Knowledge Representation and Reasoning: First order logic refresher

Chapter 7.3, 7.4, Chapter 8

Logical reasoning agent

Agent formulates a theory about its environment or about a problem it needs to solve – maybe involving other agents, maybe not.

Uses abstract (logical) representation of its theory to reason:

- ♢ deducing consequences
- ♢ exploring possibilities

Arrives at a knowledge base, which could be used for anything (prediction, communication, action, . . .)

Outline

◊ The idea of logic   // expressing!

◊ Propositional logic: connectives

◊ First order logic: quantifiers

◊ Reasoning about systems

Logic as a basis for KR

Formal declarative language for knowledge representation

Features

◇ **Clear syntax**
— well-defined recursive structure
— automation possible

◇ **Clean semantics**
— correctness (and incorrectness) are definable
— accuracy: ambiguities can be exposed and explained

◇ **General**: works for all domains
— pure logic is subject-neutral "leave the form."
— definitions depend on form, not content

◇ **Extensible**: features of target domains
— can add logic of time (past/future tense, 'while', 'until', 'next', . . .)
— can add agent attitudes (belief, intention, preference, . . .)
— can add theories, e.g. arithmetic

No absolute boundary

Deducing consequences

Given a set Γ of formulae of a formal KR language, and a specific formula A, logic determines whether A is a consequence of Γ.

◇ **Semantic definition**: A is true in every possible situation satisfying everything in Γ [Truth]
— depends on rigorous specification of meaning (truth and possibility)

◇ **Syntactic definition**: there is a derivation of A from Γ [Proof & Derivation]
— depends on rigorous specification of inference rules

◇ On either definition, some basic properties hold:
— if A is in Γ, A is a consequence of Γ;
— if $\Gamma \subseteq \Delta$ then every consequence of Γ is a consequence of Δ;
— if Γ is a set of consequences of Δ then every consequence of Γ is a consequence of Δ.

Necessary consequences, possible scenarios

◇ Consequence is a matter of **necessity**
— if this holds, that must hold as well
— having this without that is impossible

◇ Logic also defines non-consequence, and hence **possibility**
— this could hold, and that could also hold with it
— having this and that together is possible

Reasoning tasks

Some problem-solving tasks call for **proof** of logical consequence

◇ Verification that some program/plan/etc is correct

◇ Demonstration that no "bad" state is reachable

Other tasks call for **models** (examples) showing logical possibility

◇ Show how it might look if some conditions were met

◇ Produce schedules, layouts, designs, etc meeting specifications

◇ Demonstration that some "good" state is reachable

Propositional logic

The most abstract level of logical language and reasoning

◇ **Atomic** sentences p, q, r, etc
— Don't look inside them: treat them as atoms [base units]
— Logical operations (connectives) apply from outside

◇ **Connectives**
— Apply to sentences (formulae) to make longer ones
— Some unary – e.g. 'soon', 'maybe', 'Trump believes'
— Some binary – e.g. 'until', 'because', 'unless'
— etc.

◇ **Truth-functional** connectives
— Truth value (true or false) of compound determined by values of parts
— E.g. 'and', 'not'

" Truth value of sentense [O/P]
= f(truth value of "components")"
[I/P]

Propositional logic: the basic connectives

◇ Negation: $\neg A$ true iff A false (and false iff A true)

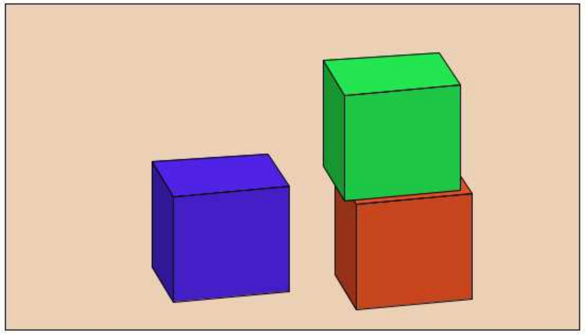
◇ Conjunction: $A \wedge B$ true iff A true and B true

◇ Disjunction: $A \vee B$ true iff A true or B true (or both)

◇ Implication: $A \rightarrow B$ true iff A false or B true

◇ Equivalence: $A \leftrightarrow B$ true iff A and B have the same truth value

expressible in terms of other connectives



greenOnRed $\wedge$ redOnTable

$\neg$(blueOnGreen $\vee$ greenOnBlue)

(redOnBlue $\wedge$ blueOnTable) $\rightarrow$ redOnGreen

Propositional logic: truth tables

connective

right

left

$\neg$	
0	1
1	0

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\rightarrow$	0	1
0	1	1
1	0	1

$\leftrightarrow$	0	1
0	1	0
1	0	1

- ◊ Truth value of any propositional formula can be computed given an assignment of the values 1 (true) and 0 (false) to the atoms
- ◊ This computation is entirely deterministic and easy (linear time)
- ◊ Gives mechanical test for validity of inferences
- ◊ However, for n atoms there are 2^n value assignments...

