Temporal Logics

Stefan Ratschan

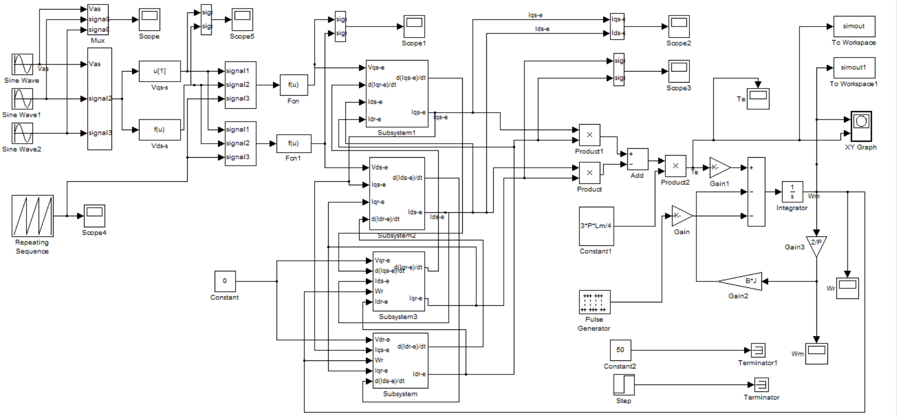
Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague

Evropský sociální fond Praha & EU: Investujeme do vaší budoucnosti

Introduction

What do we already know?

- Black-box description of systems and their components: *(discrete-time) system*
- White-box description of systems and their components: *(discrete-time) automaton*
- Interaction (*parallel composition*, *cascade composition*, etc.)



Even the composition of very simple systems/automata can show highly complex behavior

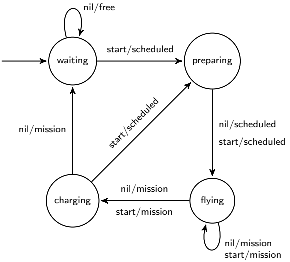
How can we be sure that the model behaves as wished?

Specification Language: Systems

$$\{(i, o) \in \Sigma_{\{nil, start\}} \times \Sigma_{\{free, scheduled, mission\}} \mid$$
$$\text{input } \textit{start} \text{ results in output } \textit{mission} \text{ within at most 10 steps}\}$$

$$\{(i, o) \in \Sigma_{\{nil, start\}} \times \Sigma_{\{free, scheduled, mission\}} \mid$$
$$\forall t \in \mathbb{N}_0 \,.\, i(t) = start \Rightarrow \exists d \in \{0, \dots, 10\} \,.\, o(t+d) = mission\}$$

Implementation: automaton



Implementation given by automaton A
fulfills specification given by system S, that is

$$[\![A]\!] \subseteq S?$$

Specification Language and Automatization

Implementation given by automaton A
fulfills specification given by system S, that is

$$[\![A]\!] \subseteq S?$$

We want to check this, if possible automatically.

For a machine to understand, we need simple and precise specification

What about the specification

$$\{(i, o) \mid o(0) = 1 \text{ iff } \mathbf{P} = \mathbf{NP}\}???$$

We need a simpler specification language!

Specification of System Behavior

Example:

- “reactor is not going to overheat”
- “if the elevator is called, it will show up eventually”
- “tomorrow the weather will be nice”
- “central locking of a car opens immediately after a crash”
- “airbag only inflates if a car crash happens”
- “server acknowledge has to be preceded by a request”

All of this can be formalized in predicate logic

Too general: difficult for both people and computers

Let’s analyze examples!

Properties (already present in classical I/O specification)

Temporal specification (this lecture)

Automata: Simplification

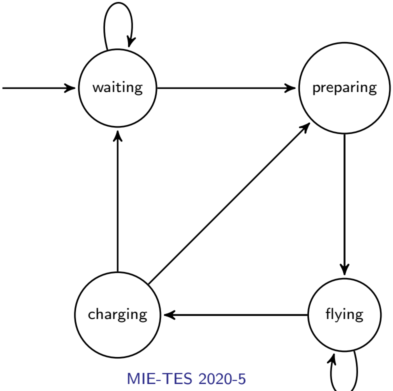
I/O structure irrelevant

So: as simple as possible automata model: *transition system*:

- set of states (*state space*) S
- non-empty set $S_0 \subseteq S$ of initial states
- transition relation $R \subseteq S \times S$ s.t.

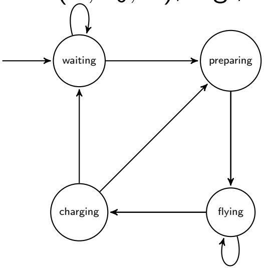
  for all $s \in S$ there is $s' \in S$ s.t. $(s, s') \in R$.

