Software Validation and Verification

Suggested Readings

About this list

- These list is just our proposal
- You can choose among the proposed works, take inspiration about the topic, or propose something different
- If you choose a work that is not from this list, send us the paper, and we will approve it or suggest something similar (if possible)

About the oral exam

- Presentations are ~30min plus questions
- Write us an email to schedule the oral exam when you are ready

Just Verification of Mutual Exclusion Algorithms, van Glabbeek et al

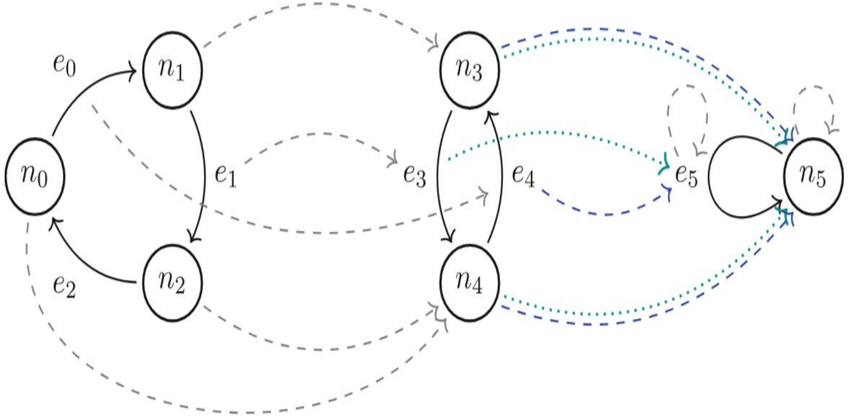
- Check correctness of a variety of mutual exclusion algorithms through model checking
- Different memory models: registers can be atomic or non-atomic
- Different assumptions to eliminate spurious counterexamples

- Find violation of correctness properties by several algorithms

- 2025 at Concur

Specification and Verification of a Linear-Time Temporal Logic for Graph Transformation, Gadducci et al

- First-order linear-time temporal logic for reasoning about the evolution of directed graphs.
- creation, duplication, merging, and deletion of elements of a graph as well as how its topology changes over time
- Its semantics is based on counterparts
- Formalization in Agda



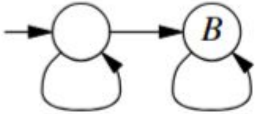
$$\psi := \mathsf{true} \mid e_1 =_E e_2 \mid n_1 =_N n_2$$

$$\phi := \psi \mid \neg\phi \mid \phi \vee \phi \mid \exists_N x.\phi \mid \exists_E x.\phi \mid \mathsf{O}\phi \mid \phi \mathsf{U} \phi \mid \phi \mathsf{W} \phi,$$

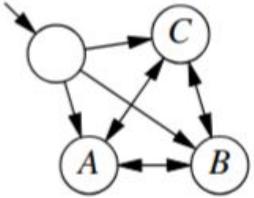
Reactive Temporal Logic, van Glabbeek

- Standard temporal logic are adequate for closed systems
- Introduce reactive temporal logic, adapted for the study of reactive systems that synchronize over actions – ingredients: labels and fairness
- From LTS to LTS with concurrency annotation
- relation validity relation ⊨ is parametrised with a set of blockable actions

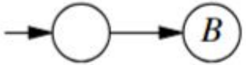
$\mathscr{E} \models \mathbf{F}B.$



Example 3 Bart is the only customer in a bar in London,
At the same time, Alice and Cameron are in a bar in Tokyo.



Example 1 Alice, Bart and Cameron stand behind a bar.



Example 2 Bart is the only customer in a bar in London

Semantics for Linear-time Temporal Logic with Finite Observations, Amjad et al

- Runtime verification: states as an infinite stream
- formulae that can be definitively said to be true or false, others are indeterminate (require further states)
- Multi-valued variant of Linear-time Temporal Logic
- Correspondence with traditional LTL semantics
  - $t \in [\![\varphi]\!]_3\ \mathrm{T} \iff \forall u \in \Sigma^\omega.\ tu \in [\![\varphi]\!]\ \mathrm{T}$
  - $t \in [\![\varphi]\!]_3\ \mathrm{F} \iff \forall u \in \Sigma^\omega.\ tu \in [\![\varphi]\!]\ \mathrm{F}$
- Proofs formalized in Isabelle/HOL.

