AI Lab 02: PL Resolution

CSC14003 "Introduction to Artificial Intelligence" @ 18CLC6
18127221 Bùi Văn Thiện

OVERVIEW

This is a naive implementation of the PL Resolution algorithm.

QUICK START

Clear the `output/` (if needed), place all the input files in `input/`, then run `PLRes.py`.

There are three versions of source code:

- **`PLRes.ipynb`**: the Jupyter Notebook that's primarily used to write and test the code. This is meant to be run on a single input file, with no output files and verbose logs printed out in runtime for inspection.
- **`PLRes.ipynb.html`**: the exact same Notebook as above, but prerendered as HTML (for machines that do not have `jupyter-notebook` or `jupyterlab` preinstalled.)
- **`PLRes.py`**: contains comment-stripped source from Notebook and a slight modification that enables batch processing all input files. Only output files are available; no verbose logs will be printed out.

Functions

`neg(x)`

Negate a single literal (`'-P'` $\rightarrow$ `'P'`, and vice versa).

`parse(file_object)`

Parse the input file and returns KB and alpha.

It mostly strips the $\vee$ connective (`' OR '`) from the source and place the clauses into a list, with the clauses themselves being a list of literals (strings)

`neg_alpha(alpha)`

Negate the whole α, so that it can be appended to KB.

There are five stages:

1. Negate all clauses
2. "Perform CNF conversion" (This is just a Cartesian product of all literals stringed together to form a list of lists that's coincidentally a valid list of or-clauses.)
3. Eliminate tautologies (clauses with both X and $\neg X$, since $X \vee \neg X \iff \text{True}$, and such clauses don't make much sense.)
4. Eliminate duplicate literals within clauses ($X \vee X \vee \cdots \vee X \iff X$)
5. Eliminate duplicate clauses

Personal reflection: since step two of is basically a hack [1], the output of this function can only be guaranteed to be the equivalent of $\neg\alpha$ in terms of truth tables / the model-checking approach, **not** the clauses themselves. However, we consider it to be good enough to append to KB (since they are *technically* CNF clauses; it's just that they are not *compact / optimal*.)

`resolve(a, b)`

Apply the inference rule to clause a and b.

Given that $a \iff P \vee Q$, and $b \iff \neg Q \vee R$:

$$P \vee Q, \neg Q \vee R \vdash P \vee R$$
or
$$a, b \vdash P \vee R$$

There are three stages:

1. String together the literals of two clauses
2. Check for contradictiory pair of clauses (X and $\neg X$). If there is one, remove that pair.
3. Check if clause is now a tautology. If it is, stop there and return nothing.
4. Check for changes in the resulting clause.
    1. If nothing's changed, return nothing.
    2. If something's changed (indicated by the difference of clause length): remove duplicate literals, sort them alphabetically, and return the resulting clause.

`iterate(input_kb, verbose=False)`

Perform one iteration of the PL-Resolution algorithm.

It simply attempts to apply `resolve()` to all (non-repeating) combination of clauses from KB and recording yields (resulting clauses.)

If `verbose` is enabled, it will output the result of each `resolve()` following this Python `.format` string:

```
{s} {a} + {b} → {r}
```

where:

- `{s}` is one of three possible statuses: `-` (normal), `k` (yield already exists in KB), and `y` (yield already exists in current yield batch.)
- `{a}`, `{b}` and `{r}` are the pair of clauses and the results of `resolve()` on said pair, respectively.

`run(input_kb, input_alpha)`

Optional parameters:

- `verbose`: toggle logs in `run()` itself
- `verbose_iterate`: controls the `verbose` flag in `iterate()`
- `write_to_file`: toggle output file writing

Run `iterate()` on given KB and α until an empty clause is found, or the yield is empty.

If `verbose` is enabled, the function will output the following:

- Initial KB (with $\neg\alpha$ appended)
- Iteration count, yields and yield count
- Final conclusion (`YES` or `NO`, once there's empty yield)

Sample run with logs

This is a sample run of the input file `X-HW03_alpha.txt` with verbose logs enabled (flag `verbose` and `verbose_iterate` in `run()` set to `True`.)

INPUT	OUTPUT
1 U 5 P OR Q -Q OR R -Q OR S -P OR U -R OR U	6 P OR R P OR S Q OR U -Q OR U -P -R 9 P OR U Q R OR U -Q S OR U R P S U 1 {} YES

