WIKIPEDIA
The Free Encyclopedia

Semantics (computer science)

In programming language theory, **semantics** is the rigorous mathematical study of the meaning of programming languages.[1] Semantics assigns computational meaning to valid strings in a programming language syntax. It is closely related to, and often crosses over with, the semantics of mathematical proofs.

Semantics describes the processes a computer follows when executing a program in that specific language. This can be shown by describing the relationship between the input and output of a program, or an explanation of how the program will be executed on a certain platform, hence creating a model of computation.

History

In 1967, Robert W. Floyd publishes the paper *Assigning meanings to programs*; his chief aim is "a rigorous standard for proofs about computer programs, including proofs of correctness, equivalence, and termination".[2][3] Floyd further writes:[2]

> A semantic definition of a programming language, in our approach, is founded on a syntactic definition. It must specify which of the phrases in a syntactically correct program represent commands, and what conditions must be imposed on an interpretation in the neighborhood of each command.

In 1969, Tony Hoare publishes a paper on Hoare logic seeded by Floyd's ideas, now sometimes collectively called *axiomatic semantics*.[4][5]

In the 1970s, the terms *operational semantics* and *denotational semantics* emerged.[5]

Overview

The field of formal semantics encompasses all of the following:

- The definition of semantic models
- The relations between different semantic models
- The relations between different approaches to meaning
- The relation between computation and the underlying mathematical structures from fields such as logic, set theory, model theory, category theory, etc.

It has close links with other areas of computer science such as programming language design, type theory, compilers and interpreters, program verification and model checking.

Approaches

There are many approaches to formal semantics; these belong to three major classes:

- **Denotational semantics**,[6] whereby each phrase in the language is interpreted as a *denotation*, i.e. a conceptual meaning that can be thought of abstractly. Such denotations are often mathematical objects inhabiting a mathematical space, but it is not a requirement that they should be so. As a practical necessity, denotations are described using some form of mathematical notation, which can in turn be formalized as a denotational metalanguage. For example, denotational semantics of functional languages often translate the language into domain theory. Denotational semantic descriptions can also serve as compositional translations from a programming language into the denotational metalanguage and used as a basis for designing compilers.
- **Operational semantics**,[7] whereby the execution of the language is described directly (rather than by translation). Operational semantics loosely corresponds to interpretation, although again the "implementation language" of the interpreter is generally a mathematical formalism. Operational semantics may define an abstract machine (such as the SECD machine), and give meaning to phrases by describing the transitions they induce on states of the machine. Alternatively, as with the pure lambda calculus, operational semantics can be defined via syntactic transformations on phrases of the language itself;
- **Axiomatic semantics**,[8] whereby one gives meaning to phrases by describing the *axioms* that apply to them. Axiomatic semantics makes no distinction between a phrase's meaning and the logical formulas that describe it; its meaning *is* exactly what can be proven about it in some logic. The canonical example of axiomatic semantics is Hoare logic.

Apart from the choice between denotational, operational, or axiomatic approaches, most variations in formal semantic systems arise from the choice of supporting mathematical formalism.

Variations

Some variations of formal semantics include the following:

- **Action semantics**[9] is an approach that tries to modularize denotational semantics, splitting the formalization process in two layers (macro and microsemantics) and predefining three semantic entities (actions, data and yielders) to simplify the specification;
- **Algebraic semantics**[8] is a form of axiomatic semantics based on algebraic laws for describing and reasoning about program semantics in a formal manner. It also supports denotational semantics and operational semantics;
- **Attribute grammars**[10] define systems that systematically compute "metadata" (called *attributes*) for the various cases of the language's syntax. Attribute grammars can be understood as a denotational semantics where the target language is simply the original language enriched with attribute annotations. Aside from formal semantics, attribute grammars have also been used for code generation in compilers, and to augment regular or context-free grammars with context-sensitive conditions;
- **Categorical (or "functorial") semantics**[11] uses category theory as the core mathematical formalism. A categorical semantics is usually proven to correspond to some axiomatic semantics that gives a syntactic presentation of the categorical structures. Also, denotational semantics are often instances of a general categorical semantics;[12]

- **Concurrency semantics**[13] is a catch-all term for any formal semantics that describes concurrent computations. Historically important concurrent formalisms have included the actor model and process calculi;
- **Game semantics**[14] uses a metaphor inspired by game theory;
- **Predicate transformer semantics**,[15] developed by Edsger W. Dijkstra, describes the meaning of a program fragment as the function transforming a postcondition to the precondition needed to establish it.

