



Fundamental Algorithms for System Modeling, Analysis, and Optimization

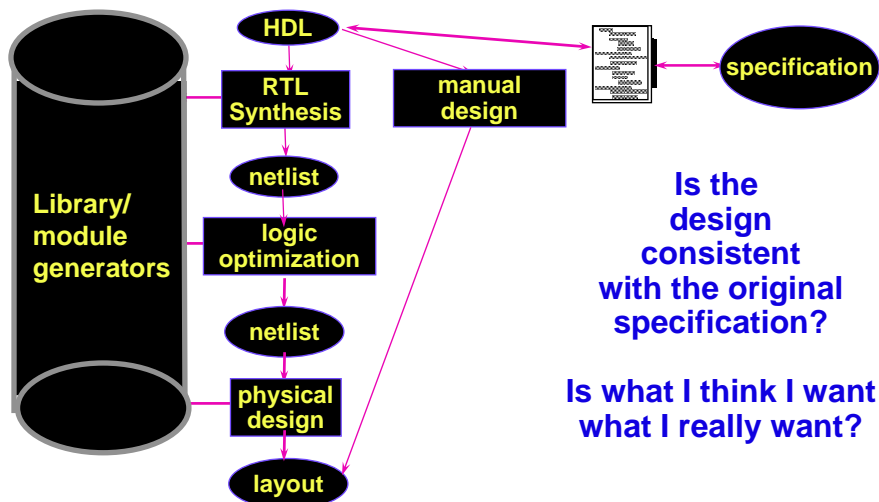
Edward A. Lee, Jaideep Roychowdhury,
Sanjit A. Seshia

UC Berkeley
EECS 144/244
Fall 2011

Copyright © 2010-11, E. A. Lee, J. Roychowdhury,
S. A. Seshia, All rights reserved

Formal Specification: Temporal Logic

Design Verification



Today's Lecture

Formal (mathematical) Specification

How do you formally state
what your design should do?

3

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

4

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 μ s (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.

5

Current status of property specification in HW

Usage of formal property specification languages is becoming widespread

- 68% in 2007 (John Cooley, DVCon'07)
- Properties often called "assertions"

Properties are used not just in formal verification, but also in simulation

- "Assertion-Based Verification" (ABV)

Some property specification languages: PSL/Sugar, System Verilog Assertions (SVA), OVA, OVL, etc.

All of these are just ways of writing variants of Temporal Logic

6

Lecture Outline

Behavior/Trace/Execution

Specifying Properties: Safety vs. Liveness

Temporal Logic (its 2 main flavors)

Synthesizing Monitors from Properties

8

Execution Trace of a State Machine

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

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

9

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.

10

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.

11

Linear Temporal Logic (LTL)

LTL formulas: Statements about an execution trace

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

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

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

12

First LTL Operator: G (Globally)

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 .

13

Second 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

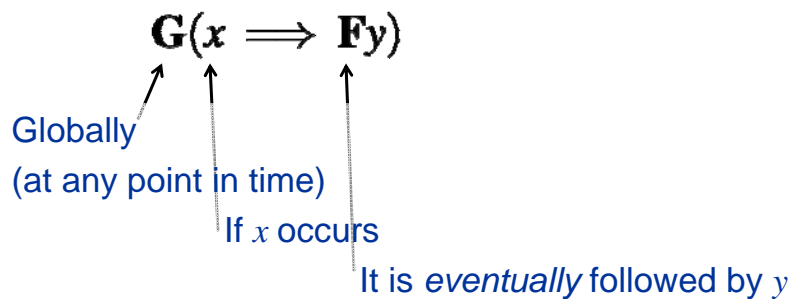
If p is a propositional logic formula, this means it holds for some q_i .

14

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



15

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

16

Third LTL Operator: X (Next)

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

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

if and only if it 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

17

Fourth LTL Operator: U (Until)

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)

18

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 steps later

Remember:

Gp p holds in all states

Fp p holds eventually

Xp p holds in the next state

19

Temporal Operators & Relationships

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

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

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

- o How about F in terms of U?

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

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

Cannot be done

20

Examples in Temporal Logic

“No more than one processor (in a 2-processor system) should have a cache line in write mode”

- wr_1 / wr_2 are respectively true if processor 1 / 2 has the line in write mode

“The grant signal must be asserted at some time after the request signal is asserted”

- Signals: grant, req

“A request signal must receive an acknowledge and the request should stay asserted until the acknowledge signal is received”

- Signals: req, ack

21

Safety vs. Liveness

Safety property

“something bad must not happen”

E.g.: system should not crash

Finite length error trace

Liveness property

“something good must happen”

E.g.: every packet sent must be received at its destination

Infinite length error trace

22

Asserts in PSL/Sugar (Verilog flavor)

$G (req \rightarrow X(X(X \text{ grant})))$

assert always req \rightarrow next[3] (grant);

$G(req \rightarrow X (\text{ack} U \text{ grant}))$

assert always req \rightarrow next (ack until grant);

23

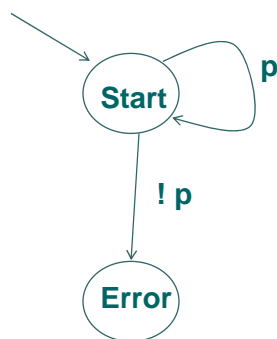
From Temporal Logic to Monitors

A monitor for a temporal logic formula
is a state machine
represents all the behaviors that satisfy the temporal
logic formula

Why are monitors useful?

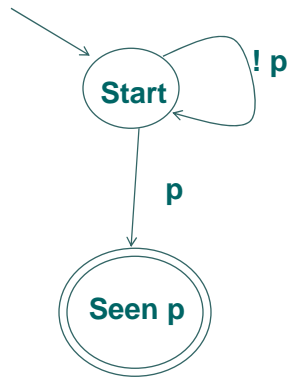
24

Monitor for $G p$, p a Boolean formula



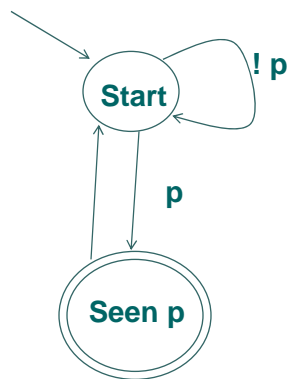
25

Monitor for $F p$, p a Boolean formula



26

Monitor for GFp , p a Boolean formula



27

Some User Reactions

“We're using SVA. I expect new RTL to be as littered with assertions as the Wisconsin countryside is littered with cheese shops & taverns.”

Make it easy to write and embed in RTL

“Started using Sugar PSL and OVA. Not clear yet as to the advantages. You have to debug the assertions, too!”

Specifications must be debugged too!

“We use 0-In assertions. I would say that our current maturity with the 0-In tools puts us at 50% efficiency. In some instances the assertions have little to no value. In some instances they are essential.” Training and improved scalability needed

“Awesome Baby!!!!!!!!!! Use PSL and they are very useful. ”

What more can one say!

Compiled by John Cooley, at DVCon'05