COMP2022: Formal Languages and Logic

2018, Semester 2, Week 9

Kalina Yacef, Joseph Godbehere

4th October, 2018

THE UNIVERSITY OF SYDNEY

Outline

- **Revision**

- Turing Machines

- Logic

FIRST and FOLLOW sets

In order to fill in the entries of the table-drived parser, we need to compute some FIRST and FOLLOW sets.

$FIRST(\alpha)$ is the set of all terminals which could start strings derived from α. We will need to calculate these for every production of G (i.e. the *right hand side* of each rule).

$FOLLOW(V)$ is the set of all terminals which which could follow the variable V at any stage of the derivation. Needed whenever V can derive ε.

Why do we need them?

Let the current input symbol be b, and the top of the stack be X.

There are two ways we might try to derive a string starting with b:

1. Derive a string starting with b from the variable X. Look for a rule $X \to \alpha$ where $b \in FIRST(\alpha)$

2. If X can be derived to ε, then we might be able to derive a string starting with b by using the symbol(s) *following* X on the stack.
   i.e. If any of the production rules $X \to \alpha$ had $\varepsilon \in FIRST(\alpha)$, then we also look at $FOLLOW(X)$

Another way to calculate FIRST sets

Let $A \to X_1 ... X_n$ be a production of A, where X_i could be a terminal or variable.

We recursively compute $FIRST(X_1...X_n)$ by looking at X_1:

- If X_1 is a terminal symbol, then $FIRST(X_1...X_n) = \{X_1\}$
- If X_1 is a variable, then $FIRST(X_1...X_n)$ contains $FIRST(X_1) \setminus \{\varepsilon\}$
- If X_1 is a variable $\varepsilon \in FIRST(X_1)$ then $FIRST(X_1...X_n)$ also contains $FIRST(X_2...X_n)$

Don't forget that $FIRST(\varepsilon) = \{\varepsilon\}$, so if every X_i can generate ε, then rule 3 will (eventually) give us $\varepsilon \in FIRST(X_1...X_n)$

Another way to calculate FOLLOW sets

If $\varepsilon \in FIRST(\alpha)$ for some production rule $A \to \alpha$ then we need to compute $FOLLOW(A)$

Consider each production rule where A appears in the *right hand side*.

Let $V \to Y_1 ... Y_n A Z_1 ... Z_m$ (Y_i, Z_i can be terminals or variables)

- If A is the start symbol, then $\$ \in FOLLOW(A)$
- $FIRST(Z_1 ... Z_m) \setminus \{\varepsilon\} \subseteq FOLLOW(A)$
- If $\varepsilon \in FIRST(Z_1 ... Z_m)$ then $FOLLOW(V) \subseteq FOLLOW(A)$

Examples

(see last week)

Constructing the parse table

Rows: one for each variable of the grammar

Columns: one for each terminal of the grammar, and for the end of string marker \$

Steps to fill the table T:

1. If there is a rule $R \rightarrow \alpha$ with $b \in FIRST(\alpha)$ then put α in $T[R, b]$

2. If there is a rule $R \rightarrow \alpha$ with $\varepsilon \in FIRST(\alpha)$ and $b \in FOLLOW(R)$, then put α in $T[R, b]$

EXAMPLE

(see last week)

Left factoring: definition

If a string w appears on the left of several rules for a variable A:

$$A \to wX_1 \mid \dots \mid wX_1$$

Then we can factor out w and introduce a new variable A':

$$A \to wA'$$

$$A' \to X_1 \mid \dots \mid X_n$$

Any other rules produced by A are unaffected.

Eliminating left recursion

Let α, β be arbitrary strings of terminals and/or variables.
Let A be a variable, and R a new variable

If A has left recursive rules:

$$A \to A\alpha \mid \beta$$

It can be replaced with:

$$A \to \beta R$$

$$R \to \alpha R \mid \varepsilon$$

Outline

- Revision

- **Turing Machines**

- Logic

Beyond CFL

An unrestricted grammar to generate $L = \{a^{2^i} \mid i \geq 1\}$:

$$
\begin{aligned}
S &\to ACaB \\
Ca &\to aaC \\
CB &\to DB \\
CB &\to E \\
aD &\to Da \\
AD &\to AC \\
aE &\to Ea \\
AE &\to \varepsilon
\end{aligned}
$$

Beyond CFL

An unrestricted grammar to generate $L = \{a^{2^i} \mid i \geq 1\}$:

How can we derive $aaaa$?

$$S \to ACaB$$

$$Ca \to aaC$$

$$CB \to DB$$

$$CB \to E$$

$$aD \to Da$$

$$AD \to AC$$

$$aE \to Ea$$

$$AE \to \varepsilon$$

Beyond CFL

An unrestricted grammar to generate $L = \{a^{2^i} \mid i \geq 1\}$:

How can we derive $aaaa$?