Not good!

Propositional logic: splitting the atom

Usually, what we want to describe has some structure. For example, things have names, we reason about relationships between them, etc.

E.g. in the blocks example
— name the three blocks R, G and B, and call the table T
— write 'on(_,_)' to say which things are on which
— so $\mathsf{on}(R,T) \leftrightarrow \neg(\mathsf{on}(R,G) \vee \mathsf{on}(R,B))$ etc.

◇ **Term** is a name or the result of applying a function symbol to terms
— picks out an individual or object

◇ **Predicate** applies to a given number of terms to form a sentence
— represents a relation (set of n-tuples)
— sentence $P(t_1, \ldots, t_n)$ true if the objects $o_1, \ldots, o_n$ picked out by those terms are in the relation represented by P

◇ **Logic** of these ground atoms is still just propositional

Expressing generality: quantifiers and variables

We often need to generalise about objects. Eg:

— any block x is "clear" iff there is no block on it
— no block is (ever) on itself
— if one block is on another, there is a block on a block on the table

◇ Need to express 'all' ($\forall$) and 'some' ($\exists$)

◇ Require using variables x, y, etc in place of names

◇ $\forall x A(x)$ means A is true of every thing x

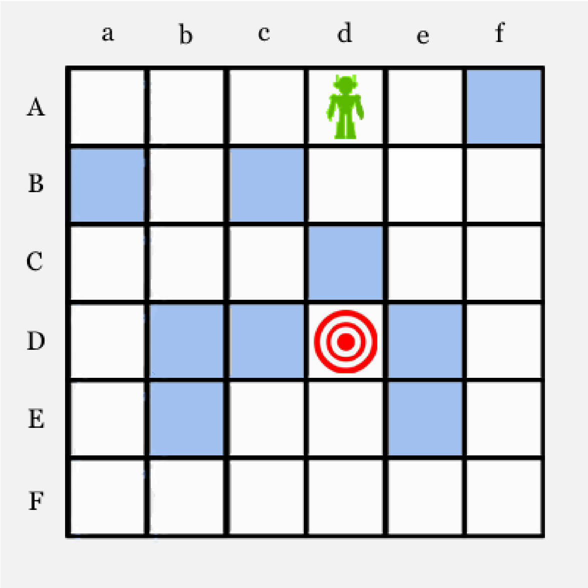
◇ $\exists x A(x)$ means A is true of at least one thing x

"someone" ← We know or do not know

So, for instance:

$$\forall x(clear(x) \leftrightarrow \neg\exists y\ \mathsf{on}(y,x))$$
$$\forall x \neg\mathsf{on}(x,x)$$
$$\exists x\exists y\,(y \neq T \wedge \mathsf{on}(x,y)) \rightarrow \exists x\exists y\,(\mathsf{on}(x,y) \wedge \mathsf{on}(y,T))$$

Example: grid world



Rows: $A, \ldots, F$

Columns: $a, \ldots, f$

Actions: North, South, East, West

States: $s_1, \ldots, s_{12}$

Functions: row(_), col(_), act(_)

Predicate: blocked(_)

"relations"

$\text{row}(s_1) = A \wedge \text{col}(s_1) = d \wedge \underline{\text{row}(s_{12})} = D \wedge \text{col}(s_{12}) = d$

$\text{blocked}(B, a) \wedge \neg\,\text{blocked}(B, b) \wedge \ \ldots$ suppose we know this

$\forall t(\text{act}(t) = \text{North} \rightarrow \text{row}(t) \neq A)$

etc.

State transition problems

◊ Very common to reason about **transitions** between **states** of a system

◊ Logic useful for representing knowledge about states and goals
— Relationships between objects in a single (static) state
— Sometimes restricted to atomic formulae, but does not have to be

◊ Can also represent knowledge about transitions
— Each transition has preconditions (describe when it can happen)
— Each transition has postconditions (describe what it changes)
— Each transition has frame conditions (describe what does not change)

◊ **Ramification problem**: calculate (relevant) consequences of changes
— Logic-based reasoning deals with this in a natural way

◊ **Frame problem**: represent and calculate all frame conditions
— Serious issue, especially where state representations are non-atomic