Artificial Intelligence: Foundations & Applications

Temporal Logic and Applications

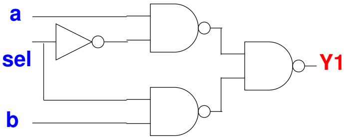
Prof. Partha P. Chakrabarti & Arijit Mondal
Indian Institute of Technology Kharagpur

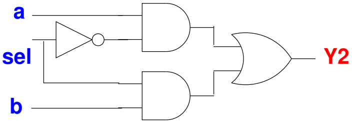
Introduction

- **Specifying the system functionality requires notion of time**
- **Propositional logic cannot be applied directly**

- **Example**
  - **b: brakes are pressed, a: accelerator is pressed, s: car stops, d: car slows down**
    - **When brakes are pressed, the car slows down in the next instant**
    - **When no accelerator is pressed then after a while the car continuously slows down**
    - **When brakes are constantly kept pressed and there is no accelerator pressed, the car slows down and eventually stops.**

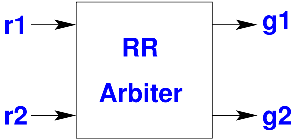
Verification of Combinational Circuits





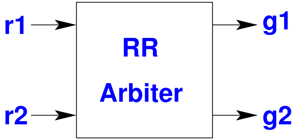
- **Are Y1 and Y2 equivalent?**
  - $Y1 = \overline{\overline{(a \wedge \neg sel)} \wedge \overline{(b \wedge sel)}}$
  - $Y2 = (a \wedge \neg sel) \vee (b \wedge sel)$
- **Canonical structure of Binary Decision Diagram can be exploited to compare Boolean functions like Y1 & Y2**

Verification of Sequential Circuits



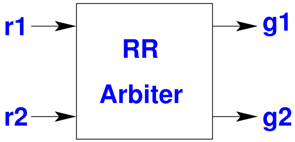
- Properties span across cycle boundaries
- Example: Two way round robin arbiter
  - If the request bit r1 is true in a cycle then the grant bit g1 has to be true within the next two clock cycles

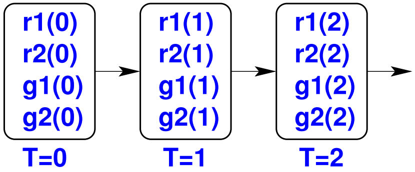
Verification of Sequential Circuits



- Properties span across cycle boundaries
- Example: Two way round robin arbiter
  - If the request bit r1 is true in a cycle then the grant bit g1 has to be true within the next two clock cycles
- Need temporal logic to specify the behavior

Verification of Sequential Circuits



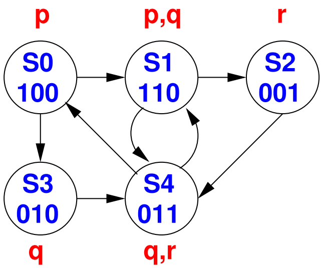


- *If the request bit r1 is true in a cycle then the grant bit g1 has to be true within the next two clock cycles*
- $\forall t[r1(t) \longrightarrow g1(t+1) \vee g1(t+2)]$
- **In propositional temporal logic time (t) is implicit**
  - **always r1 $\longrightarrow$ (next g1) $\vee$ (next next g1)**

Temporal logic

- The truth value of a temporal logic is defined with respect to a model.
- Temporal logic formula is not statically true or false in a model.
- The models of temporal logic contain several states and a formula can be true in some states and false in others.
- Example:
  - I am *always* happy.
  - I will *eventually* be happy.
  - I will be happy *until* I do something wrong.
  - I am happy.

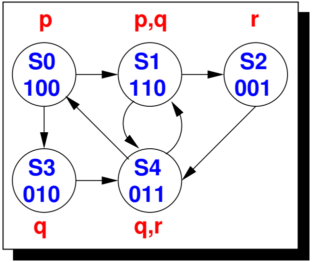
Kripke Structure



- $M = (AP, S, S_0, T, L)$
  - AP – **Set of atomic proposition**
  - S – **Set of states**
  - S_0 – **Set of initial states**
  - T – **Total transition relation ($T \subseteq S \times S$)**
  - L – **Labeling function ($S \longrightarrow 2^{AP}$)**

