Informatics 1

Michael Fourman

Lecture 19 Searching for Satisfaction

...aph
...haden.
...rtune and
...3 JAS. I
...r spent
...ureate
...adise
...Hoo
...ER

...Princeton Rev. ...ion in lite... X
...r literacy greatly in...
...efined. 1893 Athenæum...
...Edgar to trace the gradual p...
...m illiteracy to literacy.

Literal (li·tĕrăl), *a.* and *sb.* Fo...
teral, (5, 6 lyt(t)urall, 6 lyt(t)ar...
lit(t)erall, 4- **literal.** [a. OF. *li*...
teral); ad. L. *litterālis*, f. *littera* LETTE...

A. *adj.*

1. Of or pertaining to letters of the a...
of the nature of letters, alphabetical; †...
by letters, written. † Of a verse = A...
c 1475 *Partenay* 6605 And so hau...
entent, With litterall carectes fo...
Ess. Poesie (Arb.) 63 Be Lit...
pairt of zour lyne, sall ry...
...ne rynnis vpon F...
...mden) 15 Wh...
...ment of...

$\wedge \vee$ 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).

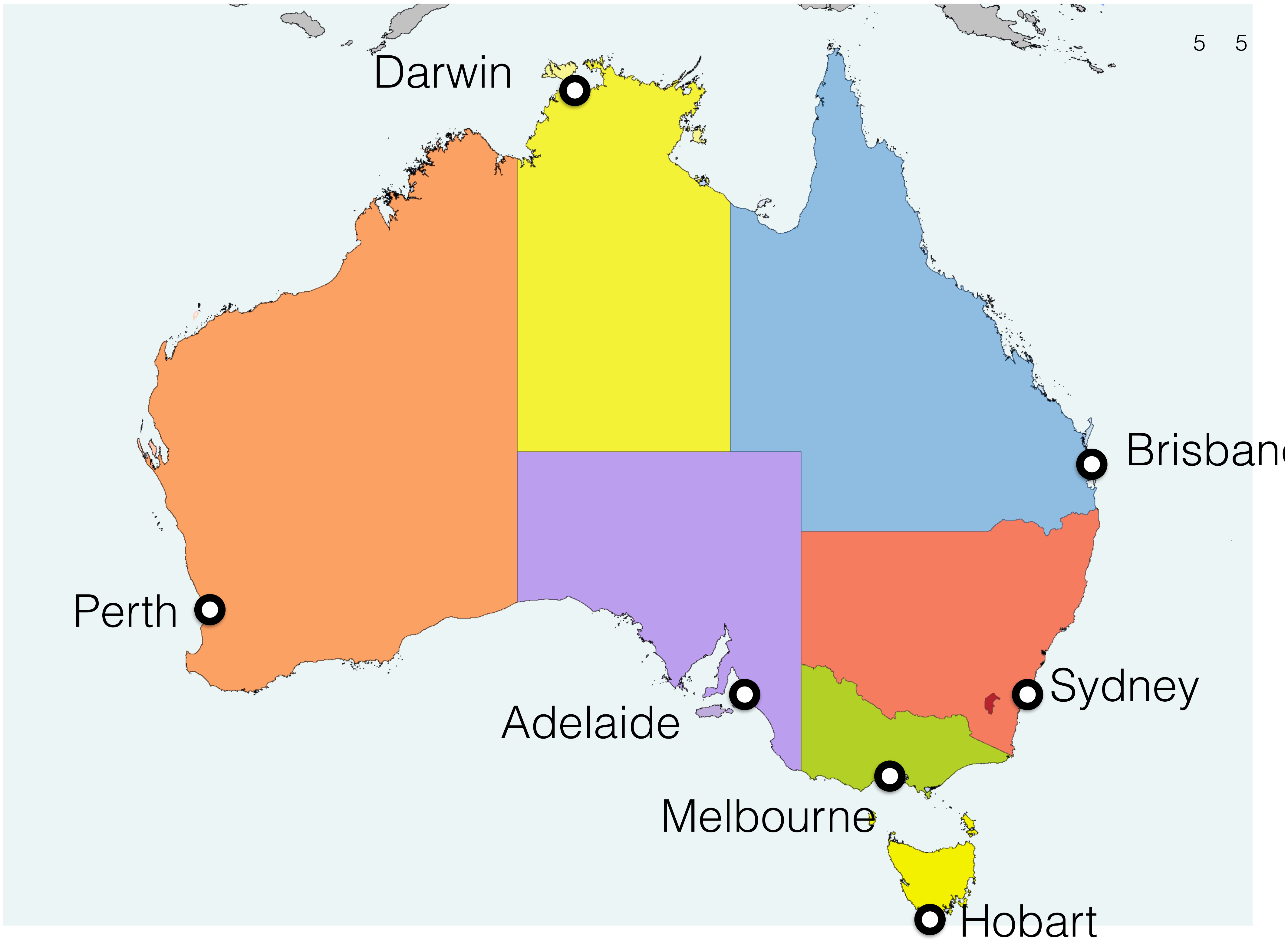
The satisfiability problem (SAT) is a fundamental problem from computer science.

