

CIS 673: Lecture 8: Oct 2
Symmetry

- If the system contains renamed copies of same module definition, then the state-space exhibits symmetries
- The processes P_1 and P_2 of Pete
- Two trains of the railroad controller
- Processes and buffers in agreement protocol
- Symmetric states: one can be obtained from the other by permuting indices
- Formalization via graph automorphisms

Graph Automorphisms

- Transition graph $G = (\Sigma, \sigma^I, \rightarrow)$.
- A G -automorphism is a bijection f from Σ to Σ such that
 - Initial states are mapped to initial states:
 $f(\sigma^I) = \sigma^I$
 - Preserves transitions: $s \rightarrow t$ iff $f(s) \rightarrow f(t)$.
- Identity function id is a G -automorphism
- Inverse of automorphism is also automorphism
- Functional composition of two automorphisms is an automorphism
- The subgroup generated by a set of automorphisms contains only automorphisms

Groups

- A group is a set A with a multiplication operation \circ :
 - \circ is associative,
 - there exists an element that is identity for \circ
 - every element has inverse with respect to \circ
- A subgroup of (A, \circ) is a subset $B \subseteq A$ such that (B, \circ) is a group
- For a subset $B \subseteq A$, the subgroup generated by B , denoted $\text{closure}(B)$, is the smallest subgroup of (A, \circ) that contains B
- The elements in B are called generators for the group $\text{closure}(B)$
- Example: the set of functions with functional composition

Symmetric Partition

- F : Set of G -automorphisms
- $\text{closure}(F)$: Subgroup generated by F
- \cong^F : two states s and t are equivalent if there exists $f \in \text{closure}(F)$ such that $t = f(s)$
- Theorem: \cong^F is stable
- If the target region σ^T is a block of every $f \in F$, then it is a block of every $f \in \text{closure}(F)$.

Strategy for Symmetric Reduction

1. Identify a set F of generator mappings over state-space of module such that
 - every $f \in F$ is an automorphism
 - the invariant is a block of every $f \in F$
2. Consider the stable partition \cong^F
3. Solve the problem on the quotient wrt \cong^F

Automorphisms for Pete

- f toggles x_1 and toggles x_2 : $t = f(s)$ iff

$$x_1[t] \neq x_1[s] \text{ and } x_2[t] \neq x_2[s],$$

$$pc_1[t] = pc_1[s] \text{ and } pc_2[t] = pc_2[s].$$
- g swaps the values of pc_1 and pc_2 , and if pc_2 is outC, toggles x_2 else toggles x_1 : $t = g(s)$ iff

$$pc_1[t] = pc_2[s] \text{ and } pc_2[t] = pc_1[s], \text{ and}$$
 if $pc_2[s] = outC$ then

$$x_1[t] = x_1[s] \text{ and } x_2[t] \neq x_2[s]$$
 else $x_1[t] \neq x_1[s]$ and $x_2[t] = x_2[s]$
- Subgroup generated by f and g : $\{f, g, id, f \circ g\}$
- Resulting partition has 12 classes of which 10 are reachable
- The region $\llbracket \neg(pc_1 = inC \wedge pc_2 = inC) \rrbracket$ is invariant under both functions f and g

Symmetry in Star Topology

- Server module P communicating with identical clients P_1, \dots, P_n
- Swapping controlled vars of two clients is an automorphism
- Set of generators: For every $1 \leq i, j \leq n$, an automorphism f_{ij}
- Partition \cong^F : one state is obtained from the other by arbitrary permutation of indices of client vars
- Example: Railroad controller

Symmetry in Ring Topology

- Identical modules P_1, \dots, P_n connected in a ring
- Each module communicates with its neighbours
- Single generator: rotate left one step: $t = f(s)$
if $\text{ctr}X_{P_{i+1}}[t] = \text{ctr}X_{P_i}[s]$.
- The subgroup generated by f : rotate left arbitrarily many steps
- Example: dining philosophers, leader election, agreement

On-the-fly Reduction

- How do we exploit symmetries during on-the-fly search?
- Find a set of representative states, one per equivalence class of \cong^F
- Find a mapping rep that maps each state to its representative
- Explore only representatives:
 - Replace *InitQueue* by $rep \circ InitQueue$
 - Replace *PostQueue* by $rep \circ PostQueue$
- Orbit problem: given a set of generators (permutations of indices), determine if two states are equivalent.
- Orbit problem is as hard as graph isomorphism (no known polynomial solution)