$$
\begin{aligned}
S &\to ACaB \\
Ca &\to aaC \\
CB &\to DB \\
CB &\to E \\
aD &\to Da \\
AD &\to AC \\
aE &\to Ea \\
AE &\to \varepsilon
\end{aligned}
$$

$$S \Rightarrow ACaB$$

Beyond CFL

An unrestricted grammar to generate $L = \{a^{2^i} \mid i \geq 1\}$:

How can we derive $aaaa$?

$$
\begin{aligned}
S &\to ACaB \\
Ca &\to aaC \\
CB &\to DB \\
CB &\to E \\
aD &\to Da \\
AD &\to AC \\
aE &\to Ea \\
AE &\to \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow ACaB \\
&\Rightarrow AaaCB
\end{aligned}
$$

Beyond CFL

An unrestricted grammar to generate $L = \{a^{2^i} \mid i \geq 1\}$:

How can we derive $aaaa$?

$$S \to ACaB$$
$$Ca \to aaC$$
$$CB \to DB$$
$$CB \to E$$
$$aD \to Da$$
$$AD \to AC$$
$$aE \to Ea$$
$$AE \to \varepsilon$$

$$S \Rightarrow ACaB$$
$$\Rightarrow AaaCB$$
$$\Rightarrow AaaDB$$

Beyond CFL

An unrestricted grammar to generate $L = \{a^{2^i} \mid i \geq 1\}$:

How can we derive $aaaa$?

$$
\begin{aligned}
S &\to ACaB \\
Ca &\to aaC \\
CB &\to DB \\
CB &\to E \\
aD &\to Da \\
AD &\to AC \\
aE &\to Ea \\
AE &\to \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow ACaB \\
&\Rightarrow AaaCB \\
&\Rightarrow AaaDB \\
&\Rightarrow AaDaB
\end{aligned}
$$

Beyond CFL

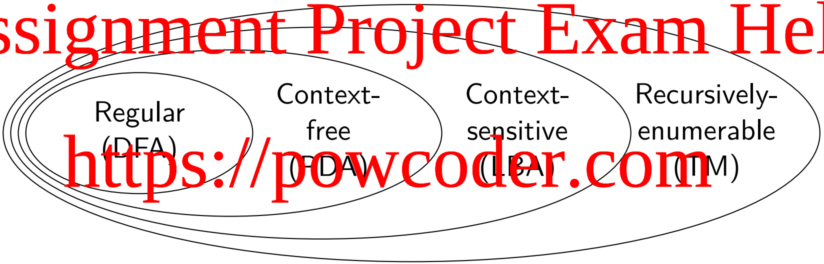
An unrestricted grammar to generate $L = \{a^{2^i} \mid i \geq 1\}$:

How can we derive $aaaa$?

$$
\begin{aligned}
S &\to ACaB \\
Ca &\to aaC \\
CB &\to DB \\
CB &\to E \\
aD &\to Da \\
AD &\to AC \\
aE &\to Ea \\
AE &\to \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow ACaB \\
&\Rightarrow AaaCB \\
&\Rightarrow AaaDB \\
&\Rightarrow AaDaB \\
&\Rightarrow ADaaB \Rightarrow ACaaB \\
&\Rightarrow AaaCaB \Rightarrow AaaaaCB \\
&\Rightarrow AaaaaE \Rightarrow AaaaEa \\
&\Rightarrow AaaEaa \Rightarrow AaEaaa \\
&\Rightarrow AEaaaa \Rightarrow aaaa
\end{aligned}
$$

Chomsky Hierarchy



Type 0 grammars (unrestricted grammars) are powerful enough to describe the set of *recursively enumerable* languages.

Turing Machines

- Finite automata have a finite memory. They recognise regular languages (regular grammars)
- Push Down Automata: Add a LIFO stack (infinite memory, but access is limited to the top). PDAs recognise context-free languages (context-free grammars)

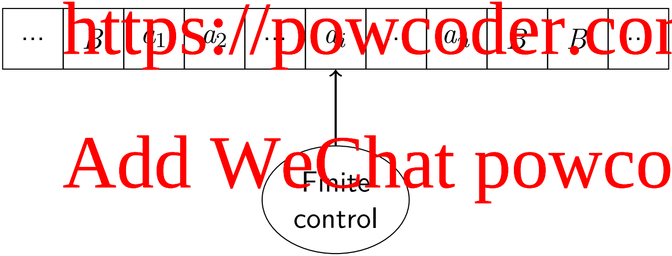
- By changing the stack for a tape (infinite memory, no limitation of access) we get Turing machines: our current computers. TM recognise recursively enumerable languages (unrestricted grammars)

- Alan Turing (1912-1954) mathematician and logician
  - http://www.alanturing.net
  - Turing machines introduced in his 1936 article

Turing Machines: schematic view

Turing machines have infinite memory: a tape.

