

# Embedded Systems Design and Modeling



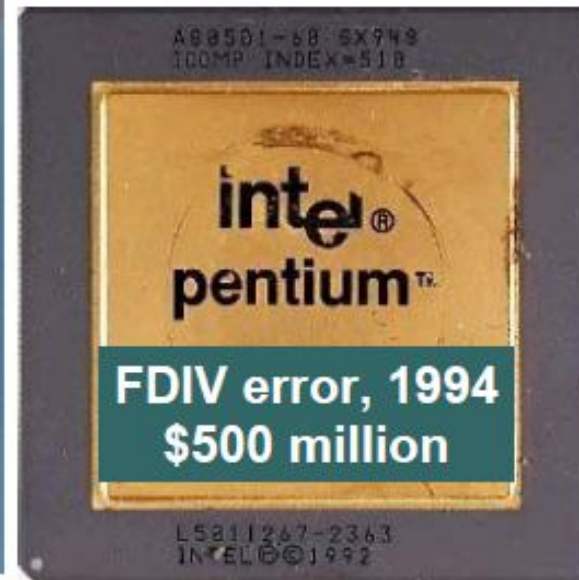
## Chapter 13 Invariants and Temporal Logic

# Correctness Definition

---

- ❑ Question: when is a design of a system “correct”?
- ❑ Answer: a design is correct when it meets its specification (requirements) in its operating environment
- ❑ Quotation: “A design without specification cannot be right or wrong, it can only be surprising!”
- ❑ To verify correctness, simply running a few tests is not enough!
- ❑ Many embedded systems are deployed in safety-critical applications (avionics, automotive, medical, ...) and require rigorous verification

# Examples From History



```
<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
```

**Estimated worst-case worm cost:**  
**> \$50 billion**

# Basic Definitions

---

## □ Specification:

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

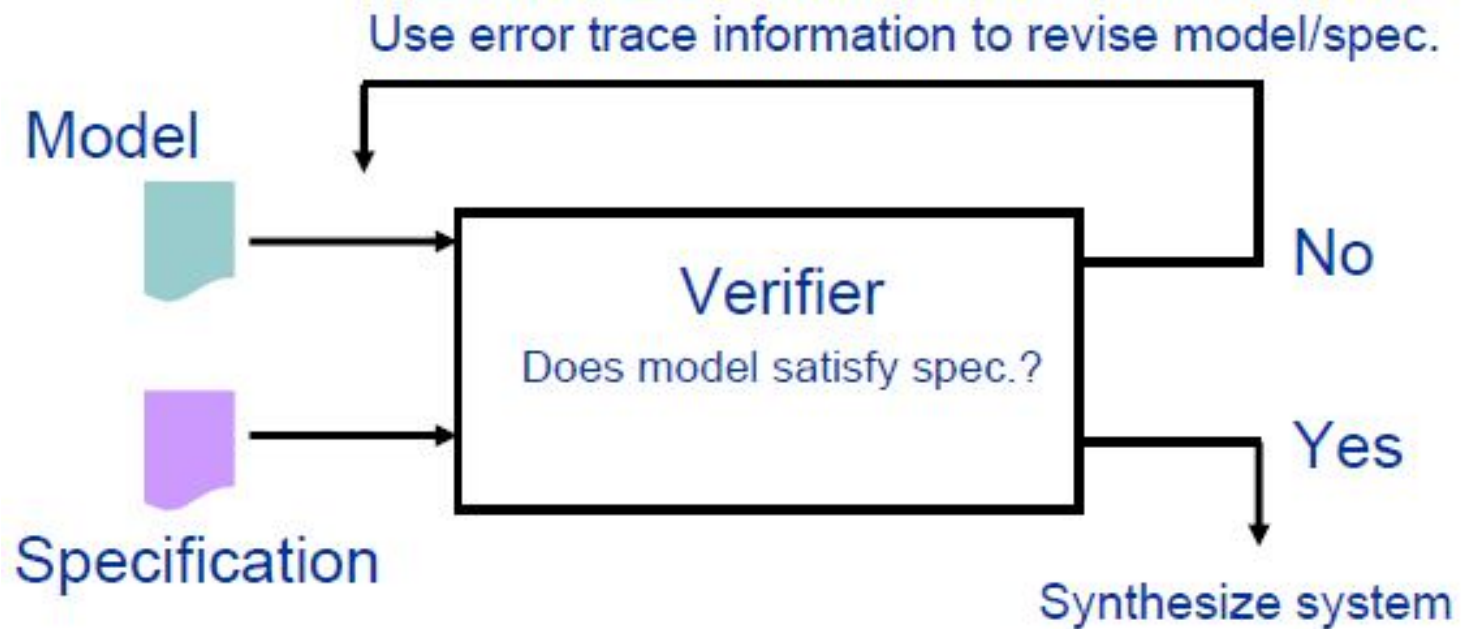
## □ Verification:

- Does the designed system achieve its objectives in the operating environment?

