SMT-Based Satisfiability of Temporal Logic

Stefano Tonetta
Fondazione Bruno Kessler
tonettas@fbk.eu

SCARE Workshop 2017
October 25, 2017
Bad Zwischenahn, Germany

The CITADEL project receives funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 700665.

Part of the material have been taken from presentations of A. Cimatti and A. Griggio

Outline

- Motivations
- Temporal Satisfiability Modulo Theories
  - Models of Time
  - First-Order Temporal Logic
  - Temporal Satisfiability Modulo Theories
  - Extensions and Equi-SAT Reductions
- SMT-Based Satisfiability
  - Symbolic Model Checking
  - LTL Model Checking
  - IC3IA
- Conclusions

The ES Unit at FBK

Fondazione Bruno Kessler (FBK) is a research non-profit public interest entity, located in Trento, Italy

Embedded Systems (ES) Unit

- Head: Alessandro Cimatti
- People: 25-30, research staff, programmers, PhD students, technologists
- Technology transfer: Intel, Boeing, NASA, Ansaldo, others under NDA

Topics:

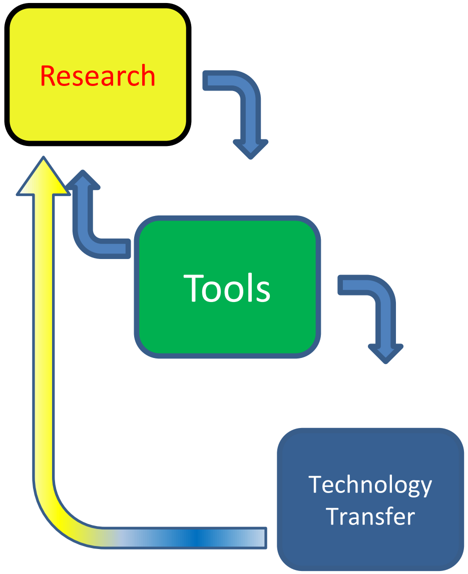
- Model checking
- Design automation with formal methods
- Autonomous reasoning and control

Beyond Model Checking:

- Requirements analysis
- Contract-based design
- Safety analysis (in case of faults)
- Fault detection, identification and recovery (FDIR)
- Planning

Strategy:

- Basic research
- Tool development
- Technology transfer

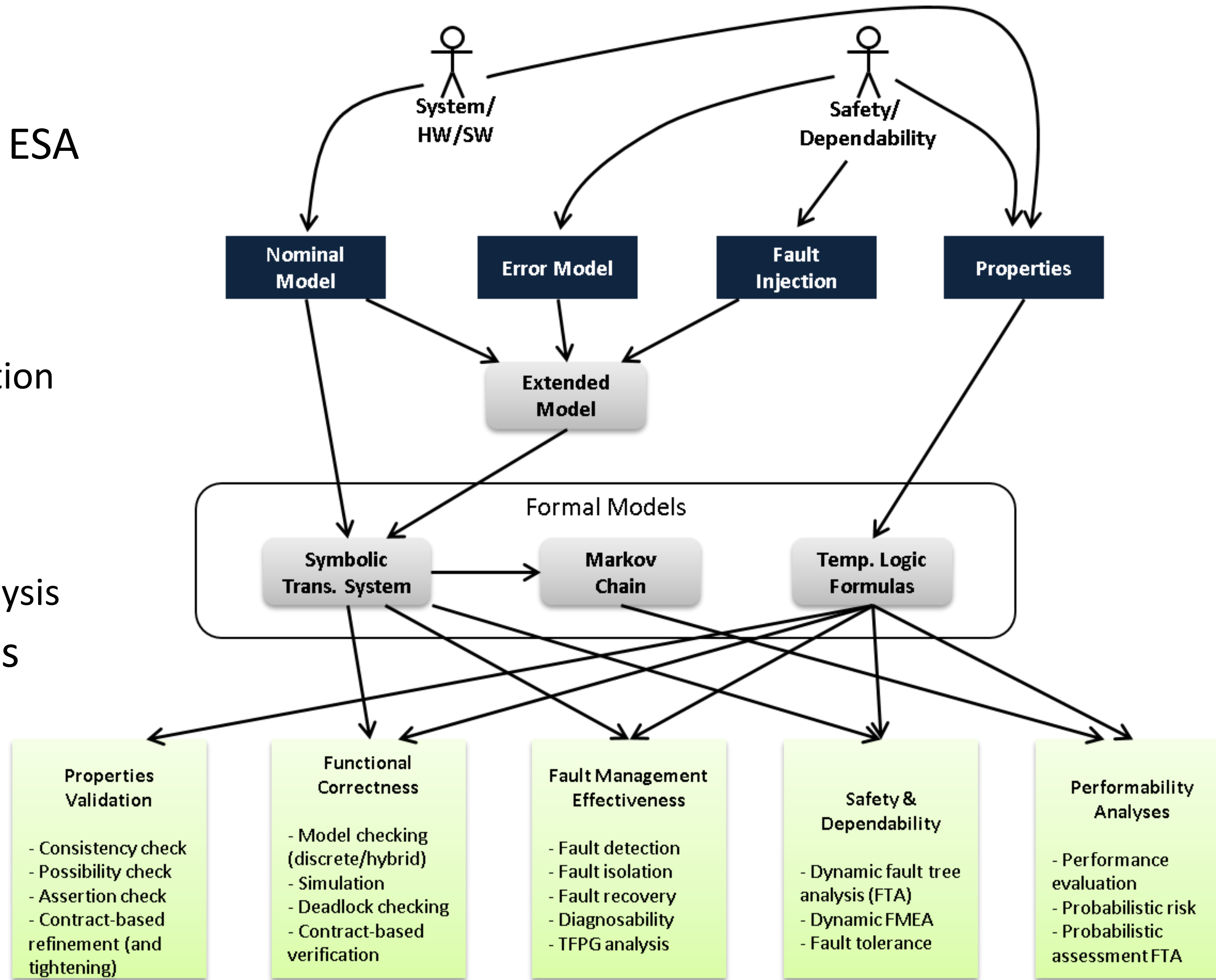


ES Tools

- NuSMV
  - Open Source, widely used: +10 years, used in +25 external projects, +500 download/month
  - Discrete-Time Finite-state systems
  - Model Checking of CTL, LTL, PSL properties (BDD and SAT)
- nuXmv
  - Discrete-Time Infinite-state Systems with Integers, Reals, and Uninterpreted Functions types
  - Implements SMT-based verification techniques, through a tight integration with MathSAT5
  - Extended model checking algorithms (Interpolation-based, IC3-based, K-liveness, …)
  - Parameter synthesis
  - Validation of properties/requirements
- HyCOMP
  - Allows modeling and verification of Asynchronous Hybrid Systems with event-based synchronization
  - Verification of Invariant properties and LTL properties
  - Discretization of hybrid systems to the nuXmv language
- xSAP
  - Safety Analysis of Discrete Infinite Synchronous Systems
  - Automatic model extension with fault specifications
  - Fault Tree Analysis (FTA) and generation of Minimal Cut Sets (MCS) for dynamic systems
  - Fault propagation analysis based on Timed Failure Propagation Graphs (TFPG)
  - Fault Detection and Isolation (FDI) design and Diagnosability Analysis
- OCRA
  - Contract Refinement
  - Contract-based compositional verification of SMV (nuXmv) and HyDI (HyCOMP)
  - Contract-based fault-tree generation
  - Validation of contracts
  - Supports synchronous and asynchronous composition

COMPASS

- Toolset for HW/SW co-design
- Developed mainly in ESA projects
- **Main functions:**
  - Requirements validation
  - Functional verification
  - Automated fault extension
  - Safety assessment
  - FDIR analysis
  - Performability analysis
- ES Tools integrated as backed

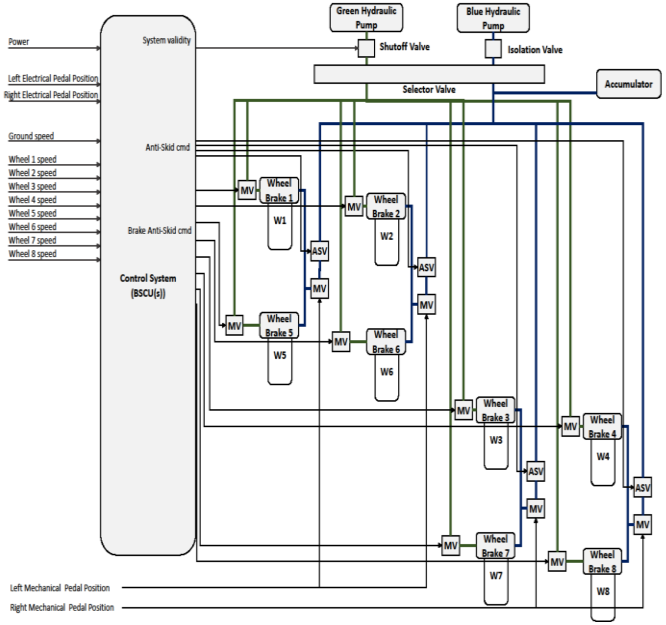


Projects with the European Space Agency (2008-2016)

