**Propositional Logic – Proof Theory**

* **Transformational proof**
  + A means of determining that 2 formulas P & Q are logically equivalent by exchanging subformulas for logically equivalent subformulas → result in P being transformed into Q
  + Aka. Boolean algebra
  + Logical laws and equivalences are expressed using ↔
  + Rules of thumb:
    - Eliminate implication & equivalence
    - Simplify as soon as you can; except:
    - Sometimes use simplification backwards to set up for distributivity
  + Transformational proof is sound and complete
    - i.e. if P ↔ Q can be proved, then P <≡> Q
    - & if P <≡>, then P ↔ Q can be proved
    - Thus if P ↔ Q can be shown, we can use the soundness of transformational proof and conclude that P <≡> Q
  + P1, P2 |= Q can be proved by P1 ∧ P2 ⇒ Q ↔ true
  + Literal – a proposition symbol or the negation of one
  + Conjunctive normal form (CNF) – conjunction of disjunctions (or literals)
    - E.g. (p ∧ q) ∨ ¬q ∨ ¬r
  + Disjunctive normal form (DNF) – disjunction of conjunctions (or literals)
    - E.g. (p ∧ q ∧ r) ∨ (¬q ∧ ¬r)
  + Every formula can be converted to CNF/DNF
    - Remove all ⇒ and ⇔ using impl and equiv laws
    - Remove negations using neg or use dm to reduce their scope (push in)
    - Use distr laws to make sure only ∧ or ∨ is on the outside
    - Simplify to make sure there are no repeated literals, clauses, and no constants
* **Natural deduction**
  + An argumentis a collection of formulas
    - One of the formulas – conclusion – is justified by the other formulas – premises
    - Deductive argument – conclusion is wholly justified by the premises
    - Inductive argument – conclude new, more general information from a small number of facts or observations
  + To show an argument is invalid:
    - Find a counterexample – a B. v. where the premises are T and the conclusion is F
  + Natural deduction is a form of forward proof – starting from premises and arriving at the conclusion
  + Natural deduction uses inference rules
    - Inference rule – a primitive valid argument form
    - To use a rule, formulas matching the premises of the rule must appear on existing lines of the proof; we can then add the formula matching the conclusion of the rule to the proof
    - The rule must apply to the whole formula on that line
  + Ex: prove a ∧ b, c |− b ∧ c
    - 1) a ∧ b premise
    - 2) c premise
    - 3) b by and\_e on 1
    - 4) b ∧ c by and\_i on 2, 3
  + Ex: b ⇒ a, a ⇒ b, b ⇔ c, a |− (a ⇔ c) ∧ ¬¬(b ∨ c)
    - 1) b ⇒ a premise
    - 2) a ⇒ b premise
    - 3) b ⇔ c premise
    - 4) a premise
    - 5) a ⇔ b by iff\_i on 1, 2
    - 6) a ⇔ c by trans on 3, 5
    - 7) b by imp\_e on 2, 4
    - 8) b ∨ c by or\_i on 7
    - 9) ¬¬(b ∨ c) by not\_not\_i on 8
    - 10) (a ⇔ c) ∧ ¬¬(b ∨ c) by and\_i on 6, 9
  + **Subordinate proofs (subproofs)**
    - Start with a formula that is assumed to be true w/in the subproof
    - Then see what can be proved based on that assumption
    - Conditional proof, indirect proof, and case analysis use subproofs
    - Must be closed before the proof is finished
    - Can be nested
    - Form:

1) Subproof\_open P {

2) … (these lines cannot be used outside of the subproof)

3) Q by … (except for in the conclusion line)

}

4) R by … on 2-3 (conclusion from subproof)

* + **Conditional proof** – discharge the assumption

Assume P {

…

Q

}

P ⇒ Q by imp\_i on …

* + **Indirect proof/proof by contradiction**

Disprove R {

…

false

}

¬R by raa on …

* + - Use not\_e (e.g. P + ¬P → false) to create contradiction
  + **Case analysis**

P ∨ Q premise

case P {

…

R

}

case Q {

…

R

}

R by cases on 1, …

* + **Disjunctive syllogism**

1) P ∨ Q premise

2) ¬P premise

3) Q by or\_e on 1, 2

* + **Law of excluded middle**

1) Q (anything)

2) P ∨ ¬P by lem

* + Strategies
    - To prove:
      * A ∧ B ← prove both A and B
      * A ∨ B ← prove either A or B
      * A ⇒ B ← assume A then prove B (conditional proof)
      * A ← disprove ¬A (indirect proof)
    - Given:
      * A ∧ B → use and\_e to get A, B
      * A ∨ B → case analysis
      * A ⇒ B → prove A, then use imp\_e to get B
      * ¬A → prove A, use not\_e to derive contradiction
  + Natural deduction is also sound and complete
* **Semantic tableaux**
  + A semantic tableaux is a tree representing all the ways that the conjunction of the formulas at the root of the tree can be true
  + A branch is a path from the root to a leaf
  + Each step can either
    - Use a rule to expand one compound formula and add new formula(s) to that branch
      * Rules are based on the outermost connective in the formula
      * Branching captures disjunction – there are > 1 way to make the formula true
    - Close a branch because it contains contradictory formulas
      * E.g. if P and ¬P both appear in a branch
  + If every branch is closed (all branches contain contradictions) then the formulas at the root are inconsistent
  + Heuristic: apply the non-branching rules first
  + For an argument to be invalid
    - There must exist a B. v. where the premises = T and conclusion = F
    - i.e. premises P1, P2, … and negation of conclusion ¬Q is a consistent set of formulas
  + To show an argument is valid
    - Show that P1, P2, … ¬Q is an inconsistent set of formulas
    - i.e. put the premises and negation of the conclusion at the root of a semantic tableaux; if all branches can be closed, they are inconsistent and thus P1, P2, … |− Q
  + Semantic tableaux is based on proof by contradiction – a form of backward proof
  + Ex:

1) premise

2) premise

3) !conclusion

by rule on #

{

4) smaller formula

}

by rule on #

{

5) smaller formula

CLOSED on #, #

}

{

6) smaller formula

CLOSED on #, #

}

* + Semantic tableaux is sound and complete