Introduction to Embedded Systems

Edward A. Lee

UC Berkeley

EECS 149/249A

Fall 2016

Chapter 13: Specification and Temporal Logic

When is a Design "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!"

[paraphrased from Young et al., 1986]

Simply running a few ad-hoc tests is not enough!

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

The Challenge of Dependable Software in Cyber-Physical Systems

Today's medical devices run on software… software defects can have life-threatening consequences.

[different device]

[From the Journal of Pacing and Clinical Electrophysiology, 2004]

"the patient collapsed while walking towards the cashier after refueling his car […] A week later the patient complained to his physician about an increasing feeling of unwell-being since the fall."

"In 1 of every 12,000 settings, the software can cause an error in the programming resulting in the possibility of producing paced rates up to 185 beats/min."

Specification, Verification, and Control

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?

Controller Synthesis

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

Temporal Logic

- A formal 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 today)
  - Real-time temporal logic
  - Signal temporal logic (used in CyberSim's autograder)
  - …

- 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.

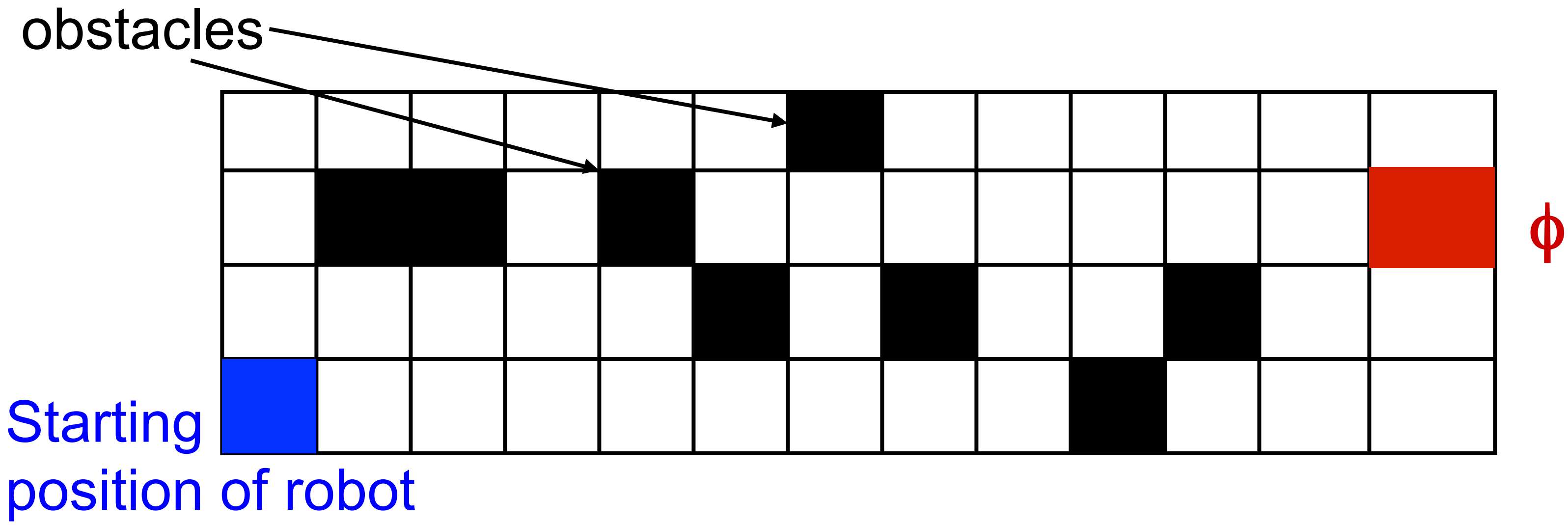
d. Whenever the reset signal is asserted the state machine shall move immediately to the *ErrorReset* state and remain there until the reset signal is de-asserted.

Example from Interrupts Lecture

```
    volatile uint timerCount = 0;
    void ISR(void) {
D →    … disable interrupts
       if(timerCount != 0) {
E →       timerCount--;
       }
       … enable interrupts
    }
    int main(void) {
       // initialization code
       SysTickIntRegister(&ISR);
       … // other init
A →    timerCount = 2000;
B →    while(timerCount != 0) {
        … code to run for 2 seconds
       }
C → … whatever comes next
    }
```

Property:
Assuming interrupts can occur infinitely often, it is always the case that position C is reached.

