Informatics 1

SATisfaction revision

Michael Fourman

$$\begin{array}{cc} 1 & B \\ \uparrow & \uparrow \\ 0 & A \end{array}$$

$0 \leq 1$

$\bot \leq \top$

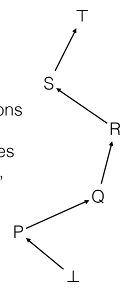
for booleans

$A \rightarrow B = \top$

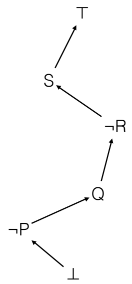
iff

$A \leq B$

If we have a chain of n-1 implications
between n variables
we can draw the line in n+1 places
making any number, from 0 to n,
of these variables true.



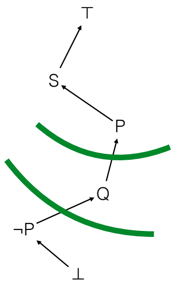
If some of the variables are negated we can do the same (but making the negated variables false when they fall above the line and true when they fall below)



If a variable appears together with its negation, we have to draw the line between them.

Here, P must be true.

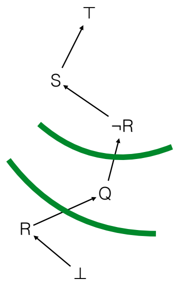
$(\neg P \to P) \to P$
is a tautology



If a variable appears together with its negation, we have to draw the line between them.

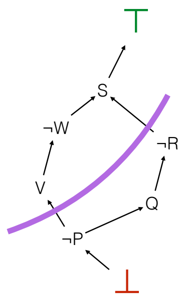
Here, R must be false.

$(R \to \neg R) \to \neg R$
is a tautology



The same trick works if our implications form a partial order.
But we have more options since we can draw a wavy line.

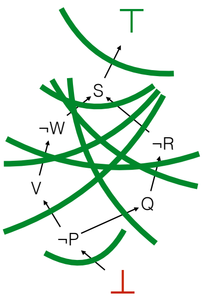
The **arrow rule** says that, whenever our line cuts an arrow, then the head must be on the side of true and the tail on the side of false.



The same trick works if our implications form a partial order.
But we have more options since we can draw a wavy line.

Not all of the valid truth assignments are represented in this diagram.

How many are missing?



Clausal Form

Clausal form is a set of sets of literals

$\{$ {¬A,C}, {¬B,D}, {¬E,B}, {¬E,A}, {A,E}, {E,B},{¬B, ¬C, ¬D} $\}$

A (partial) truth assignment makes a clause true
iff it makes at least one of its literals true
(so it can never make the empty clause {} true)

A (partial) truth assignment makes a clausal form true
iff it makes all of its clauses true
(so the empty clausal form $\{\}$ is always true).

2-SAT

A clausal form with at most two literals per clause.

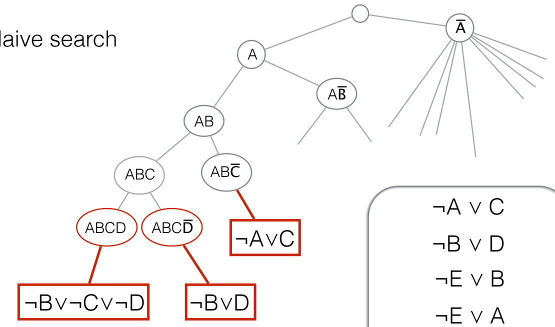
Corresponds to a conjunction of implications.

We can draw the directed graph and count the satisfying valuations.

When 3 or more are involved, satisfaction gets complicated.

In general, we must search for satisfaction.

