Finite State Automata (Speaker Notes)

Slide 1

No speaker notes for this slide.

CS UNPLUGGED at a distance

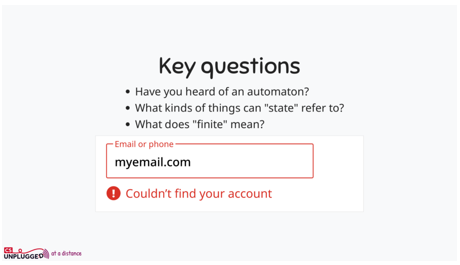
Finite State Automata

Slide 2

By the end of this session we'll know what a finite state automaton, or machine, is, and how important they are in an area called “formal languages”. On the way we will find out how a computer interface knows if an email address is valid, and how a computer can make sense of the commands we write in a program.

The title, finite state automata, might seem like strange jargon, but before we start, has anyone heard of an automaton? Put your hand up or share your experience in the chat.

And what does finite mean?


Key questions

- Have you heard of an automaton?
- What kinds of things can "state" refer to?
- What does "finite" mean?

Give time for participants to respond in the chat and keep an eye out for any raised hands, either digitally or physically.

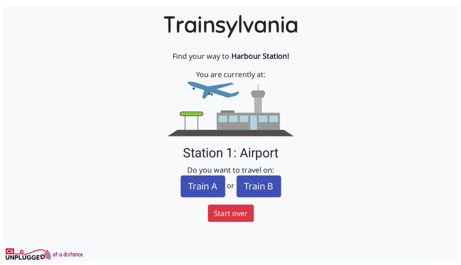
Answers for 'automaton': Participants may be aware of old-fashioned machines, toys and dolls that are self-operating; clocks such as cuckoo clocks or those where characters come out on the hour. They may have also heard the word used to signify boring manual labour, doing the same thing over and over.

Answers for 'state': To be added...

Answers for 'finite': They may know that it means that there is a limited number of items.

I can see from the responses... [make an appropriate comment, such as “several of you have had experience with automaton”, or “this session will be new learning for many of you.”]. *Automaton* essentially means a simple machine, like a wind-up toy. The plural of automaton is *automata*.

Slide 3


Trainsylvania

Find your way to **Harbour Station!**

You are currently at:

Station 1: Airport

Do you want to travel on:

We're going to use a map of a commuting service to help explain finite state automata. Have your pen and paper ready. I'm going to use this interactive to help you create a map of a city called Trainsylvania. Trainsylvania has several train stations, but there are no maps showing where the trains go. We are starting at the train station called **Station 1: Airport**.

Write down 'Airport Station' on your paper, or just a '1' is fine.

We want to get to **Station 5: Harbour**. We have been told there are two trains that leave from each station, the **A-train** and the **B-train**. We have been reassured that by **taking** a sequence of A and B trains we can get from the Airport Station to the Harbour Station. Our goal is to find a sequence to do this.

Remember, we are starting at Airport Station. In the chat tell me which train you would like to **take**. Type "A" for train A or "B" for train B. The majority will decide which train we catch. We will choose which train to **take** from each station in this way for the rest of the activity. At this stage you might as well make random choices since there's no information to base the decision on, so just quickly type whichever you want, A or B. Later you'll have more clues on which train to **take**.

Allow participants time to set up their map. Wait for a reasonable number of responses and then say the majority choice. It needn't be a carefully counted vote; at this stage it really doesn't matter too much, so just base it on your impression of which is the most common choice.

You have chosen train [A or B] and it goes to [click on Train A or Train B to find out]. You may want to write this information down. We are now at [read out the new station name]. Would you like to catch train A or train B next?

Continue like this, asking which train the participants would like to catch (A or B) and telling them where it has taken them, until they have been to 4 or 5 different stations (or they get to Harbour Station, if they are lucky!). Record the decisions they made at each station on your own map so you can refer back to it if you need to.

If participants' selections seem to be going in circles, ask them questions about how they are recording the information on their paper/map. We're starting by not saying how to record the information, but they are likely to come up with the idea of drawing lines between cities, and probably arrows since the trains only go one way; and they are also likely to realise that they should put letter names on the lines!

After 3 to 5 attempts...

I'm going to pause us for a moment now to collect our thoughts. What do we know so far about where you can travel to, or from? What have you noticed? Is there a pattern forming? What information is missing that would help us get closer to the Harbour Station? Share what you know, and would like to find out, in the chat.

