CS 412

Day 1

Formal Methods in S/W Eng

* Testing
  - Design
  - Implementation

Requirements

Design / Testing

Test Cases : Input

Expected Output $\rightarrow$ requirements

* Safety / security critical systems

Formal Methods

- Deductive method (Theorem Provers)
  - System : φ
  - Requirements : ψ
  - $\varphi \Rightarrow \psi$
  - valid
- Model Checking
  - system : M (Graph)
  - Requirements : φ (logical formula)
  - M satisfies φ
  - $M \models \varphi$
  - models

$\{p: true, q: true\} \models p \wedge q$

01/28 Day 2

Recap:

Model Checking (FM)

↳ (a) System described graph : M

↳ (b) Requirements : φ

M satisfies φ : $M \models \varphi$ ← Models

Process:

idle

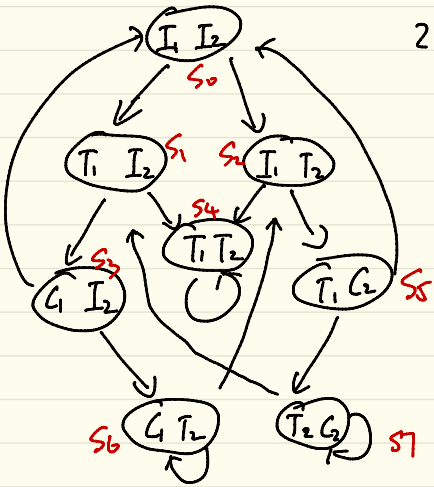
|

trying (wait if some other process is in trying or critical state)

|

critical

2 process



* Asynchronous moves

* Discrete steps/moves

$S = \{s_0, s_1, \cdots s_7\}$

$T = \{(s_0, s_1), (s_0, s_2) \cdots\}$

$L(s_0) = \{I_1, I_2\}$

Graph for system behavior

- Kripke structure

  ↳ $M(S, T, L)$

(a) S is a finite set of states

(b) T is transition relations $\subseteq S \times S$

$(s_1, s_2) \in T$: s_1 has an edge to s_2

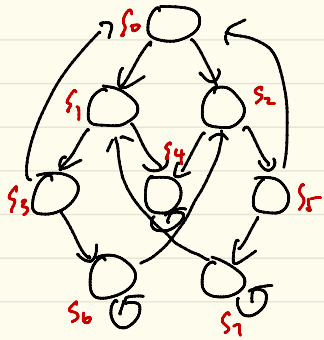
— left-total relation $\forall s \in S \ \exists s' \in S. \ (s, s') \in T$

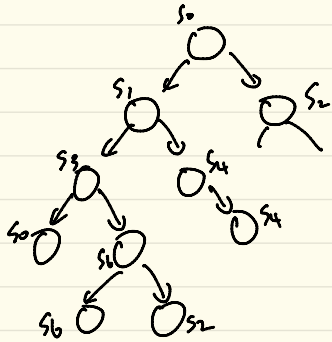
(for every state there is an outgoing edge)

(c) L is labeling function $L : S \to P(AP)$

↳ Powerset of atomic proposition

Computation Tree





$M = (S, S_0, T, L)$

↳ $S_0 \subseteq S$: set of start states

Requirements (properties)

* two processes cannot enter critical state at the same time (safety)
* whenever a process tries to enter critical state, it must always eventually enter critical state (liveness)

Computation Tree (Temporal) Logic

<u>Syntax</u>

conj (and) disj (or) not

$\varphi \rightarrow$ True / False / AP / $\varphi \wedge \varphi$ / $\varphi \vee \varphi$ / $\neg\varphi$ / $AX\varphi$ / $EX\varphi$
/ $AG\varphi$ / $EG\varphi$ / $AF\varphi$ / $EF\varphi$ / $A[\varphi \cup \varphi]$ / $E[\varphi \cup \varphi]$

CTL formula

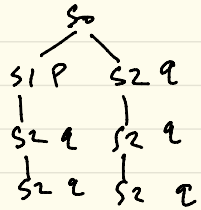
A: along all paths
X: in next state
F: in some future state
E: exists a path
G: globally (in all states)

safety property

$AG(\neg C_1 \vee \neg C_2)$: along all paths, in all states, $\neg C_1$ or $\neg C_2$ is true

$AG(T_1 \Rightarrow AF C_1)$
if T_1 then ~

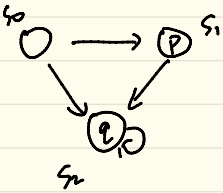
02/02 Week-2-1



Semantics

$[\![\ ]\!]$: input (a) kript structure
(b) CTL formula
output: set of states in ks that satisfies φ

KS M



$L(s_0)=\{\}$ $L(s_1)=\{p\}$ $L(s_2)=\{q\}$

φ: EXP

$[\![\ ]\!](EXP, M) =$

* $[\![EXP]\!]_M : \{s_0\}$

$[\![AXP]\!]_M = \{\}$

$[\![EFP]\!] = \{s_0, s_1\}$

$[\![AFP]\!] = \{s_1\}$

$[\![EGq]\!] = \{s_2\}$

$[\![EG(\neg p)]\!] = \{s_0, s_2\}$

Path: a sequence of states $s_i, s_{i+1}, s_{i+2} \cdots$ is a path iff $(s_i, s_{i+1}) \in T$ $(s_{i+1}, s_{i+2}) \in T \cdots$

— Given a state s, Path(s): set of paths starting from s

— π: denote a path

π: $\underbrace{\pi[0]}_{\text{first state in the path.}}, \pi[1], \pi[2]$

$\pi \in Path(s_1) \Rightarrow \pi[0] = s_1$

Def for $[\![\ ]\!]$: KS. $M = (S, T, L)$

$[\![true]\!] = S$ // set of all states

$[\![false]\!] = \{\}$

$[\![p]\!] : \{s \mid p \in L(s)\}$

$[\![\neg\varphi]\!] = S - [\![\varphi]\!]$

$[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$

$[\![EX\varphi]\!] = \{s \mid \exists \pi \in Path(s).\ \pi[1] \in [\![\varphi]\!]\}$

$[\![AX\varphi]\!] : \{s \mid \forall \pi \in Path(s),\ \pi[1] \in [\![\varphi]\!]\}$

$[\![EF\varphi]\!] : \{s \mid \exists \pi \in Path(s)\ \exists i \geq 0.\ \pi[i] \in [\![\varphi]\!]\}$

$\pi[0]\ \pi[1]\ \pi[2] \cdots \underbrace{\pi[i]}_{\varphi}$

$[\![AF\varphi]\!] = \{ s \mid \forall \pi \in Path(s).\ \exists i \geq 0.\ \pi[i] \in [\![\varphi]\!] \}$

$[\![EG\varphi]\!] = \{ s \mid \exists \pi \in Path(s).\ \forall i \geq 0.\ \pi[i] \in [\![\varphi]\!] \}$

$[\![AG\varphi]\!] = \{ \quad \forall \pi \quad \forall i \quad \}$

$[\![E(\varphi_1 \underset{\text{until}}{\cup} \varphi_2)]\!] = \{ s \mid \exists \pi \in Path(s).\ \exists i \geq 0.\ \pi[i] \in [\![\varphi_2]\!] \wedge \underbrace{\forall j < i.\ \pi[j] \in [\![\varphi_1]\!]}_{\forall j[j<i] \Rightarrow \pi[j] \in [\![\varphi_1]\!]) \ \text{false}} \}$

$[\![A(\varphi_1 \cup \varphi_2]\!] = \{ s \mid \forall \pi \cdots\cdots$

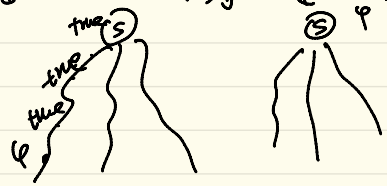
int CTL

$[\![EX(n, \varphi)]\!] = \{ s \mid \exists \pi \in path(s)\ \exists i \leq n.\ \pi[i] \in [\![\varphi]\!] \}$

<u>Equivalences</u>

phi

$[\![E(true \cup \varphi)]\!] = [\![EF\varphi]\!]$

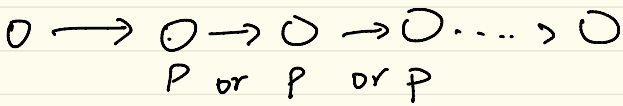


$[\![A(true \cup \varphi)]\!] = [\![AF\varphi]\!]$

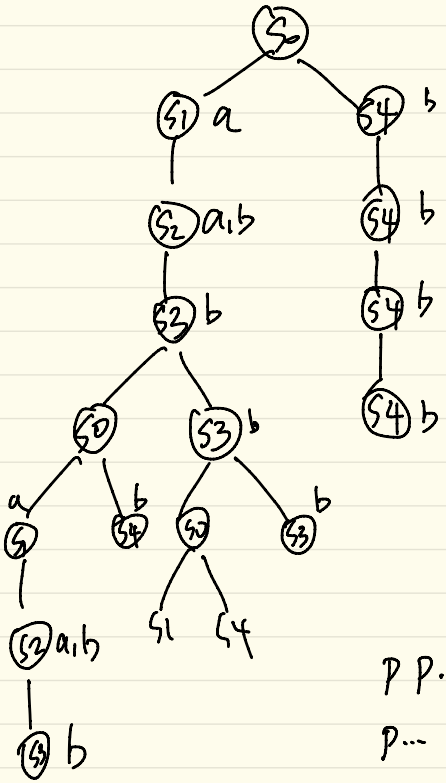
$[\![E(\varphi \cup false)]\!] = [\![false]\!]$

$[\![EF(\varphi_1 \wedge \varphi_2)]\!] \stackrel{?}{=} [\![EF\varphi_1 \wedge EF\varphi_2]\!]$

There exists a path such that 'p' holds in one or more steps from the current state



$\underset{\text{1 step}}{EX}\underset{\text{in some future}}{EF}p \stackrel{?}{=} EFEXp$



