Davis Putnam ATP

Shane Aston
Spring 2015

Automated Theorem Prover

- Takes in Premises and a Conclusion
- Returns the validity of the argument
- Davis Putnam Algorithm
    - Herbrand's theorem
    - "A formula is valid if and only if its negation is unsatisfiable"*

*http://en.wikipedia.org/wiki/Davis%E2%80%93Putnam_algorithm

Davis Putnam Algorithm

- Negate Conclusion
- Convert each statement to CNF
- Turn CNFs into a set of clauses (S)
- Satisfiable(S)
- If True, Argument Invalid
- If False, Argument Valid

```
boolean Satisfiable(S):

    if S = {} return true;

    if S = {{}} return false

    select L ∈ lit(S);

    return Satisfiable(SL) ||
    Satisfiable(SL');
```

Add-ons

- Subsumption Elimination
  - [A,B] subsumes [A,B,C]
  - remove the subsumed statements
- Pure-Literal Elimination
  - $L \in S$ AND $\sim L \notin S$
  - remove all statements containing L
- Unit Literal
  - $[L] \in S$
  - no need to branch on L and ~L
- Tautological Elimination
  - [A, ~A]
  - [A, ~A, B]

The Program

- Python 2.7
- Command Line Interface
- Flags to determine usage of add-ons
- Hosted on GitHub

Input Formats

“Classic” format:	“New” format:
A implies (N or Q)	A -> (N v Q)
not(N or not A)	~(N v ~A)
A implies Q	A -> Q

Or some combination

CNF

- Putting in CNF is Hard…
- Two approaches:
  - Regular Expressions
  - Wolfram-Alpha

A <-> Q

~A <-> Q

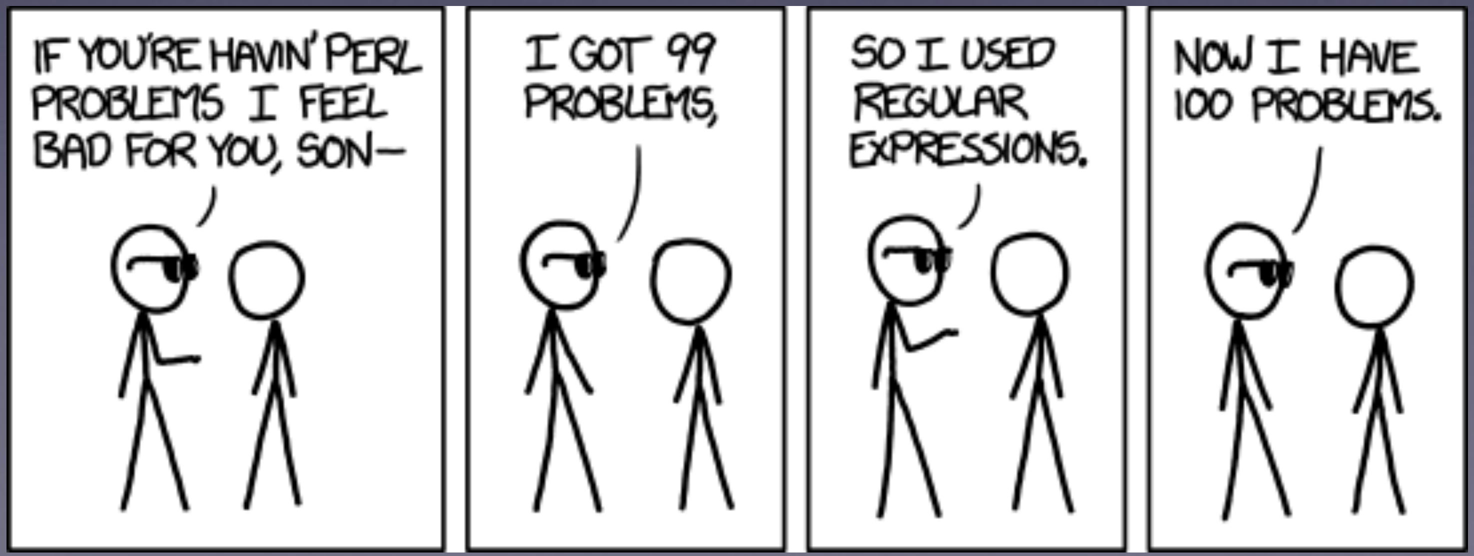
A <-> (N v Q)

~(N ^ (Q v ~P) ^ J) <-> A

Regular Expressions

- Fast
- Only for simple expressions

- A
    - re.match('(\w)$', line, re.I)
- A <-> ~B
    - m = re.match('(\w+) xnor NOT (\w+)$', line, re.I)



Wolfram Alpha

- Slow…
  - Multithreading
- Any complexity

- Free Developer API
  - 2000 queries/month
  - Simple HTTP requests
  - Returns XML

Demo!