RESOLUTION STRATEGIES

- Improve efficiency
- Preserve completeness

Clause elimination

- "pure" literal elimination
  \[ P(A) \lor Q(x, A) \lor \neg P(...), \quad P(x) \lor \neg P(x) \lor Q(A, x) \]
  unity
  no such literal

- tautology elimination
  \[ P(A) \lor \neg P(A) \lor P(B), \quad P(x) \lor \neg P(x) \lor Q(A, x) \]
  cannot help in a refutation
  eg. we would have to have clauses with literals resolvable against \( P(A) \lor \neg P(A) \)
  but then these clauses can be resolved against each other.

- subsumption elimination
  \[ P(A) \text{ subsumes } P(x) \lor Q(x, B) \]
  \[ P(x) \lor Q(x) \lor R(g, y) \]
  \[ P(A) \lor Q(A, B) \]
  \[ P(x) \lor \neg P(x) \lor Q(A, x) \]

Proof strategies

- Breadth-first
  complete but inefficient

- Depth-first
  efficient but incomplete

- Unit resolution
  (one or both parents length 1)
  efficient but incomplete

- Unit preference
  (use unit resolution if possible)
  efficient & complete

- Set of support
  denial clauses "guide" proof
  \( \neg P(x) \lor Q(x) \lor R(g, y) \)
  \( P(A) \lor Q(A, B) \)
  \( P(x) \lor \neg P(x) \)
  \( Q(e) \)
  \( R(A; B) \lor \neg Q(A) \)
  non-SOS steps
- Connection-graph method: (grads)

- Ordered resolution
  Resolve initial literals only
  Complete for Horn clauses (prolog)

\[ P(A) \iff \neg Q(x), Q(x) :\neg P(x) \]
\[ P(A) \equiv \text{True}, \quad \text{False} \equiv Q(x), \quad Q(x) \equiv P(x) \]
\[ P(A), \quad \neg Q(x), \quad P(x) \implies Q(x) \]

\[ \text{Quacks}(x) :\neg \text{Happy}(x), \text{Duck}(x) \]
\[ \quad :\neg \text{Quacks}(\text{Daffy}) \]
\[ \quad \text{goal denial} \]
\[ \quad \text{if} \quad \text{Happy}(\text{Daffy}), \text{Duck}(\text{Daffy}) \]

- Model Generation (elimination)

Efficient!

Think of "model" as "all true ground atoms"

Use prolog on horn subset
If refuted \( \rightarrow \) done
If fail \( \rightarrow \) expand horn set using a +ve literal from a non-horn clause whose antecedent (if any) we can prove.
If we refute all possibilities \( \rightarrow \) done
O.w. expand horn set again, ...

E.g.
\[ \neg P, \neg Q, \neg KVS; \neg SVPVQ, RVVPVQ \]

- can't refute
- can prove antecedent which is empty (prove antecedent \( \equiv \) eliminate +ve literals)

Add \( R \)
- can't refute \( \rightarrow \) bigger model

Add \( P \)
- can prove \( S \)
- can prove \( S \) (eliminate \( \neg S \))

Add \( Q \)
- can refute \( \rightarrow \) we're done

If we also have \( \text{Happy}(\text{Daffy}) \iff \& \quad \text{Duck}(\text{Daffy}) \iff \)
we'll have a refutation & thus a proof of Quacks (Daffy);

The latex notes contain James Allen's less trivial example,
Where we show that some bird quacks: \( \iff \text{Bird}(x), \text{Quacks}(x) \)