$AG(b) = s_4$

$A(b \mathcal{U} \neg b) = s_1, s_2, s_3$

$bb \cdots \neg b$

$AG(AF(b)) = s_0, s_1, s_2, s_3$

s_0, s_1, s_2, s_3, s_4

$EX(EG(b)) =$

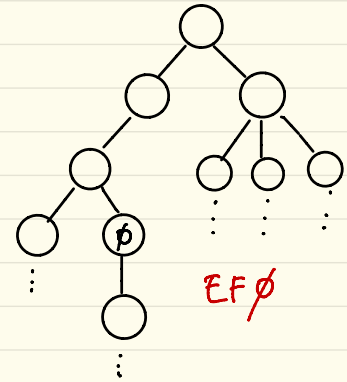
$\{s_2 s_3 s_4\}$

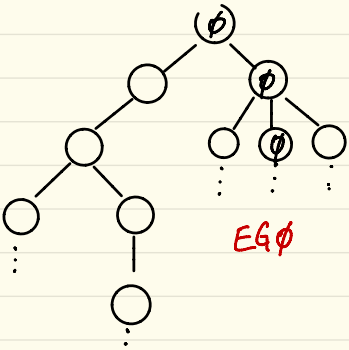
$s_0, s_1, s_2,$

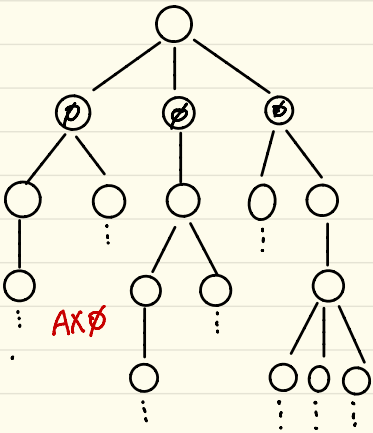
s_3, s_4

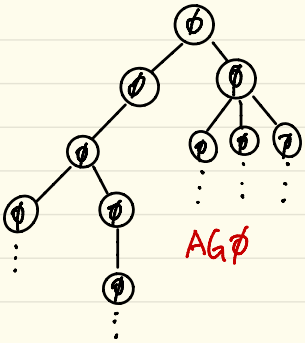
$p\ p \cdots q\ r$

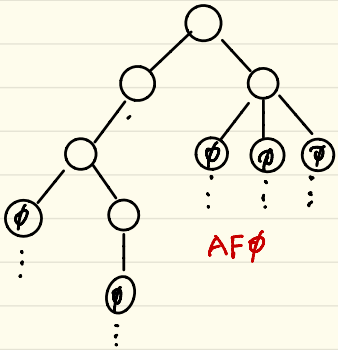
$p \cdots q \cdots r$

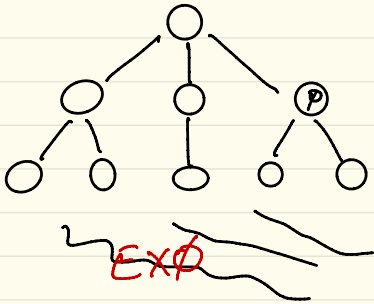

EF∅


EG∅


AX∅


AG∅


AF∅


EX∅

Along all paths r is true only between in p and q. $\neg E((\neg p \, U \, r)) \wedge \neg E((r \, U \, q))$

negation (r is true before ¬p or r is true after ¬q)

$\neg(E(\neg p \, U \, r)) \vee EF(r \wedge AG \neg q)$

E G

There exists a execution sequence such that from every configuration in the sequence

EF — It is possible to satisfy property p EG(EF(p))

AG AXAF AG

Along all paths, whenever p is true, it is followed by a state where p is false, and whenever ¬p is true, it is followed by a state where p is true.

$AG(p \rightarrow AX(AF(\neg p)) \wedge \neg p \rightarrow AX(AF(p)))$

AF = always eventually

EF = eventually

AXAG¬

Along all paths withdraw-money is never true after invalid-login

$AG(\text{invalid-login} \rightarrow AX(AG(\neg \text{withdraw})))$

open →

AG U

Along all execution sequence of an elevator behavior, if the elevator door is open then the door remains open until a request-to-move is sent to the elevator.

$AG(\text{open} \rightarrow A(\text{open} \, U \, \neg \text{open} \wedge \text{request-to-move}))$

AG AF

Whenever proposition request-site-update is true in a state, it is followed in zero or more steps by a state where proposition updating-site is true, which in turn is followed in one or more steps by a state where update-complete is true.

AXAF

$AG(\text{request-site-update} \rightarrow AF(\text{updating-site} \wedge AX(AF(\text{update-complete}))))$

Whenever p occurs, it is followed by q in zero or more steps $AG(p \rightarrow AF(q))$

Whenever p one or more steps $AG(p \rightarrow AXAF(q))$

p and q never true in the same state

There are no two states where p and q are both true.

If p is true in some state, then it is never true along any path from that state

$EF(p \rightarrow AX(AG(\neg p)))$

U

Whenever p is true, it remains true for some time but not forever $AG(p \rightarrow A(p \, U \, AF(\neg p)))$

Whenever p is followed by q, the q is followed by p. $AG((p \rightarrow AX(AG(q))) \rightarrow AX(AF(p)))$

p precedes q along all computation paths. $\neg E((\neg p) \, U \, q)$

After every p, there is some q

AG

Whenever (AG) p is true, it remains (U) true until it becomes false and q becomes true

$$AG(p \to A(p\ U\ ((\neg p) \wedge q)))$$

After p, q is never true $\quad AG(p \to AX\,AG(\neg q))$

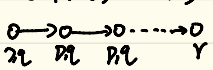
After invalid-login, along all paths, WD is never true $\quad AG(\text{invalid} \to AXAG(\neg WD))$

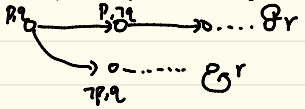
Along all paths, if q is true in some state then p is true in some other state before q

$$\neg E((\neg p)\ U\ q)$$

$$P \to Q \equiv \neg P \vee Q$$

$E((p \wedge q)\ U\ r) \equiv E(p\ U\ r) \wedge E(q\ U\ r)$ ✗





$E(p\ U\ r)$

$E(q\ U\ r)$

$A((p \wedge q)\ U\ r) \equiv A(p\ U\ r) \wedge A(q\ U\ r)$?

$E(p\ U\ (q \vee r)) \equiv E(p\ U\ q) \vee E(p\ U\ r)$ ✓

$A(p\ U\ (q \vee r)) \not\equiv A(p\ U\ q) \vee A(p\ U\ r)$ ✗

$AGEF\varphi \equiv AG\underbrace{E(\text{true}\ U\ \varphi)}_{EF\varphi}$

$EGp \vee EGq \not\equiv EG(p \vee q)$

$EFp \vee EFq \equiv EF(p \vee q)$

$AGp \wedge AGq \equiv AG(p \wedge q)$

$AFp \wedge AFq \not\equiv AF(p \wedge q)$

$\text{true} \equiv AG(p) \to EG(p)$

$[\![\neg EX\varphi]\!] = S - [\![EX\varphi]\!] = S - \{s \mid \exists \pi \in Path(s)\ \pi[i] \in [\![\varphi]\!]\} = \{s \mid \neg(\exists \pi \in Path(s).\ \pi(i) \in [\![\varphi]\!]\}$

$[\![AX\neg\varphi]\!] \leftarrow = \{s \mid \forall \pi \in Path(s).\ \pi[i] \in [\![\neg\varphi]\!]\}$

$[\![\neg AX\varphi]\!] = [\![EX\neg\varphi]\!]$

$[\![\neg AF\varphi]\!] = [\![EG\neg\varphi]\!]$

$[\![\neg AG\varphi]\!] = [\![EF\neg\varphi]\!]$

$[\![E(\text{true}\ U\ \varphi)]\!] = [\![EF\varphi]\!]$

$[\![A(\text{true}\ U\ \varphi)]\!] = [\![AF\varphi]\!]$

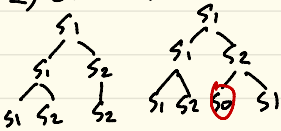
$[\![E(\varphi\ U\ \text{false})]\!] = [\![\text{false}]\!]$

$[\![EF(\varphi_1 \wedge \varphi_2)]\!] \neq [\![EF\varphi_1 \wedge EF\varphi_2]\!]$ ✗

$A \leftrightarrow E$

$G \leftrightarrow F$

$AG(\underbrace{EX(p)}_{s_1\ s_2}) \Rightarrow$ check s_1, s_2 first



✓ ✗

Week 2 – 1

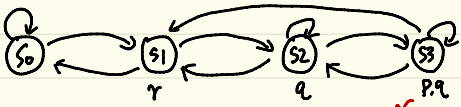
A: along all paths X: in next state F: in future state

E: exists a path G: globally in all states U: until

[[ ]] : input : (a) kripke structure (KS)

(b) CTL formula (φ)

output : set of states in KS that satisfies φ



$s_1 \leftarrow q_r \leftarrow r$

$s_2 \leftarrow q_{p,q} \leftarrow$

[[EXq]] = $s_1 s_2 s_3$

[[AXq]] = { }

☆ [[AFr]] = s_1

[[E(pUq)]] = $s_2 s_3$

[[A(qUr)]] = s_1

[[E(qUr)]] = $s_1 s_2 s_3$

[[A(qUAFr)]] : s_1

$qq \cdots s_1$

[[AGAFq]] = { }

$s_2 s_3$

[[AFAG(qUr)]] = { }

AFr = {s_1}

So can't satisfy

no r

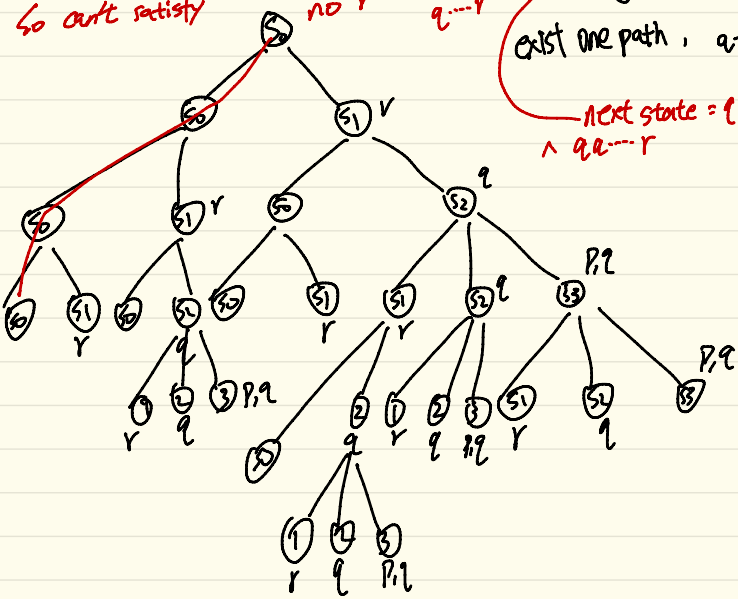
All path

$q \cdots r$

exist one path, $q \to q \to q \to \cdots r$

next state = q

$\wedge\ qq \cdots r$



q

EXq

AXq

AFr

E(pUq)

A(qUr)

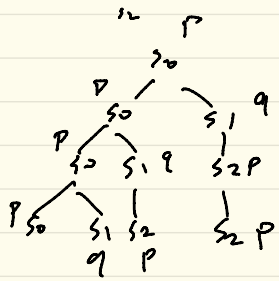
E(qUr)

A(qUAFr)

AGAFq

AFAG(qUr)

AG	s_2
AF	s_0, s_1, s_2
AG/AF	s_0, s_1, s_2
AF/AG	s_1, s_2



equivalences

$$[\![\neg EX\varphi]\!] = S - [\![EX\varphi]\!]$$
$$= S - \{\neg(\exists \pi \in Path(s).\ \pi[1] \in [\![\varphi]\!]\}$$
$$= \{s \mid \neg(\exists \pi \in Path(s).\ \pi[1] \in [\![\varphi]\!]\}$$
$$= \{s \mid \forall \pi \in Path(s).\ \pi[1] \in [\![\neg\varphi]\!]\}$$
$$= [\![AX\neg\varphi]\!]$$

$[\![\neg AX\varphi]\!] = [\![EX\neg\varphi]\!]$

$[\![\neg AF\varphi]\!] = [\![EG\neg\varphi]\!]$

$[\![\neg AG\varphi]\!] = [\![EF\neg\varphi]\!]$

$[\![\neg A(\varphi_1 \mathcal{U} \varphi_2)]\!] = ?$

$[\![\neg E(\varphi_1 \mathcal{U} \varphi_2)]\!] = ?$

Translating English properties $\rightarrow$ CTL

1. $\underbrace{\text{Whenever}}_{AG}$ the system enters faulty state, $\underbrace{\text{it is possible}}_{E}$ $\underbrace{\text{to reach}}_{F}$ a recovery state

$AG(\text{fault} \Rightarrow EF\ \text{recover})$ if faulty state then EF recover