LOG

```
Initial KB:
0: ['P', 'Q']
1: ['-Q', 'R']
2: ['-Q', 'S']
3: ['-P', 'U']
4: ['-R', 'U']
5: ['-U']


Iteration 0
- ['P', 'Q'] + ['-Q', 'R'] → ['P', 'R']
- ['P', 'Q'] + ['-Q', 'S'] → ['P', 'S']
- ['P', 'Q'] + ['-P', 'U'] → ['Q', 'U']
- ['P', 'Q'] + ['-R', 'U'] → None
- ['P', 'Q'] + ['-U'] → None
y ['-Q', 'R'] + ['P', 'Q'] → ['P', 'R']
- ['-Q', 'R'] + ['-Q', 'S'] → None
- ['-Q', 'R'] + ['-P', 'U'] → None
- ['-Q', 'R'] + ['-R', 'U'] → ['-Q', 'U']
- ['-Q', 'R'] + ['-U'] → None
y ['-Q', 'S'] + ['P', 'Q'] → ['P', 'S']
- ['-Q', 'S'] + ['-Q', 'R'] → None
- ['-Q', 'S'] + ['-P', 'U'] → None
- ['-Q', 'S'] + ['-R', 'U'] → None
- ['-Q', 'S'] + ['-U'] → None
y ['-P', 'U'] + ['P', 'Q'] → ['Q', 'U']
- ['-P', 'U'] + ['-Q', 'R'] → None
- ['-P', 'U'] + ['-Q', 'S'] → None
- ['-P', 'U'] + ['-R', 'U'] → None
- ['-P', 'U'] + ['-U'] → ['-P']
- ['-R', 'U'] + ['P', 'Q'] → None
y ['-R', 'U'] + ['-Q', 'R'] → ['-Q', 'U']
- ['-R', 'U'] + ['-Q', 'S'] → None
- ['-R', 'U'] + ['-P', 'U'] → None
- ['-R', 'U'] + ['-U'] → ['-R']
- ['-U'] + ['P', 'Q'] → None
- ['-U'] + ['-Q', 'R'] → None
- ['-U'] + ['-Q', 'S'] → None
y ['-U'] + ['-P', 'U'] → ['-P']
y ['-U'] + ['-R', 'U'] → ['-R']
yield length = 6
0: ['P', 'R']
1: ['P', 'S']
2: ['Q', 'U']
3: ['-Q', 'U']
4: ['-P']
5: ['-R']


Iteration 1
k ['P', 'Q'] + ['-Q', 'R'] → ['P', 'R']
k ['P', 'Q'] + ['-Q', 'S'] → ['P', 'S']
k ['P', 'Q'] + ['-P', 'U'] → ['Q', 'U']
```

```
- ['P', 'Q'] + ['-R', 'U'] → None
- ['P', 'Q'] + ['-U'] → None
- ['P', 'Q'] + ['P', 'R'] → None
- ['P', 'Q'] + ['P', 'S'] → None
- ['P', 'Q'] + ['Q', 'U'] → None
- ['P', 'Q'] + ['-Q', 'U'] → ['P', 'U']
- ['P', 'Q'] + ['-P'] → ['Q']
- ['P', 'Q'] + ['-R'] → None
k ['-Q', 'R'] + ['P', 'Q'] → ['P', 'R']
- ['-Q', 'R'] + ['-Q', 'S'] → None
- ['-Q', 'R'] + ['-P', 'U'] → None
k ['-Q', 'R'] + ['-R', 'U'] → ['-Q', 'U']
- ['-Q', 'R'] + ['-U'] → None
- ['-Q', 'R'] + ['P', 'R'] → None
- ['-Q', 'R'] + ['P', 'S'] → None
- ['-Q', 'R'] + ['Q', 'U'] → ['R', 'U']
- ['-Q', 'R'] + ['-Q', 'U'] → None
- ['-Q', 'R'] + ['-P'] → None
- ['-Q', 'R'] + ['-R'] → ['-Q']
k ['-Q', 'S'] + ['P', 'Q'] → ['P', 'S']
- ['-Q', 'S'] + ['-Q', 'R'] → None
- ['-Q', 'S'] + ['-P', 'U'] → None
- ['-Q', 'S'] + ['-R', 'U'] → None
- ['-Q', 'S'] + ['-U'] → None
- ['-Q', 'S'] + ['P', 'R'] → None
- ['-Q', 'S'] + ['P', 'S'] → None
- ['-Q', 'S'] + ['Q', 'U'] → ['S', 'U']
- ['-Q', 'S'] + ['-Q', 'U'] → None
- ['-Q', 'S'] + ['-P'] → None
- ['-Q', 'S'] + ['-R'] → None
k ['-P', 'U'] + ['P', 'Q'] → ['Q', 'U']
- ['-P', 'U'] + ['-Q', 'R'] → None
- ['-P', 'U'] + ['-Q', 'S'] → None
- ['-P', 'U'] + ['-R', 'U'] → None
k ['-P', 'U'] + ['-U'] → ['-P']
y ['-P', 'U'] + ['P', 'R'] → ['R', 'U']
y ['-P', 'U'] + ['P', 'S'] → ['S', 'U']
- ['-P', 'U'] + ['Q', 'U'] → None
- ['-P', 'U'] + ['-Q', 'U'] → None