## □ Controller Synthesis:

- Given an incomplete design, a strategy to complete the system so that it achieves its objectives in the operating environment

# Model-Based Design & Verification



- Requires a precise and unambiguous way to write models and specifications so that an algorithm can process it

# Natural Language Deficiency

---

- ❑ Can natural languages satisfy this requirement?
- ❑ Generally no, due to their inherent ambiguities!
- ❑ Example: Specification of the SpaceWire Protocol (European Space Agency standard)

## 8.5.2.2      **ErrorReset**

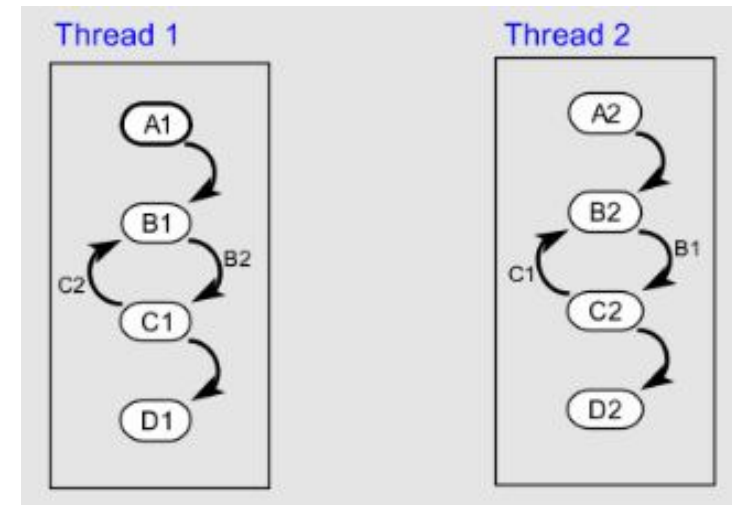
- The *ErrorReset* state shall be entered after a system reset, after link operation is terminated for any reason or if there is an error during link initialization.
- In the *ErrorReset* state the Transmitter and Receiver shall all be reset.
- When the reset signal is de-asserted the *ErrorReset* state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

Note: The exact timing of this state is not specified clearly.



# Another Example

- ❑ Recall our previous example of mutual exclusion in a multithread system
- ❑ States and/or transitions represent atomic instructions
- ❑ Sample possible specifications described in a natural language:
  - The 2-threaded program should never be in state (C1,C2)
  - Thread 1 must eventually reach D1 and thread 2 must eventually reach D2



# Motivational Observations

---

- ❑ Need a formal (mathematical) way (language) to specify the system.
- ❑ Various “logics” (mathematical languages) have been proposed to address this need.
- ❑ 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!



# Temporal Logic

---

- A precise mathematical description to express properties of a system over time
  - E.g., Behavior of an FSM or Hybrid System
  - “Temporal” emphasizes the time aspect
- Many flavors of temporal logic:
  - Propositional temporal logic
  - Linear temporal logic
  - Real-time temporal logic, etc.
- ACM Turing Award was given for the idea of using temporal logic for specification

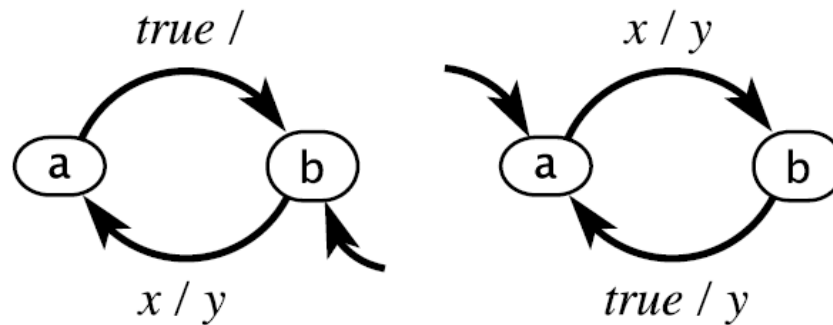
# Propositional Temporal Logic

---

- Proposition: a statement about the inputs, outputs, or states of a system
- Can be seen as expressions with true or false values
- Atomic (smallest) proposition: fine-grained statements with as few as one single input or output or state
- A propositional logic formula (or simply a proposition) is a more elaborate combination of atomic propositions

