Logic for Computer Science

08 – Relations

Wouter Swierstra

University of Utrecht

Organization

- The midterm is coming up. . . this is the last lecture with mid-term material.
- Next week Thursday: revision lecture on how to prepare for the mid-term.

Exam tips

- Don’t worry about writing beautiful mathematical symbols in Remindo!
- Feel free to use /\ rather than $\wedge$, etc. - any reasonable shorthand is fine, provided I can understand what you wrote.
- All questions must be entered in Remindo - but I will provide you with scrap paper.
- Practice, practice, practice!

Last time

Functions

Today

Relations

Previously

The **truth set** associated with a predicate P is the set:

$$\{x : P(x) \text{ holds}\}$$

We can also consider the truth set associated with predicates that take more than one argument:

$$\{(x, y, z) : R(x, y, z) \text{ holds}\}$$

For example: $R(x, y, z)$ might hold if and only if the customer with ID x ordered product y on the date z.

The corresponding truth set then defines a subset of the set $\text{CustomerID} \times \text{ProductID} \times \text{Date}$.

‘Not all pairs’

More generally, a **relation** R on A and B is a subset of a cartesian product $A \times B$.

We write R(a,b) or aRb if $(a,b) \in R$; that is, a and b are related by R.

Examples of relations

- The less-than-or equals relation on numbers, $x \leqslant 4$
- The equality relation, $x = y$
- The ‘is-an-ancestor-of’ or the parenthood relation between humans.
- The ‘equivalent’ relation between programs, describing when two programs behave the same.
- The propositionally equivalent relation between propositions.

Relations and databases

In the next period, you'll take the course on *Databases*.

There you'll see how to model databases and database queries using *relations* and *relational algebra* respectively.

The central idea is that we can model a database table as a *relation* – capturing the entries in the database.

For example:

$$\{(s, g, d) : s \text{ is a student who obtained the grade } g \text{ on the date } d \text{ for the logic class}\}$$

Functions vs relations

Functions and relations seem are similar concepts – but there are important differences.

- Given a function $f : A \to B$, we can construct the relation $\{(x, f(x)) : x \in A\}$, sometimes referred to as the **graph** of the function f.
- But not all relations are functions. For example, the 'is-an-ancestor-of' relation between me and my ancestors is not a function. Each person has many different ancestors.
- A function $f : A \to B$ associates a value in B with each $a \in A$; in a relation each $a \in A$ may be associated with zero, one or many elements of B.
- Given a relation on $A \times B$ such that each $a \in A$ is related to exactly one $b \in B$ - this determines a function $f : A \to B$

Terminology

A relation between two sets A and B is called a **binary** relation.

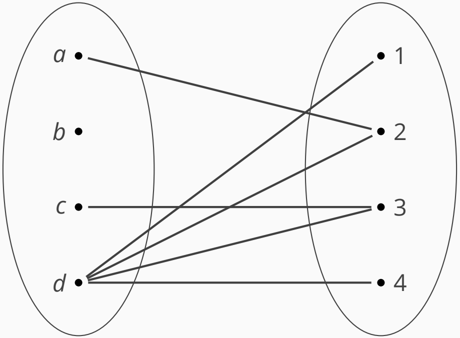
Many familiar binary relations use an **infix** operator: $\subseteq$, $=$, $\Leftrightarrow$, $\leqslant$, $\ldots$

Given a relation $R \subseteq A \times B$ we sometimes refer to A as the **source** and B as the **target** of R.

When a relation R is a subset of $A \times A$ we sometimes call R a **homogeneous relation**;

When a relation R is a subset of $A \times B$ (for two different sets A and B) we call R a **heterogeneous** relation.