- ['-P', 'U'] + ['-P'] → None
- ['-P', 'U'] + ['-R'] → None
- ['-R', 'U'] + ['P', 'Q'] → None
k ['-R', 'U'] + ['-Q', 'R'] → ['-Q', 'U']
- ['-R', 'U'] + ['-Q', 'S'] → None
- ['-R', 'U'] + ['-P', 'U'] → None
k ['-R', 'U'] + ['-U'] → ['-R']
y ['-R', 'U'] + ['P', 'R'] → ['P', 'U']
- ['-R', 'U'] + ['P', 'S'] → None
- ['-R', 'U'] + ['Q', 'U'] → None
- ['-R', 'U'] + ['-Q', 'U'] → None
- ['-R', 'U'] + ['-P'] → None
- ['-R', 'U'] + ['-R'] → None
- ['-U'] + ['P', 'Q'] → None
- ['-U'] + ['-Q', 'R'] → None
- ['-U'] + ['-Q', 'S'] → None
```

```
k ['-U'] + ['-P', 'U'] → ['-P']
k ['-U'] + ['-R', 'U'] → ['-R']
- ['-U'] + ['P', 'R'] → None
- ['-U'] + ['P', 'S'] → None
y ['-U'] + ['Q', 'U'] → ['Q']
y ['-U'] + ['-Q', 'U'] → ['-Q']
- ['-U'] + ['-P'] → None
- ['-U'] + ['-R'] → None
- ['P', 'R'] + ['P', 'Q'] → None
- ['P', 'R'] + ['-Q', 'R'] → None
- ['P', 'R'] + ['-Q', 'S'] → None
y ['P', 'R'] + ['-P', 'U'] → ['R', 'U']
y ['P', 'R'] + ['-R', 'U'] → ['P', 'U']
- ['P', 'R'] + ['-U'] → None
- ['P', 'R'] + ['P', 'S'] → None
- ['P', 'R'] + ['Q', 'U'] → None
- ['P', 'R'] + ['-Q', 'U'] → None
- ['P', 'R'] + ['-P'] → ['R']
- ['P', 'R'] + ['-R'] → ['P']
- ['P', 'S'] + ['P', 'Q'] → None
- ['P', 'S'] + ['-Q', 'R'] → None
- ['P', 'S'] + ['-Q', 'S'] → None
y ['P', 'S'] + ['-P', 'U'] → ['S', 'U']
- ['P', 'S'] + ['-R', 'U'] → None
- ['P', 'S'] + ['-U'] → None
- ['P', 'S'] + ['P', 'R'] → None
- ['P', 'S'] + ['Q', 'U'] → None
- ['P', 'S'] + ['-Q', 'U'] → None
- ['P', 'S'] + ['-P'] → ['S']
- ['P', 'S'] + ['-R'] → None
- ['Q', 'U'] + ['P', 'Q'] → None
y ['Q', 'U'] + ['-Q', 'R'] → ['R', 'U']
y ['Q', 'U'] + ['-Q', 'S'] → ['S', 'U']
- ['Q', 'U'] + ['-P', 'U'] → None
- ['Q', 'U'] + ['-R', 'U'] → None
y ['Q', 'U'] + ['-U'] → ['Q']
- ['Q', 'U'] + ['P', 'R'] → None
- ['Q', 'U'] + ['P', 'S'] → None
- ['Q', 'U'] + ['-Q', 'U'] → ['U']
- ['Q', 'U'] + ['-P'] → None
- ['Q', 'U'] + ['-R'] → None
y ['-Q', 'U'] + ['P', 'Q'] → ['P', 'U']
- ['-Q', 'U'] + ['-Q', 'R'] → None
- ['-Q', 'U'] + ['-Q', 'S'] → None
- ['-Q', 'U'] + ['-P', 'U'] → None
- ['-Q', 'U'] + ['-R', 'U'] → None
y ['-Q', 'U'] + ['-U'] → ['-Q']
- ['-Q', 'U'] + ['P', 'R'] → None
- ['-Q', 'U'] + ['P', 'S'] → None
y ['-Q', 'U'] + ['Q', 'U'] → ['U']
- ['-Q', 'U'] + ['-P'] → None
- ['-Q', 'U'] + ['-R'] → None
y ['-P'] + ['P', 'Q'] → ['Q']
- ['-P'] + ['-Q', 'R'] → None
- ['-P'] + ['-Q', 'S'] → None
```

```
- ['-P'] + ['-P', 'U'] → None
- ['-P'] + ['-R', 'U'] → None
- ['-P'] + ['-U'] → None
y ['-P'] + ['P', 'R'] → ['R']
y ['-P'] + ['P', 'S'] → ['S']
- ['-P'] + ['Q', 'U'] → None
- ['-P'] + ['-Q', 'U'] → None
- ['-P'] + ['-R'] → None
- ['-R'] + ['P', 'Q'] → None
y ['-R'] + ['-Q', 'R'] → ['-Q']
- ['-R'] + ['-Q', 'S'] → None
- ['-R'] + ['-P', 'U'] → None
- ['-R'] + ['-R', 'U'] → None
- ['-R'] + ['-U'] → None
