Lecture 12
Z Techniques

David Garlan

Carnegie Mellon University

Models of Software Systems
Fall 2016

The Story Thus Far

- We have seen a brief introduction to Z
- Main features
  - use of schemas to represent state space
  - use of a schema to represent initial states
  - use of schemas to describe operations
    - delta (∆) and xi (Ξ) conventions
    - ? and ! conventions for input and output
  - use of schema calculus to build up bigger operations out of smaller ones
    - separately specify normal and error behaviors and then combine later

This Lecture

- Relating Z to state machines
- Techniques for using Z effectively
- Example: Library
  - > Static versus dynamic information
  - > Access control through schema composition
  - > Undefined values
  - > Framing schema
  - > “Creating” new state
  - > Promotion

Z Style Reminders

- Always include prose with your Z specifications
  - explain the meaning of variables
  - motivate the invariants
- Don’t forget to account for all variables
  - if you don’t say they are unchanged, they can have arbitrary values
- Define constants as global axiomatic definitions

Relating Birthday Book to State Machines

S = {known: ℙ NAME; birthday: NAME ⇸ DATE |
known = dom(birthday)}

I = {known: ℙ NAME; birthday: NAME ⇸ DATE |
known = ∅ ∧ birthday = ∅}

A = {AddBirthday(date?:DATE, name?:NAME),
FindBirthday(date?:DATE)/name!(NAME),
Remind(date?:DATE)/names!(ℙ Name)}

δ = ...

AddBirthday

AddBirthday (date?:DATE, name?:NAME)

<u>pre</u> name? $\notin$ known

<u>post</u> birthday' = birthday $\cup$ { name? $\mapsto$ date? }

FindBirthday (date?:DATE)/name!(NAME)

<u>pre</u> name? $\in$ known

<u>post</u>

birthday' = birthday $\wedge$

known’ = known $\wedge$

name! = birthday(date?)

RAddBirthday

RAddBirthday (date?:DATE, name?:NAME)
report!(REPORT)

pre true

post

$(name? \notin known) \land$

$birthday' = birthday \cup \{ name? \mapsto date? \} \land$

$report = ok)$

$\lor$

$(name? \in known \land$

$birthday' = birthday \land$

$known' = known \land$

$report = alreadyknown)$

Library Example

Consider a simple library that supports the following operations:

1. Check out a copy of a book
2. Return a copy of a book
3. Add a copy of a book to the library
4. Remove a copy of a book from the library
5. Find out the list of books checked out by a particular borrower
6. Find out what borrower last checked out a particular copy of a book
7. Get a list of books written by a particular author or on a particular subject

There are staff users and ordinary borrowers. Only staff users can carry out 1-6, except that borrowers can execute 5 for themselves.

A user can check out a maximum of MAX books.

Books and Copies

[AUTHOR, TITLE, SUBJECT, COPYID]

Book

author: AUTHOR

title: TITLE

subjects: 𝔽 SUBJECT

Copy

Book

id: COPYID

Recall that 𝔽 is similar to ℙ but represents Finite Sets

Equivalent Definition

Copy

author: AUTHOR

title: TITLE

subjects: $\mathbb{F}$ SUBJECT

id: COPYID

Transient Book Information

[PERSON]

STATUS ::= out | in

```
┌─ Data ──────────────────────────
│ lastuser: PERSON
│ status: STATUS
└─────────────────────────────────
```

```
│ MAX: ℕ
│ nobody: PERSON
```

REAL_PERSON == PERSON \ {nobody}

Library

Library

books: $\mathbb{F}$ Copy

records: Copy ⇸ Data

users, staff: $\mathbb{F}$ REAL_PERSON

$\forall\ b_1, b_2 : \text{books} \bullet b_1.\text{id} = b_2\ .\text{id} \Leftrightarrow b_1 = \ b_2$

dom records = books

$\forall$ user: users •

$\#\ \{b: \text{books} \mid (\text{records } b).\text{lastuser} = \text{user} \wedge$

$(\text{records } b).\text{status} = \text{out}\} \leq \text{MAX}$

Operations

BookOp

Δ Library

$staff' = staff \wedge users = users'$

This is sometimes called a **framing schema**

Adding a book

AddBookCopy

BookOp

book?: Copy

$book? \notin books$

$\forall\ b: books \cdot b.id \neq book?.id$

$\exists\ d: Data \cdot d.lastuser = nobody \land d.status = in \land$

$records' = records \cup \{ book? \mapsto d\}$

Removing a book

RemoveBookCopy

BookOp

book?: Copy

book? ∈ books

records' = {book?} ⩤ records

Copies by an Author

BooksByAuthor

ΞLibrary

author?: AUTHOR

books!: 𝔽 Copy

books! = {b: books | b.author = author?}

Controlling Access

Restricted

Δ Library

doer?: PERSON

doer? $\in$ staff

UnRestricted

Δ Library

doer?: PERSON

doer? $\in$ staff $\cup$ users

SafeAddBook $\triangleq$ AddBook $\wedge$ Restricted
SafeRemoveBook $\triangleq$ RemoveBook $\wedge$ Restricted
SafeBooksByAuthor $\triangleq$ BooksByAuthor $\wedge$ Unrestricted

