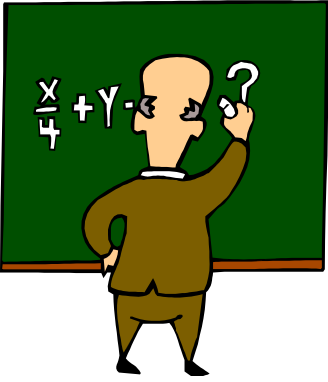


Test de Systèmes Informatiques

Partie IV : Sequence Testing (Based on Finite State Machines)

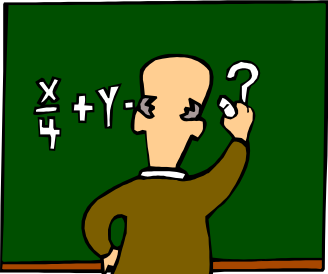
Burkhart Wolff, LRI
(basé sur Marie-Claude Gaudel)



Back in history: FSM-based testing

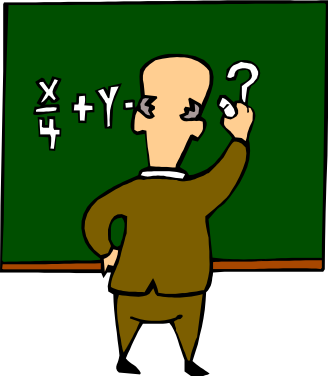


- Originally invented in the sixties for testing circuits, *thus there is a finite number of states*
- First applied to software by Chow in 78 (ref [20])
- Big corpus of knowledge, with a lot of variants on the kind of considered FSM

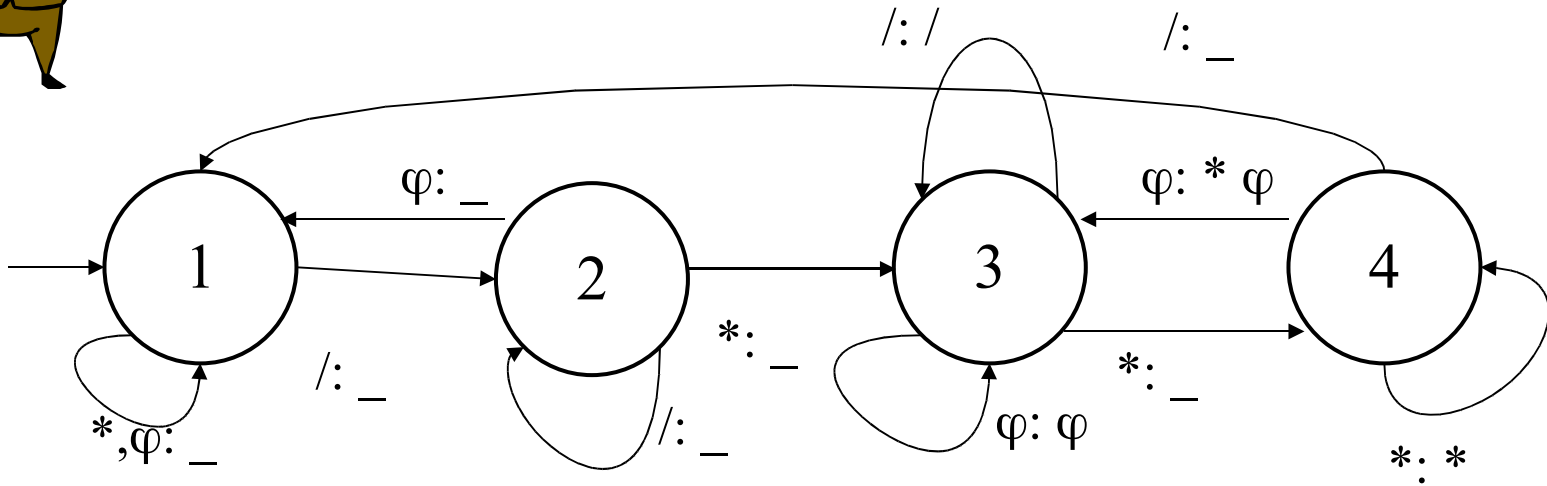


What is an FSM?