Path

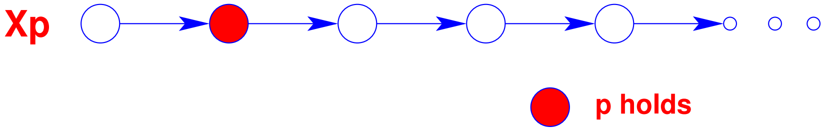
- A path $\pi = s_0, s_1, \ldots$ in a Kripke structure is a sequence of states such that $\forall i, (s_i, s_{i+1}) \in T$
- Sample paths
  - $S0, S1, S2, S4, S1, \ldots$
  - $S0, S3, S4, S0, \ldots$
  - $S0, S1, S4, S1, \ldots$
  - $\pi = \underbrace{s_0, s_1, \ldots, s_k}_{\text{prefix of } \pi_k \text{ in } \pi}, s_{k+1} \ldots$
  - $\pi = s_0, s_1, \ldots, \underbrace{s_k, s_{k+1} \ldots}_{\text{suffix of } \pi^k \text{ in } \pi}$



Temporal operators

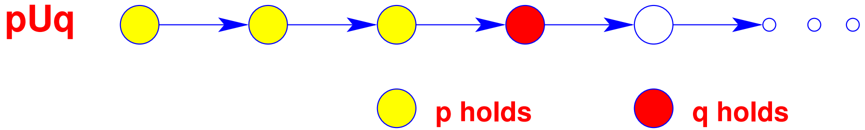
- Two fundamental path operators
  - Next operator
    - Xp – property p holds in the next state
  - Until operator
    - $p\ U\ q$ – property p holds in all states upto the state where property q holds
- Derived operators
  - Eventual/Future operator
    - Fp – property p holds eventually (in some future states)
  - Always/Globally operator
    - Gp – property p holds always (at all states)
- All these operators are interpreted over the paths in Kripke structure under consideration
- All Boolean operators are supported by the temporal logics

The *next* operator (*X*)



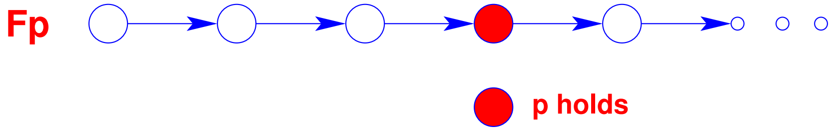
- p holds in the next state of the path
- Formally
  - $\pi \models Xp$ **iff** $\pi^1 \models p$

The *until* operator (*U*)



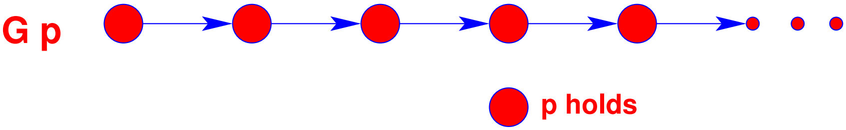
- q holds eventually and p holds until q holds
- Formally
  - $\pi \models p\, U\, q$ **iff** $\exists k$ **such that** $\pi^k \models q$ **and** $\forall j, 0 \leq j < k$ **we have** $\pi^j \models p$

The *eventual* operator (*F*)



- p holds eventually (in future)
- Formally
  - $\pi \models Fp$ iff $\exists k$ such that $\pi^k \models p$
  - This can be written as $true\ U\ p$

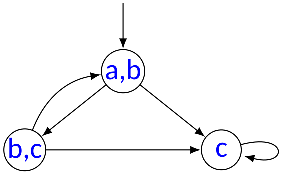
The *always* operator (G)

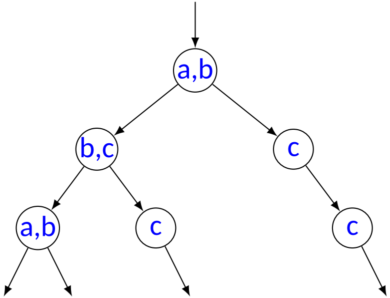


- p holds always (globally)
- Formally
  - $\pi \models Gp$ iff $\forall k$ we have $\pi^k \models p$
  - This can be written as $\neg(true\, U\, \neg p)$ or $\neg F \neg p$

Branching Time Logic

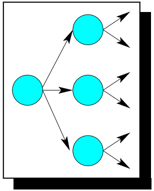
- **Interpreted over computation tree**



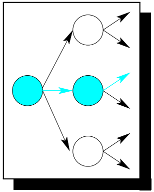


Path Quantifier

- **A: “For all paths …”**

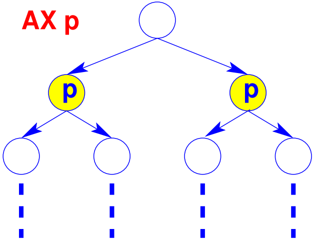


- **E: “There exists a path …”**

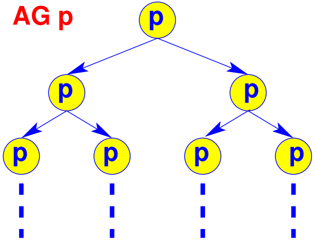


Universal Path Quantification

AX p



AG p

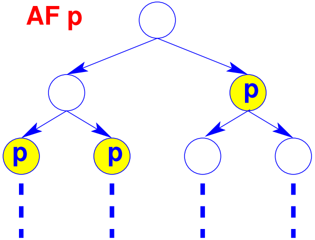


In all the next states p holds.

