SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution
Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

19CSE303 – ARTIFICIAL INTELLIGENCE

III YEAR IV SEM

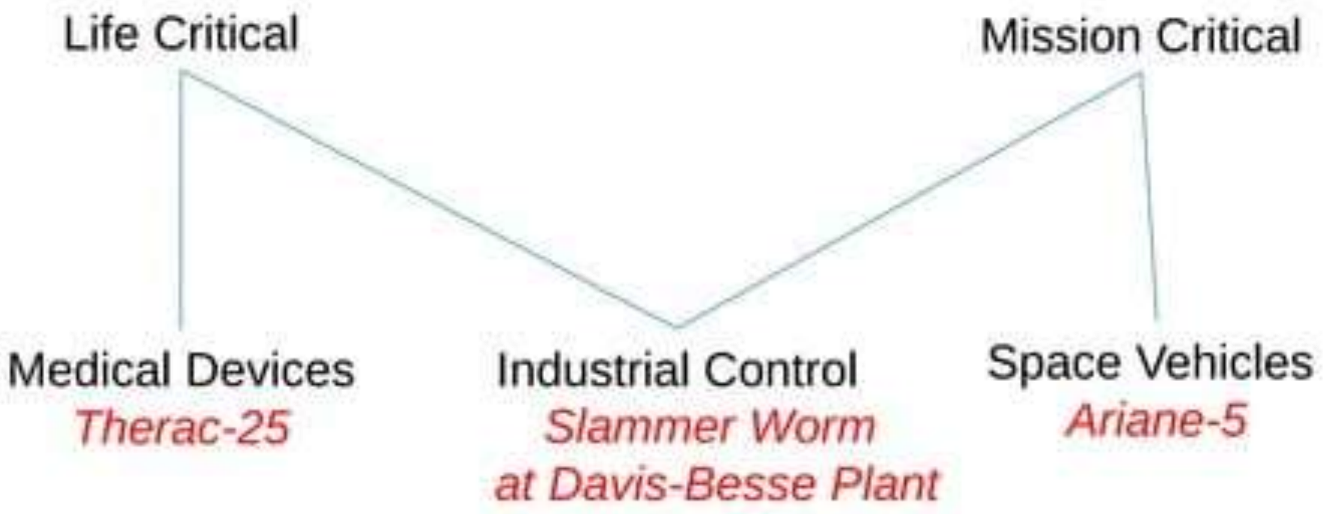
UNIT IV – **UNCERTAIN KOWLEDGE AND REASONING**

TOPIC – **Temporal Model**

Formal methods: Why?

$$\begin{tikzcd}
\overbrace{\overbrace{\text{Specification}}^{\textit{IDEAL DEVELOPMENT}} \arrow[ddd, bend right] + \Delta\text{Errors}}^{\textit{REAL DEVELOPMENT}} \arrow[d] \arrow[d, shift left=12, red] \\
\text{Design} \arrow[d] \quad\quad + \Delta\text{Errors} \arrow[d, red, "\downarrow\downarrow"] \\
\text{Coding} \arrow[d] \quad\quad + \Delta\text{Bugs} \arrow[d, red, "\downarrow\downarrow\downarrow"] \\
\text{Testing} \arrow[d] \quad\quad \text{Limited Coverage} \arrow[d, red] \\
\text{Product} \quad\quad \text{with safety issues}
\end{tikzcd}$$

Formal methods: Where?



- *An Investigation of Therac-25 Accidents* [Leveson, Turner, 93]
- *Ariane 5 Flight 501 Failure, Report by Inquiry Board* [Lions, 96]
- Slammer worm crashed Ohio nuke plant network, News Report [http://www.securityfocus.com/news/6767, 03]

Sentences (Syntax) and Models (Semantics)

- *"Every PhD student must have an advisor who is a member of faculty"*

- $F = \forall x \cdot \textit{phd-student}(x) \Rightarrow \exists y \cdot \textit{advisor-of}(y, x) \wedge \textit{faculty}(y)$

- What does F mean?
  - Classical Interpretation - A mathematical structure with:
  - *advisor-of* mapped to a binary relation on the structure
  - *phd-student, faculty* mapped to unary relations
  - Logical symbols ($\wedge$, $\Rightarrow$) have fixed interpretation
  - Quantifiers ($\forall$,$\exists$) range over elements of the underlying set

- Model of F = a satisfying interpretation for F

Fundamental reasoning tasks

- M *satisfies* F ? *Model checking problem*
- ? *satisfies* F *Satisfiability problem*
- * *satisfies* F *Validity problem*
- { a *: M* |= F(a)} *Formula (query) evaluation*

