1

CS 4390/5387
SOFTWARE V&V

LECTURE 6
SEMESTER PROJECT

Outline

2

- Semester Project Motivation
- Background
  - Linear Temporal Logic (LTL)
  - Specification Pattern System (SPS)
  - Composite Propositions (CP)
- Team Assignment

Motivation -1

3

- Software plays a major role in our daily life.
- Errors in software can be fatal.
- Conventional verification techniques, e.g., testing, are not always adequate.
  - Software errors cost U.S. economy $59.5 billion annually.
  - $22.2 billion can be saved to U.S. economy if verification is done at earlier stages (NIST, June 2002).
- Formal verification techniques are effective at detecting errors
  - Theorem Provers
  - Model Checkers
  - Run-Time Monitors

Formal Verification Techniques

4

- Provide extra layer of assurance over testing and simulation
- Discover errors missed by traditional techniques
  - Theorem Proving
  - Runtime Monitoring
  - Model Checking

Theorem Proving

- Specify system and properties as logic formulas
- Derive properties from system

- System:
  - A -> B
  - B -> C
  - A
- Property:
  - C

Runtime Monitoring

- Specify system properties in temporal logic
- Verify system correctness at runtime
- Report violations

System:

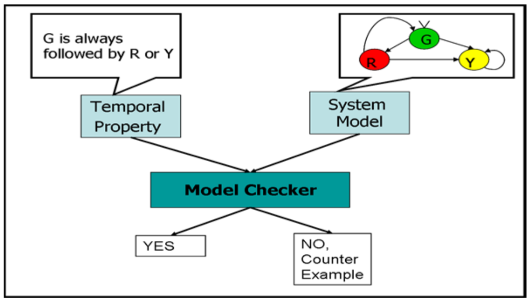
```
while result < 0 {
  if x > 5 then
    x ← x + 1
  result ← result + 1 }
```

Property

$G(x < 100)$

Model Checking

7



Motivation - 2

8

- Defining formal property specifications is a major component of any formal verification technique
- Writing formal specifications of software properties is error prone.
- Specifying behaviors, where multiple conditions and events are involved, requires extra considerations.
  - SSL (Secure Socket Layer) protocol example:

    SSL takes messages to be transmitted, fragments the data into manageable blocks, applies a MAC, encrypts, and transmits the results.

Background

9

- Linear Temporal Logic (LTL)
- Specification Pattern System (SPS)
- Composite Propositions (CP)

Logical System

10

- Two parts
  - A syntax for writing down concepts
  - A set of rules for manipulating the symbols
- Many logical systems
  - Propositional
  - First order
  - Higher order
  - Temporal
  - Non-monotonic
  - …

Logical System

11

- Two parts
  - A syntax for writing down concepts
  - A set of rules for manipulating the symbols
- Many logical systems
  - Propositional
  - First order
  - Higher order
  - Temporal
  - Non-monotonic
  - …

Why do we need logical systems?

Symbolic Reasoning, typical process

12

- Map a set of things in the real world to a set of symbols
- Operate on the symbols
- Map the result back to the real world

Symbolic Reasoning, typical process

13

- Map a set of things in the real world to a set of symbols
- Operate on the symbols
- Map the result back to the real world
- Example: 2 + 5 = 7

Symbolic Reasoning, typical process

14

- Map a set of things in the real world to a set of symbols
- Operate on the symbols
- Map the result back to the real world
- Example: 2 + 5 = 7

These are all symbols. There is nothing about the real world here. **Why is this useful?**

Symbolic Reasoning, Typical Process

15

- Map a set of things in the real world to a set of symbols
- Operate on the symbols
- Map the result back to the real world
- Example: 2 + 5 = 7

If I have 2 apples and 5 apples, in total I have 7 apples.

The "7" has meaning in the real world using the mapping that we use for "2" and "5".

Propositions

16

- Declarative sentences
- Either true or false
- Examples:
  - It is raining.
  - The door is blue.
  - The sum of 3 and 7 is 8.
  - Every natural number is the sum of two primes (Goldbach's conjecture)
- Non-examples:
  - Go climb a rock.
  - Is it Monday?