Read out the answers in the chat as they come in and comment as appropriate. For example, “yes, we know that Train A from Station 1: Airport takes us to Station 2: West.” Encourage problem solving conversations to find a successful route. This may be done through the chat or some other form of communication tool. If you have a small group, then you could unmute the participants and let them discuss their ideas.

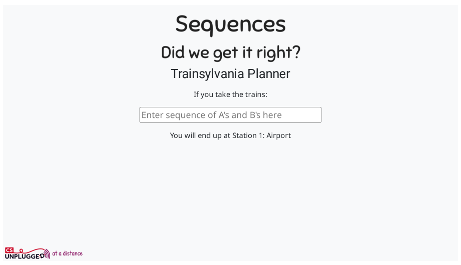
Okay. We've worked out what we do and don't know. Let's see if we can get to Harbour Station! Where are we? And which train would we like to catch?

Continue asking where participants would like to go and telling them where their choice has taken them until you reach Harbour Station.

Slide 4

Read once you reach Harbour Station...

Congratulations! We've made it to Harbour Station. Let's see if we have a sequence of A's and B's that will get us from the Airport to the Harbour. Please type in the route you would take in the chat using just As and Bs. I will copy it into the Trainsylvania Planner and see if it agrees that your route will work. The technical term for this sequence of letters is a “string”. So that I can copy and paste it quickly, please don't put any spaces or commas between your letters.



Sequences

Did we get it right?

Trainsylvania Planner

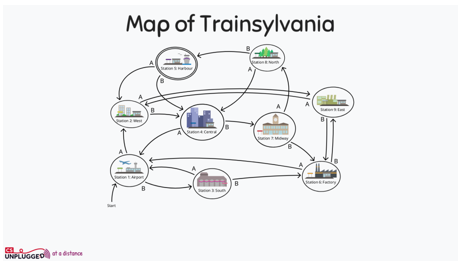
If you take the trains:

Enter sequence of A's and B's here

You will end up at Station 1: Airport

Wait for several responses. Several people will probably type in the same string of As and Bs. When this happens type the sequence of A's and B's into the text box on the planner, and it will say what station that sequence takes you to.

Slide 5



Here is the master map for this activity that you probably would have appreciated seeing earlier! How similar is it to the one you drew? Your map might have a completely different layout, particularly if you started with Airport Station in the middle of your paper, but hopefully your lines are still labelled the same between each station.

What would have happened (or did happen!) if you hadn't labelled your lines with As or Bs? Or if you didn't put the directional arrows in? As you can see on the map, even though train A took you from Airport Station to West Station, catching Train A from West Station doesn't take you back to Aiport Station, but rather to East Station.

Wait for responses in the chat and respond appropriately. Remember, it could take a while for participants to type their longer responses for this answer.

When teaching educators: With students, we deliberately let them find out as they do the activity that arrows and labels are important, because in the process they will invent for themselves the notation of a finite state automata!

Without the As and Bs it would have been confusing as to which line represented which train. Without arrows it would have been easy to forget which train went to the station and which was leaving it.

Don't be too hard on yourself, we didn't say at the beginning that you couldn't get back to your starting point on the same train you took away from it.

You can see how having arrows on each line and labelling them with As or Bs makes it easier to describe the final successful route from the Airport Station to the Harbour Station.

These lines with arrows represent what are called 'transitions'. This map represents a common idea used by computer scientists for processing complicated input for a computer, and is not really for mapping train stations (in case you hadn't realised!). We are now going to explore this notation and how it relates to computer science.

You can see in the image that each station has a circle around it. The circle indicates that this is a 'state'. The starting state (Airport Station) is shown with an arrow pointing at it labelled 'Start'.

The final destination (Harbour Station) is indicated with a double circle around it. This is called the 'final', or 'accepting' state. It is the state that says “Yes, your sequence (string) is in this machine's language, it works.” In other words, in our example any string of As and Bs that gets us from the Airport Station to the Harbour Station is accepted as valid, i.e. will get us to the 'accepting' state. It's important to note that with a finite state automata, we're not worried about the shortest way to get to the accepting state, but just whether or not you end up there for a given string.

Now that you have this map, I am going to give you a string of As and Bs. Each A or B is a transition, (railway line), between the states (stations). The starting state is Airport Station like before, and we'll be transitioning to other states. I'll give you the string, you tell me which state we end up at, that is, which train station. To make it quicker, you only need to type its number or the first three letters of the station.

Slide 6

The first string is BBB. Which station will we end up at by following this string?

Wait for answers in the chat. Once you are ready to continue, press the next slide button/key to show the answer.