Robotic Navigation: Specifying Goals



ϕ = destination for robot

Specification:

The robot eventually reaches ϕ

Suppose there are n destinations $\phi_1, \phi_2, \ldots, \phi_n$

The new specification could be that

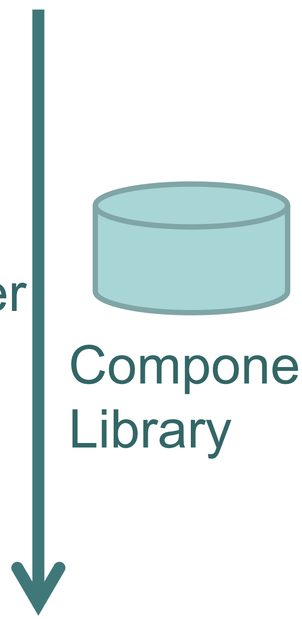
The robot visits $\phi_1, \phi_2, \ldots, \phi_n$ in that order

Multi-Robot Motion Planning from Temporal Logic: Software Synthesis for Robotics

[Saha et al., IROS 2014]

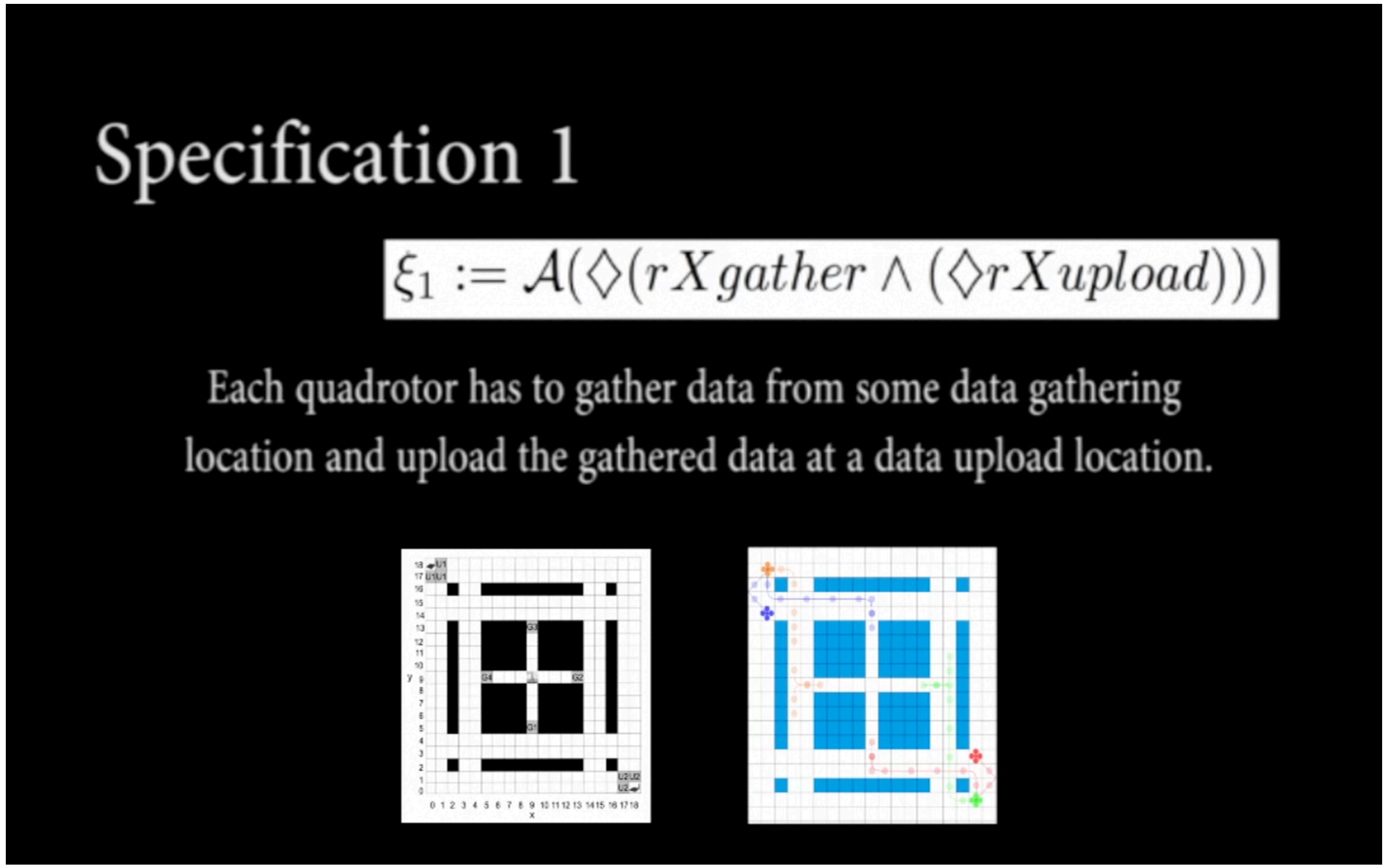
Declarative Task Specification (Temporal Logic) [+ Examples]

