4.

Resolution

Goal

Deductive reasoning in language as close as possible to full FOL

$\neg, \ \wedge, \ \vee, \ \exists, \forall$

Knowledge Level:

given KB, α, determine if KB $\models \alpha$.

or given an open $\alpha[x_1,x_2,...x_n]$, find $t_1,t_2,...t_n$ such that KB $\models \alpha[t_1,t_2,...t_n]$

When KB is finite $\{\alpha_1, \alpha_2, ..., \alpha_k\}$

KB $\models \alpha$

iff $\models [(\alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_k) \supset \alpha]$

iff KB $\cup \{\neg\alpha\}$ is unsatisfiable

iff KB $\cup \{\neg\alpha\}$ $\models$ FALSE

where FALSE is something like $\exists x.(x \neq x)$

So want a procedure to test for validity, or satisfiability, or for entailing FALSE.

Will now consider such a procedure (first without quantifiers)

Clausal representation

Formula = set of clauses

Clause = set of literals

Literal = atomic sentence or its negation

positive literal and negative literal

Notation:

If ρ is a literal, then $\bar{\rho}$ is its complement

$\overline{p} \Rightarrow \neg p \qquad \overline{\neg p} \Rightarrow p$

To distinguish clauses from formulas:

[and] for clauses: $[p, \overline{r}, s]$ { and } for formulas: $\{ [p, \overline{r}, s], [p, r, s], [\overline{p}] \}$

[] is the empty clause {} is the empty formula

So {} is different from {[]}!

Interpretation:

Formula understood as <u>conjunction</u> of clauses

Clause understood as <u>disjunction</u> of literals

Literals understood normally

$\{[p,\neg q], [r], [s]\}$	[ ]
represents	represents
$((p \vee \neg q) \wedge r \wedge s)$	FALSE

CNF and DNF

Every propositional wff α can be converted into a formula α' in Conjunctive Normal Form (CNF) in such a way that $\models \alpha \equiv \alpha'$.

1. eliminate $\supset$ and $\equiv$ using $(\alpha \supset \beta) \Rrightarrow (\neg\alpha \vee \beta)$ etc.
2. push $\neg$ inward using $\neg(\alpha \wedge \beta) \Rrightarrow (\neg\alpha \vee \neg\beta)$ etc.
3. distribute $\vee$ over $\wedge$ using $((\alpha \wedge \beta) \vee \gamma) \Rrightarrow ((\alpha \vee \gamma) \wedge (\beta \vee \gamma))$
4. collect terms using $(\alpha \vee \alpha) \Rrightarrow \alpha$ etc.

Result is a conjunction of disjunction of literals.

an analogous procedure produces DNF,
a disjunction of conjunction of literals

We can identify CNF wffs with clausal formulas

$$(p \vee \neg q \vee r) \wedge (s \vee \neg r) \Rrightarrow \{\, [p, \neg q, r], [s, \neg r] \,\}$$

So: given a finite KB, to find out if KB $\models \alpha$, it will be sufficient to

1. put $(\mathrm{KB} \wedge \neg\alpha)$ into CNF, as above
2. determine the satisfiability of the clauses

Resolution rule of inference

Given two clauses, infer a new clause:

From clause $\{ p \} \cup C_1$,

and $\{\neg p \} \cup C_2$,

infer clause $C_1 \cup C_2$.

$C_1 \cup C_2$ is called a <u>resolvent</u> of input clauses with respect to p.

Example:

clauses $[w, r, q]$ and $[w, s, \neg r]$ have $[w, q, s]$ as resolvent wrt r.

Special Case:

$[p]$ and $[\neg p]$ resolve to [] (the C_1 and C_2 are empty)

A <u>derivation</u> of a clause c from a set *S* of clauses is a sequence $c_1, c_2, \ldots, c_n$ of clauses, where $c_n = c$, and for each c_i, either

1. $c_i \in S$, or
2. c_i is a resolvent of two earlier clauses in the derivation

Write: $S \rightarrow c$ if there is a derivation

Rationale

Resolution is a symbol-level rule of inference, but has a connection to knowledge-level logical interpretations

Claim: Resolvent is <u>entailed</u> by input clauses.

Suppose $\mathfrak{I} \models (p \vee \alpha)$ and $\mathfrak{I} \models (\neg p \vee \beta)$

Case 1: $\mathfrak{I} \models p$

then $\mathfrak{I} \models \beta$, so $\mathfrak{I} \models (\alpha \vee \beta)$.

Case 2: $\mathfrak{I} \not\models p$

then $\mathfrak{I} \models \alpha$, so $\mathfrak{I} \models (\alpha \vee \beta)$.

Either way, $\mathfrak{I} \models (\alpha \vee \beta)$.

So: $\{(p \vee \alpha), (\neg p \vee \beta)\} \models (\alpha \vee \beta)$.

Special case:

$[p]$ and $[\neg p]$ resolve to [ ],

so $\{[p], [\neg p]\} \models$ FALSE

that is: $\{[p], [\neg p]\}$ is unsatisfiable

Derivations and entailment

Can extend the previous argument to derivations:

If $S \rightarrow c$ then $S \models c$

Proof: by induction on the length of the derivation.
Show (by looking at the two cases) that $S \models c_i$.

But the converse does not hold in general

Can have $S \models c$ without having $S \rightarrow c$.

Example: $\{[\neg p]\} \models [\neg p, \neg q]$ i.e. $\neg p \models (\neg p \vee \neg q)$
but no derivation

However.... Resolution *is* <u>refutation complete</u>!

