Problem: A Diplomatic Problem

You are chief of protocol for the embassy ball. The crown prince instructs you either to invite Peru or to exclude Qatar. The queen asks you to invite either Qatar or Romania or both. The king, in a spiteful mood, wants to snub either Romania or Peru or both. Who do you invite?

- Three propositional symbols $\boldsymbol{P}, \boldsymbol{Q}$ and $\boldsymbol{R}$:

$\boldsymbol{P}$	$\equiv$	invite Peru	$\neg\boldsymbol{P}$	$\equiv$	exclude Peru
$\boldsymbol{Q}$	$\equiv$	invite Qatar	$\neg\boldsymbol{Q}$	$\equiv$	exclude Qatar
$\boldsymbol{R}$	$\equiv$	invite Romania	$\neg\boldsymbol{R}$	$\equiv$	exclude Romania

- Three operators: negation $\neg$, disjunction `or`,$\vee$,conjunction `&`,$\wedge$
- Exercise: 5 minutes to formalise the problem in PL.

Representing the Diplomatic Problem in PL

- The problem can be formalized as

prince:	invite Peru or exclude Qatar	$\equiv$	$\boldsymbol{P}$ or $\neg\boldsymbol{Q}$
queen:	invite Qatar or Romania or both	$\equiv$	$\boldsymbol{Q}$ or $\boldsymbol{R}$
king:	snub Romania or Peru or both	$\equiv$	$\neg\boldsymbol{R}$ or $\neg\boldsymbol{P}$

- $\boldsymbol{\Sigma} = (\boldsymbol{P}$ or $\neg\boldsymbol{Q})$ & $(\boldsymbol{Q}$ or $\boldsymbol{R})$ & $(\neg\boldsymbol{R}$ or $\neg\boldsymbol{P})$
- Any assignment (choice of truth or falsity of the symbols) is a solution for the diplomatic problem.
- For example, let $\boldsymbol{P}$ and $\boldsymbol{Q}$ be true, and $\boldsymbol{R}$ false, then $\boldsymbol{\Sigma}$ is true.
- The assignment $\boldsymbol{P}$ =T, $\boldsymbol{Q}$=T and $\boldsymbol{R}$ =F satisfies $\boldsymbol{\Sigma}$.
- Therefore: *invite Peru and Qatar, exclude Romania* is solution!

Reasoning: Solving the diplomatic problem?

- Reminder: $\Sigma = (P \text{ or } \neg Q) \ \& \ (Q \text{ or } R) \ \& \ (\neg R \text{ or } \neg P)$
- We can use truth tables

P	Q	R	P or $\neg Q$	Q or R	$\neg R$ or $\neg P$	Σ
T	T	T	T	T	F	F
T	T	F	T	T	T	T
T	F	T	T	T	F	F
T	F	F	T	F	T	F
F	T	T	F	T	T	F
F	T	F	F	T	T	F
F	F	T	T	T	T	T
F	F	F	T	F	T	F

- Problem: we need 2^n solutions, i.e. with 3 variables 8 rows, with 20 variables 1048576, and with 50 variables 1125899906842624.

Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?

2^n where n is the number of propositional symbols.

Can we do better?

SAT was the first problem shown to be $\mathcal{NP}$*-complete* [Coo71]: all of the problems in the class $\mathcal{NP}$ can be solved by translating them (in polynomial time) into SAT.

So, if we could somehow build a *fast* solver for SAT, it could be used to solve lots of other problems.

In theory, this seems dubious, as problems in $\mathcal{NP}$ are known to take exponential time in the worst case.

Remarkably, modern SAT solvers are very fast most of the time!

Complete Methods: Davis-Putnam

- Some comments on the DP algorithm
  - not developed by Davis & Putnam, but by Davis, Logemann & Loveland
  - the original algorithm proposed by Davis & Putnam used a resolution rule instead of the splitting rule, leading to potentially exponential use of space; the improved algorithm we show is what today is known as the DP algorithm
  - a **unit clause** is a clause containing a single literal; the only way to satisfy such a clause is by assigning its literal the true value
  - a **pure literal** is a literal appearing only once in the set of clauses; we can safely assign such a literal the true value without loosing any model

