



Inf2A: Course Roadmap

Bonnie Webber
Stuart Anderson

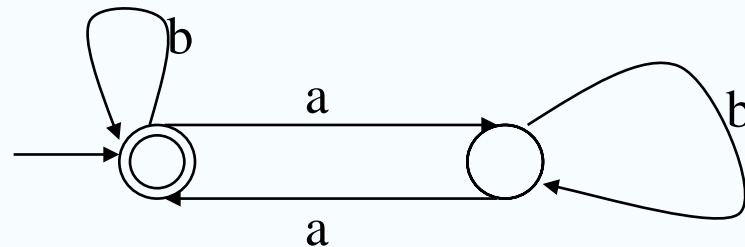
Please Read
J&M Chapter 1, Kozen Chapters 1&2

**informatics**

High Level Summary

- This course is foundational - it tries to capture the elements of the computational approach to understanding phenomena (with special reference to natural language, artificial languages and the behaviour of simple systems).
- One underlying notion is the *transition system* where we describe a collection of states or configurations and provide a relation between states (the transition relation) that describes how the system may change state or configuration.
- We look at different classes of systems potentially with structure in the state and/or in labels on the transitions.
- The notion of *Language* is a means of capturing all possible sequences in a transitions system and we can see a transitions system as a means of generating a language.
- This course is *practical* - you will consider a range of design issues and how to use your knowledge of transition systems to help design and analyze computational systems.

- What is the language recognised by this FSM?
1. Any sequence of as and bs with an even number of as
 2. Any sequence of as and bs
 3. The empty language
 4. A sequence of bs of any length



Overview

- A language is a *set* (usually infinite) of finite *sequences of symbols* (e.g. like letters or simple sounds), a particular sequence in a language is called a *sentence* of the language.
- To specify a language we specify the *alphabet* of symbols and then provide some *specification* of those sequences that are in the language.
- Specifications of languages are often given by either:
 - a *grammar* (see later) that gives a means of generating all possible sentences of a language, or
 - an *acceptor* (recall finite state acceptors from Inf1A) which is a mechanism for deciding if a given sentence is in the language. Sometimes there are outputs - *transducers*.

Overview - Continued

- We study different classes of grammar and acceptor
 - we are interested in the class of languages that can be described by a particular grammar or acceptor and the relationships between them.
- Here we will study four classes of grammar and four classes of acceptor (plus variants). These are:
 - Regular grammars, context-free grammars, context-sensitive grammars, and unrestricted grammars
 - Finite-state automata, pushdown automata, linear-bounded automata, and Turing Machines.
- To some extent these developed independently but have strong relationships.

Languages as Models

- One major component of any Natural Language are the sequences (ie, strings) of words that make up utterances or sentences in the language so our definition of languages fits well.
- We'll see that the transitions in grammars correspond to the idea of *generating* sentences using rules.
- From a machine point of view we often want to see symbols as modelling actions and sentences in the language as sequence of actions.
- From a machine point of view the language of a machine is all possible behaviours (sequences of actions).
- Sometimes this notion works fine (most of the time in Inf2A) but sometimes it is inadequate.

Probability



- If a language is all possible behaviors, in a particular context, some behaviors will be more likely than others.
- Probabilistic models approximate a collection of data (e.g. remarks on different topics).
- We can use these models to detect those of particular interest within a stream of such remarks.

