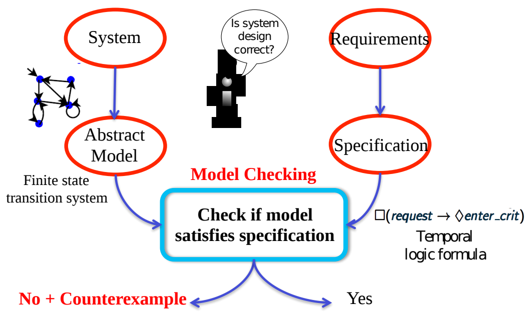
Embedded Systems



0.1. About

Specialized functionality
Constraints on power consumption, real-time scheduling, space, costs
⇒ No single optimization problem!
Different parts and interfaces: communication.
Hardware-Software partitioning

0.2. Modeling and Verification

System ↔ Model
Create an abstract Model with properties to verify that the system owns a certain property

1. Modeling

1.1. Transition Systems TS

Can be used to model ANY system!
Model $M = \{S, IRL\}$
Set of States $S = \{s_0, s_1, s_2, \dots\}$
Initial States $I \subseteq S$
Set of Actions $Act = \{\alpha, \beta, \gamma, \dots\}$
Transition Relations $R \subseteq S \times Act \times S$
Atomic Propositions $AP = \{a, b, c\}$
Label $L : S \to 2^{AP}$

TS is finite if S and Act are finite sets, otherwise its infinite.
$2^{AP} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$

A path π is an infinite sequence of states $\pi = s_0 s_1 s_2 \dots$

1.2. Control System

A control System Σ is a tuple $\Sigma = (\mathbb{R}^n, U, \mathcal{U}, f)$
state space $\mathbb{R}^n$, input set $U \subseteq R^n$, functions from $\mathbb{R}^+_0$ to U, function $f : \mathbb{R}^n \times U \to \mathbb{R}^n$
Given Σ and sampling time τ: Transform to Transition State: $X = \mathbb{R}^n$
$x \to x'$ iff $\xi_{xv}(\tau) = x'$

1.3. Timed Automata

$TA = (Z, \Sigma, C, \delta, F, Z_0)$
finite set of clocks C
state transition function $\delta : Z \times \Sigma \times P \to Z \times 2^C$
timing conditions P
initial time τ_0 and transition time τ_i

1.4. Hybrid Automata

Clock is not monoton but can be dynamic.

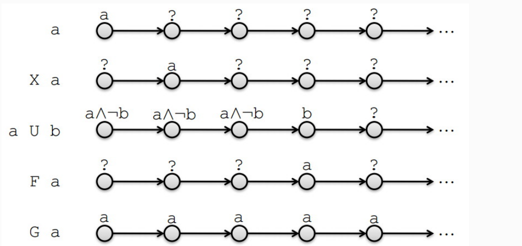
2. Verification

2.1. Properties

State formulae: properties or formulae which are tru in a specific state
Path formulae: related to paths

2.2. Temporal Logic

$p\mathcal{U}q$	U	p true until q is true
$p\mathcal{R}q$	R	q true; released after p was true
$\bigcirc p$	X	Next state has p
$\square p$	G	Globally: p is true for the entire subsequent path
$\lozenge p$	F	Future: p is true somewhere in the subsequent path
$\forall p$	A	true on all paths starting from the current path
$\exists p$	E	true on at least one subsequent path



Duality: $\neg X\varphi = X\neg\varphi, \quad \neg F\varphi = G\neg\varphi, \quad \neg G\varphi = F\neg\varphi$
Absorbtion: $FGF\varphi = GF\varphi, \qquad GFG\varphi = FG\varphi$

$p, g, f \in AP$
$s \models p$: s satisfies p

$s \models p \Leftrightarrow p \in L(s)$
$s \models f \wedge g \Leftrightarrow s \models f \wedge s \models g$
$s \models f \wedge g \Leftrightarrow s \models f \wedge s \models g$
$s \models \exists \bigcirc f \Leftrightarrow \exists\pi = s_0 s_1 \dots : s_1 \models f$
$s \models \exists(f\mathcal{U}g) \Leftrightarrow \exists\pi = s_0 s_1 \dots, \exists j \geq 0$
$[s_j \models g \wedge \forall i : 0 \leq i < j[s_i \models f]]$
$s \models \exists\square f \Leftrightarrow \exists\pi = s_0 s_1 \dots, \forall i > 0 : s_i \models f$

2.3. Logic Formula

$\varphi ::= \text{true}|a|\varphi_1 \wedge \varphi_2|\neg\varphi| \bigcirc \varphi|\varphi_1\mathcal{U}\varphi_2$
Implications:
$\lozenge\varphi := \text{true}\mathcal{U}\varphi \qquad \square\varphi := \neg\lozenge\neg\varphi$