- Can read and write symbols on it, or do nothing
- Can move to the right or to the left along the tape



i.e. it is a finite state automaton attached to an infinite tape

Formal Definition

A Turing machine is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

- Q is a finite set of states
- Σ is a finite set called the input alphabet. $B \notin \Sigma$
- Γ is a finite set of tape symbols. $\Sigma \subset \Gamma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.
  - L means 'move left', R means 'move right'
- $q_0 \in Q$ is the start state
- B is a special symbol of Γ, the *blank*
- $F \subseteq Q$ is the set of accept states

Turing machine for $\{0^n1^n \mid n > 0\}$

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, X, Y, B\}$
- $q_0 \in Q$ is the start state
- B
- $F = \{q_4\}$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is given by:

	0	1	X	Y	B
q_0	(q_1, X, R)			(q_3, Y, R)	
q_1	$(q_1, 0, R)$	(q_2, Y, L)		(q_1, Y, R)	
q_2	$(q_2, 0, L)$		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	(q_4, B, R)
q_4					

Trying M on 0011

Semi-formal description of M

Assuming:

- The tape initially contains only 0s, 1s and blanks
- We start at the first non-blank symbol on the tape

1. If we did not start on a 0, reject
2. Replace the 0 with an X
3. Move right until a 1 occurs
4. Replace the 1 with a Y
5. Move left until we find the X marker
6. Move right once. If we are on a 0, goto (2)
7. Move right across the tape, skipping X and Y.
   - If a 1 or 0 is scanned, reject
   - If a blank is scanned, accept

Turing machine for a non-CFL: $\{a^n b^n c^n \mid n > 0\}$

- $Q = \{q_0, q_a, q_b, q_c, q_y, q_z, q_f\}$
- $\Sigma = \{a, b, c\}$
- $\Gamma = \{a, b, c, X, Y, Z, B\}$
- $q_0 \in Q$ is the start state
- B
- $F = \{q_f\}$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is given by:

	a	b	c	X	Y	Z	B
q_0	(q_a, X, R)				(q_y, Y, R)		
q_a	(q_a, a, R)	(q_b, Y, R)			(q_a, Y, R)		
q_b		(q_b, b, R)	(q_c, Z, L)			(q_b, Z, R)	
q_c	(q_c, a, L)	(q_c, b, L)		(q_0, X, R)	(q_c, Y, L)	(q_c, Z, L)	
q_y					(q_y, Y, R)	(q_z, Z, R)	
q_z						(q_z, Z, R)	(q_f, B, R)
q_f							

Trying it on $aabbcc$

Language accepted by a Turing Machine

- The language accepted by a Turing machine M is the set of strings such that M started on the left of the input and reached some state of F

- M may fail to stop!
- If M stops in some non-accepting state or never stops on input w, then M does not accept w.

- Languages accepted by a Turing Machine are "recursively enumerable"
- Unrestricted grammars (type 0)

Turing machine to add 2 numbers

A number n is represented by n 0's (*unary*)

The two numbers are separated by a 1

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, B\}$
- $q_0 \in Q$ is the start state
- B
- $F = \{q_3\}$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is given by:

	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 0, R)$	(q_3, B, R)
q_1	$(q_1, 0, R)$		(q_2, B, L)
q_2	(q_3, B, R)		
q_3			

Semi-formal description

1. Move to the right until we reach a 1 (skipping 0s)
2. Rewrite the 1 with a 0
3. Move to the right until we reach a blank (skipping 0s)
4. Move to the left once, to rewrite the last 0 with a blank

For example, if our tape originally read ...$B000100B$..., it now reads ...$B00000BB$..., i.e. $3 + 2 = 5$

Turing machines can make calculations! Note the similarity with computers.

Left-bounded tapes

Can a Turing Machine with an infinite tape in *both* directions recognise more languages than tape that is only unbounded in one direction?

Does a tape with positions $0, 1, 2, 3, \ldots$ have more positions than one with positions $\ldots -2, -1, -0, 1, 2, \ldots$? i.e. are there "more" integers than positive integers?

Counter-intuitively, the answer is *no*, because we can pair up every integer with a positive integer:

1	2	3	4	5	$\ldots$
0	-1	1	-2	2	$\ldots$

So, a tape unbounded to the left does not have more positions.

This is what we mean when we say a set is *enumerable*.

Left-bounded tapes

Can a Turing Machine with an infinite tape in *both* directions recognise more languages than tape that is only unbounded in one direction?

We could make a machine that jumped around to seek the correct positions according to the bijection of integers to positive integers, but there's a much simpler way:

Add states to the TM such that whenever we want to move past the left edge of the tape, we simply shift everything on the tape one position to the right instead.

- We will need to place a *marker* at the far right of the input, so we know when to stop shifting.

Multi-tape Turing Machines

Can a multi-tape Turing Machine recognise more languages than a single-tape TM?

Multi-tape Turing Machines

Can a multi-tape Turing Machine recognise more languages than a single-tape TM?

NO

Proof: We can modify a single tape TM to simulate having multiple tapes:

- Add markers symbolising the start and end of each tape
- Add markers symbolising the current TM position on each
- Add states which simulate switching between the tapes
- Add states which extend these (finite) stretches of tape (by shifting everything beyond it by one space)

Context-free languages

Can a TM recognise every context-free language?

