

Models of Computation

Revision notes

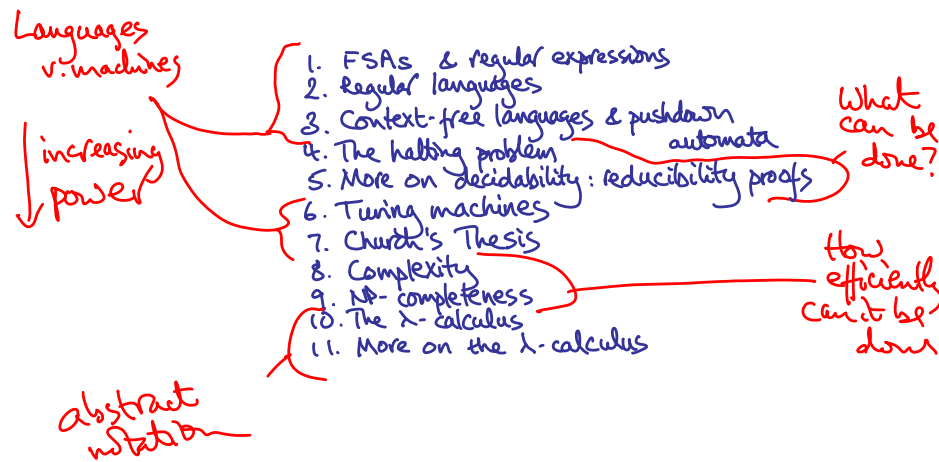
Steve Vickers

Outline

1. FSAs & regular expressions
 2. Regular languages
 3. Context-free languages & pushdown automata
 4. The halting problem
 5. More on decidability: reducibility proofs
 6. Turing machines
 7. Church's Thesis
 8. Complexity
 9. NP-completeness
 10. The λ -calculus
 11. More on the λ -calculus
- Pumping Lemma* (next to 3)
- Rice's Theorem* (next to 6)

Outlook of the module

- Practical - applying techniques
 - Should make you think - abstract
 - You see some proofs
- exam is not a memory test for proofs
- answers to "Why does it work?"



FSA's & regular expressions

- 1. FSAs & regular expressions
- 2. Regular languages
- 3. Context-free languages & pushdown automata
- 4. The halting problem
- 5. More on decidability: reducibility proofs
- 6. Turing machines
- 7. Church's Thesis
- 8. Complexity
- 9. NP-completeness
- 10. The λ -calculus
- 11. More on the λ -calculus

- ① Regular expression $E \rightarrow$ FSA that recognizes strings matching E
- \downarrow
non-deterministic FSA
- ② FSA — the strings it recognizes form a regular language

Regular languages

- 1. FSAs & regular expressions
- 2. Regular languages
- 3. Context-free languages & pushdown automata
- 4. The halting problem
- 5. More on decidability: reducibility proofs
- 6. Turing machines
- 7. Church's Thesis
- 8. Complexity
- 9. NP-completeness
- 10. The λ -calculus
- 11. More on the λ -calculus

Not every language is regular

* Pumping lemma — use to show a given language is not regular

Proof strategy: show language does not have a pumping number

Show for any number N , it can't be a pumping number for that language

If N is a pumping number:

- \forall strings $w \in$ language
 - \forall substrings of w with length $\geq N$
 - \exists subsubstring that is "pumpable" — \forall natural numbers n
- If you change w by pumping the subsubstring n times, the result is still in the language

Task: show N is not a pumping number

If N is a pumping number:

- \forall strings $w \in$ language
- \forall substrings of w with length $\geq N$
- \exists subsubstring that is "pumpable" — \forall natural numbers n

If you change w by pumping the subsubstring n times, the result is still in the language

- \exists string $w \in$ language
- \exists substring with length $\geq N$
- \forall subsubstrings — not pumpable
- $\exists n$ pumping the subsubstring n times takes w out of the language

\exists - you choose
 \forall - someone else chooses

\exists string $w \in$ language
 \exists substring with length $\geq N$
 \forall substrings - not pumpable
 $\exists n$
 pumping the substring n times takes w out of the language