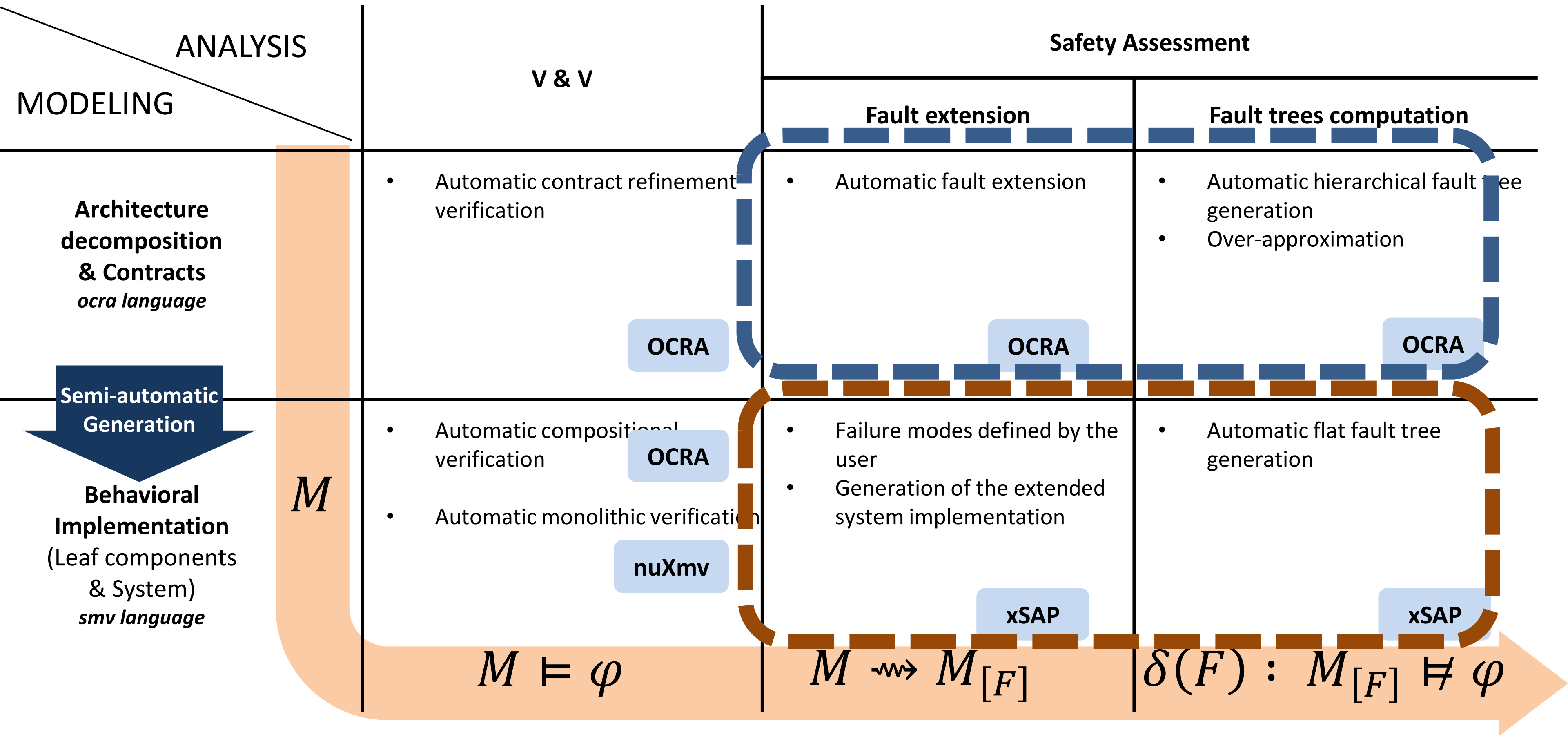
- Autonomy
  - OMC-ARE
  - IRONCAP
- Model-based co-design of Hw-Sw
  - COMPASS
  - AUTOGEF
  - FAME
  - FOREVER
  - HASDEL
  - CATSY
  - COMPASS3

AIR6110 Wheel Brake System

- Joint scientific study with Boeing
- Aerospace Information Report 6110:
  - Traditional Aircraft/System Development Process Example
  - Describes the development process of a Wheel Brake System for a fictional dual-engine aircraft
  - Analyzes different architectures with standard informal techniques
- Objectives:
  - Analyze the system safety through formal techniques
  - Repeat AIR6110 steps with formal techniques to demonstrate the usefulness and suitability of formal techniques for improving the overall traditional development and supporting aircraft certification
- Control brake for aircraft wheels
- Redundancy
  - Multiple BCSU
  - Hydraulic plants
- Functions
  - Asymmetrical braking
  - Antiskid
    - Single wheel/coupled
    - depending on control mode



WBS: Adopted approach



WBS: Conclusion

- Results:
  - Cover the process described in AIR6110 with formal methods
  - Production of modular descriptions of 5 architectures variants
    - Analysis of their characteristics in terms of a set of requirements expressed as properties
    - Production of more than 3000 fault trees
    - Production of reliability measures
  - Detection of an unexpected flaw in the process
    - Detection of the wrong position of the accumulator earlier in the process
- Highlights:
  - Going from informal to formal allows highlighting the missing information of the AIR6110 to reproduce the process
  - Automated and efficient engines as IC3 is a key factor
  - OCRA modular modeling allows a massive reuse of the design through architectures variant

Wide Literature on Contracts for CPS

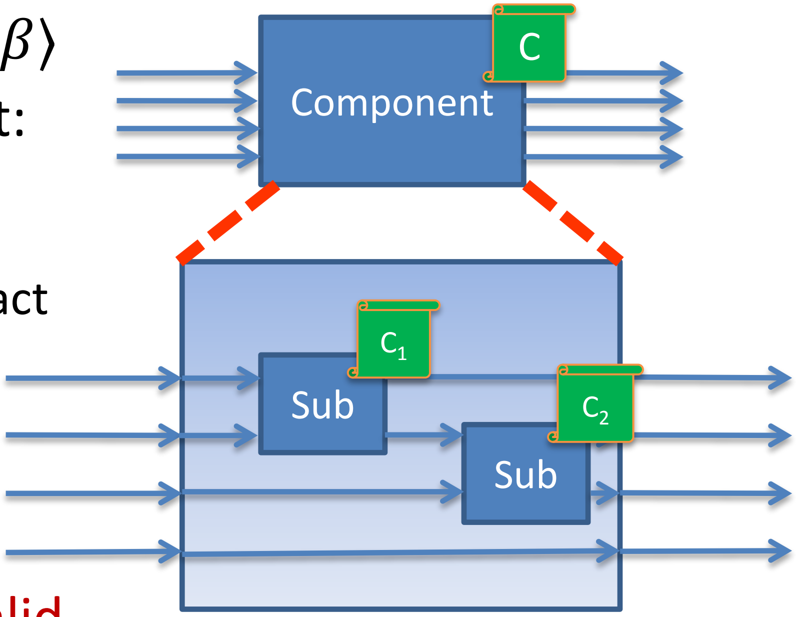
- Contracts first conceived for OO programing [Meyer, 82].
- New challenge given by CPS
  - E.g., Alberto Sangiovanni-Vincentelli, Werner Damm and Roberto Passerone. Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems. European Journal of Control, 18(3):217-238, 2012.
- Contracts theories such as
  - Albert Benveniste, Benoît Caillaud, Roberto Passerone: A Generic Model of Contracts for Embedded Systems. CoRR abs/0706.1456 (2007)
  - Sebastian S. Bauer, Alexandre David, Rolf Hennicker, Kim Guldstrand Larsen, Axel Legay, Ulrik Nyman, Andrzej Wasowski: Moving from Specifications to Contracts in Component-Based Design. FASE 2012: 43-58
  - Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto L. Sangiovanni-Vincentelli, Werner Damm, Thomas A. Henzinger, and Kim G. Larsen. Contracts for Systems Design. Rapport de recherche RR-8147, INRIA, 2012
  - Alessandro Cimatti, Stefano Tonetta: A Property-Based Proof System for Contract-Based Design. EUROMICRO-SEAA 2012: 21-28
- Tool support such as
  - Darren D. Cofer, Andrew Gacek, Steven P. Miller, Michael W. Whalen, Brian LaValley, Lui Sha: Compositional Verification of Architectural Models. NASA Formal Methods 2012: 126-140
  - Alessandro Cimatti, Michele Dorigatti, Stefano Tonetta: OCRA: A tool for checking the refinement of temporal contracts. ASE 2013: 702-705

Trace-based contract refinement

- The set of contracts $\{C_i\}$ refines C with the connection γ ($\{C_i\} \preccurlyeq_\gamma C$) iff for all correct implementations Imp_i of C_i and correct environment Env of C:
  - The composition of $\{Imp_i\}$ is a correct implementation of C.
  - For all k, the composition of Env and $\{Imp_i\}_{i \neq k}$ is a correct environment of C_k.
- $C_1 = \langle \alpha_1, \beta_1 \rangle, \dots, C_n = \langle \alpha_n, \beta_n \rangle, C = \langle \alpha, \beta \rangle$
- Proof obligations for contract refinement:
  - $\gamma\left(\left(\bigwedge_{1 \leq j \leq n}(\alpha_j \rightarrow \beta_j)\right) \rightarrow (\alpha \rightarrow \beta)\right)$

