Logics and Statistics for Language Modeling

Carlos Areces

areces@loria.fr
http://www.loria.fr/~areces/ls
INRIA Nancy Grand Est
Nancy, France

2009/2010

Today's Program

Today's Program

- Clausal Form
- The Davis Putnam Method
- Small Demo Zchaff

Moving into Clausal Form

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means: No conjunctions inside disjunctions
Negations only on propositional symbols

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means: No conjunctions inside disjunctions
Negations only on propositional symbols

- Using the following equivalences:

$$(\neg(\varphi \vee \psi)) \quad \rightsquigarrow \quad (\neg\varphi \wedge \neg\psi)$$

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means: No conjunctions inside disjunctions
Negations only on propositional symbols

- Using the following equivalences:

$$\begin{array}{lcl} (\neg(\varphi \vee \psi)) & \rightsquigarrow & (\neg\varphi \wedge \neg\psi) \\ (\neg(\varphi \wedge \psi)) & \rightsquigarrow & (\neg\varphi \vee \neg\psi) \end{array}$$

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means:
No conjunctions inside disjunctions
Negations only on propositional symbols

- Using the following equivalences:

$$\begin{array}{rcl}
(\neg(\varphi \vee \psi)) & \rightsquigarrow & (\neg\varphi \wedge \neg\psi) \\
(\neg(\varphi \wedge \psi)) & \rightsquigarrow & (\neg\varphi \vee \neg\psi) \\
(\neg\neg\varphi) & \rightsquigarrow & \varphi
\end{array}$$

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means: No conjunctions inside disjunctions
Negations only on propositional symbols

- Using the following equivalences:

$$\begin{array}{rcl}
(\neg(\varphi \vee \psi)) & \rightsquigarrow & (\neg\varphi \wedge \neg\psi) \\
(\neg(\varphi \wedge \psi)) & \rightsquigarrow & (\neg\varphi \vee \neg\psi) \\
(\neg\neg\varphi) & \rightsquigarrow & \varphi \\
(\varphi \vee (\psi \wedge \theta)) & \rightsquigarrow & ((\varphi \vee \psi) \wedge (\varphi \vee \theta))
\end{array}$$

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means: No conjunctions inside disjunctions
Negations only on propositional symbols

- Using the following equivalences:

$$\begin{aligned}
(\neg(\varphi \vee \psi)) &\rightsquigarrow (\neg\varphi \wedge \neg\psi) \\
(\neg(\varphi \wedge \psi)) &\rightsquigarrow (\neg\varphi \vee \neg\psi) \\
(\neg\neg\varphi) &\rightsquigarrow \varphi \\
(\varphi \vee (\psi \wedge \theta)) &\rightsquigarrow ((\varphi \vee \psi) \wedge (\varphi \vee \theta)) \\
((\psi \wedge \theta) \vee \varphi) &\rightsquigarrow ((\varphi \vee \psi) \wedge (\varphi \vee \theta))
\end{aligned}$$

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means: No conjunctions inside disjunctions
Negations only on propositional symbols

- Using the following equivalences:

$$\begin{array}{rcl} (\neg(\varphi \vee \psi)) & \rightsquigarrow & (\neg\varphi \wedge \neg\psi) \\ (\neg(\varphi \wedge \psi)) & \rightsquigarrow & (\neg\varphi \vee \neg\psi) \\ (\neg\neg\varphi) & \rightsquigarrow & \varphi \\ (\varphi \vee (\psi \wedge \theta)) & \rightsquigarrow & ((\varphi \vee \psi) \wedge (\varphi \vee \theta)) \\ ((\psi \wedge \theta) \vee \varphi) & \rightsquigarrow & ((\varphi \vee \psi) \wedge (\varphi \vee \theta)) \end{array}$$

The clause set associated to

$$(l_{11} \vee \ldots \vee l_{1n_1}) \wedge (l_{21} \vee \ldots \vee l_{2n_2}) \wedge \ldots \wedge (l_{k1} \vee \ldots \vee l_{kn_k}) \quad \text{is}$$

Moving into Clausal Form

- Clausal Form: Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means: No conjunctions inside disjunctions
Negations only on propositional symbols

