Contents

List of Figures

List of Tables

List of Algorithms

Preface to the first edition

It is often the case that computer science is considered merely a branch of mathematics. This (still) often motivates the belief that logic is required for computer science just because it is required for mathematics, namely for proofs. However, logic in computing goes well beyond the context of mathematical proof, being present today in fields such as artificial intelligence and cognitive science, and having significant engineering and industrial applications. This impressive plethora of computational applications of *logic* could not be possible without a large variety of *logics*, which for our purposes can be elegantly–i.e. by means of the English connector *and*–segregated in two major classes: *classical logic(s)* and *non-classical logics*.

Yet another, but perhaps not so elegant, segregation must be contemplated when speaking of computing today: *classical computing* or *non-classical computing*. While in the latter kind one can include a large variety of computation models and computers (e.g., quantum computers, artificial neural networks, evolutionary computing), we shall consider *classical computing* to be the processing of information carried out by the von Neumann, or industrial-scale digital computer, which has as a major theoretical foundation the Turing computing paradigm. This paradigm, concretized in the Turing machine, sees computation as a spatial-temporal discrete business over symbols that can best be carried out in binary code. While this paradigm does not take into account the resources available for computation, the von Neumann computer is in fact constrained by physical–i.e. spatial and temporal–resources, which means that classical computing has more or less clearly established limitations.

When logic, whether classical or non-classical, is applied in computing, either classical or non-classical, we speak of *computational logic*. This is an important label in at least two senses. Firstly, it captures the fact that there is a subfield of formal logic that can be applied in a computational setting. This subfield might be obtained by imposing restrictions (for example, on the sets of operators), but also by extensions or just plain variations. Secondly, it helps us to distinguish clearly between computation carried out with a *logical language* from computation carried out with *other* formal languages. In effect, while the latter typically is concerned with preserving the legality of symbol strings (legal strings are processed into further legal strings), the former often aims at *truth-preservation.* Say that we have a theory and wish to know whether some assertion follows logically from it, i.e. belongs to it, or is true in it. The deduction theorem allows us to express this *logical following* in a single symbol string, known as a logical formula, and our question is notoriously best concretized in the *validity* and *satisfiability problems*, which ask whether a logical formula is always true, or is true in some interpretation, respectively. When these problems–in particular the latter–are posed in a computational context, we accordingly speak of *deductive computation.* When the computational solution is to be found by means of classical computing, we then speak of *classical deductive computing.*

In this book we elaborate on *classical deductive computing with classical logic*, and we do so without a specific regard to the field of application. Our foci are first and foremost two main subjects in which classical deductive computing with classical logic has a prominent role: *automated theorem proving* and *logic programming.*

This is thus a book on *applied logic.* Furthermore, this is a book on *applied mathematical logic.* We take here the label *mathematical logic* as synonymous with *formal logic*, and this in a very narrow sense: Formal logic is logic whose foundations lie in mathematical objects and structures. Although these mathematical foundations may be inconspicuous at the object-language level, at the metalanguage level they do become more conspicuous or even explicit. Interestingly enough–though not surprising anymore–, the mathematical structures and objects usually required in mathematical logic are precisely those needed for classical deductive computing; we talk here of lattices, graphs, trees, etc., all known as *discrete structures and objects.* This accounts for a whole chapter (Chapter 1) dedicated to the topics of discrete mathematics required for a satisfactory grasping of the material in this book. More specifically, we restrictively provide the mathematical notions that are foundational for both the theory of classical computing and classical deduction. Chapter 1 constitutes Part II of this volume, Part I being the Introduction.

Were this book on formal logic alone, there would be no need for a chapter on the *theory of computing.* Although logical languages are first and foremost formal languages, outside a computational context no issues of computability or complexity arise–certainly not in the usual treatment of logic for philosophy courses, but not even in pure mathematical logic textbooks. These issues arise when we need to compute with logical languages (e.g., Turing-completeness of programming languages). Because these issues arise here, we need to approach Turing machines, which, in turn, require the fundamentals of formal languages and models of computation, in order to be satisfactorily understood. We thus provide the basics of the *general* theory of classical computing, which includes the study of formal languages and grammars, models of computation, and computability theory. As a matter of fact, we provide more than the basics, doing so in the belief that such knowledge often comes in handy for anyone interested in computational logic. This material constitutes Chapter 2, which is Part III.

