COM2109

Automata

Kirill Bogdanov
and
Robert Hierons

Semester 1

Based on the work by Lucia Specia,
M.S. Moorthy and Prof Costas Busch

Where to find information

Slides and info: on MOLE.

Books:

- Dexter C. Kozen Automata and Computability.
- Introduction to the Theory of Computation

  Michael Sipser, 2nd edition

General Info

Lectures will be provided in the recorded form.

Tutorial sheets are problems you are expected to attempt in your own time. Solutions will be provided.

Threshold quizzes are multiple-choice/multiple-answer Blackboard quizzes testing your basic knowledge of the material. There will be 2 in each semester (Autumn/Spring). You need to pass all of them (by scoring at least 70% in each of the quizzes) in order to pass the whole module. If you do that, you get 40% (that is, a pass).
Practice questions will be provided.

The main exam in January
accounts for the remaining 60%.

General Info

Tutorials and help sessions starting from week 2,

Tutorials are on Fridays and cover

- solution to the tutorial sheets and
- sample threshold quiz questions.

Depending on the social distancing rules at the time the above might be held online or in-person. This will be announced in due course.

General Info

Help sessions are starting in week 2,

- Take place on Wednesdays 9am-10am
- Demonstrators will answer your questions.
- You will need to book a Google Meet meeting covering the tutorial slot and submit
  - The URL of the meeting
  - Description of your problem

  via Google form. A demonstrator (or lecturer) will join your meeting.

Details how to book meetings are on Blackboard.

Today's lecture

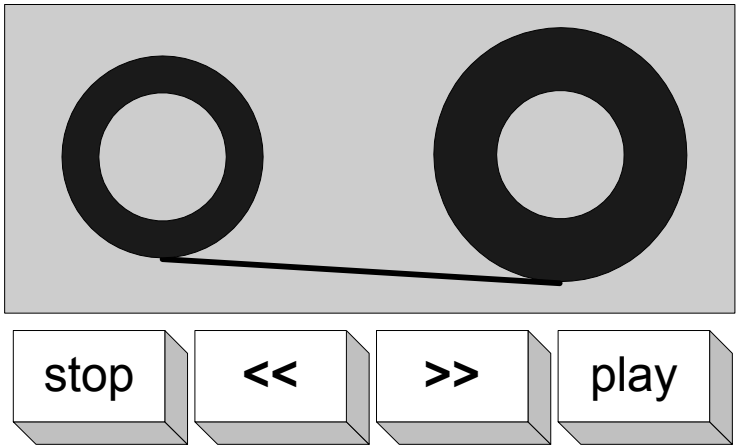
- Introduction to automata
- Types of automata
- Types of computational problems
- Structure of the course

Finite-state machines

- Used to describe and analyse software and hardware systems,
- Widely used in industry,
- A variety of different kinds of machines is also used.

