WMC via Knowledge compilation

Sharare Zolghadr - 2072251
Professors Serafini and Confalonieri
Knowledge Representation and Learning

Objective & Goal

- **Objective:** Implement and evaluate methods to compute Weighted Model Counting (WMC) for propositional formulas.

- **Goal:** Compare exact WMC methods (knowledge compilation & truth tables) with approximate methods (SampleSAT). Integrate formula generation with Google Gemini API for diverse test cases.

Project Pipeline

- **Input:** Propositional formula
- **Transformation:**
  - Convert to sd-DNNF via Shannon expansion and smoothing
  - Also transform to CNF for SAT solver validation
- **WMC Computation Methods:**
  - sd-DNNF-based WMC(exact counting)
  - Truth table enumeration(exact counting)
  - Sample SAT algorithm(approximate counting)
- **Validation:**
  - Use PySAT to generate models from CNF
  - Sum their weights and compare against own methods
- **Generation:** Generate test formulas using Google Gemini API

Key Concepts

- **NNF → DNNF → sd-DNNF:** Forms enabling tractable WMC
- **WMC:** Sum of weighted models that satisfy the formula
- **SampleSAT:** Sampling-based approximate model counter
- **SAT Solver Validation:** Compare against CNF model weights using PySAT

Definitions

- **NNF**: Only $\wedge$, $\vee$, $\neg$ ($\neg$ only on atoms)
- **DNNF**: NNF + conjuncts are decomposable
- **d-DNNF**: DNNF + disjuncts are mutually exclusive
- **sd-DNNF**: d-DNNF + disjunctions are *smooth (Same variables across branches)*

sd-DNNF Conversion

- **Shannon Expansion**: $\varphi \equiv (p \wedge \varphi \mid p) \vee (\neg p \wedge \varphi \mid \neg p)$

  Ensures **decomposability** & **determinism**

- **Smoothing**:
  Add dummy vars like ($p \vee \neg p$) to balance disjuncts

WMC

Weighted Model Counting: Sum weights of all satisfying models of formula (φ) :

$$wmc(\varphi) = \sum_{I \models \varphi} w(I)$$

In sd-DNNF:

- **AND node** → product
- **OR node** → sum

Based on structure, not full truth enumeration

SAT Solver

- **CNF Transformation** → use with standard SAT solvers
- We use **PyEDA / PySAT** to:
  - Generate **all models** of the CNF formula
  - Assign **truth values** to variables
- For each satisfying assignment:
  - Compute its **weight**
  - **Sum** all model weights to get exact WMC

Example Outcome

Input Formula:
$(A \rightarrow B) \land (B \rightarrow (C \lor D))$

Preprocessed (NNF):
$(\neg A \lor B) \land (\neg B \lor (C \lor D))$

Weights:
A: 0.25, B: 0.39, C: 0.77, D: 0.59

WMC Results:

- **sd-DNNF**: 0.810723
- **Truth Table**: 0.810723
- **SampleSAT**: ~0.8105
- **PySAT (CNF)**: 0.810723

Compiled sd-DNNF *(simplified)*:
$((B \land ((C \land (D \lor \neg D)) \lor (\neg C \land D))) \land (A \lor \neg A)) \lor ((\neg B \land \neg A) \land (C \lor \neg C) \land (D \lor \neg D))$

Results & Validation

- Exact WMC (sd-DNNF) and truth table: **Consistent results**
- Approximate WMC (SampleSAT): **Close estimates**
- CNF + PySAT: **Matched exact method**, validating correctness
- Gemini API: Provided **diverse test formulas**

Conclusion

- Developed and validated WMC computation methods
- Verified correctness using CNF + PySAT model enumeration
- Next steps:
    - Improve approximation methods
    - Enhance Gemini-based test coverage
    - Explore other SAT solving backends

References

- PySAT Toolkit
- SampleSAT Algorithm
- Gemini API
- DNNF/Knowledge Compilation Literature

Demo