Informatics 1

Lecture 8 Resolution (continued)

Michael Fourman

"I am never really satisfied that I understand anything, because, understand it well as I may, my comprehension can only be an infinitesimal fraction of all I want to understand."

Ada Lovelace, the world's first programmer, student of de Morgan, who taught her mathematics.

In this lecture we consider formal descriptions of the relationships between a finite number of individuals. We may have different types of individual

ADA LOVELACE

World's First Computer Programmer (1815 — 1852)

Ada Lovelace was the daughter of the poet Lord Byron. Her mother wanted her to be nothing like her poet father, so she made sure she had a strong education in science and mathematics. She is famous for translating the notes of Italian mathematician Menabrea on the subject of Charles Babbage's Analytical Engine. Her notes on his article were longer than the article itself. In her later correspondance with Babbage, she suggested that such a machine could later be used for composing music and producing graphics. Her prediction was correct. She also wrote instructions for the machine to calculate Bernouli numbers: the world's very first computer program.

Again, [the Analytical Engine] might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine . . . Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.

In 1852, when only 37 years of age, Ada died of cancer.

Premises

$$\frac{X \lor \neg \mathbf{D} \qquad Y \lor \mathbf{D}}{\mathbf{X} \lor \mathbf{Y}}$$

Conclusion

A valid inference	Any assignment of truth values that makes all the premises true will make the conclusion true.

The conclusion follows from the premises

Premises

$$\frac{X \lor \neg D \qquad Y \lor D}{X \lor Y}$$

Conclusion

For any valid inference	Any assignment of truth values that makes the conclusion false will make at least one of the premises false.

Premises

$$\frac{X \vee \neg\mathbf{D} \qquad Y \vee \mathbf{D}}{\mathbf{X} \vee \mathbf{Y}}$$

Conclusion

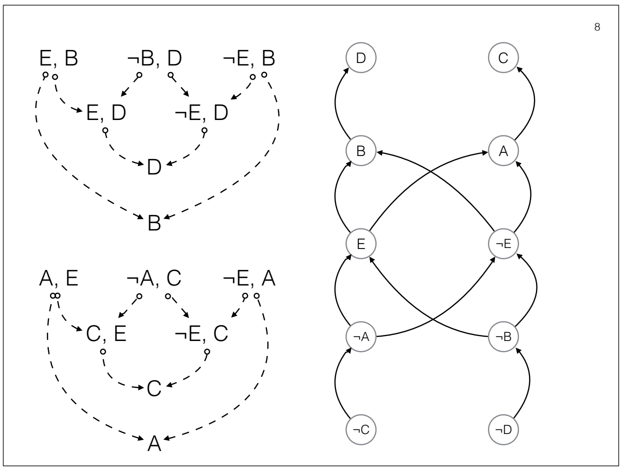
where D does not occur in X or Y

*A **special property** of this inference*

If some assignment abc of values for ABC makes the conclusion false then the assignments abc**D** and abc$\bar{\mathbf{D}}$ each make one or other of the two premises false.

Resolution

$$\frac{U \lor V \lor W \lor X \lor \neg\mathbf{C} \qquad X \lor Y \lor Z \lor \mathbf{C}}{U \lor V \lor W \lor X \lor Y \lor Z}$$



Resolution gives the same information as our earlier graphical analysis.

Clausal Form

Resolution uses CNF
a conjunction of disjunctions of literals
$(\neg A \lor C) \land (\neg B \lor D) \land (\neg E \lor B) \land (\neg E \lor A) \land (A \lor E) \land (E \lor B) \land (\neg B \lor \neg C \lor \neg D)$

Clausal form is a set of sets of literals
$\{\{\neg A, C\}, \{\neg B, D\}, \{\neg E, B\}, \{\neg E, A\}, \{A, E\}, \{E, B\}, \{\neg B, \neg C, \neg D\}\}$
Each set of literals represents the disjunction of its literals.
An empty disjunction {} represents false $\bot$.

The clausal form represents the conjunction of these disjunctions (an empty conjunction $\{\}$ represents true $\top$).

Using sets builds in idempotence, associativity and commutativity.

Clausal Form

Clausal form is a set of sets of literals

{ {¬A,C}, {¬B,D}, {¬E,B}, {¬E,A}, {A,E}, {E,B},{¬B, ¬C, ¬D} }

A (partial) truth assignment makes a clause true
iff it makes at least one of its literals true
(so it can never make the empty clause {} true)

A (partial) truth assignment makes a clausal form true
iff it makes all of its clauses true
(so the empty clausal form {} is always true).