This book is one–the first–of two volumes addressing the topic of classical deductive computing. In it we focus on computing with classical logic. Although new technologies have opened a path that led to a proliferation of *new* logics, the so-called non-classical logics, classical logic remains as the standard logical system which the other, newer, systems extend or from which they diverge. This would be reason enough to justify this volume, but the fact is that, despite the many technological advances witnessed in the last decades, classical logic is still the logical system of choice for many technological applications requiring what in this book we call deductive computation.

Although the literature on classical logic is prolific, with many good introductions to the subject, with self-containment in view we provide a whole chapter (Chapter 4) on *classical logic*. This follows a comprehensive discussion on *formal logic*, *deduction*, and *deductive computation* carried out in Chapter 3, in which such fundamental notions as logical language, from the viewpoints of both form and meaning, and logical consequence, in relation to inference and deductive systems, as well as to computation, are thoroughly discussed.

The decision problem in computational logic is overwhelmingly tackled by checking for (un)satisfiability, namely by means of the so-called SAT testers or solvers. However, we thought that a working knowledge of *classical validity testing methods* is also required. These–the *classical calculi*–we present in Chapter 5, which is followed, in Chapter 6, by the different *semantics* that provide a foundation for *meaning in classical logic*.

Chapters 3 to 6, constituting Part IV of this book, comprise our discussion of classical deduction and classical logic.

In Part V, we begin by elaborating on the *(classical) satisfiability problem*, already introduced in Chapter 2, and by providing the means to computerize classical logic with a view to finding computational solutions to this problem. This *satisfiability testing* is extensively discussed in the remaining Sections of this Chapter 7. We then proceed with extensive treatments of the aforementioned main fields of computational logic, to wit, automated theorem proving (Chapter 8) and logic programming (Chapter 9). With respect to the former, we give an equal weight to resolution and analytic tableaux. This is uncommon, as the resolution calculus has all but obliterated the analytic tableaux calculus in the context of automated theorem proving, but we think this obliteration is not justified and hope to contribute to the reassessment of the pay-offs of further automating the analytic tableaux calculus. Precisely due to this imbalance our treatment of this calculus is not as comprehensive as our elaboration on resolution. As far as logic programming is concerned, we naturally focus on Prolog, as this is the major (family of) language(s) in this programming paradigm. It is our belief that by mastering the essential aspects of Prolog related to its deductive capabilities, as well as the general theory of logic programming, the reader will be well equipped to tackle most tasks involving this programming paradigm, as well as other (sub-)languages thereof, such as Datalog and Answer Set Programming.

We restrict our elaboration on classical computing to first-order predicate logic, which is known to be adequate (i.e. sound and complete) and as such provides us with a reliable means for classical deductive computation. This by no means entails that we disfavor higher-order logics, but we leave their inclusion in this text to possible future editions thereof.

As said above, this is the first of two volumes. Born in the late 1960s / early 1970s, computational logic has quickly grown to have many subfields or subjects; many, indeed (see Introduction). Clearly, this proliferation cannot be covered by a single volume, and we decided to divide the material we find essential in two volumes, the main segregation between both being that we dedicate this (first) volume to computing with classical logic, and we shall elaborate on computation with non-classical logics in a second volume. This segregation is justified not only by the fact that classical and non-classical logics have very different computational assumptions and applications, but also by the sheer quantity of topics that need to be addressed; a single book would certainly be too voluminous and readers may be interested in only one of these, classical or non-classical logics.

An advantage of this project over other works in the field is the breadth of its covering: The reader has in it far more content on computational logic than is usually the case in a single monograph or textbook. This, like any advantage, comes at a price, though: Depth had to be relinquished. This is, however, remediated by bibliographical references to works of a more limited breadth but with greater depth of treatment. Moreover, this work contains a large selection of exercises on all the approached topics. Having in mind both that most specialized monographs and handbooks lack any exercises and the large variety of topics here approached, this is indeed yet another advantage, at least for the reader of a more practical persuasion. In our selection of exercises we included novel material (e.g., theorems not given in the main text), so that the reader is expected also to approach problems in computational logic in a creative way. Exercises asking the reader to reflect on some statements or passages, as well as to engage in research, are also included. These latter exercises are meant to complement the main text with some topics that, while not being secondary, would require some extended discussion, making of this a much larger volume.

