CSE 513T Theory of Artificial Intelligence & Machine Learning October 20, 2016

Lecture 15

Lecturer: Brendan Juba *Scribe: Liuyuan Tan, Editor: Kiron Sukesan*

Overview
1. Patial Assignments
2. SAT Solvers and resolution
3. (if time) Efficient algorithms for fragments of resolution

Reminder: Final Projects
Choose a group of ≤ 3
Choose a topic
Discuss with Prof. Juba before starting

"SAT-Solvers"

Intent. Given as input a formula φ (usually CNF), return either a satisfying assignment or "unsatisfiable."
We can understand (partially) when these algorithms are efficient in terms of resolution proofs.
Algorithms for SAT typically operates by iteratively extending partial assignments.

1 Partial Assignments

We have n Boolean attributes $x_1, x_2, ..., x_n$
A partial assignment $\varrho \in \{0, 1, \star\}^n$ fixes some attributes x_i to Boolean values (when $\varrho_i \in \{0, 1\}$), and otherwise (if $\varrho_i = \star$) leaves it unfixed.
The operation of "plugging in" a partial assignment is called "applying a restriction"

Definition: For a CNF $\varphi(x) = c_1(x) \wedge c_2(x) \wedge ... \wedge c_m(x)$, the restriction of φ under a partial assignment ϱ, $c|_\varrho$ is a formula on the attributes $\{x_i : \varrho_i = \star\}$ given as follows: for each c_i, if some literal is set to 1 by ϱ, then $c_i|_\varrho$ is the constant 1. Otherwise, $c_i|_\varrho$ is the clause consisting of the literals of c_i not falsified by ϱ (i.e. $c_i|_\varrho = \bigvee_{l \in c_i, l(\varrho) \neq 0} l$). $\varphi|_\varrho$ is then the AND of those "surviving" clauses not set to 1 unless some $c_i|_\varrho = \perp$ (the empty clause), in which case $\varphi|_\varrho = 0$
Ex: $(x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) = \varphi$
$\varrho = (\star, 0, \star, 1)$
$\varphi|_\varrho = x_3$
$\psi = (x_1 \vee \neg x_3) \wedge (x_2 \vee \neg x_4)$
$\psi|_\varrho = 0$
Note: We say that an assignment $x\{0, 1\}^n$ is consistent with a partial assignment ϱ if whenever $\varrho \neq \star, \varrho_i = x_i$. Whenever $\varphi|_\varrho$ is fixed to a constant φ takes that value on any assignment x consistent with ϱ. We also have a natural notation of composing partial assignments. If σ is a partial assignment to $\{x_i : \varrho_i = \star\}$, then $\tau = \varrho\sigma$ is the partial assignment $\tau_i = \{\varrho_i if \varrho_i \neq \star, \sigma_i \textit{ if otherwise}\}$.
Notice also that $(\varphi|_\varrho)|_\sigma = \varphi|_{\varrho\sigma}$.

2 SAT Solvers

Naively, we "build up" a satisfying assignment one variable at a time.

Naive SAT (φ) :

If φ contains $\bot$ or $\varphi = 0$, return FAIL
Else if $\varphi = 1$, return $\varrho = (\star, \star, ..., \star)$
Else, choose an arbitrary attribute x_i in φ
If Naive SAT$(\varphi|_{x_i=1}) = \varrho$ ($\neq$ FAIL)
Return $(x_i = 1)\varrho$
Else if Naive SAT$(\varphi|_{x_i=0}) = \varrho$ ($\neq$ FAIL)
Return $(x_i = 0)\varrho$
Else Return FAIL

Three modifications

Pure literals attributes that only appear positively or negatively are fixed to the approprite values first.

Unit propapagation if there are any clauses consisting of single literals, set the attributes in any such literal to the satisfying value.
The above two are called the "DPLL" algorithm.