- S : finite set of states,
 I : input alphabet,
 O : output alphabet
- T : finite set of transitions:
 - $s \xrightarrow{x:y} s' \in T, \quad s, s' \in S, x \in I, y \in O^*$
 - **Notations:** $\lambda(s,x)=y, \quad \lambda^*(s,w)=w', \quad w \in I^*, w' \in O^*$
- ***Equivalent states*** : $\forall w, \lambda^*(s,w) = \lambda^*(s',w)$
- Here, the considered FSM are: deterministic, complete ($\forall s \in S, \forall x \in I, \exists s \xrightarrow{x:y} s' \in T$), minimal (*no equivalent states*), and all states are reachable



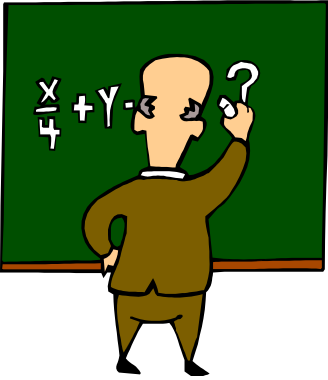
The same example



φ = any character but * and /

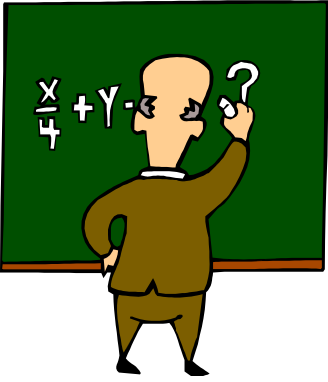
This is not a comment /* **all that / * is ** a comment** */ this is no more a comment.

This FSM removes from the input text all that is not a comment



Formalities

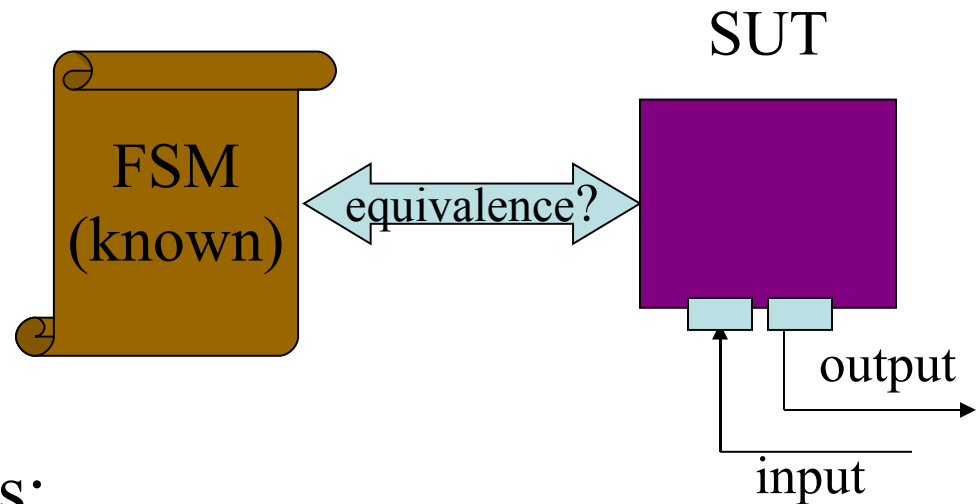
- A FSM is a “regular transducer”
- It defines a function from I^* into O^*
- *There is no memory*: given an input, the output depends only on the current state and not on the way it has been reached.



Revising history

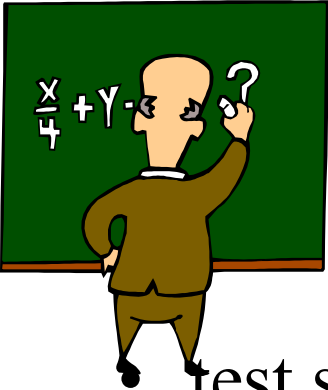


- System under test
 - unknown, but...