From Natural Projection to Partial Model Checking and Back, Costa et al

- Control theory community: natural projection
  simplifying systems built from multiple components, modeled as automata.
  - synthesize local controllers from a global specification of asynchronous discrete-event system
- Verification community: partial model checking
  - for mitigating the state explosion problem with parallel processes
  - decomposing a specification, given as a formula of the μ-calculus and analysis of the individual processes independently.
- Natural projection reduces to partial model checking and, when cast in a common setting, the two are equivalent
- Partial model checking algorithm applied to natural projection

Model Checking Spatial Logics for Closure Spaces, Ciancia et al

- Traditional verification based on temporal evolution of programs, space is typically not considered
- Spatial logic, from topological interpretations of modal logics but generalized to closure spaces, i.e. considering discrete, graph-based structures.
- Model checking procedures

$$
\begin{array}{lcll}
\Phi & ::= & a & [\textsc{Atomic proposition}] \\
 & \mid & \top & [\textsc{True}] \\
 & \mid & \neg\Phi & [\textsc{Not}] \\
 & \mid & \Phi \wedge \Phi & [\textsc{And}] \\
 & \mid & \mathcal{N}\Phi & [\textsc{Near}] \\
 & \mid & \Phi\, \mathcal{S}\, \Phi & [\textsc{Surrounded}] \\
 & \mid & \Phi\, \mathcal{P}\, \Phi & [\textsc{Propagation}]
\end{array}
$$

Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties, Schmidt et al

- Symbolic analysis of security protocols
- Protocols are multiset rewriting systems
- Security properties as first-order formulas.
- Constraint-solving algorithm
- unbounded number of protocol sessions.

$$
\begin{aligned}
(1)&\quad \mathrm{dec}(\mathrm{enc}(m,k),k) \simeq m \\
(2)&\quad \mathrm{fst}(\langle x,y\rangle) \simeq x \\
(3)&\quad \mathrm{snd}(\langle x,y\rangle) \simeq y \\
(4)&\quad x * (y * z) \simeq (x * y) * z \\
(5)&\quad x * y \simeq y * x \\
(6)&\quad x * 1 \simeq x \\
(7)&\quad x * x^{-1} \simeq 1 \\
(8)&\quad (x^{-1})^{-1} \simeq x \\
(9)&\quad (x \,\hat{}\, y) \,\hat{}\, z \simeq x \,\hat{}\, (y * z) \\
(10)&\quad x \,\hat{}\, 1 \simeq x
\end{aligned}
$$

Verifying liquidity of recursive Bitcoin contracts, Bartoletti

- Liquidity in smart contracts: assets must be redeemable by someone
- Ethereum have frozen hundreds of USD millions
- Verifying liquidity on BitML, a Domain Specific Language for smart contracts with a secure compiler to Bitcoin, featuring primitives for currency transfers, contract renegotiation and consensual recursion.

They:

- First turn infinite-state semantics of BitML into a finite-state one (sound)
- Then verify liquidity by model-checking the finite-state abstraction.

The Squirrel Prover and its Logic, Baelde et al

An interactive prover for the verification of cryptographic protocols

- **Symbolic models**: modeling cryptographic messages as first-order terms, together with an equational theory that represents attacker capabilities.

