A General Model Based Slicing Framework

Tony Clark

School of Engineering and Information Sciences, Middlesex University

September 30, 2011

Contents

Definition and Use Cases

What? Slicing is used to reduce the size of programs by removing those statements that do not contribute to the values of specified variables at a given program location.

Why? Debugging; comprehension; verification.

How? Static (backwards,forwards); dynamic; conditioned; union.

Backward Slicing: Example1

Original	Criterion	Slice
`x = 42`		
`d = 23`	`d`	`d = 23`

Backward Slicing: Example2

Original	Criterion	Slice

sum=0
prod=1
read(i)
while(i<11) {
sum=sum+i
prod=prod*i
i=i+1
}
	prod	prod = 1
		read(i)
		while (i<11) {
		
		prod=prod*i
		i=i+1
		}

Dynamic Slicing

Original	Criterion	Slice

`sum=0`
`prod=1`
`read(i)`
`while(i<11) {`
`  sum=sum+i`
`  prod=prod*i`
`  i=i+1`
`}`
	`prod, pos(next(input))=11`	`prod = 1`
		`read(i)`

Conditioned Slicing

Original	Criterion	Slice
`read(a)`		`read(a)`
`if(a<0)`		
`a=-a`		
`x=1/a`	`x, +ve(next(input))`	`x=1/a`

Unions

Original	Criterion	Slice

`a = 42`
`x = 2`
`b = 23 + a`
`y = 3`
`c = b + 2`
	`c`	`a = 42`
		
		`b = 23 + a`
		
		`c = b + 2`

`a = 42`
`x = 2`
`b = 23 + a`
`y = 3`
`c = b + 2`
	`y`	`y = 3`

`a = 42`
`x = 2`
`b = 23 + a`
`y = 3`
`c = b + 2`
	`y and c`	`a = 42`
		
		`b = 23 + a`
		`y = 3`
		`c = b + 2`

Contents

Definition and Use Cases

What? *erm - drop some bits?*

Why? Model management and comprehension; debugging transformations; debugging executable models; model verification.

How? State machines; class diagrams (syntax); specific technologies.

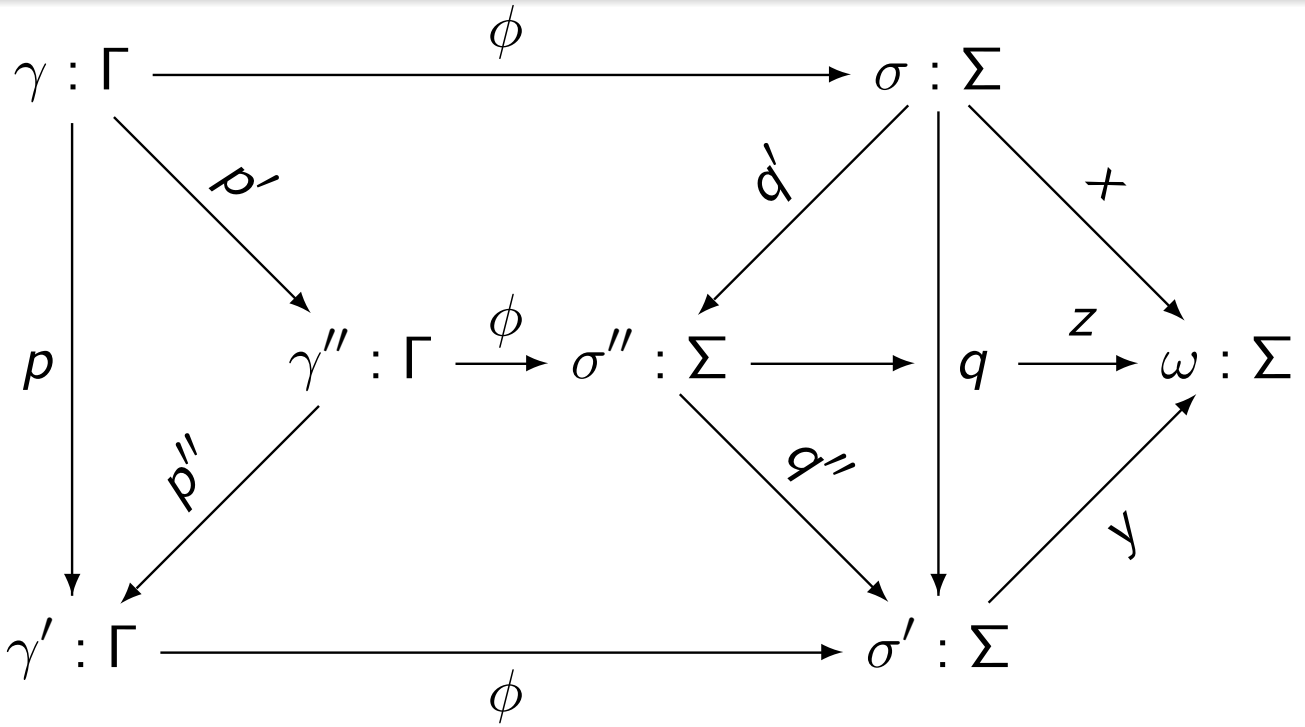
Problem No general definition for model slicing.

