An Example of the GRASP SAT Solver

Michael Carl Tschantz
with help from Himanshu Jain

15-414: Bug Catching
Professor Edmund M. Clarke
Carnegie Mellon University
2007

The Clauses

w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = ¬x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2

GRASP

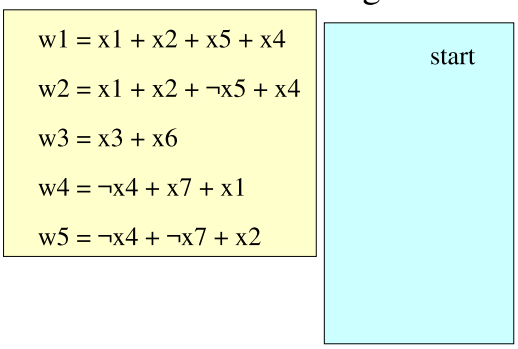
GRASP is SAT solver

Created by
Joan P. Marques Silva and
Karem A. Sakallah

See "GRASP – A New Search Algorithm for Satisfiability" by Silva and Sakallah for details

This just presents an example of the algorithm in action

Starting State



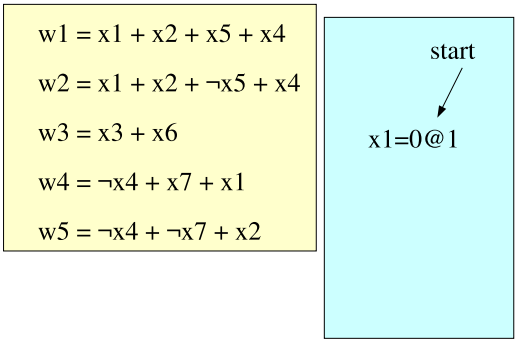
Simplifying Assumption

I will act as though GRASP selects the next variable to assign a value to in numerical order and always tries false first

In reality, GRASP selects the assignment that would satisfy as many clauses as possible

I could add clauses to my example to avoid this simplification, but I like small examples

Assign False to x1



Assign False to x1

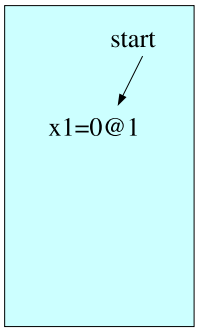
w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2



Assign False to x3

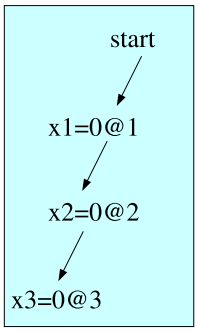
w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2



We have a unit clause! x6 must be set to true.

Assign False to x2

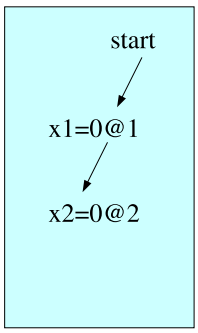
w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2



Assign False to x3

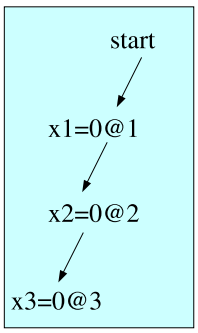
w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2



The x6 is set to true because x3 was set to false.
This might be useful info. Let's remember it.

Assign False to x3

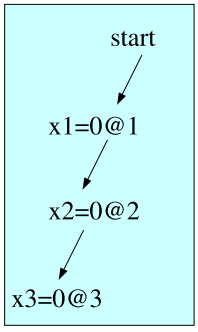
w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2



Implication Graph

An **implication graph** keeps track of why variables are assigned the value that they are

Each assigned variable has a node

Variables that force another variable to have an certain assignment point to it

Implication Graph

w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2

start

x1=0@1

x2=0@2

x3=0@3

Let's make an implication graph.

Implication Graph

w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4

start

x1=0@1

x2=0@2

x3=0@3

x1=0@1

x2=0@2

x3=0@3 —w3→ x6=1@3

Because x3 = 0, x6 must be 1.
We know this from w3

Edges connect a node to why it was assigned the way it was

Implication Graph

w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2

start

x1=0@1

x2=0@2

x3=0@3

x1=0@1

x2=0@2

x3=0@3 x6=1@3

Each assignment gets a node.