Visualizing relations



Extreme relations

- A $\times$ B is also a relation – every pair of elements (a,b) where a $\in$ A and b $\in$ B, is related.
- The empty set $\emptyset$ is also a subset of A $\times$ B – no two elements are related.
- The equality relation on a set A is defined by { (a,a) : a $\in$ A}.
- For any relation *R* on A $\times$ B, we can define the **inverse relation** on B $\times$ A as follows:

$$R^{-1} = \{(b, a) : (a, b) \in R\}$$

For example, given the relation < $\subseteq \mathbb{N} \times \mathbb{N}$, we can define the inverse relation $<^{-1}$ – more commonly known as $>$.

Combining relations

We can use familiar operations for manipulating sets to manipulate relations:

- $a \leqslant b = (a < b) \cup (a = b)$
- Parent = Father $\cup$ Mother
- Son = Child $\cap$ Male

Domain and range

Given a relation $R \subseteq A \times B$, we sometimes refer to the

- the **domain** of R is given by

$$\{a \in A : \exists b \in B \quad (a, b) \in R\}$$

- the **range** of R is given by

$$\{b \in B : \exists a \in A \quad (a, b) \in R\}$$

Properties of relations

Properties of relations

Just as we studied injective, surjective, or bijective functions, there are plenty of properties of relations worth studying:

- reflexive relations
- symmetric relations
- asymmetric relations
- antisymmetric relations
- transitive relations
- relational composition

Reflexive relations

A relation is **reflexive** if R(x,x) for all x.

Examples

- equality
- propositionally equivalent formulas;

Non-examples

- x < y (where x and y are numbers);
- The strict-subset relation on sets.
- Is-a-parent-of relation between people.

Reflexive relations

A relation is **reflexive** if R(x,x) for all x.

Examples

- equality
- propositionally equivalent formulas;

Non-examples

- x < y (where x and y are numbers);
- The strict-subset relation on sets.
- Is-a-parent-of relation between people.

If a relation R is 'never reflexive', that is, $\forall x \quad \neg(xRx)$ we call R **irreflexive**.

Symmetric relations

A relation is **symmetric** if R(x,y) implies R(y,x).

Examples

- equality
- propositionally equivalent formulas;
- the ‘is a sibling of’ relation;

Non-examples

- $x \leqslant y$ (where x and y are numbers);
- The subset relation on sets.
- The graph of the sort function.

Asymmetric relations

A relation is **asymmetric** if $R(x,y)$ implies $\neg R(y,x)$

Question: Can a relation be both symmetric and asymmetric? What is an example of an asymmetric relation?

Asymmetric relations

A relation is **asymmetric** if R(x,y) implies ¬R(y,x)

Question: Can a relation be both symmetric and asymmetric? What is an example of an asymmetric relation?

Examples

- The < relation on numbers;
- The ‘is-a-strict-prefix-of’ relation on strings.
- . . .

Antisymmetric relations

A relation is **antisymmetric** if R(x,y) and R(y,x) implies x = y.

Examples

- Equality;
- $\leqslant$ on natural numbers;
- $\subseteq$ on sets.

Non-examples

- Equivalence of propositional formulas.

Transitive relations

A relation is **transitive** if R(x,y) and R(y,z) implies R(x,z).

Question: What examples of such relations?

Transitive relations

A relation is **transitive** if R(x,y) and R(y,z) implies R(x,z).

Question: What examples of such relations?

Examples

- Subsets, equality, comparison of numbers, prefixes of strings, . . .

Relational composition

We can compose relations, just as we compose functions.

Given a relation R on A $\times$ B and a relation S on B $\times$ C, we can form the composed relation R $\circ$ S on A $\times$ C as follows:

$$R \circ S = \{(a, c) \ : \ \text{there is some } b \in B \text{ such that } aRb \land bSc\}$$

Properties

We can rephrase some of these properties in terms of subsets:

If R is a relation on $A \times A$

- R is reflexive when it contains the equality relation, $= \subseteq R$
- R is symmetric when $R^{-1} \subseteq R$ (or equivalently, when $R \subseteq R^{-1}$)
- R is transitive when $R \circ R \subseteq R$

Equivalence relations

An **equivalence relation** is a relation that is:

- reflexive – R(x,x) for all x.
- symmetric – R(x,y) implies R (y,x)
- transitive – R(x,y) and R(y,z) implies R(x,z)

The canonical example of such a relation is equality.

