A Puzzle For You

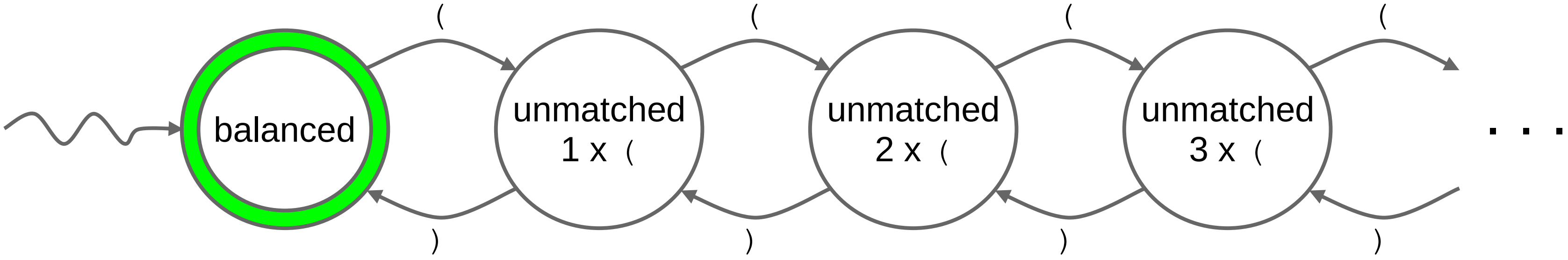
Let Σ = { ' (', ') ' } be the alphabet of parentheses.

Construct a FSM that accepts properly balanced parentheses.

E.g., Accept: λ, ( ), ( ) ( ), ((( ) ( )) ( ))

E.g., Reject: **)**, **(**, **(** ( ), ( ) **)**, **(** ( ) ( ) **(** ( ), **)** ( ) **(**

Strategy: Count the number of unmatched (



Solution requires an infinite number of states!

The Power of FSMs

CMPT 125
Mo Chen
SFU Computing Science
27/3/2020

Lecture 31

Today:

- POSIX Regular Expressions
- The Power of Regular Languages
- Non-regular Languages

Regular Languages (Review)

A regular language is a language that can be decided by a FSM.

Closed under:

- union
- catenation
- Kleene star

Can express a regular language using either:

- a FSM … OR …
- a regular expression

POSIX Extended Regular Expressions

Several tools allow you to use regular expressions

- E.g., command line shells, advanced text editors, perl
- Usually search for patterns rather than Accept / Reject

Today's Tool:

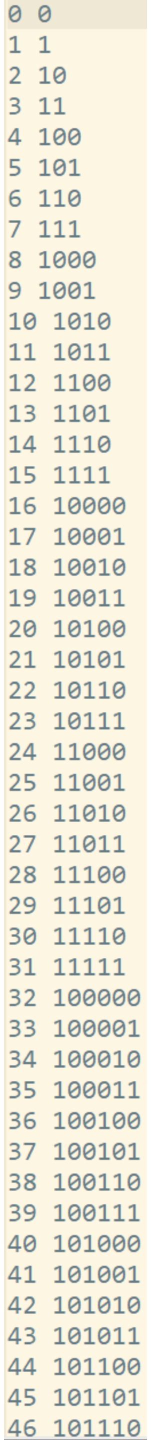
`egrep`

Typical syntax:

- `a|b` and `a*` — union and Kleene star work exactly like you expect
- `a+` — it's like `a*`, but 1 or more `a`'s instead of 0 or more `a`'s
  - E.g., `0+1+` → a block of `0`'s followed by a block of `1`'s
- `a?` — optional, i.e., 0 or 1 occurrence of `a`
  - E.g., `colou?r` → `color|colour`

Problem: Define a pattern that would locate all binary strings with two `1`'s separated by two or more `0`'s.

