Verification of hybrid systems

Model-based approaches

Zdeněk Hurák

2023-09-01

Validation vs verification

According to PMBOK Guide:

Validation
: The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers.

Verification
: The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process.

Approches to verification

- Testing
- Simulation
- Formal verification

Formal verification

- Proving satisfaction of specifications for all possible conditions/evolutions of the system.
- The specifications are given in a formal language.
- Formal mathematical methods are used.

Model checking as one of formal verification approaches

- Based on a model of the system.
  - For discrete-event and hybrid system the model is in the form of Labelled Transition System (LTS).
- The outcome of model checking is a a proof or a counterexample.

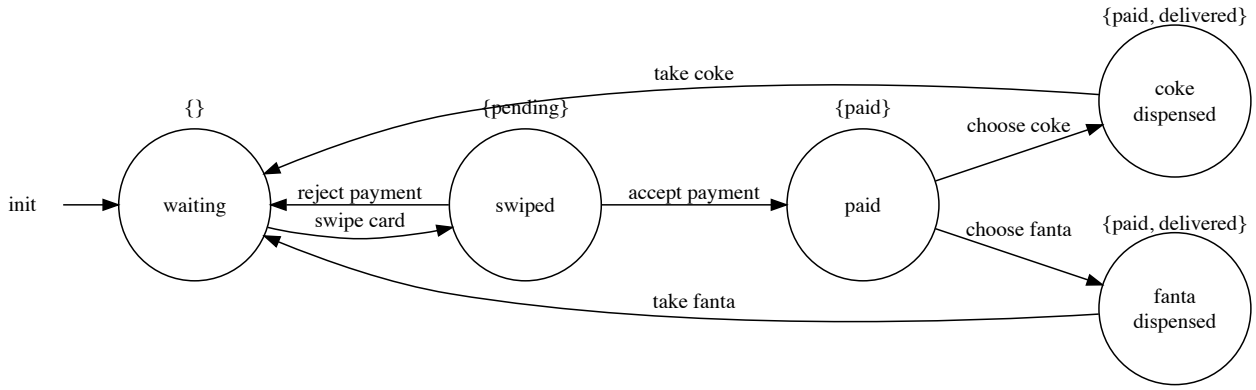
Labelled Transition system (LTS)

- A variant of (hybrid) automaton.
- Labels are attached to the states rather than the transitions.
  - The labels come from the set $\mathcal{P}$ of atomic propositions

$$\mathcal{P} = \{p_1, p_2, \ldots, p_n\}.$$

  - Labelling function $l : \mathcal{X} \to 2^{\mathcal{P}}$ assigns a set of atomic propositions to each state.

Example: beverage vending machine



$$\mathcal{P} = \{\text{pending}, \text{paid}, \text{delivered}\}$$

- Specification #1: The machine never dispenses a beverage without being paid, that is,

$$\text{delivered} \Rightarrow \text{paid}$$

- Specification #2: If the user pays, the machine will eventually dispenses a beverage.
  - But how do we express the “eventually” part?

Formal specifications

- Safety (invariance)
- Liveness (progress)
- More general specifications using temporal logics.

Safety (invariance)

- Something bad never happens.
  - The system never enters bad (unsafe) region of the state space.
- Equivalently, something good always holds.
  - The system always stays in a good (safe) region of the state space.
- How do we check this?
  - Reachability analysis
  - Barrier certificates
  - …

Reachability analysis

- For autonomous systems
  - Given a set of initial states $\mathcal{X}_0$, determine the set of states $\mathcal{X}_{\text{reach}}$ that can be reached from X_0 over the time interval $(0, t)$.
- For non-autonomous (controlled) systems
  - Given a set of initial states $\mathcal{X}_0$, determine the set of states $\mathcal{X}_{\text{reach}}$ that can be reached from X_0 over the time interval $(0, t)$ using some control.
- Robust versions: uncertain parameters, disturbances.

Reachability computations for hybrid systems

- Checking a feasibility of an optimal control problem for a MLD system.
  - As in Hybrid Toolbox (and Hysdel): http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/
- Set propagation techniques.

Basic computational steps for reachability analysis based on set propagation

- Given a set $\mathcal{X}_k$ of current states , compute the set $\mathcal{X}_{k+1}$ of next states of a discrete-time (or discretized) system such that

$$\boldsymbol{x}(k+1) = \mathbf{f}(\boldsymbol{x}(k))$$

- Example: linear discrete-time system, the initial states in a box

$$\begin{aligned}\boldsymbol{x}(k+1) &= \mathbf{A}\boldsymbol{x}(k),\\ \boldsymbol{x}(k) \in \mathcal{X}_k &= \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}_{\min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\max}\}\end{aligned}$$

- How does the set $\mathcal{X}_{k+1}$ look like?
- Typically an overapproximation of the set of all possible next states.

