Two incomplete methods: GSAT and Unit propagation

Two methods for finding a model of a formula

Incomplete=they may fail

Two different kinds of incompleteness (more later)

Both work on CNFs

GSAT

Works on CNFs: $\{C_1, ..., C_m\}$

Find a model = find an interpretation satisfying all clauses

Principle:

> if I satisfies 3 clauses and I' 5, the latter is "closer" to be a model than the former

works by picking an interpretation and iteratively trying to increase the number of satisfied clauses

---

Increasing satisfied clauses

Given an interpretation I, try to change the value of a single variable
(if true make it false and vice versa)

Does this change increase the number of satisfied clauses?

Try this for all variables

Best change

given I, change the value of a variable

determine the number of satisfied clauses

do this for all variables

pick the best, in terms of number of satisfied clauses

GSAT: the algorithm

```
1. pick a random interpretation $I$
2. for every variable $x$
    1. $I'$=same as $I$ but for the value of $x$
    2. compute number of satisfied clauses
3. $I$=interpretation $I'$ maximizing this number
4. if all clauses are satisfied by $I'$, stop
5. go to 1
```

(one point missing)

GSAT: incompleteness

limit=we change a variable at time

possible situation: $I \rightarrow I' \rightarrow I'' \rightarrow I$

- none of I, I', I'' are models
- all interpretations that differ from them by one variable satisfy less clauses than them
- a model exist, but differs from them by more than one variable

we keep cycling over them because the other close interpretations satisfy less clauses

we are stuck in a *local maximum*
(w.r.t. the surrounding interpretations, we have reached a maximum, but this is not the overall maximum)

but a model exists

GSAT: restarts

after a fixed number of iterations (say, 10000), GSAT stops and picks another random interpretation

try this for a fixed number of times (say, 10)

still, we may end up in a local maximum every time

GSAT is incomplete

GSAT: incompleteness

1. how it is incomplete?
    - if GSAT finds a model, the formula is satisfiable
    - if GSAT does not find a model, the formula may be satisfiable or not
2. when is it incomplete?

    in general, we cannot tell in advance whether GSAT will work on a particular formula
    (only way is: try it and see if it finds a model)

Unit Propagation

Still works on CNFs: $\{C_1, ..., C_m\}$

Principle:

> if a clause contains a single literal, every model makes that literal true

replace every occurrence of that variable with its value

obvious on an example (next)

Unit Propagation: example

$\{x, \neg x \lor y, x \lor \neg z, y \lor z, y \lor \neg z\}$

every model of this set has x=*true*
(otherwise, the clause x would be false)

model=interpretation that satisfies **all** clauses

(cont.)

Unit Propagation: example

$\{x, \neg x \lor y, x \lor \neg z, y \lor z, y \lor \neg z\}$

every model of this set has x=*true*

under this assumption:

- $\neg x \lor y$ = *false* $\lor\ y = y$
- $x \lor \neg z$ = *true*

Unit + Propagation

from *unit* clauses to truth values

propagate truth values to clauses

repeat, if needed

Unit propagation: full example

$\{x, \neg x \lor \neg y, x \lor \neg z, y \lor z, y \lor \neg z\}$

x=*true*

replace x=*true* in the set and simplify:

$\{x, \neg x \lor \neg y, x \lor \neg z, y \lor z, y \lor \neg z\}$ =
$\{$*true*, $\neg$*true* $\lor \neg y$, *true* $\lor \neg z, y \lor z, y \lor \neg z\}$
$\{$*true*, *false* $\lor \neg y$, *true* $\lor \neg z, y \lor z, y \lor \neg z\}$
$\{\neg y, y \lor z, y \lor \neg z\}$

simplifications based on:

- *false* $\lor$ *something* = *something*
  (remove false literals from clauses)
- *true* $\land$ *something* = *something*
  (remove true clauses from the set)

(cont.)