  Dolev-Yao Attacker — Tools like ProVerif and Tamarin

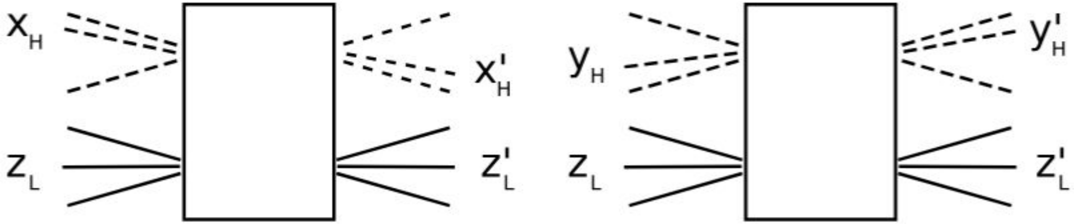
- **Computational model**: cryptographers' standard model, attackers are probabilistic polynomial-time Turing machines

  More challenging but more precise

New sound logic-based method: instead of modeling attacker by stating what the adversary can do, specify what the attacker cannot do (indistinguishablity)

Cryptographically-Masked Flows, Askarov et al

Information Flow Noninterference: no flow from secret to public data.




```
> L := H-1

> If H = 0 then
>    L := L+1

> L := enc(H)?
```

Problem: encrypted output depends on secret inputs!

… new definition to allow safe encryption, decryption, and key generation.

Epistemic temporal logic for information flow security, Balliu et al

$$\phi, \psi ::= e_1 = e_2 \mid init_x(e) \mid \phi \wedge \psi \mid \neg \phi \mid K\phi \mid \phi U \psi$$

Noninterference: no information about initial values of high identifiers (which we want to protect) can flow to final values of low identifiers (which the attacker can observe)

Declassification: acceptable, needed information leakage

Security Properties through the Lens of Modal Logic, Soloviev et al

Standard Kripke Structure (LTS)

$$w \vDash [R]\varphi \qquad \text{iff } w' \vDash \varphi \ \forall w' \text{ s.t. } (w, w') \in R$$

$$w \vDash \langle R\rangle\varphi \qquad \text{iff } \exists w' \text{ with } (w, w') \in R \text{ and } w' \vDash \varphi$$

Adapted to Security Kripke Structures (with Agents)

Models confidentiality, integrity, noninterference etc

The Hierarchy of Hyperlogics, Coenen et al

- Temporal logics only refer to a single trace or path at a time
- Temporal hyperlogics relate multiple traces or paths to each other
- Express information-flow properties such as noninterference and observational determinism.
- Logics for hyperpropertieshave been proposed: LTL and CTL$*$ with variables for traces or paths.
- Comparison of expressivity and cost for the decision problem

$$\forall \pi . \forall \pi' . \ \Box \bigwedge_{a \in AP} a_\pi \leftrightarrow a_{\pi'}$$

Model Checking for a Probabilistic Branching Time Logic with Fairness, Baier

- Non-deterministic choice between probability distributions
- The presence of non-determinism means that certain liveness properties cannot be established unless fairness is assumed
- Probabilistic branching time logic PBTL and PCTL

$$\begin{aligned}\Phi ::= tt \;\mid\; a \;\mid\; \Phi_1 \wedge \Phi_2 \;\mid\; \neg\Phi \;\mid\; [\, \exists X \; \Phi \,]_{\sqsupseteq p} \;\mid \\ [\, \forall X \; \Phi \,]_{\sqsupseteq p} \;\mid\; [\, \Phi_1 \; \exists \mathcal{U}^{\leq k} \; \Phi_2 \,]_{\sqsupseteq p} \;\mid \\ [\, \Phi_1 \; \forall \mathcal{U}^{\leq k} \; \Phi_2 \,]_{\sqsupseteq p} \;\mid\; [\, \Phi_1 \; \exists \mathcal{U} \; \Phi_2 \,]_{\sqsupseteq p} \;\mid \\ [\, \Phi_1 \; \forall \mathcal{U} \; \Phi_2 \,]_{\sqsupseteq p}\end{aligned}$$

Exploring probabilistic bisimulations, part I, Hennessy

Probabilistic and non-deterministic systems

$$P \in \text{bpCCS} ::= \mathbf{0} \quad | \quad \mu.P,\ \mu \in \text{Act}_\tau \quad | \quad P + P \quad | \quad P\ {}_p\!\oplus P,\ p \in (0,1)$$

$$s \in \text{bpCCS}_s ::= \mathbf{0} \quad | \quad \mu.P,\ \mu \in \text{Act}_\tau \quad | \quad s + s$$