Note that "$\triangleq$" (\defs) and "==" mean the same thing, but some tools prefer one or the other

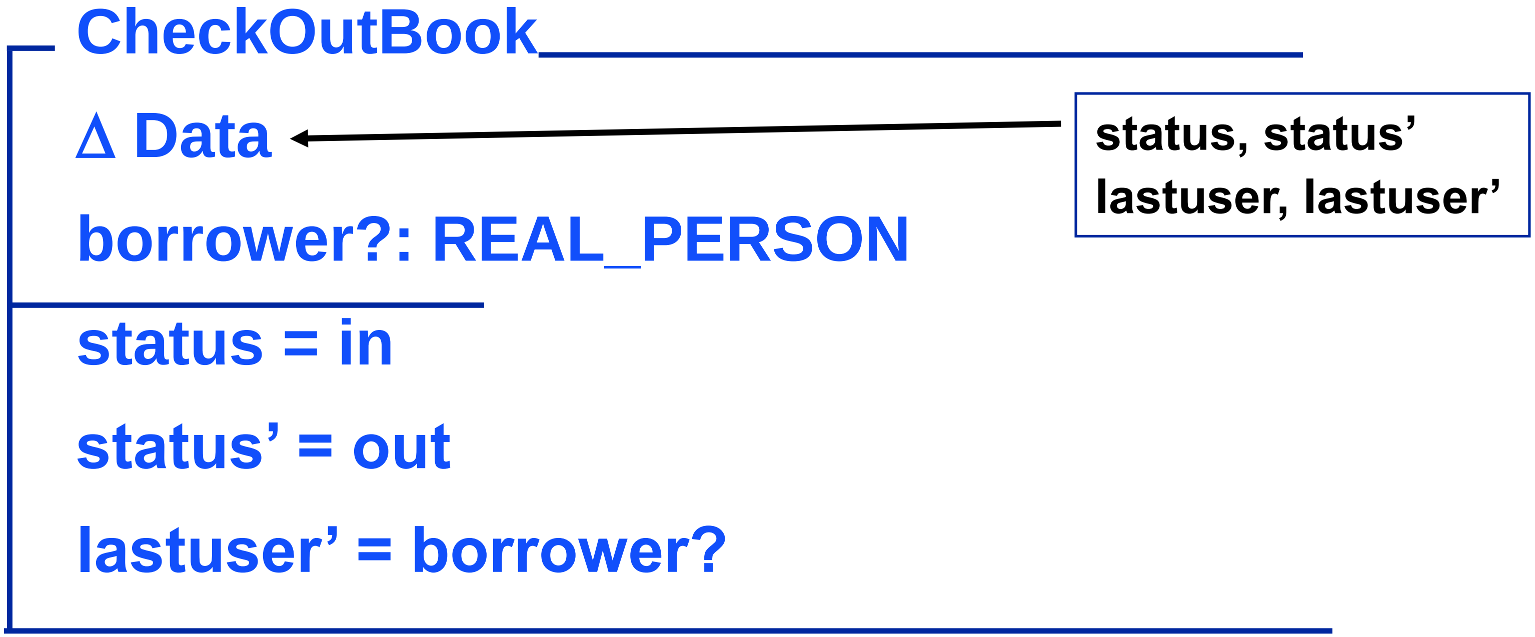
Techniques That We Have Just Seen

- Z Techniques
  - > Framing Schema to say that part of the state does not change
  - > Use of special value to represent an undefined value
  - > Separation of static and dynamic state
  - > Specification of uniqueness within a collection
  - > Separation of access control from normal behavior
  - > Creation of new state as a result of an operation
    - » Use of $\exists\ x: T \bullet P(x) \land\ Q(x)$ to “create” new instance of x

Promotion

- A common problem:
  - > define operations on some schema
  - > place these in the context of a collection of instances of that schema
- Examples:
  - > operations on the state of a game player => operations on the state of the game as a whole
  - > operations on infusion pump lines => operations on a multi-line infusion pumps
  - > operations on books => operations on libraries