You choose

string $w \sim$ in language
 substring of length N

You don't choose substring
 \therefore make sure that pumping any substrings takes w out of language.
The clever part

Context-free languages

& pushdown automata

- Context-free languages
 - derivation trees words

-
- FSAs & regular expressions
 - Regular languages
 - Context-free languages & pushdown automata
 - The halting problem
 - More on decidability: reducibility proofs
 - Turing machines
 - Church's Thesis
 - Complexity
 - NP-completeness
 - The λ -calculus
 - More on the λ -calculus

- Context-free languages \leftrightarrow pushdown automata

non-deterministic

Induction for context-free grammars

Used to show all derived strings have some property
can be used to show a string cannot be derived

- Find related properties for all non-terminals
- Show in each rule that if substrings on right hand side have appropriate properties then so does string built up from them

e.g. $S ::= U0 | T1$
 $T ::= 0 | S0 | TT1$
 $U ::= 1 | S1 | UU0$

S is start symbol

Property for S: has equal numbers of 0, 1

----- T: one more 0

----- U: one more 1

e.g. $S ::= U0$ the U-string has one more 1
 \therefore the string derived from $U0$ has equal numbers

e.g. $T ::= TT1$ each T-string on right has one more 0
 \therefore string from TT has two more 0s
 \therefore string from $TT1$ has one more 0

Halting problem

• Don't have to memorize proof

• The result : HP is undecidable

-
1. FSAs & regular expressions
 2. Regular languages
 3. Context-free languages & pushdown automata
 4. The halting problem
 5. More on decidability: reducibility proofs
 6. Turing machines
 7. Church's Thesis
 8. Complexity
 9. NP-completeness
 10. The λ -calculus
 11. More on the λ -calculus

Reducibility

If problem A were decidable
then HP would be too

\therefore A is undecidable

Saves having to reprove difficult proof for A

e.g. properties of I/O behaviour (Rice's theorem)

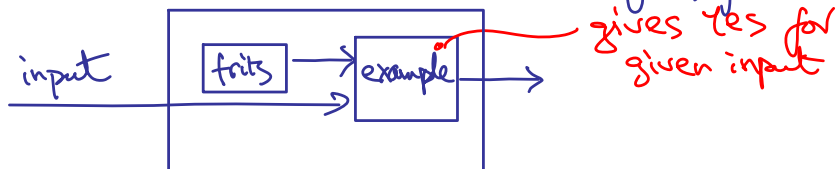
-
1. FSAs & regular expressions
 2. Regular languages
 3. Context-free languages & pushdown automata
 4. The halting problem
 5. More on decidability: reducibility proofs
 6. Turing machines
 7. Church's Thesis
 8. Complexity
 9. NP-completeness
 10. The λ -calculus
 11. More on the λ -calculus

Rice's theorem

Decidability questions about input/output behaviour

Suppose you have a decision procedure for some question like this

Reduce HP to it - does fritz.java halt?



$P = ? = NP$
solvable in polynomial time on deterministic Turing m/c

checkable in poly time on deterministic TM
 \Updownarrow
solvable in polynomial time on non-deterministic Turing m/c

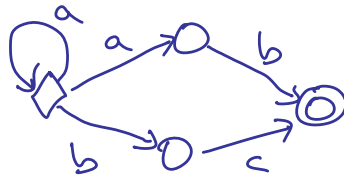
-
1. FSAs & regular expressions
 2. Regular languages
 3. Context-free languages & pushdown automata
 4. The halting problem
 5. More on decidability: reducibility proofs
 6. Turing machines
 7. Church's Thesis
 8. Complexity
 9. NP-completeness
 10. The λ -calculus
 11. More on the λ -calculus