2. It is possible to satisfy p and
It is possible to satisfy q
but p and q never gets satisfied at the same time

$$EFp \wedge EFq \wedge \underbrace{\neg EF(p \wedge q)}_{AG(\neg p \vee \neg q)}$$

Week-3-1

Equivalence

1. $EGp \vee EGq \overset{?}{\equiv} EG(p \vee q)$ No

2. $EFp \vee EFq \overset{?}{\equiv} EF(p \vee q)$

$s \in [\![EFp \vee EFq]\!]$ iff $s \in [\![EFp]\!] \cup [\![EFq]\!]$

iff $s \in [\![EFp]\!]$ or $s \in [\![EFq]\!]$

iff $\exists \pi \in Path(s).\ \exists i \geq 0.\ \pi[i] \in [\![p]\!]$

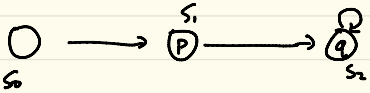
or $\exists \pi' \in Path(s).\ \exists j \geq 0.\ \pi[i] \in [\![q]\!]$

iff $\exists \pi \in Path(s).\ \exists i \geq 0.\ \pi[i] \in [\![p]\!]$ or $\pi[i] \in [\![q]\!]$

iff $s \in [\![EF(p \vee q)]\!]$

3. $AGp \wedge AGq \overset{?}{\equiv} AG(p \wedge q)$ Yes

4. $AFp \wedge AFq \overset{?}{\equiv} AF(p \wedge q)$ NO



$s_0 \in [\![AFp \wedge AFq]\!]$

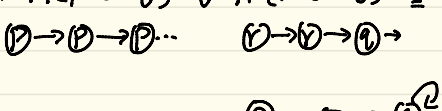
$s_0 \in [\![AF(p \wedge q)]\!]$

5. true $\overset{?}{\equiv} \underbrace{AG(p)}_{\forall i} \Rightarrow EG(p)$ Yes.

$\forall \pi$

$\Rightarrow \exists \pi\ \forall i$

6. $A(p \mathcal{U} q) \vee A(r \mathcal{U} q) \overset{?}{\equiv} A((p \vee r) \mathcal{U} q)$. NO



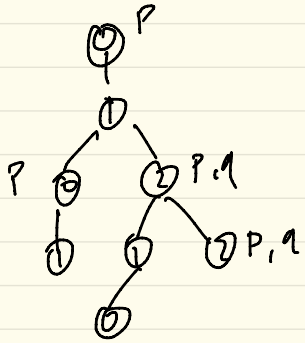
$s_0 \notin [\![LHS]\!]$

$s_0 \in [\![RHS]\!]$

7. $EXEFP \overset{?}{\equiv} EFEXp$

$s \in [\![EXEFp]\!]$ iff $\exists \pi \in Path(s) \,.\, \pi[1] \in [\![EFp]\!]$

$\exists \pi \in Path(s) \;\; [\, \exists \pi' \in Path(\pi[1]) \;\; \exists i \geq 0 \,.\, \pi'[i] \in [\![p]\!] \,]$



$EX(p) = \{1, 2\}$

$AG\,EX(p) =$

Week-5. Monotone Functions , Fairness

Fairness. (will only use Fairness running in NuSMV)
<propositional logic formula>

<u>Algorithm</u> Given KS, φ, compute set of states in KS that belongs to semantics of φ.

$\boxed{\text{Background}}$ S: set of elements

P(S): powerset of S Ex. $S=\{a,b,c\}$ Powerset: $\{\emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}\}$

* <u>Monotone functions</u> (monotonically non-decreasing function)

$f: \underbrace{P(S)}_{\text{input domain}} \rightarrow \underbrace{P(S)}_{\text{output domain}}$ is a monotone function if $\forall X, Y \;\; X \subseteq Y \Rightarrow f(X) \subseteq f(Y)$
[order preserving function]

Example:

$S = \{s_0, s_1, s_2\}$ $f(x) = x \cup \{s_0\}$

x	$f(x)$	
$\{\}$	$\{s_0\}$	
$\{s_0\}$	$\{s_0\}$	← fp (least)
$\{s_1\}$	$\{s_1, s_0\}$	
$\{s_2\}$	$\{s_2, s_0\}$	
$\{s_1, s_2\}$	$\{s_1, s_2, s_0\}$	
$\{s_0, s_2\}$	$\{s_0, s_2\}$	fp
$\{s_0, s_1\}$	$\{s_0, s_1\}$	fp
$\{s_0, s_1, s_2\}$	$\{s_0, s_1, s_2\}$	fp (greatest)

* fixed point. of a monotone function f is x if $f(x)=x$

* least fixed point
* greatest fixed point

* Tarski and Knaster theorem (*)

Given a $f: P(S) \to P(S)$ such that $|S| = n$ and f is monotonically non-decreasing

then: (a) $f^n(\{\})$ is the least fp of f

(b) $f^n(S)$ is the greatest fp of f

Proof: $\{\}$ is a subset of any other set

$\Rightarrow \{\} \subseteq f(\{\})$ ①

$\Rightarrow f(\{\}) \subseteq f(f(\{\}))$ ② because of monotomary of f

$\Rightarrow f^2(\{\}) \subseteq f^3(\{\})$ ③

$$\therefore \underbrace{\{\} \subseteq f(\{\}) \subseteq f^2(\{\}) \subseteq f^3(\{\}) \cdots \subseteq f^n(\{\})}_{n=|S|} \subseteq f^{n+1}(\{\})$$

Claim: $f^n(\{\}) = f^{n+1}(\{\})$

$= f(f^n(\{\}))$

To show: $f^n(\{\})$ is a fixed point

Assume $Z = f(Z)$ and $\boxed{Z \subset f^n(\{\})}$

$\{\} \subseteq Z$ $\quad$ Z & $f^n(\{\})$ are unrelated

$\Rightarrow f(\{\}) \subseteq f(Z)$ by monotonicity of f

$\Rightarrow f(\{\}) \subseteq Z$ because $Z = f(Z)$

$f^2(\{\}) \subseteq f(Z)$ by monotoning of f

$f^2(\{\}) \subseteq Z$ $\quad$ $Z = f(Z)$

$\Rightarrow f^n(\{\}) \subseteq Z$ $\quad$ contradiction

$\therefore$ There is no fixed point of f that is a strict subset of $f^n(\{\})$

proof (b) $S \supseteq f(S) \supseteq f^2(S) \cdots \supseteq f^n(S)$

$= f^{n+1}(S)$

KS: (S, T, L)

Defined over $P(S)$ where S is set of states in some KS

$R_\forall(Z) = \{s \mid \forall s \to s'.\ s' \in Z\}$

$= \{s \mid \forall s'.\ (s, s') \in T \Rightarrow s' \in Z\}$

$\Rightarrow R_\forall(Z)$ is the set of states whose all next states belong to Z.

* is $R_\forall$ a monotonically non-decreasing function? Yes

claim: $X \subseteq Y \Rightarrow R_\forall(X) \subseteq R_\forall(Y)$

assume: $X \subseteq Y \wedge R_\forall(X) \supset R_\forall(Y)$

$\exists s.\ s \in R_\forall(X)$ and $s \notin R_\forall(Y)$

CTL formula

set of states that satisfy φ

* $I_\varphi(Z) = [\![\varphi]\!] \cup R_\forall(Z)$ is this monotonic?

claim: $X \subseteq Y \Rightarrow I_\varphi(X) \subseteq I_\varphi(Y)$

$[\![\varphi]\!] \cup R_\forall(X) \subseteq [\![\varphi]\!] \cup R_\forall(Y)$

$R_\forall(X) \subseteq R_\forall(Y) \leftarrow$ true

claim: $[\![AF\varphi]\!]$ is the least fixed point of I_φ

$\equiv [\![AF\varphi]\!] = I_\varphi([\![AF\varphi]\!])$ and it is the subset of all fp of I_φ

Algo: $[\![AF\varphi]\!]$.

$I_\varphi(I_\varphi(\cdots\cdots(I_\varphi(\{\})) \cdots)$: T-k themem

claim (a): $[\![AF\varphi]\!] = I_\varphi([\![AF\varphi]\!])$

$[\![AF\varphi]\!] = \{s \mid \forall \pi \in Path(s).\ \exists i \geq 0.\ \pi[i] \in [\![\varphi]\!]\}$

$= \{s \mid \forall \pi \in Path(s).\ \pi[0] \in [\![\varphi]\!] \vee \exists i \geq 1.\ \pi[i] \in [\![\varphi]\!]\}$

$= \{s \mid \forall \pi \in Path(s).\ \pi[0] \in [\![\varphi]\!]\} \cup \{s \mid \forall \pi \in Path(s).\ \exists i \geq 1.\ \pi[i] \in [\![\varphi]\!]\}$

$= [\![\varphi]\!] \cup \{s \mid \forall \pi \in Path(s)\ \ \pi[1] \in [\![AF\varphi]\!]\}$

$= [\![\varphi]\!] \cup R_\forall([\![AF\varphi]\!])$

$= I_\varphi([\![AF\varphi]\!])$ $\therefore$ $[\![AF\varphi]\!]$ is a fp of I_φ

3/9 After EXAM 1

Monotonically non-decreasing function

$X \subseteq Y \Rightarrow f(X) \subseteq f(Y)$

- fp : $f(Z) = Z$
- lfp : $f^n(\{\})$ gfp : $f^n(S)$

$R_\forall(Z) = \{s \mid \forall s'.\ s \to s',\ s' \in Z\}$ monotone

$I_\varphi(Z) = [\![\varphi]\!] \cup R_\forall(Z)$ monotone

CTL set of states that satisfy φ

claim: $[\![AF\varphi]\!]$ is a lfp of I_φ

↳ claim1: $[\![AF\varphi]\!]$ is a fp of I_φ

$[\![AF\varphi]\!] = \cdots\cdots$

$= I_\varphi([\![AF\varphi]\!])$ } proved

↳ claim 2: $[\![AF\varphi]\!]$ is the least among the fp

Assume $Z = I_\varphi(Z)$ such $Z \subset [\![AF\varphi]\!]$

$\therefore \exists s_0.\ s_0 \in [\![AF\varphi]\!]$ and $s_0 \notin Z$

$\equiv \exists s_0.\ s_0 \in I_\varphi([\![AF\varphi]\!])$ and $s_0 \notin I_\varphi(Z)$

$\equiv \exists s_0.\ (s_0 \in [\![\varphi]\!]$ or $s_0 \in R_\forall([\![AF\varphi]\!])$ and $(s_0 \notin [\![\varphi]\!]$ and $s_0 \notin R_\forall(Z))$

$\equiv \exists s_0.\ (s_0 \notin [\![\varphi]\!]$ and $s_0 \in R_\forall([\![AF\varphi]\!])$ and $s_0 \notin R_\forall(Z)$

$\equiv \exists s_0.\ (s_0 \notin [\![\varphi]\!]$ and $\exists s_1.\ s_0 \to s_1\ (s_1 \in [\![AF\varphi]\!]$ and $s_1 \notin Z))$

$\therefore$ Inductively construct a path st.

