CSC 202 NOTES
Spring 2010: Amber Settle

Week 3: Monday, April 12, 2010

Announcements

- The second assignment is due now
- The third assignment is due Monday, April 19th
- We now have a grader
- The midterm is Monday, April 26th – I will post a study guide next week
- Review of the first assignment
  - Question 2: Translation of English into propositional logic
  - Other questions?

(Review of) set theory

A **set** is a collection of elements.

Examples:
- C = {Charlie, Djengo, Hugin, Joon, Marcel, Simone}
- P = {x: x is prime}

To show element **e is an element** of a set S, we write: $e \in S$
To show element **e is not an element** of a set, we write $e \notin S$

Examples: Djengo $\in$ C, Meg $\notin$ C, and $8 \notin P$

We write $|X|$ for the **cardinality** of X, that is, the number of elements X contains.

Example: $|C| = 6$

Two sets are **equal** if they contain the same elements.

Note that the order of elements in sets is arbitrary.
Also, it doesn't matter if we list elements more than once – the set is still the same.

Example: {Djengo, Djengo, Simone} = {Simone, Djengo}

A is a **subset** of B, written $A \subseteq B$, if every element of A is also an element of B.

The empty set does not contain any elements. It is written {} or ϕ.

The **union** $A \cup B$ of two sets A and B is $A \cup B = \{e : e \in A \lor e \in B\}$.
The **intersection** $A \cap B$ of two sets A and B is $A \cap B = \{x : x \in A \land x \in B\}$.
The **complement** of a set A is $\overline{A} = \{x : x \notin A\}$.
The **difference** of two sets A and B is $A - B = \{x: x \in A \land x \notin B\}$.

Tuples

A **pair** of two objects x and y is written as (x, y).

The main characteristic of a pair is that it is **ordered**: a pair has a first component and a second component.

Thus two pairs (x, y) and (u, v) are **equal** when x = u and y = v.

The **Cartesian product** of two sets X and Y is defined by: $X \times Y = \{ (x, y) : x \in X \land y \in Y\}$

Example: Let X = {Djengo, Hugin} and Y = {Simone, Joon}.

Then X x Y = {(Djengo, Simone), (Djengo, Joon), (Hugin, Simone), (Hugin, Joon)}

We can take the product of more than two sets by building **tuples** instead of pairs.

In general, if we have sets $X_1, X_2, \dots, X_n$, their Cartesian product is:
$X_1 \times X_2 \times \dots \times X_n = \{(x_1, x_2, \dots, x_n) : x_1 \in X_1 \land x_2 \in X_2 \land \dots \land x_n \in X_n\}$

New set theory

We have one last concept in set theory to discuss

Power set

The **power set** is the set of all subsets of a set.

More formally, $\wp(A) = \{X : X \subseteq A\}$.

Example: With B = {Djengo, Simone}, we have
$\wp(B) = \{\phi$, {Djengo}, {Simone}, {Djengo, Simone}}

The empty set and the entire set A always appear in the power set $\wp(A)$ since they are the trivial subsets of the set.

Exercise: Compute the power sets of the following sets

- {Djengo, Simone, Hugin}
- ϕ

What is the size of the power set of S in relation to the size of S?

$|\wp(S)| = 2^{|S|}$ for finite sets S

Why?

Subsets differ by which elements they contain, and each element can either be contained or not.
This gives us two choices per element, yielding: 2 * 2 * … * 2 (|S| times) = $2^{|S|}$

Since $2^n > n$ for all n, the power set of a set is always larger than the set itself. (This is also true for infinite sets, as we will see later).

Exercises:

1. How many elements does $\wp$({Simone, Hugin, Joon, Djengo, Marcel}) have?
2. Compute $\wp$({1, 2, 3}) and draw the result as a diagram which shows how the subsets are included in each other. Start with the empty set at the bottom, {1, 2, 3} at the top. Draw an arrow from one subset A to another B if $A \subseteq B$.

First-order logic

First-order logic can be used to describe the relationships between members as defined in predicate logic and set theory.