- `(0|1)*100+1(0|1)*` . . . or maybe just `100+1`



```
0 0
1 1
2 10
3 11
4 100
5 101
6 110
7 111
8 1000
9 1001
10 1010
11 1011
12 1100
13 1101
14 1110
15 1111
16 10000
17 10001
18 10010
19 10011
20 10100
21 10101
22 10110
23 10111
24 11000
25 11001
26 11010
27 11011
28 11100
29 11101
30 11110
31 11111
32 100000
33 100001
34 100010
35 100011
36 100100
37 100101
38 100110
39 100111
40 101000
41 101001
42 101010
43 101011
44 101100
45 101101
46 101110
```

POSIX Extended Regular Expressions

Typical syntax (cont’d):

- `.` — stands for any single character
- `[omgwtf]` — a *bracket expression* - use one of the characters within
  - E.g., `defen[cs]e` → `defence|defense`
- Hyphens are allowed in bracket expressions to denote a range
  - E.g., `1[1-4]2` → `112|122|132|142`
- `^` at the beginning of a bracket expression means “not”
  - E.g., `1[^1-4]2` → `102|152|162|172|182|192|1a2|1b2|...`
- `^` and `$` — the beginning and end of a line, respectively
- `\<` and `\>` — the beginning and end of a word, respectively
  - E.g., `\<face\>` → match the word `face`, but not `facet` or `deface`

Problem: Define a pattern that would locate all decimal numbers with a value of 200 or higher

- `\<(1[0-9]|[2-9])[0-9][0-9]+\>`

Wikipedia
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

In other projects

Wikimedia Commons

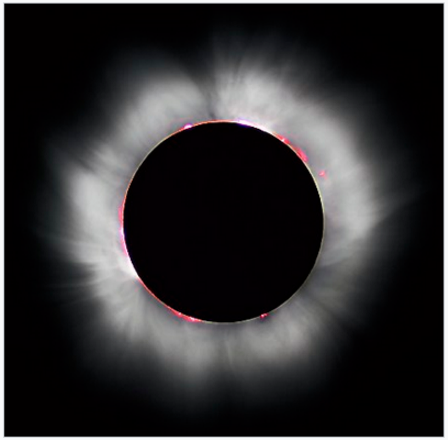
Print/export

Download as PDF
Printable version

Corona

From Wikipedia, the free encyclopedia

This article is about the plasma surrounding stars. For the disease implicated in the ongoing coronavirus pandemic, see Coronavirus disease 2019. For other uses, see Corona (disambiguation).



During a total solar eclipse, the Sun's corona and prominences are visible to the naked eye.

A **corona** (meaning "crown" in Latin, derived from Ancient Greek κορώνη, *korōnè*, "garland, wreath") is an aura of plasma that surrounds the Sun and other stars. The Sun's corona extends millions of kilometres into outer space and is most easily seen during a total solar eclipse, but it is also observable with a coronagraph. Spectroscopy measurements indicate strong ionization in the corona and a plasma temperature in excess of 1 000 000 kelvin,[1] much hotter than the surface of the Sun.

Light from the corona comes from three primary sources, from the same volume of space:

- The K-corona (K for *kontinuierlich*, "continuous" in German) is created by sunlight scattering off free electrons; Doppler broadening of the reflected photospheric absorption lines spreads them so greatly as to completely obscure them, giving the spectral appearance of a continuum with no absorption lines.
- The F-corona (F for Fraunhofer) is created by sunlight bouncing off dust particles, and is observable because its light contains the Fraunhofer absorption lines that are seen in raw sunlight; the F-corona extends to very high elongation angles from the Sun, where it is called the zodiacal light.
- The E-corona (E for emission) is due to spectral emission lines produced by ions that are present in the coronal plasma; it may be observed in broad or forbidden or hot spectral emission lines and is the main source of information about the corona's composition.[2]

Contents [hide]

1 History
1.1 Historical theories
2 Physical features
2.1 Active regions
2.1.1 Coronal loops

list of Fibonacci numbers

The http://caml.inria.fr/ocaml/index.en.htmlOCaml program used to create this list can be found http://aux.planetmath.org/files/objects/7680/fib.mlhere together with compilation and usage instructions as comments.