Consider the regular pattern "a* (ab|bc)" over the alphabet

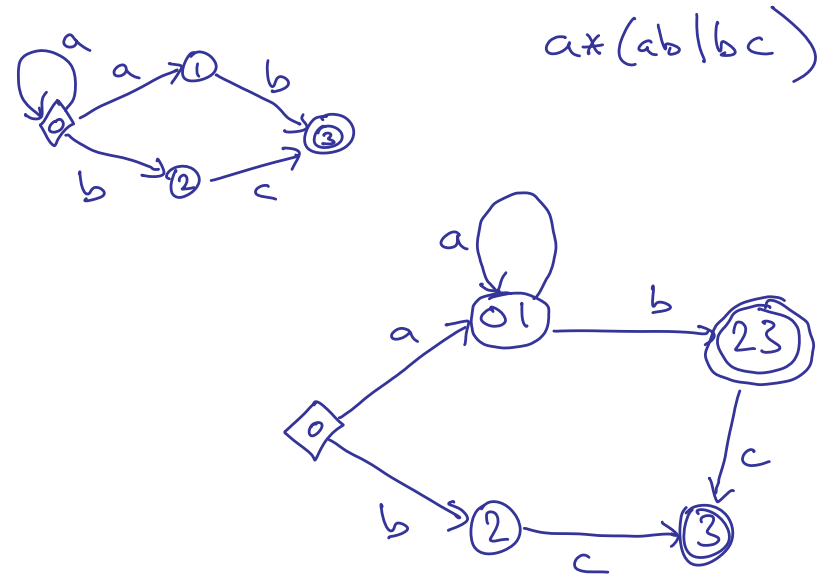
$\Sigma = \{a, b, c\}$.

Design a *deterministic* finite automaton that will accept exactly those strings over Σ that match the pattern.

Non-deterministic FSA:



Usually easier to do non-deterministic first, even though question doesn't ask for it.

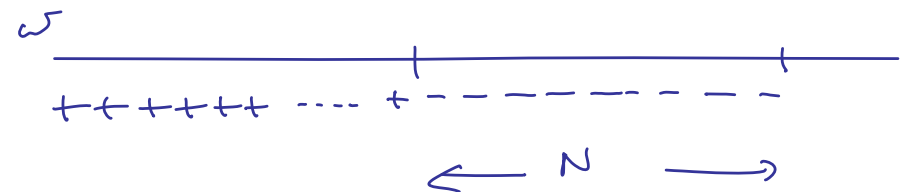


Consider strings in $\Sigma = \{+, -\}$ where + and - represent deposits and withdrawals (in units of £10) for a bank account. The bank never lets you go overdrawn, so "+++-" is acceptable but "+- -++" is not: because you would go overdrawn after the second -. Use the Pumping Lemma to show that the language of acceptable strings is not regular.

Suppose N is a pumping number. To show this is impossible, want

- \exists string $w \in$ language
- \exists substring with length $\geq N$
- \forall substrings - not pumpable
- $\exists n$ pumping the subsubstring n times takes w out of the language

Key insight in this example substring must be all - (Pumping + never takes string out of language.)



Suppose N is a pumping number. Let

$$w = +^N -^N$$

(all -)

Consider the substring $-^N$. Pumping any part of it gives an overdraft \leftarrow so takes w out of the language

$\therefore N$ is not a pumping number \therefore language is not regular

Consider the grammar given, over an alphabet $\Sigma = \{a, b\}$, by

$S ::= \epsilon \mid TS$

$T ::= \epsilon \mid aTb$

Explain why any string in the language must have an equal number of a's and b's.

We also show at the same time that any string derived from T has equal numbers of a's & b's.
Must check four rules: $S ::= \epsilon$ $T ::= \epsilon$ $S ::= TS$ $T ::= aTb$
The T, S strings on right have equal numbers, \therefore so does their concatenation.

Consider the the following problem. You are given a Java program, the file input it is to run on, and the name of an exception class. The problem is to determine whether the program will throw an uncaught exception of that class. By reducing the halting problem to it, show that this problem is undecidable.

If we had an exception decider, we could use it to decide the HP as follows.
Let fritz be a program, with some input given. We define a new exception class MyException, not used anywhere else.
Modify fritz so that at all its returns from main it throws MyException.
Then exception decider applied to modified fritz says whether original fritz would have terminated.