

CIS 673: Lecture 10: Oct 9  
Theory of Temporal Logics

- How do we compare two logics? e.g. fragments of STL
- When do two states satisfy the same STL formulas?
- Can we use quotients to model check STL?
- Distinguishing Power: a formula distinguishes two states if true in one, and false in another
- Expressive Power: equivalence of formulas

## Distinguishing Power of Logics

- Each logic  $\Phi$  induces an equivalence  $\simeq^\Phi$ : The two states  $s$  and  $t$  of  $K$  are  $\Phi$ -equivalent if for all  $\Phi$ -formulas  $\phi$ ,  $s \models_K \phi$  iff  $t \models_K \phi$ .
- To compare two logics, compare the induced equivalences.
- Any state equivalence that is as distinguishing as  $\simeq^\Phi$  is an abstract semantics for  $\Phi$ .
- The equivalence  $\simeq^\Phi$  is called the fully-abstract semantics for  $\Phi$ .

## STL and Bisimilarity

- Thm: Bisimilarity is a fully-abstract semantics of STL
  - Two bisimilar states satisfy the same set of STL formulas
  - If two state are not bisimilar, then some STL formula distinguishes them.
- Fragments of STL:
  - $\text{STL}^\circ$ : only next formulas (no until)
  - $\text{STL}^u$ : only until formulas (no next)
- If two state are not bisimilar, then some  $\text{STL}^\circ$  formula distinguishes them.
- STL and  $\text{STL}^\circ$  have same distinguishing power

## Application to Model Checking

- To solve STL model checking problem  $(K, \phi)$ :
  - Compute bisimilarity partition  $\simeq^B$  by solving partition refinement problem  $(K, \approx)$
  - Check  $\phi$  on the quotient of  $K$  wrt  $\simeq^B$
- General stratgey: for a logic  $\Phi$ , compute a quotient with an abstract semantics  $\simeq$ , and solve model checking on quotients
- Subtlety: Logic should admit abstraction
  - For every  $\phi$ , and every equivalence  $\simeq$  as distinguishing as  $\simeq^\Phi$ , characteristic region  $\llbracket \phi \rrbracket_K$  should equal  $\cup \llbracket \phi \rrbracket_{K/\simeq}$ .
  - All reasonable logics like STL admit abstraction

## Stuttering

- A reactive module stutters when its observable state stays unchanged.
- Perhaps number of rounds for which the module stutters is not important (e.g. asynchronous modules)
- Motivation: Perhaps we can get a coarser equivalence if we ignore the stuttering
- Stutter Closure of an Observation Structure  $K$ :
  - let  $s \rightarrow^S t$  if there is an source- $s$  trajectory  $\bar{s}_{0..m}$  with sink  $t$  such that for all  $0 \leq i < m$ ,  $\langle\langle s_i \rangle\rangle = \langle\langle s \rangle\rangle$
  - The stutter closure  $K^S$  is obtained by replacing  $\rightarrow$  with  $\rightarrow^S$ .

## Stutter Insensitive Equivalences

- For a state equivalence  $\simeq$ , define its stutter-closure as the equivalence  $\cong$ :
  - Two states  $s$  and  $t$  of  $K$  are  $\cong$ -equivalent if they are  $\simeq$ -equivalent in  $K^S$
- An equivalence is called stutter-insensitive if it equals its stutter closure
- Bisimilarity is not stutter insensitive, its stutter closure is called weak bisimilarity
- Stutter closure can make more states equivalent: bisimilarity is more distinguishing than weak bisimilarity

## Stutter Insensitive Logics

- A logic  $\Phi$  is stutter insensitive if the induced equivalence  $\simeq^\Phi$  is.
- STL is not stutter insensitive: next-formulas can distinguish between weakly-bisimilar states
- Are until-formulas stutter insensitive?
- Thm: Weak bisimilarity is a fully abstract semantics for  $\text{STL}^{\mathcal{U}}$ 
  - $\text{STL}^{\mathcal{U}}$  is stutter insensitive
  - To check  $\text{STL}^{\mathcal{U}}$  formulas, we can take quotients wrt weak bisimilarity (potentially fewer classes)
- $\text{STL}^{\circ}$  is more distinguishing than  $\text{STL}^{\mathcal{U}}$ .

## Expressive Power

- Can until be expressed using next and eventually?
- The logic  $\Phi$  is as expressive as the logic  $\Psi$  if for every formula  $\phi$  of  $\Psi$ , there exists a formula  $\psi$  of  $\Phi$  such that for every  $K$ , the characteristic regions  $\llbracket \phi \rrbracket_K$  and  $\llbracket \psi \rrbracket_K$  are identical.
- The logic  $\Phi$  is more expressive than the logic  $\Psi$  if  $\Phi$  is as expressive as  $\Psi$ , but not vice versa.
- Greater distinguishing power implies greater expressive power, but logics with different distinguishing powers can be equally expressive.