Naive search

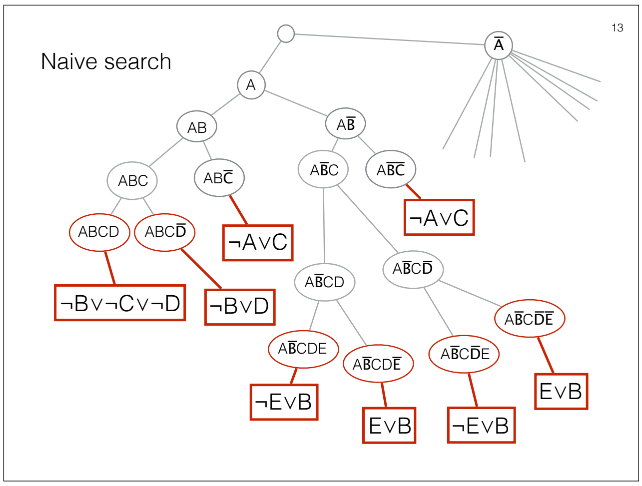


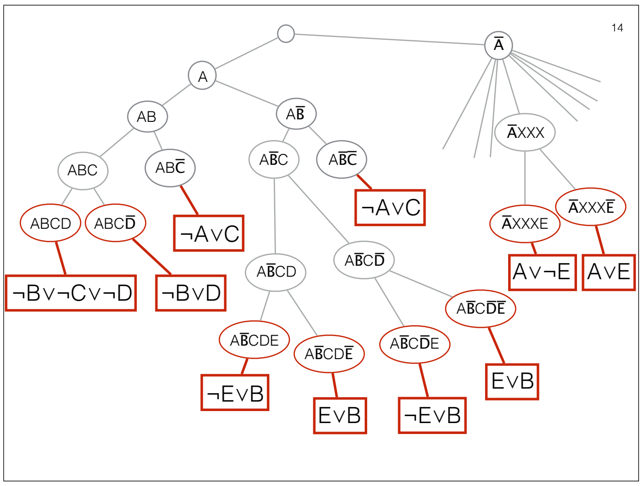
¬A ∨ C
¬B ∨ D
¬E ∨ B
¬E ∨ A
A ∨ E
E ∨ B
¬B ∨ ¬C ∨ ¬D

Naive search

```
function Naive(V, Φ)
  if V ⊨ ¬Φ then return false;
  if V ⊨  Φ  then return true;
  otherwise,
    choose an A mentioned in Φ
          but not mentioned in V
      return Naive(V^A, Φ)
             ||
              Naive(V^¬A, Φ)


                  (call Naive(∅, Φ))
```

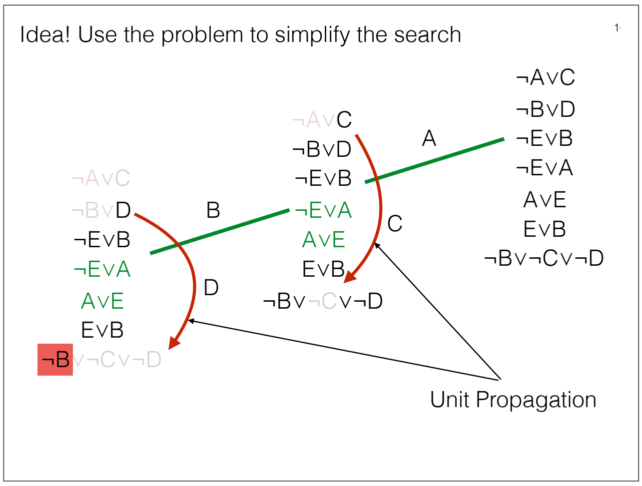

13
Naive search
$\bar{A}$
A
AB
$A\bar{B}$
ABC
$AB\bar{C}$
$A\bar{B}C$
$A\bar{B}\bar{C}$
ABCD
$ABC\bar{D}$
¬A∨C
¬A∨C
¬B∨¬C∨¬D
¬B∨D
$A\bar{B}CD$
$A\bar{B}C\bar{D}$
$A\bar{B}C\bar{D}\bar{E}$
$A\bar{B}CDE$
$A\bar{B}CD\bar{E}$
$A\bar{B}C\bar{D}E$
E∨B
¬E∨B
E∨B
¬E∨B


14
A
Ā
AB
AB̄
ĀXXX
ABC
ABC̄
AB̄C
AB̄C̄
ĀXXXE
ĀXXXĒ
ABCD
ABCD̄
¬A∨C
¬A∨C
A∨¬E
A∨E
¬B∨¬C∨¬D
¬B∨D
AB̄CD
AB̄CD̄
AB̄CD̄Ē
AB̄CDE
AB̄CDĒ
AB̄CD̄E
EvB
¬E∨B
E∨B
¬E∨B

Davis Putnam Logemann Loveland (DPLL)

```
function Naive(V, Φ)
  if V ⊨ ¬Φ then return false;
  if V ⊨  Φ  then return true;
  if V, C ⊨ X,
      where X is literal and clause C ∈ Φ
      return Naive(V^X, Φ)
  otherwise,
    choose an A mentioned in Φ
          but not mentioned in V
    return Naive(V^A, Φ)
           ||
            Naive(V^¬A, Φ)
```

