Supplemental Proof and Cactus Plot

Anonymous submission

Algorithm 1: PModelEnum

Input: formula ϕ, cutoff co
Output: if enumeration succeeded or not, models M
Ω := one-direction Tseitin translation of NNF(ϕ)
Ω_{PI} := PI translation (Jabbour et al. 2014) of Ω
(isOK, M) := **ModelEnumerator**(Ω_{PI}, co)
M' := $\{decode(m) \mid m \in M\}$
return (isOK, M')

Overview

This is a supplemental report for the proof of the properties in Indirect translation. We also include the cactus plot for all instances, which we could not include in the original paper.

Supplemental Proof for Indirect Translation

In this section, we provide a proof that the two formulas discussed in the previous section are logically equivalent:

$$\Phi_1 = \bigvee_{m \in \mathcal{M}(\phi)} \bigwedge_{l \in m} l$$

and

$$\Phi_2 = \bigvee_{m \in \textbf{PModelEnum}(\phi, \infty)} \bigwedge_{l \in m} l$$

The difference between two fomulas is $\mathcal{M}$ and **PModelEnum**, where $\mathcal{M}$ is a function that translates a Boolean formula into the set of all its (complete) models, and **PModelEnum** is the procedure described in Algorithm 1.

To provide a proof, we start with the two properties of the Tseitin translation in one direction.

Properties of One-direction Tseitin Translation

Let $A[B]$ be a source negation normal form (NNF) formula containing a subformula B. Note that an arbitrary formula can be converted to the NNF formula by using De Morgan's law. Let p be an auxiliary variable that does not appear in $A[B]$. Let A be an NNF formula such that A is obtained by all the appearances of B with the positive literal p. Then, a one-directional Tseitin translation of $A[B]$ for B can be performed by $A \wedge (\neg p \vee B)$. We clarify its properties with the following two propositions, which is based on the proof for equi-satisfiability in the literature (Tamura, Tanjo, and Banbara 2010).

Proposition 1. Suppose that α is a partial model of $A \wedge (\neg p \vee B)$, and α' is a partial model obtained by excluding p from α. Then $\alpha'(A[B]) = 1$ holds.

Proof. By definition, $\alpha(A \wedge (\neg p \vee B)) = 1$ holds. Then $\alpha(A) = 1$ and $\alpha(\neg p \vee B) = 1$ hold. We prove the given proposition by structural induction. We use an arbitrary NNF formula C that does not contain $\neg p$ instead of A to ease the explanation. If C is the variable p, then $C[B] \leftrightarrow B$ holds. Since $\alpha(p) = 1$ and $\alpha(\neg p \vee B) = 1$, $\alpha(B) = 1$. Since B does not contain p, $\alpha'(B) = 1$ hold then $\alpha'(C[B]) = 1$. If C is a variable q ($q \neq p$) or $\neg q$, then $C[B] \leftrightarrow C$ holds. Since $\alpha(C) = 1$, $\alpha(C[B]) = 1$ holds. If C is $C_1 \wedge C_2$, then $\alpha(C_1) = 1$ and $\alpha(C_2) = 1$ since $\alpha(C) = 1$. By the induction hypothesis, $\alpha'(C_1[B]) = 1$ and $\alpha'(C_2[B]) = 1$, hence $\alpha'(C[B]) = 1$ holds. If C is $C_1 \vee C_2$, the proof is done in the same way as the case of $C_1 \wedge C_2$. Therefore, $\alpha'(A[B]) = 1$ holds. $\square$

Proposition 2. Let α be a model such that $\alpha(A[B]) = 1$. Let α' be a model that extends α to the domain that includes p such that $\alpha'(p) = \alpha(B)$. Then $\alpha'(A \wedge (\neg p \vee B)) = 1$ holds.

