Week 14 - Monday

COMP 2230

Last time

- More on finite-state automata
- Simplifying finite-state automata

Questions?

Assignment 6

Logical warmup

- If three cats catch three rats in three minutes, how many cats will catch 100 rats in 100 minutes?

Simplifying FSAs

*-equivalence

- Two states of a finite-state automaton are $*$**-equivalent** if any string accepted by the automaton when it starts from one state is accepted when starting from the other
- Given an automaton A with eventual-state function N^*, we can formally say:
  - States s and t in A are $*$-equivalent iff $N^*(s, w)$ and $N^*(t, w)$ are both accepting states or both not
- It turns out that $*$-equivalence defines an equivalence relation

k-equivalence

- $*$-equivalence is hard to demonstrate directly
- Instead, we'll focus on equivalence after k or fewer inputs
- Given an automaton A with eventual-state function N^*, we can formally say:
  - States s and t in A are k-equivalent iff $N^*(s, w)$ and $N^*(t, w)$ are both accepting states or both not, for all strings w of length k or less

Facts about k-equivalence

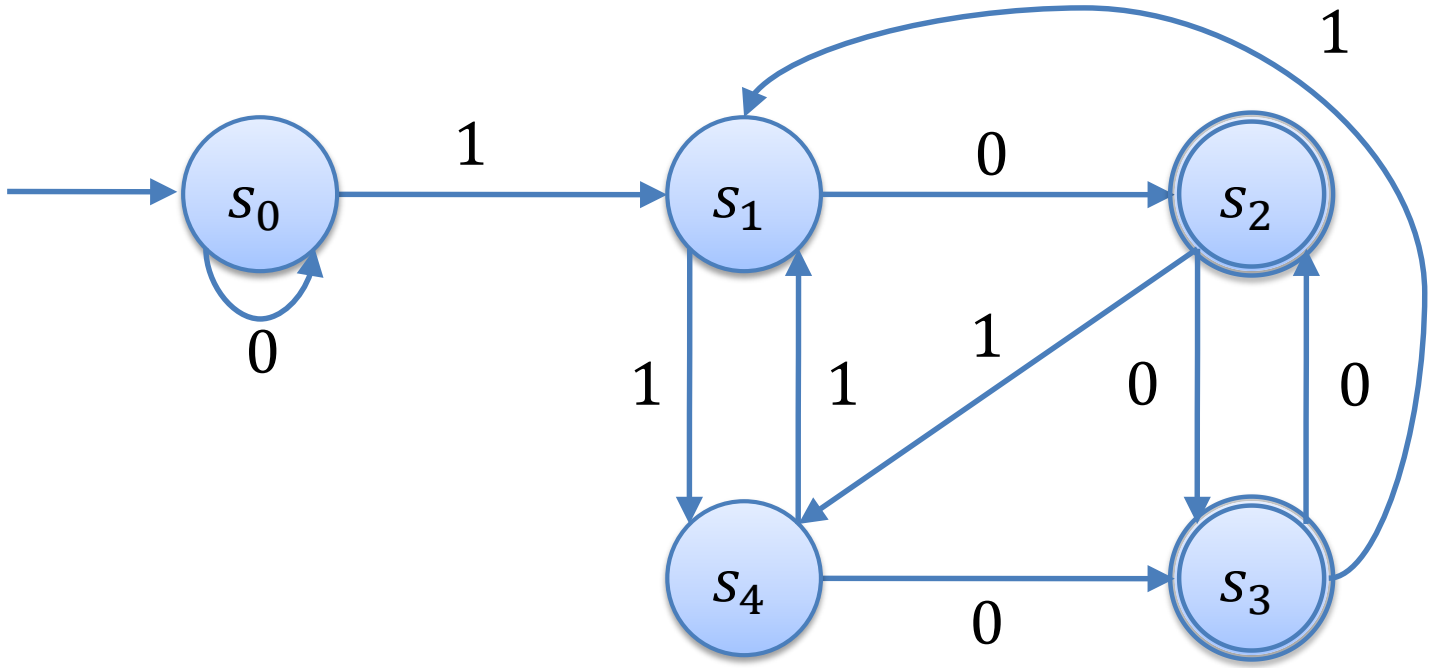
- For $k \geq 0$, k-equivalence is an equivalence relation
- For $k \geq 0$, the k-equivalence classes partition the set of all states of the automaton into a union of mutually disjoint subsets
- For $k \geq 1$, if two states are k-equivalent, they are also $(k-1)$-equivalent
- For $k \geq 1$, each k-equivalence class is a subset of a $(k-1)$-equivalence class
- Any two states that are k-equivalent for all integers $k \geq 0$ are $*$-equivalent

