# Admin

- Tutorials from week 3





# Knowledge Representation and Engineering

- Predicate calculus as a representation language
- Syntax: a language with a grammar
- Semantics: assigning meaning
- Deduction: searching for proof





# Logic as a representation language

A logic plays two roles:

- *Representation* (semantics):
  describes the state of the world
- *Inference* (deduction):
  computable operations that are defines on the representations.

Today we consider the second. We need a *language* to describe the world.





# Grammar for first-order logic

Define *terms* by

$$
\begin{aligned}
term \quad ::= \quad & constant \\
& \mid var \\
& \mid fn\_symbol \textbf{ ( } term\_list \textbf{ )} \\
term\_list \quad ::= \quad & term \\
& \mid term \textbf{ , } term\_list
\end{aligned}
$$

# Formulas (= making a statement)

$$\begin{aligned} form \;\; ::= \;\; & pred \; \textbf{(} \; term\_list \; \textbf{)} \\ & \mid \neg \; form \\ & \mid form \; \vee \; form \\ & \mid form \; \wedge \; form \\ & \mid form \; \rightarrow \; form \\ & \mid \forall \; var \; form \\ & \mid \exists \; var \; form \end{aligned}$$

Use precedence to disambiguate (or brackets).





# Semantics

We say what it is for a formula to be *true* under an interpretation in a structure.

Write $\mathcal{S}$ for a structure together with an associated interpretation $\mathtt{I}$.

Given $\mathcal{S}$, and a formula $\boldsymbol{F}$, write $\mathcal{S} \models \mathcal{F}$ for "$\boldsymbol{F}$ is true in $\mathcal{S}$".

For details, see Russell and Norvig, chapter 8, section 2.





# Quantifiers

Roughly, the idea is that for any statement $\Phi(v)$ which talks about variable $v$:

$\mathcal{S} \models \forall v_n \; (\boldsymbol{\Phi}(v_n))$ if and only if $\mathcal{S} \models \boldsymbol{\Phi}(v_n)$
for all interpretations of $v_n$

$\mathcal{S} \models \exists v_n \; (\boldsymbol{\Phi}(v_n))$ if and only if $\mathcal{S} \models \boldsymbol{\Phi}(v_n)$
for some interpretation of $v_n$





# Logical Consequence

Our semantics gives us a notion of *logical consequence*.

We say that a formula $\boldsymbol{G}$ is a logical consequence of formulae $\boldsymbol{F_1, F_2 \ldots F_n}$ (meaning that it follows logically) if and only if, for all structures with interpretation $\mathcal{S}$,

*if* $\mathcal{S} \models \boldsymbol{F_1}$ *and ... and* $\mathcal{S} \models \boldsymbol{F_n}$,
*then* $\mathcal{S} \models \boldsymbol{G}$.

When this is true, we write

$$\boldsymbol{F_1, F_2 \ldots F_n} \models \boldsymbol{G}.$$

# So what?

If we have some statements we believe to be true, we can ask:

does another statement follow from what we believe?

"Logical consequence" gives a precise way of making sense of the question.

It doesn't give us a *computational* answer, though – we need other techniques for that.





# Example: Inference System

For reasoning about statements just involving implications.

**Axioms**

A1: $A \to (B \to A)$
A2: $(A \to (B \to C)) \to$
$((A \to B) \to (A \to C))$

**Inference Rule**

MP: $\dfrac{P \qquad P \to Q}{Q}$

This language is more expressive than it looks
– eg $A \wedge B \wedge C \to D$ is the same as $A \to B \to C \to D$.





# A Derivation

How can this derivation of $p \to p$ be constructed?

| | | |
|---|---|---|
| 1 | $(p \to (p \to p) \to p) \to$ $(p \to (p \to p)) \to (p \to p)$ | A2 |
| 2 | $p \to (p \to p) \to p$ | A1 |
| 3 | $(p \to (p \to p)) \to (p \to p)$ | MP 1,2 |
| 4 | $p \to (p \to p)$ | A1 |
| 5 | $p \to p$ | MP 3,4 |

This is hard to find (in either direction).





# Search Space

We can make use of the inference rule in two directions.

**Bottom-up**:
to prove a goal $G$, start from the axioms and apply the inference rules until $G$ is found.

**Top-down**:
to prove $G$, apply the inference rules backwards until a set of axioms is found.

# Parameters to rules

Rules can often be applied in several ways; to specify exactly which way is meant, sometimes a parameter is to be supplied.

For example, when using the modus ponens rule backwards, we need the formula $P$ as a parameter:

$$\text{MP: } \frac{P \qquad P \to Q}{Q}$$

Notice that this gives an infinite branching point in the search space – we could use *any* formula at all.





# Computational properties

This inference system is *logically adequate* – the "right" formulas are provable.

But it is *computationally badly behaved* – search is unconstrained.

We can find better inference systems that have the *same* derivable formulas.





# Sequent Calculus

Instead of using formulas, use a pair of formula lists, linked by the sequent symbol:

$$F_1, F_2, \ldots, F_n \Longrightarrow G_1, \ldots, G_m$$

We usually use a *set* of formulas rather than a list here.