Note: |= denotes *satisfies* relation

Modal Logic

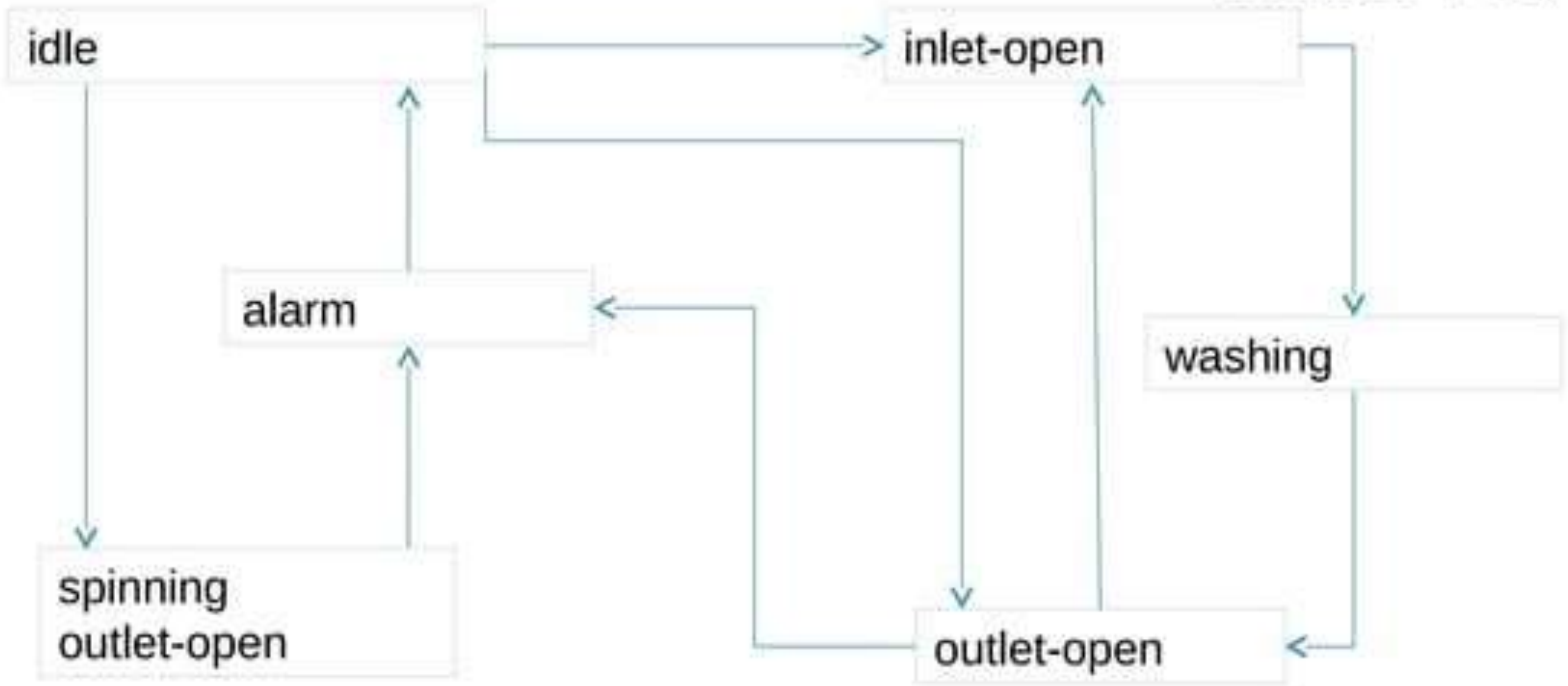
- Modalities: Necessity, knowledge, belief, obligation, tense
  - *Symbolic Logic* [Lewis,32]
- Possible World Semantics
  - Kripke Structure
- Temporal Logic
  - *Time and Modality* [Prior,57]
  - Temporal modalities: always (☐), eventually (•)

Possible Worlds & Kripke Structure for a washing machine

Accessibility Relation: {⟶ }

Kripke Structure = (States, →, L)

L : States → 2AP



- alarm

□¬(spinning ∧ washing)

□ ¬(inlet-open ∧ outlet-open)

Verification: Sequential Programs

Hoare triple: $\{\underset{\uparrow\atop\text{precondition}}{P}\}\ \overset{\text{program statement}\atop\swarrow}{S}\ \{\underset{\uparrow\atop\text{postcondition}}{Q}\}$ - using some predicate logic

- *Assigning Meanings to Programs* [Floyd,67]
- *An Axiomatic basis for Computer Programming* [Hoare, CACM69] (Hoare Logic)
- *Guarded commands, non-determinism and formal derivation of programs* [Dijstra,75] (GCL)

Hoare Logic

(Assignment Axiom)

$$\frac{}{\{Q[E/id]\}\ id{=}E;\ \{Q\}}$$

(Conditional Rule)

$$\frac{\{P \wedge E\}\ S_1\ \{Q\} \qquad \{P \wedge \neg E\}\ S_2\ \{Q\}}{\{P\}\ \texttt{if (E) \{}S_1\texttt{\} else \{}S_2\texttt{\}}\ \{Q\}}$$