SMT Solver



Component Library

Executable Software



Automated Synthesis of Multi-Robot Motion Plan from LTL Specifications

Saha, Ramaithitima, Kumar, Pappas, Seshia (**Penn** and **Berkeley**)

Simple Example

“Currently, GOOG is above 700”

Propositional Logic

Atomic formulas: Statements about an input, output, or state of a state machine (at the current time).

Examples:

formula	meaning
x	x is *present*
$x = 1$	x is *present* 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.

Example

"Currently, GOOG is above 700 and AAPL is below 150"

Propositional Logic

Propositional logic formulas: More elaborate statements about an input, output, or state of a state machine (at the current time). 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.

Quiz

If p_1 is false, what is the truth value of

$$p_1 \implies p_2$$

?

"If Lassie is a frog, then she is an amphibian."

"Lassie is a frog" implies that "Lassie is an amphibian."

"If Lassie is a frog, then she is a mammal."

Whenever it is true that "Lassie is a frog," it must be true that "Lassie is a mammal."

Execution Trace of a State Machine

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

$$q_0,\ q_1,\ q_2,\ q_3,\ \ldots,$$

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

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

Example

"GOOG started above 700"

GOOG(0) > 700

"GOOG will eventually rise above 750"

There exists a $t > 0$ s.t. GOOG(t) > 750

F(GOOG(t)) > 750

Propositional Logic on Traces

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

$$q_0,\ q_1,\ q_2,\ q_3,\ \ldots,$$

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

$$q_0,\ q_1,\ q_2,\ q_3,\ \ldots,$$

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 $q_1, q_2, \cdots$
$\phi_1 \mathbf{U} \phi_2$	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

$$q_0, q_1, q_2, q_3, \ldots,$$

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.

First LTL Operator: G (Globally)

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

$$q_0, q_1, q_2, q_3, \ldots,$$

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

$$\begin{array}{r}
q_0, q_1, q_2, q_3, \ldots \\
q_1, q_2, q_3, \ldots \\
q_2, q_3, \ldots \\
q_3, \ldots
\end{array}$$

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

G p for propositional formula p, is also termed an invariant

Second LTL Operator: F (Eventually, Finally, Future)

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

$$q_0,\ q_1,\ q_2,\ q_3,\ \ldots,$$

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

$$\begin{aligned} q_0,\ q_1,\ q_2,\ q_3,\ &\ldots \\ q_1,\ q_2,\ q_3,\ &\ldots \\ q_2,\ q_3,\ &\ldots \\ q_3,\ &\ldots \end{aligned}$$

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

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

Globally
(at any point in time)

If x occurs

It is *eventually* followed by 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

$$q_0, q_1, q_2, q_3, \ldots,$$

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:

$$\begin{array}{rl} & q_0, q_1, q_2, q_3, \ldots \\ & q_1, q_2, q_3, \ldots \quad y \text{ holds} \\ x \text{ holds} & q_2, q_3, \ldots \\ & q_3, \ldots \end{array}$$

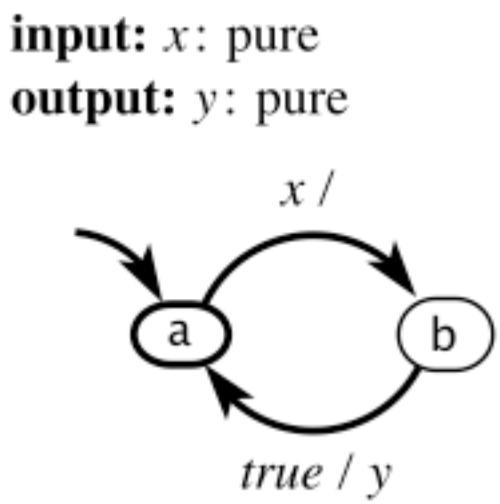
When is a Temporal Logic formula satisfied by a State Machine?

