7.2.2.2. Satisfiability. We turn now to one of the most fundamental problems of computer science: Given a Boolean formula $F(x_1, \ldots, x_n)$, expressed in so-called "conjunctive normal form" as an AND of ORs, can we "satisfy" F by assigning values to its variables in such a way that $F(x_1, \ldots, x_n) = 1$? For example, the formula

$$F(x_1, x_2, x_3) = (x_1 \vee \bar{x}_2) \wedge (x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \qquad (1)$$

is satisfied when $x_1 x_2 x_3 = 001$. But if we rule that solution out, by defining

$$G(x_1, x_2, x_3) = F(x_1, x_2, x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3), \qquad (2)$$

then G is unsatisfiable: It has no satisfying assignment.

that simplification, and with 'x_j' identical to 'j', Eq. (1) becomes

$$F \;=\; \bigl\{\{1,\bar{2}\}, \{2,3\}, \{\bar{1},\bar{3}\}, \{\bar{1},\bar{2},3\}\bigr\}.$$

And we needn't bother to represent the clauses with braces and commas either; we can simply write out the literals of each clause. With that shorthand we're able to perceive the real essence of (1) and (2):

$$F = \{1\bar{2}, 23, \bar{1}\bar{3}, \bar{1}\bar{2}3\}, \qquad G = F \cup \{12\bar{3}\}. \qquad (3)$$

Find a binary sequence $x_1 \ldots x_8$ that has no three equally spaced $0s$ *and no three equally spaced* $1s$. For example, the sequence 01001011 almost works; but it doesn't qualify, because x_2, x_5, and x_8 are equally spaced 1s.

Find a binary sequence $x_1 \ldots x_8$ that has no three equally spaced $0s$ *and no three equally spaced* $1s$. For example, the sequence 01001011 almost works; but it doesn't qualify, because x_2, x_5, and x_8 are equally spaced 1s.

integers j and k: *If* n *is sufficiently large, every binary sequence* $x_1 \ldots x_n$ *contains either* j *equally spaced* $0s$ *or* k *equally spaced* $1s$. The smallest such n is denoted

Find a binary sequence $x_1 \ldots x_8$ that has no three equally spaced 0s and no three equally spaced 1s. For example, the sequence 01001011 almost works; but it doesn't qualify, because x_2, x_5, and x_8 are equally spaced 1s.

---

integers j and k: *If n is sufficiently large, every binary sequence $x_1 \ldots x_n$ contains either j equally spaced 0s or k equally spaced 1s.* The smallest such n is denoted

---

Find Boolean formula satisfiable iff $\exists$ n-bit sequence with no j equally spaced 0s and not k equally spaced 1s. A variable x_i for each bit...

Find a binary sequence $x_1 \ldots x_8$ that has no three equally spaced 0*s and no three equally spaced* 1*s.* For example, the sequence 01001011 almost works; but it doesn't qualify, because x_2, x_5, and x_8 are equally spaced 1s.

integers j and k: *If n is sufficiently large, every binary sequence $x_1 \ldots x_n$ contains either j equally spaced* 0*s or k equally spaced* 1*s.* The smallest such n is denoted

Find Boolean formula satisfiable iff $\exists$ n-bit sequence with no j equally spaced 0s and not k equally spaced 1s. A variable x_i for each bit...

Let us accordingly define the following set of clauses when $j, k, n > 0$:

$$waerden(j, k; n) = \{(x_i \vee x_{i+d} \vee \cdots \vee x_{i+(j-1)d}) \mid 1 \le i \le n-(j-1)d,\ d \ge 1\}$$
$$\cup \{(\bar{x}_i \vee \bar{x}_{i+d} \vee \cdots \vee \bar{x}_{i+(k-1)d}) \mid 1 \le i \le n-(k-1)d,\ d \ge 1\}. \qquad (10)$$

Find a binary sequence $x_1 \ldots x_8$ that has no three equally spaced 0s and no three equally spaced 1s. For example, the sequence 01001011 almost works; but it doesn't qualify, because x_2, x_5, and x_8 are equally spaced 1s.

integers j and k: *If n is sufficiently large, every binary sequence $x_1 \ldots x_n$ contains either j equally spaced 0s or k equally spaced 1s.* The smallest such n is denoted

Find Boolean formula satisfiable iff $\exists$ n-bit sequence with no j equally spaced 0s and not k equally spaced 1s. A variable x_i for each bit...

Let us accordingly define the following set of clauses when $j, k, n > 0$:

$$waerden(j,k;n) = \big\{(x_i \vee x_{i+d} \vee \cdots \vee x_{i+(j-1)d}) \bigm| 1 \le i \le n-(j-1)d,\ d \ge 1\big\}$$
$$\cup \big\{(\bar{x}_i \vee \bar{x}_{i+d} \vee \cdots \vee \bar{x}_{i+(k-1)d}) \bigm| 1 \le i \le n-(k-1)d,\ d \ge 1\big\}. \qquad (10)$$

```
for i from 1 to n-(j-1)
  for d from 1 to floor((n-i)/(j-1))
    AddClause({i+0*d, i+1*d, ..., i+(j-1)*d})

for i from 1 to n-(k-1)
  for d from 1 to floor((n-i)/(k-1))
    AddClause({-(i+0*d), -(i+1*d), ..., -(i+(k-1)*d)})
```

```
c 3 3 8
p cnf  8 24
1 2 3 0
1 3 5 0
1 4 7 0
2 3 4 0
2 4 6 0
2 5 8 0
3 4 5 0
3 5 7 0
4 5 6 0
4 6 8 0
5 6 7 0
6 7 8 0
-1 -2 -3 0
-1 -3 -5 0
-1 -4 -7 0
-2 -3 -4 0
-2 -4 -6 0
-2 -5 -8 0
-3 -4 -5 0
-3 -5 -7 0
-4 -5 -6 0
-4 -6 -8 0
-5 -6 -7 0
-6 -7 -8 0
```

```
c 3 3 9
p cnf  9 32
1 2 3 0
1 3 5 0
1 4 7 0
1 5 9 0
2 3 4 0
2 4 6 0
2 5 8 0
3 4 5 0
3 5 7 0
3 6 9 0
4 5 6 0
4 6 8 0
5 6 7 0
5 7 9 0
6 7 8 0
7 8 9 0
-1 -2 -3 0
-1 -3 -5 0
-1 -4 -7 0
-1 -5 -9 0
-2 -3 -4 0
-2 -4 -6 0
-2 -5 -8 0
-3 -4 -5 0
-3 -5 -7 0
-3 -6 -9 0
-4 -5 -6 0
-4 -6 -8 0
-5 -6 -7 0
-5 -7 -9 0
-6 -7 -8 0
-7 -8 -9 0
```

