2020 - Daria Mikhaylova
https://github.com/dmikhaylova/logic-refcard

Classical Logic cheat sheet

Based on Nicholas J.J. Smith. Logic: the laws of truth.

letters and symbols

greek	*model, sets*
Α α alpha	𝔐 latin m (emme) in Fraktur script
Β β beta	⊨ logical sequence
Γ γ gamma	⊏ subformula
Δ δ delta	⊢ follows
Θ θ theta	∪ union
Φ φ phi	∩ intersection
Χ χ chi	\ relative complement
Ψ ψ psi	∈ element of set

bivalence

Each proposition is either **true** or **false** and not both and nothing else.

propositional logic

Syntax:
A,B,...P, Q, R - **basic** propositions — P: Mary works at ISTT lab.
( ) - parentesis — Q: Mary is a scientist.
α, β, γ - **w**ell **f**ormed **f**ormulas
wff are A,B,...,P,Q.. + connectives or wff + connectives
P→Q: Mary works at ISTI lab, so she is a scientist.

predicate logic

Syntax:

x,y,z,u,v,w	variables	
a,b,c,...t	names	(m: Mary, l: lab)
$A^1, B^1, C^1 ... A^2, B^2, C^2$	predicates	(Wxy: x works at y)

number of argument places

where P^n is predicate and $t_1 .. t_n$ is name or variable, $P^n t_1 .. t_n$ is wff

Wml - Mary works at a lab

where α e β are wff, x is variable, α e β + connectives or $\exists x\alpha$ $\forall x\alpha$ are wff

∀x(Wxl → Sx): Everyone who works at a lab is a scientist

connectives

unary - requires one wff

¬ not	negation	¬α **not** alpha (negand)	~α −α NOT α

binary - requires two wwf

∧ and	conjunction	α∧β alpha **and** beta (conjunct)	α & β α·β α β α AND β
∨ or	disjunction	α∨β alpha **or** beta	α OR β
→ if... then	conditional	α→β **if** alpha **than** beta	α ⊃ β α ⇒ β (antecedent, consequent)
↔ if and only if	biconditional	α↔β **if and only if** alpha **than** beta	α ≡ β α ⇔ β
∃x exist	existential quantifier	∃xGx **exist** at least one x such as x is G	!Only for predicate logic
∀x all	universal quantifier	∀xGx **all** x are G	

other connectives

⊤ verum	true		
⊥ falsum	false		
| nand	Sheffer stroke	α | β alpha **e non** beta **or opposite**	α ↑ β
↓ nor	Pierce arrow	α↓β **neither** alpha **nor** beta	α NOR β
⊻ xor	exclusive disjunction	α ⊻ β alpha **or** beta, but **not both**	
∗ any operator			

There are also other connectives, 2^n types for n-place connective

main connective

main connective is the one used last in **order of construction** of wff, these are 5 steps for (P∧Q) → (P∨Q):

1) P basic
2) Q basic
3) P∧Q (1,2, ∧)
4) P∨Q (1,2, ∨)
5) (P∧Q) → (P∨Q) (3,4,→)

1) P: It's sunny
2) Q: It's windy
3) P∧Q: It's sunny **and** it's windy
4) P∨Q: It's sunny **or** it's windy
5) (P∧Q) → (P∨Q): **If** it's sunny and it's windy, **than** it's sunny or it's windy

wff: P, Q, (P∧Q), ¬(P∧Q), (P∧Q) → (P∨Q)
wff: Fab, Gxy, ∀x(Fx∧Gxa)
wff: □, □, □ ∧ □, ¬(□∧ □), (□∧ □) → (□∨ □)

construct truth table

1) a truth table has 2^n rows, for n basic propositions α, β, γ
2) put each proposition in its own column.
α β
3) this table will have 2^2 = 4 rows. Start from utmost **right** column and alternate T and F, Then move left and alternate TT and FF and if you have next column TTTT FFFF:

α	β		α	β
	T		T	T
	F		T	F
	T		F	T
	F		F	F

4) write on the right all wff to calculate and fill truth value under connective, following order of construction

α	β	α∧β	→	α∨β
T	T	T	T	T
T	F	F	F	T
F	T	F	T	T
F	F	F	T	F

5) for argument, write all premises and conclusion from left to write and calculate truth values.
α ∧ β ∴ α ∨ β

