MergeSAT

NAJIB ISHAQ

Abstract will be written last.

CCS Concepts: •**Theory of computation** → **Logic and Verification**.

Additional Key Words and Phrases: P vs NP

ACM Reference Format:
Najib Ishaq. 2021. MergeSAT. 1, 1 (June 2021), 4 pages. https://doi.org/XXX

1 INTRODUCTION

Introductions are in order [1].

2 DEFINITIONS

We set up the problem with some definitions.

Definition 2.1 (Instance, Clause, Literal, Variable). A SAT instance is a conjunction of clauses. Each clause is a disjunction of literals. Each literal is either a variable or the negation of a variable. A SAT instance of size (k, m, n) contains m variables and n clauses, and each clause contains at most k literals.

Definition 2.2 (Unit Clause). A unit clause is a clause with only one literal. Such a clause can always be satisfied by setting that literal to $True$.

Definition 2.3 (Pure Literal). A pure literal is one whose negation never appears in the instance. All clauses containing a pure literal can be satisfied by setting that literal to $True$.

Definition 2.4 (Cascade). A cascade is an assignment made due to unit clauses and pure literals. A cascade can cause more cascades, so several cascades can be chained into a single cascade.

Definition 2.5 (Certificate). A certificate (χ) is a set of mappings from each variable x_i to a truth value such that the set of assignments satisfies a given instance.

$$\chi : \ x_i \mapsto \{True, \ False\} \ \forall i \in \{1 \dots m\} \mid \Phi(\chi) = True$$

Definition 2.6 (Satisfiability). If an instance has at least one certificate then the instance is *satisfiable*. Otherwise, the instance is *unsatisfiable*.

$$\Phi \ is \ satisfiable \iff \exists \ \chi \mid \Phi(\chi) = True$$
$$\Phi \ is \ unsatisfiable \iff \neg\exists \ \chi \mid \Phi(\chi) = True$$

If after assigning truth values to some, but not all, variables and evaluating the instance we find that there are some clauses left whose truth value is ambiguous, we say that the instance is, as yet, *undecided*.

Author's address: Najib Ishaq, najib_ishaq@zoho.com.

XXXX-XXXX/2021/6-ART
https://doi.org/XXX

Note that a certificate may not need to specify a mapping for every variable in an instance. Some instances can be satisfied by assigning a mapping for some, but not all, of the variables in that instance.

Definition 2.7 (Min-Certificate). A min-certificate is a special case of a certificate with the property that removing even one mapping from the certificate leaves the instance undecided.

For example:

$$\phi = (x_1 \lor \neg x_2 \lor x_3) \land (\neg x_2 \land \neg x_3 \land x_4)$$

can be satisfied by $\chi = \{x_2 \mapsto False\}$. This leaves three variables unmapped, which leads to 2^3 full certificates. χ is a min-certificate for the SAT instance Φ.

There are other possible min-certificates for this SAT instance. For example:

- $\{x_1 \mapsto True,\ x_3 \mapsto False\}$ which leads to 2^2 full certificates.
- $\{x_3 \mapsto True,\ x_4 \mapsto True\}$ which leads to 2^2 full certificates.

For the remainder of this paper, the term certificate will be overloaded with this definition of a min-certificate.

Definition 2.8 (Child Instances). Given an instance Φ with more than one clause, we define two instances, ϕ_{left} and ϕ_{right} such that every clause from Φ in either in ϕ_{left} or ϕ_{right} but not in both.

$$\phi_{left},\ \phi_{right} \subset \Phi \mid \Phi = \phi_{left} \land \phi_{right},\ \phi_{left} \cap \phi_{right} = \emptyset$$

ϕ_{left} and ϕ_{right} are child instances with Φ as the parent instance. ϕ_{left} and ϕ_{right} are siblings to each other.

Definition 2.9 (Child Certificates). If Φ is satisfiable then so are its child instances ϕ_{left} and ϕ_{right}. In other words, if Φ has a certificate then each child instance has at least one certificate. We call the certificates for child instances child certificates.

Definition 2.10 (Compatibility). Given two instances, ϕ_1 and ϕ_2, and their respective certificates, χ_1 and χ_2, we define compatibility in the following way.

$$\begin{aligned}
&\chi_1 \textit{ is compatible with } \chi_2 \\
&\Leftrightarrow \forall x_j \in \chi_1 \cap \chi_2, \chi_1(x_j) \equiv \chi_2(x_j) \\
&\textit{and } \phi_1(\chi_2) \textit{ is either True or Undecided} \\
&\textit{and } \phi_2(\chi_1) \textit{ is either True or Undecided} \\
&\textit{and } \Phi(\chi_1 \cup \chi_2) \textit{ is True, where } \Phi = \phi_1 \land \phi_2
\end{aligned}$$

Definition 2.11 (Pure Instance). A pure instance, ϕ_{pure}, is a special case of an instance in which all clauses have a variable in common.

$$\begin{aligned}
&let\ \phi_{base} \coloneqq C_1 \land C_2 \land \ldots \land C_m \\
&s.t.\ either\ x_j \in C_i\ or\ \neg x_j \in C_i \\
&for\ some\ j \in \{1 \ldots n\} \\
&for\ all\ i \in \{1 \ldots m\}
\end{aligned}$$

3 THE SOLVER

Let's first define some sub-problems and consider how to solve each of them. Then we will put them together into a complete solver.

