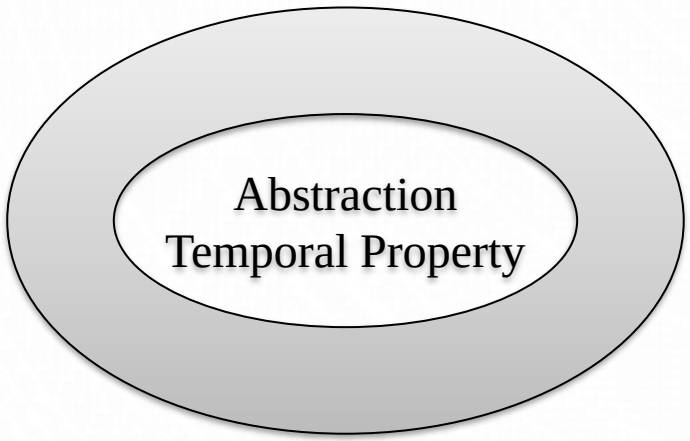
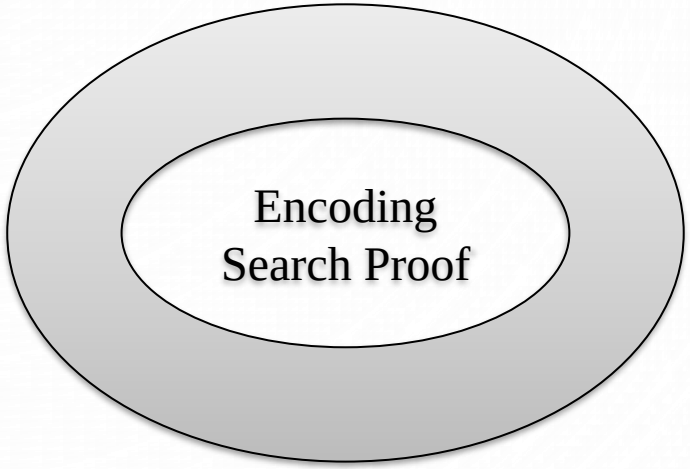
July 17 OPLSS 2018

Cyrus Liu
- Advisor, Eric Koskinen

TEMPORAL LOGIC AND PROGRAM ANALYSIS

- From Temporal Logic to Program Analysis





➢ Temporal Logic

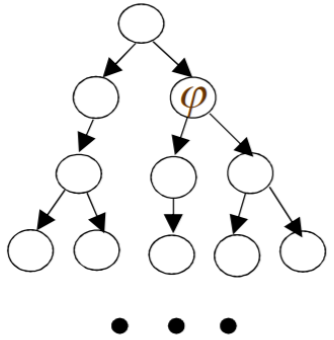
Atomic Propositions:
- Printer is busy
- He is a lawyer

Modal Logic:
- It is raining
- It will rain tomorrow
- It might rain tomorrow

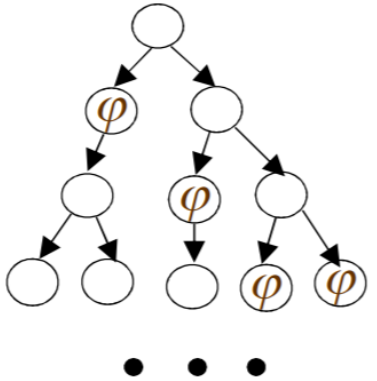
Temporal modalities

Temporal Logic

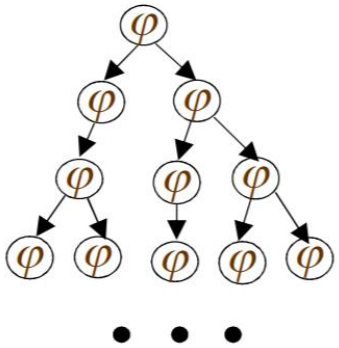
LTL, CTL
- Quantifiers over paths: A, E
- Path specific quantifiers: X, G, F, U, W