Differences to Programs

- No starting point for execution.
- Static *and* dynamic slices.
- Multiple languages.
- DSLs, profiles, meta-models.
- Highly structured.
- Weak semantics.
- Links to model transformations; transformation combinations.

Contents

A General Slicing Framework



Γ Syntax domain.
Σ Semantic domain.
p, q Projections.
ϕ Semantics.
(x, ω) Slicing criterion.

Key Features

To set up the framework for a particular domain you will need:

syntax (Γ) The things you want to slice.

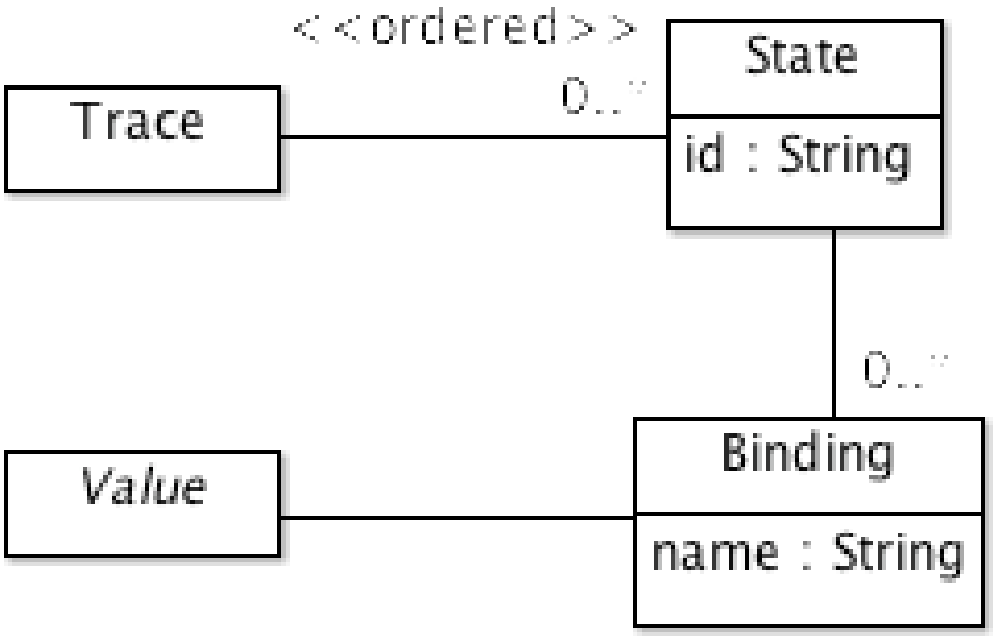
semantics (Σ, ϕ) The meaning of the syntax defined as a semantic domain and a semantic mapping.

projections (p, q) Relationships that hold between syntactic and semantic elements and that remove items that are not of interest.

criterion (x, ω) The things that remain invariant, defined in terms of the semantic domain.

products If the syntax and semantic domain also define products for elements and projections then there is scope for slicing composition.

Semantic Domain for Programs



GSF: Program Example1

Original	Criterion	Slice
`x = 42`		
`d = 23`	`d`	`d = 23`

Semantics is $\sigma = \{[\{x \mapsto 41; d \mapsto 23\}]\}$ The slice criterion requires that any trace in σ when projected onto the second step has a value for d, so $\omega = \{\{d \mapsto v\} \mid v \in V\}$ Therefore the smallest set of traces for which there exists a structural projection is $\sigma' = \{[\{d \mapsto v\} \mid v \in V\}]\}$.