- Give two bisimulations
- Distinguishing logic
- Contextual equivalence
- Prove their correspondence

Model Checking for Verification of Quantum Circuits, Ying

Model quantum circuits as a transition system

Quantum logic for the state of the qubits (with uncertainty)

CTL for the temporal evolution

Tensor network for the implementation

$$\mathcal{R}_{\mathcal{C}}(\rho) = \bigvee_{i=0}^{d-1} \operatorname{supp}\left(\mathcal{E}^i(\rho)\right) = \operatorname{supp}\left(\sum_{i=0}^{d-1} \mathcal{E}^i(\rho)\right)$$

A Calculus for Access Control in Distributed Systems, Abadi et al

Access Control with principals and their resources, logically:

- Trust, Role, Groups of Principals
- Delegation: principals on behalf of principals
- Distributed Systems: need to propagate requests and decisions

$$s ::= \mathtt{true} \mid (s \vee s) \mid (s \wedge s) \mid (s \rightarrow s) \mid A \ \mathtt{says} \ s$$

$$— \vdash (A \wedge B) \ says \ s \equiv (A \ says \ s) \wedge (B \ says \ s);$$

$$— \vdash (B|A) \ says \ s \equiv B \ says \ A \ says \ s;$$

$$— \vdash (A \Rightarrow B) \supset ((A \ says \ s) \supset (B \ says \ s)).$$

Variations in Access Control Logic, Abadi

Analysis of Different Variants for a logic of Access Control, and discussion about their adequacy w.r.t. What they should model

$$[C4] \quad \forall X.\,(A \;\mathtt{says}\; A \;\mathtt{says}\; X \to A \;\mathtt{says}\; X)$$

$$[\mathit{Unit}] \quad \forall X.\,(X \to A \;\mathtt{says}\; X)$$
$$[\mathit{Bind}] \quad \forall X, Y.\,((X \to A \;\mathtt{says}\; Y) \to (A \;\mathtt{says}\; X) \to (A \;\mathtt{says}\; Y))$$

$$[\mathit{Control\text{-}monotonicity}] \quad \forall X, Y.\, \begin{pmatrix} (X \to Y) \\ \to \\ ((A \;\mathtt{controls}\; X) \to (A \;\mathtt{controls}\; Y)) \end{pmatrix}$$

Local Action and Abstract Separation Logic, Calcagno et al

Hoare Logic: {p} C {q} – if p is true before executing C then q is true after executing C (if C terminates)

Separation logic: Hoare's logic with mutating data structures in memory

$$\frac{\{p\}\,C_1\,\{q\} \quad \{q\}\,C_2\,\{r\}}{\{p\}\,C_1; C_2\,\{r\}}$$

$$\frac{\{p\}\,C\,\{q\}}{\{p * r\}\,C\,\{q * r\}}\ \mathsf{FrameRule}$$

$$\frac{\{p_1\}\,C_1\,\{q_1\} \quad \{p_2\}\,C_2\,\{q_2\}}{\{p_1 * p_2\}\,C_1 \parallel C_2\,\{q_1 * q_2\}}\ \mathsf{ConcurrencyRule}$$

Abstracts from RAM and similar models in Separation Logic

Incorrectness Logic, O'Hearn

Logic for program incorrectness (dual of Hoare's logic of correctness)

Correctness with over-approximation

$$\{pre\text{-}condition\}\mathbf{code}\{post\text{-}condition\}$$

Incorrectness with under-approximation

$$[presumption]\mathbf{code}[result]$$

$$\frac{\{p\}C\{q \wedge r\}}{\{p\}C\{q\}} \qquad \frac{[p]C[q_1 \vee q_2]}{[p]C[q_1]}$$

Outcome Logic: A Unifying Foundation for Correctness and Incorrectness Reasoning, Zilberstein et al

- Abstract over reachability (e.g. of faulty states)
- Approach parametric on results, kind of computations and assertion logic