Theorem: $S \rightarrow []$ iff $S \models []$

Result will carry over to quantified clauses (later)

sound and complete
when restricted to [ ]

So for any set *S* of clauses: *S* is unsatisfiable iff $S \rightarrow []$.

Provides method for determining satisfiability: search all derivations for [].

So provides a method for determining all entailments

A procedure for entailment

To determine if KB $|=$ α,

If KB = { }, then we are testing the validity of α

- put KB, $\neg\alpha$ into CNF to get *S*, as before
- check if *S* $\rightarrow$ [].

Non-deterministic procedure

1. Check if [] is in *S*.

   If yes, then return **UNSATISFIABLE**

2. Check if there are two clauses in *S* such that they resolve to produce a clause that is not already in *S.*

   If no, then return **SATISFIABLE**

3. Add the new clause to *S* and go to 1.

Note: need only convert KB to CNF once

- can handle multiple queries with same KB
- after addition of new fact α, can simply add new clauses α' to KB

So: good idea to keep KB in CNF

Example 1

KB

FirstGrade

FirstGrade ⊃ Child

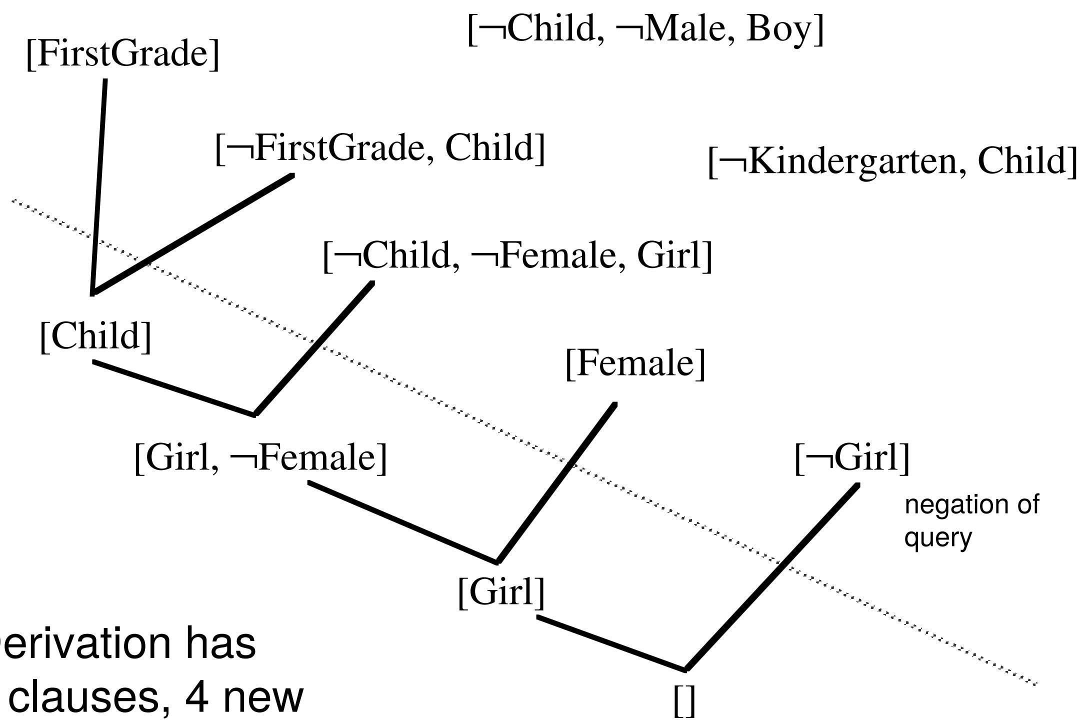
Child ∧ Male ⊃ Boy

Kindergarten ⊃ Child

Child ∧ Female ⊃ Girl

Female

Show that KB |= Girl



Derivation has
9 clauses, 4 new

Example 2

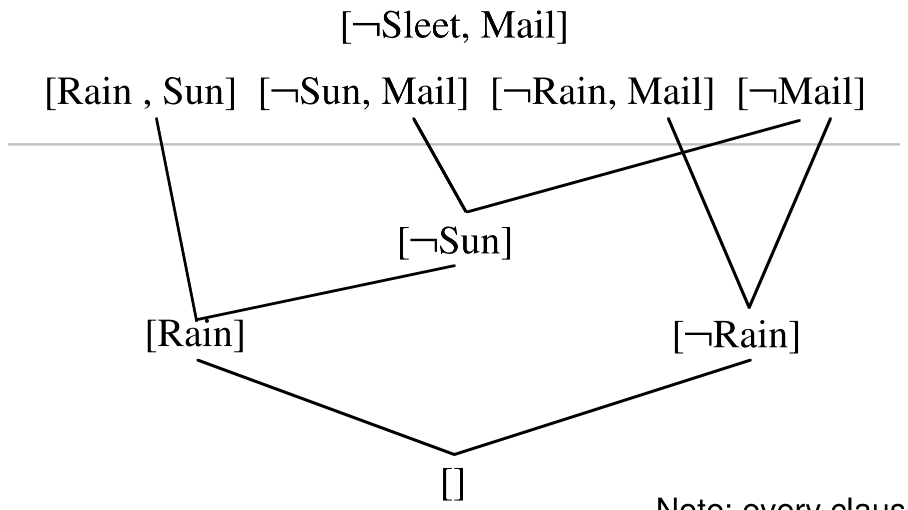
KB

(Rain ∨ Sun)

(Sun ⊃ Mail)

((Rain ∨ Sleet) ⊃ Mail)

Show KB |= Mail



Note: every clause
not in *S* has 2 parents

Similarly KB |≠ Rain

Can enumerate all resolvents given ¬Rain,
and [] will not be generated