Sets to be propagates

- Intervals
- Polyhedra
  - V and H representations.
- Ellipsoids
- Zonotopes
  - Subclass of polytopes.
  - Often more efficient than general polytopes.
  - Affine transformation of a unit box.
  - Commonly represented using generator representation (a center and a finite number of generator vectors).
- …

Software for reachability analysis based on set propagation

- In Matlab:
  - Multiparametric Toolbox (MPT): https://www.mpt3.org
    - Some functionality (invariant set computation) purely for discrete-time systems
  - CORA: https://tumcps.github.io/CORA/
    - Continuous-time and hybrid systems.
  - you can find quite a few other tools on the web but mostly unmaintained…
- In Julia:
  - ReachabilityAnalysis.jl: https://juliareach.github.io/ReachabilityAnalysis.jl/
- In Python:
  - …

Barrier certificate for continuous systems

Given two sets:

- the set of initial states $\mathcal{X}_0$
- and the set of unsafe (forbidden) states $\mathcal{X}_\mathrm{u}$,

define the barrier certificate (function) $B(\boldsymbol{x})$ such that

$$B(\boldsymbol{x}) > 0, \quad \forall \boldsymbol{x} \in \mathcal{X}_\mathrm{u},$$

$$B(\boldsymbol{x}) \leq 0, \quad \forall \boldsymbol{x} \in \mathcal{X}_0,$$

$$\nabla B(\boldsymbol{x})^\top \mathbf{f}(\boldsymbol{x}, \boldsymbol{u}) \leq 0, \quad \forall \boldsymbol{x}, \boldsymbol{u} \,\mathrm{such\ that}\, B(\boldsymbol{x}) = 0.$$

- Closely linked to Lyapunov function. But it is not the same.

Convex relaxation of the barrier certificate problem

$$B(\boldsymbol{x}) > 0, \quad \forall \boldsymbol{x} \in \mathcal{X}_{\mathrm{u}},$$

$$B(\boldsymbol{x}) \leq 0, \quad \forall \boldsymbol{x} \in \mathcal{X}_{0},$$

$$\nabla B(\boldsymbol{x})^{\top} \mathbf{f}(\boldsymbol{x}, \boldsymbol{u}) \leq 0, \quad \forall \boldsymbol{x} \in \mathcal{X}, \boldsymbol{u} \in \mathcal{U}.$$

Barrier certificate for hybrid systems

- For a hybrid automaton with l locations $\{q_1, q_2, \ldots, q_l\}$, not just one but l barrier functions/certificates are needed:

$$B_i(\boldsymbol{x}) > 0, \quad \forall \boldsymbol{x} \in \mathcal{X}_{\mathrm{u}}(q_i),$$

$$B_i(\boldsymbol{x}) \leq 0, \quad \forall \boldsymbol{x} \in \mathcal{X}_0(q_i),$$

$$\nabla B_i(\boldsymbol{x})^\top \mathbf{f}_i(\boldsymbol{x}, \boldsymbol{u}) \leq 0, \quad \forall \boldsymbol{x}, \boldsymbol{u} \,\text{such that}\, B_i(\boldsymbol{x}) = 0,$$

$$B_i(\boldsymbol{x}) \leq 0, \quad \forall \boldsymbol{x} \in \mathcal{R}(q_j, q_i, \boldsymbol{x}^-) \,\text{for some}\, q_j \\ \text{and}\, \boldsymbol{x}^- \in \mathcal{G}(q_j, q_i) \,\text{with}\, B_j(\boldsymbol{x}^-) \leq 0.$$

Convex relaxation of barrier certificates for hybrid systems

$$\nabla B_i(\boldsymbol{x})^\top \mathbf{f}_i(\boldsymbol{x}, \boldsymbol{u}) \leq 0, \quad \forall \boldsymbol{x} \in X_0(q_i), \boldsymbol{u} \in \mathcal{U}(q_i),$$

$$B_i(\boldsymbol{x}) \leq 0, \quad \forall (\boldsymbol{x}, \boldsymbol{x}^-) \,\text{such that}\, \boldsymbol{x} \in \mathcal{R}(q_j, q_i, \boldsymbol{x}^-), \\ \text{and}\, \boldsymbol{x}^- \in \mathcal{G}(q_i, q_j).$$

Liveness

- Something good eventually happens.
  - The system eventually enters a good region of the state space.
  - Stability is one example.