Some final remarks: Some of the material in this volume draws on two books of ours also published in College Publications, to wit, Augusto (2020a, b). This material either is as was first published, or has been submitted to some, often substantial, revisions and extensions. As was or revised/extended, it is mostly to be found in Chapters 1, 7, and 8, as well as in all Chapters of Part IV, though not in all Sections thereof. Chapters 2 and 9, as well as many Sections in Part IV (e.g., Sections 3.1-3), are completely novel, drawing only from folklore or from works by other authors. These are orthodoxly cited and indicated in the bibliographical references, but not always did we see it necessary to do so, especially with respect to material that has to some extent already acquired the character of mathematical or logical folklore.

Being a book on computational logic, this is, as said,–also–a book on mathematical logic. This explains the usual distinction in the main text of statements into definitions (abbreviated **Def.**), propositions (**Prop.**), and the odd undistinguished paragraph that for ends of internal reference is referred to as "§"; these are all given a number indicating the Section (two digits separated by a dot) and the order in the Section. For example, **2.1.3 (Def.)** indicates Definition 3 in Section 2.1. Theorems, as well as their companion lemmas and corollaries, are numbered in the same way but separately from the other numbered statements, and the same holds for examples. Exercises are numbered according to not only Section, but also Subsection.[1]

It is usual to provide the reader with a schematic guide for the reading of a book in the fields that are our foci. With this in mind, but not wishing to direct the reader more than the Table of Contents already is expected to do, we think that in order for the lay reader to have a *minimal* satisfactory grasping of classical deductive computing with classical logic the following topics are essential: The system of classical

[1] In the present 3rd edition this has been standardized as, for example, **Definition. 2.1.3.**, and the references by "§" were replaced by *Remarks*.

logic CL and the logic **CL** (Chapter 4), Herbrand semantics (concentrated in Sections 6.2 and 7.3.3), and Sections 7.1-2 for the satisfiability problem and for the necessary means to make logical formulae of CL amenable to computation. These are *sine-qua-non* requirements for a good understanding of automated theorem proving (Chapter 8) or logic programming (Chapter 9), or both. The novice reader wishing to gain a full grasp of our main topic cannot eschew the reading of the whole volume. It should be remarked, however, that some Chapters are self-standing in the sense that they can be used independently from the rest of the volume. This is particularly true of Chapter 2, which is largely conceived as a condensed treatment–with the usual selection of exercises–of the theory of classical computing, and thus can be of use for readers whose interest might fall exclusively on this topic.

For reasons to do with time, we do not include solutions to any of the exercises in this edition, but sooner or later they are expected to be provided, either online or in later editions. Readers wishing to contribute with original solutions to problems other than the most basic ones (e.g., proofs of theorems) are welcome to contact me for this end.

My thanks go to Dov M. Gabbay for including this work in this excellent series of College Publications, and to Jane Spurr for her usual impeccable assistance in the publication process.

Madrid, June 2018

Luis M. S. Augusto

Preface to the second edition

The first edition of the present work was rather hastily completed for many reasons. This hastiness contributed to addenda and errata lists longer than I feel comfortable with, as well as to the omission of some contents that I consider important in a comprehensive introduction to the large field of classical deductive computing with classical logic. Thus, this second edition improves on the first by both eliminating (hopefully most) addenda and errata, and including the mentioned contents. These are largely constituted by Datalog, on which I elaborate at length in a wholly new chapter (Chapter 9.3) for mainly two reasons: Firstly, Datalog has an intrinsic interest from the viewpoint of databases, thus expanding on the applications of logic programming; secondly, it provides an important illustration of the equation *Algorithm* = *Logic* in computational logic, to be contrasted with the case of Prolog, which concretizes the equation *Algorithm* = *Logic* + *Control*. On a more personal level, Datalog is a highy rewarding topic to research into; more specifically, how such a frugal logical language as Datalog can call for impressively complex formal semantics promises to keep researchers busy for a long time to come.

A few more exercises, in particular exercises aiming at connecting Part III and Parts IV-V, were added in this edition. Further minor improvements were made by redrawing some of the figures and by making minor changes to the main text.

Madrid, January 2020

Luis M. S. Augusto

Preface to the third edition

The present edition–the third–of this course book has an increased focus on algorithms, both in the theory of formal languages and automata and in computational logic. There are now fourteen clearly isolated algorithms in this book, all designed in pseudo-code for the sake of generality. This focus is also shown in the addition of a wholly new section (Section 1.4) on algorithms and programs.

