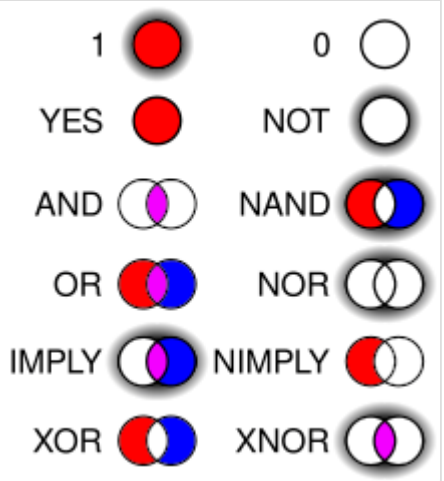
Logic in computer science

Logic in computer science covers the overlap between the field of logic and that of computer science. The topic can essentially be divided into three main areas:

- Theoretical foundations and analysis
- Use of computer technology to aid logicians
- Use of concepts from logic for computer applications



Diagrammatic representation of computer logic gates

Theoretical foundations and analysis

Logic plays a fundamental role in computer science. Some of the key areas of logic that are particularly significant are computability theory (formerly called recursion theory), modal logic and category theory. The theory of computation is based on concepts defined by logicians and mathematicians such as Alonzo Church and Alan Turing.[1][2] Church first showed the existence of algorithmically unsolvable problems using his notion of lambda-definability. Turing gave the first compelling analysis of what can be called a mechanical procedure and Kurt Gödel asserted that he found Turing's analysis "perfect.".[3] In addition some other major areas of theoretical overlap between logic and computer science are:

- Gödel's incompleteness theorem proves that any logical system powerful enough to characterize arithmetic will contain statements that can neither be proved nor disproved within that system. This has direct application to theoretical issues relating to the feasibility of proving the completeness and correctness of software.[4]
- The frame problem is a basic problem that must be overcome when using first-order logic to represent the goals of an artificial intelligence agent and the state of its environment.[5]
- The Curry–Howard correspondence is a relation between logical systems and programming languages. This theory established a precise correspondence between proofs and programs. In particular it showed that terms in the simply typed lambda calculus correspond to proofs of intuitionistic propositional logic.
- Category theory represents a view of mathematics that emphasizes the relations between structures. It is intimately tied to many aspects of computer science: type systems for programming languages, the theory of transition systems, models of programming languages and the theory of programming language semantics.[6]
- Logic programming is a programming, database and knowledge representation paradigm that is based on formal logic. A logic program is a set of sentences about some problem domain. Computation is performed by applying logical reasoning to solve problems in the domain. Major logic programming language families include Prolog, Answer Set Programming (ASP) and Datalog.

Computers to assist logicians

One of the first applications to use the term artificial intelligence was the Logic Theorist system developed by Allen Newell, Cliff Shaw, and Herbert Simon in 1956. One of the things that a logician does is to take a set of statements in logic and deduce the conclusions (additional statements) that must be true by the laws of logic. For example, if given the statements "All humans are mortal" and "Socrates is human" a valid conclusion is "Socrates is mortal". Of course this is a trivial example. In actual logical systems the statements can be numerous and complex. It was realized early on that this kind of analysis could be significantly aided by the use of computers. Logic Theorist validated the theoretical work of Bertrand Russell and Alfred North Whitehead in their influential work on mathematical logic called *Principia Mathematica*. In addition, subsequent systems have been utilized by logicians to validate and discover new mathematical theorems and proofs.[7]

Logic applications for computers

There has always been a strong influence from mathematical logic on the field of artificial intelligence (AI). From the beginning of the field it was realized that technology to automate logical inferences could have great potential to solve problems and draw conclusions from facts. Ron Brachman has described first-order logic (FOL) as the metric by which all AI knowledge representation formalisms should be evaluated. First-order logic is a general and powerful method for describing and analyzing information. The reason FOL itself is simply not used as a computer language is that it is actually too expressive, in the sense that FOL can easily express statements that no computer, no matter how powerful, could ever solve. For this reason every form of knowledge representation is in some sense a trade off between expressivity and computability. The more expressive the language is, the closer it is to FOL, the more likely it is to be slower and prone to an infinite loop.[8]

For example, IF–THEN rules used in expert systems approximate to a very limited subset of FOL. Rather than arbitrary formulas with the full range of logical operators the starting point is simply what logicians refer to as modus ponens. As a result, rule-based systems can support high-performance computation, especially if they take advantage of optimization algorithms and compilation.[9]

On the other hand, logic programming, which combines the Horn clause subset of first-order logic with a non-monotonic form of negation, has both high expressive power and efficient implementations. In particular, the logic programming language Prolog is a Turing complete programming language. Datalog extends the relational database model with recursive relations, while answer set programming is a form of logic programming oriented towards difficult (primarily NP-hard) search problems.