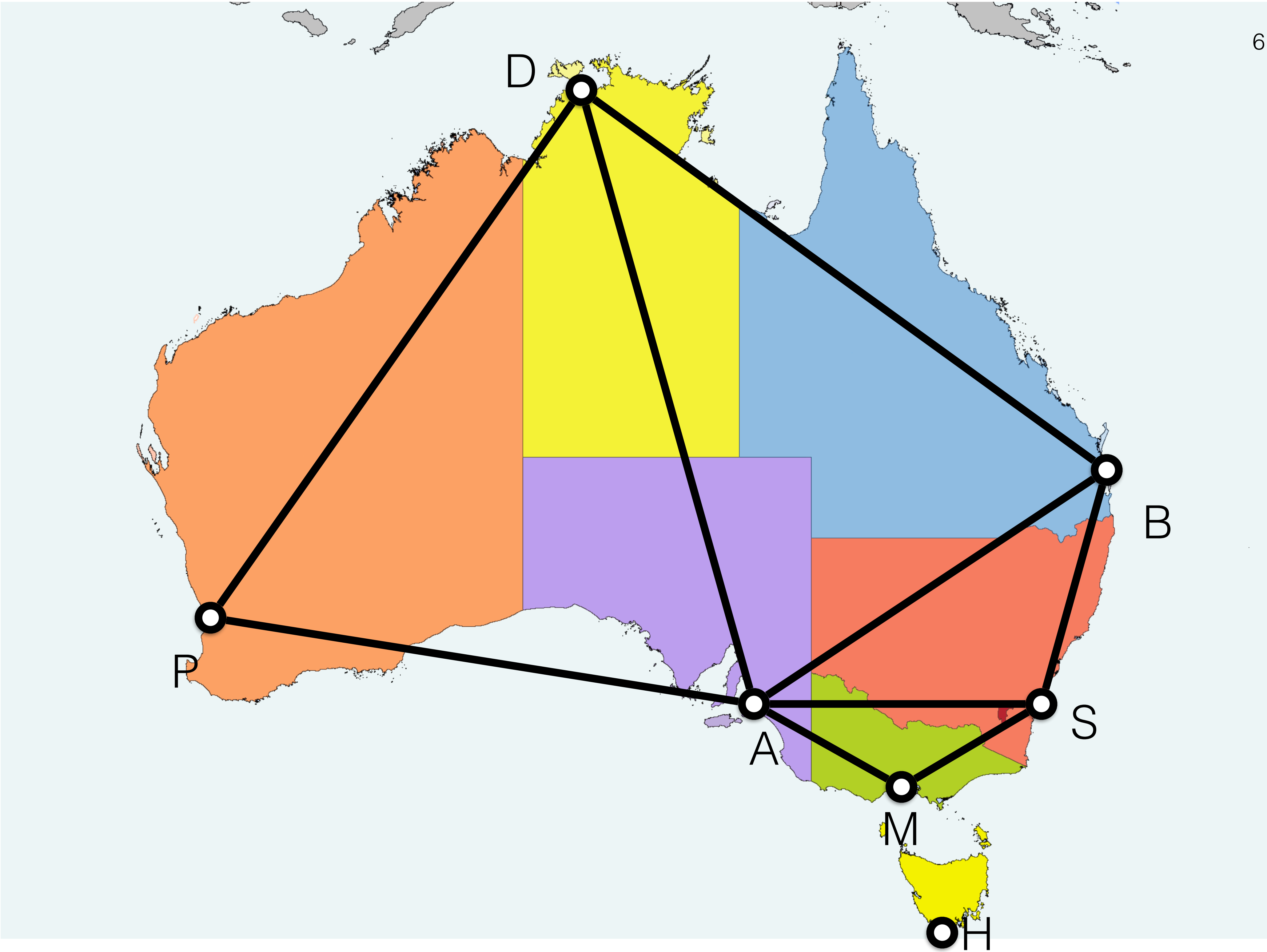
It is the problem to decide whether a formula of N Boolean variables can be satisfied, i.e. evaluated to TRUE by an assigment of the variables.

Without loss of generality one can assume that the formula is organized as conjunction of clauses, where each clause is a disjunction of literals.

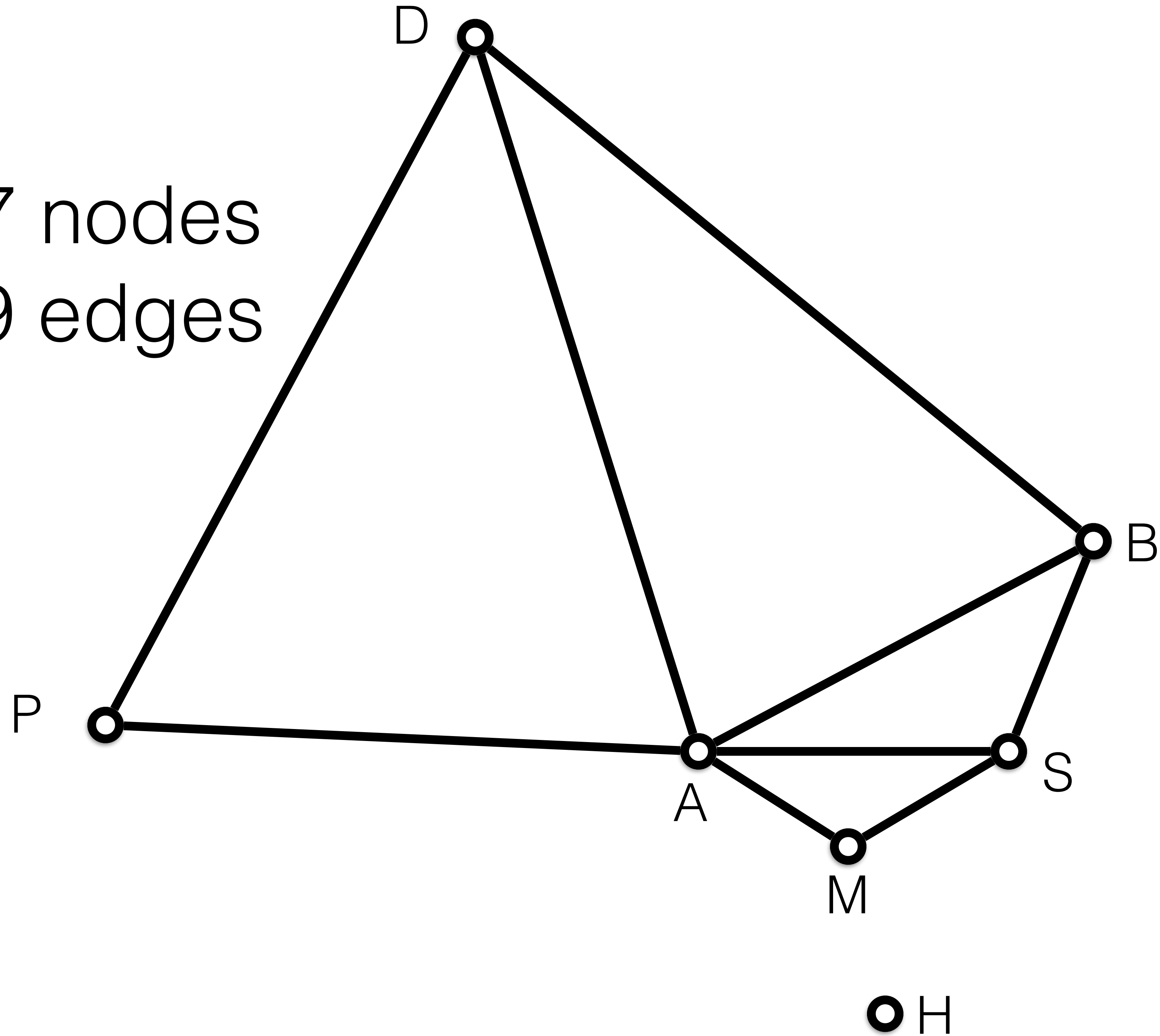
SAT is a very important problem in theoretical computer science (some people would even say: the most important problem),

but it has numerous practical applications as well.


