Case Study: Abstraction of a Birthday Book

EECS3311 A & E: Software Design
Fall 2020

CHEN-WEI WANG

Learning Objectives

Upon completing this lecture, you are expected to understand:

1. Asserting Set Equality in Postconditions (Exercise)
2. The basics of discrete math (Self-Guided Study)
   `FUN` is a `REL`, but not vice versa.
3. Creating a ***mathematical abstraction*** for a birthday book
4. Using commands and queries from two `mathmodels` classes: `REL` and `FUN`

Math Review: Set Definitions and Membership

- A *set* is a collection of objects.
  - Objects in a set are called its *elements* or *members*.
  - *Order* in which elements are arranged does not matter.
  - An element can appear *at most once* in the set.
- We may define a set using:
  - *Set Enumeration*: Explicitly list all members in a set.
    e.g., $\{1, 3, 5, 7, 9\}$
  - *Set Comprehension*: Implicitly specify the condition that all members satisfy.
    e.g., $\{x \mid 1 \leq x \leq 10 \wedge x \texttt{ is an odd number}\}$
- An empty set (denoted as $\{\}$ or $\varnothing$) has no members.
- We may check if an element is a *member* of a set:
  e.g., $5 \in \{1, 3, 5, 7, 9\}$ [*true*]
  e.g., $4 \notin \{x \mid x \leq 1 \leq 10, x \texttt{ is an odd number}\}$ [*true*]
- The number of elements in a set is called its *cardinality*.
  e.g., $|\varnothing| = 0$, $|\{x \mid x \leq 1 \leq 10, x \texttt{ is an odd number}\}| = 5$

Math Review: Set Relations

Given two sets S_1 and S_2:

- S_1 is a *subset* of S_2 if every member of S_1 is a member of S_2.

$$S_1 \subseteq S_2 \iff (\forall x \bullet x \in S_1 \Rightarrow x \in S_2)$$

- S_1 and S_2 are *equal* iff they are the subset of each other.

$$S_1 = S_2 \iff S_1 \subseteq S_2 \wedge S_2 \subseteq S_1$$

- S_1 is a *proper subset* of S_2 if it is a strictly smaller subset.

$$S_1 \subset S_2 \iff S_1 \subseteq S_2 \wedge |S1| < |S2|$$

Math Review: Set Operations

Given two sets S_1 and S_2:

- *Union* of S_1 and S_2 is a set whose members are in either.

$$S_1 \cup S_2 = \{x \mid x \in S_1 \lor x \in S_2\}$$

- *Intersection* of S_1 and S_2 is a set whose members are in both.

$$S_1 \cap S_2 = \{x \mid x \in S_1 \land x \in S_2\}$$

- *Difference* of S_1 and S_2 is a set whose members are in S_1 but not S_2.

$$S_1 \setminus S_2 = \{x \mid x \in S_1 \land x \notin S_2\}$$

Math Review: Power Sets

The *power set* of a set S is a *set* of all S' *subsets*.

$$\mathbb{P}(S) = \{s \mid s \subseteq S\}$$

The power set contains subsets of *cardinalities* 0, 1, 2, ..., $|S|$.
e.g., $\mathbb{P}(\{1,2,3\})$ is a set of sets, where each member set s has cardinality 0, 1, 2, or 3:

$$\left\{ \begin{array}{l} \varnothing, \\ \{1\},\ \{2\},\ \{3\}, \\ \{1,2\},\ \{2,3\},\ \{3,1\}, \\ \{1,2,3\} \end{array} \right\}$$

Math Review: Set of Tuples

Given n sets $S_1, S_2, \dots, S_n$, a *cross product* of theses sets is a set of n-tuples.
Each *n-tuple* $(e_1, e_2, \dots, e_n)$ contains n elements, each of which a member of the corresponding set.

$$S_1 \times S_2 \times \cdots \times S_n = \{(e_1, e_2, \dots, e_n) \mid e_i \in S_i \land 1 \le i \le n\}$$

e.g., $\{a, b\} \times \{2, 4\} \times \{\$, \&\}$ is a set of triples:

$$\begin{array}{rl} & \{a, b\} \times \{2, 4\} \times \{\$, \&\} \\ = & \{\ (e_1, e_2, e_3) \mid e_1 \in \{a, b\} \land e_2 \in \{2, 4\} \land e_3 \in \{\$, \&\}\ \} \\ = & \left\{ \begin{array}{l} (a, 2, \$), (a, 2, \&), (a, 4, \$), (a, 4, \&), \\ (b, 2, \$), (b, 2, \&), (b, 4, \$), (b, 4, \&) \end{array} \right\} \end{array}$$