Symmetry in Practice

- Enumerative model checker Mur φ at Stanford
- User-specified automorphisms in syntax:
 - Repetition type with restricted set of operations
 - Permuting indices in repetition type is guaranteed to be an automorphism
- On-the-fly symmetry reduction during DFS:
 - Each equivalence class can have multiple representatives
 - Mapping states to representatives is efficient

Temporal Safety Requirements

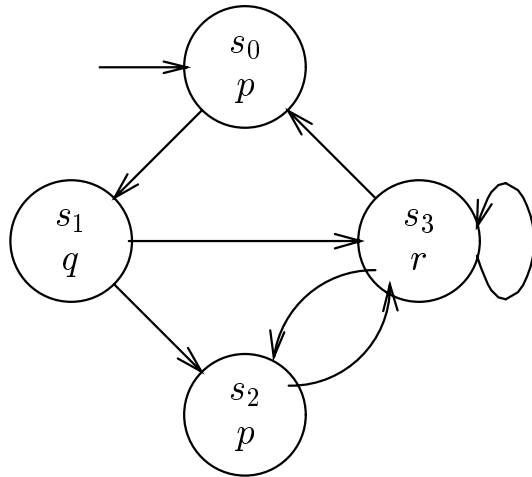
- Propositional logic: requirements about states (static)
- Temporal logic: requirements about how state evolves (dynamic)
- Origins: in philosophy (tense logic), modal logics
- Application to computer science: as a requirements language for reactive programs (Pnueli, 1977)
- Invariants can specify requirements about reachable states, but not about temporal ordering of states along trajectories
- Why are monitors not adequate?
 - Monitors increase state-space
 - More dependent on the model being debugged
 - Technical reasons: less expressive

- Temporal logic = atomic formulas + boolean connectives + temporal modalities (e.g. always, eventually)
- Many varieties: linear-time, branching-time, partial-order, real-time
- Suitable for writing requirements even before design or modeling
- For now, we study a fragment of CTL, called STL (only safety properties)
- Semantic models for temporal logics: observation structures (also called Kripke structures)

Observation Structures

- Transition graph + labeling of states with observations
- Restriction: Finite unobservable branching
- Definition: An observation structure K has
 1. a transition graph $(\Sigma, \sigma^I, \rightarrow)$
 2. observation alphabet: a set A of observations
 3. observation function: a function $\langle\langle \cdot \rangle\rangle : \Sigma \rightarrow A$ that maps each state s to an observation $\langle\langle s \rangle\rangle$ such that (i) $\langle\langle \sigma^I \rangle\rangle$ is finite, and (ii) for every state $s \in \Sigma$, $\langle\langle \text{post}_G(s) \rangle\rangle$ is finite.

Sample Observation Structure



From Modules to Observation Structures

- Observation structure K_P of a module P :
 - Transition graph G_P
 - Observation alphabet: All valuations to observable variables: $\Sigma_{\mathbf{obs}X_P}$
 - Observation function: projection.
Observation of state s is $\mathbf{obs}X_P[s]$
- Finite unobservable branching assured
 - Finitely many ways to initialize controlled state
 - Finitely many ways to update controlled state

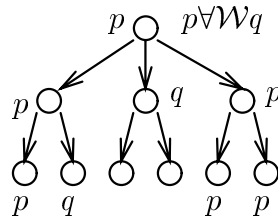
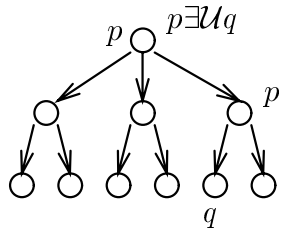
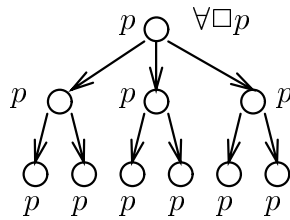
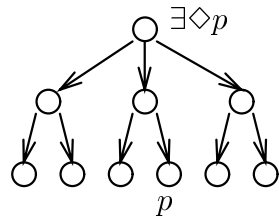
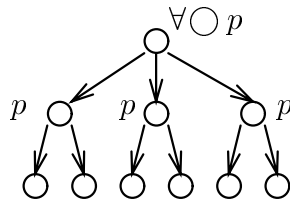
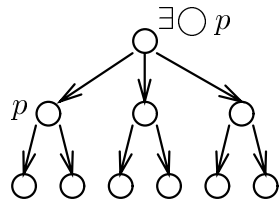
Defining Temporal Logics