(Sequencing Rule)

$$\frac{\{P\}\ S_1\ \{R\} \qquad \{R\}\ S_2\ \{Q\}}{\{P\}\ S_1\ ;\ S_2\ \{Q\}}$$

(Pre-strengthening, Post-weakening)

$$\frac{P \Rightarrow P' \qquad \{P'\}\ S\ \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\ S\ \{Q\}}$$

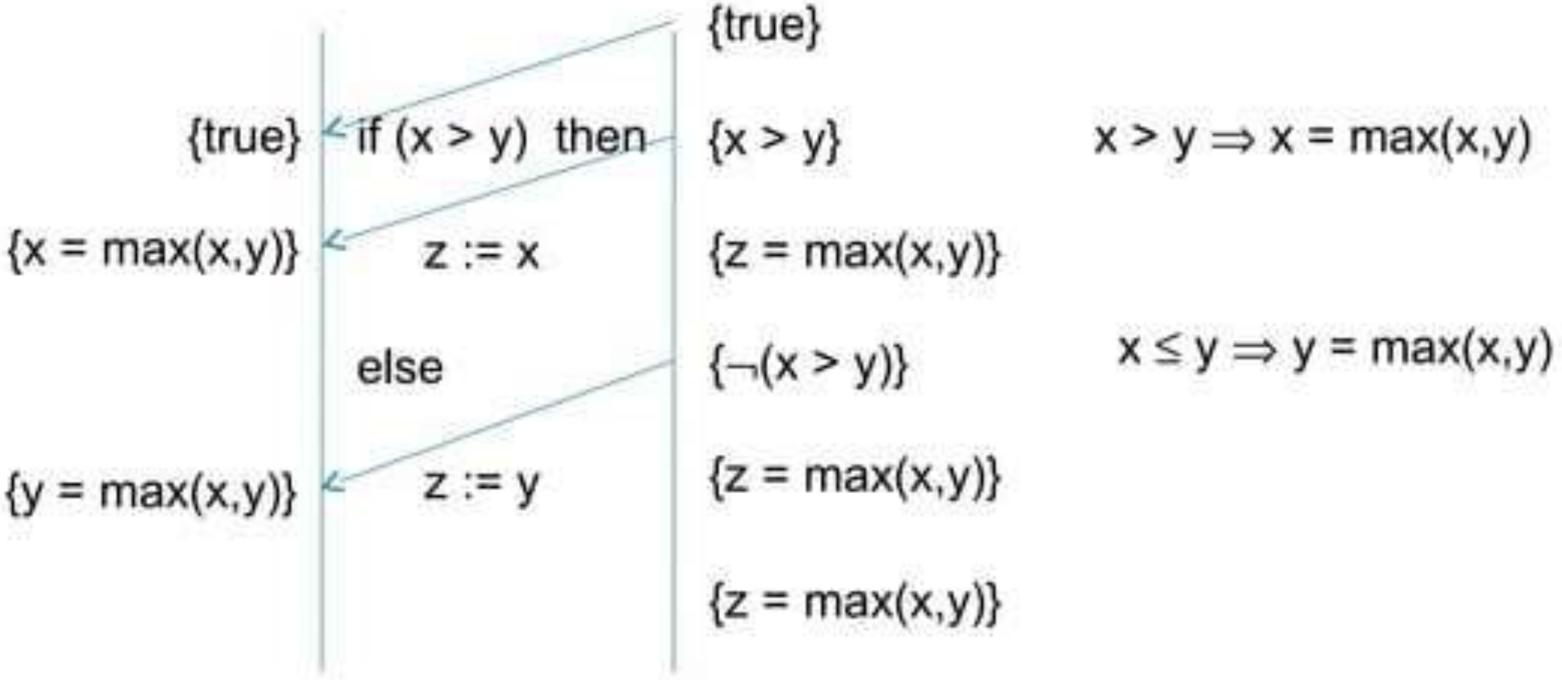
Proof Tableaux

$$\begin{array}{c|c|c} & & \{P_0\} \\ \{P_1\} & c_1; & \{Q_1\} \\ \{P_2\} & c_2; & \{Q_2\} \\ & \vdots & \\ \{P_n\} & c_n & \{Q_n\} \end{array}$$

Hoare Logic Example

- P = if (x > y) then z := x else z := y

- Prove that: {true} P {z = max(x,y)}



Concurrency

- Simple pre-condition/post-condition assertions insufficient
  - Deadlocks, Data races, Starvation!
  - Need a language for expressing concurrency properties

- Cannot ignore intermediate steps!
  - P ; Q : Intermediate states of P & Q do not interleave/interact
  - P || Q : Intermediate states interleave/interact (in exponential number of ways)