$s_0 \to s_1 \to s_2 \to \cdots$

$s_0 \notin [\![\varphi]\!],\ s_1 \notin [\![\varphi]\!],\ s_2 \notin [\![\varphi]\!], \cdots$

But $s_0 \in [\![AF\varphi]\!]$ contradiction

$\therefore$ The assumption that $Z = I_\varphi(Z)$ and $Z \subset [\![AF\varphi]\!]$ is invalid

$[\![AF\varphi]\!]$ is the lfp of I_φ

$I_\varphi(Z) = [\![\varphi]\!] \cup R_\forall(Z)$

$[\![AF\varphi]\!] = I_\varphi^n(\{\})$ (T-K the orem) init

$I_\varphi(\{\}) = [\![\varphi]\!]$ mark states with $AF\varphi$ if they are already marked with φ

$I_\varphi^2(\{\}) = [\![\varphi]\!] \cup R_\forall(I_\varphi(\{\}))$ mark states with $AF\varphi$ if all their next states

$= [\![\varphi]\!] \cup R_\forall([\![\varphi]\!])$ are already marked with $AF\varphi$

$I_\varphi^3(\{\}) = [\![\varphi]\!] \cup R_\forall(I_\varphi^2(\{\}))$

$= [\![\varphi]\!] \cup R_\forall([\![\varphi]\!] \cup R_\forall([\![\varphi]\!])$ mark states with $AF\varphi$ if all their next states

$= [\![\varphi]\!] \cup R_\forall([\![\varphi]\!]) \cup R_\forall(R_\forall([\![\varphi]\!]))$ are already marked with $AF\varphi$

mark states with $AF\varphi$ if they are marked w/ φ

Repeat : mark states w/ $AF\varphi$ if all their next states are marked w/ $AF\varphi$ $O(n)$

until no change in marking

- $R_\exists(Z) = \{s \mid \exists s'\ s \to s'\ s' \in Z\}$: monotone
- $J_{\varphi_1\varphi_2}(Z) = [\![\varphi_2]\!] \cup ([\![\varphi_1]\!] \cap R_\exists(Z))$: monotone
- $[\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$ is a lfp of $J_{\varphi_1\varphi_2}$ <u>claim</u>

claim1: $[\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$ is a fp of $J_{\varphi_1\varphi_2}$

$[\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$ = logical def

$= [\![\varphi_2]\!] \cup ([\![\varphi_1]\!] \cap R_\exists([\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]))$

$= J_{\varphi_1\varphi_2}([\![E(\varphi_1 \mathcal{U} \varphi_2)]\!])$

claim2. $[\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$ is the lfp of $J_{\varphi_1\varphi_2}$

assume $Z = J_{\varphi_1\varphi_2}(Z)$ $Z \subset [\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$

$\exists s_0.\ s_0 \in [\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$ and $s_0 \notin Z$

$\equiv \exists s_0.\ s_0 \in J_{\varphi_1\varphi_2}([\![E(\varphi_1 \mathcal{U} \varphi_2)]\!])$ and $s_0 \notin J_{\varphi_1\varphi_2}(Z)$

$\equiv \exists s_0.\ [s_0 \in [\![\varphi_2]\!]$ or $(s_0 \in [\![\varphi_1]\!]$ and $s_0 \in R_\exists([\![E(\varphi_1 \mathcal{U} \varphi_2)]\!])$ and $[s_0 \notin [\![\varphi_2]\!]$ and $(s_0 \notin [\![\varphi_1]\!]$ or $s_0 \notin R_\exists(Z))]$

$\exists s_0\ [s_0 \notin [\![\varphi_2]\!]$ and $s_0 \in [\![\varphi_1]\!]$ and $s_0 \in R_\exists([\![E(\varphi_1 \mathcal{U} \varphi_2)]\!])$ and $\underline{s_0 \notin R_\exists(Z)}$

none of the next states of s_0 belong to Z

$\exists s_0 [s_0 \notin [\![\varphi_2]\!]$ and $s_0 \in [\![\varphi_1]\!]$ and $\exists s'. s \to s'. s' \in [\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$ and $\forall s'. s \to s'. s' \notin Z$

Inductive construct a computation tree starting from s_0, such that in that computation tree

(a) φ_2 is never true in the paths where φ_1 is true all the time

(b) φ_1 is false and φ_2 is false

but $s_0 \in [\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$ Contradiction

$[\![E(\varphi_1 \mathcal{U} \varphi_2)]\!] = J^n_{\varphi_1 \varphi_2}(\{\})$

$J_{\varphi_1 \varphi_2}(\{\}) = [\![\varphi_2]\!]$

$J^2_{\varphi_1 \varphi_2}(\{\}) = [\![\varphi_2]\!] \cup ([\![\varphi_1]\!] \cap Pre_\exists([\![\varphi_2]\!]))$

mark states with $E(\varphi_1 \mathcal{U} \varphi_2)$ if they are already marked by φ_2

repeat

mark states with $E(\varphi_1 \mathcal{U} \varphi_2)$ if they are already marked with φ_1 and have at least one next state marked of $E(\varphi_1 \mathcal{U} \varphi_2)$

until no change in marking

Model Checking Algorithm

$J_{a, AFb}(Z)$

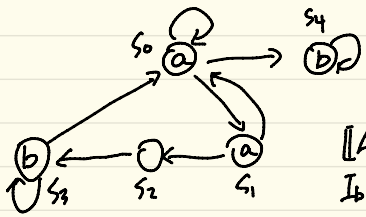
Recap

$R_\exists(Z) = \{s \mid \exists s'.\ s \to s',\ s' \in Z\}$

$R_\forall(Z) = \{s \mid \forall s'.\ s \to s',\ s' \in Z\}$

$[\![AF\varphi]\!] \leftarrow I_\varphi(Z) = [\![\varphi]\!] \cup R_\forall(Z)$

$[\![E(\varphi_1 \mathcal{U} \varphi_2)]\!] \leftarrow J_{\varphi_1,\varphi_2}(Z) = [\![\varphi_2]\!] \cup ([\![\varphi_1]\!] \cap R_\exists(Z))$



$[\![AFb]\!]$: lfp of I_b

$I_b(\{\}) : [\![b]\!] = \{s_3, s_4\}$

$I_b(I_b(\{\})) = [\![b]\!] \cup R_\forall(I_b(\{\}))$

$= \{s_3, s_4\} \cup R_\forall(\{s_3, s_4\})$

$= \{s_3, s_4\} \cup \{s_4, s_2\}$

$= \{s_2, s_3, s_4\}$

$I_b(I_b(I_b(\{\})))$

$= [\![b]\!] \cup R_\forall(\{s_2, s_3, s_4\})$

$= \{s_3, s_4\} \cup \{s_2, s_4\}$

$= \{s_2, s_3, s_4\}$

lfp 의 I_b

$[\![E(a \mathcal{U} AFb)]\!]$ - lfp of $J_{a,AFb}$ $\quad J_{a,AFb}(Z) = [\![AFb]\!] \cup ([\![a]\!] \cap R_\exists(Z))$

$J_{a,AFb}(\{\}) = [\![AFb]\!] \cup ([\![a]\!] \cap R_\exists(\{\}))$

$= \{s_2, s_3, s_4\} \cup (\{s_0, s_1\} \cap \{\}) = \{s_2, s_3, s_4\}$

$J^2_{a,AFb}(\{\}) = [\![AFb]\!] \cup ([\![a]\!] \cap R_\exists(\{s_2, s_3, s_4\}))$

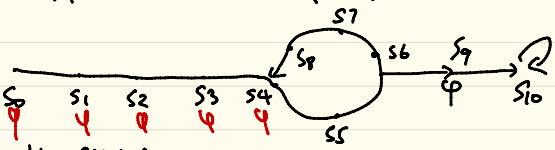
$= \{s_2, s_3, s_4\} \cup (\{s_0, s_1\} \cap \{s_0, s_1, s_2, s_3, s_4\})$

$= \{s_2, s_3, s_4\} \cup \{s_0, s_1\} = \{s_0, s_1, s_2, s_3, s_4\}$

$J^3_{a,AFb}(\{\})$ = not necessary

Instead of AF, find the semantics of EG

$[\![EG\varphi]\!]$



marking (M, φ) $\quad$ non-trivial

$O(|S|+|T|)$ Find the ν SCC in M over states that are marked with φ

$O(|S|+|T|)$ Mark states which are already marked w/ φ with $EG\varphi$ if they can reach any SCC

Model checking Algo $(\neg, \wedge, EX, AF, EU)$

Marking Algo: Given a KS M and a CTL formula φ, the algo marks the set of states in M with φ iff those states belong to the semantics of φ

mark $[\![\varphi]\!]$ with φ

marking(M, φ) $\{\neg, \wedge, EX, AF, EU\}$

$|S|$ if $\varphi \equiv$ true then mark all states with true

$|S|$ $\varphi \equiv p$ then mark states with p if they are labeled with p

$\varphi \equiv \neg\psi$ then marking (M, ψ)

$|S|$ mark states with φ if they are not mark with ψ

$\varphi \equiv \varphi_1 \wedge \varphi_2$ then marking(M, φ_1) ; marking(M, φ_2)

$|S|$ mark states with φ if they are marked already w/ φ_1 and φ_2

$\varphi \equiv EX\psi$ then marking (M, ψ)

$|T|$ mark states w/ φ if at least one of their next states are marked w/ ψ

\# of transition

$\varphi \equiv AF\psi$ then marking (M, ψ)

mark states with φ if they are already marked w/ ψ

$O(|S||T|)$ $O(|S|)$ – Repeat

$O(T)$ mark states with φ if all their next states are marked w/ φ.

Until there is no change in marking

$\varphi \equiv E(\varphi_1 U \varphi_2)$ then marking(M, φ_1) , marking (M, φ_2)

mark states with φ if they are already marked with φ_2

$O(|S|+|T|)$ BFS Repeat

mark states with φ if they are already marked w/ φ_1 and has at least one next state that is marked w/ φ

Until there is no change in marking

$O(|S||T||\varphi|)$

$\hookrightarrow$ # of nesting in φ

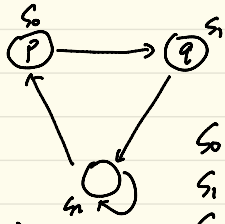
WEEK — 8 — 1

03/16

State-space explosion — McMillan — SMV

sets		logic (Propositional)
$\{s_0\}$		
$\{s_0, s_1\}$	—	$(p \wedge \neg q) \vee (\neg p \wedge q)$
$\{s_0, s_2\}$	—	$(p \wedge \neg q) \vee (\neg p \wedge \neg q) \equiv \neg q$
$\{s_0, s_1, s_2\}$	—	$(p \wedge \neg q) \vee (\neg p \wedge \neg q) \vee (\neg p \wedge q) \equiv \neg(p \wedge q)$
$\{\{s_0, s_1\}$	....	$[(p \wedge \neg q) \wedge (\neg p' \wedge q')]$
$\{s_1, s_2\}$	....	$\vee[(\neg p \wedge q) \wedge (\neg p' \wedge \neg q')]$
$\{s_2, s_2\}$	....	$\vee[(\neg p \wedge \neg q) \wedge (\neg p' \wedge \neg q')]$
$\{s_2, s_0\}\}$	....	$\vee[(\neg p \wedge \neg q) \wedge (p' \wedge \neg q')]$

T



$s_0 : p \wedge \neg q$

$s_1 : q \wedge \neg p$

$s_2 : \neg p \wedge \neg q$

$R_\exists(Z) = \{s \mid \exists s'.\ \underbrace{s \rightarrow s'}_{\text{transition}}.\ \underbrace{s' \in Z}_{\text{property of } s'}\}$

property of s' → q is true

$\exists x.(x \wedge y) =$
$(\text{true} \wedge y) \vee (\text{false} \wedge y)$
$= y$

$\exists x.(x \vee y) =$
$(\text{true} \vee y) \vee (\text{false} \vee y)$
$\equiv \text{true}$

* Find the transition relations where q' holds

$T \wedge q' = [(p \wedge \neg q) \wedge (\neg p' \wedge q')] \vee \text{false} \vee \text{false} \vee \text{false}$

$\exists p', q'.(T \wedge q') = \exists p', q'.[(p \wedge \neg q) \wedge (\neg p' \wedge \neg q')]$

$= (p \wedge \neg q)$

$R_\exists(Z)$ where Z is representing the set of states where $\neg q$ is true

Week - 8 - 2

$R_\exists(Z)$ where Z is representing the set of states where $\neg q$ is true

$$\exists p', q' \ (T \wedge \neg q') = \exists p', q' \ (b_2 \vee b_3 \vee b_4)$$
$$= (\neg p \wedge q) \vee (\neg p \wedge \neg q)$$

$R_\forall(Z) = \{ s \mid \forall s'. s \to s'. s' \in Z \} \longrightarrow$ using $R_\exists$

Semantics of CTL : C part : $[\![\ ]\!]$: input $M, \varphi \to$ set of states in M that satisfy φ)

propotition Logic

$[\![\ ]\!]$ input M , input $\varphi \to$ output the PL formula representation of set of states in M that satisfy φ.