Describing relationships

For a variety of reasons, one might wish to describe the relationships between different formal semantics. For example:

- To prove that a particular operational semantics for a language satisfies the logical formulas of an axiomatic semantics for that language. Such a proof demonstrates that it is "sound" to reason about a particular (operational) *interpretation strategy* using a particular (axiomatic) *proof system*.
- To prove that operational semantics over a high-level machine is related by a simulation with the semantics over a low-level machine, whereby the low-level abstract machine contains more primitive operations than the high-level abstract machine definition of a given language. Such a proof demonstrates that the low-level machine "faithfully implements" the high-level machine.

It is also possible to relate multiple semantics through abstractions via the theory of abstract interpretation.

See also

- Computational semantics
- Formal semantics (logic)
- Formal semantics (linguistics)
- Ontology
- Ontology (information science)
- Semantic equivalence
- Semantic technology

References

1. Goguen, Joseph A. (1975). "Semantics of computation". *Category Theory Applied to Computation and Control*. Lecture Notes in Computer Science. Vol. 25. Springer. pp. 151–163. doi:10.1007/3-540-07142-3_75 (https://doi.org/10.1007%2F3-540-07142-3_75). ISBN 978-3-540-07142-6.
2. Floyd, Robert W. (1967). "Assigning Meanings to Programs" (https://people.eecs.berkeley.edu/~necula/Papers/FloydMeaning.pdf) (PDF). In Schwartz, J.T. (ed.). *Mathematical Aspects of Computer Science* (https://books.google.com/books?id=ynigSICJflYC). Proceedings of Symposium on Applied Mathematics. Vol. 19. American Mathematical Society. pp. 19–32. ISBN 0821867288.

3. Knuth, Donald E. "Memorial Resolution: Robert W. Floyd (1936–2001)" (https://stacks.stanford.edu/file/druid:zy788sr3998/SC0193_MemorialResolution_Floyd_Robert.pdf) (PDF). *Stanford University Faculty Memorials*. Stanford Historical Society.
4. Hoare, C. A. R. (October 1969). "An axiomatic basis for computer programming" (https://dl.acm.org/doi/pdf/10.1145/363235.363259). *Communications of the ACM*. **12** (10): 576–580. doi:10.1145/363235.363259 (https://doi.org/10.1145%2F363235.363259). S2CID 207726175 (https://api.semanticscholar.org/CorpusID:207726175).
5. Winskel, Glynn (1993). *The formal semantics of programming languages : an introduction* (https://archive.org/details/formalsemanticso0000wins). Cambridge, Mass.: MIT Press. p. xv (https://archive.org/details/formalsemanticso0000wins/page/n17). ISBN 978-0-262-23169-5.
6. Schmidt, David A. (1986). *Denotational Semantics: A Methodology for Language Development*. William C. Brown Publishers. ISBN 9780205104505.
7. Plotkin, Gordon D. (1981). A structural approach to operational semantics (Report). Technical Report DAIMI FN-19. Computer Science Department, Aarhus University.
8. Goguen, Joseph A.; Thatcher, James W.; Wagner, Eric G.; Wright, Jesse B. (1977). "Initial algebra semantics and continuous algebras" (https://doi.org/10.1145%2F321992.321997). *Journal of the ACM*. **24** (1): 68–95. doi:10.1145/321992.321997 (https://doi.org/10.1145%2F321992.321997). S2CID 11060837 (https://api.semanticscholar.org/CorpusID:11060837).
9. Mosses, Peter D. (1996). Theory and practice of action semantics (Report). BRICS Report RS9653. Aarhus University.
10. Deransart, Pierre; Jourdan, Martin; Lorho, Bernard (1988). *"Attribute Grammars: Definitions, Systems and Bibliography*. Lecture Notes in Computer Science 323. Springer-Verlag. ISBN 9780387500560.
11. Lawvere, F. William (1963). "Functorial semantics of algebraic theories" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC221940). *Proceedings of the National Academy of Sciences of the United States of America*. **50** (5): 869–872. Bibcode:1963PNAS...50..869L (https://ui.adsabs.harvard.edu/abs/1963PNAS...50..869L). doi:10.1073/pnas.50.5.869 (https://doi.org/10.1073%2Fpnas.50.5.869). PMC 221940 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC221940). PMID 16591125 (https://pubmed.ncbi.nlm.nih.gov/16591125).
12. Andrzej Tarlecki; Rod M. Burstall; Joseph A. Goguen (1991). "Some fundamental algebraic tools for the semantics of computation: Part 3. Indexed categories" (https://doi.org/10.1016%2F0304-3975%2891%2990085-G). *Theoretical Computer Science*. **91** (2): 239–264. doi:10.1016/0304-3975(91)90085-G (https://doi.org/10.1016%2F0304-3975%2891%2990085-G).
13. Batty, Mark; Memarian, Kayvan; Nienhuis, Kyndylan; Pichon-Pharabod, Jean; Sewell, Peter (2015). "The problem of programming language concurrency semantics". *Proceedings of the European Symposium on Programming Languages and Systems*. Springer. pp. 283–307. doi:10.1007/978-3-662-46669-8_12 (https://doi.org/10.1007%2F978-3-662-46669-8_12).
14. Abramsky, Samson (2009). "Semantics of interaction: An introduction to game semantics". In Andrew M. Pitts; P. Dybjer (eds.). *Semantics and Logics of Computation* (https://ora.ox.ac.uk/objects/uuid:ab3ece5b-cd8d-49e6-ba33-010ea4c1a1ac). Cambridge University Press. pp. 1–32. doi:10.1017/CBO9780511526619.002 (https://doi.org/10.1017%2FCBO9780511526619.002). ISBN 9780521580571.
15. Dijkstra, Edsger W. (1975). "Guarded commands, nondeterminacy and formal derivation of programs" (https://doi.org/10.1145%2F360933.360975). *Communications of the ACM*. **18** (8): 453–457. doi:10.1145/360933.360975 (https://doi.org/10.1145%2F360933.360975). S2CID 1679242 (https://api.semanticscholar.org/CorpusID:1679242).