Specifying properties of concurrent systems

- Language: Temporal Logic
  - *The Temporal Logic of Programs* [Pnueli,77] (Linear Temporal Logic)

- Safety
  - something bad will never happen:
  - $\Box \neg(\text{spinning} \wedge \text{washing})$

- Liveness
  - Something good will eventually happen:
  - • alarm

- Fairness
  - Always something good will eventually happen
  - $\Box$ • idle

Why Temporal Logics?

$$\begin{tikzcd}
\text{Computational System} \arrow[dr, "\text{Operational Semantics}"] & & \text{Modal and Temporal Logics} \\
& \text{Labelled Transition System (Kripke Structure)} \arrow[ur, "\text{models of}"'] &
\end{tikzcd}$$

[Stirling03] http://www.fing.edu.uy/inco/eventos/wssa/

Producer-Consumer with 1-buffer

Producer

```
wtp: while (!isempty);
csp: buf = produce();
flp  : isempty = false;
```

Consumer

```
wtc: while (isempty);
csc: consume(buf);
flc  : isempty = true;
```

State space for 1-buffer system

States = *control state* $\times$ *data state*

$= \boxed{\{wtp, csp, flp\} \times \{wtc, csc, flc\}} \boxed{\times \{da\} \times \{em\}}$

wt : wait

cs : critical section

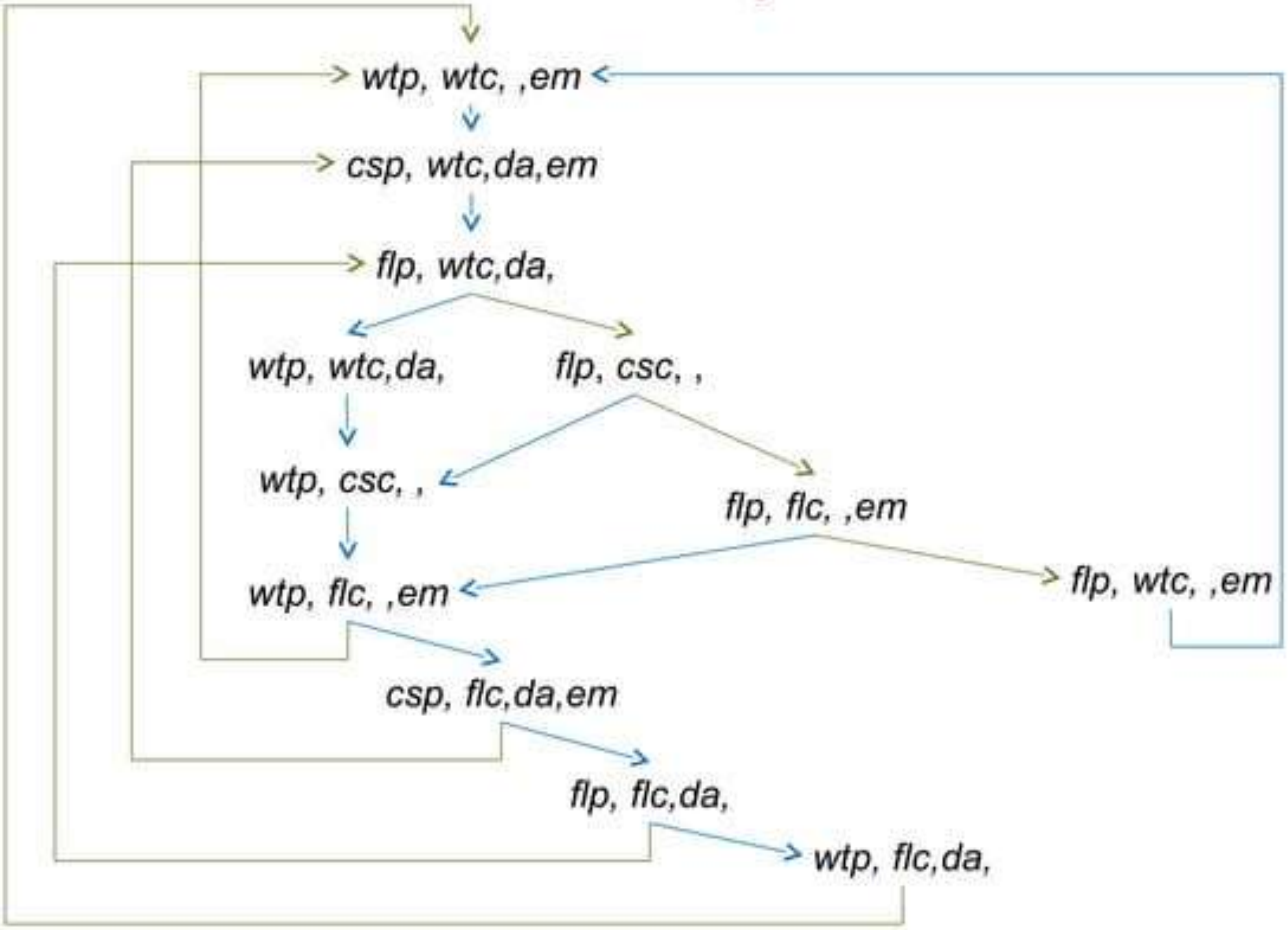
fl : flag update