Some new concepts were introduced, whether as definitions or in exercises, sometimes with additional examples, and in order to keep the book within a manageable size the contents on deterministic context-free languages and LR(k) grammars were almost entirely removed, as they are not required in the rest of the book and are more relevant to compilation than to basic formal language theory.

Given the importance of the Turing machine for the field of computational logic the contents of Section 2.2.3 were largely revised and more examples of, or exercises with, Turing machines are now provided.

I thank College Publications, London, for the willingness to publish this new edition so shortly after the second one.

Madrid, January 2022

Luis M. S. Augusto

References

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases.* Reading, MA, etc.: Addison-Wesley.

Abiteboul, S. & Vianu, V. (1991). Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, *43*(1), 62-124.

Apt, K. R. (1996). *From logic programming to Prolog.* Upper Saddle River, NJ: Prentice Hall.

Apt, K. R., Blair, H. A., & Walker, A. (1988). Towards a theory of declarative knowledge. In J. Minker (ed.), *Foundations of deductive databases and logic programming* (pp. 89-148). Los Altos, CA: Morgan Kaufmann.

Aristotle (ca. 350 BC). *Metaphysics.* Trans. by W. D. Ross (1908). Available at http://classics.mit.edu//Aristotle/metaphysics.html.

Augusto, L. M. (2019). *Formal logic: Classical problems and proofs.* London: College Publications.

Augusto, L. M. (2020a). *Many-valued logics: A mathematical and computational introduction.* 2nd ed. London: College Publications.

Augusto, L. M. (2020b). *Logical consequences. Theory and applications: An introduction.* 2nd ed. London: College Publications.

Augusto, L. M. (2021). *Languages, machines, and classical computation.* 3rd ed. London: College Publications.

Baaz, M., Egly, U., & Leitsch, A. (2001). Normal form transformations. In A. Robinson & A. Voronkov (eds.), *Handbook of automated reasoning*, vol. 1 (pp. 273-333). Amsterdam: Elsevier / Cambridge, MA: MIT Press.

Bachmair, L. & Ganziger, H. (2001). Resolution theorem proving. In A. Robinson & A. Voronkov (eds.), *Handbook of automated reasoning*, vol. 1 (pp. 19-99). Amsterdam: Elsevier / Cambridge, MA: MIT Press.

Beckert, B., Hähnle, R., & Schmitt, P. H. (1993). The *even more* liberalized δ-rule in free variable semantic tableaux. In G. Gottlob, A. Leitsch, & D. Mundici (eds.), *Proceedings of the third Kurt Gödel Colloquium KGC'93, Brno* (pp. 108-119). Springer.

Beth, E. W. (1955). Semantic entailment and formal derivability. *Mededelingen der Koninklijke Nederlandse Akademie van Wetenschappen*, *18*, 309-342.

Beth, E. W. (1960). Completeness results for formal systems. In J. A. Todd (ed.), *Proceedings of the International Congress of Mathematicians, 14-21 August 1958* (pp. 281-288). Cambridge: CUP.

Biere, A., Heule, M., van Maaren, H., & Walsh, T. (2009). *Handbook of satisfiability*. Amsterdam, etc.: IOS Press.

Bloch, E. D. (2011). *Proofs and fundamentals: A first course in abstract mathematics*. 2nd ed. New York, etc.: Springer.

Blum, M. (1967). A machine-independent theory of the complexity of recursive functions. *Journal of the Association for Computing Machinery*, *14*, 322-336.

Boole, G. (1847). *The mathematical analysis of logic. Being an essay towards a calculus of deductive reasoning.* Cambridge: Macmillan, Barclay, and Macmillan.

Boole, G. (1854). *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities.* London: Walton and Maberly.

Börger, E., Grädel, E., & Gurevich, Y. (2001). *The classical decision problem*. Berlin, etc.: Springer.

Ceri, S., Gottlob, G., & Tanca, L. (1989). What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, *1*(1), 146-166.

Ceri, S., Gottlob, G., & Tanca, L. (1990). *Logic programming and databases*. Berlin & Heidelberg: Springer.

Chang, C.-L. & Lee, R. C.-T. (1973). *Symbolic logic and mechanical theorem proving*. New York & London: Academic Press.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113-124.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, *2*(2), 113-124.

Church, A. (1936a). A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, *1*(1), 40-41.

Church, A. (1936b). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, *58*(2), 345-363.