# Atomic Propositions Example

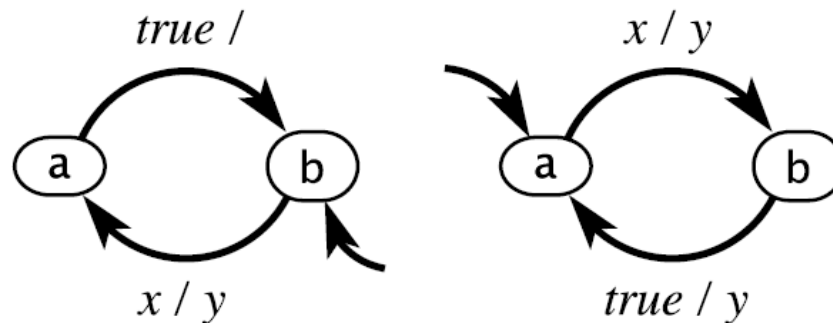
**input:**  $x$ : pure  
**output:**  $y$ : pure



$true$	Always true.
$false$	Always false.
$x$	True if input $x$ is <i>present</i> .
$x = present$	True if input $x$ is <i>present</i> .
$y = absent$	True if $y$ is <i>absent</i> .
$b$	True if the FSM is in state $b$

# Propositions Example

**input:**  $x$ : pure  
**output:**  $y$ : pure



$x \wedge y$

True if  $x$  and  $y$  are both *present*.

$x \vee y$

True if either  $x$  or  $y$  is *present*.

$x = \text{present} \wedge y = \text{absent}$

True if  $x$  is *present* and  $y$  is *absent*.

$\neg y$

True if  $y$  is *absent*.

$a \implies y$

True if whenever the FSM is in state  $a$ , the output  $y$  will be made present by the reaction

# Execution Traces

---

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

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

**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**

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

# Linear Temporal Logic

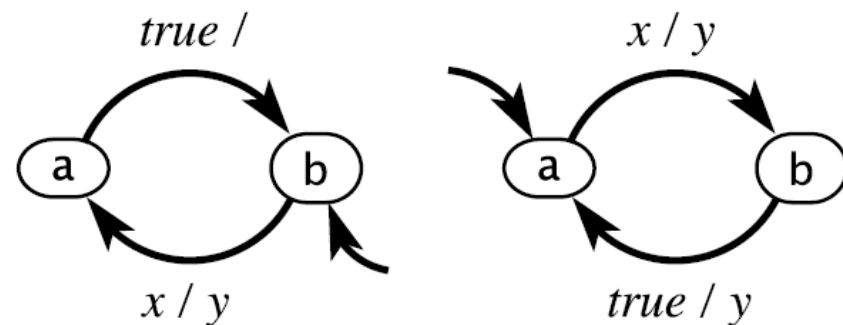
---

- LTL formula: applies to an entire trace instead of just one single element:
  - $q_0, q_1, q_2, \dots$
- If  $p$  is a proposition, then by definition, we say that LTL formula  $\Phi = p$  holds for the trace  $q_0, q_1, q_2, \dots$  if and only if  $p$  is true for  $q_0$ .
- This may seem odd, but will provide temporal logic operators ways to reason about the entire trace.
- By convention, LTL formulas are denoted as  $\Phi, \Phi_1, \Phi_2$ , etc. and propositions as  $p, p_1, p_2$ , etc.
- Given a state machine  $M$  and an LTL formula  $\Phi$ , we say that  $\Phi$  holds for  $M$  if  $\Phi$  holds for all possible traces of  $M$ .
  - This typically requires considering all possible input combinations

# LTL Example

- The LTL formula  $a$  holds for right hand side machine because all traces begin in state  $a$ .
- It does not hold for left hand side machine because there exists at least one trace that doesn't start in state  $a$ .
- The LTL formula  $x \Rightarrow y$  holds for both machines because if  $x$  is present, then  $y$  will be present.
- The LTL formula  $y$  is false for both FSMs because there is a counterexample where  $x$  is absent in the first reaction.