da : buf has data available

em : isempty is true

e.g., (*wtp*, *csc*, ,*em*) $\in$ States

LTS for 1-buffer system



Temporal Properties for 1-buffer system

- Safety: $\Box \neg(csp \wedge csc)$
  - Producer and Consumer will never be in the CS at the same time
- Liveness: $\bullet\ (da \wedge \neg\ em)$
  - Eventually data will become available and empty flag reset
- Fairness: $\Box\bullet\ csp$
  - Producer is always given a fair chance to produce

Model Checking

- Model checking: M |= F?
- *Design & Synthesis of synchronization skeletons using branching temporal logic* [Clarke &Emerson, 81]
- *Specification & Verification of Concurrent Systems in Cesar* [Queille & Sifakis, 82]
- *Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications* [Clarke, Emerson, Sistla, TOPLAS86]

Computations of LTS

- Unfold LTS $\rightarrow$ Infinite tree of computations
  - Interleaved Semantics
  - Concurrency as non-determinism

- View of computations: Linear vs Branching
  - Linear Temporal Logic
  - Computational Tree Logic

Computational Tree Logic

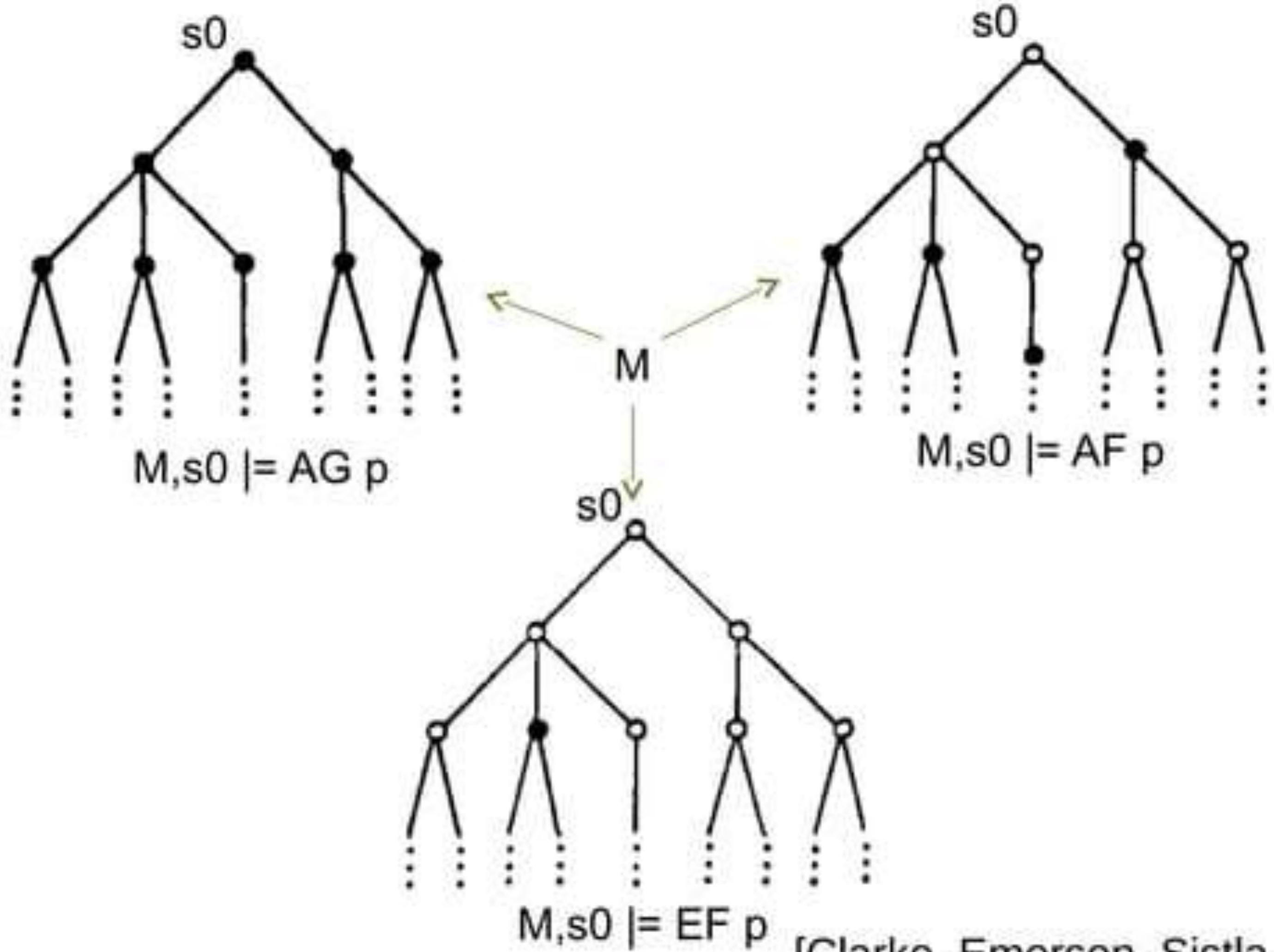
- Path quantifier
  - A: All paths (inevitably)
  - E: there Exists a path (possibly)