- Testability Hypothesis:
 - *the SUT behaves like some (unknown) FSM with the same* number of states as the description*
 - Whatever the trace leading to some state s , the execution of transition $s \xrightarrow{x:y} s'$ has the same effect (output, change of state)

*or more but this number is known



Back in history: control and observation

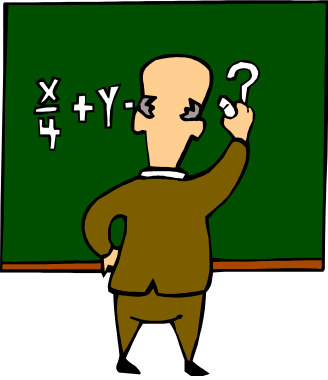


- test strategy: *transition coverage*

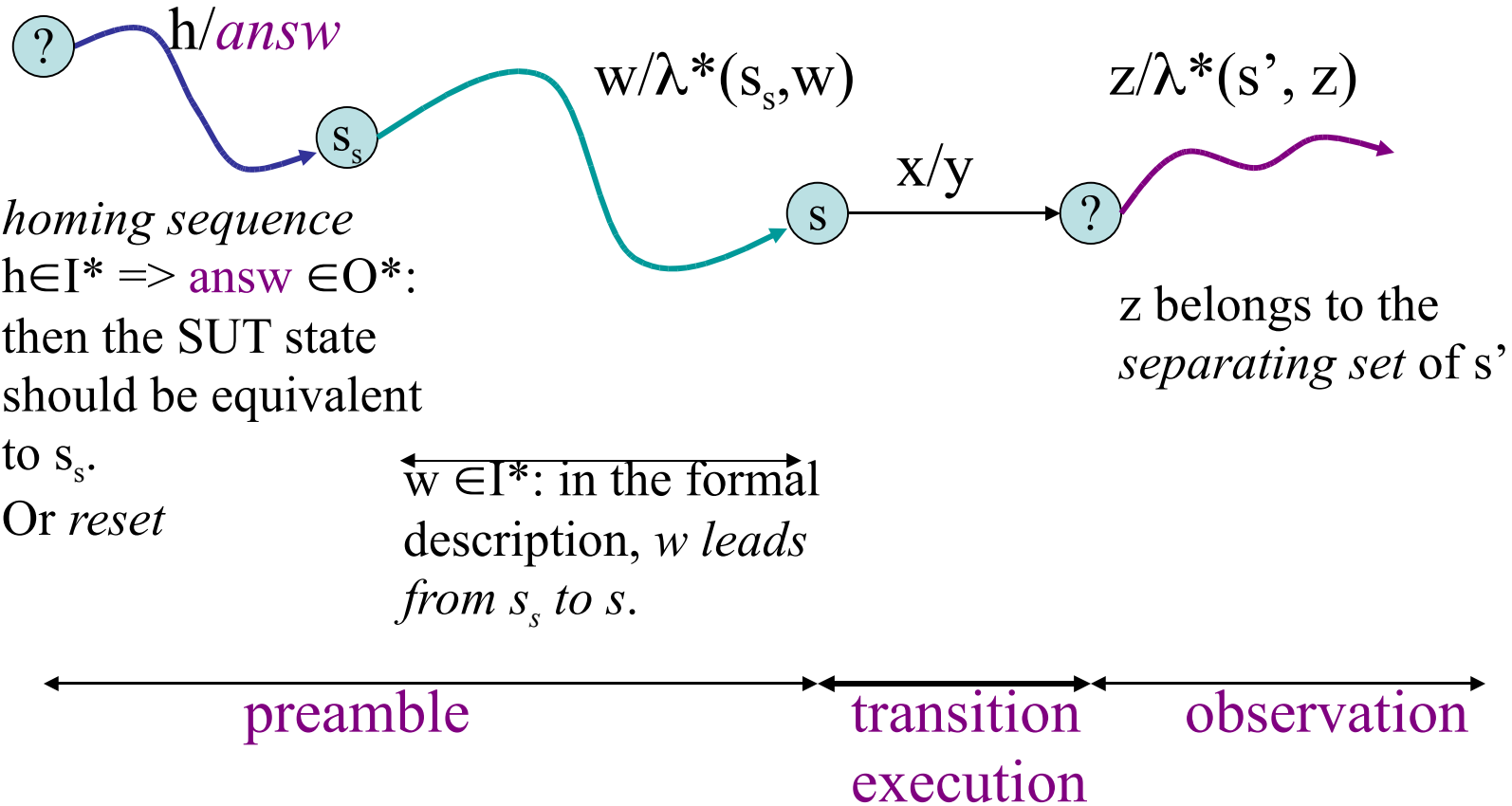
$$s \xrightarrow{x:y} s'$$

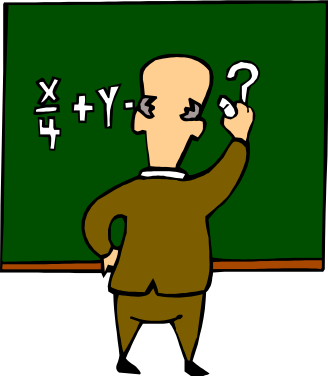
- Questions

- **control**: how to put the SUT into a state equivalent to s ?
 - solution 1: if there is an **initial state**, perform a “**reliable reset**”, and then some adequate input sequence
 - solution 2: “**homing sequence**”, and then some adequate input sequence
- **observation**: how to check that after receiving x and issuing y , the SUT is in a state equivalent to s ?
 - “**separating family**” : collection $\{Z_i\}_{i=1,\dots,n}$ of sets of input sequences whose output sequences make it possible to distinguish s_i from any other state