y ['-R'] + ['P', 'R'] → ['P']
- ['-R'] + ['P', 'S'] → None
- ['-R'] + ['Q', 'U'] → None
- ['-R'] + ['-Q', 'U'] → None
- ['-R'] + ['-P'] → None
yield length = 9
0: ['P', 'U']
1: ['Q']
2: ['R', 'U']
3: ['-Q']
4: ['S', 'U']
5: ['R']
6: ['P']
7: ['S']
8: ['U']


Iteration 2
k ['P', 'Q'] + ['-Q', 'R'] → ['P', 'R']
k ['P', 'Q'] + ['-Q', 'S'] → ['P', 'S']
k ['P', 'Q'] + ['-P', 'U'] → ['Q', 'U']
- ['P', 'Q'] + ['-R', 'U'] → None
- ['P', 'Q'] + ['-U'] → None
- ['P', 'Q'] + ['P', 'R'] → None
- ['P', 'Q'] + ['P', 'S'] → None
- ['P', 'Q'] + ['Q', 'U'] → None
k ['P', 'Q'] + ['-Q', 'U'] → ['P', 'U']
k ['P', 'Q'] + ['-P'] → ['Q']
- ['P', 'Q'] + ['-R'] → None
- ['P', 'Q'] + ['P', 'U'] → None
- ['P', 'Q'] + ['Q'] → None
- ['P', 'Q'] + ['R', 'U'] → None
k ['P', 'Q'] + ['-Q'] → ['P']
- ['P', 'Q'] + ['S', 'U'] → None
- ['P', 'Q'] + ['R'] → None
- ['P', 'Q'] + ['P'] → None
- ['P', 'Q'] + ['S'] → None
- ['P', 'Q'] + ['U'] → None
k ['-Q', 'R'] + ['P', 'Q'] → ['P', 'R']
- ['-Q', 'R'] + ['-Q', 'S'] → None
- ['-Q', 'R'] + ['-P', 'U'] → None
```

```
k ['-Q', 'R'] + ['-R', 'U'] → ['-Q', 'U']
- ['-Q', 'R'] + ['-U'] → None
- ['-Q', 'R'] + ['P', 'R'] → None
- ['-Q', 'R'] + ['P', 'S'] → None
k ['-Q', 'R'] + ['Q', 'U'] → ['R', 'U']
- ['-Q', 'R'] + ['-Q', 'U'] → None
- ['-Q', 'R'] + ['-P'] → None
k ['-Q', 'R'] + ['-R'] → ['-Q']
- ['-Q', 'R'] + ['P', 'U'] → None
k ['-Q', 'R'] + ['Q'] → ['R']
- ['-Q', 'R'] + ['R', 'U'] → None
- ['-Q', 'R'] + ['-Q'] → None
- ['-Q', 'R'] + ['S', 'U'] → None
- ['-Q', 'R'] + ['R'] → None
- ['-Q', 'R'] + ['P'] → None
- ['-Q', 'R'] + ['S'] → None
- ['-Q', 'R'] + ['U'] → None
k ['-Q', 'S'] + ['P', 'Q'] → ['P', 'S']
- ['-Q', 'S'] + ['-Q', 'R'] → None
- ['-Q', 'S'] + ['-P', 'U'] → None
- ['-Q', 'S'] + ['-R', 'U'] → None
- ['-Q', 'S'] + ['-U'] → None
- ['-Q', 'S'] + ['P', 'R'] → None
- ['-Q', 'S'] + ['P', 'S'] → None
k ['-Q', 'S'] + ['Q', 'U'] → ['S', 'U']
- ['-Q', 'S'] + ['-Q', 'U'] → None
- ['-Q', 'S'] + ['-P'] → None
- ['-Q', 'S'] + ['-R'] → None
- ['-Q', 'S'] + ['P', 'U'] → None
k ['-Q', 'S'] + ['Q'] → ['S']
- ['-Q', 'S'] + ['R', 'U'] → None
- ['-Q', 'S'] + ['-Q'] → None
- ['-Q', 'S'] + ['S', 'U'] → None
- ['-Q', 'S'] + ['R'] → None
- ['-Q', 'S'] + ['P'] → None
- ['-Q', 'S'] + ['S'] → None
- ['-Q', 'S'] + ['U'] → None
k ['-P', 'U'] + ['P', 'Q'] → ['Q', 'U']
- ['-P', 'U'] + ['-Q', 'R'] → None
- ['-P', 'U'] + ['-Q', 'S'] → None
- ['-P', 'U'] + ['-R', 'U'] → None
k ['-P', 'U'] + ['-U'] → ['-P']
k ['-P', 'U'] + ['P', 'R'] → ['R', 'U']
k ['-P', 'U'] + ['P', 'S'] → ['S', 'U']
- ['-P', 'U'] + ['Q', 'U'] → None
- ['-P', 'U'] + ['-Q', 'U'] → None
- ['-P', 'U'] + ['-P'] → None
- ['-P', 'U'] + ['-R'] → None
k ['-P', 'U'] + ['P', 'U'] → ['U']
- ['-P', 'U'] + ['Q'] → None
- ['-P', 'U'] + ['R', 'U'] → None
- ['-P', 'U'] + ['-Q'] → None