k-equivalence theorems

- Let A be an FSA with next-state function N
- Given any states s and t in A:
  1. s is 0-equivalent to t iff either s and t are both accepting states or they are both nonaccepting states
  2. For every integer $k \geq 1$, s is k-equivalent to t iff s and t are $(k-1)$-equivalent and for any input symbol m, $N(s,m)$ and $N(t,m)$ are also $(k-1)$-equivalent
- These theorems essentially allow us to create a recursive definition for testing k-equivalence

k-equivalence examples

- Find the 0-equivalence classes, the 1-equivalence classes, and the 2-equivalence classes for the following FSA:



Finding the $*$-equivalence classes

- Keep finding k-equivalence classes for larger and larger values of k
- If you ever find that the set of k-equivalence classes is equal to the set of $(k+1)$-equivalence classes, that is the set of $*$-equivalence classes
- This is known as a **fixed point** in mathematics

The quotient automaton

- We can build a new FSA from the $*$-equivalence classes
- Recall that $[s]$ means the equivalence class of s
- This FSA is called the **quotient automaton** A', and is defined from an FSA A with states S, input symbols I, and next-state function N as follows:
  1. The set of states S' of A' is the set of $*$-equivalent classes of states of A
  2. The set of input symbols I' of A' equals I
  3. The initial state of A' is $[s_0]$ where s_0 is the initial state of A
  4. The accepting states of A' are the states of the form $[s]$ where s is an accepting state of A
  5. The next-state function N': $S' \times I \rightarrow S'$ is:

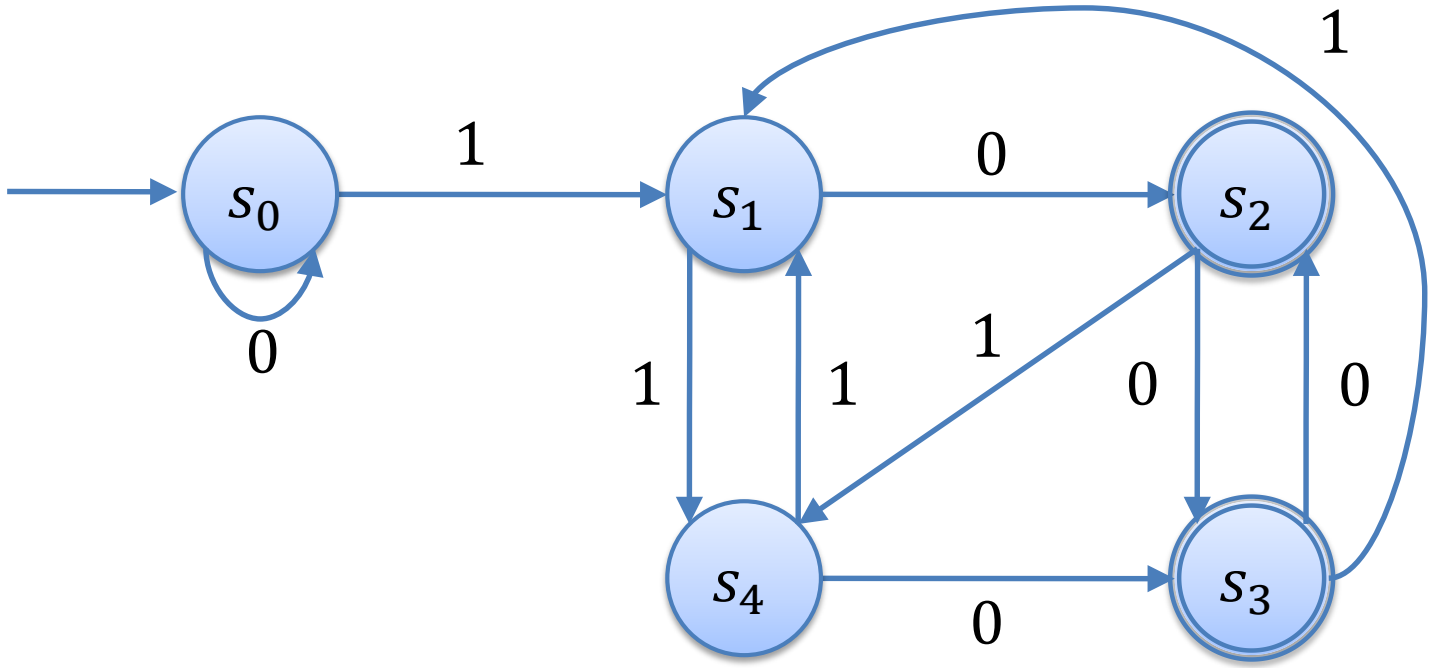
     For all states $[s]$ in S' and input symbols m, $N'([s], m) = [N(s, m)]$

Constructing a quotient automaton

- Let A be an FSA with states S, input symbols I, and next-state function N
- To build A':
  1. Find the set of 0-equivalence classes of S
  2. For each integer $k \geq 1$, find the k-equivalence classes of S until the k-equivalence classes are the same as the $(k-1)$-equivalence classes
  3. Build a quotient automaton whose states are the equivalence classes given above with transition function $N'([s], m) = [N(s, m)]$ for any input symbol m

