Satisfiability Encodings (04)

Michael L. Littman

January 26th, 1999

1 SATISFIABILITY

1.1 Recap

Last time, we talked about the definition of the Boolean satisfiability problem.
We also talked about several kinds of algorithms for solving them. Types?

> Stochastic Local Search, Systematic: DPLL.

1.2 Applications

According to Bart Selman (1996), algorithms can solve SAT problems with 10k variables and 1M constraints!
With problems this size, it becomes practical to encode real-world problems in SAT.
For example, problems from planning, scheduling, protein folding, graph coloring, and general CSPs can be converted to SAT.

2 SAT ENCODINGS

2.1 Graph Coloring

To flex our encoding muscles, let's start with graph coloring:

- Given a graph and a set of colors, color all nodes, but no adjacent nodes can have the same color.

Example: 3×3 grid.
Can write this as a CSP? Need to know the variables, domains for the variables, and the constraint matrices.

> Variables are the nodes of the graph. Edges are the same. Domains are all the same, with values the complete set of colors. Constraint matrices are $n \times n$ (if there are n colors) consisting of all 1s except 0s down the diagonal.

2.2 SAT as a CSP

How about 2-SAT as a CSP?
Example: $(x_1 + x_2)(\overline{x_1} + \overline{x_2})(x_1 + x_3)(\overline{x_1} + \overline{x_3})$ $(x_4 + x_2)(\overline{x_4} + \overline{x_2})(x_3 + x_4)(\overline{x_3} + \overline{x_4})$.
What are the variables, domains, and constraint matrices?

> Variables are variables, domains are T and F. Any two variables that share a clause have an edge between them. The constraint matrix depends on the logical relationship between the variables. Can "and" matrices together.

2.3 3-SAT as a CSP

What happens when we go to general 3-SAT?

> Need more than just matrices for constraints.

Introduce variables for clauses. Value corresponds to which literal makes it true. Example: $C_1 : (x_1 + \overline{x_2} + x_3)$.

2.4 CSP as SAT

How would you express CSP as SAT?

- Given: variables V; domains $D(x)$ for all $x \in V$; constraint matrices $M(x, y, u, v)$ for all $x, y \in V$, $u \in D(x)$, and $v \in D(y)$.
- Need: Boolean variables, clauses.

Make each CSP variable-value pair a Boolean variable: $P(x, v)$ for all $x \in V$ and $v \in D(x)$. Let n be the number of CSP variables and k be the number of values in each domain. Total Boolean variables: nk.
Constraints:

- Each variable has at least one value.

  > For all $x \in V$: $\sum_{v \in D(x)} P(x, v)$.

- Each variable has at most one value.

  > For all $x \in V$, $v \in D(x)$, $u \in D(x) - \{v\}$: $\overline{P(x, v)} + \overline{P(x, u)}$.

- No prohibited pairs of values.

  > For all x, y, u, v such that $M(x, y, u, v) = 0$: $\overline{P(x, u)} + \overline{P(y, v)}$.

Total constraints: $n + nk(k - 1) + m$, where m is the total number of prohibited pairs of values (no more than n^2k^2).

3 BATTLESHIPS

3.1 Encoding a Complex Problem

Now that we're starting to get good at converting things to SAT, let's try something really challenging.
Because of the nature of the complexity class NP, almost all discrete logic puzzles can be expressed as SAT problems.
We'll take an interesting one from Games magazine.

3.2 Battleships Instructions

Each grid represents a section of ocean in which the entire fleet is hiding. This fleet consists of one battleship (four grid squares in length), two cruisers (each three squares long), three destroyers (each two squares long), and four submarines (one square each). The ships may be oriented either horizontally or vertically, and no two ships will occupy adjacent grid squares, *even diagonally*. The digits along the right side of and below the grid indicate the number of grid squares in the corresponding rows and columns that are occupied by vessels.
In each of the puzzles below, one or more shots have been taken to start you off. These may show water (indicated by a wavy lines), a complete submarine (a circle), or the middle (a square) or the end (a rounded-off square) of a longer vessel. The puzzles get harder as you go. Only Battleships genuises will reach the rank of admiral by finding all the fleets.

3.3 Battleships

Ships:

- Battleship: ◁□□▷
- Cruiser: ◁□▷
- Destroyer: ◁▷
- Submarine: ◎
- Water: ≈

										2
										1
										1
										2
							▽			2
										3
	◁									2
										1
										5
				▽						1
3	2	2	4	4	1	1	2	0	1	

3.4 Objects

There are a number of important sets:

- Rows: $i \in \{1, \ldots, 10\}$.
- Columns: $j \in \{1, \ldots, 10\}$.
- Parts: $\{\triangle, \triangledown, \triangleright, \triangleleft, \square, \circledcirc\}$.
- Direction: { across (0), down (1) }.
- Ship Number: $k \in$ { 1,2,3,4 }.

