1 Overview of key modules

- `constants`: Global module holding the invariants, e.g. V
  - `thresh(n)`: $\alpha_n = \left(\frac{1}{2}\right)^n \frac{4}{5}$
  - `Version`: Enum of "version" being studied. A version is defined by V (restrictions on x, y, z), the positive state, negative state (i.e. subtracted from $f(\text{pos\_state})$), and name of the data directory.
    E.g. Version `XYZ` studies $f(x, y, z) - f(y, x, z)$ (i.e. Theorem 3). So:
    - `restriction`: $V = \{x, y, z : x > 0, x < y, y < z\}$
    - `pos_state`: (x, y, z)
    - `neg_state`: (y, x, z)
    - `data_dir`: "`xy/`"
  - The constant `VERSION` defines the version being studied.
  - Inside the data directory, we have a cache for $h_n(\text{pos\_state})$ and $h_n(\text{neg\_state})$.
    E.g. `h_n(x,y,z)` is a cache for $h_n(x, y, z)$, in csv format, where each row is in the form (n, constant, indicator constraints).
- `gen_h`: Contains main logic for calculating $h_n(\cdot)$, which is stored as a constant (from indicator constraints that must be true under V), and a set of indicator constraints.
- `write_h`: Writes the output of `gen_h` to the corresponding cache (see `write(n,a,b,c,filename)`).
- `verification`: Calculates $h(\cdot)$, $\Delta(\cdot)$, using real-valued states
  - `h_R(n,a,b,c)`: $h_n(a, b, c)$
  - `dh_R(n,a,b,c)`: $h_n(\text{pos\_state}(a, b, c)) - h_n(\text{neg\_state}(a, b, c))$
- `mip`: MIP (Mixed Integer Programming) models to find lower-bounds on $\Delta_n(\cdot)$ and $h_n(\text{pos\_state}) - h_n(\text{neg\_state})$
  - `delta_mip(n)`: lower bounds
    $$\Delta_n(\text{version}) - \alpha_n = \sum_{i=1}^{n} h_n(\text{pos\_state}) - \sum_{i=1}^{n} h_n(\text{neg\_state}) - \alpha_n$$
  - `dh_mip(n)`: lower bounds $h_n(\text{pos\_state}) - h_n(\text{neg\_state})$
  - `relax`: Parameter to the MIP models – whether to run a relaxation on the model after analysing the model.
    (For intuition only; not to be used in proofs.)
- `plot`: Generates plots of (x, y, z) coordinates for which $\Delta_n(x, y, z) > \alpha_n$ (see `plot_ns`).

2 Overview of helper modules

- `gen_coords`: Generates coordinates for the plot.
- `li_handler`: Handles logic of linear inequalities (i.e. test redundancy, feasibility, equivalence, etc).
- `simulation`: Simulates the game, for intuition.

3 Rough outline of studying a proof

E.g. To study a part of Theorem 3, i.e. for $0 < x < y < z$, $f(x, y, z) > f(y, x, z)$:

1. Define a new version in `constants`.
2. `VERSION :=` the version just defined.
3. Run `write_h` to generate the caches for h_n's (typically, first run $n = 1...5$).
4. Run `delta_mip(n-1)` and `dh_mip(n)` starting from $n = 2$.

    - Check the objective values.
    - If $\Delta_{n-1} + dh_n + \alpha_n > 0$, stop.
      We've found the n for which $\Delta_n > \alpha_n$, as required.
    - Else, return to step 3 and proceed with larger n values.