U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mini Project 1: Alloy

April 2025

Group 2

Afonso Osório - 202108700
Pedro Madureira - 202108866
Sofia Pinto - 202108682

1. Introduction

In this report, we present the enhancement made to the original mail system model by introducing support for spam handling. This addition implies the creation of new signatures, fields, operators and more. This way, we believe our model more accurately reflects the reality of the mail as we know it.

2. Model and Signatures

Firstly, a new mailbox, *spam*, was added to the *Mail* signature. This mailbox is intended to store messages identified as spam.

```
one sig Mail {
  -- system mailboxes
  inbox: Mailbox,
  drafts: Mailbox,
  trash: Mailbox,
  sent: Mailbox,
  spam: Mailbox,
  -- user mailboxes
  var uboxes: set Mailbox,

  userAddress: one Address,

  var op: lone Operator -- added for tracking purposes only
}
```

Figure 1: *Mail* signature

Additionally, a *userAddress* was also added to the *Mail* signature. This directly corr
elates to the new signature *Address* that was created, which will ease the process of identifying spam addresses.

```
sig Address extends Object {}
```

Figure 2: *Address* signature

Furthermore, the *SpamFilter* signature was introduced to track the set of addresses marked as spammers.

```
one sig SpamFilter {
  var spammers: set Address
}
```

Figure 3: *SpamFilter* signature

Moreover, we added an *address* field to the *Message* signature that represents the address of who sent that particular message (sender).

```
sig Message extends Object {
  var status: lone Status,
  address: Address
}
```

Figure 4: *Message* signature

Finally,we also made a couple of additional small changes: the *sboxes* function includes the *spam* mailbox, and the *Init* predicate forces it to be distinct from the rest of the system mailboxes. It also forces the address of *Fresh* messages to be the user's address, and the address of *External* messages to be different from that of the user.

3. Operators

The creation of the previous signatures and fields, lead us to come up with two new operators. These provide the functionality of adding and removing addresses from the *SpamFilter*. We named them *addToFilter* (AS) and *removeFromFilter* (RS), respectively.

***addToFilter*:** Adds an address to the list of *spammers* in the *SpamFilter* signature. Notice, however, that this address must differ from the *userAddress* and must not be in the list of *spammers* already. Additionally, this operator moves all existing messages from this address to the *spam* mailbox, unless they were already deleted (i.e. in the *trash* mailbox)

***removeFromFilter*:** Removes an address from the list of *spammers* in the *SpamFilter* signature. Notice that this address must be in the list of *spammers* already. Additionally, this operator moves all existing messages containing this address in the *spam* mailbox to the *inbox* mailbox.

We also made some changes to the existing operators. *moveMessage* has a new precondition, stating that a message cannot be directly moved to the spam mailbox. The behavior of *getMessage* was split into two cases: the first case, in which the incoming message's address is not in the spam filter, behaves identically to the original operator; the second case, in which the address is in the filter, places the message in the *spam* mailbox instead of the *inbox*.

4. Sanity Checks

We have added some key behaviors that should eventually occur.

Firstly, at some point, an address must be added to the *SpamFilter*. Likewise, addresses can also be removed from the filter. As the *SpamFilter* evolves, messages from newly flagged addresses are automatically redirected to the spam mailbox without user intervention.

On the other hand, the system also supports explicit user actions: messages can be manually removed from the spam folder, either by moving them or deleting them. Finally, when an address is removed from the spam filter, any messages from that address are implicitly cleared from the *spam* mailbox.

5. Valid Properties

The system ensures that spam handling behaves consistently with user expectations and defined rules. Any message currently found in the *spam* mailbox must originate from an address listed in the spam filter. Additionally, when a new message arrives, if it is sent from

an address contained in the *SpamFilter*, it is always directed to the *spam* mailbox, while messages from addresses not in the *SpamFilter* are delivered to the *inbox*. Note that this means that one of the properties of the original model stops being applicable: with the addition of spam, not all incoming messages pass through the inbox. The only scenario where such messages can appear outside the *spam* mailbox is if the user has explicitly moved or deleted them. Lastly, to avoid self-flagging, the spam filter is never allowed to contain the user's own email address.
Barring the previously mentioned case, all properties of the original system still hold. However, for some properties, we had to tighten the limits on the scopes, since all the new possible configuration possibilities permitted by our additions made execution times increase considerably.

6. Invalid Properties

Finally, we created a new invalid property which, at first glance, might seem impossible: “An active message can remain in its mailbox when its address is added to the *spamFilter*”. However, this behaviour is indeed possible if the given message is already in the trash. Below, we can see an example of execution that exemplifies this behaviour, achieved by negating this property and using it as an assertion in order to produce a counterexample:

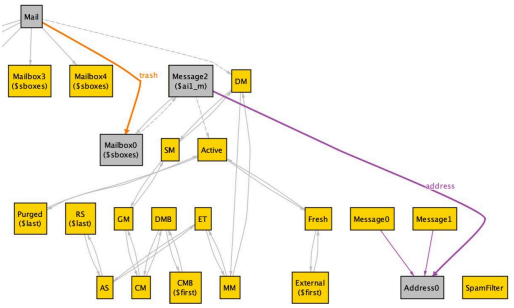

Figure 5: Before addition

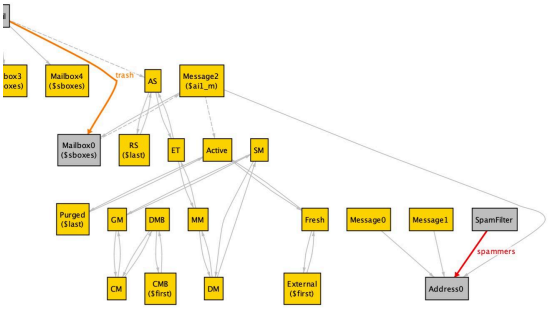

Figure 6:After addition

As one can see, in Figure 5, *Message2* contains *Address0* and is currently in the *trash* mailbox. Nevertheless, in the next step (Figure 6), when *Address0* is added to the *SpamFilter*, *Message2* remains in the *trash*, instead of being moved to the spam mailbox, as it would have happened if the message was in any other mailbox.
All other properties that were addressed in this way in the original model still behave as expected.