Online SAT Solver

Propositional theory in DIMACS format

```
c 3 3 8
p cnf  8 24
1 2 3 0
1 3 5 0
1 4 7 0
2 3 4 0
2 4 6 0
2 5 8 0
3 4 5 0
3 5 7 0
4 5 6 0
4 6 8 0
5 6 7 0
6 7 8 0
-1 -2 -3 0
-1 -3 -5 0
-1 -4 -7 0
```

solve

Answer

sat

Model

1	2	-3	-4	5
6	-7	-8		

Online SAT Solver

Propositional theory in DIMACS format

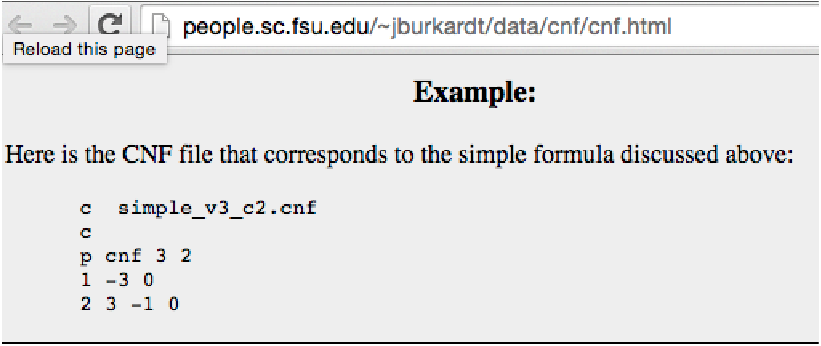
```
c 3 3 9
p cnf  9 32
1 2 3 0
1 3 5 0
1 4 7 0
1 5 9 0
2 3 4 0
2 4 6 0
2 5 8 0
3 4 5 0
3 5 7 0
3 6 9 0
4 5 6 0
4 6 8 0
5 6 7 0
5 7 9 0
6 7 8 0
```

solve

Answer

unsat

DIMACS (CNF) format and SAT Solvers

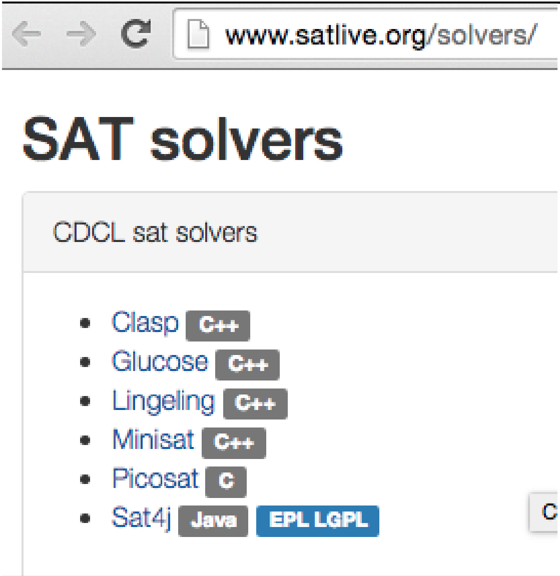


Example:

Here is the CNF file that corresponds to the simple formula discussed above:

```
c  simple_v3_c2.cnf
c
p cnf 3 2
1 -3 0
2 3 -1 0
```

DIMACS (CNF) format and SAT Solvers



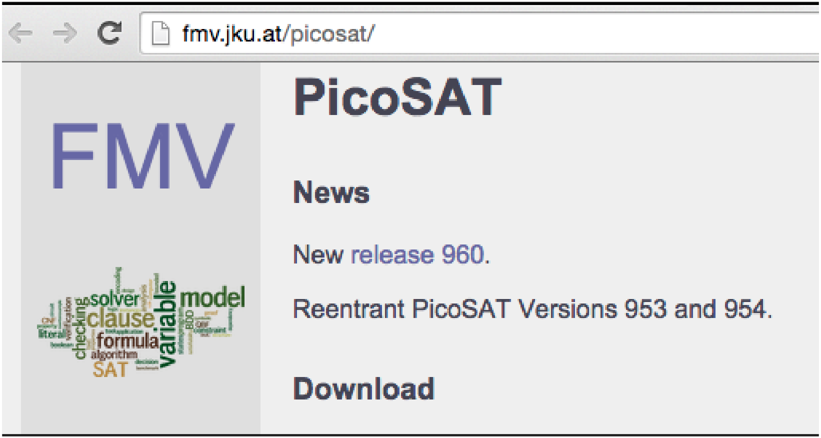

minisat.se/MiniSat.html
Main
MiniSat
MiniSat+
SatELite
Papers
Authors
Links
MiniSat
MiniSat started out 2003 as an effort to help
documentation (through the following paper).
containing all the features of the current stat
dynamic variable order, two-literal watch sche
variables.
In later versions, the code base has grown a bit
competition 2005, version 1.13 proved that Min
Below we provide a set of different versions o
extensions and suggestions for improvements,
freer licence than the LGPL, basically allowing

www.labri.fr/perso/lsimon/glucose/

The Glucose SAT Solver

Glucose is based on a new scoring scheme (well, not so new now, it was introduced in 2009) for the clause learning mechanism of so called "Modern" SAT sovlers (it is based our IJCAI'09 paper). It is designed to be parallel, since 2014. This page summarizes the techniques embedded in all the versions of glucose. The name of the Solver name is a contraction of the concept of "glue clauses", a particular kind of clauses that glucose detects and preserves during search.