Context-free languages

Can a TM recognise every context-free language?

YES

Proof: A TM can simulate a PDA.

- Write the PDA input to the tape
- Mark a section of tape beyond the end of the input as the stack
- Repeatedly move between the tape and the end of the stack
- i.e. each state in the PDA will have two states in the TM
    - One for the current input symbol, one for the top of stack

Context-free languages

Can a TM recognise every context-free language?

YES

Proof: A TM can simulate a PDA.

- Write the PDA input to the tape
- Mark a section of tape beyond the end of the input as the stack
- Repeatedly move between the tape and the end of the stack
- i.e. each state in the PDA will have two states in the TM
  - One for the current input symbol, one for the top of stack

Note: we will need a *non-deterministic* TM (our PDA are non-deterministic)

Non-deterministic TM

Is a Non-deterministic TM more powerful than a deterministic one?

Non-deterministic TM

Is a Non-deterministic TM more powerful than a deterministic one?

NO

Proof: We can simulate a NTM with a 3-tape DTM

- Tape 1 contains the original input
- Tape 2 simulates a particular computation of the TM
- Tape 3 encodes the path through the tree of possible paths through the NTM

Non-deterministic TM

Is a Non-deterministic TM more powerful than a deterministic one?

NO

Proof: We can simulate a NTM with a 3-tape DTM

- Tape 1 contains the original input
- Tape 2 simulates a particular computation of the TM
- Tape 3 encodes the path through the tree of possible paths through the NTM

This DTM will (very slowly!) explore all the possible paths through the NTM, from shortest to longest.

Universal Turing machine

- It is possible to encode a Turing machine or to write a "program" to describe a Turing machine.
- The Universal Turing Machine accepts an encoded Turing machine M together with a string w and simulates M on w.
- Exactly what general computers do:
  - They accept a program P
  - and the input to that program
  - and produces the output of P on the given input
- Again, note the similarity between TM and computers

Church-Turing Thesis

"Every effectively calculable function is a computable function."

Essentially this means that the set of *algorithms* is equivalent to the set of *Turing Machine* algorithms.

So:

- TM are a very precise model for definition of algorithm.
- TM can do *everything* that a computer can do.
- However, even a Turing machine cannot solve all problems:
  - some are beyond theoretical limits of computation

Computability, Decidability, Intractability

Decidable $\subset$ TM computable

- A function that can be computed by a Turing machine is said to be *computable* (therefore the language is *enumerable*)

- It is *decidable* if and only if the Turing machine will always halt on any input string (the Halting Problem)

- *Intractability*: The efficiency problem – can we solve the problem in a reasonable time (e.g. polynomial vs exponential)?

Example of undecidable problem: The Halting problem

- Halting problem: Is there an algorithm that can decide whether the execution of an arbitrary program halts on an arbitrary input?

- e.g.
  - $f(x)\{\ return\ 2x+1;\ \}$, will f halt on any number?
  - $h(x)\{\ while(true)\ \{x+=1;\}\ ..\}$?
  - $g(x)\{\ for(i=x;\ i<10;\ i-=1);\ ...\}$?

- Answer: NO (Church 1936, Turing 1937)

Proving the Halting Problem is undecidable

Suppose the Halting problem is decidable.

Then there exists some universal Turing machine H such that $H(a, b)$ accepts if and only if the TM represented by a would halt on input b.

The language L of H is:

$$\{a, b \mid a \text{ represents a TM, } b \text{ is an input string, } a \text{ halts on } b\}$$

Proving the Halting Problem is undecidable

$H(a, b)$ accepts iff TM a halts on input b

Let $X(c)$ be a Turing machine which either:

- If $H(c, c)$ accepts, then loop forever
- If $H(c, c)$ rejects, then halt

Proving the Halting Problem is undecidable

Now consider what happens if we use "X, X" as input to H, i.e. what does X do when given its *own* representation as input.

Proving the Halting Problem is undecidable

Now consider what happens if we use "X, X" as input to H, i.e. what does X do when given its *own* representation as input.

Case 1: If $H(X, X)$ accepts, then X must halt on input X, because H is a solution to the Halting problem. However, X will loop forever because $H(X, X)$ accepts.

Proving the Halting Problem is undecidable

Now consider what happens if we use "X, X" as input to H, i.e. what does X do when given its *own* representation as input.

Case 1: If $H(X, X)$ accepts, then X must halt on input X, because H is a solution to the Halting problem. However, X will loop forever because $H(X, X)$ accepts.

Case 2: If $H(X, X)$ rejects, then X cannot halt on input X, because H is a solution to the Halting problem. However, X will halt because $H(X, X)$ rejected.

Proving the Halting Problem is undecidable

Now consider what happens if we use "X, X" as input to H, i.e. what does X do when given its *own* representation as input.

Case 1: If $H(X, X)$ accepts, then X must halt on input X, because H is a solution to the Halting problem. However, X will loop forever because $H(X, X)$ accepts.