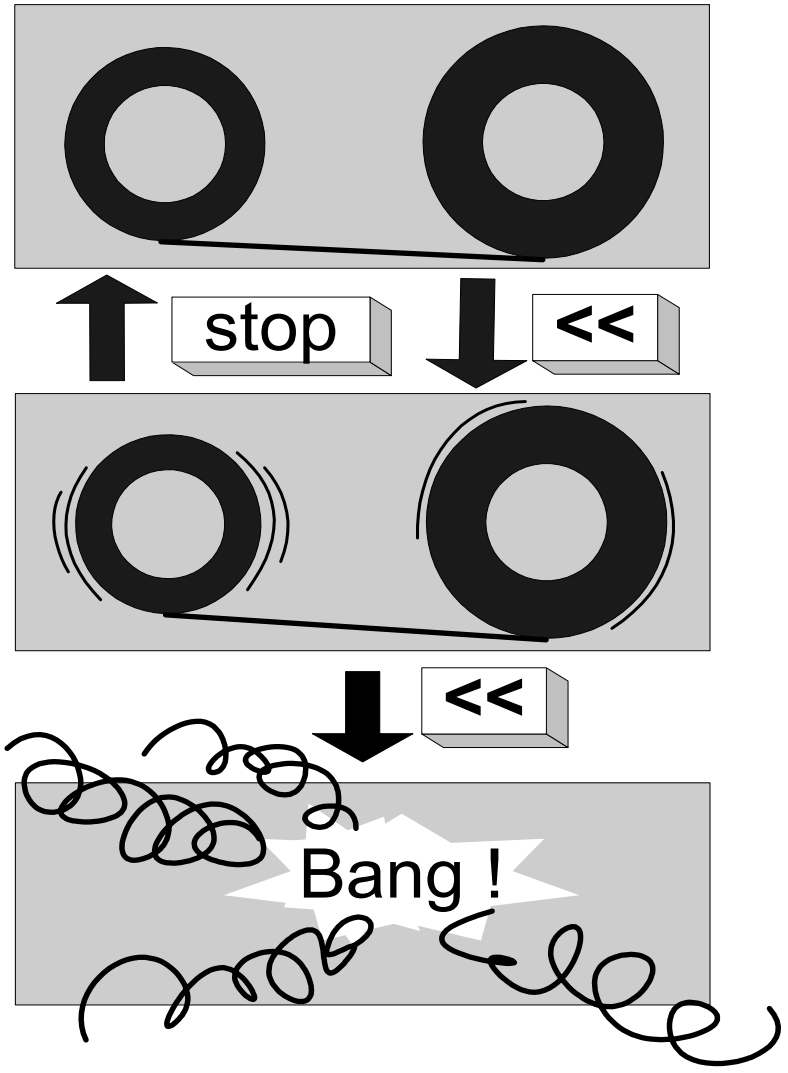
A tape recorder can

- Play tapes,
- Forward advance
- Rewind



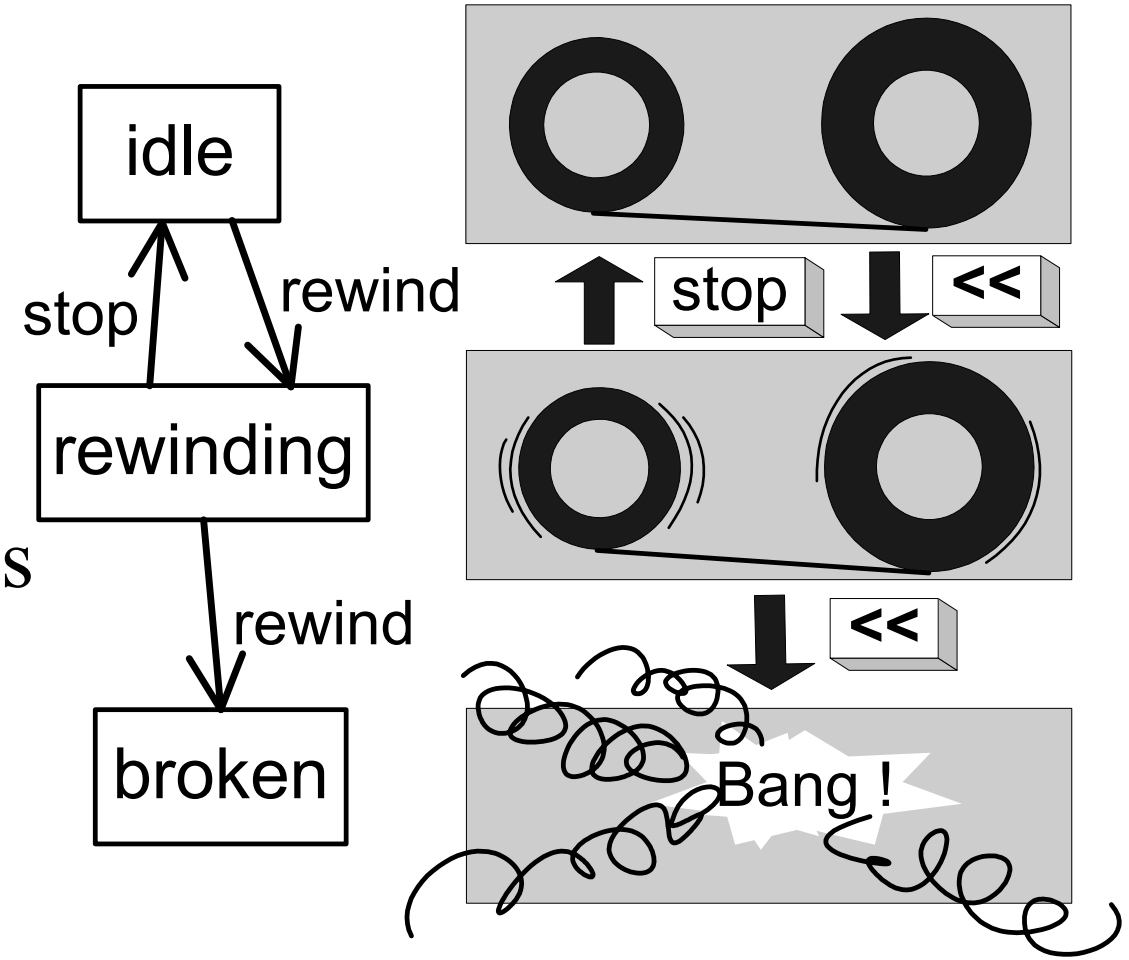
How it works

- **Behaviour** depends on **button** pressed and **mode** of operation
- Buttons are *inputs*
- Modes are *states*
- Mode changes are *transitions*