GSF: Program Example2

Original	Criterion	Slice

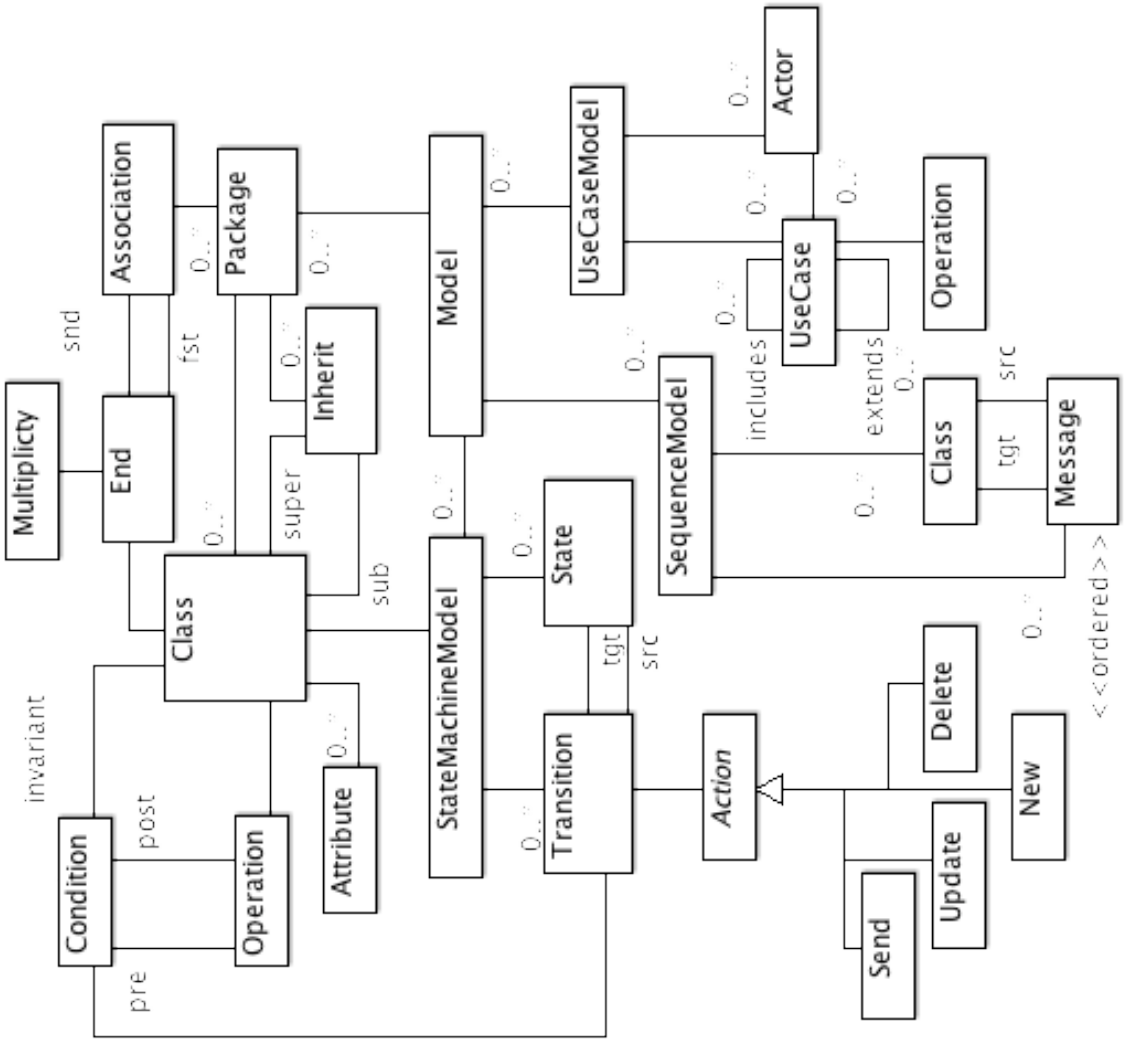
`sum=0`
`prod=1`
`read(i)`
`while(i<11) {`
`sum=sum+i`
`prod=prod*i`
`i=i+1`
`}`
	`prod`	`prod = 1`
		`read(i)`
		`while (i<11) {`
		`prod=prod*i`
		`i=i+1`
		`}`

Semantics is $\sigma = \{[\{\texttt{sum} \mapsto 0\}, \{\texttt{sum} \mapsto 0; \texttt{prod} \mapsto 1\}, \{\texttt{sum} \mapsto 0; \texttt{prod} \mapsto 1; i \mapsto v\}] + \texttt{trace}(v) \mid v \in 0 \ldots 10\}$] where $\texttt{trace}(0) = []$

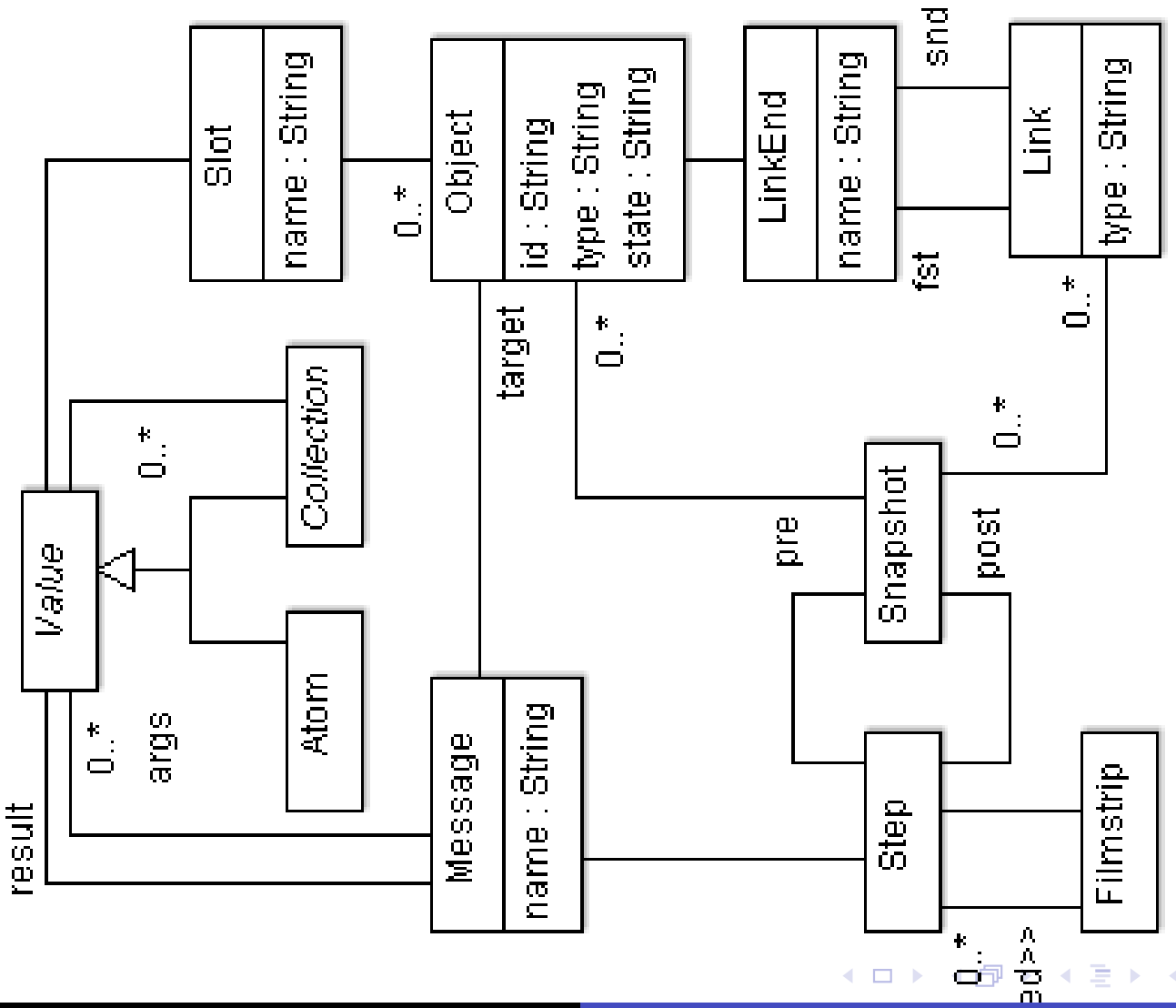
$$\texttt{trace}(v) = \texttt{trace}(v-1) + [\{\texttt{sum} \mapsto \sum_{i \in 0 \ldots v} i; \texttt{prod} \mapsto !v; i \mapsto v\}]$$

Criterion requires that $\omega = \{\texttt{prod} \mapsto v \mid v \in V\}$ and x maps each trace onto the *last* maplet for `prod`.

Modelling Language: Syntax

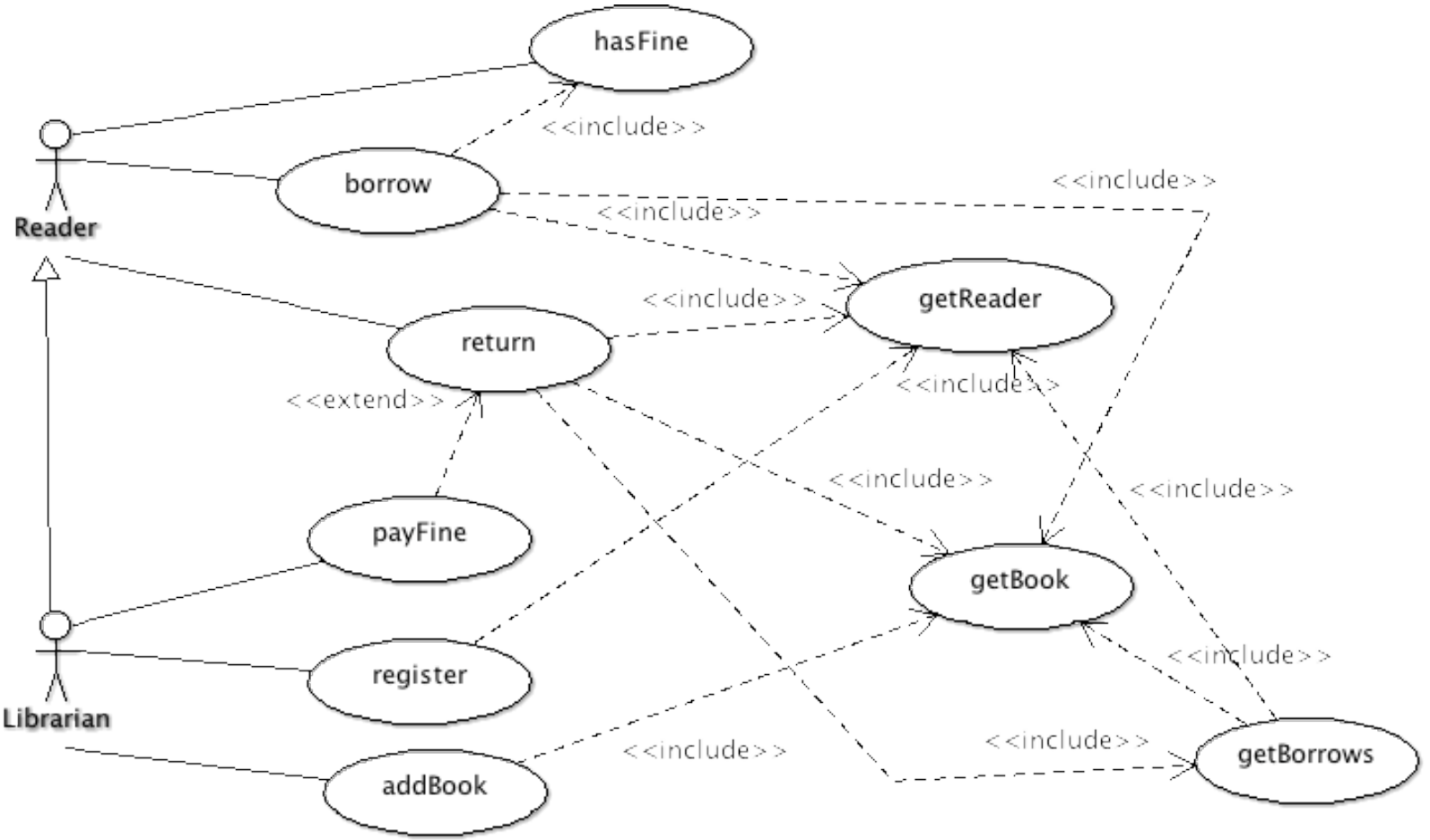


Modelling Language: Semantic Domain

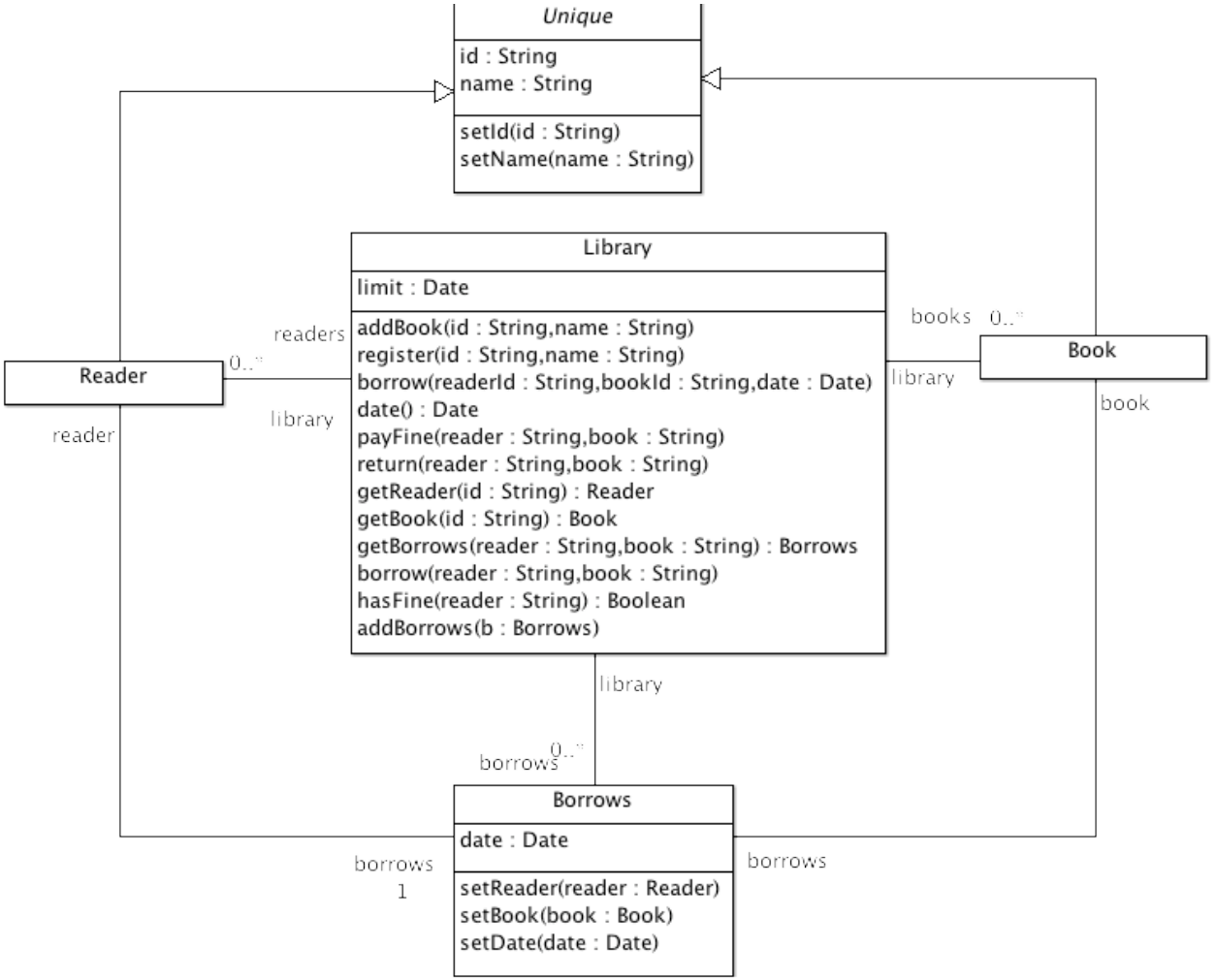


Contents

Case Study: Use Cases



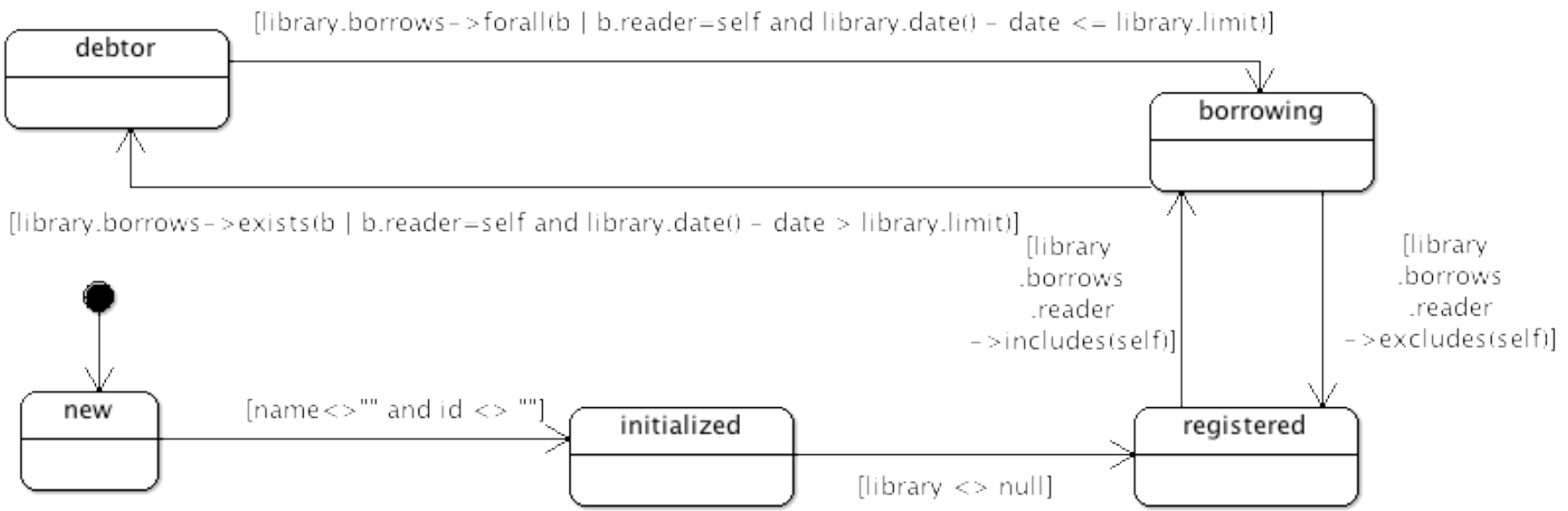
Case Study: Class Model



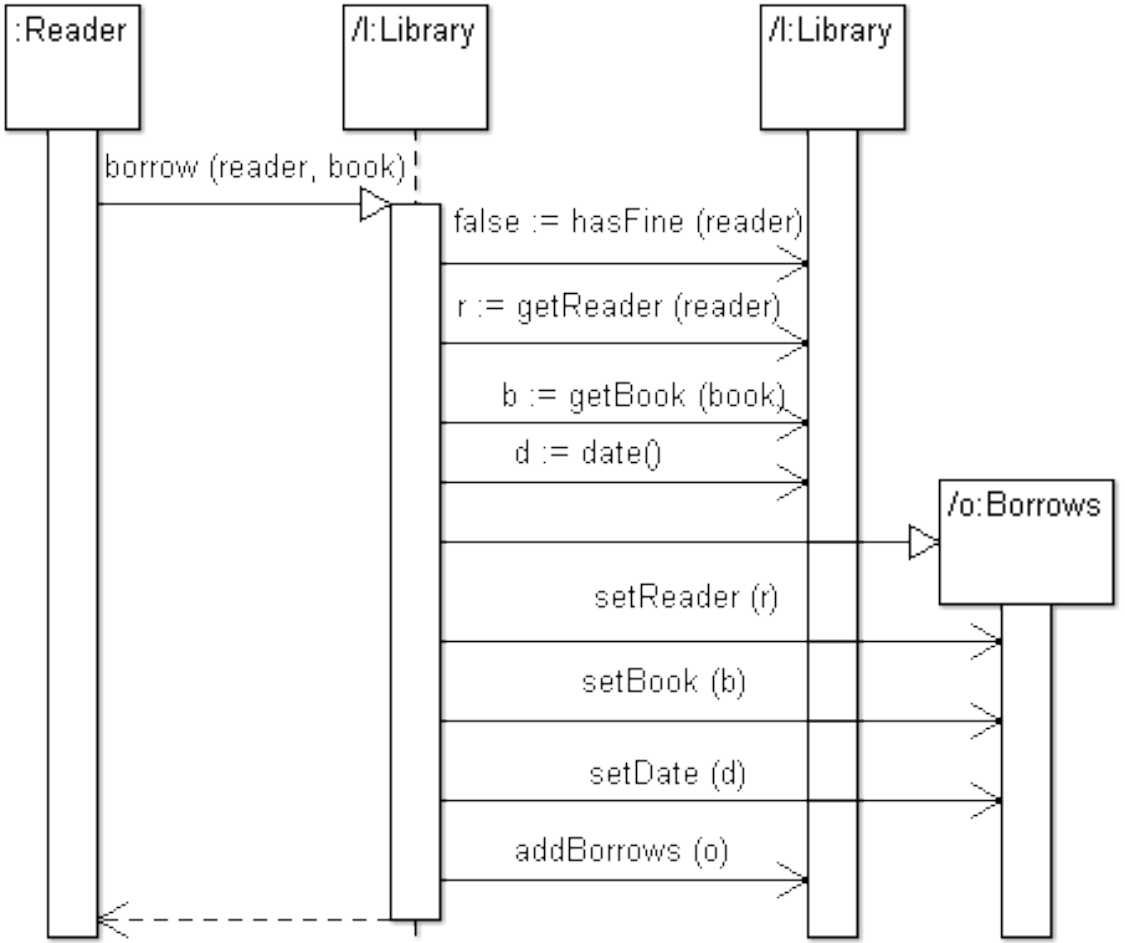
OCL

```
context Library::borrow(reader,book)
  pre not hasFine(reader) and
  post borrows->includes(b |
         b.reader = getReader(reader) and
         b.book = getBook(book) and
         b.date = date())
```

Case Study: State Machine



Case Study: Sequence Model



A Borrowing Step

```
(s1,ok = 0 <- borrow(fred,b),f,s6)
  where
    s1 = ({(0,*,Library)[limit=1],(1,*,Book)[name=n,id=1],
           (2,initialized,Reader)[name=fred,id=2]},{})
    f = [(s1,false = 0 <- hasFine(fred),...,s1),
         (s1,2 = 0 <- getReader(fred),...,s1),
         (s1,1 = 0 <- getBook(n),...,s1),
         (s1,d = 0 <- getDate(),...,s1),
         (s1,3 = 0 <- new(Borrows),[],s2),
         (s2,ok = 3 <- setReader(2),...,s4),
        ....]
    s2 = ({(0,*,Library)[limit=1],(1,*,Book)[name=n,id=1],
           (2,initialized,Reader)[name=fred,id=2],
           (3,*,Borrows)[]},{})
```

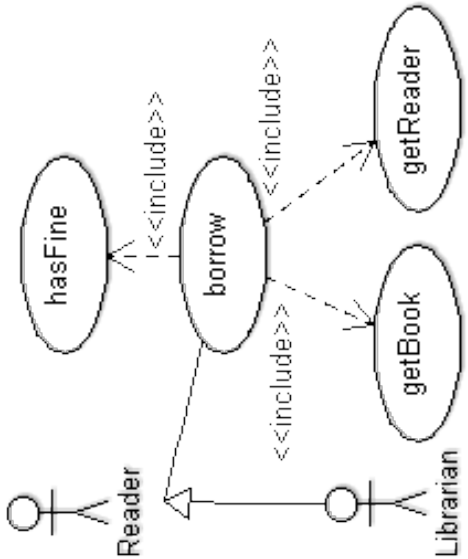
Slicing Criterion

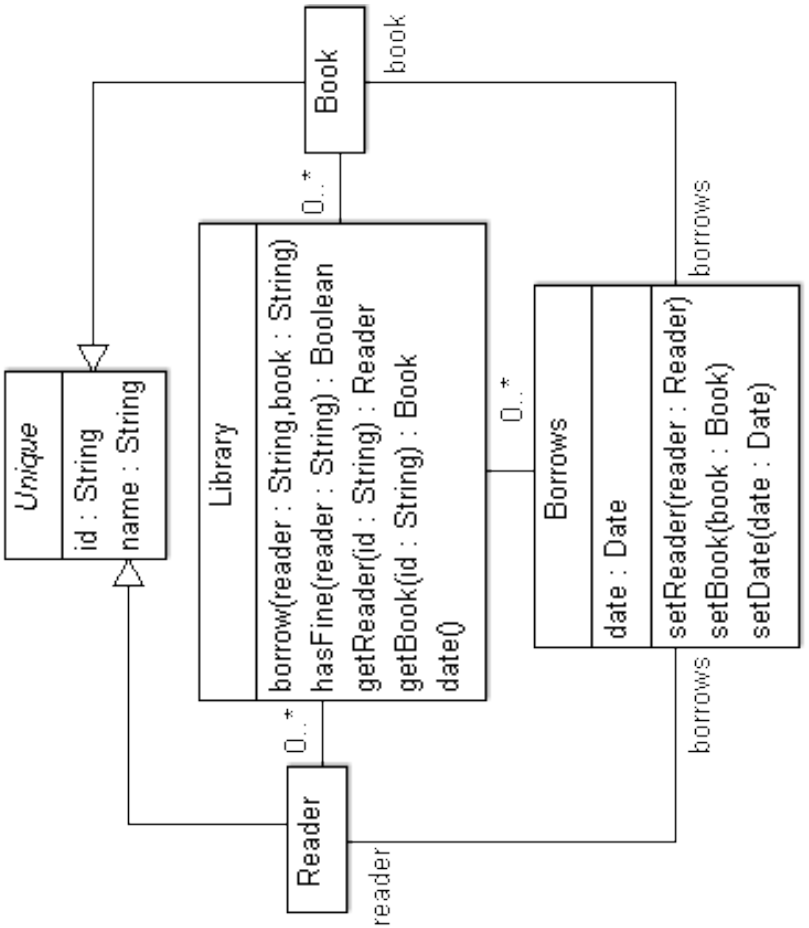
```
B={(s1,v = l <- borrow(r,b),F,s2) |
        F <- Filmstrip,
        s1 <- Snapshot,
        s2 <- Snapshot,
        v <- Value,
        l <- Id,
        r <- Str,
        b <- Str}
```

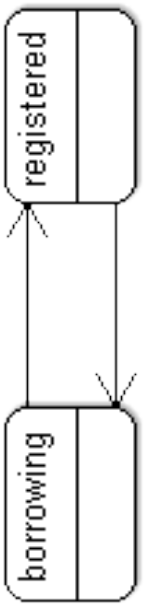
Use $\omega = B^*$ as basis for slicing criterion.

x_1 projects a filmstrip to the sequence of `borrow` steps it contains.

Sliced Model





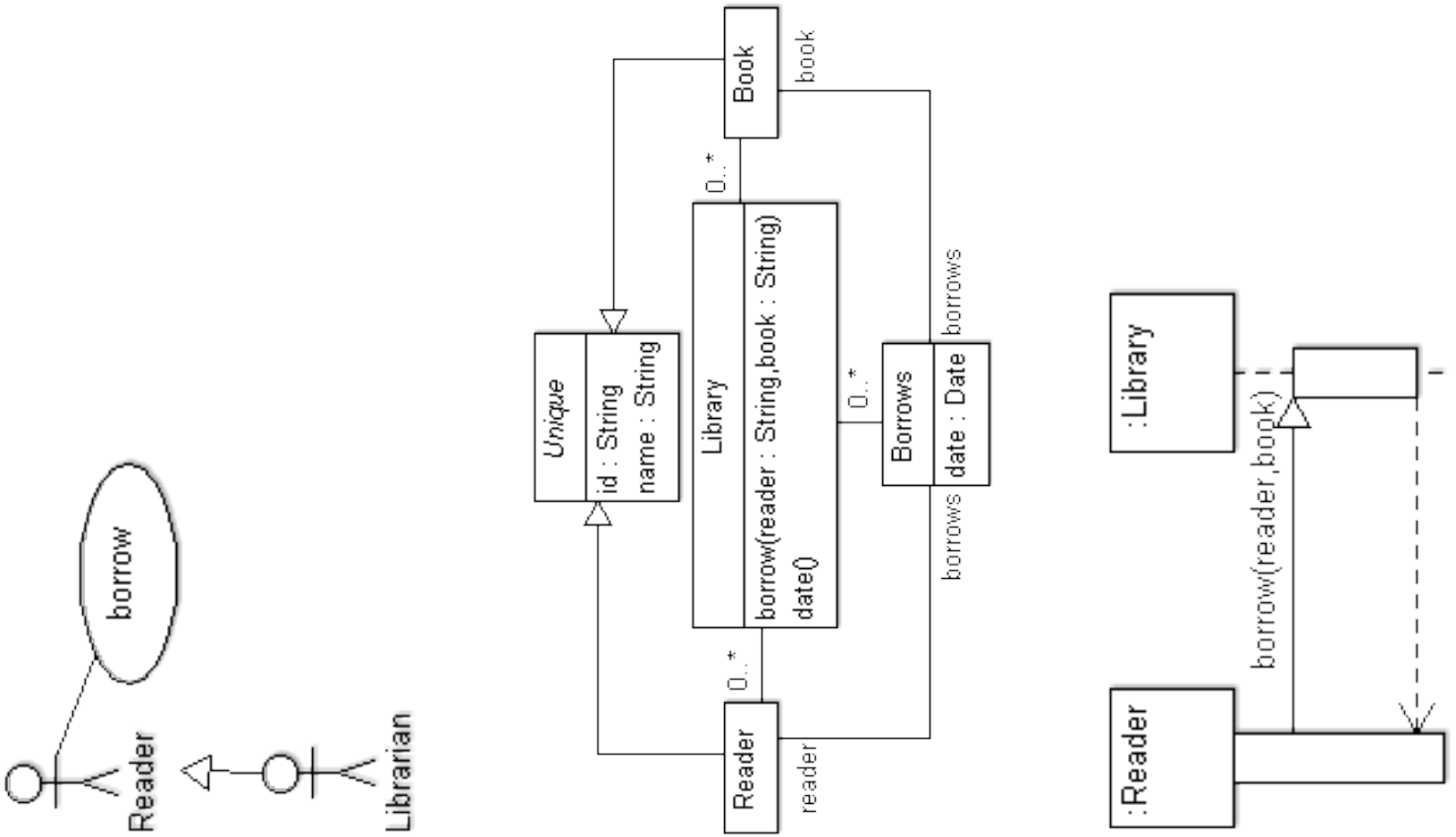


Refining the Slice

```
{[(s1,v = l <- borrow(r,b),[],s2)] |
     s1 <- Snapshot,
     s2 <- Snapshot,
     v <- Value,
     l <- Id,
     r <- Str,
     b <- Str}
```

The mapping part x_2 is a refinement of x_1 that maps all nested filmstrips to [].

Refined Slice



Contents

Issues and Problems

- Constructing slicing transformations: use of existing model transformation languages? New languages?
- Projection definition for OCL - requires theorem proving?
- Combination of slices? Consistency of slices?
- Slicing theories?
- Restricted forms of slicing: static, dynamic, conditioned?