Implication Graph

w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4

start

x1=0@1

x2=0@2

x3=0@3

x1=0@1

x2=0@2

x3=0@3 —w3→ x6=1@3

x1, x2, x3 are set to 0
because they are decision variables.
No edges to them.

Edges connect a node to why it was assigned the way it was

Implication Graph

w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2

start

x1=0@1

x2=0@2

x3=0@3

x1=0@1

x2=0@2

x3=0@3 x6=1@3

Edges connect a node to why it was assigned the way it was

Implication Graph

w1 = x1 + x2 + x5 + x4

w2 = x1 + x2 + ¬x5 + x4

w3 = x3 + x6

w4 = ¬x4 + x7 + x1

w5 = ¬x4 + ¬x7 + x2

start

x1=0@1

x2=0@2

x3=0@3

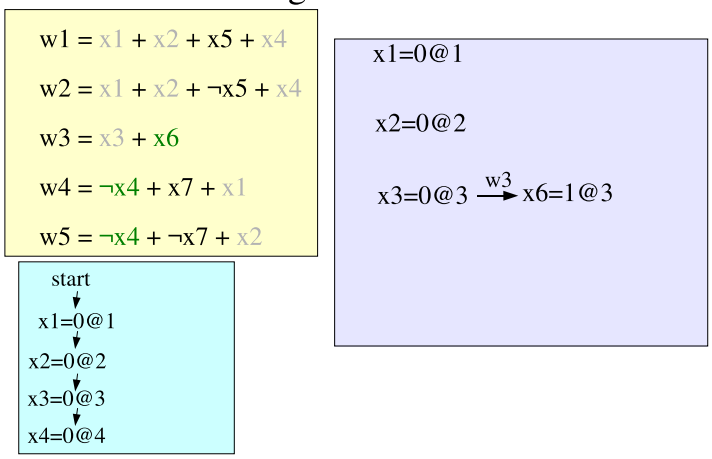
x1=0@1

x2=0@2

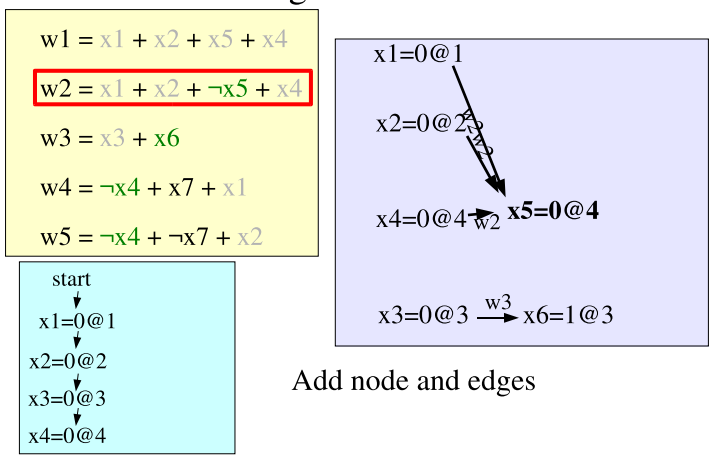
x3=0@3 —w3→ x6=1@3

The finished graph

Assign False to x4.

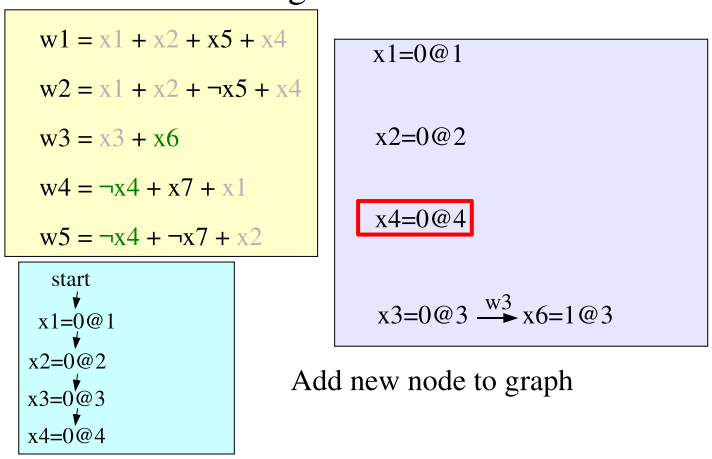


Assign False to x4.



