Informatics 1

Computation and Logic

Lecture 19

Computation: The Big Ideas

Regular expressions and finite automata

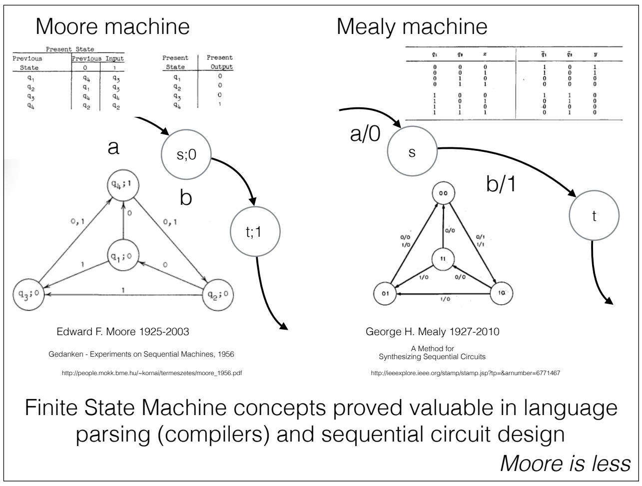
Let A be a class of such strings. We call A <u>regular</u>, if A can be described by an expression built out of the following operations (chosen in analogy to the definition of regular events in Sect. 7.1.)

The empty set and the unit set consisting of just $a_{\underline{i}}$ for any <u>i</u> are <u>regular</u>. If A and B are regular, so is their sum which we write $A \vee B$. If A and B are regular, so is the set, written AB, of strings obtained by writing a string belonging to A just left of a string belonging to B. If A and B are regular, so is A*B which abbreviates $\overbrace{A \cdots A}^{\underline{n}\text{ factors}}B$ ($\underline{n} \geq 0$), i.e., the sum of these classes for all $\underline{n} \geq 0$.

S. C. Kleene 1909–1994

Representation of events in nerve nets and finite automata 1951

https://www.rand.org/content/dam/rand/pubs/research_memoranda/2008/RM704.pdf


Moore machine
Mealy machine
a
s;0
b
t;1
a/0
s
b/1
t
Edward F. Moore 1925-2003
Gedanken - Experiments on Sequential Machines, 1956
http://people.mokk.bme.hu/~kornai/termeszetes/moore_1956.pdf
George H. Mealy 1927-2010
A Method for
Synthesizing Sequential Circuits
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6771467
Finite State Machine concepts proved valuable in language parsing (compilers) and sequential circuit design
Moore is less

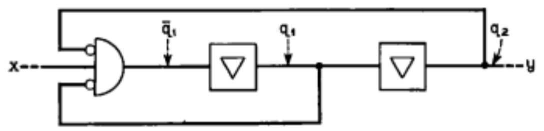
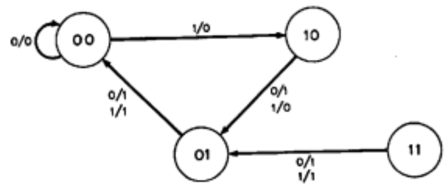


Fig. 5



A nondeterministic automaton has, at each stage of its operation, several choices of possible actions. This versatility enables us to construct very powerful automata using only a small number of internal states.

Nondeterministic automata, however, turn out to be equivalent to the usual automata. This fact is utilized for showing quickly that certain sets are definable by automata.

Dana S. Scott 1934-…

Michael O. Rabin 1931-…

Finite Automata and their Decision Problems 1959

http://www.cse.chalmers.se/~coquand/AUTOMATA/rs.pdf

Finite State Machine Parsing for Internet Protocols: Faster Than You Think (2014)

Parsers are responsible for translating unstructured, untrusted, opaque data to a structured, implicitly trusted, semantically meaningful format suitable for computing on. Parsers, therefore, are the components that facilitate the separation of data from computation and, hence, exist in nearly every conceivable useful computer system

Parsers must be correct, so that only valid input is blessed with trust; and they must be efficient so that enormous documents and torrential datastreams don't bring systems to their knees