⚠ Glucose is heavily based on Minisat, so please do cite Minisat also if you want to cite Glucose.



fmv.jku.at/picosat/

FMV

PicoSAT

News

New release 960.

Reentrant PicoSAT Versions 953 and 954.

Download

www-cs-faculty.stanford.edu/~uno/programs.html

DIMACS-TO-SAT and SAT-TO-DIMACS
    Filters to convert between DIMACS format for SAT problems and the symbolic semant

SAT0
    My implementation of Algorithm 7.2.2.2A (very basic SAT solver)

SAT0W
    My implementation of Algorithm 7.2.2.2B (teeny tiny SAT solver)

SAT8
    My implementation of Algorithm 7.2.2.2W (WalkSAT)

SAT9
    My implementation of Algorithm 7.2.2.2S (survey propagation SAT solver)

SAT10
    My implementation of Algorithm 7.2.2.2D (Davis-Putnam SAT solver)

SAT11
    My implementation of Algorithm 7.2.2.2L (lookahead 3SAT solver)

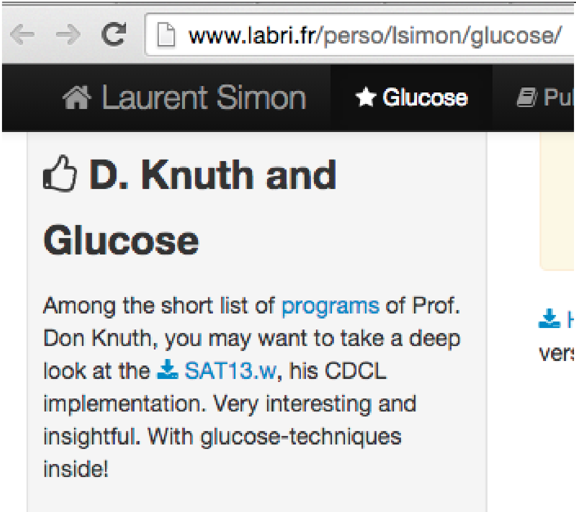
SAT11K
    Change file to adapt SAT11 to clauses of arbitrary length

SAT12 and the companion program SAT12-ERP
    My implementation of a simple preprocessor for SAT

SAT13
    My implementation of Algorithm 7.2.2.2C (conflict-driven clause learning SAT solver)

SAT-LIFE
    Various programs to formulate Game of Life problems as SAT problems (July 2013)

SATexamples
    Programs for various examples of SAT in Section 7.2.2.2 of TAOCP; also more than a

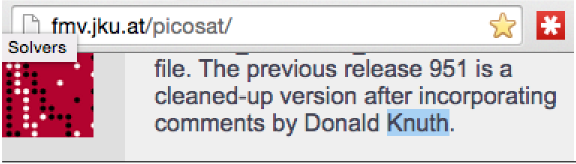


www.labri.fr/perso/lsimon/glucose/

Laurent Simon | ★ Glucose | Pu

D. Knuth and Glucose

Among the short list of programs of Prof. Don Knuth, you may want to take a deep look at the SAT13.w, his CDCL implementation. Very interesting and insightful. With glucose-techniques inside!



fmv.jku.at/picosat/

Solvers

file. The previous release 951 is a cleaned-up version after incorporating comments by Donald Knuth.

Back to binary Waerden sequences!

Recall

integers j and k: *If n is sufficiently large, every binary sequence $x_1 \ldots x_n$ contains either j equally spaced 0s or k equally spaced 1s.* The smallest such n is denoted

Back to binary Waerden sequences!

Recall

integers j and k: *If n is sufficiently large, every binary sequence $x_1 \ldots x_n$ contains either j equally spaced* 0*s or k equally spaced* 1*s.* The smallest such n is denoted

Boolean formula satisfiable iff $\exists$ n-bit sequence with no j equally spaced 0s and not k equally spaced 1s.

Let us accordingly define the following set of clauses when $j, k, n > 0$:

$$\begin{aligned} waerden(j,k;n) = &\big\{(x_i \vee x_{i+d} \vee \cdots \vee x_{i+(j-1)d}) \;\big|\; 1 \le i \le n-(j-1)d,\; d \ge 1\big\} \\ &\cup \big\{(\bar{x}_i \vee \bar{x}_{i+d} \vee \cdots \vee \bar{x}_{i+(k-1)d}) \;\big|\; 1 \le i \le n-(k-1)d,\; d \ge 1\big\}. \qquad (10) \end{aligned}$$

Back to binary Waerden sequences!

Recall

integers j and k: *If n is sufficiently large, every binary sequence $x_1 \ldots x_n$ contains either j equally spaced 0s or k equally spaced 1s.* The smallest such n is denoted

Boolean formula satisfiable iff $\exists$ *n*-bit sequence with no *j* equally spaced 0s and not *k* equally spaced 1s.

