Summary:
Given a project, a TS, and a current directory, what should the behavior be for **add, remove, commit**.

Notation:
TS, CD, PC (Data fileState = New | Altered | Unaltered | Deleted) defined in "Terms for Versioning"

Add

- Fail
  - When (file not in TS) && (file not in CD)
    - Report error
- Success
  - When (file in TS) && (file in CD)
    - Do nothing
  - When (file in TS) && (file not in CD)
    - TS = TS \ file
  - When (file not in TS) && (file in CD)
    - TS = TS ∪ file
    - Report success

Remove

- Fail
  - When (file not in TS)
    - Report error
- Success
  - When (file in TS)
    - TS = TS \ file
    - Report success

cleanTS

- Used inside commit to clear erroneous files
- For each file in TS
  - If (file not in CD)
    - TS = TS \ file

Note that add, remove and cleanTS are functions in the TS module(a sub-module of the concept module)

getFileState: project -> set -> file -> fileState

- Given a file, the file state is a function of CD, TS, and PC.

Theorem: All files in {PC u TS} must be classified as some fileState.
Pf: attached as image.

Commit: (1) Clean TS (2) get the states of all files {PC u TS} <-(see theorem above).

- Defer
    - If there are no New or Altered files
        - Report no committable changes
- Success: create a commit
    - Otherwise
        - Create commit in .dvcs subdirectory
        - Report to user

-------Note-----

- Design Decision for add. If a file is created, added, rm'ed then add'ed, it will be removed from tracked set.
- ex)
    - >> Touch b.txt
    - [CD = {b.txt}, TS = {}]
    - >> dvcs add b.txt
    - [CD = {b.txt}, TS = {b.txt}]
    - >> rm b.txt
    - [CD = {}, TS = {b.txt}]
    - >> dvcs add b.txt
    - [CD = {}, TS = {}]
    - >> touch b.txt
    - [CD = {b.txt}, TS = {}]
    - >> dvcs add b.txt
    - [CD = {b.txt}, TS = {b.txt}]