Case 2: If $H(X, X)$ rejects, then X cannot halt on input X, because H is a solution to the Halting problem. However, X will halt because $H(X, X)$ rejected.

Both cases lead to contradictions, so the assumption was incorrect. i.e. The Halting problem cannot be decidable.

Example of undecidable problem: The Halting problem

- Halting problem: Is there an algorithm that can decide whether the execution of an arbitrary program halts on an arbitrary input?

- e.g.
  - $f(x)\{\ return\ 2x+1;\}$, will f halt on any number?
  - $h(x)\{\ while(true)\ \{x+=1;\}\ ...\}$?
  - $g(x)\{\ for(i=x;\ i<10;\ i+=1);\ ...\}$?

- Answer: NO (Church 1936, Turing 1937)
- How can you restrict the halting problem to be decidable?

Computability, Decidability, Intractability

We will talk more about these concepts at the end of the course.

Outline

- Revision

- Turing Machines

- **Logic**

Logic: Introductory Example

What do you think of the following argument?

1. If a number is a multiple of $2 \times 3 \times 4$, then it is a multiple of 2 and a multiple of 3 and a multiple of 4
2. Therefore, if a number is a multiple of 2 and a multiple of 3 and a multiple of 4, then it is a multiple of $2 \times 3 \times 4$

Logic: Introductory Example

What do you think of the following argument?

1. If a number is a multiple of $2 \times 3 \times 4$, then it is a multiple of 2 and a multiple of 3 and a multiple of 4
2. Therefore, if a number is a multiple of 2 and a multiple of 3 and a multiple of 4, then it is a multiple of $2 \times 3 \times 4$

Expressing this formally:

$$M \to (P \wedge Q \wedge R)$$
$$(P \wedge Q \wedge R) \to M$$

Logic

- Logic is a language used to make some disciplines scientific by providing a way to deduce knowledge from a relatively small number of explicitly stated facts or hypotheses
- In CS: specification of requirements, program verification, some databases
- Allows expression of knowledge concisely and precisely, enabling analysis of the argument structure
- Provides a way to *reason* about the consequences of that knowledge rigourously. i.e. How to make a judgement on the validity of the argument
- Focus on validity (correctness) of the argument *form*, rather than its *contents*

Logic in real arguments

Argument 1: If I play cricket or I go to work, then I will not be going shopping. Therefore, if I go shopping, then I would neither play cricket nor would I go to work

- P: I play cricket
- Q: I go to work
- R: I go shopping

If P or Q, then not R	$(P \vee Q) \rightarrow \neg R$
Therefore, if R then not P and not Q	$R \rightarrow (\neg P \wedge \neg Q)$

Logic in real arguments

Argument 2: An object remaining stationary or moving at a constant velocity means that there is no net external force acting upon it. Therefore, if there is a net force acting upon the object, then it is neither stationary nor is it moving at a constant velocity.

- P: The object is stationary
- Q: The object is moving at a constant velocity
- R: There is a net external force acting upon the object

If P or Q, then not R	$(P \vee Q) \to \neg R$
Therefore, if R then not P and not Q	$R \to (\neg P \wedge \neg Q)$

Arugment, Premises, Deduction

An *argument* is a claim. It is composed of statements

- If P or Q, then not R
- Therefore, if R then not P and not Q

The *premises* of an arugment are the hypothesis for the argument

- If P or Q, then not R

The last statement is the *conclusion* of the argument, which needs to be *deduced* from the premises

- Therefore, if R then not P and not Q

Propositional Logic

Formalise sentences

- Propositions and connectives
- From english to propositions

Semantics

- Truth tables
- Tautologies

Formal Reasoning

Propositions

A *Proposition* is the underlying meaning of a declarative sentence (a sentence which is either **true** or **false**):

- Mammals are warm-blooded
- The sun orbits the earth
- $2 + 2 = 4$
- All integers are even

But these are not propositions:

- Can you show me the way to Redfern?
- Pay your bills on time
- Stop talking!

Well-formed formula (wff) syntax

A well-formed formaula (wff) is an expression with the correct syntax (i.e. it is a string from the language of wff)

Truth symbols are wff (**true** or **false**)

Atomic propositions are wff ($P, Q, R, \ldots$)

Complex propositions are built up using connectives:
If P and Q are wff, then (P), $\neg P$, $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$, $(P \leftrightarrow Q)$ are all also wff

To make it easier to refer to complex wff, we can set labels for them by writing, for example $Z = ((P \rightarrow Q) \vee Q)$

Semantics (truth tables)

Truth tables define the possible values that a wff can take, depending on the values of the atomic propositions that it contains.

The meaning of **true** is 1, and **false** if 0, otherwise the meaning of a wff is its truth table.