Relations

When we say something like "Chad is a brother of Amber" or "Amber is the mother of Erin", we are expressing a relation between two different objects (namely members of the Settle family).

In mathematics the objects are different (e.g. the natural numbers, the real numbers, etc.) but the idea is the same.

$3 < 4$ expresses a relationship between 3 and 4, namely that 3 comes before 4 in the natural numbers.

We write **x R y** to express that x is in relation to R.

Example: R could mean "is a brother of"

Jack R Bobby means Jack is a brother of Bobby
Bobby R Ted means that Bobby is a brother of Ted
But then, of course, we know that Jack R Ted since a brother of a brother is still a brother to the first man.

This can be written as **x R y ∧ y R z → x R z**

This means that the relation "is a brother of" is **transitive**.

There are many other relations that are transitive, for example, the relation of being smaller (or larger) than on the natural numbers.

The relation "is a brother of" is **binary**, meaning that it relates two objects.

Relations can be more complex than this by relating more than two objects at a time.

Example: A **ternary** relation relates three objects at once

1. "x places an order for y at time t"
2. "x is a giver of y to z"

Ternary relations cannot be written with the objects it is relating around it since it only has two sides and three objects it is relating.

The general way of **writing a relation** is R(x,y) or R(x,y,z) or in general $R(x_1, x_2, \ldots, x_n)$.

The number of objects in the relation is the **arity of the relation**:

1. Relations on one object are called **unary**
2. Relations on two objects are called **binary**
3. Relations on three objects are called **ternary**
4. Relations on n objects are called **nary**

Examples: You can think about a **unary relation as a property** of the object

1. "is feline"
2. "is odd"
3. "is prime"

Properties of relations

Relations have been studied extensively because they are central to mathematics, science, and computing (e.g. relational databases).

In order to understand what is known about relations we have to abstract some properties of relations.

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

Example: ≤ is reflexive but < is not (why not?)

A binary relation R is **symmetric** if R(x,y) → R(y, x) for all x and y.

Example: = is symmetric but ≤ is not

A binary relation R is **transitive** if $R(x,y) \wedge R(y,z) \rightarrow R(x,z)$ for all x, y, and z.

Example: ≤ is transitive but "being a friend of" is not

Exercises:

1. R(x,y) = "x likes y". This relation is not reflexive, symmetric, or transitive. (Why in each case?)
2. R(x,y) = "|x – y| is even" where the universe is the natural numbers. This relation is reflexive, symmetric, and transitive. (Why in each case?)
3. R(x,y) is "x and y are coprime" meaning that their greatest common divisor is 1. Show that this relation is not transitive.

Pair exercise: For the relation "is sibling of" determine whether it is reflexive, symmetric, and/or transitive. Either argue why the relation has the property or give a counterexample if it fails to have the property.

We can use these properties to define the most important class of relations: a binary relation is **an equivalence relation** if it is reflexive, symmetric, and transitive.

An equivalence relation naturally groups objects.

Let R be an equivalence relation and for every element a define **the equivalence class of a under R** as follows: $[a]_R := \{b: R(a,b)\}$

Since R is an equivalence relation, any two elements in $[a]_R$ are related by R.

Let $x \in [a]_R$ and $y \in [a]_R$.
This means that R(a, x) and R(a,y).
But then R(x,a) (by symmetry) and R(x,y) (by transitivity).

We call $[a]_R$ **the equivalence class of a**.

It contains everything equivalent to a under the relation R.

Example: Let our set for the following be {Amber, Andre, Chad, Noelle, Erin, Eva, Charlotte}

1. Let R be "has the same sex as". This divides our set into two equivalence classes: {Andre, Chad} and {Amber, Noelle, Erin, Eva, Charlotte}
2. Let R be "has the same parents as". This divides our set into 5 equivalence classes: {Amber, Chad}, {Noelle}, {Andre}, {Erin}, {Eva, Charlotte}
3. Let R be "has the same age as". This divides our set into 5 (different) equivalence classes: {Amber, Andre}, {Chad}, {Noelle}, {Erin, Eva}, {Charlotte}

Note that **an equivalence class is never empty**. It always contains at least one element.

