

CIS4930/6930 Principles of Cyber-Physical Systems

Chapter 12 Invariants & Temporal Logic

Hao Zheng

U. Of South Florida


When is a Design of a System “Correct”?

A design is correct when it meets its specification (requirements) in its operating environment


“A design without specification cannot be right or wrong, it can only be surprising!”

Simply running a few tests is not enough!

Many embedded systems are deployed in safety-critical applications (avionics, automotive, medical, ...)



Ariane disaster, 1996
\$500 million software failure



FDIV error, 1994
\$500 million

```
<msblast.exe> (the primary executable of the exploit)
I just want to say LOVE YOU SAN!!
billy gates why do you make this possible ? Stop
making money and fix your software!!
windowsupdate.com
start %s
tftp -i %s GET %s
%d.%d.%d.%d
%i.%i.%i.%i
```

Estimated SW bugs cost: > \$50 billion

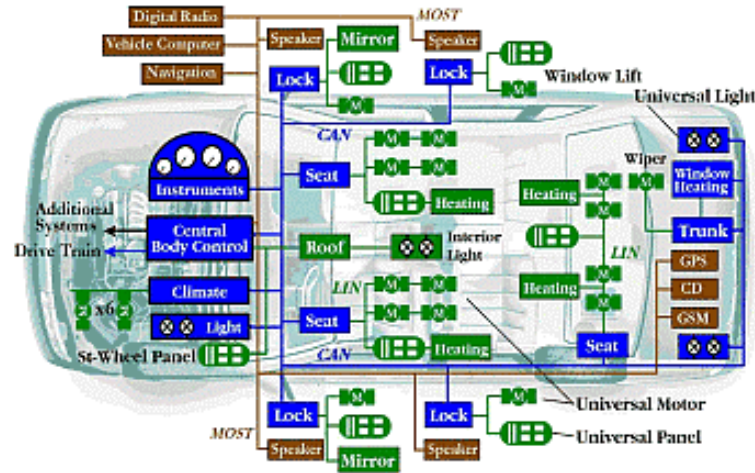
Ariane 5 Explosion



“It took the European Space Agency 10 years and \$7 billion to produce Ariane 5. All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space”

A bug and a crash, J. Gleick, New York Times, Dec 1996

Prius Brake Problems Blamed on Software Glitches



“Toyota officials described the problem as a "disconnect" in the vehicle's complex anti-lock brake system (ABS) that causes less than a one-second lag. With the delay, a vehicle going 60 mph will have traveled nearly another 90 feet before the brakes begin to take hold”