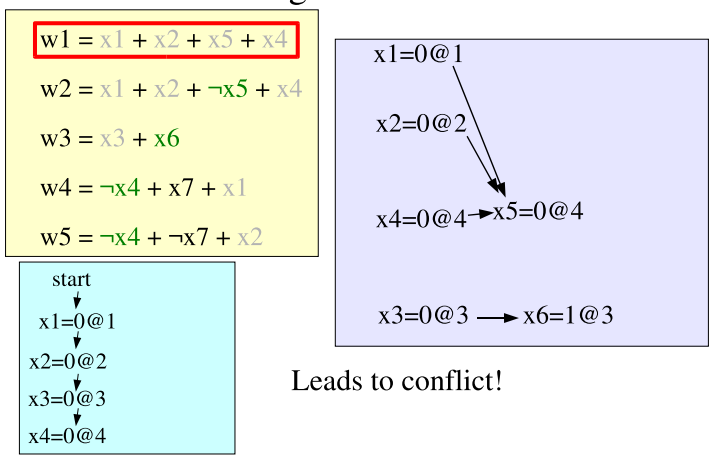
Add node and edges

Assign False to x4.



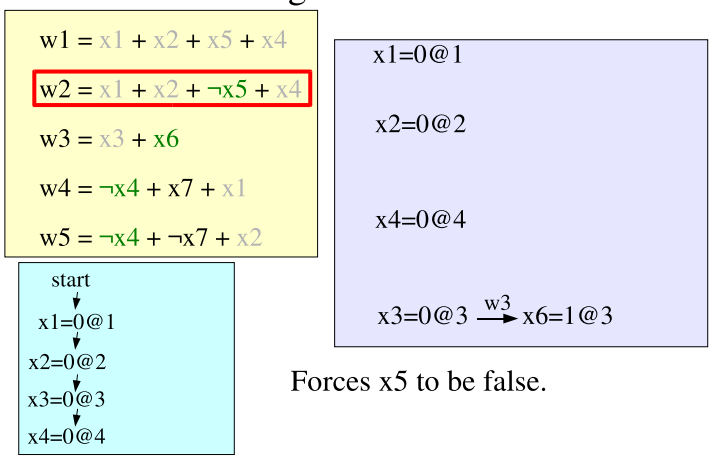
Add new node to graph

Assign False to x4.



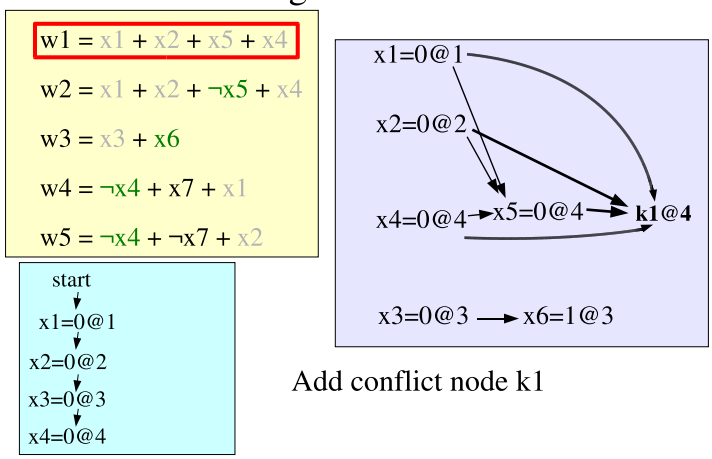
Leads to conflict!

Assign False to x4.



Forces x5 to be false.

Assign False to x4.

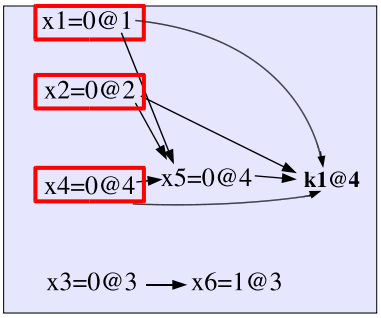


Add conflict node k1

Assign False to x4.

