(Sec. 3.3 pp. 182-187). **Algorithms**

- Key distinction re TMs and languages
  - TM T **recognizes** L if for all w in L T accepts w
    - Says nothing about what if w not in L
  - TM **decides** L if
    - T recognizes L
    - If w not in L, T always halts (in reject state)
- Hilbert's 10^{th} problem (1900): *Can any algorithm tell if a polynomial equation has any integer roots?*
  - Sample polynomial equation: $6x^3yz^2+3xy^2-x^3-10=0$
  - Example does at x=5, y=3, z=0
  - Critical point: we want **yes/no** answer for any polynomial
  - 1970: no such algorithm exists
- Key starting point: what is an "algorithm"?
- Key Definition: 1936 **Church-Turing Thesis**
  - Any function over the natural #s is computable by a algorithm iff it is computable by a TM
  - Each transition of a TM is a "**step**"
    - Step takes finite time
  - Finite # of steps to get to accepting state
- "*Does algorithm exist*" eqvt to "*Is there a TM decider*"

- Back to Hilbert
  - Define D = {p|p is a polynomial with an integral root}
  - D is **recognizable**:
    - Consider D_1={p|p a polynomial over single variable x with an integral root}
    - Recognizing TM M_1: Assume input string defines a p
      - Start an *enumerator TM* to generate 0, 1 -1, 2, -2, …
      - For each value compute p at that value
      - If a root, halt and accept
    - Note: if p has no integral roots, M_1 loops
    - TM recognizer for general D generates all cases of integers 1 at a time
  - Hilbert's 10th problem equivalent: does some TM **decide** D
    - I.e. Does some TM ***always halt*** for any p
  - For D_1 (exactly 1 variable) there are bounds that can constrain solution space (see p. 184 and problem 3.21)
    - Thus we can halt M_1 as soon as we reach these bounds
    - Thus modified M_1 is a **decider** for D_1
  - Theorem from 1970: no such bounds exist for multi-variable polynomials
    - **Cannot construct a decider for D** same way as for D_1
- When deciders exist: ***do polynomial time TMs exist?***

- (p. 184) Terminology for describing TMs
  - (p. 185) 3 ways for describing TMs
    - **Formal Description:** 7 tuple and δ
    - **Implementation Description**: use English prose to describe tape movements and tape writing
    - **High-level Description**: English prose to describe algorithm, ignoring implementation details
      - Often building one TM out of composition of others
  - (p.185)Notation for describing TM tapes(esp. initial tapes)
    - Tape always contains a **string**
    - Use strings to represent objects (#s,grammars, graphs..)
    - TM can be written to "decode" string representations
    - Notation for string representation of object O**: <O>**
    - Notation for multiple objects $O_1, O_2, \ldots O_k = \langle O_1, O_2, \ldots O_k \rangle$
    - TM algorithm described as indented lines of text
      - Each a **stage**: multiple TM operations
      - Assume initial stage checks format of input tape