- ['-P', 'U'] + ['S', 'U'] → None
- ['-P', 'U'] + ['R'] → None
k ['-P', 'U'] + ['P'] → ['U']
```

```
- ['-P', 'U'] + ['S'] → None
- ['-P', 'U'] + ['U'] → None
- ['-R', 'U'] + ['P', 'Q'] → None
k ['-R', 'U'] + ['-Q', 'R'] → ['-Q', 'U']
- ['-R', 'U'] + ['-Q', 'S'] → None
- ['-R', 'U'] + ['-P', 'U'] → None
k ['-R', 'U'] + ['-U'] → ['-R']
k ['-R', 'U'] + ['P', 'R'] → ['P', 'U']
- ['-R', 'U'] + ['P', 'S'] → None
- ['-R', 'U'] + ['Q', 'U'] → None
- ['-R', 'U'] + ['-Q', 'U'] → None
- ['-R', 'U'] + ['-P'] → None
- ['-R', 'U'] + ['-R'] → None
- ['-R', 'U'] + ['P', 'U'] → None
- ['-R', 'U'] + ['Q'] → None
k ['-R', 'U'] + ['R', 'U'] → ['U']
- ['-R', 'U'] + ['-Q'] → None
- ['-R', 'U'] + ['S', 'U'] → None
k ['-R', 'U'] + ['R'] → ['U']
- ['-R', 'U'] + ['P'] → None
- ['-R', 'U'] + ['S'] → None
- ['-R', 'U'] + ['U'] → None
- ['-U'] + ['P', 'Q'] → None
- ['-U'] + ['-Q', 'R'] → None
- ['-U'] + ['-Q', 'S'] → None
k ['-U'] + ['-P', 'U'] → ['-P']
k ['-U'] + ['-R', 'U'] → ['-R']
- ['-U'] + ['P', 'R'] → None
- ['-U'] + ['P', 'S'] → None
k ['-U'] + ['Q', 'U'] → ['Q']
k ['-U'] + ['-Q', 'U'] → ['-Q']
- ['-U'] + ['-P'] → None
- ['-U'] + ['-R'] → None
k ['-U'] + ['P', 'U'] → ['P']
- ['-U'] + ['Q'] → None
k ['-U'] + ['R', 'U'] → ['R']
- ['-U'] + ['-Q'] → None
k ['-U'] + ['S', 'U'] → ['S']
- ['-U'] + ['R'] → None
- ['-U'] + ['P'] → None
- ['-U'] + ['S'] → None
- ['-U'] + ['U'] → []
- ['P', 'R'] + ['P', 'Q'] → None
- ['P', 'R'] + ['-Q', 'R'] → None
- ['P', 'R'] + ['-Q', 'S'] → None
k ['P', 'R'] + ['-P', 'U'] → ['R', 'U']
k ['P', 'R'] + ['-R', 'U'] → ['P', 'U']
- ['P', 'R'] + ['-U'] → None
- ['P', 'R'] + ['P', 'S'] → None
- ['P', 'R'] + ['Q', 'U'] → None
- ['P', 'R'] + ['-Q', 'U'] → None
k ['P', 'R'] + ['-P'] → ['R']
k ['P', 'R'] + ['-R'] → ['P']
- ['P', 'R'] + ['P', 'U'] → None
- ['P', 'R'] + ['Q'] → None
```

```
- ['P', 'R'] + ['R', 'U'] → None
- ['P', 'R'] + ['-Q'] → None
- ['P', 'R'] + ['S', 'U'] → None
- ['P', 'R'] + ['R'] → None
- ['P', 'R'] + ['P'] → None
- ['P', 'R'] + ['S'] → None
- ['P', 'R'] + ['U'] → None
- ['P', 'S'] + ['P', 'Q'] → None
- ['P', 'S'] + ['-Q', 'R'] → None
- ['P', 'S'] + ['-Q', 'S'] → None
k ['P', 'S'] + ['-P', 'U'] → ['S', 'U']
- ['P', 'S'] + ['-R', 'U'] → None
- ['P', 'S'] + ['-U'] → None
- ['P', 'S'] + ['P', 'R'] → None
- ['P', 'S'] + ['Q', 'U'] → None
- ['P', 'S'] + ['-Q', 'U'] → None
k ['P', 'S'] + ['-P'] → ['S']
- ['P', 'S'] + ['-R'] → None
- ['P', 'S'] + ['P', 'U'] → None
- ['P', 'S'] + ['Q'] → None
- ['P', 'S'] + ['R', 'U'] → None
- ['P', 'S'] + ['-Q'] → None
- ['P', 'S'] + ['S', 'U'] → None
- ['P', 'S'] + ['R'] → None
- ['P', 'S'] + ['P'] → None
- ['P', 'S'] + ['S'] → None
- ['P', 'S'] + ['U'] → None
- ['Q', 'U'] + ['P', 'Q'] → None
k ['Q', 'U'] + ['-Q', 'R'] → ['R', 'U']