Darwin
Brisban
Perth
Adelaide
Sydney
Melbourne
Hobart

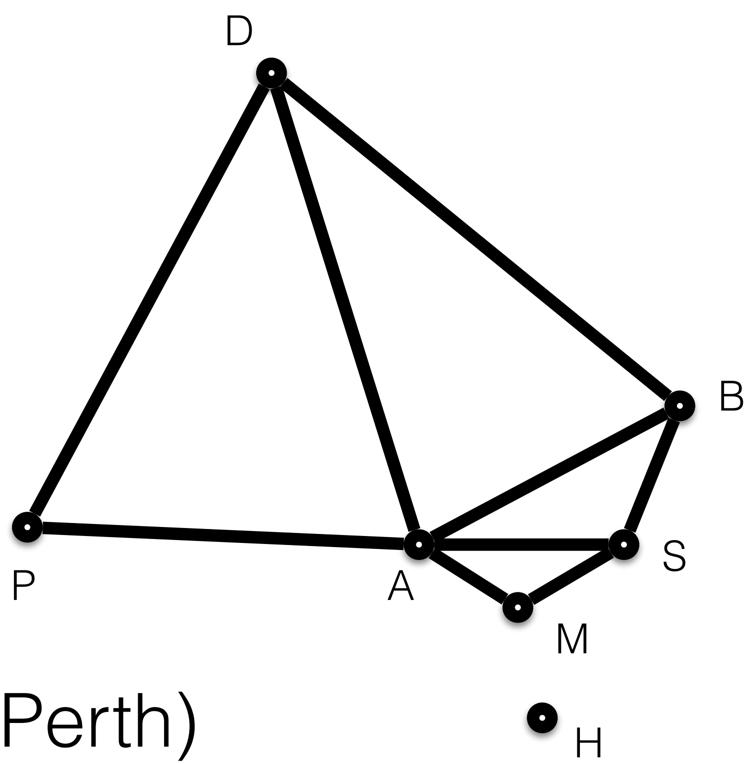

D
B
P
A
S
M
H

7 nodes
9 edges



21 atoms

		Melbourne	Sydney	Hobart	Darwin	Perth	Adelaide	Brisbane
red	🔴	Mr	Sr	Hr	Dr	Pr	Ar	Br
green	🟢	Mg	Sg	Hg	Dg	Pg	Ag	Bg
amber	🟠	Ma	Sa	Ha	Da	Pa	Aa	Ba



eg:
Pr ≡ red(Perth)

34 clauses

1 for each node (eg D)

Dr ∨ Dg ∨Da

3 for each edge (eg D–B)

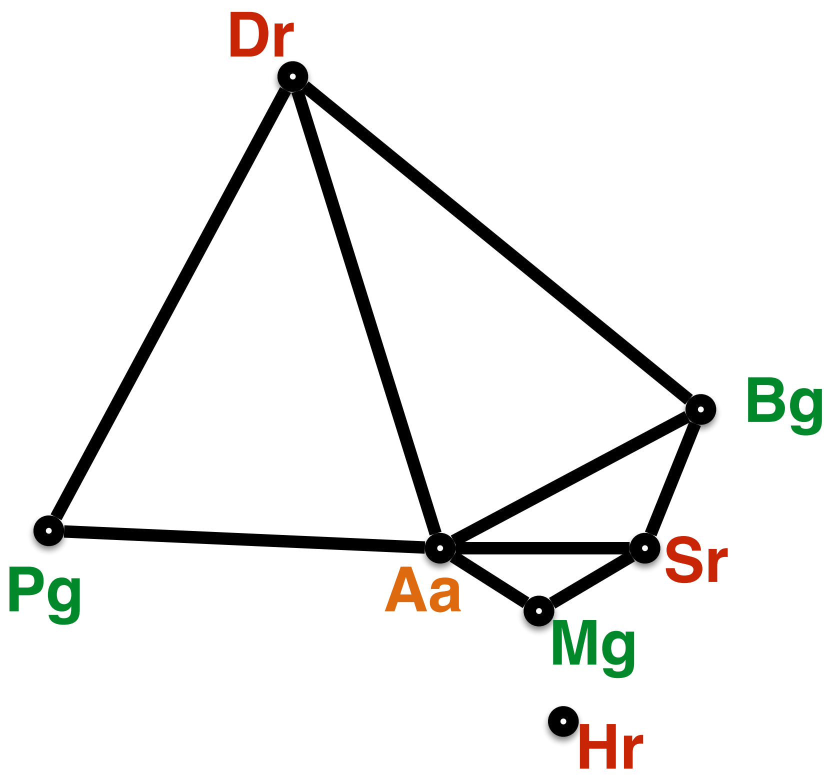
¬Dr ∨ ¬Br

¬Dg ∨ ¬Bg

¬Da ∨ ¬Ba

21 atoms

	Melbourne	Sydney	Hobart	Darwin	Perth	Adelaide	Brisbane
red	Mr	Sr	Hr	Dr	Pr	Ar	Br
green	Mg	Sg	Hg	Dg	Pg	Ag	Bg
amber	Ma	Sa	Ha	Da	Pa	Aa	Ba



34 clauses

1 for each node (e.g. D)

