Lecture 2
Modelling in Datalog

COMP24412: Symbolic AI

Giles Reger

February 2019

Aim and Learning Outcomes

The aim of this lecture is to:

Introduce you to the main concepts around *modelling* and how these are concretely realised in the *Datalog* language.

Learning Outcomes

By the end of this lecture you will be able to:

1. Describe concepts such as *consistency*, *consequence*, and *database semantics*
2. Recall and recognise the syntax for Datalog
3. Define what a *fact* and *rule* are with respect to Datalog
4. Explain the meaning of *interpretation* in the formal sense

What is a Model?

What is a Model?


What is a Model?

Definition of Model

A simplified description, especially a mathematical one, of a system or process, to assist calculations and predictions.

Let's Build Some Models

A red block is on top of a blue block.
There are 5 blocks in total.

Let's Build Some Models

A red block is on top of a blue block.
There are 5 blocks in total.

Domain

Some part of the world about which we wish to express some knowledge

Let's Build Some Models

A green block is on top of a blue block

Let's Build Some Models

A green block is on top of a blue block

Closed World Assumption

The only things that are true are the things that are stated (or derived)

Let's Build Some Models

A green block is on top of a blue block.
Smaller blocks can go on bigger blocks.

Let's Build Some Models

A green block is on top of a blue block.
Smaller blocks can go on bigger blocks.

Domain Closure Assumption

All elements of the domain are explicitly mentioned.

Let's Build Some Models

A green block is on top of a blue block.
Smaller blocks can go on bigger blocks
There is a red block.

Let's Build Some Models

A red block is on top of a blue block.
A blue block is on top of a red block.

Let's Build Some Models

A red block is on top of a blue block.
A blue block is on top of a red block.

Consistency

A model is consistent if there is at least one world that it models, otherwise it is inconsistent.

Let's Build Some Models

Block A is on top of a white block.
Block B is on top of a white block.
There are two yellow blocks.

Let's Build Some Models

Block A is on top of a white block.
Block B is on top of a white block.
There are two yellow blocks.

Unique Names Assumption

Things with different names are necessarily unique.

Database Semantics

Together the following assumptions describe Database Semantics. These are not true of logic in general and when they are assumed they change the semantics (the meaning) of a statement. Later (in a few weeks) we will show how they can be written directly in first-order logic.

Closed World Assumption

The only things that are true are the things that are stated (or derived)

Domain Closure Assumption

All elements of the domain are explicitly mentioned.

Unique Names Assumption

Things with different names are necessarily unique.

Getting Formal: Objects and Relations

We assume finite sets of object symbols $\mathcal{O}$ and relation symbols $\mathcal{R}$

A relation symbol $r \in \mathcal{R}$ has an arity given by $arr(r)$

(the arity of a relation is the number of arguments it applies to)

Getting Formal: Objects and Relations

We assume finite sets of object symbols $\mathcal{O}$ and relation symbols $\mathcal{R}$

A relation symbol $r \in \mathcal{R}$ has an arity given by $arr(r)$

(the arity of a relation is the number of arguments it applies to)

A fact is of the form $r(o_1, \ldots, o_{arr(r)})$ given $r \in \mathcal{R}$ and $o_i \in \mathcal{O}$

e.g. a relation symbol applied to the necessary number of objects

It is a convention that we write these symbols using lowercase

Relations are Tables...

...assuming Database Semantics

id	*name*	*course*
1	Adam	French
2	Raj	Fashion
3	Vlad	Music
4	Husain	Architecture
5	Sara	Engineering

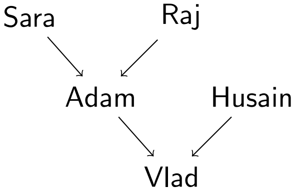
student(1, Adam, French)
student(2, Raj, Fashion)
student(3, Vlad, Music)
student(4, Husain, Architecture)
student(5, Sara, Engineering)

Datalog assumes Database Semantics

Relations are Trees/Graphs. . .

. . . but need additional rules to make $\rightarrow$ transitive, antisymmetric etc

child	*parent*
Adam	Sara
Adam	Raj
Vlad	Adam
Vlad	Husain



We'll revisit this later.

Properties

We often call unary properties (with arity 1) properties

For example, red(car), lecturer(giles), expensive(yacht).

Properties

We often call unary properties (with arity 1) properties

For example, red(car), lecturer(giles), expensive(yacht).

This places red(car), green(car), and fast(car) at the same conceptual level

Maybe colour should be a relationship between cars and colours e.g. color(car, red) and color(car, green)

Properties

We often call unary properties (with arity 1) properties

For example, red(car), lecturer(giles), expensive(yacht).