(call Naive(∅, Φ))


Idea! Use the problem to simplify the search
¬A∨C
¬B∨D
¬E∨B
¬E∨A
A∨E
E∨B
¬B∨¬C∨¬D
A
B
C
D
Unit Propagation

Davis Putnam Logemann Loveland (DPLL)
implementation - add V to Φ
unit propagation

```
function DPLL(Φ)
  if Φ is a consistent set of literals
     then return true;
  if Φ contains an empty clause
     then return false;
  for every unit clause l in Φ
     Φ ← unit-propagate(l, Φ);
   l ← choose-literal(Φ);
  return DPLL(Φ ∪ {l}) or DPLL(Φ ∪ {not(l)});
```

Clausal form is a set of sets of literals

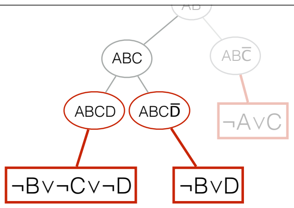
$$\mathbf{X} = \left\{ x_0, x_1, \ldots , x_{n-1} \right\}$$

Resolution rule for clauses

$$\frac{\mathbf{X} \qquad \mathbf{Y}}{(\mathbf{X} \cup \mathbf{Y}) \setminus \{ \neg A, A \}} \quad \text{where } \neg A \in \mathbf{X}, A \in \mathbf{Y}$$

If either X or Y is a singleton then this is just unit propagation.

So, *resolution is a generalisation of unit propagation.*
Search is no longer needed



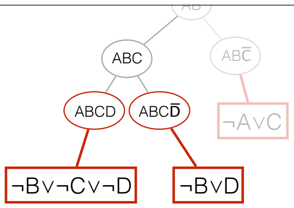
Premises

$$\frac{\neg B \vee \neg C \vee \neg \mathbf{D} \qquad \neg B \vee \mathbf{D}}{\neg \mathbf{B} \vee \neg \mathbf{C}}$$

Conclusion

A valid inference	Any assignment of truth values that makes all the premises true will make the conclusion true.

The conclusion follows from the premises



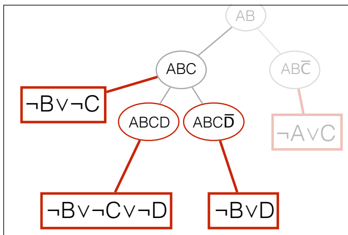
Premises

$\neg B \vee \neg C \vee \neg D \qquad \neg B \vee D$

$\neg B \vee \neg C$

Conclusion

For any valid inference	Any assignment of truth values that makes the conclusion false will make at least one of the premises false.



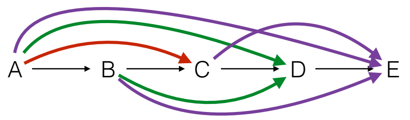
Premises

$$\frac{\neg B \vee \neg C \vee \neg \mathbf{D} \qquad \neg B \vee \mathbf{D}}{\neg \mathbf{B} \vee \neg \mathbf{C}}$$

Conclusion

*A **special property** of this inference*

If some assignment XYZ of values for ABC makes the conclusion false then the assignments XYZ**D** and XYZ$\bar{\mathbf{D}}$ each make one or other of the two premises false.



	B	C	D
¬A∨B	¬A∨C	¬A∨D	¬A∨E
¬B∨C		¬B∨D	¬B∨E
¬C∨D			¬C∨E
¬D∨E			

We keep adding clauses obtained by resolution.
Davis Putnam - choose a variable then add all instances.
Different orders for resolution will give the same results.

Davis Putnam

Take a collection $\mathcal{C}$ of clauses.

For each propositional letter, A
For each pair $(X, Y) \mid X \in \mathcal{C} \wedge Y \in \mathcal{C} \wedge \mathsf{A} \in X \wedge \neg\mathsf{A} \in Y$
if $\mathcal{R}(X, Y, \mathsf{A}) = \{\}$ return UNSAT
if $\mathcal{R}(X, Y, \mathsf{A})$ is consistent $\mathcal{C} := \mathcal{C} \cup \{ \mathcal{R}(X, Y) \}$
return SAT

Where $\mathcal{R}(X, Y, \mathsf{A}) = X \cup Y \setminus \{ \mathsf{A}, \neg\mathsf{A} \}$

Heuristic: start with variables that occur seldom.