

FIT3013 Lecture Notes

The Library Case Study

John Hurst, 2009

Computer Science and Software Engineering
Monash University

20090330

Outline

Introduction

Context

Machine

Summary

Objectives of this Lecture

- to expand the mathematical toolkit to enable modelling of relations
- to develop a case study that will widen our abstract machine repertoire
- to discuss the notions of fragile and robust operations

Objectives of this Lecture

- to expand the mathematical toolkit to enable modelling of relations
- to develop a case study that will widen our abstract machine repertoire
- to discuss the notions of fragile and robust operations

Objectives of this Lecture

- to expand the mathematical toolkit to enable modelling of relations
- to develop a case study that will widen our abstract machine repertoire
- to discuss the notions of fragile and robust operations

A Simple Library

Let us model a very simple library with the following requirements:

1. Prospective borrowers must register and are given a unique identifier.
2. Only registered users of the library may borrow books.
3. Borrowers may borrow more than one book at the same time.

A Simple Library

Let us model a very simple library with the following requirements:

1. Prospective borrowers must register and are given a unique identifier.
2. Only registered users of the library may borrow books.
3. Borrowers may borrow more than one book at the same time.

A Simple Library

Let us model a very simple library with the following requirements:

1. Prospective borrowers must register and are given a unique identifier.
2. Only registered users of the library may borrow books.
3. Borrowers may borrow more than one book at the same time.

A Simple Library

To simplify the model we will assume that there is a set *BOOK* that contains all the books that could be in the library. This could be thought of as similar to the set of all ISBN numbers, but as there will be at most only one copy of any book in the library it's more appropriate to compare it with a shelf number, or a book barcode.

When you borrow a book from a library, you don't borrow a book title, you borrow a specific physical book. The specification we are going to develop is probably not wrong when compared with a real library, nor inappropriate, rather it is **incomplete**.

Library Context

We will model the set *BOOK* as a set parameter of the machine.

We will also use a non-zero numeric constant *maxuser* to denote the size of the set of registered users.

Thus the context is

```
CONTEXT LibraryContext  
SETS  
    BOOK  
CONSTANTS  
    maxuser  
END
```

Representing Register Users

To represent the set of all possible registered users of the library we will specify a context set *USER*.

Like *BOOK*, we do not ask exactly what the elements of this set are, other than to note that they model users of the library system.

AXIOMS

In order to constrain the machine parameter *maxuser* to be non-zero, we add an axiom clause to the context:

$$\mathit{maxuser} \in \mathbb{N} - 1.$$

We wish the set *USER* to have exactly *maxuser* elements, and we use the axioms clause to specify this:

$$\mathbf{card}(USER) = \mathit{maxuser}$$

Library Context

CONTEXT *LibraryContext*

SETS

BOOK

USER

CONSTANTS

maxuser

AXIOMS

$maxuser \in \mathbb{N}_1$

$\mathbf{card}(maxuser) = maxuser$

END

Variables

We need to model:

registered users The set of currently registered users

books in the library the set of books owned by the library

books on the shelf the subset of library books that are currently on the shelves, and available for loan

books on loan the set of books out on loan, together with information as to who has borrowed them

Variables

We need to model:

registered users The set of currently registered users

books in the library the set of books owned by the library

books on the shelf the subset of library books that are currently on the shelves, and available for loan

books on loan the set of books out on loan, together with information as to who has borrowed them

Variables

We need to model:

registered users The set of currently registered users

books in the library the set of books owned by the library

books on the shelf the subset of library books that are currently on the shelves, and available for loan

books on loan the set of books out on loan, together with information as to who has borrowed them

Variables

We need to model:

registered users The set of currently registered users

books in the library the set of books owned by the library

books on the shelf the subset of library books that are currently on the shelves, and available for loan

books on loan the set of books out on loan, together with information as to who has borrowed them

Registered Users

We add a variable *users* together with a constraint $users \subseteq USER$ to keep track of which library users are registered.

Note that $users \subseteq USER$ is equivalent to $users \in \mathbb{P}(USERS)$.

Books in the Library

We add a variable *books_in_library* together with a constraint $books_in_library \subseteq BOOK$ to keep track of which books have been acquired by the library.

Note that $books_in_library \subseteq BOOK$ is equivalent to $books_in_library \in \mathbb{P}(BOOK)$.

Books on the Shelf

We add a variable *books_on_shelf* together with a constraint $books_on_shelf \subseteq books_in_library$ to keep track of which books have been acquired by the library.

Books must be acquired before they may appear on the shelf.

Note that $books_on_shelf \subseteq books_in_library$ is equivalent to $books_on_shelf \in \mathbb{P}(books_in_library)$.

Books on Loan

We add a variable *books_on_loan* together with a constraint $books_on_loan \in books_in_library \leftrightarrow users$ to keep track of which books have been borrowed.

Each book that has been borrowed must be borrowed by exactly one registered user.

A borrower may borrow more than one book.

This indicates a functional relationship between books and borrowers.

Strengthening the Invariant

The constraints on the variables are not yet strong enough.

Clearly, a book that is borrowed may not also be on the shelf in the library.

Also, a book acquired by the library is either on the shelf or on loan, at least in our simple library.

Both of these can be expressed by saying that *books_on_shelf* must be exactly the difference between *books_in_library* and the domain of the *books_on_loan* function.

Thus we need the following constraint

$$books_on_shelf = books_in_library \setminus \text{dom}(books_on_loan)$$

(\setminus is the set difference operator.)

Simple Library Machine

MACHINE SL0_Machine

SEES L0_Context

VARIABLES

books_in_library

books_on_shelf

books_on_loan

users

INVARIANTS

inv1 : $users \subseteq USER$

inv2 : $books_in_library \subseteq BOOK$

inv3 : $books_on_shelf \subseteq books_in_library$

inv4 : $books_on_loan \in books_in_library \leftrightarrow users$

inv5 :

$books_on_shelf = books_in_library \setminus \text{dom}(books_on_loan)$

Initialisation

An appropriate initialisation of the variables that will satisfy the machine state invariant is to set all the variables to the empty set.

INITIALISATION

```
users,  
books_in_library,  
books_on_shelf,  
books_on_loan := {}, {}, {}, {}
```

Adding a Book to the Library

We want to model an operation $AddBook(book)$ that adds book to the libraries collection, the set $books_in_library$.

book must be a new book – one that is not already contained in the libraries collection – to the library collection.

We will assume that at the same time as we add the book to the library collection we add it to the library shelves.

$AddBook =$

ANY $book$

WHERE

$book \in BOOK$

$book \notin books_in_library$

THEN

act1: $books_in_library := books_in_library \cup \{book\}$

act2: $books_on_shelf := books_on_shelf \cup \{book\}$

Registering a New User

We want to model an operation *NewUser* that will register a new user, provided that we have not exhausted our set of user tokens.

This operation will update a new state variable *newuser* must be used when borrowing a book. This is the result of the event.

In modelling this operation we choose any user token that has not yet been allocated. We then add this to the set *user*. This is done with a predicate based assignment.

NewUser =

WHERE

$users \neq USER$

THEN

act1: $newuser, users :—$

$newuser' \in (USER \setminus users) \wedge$

$users' = users \cup \{newuser\}$

Summary

- **a**
- b
- c

Summary

- a
- **b**
- c

Summary

- a
- b
- c