This places red(car), green(car), and fast(car) at the same conceptual level

Maybe colour should be a relationship between cars and colours e.g. color(car, red) and color(car, green)

Perhaps it is useful to treat the notion of properties uniformly e.g. hasProperty(car, red), hasProperty(yacht, expensive)

Properties

We often call unary properties (with arity 1) properties

For example, red(car), lecturer(giles), expensive(yacht).

This places red(car), green(car), and fast(car) at the same conceptual level

Maybe colour should be a relationship between cars and colours e.g. $\text{color}(\text{car}, \text{red})$ and $\text{color}(\text{car}, \text{green})$

Perhaps it is useful to treat the notion of properties uniformly e.g. $\text{hasProperty}(\text{car}, \text{red})$, $\text{hasProperty}(\text{yacht}, \text{expensive})$

The way we model things matters but there is not one best approach. Like database design, it is a bit of an art-form and there is good practice.

Although next time we will see how it can impact on reasoning

Rules

Define how new facts can be inferred from old facts

A rule is of the form $f_1, \ldots, f_n \Rightarrow f_{n+1}$ where f_i are *facts* ($n \geq 0$)

The meaning (semantics) is *if all the facts on the left (body) are true then the fact on the right (head) is true*

Rules

Define how new facts can be inferred from old facts

A rule is of the form $f_1, \ldots, f_n \Rightarrow f_{n+1}$ where f_i are *facts* ($n \geq 0$)

The meaning (semantics) is *if all the facts on the left (body) are true then the fact on the right (head) is true*

Facts in rules can contain variables (placeholders) in place of objects

Restriction: Let $var(f)$ be the variables in fact f then

$$var(f_{n+1}) \subseteq var(f_1) \cup \ldots \cup var(f_n)$$

e.g. the variables in the rule's conclusion must appear in the premises

Such rules can be seen as *templates* for ground (variable-free) rules. Due to domain closure a rule has a finite number of ground instances.

Facts and Rules Logically

If you are familiar with predicate/first-order logic then

Facts are predicates

Rules are definite clauses e.g. universally quantified disjunctions containing *exactly* one positive literal

The variable occurrence restriction is quite artificial and we'll see why it's useful next lecture

We'll return to this later in the course

Recursive Rules

It is a convention to write variables starting with an Uppercase

It is common to use rules to give recursively defined relations e.g.

$$\begin{aligned} \text{parent}(X, Y) &\Rightarrow \text{ancestor}(X, Y) \\ \text{parent}(X, Z), \text{ancestor}(Z, Y) &\Rightarrow \text{ancestor}(X, Y) \end{aligned}$$

(this is why Datalog is relational algebra + recursion)

Extensional and Intensional Relations

A relation is extensional if it is defined by facts alone e.g. it does not appear in the head of a rule.

A relation is intensional if it is (partially) defined by rules e.g. it appears in the head of a rule.

An intensional definitions gives meaning by specifying necessary and sufficient conditions

Conversely, extensional definitions enumerate everything

Example from Monday

$$\text{teaches}(\text{comp24412}, \text{logic}) \qquad \text{teaches}(\text{comp24412}, \text{prolog})$$
$$\text{about}(\text{comp24412}, \text{ai}) \qquad \text{cool}(\text{ai})$$
$$\text{language}(\text{prolog}) \qquad \text{costs}(\text{yacht}, \text{lotsOfMoney})$$

$$\text{take}(U, C), \text{teaches}(C, X) \Rightarrow \text{know}(U, X)$$
$$\text{take}(U, C), \text{about}(C, X), \text{cool}(X) \Rightarrow \text{cool}(U)$$
$$\text{know}(U, X), \text{language}(X) \Rightarrow \text{canProgram}(U)$$
$$\text{canProgram}(U) \wedge \text{know}(U, \text{logic}) \Rightarrow \text{hasGoodJob}(U)$$
$$\text{hasGoodJob}(U) \Rightarrow \text{has}(U, \text{lotsOfMoney})$$
$$\text{has}(U, X), \text{costs}(Y, X) \Rightarrow \text{has}(U, Y)$$

Let's Model Something

Let's Model Something

Datalog has quite a few restrictions that are frustrating but disappear later with Prolog or full first-order logic.

Some things that seem like restrictions at first are circumvented by the database semantics.