Clausal form is a set of sets of literals

$\{ \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \}$

where $x_i = \{ L_0, \ldots, L_{m_i-1} \}$

Resolution rule for clauses

$$\frac{\mathbf{X} \qquad \mathbf{Y}}{(\mathbf{X} \cup \mathbf{Y}) \setminus \{ \neg A, A \}} \quad \text{where } \neg A \in \mathbf{X},\ A \in \mathbf{Y}$$

If a valuation makes everything in the conclusion false then that valuation must make everything in one or other of the premises false.

If it makes A true, then it makes everything in **X** false
If it makes A false, then it makes everything in **Y** false

If we have derived {} by resolution, then, for any valuation we are given, the special property lets us find a constraint that it violates. So there are no valuations satisfying all the constraints.

Davis Putnam

Take a collection $\mathcal{C}$ of clauses.

For each propositional letter, A
    For each pair $(X, Y) \mid X \in \mathcal{C} \wedge Y \in \mathcal{C} \wedge \mathsf{A} \in X \wedge \neg\mathsf{A} \in Y$
        if $\mathcal{R}(X, Y, \mathsf{A}) = \{\}$ return UNSAT
        if $\mathcal{R}(X, Y, \mathsf{A})$ is contingent $\mathcal{C} := \mathcal{C} \cup \{ \mathcal{R}(X, Y) \}$
    remove any clauses containing A or $\neg\mathsf{A}$
return SAT

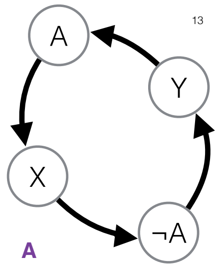
Where $\mathcal{R}(X, Y, \mathsf{A}) = X \cup Y \setminus \{ \mathsf{A}, \neg\mathsf{A} \}$, and
a clause is contingent if does not contain any
complementary pair of literals

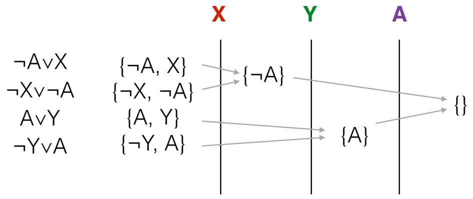
Heuristic: start with variables that occur seldom.

On this slide, indentation indicates grouping. So, for each atom, we resolve all pairs satisfying $A \in X \wedge \neg A \in Y$. Once all the A-resolvants have been produced we can forget about clauses containing A or ¬A.

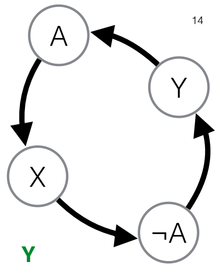
Removing clauses that contain A or ¬A will not prevent us from deriving the empty clause {}, if it can be derived. However, before we remove them, we must ensure that we have resolved **all** A, ¬A pairs.
If a clause contains both A and ¬A, then it is a tautology, and does not constrain the search.

A contradictory cycle





A contradictory cycle



		A	X	Y
¬A∨X	{¬A, X}	{¬X, Y}	{Y}	
¬X∨¬A	{¬X, ¬A}	{X, Y}	{¬Y}	{}
A∨Y	{A, Y}	{X, ¬Y}	{Y, ¬Y}	
¬Y∨A	{¬Y, A}	{¬X, ¬Y}		

15

¬A, C

¬B, D

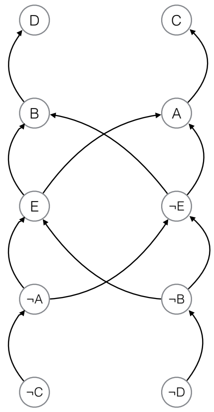
¬E, B

¬E, A

A, E

E, B

¬C, ¬D



By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.

A

¬A, C

¬B, D

¬E, B

¬E, A

A, E

E, B

¬C, ¬D

By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.