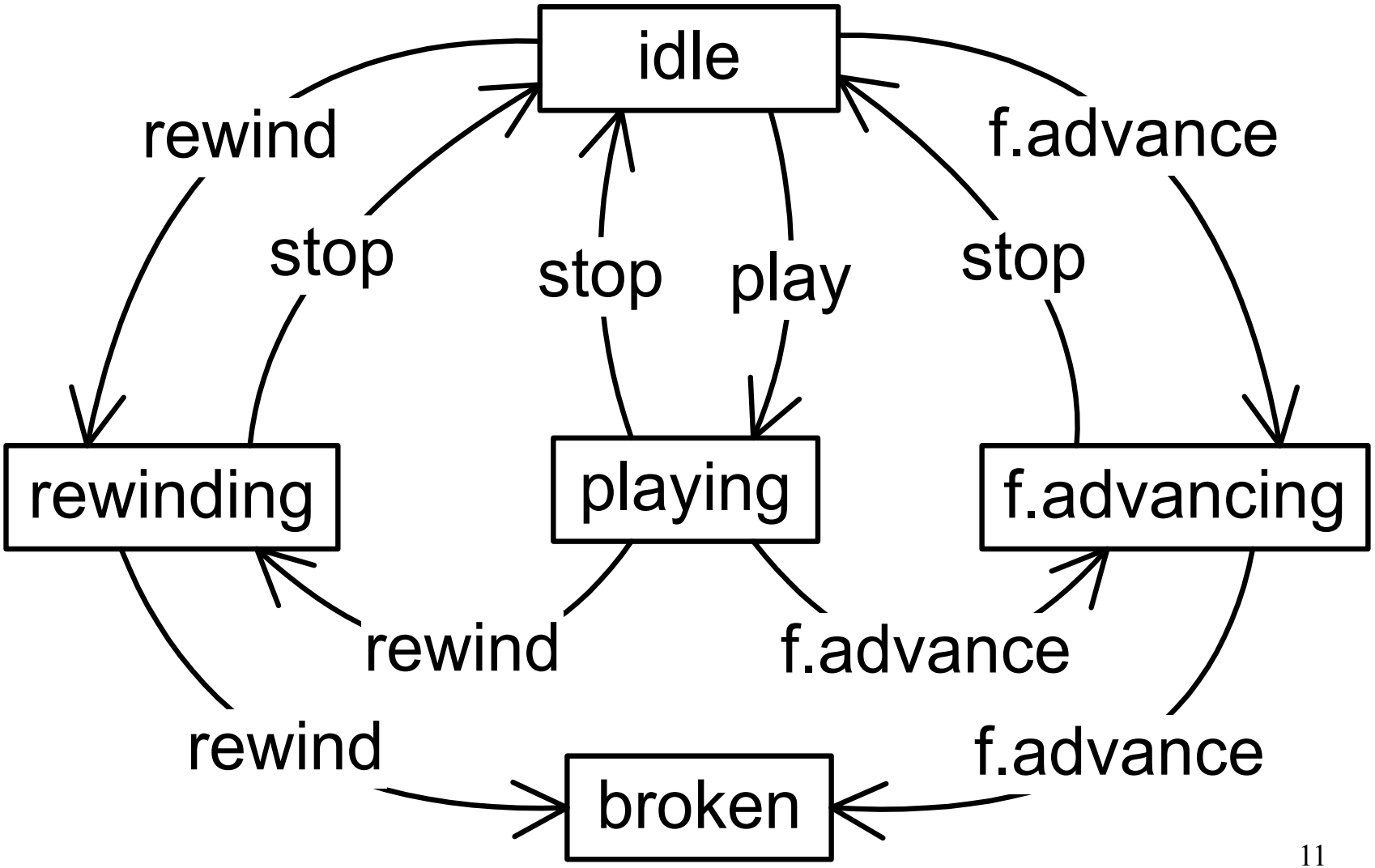


Finite-state machine

- Rectangles are states
- Arrows are transitions
- Labels on arrows are the inputs causing those transitions