Another major area of research for logical theory is software engineering. Research projects such as the Knowledge Based Software Assistant and Programmer's Apprentice programs have applied logical theory to validate the correctness of software specifications. They have also used logical tools to transform the specifications into efficient code on diverse platforms and to prove the equivalence between the implementation and the specification.[10] This formal transformation-driven approach is often far more effortful than traditional software development. However, in specific domains with appropriate formalisms and reusable templates the approach has proven viable for commercial products. The appropriate domains are usually those such as weapons systems, security systems, and real-time financial systems where failure of the system has excessively high human or financial cost. An example of such a domain is Very Large Scale Integrated (VLSI) design—the process for designing the chips used for the

CPUs and other critical components of digital devices. An error in a chip can be catastrophic. Unlike software, chips can't be patched or updated. As a result, there is commercial justification for using formal methods to prove that the implementation corresponds to the specification.[11]

Another important application of logic to computer technology has been in the area of frame languages and automatic classifiers. Frame languages such as KL-ONE can be directly mapped to set theory and first-order logic. This allows specialized theorem provers called classifiers to analyze the various declarations between sets, subsets, and relations in a given model. In this way the model can be validated and any inconsistent definitions flagged. The classifier can also infer new information, for example define new sets based on existing information and change the definition of existing sets based on new data. The level of flexibility is ideal for handling the ever changing world of the Internet. Classifier technology is built on top of languages such as the Web Ontology Language to allow a logical semantic level on top of the existing Internet. This layer is called the Semantic Web.[12][13]

Temporal logic is used for reasoning in concurrent systems.[14]

See also

- Automated reasoning
- Computational logic
- Logic programming

References

1. Lewis, Harry R. (1981). *Elements of the Theory of Computation* (https://archive.org/details/elementsoftheory00lewi). Prentice Hall.
2. Davis, Martin (11 May 1995). "Influences of Mathematical Logic on Computer Science" (https://books.google.com/books?id=YafIDVd1Z68C&pg=PA290). In Rolf Herken (ed.). *The Universal Turing Machine*. Springer Verlag. ISBN 9783211826379. Retrieved 26 December 2013.
3. Kennedy, Juliette (2014-08-21). *Interpreting Godel* (https://books.google.com/books?id=ulw3BAAAQBAJ&q=Godel+convinced+by+Turing%27s+analysis&pg=PA118). Cambridge University Press. ISBN 9781107002661. Retrieved 17 August 2015.
4. Hofstadter, Douglas R. (1999-02-05). *Gödel, Escher, Bach: An Eternal Golden Braid* (https://archive.org/details/gdelescherbachet00hofs). Basic Books. ISBN 978-0465026562.
5. McCarthy, John; P.J. Hayes (1969). "Some philosophical problems from the standpoint of artificial intelligence" (http://www-formal.stanford.edu/jmc/mcchay69.pdf) (PDF). *Machine Intelligence*. **4**: 463–502.
6. Barr, Michael; Charles Wells (1998). *Category Theory for Computing Science* (https://www.math.mcgill.ca/barr/papers/ctcs.pdf) (PDF). Centre de Recherches Mathématiques.
7. Newell, Allen; J.C. Shaw; H.C. Simon (1963). "Empirical explorations with the logic theory machine" (https://archive.org/details/computersthought00feig). In Ed Feigenbaum (ed.). *Computers and Thought*. McGraw Hill. pp. 109–133 (https://archive.org/details/computersthought00feig/page/109). ISBN 978-0262560924.