k ['Q', 'U'] + ['-Q', 'S'] → ['S', 'U']
- ['Q', 'U'] + ['-P', 'U'] → None
- ['Q', 'U'] + ['-R', 'U'] → None
k ['Q', 'U'] + ['-U'] → ['Q']
- ['Q', 'U'] + ['P', 'R'] → None
- ['Q', 'U'] + ['P', 'S'] → None
k ['Q', 'U'] + ['-Q', 'U'] → ['U']
- ['Q', 'U'] + ['-P'] → None
- ['Q', 'U'] + ['-R'] → None
- ['Q', 'U'] + ['P', 'U'] → None
- ['Q', 'U'] + ['Q'] → None
- ['Q', 'U'] + ['R', 'U'] → None
k ['Q', 'U'] + ['-Q'] → ['U']
- ['Q', 'U'] + ['S', 'U'] → None
- ['Q', 'U'] + ['R'] → None
- ['Q', 'U'] + ['P'] → None
- ['Q', 'U'] + ['S'] → None
- ['Q', 'U'] + ['U'] → None
k ['-Q', 'U'] + ['P', 'Q'] → ['P', 'U']
- ['-Q', 'U'] + ['-Q', 'R'] → None
- ['-Q', 'U'] + ['-Q', 'S'] → None
- ['-Q', 'U'] + ['-P', 'U'] → None
- ['-Q', 'U'] + ['-R', 'U'] → None
k ['-Q', 'U'] + ['-U'] → ['-Q']
- ['-Q', 'U'] + ['P', 'R'] → None
- ['-Q', 'U'] + ['P', 'S'] → None
```

```
k ['-Q', 'U'] + ['Q', 'U'] → ['U']
- ['-Q', 'U'] + ['-P'] → None
- ['-Q', 'U'] + ['-R'] → None
- ['-Q', 'U'] + ['P', 'U'] → None
k ['-Q', 'U'] + ['Q'] → ['U']
- ['-Q', 'U'] + ['R', 'U'] → None
- ['-Q', 'U'] + ['-Q'] → None
- ['-Q', 'U'] + ['S', 'U'] → None
- ['-Q', 'U'] + ['R'] → None
- ['-Q', 'U'] + ['P'] → None
- ['-Q', 'U'] + ['S'] → None
- ['-Q', 'U'] + ['U'] → None
k ['-P'] + ['P', 'Q'] → ['Q']
- ['-P'] + ['-Q', 'R'] → None
- ['-P'] + ['-Q', 'S'] → None
- ['-P'] + ['-P', 'U'] → None
- ['-P'] + ['-R', 'U'] → None
- ['-P'] + ['-U'] → None
k ['-P'] + ['P', 'R'] → ['R']
k ['-P'] + ['P', 'S'] → ['S']
- ['-P'] + ['Q', 'U'] → None
- ['-P'] + ['-Q', 'U'] → None
- ['-P'] + ['-R'] → None
k ['-P'] + ['P', 'U'] → ['U']
- ['-P'] + ['Q'] → None
- ['-P'] + ['R', 'U'] → None
- ['-P'] + ['-Q'] → None
- ['-P'] + ['S', 'U'] → None
- ['-P'] + ['R'] → None
y ['-P'] + ['P'] → []
- ['-P'] + ['S'] → None
- ['-P'] + ['U'] → None
- ['-R'] + ['P', 'Q'] → None
k ['-R'] + ['-Q', 'R'] → ['-Q']
- ['-R'] + ['-Q', 'S'] → None
- ['-R'] + ['-P', 'U'] → None
- ['-R'] + ['-R', 'U'] → None
- ['-R'] + ['-U'] → None
k ['-R'] + ['P', 'R'] → ['P']
- ['-R'] + ['P', 'S'] → None
- ['-R'] + ['Q', 'U'] → None
- ['-R'] + ['-Q', 'U'] → None
- ['-R'] + ['-P'] → None
- ['-R'] + ['P', 'U'] → None
- ['-R'] + ['Q'] → None
k ['-R'] + ['R', 'U'] → ['U']
- ['-R'] + ['-Q'] → None
- ['-R'] + ['S', 'U'] → None
y ['-R'] + ['R'] → []
- ['-R'] + ['P'] → None
- ['-R'] + ['S'] → None
- ['-R'] + ['U'] → None
- ['P', 'U'] + ['P', 'Q'] → None
- ['P', 'U'] + ['-Q', 'R'] → None
- ['P', 'U'] + ['-Q', 'S'] → None
```

```
k ['P', 'U'] + ['-P', 'U'] → ['U']
- ['P', 'U'] + ['-R', 'U'] → None
k ['P', 'U'] + ['-U'] → ['P']
- ['P', 'U'] + ['P', 'R'] → None
- ['P', 'U'] + ['P', 'S'] → None
- ['P', 'U'] + ['Q', 'U'] → None
- ['P', 'U'] + ['-Q', 'U'] → None
k ['P', 'U'] + ['-P'] → ['U']
- ['P', 'U'] + ['-R'] → None
- ['P', 'U'] + ['Q'] → None