Let us accordingly define the following set of clauses when $j, k, n > 0$:

$$waerden(j,k;n) = \{(x_i \vee x_{i+d} \vee \cdots \vee x_{i+(j-1)d}) \mid 1 \le i \le n-(j-1)d,\ d \ge 1\}$$
$$\cup \{(\bar{x}_i \vee \bar{x}_{i+d} \vee \cdots \vee \bar{x}_{i+(k-1)d}) \mid 1 \le i \le n-(k-1)d,\ d \ge 1\}. \qquad (10)$$

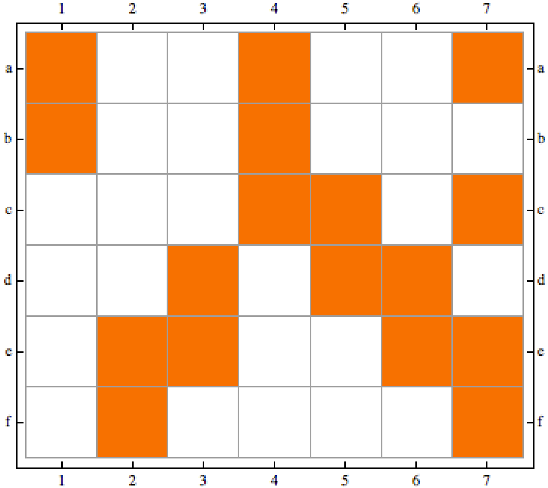
The 32 clauses in (9) are $waerden(3,3;9)$; and in general $waerden(j,k;n)$ is an appealing instance of SAT, satisfiable if and only if $n < W(j,k)$.

It's obvious that $W(1,k) = k$ and $W(2,k) = 2k - [k\,\text{even}]$; but when j and k exceed 2 the numbers $W(j,k)$ are quite mysterious. We've seen that $W(3,3) = 9$, and the following nontrivial values are currently known:

k =	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$W(3,k)$ =	9	18	22	32	46	58	77	97	114	135	160	186	218	238	279	312	349
$W(4,k)$ =	18	35	55	73	109	146	309	?	?	?	?	?	?	?	?	?	?
$W(5,k)$ =	22	55	178	206	260	?	?	?	?	?	?	?	?	?	?	?	?
$W(6,k)$ =	32	73	206	1132	?	?	?	?	?	?	?	?	?	?	?	?	?

Exact Cover

Given a $0-1$ matrix, find a selection of the rows that has exactly one 1 in each column.



Langford pairing

A permutation of $1, 1, 2, 2, 3, 3, ..., n, n$ so that the two ks are k “slots” apart.

Langford pairing

A permutation of $1, 1, 2, 2, 3, 3, \ldots, n, n$ so that the two ks are k "slots" apart.

Express as exact cover. Find a selection of the rows that has exactly one 1 in each column.

```
100010100000        1    1.1.....
100001010000        1    .1.1....
100000101000        1    ..1.1...
100000010100        1    ...1.1..
100000001010        1    ....1.1.
100000000101        1    .....1.1
010010010000         2   1..1....
010001001000         2   .1..1...
010000100100         2   ..1..1..
010000010010         2   ...1..1.
010000001001         2   ....1..1
001010001000          3 1...1...
001001000100          3 .1...1..
001000100010          3 ..1...1.
001000010001          3 ...1...1
000110000100           41....1..
000101000010           4.1....1.
000100100001           4..1....1
```

Exact Covering as SAT problem

Assign y_i to row i. Obtain conditions:

- Column 1: $y_1 + y_2 + y_3 + y_4 + y_5 + y_6 = 1$
- Column 2: $y_7 + y_8 + y_9 + y_{10} + y_{11} = 1$

...

- Column 5: $y_1 + y_7 + y_{12} + y_{16} = 1$
- Column 6: $y_2 + y_8 + y_{13} + y_{17} = 1$

...

- Column 12: $y_6 + y_{11} + y_{15} + y_{18} = 1$

```
1    1.1.....
1    .1.1....
1    ..1.1...
1    ...1.1..
1    ....1.1.
1    .....1.1
 2   1..1....
 2   .1..1...
 2   ..1..1..
 2   ...1..1.
 2   ....1..1
  3 1...1...
  3 .1...1..
  3 ..1...1.
  3 ...1...1
   41....1..
   4.1....1.
   4..1....1
```

Exact Covering as SAT problem

Assign y_i to row i. Obtain conditions:

- Column 1: $y_1 + y_2 + y_3 + y_4 + y_5 + y_6 = 1$
- Column 2: $y_7 + y_8 + y_9 + y_{10} + y_{11} = 1$

  ...
- Column 5: $y_1 + y_7 + y_{12} + y_{16} = 1$
- Column 6: $y_2 + y_8 + y_{13} + y_{17} = 1$

  ...
- Column 12: $y_6 + y_{11} + y_{15} + y_{18} = 1$

Express symmetric function
$S_1(y_1, y_2, y_3, y_4, y_5, y_6) =$
$[y_1 + y_2 + y_3 + y_4 + y_5 + y_6 = 1]$ in CNF.

```
1    1.1.....
1    .1.1....
1    ..1.1...
1    ...1.1..
1    ....1.1.
1    .....1.1
 2   1..1....
 2   .1..1...
 2   ..1..1..
 2   ...1..1.
 2   ....1..1
  3  1...1...
  3  .1...1..
  3  ..1...1.
  3  ...1...1
   41....1..
   4.1....1.
   4..1....1
```

Exact Covering as SAT problem

Assign y_i to row i. Obtain conditions:

- Column 1: $y_1 + y_2 + y_3 + y_4 + y_5 + y_6 = 1$
- Column 2: $y_7 + y_8 + y_9 + y_{10} + y_{11} = 1$

  ...
- Column 5: $y_1 + y_7 + y_{12} + y_{16} = 1$
- Column 6: $y_2 + y_8 + y_{13} + y_{17} = 1$

  ...
- Column 12: $y_6 + y_{11} + y_{15} + y_{18} = 1$

Express symmetric function
$S_1(y_1, y_2, y_3, y_4, y_5, y_6) =$
$[y_1 + y_2 + y_3 + y_4 + y_5 + y_6 = 1]$ in CNF.

