### Setting

Compute a solution for a constraint system

\[ x_i \equiv f_i(x_1, ..., x_n) \]

where \( x_i \) are unknowns (points of interest) and \( f_i \) are the constraints to be satisfied.

Used for example for static program analysis:

```c
int factorial(int n) {
    int r = 1;
    int i = n;
    while(i > 0) {
        r *= i;
        i--; 
    }
    return r;
}
```

### Motivation

Advanced solving strategies complicate reasoning about solver correctness. Implementations are often fragile and vulnerable to bugs.

\( \rightarrow \) Formal verification to justify the correctness

### Top-Down Solver

- **Generic** fixpoint algorithm
- Computes partial solution for the queried unknown of interest and all unknowns it depends on
- Tracks dependencies between unknowns on-the-fly
- An improved version [3] is used by the static analyzer GOBLINT (\( \text{G} \)oblint/\( \text{A} \)nanalyzer)

#### References