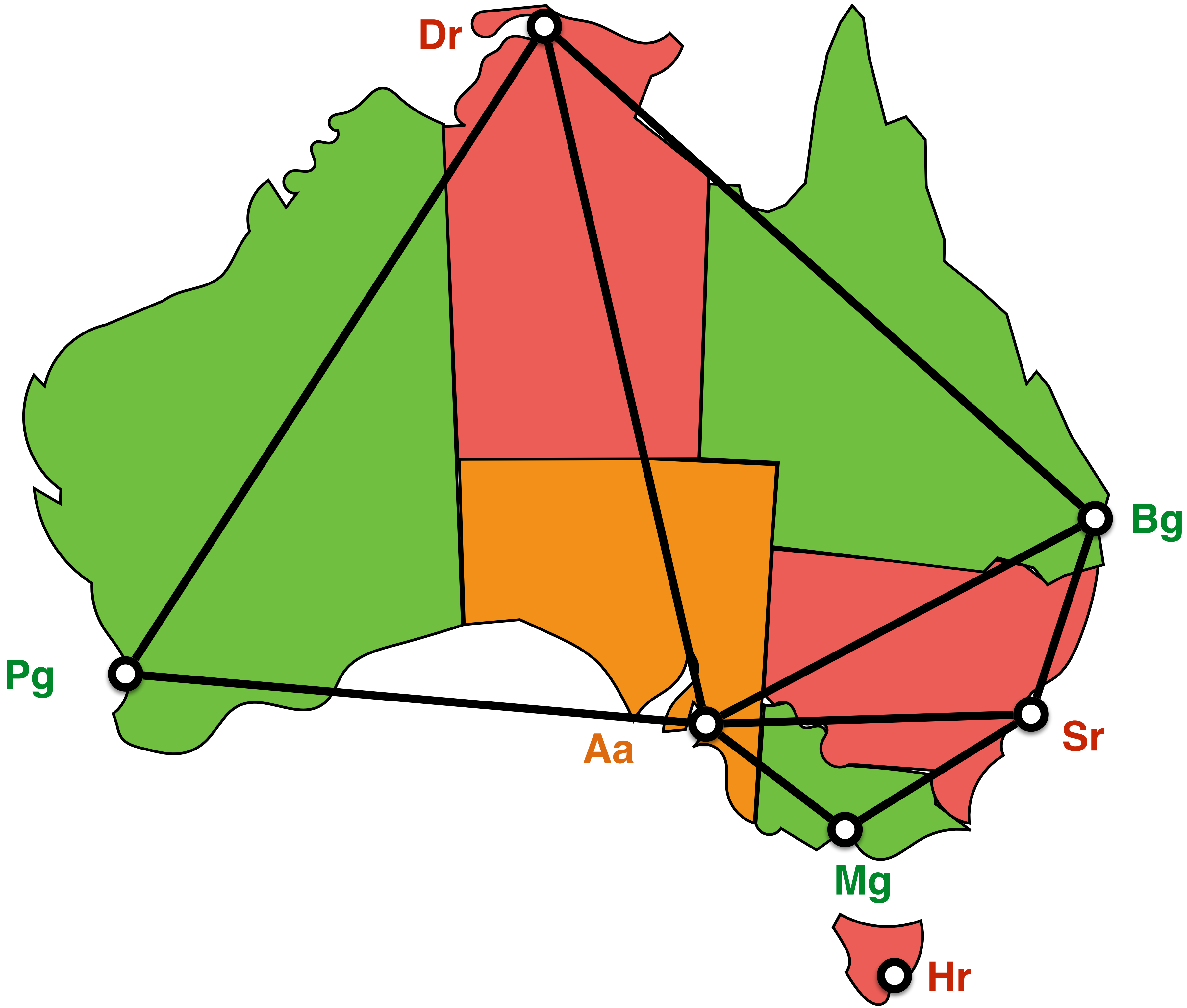
Dr ∨ Dg ∨Da

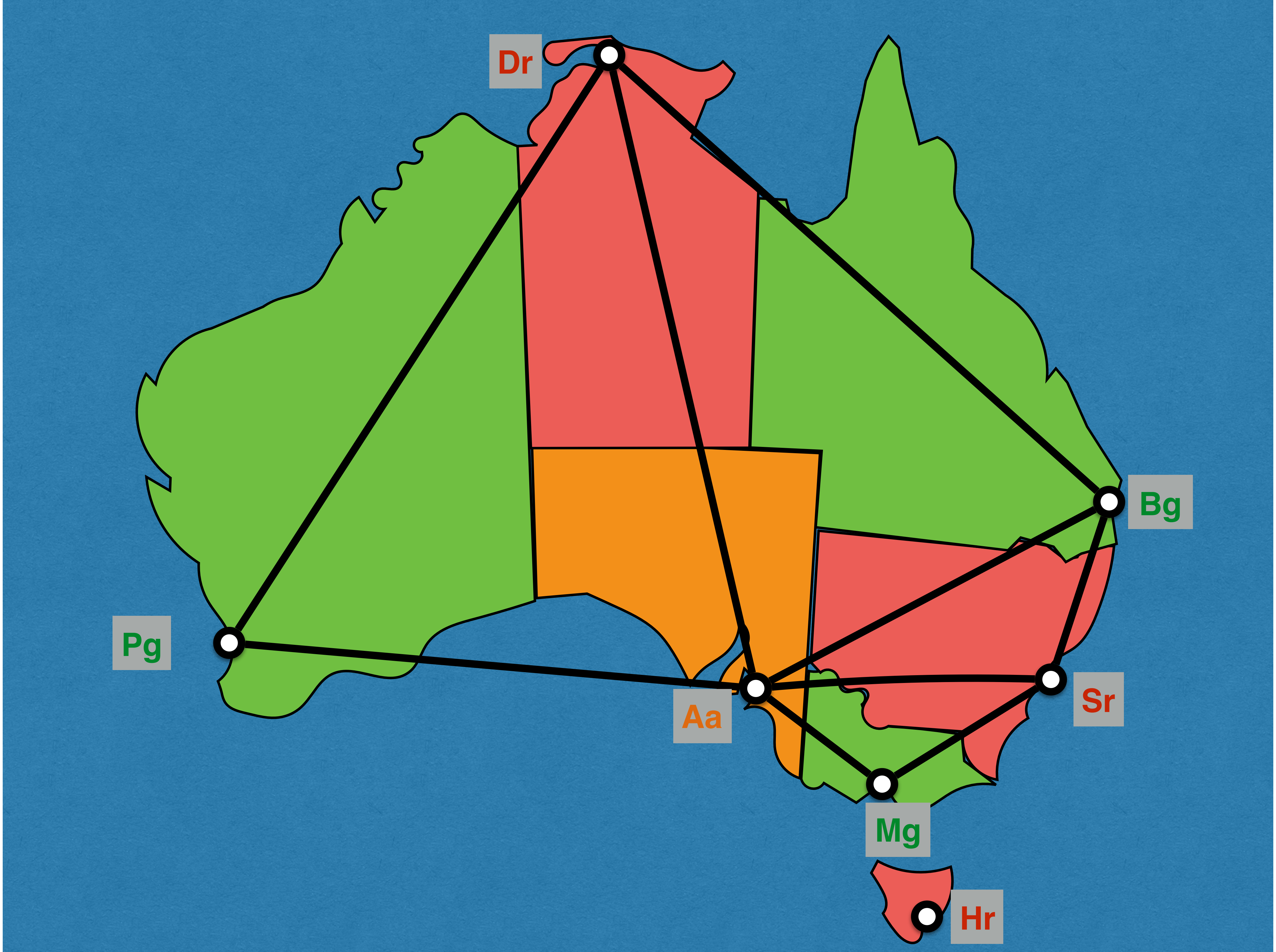
3 for each edge (e.g. D–B)

¬Dr ∨ ¬Br

¬Dg ∨ ¬Bg

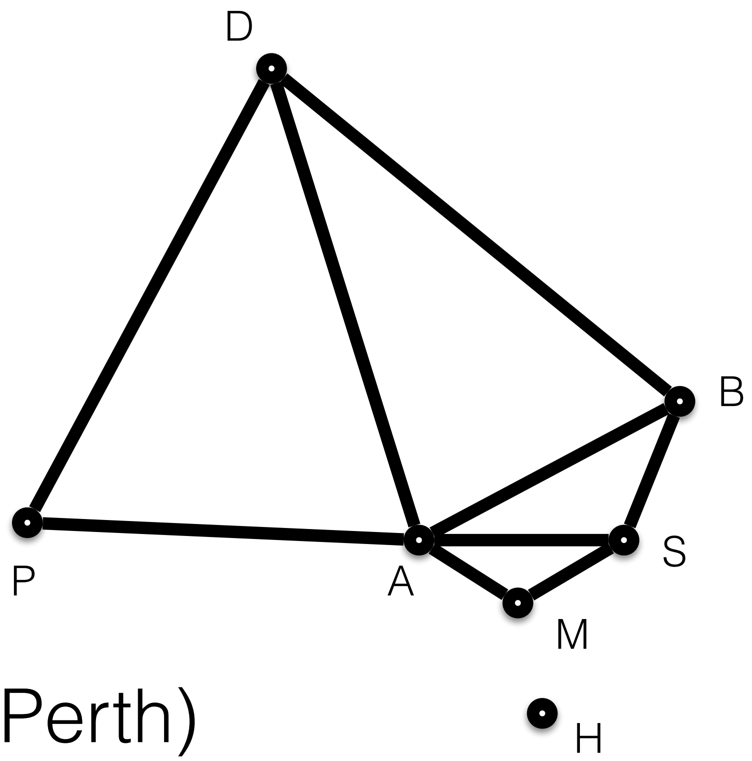
¬Da ∨ ¬Ba


Dr
Pg
Bg
Aa
Sr
Mg
Hr


Dr
Bg
Pg
Aa
Sr
Mg
Hr

21 atoms

		Melbourne	Sydney	Hobart	Darwin	Perth	Adelaide	Brisbane
red	🔴	Mr	Sr	Hr	Dr	Pr	Ar	Br
green	🟢	Mg	Sg	Hg	Dg	Pg	Ag	Bg
amber	🟠	Ma	Sa	Ha	Da	Pa	Aa	Ba



eg:
Pr ≡ red(Perth)

