System Specification, Verification and Synthesis (SSVS) – CS 4830/7485, Fall 2019

Parting thoughts

Stavros Tripakis

Northeastern University
Khoury College of Computer Sciences

WHAT WE COVERED

Systems, Specifications, Verification, Synthesis

- Systems: formal system modeling
  - State machines, transition systems, automata
- Specifications: formal requirements
  - Temporal logics, automata
- Verification: check whether the system satisfies the spec
  - We saw model-checking
  - Explicit-state (enumerative) and symbolic methods
- Synthesis: automatic generation of systems from specs

WHAT WE DID NOT COVER

(very partial list)

Abstraction

- Big topic.
- Trace inclusion, **simulation**, **bisimulation**: relations between transition systems. E.g., see Chapter 7 of [Baier and Katoen, 2008].
  - Can help reduce the size (number of states / transitions of a transition system).
  - While preserving some of its properties [Loiseaux et al., 1995].
  - Sometimes infinite-state systems become finite, e.g., timed automata! See Chapter 9 of [Baier and Katoen, 2008] and [Alur and Dill, 1994, Tripakis and Yovine, 2001].
- **Predicate abstraction**: e.g., abstract integer variable n with three predicates: $n < 0$, $n = 0$, $n > 0$ (three abstract states). See [Dams and Grumberg, 2018, Jhala et al., 2018] and [Graf and Saidi, 1997].
- **Abstract interpretation**: a mathematical theory of abstractions, compositions of abstractions, computations of abstractions using fixpoints, etc [Cousot and Cousot, 1977].

Abstraction

- Big problem: how do we come up with the right abstractions? One idea is **Counter-Example Guided Abstraction Refinement** (CEGAR): start with a coarse abstraction and refine to more fine-grain abstractions as necessary, based on **spurious counter-examples**.
- Abstraction is key in **software verification** [Ball et al., 2011]. This is a whole area by itself, with annual tool competitions (e.g., `https://sv-comp.sosy-lab.org/`), etc.
- Many interesting topics in software verification: **invariant generation**, **termination analysis**, **interpolation** [McMillan, 2018], dealing with memory and memory allocation (e.g., **separation logic** [O'Hearn, 2019]), **weak memory models** [Atig et al., 2010], ...

Refinement

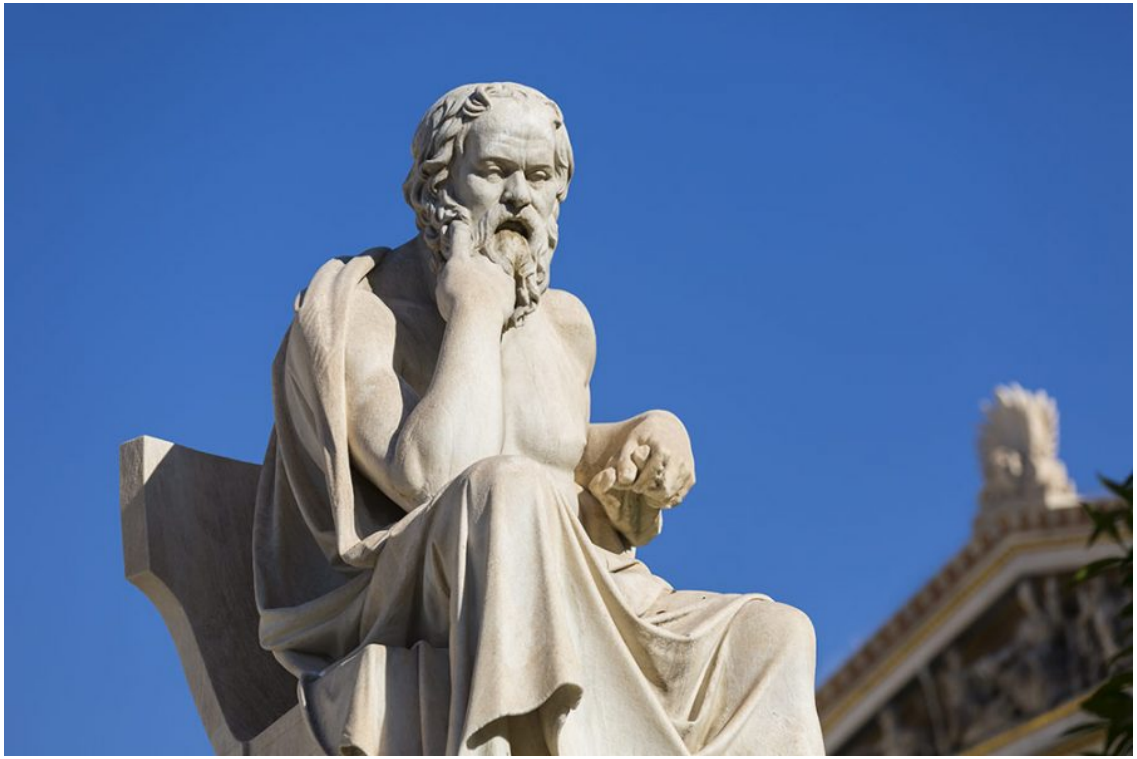
- As a relation, the counter-part to abstraction: if A abstracts B then B refines A, and vice versa.
- **Stepwise program refinement**: a top-down program design methodology: from more abstract programs, to more concrete programs, to finally code (implementations).
  - While preserving properties along the way.
  - And also preserving **compositionality**.
- Some
  references: [Floyd, 1967, Hoare, 1969, Dijkstra, 1972, Wirth, 1971, Back and Wright, 1998, Tripakis, 2016, Dragomir et al., 2018]

