Regular Languages, Continued

November 2, 2011

CS 81: Computability & Logic

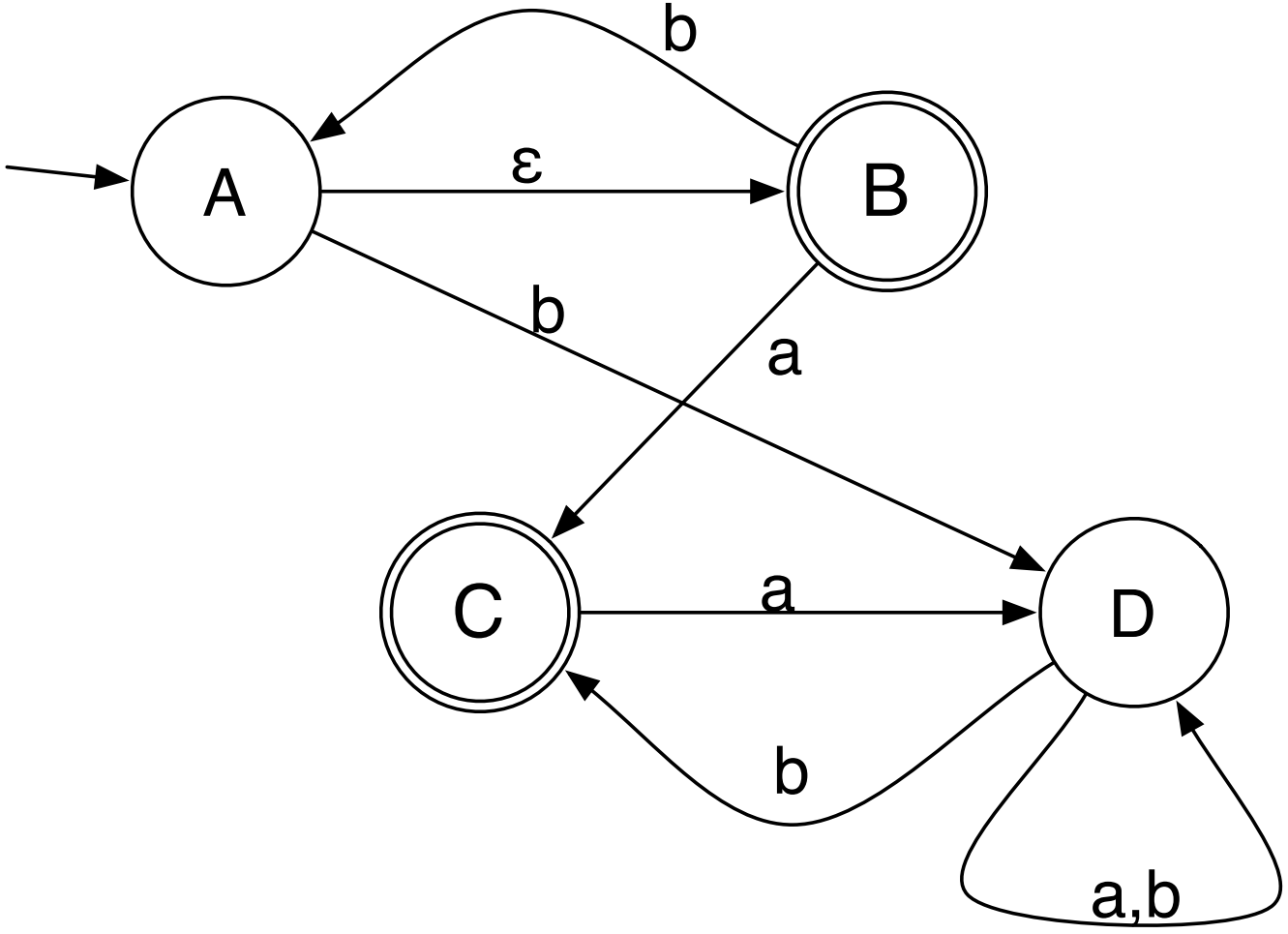
A Jewel of Theoretical Computer Science



The following are equivalent:

1. There is a DFA accepting the language L
2. [Rabin and Scott] There is an NFA accepting L
3. [Kleene] L is a regular set.