Proof. By definition, $\alpha'(A) = 1$. Since $\alpha'(p) = \alpha(B)$ holds, $\alpha'(\neg p \vee B) = 1$ holds. Therefore, $\alpha'(A \wedge (\neg p \vee B)) = 1$ holds. $\square$

Properties of PModelEnum

From here, we will also represent an assignment α as a set of literals for simplicity. For instance, suppose that ϕ is a formula over variables p, q, r. Then, an assignment $p = 1, q = 1, r = 0$ will be represented as $\{p, q, \neg r\}$. A partial assignment where $p = 1, q = 1$ will be represented as $\{p, q\}$. In addition, $\alpha(\phi) = 1$ means that any assignment satisfies $\bigwedge_{l \in \alpha} l$ also satisfies ϕ. The following proposition holds between $\mathcal{M}(\phi)$ and **PModelENum**(ϕ, ∞).

Proposition 3. Given an arbitrary formula ϕ, for any $\alpha \in \mathcal{M}(\phi)$, there exists a partial model $\alpha' \in$ **PModelEnum**(ϕ, ∞) such that $\alpha' \subseteq \alpha$ and $\alpha'(\phi) = 1$.

Proof. We prove it by contradiction. Assume that for some $\alpha \in \mathcal{M}(\phi)$, there does not exist a partial model $\alpha' \in$

PModelEnum(ϕ, ∞) such that $\alpha' \subseteq \alpha$ and $\alpha'(\phi) = 1$. For such α, by Proposition 2, there exists a model β (which is α extended with auxiliary variables) that is a model of $1TT(\phi)$. Then, let β' be a prime implicant of $1TT(\phi)$. Obviously, $\beta' \subseteq \beta$ and $\beta'(1TT(\phi)) = 1$ hold. For such β', by Proposition 1, there exists a model α' that satisfies $\alpha'(\phi) = 1$. Clearly, $\alpha' \subseteq \alpha$ holds, which contradicts the assumption. Therefore, for any $\alpha \in \mathcal{M}(\phi)$, there exists a partial model $\alpha' \in$ **PModelEnum**(ϕ, ∞) such that $\alpha' \subseteq \alpha$ and $\alpha'(\phi) = 1$. □

Then, we get the following proposition that gives the logical equivalence between Φ_1 and Φ_2.

Proposition 4. Given an arbitrary formula ϕ, any model of Φ_1 is also a model of Φ_2 and vice versa.

Proof. ($\Rightarrow$) According to Proposition 3, for any $\alpha \in \mathcal{M}(\phi)$ there exists $\alpha' \in$ **PModelEnum**(ϕ, ∞) such that $\alpha' \subseteq \alpha$. Thus, if Φ_1 has a satisfied disjunct, then Φ_2 also has a satisfied disjunct. Therefore, any model of Φ_1 is also a model of Φ_2. ($\Leftarrow$) According to Proposition 3, $\alpha'(\phi) = 1$. Therefore, there exists a complete model $\alpha \in \mathcal{M}(\phi)$, which is obtained by extending α' to a complete model. Therefore, any model of Φ_2 is also a model of Φ_1. □

Cactus Plot for All Instances

Figure 1 shows the CPU time profiles ("cactus plots") for all instances. In this plot, the y-axis shows the CPU time limit for each instance on a log scale, and the x-axis shows the number of instances solved within the time limit. D.C., I.C., and H.C. were able to count the number of singleton attractors for more than 300 instances, while others could not within 1800 seconds. Since our methods are implemented in Scala, which introduces a runtime overhead of about 1 second due to the Java Virtual Machine (JVM), this results in slower performance, especially for the first roughly 230 simpler instances. For some instances, I.C. and H.C. took a few seconds longer than D.C. because of the model enumeration process, but overall, I.C. and H.C. performed better among the proposed methods. D.C., I.C., and H.C. solved 313, 328, and 330 instances in total, respectively. Among the existing methods, AEON performed the best, with a total of 244 instances solved.

References

Jabbour, S.; Marques-Silva, J.; Sais, L.; and Salhi, Y. 2014. Enumerating Prime Implicants of Propositional Formulae in Conjunctive Normal Form. In Fermé, E.; and Leite, J., eds., *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings*, volume 8761 of *Lecture Notes in Computer Science*, 152–165. Springer.