Another example: Let R(x,y) = "|x - y| is even" over the universe of natural numbers.

We saw earlier that this is an equivalence relation, so that it defines some set of equivalence classes.

What are those classes?

$[0]_R$ contains all natural numbers that have an even difference with 0 (e.g. all x such that $|x – 0|$ is even), in other words, all even numbers.

$[1]_R$ contains all natural numbers that have an even difference from 1 (e.g. all x such that $|x – 1|$ is even), which is the set of all odd numbers.

Since $[0]_R \cup [1]_R$ is the entire set of natural numbers, this means that there are only two equivalence classes for the natural numbers under the relation R.

You can choose other numbers and consider their equivalence classes, but you will only duplicate one of the existing equivalence classes.

E.g. $[3]_R$ = ?

0 and 1 are the **standard representatives** of the natural numbers under the relation R.

Pair exercise: Consider the relationship R(x,y) = "|x – y| is a multiple of 3". Show that R is an equivalence relation and find the equivalence classes of R.

Grouping in databases

SQL allows you to build a very limited type of equivalence classes by allowing you to group records by values of fields.

Example 1: Group the records in the Student table by City

```
SELECT City, count(*)
FROM Student
GROUP BY City;
```

The addition of count(*) to the SELECT clause gives us a count of the number of elements in each group.

Example 2: Group students by degree program

```
SELECT Career, Program, count(*)
FROM Student
GROUP BY Career, Program;
```

Example 3: Give the first year each undergraduate program had a student enrolled grouped by program

```
SELECT Program, min(started)
FROM Student
WHERE Career = 'UGRD'
GROUP BY Program;
```

Example 4: Count how many students have taken each course

```
SELECT CID, Department, CourseName, count(*)
FROM Course, Enrolled
WHERE CourseID = CID
GROUP BY CID, Department, CourseName;
```

Some points about **how GROUP BY works**:

1. The WHERE is checked first, meaning that the table is reduced to the records fulfilling the WHERE condition, and then the records are grouped by the attributes listed in the GROUP BY clause (see Example 3).
2. There are other functions you can use including: sum, max, min, and avg (which do the sum, maximum, minimum, and average of the values in question).
3. If you are selecting attributes in the SELECT clause of a grouped SQL query, you can only select attributes that have been explicitly grouped (see Example 4 – writing only GROUP BY CID would not have been legal SQL).

Exercise: Write a query that lists the first year and the last year that any student started in a program grouped by programs. (This is helpful for deciding if a program is still active or should be eliminated.)

Equivalence classes

There is an alternative way of looking at equivalence classes.

Suppose that we split our universe U into several sets (or categories) that are pairwise disjoint.

Formally, we call a collection U_i where $i \in I$ a **partition** of U if $U = \bigcup_{i \in I} U_i$ and $U_i \cap U_j = \emptyset$ for all $i \neq j$, $i, j \in I$.

If we have a partition of U we can define a **relation R on U** as follows: R(x,y) is true if and only if x and y belong to the same U_i where $i \in I$.

Exercise: Show that R defined as above is an equivalence relation. This means you must verify that it is reflexive, symmetric and transitive.

On the other hand, suppose we are given an equivalence relation R over the universe U.

Consider the collection of all equivalence classes $U_a = [a]_R$ for all $a \in U$.

Note that $U = \bigcup_{a \in I} [a]_R$ since every $a \in U$ is contained in some equivalence class, namely U_a.

We don't know that this is a partition until we verify that there are no two equivalence classes that overlap.

Suppose that U_a and U_b overlap, that is, there is a c such that $c \in U_a$ and $c \in U_b$. This means that R(a,c) and R(b,c).

Since R is symmetric and transitive, we can conclude that R(a,b) so that $b \in U_a$ and therefore $U_b \subseteq U_a$

We can conclude, using a very similar argument that $U_a \subseteq U_b$ so that $U_a = U_b$.

This means that if two equivalence classes overlap, they are identical. So if we drop all the indices in U that are duplicates we have a partition of U.