Yes, that's right, BBB will take us to East Station (via South Station and Factory Station).

Question 1

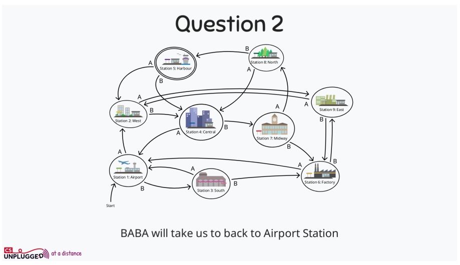
BBB will take us to East Station

Slide 7

How about BABA, where will this take us?.

Wait for responses, then push the next slide button/key to show the answer.

From Airport station back to South station, back Airport Station, twice!


Question 2

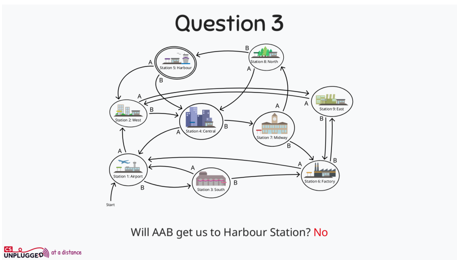
BABA will take us to back to Airport Station

Slide 8

To mix it up a bit, will this string get me to the Harbour Station? AAB. Yes or no? Type your answer in the chat.

Type AAB in the chat and wait for responses.

The answer is 'no'.


Question 3

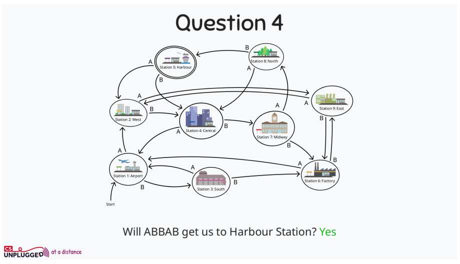
Will AAB get us to Harbour Station? No

Slide 9

What about ABBAB? Yes or no? Type your answer in the chat.

Type AAB in the chat and wait for responses.

The answer is 'yes'.


Question 4

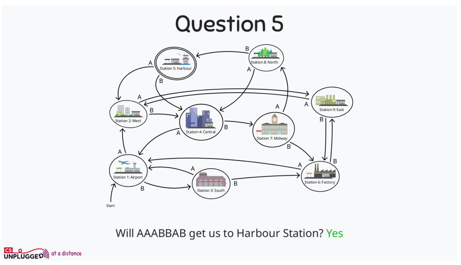
Will ABBAB get us to Harbour Station? Yes

Slide 10

What about AAABBAB? Yes or no? Type your answer in the chat.

Type AAB in the chat and wait for responses.

The answer is 'yes'.



Question 5

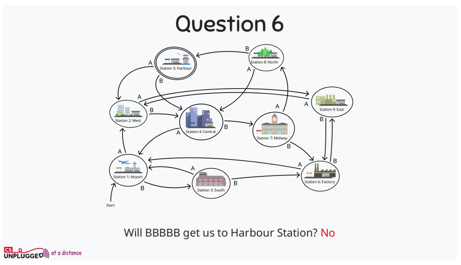
Will AAABBAB get us to Harbour Station? Yes

Slide 11

What about BBBBB? Yes or no? Type your answer in the chat.

Type AAB in the chat and wait for responses.

The answer is 'no'.



Question 6

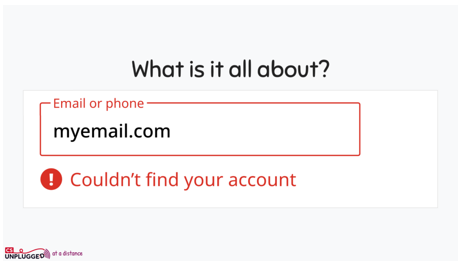
Will BBBBB get us to Harbour Station? No

Slide 12

What is a finite state automata used for in the real world? The purpose of a finite state automaton is to find out if the input string is correct in the language defined by the particular automaton. In other words, have we, the user, given the correct input into a device, whether it be by typing, or pressing buttons, to get the desired result?

Just as English is a language, there are also languages in computer science. In computer science languages there are very strict rules that can be used to check the input (for example, an email address can have strictly one "@" sign in it). Because these languages are very well defined, they are called formal languages. In contrast, natural languages (like English) aren't strictly defined, and people may argue about whether a particular spelling for, or pronunciation of, a word is correct.

Formal languages are very useful in computing. Before a computer attempts to process input information, it checks to see if the character inputs (the symbols, letters, or button presses used in the input) are in the correct language or not.