↑ transition represent

$[\![true]\!] = true$ $\quad [\![false]\!] = false$, $[\![p]\!] = p$

$[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \wedge [\![\varphi_2]\!]$ $\left([\![\varphi_1]\!] \cap [\![\varphi_2]\!] \right)$, $[\![\neg \varphi]\!] = \neg [\![\varphi]\!]$ $\quad (S - [\![\varphi]\!])$

$[\![EX\varphi]\!] = R_\exists([\![\varphi]\!]) = \exists \bar{p}' (T \wedge [\![\varphi]\!]_{\bar{p} \to \bar{p}'})$ (pre$_\exists$ image computation) $\{ s \mid \exists s'. s \to s'. s' \in [\![\varphi]\!] \}$

$[\![AX\varphi]\!] = [\![\neg EX \neg \varphi]\!]$

Monotonicity: $X \subseteq Y \Rightarrow f(x) \subseteq f(y)$

$(B_X \Rightarrow B_Y)$ $\quad (B_{f(x)} \Rightarrow B_{f(y)})$

$X : a_1 a_2 \cdots a_k$ $\qquad Y : a_1 a_2 \cdots a_k b_1 \cdots b_m$

$B_X : B_{a_1} \vee B_{a_2} \vee \cdots \vee B_{a_k}$ $\qquad B_Y : B_{a_1} \vee B_{a_2} \cdots B_{a_k} \cdots \vee B_{b_1}$

$X = Y : B_X \Leftrightarrow B_Y$

$R_\exists(Z)$ $\quad J_{\varphi_1 \varphi_2}(Z) = [\![\varphi_2]\!] \vee ([\![\varphi_1]\!] \wedge R_\exists(Z))$

* $[\![E(\varphi_1 \mathcal{U} \varphi_2)]\!]$ is the lfp of $J_{\varphi_1 \varphi_2}$

$J_{\varphi_1 \varphi_2}(false)$, $J^2_{\varphi_1 \varphi_2}(false)$ ....

$J^i_{\varphi_1 \varphi_2}(false) \Leftrightarrow J^{i+1}_{\varphi_1 \varphi_2}(false)$

– $L\varphi(Z) = [\![\varphi]\!] \wedge R_\exists(Z)$ $\qquad L\varphi(true), L^2\varphi(true) \cdots$

* $[\![EG(\varphi)]\!]$ is the gfp of $L\varphi$ $\qquad L^i\varphi(true) \Leftrightarrow L^{i+1}(true)$

Adequate set $\neg, \wedge, EX, EU, EG$.

$[\![EX\varphi]\!] = R_\exists([\![\varphi]\!])$

$[\![E(\varphi_1 \, U \, \varphi_2)]\!] = J_{\varphi_1}^{\wedge}\varphi_2$ (false) lfp of J_{φ_1,φ_2} & $n = |S|$

$[\![EG\varphi]\!] = L_\varphi^{\wedge}$ (true) gfp of L_φ

Week - 9 - 1

<u>recap</u> set: disjunction of logical formula representing each element in the set.
$(\cup : \vee,\ \cap : \wedge,\ - : \neg)$

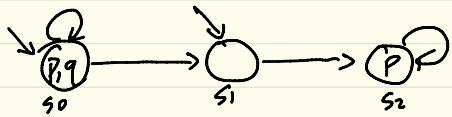
a state in KS : conjunction of literals that hold true in a state

a trans in KS : conjunction of source state representation with destination state representation (w/ primed literals)

$[\![EG\varphi]\!]$ is gfp of L_φ $\qquad L_\varphi(Z) = [\![\varphi]\!] \wedge R_\exists(Z)$

$[\![E(\varphi_1 \cup \varphi_2)]\!]$ is lfp of $J_{\varphi_1\varphi_2}$ $\qquad J_{\varphi_1\varphi_2}(Z) = [\![\varphi_2]\!] \vee ([\![\varphi_1]\!] \wedge R_\exists(Z))$

EX:



$T: [(p \wedge q \wedge p' \wedge q') \vee (p \wedge q \wedge \neg p' \wedge \neg q') \vee (\neg p \wedge \neg q \wedge p' \wedge \neg q') \vee (p \wedge \neg q \wedge p' \wedge \neg q')]$

$[\![AFAGp]\!] = [\![\neg\neg AFAGp]\!] = [\![\neg EGEF\neg p]\!] = \neg[\![EGEF\neg p]\!]$

$\Rightarrow [\![EF\neg p]\!] \equiv \varphi \longrightarrow$ compute $[\![EG\varphi]\!]$

$[\![EG\varphi]\!]$ (where $\varphi : EF\neg\varphi$) $\quad p \wedge q$

$[\![AFAGp]\!]$ is $\neg(p \wedge q) = \neg p \vee \neg q$

$L_\varphi(true) = [\![\varphi]\!] \wedge \underbrace{R_\exists(true)}_{true}$

$= [\![\varphi]\!]$

$[\![EF\neg p]\!]$: lfp of $J_{true,\neg p}$: $\neg p \vee q$

$[\![E(true\ U\ \neg p)]\!]$

$J_{true,\neg p}(false) = [\![\neg p]\!] \vee (true \wedge R_\exists(false))$

$= [\![\neg p]\!]$

$J^2_{true,\neg p}(false) = [\![\neg p]\!] \vee (true \wedge R_\exists(\neg p))$

$= \neg p \vee (p \wedge q) = \neg p \vee q$

$J^3_{true,\neg p}(false) = [\![\neg p]\!] \vee (true \wedge R_\exists(\neg p \vee q))$

$= \neg p \vee (p \wedge q) = \neg p \vee q$

$L_\varphi(true) = \neg p \vee q$

$L^2_\varphi(true) = [\![\varphi]\!] \wedge R_\exists(\neg p \vee q)$

$= (\neg p \vee q) \wedge (p \wedge q) = p \wedge q$

$L^3_\varphi(true) = [\![\varphi]\!] \wedge R_\exists(p \wedge q)$

$= (\neg p \vee q) \wedge (p \wedge q) = p \wedge q$

MC: using logical formula manip

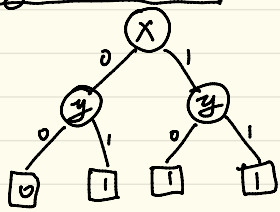
Reduce ← ordered Binary Decision Diagram (ROBDD)

Truth Table

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

$\Rightarrow$

Binary Decision Tree



$(\neg x \wedge \neg y) \vee (\neg x \wedge y) \vee (x \wedge \neg y) \vee (x \wedge y)$

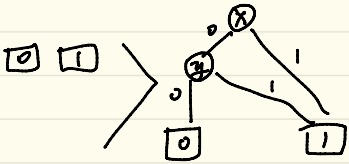
$f(a_1, a_2, \cdots, a_n)$

$= (a_1 \wedge \underbrace{f(1, a_2, \cdots, a_n)}_{a_1 \text{ is true}}) \vee (\neg a_1 \wedge \underbrace{f(0, a_2, \cdots a_n)}_{a_1 \text{ is false}})$

positive cofactor of a_1 negative ....

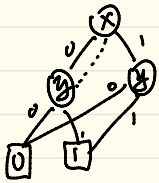
Shannon's expansion

* Remove redundant terminals (leaf nodes)



* Remove redundant decision points/nodes

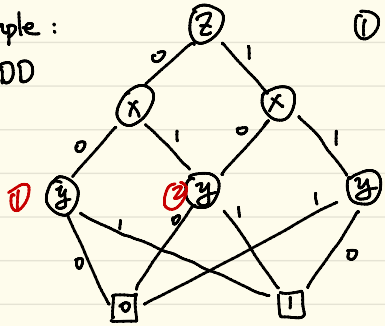
* Remove duplicate decision points/nodes



Week-9-2

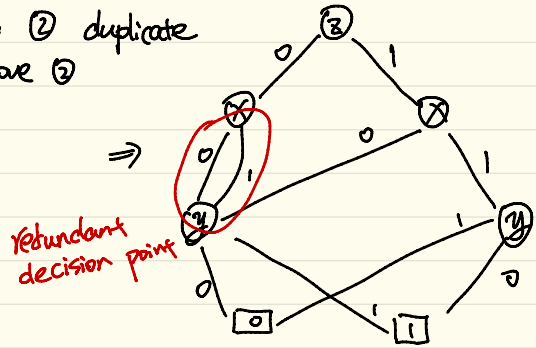
Example :

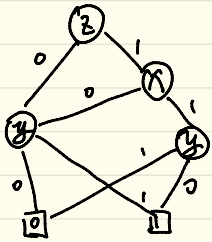
BDD



① and ② duplicate

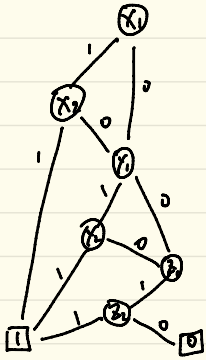
remove ②





$z = (x_1 \wedge x_2) \vee (y_1 \wedge y_2) \vee (z_1 \wedge z_2)$ $[x_1 x_2 y_1 y_2 z_1 z_2]$

$B[x_1 \mapsto 1]$



$z = (x_1 \wedge B[x \mapsto 1]) \vee (\neg x_1 \wedge B[x_1 \mapsto 0])$

Build (B)

left: Build $(B[x_1 \mapsto 1])$

right: Build $(B[x_1 \mapsto 0])$

create node any name x_1 and write pts to left & right child

Operations on BDDs

$\neg$: negating a formula

$\varphi \to D_\varphi$ (PL)

$\neg\varphi \to D_{\neg\varphi}$ (flipping / swaping the terminal nodes)

$\wedge$:

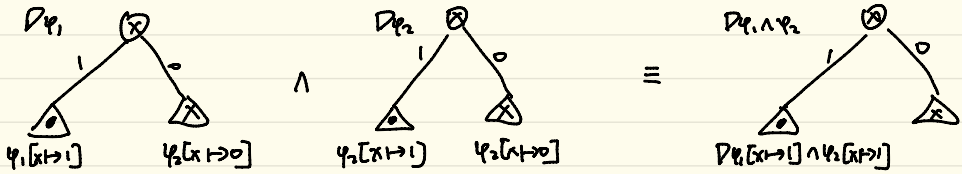
$\varphi_1 \to D_{\varphi_1}$ $\qquad$ $\varphi_2 \to D_{\varphi_2}$

