CS:5810
Formal Methods in Software Engineering

Sets and Relations
These Notes

• review the concepts of sets and relations required for working with the Alloy language

• focus on the kind of set operation and definitions used in specifications

• give some small examples of how we will use sets in specifications
Set

• Collection of distinct objects
• Each set’s objects are drawn from a larger domain of objects all of which have the same type --- sets are homogeneous
• Examples:

\{2,4,5,6,…\} set of integers
\{red, yellow, blue\} set of colors
\{true, false\} set of boolean values
\{red, true, 2\} for us, not a set!
Value of a Set

• Is the collection of its members

• Two sets $A$ and $B$ are equal iff
  – every member of $A$ is a member of $B$
  – every member of $B$ is a member of $A$

• $x \in S$ denotes “$x$ is a member of $S$”

• $\emptyset$ denotes the empty set
Defining Sets

• We can define a set by *enumeration*
  – PrimaryColors == \{red, yellow, blue\}
  – Boolean == \{true, false\}
  – Evens == \{..., -4, -2, 0, 2, 4, ...

• This works fine for finite sets, but
  – what do we mean by “...” ?
  – remember, we want to be precise
Defining Sets

• We can define a set by *comprehension*, that is, by describing a property that its elements must share.

• Notation: \[ \{ x : D \mid P(x) \} \]
  
  – Form a new set of elements drawn from domain \( D \) by including exactly the elements that satisfy predicate (i.e., Boolean function) \( P \).

• Examples:

  \[ \{ x : \mathbb{N} \mid x < 10 \} \quad \text{Naturals less than 10} \]

  \[ \{ x : \mathbb{Z} \mid (\exists y : \mathbb{Z} \mid x = 2y) \} \quad \text{Even integers} \]

  \[ \{ x : \mathbb{N} \mid x > x \} \quad \text{Empty set of natural numbers} \]
Cardinality

• The *cardinality* (#) of a finite set is the number of its elements

• Examples:
  – # \{red, yellow, blue\} = 3
  – # \{1, 23\} = 2
  – # \mathbb{Z} = ?

• Cardinalities are defined for infinite sets too, but we’ll be most concerned with the cardinality of finite sets
Set Operations

- **Union** (X, Y sets over domain D):
  
  $- X \cup Y \equiv \{ e : D \mid e \in X \text{ or } e \in Y \}$

  $- \{\text{red}\} \cup \{\text{blue}\} = \{\text{red, blue}\}$

- **Intersection**
  
  $- X \cap Y \equiv \{ e : D \mid e \in X \text{ and } e \in Y \}$

  $- \{\text{red, blue}\} \cap \{\text{blue, yellow}\} = \{\text{blue}\}$

- **Difference**
  
  $- X \setminus Y \equiv \{ e : D \mid e \in X \text{ and } e \notin Y \}$

  $- \{\text{red, yellow, blue}\} \setminus \{\text{blue, yellow}\} = \{\text{red}\}$
Subsets

• A subset holds elements drawn from another set
  – $X \subseteq Y$ iff every element of $X$ is in $Y$
  – $\{1, 7, 17, 24\} \subseteq \mathbb{Z}$

• A proper subset is a non-equal subset

• Another view of set equality
  – $A = B$ iff ($A \subseteq B$ and $B \subseteq A$)
Power Sets

• The power set of set S (denoted $Pow(S)$) is the set of all subsets of S, i.e.,

$$Pow(S) \equiv \{e \mid e \subseteq S\}$$

• Example:
  - $Pow\left(\{a, b, c\}\right) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

Note: for any S, $\emptyset \subseteq S$ and thus $\emptyset \in Pow(S)$
Exercises

• These slides include questions that you should be able to solve at this point

• They may require you to think some

• You should spend some effort in solving them – ... and may in fact appear on exams
Exercises

• Specifying using comprehension notation
  – Odd positive integers
  – The squares of integers, i.e. \{1,4,9,16,...\}

• Express the following logic properties on sets without using the \# operator
  – Set has at least one element
  – Set has no elements
  – Set has exactly one element
  – Set has at least two elements
  – Set has exactly two elements
Set Partitioning

- Sets are *disjoint* if they share no elements
- Often when modeling, we will take some set $S$ and divide its members into disjoint subsets called *blocks* or *parts*
- We call this division a *partition*
- Each member of $S$ belongs to exactly one block of the partition
Example

Model residential scenarios

- Basic domains: *Person, Residence*

- Partitions:
  - Partition *Person* into *Child, Student, Adult*
  - Partition *Residence* into *Home, DormRoom, Apartment*
Exercises

• Express the following properties of pairs of sets
  – Two sets are disjoint
  – Two sets form a partitioning of a third set
Expressing Relationships

• It’s useful to be able to refer to structured values
  – a group of values that are bound together
  – e.g., struct, record, object fields
• Alloy is a calculus of relations
• All of our Alloy models will be built using relations (sets of tuples)
• ... but first some basic definitions
Product

• Given two sets $A$ and $B$, the product of $A$ and $B$, usually denoted $A \times B$, is the set of all possible pairs $(a, b)$ where $a \in A$ and $b \in B$

$$A \times B \equiv \{(a, b) \mid a \in A, b \in B\}$$

• Example: PrimaryColor $\times$ Boolean:

\[
\{(\text{red}, \text{true}), (\text{red}, \text{false}), (\text{blue}, \text{true}), (\text{blue}, \text{false}), (\text{yellow}, \text{true}), (\text{yellow}, \text{false})\}
\]
Relation

• A binary relation $R$ between $A$ and $B$ is an element of $\text{Pow}(A \times B)$, i.e., $R \subseteq A \times B$

• Examples:
  – Parent : Person x Person
    • Parent == {(John, Autumn), (John, Sam)}
  – Square : $Z \times N$
    • Square == {(1,1), (-1,1), (-2,4)}
  – ClassGrades : Person x {A, B, C, D, F}
    • ClassGrades == {(Todd,A), (Jane,B)}
Relation

• A ternary relation $R$ between $A$, $B$ and $C$ is an element of $\text{Pow}(A \times B \times C)$

• Example:
  – FavoriteBeer : Person x Beer x Price
    • FavoriteBeer == {(John, Miller, $2), (Ted, Heineken, $4), (Steve, Miller, $2)}

• N-ary relations with n>3 are defined analogously (n is the arity of the relation)
Binary Relations

• The set of first elements is the definition domain of the relation
  – \( \text{domain (Parent)} = \{ \text{John} \} \) NOT Person!