    The subcomponents entail the parents' contract
  - $\gamma\left(\left(\bigwedge_{1 \leq j \leq n, j \neq i}(\alpha_j \rightarrow \beta_j)\right) \rightarrow (\alpha \rightarrow \alpha_i)\right)$

    The subcomponent's assumption is entailed by the subcomponent context
- $\{C_i\} \preccurlyeq_\gamma C$ iff the proof obligations are valid



OCRA tool support

- OCRA supports contracts where assumptions and guarantees are expressed in extensions of Linear-time Temporal Logic
- Supports both discrete or super-dense (hybrid) models of time
- Supports both synchronous and asynchronous composition of components
- Integrated with nuXmv for validity checking and model checking in case of discrete time
- Integrated with HyCOMP for validity checking and model checking in case of super-dense time
- Integrated with xSAP for contract-based fault-tree generation
- Integrated with CASE tools (AutoFocus3, CHESS, COMPASS)

Formalization of Component Requirements

- Components are specified as black boxes
- Visible traces of input/output data/events
- Linear-time Temporal Logic (LTL) well accepted formalism to specify component properties
- Embedded systems application needs
  - Rich data ⇨ first-order
  - Combine continuous and discrete components ⇨ hybrid (super-dense) model of time
- Moreover, component input/output events occur at different points of time
  - Need to relate data at different point of time (like storing values in extra variables and freeze them along time)
  - Need to constrain time difference (bounded response, periodicity, …)
- Reasoning needs engine to solve validity/satisfiability queries
  - Refinement proof obligation, consistency of assertions, …

Examples

- The counter is increased whenever a new valid message is received
  - $always\ (Valid(message) \rightarrow next(counter) = counter + 1)$
- The user shall switch the dispatcher to high before entering high-level data.
  - $always\ (highLevel(data) \rightarrow cmd@last(switch) = toHigh)$
- The train trip shall issue an emergency brake command, which shall not be revoked until the train has reached standstill and the driver has acknowledged the trip (ETCS SRS Sec. 3.13.8.2)
  - $always\ \left(trainTrip \rightarrow \left(brake\ until\ (speed = 0 \wedge driverAck)\right)\right)$

Temporal Satisfiability Modulo Theory

Many-Sorted First-Order LTL

- Many-Sorted Signature $\Sigma = \langle \Omega, \Phi, \Pi, Z, \gamma \rangle$
  - Ω set of constant symbols
  - Φ set of function symbols
  - Π set of predicate symbols
  - Z set of sorts
  - γ assigns sorts to symbols
- Σ-variables $V = \dot{\bigcup}_{\zeta \in Z} V_\zeta$
  - $\gamma(x) = \zeta$ iff $x \in V_\zeta$
- Terms: $u \coloneqq c \mid x \mid f(u, \dots, u)$
  - If $u = c$ and $c \in \Omega$, then u is a term and $\gamma(u) = \gamma(c)$
  - If $u = x$ and $x \in V_\zeta$, then u is a term and $\gamma(u) = \zeta$
  - If $u = f(u_1, \dots, u_n)$, $u_1, \dots, u_n$ are terms, $f \in \Phi$, and $\gamma(f) = \langle \gamma(u_1), \dots, \gamma(u_n), \zeta \rangle$, then u is a term and $\gamma(u) = \zeta$
- Temporal formulas: $\phi \coloneqq p(u, \dots, u) \mid \phi \wedge \phi \mid \neg\phi \mid \phi \widetilde{U} \phi \mid \phi \tilde{S} \phi$
  - where $p(u_1, \dots, u_n)$ is a formula if $u_1, \dots, u_n$ are terms, $p \in \Pi$, and $\gamma(p) = \langle \gamma(u_1), \dots, \gamma(u_n) \rangle$
- Quantifier-free fragment

Example:

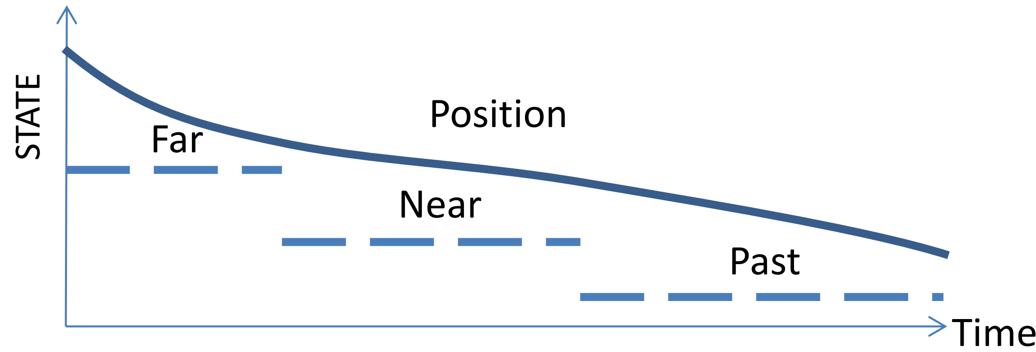
$$\Omega_r \coloneqq \{0,1\}$$
$$\Phi_r \coloneqq \{+, -, \times, f\}$$
$$\Pi_r \coloneqq \{=, <\}$$
$$Z \coloneqq \{R\}$$
$$V_r \coloneqq \{x, y\}$$
$$\big(x < f(x)\big) \widetilde{U} (x = y)$$

States and Traces

- State s $= \langle M, \mu \rangle$
  - Σ-structure M
  - Assignment μ to variables V
- Time structure $\tau = \langle T, 0, <, v \rangle$
  - T is the time domain, set of time points
  - $0 \in T$ is the initial time point
  - $<$ is a total order over T
  - $v: T \rightarrow \mathbb{R}_0^+$ is the real-time value of the time point
- Trace $\sigma = \langle M, \tau, \overline{\mu} \rangle$
  - Σ-structure M
  - Time model $\tau = \langle T, 0, <, v \rangle$
  - $\overline{\mu}: T \rightarrow M^V$
    - M^V set of states with same structure M
- $\sigma(t) \coloneqq \overline{\mu}(t)$
- Σ symbols are rigid, same interpretation in $\sigma(t)$ and $\sigma(t')$
  - As in SMT, some symbols are interpreted by the theory while others are uninterpreted
  - Uninterpreted symbols are parameters in the temporal setting

$$M = \langle \mathbb{R}, 0_{\mathbb{R}}, 1_{\mathbb{R}}, +_{\mathbb{R}}, -_{\mathbb{R}}, \times_{\mathbb{R}}, <_{\mathbb{R}}, f_M \rangle$$
$$\forall z, f_M(z) \coloneqq -_{\mathbb{R}} z$$
$$\forall t, \sigma(t)(x) \coloneqq t -_{\mathbb{R}} 1_{\mathbb{R}}$$
$$\forall t, \sigma(t)(y) \coloneqq -_{\mathbb{R}} t$$

Uniform Structure of Time



- Discrete time:
  - $T = \mathbb{N}$
  - $v(0), v(1), v(2), \ldots$

  weakly-monotonic diverging

- Dense time:
  - $T = \mathbb{R}_0^+$
  - $v(t) = t$

- Super-dense:
  - $T \subset \mathbb{N} \times \mathbb{R}_0^+$ such that $I_n = \{t \mid \langle n, t \rangle \in T\}$ is a time sequence
  - $v(\langle n, t \rangle) = t$

Temporal Satisfiability Modulo Theories

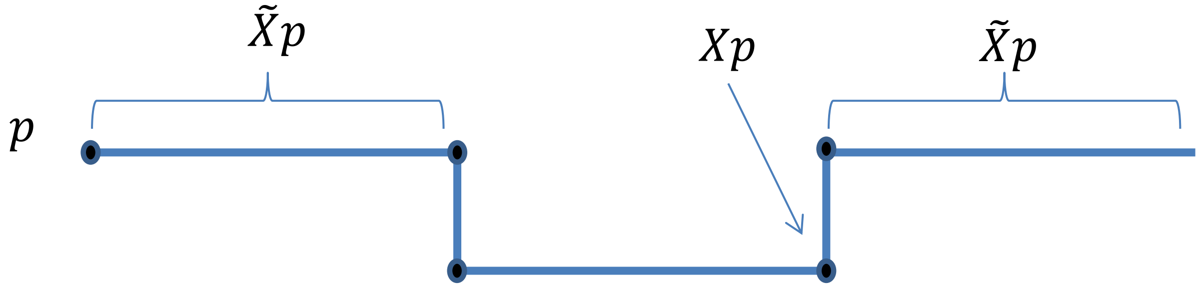
- $\sigma, t \vDash p$ iff $\sigma(t) \vDash p$
  - Implicitly modulo Σ-theory $\mathcal{T}$
    - E.g., theory of reals $\mathcal{T}_{\mathbb{R}}$
  - Finite variability assumption:
    - For every bounded interval I, the interpretation of p changes only finitely many times
- $\sigma, t \vDash \phi_1 \widetilde{\mathrm{U}} \phi_2$ iff there exists $t' > t, \sigma, t' \vDash \phi_2$ and for all $t'', t < t'' < t', \sigma, t'' \vDash \phi_1$
- $\sigma, t \vDash \phi_1 \tilde{S} \phi_2$ iff there exists $t' < t, \sigma, t' \vDash \phi_2$ and for all $t'', t' < t'' < t, \sigma, t'' \vDash \phi_1$