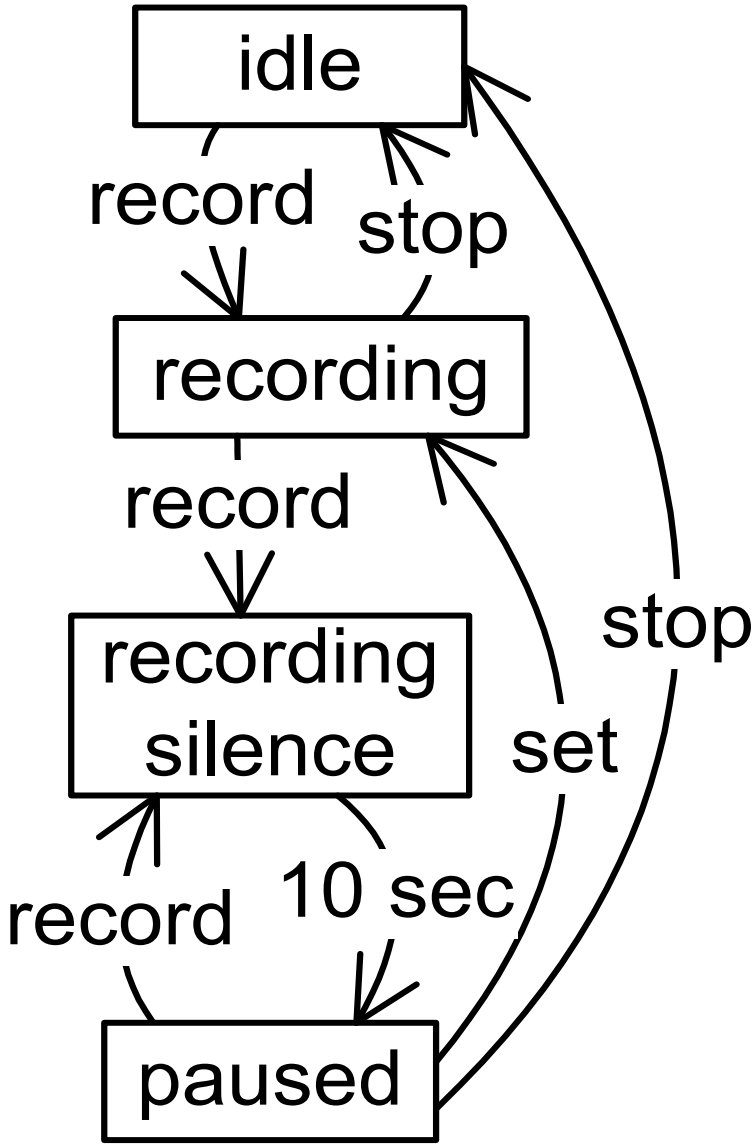
More details of the tape-recorder



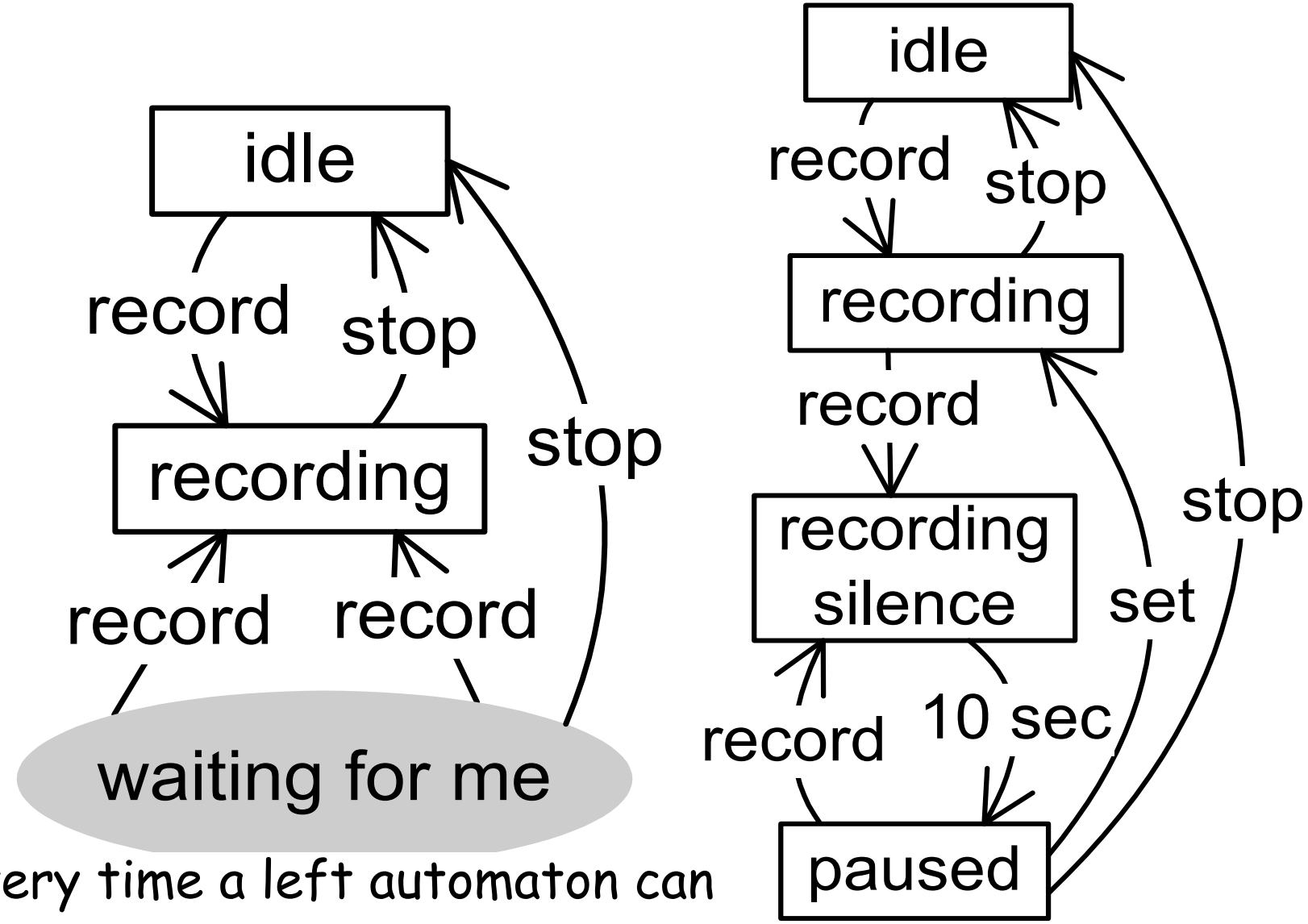
Case study

This is a fragment of a model of a real hi-fi system

There is something rather with this system.



Model checking aims to check that a model satisfies a specific property.

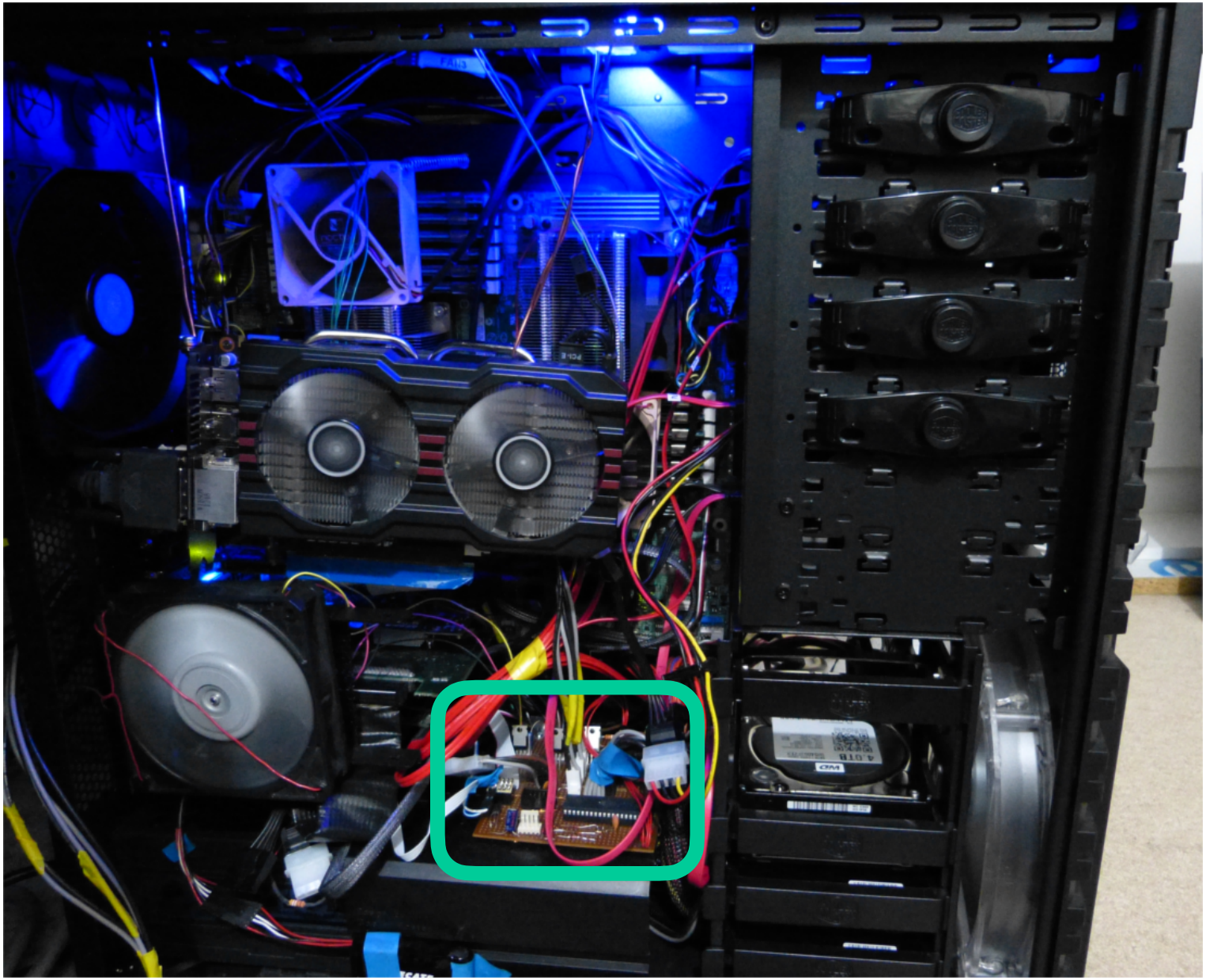


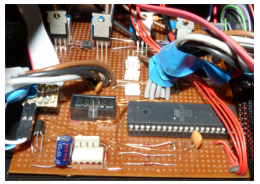
Every time a left automaton can make a transition, it should be matched by one on the right.