No i/o



Behavior of Transition Systems

We assume a transition system (S, S_0, R), e.g.,



Path: $(s_0, s_1, \dots) \in \Sigma_S$, s.t.

$s_0 \in S_0$, $(s_0, s_1) \in R$, $(s_1, s_2) \in R$, $\dots$

Notation:

$$s_0 \to s_1 \to \dots$$

One transition system may have many paths!

S can even be an infinite set (e.g., $\mathbb{N}$, $\mathbb{R}$, lists of integers).

Specification of System Behavior

- "tomorrow the weather will be nice"
- "reactor is not going to overheat"
- "if the elevator is called, it will show up eventually"
- "central locking of a car opens immediately after a crash"
- "airbag only inflates if a car crash happens"
- "Server acknowledge has to be preceded by a request"

Properties of individual states

Temporal specification

Further plan: specifying properties of

1. states (i.e., no time)
2. paths (i.e. temporal specification)
3. transition systems (i.e., several paths)

Specifying Properties of States

For now, we ignore time. How to specify properties on state?

Examples:

- $S = \{\text{waiting}, \text{preparing}, \text{flying}, \text{charging}\}$,
  onground: $\{\text{waiting}, \text{preparing}, \text{charging}\}$ → Property onground
- $S = \mathbb{R}$ (temperature), overheat: $\{T \in S \mid T > 800\}$
- $S = \{\text{rain}, \text{sunshine}\} \times \mathbb{R}$,
  nice: $\{(w, T) \in S \mid w = \mathit{sunshine} \wedge T \geq 18 \wedge T \leq 26\}$

State property: name that denotes a subset of the state space S

We assume a function $\mathcal{I}$ (*interpretation*), that
assigns to each state property a set of states.

Example: $\mathcal{I}(\text{overheat}) = \{T \in S \mid T > 800\}$

State property p *holds* on a state s ($s \models p$), iff $s \in \mathcal{I}(p)$.

Example $922 \models \text{overheat}$ iff $922 \in \{T \in S \mid T > 800\}$

Specifying Temporal Properties of Systems

- "tomorrow the weather will be nice"
- "reactor is not going to overheat"
- "if the elevator is called, it will show up eventually"
- "central locking of a car opens immediately after a crash"
- "airbag only inflates if a car crash happens"
- "Server acknowledge has to be preceded by a request"

Properties of individual states

Temporal specification

For a path π of the form $(s_0, s_1, \dots)$, we denote by

- π^i (the i-th *suffix* of π), the path $(s_i, s_{i+1}, \dots)$, and by
- $\pi(i)$ the element s_i.

Properties on Paths

Let us add time: How to specify properties of whole paths?

First: state property, e.g.,

The weather is nice

diffent!

State property p holds on a path π ($\pi \models p$) iff
holds on first element of path: $\pi(0) \models p$.

Attention: $\models$ vs. $\models$!

$(\mathit{flying}, \mathit{charging}, \mathit{waiting}, \mathit{waiting}, \mathit{waiting}, \dots) \models \mathit{onground}$
$(\mathit{flying}, \mathit{charging}, \mathit{waiting}, \mathit{waiting}, \mathit{waiting}, \dots)(0) \models \mathit{onground}$
$\mathit{flying} \models \mathit{onground}$
$\mathit{flying} \in \mathcal{I}(\mathit{onground})$
$\mathit{flying} \in \{\mathit{waiting}, \mathit{preparing}, \mathit{charging}\}$ does not hold

Properties on Paths

Tomorrow the weather will be nice

Property holds on next element of path: $\pi^1 \models p$ (vs. $\pi(1) \models p$)
($\pi \models \mathbf{X}p$: "next").

The train eventually reaches full speed

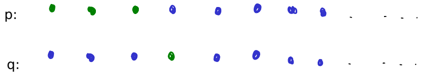
there is $k \geq 0$ s.t. $\pi^k \models p$
($\pi \models \mathbf{F}p$: "in the future")

The number of motor rotations always stays in safe area

for all $k \geq 0$, $\pi^k \models p$ ($\pi \models \mathbf{G}p$: "globally")

Properties on Paths

The train eventually stops and until then the doors remain closed

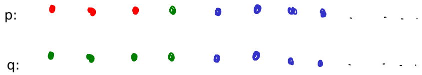


(the doors remain closed)

(the train stops)

there is i s.t. $\pi^i \models q$, and
for all $j < i$, $\pi^j \models p$ ($\pi \models p\mathbf{U}q$: "until")

As long as the plane does not reach full height
the fasten seat belts sign is on



(the plane reaches full height)

(the fast seat belts sign is on)

for all $j \geq 0$,
if for all $i < j$, $\pi^i \not\models p$
then $\pi^j \models q$ ($\pi \models p\mathbf{R}q$: "release")

Combining Operators

The train will never move with open doors

All the time $\rightarrow$ $\mathbf{G}\neg[p \wedge q]$

Whenever the elevator is called, it will eventually show up:

$$\mathbf{G}[p \Rightarrow \mathbf{F}q]$$

So: Boolean combinations ($\wedge$, $\vee$, $\neg$, $\Rightarrow$).

