CAD Programming Assignment 1
Boolean Function Minimization using Sum-of-Products

Michael Hung

September 2025

0. Objective

You are required to **write a C or C++ program** that reads a **specification file** describing a Boolean function (bit width, on-set, don't-care set) and outputs a **Sum-of-Products (SOP) file** (each line an implicant). Your submission will be validated by the provided `checker.py` program.

1. Deliverables

- Source code in C or C++ (`.c/.h` or `.cpp/.hpp`).
- A **Makefile** that builds an executable named `sop` (a sample `Makefile` will be provided for reference; you may modify it as needed; it does not need to be identical).
- The executable `sop` must generate an SOP file that passes the checker.

Submission Instructions

Please put all required files into a folder named `student_ID_PA1` (where `student_ID` should be replaced with your actual student ID). Then compress it into a `.tar` archive before submission:

```
tar cvf student_ID_PA1.tar student_ID_PA1
```

2. Program Interface

```
./sop <spec_file> <out_sop_file>
```

- Input: `<spec_file>` in the format defined below.
- Output: write a valid SOP to `<out_sop_file>`.
- Exit code 0 = success; non-zero = error.

3. File Formats

3.1 Specification File

Exactly three lines:

1. `n_bit`: $24 \geq$ integer ≥ 0.
2. `on_set`: space-separated decimal minterms ($f = 1$).
3. `dc_set`: space-separated decimal don't care minterms.

Example:

```
3
1 2 5 7
0 4 6
```

3.2 SOP Output File

- Each line = one implicant, string of length `n_bit`, chars $\in \{0, 1, -\}$.
- Bit order: MSB to LSB (leftmost = MSB).
- No extra spaces or blank lines; if n_bit = 0, output one empty line.

4. Validation Criteria (Checker Rules)

The SOP your program produces must satisfy the following:

1. All on-set minterms are covered.
2. No off-set minterms are covered.
3. No duplicate implicants.
4. Literal count $<$ |on-set| $\times$ *n_bit*.

5. Building and Running

5.1 Compilation

make

5.2 Generate SOP and Run Checker

```
./sop spec.txt out.sop
python3 checker.py spec.txt out.sop
```

6. Makefile Requirements

- Target executable: `sop`.
- Default target: `all`.
- Must support `make clean`.
- Must be able to build on the course server.

7. Example

Specification:

3
1 2 5 7
0 4 6

Valid SOP produced:

−01
1−1
010

8. Notes

- Programs may implement Quine–McCluskey, Karnaugh map simplification, or heuristics (all test cases will be huge, so that global optimization cannot produce a solution in a reasonable time duration).
- **Important:** Simply outputting each on-set minterm as an implicant (one minterm → one product term of length n_bit) is *not* a valid solution. This construction yields exactly $|\text{on-set}| \times n\_bit$ literals, which does not satisfy the strict inequality required by Rule 4 of the checker.
- **Supplement:** We will ensure that no test case is completely non-reducible.

9. Grading Policy

- There are 5 test cases in total: 2 public and 3 hidden.
- Each case is worth 20 points.
- If your SOP passes the checker and has fewer literals than the naive baseline, you receive 15 points for that case.
- The remaining 5 points per case are based on performance: fewer literals → higher score; tie is broken by faster run-time.
  - Suppose there are N valid submissions for a test case.
  - Let $\text{rank}(i)$ denote the ranking of submission i (1 = best, N = worst).
  - The performance score S_i for submission i is

$$S_i = \frac{N - \text{rank}(i) + 1}{N} \times 5.$$

  - Thus, the best submission (rank = 1) receives 5 points, and the worst submission (rank = N) still receives $\frac{5}{N}$ points.
- **Time limit:** Each program has a maximum run time of 3 minutes per case.
- If **plagiarism** or **LLM-generated code** are detected, you will receive **a score of 0**.
- **Third-party solver is prohibited**.