## Expressive Power of Fragments

- Thm1: The logic  $\text{STL}^\circ$  is not as expressive as the logic  $\text{STL}^\mathcal{U}$ .
  - The formula  $\exists \diamond p$  is not equivalent to any formula of  $\text{STL}^\circ$ .
- Thm2: The logic  $\text{STL}^\mathcal{U}$  is not as expressive as the logic  $\text{STL}^\circ$ .
  - The formula  $\exists \bigcirc p$  is not equivalent to any formula of  $\text{STL}^\mathcal{U}$ .
- Even though  $\bigcirc$  is sufficient to distinguish non-bisimilar states, it cannot express until.

## Automata theoretic approach

- Use automata instead of temporal logic, to specify requirements
- Automata-theoretic verification:
  - Module is viewed as a generator of a language over strings of observations
  - Specification is an automaton that accepts good behaviors
  - Verification problem is to check if every string generated by module is accepted by specification
- Theory of formal languages provides conceptual framework
- Sample tool: COSPAN at Bell Labs (FormalCheck)
- For now, automata over finite strings

## Traces

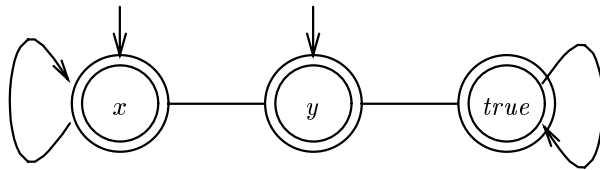
- Execution of a transition graph  $G$  results in a trajectory of  $G$ , which is a finite sequence of states.
- Trace of an observation structure: sequence of observations corresponding trajectories of underlying graph
- $L_K(s)$ : the set of source- $s$  traces of  $K$ , a language over the set  $A$  of observations
- Language  $L_K$  of  $K$ : set of initialized traces
- Language  $L_P$  of a module  $P$ : set of initialized traces of  $K_P$
- Languages of observations structures are always prefix-closed

## Automata

- An automaton  $M$  consists of
  - an observation structure  $K$
  - accepting region: a region  $\sigma^A$  of  $K$
- A trajectory is accepted if it ends in an accepting state
- Language  $L_M$  of the automaton  $M$ : set of traces corresponding to initialized accepting trajectories
- Like standard automata
  - Instead of labeling edges, labels on states
  - Languages do not contain the empty word
- Observation structure  $K$  can be viewed as an automaton, where every state is accepting

## Automata as Specifications

- Observations of a specification automaton are expressions over variables, rather than values to variables
- Motivation: Compact specifications
- Automaton corresponding to STL formula  $x \forall \mathcal{W} y$ :  
( $x$  and  $y$  can be arbitrary predicates)