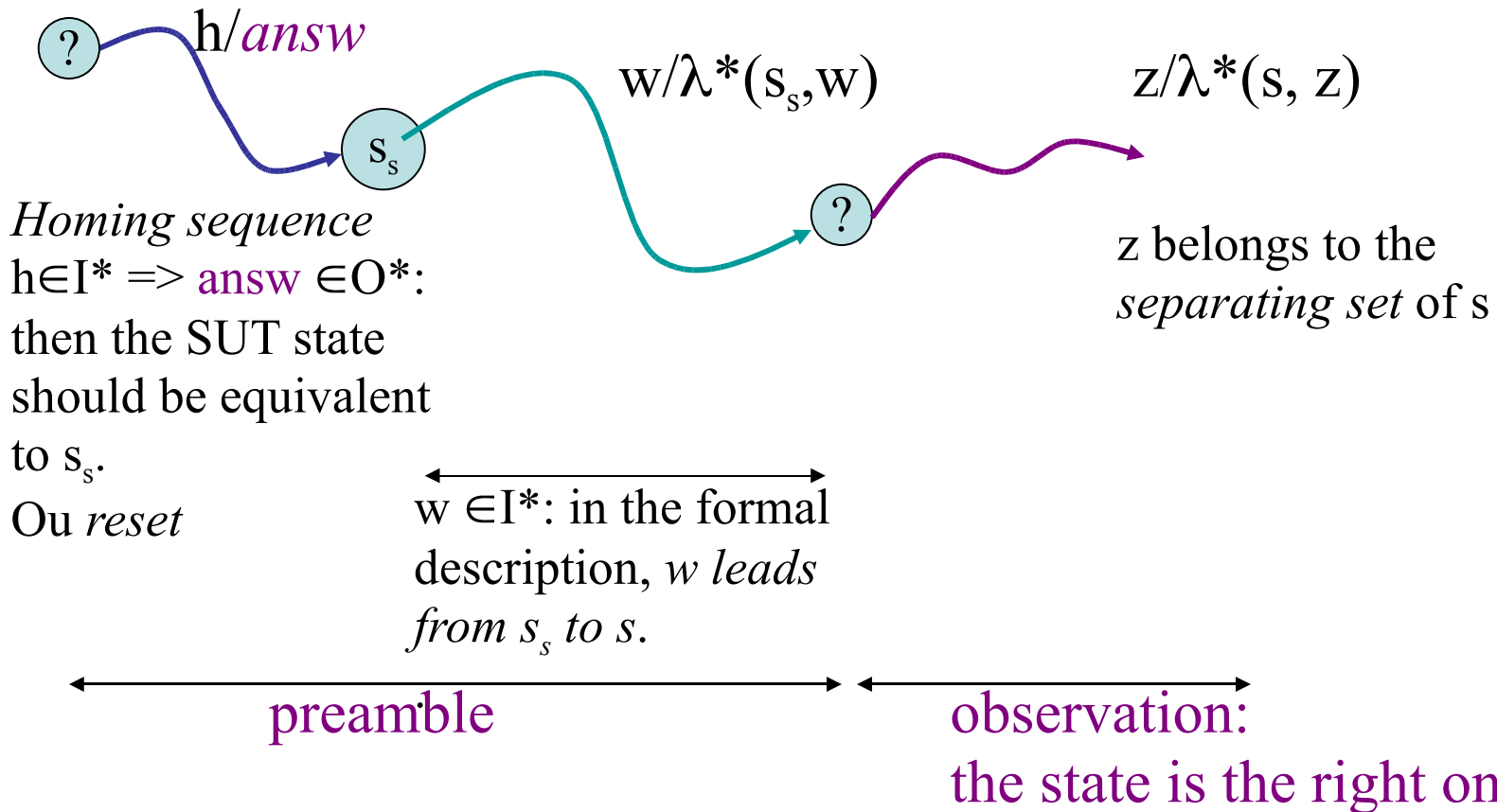


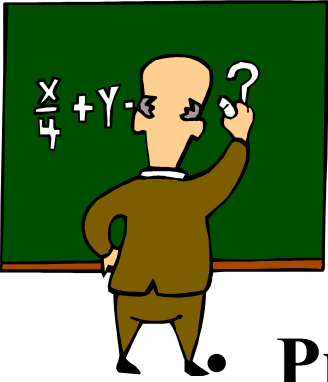
One of the tests for $s \sim_{x/y} s'$





One of the tests of the