A

¬A, C	¬E, C
¬B, D	C, E
¬E, B	

¬E, A

A, E

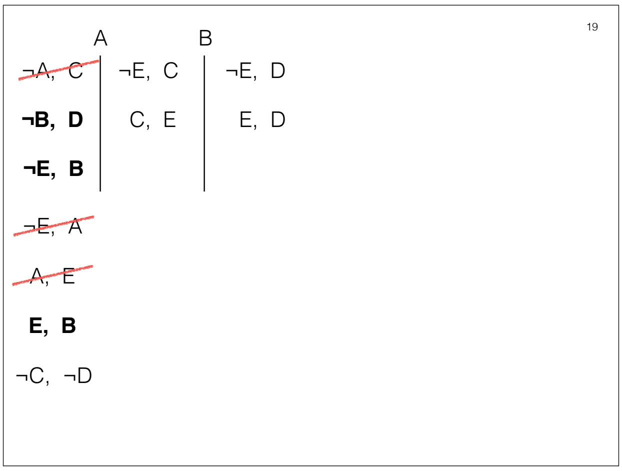
E, B

¬C, ¬D

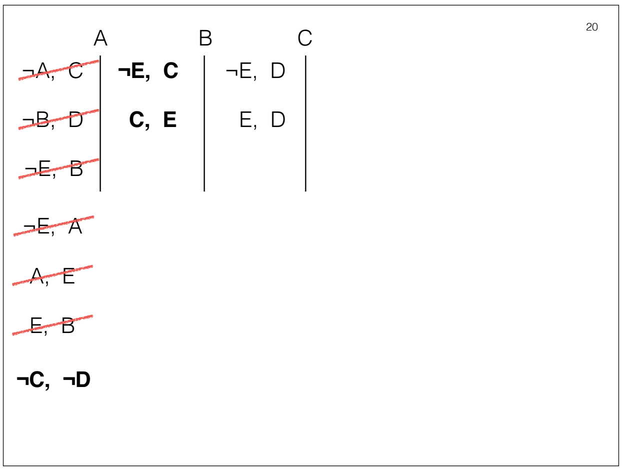
By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.

A	B
~~¬A, C~~	¬E, C
¬B, D	C, E
¬E, B	
~~¬E, A~~	
~~A, E~~	
E, B	
¬C, ¬D	

By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.



By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.



By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.

21

	A	B	C
~~¬A, C~~	**¬E, C**	¬E, D	¬E, ¬D
~~¬B, D~~	**C, E**	E, D	¬D, E
~~¬E, B~~			

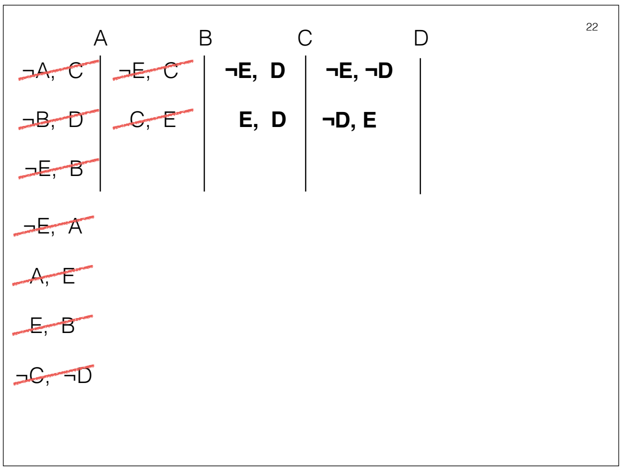
~~¬E, A~~

~~A, E~~

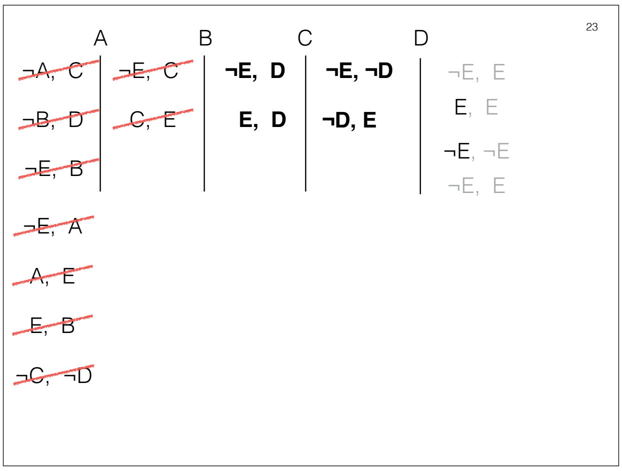
~~E, B~~

¬C, ¬D

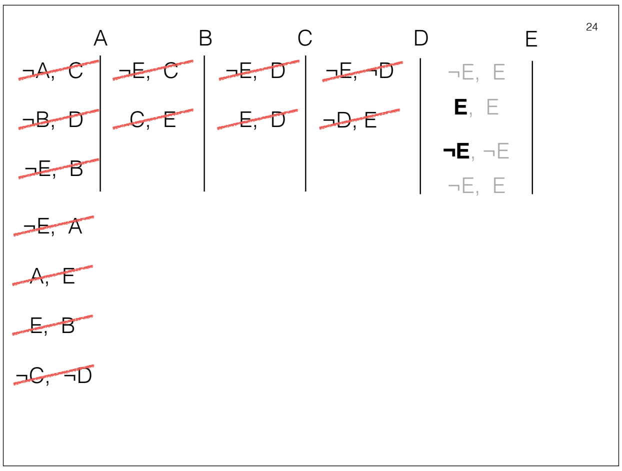
By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.