CNN Feb 4, 2010

Specification and Verification

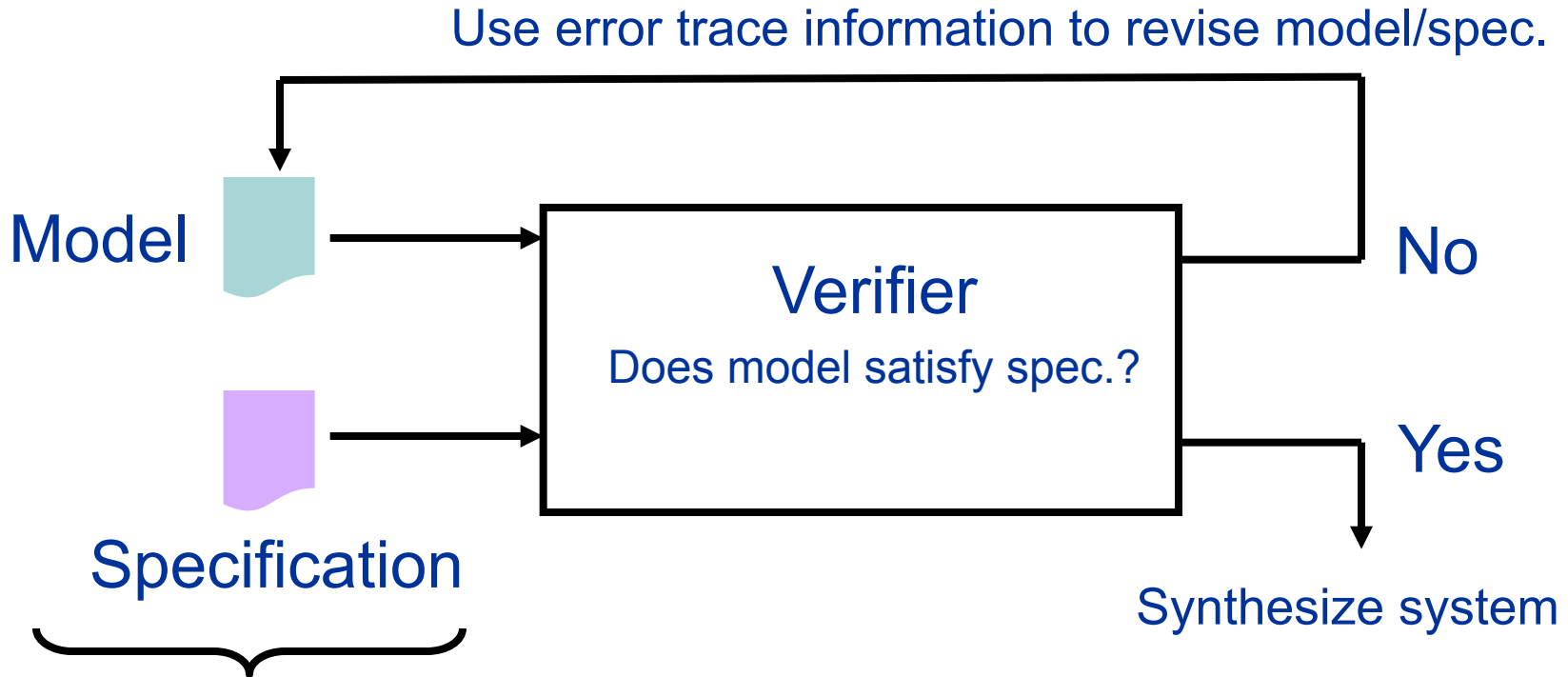
Specification

A mathematical statement of the design objective
(desired properties of the system)

Verification

Does the designed system achieve its objective in the
operating environment?

Model-Based Design: Verification



Need a mathematical way to write models and specifications so that an algorithm can process it

Temporal Logic

- A mathematical way to express properties of a system over time
 - E.g., Behavior of an FSM or Hybrid System
- Many flavors of temporal logic
 - Propositional temporal logic (we will study this)
 - Real-time temporal logic
- Amir Pnueli won ACM Turing Award, in part, for the idea of using temporal logic for specification

Example: Specification of the *SpaceWire* Protocol (European Space Agency standard)

8.5.2.2 ErrorReset

- a. The *ErrorReset* state shall be entered after a system reset, after link operation is terminated for any reason or if there is an error during link initialization.
- b. In the *ErrorReset* state the Transmitter and Receiver shall all be reset.
- c. When the reset signal is de-asserted the *ErrorReset* state shall be left unconditionally after a delay of 6,4 μs (nominal) and the state machine shall move to the *ErrorWait* state.
- d. Whenever the reset signal is asserted the state machine shall move immediately to the *ErrorReset* state and remain there until the reset signal is de-asserted.

Propositional Logic

Atomic formulas: Statements about an input, output, or state of a state machine. Examples:

formula	meaning
x	x is <i>present</i>
$x = 1$	x is <i>present</i> and has value 1
s	machine is in state s

These are propositions (true or false statements) about a state machine with input or output x and state s .

Propositional Logic

Propositional logic formulas: More elaborate statements about an input, output, or state of a state machine. Examples:

formula	meaning
$p_1 \wedge p_2$	p_1 and p_2 are both true
$p_1 \vee p_2$	either p_1 or p_2 is true
$p_1 \implies p_2$	if p_1 is true, then so is p_2
$\neg p_1$	true if p_1 is false

Here, p_1 and p_2 are either atomic formulas or propositional logic formulas.

