

Programming Languages and Compilers (CS 421)

Elsa L Gunter
2112 SC, UIUC
<http://www.cs.uiuc.edu/class/fa06/cs421/>

Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

Extra Office Hours

- Friday 3:30pm - 5:00pm
- Monday 9:00am - 10:30am

Elsa L. Gunter

Questions?

Elsa L. Gunter

Language Syntax

- Syntax is the description of which strings of symbols are meaningful expressions in a language
- It takes more than syntax to understand a language; need meaning (semantics) too
- Syntax is the entry point

Elsa L. Gunter

Elements of Syntax

- Character set – previously always ASCII, now often 64 character sets
- Keywords – usually reserved
- Special constants – cannot be assigned to
- Identifiers – can be assigned to
- Operator symbols
- Delimiters (parenthesis, braces, brackets)
- Blanks (aka white space)

Elsa L. Gunter

Elements of Syntax

- Expressions
- Type expressions
- Declarations (in functional languages)
- Statements (in imperative languages)
- Subprograms

Elsa L. Gunter

Elements of Syntax

- Modules
- Interfaces
- Classes (for object-oriented languages)

Elsa L. Gunter

Formal Language Descriptions

- Regular expressions, regular grammars, finite state automata
- Context-free grammars, BNF grammars, syntax diagrams
- Whole family more of grammars and automata – covered in automata theory

Elsa L. Gunter

Grammars

- Grammars are formal descriptions of which strings over a given character set are in a particular language
- Language designers write grammar
- Language implementers use grammar to know what programs to accept
- Language users use grammar to know how to write legitimate programs

Elsa L. Gunter

Regular Expressions

- Start with a given character set – **a, b, c...**
- Each character is a regular expression
 - It represents the set of one string containing just that character

Elsa L. Gunter

Regular Expressions

- If **x** and **y** are regular expressions, then **xy** is a regular expression
 - It represents the set of all strings made from first a string in **x** then a string in **y**
- If **x** and **y** are regular expressions, then **xvy** is a regular expression
 - It represents the set of strings in either **x** or **y**

Elsa L. Gunter

Regular Expressions

- If **x** is a regular expression, then so is **(x)**
 - It represents the same thing as **x**
- If **x** is a regular expression, then so is **x***
 - It represents strings made from concatenating zero or more strings from **x**
- **ε**
 - It represents the empty set

Elsa L. Gunter

Example Regular Expressions

- $(0^*1)^*1$
 - The set of all strings of 0's and 1's ending in 1, $\{1, 01, 11, \dots\}$
- $a^*b(a^*)$
 - The set of all strings of a's and b's with exactly one b
- $((01) \vee (10))^*$
 - You tell me
- Regular expressions (equivalently, regular grammars) important for lexing, breaking strings into recognized words

Elsa L. Gunter

Example: Lexing

- Regular expressions good for describing lexemes (words) in a programming language
 - Identifier = $(a \vee b \vee \dots \vee z \vee A \vee B \vee \dots \vee Z)(a \vee b \vee \dots \vee z \vee A \vee B \vee \dots \vee Z \vee 0 \vee 1 \vee \dots \vee 9)^*$
 - Digit = $(0 \vee 1 \vee \dots \vee 9)$
 - Number = $(1 \vee \dots \vee 9)(0 \vee \dots \vee 9) \vee \sim (1 \vee \dots \vee 9)(0 \vee \dots \vee 9)$
 - Keywords: if = if, while = while,...

Elsa L. Gunter

Implementing Regular Expressions

- Regular expressions reasonable way to generate strings in language
- Not so good for recognizing when a string is in language
- Regular expressions: which option to choose, how many repetitions to make
- Answer: finite state automata

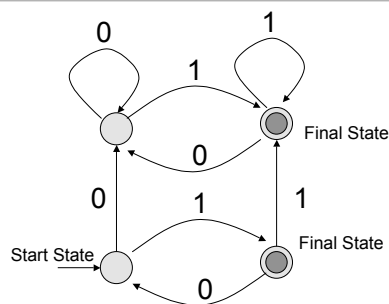
Elsa L. Gunter

Finite State Automata

- A finite state automata over an alphabet is:
 - a directed graph
 - edges are labeled with elements of alphabet, or empty string
 - some nodes (or *states*), marked as final
 - one node marked as start state
- Syntax of FSA