Quotient automaton example

- Find the quotient automaton for the following FSA

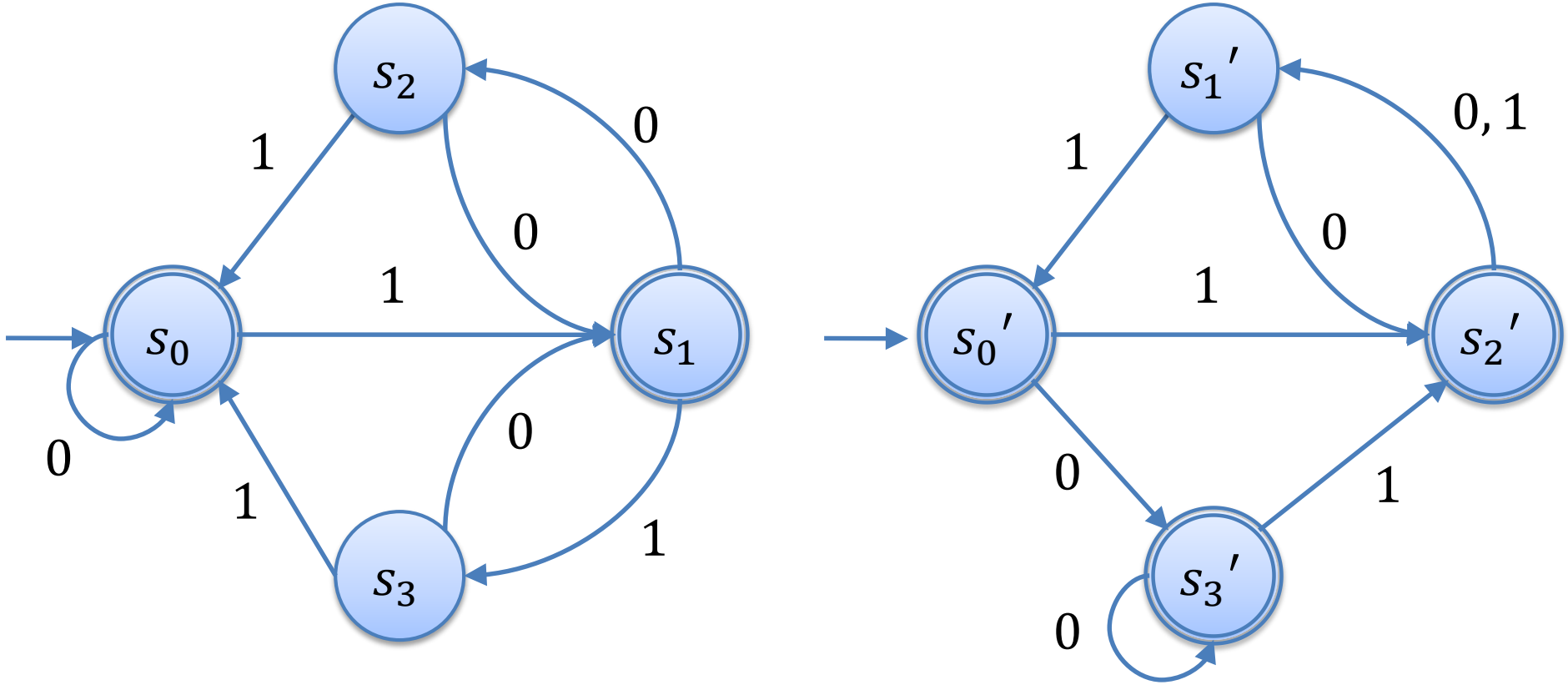


Equivalent automata

- Two automata A_1 and A_2 are equivalent iff $L(A_1) = L(A_2)$
- Proving the languages accepted by two automata can be difficult
- However, the quotient automata for both A_1 and A_2 will be the same (except for labeling) if A_1 is equivalent to A_2

Proving equivalence

- Prove that the following two automata are equivalent by finding their quotient automata



Context-Free Grammars

Three-Sentence Summary

Context Free Languages

Context free languages

- A context free language is one that can be described by a context free grammar
- Every regular language is context free, but there are context free languages that are not regular
- Classic examples:
  - Strings of k 0's followed by k 1's
  - Palindromes made up of a's and b's
  - Legally nested parentheses
- All of these involve counting arbitrary numbers of characters
  - Regular expressions can't count

Context free grammars

- Instead of using regular expressions, a context free language is often described with a **grammar**
- A grammar is a formal system of rewriting rules consisting of
  - Terminals: symbols of the alphabet
  - Non-terminals: symbols that produce other sequences of terminals and non-terminals
- Grammars often start with the non-terminal starting symbol S
- Any string that can be derived from S through some sequence of rule rewrites is a string in the language

Simple context free grammars

- The following is a grammar that produces the language of strings of 1s and 0s that end in a 1
  - $S \rightarrow A1$
  - $A \rightarrow \varepsilon \mid 1 \mid 0 \mid A1 \mid A0$
- This language is regular and is equivalent to $(0|1)^*1$
- The following is a grammar that produces the language of $a^k b^k$ where $k \geq 1$ (which is not regular)
  - $S \rightarrow A$
  - $A \rightarrow ab \mid aAb$

CFG examples

- Write a grammar that describes legal nesting of parentheses and braces in Java
  - Don't worry about the stuff that goes inside
- Write a grammar for legal mathematical expressions in Java using variables and integers, **+**, **–**, *******, **/**, and parentheses
- Write a grammar that corresponds to the same language defined by the regular expression $ab^*\ (a \mid bb)\ (ba)^*$

Upcoming

Next time…

- Pushdown automata
- Turing machines and the halting problem

Reminders

- Keep working on Assignment 6