Summary: Equivalence relations are partitions of the universe into categories.

Ordering relations

When we introduced relations, we discussed relations that order numbers such as $\geq$ and $\leq$.

We saw that these relations were not symmetric.

Nor would we want them to be!

For example, if $a \leq b$ and $b \leq a$ then $a = b$.

This is the property we are missing to produce general ordering relations.

A binary relation R is **anti-symmetric** if $R(x,y) \wedge R(y, x) \rightarrow x = y$

An ordering relation is a reflexive, anti-symmetric, and transitive relation.

Example: Let S = {Amber, André, Chad, Charlotte, Erin, Eva, Noelle}
We can order members of this set by:

1. **Age**: Amber ≥ André ≥ Noelle ≥ Chad ≥Erin ≥ Eva ≥ Charlotte
2. **Height**: Chad ≥ André ≥ Amber ≥ Noelle ≥ Erin ≥ Eva ≥ Charlotte

Pair exercise: Show that $\subseteq$ is an ordering relation on sets.

The previous examples show us that ordering relations can be quite different than $\leq$. The main difference is that two elements might not be comparable by the relation.

For numbers x and y we always have $x \leq y$ or $y \leq x$.

But it is not true that for an arbitrary ordering $\preccurlyeq$ we always have either $x \preccurlyeq y$ or $y \preccurlyeq x$.

Example: {Amber} $\not\subseteq$ {André} and {André}$\not\subseteq$ {Amber}

An ordering relation $\preccurlyeq$ for which $x \preccurlyeq y$ or $y \preccurlyeq x$ for all elements x and y is called a **total ordering**. Otherwise it is a **partial ordering**.

Example: Consider the set W of all possible "words", meaning arbitrary sequences of letters, including "parsimonious" but also including "pdjwejslk".

This set can be totally ordered as follows:

Given two words $x \neq y \in W$, let w be their longest common prefix.

- If w = x or w = y, then let $x \leq_{LEX} y$ if $|x| \leq |y|$ where $|x|$ and $|y|$ are the lengths of x and y.
- If neither w = x or w = y, then both have a letter following w. This means $x = wl_1\ldots$ and $y = wl_2\ldots$ where l_1 and l_2 are letters of the alphabet. We let $x \leq_{LEX} y$ if l_1 comes before l_2 in the alphabet.

This ordering is called the **lexicographic** or **dictionary ordering** of words. It is a simplified variant of the ordering used by dictionaries to arrange words.

Some **specific examples**:

- x = "hand" and y = "handle" have longest common prefix "hand" which is x. So "hand" $\leq_{LEX}$ "handle" since "hand" is shorter.
- x = "brutal" and y = "brunch" have longest common prefix "bru". In x "bru" is followed by "t" whereas in y "bru" is followed by "n". Since "n" < "t" is the English ordering of the alphabet, "brunch" $\leq_{LEX}$ "brutal"

The lexicographic ordering does **disagree with our standard ordering of the numbers** (if we extend the notion of words to be made from letters as well as numbers.

For example, if we consider 112 and 12 as words "112" and "12", then "112" would be listed before "12". Under standard numeric ordering, 12 comes before 112.

We didn't specify which alphabet we were using (although you probably assumed the English alphabet), and the lexicographic ordering makes sense for any alphabet. We can also use any ordering of the alphabet that we like.

Strict ordering relations

For numbers, we distinguish between < and ≤, and the same distinction can be made for arbitrary orders.

First we need a definition: A binary relation R is **anti-reflexive** if $\overline{R(x,x)}$ for all x.

We call $\prec$ a **strict ordering relation** if it is anti-reflexive, anti-symmetric, and transitive.

We can also distinguish between partial and total strict orders. A **strict ordering $\prec$ is total** if for any two distinct elements x and y we either have $x \prec y$ or $y \prec x$.

Note that we cannot require this for all pairs of elements since for $x = y$ we do not have either $x \prec y$ or $y \prec x$.

Note: By definition **strict ordering relations are not ordering relations** (since they are not reflexive).