More examples:

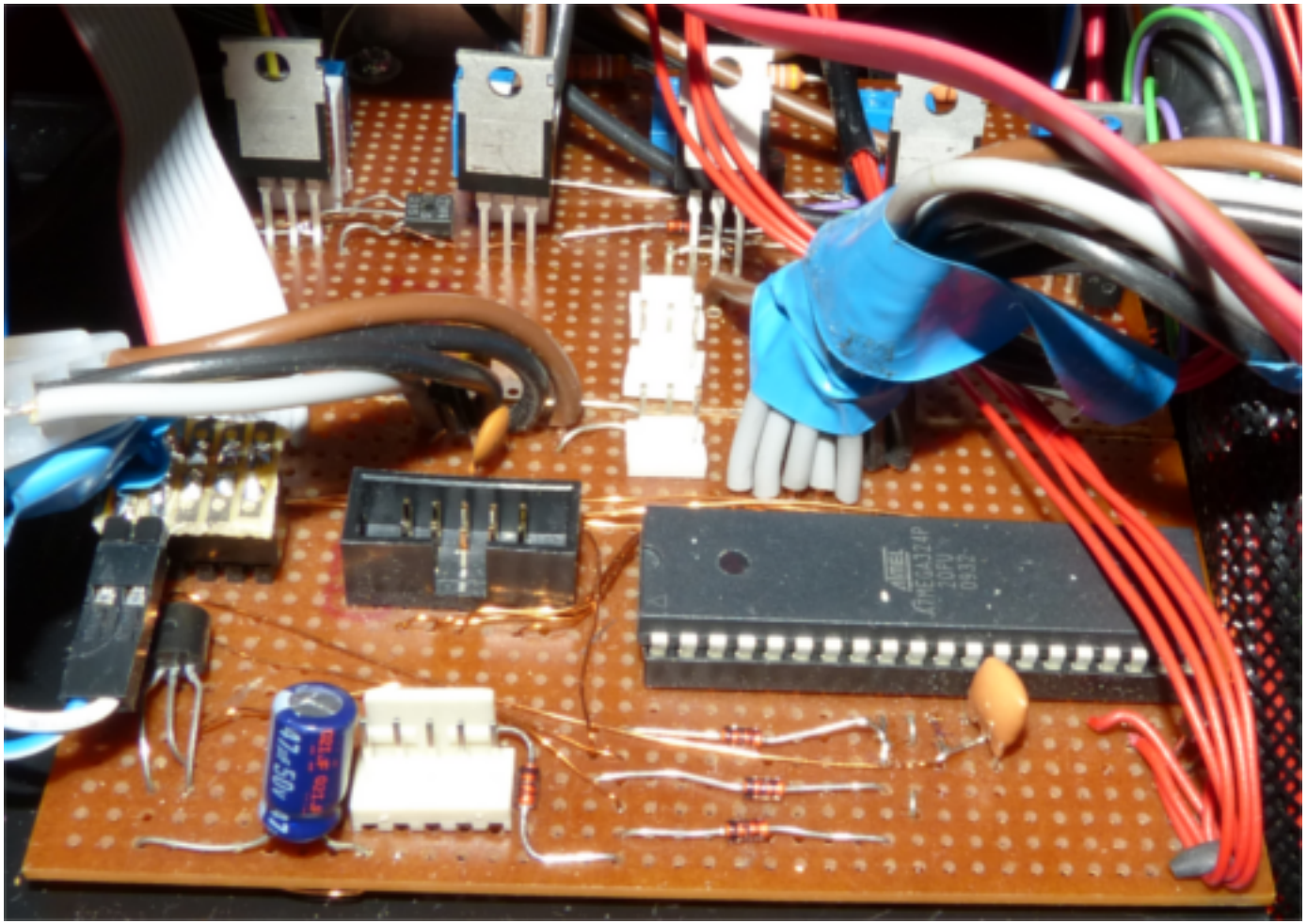
Web app and
making & drinking coffee

Model inference





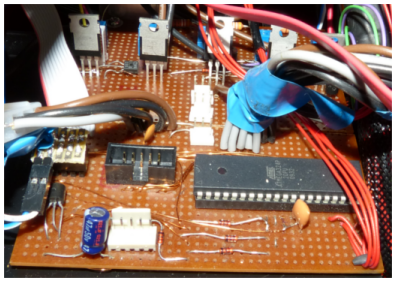
Fan controller

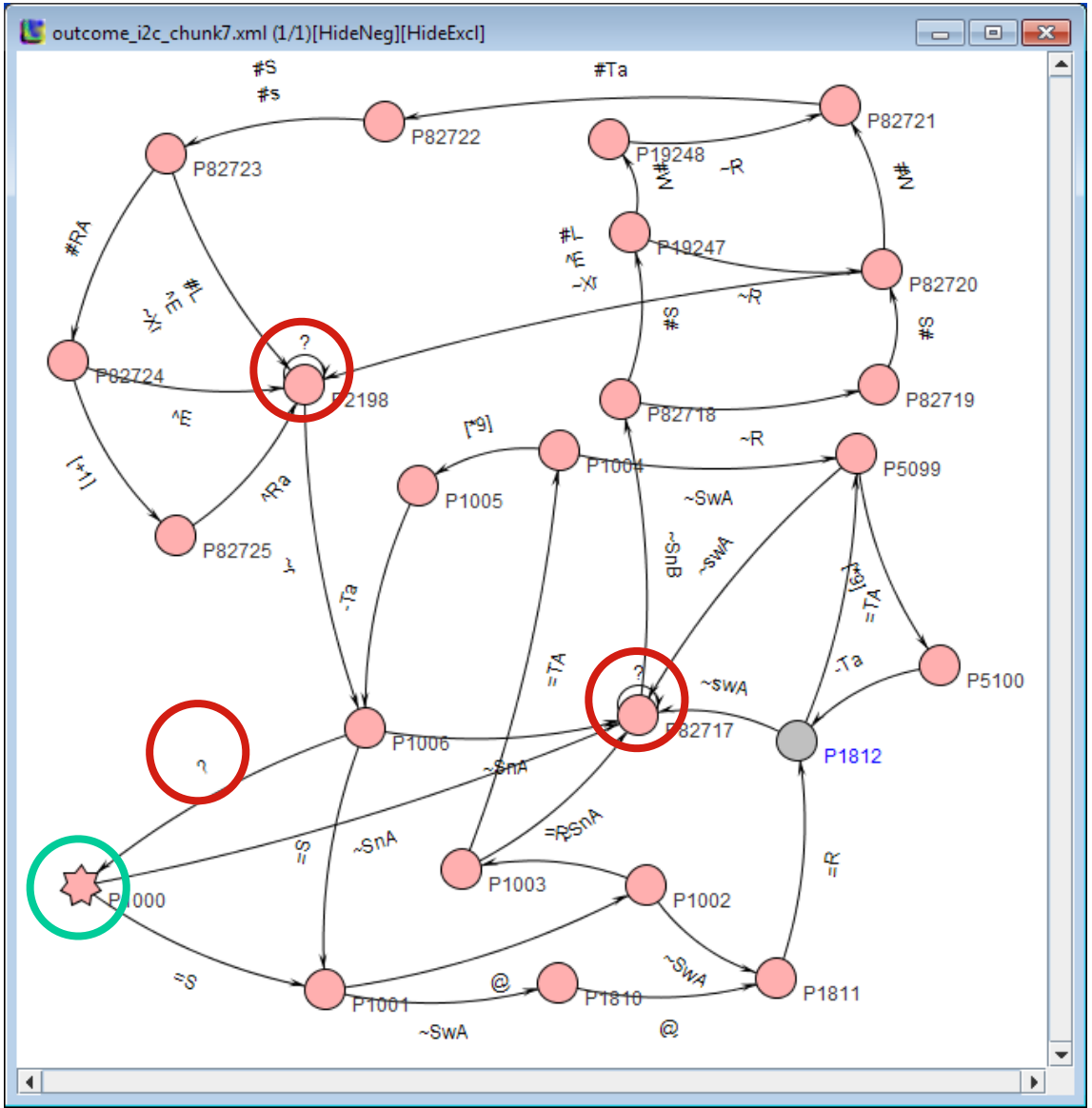


Problem: occasionally slow to respond

Fan controller

solution: log a sequence of events and *infer* a finite-state machine



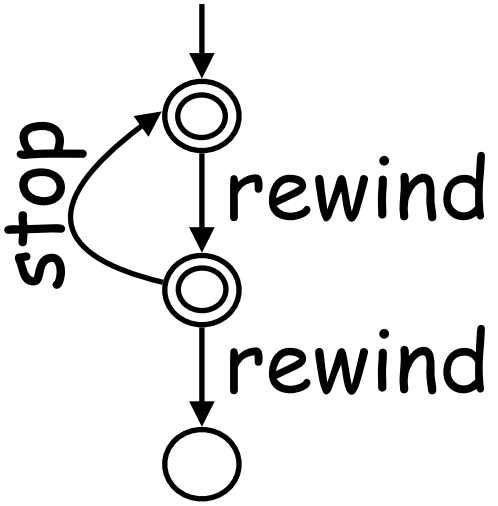


Automata are used in:

- model checking - given an automaton describing a system, check if it satisfies a property (e.g. hi-fi system).
- modelling software and systems (e.g. web application and making & drinking coffee).