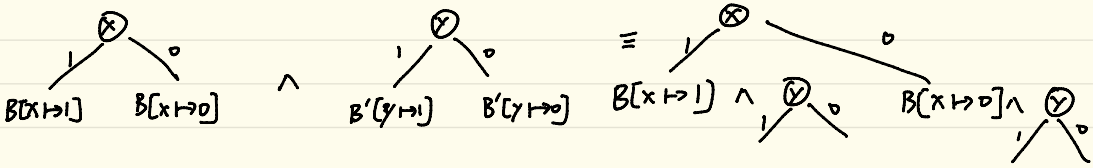
$\varphi_1 \wedge \varphi_2 \xrightarrow{?} D_{\varphi_1 \wedge \varphi_2}$

$$\varphi_1 = \underbrace{(x \wedge \varphi_1[x \mapsto 1])}_{A_1} \vee \underbrace{(\neg x \wedge \varphi_1[x \mapsto 0])}_{A_2}$$

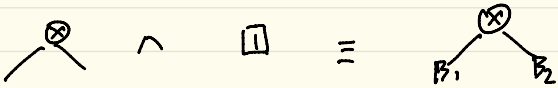
$$\varphi_2 = \underbrace{(x \wedge \varphi_2[x \mapsto 1])}_{B_1} \vee \underbrace{(\neg x \wedge \varphi_2[x \mapsto 0])}_{B_2}$$

$$\begin{aligned}
\varphi_1 \wedge \varphi_2 &\equiv (A_1 \vee A_2) \wedge (B_1 \vee B_2) \\
&\equiv (A_1 \wedge (B_1 \vee B_2)) \vee (A_2 \wedge (B_1 \vee B_2)) \\
&\equiv (A_1 \wedge B_1) \vee (A_1 \wedge B_2) \vee (A_2 \wedge B_1) \vee (A_2 \wedge B_2) \\
&\equiv (x \wedge (\varphi_1[x \mapsto 1] \wedge \varphi_2[x \mapsto 1])) \vee (\neg x \wedge (\varphi_1[x \mapsto 0] \wedge \varphi_2[x \mapsto 0]))
\end{aligned}$$

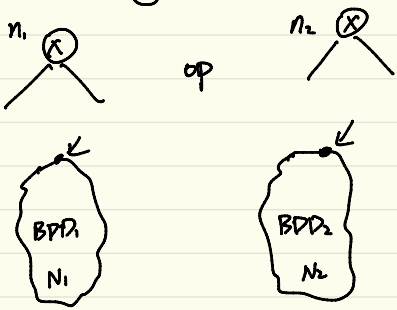




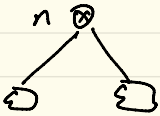
Base case



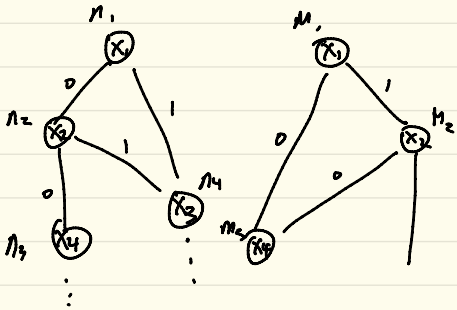
memoization (tabling

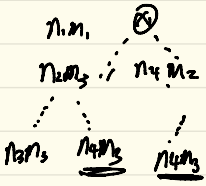


→ (op, n1, n2) in already computed
→ return the result from table
else do-recursion



Store in the table
(op, n1, n2) ↦ n





Week-10-1

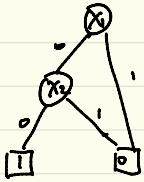
$\exists x.F = F[x \mapsto 0] \vee F[x \mapsto 1]$

$\forall x.F = F[x \mapsto 0] \wedge F[x \mapsto 1]$

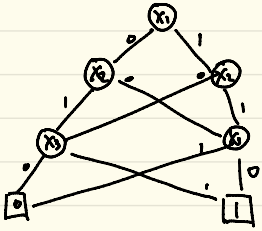
$F[x \mapsto 0]$ or $F[x \mapsto 1]$: restricting operation

Example

F:

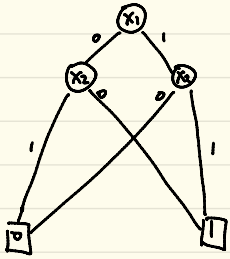


$F[x_2 \mapsto 0]$?

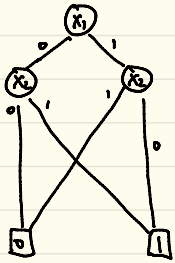


$\exists$

$F[x_3 \mapsto 0]$ $\Rightarrow$



$F[x_3 \mapsto 1]$



$\exists x_3.F \equiv F[x_3 \mapsto 0] \vee F[x_3 \mapsto 1]$

CTL : syntax + Semantics (Set-the oretic form)
↳ KS

Fixed point Characteration of semantics
↳ marking algorithm

Symbolic computation – translating sets to propositioned logic formula.
↳ BDD

Linear Temporal logic (LTL)
- CTL describe properties over tree
- LTL describes properties over path
↳ Amir Pneli (Turing Awarrd)

SPIN : → Process Meta language (Promela)
→ Simple Promela interpreter (SPIN)
Automata-theorelic model checking

<u>Syntax</u>

$\varphi \rightarrow$ true / false / $\neg\varphi$ / AP / $\varphi \vee \varphi$ / $\varphi \wedge \varphi$ ...
$X\varphi$ / $F\varphi$ / $G\varphi$ / $\varphi \, U \, \varphi$

X: in next suffix
F: in some future suffix
G: in all suffix

$\pi: s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$
π': first suffix of π
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \cdots$
$\pi^2: s_2 \rightarrow s_3 \rightarrow \cdots\cdots$
$\pi^0: \pi$

$X\varphi$ is satisfied in a path
$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$
$\Rightarrow$ p is satisfied in the path $s_1 \rightarrow s_2 \rightarrow \cdots$
$\Rightarrow p \in L(s_1)$

Fp is satisfied in a path $\pi: s_0 \rightarrow s_1 \rightarrow \cdots$
$\Rightarrow$ p is satisfied in s_0 or s_1 or s_2 ...

GFp is satisfied in a path $\pi: s_0 \to s_1 \to \cdots$

$\Rightarrow$ Fp is satisfied in $\begin{cases} s_0 \to s_1 \to \cdots \\ s_1 \to s_2 \to \cdots \\ s_2 \to s_3 \to \cdots \end{cases}$

π : path

π^i : i-th suffix of π ($\pi^0 : \pi$)

$\pi[i]$: i-th state of π ($\pi[0]$ is the start state of π)

$[\![\ ]\!]$: input φ, k_S: $M \to$ output set of states in M.

$s \in [\![\varphi]\!]_M \iff s \models_M \varphi$ CTL

satisfies

$\pi \models_M \varphi$ (π satisfies φ)

<u>Semantics</u> 1. $\pi \models true$ 2. $\pi \not\models false$

3. $\pi \models p$ iff $p \in L(\pi[0])$ 4. $\pi \models \varphi_1 \wedge \varphi_2$ iff $\pi \models \varphi_1$ and $\pi \models \varphi_2$

5. $\pi \models \neg\varphi$ iff $\pi \not\models \varphi$

6. $\pi \models X\varphi$ iff $\pi^1 \models \varphi$

7. $\pi \models F\varphi$ iff $\exists i \geq 0.\ \pi^i \models \varphi$

8. $\pi \models G\varphi$ iff $\forall i \geq 0.\ \pi^i \models \varphi$

9. $\pi \models \varphi_1 \mathcal{U} \varphi_2$ iff $\exists i \geq 0.\ \pi^i \models \varphi_2 \wedge \forall j < i : \pi^j \models \varphi_1$

<u>Example</u>

$\pi \models \neg X \neg \varphi \iff \pi \models X\varphi$ $\pi \models (true\ \mathcal{U}\ \varphi) \iff \pi \models F\varphi$

$\pi \models \neg X \neg \varphi$ iff $\pi \not\models X \neg\varphi$ iff $\pi^1 \not\models \neg\varphi$ iff $\pi^1 \models \varphi$ $\to$ iff $\pi \models X\varphi$

$\pi \models (true\ \mathcal{U}\ \varphi)$ iff $\exists i \geq 0.\ \pi^i \models \varphi \wedge \underbrace{\forall j < i.\ \pi^j \models true}_{\text{tautology}}$ iff $\exists i \geq 0.\ \pi^i \models \varphi$ iff $\pi \models F\varphi$

10-week-2

- $\pi \models \neg F \neg \varphi$ iff $\pi \models G\varphi$
  iff $\pi \not\models F\neg\varphi$
  iff $\forall i \geq 0 .\ \pi^i \not\models \neg\varphi$ iff $\forall i \geq 0 .\ \pi^i \models \varphi$ iff $\pi \models$

- $\pi \models \neg G \neg \varphi$ iff $\pi \models F\varphi$

Adequate set for LTL

$$\begin{tikzcd} X,\ U \arrow[d] & \\ F \arrow[r] & G \end{tikzcd}$$

<u>Model checking</u> Given a $KS = (S, T, L)$ and some start of the system, verify whether the start state satisfy a property (LTL)

$S \models_M \varphi$ iff $\forall \pi \in Path(s) .\ \pi \models \varphi$

$S \not\models_M \varphi$ iff $\exists \pi \in Path(s) .\ \pi \not\models \varphi$ ($\pi \models \neg\varphi$)

$\Downarrow \times$

$S \models_M \neg\varphi$ iff $\forall \pi \in Path(s) .\ \pi \models \neg\varphi$

* two processes never enter critical section at the same

$A\ G\ (\neg C_1 \vee \neg C_2)$

implicit

* There exists a path from some start state where p holds eventually (EFp)

LTL : $G\neg p$

$S \models_M G\neg p$ Yes. $\forall \pi \in Path(s) .\ \pi \models G\neg p$ $\therefore$ S doesn't satisfy R

$S \models_M G\neg p$ No. $\exists \pi \in Path(s) .\ \pi \not\models G\neg p$ iff $\pi \models \neg G\neg p$ iff $\pi \models Fp$

$\vdots$

$\therefore$ S satisfies R

CTL vs LTL

Whenever p is true, there exists a path where eventually q is true

CTL: $AG(p \to EF(q))$

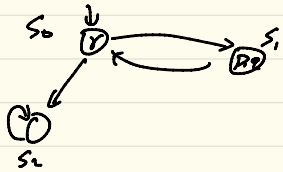
LTL: $\underbrace{G(p \to Fq)}_{AG(p \to AFq)}$ ....? .... Whenever p is true, it is always eventually followed by q

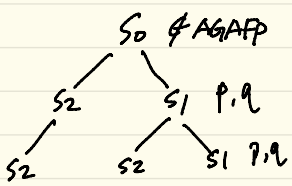
Along all paths if p is true infinitely often then q is true infinitely often.

$AGAFp \to AGAFq$

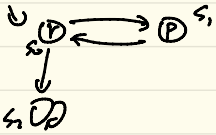
$AGAF(p \to AGAFq)$

M_1





M_2

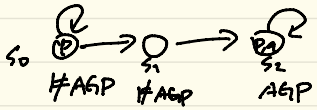


s_0 does satisfy $AGAFp \to AGAFq$ but doesn't satisfy the requirement.

