Implementing an Inference Procedure

We’ve discussed rules of inference for propositional logic.

It would be useful from a computational point of view if we had an inference procedure that carried out more simply – say, in a single operation – the variety of processes involved in reasoning with the inference rules given.

Fortunately, there is such a procedure: **Resolution**

---

**Resolution**

Recall the following rules of inference:

**Unit resolution**

\[
\alpha \lor \beta, \neg \beta \\
\alpha
\]

**Resolution**

\[
\alpha \lor \beta, \neg \beta \lor \gamma \\
\alpha \lor \gamma
\]

We will base an inference procedure on the application of these rules of inference – ignoring the other rules.

Before we can do so, however, we have to be certain that the sentences (axioms) in our knowledge base are expressed in a form to which these rules can be applied: **clause form**.

---

**Clause Form**

A sentence in clause form is one

- Without \( \land \)
- Without \( \Rightarrow \)
- In which negation applies to single terms only

For example,

\[
(a \lor b) \\
(c \lor d \lor e) \\
(f \lor \neg g)
\]

---

**Converting to Clause Form**
In order to convert a sentence in propositional logic to clause form, one can follow these steps:

- Convert a \( \Rightarrow b \) to \((\neg a \lor b)\)

- Apply deMorgan’s Laws so that any \(\neg\) refers only to a single term:
  \[
  \neg (a \land b) = (\neg a \lor \neg b) \\
  \neg (a \lor b) = (\neg a \land \neg b) \\
  \neg \neg a = a
  \]

- Apply distributive law to convert to conjunctive normal form (i.e., a conjunction of disjunctions)
  \[(a \land b) \lor c = (a \lor c) \land (b \lor c)\]

- Make a separate clause for each conjunct.
  \[(a \lor c)\]
  \[(b \lor c)\]

---

**Example. Converting the Work/Sleep Knowledge Base (KB)**

\[
\begin{align*}
\text{Sun} \lor \text{Mon} \lor \text{Tues} \lor \text{Wed} \lor \text{Thurs} & \Rightarrow \text{Work} \\
\neg (\text{Sun} \lor \text{Mon} \lor \text{Tues} \lor \text{Wed} \lor \text{Thurs}) & \lor \text{Work} \\
(\neg \text{Sun} \land \neg \text{Mon} \land \neg \text{Tues} \land \neg \text{Wed} \land \neg \text{Thurs}) & \lor \text{Work} \\
(\neg \text{Sun} \lor \text{Work}) & \land (\neg \text{Mon} \lor \text{Work}) \land (\neg \text{Tues} \lor \text{Work}) \land (\neg \text{Wed} \lor \text{Work}) \land (\neg \text{Thurs} \lor \text{Work}) \\
(\neg \text{Sun} \lor \text{Work}) \\
(\neg \text{Mon} \lor \text{Work}) \\
\text{etc.}
\end{align*}
\]

\[
\begin{align*}
\text{Party} \land \text{Work} & \Rightarrow \neg \text{Sleep} \\
\neg (\text{Party} \land \text{Work}) & \lor \neg \text{Sleep} \\
(\neg \text{Party} \lor \neg \text{Work}) & \lor \neg \text{Sleep} \\
\neg \text{Party} & \lor \neg \text{Work} \lor \neg \text{Sleep}
\end{align*}
\]

---

**Resolution: Proof by Refutation (Contradiction)**

To prove that a sentence S is true, we will assume the opposite, and show that that leads to a contradiction with the knowledge base.

High-level view of the algorithm:

1. Negate S and convert the result to clause form. Add it to the KB.
2. Repeat until either a contradiction is found or no progress can be made:
- Select two clauses. Call these the parent clauses.
- Resolve the parent clauses. Call the resulting clause the resolvent.
- If the resolvent is empty, then a contradiction has been found. If it is not, then add it to the KB.

Example. Applying resolution to the Work/Sleep problem.

Our set of axioms (i.e., our knowledge base) is:

(~Sun v Work)
(~Mon v Work)
(~Tues v Work)
(~Wed v Work)
(~Thurs v Work)
(~Thurs v Party)
(~Fri v Party)
(~Sat v Party)
¬Party v ¬Work v ¬Sleep
Thurs

To prove ¬Sleep, we add Sleep to the KB.

Thurs, (~Thurs v Work)
Work
add Work to KB

Thurs, (~Thurs v Party)
Party
add Party to KB

Work, ¬Party v ¬Work v ¬Sleep
¬Party v ¬Sleep
add ¬Party v ¬Sleep to KB

Party, ¬Party v ¬Sleep
¬Sleep
add ¬Sleep to KB

¬Sleep, Sleep CONTRADICTION

Completeness of Resolution Proof by Contradiction

The algorithm given above is complete.

On the other hand, if we applied resolution in a “forward” direction (i.e., if we did not do a proof by contradiction), it would often work – but would not be complete!
Consider beginning with an empty KB. Say you want to prove $P \lor \neg P$
You can do this with a resolution proof by contradiction. But you cannot do it in a
“forward” manner because there is nothing with which to resolve anything.

Is there anything faster?

Yes – if we restrict the expressiveness of our language.