Complete Methods: Davis-Putnam 2

- Input: A set Σ of clauses, i.e. sets of Literals
- Output: **Yes**, if Σ is satisfiable, **no** otherwise.

```
procedure DP(Σ)
(Sat)      if Σ = ∅ then ‘‘sat’’
(Empty)    if Σ contains the empty clause then ‘‘unsat’’
(Taut)     if Σ contains p ∨ ¬p then DP(Σ \ {p ∨ ¬p})
(Unit Pr)  if Σ has unit clause {l} then DP(Σ{l = true})
(Pure)     if Σ has pure literal l then DP(Σ{l = true})
(Split)    if DP(Σ{l = true}) is satisfiable then ‘‘sat’’
           else DP(Σ{l = false})
```

- Remember $\Sigma = \{\{(P, \neg Q\}, \{Q, R\}, \{\neg R, \neg P\}\}$ is a way to represent $\Sigma = (P \vee \neg Q)\ \&\ (Q \vee R)\ \&\ (\neg R \vee \neg P)$
- **Exercise:** Prove or disprove satisfiability using DP.

Solution

$\Sigma = \{\{\{P, \neg R\}, \{P, \neg Q, R\}, \{Q, R, \neg P\}, \{\neg R, \neg P, Q\}\})$
$\longrightarrow$ (Split with P=T)
$DP(\Sigma)$ is sat if $DP(\Sigma\{P = T\})$ is sat
$\Sigma\{P = T\} = \{\{Q, R\}, \{\neg R, Q\}\}$
$\longrightarrow$ (Split with Q=F)
$DP(\Sigma\{P = T\})$ is sat if $DP(\Sigma\{P = T\}\{Q = F\})$ is sat
$\Sigma\{P = T\}\{Q = F\} = \{\{R\}, \{\neg R\}\}$
$\longrightarrow$ (Unit $\longrightarrow$ R=T)
$\Sigma\{\{P = T\}\{Q = F\}\{R = T\} = \{\varnothing\}$
$DP(\Sigma\{P = T\}\{Q = F\})$ contains the empty clause and is unsat
$\longrightarrow$ (Split with Q=T) — we need to backtrack
$DP(\Sigma\{P = T\})$ is sat if $DP(\Sigma\{P = T\}\{Q = T\})$ is sat
$\Sigma\{P = T\}\{Q = T\} = \varnothing$
$\longrightarrow DP(\Sigma\{P = T\}\{Q = T\})$ is sat
$\longrightarrow DP(\Sigma\{P = T\})$ is sat
$\longrightarrow DP(\Sigma)$ is sat

Davis-Putnam: The Rules

- The `(Pure)` rule is often deleted as the cost of evaluating it might offset its benefits
- Similarly for the `(Taut)` rule: tautologies only occur at the start of search
- The `(Unit)` rule is not essential as its effect can be obtained by combination of the `(Split)` and `(Empty)` rules
- But `(Unit)` is crucial for the extremely good performance of the method. For example, by itself the `(Unit)` rule is a complete procedure for Horn clauses

Davis-Putnam: The (Split) Rule

- The (Split) rule is non-deterministic: Which literal do we chose?
  - MOM's heuristics: pick the literal that occurs **m**ost **o**ften in the **m**inimal size clauses. This method is hard to beat for speed and simplicity.
  - Jeroslow-Wang's heuristics: estimate the contribution each literal l is likely to make to satisfying the clause set and pick the best

$$\text{score}(l) = \sum_{c \in \Sigma \ \& \ l \in c} 2^{-|c|}$$

for each clause c the literal l appears in $2^{-|c|}$ is added where $|c|$ is the number of literals in c.

DP: Search Strategies

- Organizing the **search** is a fundamental issue:
  - implement intelligent backtracking: e.g. use conflict-directed backjumping to skip over irrelevant variable assignments
  - lessen the effect of early mistakes: if the first branching decision is wrong, we may have to explore half search tree for nothing
  - several techniques can be used
    - randomization and restarts
    - discrepancy and interleaved search
  - parallelization: when to explore branches in parallel or sequentially? How to balance the loads on processors?

DP: Performance

- The worst case complexity of the algorithm we show is $O(\mathbf{1,696}^n)$, and a small modification moves it to $O(\mathbf{1,618}^n)$.
- This is an improvement!. . . Notice that, for example,

$$\begin{aligned} \mathbf{2}^{\mathbf{100}} &= \mathbf{1.267.650.000.000.000.000.000.000.000.000} \\ \mathbf{1,618}^{\mathbf{100}} &= \mathbf{790.408.700.000.000.000} \end{aligned}$$

- DP can reliably solve problems with up to 500 variables
- For CNF formulas where the probability of positive, negative or no-occurrence of a variable in a clause is 1/3, DP needs on average only **quadratic time**!