CS3383 Unit 5.0: Backtracking and SAT

David Bremner

March 27, 2024

Combinatorial Search

- Backtracking
- SAT
- Tractable kinds of SAT

Background

- **DPV 9.1** Backtracking
- **DPV 8.1** SAT
- **DPV 5.3** Horn SAT
- **DPV Chapter 3 exercises** 2SAT
- **Unit Propagation** https://en.wikipedia.org/wiki/Unit_propagation

N-queens

Problem Description

Given an $n \times n$ chess board, can you place n queens so that no two are in the same row, column, or diagonal.

	Q		
			Q
Q			
		Q	

✓

	Q		
			Q
	Q		
		Q	

no

- One per row/col is easy to enforce

Representing Chessboards

- We only care about cases where there is 1 queen per column
- Represent a $n \times n$ board as an array of n integers, meaning which row.
- `None` for not chosen yet.

	Q		
			Q
Q			
		Q	

$(1, 3, 0, 2)$

		Q	
Q			

$(0, \texttt{None}, 2, \texttt{None})$

Backtracking Requirements

1. A representation for partial solutions
2. A procedure to expand a problem into smaller subproblems
3. A test for partial solutions that returns

   `True` if the solution is complete (Success)

   `False` if there is no way to complete (Failure)

   `None` if neither can be quickly determined. (Uncertainty)

Generic Backtracking

```
def backtrack(P0):
    S = [P0]
    while len(S) > 0:
        P = S.pop()
        for R in expand(P):
            result = test(R)
            if result == True:
                return R
            elif result == None:
                S.append(R)
    return False
```

Backtracking for N-Queens: Framework

representation $Q[0...n-1]$ where $Q[i]$ is row chosen, or `None`.

expand For some $Q[i] =$ `None`, try $Q[i] = 0...n-1$

```
def test(Q):
  default = True
  for i in range(len(Q)):
    if Q[i]==None:
      default = None
    else:
      for j in range(i):
        if Q[i] - Q[j] in [0,i-j,j-i]:
          return False
  return default
```

Backtracking for subset sum

Subset Sum

Given $X \subset \mathbb{Z}^+$, T

Decide Is there a subset of X that sums to T

```
def SubsetSum(X,T):
  if T == 0:
    return true
  elif T<0 or len(X) == 0:
    return False
  (y,rest) = (X[0],X[1:])
  return SubsetSum(rest, T-y) \
    or SubsetSum(rest,T)
```

The SAT Problem

Conjunctive Normal Form (CNF)

Variables $\{\, x_1 \ldots x_n \,\}$

Literals $L = \{\, x_i, \bar{x}_i \mid \text{variable } x_i \,\}$

Clauses $\{\, z_1, \ldots, z_k \,\} \subset L$

Propositional Satisfiability (SAT)

Instance Set of clauses S

Question $\exists$ setting of variables to 0, 1 such that each clause has at least one true literal?

SAT Example

$$
\begin{aligned}
\{\,\{\,1,2,3\,\},\{\,-1,-2,-3\,\}\,\} &= \{\,\{\,x_1,x_2,x_3\,\},\{\,\bar{x}_1,\bar{x}_2,\bar{x}_3\,\}\,\} \\
\text{(A)} \qquad &= (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)
\end{aligned}
$$

Truth Table

x_1	x_2	x_3	A
0	0	0	0
0	0	1	1
0	1	0	
	$\vdots$		

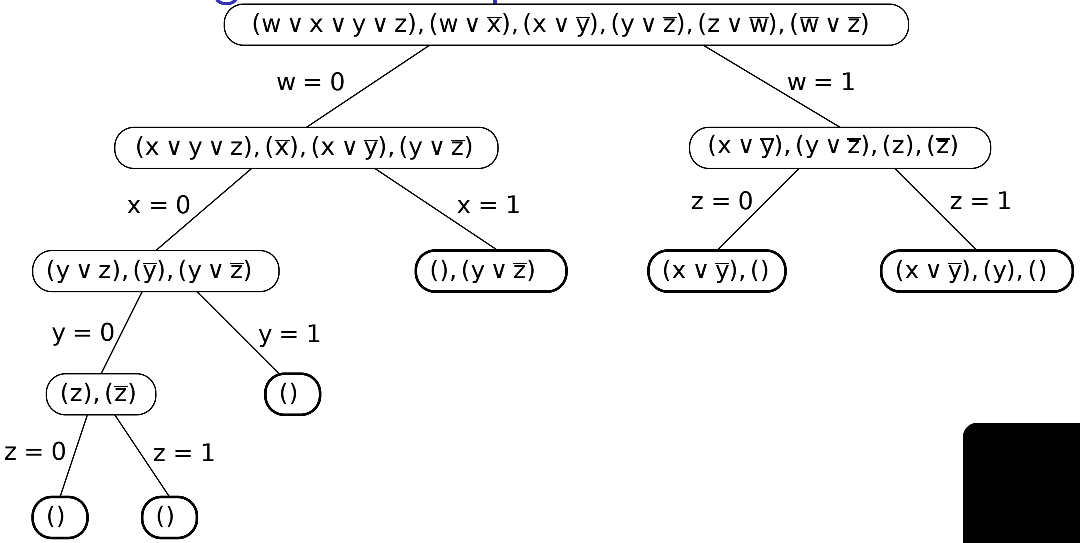
Backtracking for SAT

representation (reduced) clauses

test if empty clause, return `False`. If no clauses, return `True`. Otherwise return `None` (UNKNOWN)

expand $P_0 =$ reduce$(P, j, 0)$, $P_1 =$ reduce$(P, j, 1)$ for some j.

Backtracking SAT Example II



- We tried 11 possibilities. Maximum?

2SAT

- In 2SAT problem every clause has at most 2 elements
- 2SAT is solvable in polynomial time, but not quite trivially.
- Greedy fails on

$$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_4) \wedge (\bar{x}_4 \vee x_5) \wedge (\bar{x}_4 \vee x_6)$$

- to maximize number of clauses satisfied, choose $x_1 \leftarrow 1$, $x_4 \leftarrow 0$
- solvable with *unit propagation*

Horn SAT

Horn formulas

implication $(z \wedge w \wedge q) \Rightarrow u$. LHS is all positive, RHS one positive literal

negative clauses $(\bar{x} \vee \bar{w} \vee \bar{y})$. All literals negated.

[redacted]

Horn formulas as CNF

- negative clauses are already CNF
- implications use the following transformations

$$\left(\bigwedge_{i=1}^{k} x_i\right) \Rightarrow y$$

$$\neg\left(\bigwedge_{i=1}^{k} x_i\right) \vee y$$

$$\left(\bigvee_{i=1}^{k} \bar{x}_i\right) \vee y$$

- special CNF with at most one positive literal.

Unit propagation

```
def UnitProp(S):
 Q = [c for c in S if len(c)==1]; V=[]
 while len(Q)>0:
  z = Q.pop()[0]; T = [];  V.append(z)
  for C in S:
   C = [j for j in C if j!=-z]
   if len(C)==0: return (False,V)
   if len(C)==1: Q.append(C)
   if not z in C: T.append(C)
  if len(T)==0: return (True,V)
  S = T
 return S
```

Solving Horn SAT with Unit Propagation

1. Apply unit propagation
2. If no contradiction is detected, set the remaining variables to false.

Correctness

- If unit propagation returns `False`, the instance is unsatisfiable
- Otherwise, the resulting assignment is valid