Standard Abbreviations

- Non-strict until (standard for discrete time)
  - $\phi_1 U \phi_2 \coloneqq \phi_2 \vee \left(\phi_1 \wedge \phi_1 \widetilde{U} \phi_2\right)$
- Non strict since
  - $\phi_1 S \phi_2 \coloneqq \phi_2 \vee \left(\phi_1 \wedge \phi_1 \tilde{S} \phi_2\right)$
- In the future ϕ
  - $F\phi \coloneqq \top U \phi$
- Always ϕ
  - $\mathrm{G}\phi \coloneqq \neg F \neg \phi$
- In the past ϕ
  - $\mathrm{P}\phi \coloneqq \top \mathrm{S} \phi$ (sometimes denoted by O)
- Always in the past ϕ (historically)
  - $\mathrm{H}\phi \coloneqq \neg \mathrm{P} \neg \phi$

Next Operator

- $X\phi \coloneqq \bot\ \widetilde{U}\phi$
  - Can be true only on discrete steps
  - With discrete time, it is true in t iff ϕ is true in $t+1$
  - Always false with dense time
  - With super-dense time, it is true in $\langle n,t\rangle$ iff $\langle n+1,t\rangle$ is also in T and ϕ is true in $\langle n+1,t\rangle$
- $\tilde{X}\phi \coloneqq \phi\widetilde{U}\top \wedge \neg X\top$
  - Always false in discrete steps
  - Always false with discrete time
  - With dense time, it is true in t iff ϕ is true in (t,t') for some t'
  - With super-dense time, it is true in $\langle n,t\rangle$ iff $\langle n+1,t\rangle$ is not in T and ϕ is true in $\langle n,(t,t')\rangle$ for some t'



Next and ITE Functions

- With discrete time $next(u)$ is the value of u at time $t+1$
  - $\sigma'(t)\big(next(u)\big) \coloneqq \sigma(t+1)(u)$
  - $\sigma, t \vDash p$ iff $\sigma(t) \cdot \sigma'(t) \vDash \mathrm{p}$
    - Where $\langle M, s\rangle \cdot \langle M, s'\rangle = \langle M, s \cup s'\rangle$
  - No counterpart in dense time
- We also use if-then-else ite
  - $\sigma(t)\big(ite(\phi, u_1, u_2)\big) \coloneqq \begin{cases} \sigma(t)(u_1) & if\ \sigma, t \vDash \phi \\ \sigma(t)(u_2) & if\ \sigma, t \nvDash \phi \end{cases}$

Event-Freezing Functions

- $u@\tilde{F}\phi$ ("u at next ϕ") is the value of u at the next point in the future where ϕ holds
- $u@\tilde{P}\phi$ ("u at last ϕ") is the value of u at the last point in the past where ϕ holds
- In discrete time, if there exists a point in the future, there exists a first point

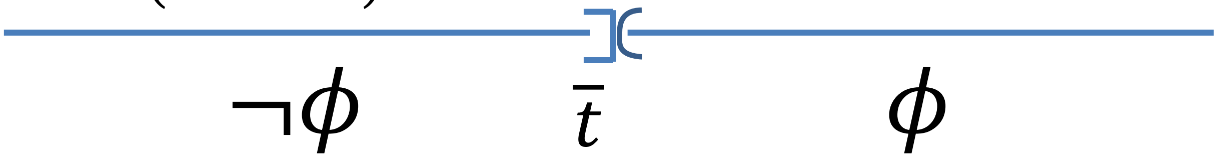
$$F\phi \equiv \neg\phi U \phi$$

- In dense time, if ϕ is true in (t, ∞) there is no first point
- But there is a first point in which ϕ or $\tilde{X}\phi$ holds

$$F\phi \equiv \neg\phi U(\phi \vee \tilde{X}\phi)$$

  – Note we are assuming finite variability

$$\sigma(t)\left(u@\tilde{F}(\phi)\right) \quad = \quad \sigma(\bar{t})(u)$$



Event-Freezing Functions

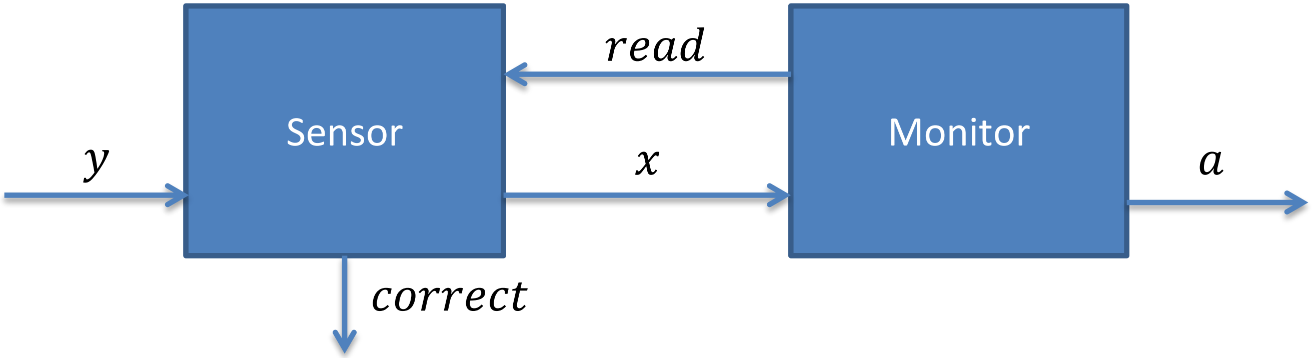
- $\sigma(t)\left(u@\tilde{F}\phi\right) = \sigma(t')(u)$ iff
  - there exists $t' > t$ such that $\sigma, t' \vDash \phi$ and for all $t'', t < t'' < t'$, $\sigma, t'' \nvDash \phi$
  - there exists $t' \geq t$ such that $\sigma, t' \vDash \tilde{X}\phi$ and for all $t'', t < t'' \leq t'$, $\sigma, t'' \nvDash \phi$
  - Else, $\sigma(t)\left(u@\tilde{F}\phi\right) = def_{u@\tilde{F}\phi}$
- $def_{u@\tilde{F}\phi}$ new symbol added to the signature Σ
- Similarly for $u@\tilde{P}\phi$
- Non-strict version as abbreviation: $u@F\phi \coloneqq ite(\phi, u, u@\tilde{F}\phi)$
- Counting: $u@\tilde{F}^{k+1}\phi \coloneqq \left(u@\tilde{F}\phi\right)@\tilde{F}^{k}\phi$

Adding Explicit Time

- We add an explicit variable $time$ to represent the time elapsed from the initial state $(\sigma(t)(time) \coloneqq v(t))$
- $time@\tilde{F}\phi$ is the time at the next point in the future where ϕ holds (call "time_until" in ocra)
- Can express different real-time properties
- Event clock logic
  - Next time in which ϕ holds is in the interval I
  - $\rhd_I \phi \coloneqq time@\tilde{F}\phi - time \in I \wedge \neg\phi\tilde{U}\phi$
- Metric temporal logic
  - ϕ will occur within p
  - $\tilde{F}_{<p}\phi \coloneqq time@\tilde{F}\phi - time < p \wedge \tilde{F}\phi$
- Counting logic
  - ϕ will occur k times within p
  - $\vec{C}^k_{<p}\phi \coloneqq time@\tilde{F}^k\phi - time < p \wedge \tilde{F}^k\phi$
- p can be a parameter!