• The set of second elements is the image of the relation
  – \( \text{image (Square)} = \{ 1, 4 \} \) NOT \( \mathbb{N} \)

• How about \( \{(1,\text{blue}), (2,\text{blue}), (1,\text{red})\} \)
  – domain? image?
Common Relation Structures

- **One-to-Many**
  - One
  - "Many" (two)

- **Many-to-One**
  - Many
  - One

- **One-to-One**
  - One
  - One

- **Many-to-Many**
  - Many
  - Many
Functions

• A *function* is a relation $F$ of arity $n+1$ containing no two distinct tuples with the same first $n$ elements,
  
  – i.e., for $n = 1$,
  
  $$\forall (a_1, b_1) \in F, \forall (a_2, b_2) \in F, (a_1 = a_2 \Rightarrow b_1 = b_2)$$

• Examples:
  
  – $\{(2, \text{red}), (3, \text{blue}), (5, \text{red})\}$
  
  – $\{(4, 2), (6, 3), (8, 4)\}$

• Instead of $F: A_1 \times A_2 \times \ldots \times A_n \times B$
  
  we write $F: A_1 \times A_2 \ldots \times A_n \rightarrow B$
Exercises

• Which of the following are functions?

  – Parent == {(John, Autumn), (John, Sam)}

  – Square == {(1, 1), (-1, 1), (-2, 4)}

  – ClassGrades == {(Todd, A), (Virg, B)}
In other words, a function is a relation that is X-to-one.
Special Kinds of Functions

• Consider a function $f$ from $S$ to $T$

• $f$ is *total* if defined for all values of $S$

• $f$ is *partial* if undefined for some values of $S$

• Examples
  
  – Squares : $\mathbb{Z} \rightarrow \mathbb{N}$, Squares = {..., (-1,1), (0,0), (1, 1), (2,4), ...}
  
  – Abs = \{(x, y) : \mathbb{Z} \times \mathbb{N} \mid (x < 0 \text{ and } y = -x) \text{ or } (x \geq 0 \text{ and } y = x)\}
Function Structures

Total Function

Partial Function

Note: the empty relation is a partial function
Special Kinds of Functions

A function $f: S \rightarrow T$ is

- **injective (one-to-one)** if no image element is associated with multiple domain elements

- **surjective (onto)** if its image is $T$

- **bijective** if it is both injective and surjective

We’ll see that these come up frequently

- can be used to define properties concisely
Function Structures

Injective Function

Surjective Function
Exercises

• What kind of function/relation is Abs?
  - Abs = {(x, y) : \(Z \times \mathbb{N}\) | (x < 0 and y = -x) or (x ≥ 0 and y = x)}

• How about Squares?
  - Squares : \(Z \times \mathbb{N}\), Squares = {(x, y) : \(Z \times \mathbb{N}\) | y = x* x}
Special Cases

Relations

Partial Functions

Surjective

Bijective

Injective

Total
Functions as Sets

• Functions are relations and hence sets

• We can apply all of the usual operators
  – ClassGrades == {(Todd, A), (Jane, B)}
  – #(ClassGrades U {(Matt, C)}) = 3
Exercises

• In the following if an operator fails to preserve a property give an example
• What operators preserve function-ness?
  – ∩?
  – ∪?
  – \?
• What operators preserve surjectivity?
• What operators preserve injectivity?
Relation Composition

• Use two relations to produce a new one
  – map domain of first to image of second
  – Given $s: A \times B$ and $r: B \times C$ then $s;r : A \times C$

\[
s;r \equiv \{(a,c) \mid (a,b) \in s \text{ and } (b,c) \in r\}
\]

• For example
  – $s == \{(\text{red},1), (\text{blue},2)\}$
  – $r == \{(1,2), (2,4), (3,6)\}$
  – $s;r = \{(\text{red},2), (\text{blue},4)\}$
Relation Closure

- Intuitively, the closure of a relation \( r: S \times S \), written \( r^+ \), is what you get when you keep navigating through \( r \) until you can’t go any farther.

\[
r^+ \equiv r \cup (r;r) \cup (r;r;r) \cup …
\]

- For example
  - GrandParent == Parent;Parent
  - Ancestor == Parent^+
Relation Transpose

• Intuitively, the transpose of a relation \( r: S \times T \), written \( \sim r \), is what you get when you reverse all the pairs in \( r \).

\[
\sim r \equiv \{(b,a) \mid (a,b) \in r\}
\]

• For example
  – ChildOf == \( \sim \)Parent
  – DescendantOf == (\( \sim \)Parent)\(^+\)
Exercises

• In the following if an operator fails to preserve a property give an example

• What properties, i.e., function-ness, onto-ness, 1-1-ness, by the relation operators?
  – composition (;)
  – closure (+)
  – transpose (~)
Acknowledgements

Some of these slides are adapted from

David Garlan’s slides from Lecture 3 of his course of Software Models entitled “Sets, Relations, and Functions” (http://www.cs.cmu.edu/afs/cs/academic/class/15671-f97/www/ )