Checking Out a Book



CheckOutBook

Δ Data

borrower?: REAL_PERSON

status = in

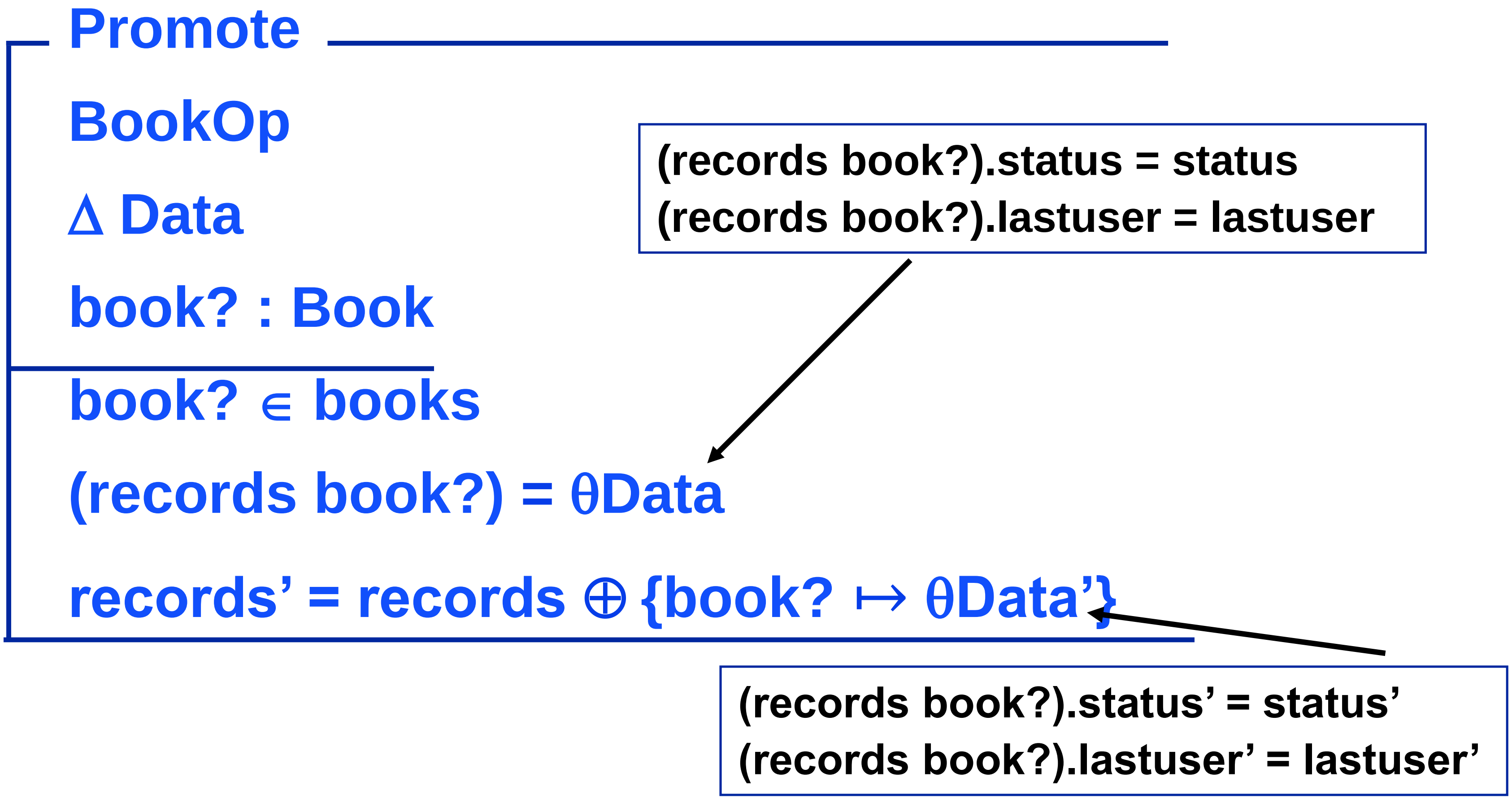
status’ = out

lastuser’ = borrower?

status, status’

lastuser, lastuser’

Framing Schema for Promotion



Promote

BookOp

Δ Data

book? : Book

book? $\in$ books

(records book?) = θData

records’ = records $\oplus$ {book? $\mapsto$ θData’}

(records book?).status = status

(records book?).lastuser = lastuser

(records book?).status’ = status’

(records book?).lastuser’ = lastuser’

The final result

CheckOut == CheckOutBook $\land$ Promote

CheckIn == CheckInBook $\land$ Promote