

Preliminaries: Sets, Relations, Functions, Booleans, and Predicates

EEC 522
Lecture #1

(Slides based on notes prepared by Prof. Paul Sivilotti, OSU)

January 21, 2009

EEC 522: SOFTWARE SYSTEMS MODELING

Set Enumeration

The simplest way to describe a set is to simply list all of its elements.

The list of elements is enclosed within braces { and }, and the elements are separated by commas.

For example,

Spectrum =
{*Violet, Indigo, Blue, Green, Yellow, Orange, Red*}

This technique is limited; describing large sets is difficult.

EEC 522: SOFTWARE SYSTEMS MODELING

2

Sets

A *set* is an unordered collection of distinct (different) elements.

Examples: the set of integers (\mathbb{Z}), the set of real numbers (\mathbb{R}), the set of National Football League teams, the set of lakes, etc.

The theory of sets is fundamental in the study of mathematics and computer science, finding applications in artificial intelligence, databases, and programming languages.

EEC 522: SOFTWARE SYSTEMS MODELING

1

Set Comprehension

Describing a set by stating properties exhibited by its members.

For example, the set comprehension

$$\{x : \mathbb{Z} \mid 0 \leq x < 5 : 2 \cdot x\}$$

denotes the set of values $2 \cdot x$ for all integers x that satisfy $0 \leq x < 5$. The enumeration of this set is

_____.

EEC 522: SOFTWARE SYSTEMS MODELING

3

Set Membership

The most basic operator with respect to sets is *set membership*, denoted by \in .

For a given set S , $e \in S$ (pronounced, “ e is in S ”) is true for every element e that is a member of S .

For all elements e that are *not* in S , the expression $\neg(e \in S)$ is true. This expression is abbreviated as $e \notin S$ (pronounced, “ e is not in S ”).

The Universe and the Empty set

The set of *all* values we consider is referred to as the *universe*, denoted by \mathbb{U} .

- The universe can be thought of as the type of every set variable in the theory.

A set that does not contain any elements is called an *empty set*. In these notes, an empty set is denoted either using \emptyset or using $\{ \}$.

- The cardinality of an empty set S is zero ($| S | = 0$).

Set Cardinality

The number of elements in a given set S is called the *cardinality* or *size* of set S .

This is denoted either using $| S |$ or $\#S$.

Subset and Superset

A set S is a *subset* of another set T if every element in S is also an element of T . This is denoted as $S \subseteq T$.

Further, if $S \neq T$ (T contains at least one element that is not in S), S is said to be a *proper subset* of T . This is denoted as $S \subset T$.

Set T is a *superset* (*proper superset*) of another set S if S is a subset (proper set) of T . Superset is denoted by operator \supseteq and proper superset is denoted by \supset .

Complement

The *complement* of a set S is the set of elements that are in the universe but not in S .

We denote the complement of a set S using $\sim S$.

By definition, $\sim \mathbb{U} = \underline{\hspace{2cm}}$ and $\sim \emptyset = \underline{\hspace{2cm}}$.

Set union, intersection, and difference

$$\{3, 5, 6\} \cup \{3, 2, 1\} = \underline{\hspace{2cm}}$$

$$\{3, 5, 6\} \cap \{3, 2, 1\} = \underline{\hspace{2cm}}$$

$$\{3, 5, 6\} - \{3, 2, 1\} = \underline{\hspace{2cm}}$$

From the definition of the complement, we see that $\sim S = \mathbb{U} - S$.

Set union, intersection, and difference

The *union* of two sets S and T , denoted as $S \cup T$, is the set of all elements in S or T , or both.

The *intersection* of two sets S and T , denoted as $S \cap T$, is the set of all elements that in both S and T .

The set *difference* of S and T , denoted as $S - T$, is the set of all elements that are in S but not in T .

Disjoint sets and Multi-sets

Sets S and T are *disjoint* if they have no elements in common.

$$S \cap T = \emptyset.$$

A *multi-set* is a set that may contain an arbitrary number of duplicates of each element.