Unit propagation: full example

so far: x=*true* and the set simplifies to:

$\{\neg y, y \lor z, y \lor \neg z\}$

all models has y false

set y=*false* and simplify:

$\{\neg y, y \lor z, y \lor \neg z\}$ =
$\{$*true*, *false* $\lor z$, *false* $\lor \neg z\}$ =
$\{z, \neg z\}$

contradiction

the set is unsatisfiable

next: formal definition of algorithm

Partial interpretations

accumulate truth values in a partial interpretation *I*

partial interpretation=truth evaluation of *some* variables

initially, no variable is evaluated

(full) interpretations are a particular case (all variables evaluated)

Unit propagation: algorithm

```
1. $I$=empty partial interpretation
2. for every unit clause $v$ or $\neg v$:
     ○ add $v$=true or $v$=false, respectively, to $I$
     ○ replace $v$ with its value in the set of clauses and simplify
3. if the set contains some other unit clause, go to 2
```

stop on contradiction in the partial interpretation

Unit propagation, in practice

try to make all propagations at the same time

```
1. $I$=empty partial interpretation
2. $I'$=empty partial interpretation
3. for every unit clause $v$ or $\neg v$:
     ○ add $v$=true or $v$=false, respectively, to $I'$
4. if $I' = \emptyset$ stop
5. replace each $v$ in $I'$ with its value in the set of clauses and simplify
6. $I' = I \cup I'$
7. go to 2
```

stop on contradiction in *I* or *I'*

Unit propagation on a consistent formula

$\{x \lor y \lor \neg z, y, \neg x \lor w, \neg y \lor w\}$

$y = \mathit{true} \rightarrow I$

formula becomes:

$\{\cancel{x \lor y \lor \neg z}, \cancel{y}, \neg x \lor w, \cancel{\neg} y \lor w\} = \{\neg x \lor w, w\}$

$w = \mathit{true} \rightarrow I$

replace in formula and simplify:

$\{\cancel{\neg x \lor w}, \cancel{w}\} = \emptyset$

all clauses are satisfied by $I = \{y = \mathit{true}, w = \mathit{true}\}$

formula is satisfied

model=any extension of *I* to all variables
(e.g., $x = \mathit{false}$, $y = \mathit{true}$, $z = \mathit{true}$, $w = \mathit{true}\}$)

Unit propagation on an inconsistent formula

$\{x \lor y, x \lor \neg y, \neg z \lor \neg x \lor y, \neg x \lor \neg y, z\}$

add $z = \mathit{true}$ to *I*

replace *z* with its truth value and simplify:

$\{x \lor y, x \lor \neg y, \cancel{\neg z} \lor \neg x \lor y, \neg x \lor \neg y, \cancel{z}\} =$

$\{x \lor y, x \lor \neg y, \neg x \lor y, \neg x \lor \neg y\}$

no unit clause

algorithm stops

(cont.)

Incompleteness of Unit propagation

formula $\{x \vee y, x \vee \neg y, \neg x \vee y, \neg x \vee \neg y\}$ is inconsistent

yet, unit propagation did not reach contradiction

happened also in the previous example, with a consistent formula

> if unit propagation does not reach contradiction, we cannot say whether formula is consistent or no

unit propagation is incomplete

Two kinds of incompleteness

GSAT and Unit Propagation are incomplete in two different ways:

GSAT

- if it finds a model, formula is surely satisfiable
- otherwise, it may be unsatisfiable or not
- no general way to predict if it works on a formula

Unit propagation

- if it reaches contradiction, formula is surely unsatisfiable
- otherwise, it may be satisfiable or not
- complete on a specific class of formulae

last point on next page

Horn formulae

Horn clause: a clause containing **at most** a positive literal

= contains zero or one positive literal

examples:

- $x \vee \neg y$
- $\neg y \vee z \vee \neg w$
- y
- $\neg x$
- $\neg x \vee \neg y \vee \neg z$

(last two examples: zero positive literal, still Horn)

Horn formula: set of Horn clauses

Unit propagation on Horn formulae

unit propagation is complete on Horn formulae

in general, it is not because it might not reach contradiction even if formula is unsatisfiable

never the case on Horn formulae

unit propagation does not reach contradiction on an Horn formula ⇒ formula is satisfiable