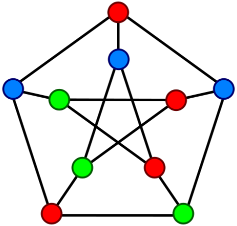
```
1    1.1.....
1    .1.1....
1    ..1.1...
1    ...1.1..
1    ....1.1.
1    .....1.1
 2   1..1....
 2   .1..1...
 2   ..1..1..
 2   ...1..1.
 2   ....1..1
  3  1...1...
  3  .1...1..
  3  ..1...1.
  3  ...1...1
   41....1..
   4.1....1.
   4..1....1
```

One of the simplest ways to express the symmetric Boolean function S_1 as an AND of ORs is to use $1 + \binom{p}{2}$ clauses:

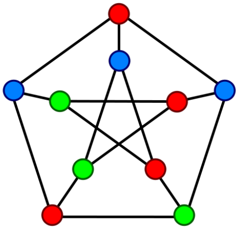
$$S_1(y_1, \ldots, y_p) = (y_1 \vee \cdots \vee y_p) \wedge \bigwedge_{1 \le j < k \le p} (\bar{y}_j \vee \bar{y}_k). \qquad (13)$$

"At least one of the y's is true, but not two." Then (12) becomes, in shorthand,

---



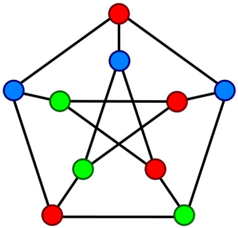
Coloring a graph. The classical problem of coloring a graph with at most d colors is another rich source of benchmark examples for SAT solvers. If the graph has n vertices V, we can introduce nd variables v_j, for $v \in V$ and $1 \leq j \leq d$, signifying that v has color j; the resulting clauses are quite simple:



Coloring a graph. The classical problem of coloring a graph with at most d colors is another rich source of benchmark examples for SAT solvers. If the graph has n vertices V, we can introduce nd variables v_j, for $v \in V$ and $1 \le j \le d$, signifying that v has color j; the resulting clauses are quite simple:

$$(v_1 \vee v_2 \vee \cdots \vee v_d) \text{ for } v \in V \text{ (“every vertex has at least one color”);} \quad (15)$$

$$(\bar{u}_j \vee \bar{v}_j) \text{ for } u \!-\!\!-\, v,\ 1 \le j \le d \text{ (“adjacent vertices have different colors”).} \quad (16)$$



Coloring a graph. The classical problem of coloring a graph with at most d colors is another rich source of benchmark examples for SAT solvers. If the graph has n vertices V, we can introduce nd variables v_j, for $v \in V$ and $1 \le j \le d$, signifying that v has color j; the resulting clauses are quite simple:

$$(v_1 \vee v_2 \vee \cdots \vee v_d) \text{ for } v \in V \text{ (“every vertex has at least one color”);} \qquad (15)$$

$$(\bar{u}_j \vee \bar{v}_j) \text{ for } u \longrightarrow v,\ 1 \le j \le d \text{ (“adjacent vertices have different colors”).} \qquad (16)$$

We could also add $n\binom{d}{2}$ additional so-called *exclusion clauses*

$$(\bar{v}_i \vee \bar{v}_j) \text{ for } v \in V,\ 1 \le i < j \le d \text{ (“every vertex has at most one color”);} \qquad (17)$$

but they're optional, because vertices with more than one color are harmless.

Factoring integers. Next on our agenda is a family of SAT instances with quite a different flavor. *Given an $(m+n)$-bit binary integer $z = (z_{m+n} \ldots z_2 z_1)_2$, do there exist integers $x = (x_m \ldots x_1)_2$ and $y = (y_n \ldots y_1)_2$ such that $z = x \times y$?* For example, if $m = 2$ and $n = 3$, we want to invert the binary multiplication

$$
\begin{array}{ccccc}
 & & y_3 & y_2 & y_1 \\
 & \times & & x_2 & x_1 \\
\hline
 & & a_3 & a_2 & a_1 \\
 & b_3 & b_2 & b_1 & \\
\hline
c_3 & c_2 & c_1 & & \\
\hline
z_5 & z_4 & z_3 & z_2 & z_1
\end{array}
\qquad
\begin{aligned}
(a_3 a_2 a_1)_2 &= (y_3 y_2 y_1)_2 \times x_1 \\
(b_3 b_2 b_1)_2 &= (y_3 y_2 y_1)_2 \times x_2
\end{aligned}
\qquad
\begin{aligned}
z_1 &= a_1 \\
(c_1 z_2)_2 &= a_2 + b_1 \\
(c_2 z_3)_2 &= a_3 + b_2 + c_1 \\
(c_3 z_4)_2 &= b_3 + c_2 \\
z_5 &= c_3
\end{aligned}
\qquad (22)
$$

when the z bits are given. This problem is satisfiable when $z = 21 = (10101)_2$, in the sense that suitable binary values x_1, x_2, y_1, y_2, y_3, a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3 do satisfy these equations. But it's unsatisfiable when $z = 19 = (10011)_2$.