- ['P', 'U'] + ['R', 'U'] → None
- ['P', 'U'] + ['-Q'] → None
- ['P', 'U'] + ['S', 'U'] → None
- ['P', 'U'] + ['R'] → None
- ['P', 'U'] + ['P'] → None
- ['P', 'U'] + ['S'] → None
- ['P', 'U'] + ['U'] → None
- ['Q'] + ['P', 'Q'] → None
k ['Q'] + ['-Q', 'R'] → ['R']
k ['Q'] + ['-Q', 'S'] → ['S']
- ['Q'] + ['-P', 'U'] → None
- ['Q'] + ['-R', 'U'] → None
- ['Q'] + ['-U'] → None
- ['Q'] + ['P', 'R'] → None
- ['Q'] + ['P', 'S'] → None
- ['Q'] + ['Q', 'U'] → None
k ['Q'] + ['-Q', 'U'] → ['U']
- ['Q'] + ['-P'] → None
- ['Q'] + ['-R'] → None
- ['Q'] + ['P', 'U'] → None
- ['Q'] + ['R', 'U'] → None
y ['Q'] + ['-Q'] → []
- ['Q'] + ['S', 'U'] → None
- ['Q'] + ['R'] → None
- ['Q'] + ['P'] → None
- ['Q'] + ['S'] → None
- ['Q'] + ['U'] → None
- ['R', 'U'] + ['P', 'Q'] → None
- ['R', 'U'] + ['-Q', 'R'] → None
- ['R', 'U'] + ['-Q', 'S'] → None
- ['R', 'U'] + ['-P', 'U'] → None
k ['R', 'U'] + ['-R', 'U'] → ['U']
k ['R', 'U'] + ['-U'] → ['R']
- ['R', 'U'] + ['P', 'R'] → None
- ['R', 'U'] + ['P', 'S'] → None
- ['R', 'U'] + ['Q', 'U'] → None
- ['R', 'U'] + ['-Q', 'U'] → None
- ['R', 'U'] + ['-P'] → None
k ['R', 'U'] + ['-R'] → ['U']
- ['R', 'U'] + ['P', 'U'] → None
- ['R', 'U'] + ['Q'] → None
- ['R', 'U'] + ['-Q'] → None
- ['R', 'U'] + ['S', 'U'] → None
- ['R', 'U'] + ['R'] → None
- ['R', 'U'] + ['P'] → None
```

```
- ['R', 'U'] + ['S'] → None
- ['R', 'U'] + ['U'] → None
k ['-Q'] + ['P', 'Q'] → ['P']
- ['-Q'] + ['-Q', 'R'] → None
- ['-Q'] + ['-Q', 'S'] → None
- ['-Q'] + ['-P', 'U'] → None
- ['-Q'] + ['-R', 'U'] → None
- ['-Q'] + ['-U'] → None
- ['-Q'] + ['P', 'R'] → None
- ['-Q'] + ['P', 'S'] → None
k ['-Q'] + ['Q', 'U'] → ['U']
- ['-Q'] + ['-Q', 'U'] → None
- ['-Q'] + ['-P'] → None
- ['-Q'] + ['-R'] → None
- ['-Q'] + ['P', 'U'] → None
y ['-Q'] + ['Q'] → []
- ['-Q'] + ['R', 'U'] → None
- ['-Q'] + ['S', 'U'] → None
- ['-Q'] + ['R'] → None
- ['-Q'] + ['P'] → None
- ['-Q'] + ['S'] → None
- ['-Q'] + ['U'] → None
- ['S', 'U'] + ['P', 'Q'] → None
- ['S', 'U'] + ['-Q', 'R'] → None
- ['S', 'U'] + ['-Q', 'S'] → None
- ['S', 'U'] + ['-P', 'U'] → None
- ['S', 'U'] + ['-R', 'U'] → None
k ['S', 'U'] + ['-U'] → ['S']
- ['S', 'U'] + ['P', 'R'] → None
- ['S', 'U'] + ['P', 'S'] → None
- ['S', 'U'] + ['Q', 'U'] → None
- ['S', 'U'] + ['-Q', 'U'] → None
- ['S', 'U'] + ['-P'] → None
- ['S', 'U'] + ['-R'] → None
- ['S', 'U'] + ['P', 'U'] → None
- ['S', 'U'] + ['Q'] → None
- ['S', 'U'] + ['R', 'U'] → None
- ['S', 'U'] + ['-Q'] → None
- ['S', 'U'] + ['R'] → None
- ['S', 'U'] + ['P'] → None
- ['S', 'U'] + ['S'] → None
- ['S', 'U'] + ['U'] → None
- ['R'] + ['P', 'Q'] → None
- ['R'] + ['-Q', 'R'] → None
- ['R'] + ['-Q', 'S'] → None
- ['R'] + ['-P', 'U'] → None
k ['R'] + ['-R', 'U'] → ['U']
- ['R'] + ['-U'] → None
- ['R'] + ['P', 'R'] → None
- ['R'] + ['P', 'S'] → None