Equivalence relations

An **equivalence relation** is a relation that is:

- reflexive – R(x,x) for all x.
- symmetric – R(x,y) implies R (y,x)
- transitive – R(x,y) and R(y,z) implies R(x,z)

The canonical example of such a relation is equality.

But many others exist!

Equivalence classes

Equivalence classes

One common construction is to study objects ‘up-to-equivalence’ under some equivalence relation:

- programs equal up to renaming of variables;
- lists equal up to reordering;
- propositional logic formulas up to equivalence;
- shapes independently of shifting along the x or y-axis;
- cars independently of their colour;
- rational numbers independently of any common divisors of the numerator and denominator.

This pops up again and again – we sometimes want to avoid certain details.

Equivalence classes

Given an equivalence relation R on $A \times A$, we can define the **equivalence class** of all the elements related to some $a \in A$ as follows:

$$[a]_R = \{x \in A \ : \ (a, x) \in R\}$$

This characterizes all the elements related to a under R.

Equivalence classes

Given an equivalence relation R on $A \times A$, we can define the **equivalence class** of all the elements related to some $a \in A$ as follows:

$$[a]_R = \{x \in A \ : \ (a, x) \in R\}$$

This characterizes all the elements related to a under R.

Now consider all the equivalence classes, written A/R:

$$A/R = \{[a]_R \ : \ a \in A\}$$

Partitions

A **partition** of a set A consists of a series of non-empty sets $A_1, A_2, A_3, \ldots A_n$ such that:

- $A_i \cap A_j = \emptyset$ when $i \neq j$;
- $A_1 \cup A_2 \cup \ldots \cup A_n = A$

Intuitively a partition divides the original set A into n separate pieces.

(This definition is a bit simpler than the one in the book and only works for finite sets)

Partitions

A **partition** of a set A consists of a series of non-empty sets $A_1, A_2, A_3, \ldots A_n$ such that:

- $A_i \cap A_j = \emptyset$ when $i \neq j$;
- $A_1 \cup A_2 \cup \ldots \cup A_n = A$

Intuitively a partition divides the original set A into n separate pieces.

(This definition is a bit simpler than the one in the book and only works for finite sets)

Theorem Given a relation $R \subseteq A \times A$, the equivalence classes $\{[a]_R : a \in A\}$ form a partition of A.

Theorem Given a relation $R \subseteq A \times A$, the equivalence classes $\{[a]_R : a \in A\}$ form a partition of A.

Proof

We need to show:

- Each equivalence class $[a]_R$ is non-empty.
- The union of all equivalence classes is A.
- The equivalence classes are disjoint.

Question

Why do these three properties hold?

Proof

Theorem Given a relation $R \subseteq A \times A$, the equivalence classes $\{[a]_R : a \in A\}$ form a partition of A.

Proof

We need to show:

- Each equivalence class $[a]_R$ is non-empty as $a \in [a]_R$ and R is reflexive.
- The union of all equivalence classes is A. Once again, because $a \in [a]_R$ for each $a \in A$, the union of all equivalence classes is equal to A.
- The equivalence classes are disjoint.

To prove this last point we must show that if $x \in [a]_R$ and $x \in [b]_R$, then $[a] = [b]$.

From our assumption we know that xRa and xRb.

From the symmetry and transitivity of R we can conclude that aRb and hence $[a] = [b]$.

Equivalence relations

Working with equivalence classes lets us ignore certain details that are not of importance – choice of variable names, colour of cars, position of shapes, etc.

This construction in Computer Science pops up over and over!

Let's look at an example. . .