- Using the following equivalences:

$$\begin{array}{rcl}
(\neg(\varphi \vee \psi)) & \rightsquigarrow & (\neg\varphi \wedge \neg\psi) \\
(\neg(\varphi \wedge \psi)) & \rightsquigarrow & (\neg\varphi \vee \neg\psi) \\
(\neg\neg\varphi) & \rightsquigarrow & \varphi \\
(\varphi \vee (\psi \wedge \theta)) & \rightsquigarrow & ((\varphi \vee \psi) \wedge (\varphi \vee \theta)) \\
((\psi \wedge \theta) \vee \varphi) & \rightsquigarrow & ((\varphi \vee \psi) \wedge (\varphi \vee \theta))
\end{array}$$

The clause set associated to

$$(l_{11} \vee \ldots \vee l_{1n_1}) \wedge (l_{21} \vee \ldots \vee l_{2n_2}) \wedge \ldots \wedge (l_{k1} \vee \ldots \vee l_{kn_k}) \quad \text{is}$$
$$\{\{l_{11}, \ldots, l_{1n_1}\}, \ \{l_{21}, \ldots, l_{2n_2}\} \ , \ldots , \{l_{k1}, \ldots, l_{kn_k}\}\}$$

Example 1

Example 1

The Diplomatic Problem:

$$(P \vee \neg Q) \wedge (Q \vee R) \wedge (\neg R \vee \neg P)$$

Example 1

The Diplomatic Problem:

$$(P \vee \neg Q) \wedge (Q \vee R) \wedge (\neg R \vee \neg P)$$
$$\{\{P, \neg Q\}, \{Q, R\}, \{\neg R, \neg P\}\}$$

Example 2

Example 2

1. $\neg((p \vee q) \rightarrow (\neg q \rightarrow (p \vee q)))$

Example 2

1. $\neg((p \vee q) \rightarrow (\neg q \rightarrow (p \vee q)))$
2. $\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q)))$

Example 2

1. $\neg((p \vee q) \rightarrow (\neg q \rightarrow (p \vee q)))$
2. $\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q)))$
3. $\neg(\neg(p \vee q) \vee (q \vee (p \vee q)))$

Example 2

1. $\neg((p \vee q) \rightarrow (\neg q \rightarrow (p \vee q)))$
2. $\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q)))$
3. $\neg(\neg(p \vee q) \vee (q \vee (p \vee q)))$
4. $(\neg\neg(p \vee q) \wedge \neg(q \vee (p \vee q)))$

Example 2

1. $\neg((p \vee q) \rightarrow (\neg q \rightarrow (p \vee q)))$
2. $\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q)))$
3. $\neg(\neg(p \vee q) \vee (q \vee (p \vee q)))$
4. $(\neg\neg(p \vee q) \wedge \neg(q \vee (p \vee q)))$
5. $((p \vee q) \wedge \neg(q \vee (p \vee q)))$

Example 2

1. $\neg((p \vee q) \rightarrow (\neg q \rightarrow (p \vee q)))$
2. $\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q)))$
3. $\neg(\neg(p \vee q) \vee (q \vee (p \vee q)))$
4. $(\neg\neg(p \vee q) \wedge \neg(q \vee (p \vee q)))$
5. $((p \vee q) \wedge \neg(q \vee (p \vee q)))$
6. $((p \vee q) \wedge (\neg q \wedge \neg(p \vee q)))$

Example 2

1. $\neg((p \vee q) \rightarrow (\neg q \rightarrow (p \vee q)))$
2. $\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q)))$
3. $\neg(\neg(p \vee q) \vee (q \vee (p \vee q)))$
4. $(\neg\neg(p \vee q) \wedge \neg(q \vee (p \vee q)))$
5. $((p \vee q) \wedge \neg(q \vee (p \vee q)))$
6. $((p \vee q) \wedge (\neg q \wedge \neg(p \vee q)))$
7. $((p \vee q) \wedge (\neg q \wedge (\neg p \wedge \neg q)))$