Unit propagation complete on Horn formulae

assume unit propagation does not reach contradiction

let I be the partial interpretation and F the set of remaining clauses

- F does not contain any variable in I
  (otherwise, F would have been simplified)
- F does not contain any unit clause
  (otherwise the algorithm would not terminate)

all clauses are made of at least two literals, none of them evaluated in I

at most one positive=at least one negative

setting all variables to false, all clauses are satisfied

conclusion:

if unit propagation does not reach contradiction, the set is satisfiable
(model is I plus all other variables set to false

Up on Horn CNFs: example

$\{ x_1 \vee \neg x_2 \vee \neg x_3,\ \neg x_1 \vee x_2 \vee \neg x_3,\ x_4 \vee \neg x_5,\ x_5,\ \neg x_1 \vee x_3 \vee \neg x_4,\ \neg x_2 \vee \neg x_4 \vee \neg x_5 \}$

(cont.)

Up on Horn CNFs: example (2)

$\{ x_1 \vee \neg x_2 \vee \neg x_3,\ \neg x_1 \vee x_2 \vee \neg x_3,\ x_4 \vee \neg x_5,\ x_5,\ \neg x_1 \vee x_3 \vee \neg x_4,\ \neg x_2 \vee \neg x_4 \vee \neg x_5 \}$

x_5 is a unit clause

set x_5=*true*

Up on Horn CNFs: example (3)

replace x_5 with *true*

$\{ x_1 \vee \neg x_2 \vee \neg x_3,\ \neg x_1 \vee x_2 \vee \neg x_3,\ x_4 \vee \cancel{\neg x_5},\ \cancel{x_5},\ \neg x_1 \vee x_3 \vee \neg x_4,\ \neg x_2 \vee \neg x_4 \vee \cancel{\neg x_5} \}$

simplify:

$\{ x_1 \vee \neg x_2 \vee \neg x_3,\ \neg x_1 \vee x_2 \vee \neg x_3,\ x_4,\ \neg x_1 \vee x_3 \vee \neg x_4,\ \neg x_2 \vee \neg x_4 \}$

(cont.)

Up on Horn CNFs: example (4)

$\{ x_1 \vee \neg x_2 \vee \neg x_3,\ \neg x_1 \vee x_2 \vee \neg x_3,\ x_4,\ \neg x_1 \vee x_3 \vee \neg x_4,\ \neg x_2 \vee \neg x_4 \}$

set x_4=*true*

replace:

$\{ x_1 \vee \neg x_2 \vee \neg x_3,\ \neg x_1 \vee x_2 \vee \neg x_3,\ \cancel{x_4},\ \neg x_1 \vee x_3 \vee \cancel{\neg x_4},\ \neg x_2 \vee \cancel{\neg x_4} \}$

simplify:

$\{ x_1 \vee \neg x_2 \vee \neg x_3,\ \neg x_1 \vee x_2 \vee \neg x_3,\ \neg x_1 \vee x_3,\ \neg x_2 \}$

(cont.)

Up on Horn CNFs: example (5)

$\{ x_1 \vee \neg x_2 \vee \neg x_3,\ \neg x_1 \vee x_2 \vee \neg x_3,\ \neg x_1 \vee x_3,\ \neg x_2 \}$

clause $\neg x_2$ is unary

set x_2=*false*

replace:

$\{ \cancel{x_1 \vee \neg x_2 \vee \neg x_3},\ \neg x_1 \vee \cancel{x_2} \vee \neg x_3,\ \neg x_1 \vee x_3,\ \cancel{\neg x_2} \}$

simplify:

$\{ \neg x_1 \vee \neg x_3,\ \neg x_1 \vee x_3 \}$

Up on Horn CNFs: example (6)

$\{ \neg x_1 \vee \neg x_3,\ \neg x_1 \vee x_3 \}$

no unit clause

propagation stops

set remaining variables to *false*:
all clauses are binary and Horn,
so they contain at least a negative literal each
setting all variables to *false* makes them true

result is assignment $\{ x_1 = \mathit{false},\ x_2 = \mathit{false},\ x_3 = \mathit{false},\ x_4 = \mathit{true},\ x_5 = \mathit{true} \}$

this is a model of the original CNF

Comparison GSAT-UP, in practice

GSAT
    good in practice, no way to predict if it works on a specific formula
Unit propagation
    complete on a specific kind of formulae (Horn formulae)