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2

start
x1=0@1
x2=0@2
x3=0@3
x4=0@4

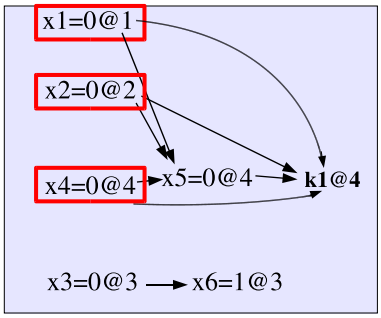


What caused conflict k1?
Follow the edges back to see

Assign False to x4.

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2

start
x1=0@1
x2=0@2
x3=0@3
x4=0@4



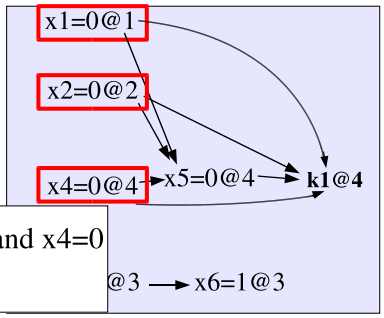
How can we avoid k1?

Assign False to x4.

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1

It was that x1=0, x2=0, and x4=0
i.e., ¬x1 & ¬x2 & ¬x4

x2=0@2
x3=0@3
x4=0@4



What caused conflict k1?
Follow the edges back to see

Conflict Clauses

k1 was caused by ¬x1 & ¬x2 & ¬x4

To avoid k1, force this not to be true

Add conflict clause

w(k1) = ¬(¬x1 & ¬x2 & ¬x4)
= x1 + x2 + x4

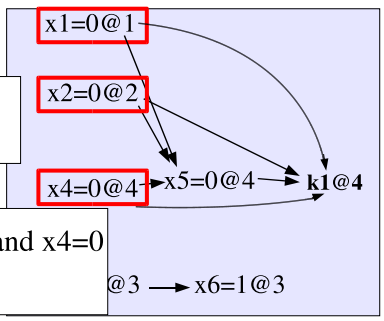
Assign False to x4.

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4

We don't need x5=0
since it is implied

It was that x1=0, x2=0, and x4=0
i.e., ¬x1 & ¬x2 & ¬x4

x2=0@2
x3=0@3
x4=0@4

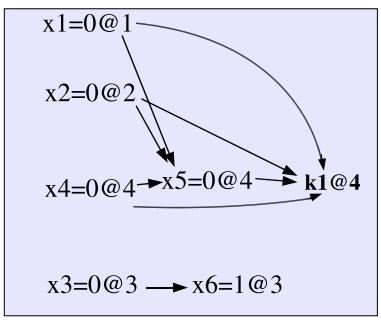


What caused conflict k1?
Follow the edges back to see

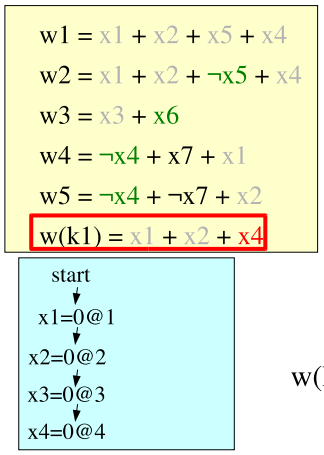
Add Conflict Clause

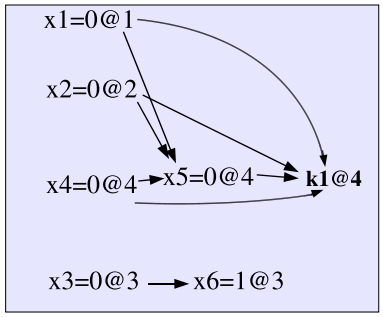
w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4