preamble of $s \xrightarrow{-x/y-} s'$

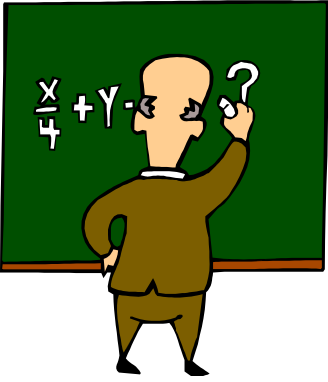




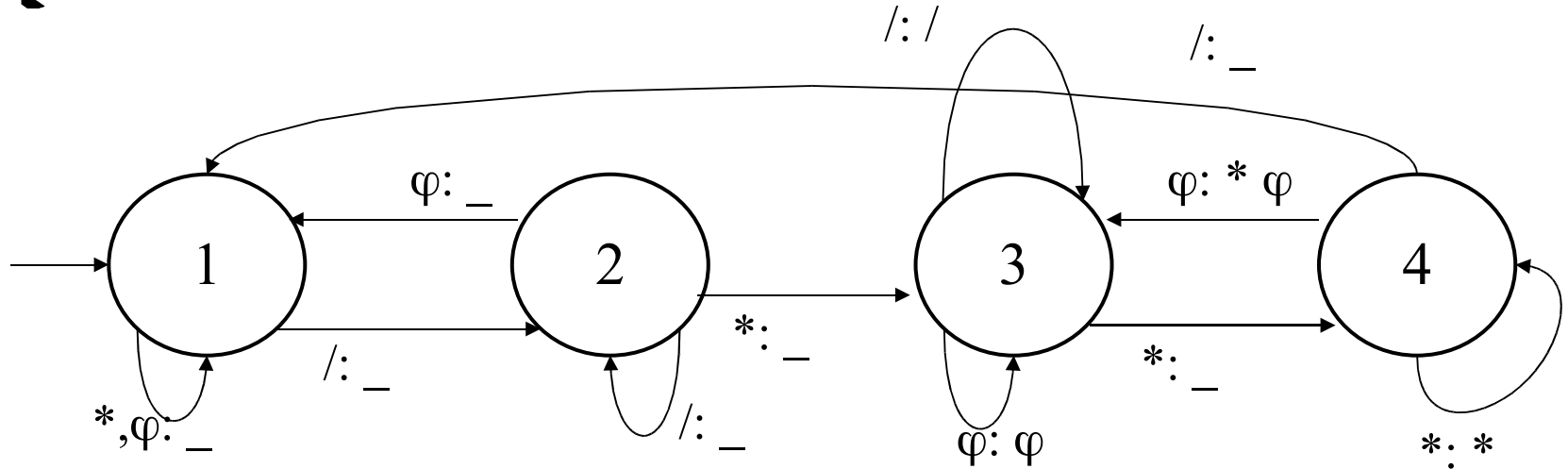
Back in history [Chow 78]



