Università della Svizzera italiana
Faculty of Informatics

Theory of Computation **2025**

Students: Edoardo Ababei, Mykhailo Bozhko, Davide Frova, Mehmet Fatih Tekin

---

Group D — Fashion Store Problem

The Fashion Store problem is solved using Boolean Satisfiability (SAT). This report details the comprehensive SAT encoding, implementation, user interface, experiments, and results. Constraints including garment selection, color harmony, layering rules, seasonal appropriateness, and user style preferences are systematically encoded. Python and the Z3 solver demonstrate the practicality and efficiency of SAT for combinational problems.

1. Introduction

The Fashion Store problem involves selecting garments and colors based on specific constraints to create valid outfit combinations. The objective is to find combinations using a SAT solver that adhere to predefined aesthetic and practical guidelines.

2. Encoding into SAT

1. Variables

Each garment-color pair is represented as a Boolean variable $x_{g,c}$, indicating the selection state. Variables are mapped to integers for SAT solver compatibility.

2. Constraints

Detailed constraints include:

- **Outfit Size Constraints**: Between 3 and 6 garments.
- **Type Coverage**: At least one garment per category (hat, coat, top, bottom, shoes, gloves).
- **Palette Size Constraints**: Between 2 and 4 distinct colors.
- **Color Clashes**: Certain combinations like red and pink are forbidden.
- **Complement Harmony**: Inclusion of cool colors when warm colors are selected.
- **Layering Order**: Logical layering rules, e.g., coats require tops underneath.
- **One-Per-Body-Part**: Each garment category can have only one selected garment.
- **Season/Context (Winter)**: Must include either coat or gloves.
- **Style Preferences (Soft Constraints)**: Preferences like black gloves over white gloves are encoded as soft constraints.

3. CNF Conversion

Constraints are converted into Conjunctive Normal Form (CNF). For instance, color clash constraints:

$$(\neg x_{\text{hat,red}} \vee \neg x_{\text{top,pink}})$$

3. Implementation

1. Software and Libraries

The Z3 solver was chosen for its efficiency and versatility. Python scripts handle CNF generation, input parsing, and solver execution. Flask and Bootstrap are employed for the user-friendly web interface.

2. Solver Integration

The implementation pipeline includes:

1. Parsing user input (file or direct input).
2. Generating CNF clauses.
3. Solving with Z3 solver.
4. Parsing solver output to user-friendly results.

4. User Interface

A Flask-based GUI provides intuitive interactions:

- File upload and direct input options.
- Quick test buttons (SAT/UNSAT).
- Interactive constraint exploration.
- Readable solver outputs displayed clearly.

5. Experiments and Results

1. Testing Methodology

Comprehensive testing includes predefined SAT and UNSAT scenarios designed to rigorously validate constraints.

2. Results

Results showed the solver quickly and accurately determining garment selections with very low solving times, typically milliseconds.

6. Challenges and Alternatives

Challenges included complex constraint encodings and GUI responsiveness. Alternative encodings and different solvers were considered but ultimately Z3 proved optimal.

7. Conclusion

SAT-based solutions effectively addressed the Fashion Store problem, with a robust solver and user-friendly interface. Future improvements include more dynamic soft constraint management and extended customization.

References

- Z3 Solver Documentation, Microsoft Research
- Flask Framework Documentation
- Bootstrap Frontend Toolkit Documentation