Combining temporal operators. For example:

- $\mathbf{FG}p$: Eventually property p will hold forever.
- $\mathbf{GF}p$: Always eventually p will hold.

  Whenever I come to the station,
  soon or later a train will come

Result: *Linear Temporal Logic (LTL)*

Further examples: `ltllib.lsal`

Syntax of LTL

- Every state property is an LTL formula
- If p and q are LTL formulas then also $\mathbf{X}p$, $p\mathbf{U}q$, $p\mathbf{R}q$, $\mathbf{F}p$, $\mathbf{G}p$, $p\mathbf{U}q$, $\neg p$, $p \vee q$, and $p \wedge q$, are LTL formulas.

Operator priority:

1. $\neg$, $\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$
2. $\mathbf{U}$, $\mathbf{R}$
3. $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$

$p\mathbf{U}q\mathbf{U}r$??? $[p\mathbf{U}q]\mathbf{U}r$ vs. $p\mathbf{U}[q\mathbf{U}r]$

U, **R**: right-associative in literature, but
left-associative in language PROMELA

Semantics of LTL

SUMMARY

For a path π and LTL formulas p, q,

- $\pi \models p$ where p is a state property
  iff $\pi(0) \models p$
- $\pi \models \mathbf{X}p$ iff $\pi^1 \models p$
- $\pi \models \mathbf{F}p$ iff there is k s.t. $\pi^k \models p$
- $\pi \models \mathbf{G}p$ iff for all k, $\pi^k \models p$
- $\pi \models p\mathbf{U}q$ iff there is i s.t. $\pi^i \models q$ and for all $j < i$, $\pi^j \models p$
- $\pi \models p\mathbf{R}q$ iff for all j, [if for all $i < j$, $\pi^i \not\models p$, then $\pi^j \models q$]
- $\pi \models \neg p$ iff not $\pi \models p$
- $\pi \models p \wedge q$ iff $\pi \models p$ and $\pi \models q$
- $\pi \models p \vee q$ iff $\pi \models p$ or $\pi \models q$

where i, j, k range over $\mathbb{N}_0$.

Attention: Recursive definition,
recursion ends on right-hand side of first line!

Example

Path π: $(a, b, c, d, a, b, c, d, \dots) \in \Sigma_{\{a,b,c,d\}}$

State property p with interpretation $\mathcal{I}(p) = \{c, d\}$

$\pi \models \mathbf{F} p$?

there is k s.t. $\pi^k \models p$

there is k s.t. $\pi^k(0) \models p$

there is k s.t. $\pi^k(0) \in \mathcal{I}(p)$

there is k s.t. $\pi^k(0) \in \{c, d\}$

there is k s.t. $(a, b, c, d, a, b, c, d, \dots)^k(0) \in \{c, d\}$

Example Continued

there is k s.t. $(a, b, c, d, a, b, c, d, \dots)^k(0) \in \{c, d\}$

First attempt $k = 0$:

$(a, b, c, d, a, b, c, d, \dots)^0(0) \in \{c, d\}$
$(a, b, c, d, a, b, c, d, \dots)(0) \in \{c, d\}$
$a \in \{c, d\}$
$\bot$

Second attempt $k = 1$:

$(a, b, c, d, a, b, c, d, \dots)^1(0) \in \{c, d\}$
$(b, c, d, a, b, c, d, \dots)(0) \models \{c, d\}$
$b \in \{c, d\}$
$\bot$

Third attempt $k = 2$:

$(a, b, c, d, a, b, c, d, \dots)^2(0) \in \{c, d\}$
$(c, d, a, b, c, d, \dots)(0) \in \{c, d\}$
$c \in \{c, d\}$
$\top$

Further Example

$\pi \models \mathbf{GF}p$?