start
x1=0@1
x2=0@2
x3=0@3
x4=0@4



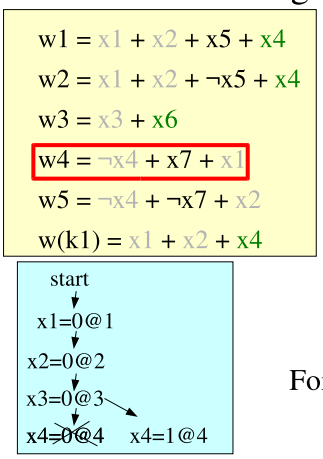
Add Conflict Clause

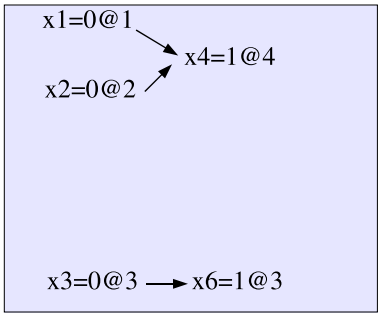




w(k1) forces x4 to be true

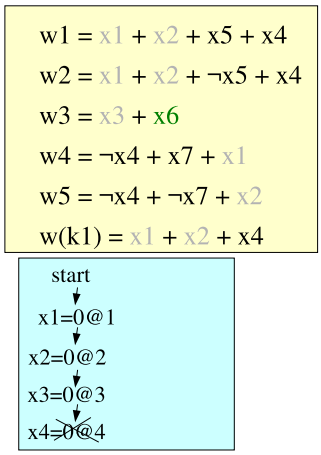
Assign True to x4

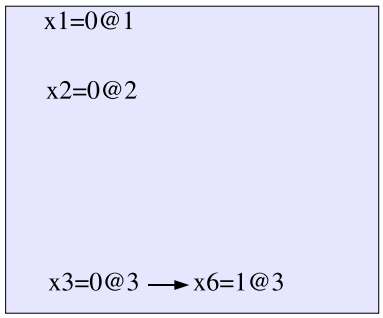




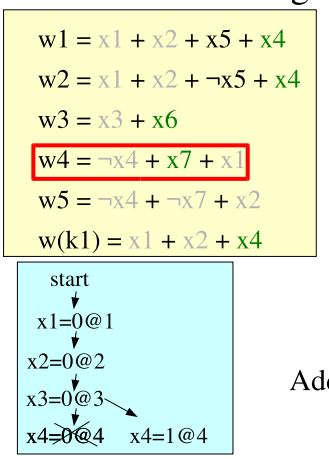
Forces x7 to be true

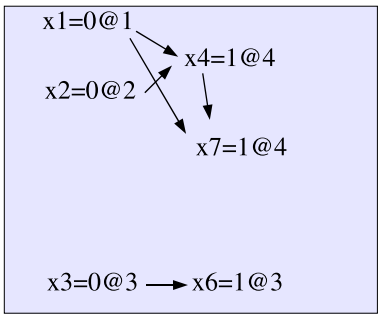
Undo x4=0@4





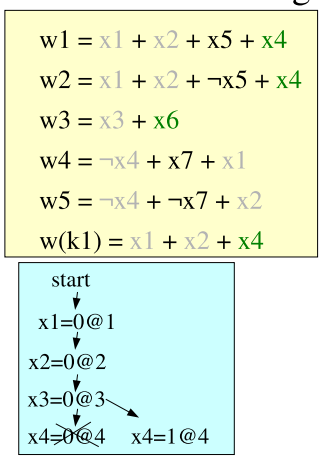
Assign True to x4

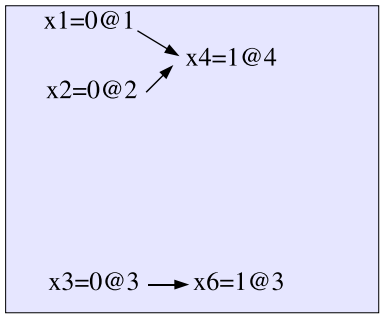




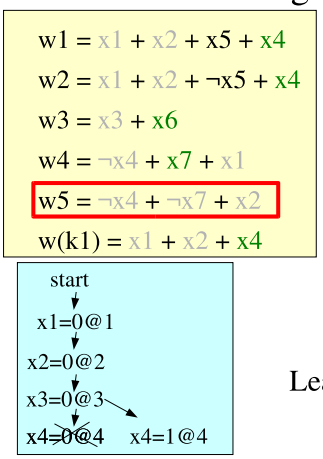
Add node and edges

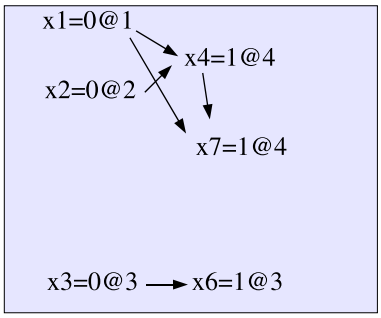
Assign True to x4





Assign True to x4

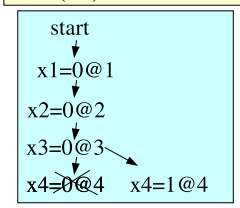


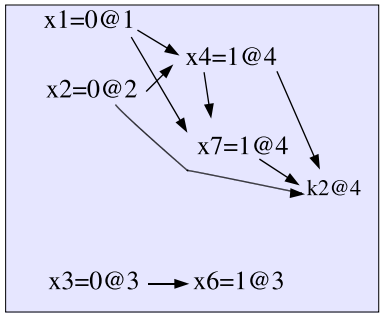


Leads to conflict!

Assign True to x4