Clark, K. L. (1978). Negation as failure. In H. Gallaire & J. Minker (eds.), *Logic and data bases* (pp. 293-322). New York: Plenum.

Cleave, J. P. (1991). *A study of logics.* Oxford: Clarendon Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms.* 3rd ed. Cambridge, MA & London, UK: MIT Press.

Curry, H. B. (1963). *Foundations of mathematical logic.* New York, etc.: McGraw-Hill.

D'Agostino, M. (1999). Tableau methods for classical propositional logic. In M. D'Agostino et al. (eds.), *Handbook of tableau methods* (pp. 45-123), Dordrecht: Kluwer.

Date, C. J. (2004). *Introduction to database systems.* 8th ed. Reading, MA: Addison-Wesley.

Davis, M. (2001). The early history of automated deduction. In A. Robinson & A. Voronkov (eds.), *Handbook of automated reasoning*, vol. 1 (pp. 1-15). Amsterdam: Elsevier / Cambridge, MA: MIT Press.

Davis, M. & Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the ACM*, *7*(3), 201-215.

Davis, M. D. & Weyuker, E. J. (1983). *Computability, complexity, and languages. Fundamentals of theoretical computer science.* Orlando, etc.: Academic Press.

Davis, M., Logemann, G., & Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*, *5*(7), 394-397.

Deransart, P. & Małuszyński, J. (1993). *A grammatical view of logic programming.* Cambridge, MA: MIT Press.

Digricoli, V. J. & Harrison, M. C. (1986). Equality-based binary resolution. *Journal of the Association for Computing Machinery, 33*(2), 253-289.

Doets, K. (1994). *From logic to logic programming.* Cambridge, MA & London, England: The MIT Press.

Enderton, H. B. (2001). *A mathematical introduction to logic.* 2nd ed. San Diego, etc.: Harcourt Academic Press.

Etchemendy, J. (1999). *The concept of logical consequence.* Stanford: CSLI Publications.

Fitting, M. (1996). *First order logic and automated theorem proving.* 2nd ed. New York, etc.: Springer.

Fitting, M. (1999). Introduction. In M. D'Agostino et al. (eds.), *Handbook of tableau methods* (pp. 1-44). Dordrecht: Kluwer.

Frege, G. (1892). Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik, 100*(1), 25-50.

Gabbay, D. M. & Woods, J. (2003). *A practical logic of cognitive systems. Vol. 1: Agenda relevance. A study in formal pragmatics.* Amsterdam, etc.: Elsevier.

Gabbay, D. M., Hogger, C. J., & Robinson, J. A. (eds.) (1998). *Handbook of logic in artificial intelligence and logic programming. Vol. 5: Logic Programming.* Oxford: Clarendon Press.

Gallaire, H., Minker, J., & Nicolas, J.-M. (1984). Logic and databases: A deductive approach. *Computing Surveys, 16*(2), 153-185.

Gallier, J. (2011). *Discrete mathematics.* New York, etc.: Springer.

Garey, M. R. & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness.* New York: W. H. Freeman and Company.

Gelfond, M. & Lifschitz, V. (1988). The stable model semantics for logic programming. In R. Kowalski & K. Bowen (eds.), *Logic programming: Proceedings of the 5th international conference and symposium* (pp. 1070-1080). MIT Press.

Gentzen, G. (1934-5). Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, *39*, 176-210, 405-431. (Engl. trans.: Investigations into logical deduction. In M. E. Szabo (ed.), *The Collected Papers of Gerhard Gentzen* (pp. 68-131). Amsterdam: North-Holland.)

Gilmore, P. (1960). A proof method for quantification theory: Its justification and realization. *IBM Journal of Research and Development*, *4* (1), 28-35.

Gödel, K. (1930). Die Vollständigkeit der Axiome des logischen Funktionkalküls. *Monatshefte für Mathematik*, *37*, 349-360. (Engl. trans.: The completeness of the axioms of the functional calculus of logic. In S. Feferman et al. (eds.), *Collected works. Vol. 1: Publications 1929-1936* (pp. 103-123). New York: OUP & Oxford: Clarendon Press, 1986.)

Gödel, K. (1931). Über formal unentscheidbare Sätze der *Principia Mathematica* und verwandter Systeme, I. *Monatshefte für Mathematik und Physik*, *38*, 173-198. (Engl. trans.: On formally undecidable propositions of *Principia Mathematica* and related systems, I. In S. Feferman et al. (eds.), *Collected works. Vol. 1: Publications 1929-1936* (pp. 144-195). New York: OUP & Oxford: Clarendon Press, 1986.)

