David Pick
CM 2403

I spent 3 hours on this assignment.

```
/*****************************************************************************
  This specification models an email client.

  by David Pick
*****************************************************************************/
module homework/hw06
open util/ordering[Time] as TO

/*****************************************************************************
  The system meta-model.
*****************************************************************************/
-- Time flies like an arrow; fruit flies like a banana
sig Time {}

-- Mailboxes are the primary grouping mechanism
abstract sig Mailbox {
  contents: (Message + Mailbox) -> Time
}
abstract sig SpecialMailbox extends Mailbox {}
one sig Inbox, Outbox, Sent, Drafts, Trash extends SpecialMailbox {}
sig CustomMailbox extends Mailbox {}

-- For abstraction, we ignore the message content
sig Message {
  from: Address,
  to: set Address,
  sent: lone Time   -- not all messages are necessarily sent
}

sig Address {}

-- BONUS: add filtering rules

/*****************************************************************************
  Constraints on system design.
*****************************************************************************/
fact SpecialMailboxRules {
  all smb: SpecialMailbox {
    -- Special mailboxes cannot contain other mailboxes
    no (Mailbox <: smb.contents)
    -- Special mailboxes cannot be contained in other mailboxes
    no Mailbox.contents[smb]
  }
}

-- Checks that a custom mailbox obeys the invariants at a given time.
pred CustomMailboxOK[cmb: CustomMailbox, t: Time] {
  -- no cycles
  cmb not in cmb.^(contents.t)
```

```
  -- at most one parent
  lone container[cmb, t]
}

-- Checks that a message obeys the invariants at a given time.
pred MessageOK[m: Message, t: Time] {
  lone container[m, t]
}


/*****************************************************************************
  Helper functions and predicates.
*****************************************************************************/
fun container[m: Message + Mailbox, t: Time]: set Mailbox {
  contents.t.m
}


/*****************************************************************************
  System events.
*****************************************************************************/
abstract sig Event {
  pre, post: Time
}

abstract sig MessageEvent extends Event {
  msg: Message
}

-- Transfers new message to Inbox
sig Receive extends MessageEvent {} {
  -- Message not in any of our mailboxes at start
  no container[msg, pre]
  -- Message must have been sent prior to being received
  some msg.sent and msg.sent in TO/prevs[pre] + pre
  -- Change in containment is restricted to just moving msg to Inbox
  contents.post = contents.pre + Inbox -> msg
}

-- Begins a new message in Drafts
sig Create extends MessageEvent {} {
-- Message doesn't exist when we start
no container[msg, pre]
-- Message must not have been sent yet
no msg.sent
-- No changes when we create the msg
contents.post = contents.pre + Drafts -> msg
}

-- Moves a message from Drafts to Outbox
sig PrepareToSend extends MessageEvent {} {
  container[msg, pre] = Drafts
  -- Message must not have been sent yet
  no msg.sent or msg.sent in TO/nexts[post]
```

```
  -- Change in containment is restricted to just moving msg to Outbox
  contents.post = contents.pre - (Drafts -> msg) + (Outbox -> msg)
}

-- Sends a message from Outbox, files it in Sent
sig Send extends MessageEvent {} {
-- Message starts in the outbox
container[msg, pre] = Outbox
-- Message has been sent already
msg.sent in TO/prevs[pre] + pre
-- Restrict the change in containment to just moving msg to Sent
contents.post = contents.pre - (Outbox -> msg) + (Sent -> msg)
}

-- Moves a message from one of our mailboxes to another.
-- Cannot move a message to Outbox, Sent, or Drafts.
sig FileMessage extends MessageEvent {
  dest: (Mailbox - Outbox - Sent - Drafts)
} {
one preContainer: container[msg, pre] {
-- move message from previous container to new one
contents.post = contents.pre - (preContainer -> msg) + (dest -> msg)
}
}

-- Moves a message from one of our mailboxes to the Trash.
sig TrashMessage extends FileMessage {} {
--Set the messages container to Trash
dest = Trash
}

-- Rearranges the custom mailbox hierarchy
sig Rearrange extends Event {
  box: CustomMailbox,
  dest: CustomMailbox
} {
--dest is not in box or its children
dest not in box.*(contents.pre)

-- Change the boxes around
contents.post = contents.pre - ((contents.pre).box -> box) + (dest -> box)
}


/*****************************************************************************
  Constraints on Time.
*****************************************************************************/
pred init[t: Time] {
  -- System is initialized with no messages and no custom mailboxes,
  -- where "no" is represented by having no container.
  all cmb: CustomMailbox | no container[cmb, t]
  all msg: Message | no container[msg, t]
}
```

```
fact Traces {
  init[TO/first[]]

  all t: Time - TO/last[] | let t' = TO/next[t] |
    some e: Event {
      -- constrains event to one time step
      e.pre = t and e.post = t'
      -- restricts the changes that can occur to a particular category
      -- of event
      (contents.t :> Message) != (contents.t' :> Message)
        => e in MessageEvent
      (contents.t :> CustomMailbox) != (contents.t' :> CustomMailbox)
        => e in Rearrange
    }

}


/*****************************************************************************
  Predicates for visualizing the system.
*****************************************************************************/
pred show[] {}
run show for 3 but exactly 8 Mailbox, exactly 3 Message, exactly 2 Event

pred showEvent[] {
}
run showEvent for 2 but exactly 8 Mailbox, exactly 1 Receive
run showEvent for 2 but exactly 8 Mailbox, exactly 1 Create
run showEvent for 3 but exactly 8 Mailbox, exactly 1 Create,
  exactly 1 PrepareToSend
run showEvent for 4 but exactly 8 Mailbox, exactly 1 Create,
  exactly 1 PrepareToSend, exactly 1 Send
run showEvent for 3 but exactly 8 Mailbox, exactly 1 Receive,
  exactly 1 FileMessage
run showEvent for 3 but exactly 8 Mailbox, exactly 1 Create,
  exactly 1 TrashMessage
run showEvent for 2 but exactly 8 Mailbox, exactly 1 Rearrange

pred showTrace[] {
}
run showTrace for 8 but exactly 8 Mailbox, exactly 8 Time


/*****************************************************************************
  Assertions for checking the system.
*****************************************************************************/
assert NoNestingInSpecial {
  all mb: SpecialMailbox |
    no (Mailbox <: mb.contents)
}
check NoNestingInSpecial for 6

assert NoTwoParents {
  all t: Time |
```

```
    no disj mb, mb', mb'': Mailbox |
      mb'' in mb.contents.t and mb'' in mb'.contents.t
}
check NoTwoParents for 8

assert CustomMailboxesOK {
-- For all Custom mailboxes ensure that the custom mailbox
-- predicate holds for any initial time and then the time
-- directly following
  all cmb: CustomMailbox, t, t': Time |
t' = next[t] and
CustomMailboxOK[cmb, t] implies
CustomMailboxOK[cmb, t']
}
check CustomMailboxesOK for 6 but 2 Address, 8 Mailbox, 8 Time, 7 Event

assert MessagesOK {
-- For all messages ensure that the message
-- predicate holds for any initial time and then the time
-- directly following
  all m: Message, t, t': Time |
t' = next[t] and
MessageOK[m, t] implies
MessageOK[m, t']
}
check MessagesOK for 5 but 2 Address, 8 Mailbox, 6 Time, 5 Event
```