Factoring integers. Next on our agenda is a family of SAT instances with quite a different flavor. *Given an $(m+n)$-bit binary integer $z = (z_{m+n} \ldots z_2 z_1)_2$, do there exist integers $x = (x_m \ldots x_1)_2$ and $y = (y_n \ldots y_1)_2$ such that $z = x \times y$?* For example, if $m = 2$ and $n = 3$, we want to invert the binary multiplication

$$\begin{array}{r} y_3\, y_2\, y_1 \\ \times \quad x_2\, x_1 \\ \hline a_3\, a_2\, a_1 \\ b_3\, b_2\, b_1 \quad \\ \hline c_3\, c_2\, c_1 \quad\quad \\ \hline z_5\, z_4\, z_3\, z_2\, z_1 \end{array} \qquad \begin{aligned} (a_3 a_2 a_1)_2 &= (y_3 y_2 y_1)_2 \times x_1 \\ (b_3 b_2 b_1)_2 &= (y_3 y_2 y_1)_2 \times x_2 \end{aligned} \qquad \begin{aligned} z_1 &= a_1 \\ (c_1 z_2)_2 &= a_2 + b_1 \\ (c_2 z_3)_2 &= a_3 + b_2 + c_1 \\ (c_3 z_4)_2 &= b_3 + c_2 \\ z_5 &= c_3 \end{aligned} \qquad (22)$$

when the z bits are given. This problem is satisfiable when $z = 21 = (10101)_2$, in the sense that suitable binary values $x_1, x_2, y_1, y_2, y_3, a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3$ do satisfy these equations. But it's unsatisfiable when $z = 19 = (10011)_2$.

Express as a Boolean Chain (Section 7.1.2) ...

One such chain, if we identify a_1 with z_1 and c_3 with z_5, is

$$\begin{array}{llllll} z_1 \leftarrow x_1 \wedge y_1, & b_1 \leftarrow x_2 \wedge y_1, & z_2 \leftarrow a_2 \oplus b_1, & s \leftarrow a_3 \oplus b_2, & z_3 \leftarrow s \oplus c_1, & z_4 \leftarrow b_3 \oplus c_2, \\ a_2 \leftarrow x_1 \wedge y_2, & b_2 \leftarrow x_2 \wedge y_2, & c_1 \leftarrow a_2 \wedge b_1, & p \leftarrow a_3 \wedge b_2, & q \leftarrow s \wedge c_1, & z_5 \leftarrow b_3 \wedge c_2, \\ a_3 \leftarrow x_1 \wedge y_3, & b_3 \leftarrow x_2 \wedge y_3, & & & c_2 \leftarrow p \vee q, & \end{array} \qquad (23)$$

using a "full adder" to compute $c_2 z_3$ and "half adders" to compute $c_1 z_2$ and $c_3 z_4$

Express Boolean Chain in CNF using Tseytin encoding.

One such chain, if we identify a_1 with z_1 and c_3 with z_5, is

$$
\begin{array}{llllll}
z_1 \leftarrow x_1 \wedge y_1, & b_1 \leftarrow x_2 \wedge y_1, & z_2 \leftarrow a_2 \oplus b_1, & s \leftarrow a_3 \oplus b_2, & z_3 \leftarrow s \oplus c_1, & z_4 \leftarrow b_3 \oplus c_2, \\
a_2 \leftarrow x_1 \wedge y_2, & b_2 \leftarrow x_2 \wedge y_2, & c_1 \leftarrow a_2 \wedge b_1, & p \leftarrow a_3 \wedge b_2, & q \leftarrow s \wedge c_1, & z_5 \leftarrow b_3 \wedge c_2, \\
a_3 \leftarrow x_1 \wedge y_3, & b_3 \leftarrow x_2 \wedge y_3, & & & c_2 \leftarrow p \vee q, &
\end{array}
\qquad (23)
$$

using a "full adder" to compute $c_2 z_3$ and "half adders" to compute $c_1 z_2$ and $c_3 z_4$

$$
\begin{array}{l}
t \leftarrow u \wedge v \text{ becomes } (u \vee \bar{t}) \wedge (v \vee \bar{t}) \wedge (\bar{u} \vee \bar{v} \vee t); \\
t \leftarrow u \vee v \text{ becomes } (\bar{u} \vee t) \wedge (\bar{v} \vee t) \wedge (u \vee v \vee \bar{t}); \\
t \leftarrow u \oplus v \text{ becomes } (\bar{u} \vee v \vee t) \wedge (u \vee \bar{v} \vee t) \wedge (u \vee v \vee \bar{t}) \wedge (\bar{u} \vee \bar{v} \vee \bar{t}).
\end{array}
\qquad (24)
$$

$$(x_1 \vee \bar{z}_1) \wedge (y_1 \vee \bar{z}_1) \wedge (\bar{x}_1 \vee \bar{y}_1 \vee z_1) \wedge \cdots \wedge (\bar{b}_3 \vee \bar{c}_2 \vee \bar{z}_4) \wedge (b_3 \vee \bar{z}_5) \wedge (c_2 \vee \bar{z}_5) \wedge (\bar{b}_3 \vee \bar{c}_2 \vee z_5)$$

Express Boolean Chain in CNF using Tseytin encoding.

One such chain, if we identify a_1 with z_1 and c_3 with z_5, is

$$\begin{array}{llllll} z_1 \leftarrow x_1 \wedge y_1, & b_1 \leftarrow x_2 \wedge y_1, & z_2 \leftarrow a_2 \oplus b_1, & s \leftarrow a_3 \oplus b_2, & z_3 \leftarrow s \oplus c_1, & z_4 \leftarrow b_3 \oplus c_2, \\ a_2 \leftarrow x_1 \wedge y_2, & b_2 \leftarrow x_2 \wedge y_2, & c_1 \leftarrow a_2 \wedge b_1, & p \leftarrow a_3 \wedge b_2, & q \leftarrow s \wedge c_1, & z_5 \leftarrow b_3 \wedge c_2, \\ a_3 \leftarrow x_1 \wedge y_3, & b_3 \leftarrow x_2 \wedge y_3, & & & c_2 \leftarrow p \vee q, & \end{array} \tag{23}$$