Elsa L. Gunter

Example FSA



Elsa L. Gunter

Deterministic FSA's

- If FSA has for every state exactly one edge for each letter in alphabet then FSA is *deterministic*
 - No edge labeled with ϵ
- In general FSA is *non-deterministic*.
 - NFA also allows edges labeled by ϵ
- Deterministic FSA special kind of non-deterministic FSA

Elsa L. Gunter

DFSA Language Recognition

- Think of a DFSA as a board game; DFSA is board
- You have string as a deck of cards; one letter on each card
- Start by placing a disc on the start state

Elsa L. Gunter

DFSA Language Recognition

- Move the disc from one state to next along the edge labeled the same as top card in deck; discard top card
- When you run out of cards,
 - if you are in final state, you win; string is in language
 - if you are not in a final state, you lose; string is not in language

Elsa L. Gunter

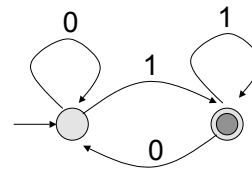
DFSA Language Recognition - Summary

- Given a string over alphabet
- Start at start state
- Move over edge labeled with first letter to new state
- Remove first letter from string
- Repeat until string gone
- If end in final state then string in language
- Semantics of FSA

Elsa L. Gunter

Example DFSA

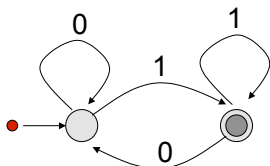
- Regular expression: $(0 \vee 1)^* 1$
- Deterministic FSA



Elsa L. Gunter

Example DFSA

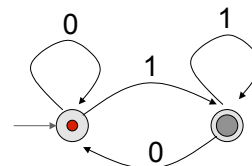
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string 0 1 1 0 1



Elsa L. Gunter

Example DFSA

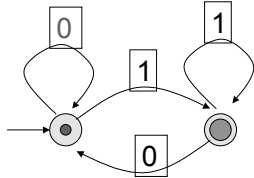
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string 0 1 1 0 1



Elsa L. Gunter

Example DFSA

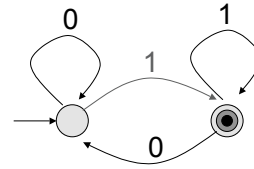
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~ 1 1 0 1



Elsa L. Gunter

Example DFSA

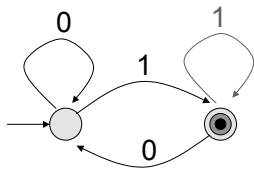
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~ 1 0 1



Elsa L. Gunter

Example DFSA

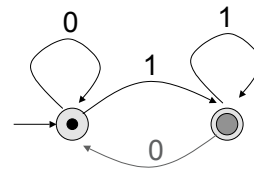
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~~~0~~ 1



Elsa L. Gunter

Example DFSA

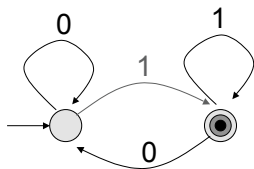
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~~~0~~~~1~~ 1



Elsa L. Gunter

Example DFSA

- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~~~0~~~~1~~~~0~~~~1~~



Elsa L. Gunter

Non-deterministic FSA's

- NFSA generalize DFSA in two ways:
 - Include edges labeled by ϵ
 - Allows process to non-deterministically change state
- Each state can have zero, one or more edges labeled by each letter
 - Given a letter, non-deterministically choose an edge to use

Elsa L. Gunter

NFSA Language Recognition

- Play the same game as with DFSA
- Free move: move across an edge with empty string label without discarding card
- When you run out of letters, if you are in final state, you win; string is in language
- You can take one or more moves back and try again
- If have tried all possible paths without success, then you lose; string not in language