8. Levesque, Hector; Ronald Brachman (1985). "A Fundamental Tradeoff in Knowledge Representation and Reasoning" (https://archive.org/details/readingsinknowle00brac). In Ronald Brachman and Hector J. Levesque (ed.). *Reading in Knowledge Representation* (https://archive.org/details/readingsinknowle00brac/page/49). Morgan Kaufmann. p. 49 (https://archive.org/details/readingsinknowle00brac/page/49). ISBN 0-934613-01-X. "The good news in reducing KR service to theorem proving is that we now have a very clear, very specific notion of what the KR system should do; the bad new is that it is also clear that the services can not be provided... deciding whether or not a sentence in FOL is a theorem... is unsolvable."
9. Forgy, Charles (1982). "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem*" (https://web.archive.org/web/20131227044049/http://web.yonsei.ac.kr/yusong/lecture/data/BI/Materials/1.1.Rete%20-%20A%20Fast%20Algorithm%20for%20the%20Many%20Pattern,%20Many%20Object%20Pattern%20Match%20Problem.pdf) (PDF). *Artificial Intelligence*. **19**: 17–37. doi:10.1016/0004-3702(82)90020-0 (https://doi.org/10.1016%2F0004-3702%2882%2990020-0). Archived from the original (http://web.yonsei.ac.kr/yusong/lecture/data/BI/Materials/1.1.Rete%20-%20A%20Fast%20Algorithm%20for%20the%20Many%20Pattern,%20Many%20Object%20Pattern%20Match%20Problem.pdf) (PDF) on 2013-12-27. Retrieved 25 December 2013.
10. Rich, Charles; Richard C. Waters (November 1987). "The Programmer's Apprentice Project: A Research Overview" (https://web.archive.org/web/20170706115702/ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-1004.pdf) (PDF). *IEEE Expert*. Archived from the original (ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-1004.pdf) (PDF) on 2017-07-06. Retrieved 26 December 2013.
11. Stavridou, Victoria (1993). *Formal Methods in Circuit Design* (https://books.google.com/books?id=Hf_AZfW2YWsC&q=VLSI+chip+design+formal+methods&pg=PA14). Press Syndicate of the University of Cambridge. ISBN 0-521-443369. Retrieved 26 December 2013.
12. MacGregor, Robert (June 1991). "Using a description classifier to enhance knowledge representation". *IEEE Expert*. **6** (3): 41–46. doi:10.1109/64.87683 (https://doi.org/10.1109%2F64.87683). S2CID 29575443 (https://api.semanticscholar.org/CorpusID:29575443).
13. Berners-Lee, Tim; James Hendler; Ora Lassila (May 17, 2001). "The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities" (https://web.archive.org/web/20130424071228/http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html). *Scientific American*. **284**: 34–43. doi:10.1038/scientificamerican0501-34 (https://doi.org/10.1038%2Fscientificamerican0501-34). Archived from the original (http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html) on April 24, 2013.
14. Colin Stirling (1992). "Modal and Temporal Logics". In S. Abramsky; D. M. Gabbay; T. S. E. Maibaum (eds.). *Handbook of Logic in Computer Science*. Vol. II. Oxford University Press. pp. 477–563. ISBN 0-19-853761-1.

Further reading

- Ben-Ari, Mordechai (2012). *Mathematical Logic for Computer Science* (https://www.weizmann.ac.il/sci-tea/benari/research-activities/mathematical-logic-computer-science-third-edition) (3rd ed.). Springer-Verlag. ISBN 978-1447141280.
- Harrison, John (2009). *Handbook of Practical Logic and Automated Reasoning* (https://www.cl.cam.ac.uk/~jrh13/atp/) (1st ed.). Cambridge University Press. ISBN 978-0521899574.
- Huth, Michael; Ryan, Mark (2004). *Logic in Computer Science: Modelling and Reasoning about Systems* (http://www.cs.bham.ac.uk/research/lics/) (2nd ed.). Cambridge University Press. ISBN 978-0521543101.
- Burris, Stanley N. (1997). *Logic for Mathematics and Computer Science* (https://www.math.u

waterloo.ca/~snburris/htdocs/lmcs.html) (1st ed.). Prentice Hall. ISBN 978-0132859745.

External links

- Article on *Logic and Artificial Intelligence* (http://plato.stanford.edu/entries/logic-ai/) at the *Stanford Encyclopedia of Philosophy*.
- IEEE Symposium on Logic in Computer Science (https://web.archive.org/web/20051025052601/http://www.informatik.hu-berlin.de/lics/) (LICS)
- Alwen Tiu, Introduction to logic (http://videolectures.net/ssll09_tiu_intlo/) video recording of a lecture at ANU Logic Summer School '09 (aimed mostly at computer scientists)