What is it all about?

Email or phone

myemail.com

Couldn't find your account

Slide 13

For a natural language like English, an "English checker" would ask is this English language: "I'm gonna be late."

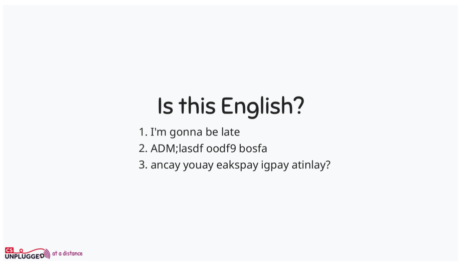
Push the next button to show each prompt individually.

Is this English: "Adm;lasdf oodf9 bosfa"?

Is this English? "ancay youay eakspay igpay atinlay?".

A finite state automaton is one way of applying the strict yes or no rules for the formal machine language. For example, the language for an email address can be expressed using this finite state automaton...

If any one asks, the third bullet points is written in Pig Latin.


Is this English?

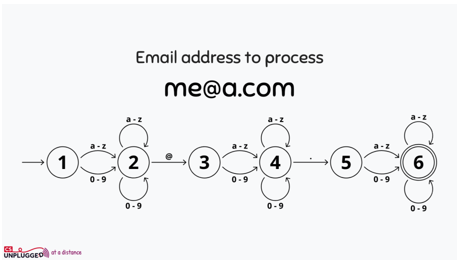
1. I'm gonna be late
2. ADM;lasdf oodf9 bosfa
3. ancay youay eakspay igpay atinlay?

Slide 14

Suppose someone types in the email address “me@a.com”. This finite state automaton shows that it (the machine) will start at 1 and move onto 2 when the “m” is processed (since it's in the range a-z). The next letter in the string is “e”, and this follows the loop back to 2. It will stay in the 2 state for any number of letters or numbers until an @ symbol is entered, which will move it on to 3. Here it will move on the “a” to 4, and then the full stop (.) will move onto 5. The next character, “c” will move it to the 'accepting' state, 6, and the “o” and “m” will make it stay in that state. Because 6 has the double circle, it's an “accepting state”, and tells us that “me@a.com” follows the rules for an email address. Of course, it doesn't mean that it's a valid one, but at least we know that it is in the right format.

For this finite state automaton the shortest input that will get it to the accepting state (6) is any character followed by an @ symbol, followed by another character, a full stop, and then another character, for example “a@b.c”. You can see why we're not usually interested in the shortest path to the accepting state, since it probably corresponds to a very simple string. A finite state automata is about whether or not a particular string gets accepted.

One the other hand, the email address “jo@” would get to 3, but if that is all of the input, then it isn't accepted because state 3 doesn't have the double circle. It would show an error message to warn us that the address doesn't look right.

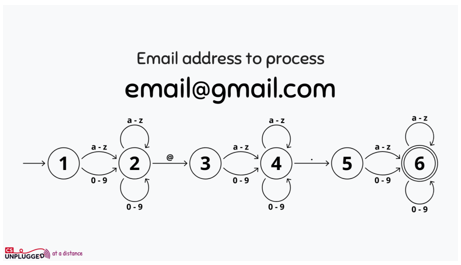

Email address to process

me@a.com

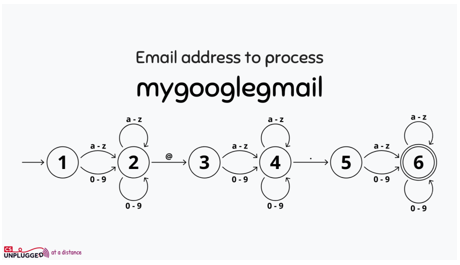
Slide 15

Using the text “email@gmail.com”, does it get to the accepting state? Let's step through it together. The beginning E takes us to 2, then the next M, A, I and L stay in 2. The @ moves it to state 3. The remaining characters go to 4, then 5 with the dot, and 6 with the characters after the dot. 6 has a double circle, so the string is accepted - it can be marked as a legitimate looking email address.

You could break up the process, for example, asking them which state it would be in after “email”, and after “email@gmail”



Slide 16



Here's another possible email address: mygooglegmail. Will the automaton let it through? If not, why not? Share your ideas in the chat.

“mygooglegmail” won't get to the accepting state as it stops at 2, which doesn't have a double circle.

Where else do we see inputs having to be in the right format for the intended output to happen? Think about when you have to enter information in a certain way to match the format required.