**input:**  $x$ : pure  
**output:**  $y$ : pure



# LTL Formulas

---

formula	mnemonic	meaning
$p$	proposition	$p$ holds in $q_0$
$G\phi$	globally	$\phi$ holds for every suffix of the trace
$F\phi$	finally, future, eventually	$\phi$ holds for some suffix of the trace
$X\phi$	next state	$\phi$ holds for the trace $q_1, q_2, \dots$
$\phi_1 U \phi_2$	until	$\phi_1$ holds for all suffixes of the trace until a suffix for which $\phi_2$ holds.



# G Operator

## □ Globally $\Phi$ :

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  $q_i$ .

- Example:  $G(x \Rightarrow y)$  is true for all traces of the right hand side machine, and hence holds for the machine.
- $G(x \wedge y)$  does not hold for the machine, because it is false for any trace where  $x$  is absent in any reaction.

# F Operator

---

## □ Finally $\Phi$ or eventually $\Phi$ :

The LTL formula  **$Fp$**  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:

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

$q_1, q_2, q_3, \dots$

$q_2, q_3, \dots$

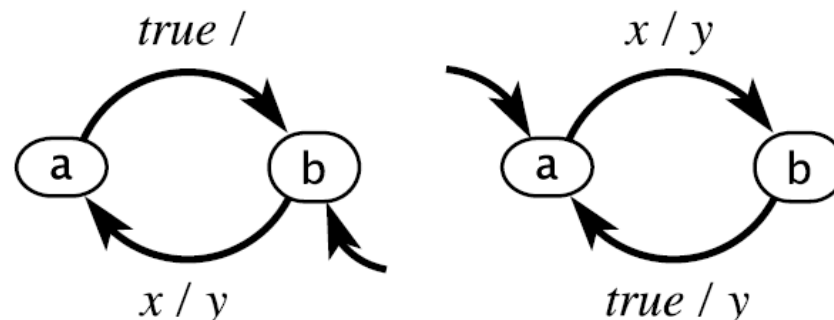
$q_3, \dots$

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

# F Operator Examples

- $G(x \Rightarrow Fb)$  holds for both machines:
  - If  $x$  is present in any reaction, then the machine will eventually be in state  $b$ .
  - True even in suffixes that start in state  $a$ .
- Parentheses (order) can be important in interpreting an LTL formula:
  - $(Gx) \Rightarrow (Fb)$  is trivially true because  $Fb$  is true for all traces
- $F\neg\Phi$  holds if and only if  $\neg G\Phi$ :
  - $(\Phi \text{ is eventually false}) = (\Phi \text{ is not always true})$

**input:**  $x$ : pure  
**output:**  $y$ : pure

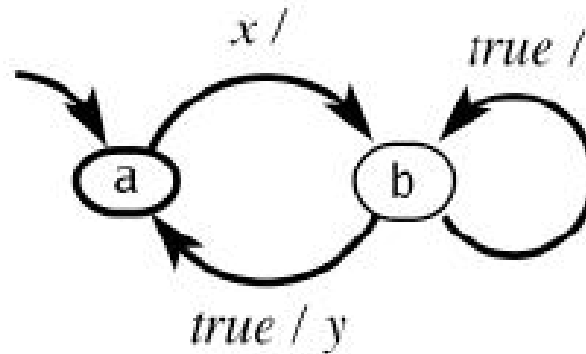


Embedded Syster

# F Operator Examples (Continued)

- Does  $G(x \Rightarrow Fy)$  hold for this machine?
  - No because there is a counterexample in which  $y$  is not present even though  $x$  is present.

**input:**  $x$ : pure  
**output:**  $y$ : pure



# X Operator

---

## □ Next state $\Phi$ :

The LTL formula  **$X\Phi$**  holds for a trace

**$q_0$** ,  $q_1$ ,  $q_2$ ,  $q_3$ , ...,

if and only if it holds for the suffix  $q_1$ ,  $q_2$ ,  $q_3$ , ...

$q_0$ ,  $q_1$ ,  $q_2$ ,  $q_3$ , ...

