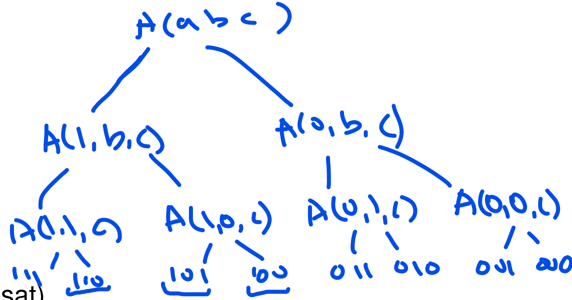
Lecture 10: Resolution, friends + SAT

Barnaby Martin
barnaby.d.martin@durham.ac.uk

A search algorithm

A(a, b, c) $a \wedge b \wedge c \Rightarrow b \vee c$



- Algorithm A(F)
- outputs *sat* or *unsat*
- Pseudocode:

if $\mathrm{var}(F) = \emptyset$ then
  if $F = \emptyset$ then exit(sat).
  else exit(unsat). /* $F = \{\Box\}$ where $\Box$ is empty clause */
else choose $x \in \mathrm{var}(F)$,
/* x is the branching variable */
if A($F[x = 0]$) = sat then exit(sat).
else if A($F[x = 1]$) = sat then exit(sat).
else exit(unsat).

$\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \Rightarrow \varphi_4$

$\varphi_1, \varphi_2, \varphi_3 \vdash \varphi_4$

Sequent

$\ldots \vdash \varphi$

DPLL

$\boxed{1+2+3+4+5+6}$

reduction 3, 3, 4, 5, 6

6, 4, 5, 6

10, 5, 6

15, 6

21 → return

- Algorithm A is the foundation of the *Davis-Putnam-Logmann-Loveland (DPLL) algorithm*.
- Essentially, DPLL is algorithm A plus some *data reduction rules*.
- Complete state-of-the-art SAT solvers are based on DPLL.

- Most complete AR solvers use an inference system.
- For propositional logic, it's usually a variant of resolution.

Resolution Rule

- Let C and D be two clauses, and let $x \in C$ and $\overline{x} \in D$.
- The resolution rule allows us to derive from C and D the clause $(C \setminus \{x\}) \cup (D \setminus \{\overline{x}\})$.
- As a figure:

$$\frac{C \qquad\qquad D}{(C \setminus \{x\}) \cup (D \setminus \{\overline{x}\})} \qquad \frac{\neg p \vee q \qquad p \vee r}{q \vee r}$$

- $(C \setminus \{x\}) \cup (D \setminus \{\overline{x}\})$ is a resolvent of C and D wrt literal x.
- Example:

$$\frac{\{x, y, \overline{z}\} \qquad\qquad \{v, \overline{y}, \overline{z}\}}{\{v, x, \overline{z}\}}$$

Resolution Derivations

- A resolution derivation from a clause-set F is a sequence of clauses $C_1, \ldots, C_s$ such that, for each i,
  - — $C_i \in F$, i.e. C_i is an axiom, or
  - — C_i is the resolvent of clauses C_j, and C_k for $1 \leq j, k < i$, C_i is a derived clause
- A clause C can be derived by resolution from F if there is a resolution derivation $\mathcal{D}$ from F that containing C; written

$$F \vdash_R C$$

Example

$a, b, c, \quad a, \bar{b} \quad \bar{a}, b \quad \bar{a}, \bar{b}, \bar{c}, \quad \bar{b}, c$

- $F = \{\{x, y, z\}, \{x, \overline{z}\}, \{\overline{x}, y\}, \{\overline{x}, \overline{y}, \overline{z}\}, \{\overline{y}, z\}\}$.
- A resolution derivation from F:
  1. $C_1 = \{x, y, z\}$ (axiom)
  2. $C_2 = \{x, \overline{z}\}$ (axiom)
  3. $C_3 = \{x, y\}$ (from C_1 and C_2)
  4. $C_4 = \{\overline{x}, y\}$ (axiom)
  5. $C_5 = \{y\}$ (from C_3 and C_4)
  6. $C_6 = \{\overline{x}, \overline{y}, \overline{z}\}$ (axiom)
  7. $C_7 = \{\overline{y}, \overline{z}\}$ (from C_6 and C_2)
  8. $C_8 = \{\overline{y}, z\}$ (axiom)
  9. $C_9 = \{\overline{y}\}$ (from C_7 and C_8)
  10. $C_{10} = \square$ (from C_5 and C_9)

$\{C_1, C_2\} \vdash_R \{C_3\}$

$\{\{x, y, z\}, \{x, \bar{z}\}\} \vdash_R \{x, y\}$

- Hence $F \vdash_R \square$.