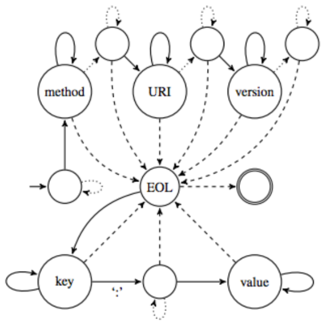
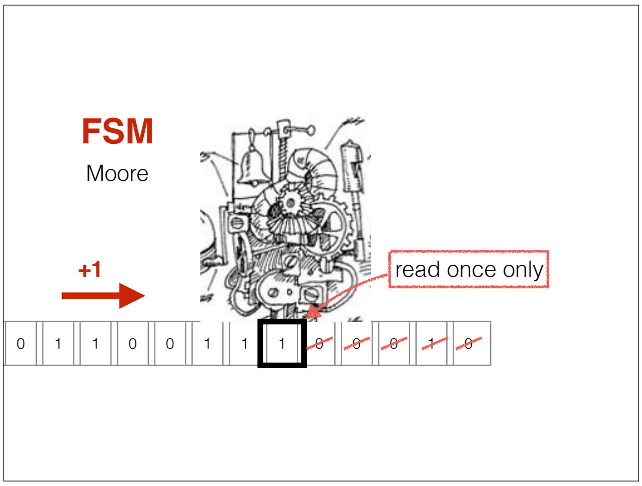
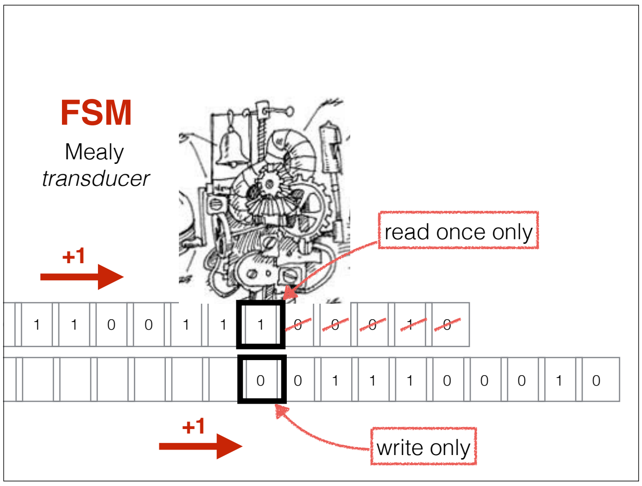


Fig. 2. DFA that recognizes HTTP headers, with the request on the first line, followed by arbitrary key-value pairs on subsequent lines, ending with two consecutive newlines. Solid edges represent any printable character, dotted lines represent a space, dashed lines represent a newline. The unlabeled states just eat spaces.

A Practical Introduction to Hardware/Software Codesign,
Chapter 4. Finite State Machine with Datapath (2010)

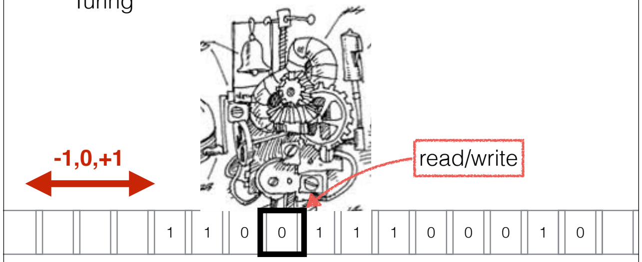
Abstract In this chapter, we introduce an important building block for efficient custom hardware design: the Finite State Machine with Datapath (FSMD). An FSMD combines a controller, modeled as a finite state machine (FSM) and a datapath. The datapath receives commands from the controller and performs operations as a result of executing those commands. The controller uses the results of data path operations to make decisions and to steer control flow. The FSMD model will be used throughout the remainder of the book as the reference model for the 'hardware' part of hardware/software codesign.


FSM
Moore
+1
read once only
0 1 1 0 0 1 1 1 0 0 0 1 0

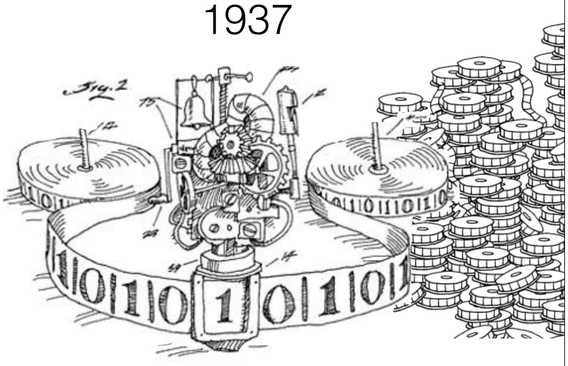

FSM
Mealy
transducer
+1
read once only
1 1 0 0 1 1 1 0 0 0 1 0
0 0 1 1 1 0 0 0 1 0
+1
write only

Turing Machine

Turing



Universal Turing Machine
1937



Alan Turing 1912-1954