Math Models: Relations (1)

- A *relation* is a collection of mappings, each being an *ordered pair* that maps a member of set S to a member of set T.
  e.g., Say $S = \{1, 2, 3\}$ and $T = \{a, b\}$
  - $\varnothing$ is an empty relation.
  - $S \times T$ is a relation (say r_1) that maps from each member of S to each member in T: $\{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$
  - $\{(x, y) : S \times T \mid x \neq 1\}$ is a relation (say r_2) that maps only some members in S to every member in T: $\{(2, a), (2, b), (3, a), (3, b)\}$.
- Given a relation r:
  - *Domain* of r is the set of S members that r maps from.

$$\mathrm{dom}(r) = \{s : S \mid (\exists t \bullet (s, t) \in r)\}$$

e.g., $\mathrm{dom}(r_1) = \{1, 2, 3\}$, $\mathrm{dom}(r_2) = \{2, 3\}$

  - *Range* of r is the set of T members that r maps to.

$$\mathrm{ran}(r) = \{t : T \mid (\exists s \bullet (s, t) \in r)\}$$

e.g., $\mathrm{ran}(r_1) = \{a, b\} = \mathrm{ran}(r_2)$

Math Models: Relations (2)

- We use the power set operator to express the set of *all possible relations* on S and T:

$$\mathbb{P}(S \times T)$$

- To declare a relation variable r, we use the colon (`:`) symbol to mean *set membership*:

$$r : \mathbb{P}(S \times T)$$

- Or alternatively, we write:

$$r : S \leftrightarrow T$$

where the set $S \leftrightarrow T$ is synonymous to the set $\mathbb{P}(S \times T)$

Math Models: Relations (3.1)

Say $r = \{(a,1),(b,2),(c,3),(a,4),(b,5),(c,6),(d,1),(e,2),(f,3)\}$

- r.***domain*** : set of first-elements from r
  - r.**domain** $= \{ \ d \mid (d,r) \in r \ \}$
  - e.g., r.**domain** $= \{a,b,c,d,e,f\}$
- r.***range*** : set of second-elements from r
  - r.**range** $= \{ \ r \mid (d,r) \in r \ \}$
  - e.g., r.**range** $= \{1,2,3,4,5,6\}$
- r.***inverse*** : a relation like r except elements are in reverse order
  - r.**inverse** $= \{ \ (r,d) \mid (d,r) \in r \ \}$
  - e.g., r.**inverse** $= \{(1,a),(2,b),(3,c),(4,a),(5,b),(6,c),(1,d),(2,e),(3,f)\}$

Math Models: Relations (3.2)

Say $r = \{(a,1),(b,2),(c,3),(a,4),(b,5),(c,6),(d,1),(e,2),(f,3)\}$

- r.***domain_restricted***(ds) : sub-relation of r with domain ds.
  - r.**domain_restricted**(ds) $= \{ \ (d,r) \mid (d,r) \in r \land d \in ds \ \}$
  - e.g., r.**domain_restricted**({a, b}) $= \{(\mathbf{a},1),(\mathbf{b},2),(\mathbf{a},4),(\mathbf{b},5)\}$
- r.***domain_subtracted***(ds) : sub-relation of r with domain <u>not</u> ds.
  - r.**domain_subtracted**(ds) $= \{ \ (d,r) \mid (d,r) \in r \land d \notin ds \ \}$
  - e.g., r.**domain_subtracted**({a, b}) $=$ $\{(\mathbf{c},3),(\mathbf{c},6),(\mathbf{d},1),(\mathbf{e},2),(\mathbf{f},3)\}$
- r.***range_restricted***(rs) : sub-relation of r with range rs.
  - r.**range_restricted**(rs) $= \{ \ (d,r) \mid (d,r) \in r \land r \in rs \ \}$
  - e.g., r.**range_restricted**({1, 2}) $= \{(a,\mathbf{1}),(b,\mathbf{2}),(d,\mathbf{1}),(e,\mathbf{2})\}$
- r.***range_subtracted***(ds) : sub-relation of r with range <u>not</u> ds.
  - r.**range_subtracted**(rs) $= \{ \ (d,r) \mid (d,r) \in r \land r \notin rs \ \}$
  - e.g., r.**range_subtracted**({1, 2}) $=$ $\{\{(c,\mathbf{3}),(a,\mathbf{4}),(b,\mathbf{5}),(c,\mathbf{6}),(f,3)\}\}$

Math Models: Relations (3.3)

