Knowledge Representation and Reasoning: Logical Foundations – Computability and Logic

Yongmei Liu
ymliu@mail.sysu.edu.cn

Dept. of Computer Science
Sun Yat-sen University

Fall 2019

Outline

- KRR and logic
- Course organization
- Chap 1-2: Distinguish between two kinds of infinite sets, the enumerable and the nonenumerable (可枚举集和不可枚举集)

What is KRR?

Symbolic encoding of propositions believed by some agent and their manipulation to produce representations of propositions that are believed by the agent but not explicitly represented

An example

- Explicitly represented beliefs:
  $GradStu(Ann)$, $GradStu(Bob)$,
  $\forall x(GradStu(x) \to Student(x))$
- Implicitly represented beliefs:
  $Student(Ann)$, $Student(Bob)$,
  $\forall x(\neg Student(x) \to \neg GradStu(x))$

We need knowledge to answer questions

Could a crocodile run a steeplechase? [Levesque 88]

- Yes
- No

The intended thinking: short legs, tall hedges $\Rightarrow$ No!

Yet another example

Consider a question about materials:

> The large ball crashed right through the table because it was made of XYZZY. What was made of XYZZY?
> - the large ball
> - the table

Now suppose that you learn some facts about XYZZY.

1. It is a trademarked product of the Dow Chemical Company.
2. It is usually white, but there are green and blue varieties.
3. It is ninety-eight percent air, making it lightweight and buoyant.
4. It was first discovered by a Swedish inventor, Carl Georg Munters.

Ask: At what point does the answer stop being just a guess?

Why KRR?

- KR hypothesis: any artificial intelligent system is knowledge-based
  - Much of AI involves building systems that are knowledge-based
  - Some, to a certain extent, *e.g.*, game-playing, vision, etc.
  - Some, to a much lesser extent, *e.g.*, speech, motor control, etc.
- Knowledge-based system: system with structures that
  - can be interpreted propositionally and
  - determine the system behavior

  such structures are called its knowledge base (KB)

Two examples

Example 1

```
printColour(snow) :- !, write("It's white.").
printColour(grass) :- !, write("It's green.").
printColour(sky) :- !, write("It's yellow.").
printColour(X) :- write("Beats me.").
```

Example 2

```
printColour(X) :- colour(X,Y), !,
        write("It's "), write(Y), write(".").
printColour(X) :- write("Beats me.").

colour(snow,white).
colour(sky,yellow).
colour(X,Y) :- madeof(X,Z), colour(Z,Y).
madeof(grass,vegetation).
colour(vegetation,green).
```

Why bother?

- Why not "compile out" knowledge into specialized procedures?
    - distribute KB to procedures that need it (as in Example 1)
    - almost always achieves better performance
- No need to think. Just do it!
    - riding a bike
    - driving a car

Advantages

Knowledge-based system most suitable for *open-ended* tasks

can structurally isolate *reasons* for particular behaviour

Good for

- explanation and justification
  - "Because grass is a form of vegetation."
- informability: debugging the KB
  - "No the sky is not yellow. It's blue."
- extensibility: new relations
  - "Canaries are yellow."
- extensibility: new applications
  - returning a list of all the white things
  - painting pictures

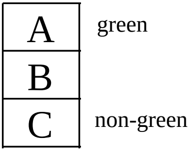
KRR and logic

Logic is the main tool for KRR, because logic studies

- How to formally represent agent's beliefs
- Given the explicitly represented beliefs, what are the implicitly represented beliefs

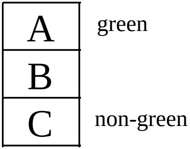
There are many kinds of logics. In this course, we will study first-order logic (FOL).

A blocks world example



- Given the scene, human can easily draw the conclusion "there is a green block directly on top of a non-green block"
- How can a machine do the same?

Formalization in FOL



- $S = \{On(a,b), On(b,c), Green(a), \neg Green(c)\}$
- $\alpha = \exists x \exists y [Green(x) \wedge \neg Green(y) \wedge On(x,y)]$
- S logically entails α

An example

- Tony, Mike, and John belong to the Alpine Club.
- Every member of the Alpine Club who is not a skier is a mountain climber.
- Mountain climbers do not like rain, and anyone who does not like snow is not a skier.
- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.
- Tony likes rain and snow.
- Is there a member of the Alpine Club who is a mountain climber but not a skier?

An example (cont'd)

- Intelligence is needed to answer the question
- Can we make machines answer the question?
- A possible approach
  - First, translate the sentences and question into FOL formulas
    - Of course, this is hard, and we do not have a good way to automate this step
  - Second, check if the formula of the question is logically entailed by the formulas of the sentences
    - There are ways to automate this step

Alphabet

- Individuals (constants or 0-ary functions):
  - tony, mike, john
  - rain, snow
- Types (unary predicates):
  - $A(x)$ means that x belongs to Alpine Club
  - $S(x)$ means that x is a skier
  - $C(x)$ means that x is a mountain climber
- Relationships (binary predicates):
  - $L(x, y)$ means that x likes y

Basic facts

- Tony, Mike, and John belong to the Alpine Club.
  $A(tony), A(mike), A(john)$
- Tony likes rain and snow.
  $L(tony, rain), L(tony, snow)$

Complex facts

- Every member of the Alpine Club who is not a skier is a mountain climber.
  $\forall x(A(x) \land \neg S(x) \to C(x))$
- Mountain climbers do not like rain, and anyone who does not like snow is not a skier.
  $\forall x(C(x) \to \neg L(x, rain))$
  $\forall x(\neg L(x, snow) \to \neg S(x))$
- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.
  $\forall x(L(tony, x) \to \neg L(mike, x))$
  $\forall x(\neg L(tony, x) \to L(mike, x))$
- Is there a member of the Alpine Club who is a mountain climber but not a skier?
  $\exists x(A(x) \land C(x) \land \neg S(x))$

Course content: Part 1

An in-depth study of predicate calculus, including

- proof system for predicate calculus
- why the proof system is sound and complete
  - *i.e.*, a conclusion follows from a set of hypotheses iff there is a proof of the conclusion starting from the hypotheses
  - The completeness theorem for predicate calculus, proved by Gödel in 1930, ranks as one of the great results in logic in 20th century

Course content: Part 2

An introduction to computability theory, we will study

- there is no algorithm which will decide if a program will halt
- there is no algorithm which can decide whether a first-order sentence is satisfiable or not

Textbook and course evaluation

- Classic Textbook: G. S. Boolos, J. P. Burgess and R. C. Jeffrey, Computability and Logic, Fourth/Fifth Edition, Cambridge University Press, 2002/2007
- We will cover Chap1-14 of the textbook
- 4 assignments (40%) + final exam (60%)