- testing of software (e.g. coffee) - there are methods that are proven to find all faults in the transition diagram.
- learning the behaviour of complex systems (e.g. inference).

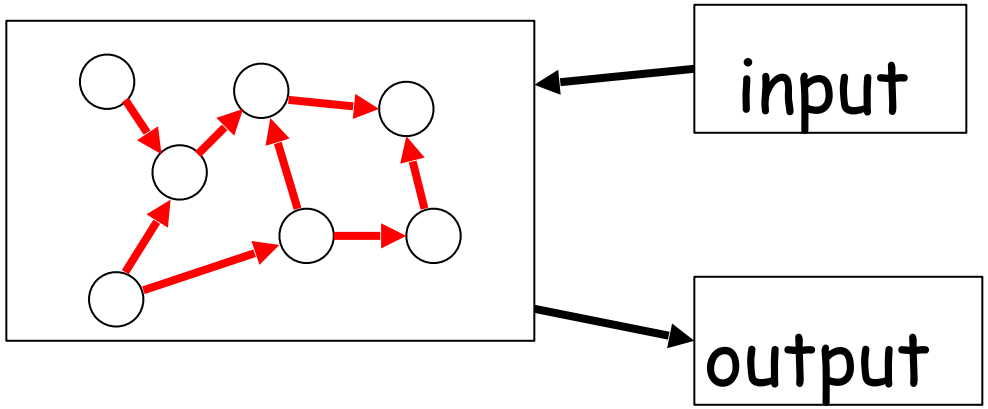
Finite Automaton

- **alphabet** is a set of symbols, such as { stop, rewind }
- **input** is a **sequence** of elements from an **alphabet** such as rewind, stop, rewind
- **output** is either 'accept' or 'reject'.
- An input sequence is accepted if automaton entered an accept state after going through it.
- We can talk about sets of sequences accepted by an automaton or how to build an automaton that accepts a specific set of sequences.