Other not covered topics

- **Probabilistic Model Checking** [Baier et al., 2018]
- **Hybrid Systems** [Doyen et al., 2018]
- **Process Algebras** [Cleaveland et al., 2018]
- **Model Checking Security Protocols** [Basin et al., 2018]
- **Conformance Testing** and combinations of model-checking and testing [Broy et al., 2005, Godefroid and Sen, 2018]
- **Static Program Analysis** [Nielson et al., 1999]
- **Debugging** [Zeller, 2005]
- Software engineering
- Theorem proving
- ...

PHILOSOPHY

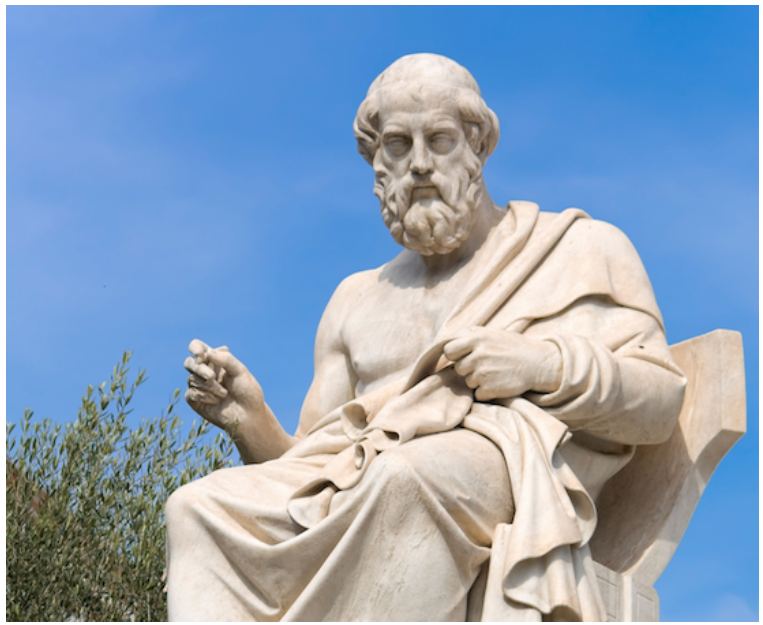
Corrupting young minds



Socrates

Language

- And logic, and reason.
- Definitions matter.
- Human language typically very poor to express complex concepts.
  - C.f. Plato's dialogues.
- A lesson we mostly forget these days, with catastrophic consequences.



Plato

Systems

- Everything is a system.
- Beautifully complex, even when small: c.f., Collatz.

Democracy

- Political systems are systems.
- Anarchy, Monarchy, Oligarchy, Democracy.
- Like all systems, they satisfy some properties, and violate other properties.
- For example, none of anarchy, monarchy, nor oligarchy, can guarantee human rights.
- Can democracy guarantee human rights?

Freedom and human rights

- What exactly does freedom mean?
- Which are our rights, exactly?
- The law.
- The justice system.

Predictions

- About the future.
- Science.
- Intelligence.
- About the past.
- Simulation and verification.

Causality

- We ask *Why ... ?* questions all the time.
- Correlation vs. causation is still an unsolved problem in science.
- Many questions beginning with *Why* are ill-defined.
- *Why am I angry?*
- *Why does my stomach hurt?*
- *Why did Nokia fail?*
- *Why did Greece go bankrupt?*
- *What caused the financial crisis in 20XX?*
- *Why is this state machine in state X?*

Nature vs nurture

- Another famous question that has been bothering mankind for centuries.
  - Famous professors still publish best-sellers on it.
- Yet from a system-theoretic point of view the question makes little sense.
- When a state machine exhibits a certain behavior, is that due to the structure of the state machine, or due to the inputs that the machine has been given?