Execution Trace of a State Machine

An **execution trace** is a sequence of the form

$$q_0, q_1, q_2, q_3, \dots, \quad \text{State : } q_i$$

where $q_j = (x_j, s_j, y_j)$ where s_j is the state at step j , x_j is the input valuation at step j , and y_j is the output valuation at step j . Can also write as

$$s_0 \xrightarrow{x_0/y_0} s_1 \xrightarrow{x_1/y_1} s_2 \xrightarrow{x_2/y_2} \dots$$

$$\text{Reaction : } s_i \xrightarrow{x_i/y_i} s_{i+1}$$

Propositional Logic on Traces

A propositional logic formula p **holds** for a trace

$$q_0, q_1, q_2, q_3, \dots,$$

if and only if it holds for q_0 .

This may seem odd, but we will provide temporal logic operators to reason about the trace.

Linear Temporal Logic (LTL)

LTL formulas: Statements about an execution trace

$q_0, q_1, q_2, q_3, \dots,$

formula	meaning
p	p holds in q_0
$\mathbf{G}\phi$	ϕ holds for every suffix of the trace
$\mathbf{F}\phi$	ϕ holds for some suffix of the trace
$\mathbf{X}\phi$	ϕ holds for the trace q_1, q_2, \dots
$\phi_1 \mathbf{U} \phi_2$	ϕ_1 holds for all suffixes of the trace until a suffix for which ϕ_2 holds.

Here, p is propositional logic formula and ϕ is either a propositional logic or an LTL formula.

Linear Temporal Logic (LTL)

LTL formulas: Statements about an execution trace

$q_0, q_1, q_2, q_3, \dots,$

formula	mnemonic
p	proposition
$\mathbf{G}\phi$	globally
$\mathbf{F}\phi$	finally, future, eventually
$\mathbf{X}\phi$	next state
$\phi_1 \mathbf{U}\phi_2$	until

Here, p is propositional logic formula and ϕ is either a propositional logic or an LTL formula.

LTL Operator: G (Globally, Always)

The LTL formula Gp holds for a trace

$q_0, q_1, q_2, q_3, \dots,$

if and only if it holds for every suffix of the trace:

$q_0, q_1, q_2, q_3, \dots$

q_1, q_2, q_3, \dots

q_2, q_3, \dots

q_3, \dots

If p is a propositional logic formula, this means it holds for each q_i .

LTL Operator: F (Eventually, Finally)

The LTL formula $\mathbf{F}p$ holds for a trace

$q_0, q_1, q_2, q_3, \dots,$

if and only if it holds for some suffix of the trace:

$q_0, q_1, q_2, q_3, \dots$

q_1, q_2, q_3, \dots

q_2, q_3, \dots

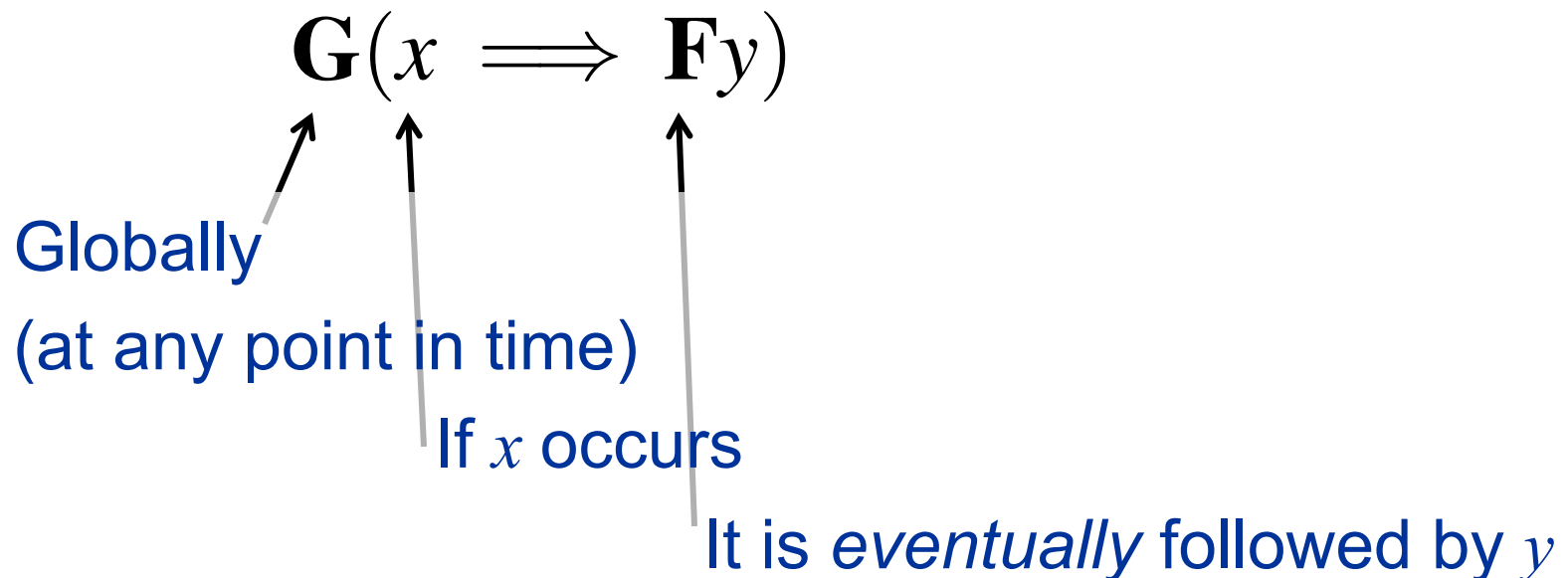
q_3, \dots

If p is a propositional logic formula, this means it holds for some q_i .

Propositional Linear Temporal Logic

LTL operators can apply to LTL formulas as well as to propositional logic formulas.

E.g. Every input x is eventually followed by an output y



Every input x is eventually followed by an output y

The LTL formula $\mathbf{G}(x \implies \mathbf{F}y)$ holds for a trace

$q_0, q_1, q_2, q_3, \dots,$

if and only if it holds for any suffix of the trace where x holds, there is a suffix of that suffix where y holds:

$q_0, q_1, q_2, q_3, \dots$
 q_1, q_2, q_3, \dots y holds
 x holds q_2, q_3, \dots
 q_3, \dots

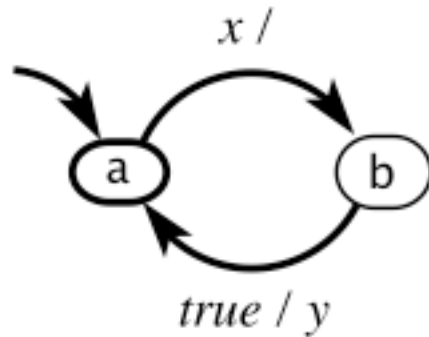
Propositional Temporal Logic

Does the following hold?

$$\mathbf{G}(x \implies \mathbf{F}y)$$

input: x : pure

output: y : pure



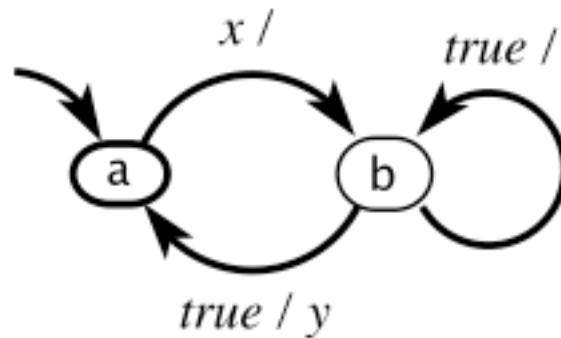
Propositional Temporal Logic

Does the following hold?