Example 2

1. $\neg((p \vee q) \rightarrow (\neg q \rightarrow (p \vee q)))$
2. $\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q)))$
3. $\neg(\neg(p \vee q) \vee (q \vee (p \vee q)))$
4. $(\neg\neg(p \vee q) \wedge \neg(q \vee (p \vee q)))$
5. $((p \vee q) \wedge \neg(q \vee (p \vee q)))$
6. $((p \vee q) \wedge (\neg q \wedge \neg(p \vee q)))$
7. $((p \vee q) \wedge (\neg q \wedge (\neg p \wedge \neg q)))$
8. $\{\{p, q\}, \{\neg q\}, \{\neg p\}\}$

Example 3

Example 3

1. $(p \leftrightarrow q) \vee r$

Example 3

1. $(p \leftrightarrow q) \vee r$
2. $((p \rightarrow q) \wedge (q \rightarrow p)) \vee r$

Example 3

1. $(p \leftrightarrow q) \vee r$
2. $((p \rightarrow q) \wedge (q \rightarrow p)) \vee r$
3. $((\neg p \vee q) \wedge (\neg q \vee p)) \vee r$

Example 3

1. $(p \leftrightarrow q) \vee r$
2. $((p \rightarrow q) \wedge (q \rightarrow p)) \vee r$
3. $((\neg p \vee q) \wedge (\neg q \vee p)) \vee r$
4. $(((\neg p \vee q) \vee r) \wedge ((\neg q \vee p) \vee r))$

Example 3

1. $(p \leftrightarrow q) \vee r$
2. $((p \rightarrow q) \wedge (q \rightarrow p)) \vee r$
3. $((\neg p \vee q) \wedge (\neg q \vee p)) \vee r$
4. $(((\neg p \vee q) \vee r) \wedge ((\neg q \vee p) \vee r))$
5. $\{\{\neg p, q, r\}, \{\neg q, p, r\}\}$

The Davis-Putnam Algorithm

The Davis-Putnam Algorithm

- The Davis-Putnam method is perhaps one of the most widely used algorithms for solving the SAT problem of PL

The Davis-Putnam Algorithm

- The Davis-Putnam method is perhaps one of the most widely used algorithms for solving the SAT problem of PL
- Despite its age, it is still one of the most popular and successful complete methods

The Davis-Putnam Algorithm

- The Davis-Putnam method is perhaps one of the most widely used algorithms for solving the SAT problem of PL
- Despite its age, it is still one of the most popular and successful complete methods

Let Σ be the clause set associated to a formula φ

```
procedure DP(Σ)
```

The Davis-Putnam Algorithm

- The Davis-Putnam method is perhaps one of the most widely used algorithms for solving the SAT problem of PL
- Despite its age, it is still one of the most popular and successful complete methods

Let Σ be the clause set associated to a formula φ

```
procedure DP(Σ)
if Σ={} then return SAT                    // (SAT)
```

The Davis-Putnam Algorithm

- The Davis-Putnam method is perhaps one of the most widely used algorithms for solving the SAT problem of PL
- Despite its age, it is still one of the most popular and successful complete methods

Let Σ be the clause set associated to a formula φ

```
procedure DP(Σ)
if Σ={} then return SAT                 // (SAT)
if {} ∈ Σ then return UNSAT             // (UNSAT)
```

The Davis-Putnam Algorithm

- The Davis-Putnam method is perhaps one of the most widely used algorithms for solving the SAT problem of PL
- Despite its age, it is still one of the most popular and successful complete methods

Let Σ be the clause set associated to a formula φ

```
procedure DP(Σ)
if Σ={} then return SAT                 // (SAT)
if {} ∈ Σ then return UNSAT             // (UNSAT)
if Σ has unit clause {l}
   then DP(Σ[{l=true}])                 // (Unit Pr.)
```

The Davis-Putnam Algorithm

- The Davis-Putnam method is perhaps one of the most widely used algorithms for solving the SAT problem of PL
- Despite its age, it is still one of the most popular and successful complete methods

Let Σ be the clause set associated to a formula φ

```
procedure DP(Σ)
if Σ={} then return SAT                    // (SAT)
if {} ∈ Σ then return UNSAT                // (UNSAT)
if Σ has unit clause {l}
   then DP(Σ[{l=true}])                    // (Unit Pr.)
Choose literal l and
   if DP(Σ[{l=true}]) return SAT
      then return SAT
      else return DP(Σ[{l=false}])         // (Split)
```