The list can be downloaded in tab delimited format (UNIX line terminated) \htmladdnormallinkhere http://aux.planetmath.org/files/objects/7680/fib.txt

n	$f(n)$
0	0
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
11	89
12	144
13	233
14	377
15	610
16	987
17	1597
18	2584
19	4181
20	6765
21	10946
22	17711
23	28657
24	46368
25	75025
26	121393
27	196418
28	317811
29	514229
30	832040
31	1346269
32	2178309
33	3524578
34	5702887
35	9227465
36	14930352
37	24157817
38	39088169
39	63245986
40	102334155
41	165580141
42	267914296
43	433494437
44	701408733
45	1134903170
46	1836311903
47	2971215073

Pattern Exercises

Define a pattern for each of the following:

1. All instances of inline C/C++ comments
   - E.g., `puts("Hello"); // inline comment`
   - `//.*$`

2. C-style hexadecimal numbers
   - E.g., `0xffe4`
   - `\<0[xX][0-9a-fA-F]+\>`

The Power of FSMs and Regex

We saw, in the opening exercise, that FSMs can’t decide parenthesis matching

- We have seen a simple algorithm for this earlier in the course, which uses a stack.

If you augment a FSM with an unbounded stack, you can decide balanced parentheses.

- Called a *pushdown automaton*
- Transitions are based on the current state, the next input character, and the topmost stack symbol.
- Actions include push, pop and next input

Non-regular Languages

FSMs are powerful enough to decide regular languages.

- When the language is non-regular, you need a stronger machine.

Pushdown automata are powerful enough to decide *context-free languages*

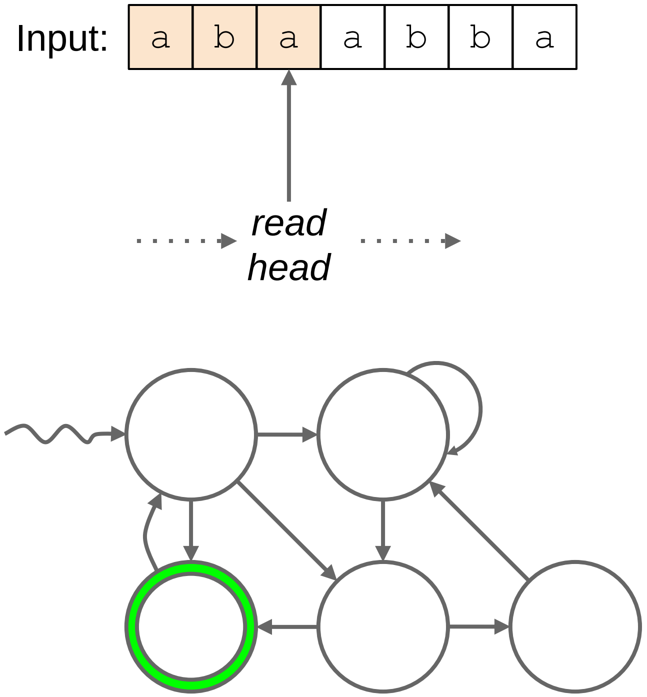
- E.g., Balanced parentheses, valid postfix expressions.