from definition: $\pi \models \mathbf{G}p$ iff for all k, $\pi^k \models p$

for all k, $\pi^k \models \mathbf{F}p$

from definition: $\pi \models \mathbf{F}p$ iff there is k s.t. $\pi^k \models p$

for all l, $\pi^l \models \mathbf{F}p$

for all l,
  there is k s.t. $(\pi^l)^k \models p$

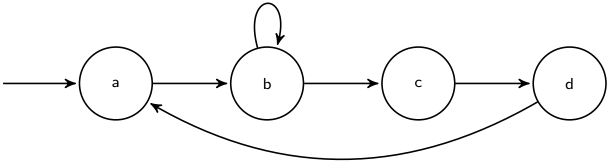
for all l,
  there is k s.t. $\pi^{l+k} \models p$

for all l,
  there is k s.t. $\pi(l+k) \models p$

for all l,
  there is k s.t. $\pi(l+k) \in \mathcal{I}(p)$

Properties of Transition Systems

1. states (i.e., no time)
2. paths (i.e. temporal specification)
3. transition systems (i.e., several paths)



$\mathbf{F}p$, where $\mathcal{I}(p) = \{d\}$?

$(S, S_0, R) \models p$ iff
 for all paths π of the given transition system (S, S_0, R), $\pi \models p$

Hence: $(S, S_0, R) \models \neg p$ is not equivalent to $(S, S_0, R) \not\models p$!

If the transition system is clear from the context one often writes $\models p$.

Some Equivalences

For an arbitrary LTL formula ϕ, we have:

- $\neg\mathbf{G}\phi$ is equivalent to $\mathbf{F}\neg\phi$
- $\neg\mathbf{F}\phi$ is equivalent to $\mathbf{G}\neg\phi$
- $\neg\mathbf{X}\phi$ is equivalent to $\mathbf{X}\neg\phi$
- $\mathbf{G}\phi$ is equivalent to $\phi \wedge \mathbf{XG}\phi$
- $\mathbf{F}\phi$ is equivalent to $\phi \vee \mathbf{XF}\phi$
- $\mathbf{GG}\phi$ is equivalent to $\mathbf{G}\phi$
- $\mathbf{FF}\phi$ is equivalent to $\mathbf{F}\phi$
- $\mathbf{FGF}\phi$ is equivalent to $\mathbf{GF}\phi$
- $\mathbf{GFG}\phi$ is equivalent to $\mathbf{FG}\phi$

Equivalences Continued

For arbitrary LTL formulas ϕ and ψ, we have:

- $\mathbf{F}[\phi \vee \psi]$ is equivalent to $\mathbf{F}\phi \vee \mathbf{F}\psi$
- $\mathbf{G}[\phi \wedge \psi]$ is equivalent to $\mathbf{G}\phi \wedge \mathbf{G}\psi$
- $\mathbf{F}\phi$ is equivalent to $\top\mathbf{U}\phi$
- $\mathbf{G}\phi$ is equivalent to $\bot\mathbf{R}\phi$
- $\neg[\phi\mathbf{U}\psi]$ is equivalent to $\neg\phi\mathbf{R}\neg\psi$
- $\neg[\phi\mathbf{R}\psi]$ is equivalent to $\neg\phi\mathbf{U}\neg\psi$

Operator Redundancy

Theorem

Every LTL formula can be expressed in terms of $\vee$, $\neg$, X, U *only.*

How?

- $\phi \wedge \psi \equiv \neg[\neg\phi \vee \neg\psi]$
- $\mathbf{F}\phi \equiv \top\ \mathbf{U}\phi$
- $\mathbf{G}\phi \equiv \neg\mathbf{F}\neg\phi$
- $\phi\mathbf{R}\psi \equiv \neg[\neg\phi\mathbf{U}\neg\psi]$

Inputs, Outputs, and Transition Systems

Languages used in practice usually also support inputs and outputs

Such specifications then describe (discrete time) systems.

`Demo` SAL (Symbolic Analysis Laboratory)

State properties refer not only to states, but also to inputs and outputs

Counter-examples

Recall: $(S, S_0, R) \models \phi$ iff
for all paths π of (S, S_0, R), $\pi \models \phi$.

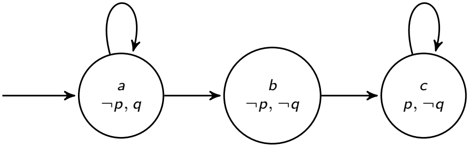
`Demo` SAL (Symbolic Analysis Laboratory)

If not $\models \phi$, information for debugging?

path π of (S, S_0, R) s.t. $\pi \not\models \phi$
(*counter-example*)

Note: $\pi \not\models \phi$ is equivalent to $\pi \models \neg\phi$

Example:



$\models p \, \mathbf{R} \, q$? where $\mathcal{I}(p) = \{c\}$, $\mathcal{I}(q) = \{a\}$

From the definitions:

for every path π

for all j, [if for all $i < j$, $\pi^i \not\models p$, then $\pi^j \models q$]

$(a, a, a, a, \ldots)$? $(a, b, c, \ldots)$?