Say $r = \{(a,1),(b,2),(c,3),(a,4),(b,5),(c,6),(d,1),(e,2),(f,3)\}$

- r.***overridden***(t) : a relation which agrees on r outside domain of t.*domain*, and agrees on t within domain of t.*domain*
  - r.**overridden**(t) $= t \cup r.$**domain_subtracted**$(t.$**domain**$)$
  -

$$
\begin{aligned}
& r.\textbf{overridden}(\underbrace{\{(a,3),(c,4)\}}_{t}) \\
= \ & \underbrace{\{(a,3),(c,4)\}}_{t} \cup \underbrace{\{(b,2),(b,5),(d,1),(e,2),(f,3)\}}_{r.\textbf{domain\_subtracted}(\underbrace{t.\textbf{domain}}_{\{a,c\}})} \\
= \ & \{(a,3),(c,4),(b,2),(b,5),(d,1),(e,2),(f,3)\}
\end{aligned}
$$

Math Review: Functions (1)

A ==*function*== f on sets S and T is a *specialized form* of relation: it is forbidden for a member of S to map to more than one members of T.

$$\forall s : S;\ t_1 : T;\ t_2 : T \bullet (s, t_1) \in f \land (s, t_2) \in f \Rightarrow t_1 = t_2$$

e.g., Say $S = \{1, 2, 3\}$ and $T = \{a, b\}$, which of the following relations are also functions?

- $S \times T$ [No]
- $(S \times T) - \{(x, y) \mid (x, y) \in S \times T \land x = 1\}$ [No]
- $\{(1, a), (2, b), (3, a)\}$ [Yes]
- $\{(1, a), (2, b)\}$ [Yes]

Math Review: Functions (2)

- We use *set comprehension* to express the set of all possible functions on S and T as those relations that satisfy the ==*functional property*== :

$$\{r : S \leftrightarrow T \mid (\forall s : S;\ t_1 : T;\ t_2 : T \bullet (s, t_1) \in r \land (s, t_2) \in r \Rightarrow t_1 = t_2)\}$$

- This set (of possible functions) is a subset of the set (of possible relations): $\mathbb{P}(S \times T)$ and $S \leftrightarrow T$.
- We abbreviate this set of possible functions as $S \to T$ and use it to declare a function variable f:

$$f : S \to T$$

Math Review: Functions (3.1)

Given a function $f : S \to T$:

- f is *injective* (or an injection) if f does not map two members of S to the same member of T.

$$f \texttt{ is injective} \iff (\forall s_1 : S;\ s_2 : S;\ t : T \bullet (s_1, t) \in r \land (s_2, t) \in r \Rightarrow s_1 = s_2)$$

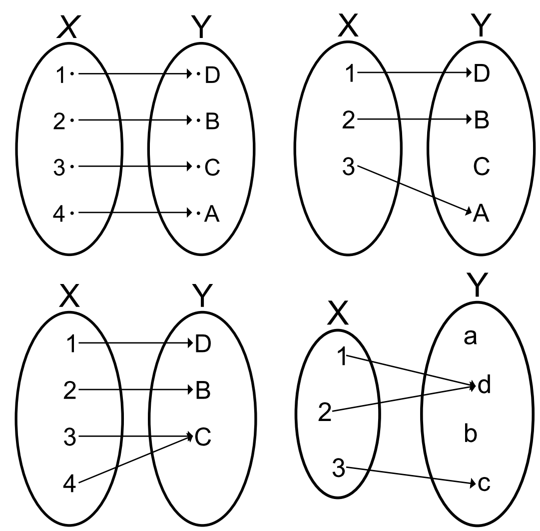
  e.g., Considering an array as a function from integers to objects, being injective means that the array does not contain any duplicates.

- f is *surjective* (or a surjection) if f maps to all members of T.

$$f \texttt{ is surjective} \iff \mathrm{ran}(f) = T$$

- f is *bijective* (or a bijection) if f is both injective and surjective.

Math Review: Functions (3.2)



Math Models: Command-Query Separation

Command	*Query*
domain_restrict	domain_restrict**ed**
domain_restrict_by	domain_restrict**ed**_by
domain_subtract	domain_subtract**ed**
domain_subtract_by	domain_subtract**ed**_by
range_restrict	range_restrict**ed**
range_restrict_by	range_restrict**ed**_by
range_subtract	range_subtract**ed**
range_subtract_by	range_subtract**ed**_by
override	overrid**den**
override_by	overrid**den**_by

Say $r = \{(a,1),(b,2),(c,3),(a,4),(b,5),(c,6),(d,1),(e,2),(f,3)\}$

- ***Commands*** modify the context relation objects.

  `r.domain_restrict({a})` changes r to $\{(a,1),(a,4)\}$
- ***Queries*** return new relations without modifying context objects.

