Specifying Program for Correctness

- There are two notions of correctness of a program
  (i) **syntactical** : program has correct syntax so it will compile and run
  (ii) **semantical** : program correctly implements the intended specification

Example: "Tree can move", is syntatically correct but not semantically correct.

- Semantic correctness of program requires (i) termination of program, and (ii) correctness of function implemented
- (ii) alone called partial correctness; (i) & (ii) together called total correctness
- Termination of a program on a given input is in general undecidable. (popularly known as halting problem), and so is proving functional correctness. (Meaning both require human reasoning).

- We discuss how to specify (and verify) for functional correctness. 1st order logic can be used to specify functional correctness

prog. P with n inputs & m outputs

- Precondition, $pre(i_1, \ldots, i_n)$: 1st order formula specifying property of inputs prior to P's execution

  Postcondition, $post(i_1 \ldots i_n, o_1 \ldots o_m)$: 1st-order formula specifying property of inputs & outputs after P's execution

$\vec{pre}(\vec{i})\ \{P\}\ post(\vec{i}, \vec{o})$ ← known as "Hoare triple"

- Function correctness specified by pre & post conditions read as: If $pre(\vec{i})$ true before execution, then $post(\vec{i}, \vec{o})$ after execution.