Lecture 13: Knowledge and Inference

Dr John Levine

CS103 Machines, Languages and Computation
November 9th 2015

University of Strathclyde Science

The Second Half of the Class

- Week 7: Recursion and Languages: how to use recursion to create potentially unbounded sentences.
- Week 7½: Recursion and Strings: how to specify an infinite set of strings using a finite set of rules.
- Week 8: Recursion and Logical Definitions: what does it mean to be someone's ancestor?
- Week 9: The λ-calculus: how can we create and use computable functions using only symbols?
- Week 10: Recursion and Function Definitions: how can we use recursion to create functions?

University of Strathclyde Science

The Second Half of the Class

- Week 7: Recursion and Languages: how to use recursion to create potentially unbounded sentences.
- Week 7½: Recursion and Strings: how to specify an infinite set of strings using a finite set of rules.
- Week 8: Recursion and Logical Definitions: what does it mean to be someone's ancestor?
- Week 9: The λ-calculus: how can we create and use computable functions using only symbols?
- Week 10: Recursion and Function Definitions: how can we use recursion to create functions?

University of Strathclyde Science

Grammars and Languages

- We can use regular expressions to describe the set of strings generated by a grammar.
- Such a set of strings is often referred to as a *language*.
- For example:

$$S \rightarrow aS$$
$$S \rightarrow b$$

- Exercise: describe the language (i.e. the set of strings) generated by this grammar as a regular expression.
- Answer: a*b.

University of Strathclyde Science

Assignment 7

- Write a grammar which can generate strings of the form a^+b^+, e.g. ab, aaab, abbbb, abbb, but *not* aaa, bbb ($^+$ means “repeated at least once”).
- Write a grammar which can generate strings of the form $(a^+b^+)^+$, e.g. ab, aabb, aaaabbaabb, abababbb, but *not* aaaa, aabbaa, bbabbaaabb, etc.
- Can you write a grammar to can generate strings of the form a^nb^n, e.g. ab, aabb, aaabbb, aaaabbbb, but *not* abbb, aaab, aaaabb, etc?
- Please hand in workbooks by 3pm on Wednesday.

University of Strathclyde Science

The Second Half of the Class

- Week 7: Recursion and Languages: how to use recursion to create potentially unbounded sentences.
- Week 7½: Recursion and Strings: how to specify an infinite set of strings using a finite set of rules.
- Week 8: Recursion and Logical Definitions: what does it mean to be someone's ancestor?
- Week 9: The λ-calculus: how can we create and use computable functions using only symbols?
- Week 10: Recursion and Function Definitions: how can we use recursion to create functions?

University of Strathclyde Science

Knowledge and Inference

- Using symbolic systems to represent knowledge about the world and make legal inferences:

    1. Tom is a cat.
    2. Jerry is a mouse.
    3. Cats chase mice.
    4. Therefore, Tom chases Jerry.

- Given facts 1, 2 and 3, fact 4 is a logical consequent
- How can we represent facts and rules?
- How can we make legal inferences?

University of Strathclyde Science

Simple Facts

- We need to be able to encode simple facts like “Tom is a cat”.
- Intuitively, this means “The object labelled as “Tom” is a member of the set of cats”.
- We can invent a simple function (a *predicate*) called “cat(x)” which returns true if x is a cat.
- A predicate is a function which returns true or false
- So we can say “cat(tom)” is true, but “cat(jerry)” is false
- Similarly, “mouse(jerry)” is true, “mouse(tom)” is false

University of Strathclyde Science

More Complicated Facts

- Some facts involve more than one object, such as “Spike is father of Tyke”.
- We can invent predicates for this sort of thing, by using more than one argument to the predicate:

  father(spike,tyke) is true
  father(tom,jerry) is false

- We call these “two-place predicates” or “relations”

University of Strathclyde Science

Keeping a Database

- We'll keep a note of only the true facts about the world in a database:

  cat(tom)
  cat(butch)
  mouse(jerry)
  dog(spike)
  dog(tyke)
  father(spike,tyke)

- All facts which are not in the database (e.g. cat(jerry)) are assumed to be false.