Q. Can you add even more strength to the machine and get even more languages?

The Ultimate Model of Computation

Augment the FSM with an unbounded data tape

- tape is initialized with the input word

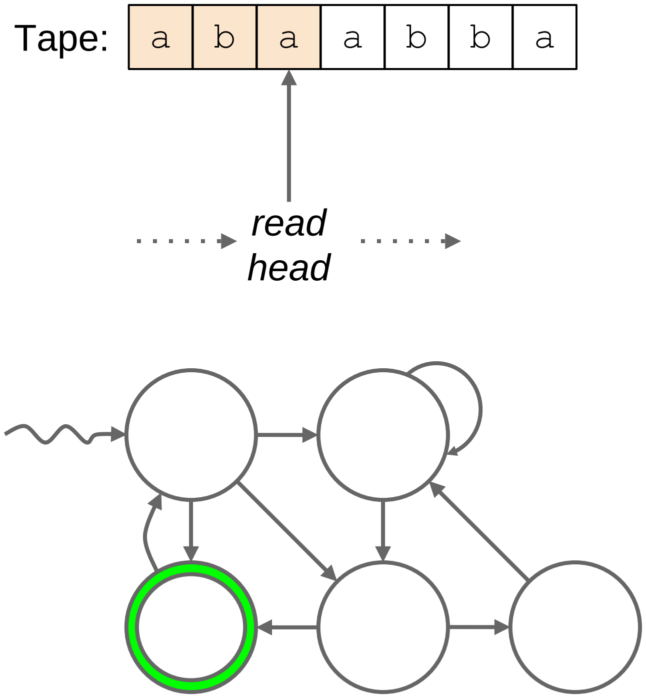


Finite State Machine

The Ultimate Model of Computation

Augment the FSM with an unbounded data tape

- tape is initialized with the input word
- allowed actions:
  - may read or write at current position

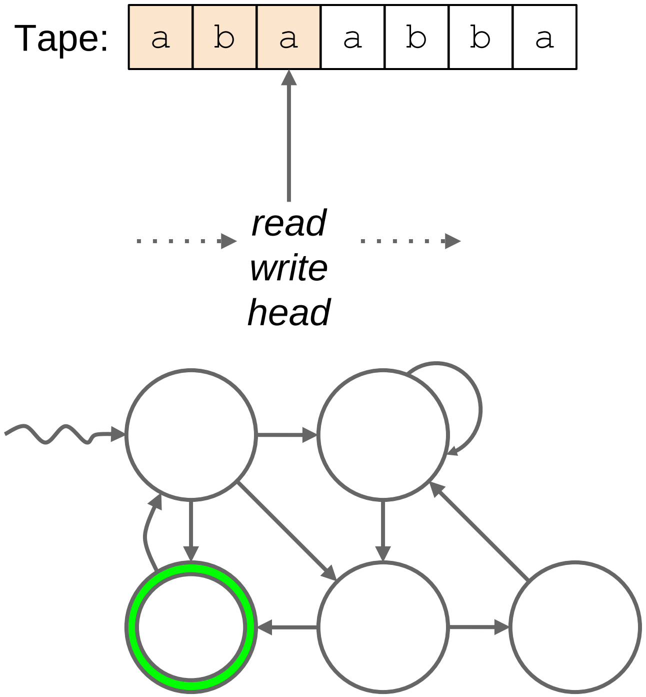


Finite State Machine

The Ultimate Model of Computation

Augment the FSM with an unbounded data tape

- tape is initialized with the input word
- allowed actions:
    - may read or write at current position

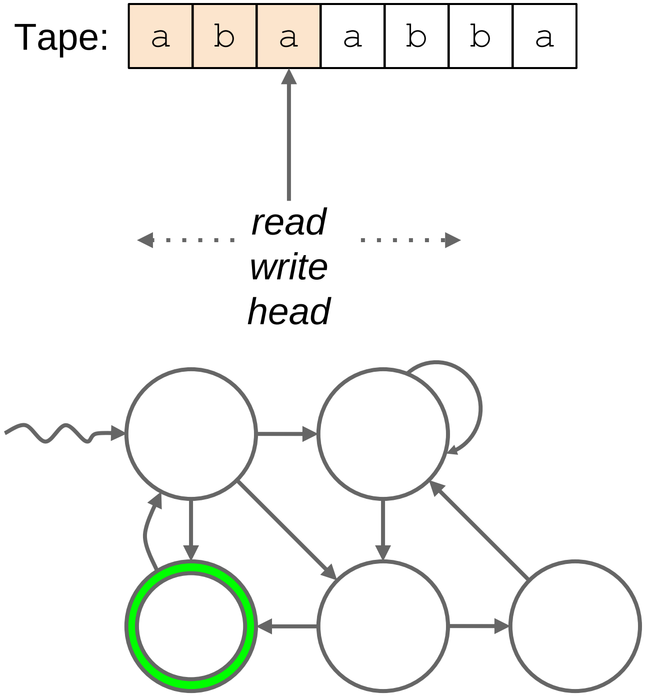


Finite State Machine

The Ultimate Model of Computation

Augment the FSM with an unbounded data tape

- tape is initialized with the input word
- allowed actions:
  - may read or write at current position
  - may move one step left or right