Gödel, K. (1964). Postscriptum to Gödel (1934). In *Collected works I* (pp. 369-371), Oxford: OUP, 1986.

Greco, S. & Molinaro, C. (2016). *Datalog and logic databases.* Morgan & Claypool.

Grune, D. & Jacobs, C. J. H. (2010). *Parsing techniques: A practical guide.* 2nd ed. New York, NY: Springer.

Hähnle, R. & Schmitt, P. H. (1994). The liberalized δ-rule in free-variable semantic tableaux. *Journal of Automated Reasoning*, *13* (2), 211-221.

Henkin, L. (1949). The completeness of the first-order functional calculus. *Journal of Symbolic Logic*, *14* (3), 159-166.

Herbrand, J. (1930). *Recherches sur la théorie de la démonstration.* Thèses présentées à la Faculté des Sciences de Paris.

Hilbert, D. & Ackermann, W. (1928). *Grundzüge der theoretischen Logik.* Berlin: Springer.

Hintikka, J. (1955). Form and content in quantification theory. *Acta Philosophica Fennica*, *8*, 7-55.

Hopcroft, J. E., Motwani, R., & Ullman, J. (2013). *Introduction to automata theory, languages, and computation.* 3rd ed. Boston, etc.: Pearson.

Hurley, P. J. (2012). *A concise introduction to logic.* 11th ed. Boston, MA: Wadsworth.

Jaśkowski, S. (1934). On the rules of suppositions in formal logic. *Studia Logica*, *1*(1), 5-32.

Kleene, S. C. (1952). *Introduction to metamathematics.* Princeton, NJ: D. van Nostrand Co.

Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In C. E. Shannon & J. McCarthy (eds.), *Automata studies* (pp. 3-42). Princeton: Princeton University Press.

Leary, C. C. & Kristiansen, L. (2015). *A friendly introduction to mathematical logic.* 2nd ed. Geneseo, NY: Milne Library.

Leitsch, A. (1997). *The resolution calculus.* Berlin, etc.: Springer.

Letz, R. (1999). First-order tableau methods. In M. D'Agostino et al. (eds.), *Handbook of tableau methods* (pp. 125-196), Dordrecht: Kluwer.

Libkin, L. (2012). *Elements of finite model theory.* Berlin, etc.: Springer.

MacKenzie, D. (1995). The automation of proof: A historical and sociological exploration. *IEEE Annals of the History of Computing*, *17*, 7-29.

Makinson, D. (2008). *Sets, logic, and maths for computing.* London: Springer.

Martin, N. M. & Pollard, S. (1996). *Closure spaces and logic.* Dordrecht: Kluwer.

Matiyasevich, Y. V. (1993). *Hilbert's Tenth Problem.* Cambridge, MA: MIT Press.

Mendelson, E. (2015). *Introduction to mathematical logic.* 6th ed. Boca Raton, FL: Taylor & Francis Group.

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

Minsky, M. (1974). A framework for representing knowledge. Report AIM, 306, Artificial Intelligence Laboratory, MIT.

Newell, A. (1973). Production systems: Models of control structures. In W. G. Chase (ed.), *Visual information processing* (pp. 463-526), New York: Academic Press.

Newell, A. (1990). *Unified theories of cognition.* Cambridge, MA: Harvard University Press.

Nieuwenhuis, R. & Rubio, A. (2001). Paramodulation-based theorem proving. In A. Robinson & A. Voronkov (eds.), *Handbook of automated reasoning*, vol. 1 (pp. 371-443). Amsterdam: Elsevier / Cambridge, MA: MIT Press.

Ouyang, M. (1998). How good are branching rules in DPLL? *Discrete Applied Mathematics*, *89*(1-3), 281-286.

Prawitz, D. (1965). *Natural deduction. A proof-theoretical study.* Stockholm: Almqvist & Wiksell.

Priest, G. (2008). *An introduction to non-classical logic.* Cambridge: Cambridge University Press.

Przymusinski, T. C. (1989). On the declarative and procedural semantics of logic programs. *Journal of Automated Reasoning*, *5*, 167-205.

Quine, W. V. O. (1938). Completeness of the propositional calculus. *Journal of Symbolic Logic*, *3*(1), 37-40.

Rahwan, I. & Simari, G. R. (eds.) (2009). *Argumentation in artificial intelligence.* Dordrecht, etc.: Springer.