Running time

GSAT
    can find a model at the first step, or after 100000
    if it does not, running time is maximum number of flips per maximum number of tries
Unit propagation
    runs in linear time
    (proof: see next)

Unit propagation: dumb version

1. $I=\emptyset$
2. scan formula for unit clauses, collecting values in I'
3. scan formula, replacing variables in I' and simplifying
4. $I=I \cup I'$
5. if $I' \neq \emptyset$ goto 2

worst case?

Worst case for UP, dumb version

$\{ x_1 \vee \neg x_2, x_2 \vee \neg x_3, \ldots x_{98} \vee \neg x_{99}, x_{99} \vee \neg x_{100}, x_{100} \}$

first scan produces $I'=\{x_{100}=true\}$

scan for replacing removes x_{100} and simplifies $x_{99} \vee \neg x_{100}$ to x_{99}

second scan produces $I'=\{x_{99}=true\}$

scan for replacing removes x_{99} and simplifies $x_{98} \vee \neg x_{99}$ to x_{98}

etc.

Worst case, running time

- two complete scans of the set for x_{100}
- two scans up to the second-last clause for x_{99}
- ...

100+99+98...=?

like a triangle, area is base*height/2

cost is in the order of n^2, where n is the number of variables

Better algorithm?

the problem with the previous set was:

we need to scan the whole set just to find out that x_{100} was only contained in the last two clauses

the solution:

create a data structure for finding the clauses that contain a given variable

---

Data structure

array of n lists, where n is the number of variables

each list contains the clauses that contain a variables

first, assign number to clauses:

$$\begin{array}{ccccc} \{\, x_1 \lor \neg x_2, & x_2 \lor \neg x_3, & \ldots\ x_{98} \lor \neg x_{99}, & x_{99} \lor \neg x_{100}, & x_{100}\ \} \\ C_1 & C_2 & \ldots\ C_{98} & C_{99} & C_{100} \end{array}$$

second, scan the set and add each clauses to the lists

x_1	: C_1
x_2	: C_1 C_2
...	
x_{99}	: C_{98} C_{99}
x_{100}	: C_{99} C_{100}

Data structure, on the example

$$\begin{array}{ccccc} \{\, x_1 \lor \neg x_2, & x_2 \lor \neg x_3, & \ldots\ x_{98} \lor \neg x_{99}, & x_{99} \lor \neg x_{100}, & x_{100}\ \} \\ C_1 & C_2 & \ldots\ C_{98} & C_{99} & C_{100} \end{array}$$

x_1	: C_1
x_2	: C_1 C_2
...	
x_{99}	: C_{98} C_{99}
x_{100}	: C_{99} C_{100}

scan the set, get x_{100}=*true*

the list of x_{100} contains C_{99} and C_{100}

simplify these clauses

C_{99} becomes unary, set x_{99}=*true*

repeat

the data structure eliminates the need for scanning the set of clauses for simplifying

---

Unit propagation, with the data structure

```
1. build the array of lists
2. $I=\emptyset$
3. scan formula for unit clauses, collecting values in $I'$
4. if $I'=\emptyset$ end
5. $I''=\emptyset$
6. for each $x_i$=value in $I'$:
     ○ for each clause $C_j$ in the list of $x_i$:
          ▪ simplify the clause
          ▪ if it becomes unary, set its variable in $I''$
7. $I=I \cup I'$
8. $I'=I''$
9. go to 4
```

Cost of the new algorithm

building the data structure is linear:
scan the set once, adding each clause to the lists of all variables it contains

every time we get a new value, we can immediately find the clauses that need simplification

linear for 3CNF (in general, for clauses of fixed length)

can be made linear in general

(use two lists, one for the clauses containing x_i and one for the clauses containing $\neg x_i$; use a counter for the number of literals left in clauses)