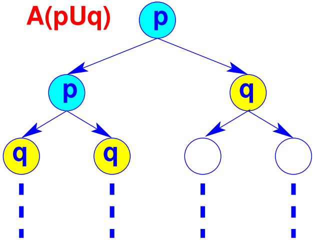
Along all the paths p holds forever.

Universal Path Quantification

AF p



A(pUq)

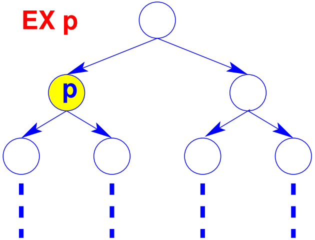


Along all the paths p holds eventually.

Along all the paths p holds until q holds.

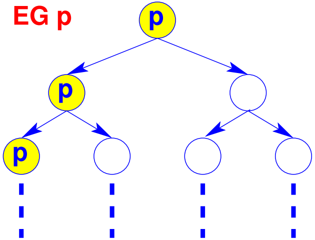
Existential Path Quantification

EX p



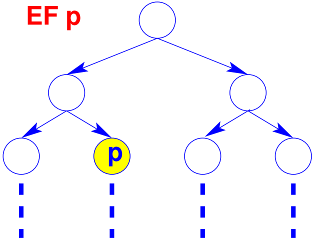
There exists a next state where p holds.

EG p

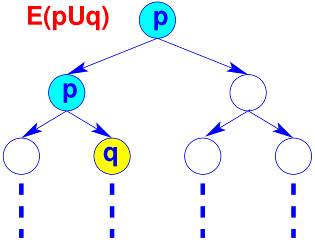


there exists a path along which p holds forever.

Existential Path Quantification



There exists a path along which p holds eventually.



There exists a path along which p holds until q holds.

Duality between Always & Eventual operators

- $Gp = p \wedge (\textbf{next } p) \wedge (\textbf{next next } p) \wedge (\textbf{next next next } p) \ldots$

  $= \neg(\neg(p \wedge (\textbf{next } p) \wedge (\textbf{next next } p) \wedge (\textbf{next next next } p) \wedge \ldots))$

  applying De Morgan's law

  $= \neg(\neg p \vee (\textbf{next } \neg p) \vee (\textbf{next next } \neg p) \vee (\textbf{next next next } \neg p) \vee \ldots)$

  $= \neg(F\neg p)$

- **Therefore we have**
  - $Gp = \neg F \neg p$
  - $Fp = \neg G \neg p$

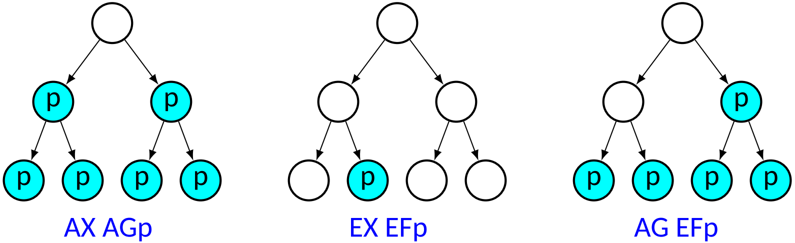
Computation Tree Logic (CTL)

- **Syntax:**
  - **Given a set of Atomic Propositions (AP):**
    - **All Boolean formulas of over AP are CTL properties**
    - **If f and g are CTL properties then so are $\neg f$, AXf, $A(f\,U\,g)$, EXf and $E(f\,U\,g)$,**
  - **Properties like AFp, AGp, EGp, EFp can be derived from the above**

- **Semantics:**
  - **The property Af is true at a state s of the Kripke structure iff the path property f holds on all paths starting from s**
  - **The property Ef is true at a state s of the Kripke structure iff the path property f holds on some path starting from s**

Nested properties in CTL

- *AX AGp*
  - **From all the next state *p* holds forever along all paths**
- *EX EFp*
  - **There exist a next state from where there exist a path to a state where *p* holds**
- *AG EFp*
  - **From any state there exist a path to a state where *p* holds**



Thank you!