Time

- What is time?
- Isn't it just the change of state?
- Is it possible to "travel back in time"?
- Does it mean "roll back your state" (like "undo")?
- Or roll back the state of the universe, while maintaining your state (age, memories, etc.) unchanged?

Don't forget to answer the course evaluation survey!

Bibliography I

Alur, R. and Dill, D. (1994).

A theory of timed automata.

Theoretical Computer Science, 126:183–235.

Atig, M. F., Bouajjani, A., Burckhardt, S., and Musuvathi, M. (2010).

On the verification problem for weak memory models.

In *37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '10, pages 7–18. ACM.

Back, R.-J. and Wright, J. (1998).

Refinement Calculus.

Springer.

Baier, C., de Alfaro, L., Forejt, V., and Kwiatkowska, M. (2018).

Model checking probabilistic systems.

In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 963–999. Springer.

Baier, C. and Katoen, J.-P. (2008).

Principles of Model Checking.

MIT Press.

Ball, T., Levin, V., and Rajamani, S. K. (2011).

A Decade of Software Model Checking with SLAM.

Commun. ACM, 54(7):68–76.

Basin, D., Cremers, C., and Meadows, C. (2018).

Model checking security protocols.

In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 727–762. Springer.

Bibliography II

Broy, M., Jonsson, B., Katoen, J.-P., Leucker, M., and Pretschner, A. (2005).

Model-Based Testing of Reactive Systems: Advanced Lectures (Lecture Notes in Computer Science).
Springer.

Cleaveland, R., Roscoe, A. W., and Smolka, S. A. (2018).

Process algebra and model checking.
In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 1149–1195. Springer.

Cousot, P. and Cousot, R. (1977).

Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints.
In *4th ACM Symp. POPL*.

Dams, D. and Grumberg, O. (2018).

Abstraction and abstraction refinement.
In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 385–419. Springer.

Dijkstra, E. (1972).

Notes on structured programming.
In Dahl, O., Dijkstra, E., and Hoare, C., editors, *Structured programming*, pages 1–82. Academic Press, London, UK.

Doyen, L., Frehse, G., Pappas, G. J., and Platzer, A. (2018).

Verification of hybrid systems.
In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 1047–1110. Springer.

Bibliography III

Dragomir, I., Preoteasa, V., and Tripakis, S. (2018).

The Refinement Calculus of Reactive Systems Toolset.

In *Tools and Algorithms for the Construction and Analysis of Systems – TACAS*, volume 10806 of *LNCS*. **Distinguished Artifact Award**.

Floyd, R. (1967).

Assigning meanings to programs.

In *In. Proc. Symp. on Appl. Math. 19*, pages 19–32. American Mathematical Society.

Godefroid, P. and Sen, K. (2018).

Combining model checking and testing.

In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 613–649. Springer International Publishing, Cham.

Graf, S. and Saidi, H. (1997).

Construction of abstract state graphs with PVS.

In Grumberg, O., editor, *Computer Aided Verification*, pages 72–83. Springer.

Hoare, C. A. R. (1969).

An axiomatic basis for computer programming.

Comm. ACM, 12(10):576–580.

Jhala, R., Podelski, A., and Rybalchenko, A. (2018).

Predicate abstraction for program verification.

In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 447–491. Springer.

Loiseaux, C., Graf, S., Sifakis, J., Bouajjani, A., and Bensalem, S. (1995).

Property preserving abstractions for the verification of concurrent systems.

Formal Methods in System Design.

Bibliography IV

McMillan, K. L. (2018).
Interpolation and model checking.
In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 421–446. Springer.

Nielson, F., Nielson, H. R., and Hankin, C. (1999).
Principles of Program Analysis.
Springer.

O'Hearn, P. (2019).
Separation logic.
Comm. ACM, 62(2):86–95.

Tripakis, S. (2016).
Compositionality in the Science of System Design.
Proceedings of the IEEE, 104(5):960–972.

Tripakis, S. and Yovine, S. (2001).
Analysis of Timed Systems using Time-abstracting Bisimulations.
Formal Methods in System Design, 18(1):25–68.

Wirth, N. (1971).
Program development by stepwise refinement.
Comm. ACM, 14(4):221–227.

Zeller, A. (2005).
Why Programs Fail: A Guide to Systematic Debugging.
Morgan Kaufmann.