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4

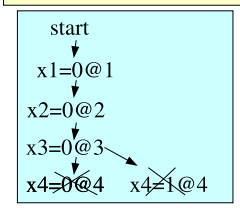


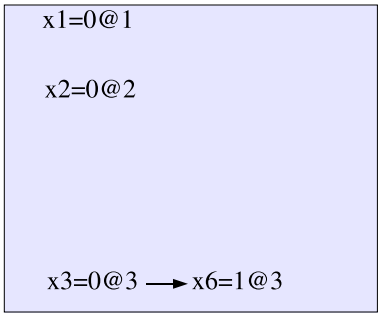


Add conflict node k2

Backtracking

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2

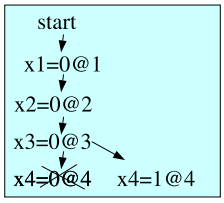


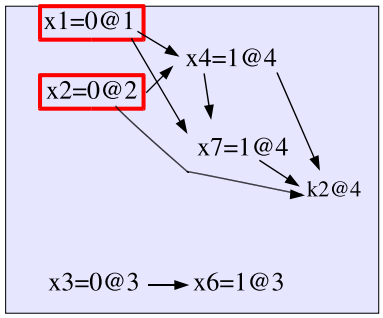


We need to undo
x4=1@4

Assign True to x4

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4

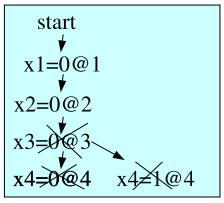


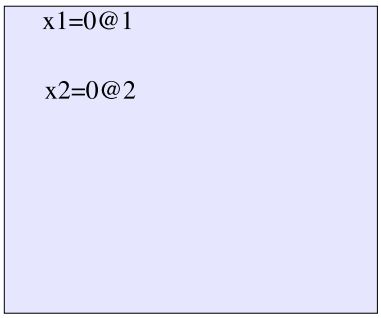


Go backwards from k2 to find
root causes

Backtracking

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2

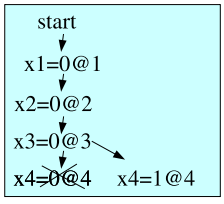


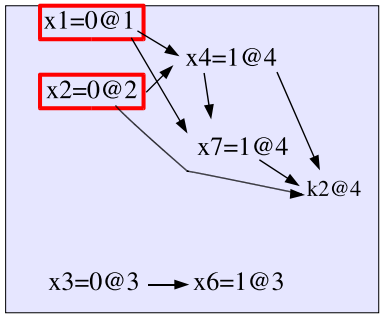


Since we tried both x4=0 and x4=1,
we'll also have to undo x3=0

Add Conflict Clause w(k2)