using a "full adder" to compute $c_2 z_3$ and "half adders" to compute $c_1 z_2$ and $c_3 z_4$

$$\begin{array}{ll} t \leftarrow u \wedge v & \text{becomes } (u \vee \bar{t}) \wedge (v \vee \bar{t}) \wedge (\bar{u} \vee \bar{v} \vee t); \\ t \leftarrow u \vee v & \text{becomes } (\bar{u} \vee t) \wedge (\bar{v} \vee t) \wedge (u \vee v \vee \bar{t}); \\ t \leftarrow u \oplus v & \text{becomes } (\bar{u} \vee v \vee t) \wedge (u \vee \bar{v} \vee t) \wedge (u \vee v \vee \bar{t}) \wedge (\bar{u} \vee \bar{v} \vee \bar{t}). \end{array} \tag{24}$$

$$(x_1 \vee \bar{z}_1) \wedge (y_1 \vee \bar{z}_1) \wedge (\bar{x}_1 \vee \bar{y}_1 \vee z_1) \wedge \cdots \wedge (\bar{b}_3 \vee \bar{c}_2 \vee \bar{z}_4) \wedge (b_3 \vee \bar{z}_5) \wedge (c_2 \vee \bar{z}_5) \wedge (\bar{b}_3 \vee \bar{c}_2 \vee z_5)$$

How do we obtain a CNF formula satisfiable iff $z = (10101)_2$ can be factored into x and y?

Guess that Boolean function!

$f(x_1, x_2, \ldots, x_N)$ is an unknown Boolean function that evaluates to 1 or 0 on the tabulated points.

VALUES TAKEN ON BY AN UNKNOWN FUNCTION

Cases where $f(x) = 1$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9					…						x_{20}
1	1	0	0	1	0	0	1	0	0	0	0	1	1	1	1	1	1	0	1
1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1
0	1	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	1	1
0	1	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	1	0
0	1	1	0	0	0	1	0	1	0	0	0	1	0	1	1	1	0	0	0
0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	1	1	1	0	0
1	1	0	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0
1	0	0	0	1	0	1	0	0	1	1	0	0	1	1	1	1	1	0	0
1	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	1	1	1	0	1	1	1	1	1	0	1	0	1	0
0	1	1	0	0	0	1	1	1	0	1	1	0	0	0	1	0	0	1	1
1	0	0	1	1	0	1	1	0	0	1	0	0	0	1	0	0	1	0	1
0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	0
0	1	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1
0	1	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	1	0	1

Cases where $f(x) = 0$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9					…						x_{20}
1	0	1	0	1	1	0	1	1	1	1	1	1	0	0	0	0	1	0	1
0	1	0	0	0	1	0	1	1	0	0	0	1	0	1	0	0	0	1	0
1	0	1	1	1	0	1	1	0	1	0	0	1	0	1	0	1	0	0	1
1	0	1	0	1	0	1	0	1	1	1	1	1	1	0	1	1	1	0	0
0	1	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	0
0	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	1	0	0
1	1	1	1	0	0	0	1	1	1	0	1	1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	1	1
1	1	0	0	1	1	1	0	0	0	1	0	1	1	0	1	0	0	1	1
0	1	1	0	1	0	0	1	0	1	0	1	1	0	1	0	1	0	0	1
1	1	1	0	0	0	0	1	0	0	1	1	0	1	1	0	0	1	0	0
0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0	0
0	0	1	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0
1	1	0	0	1	0	0	1	0	0	1	1	1	0	0	1	1	1	0	1
1	1	0	0	1	1	1	0	0	0	1	0	0	1	0	0	1	0	0	1
1	0	1	1	0	0	1	1	1	1	1	0	1	1	1	1	1	0	0	1

almost immediately that a very simple formula is consistent with all of the data:

$$f(x_1, \ldots, x_{20}) = \bar{x}_2\bar{x}_3\bar{x}_{10} \vee \bar{x}_6\bar{x}_{10}\bar{x}_{12} \vee x_8\bar{x}_{13}\bar{x}_{15} \vee \bar{x}_8 x_{10}\bar{x}_{12}. \qquad (27)$$

Guess that Boolean function!

$f(x_1, x_2, \ldots, x_N)$ is an unknown Boolean function that evaluates to 1 or 0 on the tabulated points.

VALUES TAKEN ON BY AN UNKNOWN FUNCTION

Cases where $f(x) = 1$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9					$\cdots$						x_{20}
1	1	0	0	1	0	0	1	0	0	0	0	1	1	1	1	1	1	0	1
1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1
0	1	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	1	1
0	1	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	1	0
0	1	1	0	0	0	1	0	1	0	0	0	1	0	1	1	1	0	0	0
0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	1	1	1	0	0
1	1	0	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0
1	0	0	0	1	0	1	0	0	1	1	0	0	1	1	1	1	1	0	0
1	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	1	1	1	0	1	1	1	1	1	0	1	0	1	0
0	1	1	0	0	0	1	1	1	0	1	1	0	0	0	1	0	0	1	1
1	0	0	1	1	0	1	1	0	0	1	0	0	0	1	0	0	1	0	1
0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	0
0	1	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	0	1	1
0	1	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	1	0	1