A linear temporal logic (LTL) formula is satisfied by a state machine iff *every trace* of that state machine satisfies the LTL formula.

Test Your Understanding: Qn 1

Does the following temporal logic property hold for the state machine below?

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

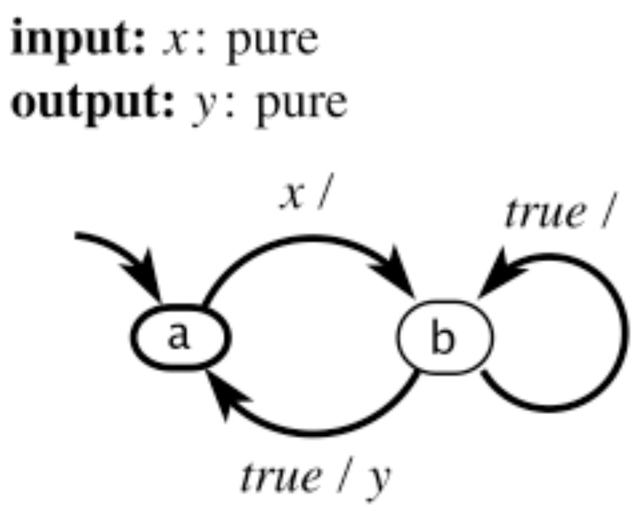


Yes

Test Your Understanding: Qn 2

Does the following hold?

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



No. What's the error trace?

Third LTL Operator: X (Next)

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

$$q_0,\ q_1,\ q_2,\ q_3,\ \ldots,$$

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

$$\begin{array}{r}
q_0,\ q_1,\ q_2,\ q_3,\ \ldots \\
q_1,\ q_2,\ q_3,\ \ldots \\
q_2,\ q_3,\ \ldots \\
q_3,\ \ldots
\end{array}$$

Fourth LTL Operator: U (Until)

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

$$q_0, q_1, q_2, q_3, \ldots,$$

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

$$q_0, q_1, q_2, q_3, \ldots$$
$$q_1, q_2, q_3, \ldots$$
$$q_2, q_3, \ldots$$
$$q_3, \ldots$$

p_1 holds

p_2 holds (and maybe p_1 also)

Note: A variant, called "weak until," written W, does not require p_2 to eventually hold. The "U" version does.

Alternate Notation

Sometimes you'll see alternative notation in the literature:

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

Examples: What do they mean?

<u>Remember:</u>
Gp   p holds in all states
Fp   p holds eventually
Xp   p holds in the next state

- $\mathbf{G}\ \mathbf{F}\ p$

  p holds infinitely often

- $\mathbf{F}\ \mathbf{G}\ p$

  Eventually, p holds henceforth

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

  Every p is eventually followed by a q

- $\mathbf{F}(\ p \Rightarrow (\mathbf{X}\ \mathbf{X}\ q)\ )$

  If p occurs, then on some occurrence it is followed by a q two reactions later

Temporal Operators & Relationships

G, F, X, U: All express properties along system traces

- Can you express G p purely in terms of F, p, and Boolean operators ?

$$\mathbf{G}\phi = \neg\mathbf{F}\neg\phi$$

- How about F in terms of U?

$$\mathbf{F}\phi = \mathit{true}\ \mathbf{U}\ \phi$$

- What about X in terms of G, F, or U?

Cannot be done

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!

- Interesting research directions:
  - Inferring temporal logic from system traces
  - Translating natural language into (temporal) logic

Exercises: Write in Temporal Logic

1. "Whenever the iRobot is at the ramp-edge (cliff), eventually it moves 5 cm away from the cliff."
   - p – iRobot is at the cliff
   - q – iRobot is 5 cm away from the cliff

2. "Whenever the distance between cars is less than 2m, cruise control is deactivated"
   - p – distance between cars is less than 2 m
   - q – cruise control is active

More Exercises

Write the SpaceWire specs. in Temporal Logic

Also write the specification for the Robot and Interrupt-based Program examples in Temporal Logic