Ordering in databases

SQL allows you to sort your output by any field using ORDER BY.

Example: We can list the students in the Student table in the order in which they started.

```
SELECT *
FROM Student
ORDER BY Started;
```

Example: We can sort students in the Student table within each year by last name and then by first name.

```
SELECT *
FROM Student
ORDER BY Started, LastName, FirstName;
```

Inverse relations

The ordering relations we are familiar with on numbers typically come in pairs. For example, we have < and >.

In general, starting with a binary relation R(x,y) we can always consider the **inverse relation** $R^{-1}(x,y) := R(y,x)$

Note that this works on all binary relations, not just ordering relations.

Examples: What are the inverses of the following relations?

- R(x,y) = "$x < y$"
- R(x,y) = "$x \geq y$"
- R(x,y) = "x is a parent of y"
- R(x,y) = "x is a brother of y"
- R(x,y) = "x is an uncle of y"
- R(x,y) = "x is coprime to y"
- R(x,y) = "x is older than y"

Exercise: Show that R is symmetric if and only if $R^{-1} = R$.

Joining relations

Another natural operation on relations is to combine them.

Let R(x,y) and S(u,v) be two relations. The **join of R and S, written R ∘ S**, is the relation that holds between x and v if there is a y such that R(x,y) and S(y,v).

Examples:

- R(x,y) = "x is a parent of y" and S(u,v) = "u is a parent of v", then R ∘ S(x,v) is true if there is a y such that x is a parent of y and y is a parent of v. In other words, R ∘ S(x,v) is true if x is a grandparent of v.
- R(x,y) = "x is a sibling of y" and S(u,v) = "u is a parent of v". Then R ∘ S(x,v) is true if there is a y such that x is a sibling of y and y is a parent of v. In other words, R ∘ S(x,v) means that x is an uncle or aunt of v (excluding uncles or aunts by marriage).
- Suppose R(x,y) = "x = 2y" and S(u,v) = "u = 3v". Then R ∘ S(x,v) is true if there is a y such that x = 2y and y = 3v. This is true if x = 2(3v) = 6v.

Exercises: What is the join R ∘ S if R(x,y) = "x is married to y" and S(u,v) = "u is a child of v"

The join of a relation is closely related to its transitivity.

For example, "parent of parent" is "grandparent" and is not the same as "parent". Being a parent is not transitive.

But "taller than someone taller than" is the same as "taller than". Being taller than is transitive.

Joins and databases

Joins can be defined on arbitrary, not just binary, relations.

In that case we need to specify which argument of R is matched up with which argument of S.

We have **used joins before when connecting multiple tables** in a database. We join them by requiring that the foreign key be the same as the primary key.

Example: This query lists the last names of presidents of student groups.

```
SELECT LastName, SID, Name
FROM Student, StudentGroup
WHERE PresidentID = SID;
```

This is a join of Student(ln, fn, sid, cr, pr, ct, st) and StudentGroup(pid, gn, fd), joined by the condition that sid=pid.

The extensional view of relations

The examples of relations we have seen included many natural relations such as "sibling", "parent", <, etc.

These relations differ from the type we have seen in our database examples.

A relation that is described through words or mathematics is called an **intensional** relation, since it is determined by its intension or meaning.

The relations found in a database are explicitly listed (e.g. Abigail Winter took CSC 440 in the Fall 2005), and such relations are called **extensional** since they are determined by their explicit listing.

If we take an extensional view of relations, we can view them as sets. This means that we can take unions, intersections, and differences (complements) of them.

Examples:

- Let R(x,y) = "$x < y$" and S(x,y) = "$x = y$". Then $R \cup S(x,y)$ = "$x \leq y$". $R \cap S(x,y) = \emptyset$ since there are no x and y such that $x < y$ and $x = y$.
- Let R(x,y) = "x is a sister of y" and S(x,y) = "x is older than y". Then $R \cup S(x,y)$ = "x is older than y or a sister of y". $R \cap S(x,y)$ = "x is an older sister of y". $R – S(x,y)$ = "x is a younger (or same-age) sister of y".

