Irixa Vales Torres  
INSO4101-060  
Homework #2

**8.1 Transportation Net Algebra**

**type**

Net, Segment, Connection, Location, Block, Conveyor

**values**

insert segment: Segment x Connection x Connection x Net -> Net

delete segment: Net x Segment -~-> Segment x Net

insert connection: Connection x Location x Net -> Net

delete connection: Net x Connection x Location -~-> Connection x Net

**axioms**

for all s: Segment, n: Net, c1, c2: Connection, l: Location

insert segment ((delete segment (n)(s)) (c­1)(c2)) = n

delete segment (insert segment (s)(c1)(c2)(n)) = n

insert connection (delete (connection (c1) (l) (n))) = n

delete connection (insert connection (c1) (l) (n)) = n

delete segment (empty Net) = chaos

delete connection (empty Net) = chaos

**8.2 Container Logistics Algebra**

**types**

Container Ship, Container, Container Storage Area, Quay, Bay, Row, Stack, Location

**values**

Load to ship: (Quay -> Container) x (Container Ship -> Row -> Stack) -> Stack

Unload from ship : (Container Ship -> Row -> Stack) x Quay -> Stack

Load to storage area : (Quay -> Container) x (Container Storage Area -> Stack) -> Stack

Unload from storage area : (Container Storage Area -> Stack) x Quay -> Stack x Quay

**axioms**

1. If you try to unload a container from an empty ship/storage area, it will create chaos.

2. If you try to load a container to a ship/storage area from an empty quay, it will create chaos.

3. If you load a container from a quay to a ship, which you unloaded from the ship to the quay, you will have the same initial stack.

4. If you unload a container from a ship to a quay, which you loaded from the quay to the ship, you will have the same initial stack.

5. If you load a container from a quay to a storage area, which you unloaded from the storage area to the quay, you will have the same initial stack.

6. If you unload a container from a storage area to a quay, which you loaded from the quay to the storage area, you will have the same initial stack.

**8.3 Financial Service System Algebra**

**types**

Customer, Bank Account, Bank, Bank Book, Money, Location

**values**

Open Account : Bank x Bank Book x Customer -> Bank Book x Account

Close Account : Bank x Bank Book x Account -> Bank Book

Establish Shared Accounts : Bank x Bank Book x Account x Customer x Customer -> Bank Book

Deposit Funds : Bank x Bank Book x Bank Account x Money -> Bank Account x Bank Book

Withdraw Funds : Bank x Bank Book x Bank Account x Money -> Bank Account x Bank Book

Transfer funds between accounts : Bank x Bank Book x Bank Account x Money x Bank Account -> Bank Book x Bank Account x Bank Account

**axioms**

1. If you close an account, which you just opened, you will have the same initial bank book.
2. If you try to close an account, which has not been opened, it will create chaos.
3. If you deposit funds, which you just withdrew, you will have the same initial balance.
4. If you withdraw funds, which you just deposited, you will have the same initial balance.
5. If you withdraw funds, which were just transferred to your account, you will have the same initial balance.
6. If you deposit funds and transfer those same funds to another account, you will have the same initial balance.

**9.1 Predicates over Transportation Net domain**

1.

**type**

Net, Segment, Connection

**value**

is\_visible: Segment x Connection -> Bool

is\_visible: Connection x Segment -> Bool

**axiom**

for all s, c, s: Segment, c: Connection

is\_visible(s, c) is\_visible(c, s) if from a segment a connection is visible, from that connection the segment is visible end

is\_visible(c, s) is\_visible(s, c) if from a connection a segment is visible, from that segment the connection is visible end

2. No

3.

**type**

Net, Segment, Connection

**value**

segment\_exists: Segment x Net -> Bool

connection\_exists: Connection x Net -> Bool

insert\_segment: Segment x Connection x Connection x Net -> Net

**axiom**

for all s: Segment, c1, c2: Connection, n: Net

insert\_segment(s, c1, c2, n)

pre segment\_exists(s) = false ^ (connection\_exists(c1) ^ connection\_exists(c2)) = true

post segment\_exists(s) = true

end

4.

**type**

Net, Segment, Connection

**value**

segment\_exists: Segment x Net -> Bool

connection\_exists: Connection x Net -> Bool

insert\_connection: Connection x Segment x Net -> Net

**axiom**

for all s: Segment, c: Connection, n: Net

insert\_connection(c, s, n)

pre connection\_exists(c) = false ^ segment\_exists(c) = true

post connection\_exists(c) = true

end

**9.2 Predicate over Container Logistics Domain**

**type**

Container Ship, Container, Container Storage Area, Quay, Bay, Row, Stack, Location, Height

**value**

max\_height: Bay -> Height

get\_height: Stack -> Height

**axiom**

for all bay: Bay, h: Height

for all r: Row, s: Stack, r bay

if get\_height(s) < max\_height(bay) then true

end