Examples:
Mutal Exclusion: $\square(\neg\text{crit}_1 \vee \neg\text{crit}_2)$
railroad-crossing: $\square(\neg\text{train\_near} \to \neg\text{gate\_closed})$
progress: $\square(\text{Request} \to \lozenge\text{Response})$

3. Model Checking

3.1. Basic Idea

Given: a finite transition system $\mathcal{T}$ over a set of atomic propositions AP and an LTL formula φ over AP.
Model checking question: Does $\mathcal{T} \models \varphi$ hold?
Try to refute $\mathcal{T} \models \varphi$ by searching for a path π in $\mathcal{T}$ such that $\pi \not\models \varphi$ or $\pi \models \neg\varphi$

3.2.

Check: $f = \exists\lozenge g$
S_g set of states that satisfy g ind M
Start with $Q' = S_g$
Iteratively add all states that in the next state satisfy g
Result: S_f, if $S_f \cap I \neq \emptyset$, then f is satisfied by M

$\text{Pre}(Q)$: set of states that have a transition to a state in Q
$\text{Post}(Q)$: set of states that can be reached by a transition from a state in Q
Path π: infinite sequence of valid states.
$\text{Trace}(\pi)$: sequence of atomic propositions of the path π

3.3.

$\text{Trace}(s) = \text{Trace}(\pi(\text{Post}(s))$

3.4. Linear time (LT) properties

Linear time (LT) properties specify traces that a transition system must exhibit. LT properties describe admissible behaviors of the system under consideration. A LT property is a subset of $\{2^A P\}^\omega$

3.5. Invariants

An invariant is an LT property that can be expressed by a logic condition Φ for states.
The invariant dictates that Φ is true for all reachable states
Example: (Mutual Exclusion): $\Phi = \neg c_1 \vee \neg c_2$

3.6. Safety Properties

"Nothing bad should happen"
Any infinite run violating the property will have a finite prefix that is "bad"
State Property: Look at the bad states.
$P_{\text{inv}} = A_0 A_1 A_2 \ldots \in (2^{AP})^\omega | \forall j \geq 0, A_j \models \Phi$
Path property: Look at the bad paths. Find a finite prexif of a path that will violate the safety property.
Example ATM: money can only be drawn after correct PIN.

A LT Property P_{safe} over AP is a safety Property if, $\forall\sigma \in (2^{AP})^\omega \setminus P_{\text{safe}}, \exists\hat{\sigma}_{\text{finit}} \subset \sigma$:

$TS \models P_{\text{safe}}$ iff $\text{Trace}(TS) \cap BadPref(P_{\text{safe}}) = \emptyset$ P is a safety Property iff $\text{Closure}(P) = P$
Because there is no

3.7. Closure

The closure of a LT property P is the set of infinite traces whose finite prefixes are also prefixes of P
This means: All loops and(union) their prefixes that are reached by P.
P is a safety Property iff $\text{Closure}(P) = P$
This means: all loops can be reached without finite prefixes (but may ALSO be reached with finite prefixes)

3.8. Liveness

Liveness properties ensure that something good will eventually happen.
Liveness property can only be violated in infinite time.
P is a liveness property if $\text{Closure}(P) = (2^{AP})^\omega$
Because no prefix is ruled out by P
Example: The program eventually terminates
Or: If you request an elevator it will eventually come.
A process will eventually enter its critical section
A process will enter its critical section infinitely often
Every waiting process will eventually enter its critical section

Any LT property is equivalent to a conjunction of a safety and a liveness property

3.9. Languages

A Language L consists of strings of symbols from an alphabet. An alphabet $\Sigma = \{a, b, c\}$
Strings: $\varepsilon, aab, baabc$
ε is the empty string. $\Sigma*$ is the set of all finite strings over Σ
Language $L = \{\varepsilon, a, b, aa, ab\}$

3.10. Regular Expressions

$\alpha ::= \emptyset|\varepsilon|A|\alpha_1 + \alpha_2|\alpha_1.\alpha_2|\alpha* \qquad \alpha \mapsto \mathcal{L}(\alpha) \subseteq \Sigma^*$

Empty set $\emptyset$ is not the same as empty string ε
Basic Ops: Union $(+)$, Concatenation $(.)$, Finite Repition $(*)$
$\alpha^+ = \alpha\alpha* = \alpha * \alpha$ (without empty string, means at least one α)