Finite State Machine

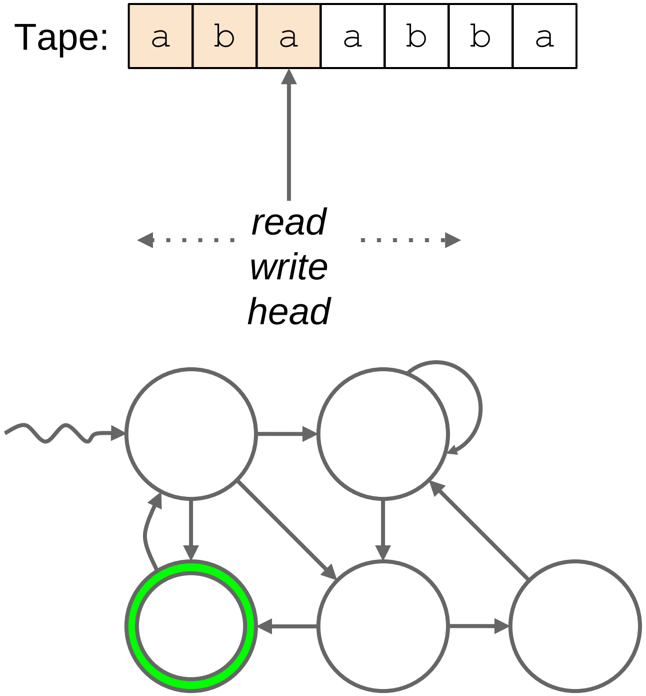
The Ultimate Model of Computation

Augment the FSM with an unbounded data tape

- tape is initialized with the input word
- allowed actions:
    - may read or write at current position
    - may move one step left or right

Alan Turing

Alonzo Church



Tape:							
	a	b	a	a	b	b	a

Turing Machine

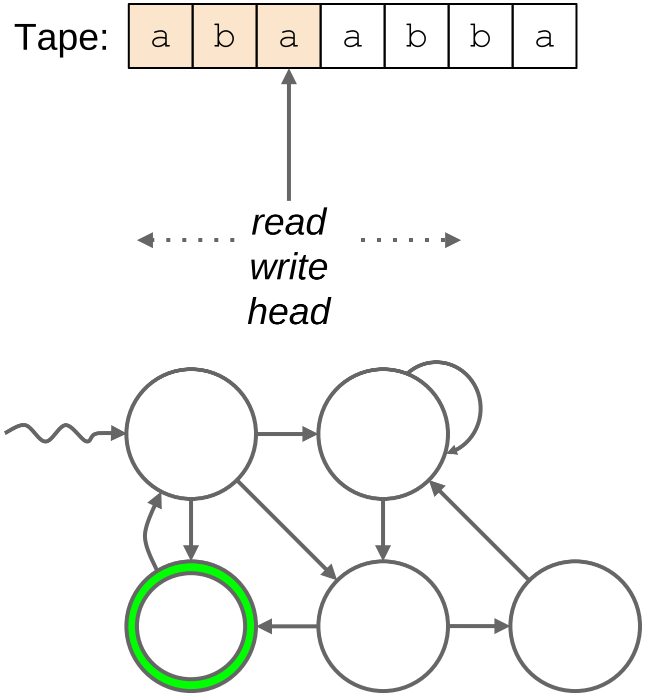
Church-Turing Thesis

Augment the FSM with an unbounded data tape

- tape is initialized with the input word
- allowed actions:
  - may read or write at current position
  - may move one step left or right

Alan Turing

Alonzo Church



Turing Machine