Tamura, N.; Tanjo, T.; and Banbara, M. 2010. Constraint Optimization Problems and SAT Encodings. , 25(1): 77–85.

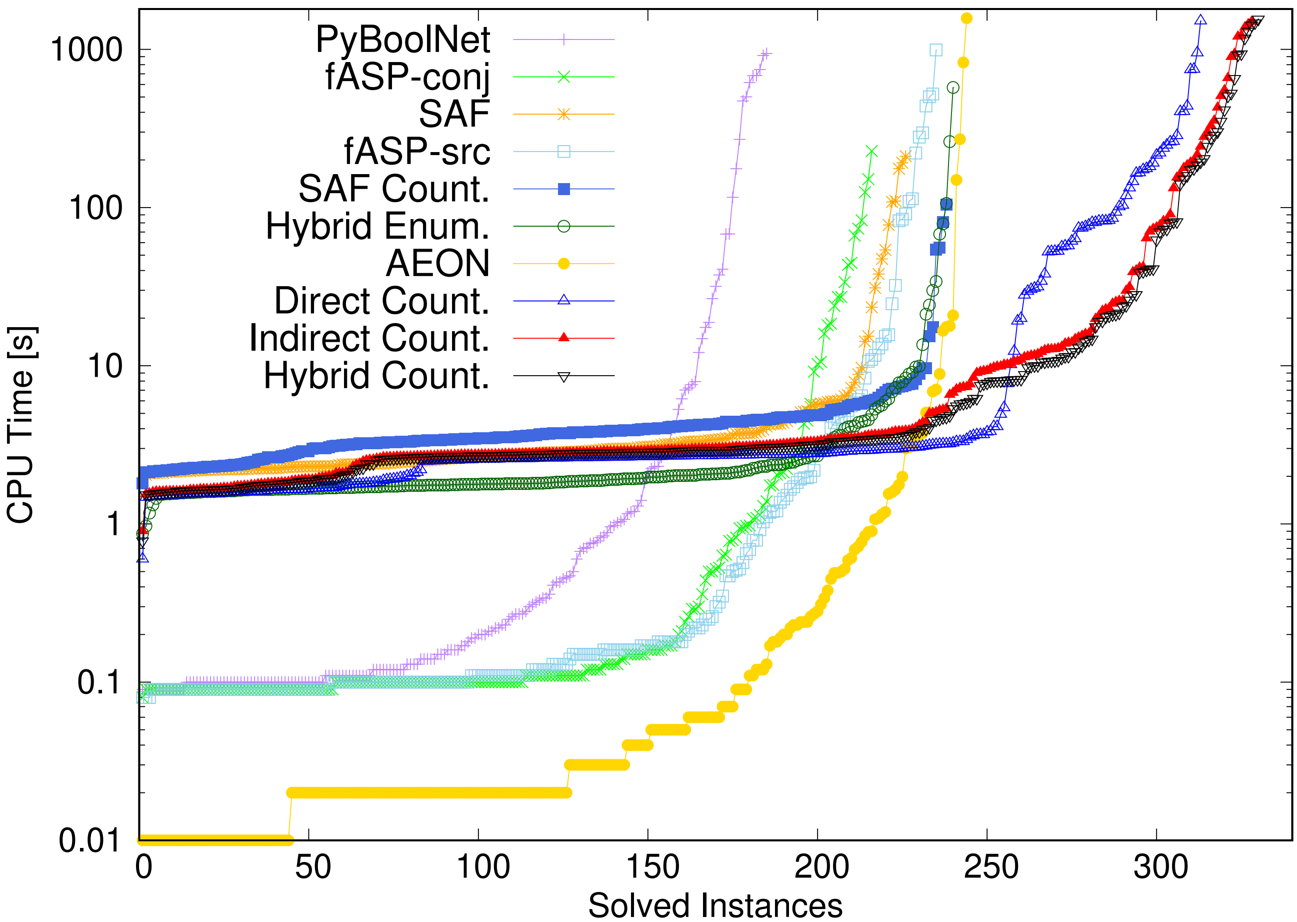

Figure 1: CPU time (log-scale) profiles for all 643 instances