Reiter, R. (1978). On closed world data bases. In H. Gallaire & J. Minker (eds.), *Logic and data bases* (pp. 55-76). New York: Plenum.

Reiter, R. (1984). Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopolous, & J. W. Schmidt (eds.), *On conceptual modeling. Perspectives from artificial intelligence, databases, and programming languages* (pp. 191-238). New York: Springer.

Robinson, A. J. (1965). A machine-oriented logic based on the resolution principle. *Journal of ACM*, *12*(1), 23-41.

Robinson, G. & Wos, L. (1969). Paramodulation and theorem-proving in first-order theories with equality. *Machine Intelligence*, *4*, 135-150.

Scott, M. L. (2009). *Programming language pragmatics.* 3rd ed. Amsterdam, etc.: Morgan Kaufmann.

Sebesta, R. W. (2012). *Concepts of programming languages.* 10th ed. Boston, etc.: Pearson.

Shepherdson, J. C. (1984). Negation as failure: A comparison of Clark's completed data base and Reiter's closed world assumption. *Journal of Logic Programming*, *1*(1), 1-48.

Siekmann, J. H. (ed.) (2014). *Handbook of the history of logic. Vol. 9: Computational logic.* Amsterdam, etc.: North-Holland, Elsevier.

Sippu, S. & Soisalon-Soininen, E. (1990). *Parsing theory. Vol. II: LR(k) and LL(k) parsing.* Berlin, Heidelberg: Springer.

Smullyan, R. M. (1968). *First-order logic.* Mineola, NY: Dover.

Stepney, S. et al. (2005). Journeys in non-classical computation I: A grand challenge for computing research. *International Journal of Parallel, Emergent and Distributed Systems*, *20*(1), 5-19.

Sterling, L. & Shapiro, E. (1994). *The art of Prolog.* Cambridge, MA & London, England: The MIT Press.

Stone, M. H. (1936). The theory of representation for Boolean algebras. *Transactions of the American Mathematical Society*, *40*(1), 37-111.

Tarski, A. (1930). Fundamentale Begriffe der Methodologie der deduktiven Wissenschaften. I. *Monatshefte für Mathematik und Physik*, *37*, 361-404. (Engl. trans.: Fundamental concepts of the methodology of the deductive sciences. In A. Tarski, *Logic, semantics, metamathematics: Papers from 1923 to 1938* (pp. 60-109). Oxford: Clarendon Press, 1956.)

Tarski, A. (1935). Der Wahrheitsbegriff in formalisierten Sprachen. *Studia Philosophica*, *1*, 261-405 (Engl. trans.: The concept of truth in formalized languages. In A. Tarski, *Logic, semantics, metamathematics: Papers from 1923 to 1938* (pp. 152-278). Trans. by J. H.

Woodger. Oxford: Clarendon Press, 1956) (Originally published in Polish in 1933.)

Tarski, A. (1994). *Introduction to logic and to the methodology of deductive sciences.* 4th ed. J. Tarski (ed.). New York & Oxford: Oxford University Press.

Troelstra, A. S. & Schwichtenberg, H. (2000). *Basic proof theory.* 2nd ed. Cambridge: Cambridge University Press.

Tseitin, G. S. (1968). On the complexity of derivations in the propositional calculus. In A. O. Slisenko (ed.), *Studies in constructive mathematics and mathematical logic. Part 2. Seminar in mathematics* (pp. 115-125). Steklov Mathematical Institute.

Turing, A. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, *41*, 230-265.

van Emden, M. H. & Kowalski, R. A. (1976). The semantics of predicate logic as a programming language. *Journal of the Association for Computing Machinery*, *23*(4), 733-742.

van Gelder, A. (1986). Negation as failure using tight derivations for general logic programs. In *Proceedings of the Third IEEE Symposium on Logic Programming* (pp. 137-146).

van Gelder, A., Ross, K. A., & Schlipf, J. S. (1991). The well-founded semantics for general logic programs. *Journal of the ACM*, *38*(3), 620-650.

Walther, C. (1985). A mechanical solution of Schubert's Steamroller by many-sorted resolution. *Artificial Intelligence*, *26*, 217-224.

Wójcicki, R. (1988). *Theory of logical calculi: Basic theory of consequence operations.* Dordrecht: Kluwer.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, *10*(2), 189-208.

Index

Index

D

Q

R