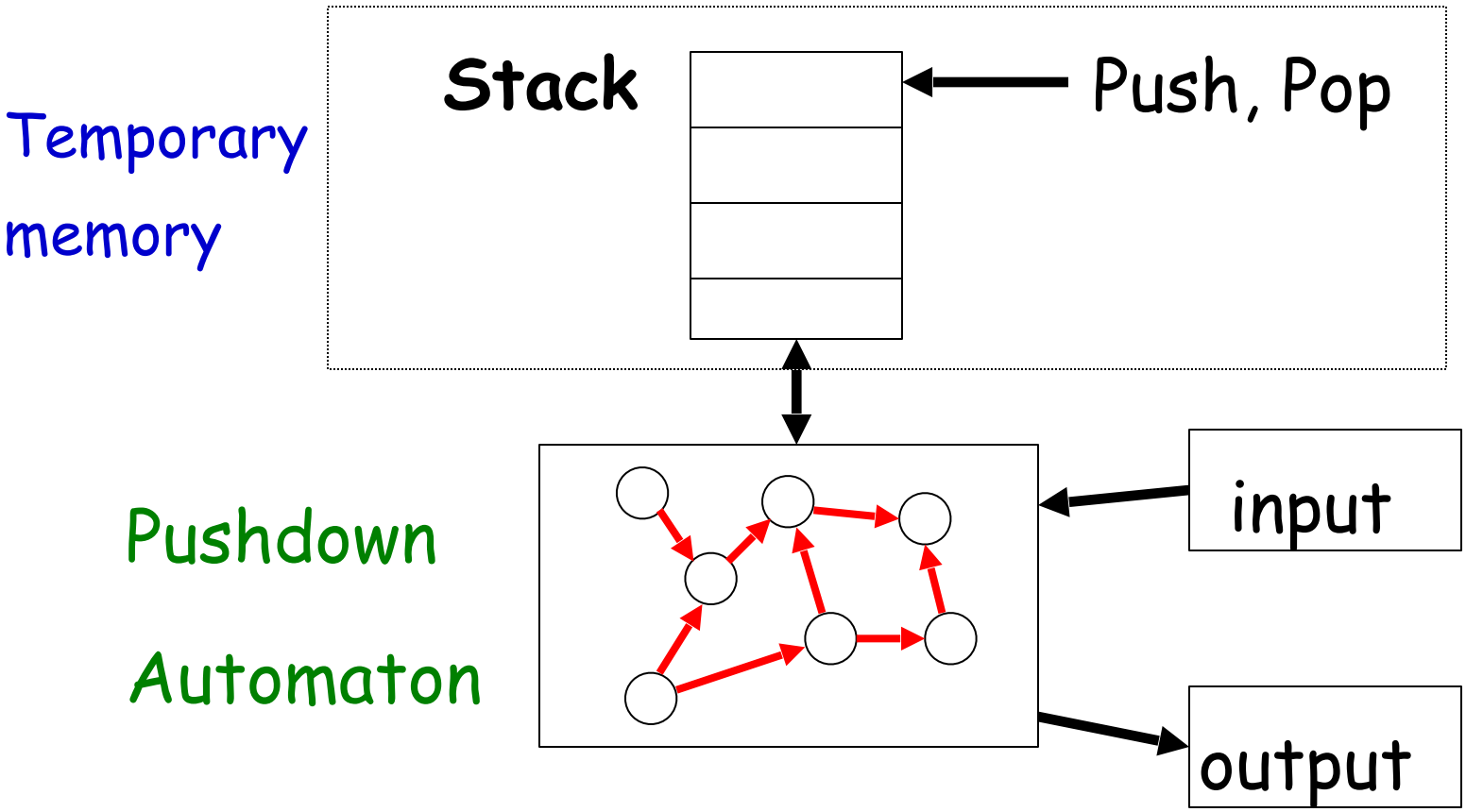
Finite Automaton - small computing power

Finite Automaton



E.g.: Elevators, vending machines, automatic doors

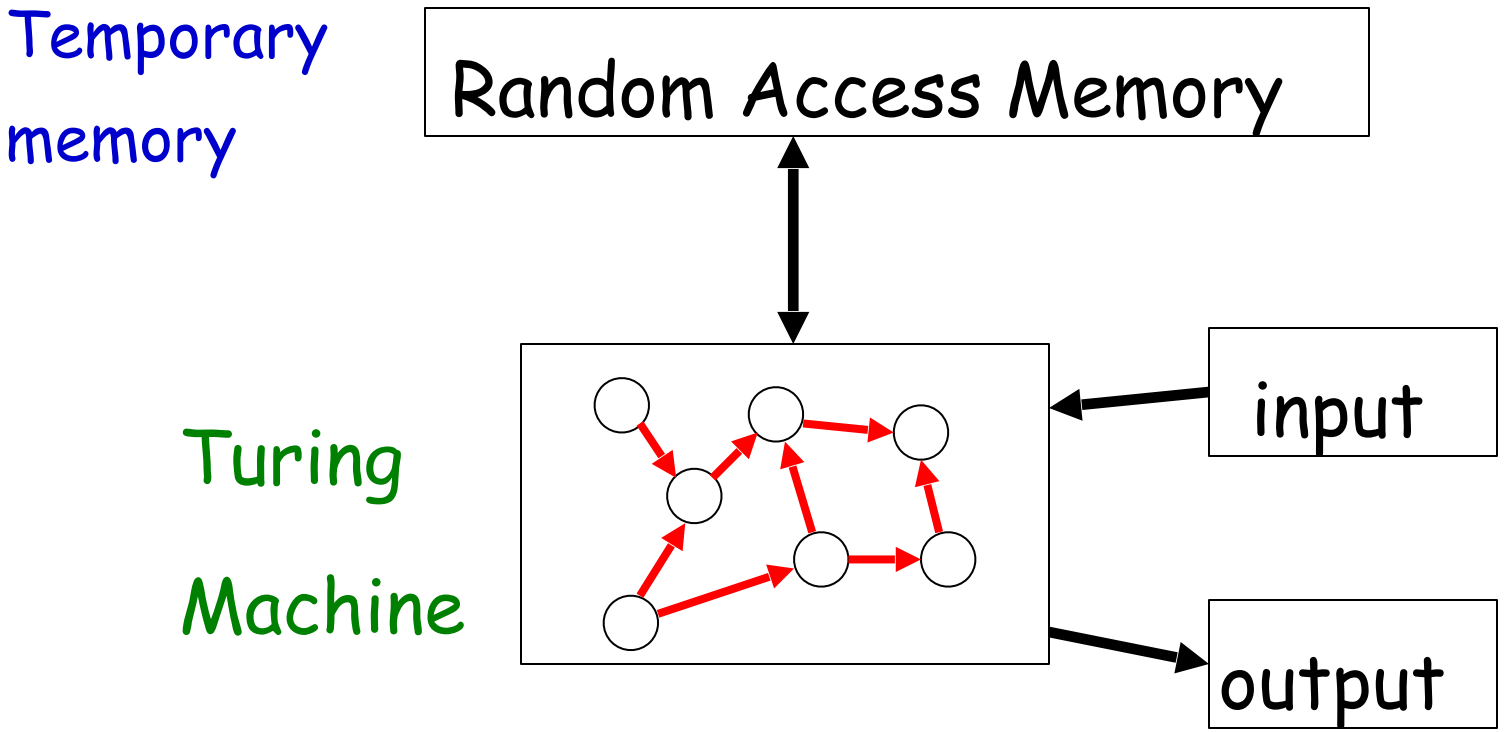
Pushdown Automaton - medium computing power)



