**Theories of Arithmetic**

* Natural numbers: N = {0, 1, 2, …}
* Theory of arithmetic contains:
  + Constants: 0
  + Functions:
    - suc → suc(x) means x + 1
      * suc(0) = 1; suc(suc(0)) = 2
    - + → addition
    - × → multiplication
  + Predicates:
    - < → less than
* **Peano’s Axioms**
  + Axiom 1 – 0 is not a successor
    - ∀n . ¬(suc(n) = 0)
  + Axiom 2 – successors are distinct
    - ∀x, y . (suc(x) = suc(y)) ⇒ (x = y)
  + Axiom 3 – 0 is the identity of addition
    - ∀m . m + 0 = m
  + Axiom 4 – addition
    - ∀m, n . m + suc(n) = suc(m + n)
    - e.g. 3 + 2 = 5
      * 3 + 2

= suc(suc(suc(0))) + suc(suc(0))

= suc(suc(suc(suc(0)) + suc(0)) by axiom 4

= suc(suc(suc(suc(suc(0))) + 0) by axiom 4

= suc(suc(suc(suc(suc(0)))) by axiom 3

= 5

* + Axiom 5 – multiplication by 0
    - ∀n . n × 0 = 0
  + Axiom 6 – multiplication
    - ∀m, n . m × suc(n) = m × n + m
    - e.g. 2 × 2 = 4
      * 2 × 2

= suc(suc(0)) × suc(suc(0))

= suc(suc(0)) × suc(0) + suc(suc(0)) by axiom 6

= suc(suc(0)) × 0 + suc(suc(0)) + suc(suc(0)) by axiom 6

= 0 + suc(suc(0)) + suc(suc(0)) by axiom 5

= 4 using axiom 3, 4

* + Axiom 7 – induction
    - For all P: **P(0) ∧ (∀k . P(k) ⇒ P(suc(k))) ⇒ ∀n . P(n)**
    - P(0) – base case
    - P(k) – induction hypothesis
    - ∀k . P(k) ⇒ P(suc(k)) – induction proof/step
  + Intended model for theory of arithmetic:
    - Domain = N
    - Mapping:
      * 0 → 0
      * suc(.) → Suc(x) := x + 1
      * . + . → Plus(x, y) := x + y
      * . × . → Times(x, y) := x \* y
      * . < . → LessThan(x, y) := x < y
* **Mathematical induction**
  + Mathematical induction is a type of deductive reasoning
  + Induction works because N is ordered
    - Between any m, n ∈ N, there are only a finite number of x such that m < x < n
  + Induction inference rule
    - bc) P(0)

ih) inductionstep P(kg) {

…

x) P(kg + 1)

}

∀n . P(n) by induction on bc, ih-x

* + - This is a shorthand for a forall\_i (for every kg …) and imp\_i (assume P(kg) …)
    - We are proving that the predicate P holds true for all n ≥ the base case
* Practical approach
  + Use numbers as constants (0, 1, 2 …)
  + Use x + 1 instead of suc(x)
  + Use mathematical functions (+, ×, ∑ etc.)
  + Use predicates with “well-understood” meanings (<, >, even, odd, etc.)
  + Can use arithmetic to operate on subformulas
  + Use axioms that all numbers are distinct (¬(1 = 2), ¬(5 = 7), etc.)
  + Equalities:
    - equals f(x) {

g(x)

}

* + - Is equivalent to
    - f(x) = g(x)
* Recursive function
  + Defined by:
    - Its value for 0
    - Its value for n + 1 in terms of n, or for n in terms of n – 1
    - E.g. ∑(i = 0 → n) = n + ∑(i = 0 → n – 1)