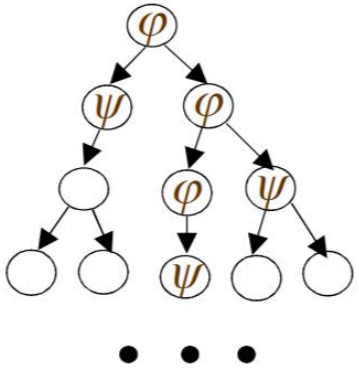
EX φ (exists next)



AF φ (all future)



AG φ (all global)



A[φ U ψ] (all until)

- Semantics of $\forall CTL$

$$\frac{\alpha(s)}{R, s \vDash \alpha} \qquad \frac{R, s \vDash \varphi_1 \quad R, s \vDash \varphi_2}{R, s \vDash \varphi_1 \wedge \varphi_2} \qquad \frac{R, s \vDash \varphi_1 \ \vee \ R, s \vDash \varphi_2}{R, s \vDash \varphi_1 \vee \varphi_2}$$

$$\frac{\forall (s_0, s_1, \ldots).\ s_0 = s \Rightarrow \exists i \geq 0.\ R, s_i \vDash \varphi}{R, s \vDash \mathsf{AF}\varphi}$$

$$\frac{\forall (s_0, s_1, \ldots).\ s_0 = s \Rightarrow \forall i \geq 0.\ R, s_i \vDash \varphi}{R, s \vDash \mathsf{AG}\varphi}$$

$$\frac{\begin{array}{l}\forall (s_0, s_1, \ldots).\ s = s_0 \Rightarrow (\forall i \geq 0.\ R, s_i \vDash \varphi_1) \vee \\ (\exists j \geq 0.\ R, s_j \vDash \varphi_2 \wedge \forall i \in [0, j).\ R, s_i \vDash \varphi_1)\end{array}}{R, s \vDash \mathsf{A}[\varphi_1 \mathsf{W} \varphi_2]}$$

A universal CTL formula:

- it uses only universal temporal connectives (AX, AF, AU, AG) with negation applied to the level of atomic propositions.

➢ Temporal Properties

Safety
Something “bad” will never happen

- AG ¬bad
- e.g., mutual exclusion: no two processes are in their critical section at once
- Safety = if false then there is a finite counter example

Liveness
Something “Good” will always happen

- AG AF good
- e.g., every request is eventually serviced
- Liveness = if false then there is an infinite counterexample

Every universal temporal logic formula can be decomposed into a conjunction of safety and liveness.

➢ Program Analysis

- Derive properties of a program for all possible input values, in order to characterize the set of all possible output values or to find problems in its internal structure

$$\mathrm{M} = (\mathrm{S}, \mathrm{R}, \mathrm{I}) \mid R \in S \times S$$

$$\pi = (s_0, s_1, s_2, \ldots s_t), s_0 \in I$$

$$\xrightarrow{\mathrm{P}}$$

$$\sigma : \mathrm{var} \rightarrow \mathrm{var}$$

$$\pi = \sigma_0, \sigma_1, \sigma_2, \ldots \mathrm{s.t.} \sigma_0 \in I$$

$$Infinit : \forall \sigma \in \Sigma, \exists \sigma', (\sigma, \sigma') \in \mathrm{R}$$

l_0 : x := 3;
l_1 : if (y > 0)
l_2 : C1
else
l_3 : C2

$$\pi = \begin{matrix} x: \\ y: \\ pc: \end{matrix} \begin{bmatrix} 0 & 3 & 3 \\ 1 & 1 & 1 \\ l_0 & l_1 & l_2 \end{bmatrix}$$