Examples

Examples

$$\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q))) \ -\mathrm{CNF}\!\rightarrow\ \{\{p, q\}, \{\neg q\}, \{\neg p\}\}$$

Examples

$$\neg(\neg(p \vee q) \vee (\neg\neg q \vee (p \vee q))) \ -\mathrm{CNF}\!\rightarrow \{\{p, q\}, \{\neg q\}, \{\neg p\}\}$$

$$\{\{P, \neg Q\}, \{Q, R\}, \{\neg R, \neg P\}\}$$

DP: Performance

DP: Performance

- The worst case complexity of the algorithm we show is $O(1.696^n)$, and a small modification moves it to $O(1.618^n)$.
- This is an improvement!...

DP: Performance

- The worst case complexity of the algorithm we show is $O(1.696^n)$, and a small modification moves it to $O(1.618^n)$.
- This is an improvement!... Notice that, for example,

$$2^{100} = 1.267.650.000.000.000.000.000.000.000$$

DP: Performance

- The worst case complexity of the algorithm we show is $O(1.696^n)$, and a small modification moves it to $O(1.618^n)$.
- This is an improvement!... Notice that, for example,

$$\begin{array}{rcr} 2^{100} & = & 1.267.650.000.000.000.000.000.000.000 \\ 1.696^{100} & = & 87.616.270.000.000.000.000.000 \end{array}$$

DP: Performance

- The worst case complexity of the algorithm we show is $O(1.696^n)$, and a small modification moves it to $O(1.618^n)$.
- This is an improvement!... Notice that, for example,

$$
\begin{array}{rcr}
2^{100} & = & 1.267.650.000.000.000.000.000.000.000.000 \\
1.696^{100} & = & 87.616.270.000.000.000.000.000 \\
1.618^{100} & = & 790.408.700.000.000.000.000
\end{array}
$$

DP: Performance

- The worst case complexity of the algorithm we show is $O(1.696^n)$, and a small modification moves it to $O(1.618^n)$.
- This is an improvement!... Notice that, for example,

$$\begin{array}{rcr} 2^{100} & = & 1.267.650.000.000.000.000.000.000.000.000 \\ 1.696^{100} & = & 87.616.270.000.000.000.000.000 \\ 1.618^{100} & = & 790.408.700.000.000.000.000 \end{array}$$

- DP can reliably solve problems with up to 500 variables

DP: Performance

- The worst case complexity of the algorithm we show is $O(1.696^n)$, and a small modification moves it to $O(1.618^n)$.
- This is an improvement!... Notice that, for example,

$$\begin{array}{rcr} 2^{100} & = & 1.267.650.000.000.000.000.000.000.000.000 \\ 1.696^{100} & = & 87.616.270.000.000.000.000.000 \\ 1.618^{100} & = & 790.408.700.000.000.000.000 \end{array}$$

- DP can reliably solve problems with up to 500 variables
- Sadly real world applications easily go into the thousands of variables (remember coloring: #nodes $\times$ #colors).

DP: Performance

- The worst case complexity of the algorithm we show is $O(1.696^n)$, and a small modification moves it to $O(1.618^n)$.
- This is an improvement!... Notice that, for example,

$$
\begin{array}{rcr}
2^{100} & = & 1.267.650.000.000.000.000.000.000.000.000 \\
1.696^{100} & = & 87.616.270.000.000.000.000.000 \\
1.618^{100} & = & 790.408.700.000.000.000.000
\end{array}
$$

- DP can reliably solve problems with up to 500 variables
- Sadly real world applications easily go into the thousands of variables (remember coloring: #nodes × #colors).
- But this is worst time complexity. You might get lucky...

Zchaff

- A highly optimized system implementing a 'flavor' of DP (known as the chaff algorithm).
- Site: `http://www.princeton.edu/~chaff/zchaff.html`
- Also known as the 'Princeton Prover'.
- Success stories of zChaff solving problems with more than one million variables and 10 million clauses. (Of course, it can't solve every such problem!).
- Integrated into the AI Planner BlackBox, the Model Checker NuSMV, the Theorem Prover GrAnDe, etc.