From NFA to DFA: the Subset Construction



From Regular Expression to NFA

Construct $\mathrm{NFA}(r)$ by induction/recursion on the regular expression r.

- ✓ $\emptyset$ is a regexp
- ✓ ε is a regexp
- ✓ a is a regexp for any $a \in \Sigma$.
- ✓ If r_1 and r_2 are regexps, then so is $(r_1 r_2)$ and $(r_1 | r_2)$.
- ✓ If r is a regexp, then so is (r^*).

Regular Expression Review

- ✓ What are the usual "primitive" regular expressions?

- ✓ What other regular expressions have you seen?

```
M[ou]'?am+[ae]r .*([AEae]l[- ])?[GKQ]h?[aeu]+([dtz][dhz]?)+af[iy]
```

Give two strings matching this regular expression.

```
Muammar Qaddafi
Mo'ammar Gadhafi
Muammar Kaddafi
Muammar Qadhafi
Moammar El Kadhafi
Muammar Gadafi
Mu'ammar al-Qadafi
Moamer El Kazzafi
Moamar al-Gaddafi
Mu'ammar Al Qathafi
Muammar Al Qathafi
Mo'ammar el-Gadhafi
Moamar El Kadhafi
Muammar al-Qadhafi
Mu'ammar al-Qadhdhafi
Mu'ammar Qadafi
```

```
Moamar Gaddafi
Mu'ammar Qadhdhafi
Muammar al-Khaddafi
Mu'amar al-Kadafi
Muammar Ghaddafy
Muammar Ghadafi
Muammar Ghaddafi
Muamar Kaddafi
Muammar Quathafi
Muammar Gheddafi
Muamar Al-Kaddafi
Moammar Khadafy
Moammar Qudhafi
Mu'ammar al-Qaddafi
Mu'ammar Muhammad Abu Minyar al-Qadhafi
```

From the RX library

Exercise

Give a regular expression for C identifiers:

- ✓ Can contain letters, digits, and underscores
- ✓ Must begin with a letter or underscore.
- ✓ E.g., `main` or `__Z6rotateiii`

Give a regular expression for Ada identifiers, which

- ✓ Can contain letters, digits and underscores
- ✓ Begin with a letter
- ✓ Have no consecutive underscores or an underscore at the end
- ✓ E.g., `woohoo32` or `Last_Nonzero_Row`

Exercise: Integer Constants in C

Kernighan & Ritchie
"The C Programming Language"

A2.5.1 Integer Constants

An integer constant consisting of a sequence of digits is taken to be octal if it begins with `0` (digit zero), decimal otherwise. Octal constants do not contain the digits `8` or `9`. A sequence of digits preceded by `0x` or `0X` (digit zero) is taken to be a hexadecimal integer. The hexadecimal digits include `a` or `A` through `f` or `F` with values 10 through 15.

An integer constant may be suffixed by the letter `u` or `U`, to specify that it is unsigned. It may also be suffixed by the letter `l` or `L` to specify that it is long.

The type of an integer constant depends on its form, value and suffix. (See §A4 for a discussion of types.) If it is unsuffixed and decimal, it has the first of these types in which its value can be represented: `int`, `long int`, `unsigned long int`. If it is unsuffixed octal or hexadecimal, it has the first possible of these types: `int`, `unsigned int`, `long int`, `unsigned long int`. If it is suffixed by `u` or `U`, then `unsigned int`, `unsigned long int`. If it is suffixed by `l` or `L`, then `long int`, `unsigned long int`.

> The elaboration of the types of integer constants goes considerably beyond the first edition, which merely caused large integer constants to be `long`. The `U` suffixes are new.