The sequent should be interpreted as saying:
if all the lhs formulas are true, then one of the rhs formulas is true.





We want an inference system that is equivalent to the old one (using A1, A2, MP).

So we want axioms and inference rules for the system $NEW$ such that

$$\vdash_{\mathit{OLD}} F \quad \text{iff} \quad \vdash_{\mathit{NEW}} [\,] \Longrightarrow F$$

We have introduced extra syntax, so you might expect any inference system to be more complex; not so!

## Sequent Rules

**Axiom**: $\ldots F \ldots \Longrightarrow \ldots F \ldots$

**Inference Rules**

impI $$\frac{F, F_1, \ldots, F_n \Longrightarrow G, H_1, \ldots, H_m}{F_1, \ldots, F_n \Longrightarrow F \rightarrow G, H_1, \ldots, H_m}$$

impE $$\frac{F \rightarrow G, F_1, \ldots, F_n \Longrightarrow F, H_1, \ldots, H_m \qquad G, F_1, \ldots, F_n \Longrightarrow H_1, \ldots, H_m}{F \rightarrow G, F_1, \ldots, F_n \Longrightarrow H_1, \ldots, H_m}$$

Now the proof of $p \rightarrow p$ is very easy.





## Top-down proof search

Suppose we have use the rules in the hand-out for the third week tutorial.

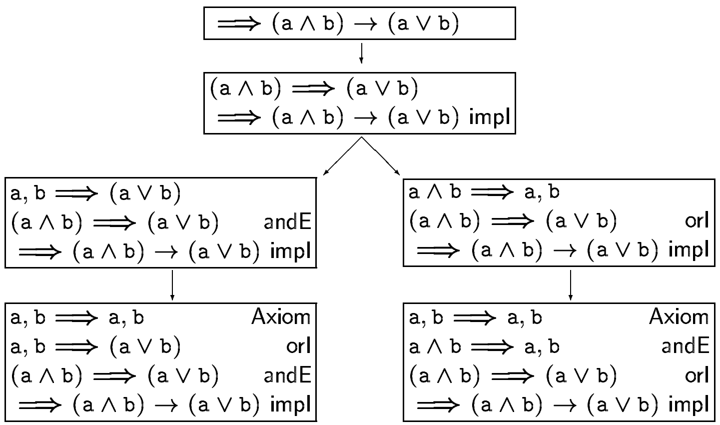
Consider the goal

$$[\,] \Longrightarrow (a \wedge b) \rightarrow (a \vee b)$$

Applying the rules backwards gives a goal tree that describes the search space – branching occurs where there is more than one way of extending a derivation backwards from some unjustified line in the derivation.












## Looping Search

Notice that the branching in the search tree is *finite* here.

However, search may loop.

For example, if we apply $impE$ backwards to $p \rightarrow q \Longrightarrow p$, we get the original goal repeated:

$$\frac{p \rightarrow q \Longrightarrow p \qquad p \rightarrow q, q \Longrightarrow p}{p \rightarrow q \Longrightarrow p}$$

# Dealing with looping

There are two possible solutions, if we want to find a complete inference procedure:

1. Change the inference system — find one that generates no loops.
2. Change the inference strategy — don't use depth first search; or build in some check for looping.





Here is a possible alternative rule.

Instead of

$$\text{impE}\frac{F \to G, F_1, \ldots, F_n \Longrightarrow F, H_1, \ldots, H_n \qquad G, F_1, \ldots, F_n \Longrightarrow H_1, \ldots, H_n}{F \to G, F_1, \ldots, F_n \Longrightarrow H_1, \ldots, H_n}$$

use

$$\text{impE2}\frac{F_1, \ldots, F_n \Longrightarrow F, H_1, \ldots, H_n \qquad G, F_1, \ldots, F_n \Longrightarrow H_1, \ldots, H_n}{F \to G, F_1, \ldots, F_n \Longrightarrow H_1, \ldots, H_n}$$

This will stop the problem of looping. But is can we still derive the same formulas as before? Yes!! (This needs some thought, though.)





# Alternative

Decide in the search space on some condition when the rule that gives looping should not be applied.

For example:

if looking at $F \to G, \ldots \Longrightarrow C$, only use impE if $C \neq F$.

Do we know that we do not lose any provable statements this way — we *do* lose some derivations, but a complete inference procedure is not required to find all the proofs, just to find *some* proof.





# Infinite choice points

A full set of rules for sequent calculus has rules for the quantifiers.

A rule for $\exists$ is:

$$\frac{\ldots \Longrightarrow F(t)}{\ldots \Longrightarrow \exists x\ F(x)}$$

where $t$ can be any term. So here the branching is infinite.

Here resolution gives us a hint — the choice of candidate terms that are worth investigating comes from unification with terms that are already in the formula.

# Special case

Suppose that there are only finitely many constants, and no function symbols.

The we need only look at finitely many possible terms $t$, so the branching is finite.

(Why is this? — given that there are still infinitely many variables.)





# Summary

- Predicate Calculus as a KR language
- Sequent Calculus for top down search
- Problem of looping, and infinite branching
- A Special case.