- **Preliminary theorem:** every such FSM has a *characterizing set* $W = \{w_1, \dots, w_m\} \subseteq I^+$, which allows to distinguish the states
 - $s \neq s' \Rightarrow \exists w_i \in W$ such that $\lambda^*(s, w_i) \neq \lambda^*(s', w_i)$
- **Test sequences:** $p.z$, where $p \in P$, $z \in Z$
 - P : for every transition $s \xrightarrow{x:y} s'$, there are two sequences in P , **p** and **$p.x$** , such that p leads from the initial state to s
 - $Z = W$ (or W^k if there are k more states)
 - i.e., coverage of transitions, with observation of the origin and destination states
- The FSM has an initial state, and the SUT provides a *reliable reset*



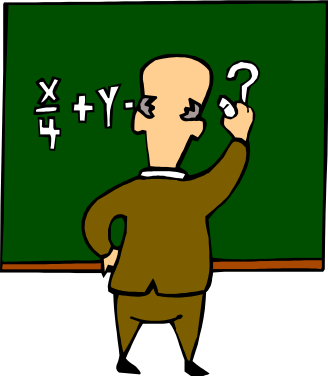
“W” for the example



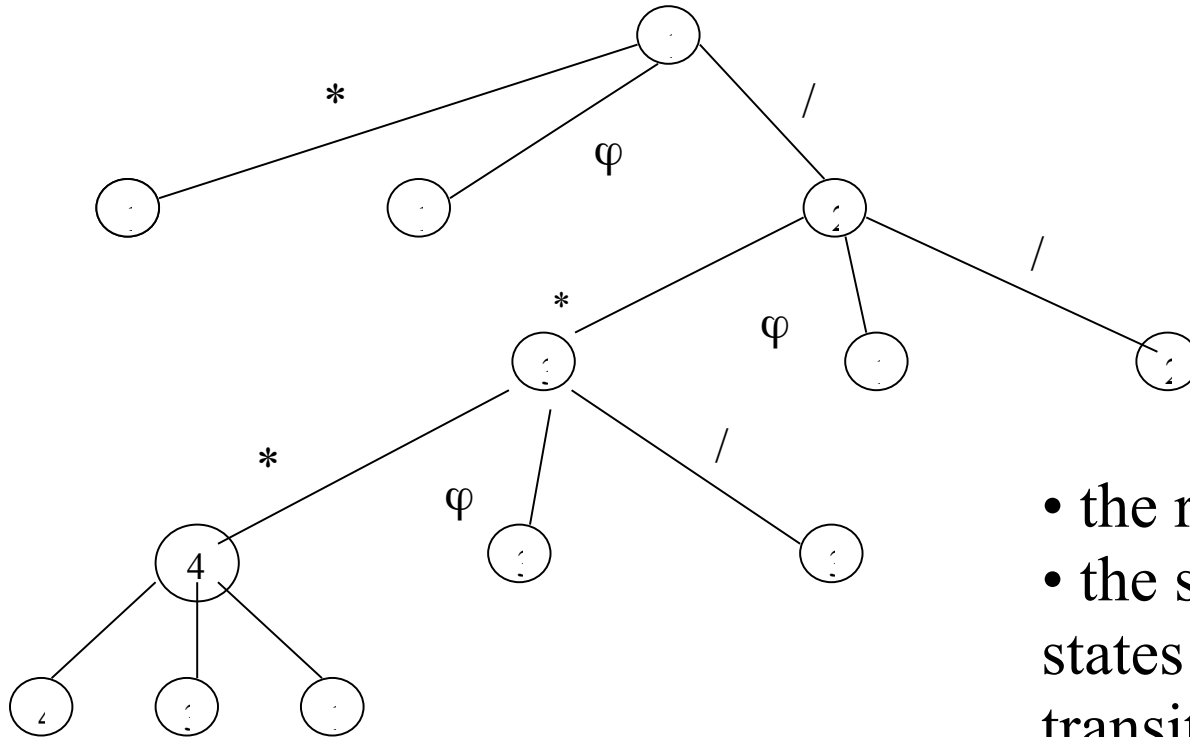
Characterizing set : $\mathbf{W} = \{\ast\varphi\}$

