Comp409, Lecture 13

Thursday, February 27, 2003

1 Review

We've been discussing boundaries between tractable and intractable problems. In the class of k-CNF problems we have:

1-CNF: easy
2-CNF: easy
3-CNF: NP-complete

We also looked at the class of k^+-CNF problems, which have a maximum of k positive literals per clause. For this class of problems, we have:

0^+-CNF: easy
1^+-CNF (Horn-CNF): We will show this to be easy in this lecture.
2^+-CNF: NP-complete

2 Horn Formulas

We have the following nomenclature :

- Literal : A literal is either a proposition (p) or the negation of a proposition ($\neg p$) and is called a *positive* or a *negative literal* respectively.
- Clause : A clause is a set of literals.
- Horn Clause : A clause with at most 1 positive literal.
- Horn Formula : A set of Horn Clauses.

A Horn Clause can be of the following type :

1. There are no positive literals in the clause. In $3-CNF$ form the clause would look like

   $(\neg p_1 \vee \neg p_2 \vee \neg p_3) \models\!\!\!\!\dashv \neg(p_1 \wedge p_2 \wedge p_3)$

This is known as a *constraint*

2. The clause contains precisely 1 positive literal.

   There can be two cases :

   (a) There are no negative literals in the clause (this implies that there is exactly one literal in the clause and that is a positive literal)

   $$p.$$

   This is called a *fact.*

   (b) There are some negative literals in the clause.

   $$(\neg p_1 \vee \neg p_2 \vee q) \models\Dashv (p_1 \wedge p_2) \rightarrow q$$

   This is called a *rule.*

3 DPLL

Recall the splitting algorithm:

$SAT(\varphi)$
 if $\varphi \in 0.1$ return φ
 choose $p \in AP(\varphi)$ and $c \in 0,1$
 if $SAT(\varphi[p \mapsto c])$ return 1
 if $SAT(\varphi[p \mapsto 1-c])$ return 1
 return 0

Let $\varphi = \{k_1, \ldots, k_n\}_\wedge$ and $k_i = \{l_1, \ldots, l_n\}_\vee$. For a given clause k_i, the possible results of making the substitution $[p \rightarrow c]$ are:

p	other props in k_i?	c	$k_i[p \rightarrow c]$
not in k_i	doesn't matter	any	k_i
positive	doesn't matter	1	1
negative	doesn't matter	0	1
positive	no other props	0	0
positive	there are other props	0	$k_i - \{p\}$
negative	no other props	1	0
negative	there are other props	1	$k_i - \{\neg p\}$

Note that Horn clauses are preserved under substitutions, i.e, the result of a subsitution is also a Horn clause. The same holds for Horn formulas.

This suggests a heuristic for choosing which variable to assign next:

Unit Clause: A clause consisting of only one literal.

Unit Preference: If there is a unit clause, then if it is a positive literal p, make the assignment $[p \to 1]$. If it is a negative literal $\neg p$, make the assignment $[p \to 0]$.

Using the splitting method with unit preference gives us the DPLL(*Davis-Putnam-Longemann-Loveland*) algorithm.

4 Properties of Horn Formulas

First note that both rules and constraints are satisfied by the truth assignment that assigns 0 to all propositions.

Lemma 1 *A fact-free Horn formula is satisfiable.*

Similarly, both rules and facts are trivially satisfiable by assigning all propositions the value 1.

Lemma 2 *A constraint-free Horn formula is satisfiable.*

Definition 1 *A CNF-formula is* all-neg*(resp.,* all-pos*) if all clauses contain at least one negative (resp., positive) literal.*

Note that fact-free Horn formulas are all-neg and constraint-free Horn formulas are all-pos. All-neg formulas are satisfiable by the universal 0 assignment, while all-pos formulas are satisfiable by the universally 1 asisgnment.

Finally, recall from last lecture that when you simplify a clause with at most one positive literal, you still have at most one positive literal.

Fact 1 *Horn formulas are closed under simplifaction.*

5 Horn formulas and DPLL

Let's make the following observations:

1. A fact-free formula is easily satisfiable.
2. If a formula is not easily satisfiable, then it's not fact free. We will then choose a fact, assign it true, then recurse. Notice that in this way we never have to split.

The DPLL algorithm can be rewritten for Horn formulas as follows:

```
DPLL-Horn(φ) [Where φ is a Horn formula]
   If φ is 1 or fact-free, return 1
```

```
If φ is 0, return 0
Choose p, a fact in φ
return DLL − Horn(φ[p ↦ 1])
```

Splitting has an exponential search space. By eliminating splitting, the above algorithm calls itself only a linear number of times.

Theorem 1 *DLL-Horn runs in $PTIME$ and $PSPACE$.*

Corollary 1 *HORN-SAT is in $PTIME$.*

Once it was shown that HORN-SAT is in $PTIME$ and $PSPACE$, the question arises: Is HORN-SAT in $NLOGSPACE$?

The question: "Where does HORN-SAT belong?" is still unanswered. Whenever such a problem is found, the next step is to find a class of equivalent problems for which the answer is not known. As such, we define the following:

Definition 2 *A problem L is $PTIME$-complete if $L \in PTIME$ and for every $L' \in PTIME$, we have $L' \leq_{LOG} L$.*

Theorem 2 (Cook) *HORN-SAT is $PTIME$-complete.*

HORN-SAT is then as hard as anything in $PTIME$. Thus, if HORN-SAT $\in NLOGSPACE$, then $NLOGSPACE = PTIME$, which is unlikely. Whether $NLOGSPACE = PTIME$ is an open question.

Theorem 3 *2-CNF-SAT is $NLOGSPACE$-complete*

Another open question is whether 2CNF-SAT$\in LOGSPACE$. We can define $NLOGSPACE$-completeness in a similar way to $PTIME$-completeness. Thus if 2-CNF-SAT is in $LOGSPACE$, $LOGSPACE = NLOGSPACE$.

We have now established representative problems for several complexity classes:

$$L \subseteq NL(2{-}SAT) \subseteq P(Horn{-}SAT) \subseteq NP(SAT) \subseteq PSPACE(quatifiedSAT)$$

6 Proof systems

The first question in this class was: "What is logic?" Here are two of the answers:

1. Logic is concerned with the truth - this is handled by the model theory
2. Logic is concerned with evidence and compelling arguments - proof theory

A **Formal System**, **Proof System**, or a **Deductive System** (which are all equivalent terms) is some $\Gamma \subseteq (Form)^*$. An *inference rule* $\varphi_1 ... \varphi_k \in \Gamma$ (usually written as $\frac{\varphi_1,...,\varphi_{k-1}}{\varphi_k}$) means that φ_k follows from $\varphi_1, ..., \varphi_{k-1}$. A special case of this is $\frac{}{\varphi}$, which is called an *axiom* [1].

Example A common rule of inference is modus ponens: "If p holds and $p \to q$ holds, then q holds". This can be written $\frac{p, p \to q}{q}$.

Definition 3 *A* deduction with respect to Γ *is a sequence* $\varphi_1, ..., \varphi_n$ *such that for every* φ_j*, there are* $j_1 < j_2 < ... < j_{k-1} < j$ *such that* $\frac{\varphi_{j_1} \varphi_{j_2} \cdots \varphi_{j_{k-1}}}{\varphi_j} \in \Gamma$.

[1] There is usually a formal distinction between axioms and inference rules. For this class axioms will be considered as a particular type of inference rules.