University of Strathclyde Science

Representing Rules

- Our database of facts are like axioms in MIU
- We can also infer new facts from these “axioms” using *rules of inference*:

  All mice are small.
  mouse(X) → small(X)

  All cats chase mice.
  cat(X), mouse(Y) → chases(X,Y) (, = “and”)

- We keep these rules in a separate part of the database and can derive new facts using them

University of Strathclyde Science

Deriving Facts

- We apply the rules just like in MIU to derive new facts:

  cat(tom)
  cat(butch)
  mouse(jerry)

  mouse(X) → small(X)
  cat(X), mouse(Y) → chases(X,Y)

- Apply the first rule to “mouse(jerry)” and we can infer “small(jerry)”

University of Strathclyde Science

Deriving Facts

- We apply the rules just like in MIU to derive new facts:

  cat(tom)
  cat(butch)
  mouse(jerry)

  mouse(X) → small(X)
  cat(X), mouse(Y) → chases(X,Y)

- Apply the second rule with X=tom and Y=jerry and we can derive “chases(tom,jerry)”

University of Strathclyde Science

Deriving Facts

- To derive a new fact, we apply a rule to some existing facts, setting the variables in the rule consistently
- We write a derivation like this:

  rule to be applied
  existing facts to be used
  variable values
  → *new fact*

- We are then allowed to use the new fact in further derivations.

University of Strathclyde Science

Example Derivation

- Derive chases(tom,jerry)

  cat(X), mouse(Y) → chases(X,Y)
  cat(tom), mouse(jerry)
  [X = tom, Y = jerry]
  → chases(tom,jerry) ■

University of Strathclyde Science

Knowledge Engineering

- The process of turning known facts about the world into facts and rules is called *knowledge engineering*
- A simple approach:
  1. Identify all the objects in the world – Tom, Jerry, Spike, Tyke. These are the things that appear inside the brackets. Give them all names.
  2. Identify the sets these can belong in – the set of all cats, the set of all mice, the set of all small things – these are the predicates, e.g. cat(tom).
  3. Identify relationships between objects – these are the relations, e.g. hates(spike,tom).

University of Strathclyde Science

Translating Rules into Logic

- Rules written in English can be hard to transform into logical rules:

  Every dog likes Spike.
  $dog(X) \rightarrow like(X,spike)$

  Dogs chase cats.
  $dog(X), cat(Y) \rightarrow chase(X,Y)$

  A cat is an animal.
  $cat(X) \rightarrow animal(X)$

  Fat people are jolly.
  $fat(X), person(X) \rightarrow jolly(X)$

University of Strathclyde Science

Exercise

- Translate these sentences into facts and rules, using the knowledge engineering approach suggested:

  Ford is an alien.
  Arthur is a human.
  Marvin is an android.
  Humans are lifeforms.
  Aliens are lifeforms.
  Androids are cleverer than lifeforms.

- Can you derive the fact that Marvin is cleverer than Arthur?

University of Strathclyde Science

Exercise

- Objects: ford, arthur, marvin.
- Sets: alien, human, android, lifeform.
- Relations: is-cleverer-than(X,Y).

alien(ford)
human(arthur)
android(marvin)

human(X) → lifeform(X)
alien(X) → lifeform(X)
android(X), lifeform (Y) → is-cleverer-than(X,Y)

University of Strathclyde Science

Exercise

- Can you derive the fact that Marvin is cleverer than Arthur?

  human(X) → lifeform(X)
  human(arthur)
  [X = arthur]
  → lifeform(arthur)

  android(X), lifeform(Y) → is-cleverer-than(X,Y)
  android(marvin), lifeform(arthur)
  [X = marvin, Y = arthur]
  → is-cleverer-than(marvin, arthur)

University of Strathclyde Science

Recursive Logical Definitions

- Consider part of the Simpsons' family tree:

```
parent(orville,grampa)      parent(homer,bart)
parent(yuma,grampa)         parent(marge,bart)
parent(grampa,homer)        parent(homer,lisa)
parent(mona,homer)          parent(marge,lisa)
parent(jackie,marge)        parent(homer,maggie)
parent(clancy,marge)        parent(marge,maggie)
```

- Who are Bart's ancestors?
- How can we define the ancestor(X,Y) relation?