Example: $\{2, 3, 4, 4, 5\}$.

Power set

The *power set* of a set S denoted as $\mathcal{P}.S$ is the set of all subsets of S .

The power set of S is also denoted as 2^S . For example,

$$\mathcal{P}.\{1, 2, 3\} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

Cross product

The *cross product* or *Cartesian product* $S \times T$ of two sets S and T is the set of all pairs $\langle b, c \rangle$ such that b is in S and c is in T .

Examples:

$\mathbb{Z} \times \mathbb{Z}$ denotes all integral points in a plane.

$\mathbb{R} \times \mathbb{R}$ denotes all points in the plane.

$\{2, 5\} \times \{1, 2, 3\}$ is the set _____.

Tuples and pairs

An *n-tuple* is a sequence of length n : it is an ordered list of values (without names).

Example: $\langle 1, 1, 2, 3, 5 \rangle$

For two expressions b and c , the 2-tuple $\langle b, c \rangle$ is called an *ordered pair*, or simply a *pair*.

Relation

A *relation* on a cross product $B_1 \times \cdots \times B_n$ is a subset of $B_1 \times \cdots \times B_n$.

- Thus, a relation is a set of n -tuples (for some fixed n).

A *binary relation* over $B \times C$ is a subset of $B \times C$.

- Thus, a binary relation is a set of pairs.

Relation

A relation r is simply a *mapping* between elements of two sets.

The relation maps elements in the *domain* (denoted as $\mathcal{D}.r$) to elements in the *range* (denoted as $\mathcal{R}.r$).

Example: the relation *parent* is the set of pairs $\langle b, c \rangle$ where b is a parent of c . Domain is the set of all adults, and range the set of all children.

$$\textit{parent} : \textit{Adults} \rightarrow \textit{Children}$$

Function

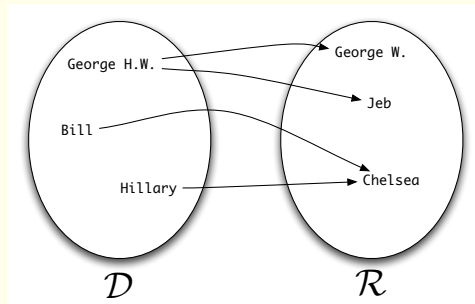
A *function* is a special kind of relation.

A function maps each element in the domain to *exactly one element* in the range.

More formally, we can regard a binary relation on $B \times C$ as a function f that contains all pair $\langle b, c \rangle$, such that $f.b = c$ and for every value of b , there is exactly one value of c .

Relation: Graphical Representation

The relation is represented by the set of arrows from elements in the domain (\mathcal{D}) to elements in the range (\mathcal{R}).



Function: Example

$$\textit{sqrt} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$$

This describes a function that maps elements in \mathbb{R}^+ to \mathbb{R}^+ . The domain and the range of this function are the same:

$$\mathcal{D}.\textit{sqrt} = \mathcal{R}.\textit{sqrt} = \mathbb{R}^+$$

Function application

The operator that we use for *function application* is the dot operator (“.”).

For example, we would write the *sqrt* function to determine the square root of 25 as:

sqrt.25

Total and Partial Functions

A function f on $B \times C$ is *total* if $B = \mathcal{D}.f$, i.e., the function is defined for *every* member of the domain set B .

Any function that is not total is referred to as a *partial* function.

Currying

Currying is a technique that is used to transform a function with multiple arguments into a function that takes a single argument.

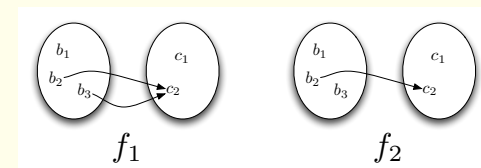
For example, the function $f(a_1, \dots, a_n)$ would be written as $f.a_1. \dots .a_n$.

We do this primarily for simplicity.

Injective (One-to-one) Functions

A function f is called *injective* if it is “range-unique” — every element in the domain maps to a unique element in the range.