Propositional Variables

17

- A countable infinite set of symbols: p, q, r (numbered with subscripts as needed)
- We assign atomic propositions to the propositional variables
- Example: r == "It is raining"

Propositional Logic Syntax

18

- Var: the (infinite) set of propositional variables
  - We'll use p, q, r, r_2, …
- Logical Connectives: $\neg\wedge\vee\rightarrow$
- F: the set of all propositional formulas
  - If $P \in$ Var, $P \in F$
  - If $p \in F$, then $\neg p \in F$
  - If $p,q \in F$, then $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q) \in F$
  - Only these are propositional formulas

Example Propositional Formulas

19

- p
- $\neg\neg p$
- $((p \wedge q) \vee r)$
- $\neg(q \vee r)$

Propositional Logic: Semantics

20

- Binding:
  - $\neg$ binds most tightly
  - $\wedge\vee$ bind next.
  - $\rightarrow$ is weakest.
  - $P \wedge Q \rightarrow \neg R \vee Q == ((P \wedge Q) \rightarrow ((\neg R) \vee Q))$

  Parenthesis are added for emphasis
  We'll omit the parenthesis when convenient

21

LOGIC IN COMPUTER SCIENCE

We want to use the mathematical formalism of logic to help us ensure the correctness of programs

Mutual exclusion
Hyman in Communications of ACM, 1966

22

```
Boolean array b(0;1)                    ;; flag: true if critical section is not requested
Integer i;                              ;; process i, i=0 or 1
Integer k;                              ;; k is process requesting critical section
                                        ;;   initially k = 1-i
c0: b(i) := false                       ;; process i requests critical section
c1: if k != i then
        begin
          c2: if not b(1-i) then go to c2;   ;; loop until other process frees critical section
                else k := i;                 ;; other process freed, set request to this one
                goto c1
        end;
    else critical section;              ;; do the controlled section
    b(i) := true;                       ;; done with critical section
    remainder of program;
    go to c0;
end
```

Does this ensure that only one process can be here at a time?

Specification Language: Linear Temporal Logic (LTL)

23

- Widely used property specification language.
- Expressibility allows modeling of software properties such as liveness and safety.
- Applicable to numerous verification tools
  - Model checkers Spin, NuSMV, and Java Pathfinder
  - Runtime verification of Java programs

LTL Syntax:

24

- Atoms: atomic propositions as in propositional logic
- Formulas (Q is an LTL formula):

  Q ::= true | false | p | (¬Q) | (Q→Q) | (Q∧Q) | (Q∨Q) |
  (X Q) | (F Q) | (G Q) | (Q U Q) | (Q W Q)

  - p ∈ ATOMS
  - X (Next)
  - F (Future, <>)
  - G (Global [])
  - U (Until)
  - W (Weak until)

LTL Formulas (Manna et. al. 89)

25

- "*True*" is a well-formed LTL formula.
- "*False*" is a well-formed LTL formula.
- If *a* and *b* are well-formed LTL formulas then so is:
  - ! a !a- - - - - - - - - -
  - a ∨ b b - - - - - - - - - - a - - - - - - - - - -
  - a ∧ b (ab) - - - - - - - -
  - a U b (a until b) a a a a b - - - - -
  - X a (next a) - a - - - - - - - -
  - ◊ a (eventually a) - - - - a - - - - - -
  - □ a (always a) a a a a a a a a a a

Examples

26

- (((Fp) /\ (Gq)) -> (p W r)
- (F (p->(Gr)) V((~q) U p))
- (p W (q W r))
- ((G(Fp)) -> (F (q V s)))

Examples:

27

- The printer is always ready:
  - G p
- The printer will eventually be ready:
  - F p
- The printer will be ready until a job arrives:
  - p U j
- After a job arrives, sooner or later the printer will be ready
  - G(j -> F p)

Clause 10: Until

28

- P U Q

P P P Q _ _	TRUE
_ P Q _ _	FALSE
Q _ _ _ _ _ _	TRUE
P P P P P P P	FALSE
Q Q Q P _ _	TRUE
Q Q Q Q Q	TRUE
Q Q Q (PQ) Q	TRUE

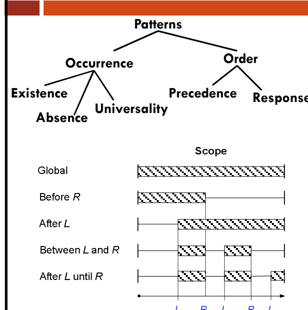
- The first thing holds continuously until the second thing holds.
- They don't have to hold in the same state
- The second thing has to occur

Examples

29

- It is impossible to get to a state where we are *started* but not *ready*.
  - G ~(started /\ ready)
- If a request is made, it will be serviced
  - G(requested -> F serviced)
- P is enabled infinitely often on every path
  - G F enabled
- Whatever happens, P will eventually become permanently deadlocked
  - F G deadlock
- If the process is enabled infinitely often, then it runs infinitely often
  - G F enabled -> G F running
- An elevator moving up at the second floor does not go down if it has passengers traveling to the 5th floor
  - G (floor2 /\ directionup /\ Button5) -> (directionup U Floor5)

Automated Support: SPS (Dwyer et al., 2002)