$s_0 \models_M GFp \to GFq \quad \forall \pi \in Path(s_0) \quad \pi \models (GFp \to GFq)$

* AFAGP FGP

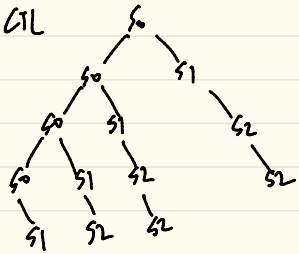
$s \models AFAGP$ vs $s \models FGP$: claim X



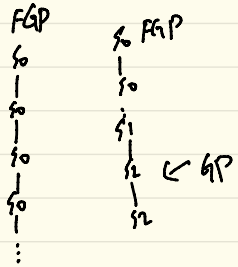
$s_0 \models AFAGP$

$s_0 \in [\![AFAGP]\!]$ X

CTL



LTL



11-Week-2

Equivalences

$\varphi_1\ \varphi_2$: LTL ($\pi \models \varphi_1$ iff $\pi \models \varphi_2$)

$FX p \equiv XFp \equiv AXAFp \not\equiv AFAXp$

$FGp \not\equiv AFAGp$

$GFp \to GFq \not\equiv AGAFp \to AGAFq$

$GFp \equiv AGAFp$

1. $F\varphi_1 \vee F\varphi_2 = F(\varphi_1 \vee \varphi_2)$

   $F\varphi_1 \to F(\varphi_1 \vee \varphi_2)$

   $F\varphi_2 \to F(\varphi_1 \vee \varphi_2)$

   $F(\varphi_1 \vee \varphi_2) \to F\varphi_1 \vee F\varphi_2$

2. $F\varphi_1 \wedge F\varphi_2 \equiv F(\varphi_1 \wedge \varphi_2)$ ✗

3. $G(\varphi_1 \vee \varphi_2) \equiv G\varphi_1 \vee G\varphi_2$ ✗

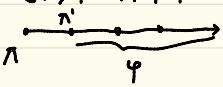
   implies

4. $G(\varphi_1 \wedge \varphi_2) \equiv G\varphi_1 \wedge G\varphi_2$ ✓

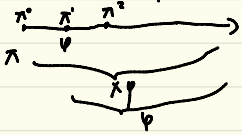
5. $XF\varphi \equiv FX\varphi$ ✓

   $\pi' \models F\varphi$

   $\exists i \geq 1.\ \pi^i \models \varphi$



   $\exists i \geq 0.\ \pi^i \models X\varphi \longrightarrow \exists i \geq 0.\ \pi^{i+1} \models \varphi$



   $\exists i \geq 1\ \pi^i \models \varphi$

6. $XG\varphi \equiv GX\varphi$ ✓

7. $G\varphi \equiv \varphi \wedge XG\varphi$ $\quad F\varphi \equiv \varphi \vee XF\varphi$ $\quad \varphi_1 \mathcal{U} \varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge X(\varphi_1 \mathcal{U} \varphi_2))$

English → LTL (Expressibility and verifiability)

1. Whenever p true, q true after that in one or more steps. $\quad s \models \varphi$ iff $\forall \pi \in Path(s).\ \pi \models \varphi$

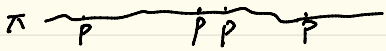
   $G(p \to XFq)$ $\quad$ CTL: $AG(p \to AXAFq)$

2. Along all paths. p true eventually and q true eventually, p and q never true at the same time.

   $Fp \wedge Fq \wedge \neg F(p \wedge q)$ $\quad G(\neg p \vee \neg q)$

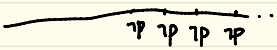
3. There exists a path where p true infinitely often

   Verify: $FG\neg p$



   $s \models FG\neg p$ $\quad \forall \pi \in Path(s).\ \pi \models FG\neg p$

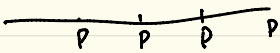
   s doesn't satisfy the requirement



   $s \not\models FG\neg p$ $\quad \exists \pi \in Path(s)\ \pi \not\models FG\neg p$

   $\pi \models FGp$

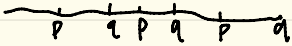
   satisfy



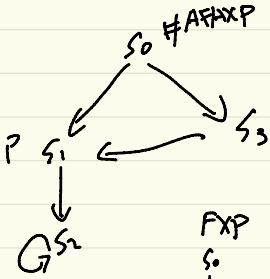
4. Along all paths, if q true at some state, then p time in some other state before q

$Fq \rightarrow (\neg q \, U \, p) \equiv G\neg q \vee (\neg q \, U \, p)$

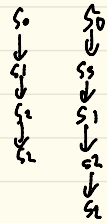
CTL: $\neg E(\neg p \, U \, q)$



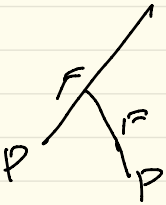
5. Requirement : AFAXp to represent some property AF:Fxp to represent



FXp



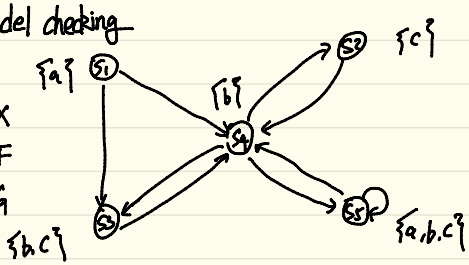
6. Requirement : AFAGp to represent a requirement GFP to represent



Week-12-1

Model checking

O: X
<>: F
[]: G



1. FGc does S_1 satisfy FGc? No
   $S_1 \to S_3 \to S_4 \to S_3 \to S_4 \cdots$
2. GFc does S_1 satisfy GFc? Yes
3. $a \mathcal{U} G(b \vee c)$ $\{S_1, S_2, S_3, S_4, S_5\}$
4. $(X X b) \mathcal{U} (b \vee c)$ $\{S_2, S_3, S_4, S_5\}$ $S_1 \to S_4 \to S_2$

Spin Model Checker

Process Meta language: Promela $\xrightarrow{\text{translate}}$ C program $\xrightarrow{\text{gcc/cc}}$ executable

1. Spin -a $\underbrace{\text{test.txt}}_{\text{promela}}$
   ↓ pan.c
2. gcc -o pan pan.c
   ↓ execute
3. pan -a → model check the system specified in test.txt against some LTL formula
   or
   pan -a -N r0
   ↓ if error → generate test.txt.trail
4. spin -t test.txt (spin -t -r -s test.txt)

```
proctype sender(....) {
        . .
        : :
        . .
}
```

```
init() {
    run sender(...) arg
    run
}
```

```
active proctype sender(){

}
```

init is not necessary.

Week-12-2

LTL MC Algo

<u>Obj</u> Given KS and some start state

we want to verify whether all paths from the start state belongs to the semantics of some LTL formula. φ

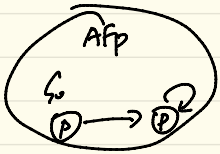
$L(M)$ = set of all paths starting from start state

$L(\varphi)$ = set of all paths that belong to semantics of φ

$L(M) \subseteq L(\varphi)$: start state satisfies φ

EX

$\pi : s_0 \to s_1 \to s_2 \cdots$
$(p \wedge \neg q) \to (p \wedge q) \to (p \wedge q)$ } KS



$Fp : p \to true \to true \to \cdots$
$true \to p \to true \to \cdots$
$true \to true \to p \to \cdots$ } φ

$p \to \neg p \to \neg p \cdots$
$\neg p \to p \to \neg p \cdots$
$\neg p \to \neg p \to p \cdots$

<u>Automata</u>

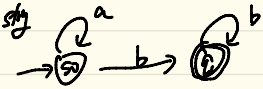
$A = (Q, Q_0, \Gamma, \Delta, F)$

states start state, alph, edge, final/accepting

$Q_0 \subseteq Q \qquad F \subseteq Q \qquad \Delta \subseteq Q \times \Gamma \times Q \; (q \xrightarrow{a} q')$

Run in automata over a finite string (Regular Automata)

v: finite length string $\quad v[i]$: ith letter in the str



A run ρ over v in an automata A is

a function $\rho : \{0, 1, 2, \cdots |v|\} \to Q$

st. (a) $\rho(0) \in Q_0$

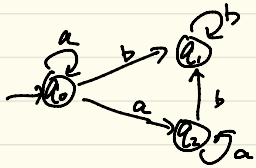
(b) $\forall i \in [0, |v|-1] : \rho(i) \xrightarrow{v[i]} \rho(i+1) \in \Delta$

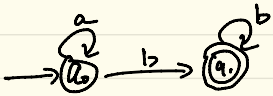
a b b b

$q_0 \to q_0 \to q_1 \to q_1 \to q_1$

$(\rho_0)\ (\rho_1)\ (\rho_2)\ (\rho_3)\ (\rho_4)$

Accepting run ρ is such that $\rho(|v|) \in F$ } accepting condition for regular automata.





string: ab

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1$ ✓

string: bb

$q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_1$ ✓

string: aaa

$q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0$ ✗

string aba

$\underset{(p_0)}{q_0} \xrightarrow{a} \underset{(p_1)}{q_1} \xrightarrow{b} \underset{(p_2)}{q_1}$ ✗

$L(A)$ = set of strings for which the A has an accepting run

$= \{b, ab, abb, abbb, \cdots\}$

<u>w-regular automata</u> (Büchi automata)

v : string of infinite length

A run ρ over v is a function

$\rho : \{0, 1, 2, \cdots\} \longrightarrow Q$ such that

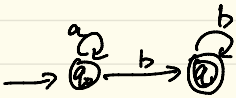
(a) $\rho(0) \in Q_0$

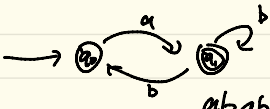
(b) $\forall i \geq 0 .\ \rho(i) \xrightarrow{v(i)} \rho(i+1) \in \Delta$

$\inf(\rho) = \{q \in Q \mid \forall i \geq 0 .\ \exists j > i .\ \rho(j) = q\}$

$\underbrace{abbb\cdots}$

$q_0 \to q_0 \to q_1 \to q_1 \to q_1 \cdots$





abab ab

$\inf(\rho) = \{q_0, q_1\}$

Accepting run ρ is such that

$\inf(\rho) \cap F \neq \{\}$

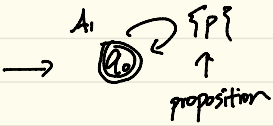
$L(A)$ = language of Büchi automata

= set of string that has an accepting run

$= \{bbb\cdots, abbb\cdots, aabbb\cdots, \cdots\}$

Week-13-1

EX1

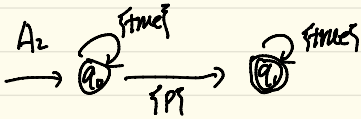


$L(A_1) = \{ \{p\}.\{p\}.\{p\}.\{p\}\ldots\ldots \}$

$\equiv [\![Gp]\!]$

= set of all paths such every suffix satisfies p.

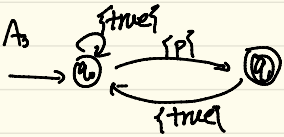
EX2



$L(A_2) = \{ \{p\}.\{true\}.\{true\}.\cdots$
$\{true\}.\{p\}.\{true\}.\cdots$
$\{true\}.\{true\}.\{p\}\cdots \quad \cdots \}$

Fp

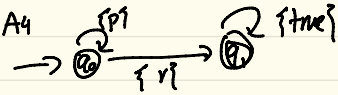
Ex3



$L(A_3) = \{\ldots\ldots\}$

Gfp

EX4



$p \mathcal{U} r$

All the paths that belongs to semantics $\neg\varphi$

$L(A_{\neg\varphi}) \equiv$ Semantics of $\neg\varphi$

$Fp \equiv p \vee XFp$

$Gp \equiv p \wedge XGp$

$p \mathcal{U} r \equiv r \vee (p \wedge X(p \mathcal{U} r))$

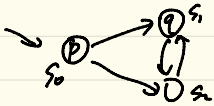
Spin → "[]p"

Runtime for LTL to Büchi translate in exp wrt size of LTL formula

Find all the paths in the KS starting from starte state

Ex. Translating KS paths → Buchi A.