$$\mathbf{G}(x \implies \mathbf{F}y)$$

input: x : pure

output: y : pure



no

LTL Operator: X (Next)

The LTL formula Xp holds for a trace

$q_0, q_1, q_2, q_3, \dots,$

if and only if p holds for the suffix q_1, q_2, q_3, \dots

$q_0, q_1, q_2, q_3, \dots$

q_1, q_2, q_3, \dots

q_2, q_3, \dots

q_3, \dots

LTL Operator: U (Until)

The LTL formula $p_1 \mathbf{U} p_2$ **holds** for a trace

$q_0, q_1, q_2, q_3, \dots,$


if and only if p_2 holds for some suffix of the trace, and p_1 holds for all previous suffixes:

$q_0, q_1, q_2, q_3, \dots$

q_1, q_2, q_3, \dots

q_2, q_3, \dots

q_3, \dots

 p_1 holds

 p_2 holds (and maybe p_1 also)

Examples: What do they mean?

- **G F p**

p holds infinitely often

- **F G p**

Eventually, p holds henceforth

- **G(p => F q)**

Every p is eventually followed by a q

- **F(p => (X X q))**

Every p is followed by a q two reactions later

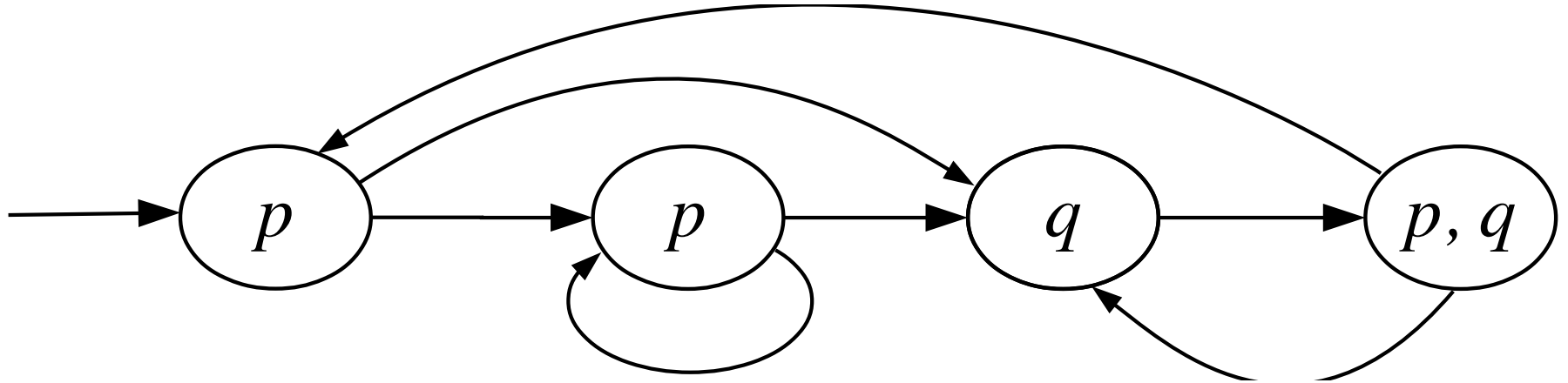
Remember:

Gp p holds in all states

Fp p holds eventually

Xp p holds in the next state

Example



$\mathbf{G}(p \vee q)$

$\mathbf{F}(p \vee q)$

$\mathbf{F}(p \wedge q)$

$\mathbf{G}(p \Rightarrow (p \mathbf{U} q))$

$\mathbf{G}(q \Rightarrow \mathbf{F}p)$

Some Points to Ponder

- A mathematical specification only includes properties that the system must or must not have
- It requires human judgment to decide whether that specification constitutes “correctness”
- Getting the specification right is often as hard as getting the design right!
 - Often the specification stage can reveal a lot of design flaws.