  `r.domain_restricted({a})` returns $\{(a,1),(a,4)\}$ with r untouched

Math Models: Example Test

```
test_rel: BOOLEAN
 local
  r, t: REL[STRING, INTEGER]
  ds: SET[STRING]
 do
  create r.make_from_tuple_array (
   <<["a", 1], ["b", 2], ["c", 3],
     ["a", 4], ["b", 5], ["c", 6],
     ["d", 1], ["e", 2], ["f", 3]>>)
  create ds.make_from_array (<<"a">>)
  -- r is not changed by the query 'domain_subtracted'
  t := r.domain_subtracted (ds)
  Result :=
   t /~ r and not t.domain.has ("a") and r.domain.has ("a")
  check Result end
  -- r is changed by the command 'domain_subtract'
  r.domain_subtract (ds)
  Result :=
   t ~ r and not t.domain.has ("a") and not r.domain.has ("a")
 end
```

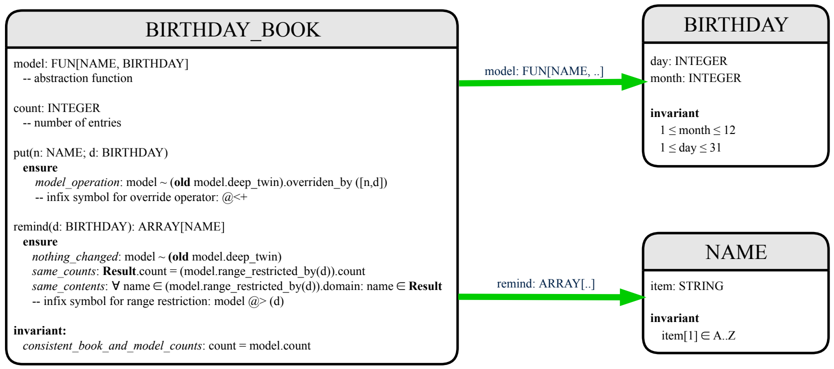

Case Study: A Birthday Book

- A birthday book stores a collection of entries, where each entry is a pair of a person's name and their birthday.
- No two entries stored in the book are allowed to have the same name.
- Each birthday is characterized by a month and a day.
- A birthday book is first created to contain an empty collection of entires.
- Given a birthday book, we may:
  - Inquire about the number of entries currently stored in the book
  - Add a new entry by supplying its name and the associated birthday
  - Remove the entry associated with a particular person
  - Find the birthday of a particular person
  - Get a reminder list of names of people who share a given birthday

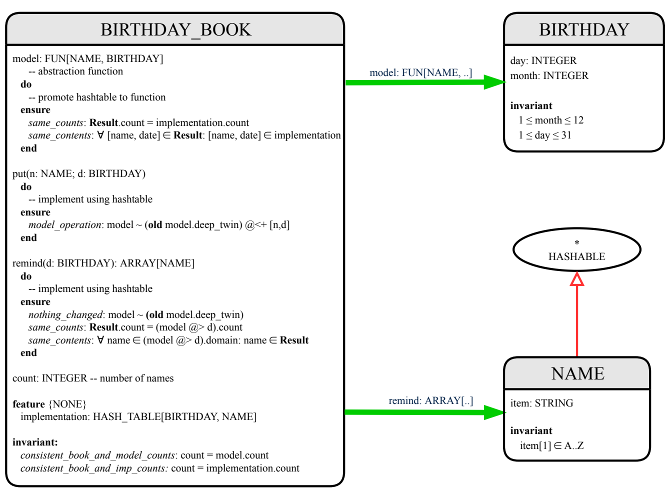
Birthday Book: Decisions

- ***Design*** Decision
  - Classes
  - Client Supplier vs. Inheritance
  - Mathematical Model? [e.g., `REL` or `FUN`]
  - Contracts
- ***Implementation*** Decision
  - Two linear structures (e.g., arrays, lists) [$O(n)$]
  - A balanced search tree (e.g., AVL tree) [$O(log \cdot n)$]
  - A hash table [$O(1)$]
- Implement an *abstraction function* that maps implementation to the math model.

Birthday Book: Design



Birthday Book: Implementation



Beyond this lecture ...

- Familiarize yourself with the features of class `REL`, `FUN`, and `SET`.
- **Exercise**:
  - Consider an alternative implementation using two linear structures (e.g., here in Java).
  - Implement the design of birthday book covered in lectures.
  - Create another `LINEAR_BIRTHDAY_BOOK` class and modify the implementation of abstraction function accordingly.
    Do all contracts still pass? What should change? What remain unchanged?

Index (1)

Index (2)