E.g.: Models of programs where

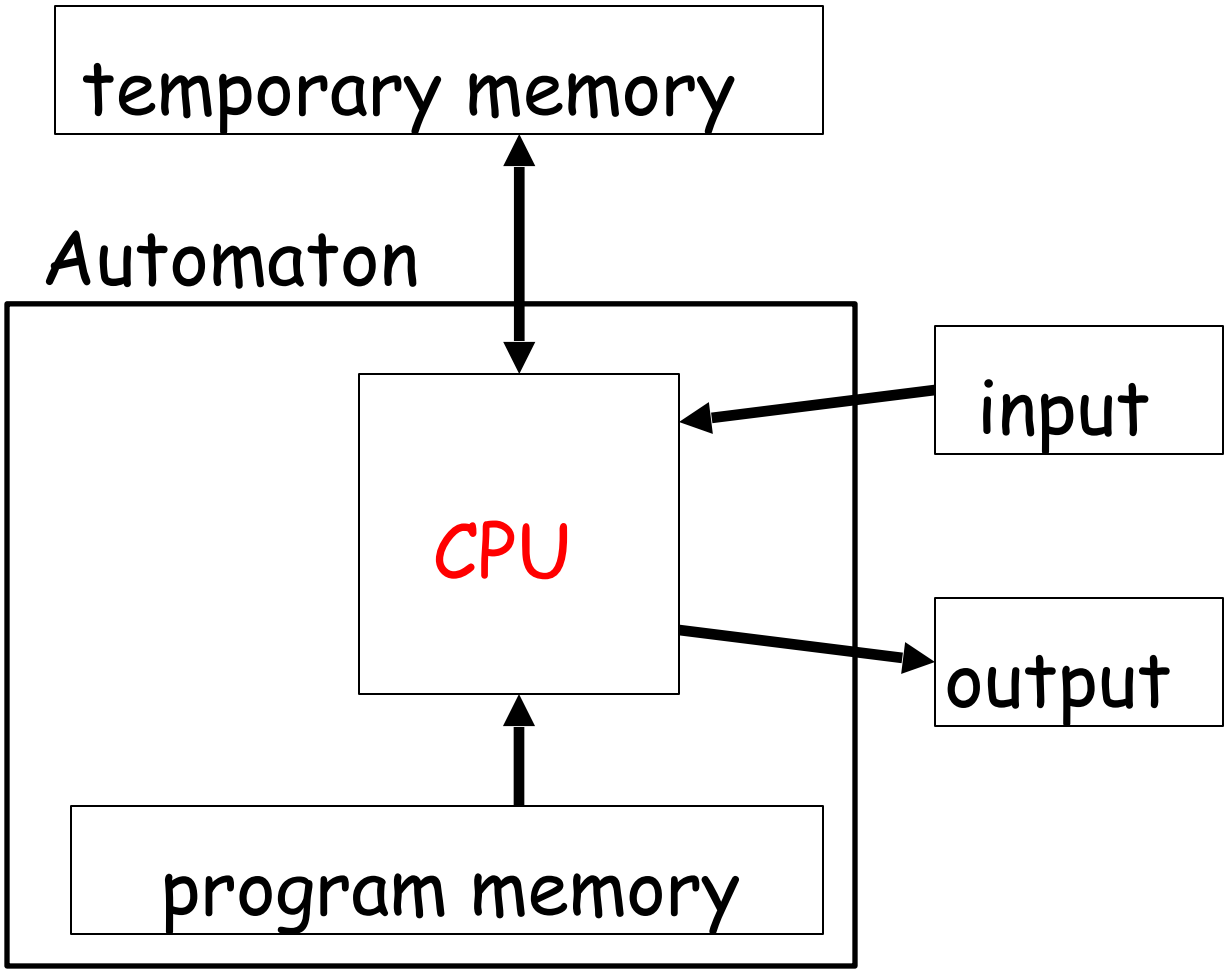
- each function is described by an automaton and
- each function can call another function (using call stack)

Turing Machine - highest computing power



E.g.: Any algorithm

Automata as computational model of a computer



Different Kinds of Automata

Automata are distinguished by their temporary memory:

- **Finite Automata:** no temporary memory
- **Pushdown Automata:** stack
- **Turing Machines:** random access memory

Turing Machine is the most powerful computational model known

Question: Are there computational problems that a Turing Machine cannot solve?

Answer: Yes (unsolvable problems)

This is important: it shows that with conventional computers, there problems that cannot be solved.

Many interesting problems are such, hence either

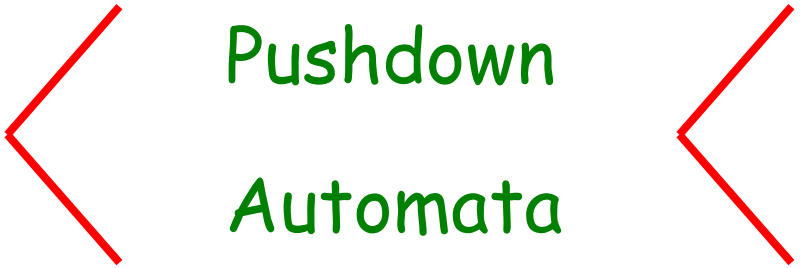
- build different computers (such as quantum), or
- cheat (build solutions that work most of the time)

Power of Automata

Simple Problems

More complex problems

Hardest problems



Less power

More power

Solve more

computational problems

Structure of the course

- Regular languages
  - Finite automata
  - Non-deterministic automata
  - Regular expressions
- Context-free languages
  - Context-free grammars
  - Push-down automata
- Turing machines:
  computability and decidability.