Refutations

A resolution refutation = resolution derivation containing $\Box$

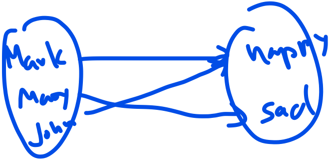
Theorem

A clause-set F is unsatisfiable iff $F \vdash_R \Box$

In words, resolution is

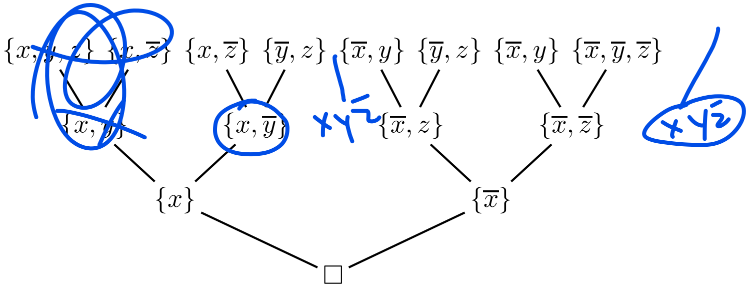
- sound, i.e. it can refute only unsat clause-sets, and
- refutationally complete, i.e. any unsat clause-set can be refuted in it.

Mark, Mary, John $\Rightarrow \{\Box\}$



Resolution Trees

Tree-like resolution: If every derived clause in a resolution derivation is used at most once for obtaining a clause below.



Resolution trees and DAGs

Fact

Tree-like resolution is sound and refutationally complete.

- If subderivations are shared, a resolution derivation can be represented by a DAG.
- dag-like resolution = general resolution.
- dag-like resolution derivations can be significantly shorter than tree-like resolution derivations.

- Algorithm A for satisfiability can be modified so that it outputs
  - a satisfying truth assignment if the input F is sat, and
  - a tree-like resolution refutation if input F is unsat.
- In both cases it is easy to verify the certificate.

Long and short resolution refutations

- Let F be a clause-set and R a resolution refutation of F.
- We say the resolution refutation R is *short* if the number of clauses in R is polynomial in the number of clauses of F.
- Otherwise we call the resolution refutation *long*.
- Is there a family of unsatisfiable clause-sets that require long resolution refutations?
- If every unsatisfiable clause-set had a short resolution refutation, then NP $=$ coNP would follow (which is believed not to be the case).

Pigeonhole clause-sets

- Let $\mathbf{PH}_n$ be the clause-set that represents the (untrue) statement that $n + 1$ items can be fitted into n drawers such that each drawer contains at most one item.
- $\mathbf{PH}_n$ contains variables $x_{i,j}$ for $1 \le i \le n + 1$ and $1 \le j \le n$; we think of $x_{i,j}$ as true if item i is in drawer j.
- $\mathbf{PH}_n$ contains two groups of clauses:
  - a clause $\{x_{i,1}, \ldots, x_{i,n}\}$ for each $1 \le i \le n + 1$ "item i is in one of the n drawers";
  - a clause $\{\overline{x_{i,j}}, \overline{x_{i',j}}\}$ for each $1 \le i < i' \le n + 1$ and $1 \le j \le n$ "in each drawer there is at most one item".
- Observe that $\mathbf{PH}_n$ is unsatisfiable.

Haken's Theorem

Theorem (Haken, 1985)

Pigeonhole clause-set require long resolution refutations.

A shortest resolution refutation of $\mathbf{PH}_n$ contains $2^{\Omega(n)}$ clauses.

Later, Chávtal and Szemerédi have shown that randomly chosen unsatisfiable k-SAT clause-sets require long resolution refutations.