A2.5.2 Character Constants

Concrete Syntaxes for Regular Expressions

Different applications may use different concrete syntaxes for regular expressions:

- ✓ perl: `b(ea|a)(r|d)`
- ✓ sed: `b\(ea|a\)\(r|d\)`
- ✓ emacs: `b\(ea\|a\)\(r\|d\)`

Globs

Also, OS shells often support another variant of regular-expression-like syntax (*globs*):

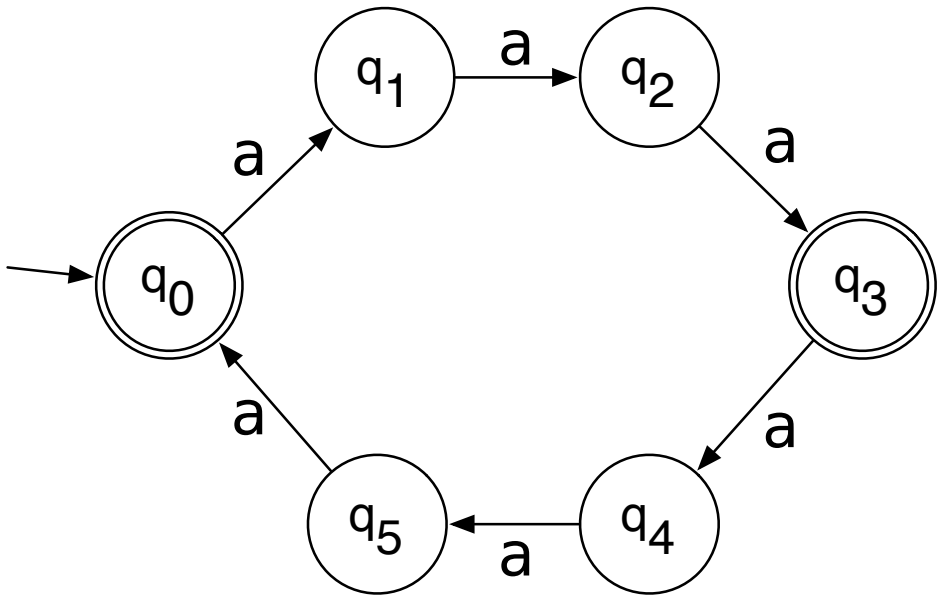
```
> ls
bad bag bar bead beg bear beer bug ear rag rear rug
> ls b*r
bar bear beer
> ls b?ar
bear
> ls [br]ear
bear rear
> ls b{ea,a}{r,d}
bad bar bead bear
> ls {?,r?}ar
bar ear rear
> ls b.g
ls: b.g: No such file or directory
```

Completing the Equivalence: Automata to Regular Expressions

Two approaches:

1. Solving equations
2. Generalized NFAs

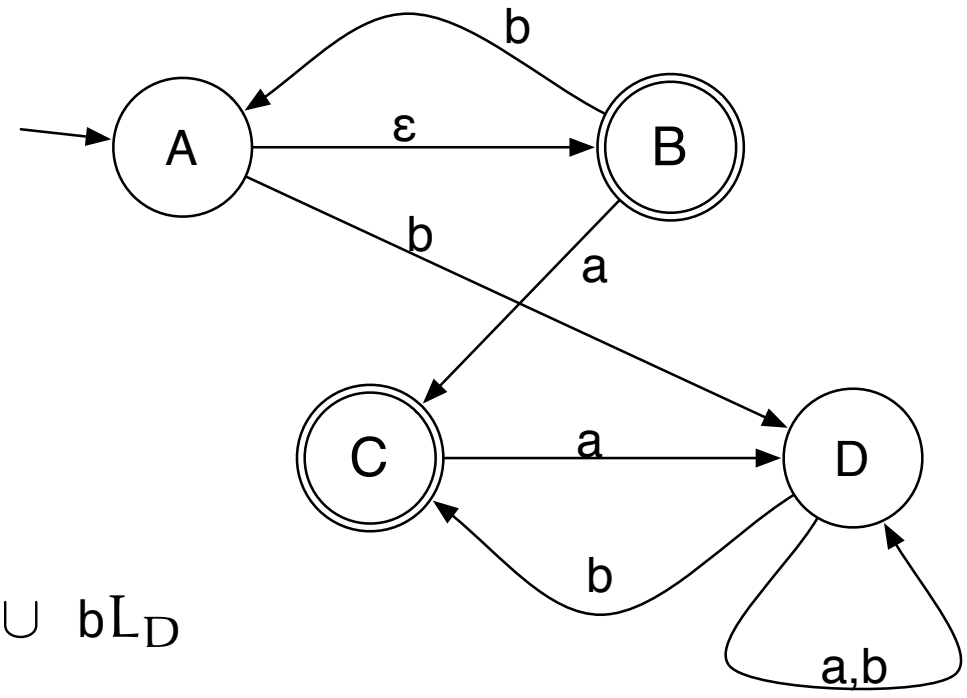
The Language of a State



Let L_q be the set of strings are accepted when starting from state q.