- Temporal operator
  - X: neXt state
  - F: some Future state (eventually)
  - G: Globally; all future states
  - U: Until

- e.g., AF: for all paths eventually, EG: for some path globally

CTL semantics

● $= p$, ○ $= \neg p$.



[Clarke, Emerson, Sistla, TOPLAS86

CTL examples

It is possible to get to a state where **started** holds, but **ready** doesn't:
$\mathrm{EF}\,(\mathsf{started} \land \neg \mathsf{ready})$.

For any state, if a **request** (of some resource) occurs, then it will eventually be
acknowledged:
$\mathrm{AG}\,(\mathsf{requested} \to \mathrm{AF}\ \mathsf{acknowledged})$.

From any state it is possible to get to a **restart** state:
$\mathrm{AG}\,(\mathrm{EF}\ \mathsf{restart})$.

A certain process is **enabled** infinitely often on every computation path:
$\mathrm{AG}\,(\mathrm{AF}\ \mathsf{enabled})$.

The lift can remain idle on the third floor with its doors closed:
$\mathrm{AG}\,(\mathsf{floor3} \land \mathsf{idle} \land \mathsf{doorclosed} \to \mathrm{EG}\,(\mathsf{floor3} \land \mathsf{idle} \land \mathsf{doorclosed}))$.

Logic in Computer Science [Huth, Ryan, 04]

Computational Tree Logic

- $\phi ::= \quad T \mid F \mid p$
  $\mid \neg\phi \mid \phi\wedge\phi \mid \phi\vee\phi \mid \phi\rightarrow\phi$
  $\mid AX\ \phi \mid EX\ \phi$
  $\mid AF\ \phi \mid EF\ \phi$
  $\mid AG\ \phi \mid EG\ \phi$
  $\mid A\ [\phi\ U\ \phi] \mid E\ [\phi\ U\ \phi]$

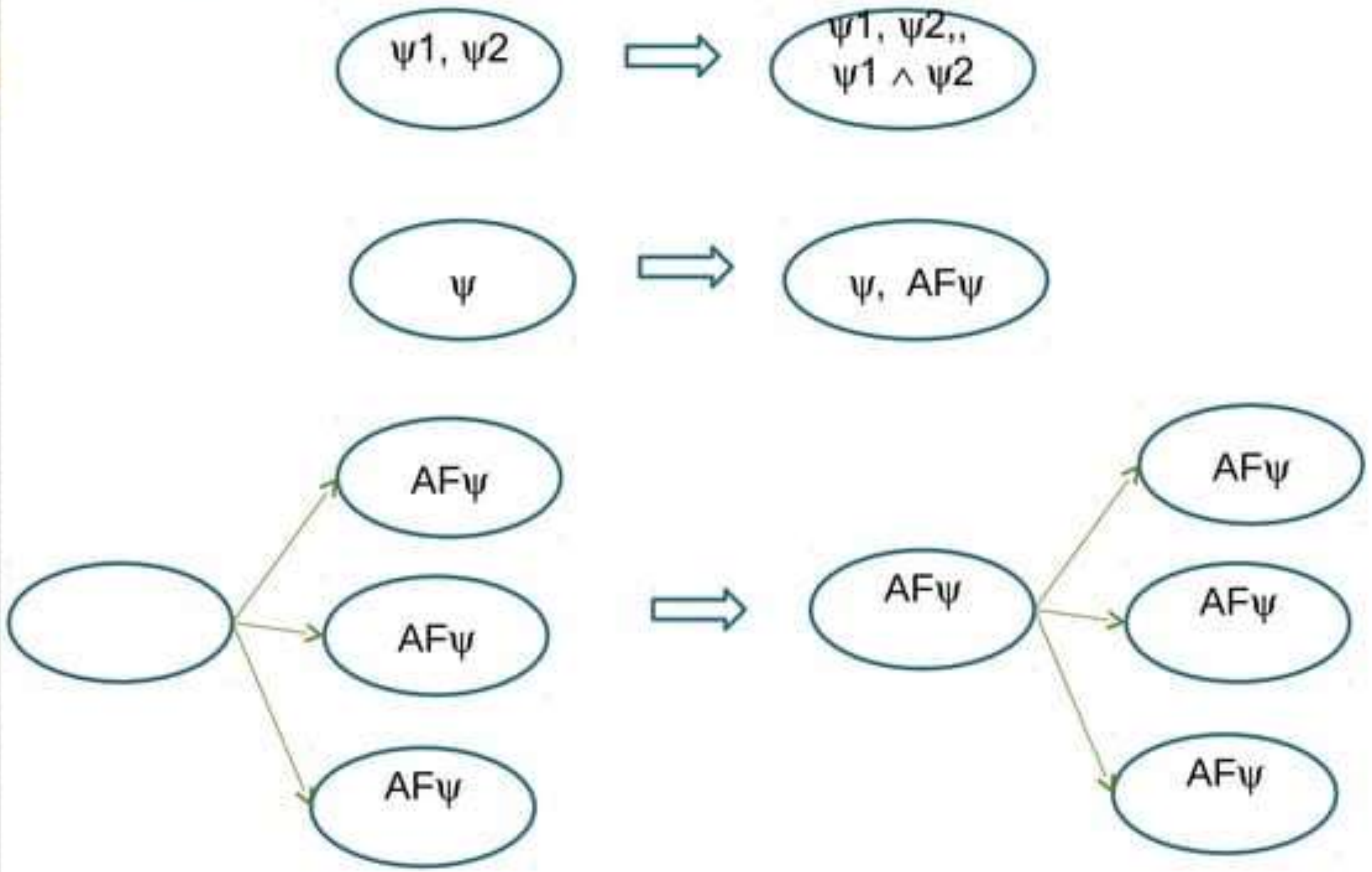
A: inevitably (along all paths)
E : possibly (there exists a path)
G: globally (always), F: in future (eventually)
X: neXt state, U: until