Elsa L. Gunter

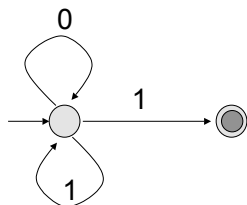
FSA Language Recognition

- Move the disc from one state to next if edge between labeled the same as top card in deck; discard top card
- Free move: move across an edge with empty string label without discarding card
- When you run out of cards, if you are in final state, you win; string is in language
- You can take a move back and try another
- If you have cards left and you have tried all possible edges without success, then you lose; string not in language

Elsa L. Gunter

Example NFSA

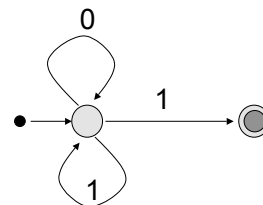
- Regular expression: $(0 \vee 1)^* 1$
- Non-deterministic FSA



Elsa L. Gunter

Example NFSA

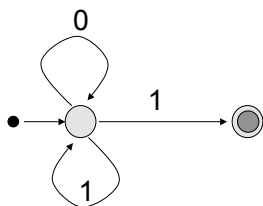
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string 0 1 1 0 1



Elsa L. Gunter

Example NFSA

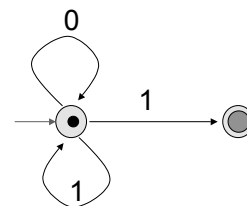
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string 0 1 1 0 1



Elsa L. Gunter

Example NFSA

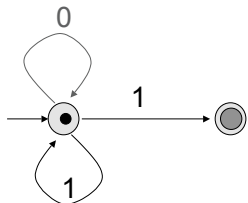
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string 0 1 1 0 1



Elsa L. Gunter

Example NFSA

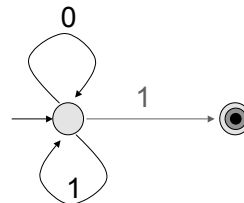
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~1101



Elsa L. Gunter

Example NFSA

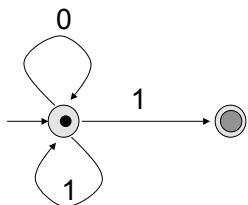
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~101
- Guess



Elsa L. Gunter

Example NFSA

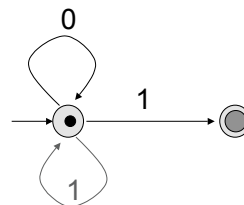
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~1101
- Backtrack



Elsa L. Gunter

Example NFSA

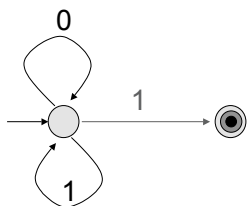
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~101
- Guess again



Elsa L. Gunter

Example NFSA

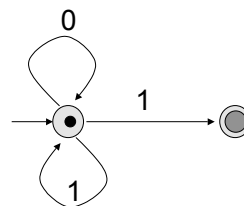
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~~~0~~1
- Guess



Elsa L. Gunter

Example NFSA

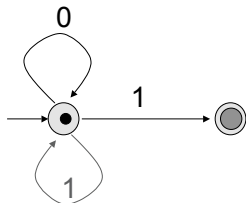
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~101
- Backtrack



Elsa L. Gunter

Example NFSA

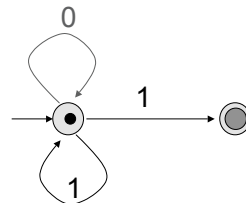
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~0 1
- Guess again



Elsa L. Gunter

Example NFSA

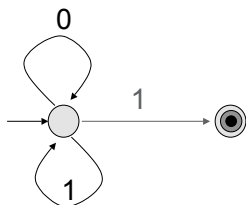
- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~~~0~~1



Elsa L. Gunter

Example NFSA

- Regular expression: $(0 \vee 1)^* 1$
- Accepts string ~~0~~~~1~~~~0~~~~1~~
- Guess (Hurray!!)



Elsa L. Gunter

Rule Based Execution

- Search
- When stuck backtrack to last point with choices remaining
- Executing the NFSA in last example was example of rule based execution
- FSA's are rule-based programs; transitions between states (labeled edges) are rules; set of all FSA's is programming language

Elsa L. Gunter