

Davis Putnam ATP

Shane Aston
Spring 2015

Davis Putnam Algorithm

- Negate Conclusion
- Convert each statement to CNF
- Turn CNFs into clauses
- Run DP algorithm

```
boolean Satisfiable(S)
```

```
begin
```

```
    if  $S = \{\}$  return true;
```

```
    if  $S = \{\{\}\}$  return false
```

```
    select  $L \in \text{lit}(S)$ ;
```

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

```
end
```

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
 - $[L] \in S$
 - don't branch
- Tautological Elimination

Input Parsing

- Two formats:
 - “Classic”
 - Natural language (kinda)
 - “New”

Formats

“Classic” format:

A implies (N or Q)

not(N or not A)

A implies Q

“New” format:

$A \rightarrow (N \vee Q)$

$\sim(N \vee \sim A)$

$A \rightarrow Q$

CNF

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

$$A \leftrightarrow Q$$

$$\sim A \leftrightarrow Q$$

$$A \leftrightarrow (N \vee Q)$$

$$\sim(N \wedge (Q \vee \sim P) \wedge J) \leftrightarrow A$$

Regular Expressions

- Fast
- Only for simple expressions
- A
 - `re.match('(\w)$', line, re.I)`
- $A \leftrightarrow \sim B$
 - `m = re.match('(\w+) xnor NOT (\w+)$', line, re.I)`

Wolfram Alpha

- Slow...
- Any complexity
- Free Developer API
 - 2000 queries/month
 - Simple HTTP requests
 - Returns XML

Demo!