Further reading

Textbooks

- Floyd, Robert W. (1967). "Assigning Meanings to Programs" (https://www.cs.tau.ac.il/~nachumd/term/FloydMeaning.pdf) (PDF). In Schwartz, J.T. (ed.). *Mathematical Aspects of Computer Science* (https://books.google.com/books?id=ynigSICJflYC). Proceedings of Symposium on Applied Mathematics. Vol. 19. American Mathematical Society. pp. 19–32. ISBN 0821867288.
- Hennessy, M. (1990). *The semantics of programming languages: an elementary introduction using structural operational semantics*. Wiley. ISBN 978-0-471-92772-3.
- Tennent, Robert D. (1991). *Semantics of Programming Languages* (https://books.google.com/books?id=K7N7QgAACAAJ). Prentice Hall. ISBN 978-0-13-805599-8.
- Gunter, Carl (1992). *Semantics of Programming Languages*. MIT Press. ISBN 0-262-07143-6.
- Nielson, H. R.; Nielson, Flemming (1992). *Semantics With Applications: A Formal Introduction* (https://web.archive.org/web/20120417112149/http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.pdf) (PDF). Wiley. ISBN 978-0-471-92980-2. Archived from the original (http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.pdf) (PDF) on 2012-04-17. Retrieved 2011-05-27.
- Winskel, Glynn (1993). *The Formal Semantics of Programming Languages: An Introduction*. MIT Press. ISBN 0-262-73103-7.
- Mitchell, John C. (1995). *Foundations for Programming Languages* (http://www.lix.polytechnique.fr/~catuscia/teaching/cg520/papers_and_books/Mitchell_book.ps.gz) (Postscript).
- Slonneger, Kenneth; Kurtz, Barry L. (1995). *Formal Syntax and Semantics of Programming Languages* (http://www.cs.uiowa.edu/~slonnegr/plf/Book/). Addison-Wesley. ISBN 0-201-65697-3.
- Reynolds, John C. (1998). *Theories of Programming Languages* (https://archive.org/details/theoriesofprogra0000reyn). Cambridge University Press. ISBN 0-521-59414-6.
- Harper, Robert (2006). *Practical Foundations for Programming Languages* (https://web.archive.org/web/20070627041059/https://www.cs.cmu.edu/~rwh/plbook/book.pdf) (PDF). Archived from the original (https://www.cs.cmu.edu/~rwh/plbook/book.pdf) (PDF) on 2007-06-27. (Working draft)
- Nielson, H. R.; Nielson, Flemming (2007). *Semantics with Applications: An Appetizer* (https://books.google.com/books?id=oPi0yERDUeYC). Springer. ISBN 978-1-84628-692-6.
- Stump, Aaron (2014). *Programming Language Foundations*. Wiley. ISBN 978-1-118-00747-1.
- Krishnamurthi, Shriram (2012). "Programming Languages: Application and Interpretation" (http://cs.brown.edu/courses/cs173/2012/book/) (2nd ed.).

Lecture notes

- Winskel, Glynn. "Denotational Semantics" (http://www.cl.cam.ac.uk/~gw104/dens.pdf) (PDF). University of Cambridge.

External links

- Aaby, Anthony (2004). *Introduction to Programming Languages* (https://web.archive.org/web/20150619164601/http://www.emu.edu.tr/aelci/Courses/D-318/D-318-Files/plbook/semantic.htm). Archived from the original (http://www.emu.edu.tr/aelci/Courses/D-318/D-318-Files/plbook/semantic.htm) on 2015-06-19. Semantics.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Semantics_(computer_science)&oldid=1187106449"