Alternatively: search for counter-example:

path π s.t. $\pi \not\models p \, \mathbf{R} \, q$

path π s.t. $\pi \models \neg[p \, \mathbf{R} \, q]$

path π s.t. $\pi \models \neg p \, \mathbf{U} \, \neg q$

From the definition:

there is i s.t. $\pi^i \models \neg q$ and for all $j < i$, $\pi^j \models \neg p$

Time Model

Here: discrete time model

One time step might correspond to

- fixed time unit (e.g., clock cycle, 1sec)
- events (e.g., key pressed)

For many applications a more flexible time model is needed (viz. further lectures).

Final Remarks

More temporal logics (CTL, CTL*,. . .)

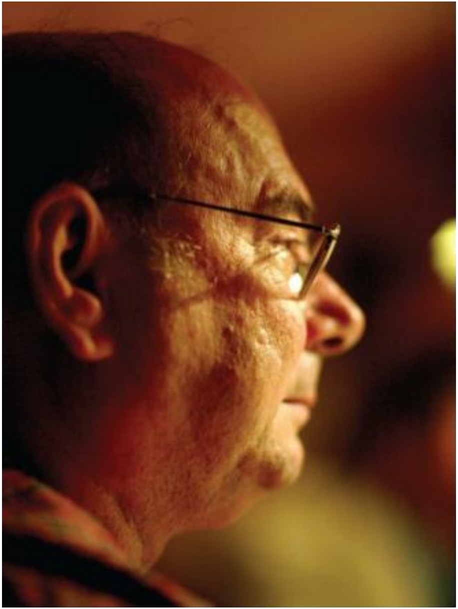
All of these logics are defined using first-order predicate logic expressions over paths. Why not directly use those expressions?

- One has to understand only a few basic operators,
  complex properties: combinations.
- Efficient algorithms ("model checking") for automatic checking
  temporal logic properties on finite state systems.

Practical usage:
LTL is the basis for various specification languages,
that are used in industry.

Example: Property Specification Language (PSL)

Amir Pnueli—Turing Award Winner



Programs as Transition Systems

$i, k \in \mathbb{N}$; $r \in \{\mathtt{prime}, \mathtt{nonprime}\}$
1 : $r \leftarrow \mathtt{prime}$
2 : $i \leftarrow 2$
3 : **while** $i < k$ and $r = \mathtt{prime}$ **do**
4 :   **if** i divides k **then** $r \leftarrow \mathtt{nonprime}$
5 : $i \leftarrow i + 1$
6 : **done**

$S = \{1, \dots, 6\} \times \mathbb{N} \times \mathbb{N} \times \{\mathtt{prime}, \mathtt{nonprime}\}$
$S_0 = \{(1, i, k, r) \mid i \in \mathbb{N}, k \in \mathbb{N}, r \in \{\mathtt{prime}, \mathtt{nonprime}\}\}$
$R = R_{1,2} \cup R_{2,3} \cup R_{3,4} \cup R_{3,6} \cup R_{4,5} \cup R_{5,3}$ where

- $R_{1,2} = \{((1, i, k, r), (2, i', k', r')) \mid i' = i \wedge k' = k \wedge r' = \mathtt{prime}\}$
- $R_{2,3} = \{((2, i, k, r), (3, i', k', r')) \mid i' = 2 \wedge k' = k \wedge r' = r\}$
- $R_{3,4} = \{((3, i, k, r), (4, i', k', r')) \mid i' = i \wedge k' = k \wedge r' = r \wedge i < k \wedge r = \mathtt{prime}\}$
- $R_{3,6} = \{((3, i, k, r), (6, i', k', r')) \mid i' = i \wedge k' = k \wedge r' = r \wedge \neg[i < k \wedge r = \mathtt{prime}]\}$
- ...

course MI-FME (Formal Methods and Specification)

Comments/Literature Guide

We give state properties a certain interpretation $\mathcal{I}$.

In the literature (e.g.,[Clarke et al., 1999]), often
predicates are added to transition system (instead of handled by logic).

The literature calls result "*Kripke structure*"

Alternative notation:

- $\bigcirc$: X
- $\Diamond$: F
- $\Box$: G

Literature

Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

Michael Huth and Mark Ryan. *Logic in Computer Science*. Cambridge University Press, 2004.