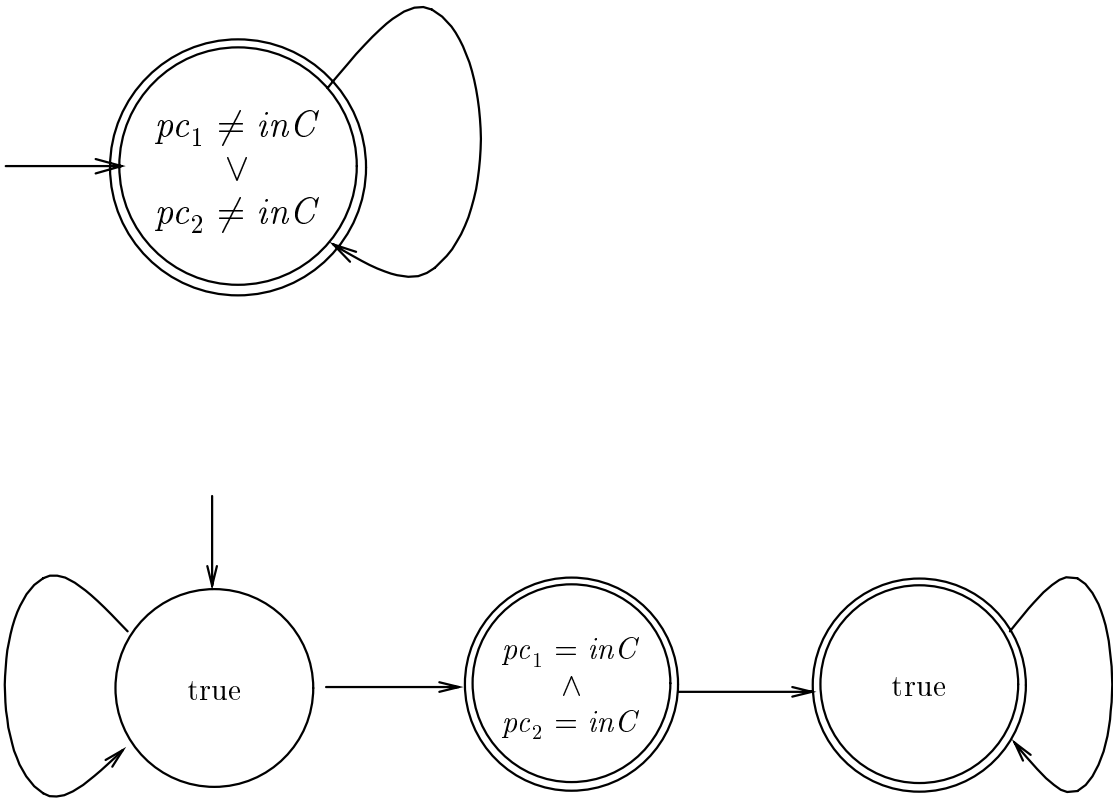
## Automaton Logic: SAL

- Syntax: Boolean combinations of specification automata

$$\phi ::= M \mid \neg\phi \mid \phi \vee \phi$$

- A formula of SAL can be interpreted at states of  $K$  if if each observation of  $K$  is a valuation for a superset of the variables appearing in the observations of all automata occurring in  $\phi$
- Semantics: A state  $s$  of  $K$  satisfies the SAL formula  $M$  if for every source- $s$  trajectory  $\bar{s}_{0..m}$  of  $K$  there is a trace  $\bar{a}_{0..m} \in L_M$  such that for all  $i$ ,  $s_i \models a_i$
- Like other state logics, we can define characteristic regions, satisfaction  $K \models \phi$  etc.

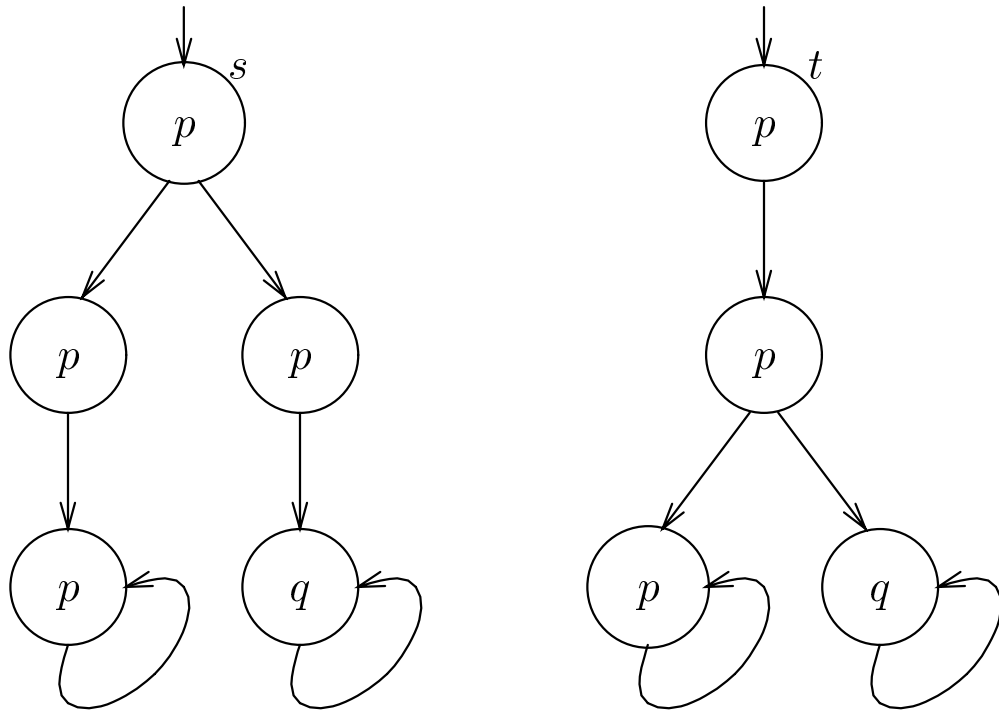
# Specifying Mutual Exclusion



## Trace Equivalence

- Two states  $s$  and  $t$  of an observation structure  $K$  are trace equivalent, denoted  $s \simeq^L t$ , if  $L_K(s) = L_K(t)$ .
- Trace equivalence is less distinguishing than bisimilarity: two bisimilar states are guaranteed to be trace equivalent, but not vice versa.
- Trace equivalence: linear,  
bisimilarity: branching

# Trace Equivalence vs. Bisimilarity



## SAL versus STL

- Recall: bisimilarity is fully abstract semantics for STL
- Theorem: Trace equivalence is fully abstract semantics for SAL.
  - no SAL formula can distinguish between two states that are trace equivalent
  - for every two states that are not trace equivalent, there exists a SAL formula that is satisfied by only one of the two states
- Corollary: SAL is less distinguishing than STL
- Corollary: To solve model checking problems for SAL, we can first compute bisimilarity quotient

## Expressive Power

- SAL and STL have incomparable expressive powers
- If  $s$  and  $t$  are trace-equivalent, but not bisimilar, then they agree on all SAL formulas, but STL can distinguish between them
- In general, SAL cannot express existential properties e.g. from every reachable state, some  $p$  state is reachable
- STL cannot count: no STL formula is equivalent to the automaton  $M_{even}$  ( $x$  holds in every even numbered state in every trajectory)