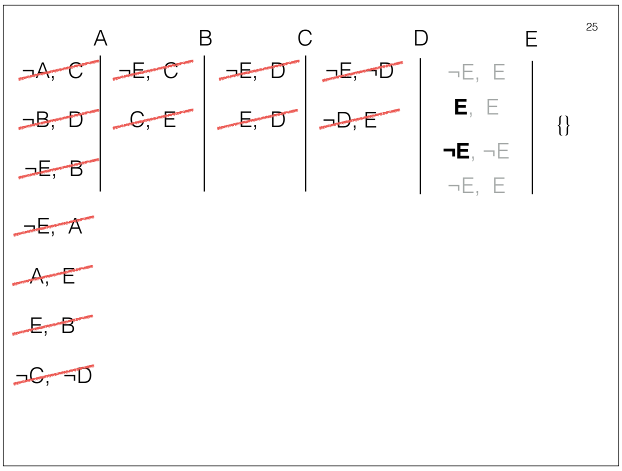
By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.



By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.



By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.



By our analysis of the picture, we know that any valuation satisfying the binary constraints must make A, B, C, D all true. So adding this new constraint makes an inconsistent set of constraints. Use resolution to show this directly.

Resolution

A complete proof procedure for propositional logic that works on formulas expressed in conjunctive normal form. (Robinson 1965)

Conjunctive Normal Form (CNF)

Literal: a propositional variable p or its negation ¬p
Clause: a disjunction of (a set of) literals.
CNF: a conjunction of clauses.

Resolution

From two clauses

$$C_1 = (X \cup \{A\}),\ C_2 = (Y \cup \{\neg A\})$$

the resolution rule generates the new clause

$$(X \cup Y) = R(C_1, C_2)$$

where X and Y are sets of literals, not containing A or ¬A.

$(X \cup Y)$ is the resolvant
A is the variable resolved on

Resolution

A **resolution refutation** of a CNF F (a set of clauses) is a sequence $C_1, C_2, \dots, C_m$ of clauses such that

$C_m = \{\}$, and

each C_i is either

a member of F

or

the resolvant of two previous clauses in the proof:

$C_i = R(C_j, C_k)$, where $j,k < i$

Resolution

Any resolution proof can be represented as a DAG nodes are clauses in the proof.

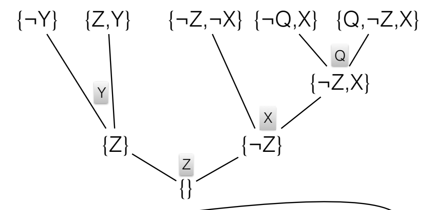
Clauses in F are leaves: they have no incoming edges.

Every clause C_i that arises from a resolution step has two incoming edges. One from each of the clauses (C_j,C_k) that were resolved together to obtain $C_i = R(C_j,C_k)$.

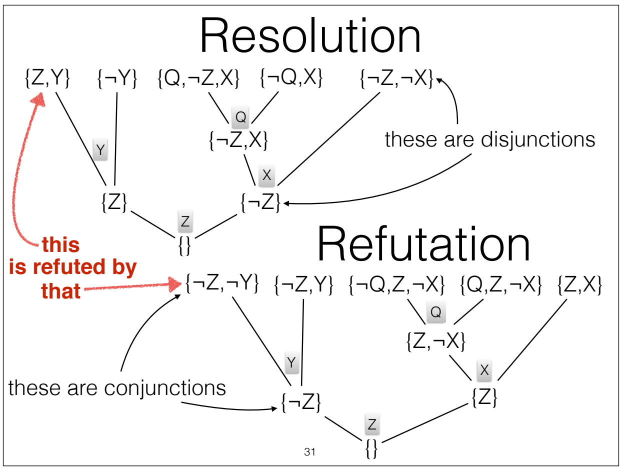
Each non-leaf node C_i is labeled by the variable that was resolved away to obtain it.

Resolution

F = {{¬Y}, {Z,Y}, {¬Z,¬X}, {¬Q,X}, {Q,¬Z,X}}



[**{¬Y}, {Z,Y}, {¬Z,¬X}, {¬Q,X}, {Q,¬Z,X}**,{¬Z,X}, {¬Z}, {Z},{}]



From the resolution proof we cn derive a refutation.
The lower tree demonstrates the fact that whatever values we choose for the variables, we will arrive at a clause that is false for our chosen values. This suffices to show that, no matter what choice of values we make, the conjunction is false. The CNF is not satisfiable.
In the next lecture we will build the refutation tree directly, by searching for a satisfying valuation.

(A?B:C)