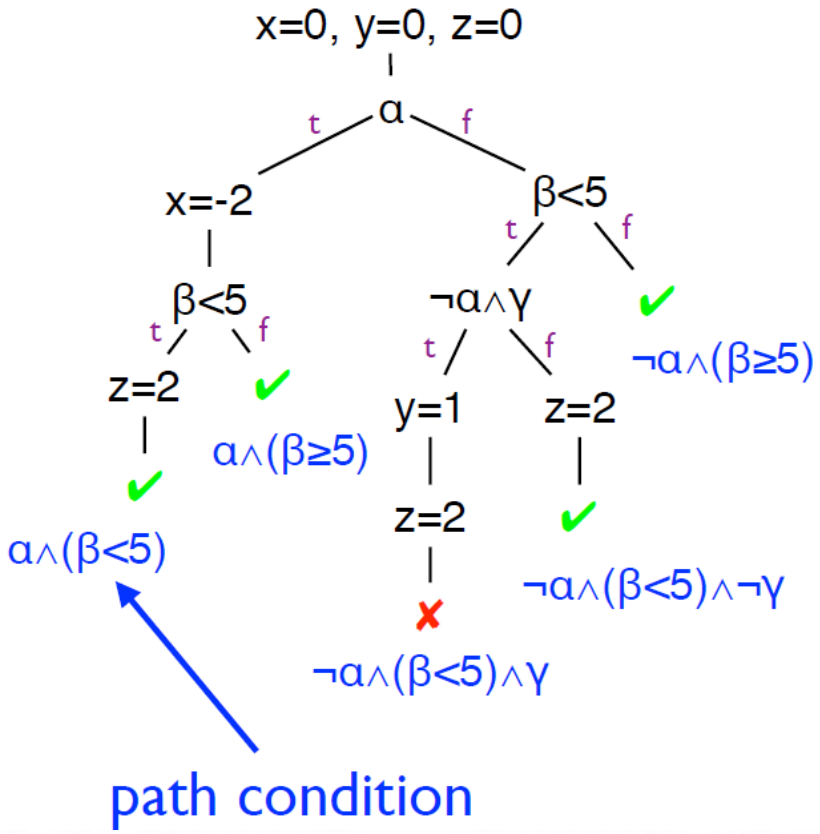
$$\pi' = \begin{matrix} x: \\ y: \\ pc: \end{matrix} \begin{bmatrix} 0 & 3 & 3 \\ -2 & -2 & -2 \\ l_0 & l_1 & l_3 \end{bmatrix}$$

Rank Functions

$$M = \{\mathrm{f}_i \mid \forall (\mathrm{s}, \mathrm{s}') \in \mathrm{R}, \mathrm{f}_i(\mathrm{s}') \prec \mathrm{f}_i(\mathrm{s})\}$$

➢ Symbolic Execution

```
1. int a = α, b = β, c = γ;
2.                  // symbolic
3. int x = 0, y = 0, z = 0;
4. if (a) {
5.   x = -2;
6. }
7. if (b < 5) {
8.   if (!a && c)  { y = 1; }
9.   z = 2;
10.}
11.assert(x+y+z!=3)
```




```
; Variable declarations
(declare-fun x () Int)
(declare-fun y () Int)
(declare-fun z () Int)

; Constraints
(assert (= x -2))
(assert (= y 0))
(assert (= z 2))

; Solve
(assert(not (= (+ (+ x y) z)
3)))
(check-sat)
```

➢ Encoding Temporal Property as Program Analysis

Cook B., Koskinen E., Vardi M. (2011) *Temporal Property Verification as a Program Analysis Task*. CAV 2011.