34 clauses

1 for each node (eg D)

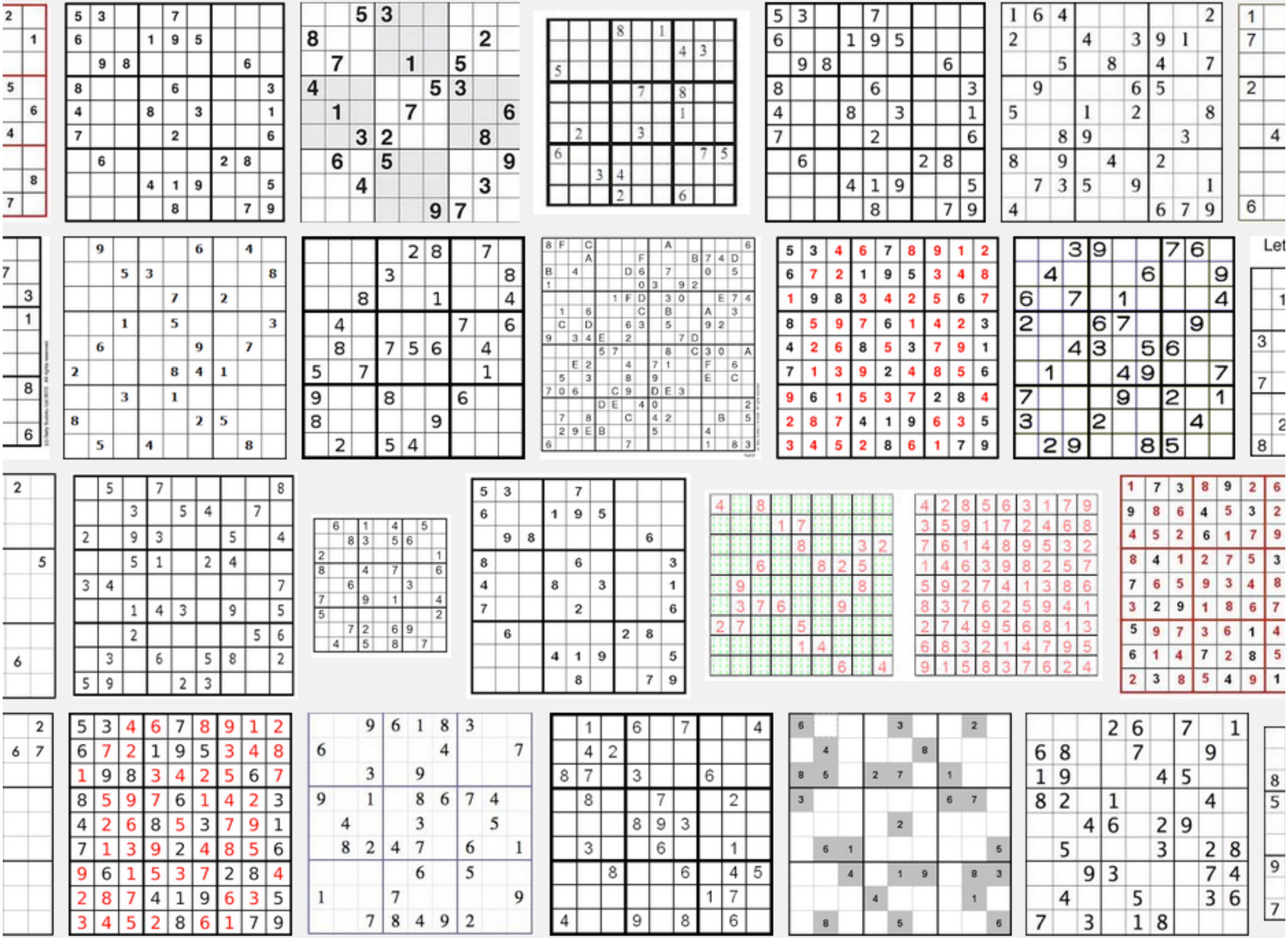
Dr ∨ Dg ∨Da

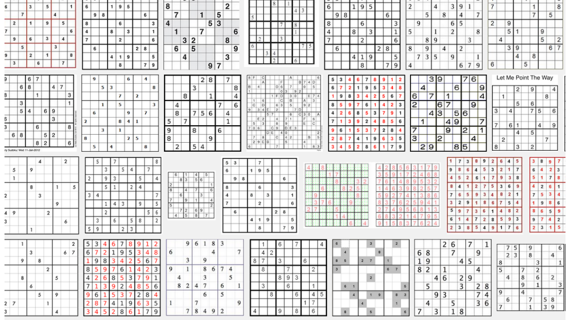
3 for each edge (eg D–B)

¬Dr ∨ ¬Br

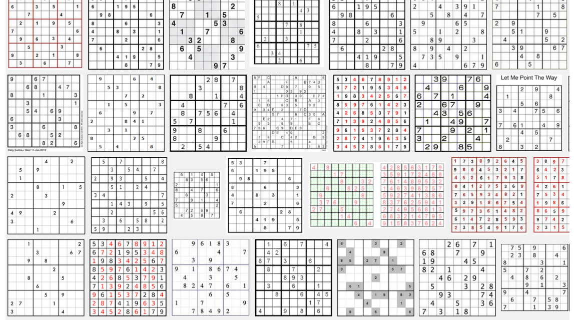
¬Dg ∨ ¬Bg

¬Da ∨ ¬Ba





Sudoku



Squares i, j (i, j ∈ (1..9))

Numbers k (k ∈ (1..9))

729 (= 9^3) Atoms $\mathbf{p}_{\mathbf{i,\,j,\,k}}$

$\mathbf{p}_{\mathbf{i,\,j,\,k}}$

means

the number k is in square i,j

A sudoku problem is defined
by saying which numbers are in which squares

```
(((p 1 2 3) and ((p 1 6 1) and ((p 2 3 6) and ((p 2 8 5) and ((p 3 1 5) and ((p 3 7 9) and (p 3 8 8))
```

$(p_{1,2,3} \wedge p_{1,6,1} \wedge p_{2,3,6} \wedge p_{2,8,5} \wedge p_{3,1,5} \wedge p_{3,7,9} \wedge p_{3,8,8})$

```
(((p 4 2 8) and ((p 4 6 6) and ((p 4 7 3) and ((p 4 9 2) and ((p 5 5 5) and ((p 6 1 9) and ((p 6 3 3) and ((p 6 4 8) and (p 6 8 6))
```