- Specification Pattern System (SPS)
  - Introduced patterns and scopes
  - Patterns: high-level abstractions that provide descriptions of common properties
  - Scopes: describe the extent of program execution over which a property holds
- SPS provides formal specifications in multiple formalisms, e.g., LTL, CTL

Patterns: Examples

31

- Existence of P - - - - P - - - - - - - - - -
- Q Precedes P - - - - Q - - - - P - - - - -
- Q Responds to P - - P - - Q - - P - - Q - -

Example (SPS)

32

- When a connection is made to the SMTP server, all queued messages in the Outbox mail will be transferred to the server.
  - P: Connection is made to the SMTP.
  - R: Queued messages in the Outbox are transferred to the server.
  - Existence (P) Before (R)
  - LTL formula: $(\Diamond R) \rightarrow (!R \; U \; (P \wedge !R))$

Automated Support: Composite Propositions (Mondragon et al. 2004)

33

- Need for Composite Propositions (CPs)
  - Specify concurrency and sequences
  - Help practitioner consider different behaviors
- Example revisited: SSL protocol

  SSL takes messages to be transmitted, fragments the data into manageable blocks, applies a MAC, encrypts, and transmits the results.

  - *f* - Data is fragmented . *m* - MAC is applied. *e* - Data is encrypted.
  - Possible interpretations in LTL:
    - $\Diamond (f \wedge m \wedge e)$
    - $(f \rightarrow \Diamond m) \wedge (m \rightarrow \Diamond e)$
    - $(f \wedge \Diamond (m \wedge \Diamond e))$
    - $f \wedge X \Diamond (m \wedge X \Diamond e)$

SPS: LTL Specifications

34

PATTERN	SCOPE	LTL Formula
Absence	*Global*	$\Box(\neg P)$
	Before R	$\Diamond R \rightarrow (\neg P U R)$
	After L	$\Box(L \rightarrow \Box(\neg P))$
	Between L and R	$\Box((L \wedge \neg(R) \wedge \Diamond R) \rightarrow (\neg(P) U R))$
	After L Until R	$\Box((L \wedge \neg(R)) \rightarrow (\neg(P) W R))$
Existence	*Global*	$(\Diamond P)$
	Before R	$\neg R W (P \wedge \neg R)$
	After L	$\Box(\neg L) \vee \Diamond(L \wedge \Diamond P))$
	Between L and R	$\Box((L \wedge \neg(R)) \rightarrow (\neg(R) W (P \wedge \neg R)))$
	After L Until R	$\Box((L \wedge \neg(R)) \rightarrow (\neg(R) U (P \wedge \neg R)))$
Universality	*Global*	$\Box P$
	Before R	$\Diamond R \rightarrow (P U R)$
	After L	$\Box(L \rightarrow \Box(P))$
	Between L and R	$\Box((L \wedge \neg(R) \wedge \Diamond R) \rightarrow (P U R))$
	After L Until R	$\Box((L \wedge \neg(R)) \rightarrow (P W R))$
Presedence	*Global*	$\neg(P) W T$
	Before R	$\Diamond R \rightarrow (\neg(P) U (T \vee R))$
	After L	$\Box \neg L \vee \Diamond(L \wedge (\neg P W T))$
	Between L and R	$\Box((L \wedge \neg(R) \wedge \Diamond R) \rightarrow (\neg(P) U ((T \vee R)))$
	After L Until R	$\Box(L \wedge \neg(R) \rightarrow (\neg(P) W (T \vee R)))$
Response	*Global*	$\Box(P \rightarrow \Diamond T)$
	Before R	$\Diamond R \rightarrow ((P \rightarrow (\neg(R) U (T \wedge \neg(R))))U R)$
	After L	$\Box(L \rightarrow \Box(P \rightarrow \Diamond T))$
	Between L and R	$\Box((L \wedge \neg(R) \wedge \Diamond R) \rightarrow (P \rightarrow (\neg(R) U (T \wedge \neg(R))))U R$
	After L Until R	$\Box((L \wedge \neg(R) \rightarrow (P \rightarrow (\neg(R) U (T \wedge \neg(R))))W R)$