α	β	α ∨ β	α ∧ β	
T	T	T	T	← ok
T	F	**T**	**F**	← not valid, counterexample
F	T	**T**	**F**	← not valid, counterexample
F	F	F	F	

tables for connectives

α	β	and α∧β	or α∨β	if.. than α→β	iff α↔β	xor α⊻β	nor α↓β	nand α|β
T	T	T	T	T	T	F	F	F
T	F	F	T	F	F	T	F	T
F	T	F	T	T	F	T	F	T
F	F	F	F	T	T	F	T	T

not

α	¬α
T	F
T	T

0-place

⊤	⊥
T	F

argument

P ← premise 1
Q ← premise 2
∴ P ∧ Q ← conclusion (therefore)
also → P ∧ Q or ⊨ P ∧ Q

necessary truth preserving (NTP)
Argument is NTP if it's impossible that premises are true and conclusion is false. Argument can be NTP by it's content.

validity

Validity is NTP by form, not by content. Argument is valid, when there is no scenario in which all premises are true and conclusion is false.

check validity with truth table
Translate argument into PL
Construct truth table
Check the rows of the table, where the premises are T and see if the conclusion is F. If it's the case, the argument is invalid.

bound variable

scope of quantifier is subformula of it, the wff to which quantifier was attached during construction.
If variable in subformula is the same of quantifier, it is **bound**, otherwise it is **free**.
∀xRy - y is free
∀xFx - x is bound
∀x(Rx → Fx) - x is bound

model

Model 𝔐 is a scenario, situation in which proposoition of predicate logic is true or false.
$\mathfrak{M}_1$
Model has a **domain,** set of all objects that we are taking in account (= everything).
D: {Dante, Pirandello, Baricco, Vasari}
Extension of predicate is a subset, part of the domain objects that are true for this predicate (property).
Px: x is painter; P:{Vasari}
Wx: x is writer; W:{Dante, Baricco}
Referent is an object that respond to particular name
v: Vasari, b: Baricco

soundness

soundness
Actual row of true table - is that that represents the actual truth values of propositions (by their meaning)
Argument is **sound** if it's valid and its premises are true in actual row.

1) Grass is green. Snow is white ∴ Grass is green and show is white. ← valid and sound
2) Grass is red. Snow is green ∴ Grass is red and show is green. ← valid, not sound

S	G	S∧G	
T	T	T	← actual for 1st
T	F	F	
F	T	F	
F	F	F	← actual for 2nd

rules for semantic trees

one under the other - non branching

$(\alpha \wedge \beta)$
α
β

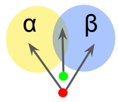
unnegated ∧
the conjunction is true when **both** of conjuncts is true

α	β	α∧β
T	T	T
T	F	F
F	T	F
F	F	F

create two branches - branching

$\neg(\alpha \wedge \beta)$
∧
$\neg\alpha \quad \neg\beta$

negated ∧
the conjunction is false when first conjunct is false **or** second conjunct is false.



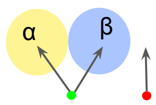
$(\alpha \vee \beta)$
∧
$\alpha \quad \beta$

unnegated ∨
the disjunction is true when first **or** second disjunct is true.

α	β	α∨β
T	T	T
T	F	T
F	T	T
F	F	F

$\neg(\alpha \vee \beta)$
$\neg\alpha$
$\neg\beta$

negated ∨
the disjunction is false when **both** of disjunct are false



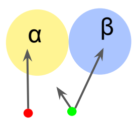
$(\alpha \rightarrow \beta)$
∧
$\neg\alpha \quad \beta$

unnegated →
α follows from β, conditional is true, if α is false or β is true.

α	β	α→β
T	T	T
T	F	F
F	T	T
F	F	T

$\neg(\alpha \rightarrow \beta)$
α
$\neg\beta$

negated →
α doesn't follow from β, conditional is false, if both α is true and β is false..



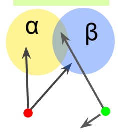
$(\alpha \leftrightarrow \beta)$
∧
$\alpha \quad \neg\alpha$
$\beta \quad \neg\beta$

unnegated ↔
biconditional is true, when both α and β have the same values

α	β	α↔β
T	T	T
T	F	F
F	T	F
F	F	T

$\neg(\alpha \leftrightarrow \beta)$
∧
$\alpha \quad \neg\alpha$
$\neg\beta \quad \beta$

negated ↔
biconditional is false when both α and β have opposite values



$\neg\neg\alpha$
α

double negation
negation of negation of expression is false, when expression is true.