General specs using temporal logics

- Temporal logis use some **temporal operators** together with logical operators and quantifiers.
- Several temporal logics
  - Linear Temporal Logic (LTL)
  - Computation Tree Logic (CTL)
  - Mix of the two (CTL*)
  - Timed CTL (TCLT) – timing added to CTL, used by UPAAL.
  - Metric temporal logic (MTL) – timing added to LTL.
  - Signal temporal logig (STL) – real-valued signals.

Temporal operators

- $\mathbf{F}$ or $\Diamond$: Eventually (or **F**inally)
- $\mathbf{G}$ or $\Box$: **G**lobally (or Always)
- $\mathbf{X}$ or $\bigcirc$: Ne**X**t
- $\mathbf{U}$ or $\sqcup$: **U**ntill

Logical operators

- $\neg, \vee, \wedge, \Rightarrow$
- $\models$

Linear temporal logic (LTL)

- “Linear” refers to linearity in time (as opposed to branching).
- Corresponding to the sequence (trajectory) of discrete states $x(0), x(1), x(2), \ldots$ of the discrete-event or hybrid system.
- We consider some property $\phi(x)$ of a sequence of states.
- ϕ is an LTL formula:

$$\phi = \text{true} \,|\, p \,|\, \neg\phi_1 \,|\, \phi_1 \wedge \phi_2 \,|\, \phi_1 \vee \phi_2$$
$$|\, \mathbf{X}\phi_1 \,|\, \mathbf{F}\phi_1 \,|\, \mathbf{G}\phi_1 |\, \phi_1 \mathbf{U} \phi_2$$

- for a given state x we write

$$x \models \phi$$

  if ϕ is true for all possible state trajectories starting at this state.

Examples of LTL formulas

$$\mathbf{G}\neg\phi$$

$$\mathbf{GF}\phi$$

$$\mathbf{FG}\phi$$

$$\mathbf{F}(\phi_1 \wedge \mathbf{XF}\phi_2)$$

CTL* (CTL mixed with LTL) supports branching

- Existential quantifiers needed
  - $\mathbf{A}$: For all
  - $\mathbf{E}$: There exists

Literature

General

- Lin, Hai, and Panos J. Antsaklis. Hybrid Dynamical Systems: Fundamentals and Methods. Advanced Textbooks in Control and Signal Processing. Cham: Springher, 2022. Chapter 3.
- Mitra, Sayan. Verifying Cyber-Physical Systems: A Path to Safe Autonomy. Cyber Physical Systems Series. Cambridge, MA, USA: MIT Press, 2021. https://sayanmitracode.github.io/cpsbooksite/about.html.

Set propagation based reachability analysis

- Althoff, Matthias, Goran Frehse, and Antoine Girard. 'Set Prop Techniques for Reachability Analysis'. Annual Review of Contr and Autonomous Systems 4, no. 1 (2021): 369–95.
- Althoff, Matthias, Niklas Kochdumper, Mark Wetzlinger, and T 'CORA 2024 Manual'. Matlab, 2023. https://tumcps.github.io/CORA/data/archive/manual/Cora20

Barrier certificates

- Prajna, Stephen, and Ali Jadbabaie. 'Safety Verification of Hybrid Systems Using Barrier Certificates'. In Hybrid Systems: Computation and Control, edited by Rajeev Alur and George J. Pappas, 477–92. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004.

Temporal logics

- Baier, Christel, and Joost-Pieter Katoen. Principles of Model Checking. Cambridge, MA, USA: MIT Press, 2008.
- Clarke, Edmund M., Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. Model Checking. 2nd ed. Cyber Physical Systems Series. Cambridge, MA, USA: MIT Press, 2018.
- Murray, Richard M, Ufuk Topcu, and Nok Wongpiromsarn. 'Lecture 3 Linear Temporal Logic (LTL)'. Lecture presented at the EECI-IGSC, Belgrade (Serbia), 9 March 2020. http://www.cds.caltech.edu/~murray/courses/eeci-sp2020/L3_ltl-09Mar2020.pdf.
- Wongpiromsarn, Nok, Richard M. Murray, and Ufuk Topcu. 'Lecture 4 Model Checking and Logic Synthesis'. Lecture presented at the EECI-IGSC, Belgrade (Serbia), 9 March 2020. http://www.cds.caltech.edu/~murray/courses/eeci-sp2020//L4_model_checking-09Mar2020.pdf.