LTL Semantics of CP Classes

35

CP Class	Informal Description	Semantics in LTL
$AtLeastOne_C$	At least one proposition in *S* holds.	$p_1 \vee p_2 \vee \ldots \vee p_n$
$AtLeastOne_E$	At least one proposition in *S* becomes *true.*	$(!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) \wedge ((!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) U (p_1 \vee p_2 \vee \ldots \vee p_n))$
$Parallel_C$	All propositions in *S* hold.	$p_1 \wedge p_2 \wedge \ldots \wedge p_n$
$Parallel_E$	All propositions in *S* become *true.*	$(!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) \wedge ((!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) U (p_1 \wedge p_2 \wedge \ldots \wedge p_n))$
$Consecutive_C$	Each proposition in *Q* holds in a specified order, one at each successive state.	$p_1 \wedge X(p_2 \wedge X(p_3 \wedge X(\ldots \wedge X(p_n) \ldots)))$
$Consecutive_E$	Each proposition in *Q* becomes *true* in a specified order, one at each successive state.	$(!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) \wedge ((!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) U (p_1 \wedge !p_2 \wedge \ldots \wedge !p_n \wedge X(p2 \wedge !p3 \wedge \ldots \wedge !p_n \wedge (\ldots \wedge X(p_{n-1} \wedge !p_n \wedge X p_n)\ldots)))))$
$Eventually_C$	Each proposition in *Q* is asserted to hold in a specified order and in distinct and possibly nonconsecutive states.	$p_1 \wedge X((!p_2 U (p_2 \wedge X (\ldots \wedge X(!p_n U p_n)\ldots)))$
$Eventually_E$	Each proposition in *Q* becomes *true* in a specified order and in distinct and possibly nonconsecutive states.	$(!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) \wedge ((!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) U (p_1 \wedge !p_2 \wedge \ldots \wedge !p_n \wedge ((!p_2 \wedge \ldots \wedge !p_n) U (p_2 \wedge !p_3 \wedge \ldots \wedge !p_n \wedge ((!p_3 \wedge \ldots \wedge !p_n) U (p_3 \wedge !p_3 \wedge \ldots \wedge !p_n \wedge ((\ldots ((p_{n-1} \wedge !p_n \wedge (!p_n U p_n)))\ldots))))))))$

LTL Semantics of CP Classes

36

CP Class	Informal Description	Semantics in LTL
$Parallel_E$	**All propositions in *S* become *true at the same state.***	**$(!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) \wedge ((!p_1 \wedge !p_2 \wedge \ldots \wedge !p_n) U (p_1 \wedge p_2 \wedge \ldots \wedge p_n))$**
$Eventual_C$	**Each proposition in *Q* is asserted to hold in a specified order and in distinct and possibly nonconsecutive states.**	**$p_1 \wedge X((!p_2 U (p_2 \wedge X (\ldots \wedge X(!p_n U p_n)\ldots)))$**

Problem

37

- Mondragon used direct substitutions of CP formulas into patterns and scope to generate formulas.
- Approach worked for formulas written in Future Interval Logic (FIL).
- Direct substitutions do not work on LTL.

Example

- The delete button is enabled in the main window only if the user is logged in as administrator and the main window is invoked by selecting it from the Admin menu.
- **Existence (Eventual$_C$ (p_1, p_2)) Before R**

p_1: User logged in as an admin.

p_2: Main window is invoked.

R: Delete button is enabled.

- Existence (P) Before R
  $(\Diamond R) \rightarrow (!R \; U \; (P \wedge !R))$
- Eventual$_C$:
  $(p_1 \wedge X \; (!p_2 \; U \; p_2))$
- By direct substitution:
  $\Diamond R \rightarrow (!R \; U \; ((p_1 \wedge X \; (!p_2 \; U \; p_2)) \wedge !R))$
- The following behavior is accepted:
  - - - - p_1 - - R - - p_2

  The delete button can be enabled between the time the admin logs in and the admin invokes the main window.

Challenge

39

- LTL formulas that use multiple propositions are:
  - Hard to specify
  - **Hard to verify**
- There are ≈ 47,000 combinations of patterns, scope, and CP.

40

Pattern/Scope	global	before R	after L	btn L&R	after L until R	
absence (P)	9	81	81	729	729	
existence (P)	9	81	81	729	729	
(Q) responds to (P)	81	729	729	6561	6561	
(Q) precedes (P)	81	729	729	6561	6561	
(Q) strictly precedes (P)	81	729	729	6561	6561	
Total	261	2349	2349	21141	21141	47241

So, on to your tasks

41

- Yadira: Precedence between L & R
- Carlos: Response between L & R
- Luis: Response after L Until R
- Florencia: Precedence after L Until R
- Salah: Strict Precedence between L & R

So, on to your tasks

42

- Your team is to develop a complete test plan for your assigned pattern/scope
  - You will have to significantly reduce the number of formulas to test (each team's assigned pattern/scope yields 6561 formulas)
  - You will have to describe a systematic approach to test the new set of formulas
  - Your deliverable (by class time on Thursday) consists of:
    - Test plan
    - Test cases
    - Presentation of your result to the class

Note that you do not need to run your test cases. You simply need to specify them