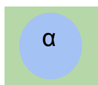
α	¬¬α
T	T
F	F

$\alpha \quad \neg I^2aa$
$\neg\alpha \quad (a \neq a)$
X X

closure
tree is closed when α and $\neg\alpha$ appear on a branch

$\exists x\alpha(x)$ ✓a (new a)
$\alpha(a/x)$

unnegated ∃
Exist x such as x is α, when there is at least one *a* (name) that is α.



$\neg\exists x\alpha(x)$
$\forall x\neg\alpha(x)$

negated ∃
It doesn't exist x such as x is α, when all x are not α.

$\forall x\alpha(x)$ \a (any a)
$\alpha(a/x)$

unnegated ∀
All x are α, when all names *a* , b, c... are α.

$\neg\forall x\alpha(x)$
$\exists x\neg\alpha(x)$

negated ∀
Not all x are α, when exist at least on x that are not α.

order of rule application

1. rules without quantifiers, nonbranching
2. rules without quantifiers, branching
3. negated quantifiers (¬∃ , ¬∀)
4. unnegated ∃ rule *(substitute for one new name)*
5. unnegated ∀ rule *(substitute for all names that appeared on the tree before)*

For large tree apply rule for the proposition with smaller "address", where address is assigned in this way:

```
  ∧
 11   12
111   121
1111   ∧
    1211  1212
```

infinite trees may appear with ∀x∃y

```
  α1
  ∧
α2  α3
     ∧
   α4  α5
        ∧
      α6 α7
```

Move quantifier for antecedent:
$(\forall x\alpha \rightarrow \beta) := \exists x(\alpha \rightarrow \beta)$
$(\exists x\alpha \rightarrow \beta) := \forall x(\alpha \rightarrow \beta)$
For rest of connectives:
$(\forall x\alpha \wedge \beta) := \forall x(\alpha \wedge \beta)$
$(\beta \wedge \forall x\alpha) := \forall x(\beta \wedge \alpha)$

tests for logical properties

Test if argument is **valid**: α, β ∴ γ	Check if it's possible that all premises are true and conclusion is false, so set is satisfiable. Solve the tree: α β ¬γ If all paths are closed argument is valid, if not - argument is invalid. Read from the tree.
Test if proposition α is **tautology** (always true)	Check if tree for the negation has solutions. Means negation cannot be true, so a proposition is always true. Solve the tree: **¬α** If all paths are closed, means proposition is tautology, otherwise read from the tree
Test if two propositions α, β are **equivalent**	Check if the negated biconditional has solutions. Solve the tree: **¬ (α ↔ β)** If all paths are closed α, β are equivalent, if some paths are open they are not. Read the case from the tree.
Test whether proposition α is **satisfiable** or a **contradiction**	Solve the tree: **α** If all paths close, proposition is contradiction. If a path remains open it's satisfiable
Test whether **set of propositions α, β are satisfiable**	Solve the tree: **α** **β** If all paths are closed, set is unsatisfiable. If not it is satisfiable.
Test whether two jointly unsatisfiable propositions are **contraries or contradictories**	Solve the tree: **¬ α** **¬ β** If all paths are closed propositions are contradictories, if a path is open they are contraries. Read off from open path a scenario in which both are F.

construct a tree

To check if proposition Fa→ ∃xFx is tautology. Write down a negation of the proposition:
¬ (Fa→ ∃xFx)
Assume it's T. Identify the main connective: **negated →** and apply the corresponding rule, write down resulting propositions.
1) ¬ (Fa→ ∃xFx) ✓
2) Fa ✓ {1, negated →, α}
3) ¬∃xFx ✓ {1. negated →, $\neg\beta$}
Identify next rule to apply: ¬∃ quantifier
4) ∀x¬Fx **\a** {3, negated ∃, ∀x¬α}
Substitute x with any name - a - (a/x), write \a on the row above
5) ¬F**a** ✓ {4, a}
Fa and ¬Fa appear in the tree. Close the tree.
6) x

Russelian description

Uniqueness assumption:
The Florence main cathedral is high. f: Florence, Cxy: x is m. cathedral of y
There is exactly one main cathedral: ∃x∀y(Cyf ↔ y = x)
Exist x such as for all y, if y is cathedral of Florence, it's actually x)
This cathedral is high: ∃x(∀y(Cyf ↔ y = x) ∧ Hx)
The architect of the Florence main cathedral is talented:
∃x(∀y(Cyf ↔ y = x) ∧ ∃z∀y(Ayx ↔ y = z))