3.10.1 ω-Regular Expression

$*$: finite repetition, $^\omega$: infinite repitition
General ω-regular expression: $\gamma = \alpha_1\beta_1^\omega + \ldots + \alpha_n\beta_n^\omega$
An ω-regular Language $\mathcal{L}_\omega(\gamma) = \bigcup_{1 \leq i \leq n} \mathcal{L}(\alpha_i)\mathcal{L}(\beta_i)^\omega \subseteq \Sigma^\omega$ is the set of infinite words over 2^{AP} that have an accepting run in γ

3.10.2 ω-regular Property $E \subseteq (2^{\text{AP}})^\omega$

"E is called an ω-regular property iff there exists an ω-regular expression γ over 2^{AP} such that $E = \mathcal{L}_\omega(\gamma)$"
$E = (a * b*)^\omega$ is no reg. expr. because it contains ε^ω

3.11. Nondeterministic Büchi Automata (NBA)

Can recognize ω-regular languages
NFA: A word is accepted if a final state is reached by an infinite path.
Nondeterministic Büchi Automata: A word is accepted if a final state is reached infinitely often by an infinite path.

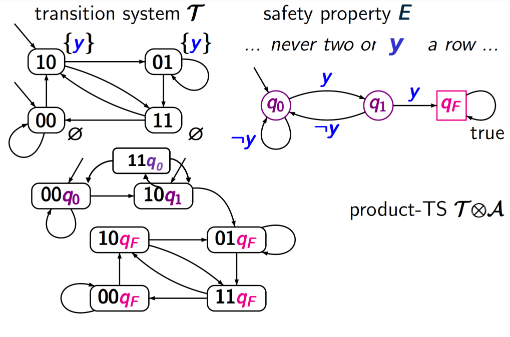
"For each NBA $\mathcal{A}$ there is an ω-regular expression γ with $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\gamma)$"

3.12.

Let E

3.13. Checking ω-regular Properties

1. construct a NBA $\mathcal{A}$ for the bad behavior for the LT property E
2. build the product TS $\mathcal{T} \otimes \mathcal{A}$ check



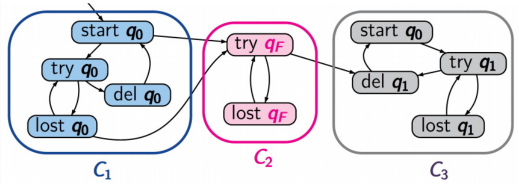
3.14. Persistence Checking

Given finite transition $\mathcal{T}$ over AP and persistence condition a

Question: Does $\mathcal{T} \models$ "eventtually forever a" hold?
Equivalent:

3.14.1 Strongly Connected Component (SCC)

Maximal set of states that are reachable from each other (loops):



C_2 is non trivial because it has more than one edge.

3.15. ReCap

3.15.1 Modeling

TS to model any system. Properties: reachable, terminal. Paths Typical: draw a TS of a logic. differential equation of el. or mech. system

Find invariant condition Φ
Each invariant is a safety property but not vis versa

3.15.2 Language and Automata Theory

Clenees? Theory: Every reg. expr. can be produces by automata.
Typ: Identify/construct/dterminise NFA
Linear temporal Logic

3.15.3 Model checking

Really important! Compute product

3.15.4 Abstraction and Synthesis

Model checking: Does TS satisfy φ?
Synthesis: Is there a controller C such that $TS \times C \models \varphi$?

Abstraction: $TS \leq TS'$ such that $traces(TS) \leq traces(TS')$

Tutorials

Tut1, Ex1: AP $= \{a, b\}$, $L = \{L(s_0) = \{a\}, L(s_1) = \{b\}\}$
Tut1, Ex2: b) α in left and right TS are not the same. c) same name, same signals! chack all transitions for every state

Tut2, Ex1: a) $S = \mathbb{R}^2$
Tut2, Ex1: c) deterministic if $|\operatorname{Post}(s, \alpha)| = 1$
Tut2, Ex1: d) Every S has a post state
Tut2, Ex1: e) $P_1 = \{s \in S \mid \|s\| \leq 0.1\}$

Tut3, Ex2: $s \models p \Leftrightarrow p \in L(s)$
Tut4, Ex1: $a(a + \varepsilon)(a, ba)^\omega$ Tut4, Ex2: a) b^ω liveness, b) $b*(a + a, b)b*$ safety c) $(ab)^\omega$ liveness d) none Tut4, Ex3: a) ? b) $0(01)^\omega$ c) $1(1 + 0)^\omega$ d)
Tut5, Ex1: a) Tut5, Ex2:
Tut6, Ex1: a) s_2, s_3 b) s_1, s_2, s_4 c) s_1, s_4 d) s_2, s_3, s_4 e) s_1 f) all states g) $\emptyset$ h) $\emptyset$ i) s_1, s_2, s_3 j) $\emptyset$ Tut6, Ex2: