Addendum to 'Teaching Touchy Transformations'

Martin Gogolla

University of Bremen (D), CS Department, Database Systems Group

Abstract: This paper is an addendum to the paper 'Teaching Touchy Transformations' presented at the Educator's Symposium during the MODELS conference in 2008 by the same author. That paper reports on a teaching unit on model transformations in which one example model is first developed and afterwards transformed into various other models. The current paper collects the material used for the described teaching unit.

Context

The paper at the Educator's Symposium during the MODELS conference in 2008 describes a library system with different models in different sections. The models and sections are abbreviated and numbered as follows:

- Informal (Section 3.1)
- MaxInvsMinPrepos (Section 3.2)
- MaxPrepos (Section 3.3)
- Assoc2Attr (Section 3.4)
- RelDB1NF (Section 3.5)
- Invs2Super (Section 3.6)
- CompFrame (Section 3.7)

In the following we present the material used for the respective section using the above numbering scheme thus allowing an easy reference to the original paper.

3.1 Informal

3.1.1 Cloze Text

The example describes a digital support system for a library. The library offers book copies to users. A user can borrow a copy or in other words, an exemplar, of a book. A book is characterized by an author list, a year of publication, and a unique title. A copy is determined by the number of return actions of the copy, the book of which the copy is an exemplar of, and a unique signature. A user has an address and a unique name. At most one user can borrow a copy of a book at one particular point in time. Book, copy, and user properties are first manipulated by initialization actions. Both users and copies are able to perform actions for borrowing and returning.
Additionally, certain conditions must hold. If properties such as author, title, signature, address, and name are described by strings, the string is not allowed to be undefined or to be equal to the empty string. A year of publication is equal to or greater than 1455 (the year in which the Gutenberg bible was published). A user having borrowed a copy of a particular book is not allowed to borrow another copy of the same book at the same time. An author can appear at most once in an author list. Finally, as already indicated above, certain properties such as title, signature, and name are unique.

Initialization, borrow and return actions have to respect the above conditions. They can only be performed meaningfully in reasonable situations. They have to fulfill their expected functionality.

3.1.2 Cloze Text with Missing Words Substituted (Last Paragraphs Only)

Additionally, certain conditions must hold. If properties such as author, title, signature, address, and name are described by strings, the string is not allowed to be undefined or to be equal to the empty string. A year of publication is equal to or greater than 1455 (the year in which the Gutenberg bible was published). A user having borrowed a copy of a particular book is not allowed to borrow another copy of the same book at the same time. An author can appear at most once in an author list. Finally, as already indicated above, certain properties such as title, signature, and name are unique.

Initialization, borrow and return actions have to respect the above conditions. They can only be performed meaningfully in reasonable situations. They have to fulfill their expected functionality.

3.2 MaxInvsMinPrepos

3.2.1 Class Diagram
3.2.2 USE Model (Classes, Associations, Invariants, Pre- and postconditions)

-- Library
model Library

class User
attributes
    name: String -- key
    address: String
operations
    init(aName: String, anAddress: String)
    borrow(aCopy: Copy)
    return(aCopy: Copy)
end

class Copy
attributes
    signature: String -- key
    numReturns: Integer
operations
    init(aSignature: String, aBook: Book)
    borrow(aUser: User)
    return()
end

class Book
attributes
    title: String -- key
    authSeq: Sequence(String)
    year: Integer
operations
    init(aTitle: String, anAuthSeq: Sequence(String), aYear: Integer)
end

association Borrows between
    User[0..1] role user
    Copy[0..*] role copy
end

association BelongsTo between
    Copy[0..*] role copy
end

-- constraints
context u: User inv nameAddressFormatOk:
    u.name<>oclUndefined(String) and u.name<>'' and
    u.address<>oclUndefined(String) and u.address<>''
context ul: User inv nameIsKey: User.allInstances->forAll(u2 | ul<>u2 implies ul.name<>u2.name)
context u: User inv noDoubleBorrowings:
    not(u.copy->exists(c1,c2|c1<>c2 and c1.book=c2.book))

context c: Copy inv signatureFormatOk:
    c.signature<>oclUndefined(String) and c.signature<>''
context cl: Copy inv signatureIsKey: Copy.allInstances->forAll(c2 | cl<>c2 implies cl.signature<>c2.signature)
context b:Book

\text{inv titleFormatOk:}
\begin{align*}
\text{\quad b.title} & \text{<> Undefined(String) and b.title<>''} \\
\text{inv titleIsKey: Book.allInstances} & \text{->forall(b2 |}
\text{\quad b1<>b2 implies b1.title<>b2.title)}
\end{align*}

context b:Book

\text{inv authSeqFormatOk: Set[1..b.authSeq->size()] ->forall(i |}
\text{\quad authSeq->at(i)<>Undefined(String) and authSeq->at(i)<>''})

context b:Book

\text{inv authSeqExistsAndUnique: b.authSeq->size()>0 and}
\text{Set[1..b.authSeq->size()-1] ->forall(i |}
\text{\quad Set[i+1..b.authSeq->size()] ->forall(j |}
\text{\quad authSeq->at(i)<>authSeq->at(j))}

context b:Book

\text{inv yearPlausible:}
\begin{align*}
1455 & \text{<> b.year}
\end{align*}

\text{----------------------------------------------- User::init}

context User::init(aName:String, anAddress:String)

pre freshUser:
\begin{align*}
\text{self.name = Undefined(String) and}
\text{self.address = Undefined(String) and self.copy->isEmpty()}
\end{align*}

post attrsAssigned:
\begin{align*}
\text{aName = self.name and anAddress = self.address}
\end{align*}

\text{----------------------------------------------- User::borrow}

context User::borrow(aCopy:Copy)

pre copyOk:
\begin{align*}
\text{aCopy = Undefined(Copy) and aCopy.user->isEmpty()}
\end{align*}

post linkAssigned:
\begin{align*}
\text{self.copy@pre->including(aCopy) = self.copy}
\end{align*}

\text{----------------------------------------------- User::return}

context User::return(aCopy:Copy)

pre aCopyOk:
\begin{align*}
\text{aCopy = Undefined(Copy) and self.copy->includes(aCopy) and}
\text{self.user->isEmpty() and self.book->isEmpty()}
\end{align*}

post bookOk:
\begin{align*}
\text{aBook = Undefined(Book) and}
\text{self.signature = Undefined(String) and}
\text{self.numReturns = Undefined(Integer) and}
\text{self.user->isEmpty() and self.book->isEmpty()}
\end{align*}

\text{----------------------------------------------- Copy::init}

context Copy::init(aSignature:String, aBook:Book)

pre freshCopy:
\begin{align*}
\text{self.signature = Undefined(String) and}
\text{self.numReturns = Undefined(Integer) and}
\text{self.user->isEmpty() and self.book->isEmpty()}
\end{align*}

post bookOk:
\begin{align*}
\text{aBook = Undefined(Book) and}
\text{self.signature = aSignature and 0 = self.numReturns and}
\text{aBook = self.book}
\end{align*}

\text{----------------------------------------------- Copy::borrow}

context Copy::borrow(aUser:User)

pre userOk:
\begin{align*}
\text{aUser = Undefined(User) and notBorrowed:}
\text{self.user->isEmpty() and self.book->isEmpty()}
\end{align*}

post linkAssigned:
\begin{align*}
\text{aUser = self.user}
\end{align*}
3.2.3 Operation Implementations with Command Files

--- --- --- --- --- --- --- --- User::init(aName:String, anAddress:String)
!set self.name:=aName
!set self.address:=anAddress

--- --- --- --- --- --- --- --- User::borrow(aCopy:Copy)
!insert (self,aCopy) into Borrows

--- --- --- --- --- --- --- --- User::return(aCopy:Copy)
!set aCopy.numReturns:=aCopy.numReturns+1
!delete (self,aCopy) from Borrows

--- --- --- --- --- --- --- --- Copy::init(aSignature:String, aBook:Book)
!set self.signature:=aSignature
!set self.numReturns:=0
!insert (self,aBook) into BelongsTo

--- --- --- --- --- --- --- --- Copy::borrow(aUser:User)
!insert (aUser,self) into Borrows

--- --- --- --- --- --- --- --- Copy::return()
!set self.numReturns:=self.numReturns+1
!delete (self.user,self) from Borrows

--- --- --- --- --- --- --- --- Book::init(aTitle:String, anAuthSeq:Sequence(String), aYear:Integer)
!set self.title:=aTitle
!set self.authSeq:=anAuthSeq
!set self.year:=aYear
3.2.4 Sequence and Object Diagram

3.2.5 Complete Command Line Protocol

use> ------------------------------- library.use
use> open library.use

use> ------------------------------- ada:User
use> !create ada:User
use> ?Tuple{name:ada.name,address:ada.address,copy:ada.copy}
    Tuple{name:Undefined,address:Undefined,copy:Set{}} :
    Tuple{name:undefined,address:undefined,copy:Set(Copy)}
use> !openter ada.init('Ada','NY')
precondition `freshUser' is true
use> read User_init.cmd
User::init(aName: String, anAddress: String)
  !set self.name:=aName
  !set self.address:=anAddress

use> !opexit
postcondition `attrsAssigned' is true

use> ?Tuple{name:ada.name,address:ada.address, copy:ada.copy}:
  Tuple(name:'Ada', address:'NY', copy: Set())

use> -- ------------------------------------------------------ bob: User
use> !create bob: User
use> !openter bob init ('Bob', 'CA')
precondition `freshUser' is true
use> read User_init.cmd
-- ------------------------------------------------------ User::init(aName: String, anAddress: String)
  !set self.name:=aName
  !set self.address:=anAddress
use> !opexit
postcondition `attrsAssigned' is true

use> -- ------------------------------------------------------ date: Book
use> !create date: Book
use> !openter date init ('Intro to DBS', Sequence('Date'), 1983)
precondition `freshBook' is true
use> read Book_init.cmd
  -- Book::init(aTitle: String, anAuthSeq: Sequence(String), aYear: Integer)
  !set self.title:=aTitle
  !set self.authSeq:=anAuthSeq
  !set self.year:=aYear
use> !opexit
postcondition `attrsAssigned' is true

use> -- ------------------------------------------------------ dbs42: Copy
use> !create dbs42: Copy
use> !openter dbs42 init ('DBS42', date)
precondition `freshCopy' is true
precondition `bookOk' is true
use> read Copy_init.cmd
-- ------------------------------------------------------ Copy::init(aSignature: String, aBook: Book)
  !set self.signature:=aSignature
  !set self.numReturns:=0
  !insert (self,aBook) into BelongsTo
use> !opexit
postcondition `attrsAndLinkAssigned' is true

use> ?Tuple{name:ada.name,address:ada.address, copy:ada.copy}:
  Tuple(name:'Ada', address:'NY', copy: Set())
use> !openter dbs42 borrow (ada)
precondition `userOk' is true
precondition `notBorrowed' is true
use> read Copy_borrow.cmd
-- ------------------------------------------------------ Copy::borrow(aUser: User)
  !insert (aUser,self) into Borrows
use> !opexit
postcondition `linkAssigned' is true

use> ?Tuple{name:ada.name,address:ada.address, copy:ada.copy}:
  Tuple(name:'Ada', address:'NY', copy: Set(db42))
  Tuple(name: String, address: String, copy: Set(Copy))
use> !create elNa:Book
use> !openter elNa init('Funds of DBS',Sequence('Elmasri','Navathe'),1994) precondition freshBook' is true
use> read Book_init.cmd
-- Book::init(aTitle: String, anAuthSeq: Sequence(String), aYear: Integer)
!set self.title:=aTitle
!set self.authSeq:=anAuthSeq
!set self.year:=aYear
use> !opexit
postcondition 'attrsAssigned' is true

use> ------------------------------- dbs43:Copy
use> !create dbs43:Copy
use> !openter dbs43 init('DBS43',date) precondition 'freshCopy' is true precondition 'bookOk' is true
use> read Copy_init.cmd
-- Copy::init(aSignature: String, aBook: Book)
!set self.signature:=aSignature
!set self.numReturns:=0
!insert (self,aBook) into BelongsTo
use> !opexit
postcondition 'attrsAndLinkAssigned' is true

use> ------------------------------- dbs52:Copy
use> !create dbs52:Copy
use> !openter dbs52 init('DBS52',elNa) precondition 'freshCopy' is true precondition 'bookOk' is true
use> read Copy_init.cmd
-- Copy::init(aSignature: String, aBook: Book)
!set self.signature:=aSignature
!set self.numReturns:=0
!insert (self,aBook) into BelongsTo
use> !opexit
postcondition 'attrsAndLinkAssigned' is true

use> ------------------------------- User::borrow
use> !openter bob borrow(dbs43) precondition 'copyOk' is true
use> read User_borrow.cmd
-- User::borrow(aCopy: Copy)
!insert (self,aCopy) into Borrows
use> !opexit
postcondition 'linkAssigned' is true

use> ------------------------------- Copy::borrow
use> !openter dbs52 borrow(ada) precondition 'userOk' is true precondition 'notBorrowed' is true
use> read Copy_borrow.cmd
-- Copy::borrow(aUser: User)
!insert (aUser,self) into Borrows
use> !opexit
postcondition 'linkAssigned' is true

use> ------------------------------- Copy::return
use> !openter dbs52 return() precondition 'copyOk' is true
use> read Copy_return.cmd
```plaintext
-- set self.numReturns:=self.numReturns+1
!set self.numReturns:=self.numReturns+1
!delete (self.user,self) from Borrows
use>
!opexit
postcondition `linkRemoved' is true
postcondition `numReturnsIncreased' is true
use>
?Tuple{signature:db52.signature,numReturns:db52.numReturns,
user:db52.user,book:db52.book}
Tuple{signature:'DBS52',numReturns:1, user:Undefined, book:@elNa} :
Tuple{signature:String,numReturns:Integer, user:User, book:Book}

use> ------------------------------------------- User::return
use>
!openter db43.user return(db43)
precondition `aCopyOk' is true
use>
read User_return.cmd
-- set aCopy.numReturns:=aCopy.numReturns+1
!set aCopy.numReturns:=aCopy.numReturns+1
!delete (self,aCopy) from Borrows
use>
!opexit
postcondition `linkRemoved' is true
postcondition `numReturnsIncreased' is true
use> -------------------------------------------

3.3 MaxPrepos

3.3.1 Class Diagram

Identical to class diagram in MaxInvsMinPrepos.

3.3.2 USE Model

Changes with respect to MaxInvsMinPrepos are indicated by '-- inv'.

------------------------------------------- Library
model Library
------------------------------------------- class User
class User
attributes
  name: String -- key
  address: String
operations
  init(aName: String, anAddress: String)
  borrow(aCopy: Copy)
  return(aCopy: Copy)
end
------------------------------------------- class Copy
class Copy
attributes
  signature: String -- key
  numReturns: Integer
operations
  init(aSignature: String, aBook: Book)
  borrow(aUser: User)
  return()
end
```
class Book
attributes
  title: String -- key
  authSeq: Sequence(String)
  year: Integer
operations
  init(aTitle: String, anAuthSeq: Sequence(String), aYear: Integer)
end

association Borrows between
  User[0..1] role user
  Copy[0..*] role copy
end

association BelongsTo between
  Copy[0..*] role copy
end

constraints
-- invariants transformed into pre-conditions of operations
context User::init
pre
  nameAddressFormatOk:                        -- inv
    aName<>oclUndefined(String) and aName<>'' and
    anAddress<>oclUndefined(String) and anAddress<>''
pre
  nameIsKey:                                  -- inv
    User.allInstances->collect(u|u.name)->excludes(aName)
pre
  freshUser:
    self.name=oclUndefined(String) and
    self.address=oclUndefined(String) and self.copy->isEmpty()
post
  attrsAssigned:
    aName=self.name and anAddress=self.address
context User::borrow
pre
  noDoubleBorrowings:                         -- inv
  self.copy.book->excludes(aCopy.book)
pre
  copyOk:
    aCopy<>oclUndefined(Copy) and aCopy.user->isEmpty()
post
  linkAssigned:
    self.copy@pre->including(aCopy)=self.copy
context User::return
pre
  aCopyOk:
    aCopy<>oclUndefined(Copy) and self.copy->includes(aCopy)
post
  linkRemoved:
    self.copy@pre->excluding(aCopy)=self.copy
post
  numReturnsIncreased:
    aCopy.numReturns@pre+1=aCopy.numReturns
context Copy::init(aSignature:String, aBook:Book)
pre signatureFormatOk:
  aSignature<>oclUndefined(String) and aSignature<>''
pre signatureIsKey:
  Copy.allInstances->collect(c|c.signature)->excludes(aSignature)
pre freshCopy:
  self.signature=oclUndefined(String) and
  self.numReturns=oclUndefined(Integer) and
  self.user->isEmpty() and self.book->isEmpty()
pre bookOk:
  aBook<>oclUndefined(Book)
post attrsAndLinkAssigned:
  aSignature=self.signature and 0=self.numReturns and
  aBook=self.book
----------------------------- Copy::borrow
context Copy::borrow(aUser:User)
pre noDoubleBorrowings:
  -- inv
pre userOk:
  aUser<>oclUndefined(User)
pre notBorrowed:
  self.user->isEmpty()
post linkAssigned:
  aUser=self.user
----------------------------- Copy::return
context Copy::return()
pre copyOk:
  self.user->notEmpty()
post linkRemoved:
  self.user->isEmpty()
post numReturnsIncreased:
  self.numReturns@pre+1=self.numReturns
------------- Book::init
context Book::
  init(aTitle:String, anAuthSeq:Sequence(String), aYear:Integer)
pre titleFormatOk:
  aTitle<>oclUndefined(String) and aTitle<>''
pre titleIsKey:
  Book.allInstances->collect(b|b.title)->excludes(aTitle)
pre authSeqFormatOk: Set{1..anAuthSeq->size()}->forAll(i|
  anAuthSeq->at(i)<>oclUndefined(String) and anAuthSeq->at(i)<>'')
pre authSeqExistsAndUnique:
  anAuthSeq->size()>0 and
  Set{1..anAuthSeq->size()-1}->forAll(i|
    Set{i+1..anAuthSeq->size()}->forAll(j|
      anAuthSeq->at(i)<>anAuthSeq->at(j)))
pre yearPlausible:
  1455<=aYear
pre freshBook:
  self.title=oclUndefined(String) and
  self.authSeq=oclUndefined(Sequence(String)) and
  self.year=oclUndefined(Integer) and
  self.copy->isEmpty()
post attrsAssigned:
  aTitle=self.title and anAuthSeq=self.authSeq and aYear=self.year
3.4 Assoc2Attr

3.4.1 Class Diagram

![Class Diagram](image)

3.4.2 USE Model

```plaintext
------------------------------ Library
------------------------------ class User
class User
attributes
  name: String -- key
  address: String
  copy: Set(Copy)
operations
  init(aName: String, aAddress: String)
  borrow(aCopy: Copy)
  return(aCopy: Copy)
end
------------------------------ class Copy
class Copy
attributes
  signature: String -- key
  numReturns: Integer
  user: User
  book: Book
operations
  init(aSignature: String, aBook: Book)
  borrow(aUser: User)
  return()
end
```
class Book
attributes
title: String -- key
authSeq: Sequence(String)
year: Integer
copy: Set(Copy)
operations
init(aTitle: String, anAuthSeq: Sequence(String), aYear: Integer)
end
constraints
--- ---- ---- ---- -------- -- guarantee for multiplicity consistency
context c: Copy inv bookIsDefined:
  c.book<>ocl Undefined(Book)
--- ---- ---- ---- -------- -- guarantee for association consistency
context u: User inv userCopyUserEQuser:
  u.copy<>ocl Empty(Set(Copy)) implies u.copy.user->asSet()=Set{u}
context c: Copy inv copyUserCopyEQcopy:
  c.user<>ocl Undefined(User) implies c.user.copy->includes(c)
context b: Book inv bookCopyBookEQbook:
  b.copy<>ocl Empty(Set(Copy)) implies b.copy.book->asSet()=Set{b}
context c: Copy inv copyBookCopyEQcopy:
  c.book.copy->includes(c)
context u: User inv nameAddressFormatOk:
  u.name<>ocl Undefined(String) and u.name<>''
  and u.address<>ocl Undefined(String) and u.address<''
context u1: User inv nameIsKey: User.allInstances->forAll(u2 |
  u1<>u2 implies u1.name<>u2.name)
context u: User inv noDoubleBorrowings:
  not(u.copy->exists(c1,c2|c1<>c2 and c1.book=c2.book))
--- ---- ---- ---- -------- -- Copy
context c: Copy inv signatureFormatOk:
  c.signature<>ocl Undefined(String) and c.signature<''
context c1: Copy inv signatureIsKey: Copy.allInstances->forAll(c2 |
  c1<>c2 implies c1.signature<>c2.signature)
--- ---- ---- ---- -------- -- Book
context b: Book inv titleFormatOk:
  b.title<>ocl Undefined(String) and b.title<''
context b1: Book inv titleIsKey: Book.allInstances->forAll(b2 |
  b1<>b2 implies b1.title<>b2.title)
context b: Book inv authSeqFormatOk: Set(1..b.authSeq->size())->forAll(i |
  b.authSeq->at(i)<ocl Undefined(String) and b.authSeq->at(i)<>''
  and Set(1..b.authSeq->size()-1)->forAll(i |
  Set(i+1..b.authSeq->size())->forAll(j |
  authSeq->at(i)<authSeq->at(j))))
context b: Book inv yearPlausible:
  1455<=b.year
--- ---- ---- ---- -------- -- User::init
context User::init(aName: String, anAddress: String)
pre freshUser:
  self.name=ocl Undefined(String) and
  self.address=ocl Undefined(String) and
  self.copy=ocl Undefined(Set(Copy))                       -- forced change
post attrsAssigned:
  aName=self.name and anAddress=self.address


context User::borrow(aCopy:Copy)
pre copyOk:
    aCopy<>oclUndefined(Copy) and
    aCopy.user=oclUndefined(User)  -- forced change
post linkAssigned:
    self.copy@pre->including(aCopy)=self.copy
context User::return(aCopy:Copy)
pre aCopyOk:
    aCopy<>oclUndefined(Copy) and self.copy->includes(aCopy)
post linkRemoved:
    self.copy@pre->excluding(aCopy)=self.copy
post numReturnsIncreased:
    aCopy.numReturns@pre+1=aCopy.numReturns
context Copy::init(aSignature:String, aBook:Book)
pre freshCopy:
    self.signature=oclUndefined(String) and
    self.numReturns=oclUndefined(Integer) and
    self.user=oclUndefined(User) and
    self.book=oclUndefined(Book)  -- forced change
pre bookOk:
    aBook<>oclUndefined(Book)
post attrsAndLinkAssigned:
    aSignature=self.signature and 0=self.numReturns and
    aBook=self.book
context Copy::borrow(aUser:User)
pre userOk:
    aUser<>oclUndefined(User)
pre notBorrowed:
    self.user=oclUndefined(User)  -- forced change
post linkAssigned:
    aUser=self.user
context Copy::return()
pre copyOk:
    self.user<>oclUndefined(User)  -- forced change
post linkRemoved:
    self.user=oclUndefined(User)  -- forced change
post numReturnsIncreased:
    self.numReturns@pre+1=self.numReturns
context Book::
    init(aTitle: String, anAuthSeq: Sequence(String), aYear: Integer)
pre freshBook:
    self.title=oclUndefined(String) and
    self.authSeq=oclUndefined(Sequence(String)) and
    self.year=oclUndefined(Integer) and
    self.copy=oclUndefined(Set(Copy))  -- forced change
post attrsAssigned:
    aTitle=self.title and anAuthSeq=self.authSeq and aYear=self.year
3.4.3 Operation Implementations with Command Files

```plaintext
-- - - - - - - - - - - - - - User::init(aName:String, anAddress:String)
!set self.name:=aName
!set self.address:=anAddress
!set self.copy:=oclEmpty(Set(Copy))

-- !insert (self,aCopy) into Borrows
!set self.copy:=self.copy->including(aCopy)
!set aCopy.user:=self

-- - - - - - - - - - - - - - User::borrow(aCopy:Copy)
!set aCopy.numReturns:=aCopy.numReturns+1
-- !delete (self,aCopy) from Borrows
!set self.copy:=self.copy->excluding(aCopy)
!set aCopy.user:=oclUndefined(User)

-- - - - - - - - - - - - - - Copy::init(aSignature:String, aBook:Book)
!set self.signature:=aSignature
!set self.numReturns:=0
-- !insert (self,aBook) into BelongsTo
!set self.book:=aBook
!set aBook.copy:=aBook.copy->including(self)

-- - - - - - - - - - - - - - Copy::borrow(aUser:User)
-- !insert (aUser,self) into Borrows
!set self.user:=aUser
!set aUser.copy:=aUser.copy->including(self)

-- - - - - - - - - - - - - - Copy::return()
!set self.numReturns:=self.numReturns+1
-- !delete (self.user,self) from Borrows
!set self.user.copy:=self.user.copy->excluding(self)
!set self.user:=oclUndefined(User)

-- Book::init(aTitle:String, anAuthSeq:Sequence(String), aYear:Integer)
!set self.title:=aTitle
!set self.authSeq:=anAuthSeq
!set self.year:=aYear
!set self.copy:=oclEmpty(Set(Copy))
```

---
3.4.4 Sequence Diagram (Excerpt)
3.4.5 Object Diagram

3.5 RelDB1NF

3.5.1 Class Diagram

```
Library

db: Tuple(User: Set(Tuple(name: String, address: String)),
          Copy: Set(Tuple(signature: String, numReturns: Integer, name: String, title: String)),
          Book: Set(Tuple(title: String, year: Integer)),
          authSeq: Set(Tuple(title: String, pos: Integer, author: String)))
```

3.5.2 USE Model

```
--- Library

model Library

---

class Library

attributes

db:Tuple(User: Set(Tuple(name: String, address: String)),
          Copy: Set(Tuple(signature: String, numReturns: Integer, name: String, title: String)),
          Book: Set(Tuple(title: String, year: Integer)),
          authSeq: Set(Tuple(title: String, pos: Integer, author: String)))
```
operations

Library_init()
User_init(aName:String, anAddress:String)
User_borrow(aName:String, aSignature:String)
User_return(aName:String, aSignature:String)
Copy_init(aSignature:String, aTitle:String)
Copy_borrow(aSignature:String, aName:String)
Copy_return(aSignature:String)
Book_init(aTitle:String, anAuthSeq:Sequence(String), aYear:Integer)

User_name2tuple(aName:String):
  Tuple(name:String, address:String) =
  self.db.User->select(name=aName)->any(true)

Copy_signature2tuple(aSignature:String):
  Tuple(signature:String, numReturns:Integer, name:String, title:String) =
  self.db.Copy->select(signature=aSignature)->any(true)

Book_title2tuple(aTitle:String):
  Tuple(title:String, year:Integer) =
  self.db.Book->select(title=aTitle)->any(true)

authSeq_titlePos2tuple(aTitle:String, aPos:Integer):
  Tuple(title:String, pos:Integer, author:String) =
  self.db.authSeq->select(title=aTitle and pos=aPos)->any(true)

constraints

User context Library inv nameAddressFormatOk:
  self.db.User->forAll(u:Tuple(name:String, address:String) | u.name<>oclUndefined(String) and u.name<>'' and
  u.address<>oclUndefined(String) and u.address<>'')

context Library inv nameIsKey:
  self.db.User->forAll(u1, u2:Tuple(name:String, address:String) | u1<>u2 implies u1.name<>u2.name)

context Library inv noDoubleBorrowings:
  self.db.User->forAll(u:Tuple(name:String, address:String) | not (self.db.Copy->exists(c1, c2: Tuple(signature:String, numReturns:Integer, name:String, title:String) | c1<>c2 and c1.name=u.name and c2.name=u.name and
  c1.title=c2.title)))

Copy context Library inv signatureFormatOk:
  self.db.Copy->forAll(c: Tuple(signature:String, numReturns:Integer, name:String, title:String) | c.signature<>oclUndefined(String) and c.signature>'')

context Library inv signatureIsKey:
  self.db.Copy->forAll(c1, c2: Tuple(signature:String, numReturns:Integer, name:String, title:String) | c1<>c2 implies c1.signature<>c2.signature)
context Library inv titleFormatOk:
  self.db.Book->forAll (b:Tuple (title: String, year: Integer) |
                       b.title<>oclUndefined(String) and b.title<>'')
context Library inv titleIsKey:
  self.db.Book->forAll (b1, b2:Tuple (title: String, year: Integer) |
                        b1<>b2 implies b1.title<>b2.title)
context Library inv authSeqFormatOk:
  self.db.authSeq->forAll (aS: Tuple (title: String, pos: Integer, author: String) |
                           aS.author<>oclUndefined(String) and aS.author<>'')
context Library inv authSeqExistsAndUnique:
  self.db.Book->forAll (b:Tuple (title: String, year: Integer) |
                         self.db.authSeq->exists (aS: Tuple (title: String, pos: Integer, author: String) |
                                                  b.title=aS.title and aS.pos=1) and
                         self.db.authSeq->forAll (aS1, aS2: Tuple (title: String, pos: Integer, author: String) |
                                                  b.title=aS1.title and b.title=aS2.title and aS1.pos<>aS2.pos implies aS1.author<>aS2.author) and
                         self.db.authSeq->forAll (aS1: Tuple (title: String, pos: Integer, author: String) |
                                                  b.title=aS1.title and 1<aS1.pos implies
                         self.db.authSeq->exists (aS2: Tuple (title: String, pos: Integer, author: String) |
                                                  aS2.title=aS1.title and aS2.pos=aS1.pos-1)))
context Library inv yearPlausible:
  self.db.Book->forAll (b:Tuple (title: String, year: Integer) |
                         1455<year)

3.5.3 Operation Implementations with Command Files

-- Library_init()
!set self.db:=
  Tuple (User:oclEmpty (Set(Tuple (name: String, address: String)))),
  Copy:oclEmpty (Set(Tuple (signature: String, numReturns: Integer, name: String, title: String))),
  Book:oclEmpty (Set(Tuple (title: String, year: Integer))),
  authSeq:oclEmpty (Set(Tuple (title: String, pos: Integer, author: String))))

-- User_init(aName: String, anAddress: String)
!set self.db:=
  Tuple (User:self.db.User->
        including(Tuple (name: aName, address: anAddress))),
  Copy:self.db.Copy,
  Book:self.db.Book,
  authSeq:self.db.authSeq)
-- User_borrow(aName:String, aSignature:String)
!set self.db:=
  Tuple{User:self.db.User,
       Copy:self.db.Copy->
         reject(t|t.signature=aSignature)->
           including(Tuple{signature:aSignature,
                         numReturns:self.db.Copy->
                           select(signature=aSignature)->
                             any(true).numReturns,
                           name:aName,
                           title:self.db.Copy->
                             select(signature=aSignature)->
                               any(true).title})},

Book:self.db.Book,
authSeq:self.db.authSeq}

-- User_return(aName:String, aSignature:String)
!set self.db:=
  Tuple{User:self.db.User,
       Copy:self.db.Copy->
         reject(t|t.signature=aSignature)->
           including(Tuple{signature:aSignature,
                         numReturns:self.db.Copy->
                           select(signature=aSignature)->
                             any(true).numReturns+1,
                           name:oclUndefined(String),
                           title:self.db.Copy->
                             select(signature=aSignature)->
                               any(true).title})},

Book:self.db.Book,
authSeq:self.db.authSeq}

-- Copy_init(aSignature:String, aTitle:String)
!set self.db:=
  Tuple{User:self.db.User,
       Copy:self.db.Copy->including(Tuple{signature:aSignature,
                                           numReturns:0,
                                           name:oclUndefined(String),
                                           title:aTitle})},

Book:self.db.Book,
authSeq:self.db.authSeq}

-- Copy_borrow(aSignature:String, aName:String)
!set self.db:=
  Tuple{User:self.db.User,
       Copy:self.db.Copy->
         reject(t|t.signature=aSignature)->
           including(Tuple{signature:aSignature,
                         numReturns:self.db.Copy->
                           select(signature=aSignature)->
                             any(true).numReturns,
                         name:aName,
                         title:self.db.Copy->
                             select(signature=aSignature)->
                               any(true).title})},

Book:self.db.Book,
authSeq:self.db.authSeq}
3.5.4 System State (Object Diagram)

Tuple{
  User=Set{Tuple{name='Ada',address='NY'},
           Tuple{name='Bob',address='CA'}},
  Copy=Set{Tuple{signature='DBS42',numReturns=0,
                 name='Ada',title='Intro to DBS'},
           Tuple{signature='DBS43',numReturns=1,
                 name=Undefined,title='Intro to DBS'},
           Tuple{signature='DBS52',numReturns=1,
                 name=Undefined,title='Funds of DBS'}},
  Book=Set{Tuple{title='Funds of DBS',year=1994},
            Tuple{title='Intro to DBS',year=1983}},
  authSeq=Set{Tuple{title='Funds of DBS',pos=1,author='Elmasri'},
              Tuple{title='Funds of DBS',pos=2,author='Navathe'},
              Tuple{title='Intro to DBS',pos=1,author='Date'}},

Tuple(User:Set{Tuple(name:String,address:String)}),
    Copy:Set{Tuple(signature:String,numReturns:Integer,
                name:String,title:String)},
    Book:Set{Tuple(title:String,year:Integer)},
    authSeq:Set{Tuple(title:String,pos:Integer,author:String))}
3.6 Invs2Super

3.6.1 Class Diagram

![Class Diagram Image]

3.6.2 USE Model

```plaintext
library Library

abstract class Keyed

operations
  keyValue(): OclAny = oclUndefined(OclAny)
  comparableTo(o: OclAny): Boolean = oclUndefined(Boolean)

constraints
  inv self: diffObjectsDiffKeys:
    Keyed.allInstances -> forall(self2 |
      self <> self2 and self.comparableTo(self2) implies
      self.keyValue() <> self2.keyValue())

end

class User < Keyed

attributes
  name: String -- key
  address: String

operations
  init(aName: String, anAddress: String)
  borrow(aCopy: Copy)
  return(aCopy: Copy)
  keyValue(): String = name
  comparableTo(o: OclAny): Boolean = o.oclIsTypeOf(User)

end
```

22
class Copy < Keyed
attributes
  signature: String -- key
  numReturns: Integer
operations
  init(aSignature: String, aBook: Book)
  borrow(aUser: User)
  return()
  keyValue(): String = signature
  compareTo(o: OclAny): Boolean = o.oclIsTypeOf(Copy)
end

class Book < Keyed
attributes
  title: String -- key
  authSeq: Sequence(String)
  year: Integer
operations
  init(aTitle: String, anAuthSeq: Sequence(String), aYear: Integer)
  keyValue(): String = title
  compareTo(o: OclAny): Boolean = o.oclIsTypeOf(Book)
end

association Borrows between
  User[0..1] role user
  Copy[0..*] role copy
end

association BelongsToTo between
  Copy[0..*] role copy
end

invariants

constraints

User context u: User inv nameAddressFormatOk:
  u.name<>oclUndefined(String) and u.name<>'' and
  u.address<>oclUndefined(String) and u.address<>''
-- context u1: User inv nameIsKey: User.allInstances->forAll(u2 | --
  u1<u2 implies u1.name<>u2.name) --
context u: User inv noDoubleBorrowings:
  not(u.copy->exists(c1,c2|c1<>c2 and c1.book=c2.book))
-- context c: Copy inv signatureFormatOk:
  c.signature<oclUndefined(String) and c.signature<>''
-- context c1: Copy inv signatureIsKey: Copy.allInstances->forAll(c2 | --
  c1<>c2 implies c1.signature<>c2.signature)
-- context b: Book inv titleFormatOk:
  b.title<>oclUndefined(String) and b.title<>''
-- context b1: Book inv titleIsKey: Book.allInstances->forAll(b2 | --
  b1<>b2 implies b1.title<>b2.title)
context b: Book inv authSeqFormatOk: Set(1..b.authSeq->size())->forAll(i|
  authSeq->at(i)<oclUndefined(String) and authSeq->at(i)<>'')
context b: Book inv authSeqExistsAndUnique: b.authSeq->size() > 0 and
    \{ \{ i \in \ldots b.authSeq->size() - 1 \} \rightarrow \forall i \}
    \{ \{ i + 1 \leq b.authSeq->size() \} \rightarrow \forall j \}
    authSeq->at(i) \neq authSeq->at(j)\}
context b: Book inv yearPlausible:
    1455 \leq b.year

----------------------------------------------- pre- and postconditions
-----------------------------------------------
context User::init(aName: String, anAddress: String)
pre freshUser:
    self.name=oclUndefined(String) and
    self.address=oclUndefined(String) and self.copy->isEmpty()
post attrsAssigned:
    aName=self.name and anAddress=self.address

context User::borrow(aCopy: Copy)
pre copyOk:
    aCopy<oclUndefined(Copy) and aCopy.user->isEmpty()
post linkAssigned:
    self.copy@pre->including(aCopy)=self.copy

context User::return(aCopy: Copy)
pre copyOk:
    aCopy<oclUndefined(Copy) and self.copy->includes(aCopy)
post linkRemoved:
    self.copy@pre->excluding(aCopy)=self.copy
post numReturnsIncreased:
    aCopy.numReturns@pre+1=aCopy.numReturns

context Copy::init(aSignature: String, aBook: Book)
pre freshCopy:
    self.signature=oclUndefined(String) and
    self.numReturns=oclUndefined(Integer) and
    self.user->isEmpty() and self.book->isEmpty()
pre bookOk:
    aBook<oclUndefined(Book)
post attrsAndLinkAssigned:
    aSignature=self.signature and 0=self.numReturns and
    aBook=self.book

context Copy::borrow(aUser: User)
pre userOk:
    aUser<oclUndefined(User)
pre notBorrowed:
    self.user->isEmpty()
post linkAssigned:
    aUser=self.user

context Copy::return()
pre copyOk:
    self.user->notEmpty()
post linkRemoved:
    self.user->isEmpty()
post numReturnsIncreased:
    self.numReturns@pre+1=self.numReturns
3.7 CompFrame

3.7.1 Class Diagram

Identical to class diagram in MaxInvsMinPrepos.

3.7.2 USE Model

--- Library ---
model Library
--- class User ---
class User
attributes
  name: String -- key
  address: String
operations
  init(aName: String, anAddress: String)
  borrow(aCopy: Copy)
  return(aCopy: Copy)
  doNothing()
end
--- class Copy ---
class Copy
attributes
  signature: String -- key
  numReturns: Integer
operations
  init(aSignature: String, aBook: Book)
  borrow(aUser: User)
  return()
end
--- class Book ---
class Book
attributes
  title: String -- key
  authSeq: Sequence(String)
  year: Integer
operations
  init(aTitle: String, anAuthSeq: Sequence(String), aYear: Integer)
end
--- association Borrows ---
association Borrows between
  User[0..1] role user
  Copy[0..*] role copy
end
association BelongsTo between
  Copy[0..*] role copy
end

constraints

context u:User inv nameAddressFormatOk:
  u.name<>oclUndefined(String) and u.name<>'' and
  u.address<>oclUndefined(String) and u.address<>''
context u1:User inv nameIsKey: User.allInstances->forAll(u2 |
  u1<>u2 implies u1.name<>u2.name)
context u:User inv noDoubleBorrowings:
  not(u.copy->exists(c1,c2|c1<>c2 and c1.book=c2.book))

context c:Copy inv signatureFormatOk:
  c.signature<>oclUndefined(String) and c.signature<>''
context cl:Copy inv signatureIsKey: Copy.allInstances->forAll(c2 |
  c1<>c2 implies c1.signature<>c2.signature)

context b:Book inv titleFormatOk:
  b.title<>oclUndefined(String) and b.title<>''
context b1:Book inv titleIsKey: Book.allInstances->forAll(b2 |
  b1<>b2 implies b1.title<>b2.title)
context b:Book inv authSeqFormatOk: Set(1..b.authSeq->size()-1)->forAll(i|
  authSeq->at(i)<>oclUndefined(String) and authSeq->at(i)<>'')
context b:Book inv authSeqExistsAndUnique: b.authSeq->size()>0 and
  Set(1..b.authSeq->size())->forAll(i|
  Set(i+1..b.authSeq->size())->forAll(j|
    authSeq->at(i)<>authSeq->at(j))))
context b:Book inv yearPlausible:
  1455<=b.year

context User::doNothing() -- systematic description of 'unchanged'
pre neverCalled: false
post unChanged:
  User.allInstances@pre=User.allInstances and
  User.allInstances->forAll(u|
    u.name@pre=u.name and u.address@pre=u.address and
    u.copy@pre=u.copy)
post copyUnChanged:
  Copy.allInstances@pre=Copy.allInstances and
  Copy.allInstances->forAll(c|
    c.signature@pre=c.signature and c.numReturns@pre=c.numReturns and
    c.user@pre=c.user and c.book@pre=c.book)
post bookUnChanged:
  Book.allInstances@pre=Book.allInstances and
  Book.allInstances->forAll(b|
    b.title@pre=b.title and b.authSeq@pre=b.authSeq and
    b.year@pre=b.year and b.copy@pre=b.copy)

context User::init(aName: String, anAddress: String)
pre freshUser:
  self.name=oclUndefined(String) and
  self.address=oclUndefined(String) and self.copy->isEmpty()
post attrsAssigned:
  aName=self.name and anAddress=self.address
post userNearlyUnchanged:
User.allInstances@pre=User.allInstances and
User.allInstances->forAll(u|
  (u<>self implies u.name@pre=u.name) and
  (u<>self implies u.address@pre=u.address) and
  u.copy@pre=u.copy)

post copyUnchanged:
Copy.allInstances@pre=Copy.allInstances and
Copy.allInstances->forAll(c|
  c.signature@pre=c.signature and c.numReturns@pre=c.numReturns and
  c.user@pre=c.user and c.book@pre=c.book)

context User:: borrow
pre copyOk:
  aCopy<>oclUndefined(Copy) and aCopy.user->isEmpty()
post linkAssigned:
  self.copy@pre->including(aCopy)=self.copy
post userNearlyUnchanged:
User.allInstances@pre=User.allInstances and
User.allInstances->forAll(u|
  u.name@pre=u.name and u.address@pre=u.address and
  (u<>self implies u.copy@pre=u.copy))

context User:: return
pre aCopyOk:
  aCopy<>oclUndefined(Copy) and self.copy->includes(aCopy)
post linkRemoved:
  self.copy@pre->excluding(aCopy)=self.copy
post numReturnsIncreased:
  aCopy.numReturns@pre+1=aCopy.numReturns
post userNearlyUnchanged:
User.allInstances@pre=User.allInstances and
User.allInstances->forAll(u|
  u.name@pre=u.name and u.address@pre=u.address and
  (u<>self implies u.copy@pre=u.copy))

context User:: borrow
pre copyOk:
  aCopy<>oclUndefined(Copy) and aCopy.user->isEmpty()
post linkAssigned:
  self.copy@pre->including(aCopy)=self.copy
post userNearlyUnchanged:
User.allInstances@pre=User.allInstances and
User.allInstances->forAll(u|
  u.name@pre=u.name and u.address@pre=u.address and
  (u<>self implies u.copy@pre=u.copy))

context User:: return
pre aCopyOk:
  aCopy<>oclUndefined(Copy) and self.copy->includes(aCopy)
post linkRemoved:
  self.copy@pre->excluding(aCopy)=self.copy
post numReturnsIncreased:
  aCopy.numReturns@pre+1=aCopy.numReturns
post userNearlyUnchanged:
User.allInstances@pre=User.allInstances and
User.allInstances->forAll(u|
  u.name@pre=u.name and u.address@pre=u.address and
  (u<>self implies u.copy@pre=u.copy))

context User:: borrow
pre copyOk:
  aCopy<>oclUndefined(Copy) and aCopy.user->isEmpty()
post linkAssigned:
  self.copy@pre->including(aCopy)=self.copy
post userNearlyUnchanged:
User.allInstances@pre=User.allInstances and
User.allInstances->forAll(u|
  u.name@pre=u.name and u.address@pre=u.address and
  (u<>self implies u.copy@pre=u.copy))

context User:: return
pre aCopyOk:
  aCopy<>oclUndefined(Copy) and self.copy->includes(aCopy)
post linkRemoved:
  self.copy@pre->excluding(aCopy)=self.copy
post numReturnsIncreased:
  aCopy.numReturns@pre+1=aCopy.numReturns
post userNearlyUnchanged:
User.allInstances@pre=User.allInstances and
User.allInstances->forAll(u|
  u.name@pre=u.name and u.address@pre=u.address and
  (u<>self implies u.copy@pre=u.copy))
post bookUnchanged:
  Book.allInstances@pre=Book.allInstances and
  Book.allInstances->forall(b|
    b.title@pre=b.title and b.authSeq@pre=b.authSeq and
    b.year@pre=b.year and b.copy@pre=b.copy)

context Copy::init(aSignature:String, aBook:Book)
pre freshCopy:
  self.signature=oclUndefined(String) and
  self.numReturns=oclUndefined(Integer) and
  self.user->isEmpty() and self.book->isEmpty()
post bookOk:
  aBook<>oclUndefined(Book)
post attrsAndLinkAssigned:
  aSignature=self.signature and 0=self.numReturns and
  aBook=self.book
post userUnchanged:
  User.allInstances@pre=User.allInstances and
  User.allInstances->forall(u|
    u.name@pre=u.name and u.address@pre=u.address and
    u.copy@pre=u.copy)
post copyNearlyUnchanged:
  Copy.allInstances@pre=Copy.allInstances and
  Copy.allInstances->forall(c|
    (c<>self implies c.signature@pre=c.signature) and
    (c<>self implies c.numReturns@pre=c.numReturns) and
    c.user@pre=c.user and
    (c<>self implies c.book@pre=c.book))
post bookNearlyUnchanged:
  Book.allInstances@pre=Book.allInstances and
  Book.allInstances->forall(b|
    b.title@pre=b.title and b.authSeq@pre=b.authSeq and
    b.year@pre=b.year and
    (b<>self.book implies b.copy@pre=b.copy))

context Copy::borrow(aUser:User)
pre userOk:
  aUser<>oclUndefined(User)
pre notBorrowed:
  self.user->isEmpty()
post linkAssigned:
  aUser=self.user
post userNearlyUnchanged:
  User.allInstances@pre=User.allInstances and
  User.allInstances->forall(u|
    u.name@pre=u.name and u.address@pre=u.address and
    (u<>aUser implies u.copy@pre=u.copy))
post copyNearlyUnchanged:
  Copy.allInstances@pre=Copy.allInstances and
  Copy.allInstances->forall(c|
    c.signature@pre=c.signature and c.numReturns@pre=c.numReturns and
    (c<>self implies c.user@pre=c.user) and
    c.book@pre=c.book)
post bookUnchanged:
  Book.allInstances@pre=Book.allInstances and
  Book.allInstances->forall(b|
    b.title@pre=b.title and b.authSeq@pre=b.authSeq and
    b.year@pre=b.year and b.copy@pre=b.copy)
context Copy::return()
pre copyOk:
    self.user->notEmpty()
post linkRemoved:
    self.user->isEmpty()
post numReturnsIncreased:
    self.numReturns@pre+1=self.numReturns
post userNearlyUnchanged:
    User.allInstances@pre=User.allInstances and
    User.allInstances-->forall(u|
        u.name@pre=u.name and u.address@pre=u.address and
        (u<>self.user@pre implies u.copy@pre=u.copy))
post copyNearlyUnchanged:
    Copy.allInstances@pre=Copy.allInstances and
    Copy.allInstances-->forall(c|
        c.signature@pre=c.signature and
        c.numReturns@pre=c.numReturns and
        c.user@pre=c.user and
        c.book@pre=c.book)
post bookUnchanged:
    Book.allInstances@pre=Book.allInstances and
    Book.allInstances-->forall(b|
        b.title@pre=b.title and b.authSeq@pre=b.authSeq and
        b.year@pre=b.year and b.copy@pre=b.copy)

context Book::init
init(aTitle:String, anAuthSeq:Sequence(String), aYear:Integer)
pre freshBook:
    self.title=oclUndefined(String) and
    self.authSeq=oclUndefined(Sequence(String)) and
    self.year=oclUndefined(Integer) and
    self.copy-->isEmpty()
post attrsAssigned:
    aTitle=self.title and anAuthSeq=self.authSeq and aYear=self.year
post userUnchanged:
    User.allInstances@pre=User.allInstances and
    User.allInstances-->forall(u|
        u.name@pre=u.name and u.address@pre=u.address and
        u.copy@pre=u.copy)
post copyUnchanged:
    Copy.allInstances@pre=Copy.allInstances and
    Copy.allInstances-->forall(c|
        c.signature@pre=c.signature and c.numReturns@pre=c.numReturns and
        c.user@pre=c.user and c.book@pre=c.book)
post bookNearlyUnchanged:
    Book.allInstances@pre=Book.allInstances and
    Book.allInstances-->forall(b|
        b<>self implies b.title@pre=b.title) and
        b<>self implies b.authSeq@pre=b.authSeq) and
        b<>self implies b.year@pre=b.year) and
        b.copy@pre=b.copy)
3.7.3 Command Line Protocol Showing Difference to MaxInvsMinPrepos

use> open libraryWithoutFrameConditions.use
use> ada:User
use> create ada:User
use> openter ada init('Ada','NY')
    precondition `freshUser' is true
use> set self.name:=aName
use> set self.address:=anAddress
use> opexit
    postcondition `attrsAssigned' is true
use> bob:User
use> create bob:User
use> openter bob init('Bob','CA')
    precondition `freshUser' is true
use> set self.name:=aName
use> set self.address:=anAddress
use> set ada.address:='TX'
use> opexit
    postcondition `attrsAssigned' is true

use> open libraryWithFrameConditions.use
use> ada:User
use> create ada:User
use> openter ada init('Ada','NY')
    precondition `freshUser' is true
use> set self.name:=aName
use> set self.address:=anAddress
use> opexit
    postcondition `attrsAssigned' is true
    postcondition `userNearlyUnchanged' is true
    postcondition `copyUnchanged' is true
    postcondition `bookUnchanged' is true

use> bob:User
use> create bob:User
use> openter bob init('Bob','CA')
    precondition `freshUser' is true
use> set self.name:=aName
use> set self.address:=anAddress
use> set ada.address:='TX'
use> opexit
    postcondition `attrsAssigned' is true
    postcondition `userNearlyUnchanged' is false -- postcondition fails
    postcondition `copyUnchanged' is true
    postcondition `bookUnchanged' is true
Table of Contents

3.1 Informal
3.1.1 Cloze Text
3.1.2 Cloze Text with Missing Words Substituted (Last Paragraphs Only)

3.2 MaxInvsMinPrepos
3.2.1 Class Diagram
3.2.2 USE Model (Classes, Associations, Invariants, Pre- and postconditions)
3.2.3 Operation Implementations with Command Files
3.2.4 Sequence and Object Diagram
3.2.5 Complete Command Line Protocol

3.3 MaxPrepos
3.3.1 Class Diagram
3.3.2 USE Model

3.4 Assoc2Attr
3.4.1 Class Diagram
3.4.2 USE Model
3.4.3 Operation Implementations with Command Files
3.4.4 Sequence Diagram (Excerpt)
3.4.5 Object Diagram

3.5 RelDB1NF
3.5.1 Class Diagram
3.5.2 USE Model
3.5.3 Operation Implementations with Command Files
3.5.4 System State (Object Diagram)

3.6 Invs2Super
3.6.1 Class Diagram
3.6.2 USE Model

3.7 CompFrame
3.7.1 Class Diagram
3.7.2 USE Model
3.7.3 Command Line Protocol Showing Difference to MaxInvsMinPrepos