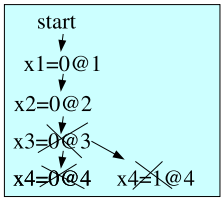
w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2

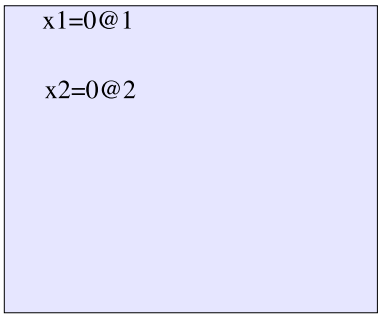




Backtracking

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2





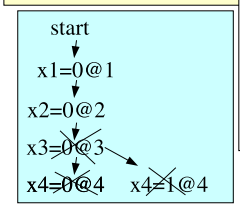
Now we try x3=1@3

Backtracking

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2

x1=0@1

x2=0@2



But wait!
x3 had nothing to do with k2.
We can skip trying x3=1@3
and backtrack up to x2=0@2

Non-Chronological Backtracking

GRASP gets away with this because the conflict clause made from the implication graph allows it to know what assignments did not play a role in the conflict

GRASP then jumps up to the latest assignment that did play a role

This called **conflict-driven** backtracking

Non-Chronological Backtracking

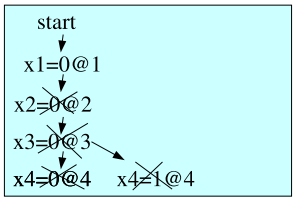
SAT solvers like Davis-Putnam undo assignments in the reserve order in which they are made and always try the other assignment

GRASP does non-chronological backtracking: it will skip assignment points all together and jump up higher in the decision tree

Backtracking

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2

x1=0@1

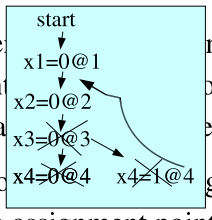


Undo x2=0@2

Non-Chronological Backtracking

SAT solve nam undo
assignmen order in which they
are made a e other assignment

GRASP do gical backtracking:
it will skip assignment points all together and
jump up higher in the decision tree

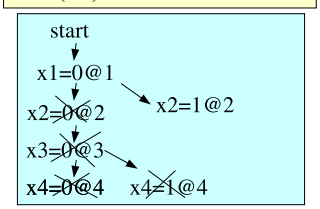


Assign True to x2

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2

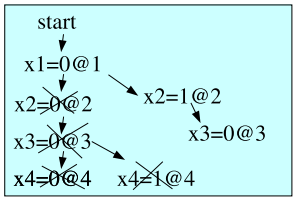
x1=0@1

x2=1@2



Assign False to x3

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2



x1=0@1

x2=1@2

x3=0@3 ⟶ x6=1@3

Key Points

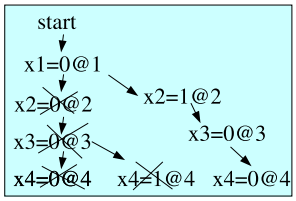
The **implication graph** allows us to know why a conflict forms

A **conflict clause** summarizes how to avoid a conflict

Knowing this allows for **conflict-driven non-chronological backtracking**

Assign False to x4

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2



x1=0@1

x2=1@2

x4=0@4

x3=0@3 ⟶ x6=1@3

Something to Ponder

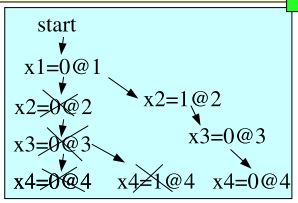
Keeping an implication graph takes both time and memory

Is the ability to do conflict-driven backtracking worth this cost?

How would you find out?

Assign False to x4

w1 = x1 + x2 + x5 + x4
w2 = x1 + x2 + ¬x5 + x4
w3 = x3 + x6
w4 = ¬x4 + x7 + x1
w5 = ¬x4 + ¬x7 + x2
w(k1) = x1 + x2 + x4
w(k2) = x1 + x2



x1=0@1

All Clauses Satisfied
We Are Done!

x3=0@3 ⟶ x6=1@3