Rationals from pairs of naturals

We could define the (positive) rationals as the pair $\mathbb{N} \times \mathbb{N} \ldots$

But then: (1,2) $\neq$ (2,4) – which is not what we want.

Instead, we consider the relation $(a, b) \sim (c, d)$ that holds when $a \times d = b \times c$.

Question

Prove this is an equivalence relation.

Equivalence class of the rationals

We can define the rationals as $\mathbb{N} \times \mathbb{N}_{>0} / \sim$, that is:

equivalence classes of pairs of natural numbers (where the second number is greater than zero);

I claim that any function we define over the rationals cannot distinguish between (1,2) and (2,4). . .

For example consider the following 'function':

wrong(x,y) = x + y

Claim

This does not define a function on the rationals.

What is wrong?

wrong(x,y) = x + y

Every function should map an input to a *unique* output.

This wrong function maps:

wrong(1,2) = 3

wrong(2,4) = 6

Yet (1,2) and (2,4) are *in the same equivalence class*.

Hence ‘wrong’ maps *the same input* to **different** outputs.

Therefore ‘wrong’ is not a valid function!

Functions over equivalence classes

To define a function $f : A/R \rightarrow B$ over equivalence classes, we need to check that

for all $a \in A$ and $a' \in A$, if aRa' then $f(a) = f(a')$.

In words, f maps *related* inputs to the *same* output.

Or put differently, f cannot distinguish between related inputs.

Applications

We do this in computer science all the time:

- a compiler is a function on programs (that should not distinguish between the same program using different variable names);
- calculating the surface area of a shape should be independent of *where* the shape is located;
- I can represent a set of elements as an array (provided I never observe the *order* of the elements).

Working with equivalence classes gives us a mathematical construction to *hide* certain unimportant information.

Example

Suppose we have some class `Car` storing information about a cars make, model, colour, etc.

We can define an equivalence relation on `Car` objects easily enough:

$c_1 \sim c_2$ if and only if c_1`.colour` = c_2`.colour`

(Why is is an equivalence relation?)

What kind of functions can we define on the equivalence classes that we get by partitioning all cars by their colour?

Example

Does this define a function on equivalence classes?

```
showColour : Car ⟶ String
showColour(c) = toString(c.colour)
```

Example

Does this define a function on equivalence classes?

```
showColour : Car → String
showColour(c) = toString(c.colour)
```

Yes! The `showColour` returns the same string for two cars in the same equivalence class: a red Fiat and a red Ferrari will both produce the string “red”.

Example

Does this define a function on equivalence classes?

```
isEV : Car ⟶ Bool
isEV (c) = isElectric(c.motor)
```

Example

Does this define a function on equivalence classes?

```
isEV : Car ⟶ Bool
isEV (c) = isElectric(c.motor)
```

No! A red Tesla and a red Ferrari are in the same equivalence class (they are both red) – yet one will produce `True`; the other will produce `False`.

Example - cars and colours

This example shows that by considering the *equivalence classes* of cars, we limit the information you can use:

- We can observe a car's colour;
- But cannot inspect it's make, model, motor, etc.

This is a common pattern in Computer Science, where you want to *hide* certain implementation details.

Equivalence classes gives us the mathematics to do so.

Theorems

We can use this result to turn any *surjection* into a bijection. . .

Theorems

We can use this result to turn any *surjection* into a bijection. . .

Given a function $f : A \rightarrow B$, we can define the relation $R_f \subseteq A \times A$ as:

$$xR_fx' \text{ iff } f(x) = f(x')$$

Theorem Any surjection f gives rise to a bijection A/R_f and B.

Theorems

We can use this result to turn any *surjection* into a bijection. . .

Given a function $f : A \to B$, we can define the relation $R_f \subseteq A \times A$ as:

$$xR_fx' \text{ iff } f(x) = f(x')$$

Theorem Any surjection f gives rise to a bijection A/R_f and B.

Proof

- We need to show that R_f is an equivalence relation;
- that we can define a function $\widetilde{f}$ from A/R_f to B;
- and that this function is a bijection.