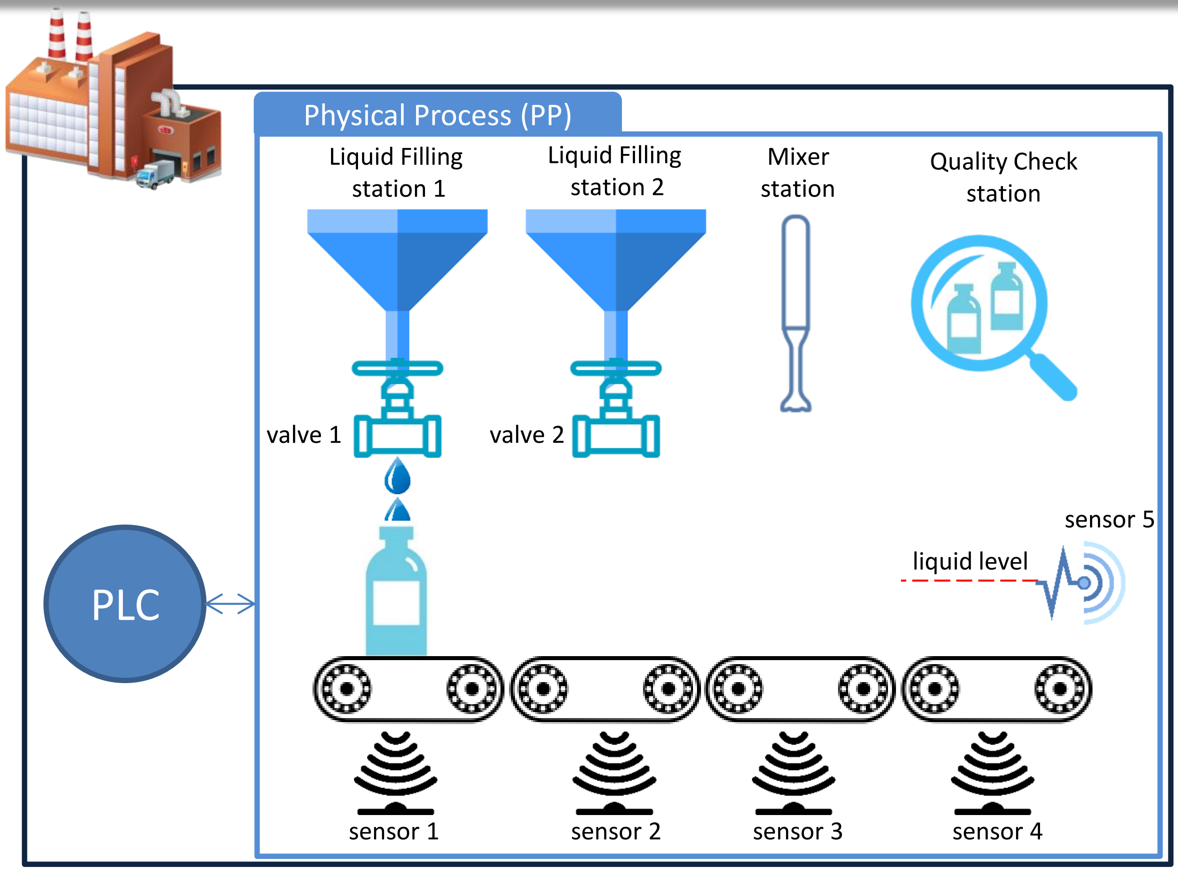
A Sensor Example

- Output x always equal to the last correct input y: $G\big(x = y@P(correct)\big)$
- Permanent failure: $G(\neg correct \rightarrow G(\neg correct))$
- Read periodically: $p > 0 \wedge read \wedge G\big(read \rightarrow \rhd_{=\mathrm{p}}\ read\big)$
- Trigger an alarm when last two readings are not equal: $G\left(a \leftrightarrow x@\tilde{P}(read) \neq x@\tilde{P}^2(read)\right)$
- Property: $G\big(\neg correct \rightarrow F_{\leq 2*p}\, a\big)$



A Factory Example

- PLC interface:
  - Load bottle
  - Move belt
  - Open/close valve1/valve2
  - Sense liquid level
- Property:
  - The final liquid level in the bottle is equal the filling rate times the time difference between closing and opening the valve



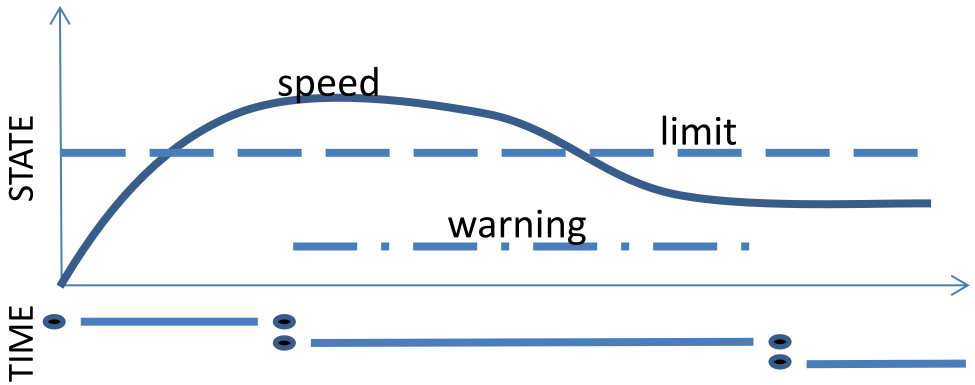
- One bottle at a time:
  - $G(sensor5 = bottleOK \rightarrow ingredient1 = fillingRate * (time@P(valve1Close) - time@P(valve1Open)))$
- Many bottles:
  - $G(sensor5 = bottleOK \rightarrow ingredient1 = fillingRate * (time@P(valve1Close) - time@P(valve1Open))@P^3(moveBelt))$

Extension for hybrid systems

- In hybrid systems, some variables evolve continuously in time
- The trace σ is continuous and differentiable almost everywhere
- Super-dense time: $T \subset \mathbb{N} \times \mathbb{R}_0^+$ such that $I_n = \{t \mid \langle n, t \rangle \in T\}$ is a time sequence
- Discontinuous changes are allowed on discrete steps time points (i.e., in $\langle n, t \rangle$ if $\langle n+1, t \rangle$ is also in T)
- Add predicates over derivatives: $der(X) \geq 0$
- Restricted to linear constraints over derivatives
  - No comparison with variables (e.g., $der(x) = x$)
  - To allow reduction to LTL with linear SMT constraints

Examples

- The derivative of “x” is always less than 2:
  - $G(der(x) < 2)$
- Whenever “a” holds, the derivative of “x” is zero
  - $G(a \rightarrow der(x) = 0)$
- Whenever “a” holds, “b” remain true until the derivative of “x” is less or equal to 5
  - $G\big(a \rightarrow (b\ U\ der(x) \leq 5)\big)$



$G\ (speed > limit \rightarrow F\ warning)$

Equi-Satisfiability Reduction

- Consider ϕ with super-dense semantics (dense time is a subcase)

1. Discretize time
   - Equi-satisfiable LTL formula with $next$
2. Remove event-freezing functions
   - Equi-satisfiable LTL formula with monitors
3. Check validity with SMT-based model checking

Discretization idea

- Exploit finite variability
- Split the trace into intervals that are fine for the subformulas of input formula ϕ
- Consider only singular and open intervals
- In case of open interval, the discrete state is a sample for some timepoint in the interval
- One discrete timepoint for each interval
- Introduce extra variables to represent the super-dense structure
  - ι is true iff the current discrete timepoint corresponds to a singular interval
  - δ is a real variable that stores the time elapsed between current and next timepoint

$$G((\iota \wedge X\iota \wedge \delta = 0) \vee (\iota \wedge X\neg\iota \wedge \delta > 0) \vee (\neg\iota \wedge X\iota \wedge \delta > 0)$$

- In case of hybrid constraints add also
  - Constraints over derivatives.
  - Continuity of predicates along continuous evolution (e.g., $(\neg\iota \wedge x<0) \rightarrow x\leq 0$)

Rewriting the Event-Freezing Functions

- Replace $u@F\phi$ with a prophecy variable $p_{u@F\phi}$
- Add monitoring conditions:
  - $G\left(F\phi \rightarrow (\neg\phi \wedge next(p_{u@F\phi}) = p_{u@F\phi})U(\phi \wedge p_{u@F\phi} = u)\right)$
  - $G(G\neg\phi \rightarrow p_{u@F\phi} = def_{u@F\phi})$

SMT-Based Temporal Satisfiability

Symbolic Transition System

- As before
  - Given a many-sorted first-order signature Σ and a Σ-theory $\mathcal{T}$
  - Formulas implicitly mean Σ-formulas and $\vDash$ implicitly means $\vDash_{\mathcal{T}}$
- A transition system is a tuple $\langle V, I, T \rangle$ where:
  - V is a finite set of variables
  - The set of initial states represented by the formula $I(V)$
  - The transition relation represented by the formula $T(V, V')$ where $V' \coloneqq \{next(v) \mid v \in V\}$
- A state $s = \langle M, \mu \rangle$ is given by Σ-structure M and an assignment μ to variables V
- Trace $\sigma = \langle M, \overline{\mu} \rangle$ (implicitly with a discrete time model)
  - Σ-structure M
  - $\overline{\mu}: \mathbb{N} \rightarrow M^V$
- In other words, a trace is a sequence $s_0, s_1, s_2, \ldots$ of states with the same structure M
  - A finite trace is a finite sequence
- A trace of the system S is a trace $s_0, s_1, s_2, \ldots$ of states such that $s_0 \vDash I$ and for all i, $s_i \cdot s'_i \vDash T$

Example:
$V = \{x, y\}$
I:=x=1∧y=1
T:=x'=x+1∧y'=y+x

Model Checking

- Given an LTL formula ϕ, $M \vDash_{LTL} \phi$ iff for every trace σ of M, $\sigma \vDash \phi$
- A state s is reachable iff there exists a trace $s_0, s_1, \ldots, s_k$ such that $s = s_k$
- A formula $P(V)$ is an invariant ($M \vDash_{INV} P$) iff for all traces $s_0, s_1, \ldots, s_k$, for all i, $s_i \vDash P$
- Equivalent to say that no state in $\neg P$ is reachable
- Similar to LTL property $G(P)$ but slightly different:
  - Invariants defined over finite traces
  - LTL defined over infinite traces
  - There may be a counterexample σ for the invariant P and for LTL $G(P)$ because σ cannot be extended to an infinite trace
  - If $M \vDash_{INV} P$ then $M \vDash_{LTL} G(P)$
  - If M is deadlock free and $M \vDash_{LTL} G(P)$, then $M \vDash_{INV} P$