$(p_{4,2,8} \wedge p_{4,6,6} \wedge p_{4,7,3} \wedge p_{4,9,2} \wedge p_{5,5,5} \wedge p_{6,1,9} \wedge p_{6,3,3} \wedge p_{6,4,8} \wedge p_{6,8,6})$

```
(((p 7 1 7) and ((p 7 2 1) and ((p 7 3 4) and ((p 7 9 9) and ((p 8 2 2) and ((p 8 7 8) and ((p 9 4 4) and (p 9 8 3))
```

$(p_{7,1,7} \wedge p_{7,2,1} \wedge p_{7,3,4} \wedge p_{7,9,9} \wedge p_{8,2,2} \wedge p_{8,7,8} \wedge p_{9,4,4} \wedge p_{9,8,3})$

```
((bigand i((1 .. 9)) (bigand j((1 .. 9)) (bigand n((1 .. 9)) (bigand m((1 .. 9)) ((m diff n)) (((p i j n)) imply (not ((p i j m)))))))))
```

$$\bigwedge_{i\in\{1,..,9\}}\bigwedge_{j\in\{1,..,9\}}\bigwedge_{n\in\{1,..,9\}}\bigwedge_{m\in\{1,..,9\}|(m\neq n)}(p_{i,j,n}\rightarrow\neg p_{i,j,m})$$

at most one number per square

```
((bigand n((1 .. 9)) (bigand i((1 .. 9)) (bigor j((1 .. 9)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigwedge_{i\in\{1,..,9\}}\bigvee_{j\in\{1,..,9\}}p_{i,j,n}$$

every number occurs in each row

```
((bigand n((1 .. 9)) (bigand j((1 .. 9)) (bigor i((1 .. 9)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigwedge_{j\in\{1,..,9\}}\bigvee_{i\in\{1,..,9\}}p_{i,j,n}$$

every number occurs in each column

```
((bigand n((1 .. 9)) (bigor i((1 .. 3)) (bigor j((1 .. 3)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{1,..,3\}}\bigvee_{j\in\{1,..,3\}}p_{i,j,n}$$

every number occurs in top-left square

```
((bigand n((1 .. 9)) (bigor i((4 .. 6)) (bigor j((1 .. 3)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{4,..,6\}}\bigvee_{j\in\{1,..,3\}}p_{i,j,n}$$

every number occurs in centre-left square

```
((bigand n((1 .. 9)) (bigor i((7 .. 9)) (bigor j((1 .. 3)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{7,..,9\}}\bigvee_{j\in\{1,..,3\}}p_{i,j,n}$$

every number occurs in bottom-left square

```
((bigand n((1 .. 9)) (bigor i((1 .. 3)) (bigor j((4 .. 6)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{1,..,3\}}\bigvee_{j\in\{4,..,6\}}p_{i,j,n}$$

every number occurs in top-middle square

```
(bigand n ((1 .. 9)) ((bigor i ((4 .. 6)) ((bigor j ((4 .. 6)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{4,..,6\}}\bigvee_{j\in\{4,..,6\}} p_{i,j,n}$$

every number occurs in centre-middle square

```
(bigand n ((1 .. 9)) ((bigor i ((7 .. 9)) ((bigor j ((4 .. 6)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{7,..,9\}}\bigvee_{j\in\{4,..,6\}} p_{i,j,n}$$

every number occurs in bottom-middle square

```
(bigand n ((1 .. 9)) ((bigor i ((1 .. 3)) ((bigor j ((7 .. 9)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{1,..,3\}}\bigvee_{j\in\{7,..,9\}} p_{i,j,n}$$

every number occurs in top-right square

```
(bigand n ((1 .. 9)) ((bigor i ((4 .. 6)) ((bigor j ((7 .. 9)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{4,..,6\}}\bigvee_{j\in\{7,..,9\}} p_{i,j,n}$$

every number occurs in centre-right square

```
(bigand n ((1 .. 9)) ((bigor i ((7 .. 9)) ((bigor j ((7 .. 9)) ((p i j n))))))
```

$$\bigwedge_{n\in\{1,..,9\}}\bigvee_{i\in\{7,..,9\}}\bigvee_{j\in\{7,..,9\}} p_{i,j,n}$$

every number occurs in bottom-right square

729 atoms

structural constraints include

many, many occurrences of literals

How Many?

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 clauses with 3 or more literals are involved, satisfaction gets complicated.

In general, we must search for satisfaction.

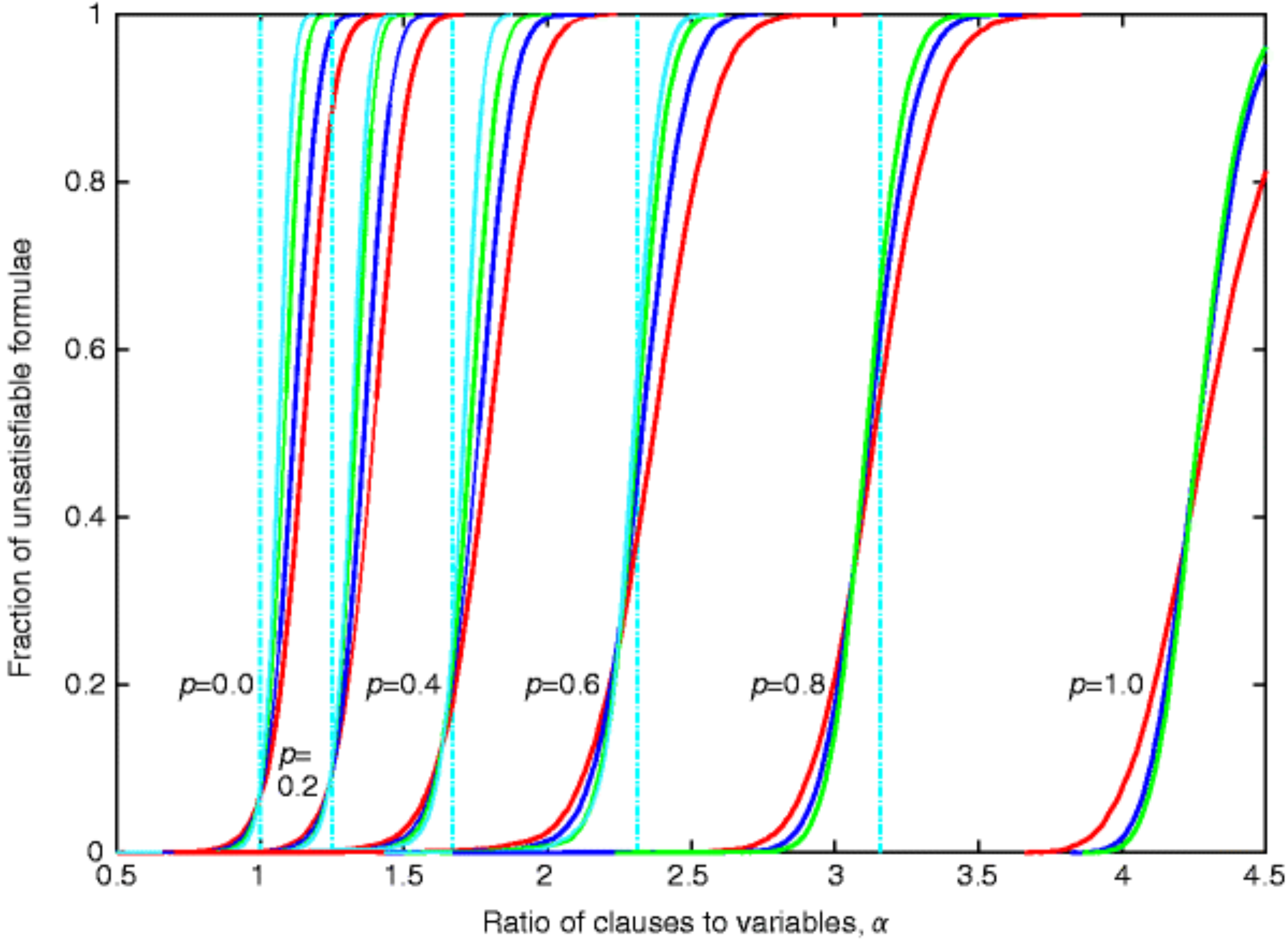
If every clause has 2 variables (2-SAT) the problem is easy.
If every clause has 3 variables (3-SAT) the problem is hard.