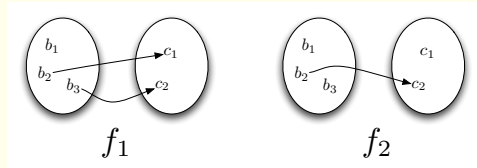
Injective functions are also called *one-to-one* functions.



Surjective (Onto) Functions

A *surjective* function f is one such that there is at least one domain element that maps to *every* element in the range; no range element is left unmapped.

Surjective functions are also called *onto* function.



Boolean

Boolean refers to a set with exactly two elements: **true** and **false**.

The set is commonly denoted using \mathbb{B} .

Several other symbols have been used to refer to the members of this set, such as T/F, 1/0, \top/\perp .

Bijjective Functions

A *bijjective* function f is one that is both injective and surjective (one-to-one and onto).

Some Boolean Operators

- conjunction (\wedge , pronounced “and”)
- disjunction (\vee , pronounced “or”)
- negation (\neg , pronounced “not”)
- equivalence (\equiv , pronounced “equivalents”)
- implication (\Rightarrow , pronounced “implies”)
- etc...

Predicate

A *predicate* P is any function whose range is boolean (\mathbb{B}).

$$P : \mathcal{D} \rightarrow \mathbb{B}, \text{ for any domain } \mathcal{D}$$

For example, $odd : \mathbb{Z} \rightarrow \mathbb{B}$ is a predicate that is **true** exactly when it is applied to an odd number, and is **false** otherwise.

($odd.3$ is **true**, and $odd.14$ is **false**).

Lifting: Example

Consider a predicate $prime : \mathbb{Z} \rightarrow \mathbb{B}$ that is **true** when applied to a prime number, along with the predicate odd . The expression $odd \wedge prime$ is a predicate! Its signature look like this:

$$odd \wedge prime : \mathbb{Z} \rightarrow \mathbb{B}$$

Therefore, it can be applied to elements of the domain (\mathbb{Z}), and results in a boolean:

$$(odd \wedge prime).13 \equiv \mathbf{true}$$

Lifting

All operators on booleans (\wedge , \vee , \equiv , ...) can also be applied to predicates.

Although the symbols used are the same, they are technically *different* operators, since their signatures are different:

$$\wedge : \text{boolean} \times \text{boolean} \rightarrow \text{boolean}$$

$$\wedge : \text{predicate} \times \text{predicate} \rightarrow \text{_____}$$

Lifting

Such overloading of operator symbols is known as *lifting*.

Lifting can not only apply to operators, it can also be applied to the constants **true** and **false**.

That is, they can be lifted to be predicates, instead of boolean values.

The predicate **true** means "**true everywhere**", and the predicate **false** means "**false everywhere**".

Everywhere brackets

What does *everywhere* mean?

We use the *everywhere brackets* operator (denoted by a pair of square brackets, []) to say that a predicate is **true** for every element in its domain.

Everywhere brackets

But can we ask the question: “do *odd* and *prime* mean the same thing?” or “Is being odd the same as being prime?”.

Precisely, “For every number, is being odd the same as being prime?”.

This statement can be written as:

$$[odd \equiv prime]$$

This expression is now a _____, whose value is _____.

Everywhere brackets

Consider the expression:

$$odd \equiv prime$$

This expression is a _____. We can evaluate this at various points in the domain (\mathbb{Z}):

$$(odd \equiv prime).13 \text{ is } \underline{\hspace{2cm}}$$

$$(odd \equiv prime).142 \text{ is } \underline{\hspace{2cm}}$$

$$(odd \equiv prime).2 \text{ is } \underline{\hspace{2cm}}$$

Everywhere brackets

In fact, we can enclose any predicate with everywhere brackets, resulting in a _____.

$$[odd] \text{ is } \underline{\hspace{2cm}}$$

$$[odd \vee prime] \text{ is } \underline{\hspace{2cm}}$$

$$[odd \vee \neg odd] \text{ is } \underline{\hspace{2cm}}$$