By the Way: Datalog Syntax

You will probably see rules written as

```
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

That's proper Datalog... I'm using logical notation to get you used to it for later. The above is the syntax you'll see in Prolog.

Semantics

Let us call a collection of facts and rules a knowledge base

What is the *meaning* of a knowledge base i.e. its semantics?

What do we mean by this?
What are the set of facts that follow from the knowledge base.

For this we are going to turn to the heavyweight pursuit of *model theory*

Interpretations

An interpretation $\mathcal{I}$ is a function that assigns meaning (truth) to symbols, and hence the facts and rules that contain them

An interpretation $\mathcal{I}$ gives a truth value to each fact built from $\mathcal{O}$ and $\mathcal{R}$. If $\mathcal{I}$ applied to fact f is *true* then we say $\mathcal{I}$ satisfies f.

Interpretations

An interpretation $\mathcal{I}$ is a function that assigns meaning (truth) to symbols, and hence the facts and rules that contain them

An interpretation $\mathcal{I}$ gives a truth value to each fact built from $\mathcal{O}$ and $\mathcal{R}$. If $\mathcal{I}$ applied to fact f is *true* then we say $\mathcal{I}$ satisfies f.

A substitution is a *map* (function with finite domain) from variables to objects. We can apply a substitution to a fact containing variables to produce a new fact e.g. $\{X \mapsto a\}(f(X, Y)) = f(a, Y)$.

An interpretation $\mathcal{I}$ satisfies a rule $f_1, \ldots, f_n \Rightarrow f_{n+1}$ if for every substitution σ whenever $\mathcal{I}$ satisfies $\sigma(f_1), \ldots \sigma(f_n)$ it also satisfies $\sigma(f_{n+1})$.

Interpretations are different Realities

If our knowledge base is

$$\mathsf{red}(\mathsf{block1}) \qquad \mathsf{blue}(\mathsf{block2}) \qquad \mathsf{onTop}(\mathsf{red}, \mathsf{blue})$$

then we can have multiple interpretations of the knowledge base

Interpretations are different Realities

If our knowledge base is

$$\mathsf{red}(\mathsf{block1}) \qquad \mathsf{blue}(\mathsf{block2}) \qquad \mathsf{onTop}(\mathsf{red}, \mathsf{blue})$$

then we can have multiple interpretations of the knowledge base

Unless we have the closed world assumption

In which case, there is one minimal interpretation

Note that an interpretation can be defined exactly by the facts true in it

Consistency and Consequence

An interpretation satisfies a knowledge base if it satisfies all facts and clauses in the knowledge base.

A knowledge base is consistent if there is at least one interpretation that satisfies it.

A fact f is a consequence of the knowledge base if every interpretation that satisfies the knowledge base also satisfies f.

If a fact f is a consequence of a knowledge base $\mathcal{KB}$ we write $\mathcal{KB} \models f$

Aside: Symbols are just Symbols

$\text{rich}(\text{giles}) \qquad \text{rich}(X) \Rightarrow \text{happy}(X)$

Aside: Symbols are just Symbols

$$\mathrm{rich}(\mathrm{giles}) \qquad \mathrm{rich}(X) \Rightarrow \mathrm{happy}(X) \qquad \mathrm{happy}(\mathrm{giles})$$

Aside: Symbols are just Symbols

$$\text{rich}(\text{giles}) \qquad \text{rich}(X) \Rightarrow \text{happy}(X) \qquad \text{happy}(\text{giles})$$

$$\text{fruit}(\text{giles}) \qquad \text{fruit}(X) \Rightarrow \text{dancing}(X)$$

Aside: Symbols are just Symbols

$$\text{rich(giles)} \qquad \text{rich}(X) \Rightarrow \text{happy}(X) \qquad \text{happy(giles)}$$

$$\text{fruit(giles)} \qquad \text{fruit}(X) \Rightarrow \text{dancing}(X) \qquad \text{dancing(giles)}$$

Aside: Symbols are just Symbols

$$\text{rich}(\text{giles}) \qquad \text{rich}(X) \Rightarrow \text{happy}(X) \qquad \text{happy}(\text{giles})$$

$$\text{fruit}(\text{giles}) \qquad \text{fruit}(X) \Rightarrow \text{dancing}(X) \qquad \text{dancing}(\text{giles})$$

The consequences are equivalently valid. The symbols don't care that they don't make sense. You have to make sure the knowledge base is *sensible*.

Summary

Warnings:

- I have been a little lazy with some bits that we will revisit properly later when considering reasoning in full first-order logic
- I'm treating Datalog relatively logically (rather than programatically) so have ignored some 'features'
- I'm avoiding the word *model* in the logical sense on purpose at the moment

The idea is to introduce you to the basic concepts in a simple setting

Next time: we now know what a knowledge base is, but how do we use it? We will look at what a query is and what the answer to a query is, and how to compute answers.