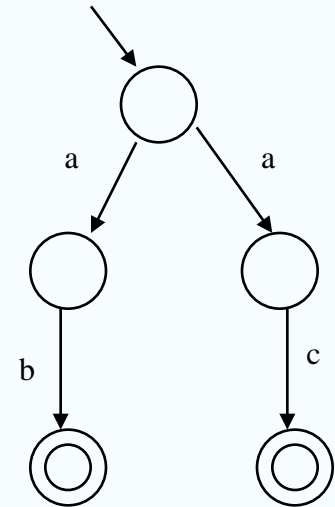
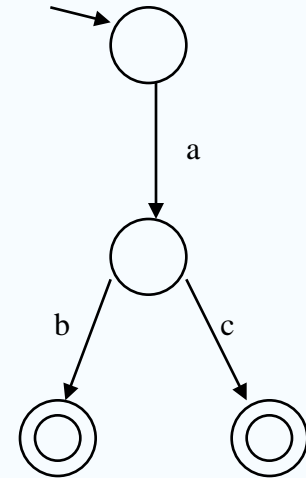
A screenshot of the BBC News website. The top left features the 'BBC NEWS' logo. To the right, there is a link for 'OPEN BBC News in video and audio'. Below the logo is a navigation menu with categories like 'News Front Page', 'Africa', 'Americas', 'Asia-Pacific', 'Europe', 'Middle East', 'South Asia', 'UK', 'Business', 'Health', 'Science/Nature', and 'Technology'. The main content area shows a headline: 'Bush defends phone-tapping policy'. Below the headline is a sub-headline: 'US President George W Bush has again defended his decision to allow eavesdropping on Americans in the wake of the 2001 terror attacks.' To the right of the text is a photograph of George W. Bush. Below the photo is a 'VIDEO Watch the statement' link. At the top right of the news section, there is a 'Last Updated: Monday, 19 December 2005, 17:20 GMT' timestamp and links for 'E-mail this to a friend' and 'Printable version'.

Kinds of things we are concerned with (increasingly meta)

- The definitions of state/configuration and transition and computation.
- The design and operation of particular systems - e.g. designing a traffic light controller we want to ensure two opposing signals never both display green.
- Issues about collections of machines - e.g. are two machines "equal"?
- Issues about all machines of a particular class - e.g. "there is no machine of kind X that does this..."
- Issues across classes of machines - e.g. "for any machine of a particular class there is another that *does the same thing*"

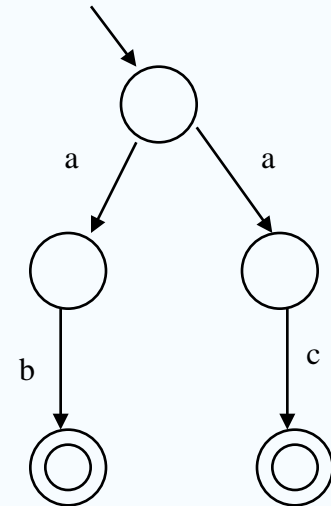
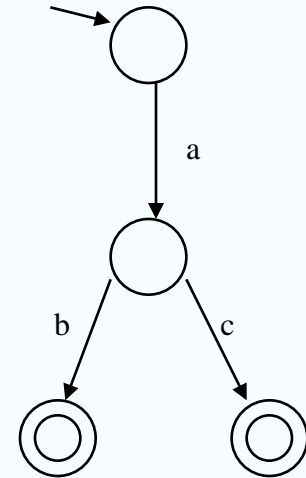
Equality

- Are these two FSMs equivalent?
- True, or
- False?



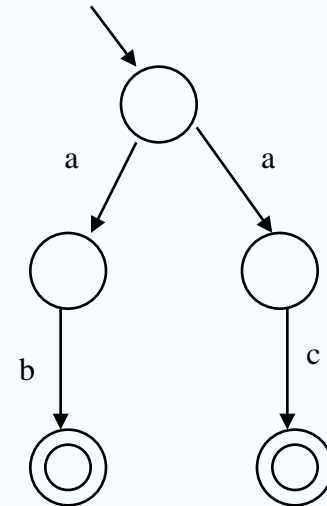
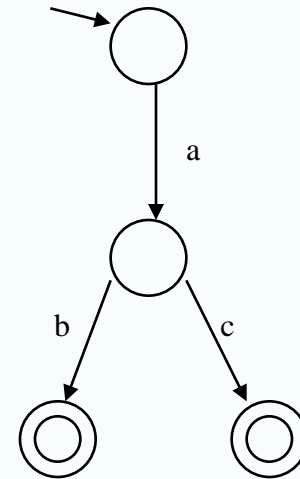
Equality

- Are these two FSMs equivalent?
- Why?
- Because:
 1. They are the same because they recognise the same language.
 2. They are different because after accepting an a either a b or a is acceptable in one but not the other.