Model Checking

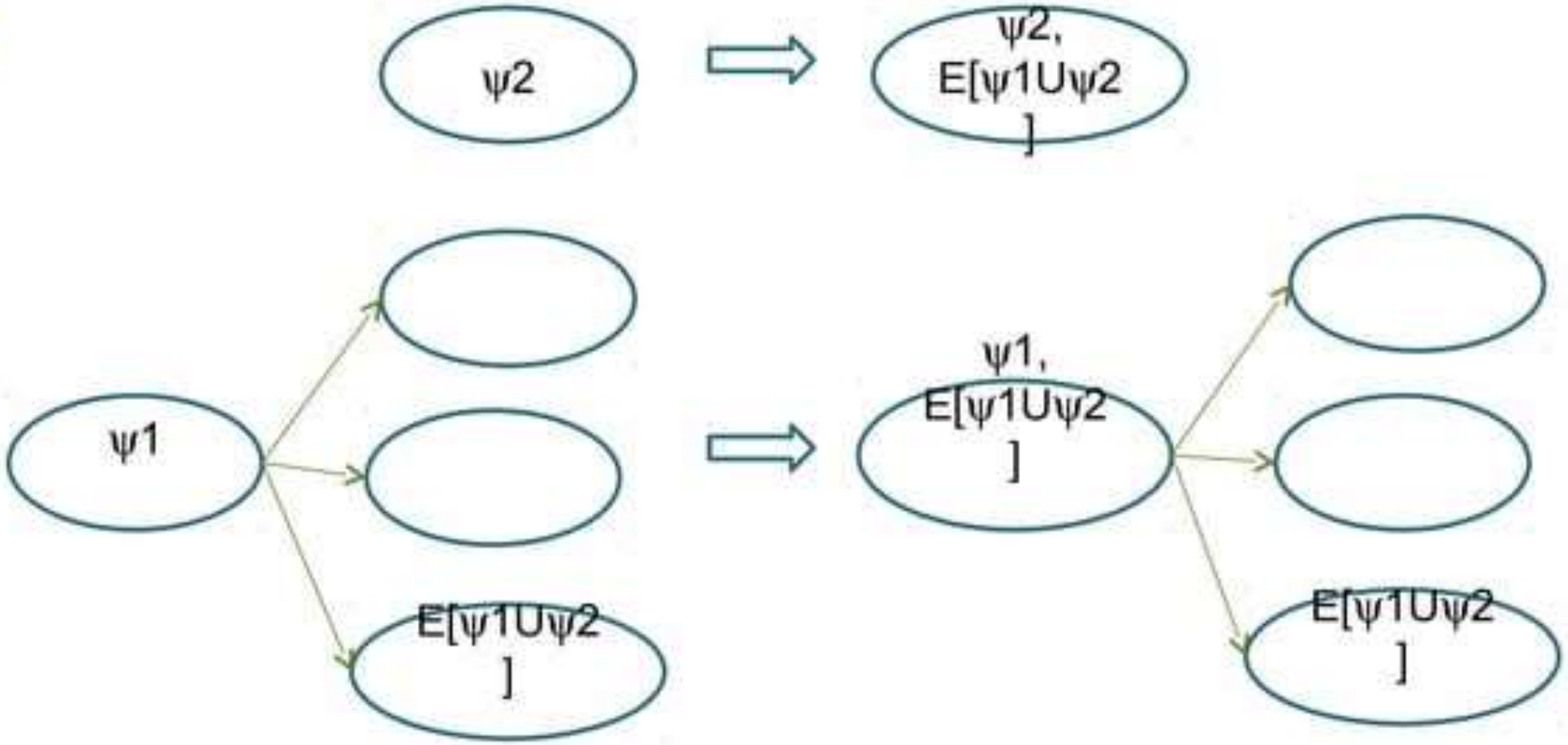
- $M \models F$?
- SAT(M, ϕ) : 2S
  - INPUT:
    - 熛CTL model $M = (S, \rightarrow, L)$
    - 熛CTL formula ϕ
  - OUTPUT:
    - 熛Set of states ($\subseteq$ S)that satisfy ϕ
  - Complexity: $O(f \,.\, |S| \,.\, (|S| + |\rightarrow|))$