P	Q	$P \wedge Q$
1	1	1
1	0	0
0	1	0
0	0	0

Connectives

Negation (not)	$\neg P$ is true iff P is false
Conjunction (and)	$(P \wedge Q)$ is true iff both P and Q are true
Disjunction (or)	$(P \vee Q)$ is true iff P or Q is true
Implication	$(P \rightarrow Q)$ is false iff P is true and Q is false
Equivalence	$(P \leftrightarrow Q)$ is true iff P has the same truth value as Q

P	Q	$\neg P$	$(P \wedge Q)$	$(P \vee Q)$	$(P \rightarrow Q)$	$(P \leftrightarrow Q)$
1	1					
1	0					
0	1					
0	0					

Connectives

Negation (not)	$\neg P$ is true iff P is false
Conjunction (and)	$(P \wedge Q)$ is true iff both P and Q are true
Disjunction (or)	$(P \vee Q)$ is true iff P or Q is true
Implication	$(P \rightarrow Q)$ is false iff P is true and Q is false
Equivalence	$(P \leftrightarrow Q)$ is true iff P has the same truth value as Q

P	Q	$\neg P$	$(P \wedge Q)$	$(P \vee Q)$	$(P \rightarrow Q)$	$(P \leftrightarrow Q)$
1	1	0				
1	0	0				
0	1	1				
0	0	1				

Connectives

Negation (not)	$\neg P$ is true iff P is false
Conjunction (and)	$(P \wedge Q)$ is true iff both P and Q are true
Disjunction (or)	$(P \vee Q)$ is true iff P or Q is true
Implication	$(P \rightarrow Q)$ is false iff P is true and Q is false
Equivalence	$(P \leftrightarrow Q)$ is true iff P has the same truth value as Q

P	Q	$\neg P$	$(P \wedge Q)$	$(P \vee Q)$	$(P \rightarrow Q)$	$(P \leftrightarrow Q)$
1	1	0	1			
1	0	0	0			
0	1	1	0			
0	0	1	0			

Connectives

Negation (not)	$\neg P$ is true iff P is false
Conjunction (and)	$(P \wedge Q)$ is true iff both P and Q are true
Disjunction (or)	$(P \vee Q)$ is true iff P or Q is true
Implication	$(P \rightarrow Q)$ is false iff P is true and Q is false
Equivalence	$(P \leftrightarrow Q)$ is true iff P has the same truth value as Q

P	Q	$\neg P$	$(P \wedge Q)$	$(P \vee Q)$	$(P \rightarrow Q)$	$(P \leftrightarrow Q)$
1	1	0	1	1		
1	0	0	0	1		
0	1	1	0	1		
0	0	1	0	0		

Connectives

Negation (not)	$\neg P$ is true iff P is false
Conjunction (and)	$(P \wedge Q)$ is true iff both P and Q are true
Disjunction (or)	$(P \vee Q)$ is true iff P or Q is true
Implication	$(P \rightarrow Q)$ is false iff P is true and Q is false
Equivalence	$(P \leftrightarrow Q)$ is true iff P has the same truth value as Q

P	Q	$\neg P$	$(P \wedge Q)$	$(P \vee Q)$	$(P \rightarrow Q)$	$(P \leftrightarrow Q)$
1	1	0	1	1	1	
1	0	0	0	1	0	
0	1	1	0	1	1	
0	0	1	0	0	1	

Connectives

Negation (not)	$\neg P$ is true iff P is false
Conjunction (and)	$(P \wedge Q)$ is true iff both P and Q are true
Disjunction (or)	$(P \vee Q)$ is true iff P or Q is true
Implication	$(P \rightarrow Q)$ is false iff P is true and Q is false
Equivalence	$(P \leftrightarrow Q)$ is true iff P has the same truth value as Q

P	Q	$\neg P$	$(P \wedge Q)$	$(P \vee Q)$	$(P \rightarrow Q)$	$(P \leftrightarrow Q)$
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

Example

Construct the truth table for

$$X = (((P \wedge Q) \vee \neg Q) \to ((P \vee Q) \wedge P))$$

P	Q	$(P \wedge Q)$	$\neg Q$	$((P \wedge Q) \vee \neg Q$	$(P \vee Q)$	$(P \vee Q) \wedge P$	X
1	1						
1	0						
0	1						
0	0						

Formalising English as logic

On Friday morning Mary went for a walk, in the afternoon she went to work and on Saturday she stayed home while her house was being painted.

Although John is not tall, John has a better chance of winning the next match of tennis, despite Mark's experience.

If Gromit is not in his kennel, then he is reading the paper.

Increased spending overheats the economy

Increased spending coupled with tax cuts overheats the economy

Formalising English as logic

On Friday morning Mary went for a walk, in the afternoon she went to work and on Saturday she stayed home while her house was being painted.

$$(W \wedge J \wedge H \wedge P)$$

Although John is not tall, John has a better chance of winning the next match of tennis, despite Mark's experience.

If Gromit is not in his kennel, then he is reading the paper.

Increased spending overheats the economy

Increased spending coupled with tax cuts overheats the economy

Formalising English as logic

On Friday morning Mary went for a walk, in the afternoon she went to work and on Saturday she stayed home while her house was being painted.

$$(W \wedge J \wedge H \wedge P)$$

Although John is not tall, John has a better chance of winning the next match of tennis, despite Mark's experience.

$$(\neg T \wedge W \wedge E)$$

If Gromit is not in his kennel, then he is reading the paper.