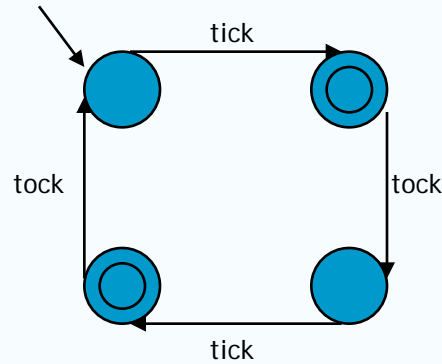
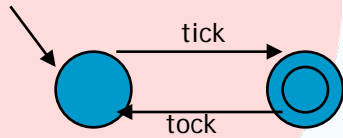


Equality

- Are these two FSMs equivalent?
- Why?
- Because:
 1. They are the same because they recognise the same language.
 2. They are different because after accepting an a either a b or a is acceptable in one but not the other.
- Answer 1 is the standard notion of equality of finite-state machines.

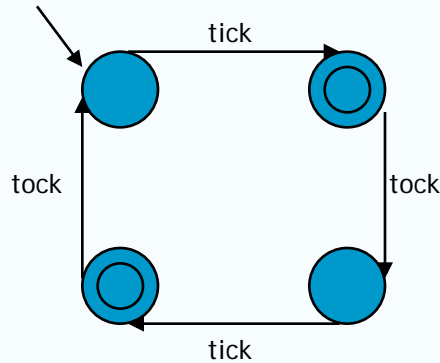
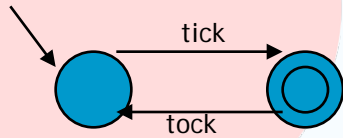


More on Equality

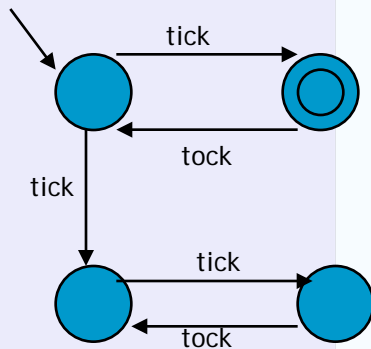


1. Are these two machines equal?
 - True, or
 - False
- How would you convince me?

More on Equality

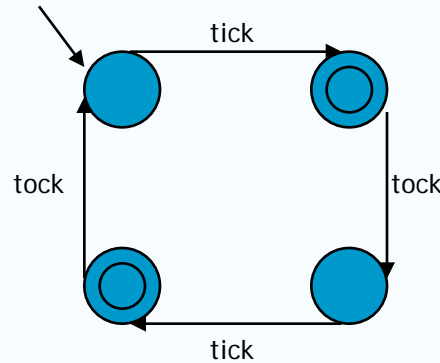
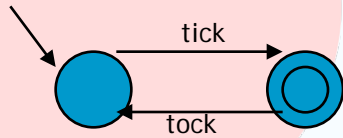


- Are these two machines equal?
 - True, or
 - False
- How would you convince me?

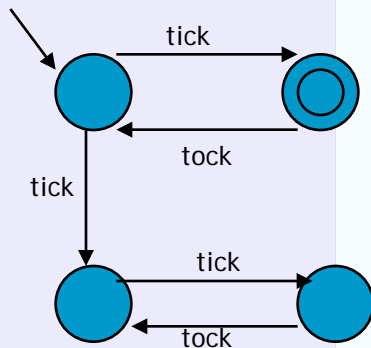


- Is this machine equal to the two-state machine above?
 - True, or
 - False
- How would you convince me?