LTL Satisfiability as Model Checking

- Consider universal model $M_U \coloneqq \langle \mathrm{V}, \top, \top \rangle$
- ϕ is satisfiable iff $M_U \nvDash \neg\phi$
- Automata-theoretic approach:
  - Build a transition system M_ϕ with a fairness condition f_ϕ, such that $\sigma \vDash GF f_\phi$ iff $\sigma \vDash \phi$ for every trace σ of M_ϕ
  - ϕ is satisfiable iff $M_\phi \nvDash FG \neg f_\phi$
- Standard techniques to build M_ϕ [Vardi95] in case of propositional LTL
- In case of first-order LTL:
  - $\hat{\phi} \coloneqq \phi[p \mapsto v_p]_{p \in Sub(\phi)}$ where v_p are fresh Boolean variables
  - If $M_{\hat{\phi}} = \langle I_{\hat{\phi}}, T_{\hat{\phi}} \rangle$ then $M_\phi \coloneqq \langle I_{\hat{\phi}}, T_{\hat{\phi}} \wedge \bigwedge_{p \in Sub(\phi)} p \leftrightarrow v_p \rangle$

K-liveness

- Symbolic techniques for proving FG require a doubly-nested fixpoint
- SAT-based approaches typically reduce the problem to invariant checking
- K-Liveness: simple but effective technique for LTL verification of finite-state systems [ClaessenSörensson12]
- Key insight: $M \models FG\neg f_{\neg\phi}$ iff there exists k such that $f_{\neg\phi}$ is visited at most k times
  - Again, a safety property
- K-liveness: increase k incrementally
  - Liveness checking as a sequence of safety checks
- From finite to infinite
  - No change
  - Sound to prove properties, but not complete
  - Property may hold, but fairness can be visited an unbounded number of times
  - Extended in [Cimatti et al. 14] for hybrid systems

Other techniques

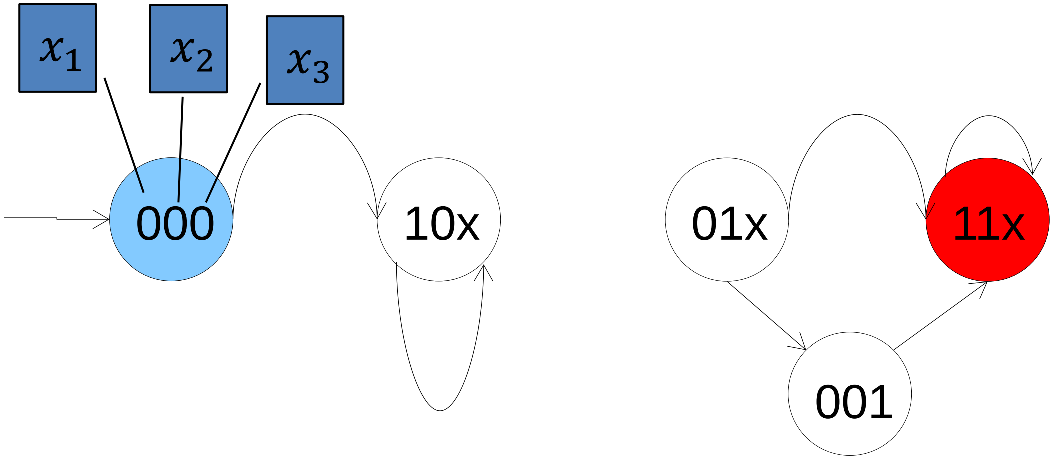
- BMC
  - Add encoding of lasso-shape and fairness
  - Sound for finding traces, but not complete
  - The only counterexample may be not lasso-shape
- Liveness2safety
  - Not sound (proving absence of lasso-shape is not sufficient)
  - Extended in [SchuppanBiere06] for specific classes of systems
  - Extended in [Daniel et al. 16] to consider only abstract lasso-shape paths and to integrate ranking functions like for termination

Induction

- P is an inductive invariant iff
  - Base case: check if the initial state satisfies P
  $$I(V) \vDash P$$
  - Inductive case: check if the transitions preserve the invariant
  $$P(V) \wedge T(V, V') \vDash P(V')$$

Example

- $V := \{x_1, x_{2,}, x_3\}$
- $I := \neg x_1 \wedge \neg x_2 \wedge \neg x_3$
- $Bad := x_1 \wedge x_2$
- $P := \neg x_1 \vee \neg x_2$
- Inductive?
- Inductive invariant?



Induction proof strategies

- Different strategies to prove P (see [MannaPnueli95])
- Stronger assertion: find S such that
  - S is inductive
  - $S \vDash P$
- Incremental strategy:
  - Use previously proved invariant S
  - Check if P is inductive relative to S

$$S \wedge P \wedge T \vDash P'$$

- Note that the set of reachable states is the strongest inductive invariant
  - Needs quantifiers
  - SMT with quantifiers is in many theories very complex or undecidable

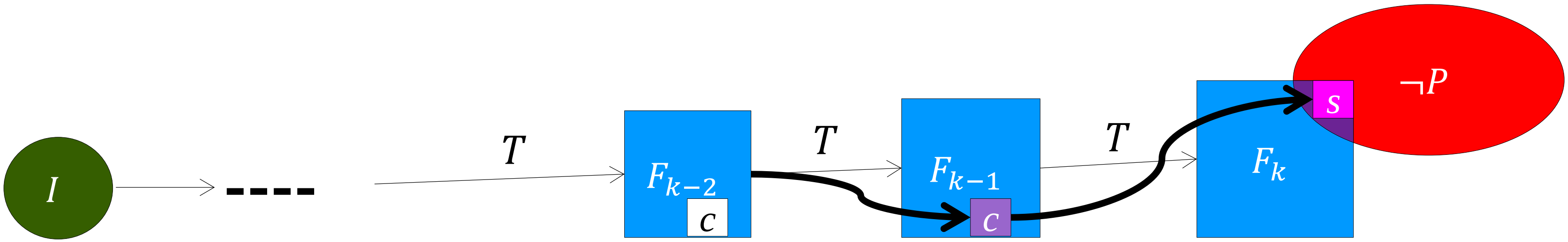
Incremental induction example

- Example:
  - $I \coloneqq x = 1 \land y = 1$
  - $T \coloneqq x' = x + 1 \land y' = y + x$
  - $P \coloneqq y \geq 1$
- Is P inductive?
- P is inductive relative to …

IC3

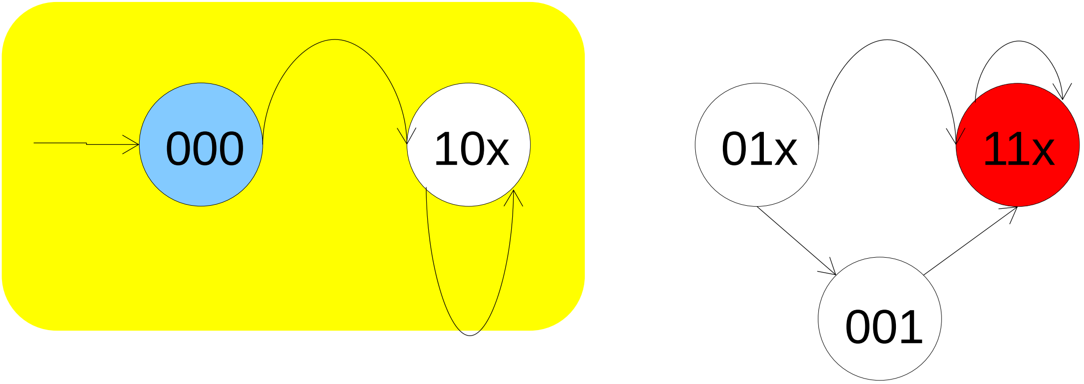
- Very successful SAT-based model checking algorithm proposed by Bradley in 2011
- Inductive invariant built incrementally
  - Trace of formulas $F_0 \equiv I, F_1, \dots, F_k$ s.t:
    - for $i > 0$, F_i is a set of clauses, overapproximation of states reachable in up to i steps
    - $F_{i+1} \subseteq F_i$ (so $F_i \vDash F_{i+1}$)
    - $F_i \wedge T \vDash F'_{i+1}$
    - For all $\mathrm{i} < \mathrm{k}, F_i \vDash P$
- Strengthen formulas until $F_i = F_{\mathrm{i}+1}$ for some i
- Exploiting efficient SAT solvers

IC3 key points



- Blocking phase:
  - If $F_{i-1} \wedge \neg c \wedge T \vDash \neg c'$ and $I \vDash \neg c$, add generalize c to g and block by adding $\neg g$ to $F_i, F_{i-1}, \dots, F_1$
- Propagation phase:
  - If $c \in F_i$ and $F_i \wedge T \vDash c'$, add c to F_{i+1}