3.5 Variables

- Filled: for all i in Rows, j in Columns, $f(i,j)$ if position i,j contains a ship.
- Part: for all i in Rows, j in Columns, $pa(i,j)$, $pb(i,j)$, and $pc(i,j)$ encode the type of ship part in the given grid cell. i,j contains a ship.
- Battleship: $bi(i)$, $bj(j)$, bd encodes the position and direction of the battleship.
- Cruisers: $ci(k,i)$, $cj(k,j)$, $cd(k)$ encodes the position and direction of each cruiser, $k \in \{1,2\}$.
- Destroyers: $di(k,i)$, $dj(k,j)$, $dd(k)$ encodes the position and direction of each destroyer, $k \in \{1,2,3\}$.
- Submarines: $si(k,i)$, $sj(k,j)$ encodes the position of each submarine, $k \in \{1,2,3,4\}$.

Total variables: $10\times10+3(10\times10)+(10+10+1)+2(10+10+1)+3(10+10+1)+4(10+10) = 606$.

3.6 Initial Conditions

The starting information in the grid forces some of the variables to become true.
Ship parts are encoded using the pa, pb, and pc variables:

part	pa	pb	pc
$\triangle$	0	0	0
$\triangledown$	0	0	1
$\triangleright$	0	1	0
$\triangleleft$	0	1	1
$\square$	1	0	0
$\circledcirc$	1	0	1

For example: $f(5,8)$, $\overline{pa(5,8)}$, $\overline{pb(5,8)}$, $pc(5,8)$, $f(7,2)$, $\overline{pa(7,2)}$, $pb(7,2)$, $pc(7,2)$, $f(10,5)$, $\overline{pa(10,5)}$, $\overline{pb(10,5)}$, $pc(10,5)$.
Water can be encoded as well by making an f variable false.
Total Constraints: Never more than rows by columns by 4 variables (400). Rarely over 30.

3.7 Footprints

Each ship leaves a "footprint" on the water. That is, grid cells take on a particular value depending on where the ships are.
Example: $di(2,6)dj(2,5)\overline{dd(2)} \to \overline{f(5,4)}$. This says that a destroyer at (6,5) going across means that there is water at (5,4).
This can be made into a clause by negating the antecendent:
$di(2,6)dj(2,5)\overline{dd(2)} \to \overline{f(5,4)}$
Total: $2 \times 7 \times 10 \times 18 + 2 \times 2 \times 8 \times 10 \times 15 + 3 \times 2 \times 9 \times 10 \times 12 + 4 \times 10 \times 10 \times 9 = 17400$.

3.8 Counts

Each row and column (20 altogether) must have the number of filled squares sum to the number indicated.
We can introduce some more variables ($3 \times 20 = 60$), to represent the sum. Then, a set of clauses ($3 \times 20 = 60$) match the actual total to the desired total.
There are lots of ways to compute the sum, that vary in the number of variables and clauses they add. If you have an efficient addition circuit, it can be encoded in SAT!
Here's the simplest thing I could come up with. Say we want to know if τ of n Boolean variables $x_1, \ldots, x_n$ are on.

- Create variables $\sigma(i,t)$ for $2 \le i \le n$ and $0 \le t \le \tau$.
- We define $\sigma(i,t)$ to indicate whether $\sum_{j=1}^{i} x_j = t$ (this is a real sum this time, not an "or").
- Ultimately, we make a clause $\sigma(n,\tau)$ to force the sum constraint.

  Base case: $\sigma(1,1) = x_1$, $\sigma(1,0) = \overline{x_1}$, $\sigma(1,t) = 0$ for $t > 1$, $\sigma(i,-1) = 0$ for all i.
- $\sigma(i-1,t-1)x_i \to \sigma(i,t)$, $\sigma(i-1,t)\overline{x_i} \to \sigma(i,t)$, $\sigma(i-1,t-1)\overline{x_i} \to \overline{\sigma(i,t)}$, $\sigma(i-1,t)x_i \to \overline{\sigma(i,t)}$.
- We don't need exclusion constraints on $\sigma(i,\cdot)$.

For Battleships, we get $20 \times 8 \times 9 = 1440$ new variables and $4 \times 20 \times 8 \times 9 = 5760$ new clauses.

3.9 All Else Empty

How do we state that anywhere there *isn't* a ship is water?

> Don't have to. There are 20 boat pieces total, and all are forced to appear by row and column counts. Nothing extra will be allowed.

So, total variables: 2046. Total constraints: 23560.

3.10 Encoding Inference Rules

Solving shortcuts you discover can be encoded as inference rules.
For the planning domain, this speeds things up tremendously!

4 USING THE SOLVER

4.1 Input Format

- Variables given numbers from 1 to n.
- Variable negation shown by negating the variable number.
- Lines are zero terminated (not for tableau, unfortunately).
- Comment lines begin with 'c'.
- First content line has "p cnf n m" where n is the number of variables and m the number of clauses.
- After that, it's one clause per line.

Example:

```
c generated by makewff, seed= 702008999
p cnf 4 8
1 -2 -4 0
1 -2 4 0
1 2 -3 0
1 2 3 0
-1 -2 3 0
-1 -3 4 0
-1 2 -4 0
-2 -3 -4 0
```

4.2 Output Format

If a satisfying assignment is found, output is the assignment (using `-solcnf` flag):

```
v 1
v -2
v -3
v -4
```