KS.

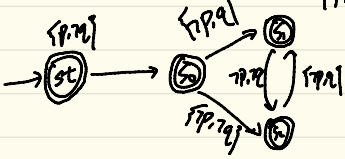


$s_0 \to s_1 \to s_2 \to s_1 \to s_2 \cdots$

$\{p, \neg q\} \to \{\neg p, q\} \to \neg p, \neg q \to \neg p, q \cdots$

$s_0 \to s_2 \to s_1 \cdots$

$\{p, \neg q\}\ \{\neg p, \neg q\}\ \{\neg p, q\} \cdots$



$L(A_{KS})$

$O(|S| + |T|)$

$\boxed{KS: \quad \varphi}$ →

check whether there is any path in KS that belongs to semantics of $\neg\varphi$

$L(A_{KS}) \cap L(A_{\neg\varphi}) \overset{?}{=} \{\}$

crossproduct $\otimes$

$L(A_{KS} \times A_{\neg\varphi})$

$\equiv L(A_{KS}) \cap L(A_{\neg\varphi})$

$A_{KS} \equiv (Q_{KS}, Q_{0KS}, \Gamma_{KS}, \Delta_{KS}, F_{KS})$

$A_{\neg\varphi} \equiv (Q_{\neg\varphi}, Q_{0\neg\varphi}, \Gamma_{\neg\varphi}, \Delta_{\neg\varphi}, F_{\neg\varphi})$

$A \equiv (Q, Q_0, \Gamma, \Delta, F)$

$Q \subseteq Q_{KS} \times Q_{\neg\varphi}$

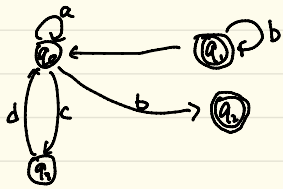
$Q_0 \subseteq Q_{0KS} \times Q_{0\neg\varphi}$

$\Delta = \{(s_i, q_j) \xrightarrow{A \cup B} (s_i', q_i') \mid s_i \xrightarrow{A} s_i' \in \Delta_{KS} \wedge q_j \xrightarrow{B} q_j' \in \Delta_{\neg\varphi} \wedge A \cup B \text{ is consistent}\}$

$\Gamma \subseteq \Gamma_{KS} \times \Gamma_{\neg\varphi}$

$F = Q_{KS} \times F_{\neg\varphi}$

A: check whether $L(A) = \{\}$



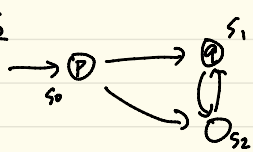
A: $\{p, \neg q\}$

B: $\{p, q\}$

$A \cup B$: inconstant

Week-13-2

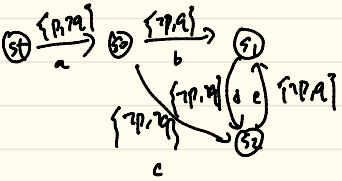
EX KS



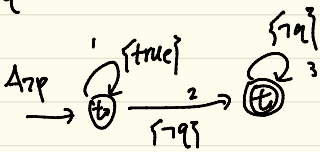
$\varphi : Gfq$

$\neg\varphi : FG\neg q$

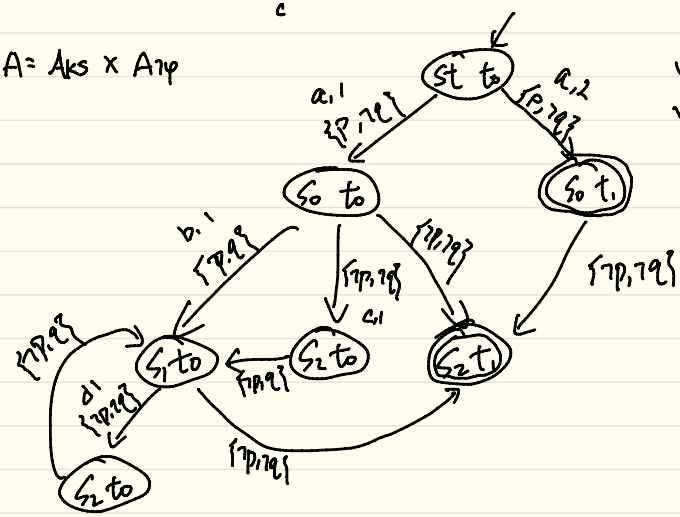
A_{KS}



$A_{\neg\varphi}$



$A = A_{KS} \times A_{\neg\varphi}$



KS $\qquad$ φ

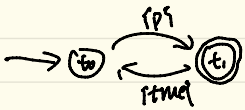
✓ $L(A_{KS}) \subseteq L(A_\varphi)$

✓ $\equiv L(A_{KS}) \cap L(A_\varphi) = \underline{\Phi}$

$L(A) = L(A_{KS} \times A_{\neg\varphi}) = \underline{\Phi}$

$\equiv L(A_{KS}) \cap L(A_{\neg\varphi}) = \underline{\Phi}$

A

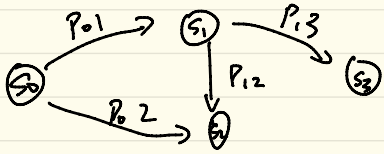


$L(A) = \{ \{p\} \{true\} \{p\} \{true\} \{p\} \cdots \}$

$s_0 \quad s_1 \quad s_2 \quad s_3 \quad s_4$

$p \wedge G(p \rightarrow XXp)$

Probalistic Systems



$M = (S, T, L)$

S: finite set of states

$T: S \times S \to [0,1]$ st.

(a) $\forall s. \exists s'. T(s, s') > 0$ (Kripke condition)

(b) $\forall s. \sum T(s, s') = 1$ (stochastic condition)

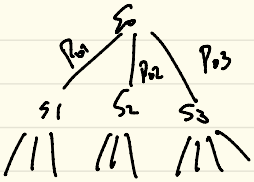
(c) $\forall \pi \in Path(s)$ $T(\pi[i], \pi[i+1])$ depends only on $\pi[i]$ (Markov Condition)

⊛ Probability of a finite path in DTMC

$\pi: \pi[0] \to \pi[1] \to \pi[2] \cdots \to \pi[n-1]$

$$\prod_{i=0}^{n-1} T(\pi[i], \pi[i+1])$$

⊛ Sum of the probability of all finite paths from a state



Sum of Prob of all finite paths from s_0 that of size 1 $= 1$

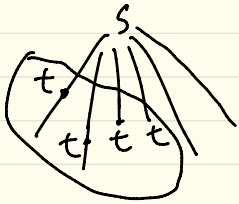
Sum of Prob of all finite paths from s_0 that one of size 2 $= 1$

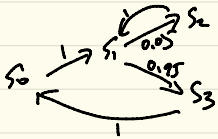
Week-14-1

Prob System (DTMC)

Reachability : Given some state s, what is the prob. of s reaching another state t

[Sum of the prob. of finite paths from s to t]



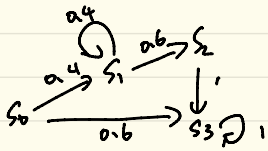


$s_0 \sim s_2$

$= 0.05 + 0.95 \times 0.05 + 0.95^2 \times 0.05 + \cdots$

$= 0.05 (1 + 0.95 + 0.95^2 + 0.95^3 + \cdots)$

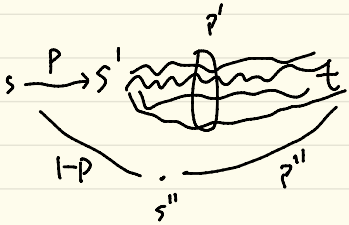
$= 0.05 \frac{1}{1-0.95} = 1$



$s_0 \sim s_2$

$= 0.24 (1 + 0.4 + 0.4^2 + 0.4^3 + \cdots)$

$= 0.24 \frac{1}{1-0.4} = \frac{0.24}{0.6} = 0.4$

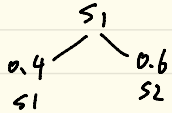


x_i : prob of s_i reaching s_2

$x_2 = 1 \qquad x_3 = 0$

$x_1 = 0.4 x_1 + 0.6 x_2 = 1$

$x_0 = 0.4 x_1 + 0.6 x_3 = 0.4$



PRISM

Initial approx	1st approx	2nd approx
x_i are set to 0	$x_2 = 1$	$x_2 = 1$. $x_1 = 0.6$
	other x_i's 0	

PCTL (Prob CTL)

$\varphi \to true / AP / \neg\varphi / \varphi \vee \varphi / \cdots / P_{\bowtie r}\,\psi$ $\quad \bowtie : \{<, >, \leq, \geq\}$ $\quad r \in [0,1]$

$\psi \to X\varphi \,/\, \varphi \,\mathcal{U}\, \varphi \,/\, \varphi \,\mathcal{U}^{\leq k} \varphi$ $\quad$ k is some integer.

[$P_{\bowtie r}\psi$: (sum of prob. of paths starting from s that satisfy ψ $\bowtie r$)

semantics of $P_{\leq 0.6}(a \,\mathcal{U}\, b)$

Semantics. $\pi : \pi[i]$

$s \models true$ $\qquad$ 2. $s \models p$ iff $p \in L(s)$

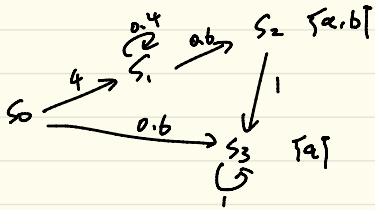
$s \models \neg\varphi$ iff $s \not\models \varphi$ $\qquad$ 4. $\models \varphi_1 \vee \varphi_2$ iff $s \models \varphi_1$ or $s \models \varphi_2$

$s \models P_{\bowtie r}\psi$ iff $\sum Prob(\pi : \pi \in Path(s).\ \pi \models \psi) \bowtie r$.

$\pi \models X\varphi$ iff $\pi[1] \models \varphi$

$\pi \models \varphi_1 \,\mathcal{U}\, \varphi_2$ iff $\exists i \geq 0.\ \pi[i] \models \varphi_2 \wedge \forall j < i.\ \pi[j] \models \varphi_2$

$\pi \models \varphi_1 \,\mathcal{U}^{\leq k} \varphi_2$ iff $\exists i \geq 0.\ i \leq k.\ \pi[i] \models \varphi_2 \wedge \forall j < i.\ \pi[j] \models \varphi_1$



$P \leq 0.5 (a \,\mathcal{U}\, b)$

x_i: sum of prob. of paths starting from S_i that satisfies $a \,\mathcal{U}\, b$

$x_2 = 1 \qquad x_3 = 0$

$x_1 = 0.4x_1 + 0.6x_2 = 1$

$x_0 = 0.4x_1 + 0.6x_3 = 0.4$

$P_{\leq 0.5}(P_{>0.6}(a \,\mathcal{U}\, b) \,\mathcal{U}\, c)$

set of states that satisfy $\underbrace{P_{>0.6}(a \,\mathcal{U}\, b)}_{d}$

$P_{\leq 0.5}(d \,\mathcal{U}\, c)$

$S \models P_{>0}(true \,\mathcal{U}\, q)$ $\quad \sim\sim \quad$ $s \models \varphi$ (CTL)

$S \models P_{\geq 1}(true \,\mathcal{U}\, q)$ $\qquad$ $s \models \underline{\varphi}$ (CTL)