Increased spending overheats the economy

Increased spending coupled with tax cuts overheats the economy

Formalising English as logic

On Friday morning Mary went for a walk, in the afternoon she went to work and on Saturday she stayed home while her house was being painted.

$$(W \wedge J \wedge H \wedge P)$$

Although John is not tall, John has a better chance of winning the next match of tennis, despite Mark's experience.

$$(\neg T \wedge W \wedge E)$$

If Gromit is not in his kennel, then he is reading the paper.

$$(\neg K \rightarrow P)$$

Increased spending overheats the economy

Increased spending coupled with tax cuts overheats the economy

Formalising English as logic

On Friday morning Mary went for a walk, in the afternoon she went to work and on Saturday she stayed home while her house was being painted.

$$(W \wedge J \wedge H \wedge P)$$

Although John is not tall, John has a better chance of winning the next match of tennis, despite Mark's experience.

$$(\neg T \wedge W \wedge E)$$

If Gromit is not in his kennel, then he is reading the paper.

$$(\neg K \to P)$$

Increased spending overheats the economy

$$(S \to E)$$

Increased spending coupled with tax cuts overheats the economy

Formalising English as logic

On Friday morning Mary went for a walk, in the afternoon she went to work and on Saturday she stayed home while her house was being painted.

$$(W \wedge J \wedge H \wedge P)$$

Although John is not tall, John has a better chance of winning the next match of tennis, despite Mark's experience.

$$(\neg T \wedge W \wedge E)$$

If Gromit is not in his kennel, then he is reading the paper.

$$(\neg K \rightarrow P)$$

Increased spending overheats the economy

$$(S \rightarrow E)$$

Increased spending coupled with tax cuts overheats the economy

$$((S \wedge T) \rightarrow E)$$

Possible interpretations in English

$\neg P$	not P P does not hold it is not the case that P P is false
$(P \wedge Q)$	P and Q P but Q not only P but Q P while Q P despite Q P yet Q P although Q
$(P \vee Q)$	P or Q P or Q or both P and/or Q

Possible interpretations in English

$(P \to Q)$	If P then Q Q if P P only if Q P is sufficient for Q Q is necessary for P P implies Q
$(P \leftrightarrow Q)$	P if and only if Q P iff Q P is necessary and sufficient for Q

Be careful with disjunction!

We need to be careful with the definition of disjunction

- Inclusive "or": $(P \vee Q)$
- Exclusive "or": $(P \vee Q) \wedge \neg(P \wedge Q)$

Be careful with disjunction!

We need to be careful with the definition of disjunction

- Inclusive "or": $(P \vee Q)$
- Exclusive "or": $(P \vee Q) \wedge \neg(P \wedge Q)$

Examples:

You can go to the airport by taxi or bus

You can choose to save your money or your life

The error is in the program or the sensor data

The program or the sensor data are erroneous

Be careful with disjunction!

We need to be careful with the definition of disjunction

- Inclusive "or": $(P \vee Q)$
- Exclusive "or": $(P \vee Q) \wedge \neg(P \wedge Q)$

Examples:

You can go to the airport by taxi or bus	Inclusive
You can choose to save your money or your life	
The error is in the program or the sensor data	
The program or the sensor data are erroneous	

Be careful with disjunction!

We need to be careful with the definition of disjunction

- Inclusive "or": $(P \lor Q)$
- Exclusive "or": $(P \lor Q) \land \neg(P \land Q)$

Examples:

You can go to the airport by taxi or bus	Inclusive
You can choose to save your money or your life	Exclusive
The error is in the program or the sensor data	
The program or the sensor data are erroneous	

Be careful with disjunction!

We need to be careful with the definition of disjunction

- Inclusive "or": $(P \vee Q)$
- Exclusive "or": $(P \vee Q) \wedge \neg(P \wedge Q)$

Examples:

You can go to the airport by taxi or bus	Inclusive
You can choose to save your money or your life	Exclusive
The error is in the program or the sensor data	Exclusive?
The program or the sensor data are erroneous	

Be careful with disjunction!

We need to be careful with the definition of disjunction

- Inclusive "or": $(P \lor Q)$
- Exclusive "or": $(P \lor Q) \land \neg(P \land Q)$

Examples:

You can go to the airport by taxi or bus	Inclusive
You can choose to save your money or your life	Exclusive
The error is in the program or the sensor data	Exclusive?
The program or the sensor data are erroneous	Inclusive?

Be careful with implication/equivalence!

Sometimes in english the syntax and terms used do not reflect the logical meaning

Be careful with implication/equivalence!

Sometimes in english the syntax and terms used do not reflect the logical meaning

"Eating fast-food is equivalent to aiding the destruction of the world's rainforests"

- This looks like an equivalence, but the speaker actually means implication (it's unlikely that they are trying to claim that destroying rainforests implies eating fast food.)