A clause with K literals excludes $1/2^K$ of the 2^N possible assignments and the whole formula is satisfiable if the number of clauses is small compared to the number of variables.

Just by counting, we can see that a K-SAT problem is satisfiable if the number of clauses is less than 2^K, and it is easy to exclude all valuations with 2^K clauses

For a large number of variables, N,
a **random** 3-SAT problem with N variables,
with less than ~ 4.2 N clauses if is probably satisfiable;
with more than ~ 4.2 N clauses if is probably not satisfiable.

Hard problems appear near the boundary.


Fraction of unsatisfiable formulae
Ratio of clauses to variables, α
p=0.0
p=
0.2
p=0.4
p=0.6
p=0.8
p=1.0
0
0.2
0.4
0.6
0.8
1
0.5
1
1.5
2
2.5
3
3.5
4
4.5

Naïve search

V is a partial valuation
(a consistent set of literals)
$V \wedge A = V \cup \{A\}$

```
function SAT(Φ,V)
  Φ|V = {}
  ||
    {} ∉ Φ|V
    &&
    let A = chooseLiteral (Φ,V)
    in
      SAT (Φ,V ^ A)
      ||
      SAT (Φ,V ^ ¬A)
```

Φ is a set of clauses

$\Phi \mid V$ is the result of simplifying Φ using V:
For each literal $L \in V$

- remove clauses containing L
- delete $\neg L$ from remaining clauses

chooseLiteral(Φ,V) returns a literal occurring in $\Phi \mid V$

partial valuations

Φ

A	B	C
$\neg C$	B	D
$\neg A$	B	C
$\neg A$	$\neg B$	$\neg C$

$\Phi \mid V$

A	B	C
$\neg C$	B	D
$\neg A$	B	C
$\neg A$	$\neg B$	$\neg C$

V : []

search

[]

Φ

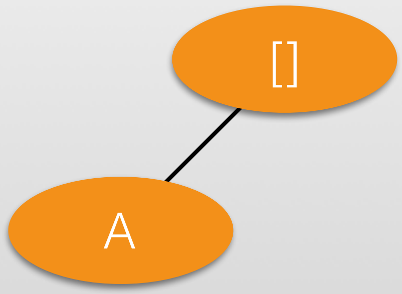
A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

Φ | V

A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

V : [A]

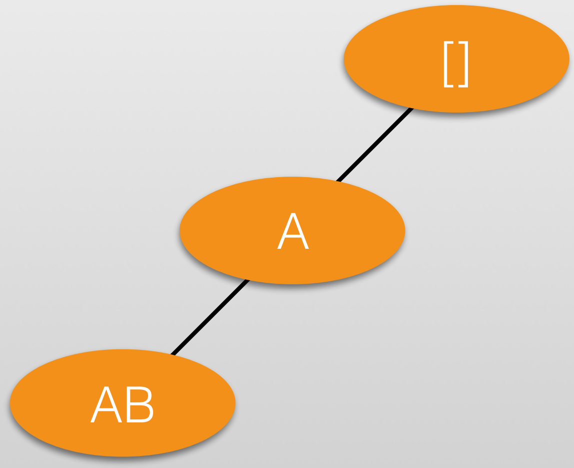
search



Φ	Φ | V
A B C	A B C
¬C B D	¬C B D
¬A B C	¬A B C
¬A ¬B ¬C	¬A ¬B ¬C

V : [A,B]

search

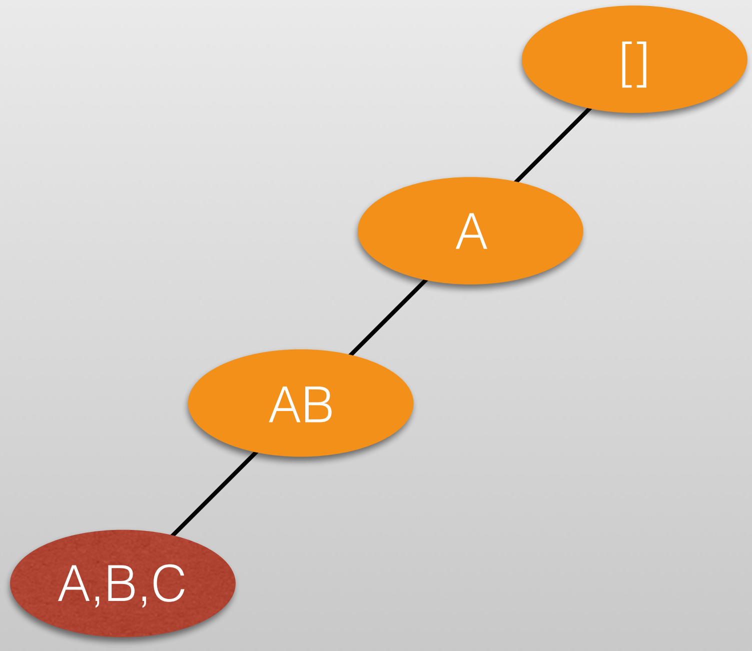


Φ		
A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

$\Phi \mid V$		
A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

V :[A,B,C]

search



Φ

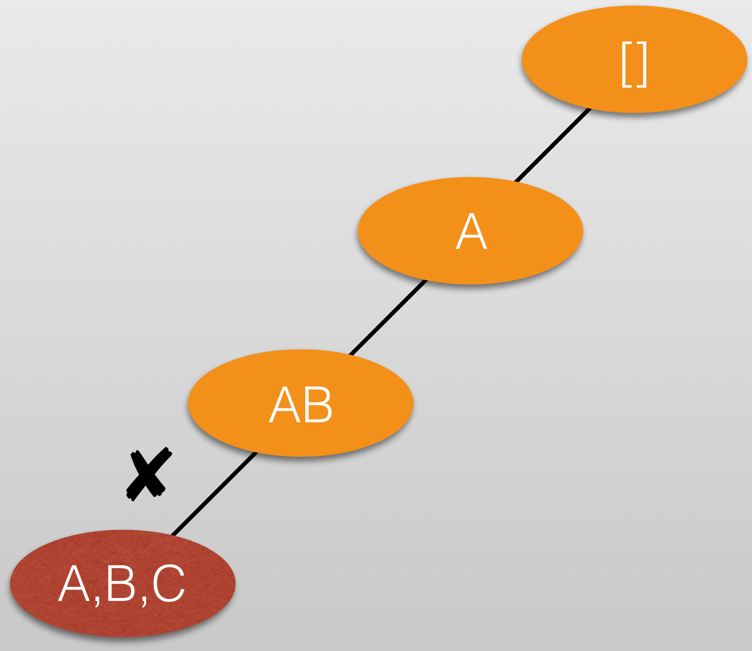
A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