Linear Resolution

- A resolution derivation linear if it has the following form: $C_1, D_1, C_2, D_2, \ldots, C_{n-1}, D_{n-1}, C_n$ where
  - C_1 is an axiom, and each D_i is either an axiom or C_j for some $j < i$.
  - C_{i+1} is a resolvent of C_i and D_i, for $1 \leq i \leq n-1$.
- The clause C_1 is the *base clause*, the clauses D_i are called *side clauses*.

Theorem

Linear resolution is refutationally complete.

- A linear resolution derivation is called an input resolution derivation if all side clauses are axioms.
- Note that input resolution derivations are always tree-like but linear resolution derivations are not always tree-like.

Example

- $F = \{\{x, y, z\}, \{x, \overline{z}\}, \{\overline{x}, y\}, \{\overline{x}, \overline{y}, \overline{z}\}, \{\overline{y}, z\}\}$.
- A linear resolution refutation of F
  1. $C_1 = \{x, y, z\}$ (axiom)
  2. $D_1 = \{x, \overline{z}\}$ (axiom)
  3. $C_2 = \{x, y\}$ (from C_1 and D_1)
  4. $D_2 = \{\overline{x}, y\}$ (axiom)
  5. $C_3 = \{y\}$ (from C_2 and D_2)
  6. $D_3 = \{\overline{x}, \overline{y}, \overline{z}\}$ (axiom)
  7. $C_4 = \{\overline{x}, \overline{z}\}$ (from C_3 and D_3)
  8. $D_4 = \{x, \overline{z}\}$ (axiom)
  9. $C_5 = \{\overline{z}\}$ (from C_4 and D_4)
  10. $D_5 = \{\overline{y}, z\}$ (axiom)
  11. $C_6 = \{\overline{y}\}$ (from C_5 and D_5)
  12. $D_6 = \{y\}$ $(= C_3)$
  13. $C_7 = \square$ (from C_6 and D_6)
- Is this an input resolution derivation?

Incompleteness of Input Resolution

- Let $F = \{\{x, y\}, \{\overline{x}, y\}, \{x, \overline{y}\}, \{\overline{x}, \overline{y}\}\}$.
- The following is a linear resolution refutation of F.
  $C_1 = \{x, y\}$, $D_1 = \{x, \overline{y}\}$, $C_2 = \{x\}$, $D_2 = \{\overline{x}, y\}$,
  $C_3 = \{y\}$, $D_3 = \{\overline{x}, \overline{y}\}$ $C_4 = \{\overline{x}\}$, $D_4 = C_2 = \{x\}$, $C_5 = \square$.
- However, F has no input resolution refutation! (Why?)

Theorem

Input resolution is not refutationally complete.

- We shall see that input resolution is refutationally complete if all input clauses have a special property: they are Horn clauses.

Horn clauses and unit resolution

- A clause is a Horn clause if it contains at most one positive literal.
- It is called definite Horn clause if it contains exactly one positive literal.
- Examples: $\{\overline{x}, \overline{z}, y\}$, $\{\overline{x}, \overline{z}, \overline{y}\}$, $\{x\}$, $\{\overline{x}\}$, $\square$ are Horn clauses. The first and third clauses are definite Horn.
- Rules and facts can be directly stated as Horn clauses.
- A clause-set is called (definite) Horn if it contains only (definite) Horn clauses.
- Unit clause = a clause with one literal
- Unit resolution = every step involves a unit clause

Theorem

A Horn clause-set F is unsatisfiable if and only if $F \vdash_{UR} \square$.

SLD Resolution

SLD Resolution is a special type of linear resolution, it is defined for Horn clauses only. SLD stands for **linear** resolution with **selection** function for **definite** clauses.

An input resolution derivation is an SLD resolution derivation if:

- the base clause is a *negative clause*, i.e., all its literals are negative.
- Each side clause is a *definite Horn clause*, i.e., it contains exactly one positive literal.

Theorem

SLD resolution is refutationally complete for Horn clause-sets.