Note: it is a destructive observation

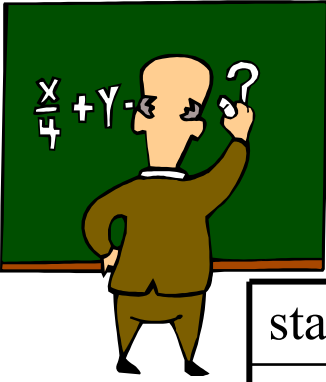
state	$\ast\varphi$
1	$_$
2	φ
3	$\ast\varphi$
4	$\ast\ast\varphi$



Construction of some P (covering tree)



- the root is the initial state
- the sons of a node are those states reachable by some transition
- if a state is already in the tree it is terminal

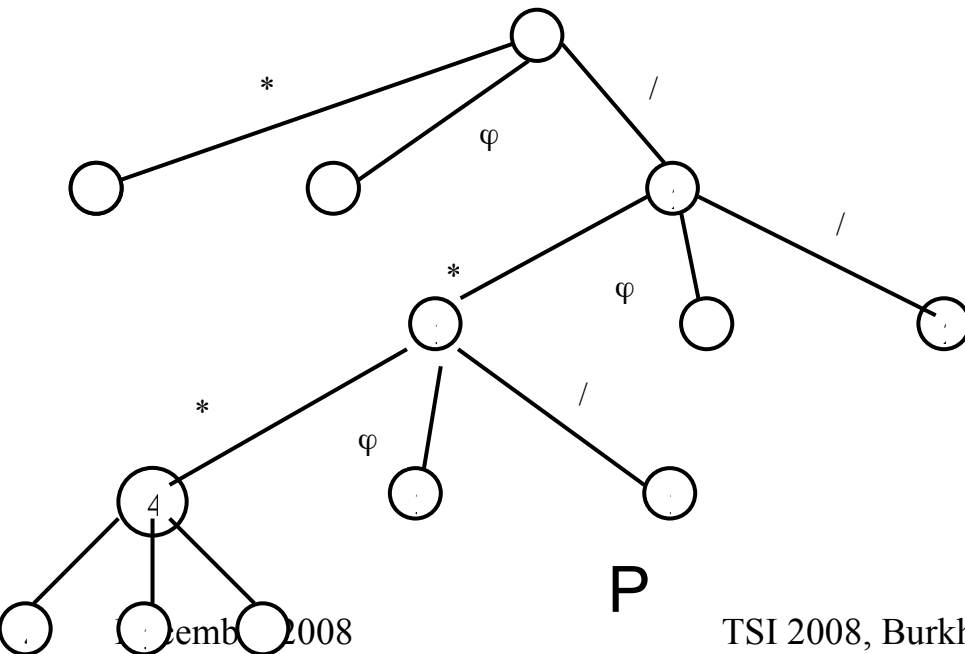


Tests and expected results



state	* φ
1	—
2	φ
3	* φ
4	** φ

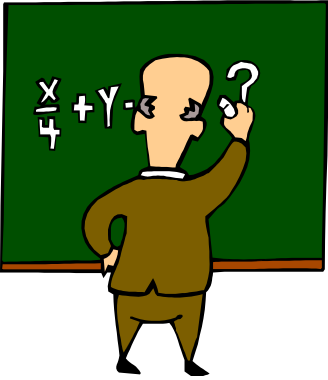
W



P

<u>*φ</u>	nothing
** <u>φ</u>	nothing
φ <u>*φ</u>	nothing
/ <u>*φ</u>	φ
// <u>*φ</u>	φ
/ φ <u>*φ</u>	nothing
/ <u>**φ</u>	* φ