Logic in Computer Science [Huth, Ryan, 04]

SAT: Conjunction & Inevitability



SAT: Possibly Until



SAT(ϕ) : 2S

```
begin
  case($\phi$)
   T          : return S
   $\bot$          : return $\varnothing$
     $\neg\psi$        : return S – SAT($\psi$)
     $\phi1\wedge\phi2$  : return SAT($\phi1$) $\cap$ SAT($\phi2$)
     $\phi1\vee\phi2$  : return SAT($\phi1$) $\cup$ SAT($\phi2$)
     $\phi1\Rightarrow\phi2$ : return SAT($\neg\phi1\vee\phi2$)
…
```

SAT (case(ϕ) cont.d)

```
    AF$\psi$        : SATAF($\psi$)
    E[$\phi$1U$\phi$2] : SATEU($\phi$1,$\phi$2)
  end case
end
```

SATAF(ϕ)

```
begin
  Y := $\emptyset$;
  repeat
    X := Y;
    Y := f($\phi$,Y);
  until X = Y
end
```

```
function f($\phi$,Y)
begin
  if Y = $\emptyset$ then
    return SAT($\phi$)
  else
    return Y $\cup$
  pre$\forall$(Y)
end
```

Fixpoint characterization

- Consider, F: 2S $\rightarrow$ 2S

- Formula are identified with their characteristic set
  - e.g. ϕ denotes set of all states where ϕ is true

- Subsets of S ($\in$ 2S) form complete lattice
  - Partial order: $\subseteq$, Join: $\cup$, Meet: $\cap$

- Knaster-Tarski theorem
  - "Monotone functions on a complete lattice possess least and greatest fixpoints": *A lattice-theoretical fixpoint theorem and its applications* [Tarski,55]

Fixpoint characterization: Eventually, Until

- $EF\phi = \mu Z.\ \phi \vee EX\ Z$
- $AF\phi = \mu Z.\ \phi \vee AX\ Z$
- $E[\psi 1 U \psi 2] = \mu Z.\ \psi 2 \vee (\psi 1 \wedge EX\ Z)$
- $A[\psi 1 U \psi 2] = \mu Z.\ \psi 2 \vee (\psi 1 \wedge AX\ Z)$

Fixpoint characterization: globally

- $AG\phi = \nu Z.\phi \wedge AX\ Z$

- $EG\phi = \nu Z.\phi \wedge EX\ Z$

LTL Model checking

- *The complexity of propositional linear temporal logics* [Sistla, Clarke, 85]
- *Checking that finite state concurrent programs satisfy their linear specification,* [Lichtenstein, Pnueli, POPL85]
- *An automata-theoretic approach to automatic program verification* [Vardi, Wolper, 86]
  - *LTL to Buchi Automata*

State explosion

- The state explosion problem [Clarke, Grumberg, 87]
    - Number of concurrent processes
    - Number of variables
- Partial-order reduction
    - *An Introduction to Trace Theory* [Mazurkiewicz,95]
- *Symbolic Model Checking: 1020 states and beyond* [Burch, Clarke, McMillan, Dill, Hwang, 92]
    - OBDD: *Ordered Binary Decision Diagrams* [Bryant,86]
    - μ-Calculus: *Finiteness is μ-ineffable* [Park,74]
- Abstraction
- Combining theorem-proving & model checking
- On-the-fly model checking

Some Tools

- SPIN/Promela
  - http://spinroot.com
- Java Pathfinder
  - http://javapathfinder.sourceforge.net/
- NuSMV
  - http://nusmv.irst.itc.it/

Thank You

Have a great day!