Clause learning & Restarts when a partial assignment falsifies φ, we add a new clause ruling out this partial assignment. (For example, the clause $\neg_{l:l(\varrho)=1}(\bigwedge l) = \bigvee_{l:l(\varrho)=1} l$ Many other strategies are useful.)
The algorithms using above modification are called the "CDCL" algorithms. Occasionally, the partial assignment is dropped and the search restarts with the new clauses.
Whenever any of those algorithms fails to find a satisfying assignment, the "trace" ofits execution gives a resolution refutation of of the formula (so its size is a lower bound on the running time of the algorithm). Moreover, for Naive SAT or DPLL, this is a treelike resolution refutation.
We can use this fact to find resolution proofs.

Naive SAT (φ):

If φ contains $\bot$ or $\varphi = 0$, return FAIL
Else if $\varphi = 1$, return $\varrho = (\star, \star, ..., \star)$
Else, choose an arbitrary attribute x_i in φ
If Naive SAT$(\varphi|_{x_i=1}) = \varrho$ ($\neq$ FAIL)
Return $(x_i = 1)\varrho$
Else if Naive SAT$(\varphi|_{x_i=0}) = \varrho$ ($\neq$ FAIL)
Return $(x_i = 0)\varrho$
Else for $\pi^{(0)} = NaiveSAT(\varphi|_{x_i} = 0), \pi^{(1)=NaiveSAT(\varphi|_{x_i=1})}$
If using the corresponding clauses we obtain a (treelike) refutation π of φ, return π
Else, return the proof

$$\begin{array}{ccc} & \bot & \text{(with cut on } x_i\text{)} \\ \pi^{(1)} & & \pi^{(0)} \end{array}$$

Claim: Modified Naive RES returns a treelike refutation of any unsatisfiable CNF φ

Proof: by induction on # of free variables
Base: (no free variable)
Some φ is unsatisfiable, $\phi \neq 1$
So if $\varphi \neq 0$, φ must consist of only empty clauses, so $\bot$ is a refutation.
Induction Step: Since φ is unsatisfiable, both $\varphi|_{x_i=1}$ and $\varphi|_{x_i=0}$ are unsatisfiable (otherwise we could extend to a satisfying assignment of $\varphi|$). So, since these each have one less free variable, Naive SAT returns refutations of $\varphi|_{x_i=1}, \pi^{(1)}$, and $\varphi|_{x_i=0}, \pi^{(0)}$
Let's consider the corresponding derivations $\tilde{\pi}^{(0)}$ and $\tilde{\pi}^{(1)}$ on the original clauses of φ (following the same proof steps)
Recalling that a resolution refutation derives $\bot$, these are three possiblities: either one of $\tilde{\pi}^{(0)}$ or $\tilde{\pi}^{(1)}$ it self derives $\bot$, or, $\tilde{\pi}^{(0)}$ derives x_i and $\tilde{\pi}^{(1)}$ derives $\neg x_i$. In the first two cases, Naive SAT returns the appropriate one of $\tilde{\pi}^{(0)}$ or $\tilde{\pi}^{(1)}$, and in the third case, Naive SAT returns the resolution proof

$$\begin{array}{ccc} & \bot & \\ \pi^{(1)} & & \pi^{(0)} \end{array}$$

(using a cut on x_1). Since $\tilde{\pi}^{(0)}$ and $\tilde{\pi}^{(1)}$ are not sharing lines, this is a treelike proof.

Corollary: treelike resolution is complete (for refuting CNFs)

Remarks

(1) We'll see on HW3 that if unit propagation falsifies a formula, we can still extract a resolution refutation from the unit propagation inferences (so, the size of a treelike RES refutation also bounds the running time of DPLL on unsatisfiable formulas).

(2) The size of a general ("DAG-like") resolution refutation can similarly be shown to give a lower bound for the running time of CDCL algorithms.
Interesting open question: can CDCL algorithms find resolution proofs efficiently?