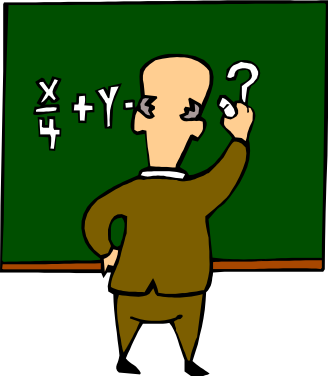
/*/ <u>*φ</u>	/* φ
/* φ <u>*φ</u>	φ * φ
/*** <u>*φ</u>	** φ
/**** <u>*φ</u>	*** φ
/*** φ <u>*φ</u>	* φ * φ
/***/ <u>*φ</u>	nothing



Exhaustivity of P.Z

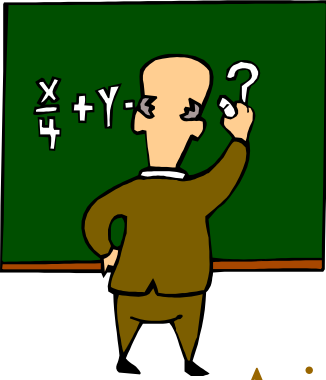


- Let A, B , two FSM with the same I and O ;
 - Let $V \subseteq I^*$;
 - s_A is a state of A , s_B is a state of B ;
 - s_A and s_B are V -equivalent iff
 - $\forall w \in V, \lambda^*(s_A, w) = \lambda^*(s_B, w)$
- A and B are V -equivalent \Leftrightarrow their initial states are V -equivalent
- **Chow's theorem:**
 A and B are equivalent \Leftrightarrow **A and B are P.Z-equivalent**



I like it!
It was the first “extrapolation”
theorem applicable to software
testing

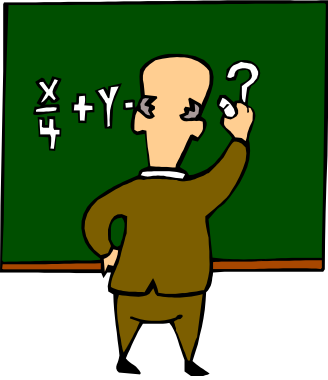




Application to testing



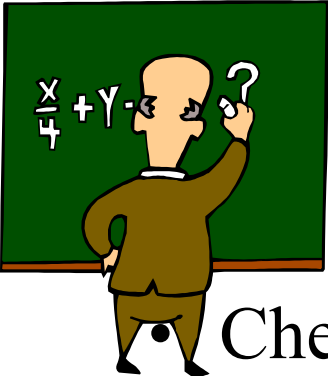
- A is a description/specification/model
- B is a system under test (SUT) that behaves like some FSM
 - One knows that I and O are the same sets
 - One knows that B has the same number of states than A, or a known number of additional states
 - There is a reliable reset of B
 - It is all that is known about B
- From A one builds P.Z
 - One tests B against the sequences of P.Z
 - If all the output results are the same as for A, B is equivalent to A



Testability Hypotheses and all that...



- Hypotheses:
 - *the SUT behaves like some FSM with the same* number of states as the specification*
 - *the SUT provides a reliable reset*
- Under these testability hypotheses, the success of a test set P.Z ensures equivalence of the SUT and the specification FSM
- Here the satisfaction relation is *equivalence*
- P.Z is exhaustive/these hypotheses and this relation
- *the SUT behaves like some FSM with a known nb of states and it provides a reliable reset*
 \Rightarrow (SUT passes P.Z \Leftrightarrow SUT equiv SP)



Before or after Chow

Checking sequence:

- covers every transition and its separating set; *distinguishes the description FSM from any other FSM with the same number of states, no need of reset*
- Finite, *but may be exponential/nb of states...* in length, in construction
- Exhaustivity
 - Transition (+ separating set) coverage
- Control
 - homing sequence, or reliable reset
 - Non-determinism \Rightarrow adaptive test sequences
- Observation
 - distinguishing sets, UIO, or variants (plenty of them!)