Logic Programming

- The program consists of an ordered list of definite horn clauses.
- Each clause is given as an ordered set of literals, with the positive literal first.
  E.g., the clause $\{x, \overline{y}, \overline{z}\}$ is given as
  ```
  x <== y and z.
  ```
  (or
  ```
  x :- y, z.
  ```
  in PROLOG syntax)
- The goal clause is given with a question mark, e.g.,
  ```
  ?  x,y.
  ```
- The interpreter/compiler searches for an SDL resolution refutation with the goal clause as base clause.
- Side clauses are tried according to the ordering they are given in.

Example

- From Monty Python's Holy Grail: A witch is a female who burns. Witches burn - because they're made of wood. Wood floats. What else floats on water? A duck; if something has the same weight as a duck it must float. A duck and scales are fetched. The girl and the duck balance perfectly. "It's a fair cop."
- This reasoning can be represented by the following logic program.

```
witch <= burns and female.
burns <= wooden.
wooden <= floats.
floats <= sameweightDuck.
female.
sameweightDuck.
?  witch.
```

Example cont'd

The interpreter produces the following SLD resolution refutation.

1. $C_1 = \{\overline{\mathtt{witch}}\}$
2. $D_1 = \{\mathtt{witch}, \overline{\mathtt{burns}}, \overline{\mathtt{female}}\}$
3. $C_2 = \{\overline{\mathtt{burns}}, \overline{\mathtt{female}}\}$
4. $D_2 = \{\mathtt{burns}, \overline{\mathtt{wooden}}\}$
5. $C_3 = \{\overline{\mathtt{wooden}}, \overline{\mathtt{female}}\}$
6. $D_3 = \{\mathtt{wooden}, \overline{\mathtt{floats}}\}$
7. $C_4 = \{\overline{\mathtt{floats}}, \overline{\mathtt{female}}\}$
8. $D_4 = \{\mathtt{floats}, \overline{\mathtt{sameweightDuck}}\}$
9. $C_5 = \{\overline{\mathtt{sameweightDuck}}, \overline{\mathtt{female}}\}$
10. $D_5 = \{\mathtt{sameweightDuck}\}$
11. $C_6 = \{\overline{\mathtt{female}}\}$
12. $D_6 = \{\mathtt{female}\}$
13. $C_7 = \square$

Recall our tree resolution of
$F := (x \lor y) \land (\neg y \lor z) \land (\neg x \lor y) \land (\neg z)$.

$$\begin{tikzcd}
(\neg z) \arrow[dr] & (\neg y \lor z) \arrow[d] & & (\neg z) \arrow[d] & (\neg y \lor z) \arrow[dl] \\
(x \lor y) \arrow[dr] & (\neg y) \arrow[d] & & (\neg y) \arrow[d] & (\neg x \lor y) \arrow[dl] \\
& (x) \arrow[dr] & & (\neg x) \arrow[dl] & \\
& & () & &
\end{tikzcd}$$

It may be inverted to provide a boolean search tree for
$F := (x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee y) \wedge (\neg z)$.

$$
\begin{tikzcd}
(z) & (y \wedge \neg z) & & (z) & (y \wedge \neg z) \\
(\neg x \wedge \neg y) & (y) \arrow[ul] \arrow[u] & & (y) \arrow[u] \arrow[ur] & (x \wedge \neg y) \\
 & (\neg x) \arrow[ul] \arrow[u] & & (x) \arrow[u] \arrow[ur] & \\
 & & () \arrow[ul] \arrow[ur] & &
\end{tikzcd}
$$

The *least number principle* **LNP**$_n$ is a tautology asserting that any strict (total) n-order has a minimal element.
Its negation may be given in CNF by the following clauses:

$$\begin{array}{ll} \neg x_{i,i} & i \in [n] \\ x_{i,j} \vee x_{j,i} & i \neq j \in [n] \\ \neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k} & i,j,k \in [n] \\ \bigvee_{i=1}^{n} x_{i,j} & j \in [n] \end{array}$$

LNP$_n$ admits polynomial-sized refutations in resolution. Let $k \in [n]$ and $i \in [k]$ and move between instances of:

$$x_{i,1} \wedge \ldots \wedge x_{i,i-1} \wedge x_{i,i+1} \wedge \ldots \wedge x_{i,k}$$