- ✓ What is $L_{q_0}, L_{q_1}, L_{q_2}, \ldots$?
- ✓ How is L_{q_1} related to L_{q_2} ?

Automaton as a System of Equations



- ✓ $L_A = \varepsilon L_B \cup bL_D$
- ✓ $L_B =$
- ✓ $L_C =$
- ✓ $L_D =$

Solving Equations using Arden's Rule

- ✓ The equation

$$L = AL \cup B$$

has the solution

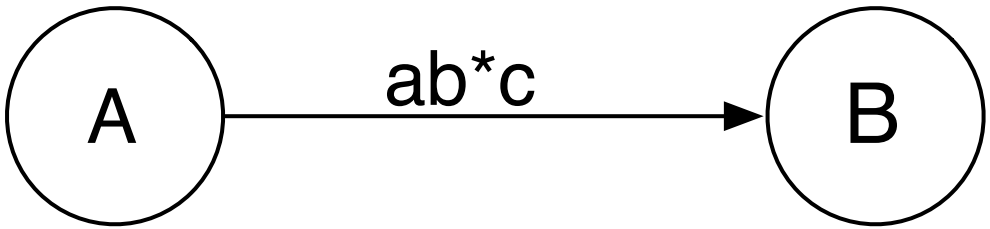
$$L = A^*B$$

- ✓ This is the smallest solution
  - If $\varepsilon \notin A$, the unique solution
  - Otherwise A^*C is a solution for any $B \subseteq C$.

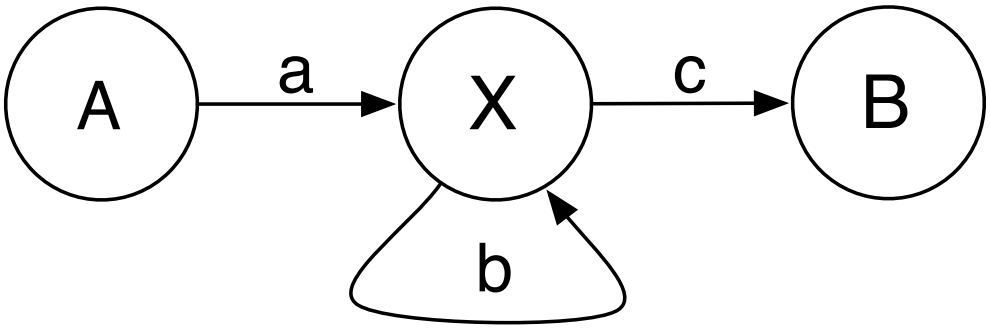
$$\begin{aligned} L_A &= L_B \cup bL_D \\ L_B &= \varepsilon \cup bL_A \cup aL_C \\ L_C &= \varepsilon \cup aL_D \\ L_D &= (a \cup b)L_D \cup bL_C \end{aligned}$$

Generalized NFAs

Just like an NFA, but *edges* have regular expressions rather than single symbols



