CS120: Intro. to Algorithms and their Limitations Hesterberg & Vadhan

Lecture 24: Satisfiability Modulo Theories and Cook–Levin

Harvard SEAS - Fall 2022 *2022-11-22*

1 Announcements

Recommended Reading:

- De Moura & Bjørner: https://cacm.acm.org/magazines/2011/9/122785-satisfiability-modulo-theories/abstract

Announcements:

- PS9 due this Fri 12/2
- Final exam Thu 12/8
- Final review sessions starting this Thurs
- PP3, revisions on PS8 & 9 due 12/8 but will be accepted on 12/9 without penalty
- Please fill out Q survey

2 Satisfiability Modulo Theories

Definition 2.1. A *theory* $\mathcal{T}$ consists of a domain $\mathcal{D}$ and a collection $\mathcal{P}$ of predicates $p : \mathcal{D}^k \to \{0, 1\}$, $k \geq 0$.

Examples:

- Theory of Naturals: $\mathcal{D} = \mathbb{N}$, with constraints

- Theory of Difference Arithmetic: $\mathcal{D} = \mathbb{Q}$, with constraints

- Theory of Disjunctions: $\mathcal{D} = \{0, 1\}$, with constraints

- Theory of Bitvectors of length w: $\mathcal{D} = \{0, 1, \ldots, 2^w - 1\} \equiv \{0, 1\}^w$, with constraints

Input	**:** A CNF formula $\varphi(x_0, \ldots, x_{n_p-1}, y_0, \ldots, y_{n_q-1})$ on n_p *propositional variables* and n_q *auxiliary variables*, a sequence $z = (z_0, \ldots, z_{n_t-1})$ of n_t *theory variables*, and a sequence of n_q *theory predicates* $P_0(z), \ldots, P_{n_q-1}(z)$, each of which is obtained by applying a predicate $p \in \mathcal{P}$ to a sequence $(z_{i_0}, \ldots, z_{i_{k-1}})$ of theory variables.
Output	**:** Assignments

Computational Problem SatisfiabilityModuloT

Example:

The solvability of Satisfiability Modulo $\mathcal{T}$ depends on the choice of the theory $\mathcal{T}$. It can be simplified using the following formulation.

Input	**:** A sequence $z = (z_0, \ldots, z_n)$ of n *theory variables*, and a sequence of *theory predicates* $Q_0(z), \ldots, Q_{m-1}(z)$, each of which is obtained by applying a predicate $p \in \mathcal{P}$ *or its negation* to a sequence $(z_{i_0}, \ldots, z_{i_{k-1}})$ of theory variables.
Output	**:** An assignment

Computational Problem ConstraintSatisfactionInT

Examples:

Theorem 2.2. *For every theory $\mathcal{T}$, Satisfiability Modulo $\mathcal{T}$ is solvable iff Constraint Satisfaction in $\mathcal{T}$ is solvable.*

Proof sketch.
$\Rightarrow$

$\Leftarrow$ $\square$

In practice: use a SAT solver to find a satisfying assignment, then apply a constraint satisfaction solver. If the constraint satisfaction solver determines there is no corresponding assignment to the theory variables, construct a new clause to add to φ.

It turns that the Theory of Naturals is unsolvable (proven similarly to the optional problem 3a on PS9), but the Theory of Difference Arithmetic and the Theory of Bitvectors are solvable.

3 Program Verification and the Cook–Levin Theorem

SMT Solvers are a powerful tool for verifying properties of time-bounded algorithms, like we saw with Binary Search. Indeed, we will see how they can be used to solve the following very general problem.

Input : A Word RAM program P, an array of natural numbers $x = (x_0, \ldots, x_n)$ and parameters $w, m, t \in \mathbb{N}$.

Output : An array $y = (y_0, \ldots, y_{m-1})$ of natural numbers y such that:

1. Throughout the computation of P on input (x, y) the word length is at most w.
2. P halts on input (x, y) within t steps, and
3. $P(x, y) = 1$,

if such a y exists.

Computational Problem WordRAMSatisfiability

Many debugging problems can easily be reduced to WordRAMSatisfiability, if we fix bounds on the input length and the running time that we care about. For example, if we take $n = 0$ and modify P to output 1 only when an arithmetic overflow occurs, then WordRAMSatisfiability will tell us whether there is an input y of length m that causes an arithmetic overflow within t steps.

Furthermore, we can easily reduce every $\mathsf{NP}_{\mathsf{search}}$ problem Π to WordRAMSatisfiability, by taking P to be the verifier of solutions for Π: given an instance x of Π, we can set $m = n^{O(1)}$ to be the length of solutions for Π on input x and $t = n^{O(1)}$ to be the running time of the verifier. By using bignums, it can be shown that the word length w can be assumed to be $O(\log n)$ without loss of generality.

Theorem 3.1. *WordRAMSatisfiability can be reduced to Satisfiability Modulo the Theory of Bitvectors. Given an instance (P, x, w, m, t) of WordRAMSatisfiability, the (mapping) reduction produces an SMT instance with the same word size parameter w and runs in time polynomial in $|P|$, $|x|$, m, and t.*

Proof sketch. Given a WordRAMSatisfiability instance (P, x, w, m, t), we construct our SMT instance as follows:

- Propositional variables:

- Theory variables:

- Constraints:

□

Theorem 3.2. *Satisfiability Modulo the Theory of Bitvectors reduces to SAT in polynomial time.*

Proof sketch. In case $w = O(\log n)$:

A more efficient solution (to avoid the exponential dependence on w):

□

Note that the $\mathsf{NP}_{\mathsf{search}}$-hardness of SAT (i.e. the hard part of the Cook–Levin Theorem, which we stated without proof in Lecture 19) follows by combining Theorems 3.1 and 3.2.