Example



$$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$$
$$P = \neg x_1 \vee \neg x_2$$

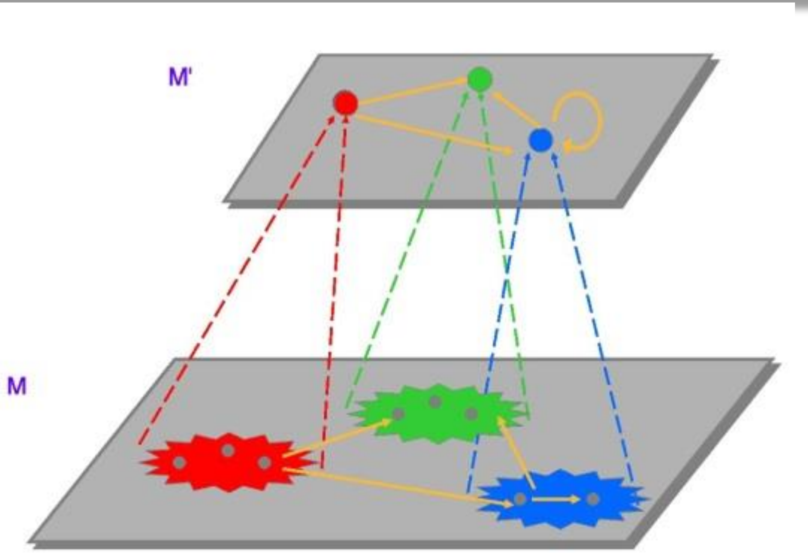
$$F_0 = I$$
$$F_1 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_2 = \neg x_2 \wedge (x_1 \vee \neg x_3)$$
$$F_3 = \top$$

Inductive invariant:

$$F_1 \equiv F_2 \equiv \neg x_2 \wedge (x_1 \vee \neg x_3)$$

Predicate Abstraction

- Reduction to finite-state MC [GrafSaidi97]
- Predicates $\mathbb{P}$ over concrete variables to define the abstraction
- Abstract state space given by Boolean variables, one for each predicate $\widehat{\mathrm{V}} = \{v_p \mid p \in \mathbb{P}\}$
- Abstract state $\alpha(s) = \{v_p \mid s(p) = \top\}$



- Abstract transition iff there exists a concrete transition between two corresponding concrete states

$$\hat{T} = \{\langle \hat{s}, \hat{s}' \rangle | \exists s, s', \alpha(s) = \hat{s}, \alpha(s') = \hat{s}', T(s, s')\}$$

- Spurious paths can be removed by adding more predicates (automatically extracted by the path) – see CEGAR in [Clarke et al.00]

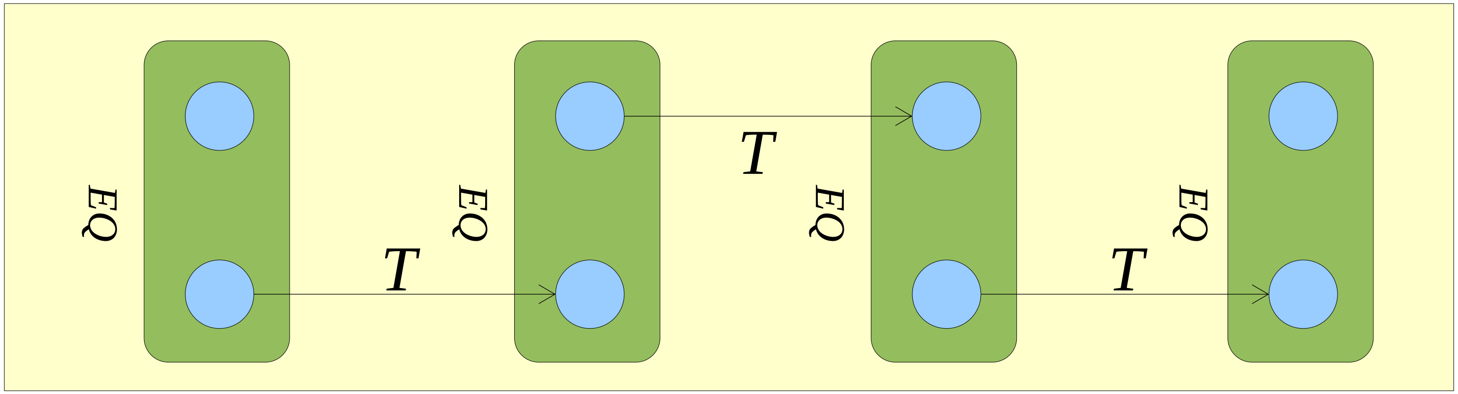
Implicit Predicate Abstraction

- Abstract version of BMC and k-induction, avoiding explicit computation of the abstract transition relation [Tonetta09]
  - By embedding the abstraction in the SMT encoding
  - $EQ(V_1, V_2) \coloneqq \bigwedge_{p \in \mathbb{P}} p(V_1) \leftrightarrow p(V_2)$
- For example, given the encoding a sequence of concrete transitions

$$T(V_0, V_1) \wedge T(V_1, V_2) \wedge \cdots \wedge T(V_{k-1}, V_k)$$

- The abstract version is

$$T\left(V_0, \overline{V}_1\right) \wedge EQ\left(\overline{V}_1, V_1\right) \wedge T\left(V_1, \overline{V}_2\right) \wedge EQ\left(\overline{V}_2, V_2\right) \wedge T(V_2, V_3) \wedge \cdots$$



IC3 with Implicit Abstraction

- Integrate the idea of Implicit Abstraction within IC3 [Cimatti et al. 14]
- P is inductive relative to S iff the following is unsat

$$S(V) \land P(V) \land T(V, V') \land \neg P(V')$$

- P is inductive relative to S in the abstraction with predicates $\mathbb{P}$ iff the following is unsat

$$S(V) \land P(V) \land T\left(V, \overline{V}\right) \land \bigwedge_{p \in \mathbb{P}} \left(p(V') \leftrightarrow p\left(\overline{V}\right)\right) \land \neg P(V')$$

- No theory-specific technique needed
- No preimage needed
- Same generalization as in the finite-state case

Abstraction refinement

- Abstract counterexample check can use incremental SMT
- Abstraction refinement is *fully incremental*
- No restart from scratch
- Can keep all the clauses of $F_1, \ldots, F_k$
- Refinements monotonically strengthen T

$$T_{new} := T_{old} \wedge \bigwedge_{p \in new\mathbb{P}} \big(p(V) \leftrightarrow p(W)\big) \wedge \big(p(V') \leftrightarrow p(W')\big)$$

- All IC3 invariants on $F_1, \ldots, F_k$ are preserved
  - $F_{i+1} \subseteq F_i$ (so $F_i \vDash F_{i+1}$)
  - $F_i \wedge T \vDash F'_{i+1}$
  - For all $\mathrm{i} < \mathrm{k}, F_i \vDash P$

Example

- System with 2 state vars c and d
  - Init: $(d = 1) \wedge (c \geq d)$
  - Trans: $(c' = c + d) \wedge (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Predicates $\mathbb{P}$
  - $(d = 1), (c \geq d),$
  - $(d > 2), (c > d),$
  - $(d \geq 2), (d \geq 3)$
- Final trace
  - $F_0 \coloneqq Init$
  - $F_1 \coloneqq \neg(d > 2) \wedge (c \geq d) \wedge F_2$
  - $F_2 \coloneqq F_3 \coloneqq (c \geq d) \vee \neg(d \geq 2) \wedge (d = 1) \vee (d \geq 2) \wedge F_4$
  - $F_4 \coloneqq (c > d) \vee \neg(d > 2)$

Summary

Contract-based reasoning

(Extended) Metric/Hybrid LTL SMT over (super)dense time

(Extended) LTL SMT over discrete time

LTL SMT-based Model Checking

Invariant SMT-based Model Checking (IC3 with Implicit Abstraction)

Conclusions

- Overview of extensions of LTL with SMT-based support
- Motivated by component-based design of embedded systems
- Rich language that includes:
  - First-order terms to represent the state
  - Discrete, dense, or super-dense models of time
  - Event-freezing functions which can express metric time, event clocks, and counting constraints
- Supported by complex layered tool chain
- Effective in practice for many problems

Future Directions

- From satisfiability to realizability
- Parameter synthesis (see work on tightening [SEFM16])
- Integrating security properties such as information flow
- Integrating probabilities
- Checking diagnosability and synthesis of monitors
- Specification mining from traces

- Contact me for next open positions for post-doc!