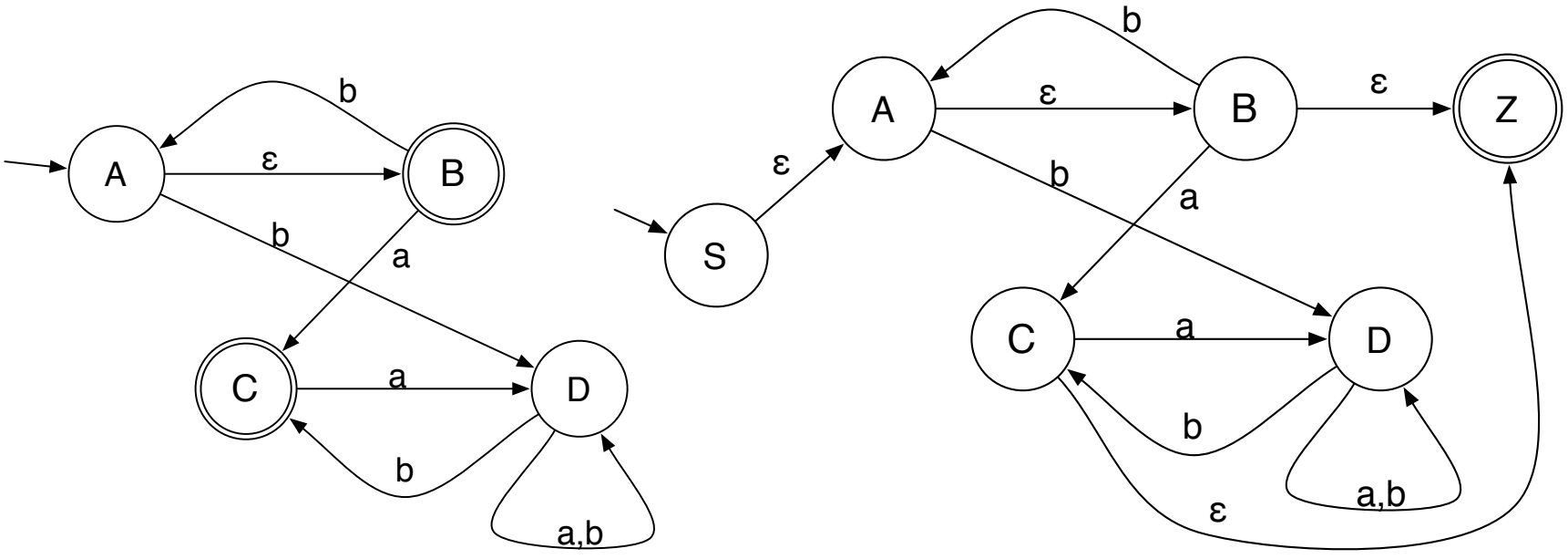
Since regular expressions can be turned into NFAs, we aren't adding any extra power.



Regexp by Removing States

The strategy:

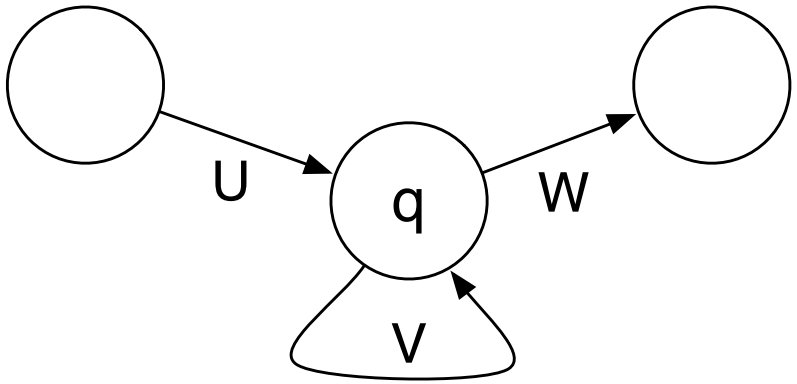
- ✓ Make sure our NFA has
  - One start state, with edges only going out
  - One accept state, with edges only going in.



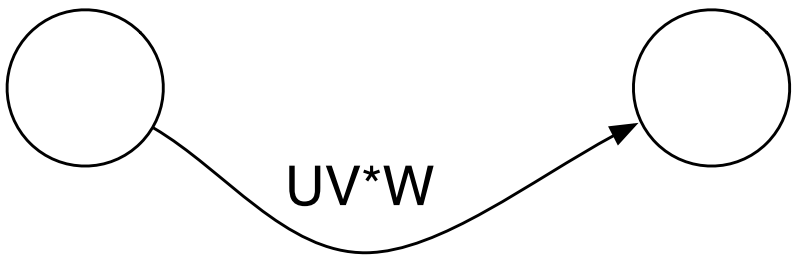
- ✓ Remove all the intermediate states (A–D), one at a time.
- ✓ In the end, we have one edge, labeled by our regexp.

Removing States

- ✓ When removing state q, replace every pair of in/out edges



by a single edge



Example