**$q_1$** ,  $q_2$ ,  $q_3$ , ...

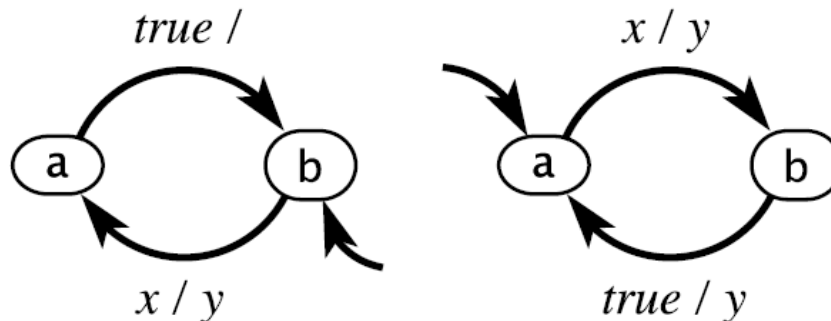
$q_2$ ,  $q_3$ , ...

$q_3$ , ...

# X Operator Examples

- $x \Rightarrow Xa$  holds for the left side state machine:
  - If  $x$  is present in the first reaction, then the next state will be  $a$ .
- $G(x \Rightarrow Xa)$  does not hold for the same state machine:
  - It does not hold for any suffix that begins in state  $a$ .
- $G(b \Rightarrow Xa)$  holds for the right side state machine.

**input:**  $x$ : pure  
**output:**  $y$ : pure



# U Operator

## □ Until operator:

The LTL formula  $p_1 U p_2$  **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:

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

$q_1, q_2, q_3, \dots$

$q_2, q_3, \dots$

$q_3, \dots$

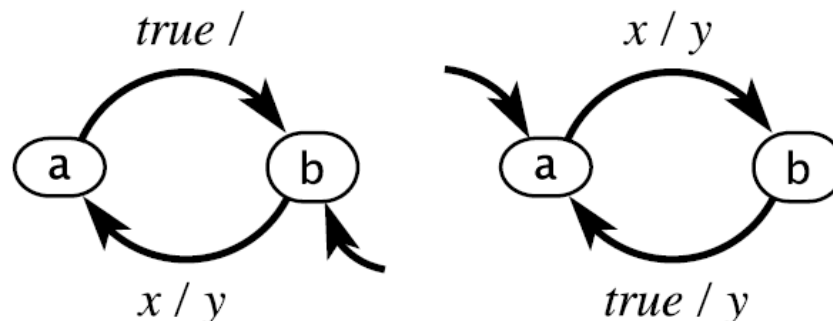
  $p_1$  holds

  $p_2$  holds (and maybe  $p_1$  also)

# U Operator Examples

- For the right side machine,  $aUx$  is true for any trace for which  $Fx$  holds.
- Since this does not include all traces,  $aUx$  does not hold for the state machine.

**input:**  $x$ : pure  
**output:**  $y$ : pure





# What do these mean?

---

- $G F p$ :
  - $p$  holds infinitely often
- $F G p$ :
  - Eventually,  $p$  holds henceforth (steady state)
- $G(p \Rightarrow F q)$ :
  - Every  $p$  is eventually followed by a  $q$  (request-response)
- $F(p \Rightarrow (XXq))$ :
  - If  $p$  occurs, then on some occurrence it is followed by a  $q$  two reactions later

# Operator Relationships

---

- Can one express  $G\Phi$  purely in terms of  $F$ ,  $p$ , and Boolean operators?
  - Yes:  $G\Phi = \neg F \neg \Phi$
- How about  $F$  in terms of  $U$ ?
  - $F\Phi = \text{true} \cup \Phi$
- What about  $X$  in terms of  $G$ ,  $F$ , or  $U$ ?
  - Cannot be done!

# Invariants

---

- An invariant is a property that holds for a system if it remains true at all times during operation of the system.
  - An invariant holds for a system if it is true in the initial state of the system, and it remains true as the system evolves, after every reaction, in every state.
- Example: In the model of a traffic light controller, there is no pedestrian crossing when the traffic light is green.
- This property must always remain true of this system, and hence is a system invariant.

# Invariants (Continued)

---

- ❑ Invariant properties must include both software and hardware aspects of an embedded system
  - Software:
    - ❑ Correct programming style
    - ❑ Deadlock prevention in mutexes
    - ❑ Input data restrictions
    - ❑ etc.
  - Hardware:
    - ❑ Timing requirements
    - ❑ Data settlement issues
    - ❑ etc.

# Homework Assignments

---

- Chapter 13:
  - Mandatory: 2, 4
  - Due date Sunday 1403/3/13
  - The rest optional