3.1 Partitioning an Instance

Given an instance Φ with variables $\{x_1 \ldots x_n\}$, we can partition Φ into two child instances about the variable x_i by whether the clauses in Φ contain the variable x_i. Without loss of generality, we can have ϕ_{left} be the conjunction of clauses that each **contain** the variable x_1, and ϕ_{right} be the conjunction of clauses that each **do not contain** the variable x_1.

Note that ϕ_{left} is a pure instance and ϕ_{right} is a smaller instance of the same form as Φ. ϕ_{left} can be further partitioned into two instances, $\phi_{left,1}$ and $\phi_{left,2}$, such that x_1 is a pure literal in $\phi_{left,1}$ and $\neg x_1$ is a pure literal in $\phi_{left,2}$. ϕ_{left_1} and $\phi_{left,2}$ are of sizes $(k_l,\ m_l,\ n_l)$ with $k_l \le k-1$, $m_l \le m$ and $n_l \le n$. ϕ_{right} is of size $(k_r,\ m_r,\ n_r)$ with $k_r \le k$, $m_r \le m-1$ and $n_r \le n-2$.

After each partition, we remove all unit clauses and pure literals from the child instances.

3.2 Solving a Pure Instance

Given a pure instance ϕ with the common *variable* x_j, we can further partition ϕ into two child instances ϕ_1 and ϕ_2 such that ϕ_1 contains only those clauses that contain the literal x_j and ϕ_2 contains only those clauses that contain the literal $\neg x_j$.

ϕ is satisfiable if and only if ϕ_1 and ϕ_2 are both satisfiable and there are certificates χ_1 for ϕ_1, and χ_2 for ϕ_2 such that χ_1 is compatible with χ_2.

We find all certificates for ϕ_1 and for ϕ_2. For ϕ_1, one possible certificate is $\{x_j \mapsto True\}$ but this is not compatible with any certificate for ϕ_2. In order to find the other certificates, we start with $\{x_j \mapsto False\}$ and reduce ϕ_1 to a smaller SAT instance. We then recursively find all certificates for that instance. By symmetry, ϕ_2 must be satisfied by starting with $\{x_j \mapsto True\}$ and finding all certificates for the resulting instance.

3.3 Combining Two Certificates

Given two instances ϕ_1 and ϕ_2 and their respective certificates χ_1 and χ_2, if χ_1 and χ_2 are compatible with each other, they can be merged. If χ_1 and χ_2 are compatible, they are two sets of assignments for variables with no conflicts among assignments, χ_1 and χ_2 can be combined into a certificate for $\Phi := \phi_1 \wedge \phi_2$ by simply taking the union of the two sets. Thus, $\chi := \chi_1 \cup \chi_2$ is a certificate for Φ.

3.4 Finding Parent Certificates

Given two sibling instances and all of their respective certificates, $(\phi_{left},\ \{\chi_{left}\})$ and $(\phi_{right},\ \{\chi_{right}\})$, we can find all certificates for the parent instance by performing a pair-wise merge of all compatible pairs of certificates from $\{\chi_{left}\}$ and $\{\chi_{right}\}$.

Note that Φ is satisfiable if and only if $\exists\ \chi_{left}, \chi_{right}$ such that χ_{left} and χ_{right} are mutually compatible. If there is no such pair of compatible certificates, then the parent instance is unsatisfiable.

3.5 The Solver

Let's piece together the sub-problems into the complete algorithm.

Given a SAT instance Φ, we can partition it into a pure instance ϕ_{pure} and a smaller instance ϕ_{right}. We solve ϕ_{pure} as discussed in 3.2 and find all certificates for ϕ_{pure}. We recursively solve ϕ_{right} to find all certificates for ϕ_{right}. We then find all certificates for Φ as discussed in 3.4.

4 ANALYSIS

We analyze the time and space complexity of the proposed algorithm. We will proceed under the assumption that variables and their negations are uniformly distributed among clauses and that no one clause has both a variable and its negation. This eases the analysis, but it should hold even if the uniformity assumption is incorrect.

4.1 Partitioning an Instance

...

4.2 Solving a Pure Instance

...

4.3 Combining Two Certificates

...

4.4 Finding Parent Certificates

...

Since there are more clauses for each certificate to satisfy, we expect a majority of certificates to be discarded in this step.

4.5 The Solver

...

5 BENCHMARKS

Let's run tests and compare against competitors.

6 CONCLUSIONS

We learned some things.

REFERENCES

[1] Bengt Aspvall, Michael F. Plass, and Robert E. Trajan. 1979. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inform. Process. Lett.* 8, 3 (03 1979).