Triple Name	Syntax		Semantics				
Hoare Logic	$\vDash \{P\}\ C\ \{Q\}$	iff	$\forall \sigma \vDash P.$	$\forall \tau.$	$\tau \in [\![C]\!]\,(\sigma)$	$\Rightarrow$	$\tau \vDash Q$
Incorrectness Logic (IL) / Reverse Hoare Logic (RHL)	$\vDash [P]\ C\ [Q]$	iff	$\forall \tau \vDash Q.$	$\exists \sigma.$	$\tau \in [\![C]\!]\,(\sigma)$	and	$\sigma \vDash P$
Outcome Logic (OL)	$\vDash \langle P \rangle\ C\ \langle Q \rangle$	iff	$\forall m.$	$m \vDash P$	$\Longrightarrow$		$[\![C]\!]^{\dagger}(m) \vDash Q$

Fig. 1. Semantics of triples where P and Q are logical formulae, C is a program, Σ is the set of all program states, $\sigma, \tau \in \Sigma$, and $[\![C]\!] : \Sigma \to 2^{\Sigma}$ is the reachable states function. In the last line of the table, M is a monad, $m \in M\Sigma$ and $[\![C]\!]^{\dagger} : M\Sigma \to M\Sigma$ is the monadic lifting of $[\![\cdot]\!] : \Sigma \to M\Sigma$.

Linear logic as a logic of computations, Kanovich

Linear Logic: constraints on weakening and contraction rules

LL Fragment(s)
VS
Program (graphs)
VS
Petri Nets

$$\mathbf{I}\quad \frac{}{A \vdash A}$$

$$\mathbf{L}\otimes\quad \frac{\Sigma, A, B \vdash C}{\Sigma, (A \otimes B) \vdash C}$$

$$\mathbf{L}\oplus\quad \frac{\Sigma, A \vdash C \qquad \Sigma, B \vdash C}{\Sigma, (A \oplus B) \vdash C}$$

$$\mathbf{C!}\quad \frac{\Sigma, !A, !A \vdash C}{\Sigma, !A \vdash C}$$

$$\mathbf{L}\multimap\quad \frac{\Sigma_1 \vdash A \qquad B, \Sigma_2 \vdash C}{\Sigma_1, (A \multimap B), \Sigma_2 \vdash C}$$

$$\mathbf{R}\otimes\quad \frac{\Sigma_1 \vdash A \qquad \Sigma_2 \vdash B}{\Sigma_1, \Sigma_2 \vdash (A \otimes B)}$$

$$\mathbf{L!}\quad \frac{\Sigma, A \vdash C}{\Sigma, !A \vdash C}$$

$$\mathbf{W!}\quad \frac{\Sigma \vdash C}{\Sigma, !A \vdash C}$$

Compositional Symbolic Execution for Correctness and Incorrectness Reasoning, Lööw et al

- Tool for symbolic execution
- Symbolic execution has scalability problem
- Separation logic can help in determining how to work compositionally on smaller components of the system to verify
- Supports bot Correctness and Incorrectness reasoning
- Interesting details about the implementation of a real world verification tool

Compositional Symbolic Execution for the Next 700 Memory Models (Extended Version), Lööw et al

- compositional symbolic execution exploits separation logic for compositional verification
- Separation logic speaks of memory accesses and modification
- Different assumptions on the memory model can change the result of the analysis
- Formal foundation for memory-model-parametric composition symbolic execution platforms.
- The model is mechanised in Rocq
- Model instantiating to concrete useful examples, e.g. C language

Non-Termination Proving: 100 Million LoC and Beyond, Vanegue et al

Halting problem: we cannot verify precisely if a program P terminates with input i

- Tool for proving non-termination of programs (of course with approximation)
- Focus on large programs by using a compositional approach
- Look for repeating states (loops) but in an abstract semantics
- Combines symbolic execution (for abstraction) and separation logic (for compositionality)
- For soundness, repeating states are under-approximated therefore incorrectness logic is needed instead of traditional Hoare logic