Be careful with implication/equivalence!

Sometimes in english the syntax and terms used do not reflect the logical meaning

"Eating fast-food is equivalent to aiding the destruction of the world's rainforests"

- This looks like an equivalence, but the speaker actually means implication (it's unlikely that they are trying to claim that destroying rainforests implies eating fast food.)

"I will give you a lift to the city, if you are going to the city"

- This is really an equivalence, not the implication that the sentence structure suggests.

More examples

Max is home and Claire is at the library
Max is home or Claire is at the library
Max is home if Claire is at the library
Max is home only if Claire is at the library
Max is home if and only if Claire is at the library
Max is not home nor Claire is at the library
Max is home although Claire is at the library
Max is home unless Claire is at the library

More examples

Max is home and Claire is at the library $(H \wedge L)$

Max is home or Claire is at the library

Max is home if Claire is at the library

Max is home only if Claire is at the library

Max is home if and only if Claire is at the library

Max is not home nor Claire is at the library

Max is home although Claire is at the library

Max is home unless Claire is at the library

More examples

Max is home and Claire is at the library $(H \wedge L)$
Max is home or Claire is at the library $(H \vee L)$
Max is home if Claire is at the library

Max is home only if Claire is at the library
Max is home if and only if Claire is at the library
Max is not home nor Claire is at the library
Max is home although Claire is at the library

Max is home unless Claire is at the library

More examples

Max is home and Claire is at the library	$(H \wedge L)$
Max is home or Claire is at the library	$(H \vee L)$
Max is home if Claire is at the library	$(L \rightarrow H)$
Max is home only if Claire is at the library	
Max is home if and only if Claire is at the library	
Max is not home nor Claire is at the library	
Max is home although Claire is at the library	
Max is home unless Claire is at the library	

More examples

Max is home and Claire is at the library $(H \wedge L)$

Max is home or Claire is at the library $(H \vee L)$

Max is home if Claire is at the library $(L \rightarrow H)$

Max is home only if Claire is at the library $(H \rightarrow L)$

Max is home if and only if Claire is at the library

Max is not home nor Claire is at the library

Max is home although Claire is at the library

Max is home unless Claire is at the library

More examples

Max is home and Claire is at the library	$(H \wedge L)$
Max is home or Claire is at the library	$(H \vee L)$
Max is home if Claire is at the library	$(L \rightarrow H)$
Max is home only if Claire is at the library	$(H \rightarrow L)$
Max is home if and only if Claire is at the library	$(H \leftrightarrow L)$
Max is not home nor Claire is at the library	
Max is home although Claire is at the library	
Max is home unless Claire is at the library	

More examples

Max is home and Claire is at the library	$(H \wedge L)$
Max is home or Claire is at the library	$(H \vee L)$
Max is home if Claire is at the library	$(L \rightarrow H)$
Max is home only if Claire is at the library	$(H \rightarrow L)$
Max is home if and only if Claire is at the library	$(H \leftrightarrow L)$
Max is not home nor Claire is at the library	$(\neg H \wedge \neg L)$
Max is home although Claire is at the library	
Max is home unless Claire is at the library	

More examples

Max is home and Claire is at the library	$(H \wedge L)$
Max is home or Claire is at the library	$(H \vee L)$
Max is home if Claire is at the library	$(L \to H)$
Max is home only if Claire is at the library	$(H \to L)$
Max is home if and only if Claire is at the library	$(H \leftrightarrow L)$
Max is not home nor Claire is at the library	$(\neg H \wedge \neg L)$
Max is home although Claire is at the library	$(H \wedge L)$
Max is home unless Claire is at the library	

More examples

Max is home and Claire is at the library	$(H \wedge L)$
Max is home or Claire is at the library	$(H \vee L)$
Max is home if Claire is at the library	$(L \rightarrow H)$
Max is home only if Claire is at the library	$(H \rightarrow L)$
Max is home if and only if Claire is at the library	$(H \leftrightarrow L)$
Max is not home nor Claire is at the library	$(\neg H \wedge \neg L)$
Max is home although Claire is at the library	$(H \wedge L)$
Max is home unless Claire is at the library	$(L \rightarrow \neg H)$

More examples

Max is home and Claire is at the library	$(H \wedge L)$
Max is home or Claire is at the library	$(H \vee L)$
Max is home if Claire is at the library	$(L \rightarrow H)$
Max is home only if Claire is at the library	$(H \rightarrow L)$
Max is home if and only if Claire is at the library	$(H \leftrightarrow L)$
Max is not home nor Claire is at the library	$(\neg H \wedge \neg L)$
Max is home although Claire is at the library	$(H \wedge L)$
Max is home unless Claire is at the library	$(L \rightarrow \neg H)$
	$(H \vee L)$

Announcements

- Assignment 2
  - Appendices (on extensions) are available
  - Some test cases will be released to you this weekend
  - Submission system will be available next week

- Week 10 quiz tests concepts from the Week 8 tutorial