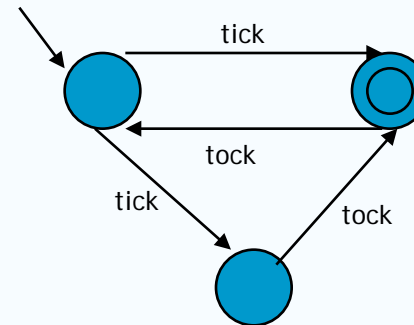
More on Equality



- Are these two machines equal?
 - True, or
 - False
- How would you convince me?



- Is this machine equal to the two-state machine above?
 - True, or
 - False
- How would you convince me?



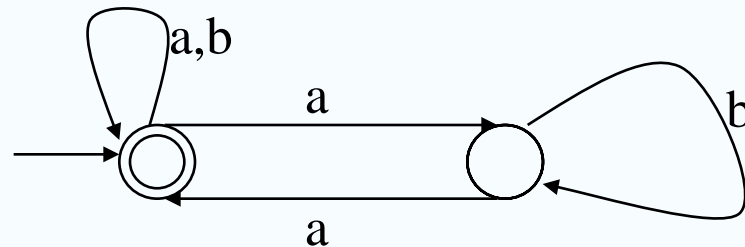
- Is this machine equal to the two-state machine above?
 - True, or
 - False

What do we do with Grammars and Machines?

- We can use grammars and machines to describe particular languages we are interested in. We consider:
 - Using these mechanisms (particularly grammars) to *describe* a naturally occurring language (e.g. English or Hindi):
 - Here we are constructing a model of some mechanism we can empirically observe.
 - We worry about the adequacy of the model, whether the mechanism explains anything about the phenomenon
 - Using these mechanisms to *design* a new artificial language (e.g. a programming language or some interchange format between two computer systems):
 - We worry about properties of the language e.g. how easy is it to *parse*, is it unambiguous, is it easy to detect and recover from errors in a sentence in the language.

Revision

- What regular expression describes the language recognised by this machine?
 - $(a+b)^*$
 - $(a^*b^*)^*$
 - $(b^*ab^*a)^*b^*$
 - $(aba)^*$



What do we do with Grammars and Machines?

- We can explore the definitional power of a particular mechanism (either grammar or machine) and see how it relates to other mechanisms. This is the study of the foundations of computation. Our concerns are questions like:
 - Is a particular mechanism more or less powerful than another?
 - Is it possible to describe any conceivable language in one of these mechanisms?
 - Are there languages that are impossible to describe?
 - For a given language description is it always possible to decide whether a sentence is in the language or not?

Natural Language

- Complex, naturally occurring phenomenon, so our models are always approximate. Areas of study:

- Phonetics and phonology: study of linguistic sounds

[aɪ p^hi: eɪ]

- Morphology: study of the structure of words in.sur.mount.able, sale.s.manager

- Syntax: study of sentence structure

fruit flies like a banana

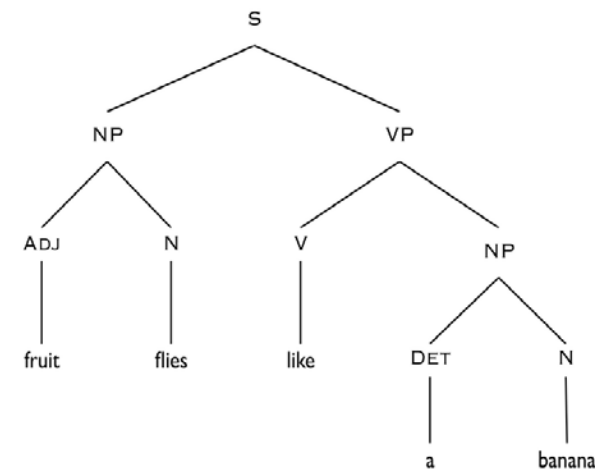
- Semantics: the study of meaning

A student failed every course:

$(\exists x)(\text{student}(x) \wedge (\forall y)(\text{course}(y) \rightarrow \text{failed}(x,y)))$

- Pragmatics and Discourse: study of language use and of larger linguistic units (dialogues, texts)

It's freezing in here → Command: close the window



Designing Artificial Languages

- Here we are in control of the language so we try to design in "good" properties like being easy to check if a sentence is correct, make it easy for the checker to recover from human errors (e.g. omissions, misspelling, ...), make it easy for a human to understand. Typically we study a subset of the areas studied for natural language:
 - Lexical analysis (part of morphology) studies how the symbols of the language are built from the components that make them up (e.g. a name and the letters making up a name).
 - Syntax: the study of the structure of sentences.
 - Semantics: how to relate meaning to sentences (e.g. in a programming language, relating the text of the program to its behaviour (ideally in a way that is independent of a particular implementation).

Contrast: The attitude to ambiguity

- Natural utterances are full of **ambiguity**. The less context, the harder to decide what was intended:
 - [J&M] "*I made her duck*"
 - All meanings are valid in this context (and each has a different structure).
 - We don't want to throw away any possibilities until we know more.
- In the design of programming languages, ambiguity is not often tolerated :
 - $y = 1; \text{ if } x > 3 \text{ then if } x < 5 \text{ then } y = 2 \text{ else } y = 3$
 - If $x == 2$ then what is the value of y after executing this?
 - Solution: either don't allow it or always ensure the syntax is unambiguous.

Natural Language Ambiguity

- Part of speech ambiguity in the BNC (Inf 1B):
 - I: PNP CRD ZZO NPO (personal pronoun, cardinal, symbol, proper noun)
 - Made: VVN VVD (verb in past tense or part participle)
 - Her: DPS PNP (personal pronoun, possessive pronoun)
 - Duck: NNO VVI VVB NPO (common noun, verb in infinitive, verb in base form, proper noun)
- Syntactic ambiguity:
 - [I [made [her duck]]]
 - [I [made her duck]]
 - [[I [made her]] Duck]

Computation

- Here we don't worry about individual languages
- We are concerned with the collection of all languages that are describable using a particular method (e.g. Context-Free Grammars or Finite State Machines).
- We might ask questions like:
 - Is every language I can describe using this method describable by some other method?
 - What languages are not describable using some particular method?
 - It's clear that not all languages are describable - is it?
 - Is there a general method of deciding, given a description of a language and a string, whether the string is in the language?
 - Can we construct efficient, general-purpose, *parsers*.

Abstraction

- Natural languages are quite complex and difficult to analyse.
- Real-world computers are also quite complex, and difficult to analyse cleanly.
- In studying language and computation, we always *abstract* in order to study a problem in as simple a situation as we can while retaining the essence of the real-world issue.
- An abstraction we will study is *effective computability*, i.e. that we can in principle calculate the solution to some problem.
- In the early to mid-1900s this was studied intensively using different models - however all turned out to be equivalent.
- The stability of this abstraction in all of the models is enshrined in the "*Church-Turing*" thesis, which claims that all definitions of computational mechanisms are equivalent.

Summary

- Formal languages are an important tool in the study of natural language and computer languages and systems.
- The basic definitions are common across both fields but the phenomena we study are different.
- Algorithms and techniques to process languages and automata are also common but they are specialised to the application area in quite different ways.
- We have a range of different kinds of mechanisms for defining languages that vary in expressiveness - in general more expressive means less amenable to the use of automated tools.

Question for Next Time

- Is there a finite state machine that recognises all those strings from the alphabet $\{a,b\}$ where the difference between the number of a's and number of b's is less than k for some constant k ???