- A logic is defined by
 - Syntax: the set of well-formed formulas of the logic
 - Semantics: how to interpret the formulas
- Formulas interpreted over states of observation structures
- A formula ϕ is built from atomic formulas
- ϕ can be interpreted over states of K if an observation in K is a valuation for superset of variables occurring in ϕ (then, we call K is a ϕ -structure)
- Semantics defined by giving rules for $s \models_K \phi$ (state s of K satisfies ϕ)

Safe Temporal Logic

- Trajectory selectors: \exists (along some trajectory), \forall (along all trajectories)
- Temporal operators select states on a trajectory: \bigcirc : next, \diamond : eventually, \square : always, \mathcal{U} : until
- Possibly-next $\exists\bigcirc$: $\exists\bigcirc p$ holds in a state s if some successor of s satisfies p
- Possibly-eventually $\exists\diamond$: $\exists\diamond p$ holds in a state s if some state satisfying p is reachable from s
- Inevitably-always $\forall\square$: $\forall\square p$ holds in a state s if all states that are reachable from s satisfy p
- Possibly-until $\exists\mathcal{U}$: $p\exists\mathcal{U}q$ holds in a state s if there a source- s trajectory whose last state satisfies q and previous states satisfy p

STL Operators



Formal Definition of STL

- Syntax: defined by inductive grammar

$$\phi ::= p \mid \phi \vee \phi \mid \neg\phi \mid \exists \bigcirc \phi \mid \phi \exists \mathcal{U} \phi,$$

where p is an atomic formula

- Semantics: satisfaction relation defined inductively

$$\begin{aligned} s \models_K p & \quad \text{iff } \langle\langle s \rangle\rangle \models p; \\ s \models_K \phi \vee \psi & \quad \text{iff } s \models_K \phi \text{ or } s \models_K \psi; \\ s \models_K \neg\phi & \quad \text{iff } s \not\models_K \phi; \\ s \models_K \exists \bigcirc \phi & \quad \text{iff there is a state } t \in \text{post}_K(s) \text{ such that} \\ & \quad t \models_K \phi; \\ s \models_K \psi \exists \mathcal{U} \phi & \quad \text{iff there is a source-}s \text{ trajectory } \bar{s}_{0..m} \text{ of } K \\ & \quad \text{such that (1) } s_m \models_K \phi \text{ and} \\ & \quad \text{(2) for all } 0 \leq i < m, s_i \models_K \psi. \end{aligned}$$

Derived Connectives

- The propositional connectives \wedge (conjunction), \rightarrow (implication), and \leftrightarrow (equivalence) can be defined using the connectives \vee (disjunction) and \neg (negation).

- Additional temporal operators:

Inevitably-next	$\forall \bigcirc \phi$	for	$\neg \exists \bigcirc \neg \phi$;
Possibly-eventually	$\exists \diamond \phi$	for	<i>true</i> $\exists \mathcal{U} \phi$;
Inevitably-always	$\forall \square \phi$	for	$\neg \exists \diamond \neg \phi$;
Inevitably-waiting-for	$\phi \forall \mathcal{W} \psi$	for	$\neg((\neg \psi) \exists \mathcal{U} \neg(\phi \vee \psi))$.

STL Specifications

- Mutual exclusion specified by

$$\forall \square \neg (pc_1 = in \wedge pc_2 = in)$$

equivalently by

$$\neg \exists \diamond (pc_1 = in \wedge pc_2 = in).$$

- First-come-first-in requirement

$$\forall \square ((pc_1 = req \wedge pc_2 = out) \rightarrow (pc_2 \neq in) \forall \mathcal{W} (pc_1 = in))$$

- Deadlock-freedom

$$\forall \square (pc_1 = req \rightarrow \exists \diamond (pc_1 = in))$$

Tree Interpretation of STL

- For a state s of an observation structure K , let $T_K(s)$ be the tree obtained by unwinding the structure
 - s is the root
 - Nodes of the tree correspond to source- s trajectories
 - $T_K(s)$ can be viewed as another observation structure
- Semantics of STL is insensitive to such unwinding:

$$s \models_{T_K(s)} \phi \text{ iff } s \models_K \phi$$

- Satisfaction of an STL-formula at a state s depends only on substructure reachable from s

Sample tree