Brainstorm with participants applications for using this process to recognise inputs. You may have to share one of these examples to get the ball rolling to start.

Possible answers:

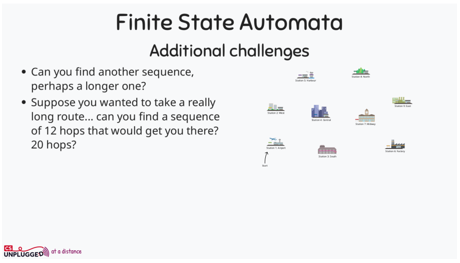
- Dates that should be entered in a particular format such as 9-Mar-2011 (for example, 123-Marc-2011 shouldn't be accepted)
- Dollar amounts typed into a banking app, such as $12.23 (for example, $12.23.45 shouldn't be accepted even though it uses legitimate characters, and $12 should be accepted even though it has no decimal point).
- If you've used a text-based programming language, it will be full of rules that use a FSA to check them (like a variable name must start with a letter, and only contain letters and digits).
- FSAs can be used to understand and evaluate interfaces on gadgets like alarms, remote controls, microwave ovens, buttons on headphones, heating or cruise controls on a car, digital clocks and watches, and even vending machines (and sometimes the instructions for these gadgets look like a simple FSA, with boxes showing “states” (such as “defrost mode”) and arrows labelled with button names showing how to get between them).
- Predictive text: if 'th' has been typed the computer is unlikely to suggest 'j' as the next letter, but will

probably suggest 'e'. This can be modelled by a type of FSA.

Text-based programming languages use finite state automata to check code is written correctly in order for the program to read it and run as expected. Have you ever received the message 'syntax error' for a program? If you have, a finite state automata has probably been used to check your code for errors.

This is a simplified FSA for email addresses. An FSA to test all valid email addresses is more complicated.

Slide 17


Finite State Automata

Additional challenges

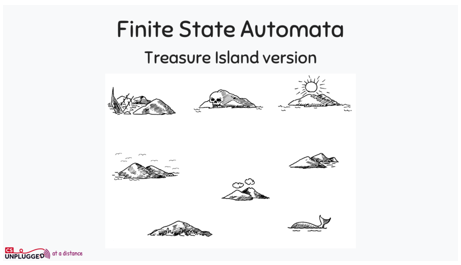
- Can you find another sequence, perhaps a longer one?
- Suppose you wanted to take a really long route... can you find a sequence of 12 hops that would get you there? 20 hops?

Going back to our original train station activity, there's a couple of fun ways you can extend it with your students. Challenge them to find a different string of As and Bs that will get them from Airport Station to Harbour Station. [Hint: to make the string longer they just need to find a loop, such as BA, which takes them from Airport Station, to South Station, and back again, and use this a few times.]

You can also ask them how many different successful routes (strings) they think there are for this map? (There is an infinite number of accepted strings! And there is also an infinite number of strings that won't be accepted!)

Slide 18

We've been using an online version of the game based on trains, but if you can do this activity in person, there's a version where the students run around between the locations, which can be even more engaging! Instead of train stations it uses islands navigated by a friendly pirate commuting service, and has people stationed around the playground (or room) holding the information about where route A and B go from each island. The general principles and goal of the activity are the same. There's more information about this on the CS Unplugged site, we will share a link on the next slide.


Finite State Automata

Treasure Island version

Slide 19

Here are some supporting resources for you. I'll paste these links in the chat.

```
Classic CS Unplugged - Treasure Hunt activity
https://classic.csunplugged.org/activities/finite-
state-automata/

Classic CS Unplugged - Video on FSA
https://classic.csunplugged.org/videos/#finite-state-
automata

Supporting high school learning - Computer Science
Field Guide
https://csfieldguide.org.nz/en/chapters/formal-
languages/finite-state-automata/

Video supporting high school learning - Regular
Languages (especially up to 4:13)
https://www.youtube.com/watch?v=PK3wL7DXuuw
```

Supporting Resources

- Classic CS Unplugged - Treasure Hunt activity
  https://classic.csunplugged.org/activities/finite-state-automata/
- Classic CS Unplugged - Video on FSA
  https://classic.csunplugged.org/videos/#finite-state-automata
- Supporting high school learning - Computer Science Field Guide
  https://csfieldguide.org.nz/en/chapters/formal-languages/finite-state-automata/
- Video supporting high school learning - Regular Languages (especially up to 4:13)
  https://www.youtube.com/watch?v=PK3wL7DXuuw