$\Phi \mid V$

A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

V : [A,B]

search



Φ

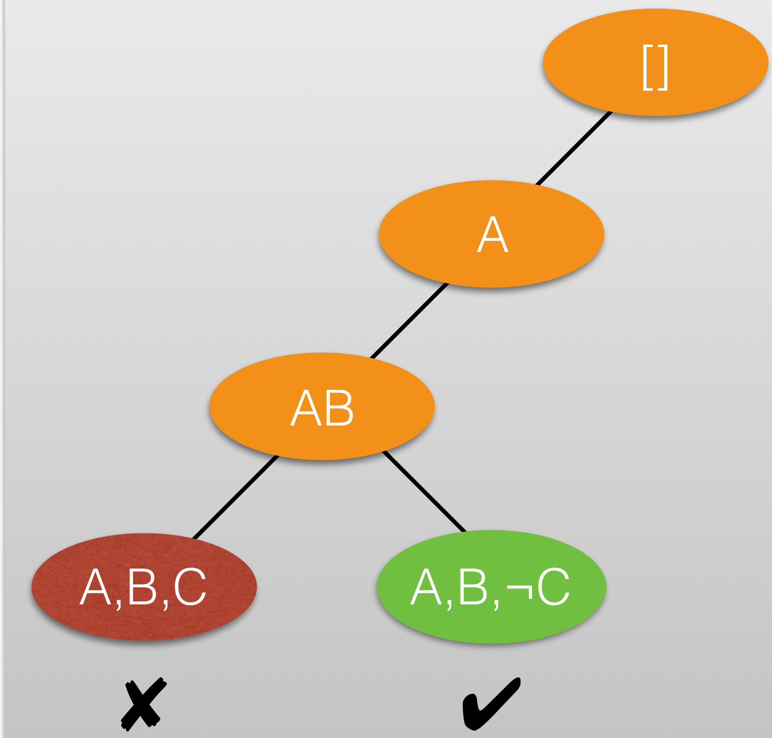
A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

$\Phi \mid V$

A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

V :[A,B,C]

search



Φ

A B C

¬C B D

¬A B C

¬A ¬B ¬C

Φ | V

A B C

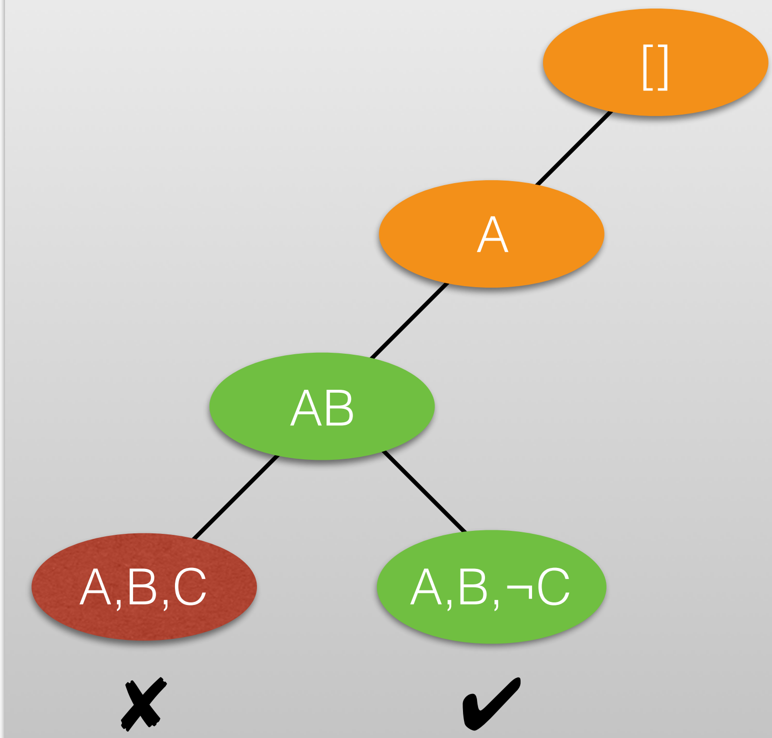
¬C B D

¬A B C

¬A ¬B ¬C

V :[A,B,C]

search



Φ

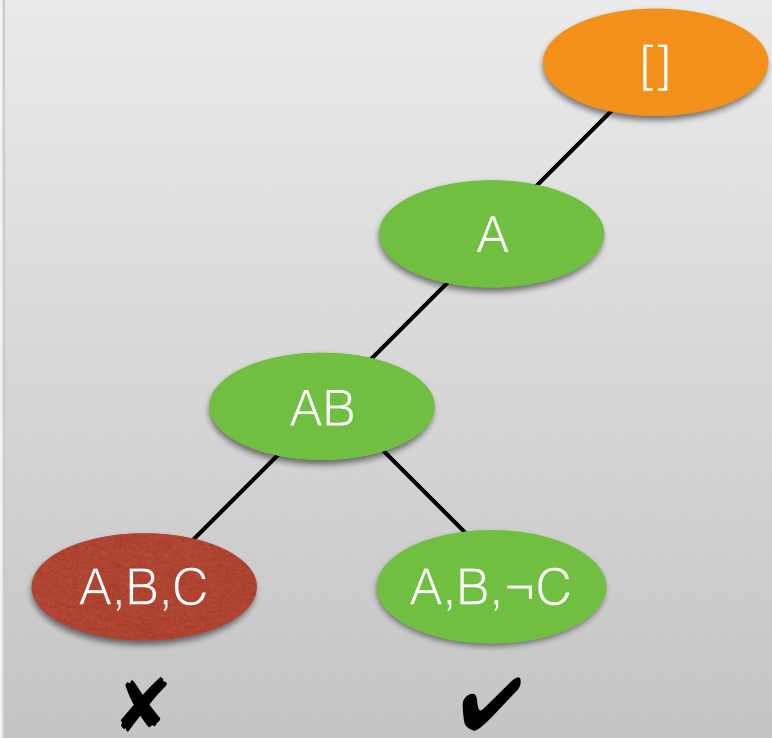
A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

Φ | V

A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

V :[A,B,C]

search



Φ

A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

$\Phi \mid V$

A	B	C
¬C	B	D
¬A	B	C
¬A	¬B	¬C

V :[A,B,C]

search