Note that the set-theoretic operations correspond to logical operations on the relation.

- $R \cup S$ is the same as $(R \vee S)(x,y) := R(x,y) \vee S(x,y)$
- $R \cap S$ is the same as $(R \wedge S)(x,y) := R(x,y) \wedge S(x,y)$
- $R – S$ is the same as $(R \wedge \overline{S})(x,y) := R(x,y) \wedge \overline{S}(x,y)$

Representing binary relations

For binary relations there is a good alternative representation to what we have seen.

Example: Consider the set S = {Chad, Charlotte, Eva, Noelle} and the relation R(x,y) = "x is a parent of y".

We can represent this as a **table**, where the row corresponds to x and the column corresponds to y.

For example, Chad is a parent of Eva and Charlotte so in the following table the row corresponding to Chad has two true entries, one in the column labeled Eva and one in the column labeled Charlotte:

	Chad	Charlotte	Eva	Noelle
Chad	F	T	T	F
Charlotte	F	F	F	F
Eva	F	F	F	F
Noelle	F	T	T	F

Another way to represent this is to remove the labels on the row and columns (assuming that we know and remember what they mean) and replace the T/F entries with 1/0 entries to get a matrix representation of the set:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

This matrix encodes all of the information about "is a parent of" for my brother's family.

M[i,j] is the entry in row i and column j
M[i,j] = 1 if and only if the ith Settle is a parent of the jth Settle where we order the Settles as specified in the set S above.

This is called the **matrix view** of binary relations.

Exercises: Construct matrices for the following relations on the Settles ordered as in the set S above:

- "is a child of" (compare to the above – do you see a connection?)
- "has the same sex as"
- "is identical to"

- "is younger than" compared with "is older than" – what is the connection and what happens at the diagonal?

Relation properties and the matrix view

Some of the properties of relations that we covered earlier translate into visually clear properties of the matrices that represent them.

To aid our translations we need a bit of matrix background.

The **transpose** of a matrix M is written M^T and is defined by $M^T[i,j] = M[j,i]$

Example: Find the transpose of the matrix representing the parent relationships in the Settle family set.

Translations:

- A relation is **reflexive** if the diagonal of the associated matrix contains only 1s.
- A relation is **symmetric** if the associated matrix remains unchanged when transposing it, or precisely stated, $M^T = M$. This is the same as saying that $R^{-1} = R$.

Examples: Consider the transposes of the matrices produced for the set S given above under the following relations

- "is identical to"
- "has the same sex as"

Transitive closure

In the previous section we did not characterize transitivity.

Transitivity is closely related to the join of a binary relation with itself.

Example: "parent of parent" is "grandparent" and not the same as "parent" but "taller than someone taller than" is the same as "taller than"

Moving back to the set theoretic view of relations we can characterize this further.

Lemma: $R \circ R \subseteq R$ if and only if R is transitive
Proof: Assume that $R \circ R \subseteq R$. Pick an arbitrary $(x,y) \in R$ and $(y,z) \in R$. By definition of $R \circ R$, $(x,z) \in R \circ R$. But by assumption $R \circ R \subseteq R$ so that $(x,z) \in R$. This means that R is transitive.

Assume that R is transitive. Pick an arbitrary $(x,z) \in R \circ R$. By definition of $\in R \circ R$ there has to be a y such that $(x,y) \in R$ and $(y,z) \in R$. But then by transitivity of R, we know that $(x,z) \in R$. Thus we know that $R \circ R \subseteq R$.

Transitivity can be expressed in the matrix view using multiplication. Matrix multiplication is beyond the scope of this class, but you will very likely see it in other computer science classes.

What do we do if a relation isn't transitive, but we would like it to be?

Example: "is a parent of" is not transitive because your grandparents are not your parents. But we could include grandparents. But now your great grandparents are violating transitivity. So we could add them.

This is equivalent to replacing R with $R \circ R$ at each step. Repeating this over and over until we don't add any additional connections, we produce the **transitive closure** of the original relation R.

Exercise: What is the transitive closure of "is a child of"?