References

- Tools
  - NuSMV http://nusmv.fbk.eu
  - nuXmv https://nuxmv.fbk.eu
  - HyCOMP https://hycomp.fbk.eu
  - xSAP https://xsap.fbk.eu
  - OCRA https://ocra.fbk.eu
  - COMPASS http://www.compass-toolset.org/
- WBS case study
  - Marco Bozzano, Alessandro Cimatti, Anthony Fernandes Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson, Stefano Tonetta: Formal Design and Safety Analysis of AIR6110 Wheel Brake System. CAV (1) 2015: 518-535

References

- Linear-time Temporal Logic

  - Amir Pnueli: The Temporal Logic of Programs. FOCS 1977: 46-57
  - Orna Lichtenstein, Amir Pnueli, Lenore D. Zuck: The Glory of the Past. Logic of Programs 1985: 196-218
  - Zohar Manna, Amir Pnueli: The temporal logic of reactive and concurrent systems - specification. Springer 1992, ISBN 978-3-540-97664-6, pp. I-XIV, 1-427
  - Ron Koymans: Specifying Real-Time Properties with Metric Temporal Logic. Real-Time Systems 2(4): 255-299 (1990)
  - Rajeev Alur, Thomas A. Henzinger: Logics and Models of Real Time: A Survey. REX Workshop 1991: 74-106
  - Rajeev Alur, Thomas A. Henzinger: A Really Temporal Logic. J. ACM 41(1): 181-204 (1994)
  - Rajeev Alur, Tomás Feder, Thomas A. Henzinger: The Benefits of Relaxing Punctuality. J. ACM 43(1): 116-146 (1996)
  - Jean-François Raskin, Pierre-Yves Schobbens: The Logic of Event Clocks - Decidability, Complexity and Expressiveness. Journal of Automata, Languages and Combinatorics 4(3): 247-286 (1999)
  - Alexander Moshe Rabinovich: On the Decidability of Continuous Time Specification Formalisms. J. Log. Comput. 8(5): 669-678 (1998)
  - Yoram Hirshfeld, Alexander Moshe Rabinovich: An Expressive Temporal Logic for Real Time. MFCS 2006: 492-504
  - Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, Daniele Zucchelli: Combination Methods for Satisfiability and Model-Checking of Infinite-State Systems. CADE 2007: 362-378
  - Carlo A. Furia, Matteo Rossi: On the Expressiveness of MTL Variants over Dense Time. FORMATS 2007: 163-178
  - Stéphane Demri, Ranko Lazic: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. 10(3): 16:1-16:30 (2009)
  - Alessandro Cimatti, Marco Roveri, Stefano Tonetta: HRELTL: A temporal logic for hybrid systems. Inf. Comput. 245: 54-71 (2015)
  - Stefano Tonetta: Linear-time Temporal Logic with Event Freezing Functions. GandALF 2017: 195-209

References

- Contract-based design for embedded systems/CPS

  - Albert Benveniste, Benoît Caillaud, Roberto Passerone: A Generic Model of Contracts for Embedded Systems. CoRR abs/0706.1456 (2007)
  - Albert Benveniste, Benoît Caillaud, Alberto Ferrari, Leonardo Mangeruca, Roberto Passerone, Christos Sofronis: Multiple Viewpoint Contract-Based Specification and Design. FMCO 2007: 200-225
  - Werner Damm, Hardi Hungar, Bernhard Josko, Thomas Peikenkamp, Ingo Stierand: Using contract-based component specifications for virtual integration testing and architecture design. DATE 2011: 1023-1028
  - Sebastian S. Bauer, Alexandre David, Rolf Hennicker, Kim Guldstrand Larsen, Axel Legay, Ulrik Nyman, Andrzej Wasowski: Moving from Specifications to Contracts in Component-Based Design. FASE 2012: 43-58
  - Sebastian S. Bauer, Rolf Hennicker, Axel Legay: A meta-theory for component interfaces with contracts on ports. Sci. Comput. Program. 91: 70-89 (2014)
  - Alberto L. Sangiovanni-Vincentelli, Werner Damm, Roberto Passerone: Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems. Eur. J. Control 18(3): 217-238 (2012)
  - Antonio Iannopollo, Pierluigi Nuzzo, Stavros Tripakis, Alberto L. Sangiovanni-Vincentelli: Library-based scalable refinement checking for contract-based design. DATE 2014: 1-6
  - Pierluigi Nuzzo, Alberto L. Sangiovanni-Vincentelli, Davide Bresolin, Luca Geretti, Tiziano Villa: A Platform-Based Design Methodology With Contracts and Related Tools for the Design of Cyber-Physical Systems. Proceedings of the IEEE 103(11): 2104-2132 (2015)
  - Alessandro Cimatti, Stefano Tonetta: A Property-Based Proof System for Contract-Based Design. EUROMICRO-SEAA 2012: 21-28
  - Alessandro Cimatti, Michele Dorigatti, Stefano Tonetta: OCRA: A tool for checking the refinement of temporal contracts. ASE 2013: 702-705
  - Marco Bozzano, Alessandro Cimatti, Cristian Mattarei, Stefano Tonetta: Formal Safety Assessment via Contract-Based Design. ATVA 2014: 81-97
  - Alessandro Cimatti, Stefano Tonetta: Contracts-refinement proof system for component-based embedded systems. Sci. Comput. Program. 97: 333-348 (2015)
  - Alessandro Cimatti, Rance DeLong, Davide Marcantonio, Stefano Tonetta: Combining MILS with Contract-Based Design for Safety and Security Requirements. SAFECOMP Workshops 2015: 264-276
  - Alessandro Cimatti, Ramiro Demasi, Stefano Tonetta: Tightening a Contract Refinement. SEFM 2016: 386-402

References

- IC3 for Finite-State Transition Systems:
  - Aaron R. Bradley: SAT-Based Model Checking without Unrolling. VMCAI 2011: 70-87
  - Fabio Somenzi, Aaron R. Bradley: IC3: where monolithic and incremental meet. FMCAD 2011: 3-8
  - Krystof Hoder, Nikolaj Bjørner: Generalized Property Directed Reachability. SAT 2012: 157-171
- IC3 for Infinite-State Systems:
  - Alessandro Cimatti, Alberto Griggio: Software Model Checking via IC3. CAV 2012: 277-293
  - Alessandro Cimatti, Alberto Griggio, Sergio Mover, Stefano Tonetta: IC3 Modulo Theories via Implicit Predicate Abstraction. TACAS 2014: 46-61
  - Johannes Birgmeier, Aaron R. Bradley, Georg Weissenbacher: Counterexample to Induction-Guided Abstraction-Refinement (CTIGAR). CAV 2014: 831-848
  - Yakir Vizel, Arie Gurfinkel: Interpolating Property Directed Reachability. CAV 2014: 260-276
  - Nikolaj Bjørner, Arie Gurfinkel: Property Directed Polyhedral Abstraction. VMCAI 2015: 263-281
- Predicate abstraction
  - Susanne Graf, Hassen Saïdi: Construction of Abstract State Graphs with PVS. CAV 1997: 72-83
  - E. Clarke, O. Grumberg, S. Jha, Y. Lu, and V.H., “Counterexample-guided abstraction refinement,” in Computer Aided Verification, 2000, pp. 154–169.
  - Stefano Tonetta: Abstract Model Checking without Computing the Abstraction. FM 2009: 89-105
- LTL Model Checking:
  - Amir Pnueli: The Temporal Logic of Programs. FOCS 1977: 46-57
  - Moshe Y. Vardi: An Automata-Theoretic Approach to Linear Temporal Logic. Banff Higher Order Workshop 1995: 238-266
  - Edmund M. Clarke, Orna Grumberg, Kiyoharu Hamaguchi: Another Look at LTL Model Checking. Formal Methods in System Design 10(1): 47-71 (1997)
- Liveness to safety:
  - Viktor Schuppan, Armin Biere: Efficient reduction of finite state model checking to reachability analysis. STTT 5(2-3): 185-204 (2004)
  - Koen Claessen, Niklas Sörensson: A liveness checking algorithm that counts. FMCAD 2012: 52-59

References

- Liveness to Safety for Infinite-State Systems:
  - Viktor Schuppan, Armin Biere: Liveness Checking as Safety Checking for Infinite State Spaces. Electr. Notes Theor. Comput. Sci. 149(1): 79-96 (2006)
  - Andreas Podelski, Andrey Rybalchenko: Transition predicate abstraction and fair termination. ACM Trans. Program. Lang. Syst. 29(3) (2007)
  - Alessandro Cimatti, Alberto Griggio, Sergio Mover, Stefano Tonetta: Verifying LTL Properties of Hybrid Systems with K-Liveness. CAV 2014: 424-440
  - Jakub Daniel, Alessandro Cimatti, Alberto Griggio, Stefano Tonetta, Sergio Mover: Infinite-State Liveness-to-Safety via Implicit Abstraction and Well-Founded Relations. CAV (1) 2016: 271-291
- Hybrid systems
  - Thomas A. Henzinger: The Theory of Hybrid Automata. LICS 1996: 278-292
  - Rajeev Alur: Formal verification of hybrid systems. EMSOFT 2011: 273-278
- SMT-Based Verification of Hybrid Systems
  - Gilles Audemard, Marco Bozzano, Alessandro Cimatti, Roberto Sebastiani: Verifying Industrial Hybrid Systems with MathSAT. Electr. Notes Theor. Comput. 119(2): 17-32 (2005)
  - Andreas Eggers, Martin Fränzle, Christian Herde: SAT Modulo ODE: A Direct SAT Approach to Hybrid Systems. ATVA 2008: 171-185
  - Alessandro Cimatti, Sergio Mover, Stefano Tonetta: SMT-based scenario verification for hybrid systems. Formal Methods in System Design 42(1): 46-66 (2013)