Cases where $f(x) = 0$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9					$\cdots$						x_{20}
1	0	1	0	1	1	0	1	1	1	1	1	1	0	0	0	0	1	0	1
0	1	0	0	0	1	0	1	1	0	0	0	1	0	1	0	0	0	1	0
1	0	1	1	1	0	1	1	0	1	0	0	1	0	1	0	1	0	0	1
1	0	1	0	1	0	1	0	1	1	1	1	1	1	0	1	1	1	0	0
0	1	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	0
0	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	1	0	0
1	1	1	1	0	0	0	1	1	1	0	1	1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	1	1
1	1	0	0	1	1	1	0	0	0	1	0	1	1	0	1	0	0	1	1
0	1	1	0	1	0	0	1	0	1	0	1	1	0	1	0	1	0	0	1
1	1	1	0	0	0	0	1	0	0	1	1	0	1	1	0	0	1	0	0
0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0	0
0	0	1	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0
1	1	0	0	1	0	0	1	0	0	1	1	1	0	0	1	1	1	0	1
1	1	0	0	1	1	1	0	0	0	1	0	0	1	0	0	1	0	0	1
1	0	1	1	0	0	1	1	1	1	1	0	1	1	1	1	1	0	0	1

almost immediately that a very simple formula is consistent with all of the data:

$$f(x_1, \ldots, x_{20}) = \bar{x}_2\bar{x}_3\bar{x}_{10} \vee \bar{x}_6\bar{x}_{10}\bar{x}_{12} \vee x_8\bar{x}_{13}\bar{x}_{15} \vee \bar{x}_8 x_{10}\bar{x}_{12}. \qquad (27)$$

Problem: find a DNF formula on M terms that agrees with the tabulated data.

This formula was discovered by constructing clauses in $2MN$ variables $p_{i,j}$ and $q_{i,j}$ for $1 \le i \le M$ and $1 \le j \le N$, where M is the maximum number of terms allowed in the DNF (here $M = 4$) and where

$$p_{i,j} \;=\; [\text{term } i \text{ contains } x_j], \qquad q_{i,j} \;=\; [\text{term } i \text{ contains } \bar{x}_j]. \tag{28}$$

If the function is constrained to equal 1 at P specified points, we also use auxiliary variables $z_{i,k}$ for $1 \le i \le M$ and $1 \le k \le P$, one for each term at every such point.

This formula was discovered by constructing clauses in $2MN$ variables $p_{i,j}$ and $q_{i,j}$ for $1 \leq i \leq M$ and $1 \leq j \leq N$, where M is the maximum number of terms allowed in the DNF (here $M = 4$) and where

$$p_{i,j} \;=\; [\text{term } i \text{ contains } x_j], \qquad q_{i,j} \;=\; [\text{term } i \text{ contains } \bar{x}_j]. \tag{28}$$

If the function is constrained to equal 1 at P specified points, we also use auxiliary variables $z_{i,k}$ for $1 \leq i \leq M$ and $1 \leq k \leq P$, one for each term at every such point.

Table 2 says that $f(1,1,0,0,\ldots,1) = 1$, and we can capture this specification by constructing the clause

$$(z_{1,1} \vee z_{2,1} \vee \cdots \vee z_{M,1}) \tag{29}$$

together with the clauses

$$(\bar{z}_{i,1} \vee \bar{q}_{i,1}) \wedge (\bar{z}_{i,1} \vee \bar{q}_{i,2}) \wedge (\bar{z}_{i,1} \vee \bar{p}_{i,3}) \wedge (\bar{z}_{i,1} \vee \bar{p}_{i,4}) \wedge \cdots \wedge (\bar{z}_{i,1} \vee \bar{q}_{i,20}) \tag{30}$$

for $1 \leq i \leq M$. Translation: (29) says that at least one of the terms in the DNF must evaluate to true; and (30) says that, if term i is true at the point $1100\ldots1$, it cannot contain $\bar{x}_1$ or $\bar{x}_2$ or x_3 or x_4 or $\cdots$ or $\bar{x}_{20}$.

This formula was discovered by constructing clauses in $2MN$ variables $p_{i,j}$ and $q_{i,j}$ for $1 \leq i \leq M$ and $1 \leq j \leq N$, where M is the maximum number of terms allowed in the DNF (here $M = 4$) and where

$$p_{i,j} = [\text{term } i \text{ contains } x_j], \qquad q_{i,j} = [\text{term } i \text{ contains } \bar{x}_j]. \tag{28}$$

If the function is constrained to equal 1 at P specified points, we also use auxiliary variables $z_{i,k}$ for $1 \leq i \leq M$ and $1 \leq k \leq P$, one for each term at every such point.

Table 2 says that $f(1, 1, 0, 0, \ldots, 1) = 1$, and we can capture this specification by constructing the clause

$$(z_{1,1} \vee z_{2,1} \vee \cdots \vee z_{M,1}) \tag{29}$$

together with the clauses

$$(\bar{z}_{i,1} \vee \bar{q}_{i,1}) \wedge (\bar{z}_{i,1} \vee \bar{q}_{i,2}) \wedge (\bar{z}_{i,1} \vee \bar{p}_{i,3}) \wedge (\bar{z}_{i,1} \vee \bar{p}_{i,4}) \wedge \cdots \wedge (\bar{z}_{i,1} \vee \bar{q}_{i,20}) \tag{30}$$

for $1 \leq i \leq M$. Translation: (29) says that at least one of the terms in the DNF must evaluate to true; and (30) says that, if term i is true at the point $1100 \ldots 1$, it cannot contain $\bar{x}_1$ or $\bar{x}_2$ or x_3 or x_4 or $\cdots$ or $\bar{x}_{20}$.

Table 2 also tells us that $f(1, 0, 1, 0, \ldots, 1) = 0$. This specification corresponds to the clauses

$$(q_{i,1} \vee p_{i,2} \vee q_{i,3} \vee p_{i,4} \vee \cdots \vee q_{i,20}) \tag{31}$$

for $1 \leq i \leq M$. (Each term of the DNF must be zero at the given point; thus either $\bar{x}_1$ or x_2 or $\bar{x}_3$ or x_4 or $\cdots$ or $\bar{x}_{20}$ must be present for each value of i.)