- ['R'] + ['Q', 'U'] → None
- ['R'] + ['-Q', 'U'] → None
- ['R'] + ['-P'] → None
y ['R'] + ['-R'] → []
- ['R'] + ['P', 'U'] → None
```

```
- ['R'] + ['Q'] → None
- ['R'] + ['R', 'U'] → None
- ['R'] + ['-Q'] → None
- ['R'] + ['S', 'U'] → None
- ['R'] + ['P'] → None
- ['R'] + ['S'] → None
- ['R'] + ['U'] → None
- ['P'] + ['P', 'Q'] → None
- ['P'] + ['-Q', 'R'] → None
- ['P'] + ['-Q', 'S'] → None
k ['P'] + ['-P', 'U'] → ['U']
- ['P'] + ['-R', 'U'] → None
- ['P'] + ['-U'] → None
- ['P'] + ['P', 'R'] → None
- ['P'] + ['P', 'S'] → None
- ['P'] + ['Q', 'U'] → None
- ['P'] + ['-Q', 'U'] → None
y ['P'] + ['-P'] → []
- ['P'] + ['-R'] → None
- ['P'] + ['P', 'U'] → None
- ['P'] + ['Q'] → None
- ['P'] + ['R', 'U'] → None
- ['P'] + ['-Q'] → None
- ['P'] + ['S', 'U'] → None
- ['P'] + ['R'] → None
- ['P'] + ['S'] → None
- ['P'] + ['U'] → None
- ['S'] + ['P', 'Q'] → None
- ['S'] + ['-Q', 'R'] → None
- ['S'] + ['-Q', 'S'] → None
- ['S'] + ['-P', 'U'] → None
- ['S'] + ['-R', 'U'] → None
- ['S'] + ['-U'] → None
- ['S'] + ['P', 'R'] → None
- ['S'] + ['P', 'S'] → None
- ['S'] + ['Q', 'U'] → None
- ['S'] + ['-Q', 'U'] → None
- ['S'] + ['-P'] → None
- ['S'] + ['-R'] → None
- ['S'] + ['P', 'U'] → None
- ['S'] + ['Q'] → None
- ['S'] + ['R', 'U'] → None
- ['S'] + ['-Q'] → None
- ['S'] + ['S', 'U'] → None
- ['S'] + ['R'] → None
- ['S'] + ['P'] → None
- ['S'] + ['U'] → None
- ['U'] + ['P', 'Q'] → None
- ['U'] + ['-Q', 'R'] → None
- ['U'] + ['-Q', 'S'] → None
- ['U'] + ['-P', 'U'] → None
- ['U'] + ['-R', 'U'] → None
y ['U'] + ['-U'] → []
- ['U'] + ['P', 'R'] → None
- ['U'] + ['P', 'S'] → None
```

```
- ['U'] + ['Q', 'U'] → None
- ['U'] + ['-Q', 'U'] → None
- ['U'] + ['-P'] → None
- ['U'] + ['-R'] → None
- ['U'] + ['P', 'U'] → None
- ['U'] + ['Q'] → None
- ['U'] + ['R', 'U'] → None
- ['U'] + ['-Q'] → None
- ['U'] + ['S', 'U'] → None
- ['U'] + ['R'] → None
- ['U'] + ['P'] → None
- ['U'] + ['S'] → None
yield length = 1
0: []
YES
```

Personal reflection: as evidenced by the output log, there is currently no data structure or method of keeping track of which pairs of clauses have previously been put through the inference rule, which means everything will be calculated again from the beginning of each iteration. This is considered waste of computation, and optimization in this aspect can be one of the first steps towards better runtime.

References

- Slides and lecture notes on Propositional Logic
- Wolfram|Alpha (for checking $\neg\alpha$ conversion and whether the algorithm ran correctly)
- This answer on StackOverflow (for `itertools.groupby` – used in `neg_alpha()` for quickly eliminating duplicate clauses)

---

1. We're on a tight time budget and we can't figure out how to properly codify the CNF conversion. It's sad `:(`. ↩