Equivalence relation

For any function $f : A \to B$, we can construct the following equivalence relation:

$$xR_f x' \text{ iff } f(x) = f(x')$$

Question

Show that this is an equivalence relation.

Function

For any function $f : A \to B$, we can construct the following equivalence relation:

$$xR_f x' \text{ iff } f(x) = f(x')$$

Furthermore, we can define a function $\widetilde{f}$ from A/R_f to B by:

$$\widetilde{f}([x]) = f(x)$$

Function

For any function $f : A \rightarrow B$, we can construct the following equivalence relation:

$$xR_f x' \text{ iff } f(x) = f(x')$$

Furthermore, we can define a function $\widetilde{f}$ from A/R_f to B by:

$$\widetilde{f}([x]) = f(x)$$

We need to check that if $xR_f x'$ then $\widetilde{f}([x]) = \widetilde{f}([x'])$.

But this follows from the definition of R_f!

Bijection

For any function $f : A \rightarrow B$, we can construct the following equivalence relation:

$$xR_f x' \text{ iff } f(x) = f(x')$$

Furthermore, we can define a function $\widetilde{f}$ from A/R_f to B by:

$$\widetilde{f}([x]) = f(x)$$

If f is a *surjection*, then $\widetilde{f}$ is a *bijection*.

Bijection

For any function $f : A \to B$, we can construct the following equivalence relation:

$$xR_f x' \text{ iff } f(x) = f(x')$$

Furthermore, we can define a function $\widetilde{f}$ from A/R_f to B by:

$$\widetilde{f}([x]) = f(x)$$

If f is a *surjection*, then $\widetilde{f}$ is a *bijection*.

- To show $\widetilde{f}$ is surjective, we use the fact that f is already surjective.

Bijection

For any function $f : A \to B$, we can construct the following equivalence relation:

$$xR_fx' \text{ iff } f(x) = f(x')$$

Furthermore, we can define a function $\widetilde{f}$ from A/R_f to B by:

$$\widetilde{f}([x]) = f(x)$$

If f is a *surjection*, then $\widetilde{f}$ is a *bijection*.

- To show $\widetilde{f}$ is surjective, we use the fact that f is already surjective.
- To prove injectivity amounts to showing that if $[x], [x'] \in A/R_f$ and $\widetilde{f}([x]) = \widetilde{f}([x'])$, then $[x] = [x']$.
  But by definition of $\widetilde{f}$, we know that if $\widetilde{f}([x]) = \widetilde{f}([x'])$ then $f(x) = f(x')$, but then by definition of R_f we know that xR_fx' and therefore $[x] = [x']$.

Why equivalence classes?

This is a first ‘non-obvious’ example of a mathematical construction that has many applications.

The previous proof relies on bringing together a great deal of material we’ve covered in the previous weeks:

- propositional and predicate logic;
- proof sketches;
- notions of injectivity and surjectivity;
- relations and equivalence classes;
- . . .

Understanding the proof is a good way to stress test your own understanding of this material.

Today

- Functions and their properties
- Relations and their properties

Defining relations

When I first learned about relations, I really didn't understand them well.

We are used to defining functions such as:

$f(x) = x^3 + 17$

```
public int triple(int x) {...}
```

And we can study and define these functions without every talking about their graphs.

Defining relations

When I first learned about relations, I really didn't understand them well.

We are used to defining functions such as:

$f(x) = x^3 + 17$

```
public int triple(int x) {...}
```

And we can study and define these functions without every talking about their graphs.

But many books only mention relations as being defined as a subset of $A \times B\ldots$

And don't give you a 'language' to **define** relations.

Once we cover induction and recursion, I can give a more precise account of how to define relations on infinite sets – and define more interesting relations than the ones we have covered today.

Material

Modelling Computing Systems – Chapter 7

Supporting material on defining functions over equivalence classes