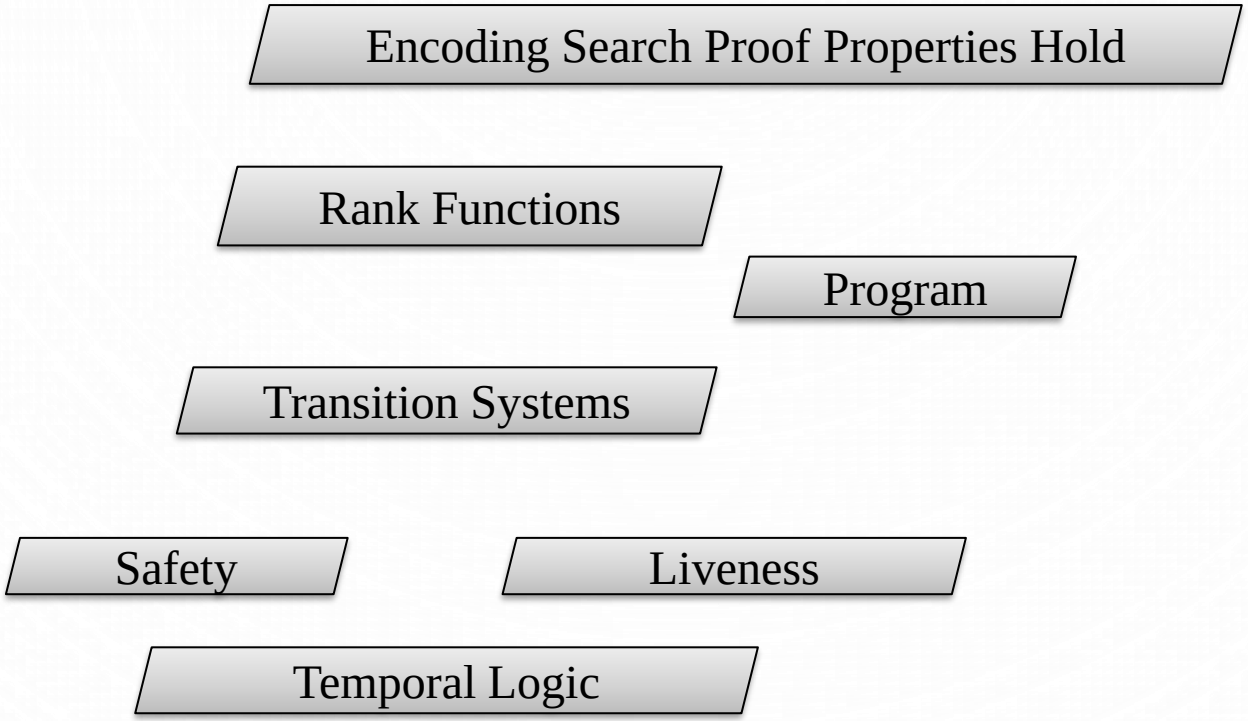
$INV_1 : \forall s, \psi, \mathcal{M}, R.$ if $R, s \not\models \varphi$ then $\mathcal{E}(\langle s, \varphi\rangle, \mathcal{M}, R)$ can return false

$INV_2 : \forall s, \psi, \mathcal{M}, R.\ \mathcal{E}(\langle s, \psi\rangle, \mathcal{M}, R)$ can return true

$$\exists \mathcal{M}.\ \forall s \in I.\ \mathcal{E}_R^{\mathcal{M}}(s, \varphi) \text{ cannot return false} \quad \Rightarrow \quad P \models \varphi$$

```
let rec E(⟨s, ψ⟩, M, R) : bool =
  match(ψ) with
  | α → return α(s)
  | ψ′∧ψ″ →
      if (*) return E(⟨s, ψ′⟩, M, R)
      else return E(⟨s, ψ″⟩, M, R);
  | ψ′∨ψ″ →
      if (E(⟨s, ψ′⟩, M, R)) return true;
      else return E(⟨s, ψ″⟩, M, R);
  | AGψ′ →
       while (true) {
          if (¬ E(⟨s, ψ′⟩, M, R))
            return false;
          if (*) return true;
          s := choose({s′ | R(s, s′)});
       }
  | AFψ′ → local dup := false; local ′s ;
      while (true) {
         if (E(⟨s, ψ′⟩, M, R)) return true;
         if (dup ∧ ¬(∃f ∈ M. f(s) ≺ f(′s)))
            return false;
         if (¬ dup ∧ *) { dup := true; ′s := s; }
         if (*) return true;
         s := choose({s′ | R(s, s′)});
      }
  | A[ψ′Wψ″] →
     while(true) {
        if (¬E(⟨s, ψ′⟩, M, R))
          return E(⟨s, ψ″⟩, M, R);
        if (*) return true;
        s := choose({s′ | R(s, s′)});
     }
```

- Quick Take Away



➢ TEMPORAL LOGIC AND PROGRAM ANALYSIS

July 17 OPLSS 2018

Cyrus Liu
- Advisor, Eric Koskinen

Q&A

Thanks!