XII. Routing Algorithm for a Mobile Agent (October 2008)

The purpose of the example developed in this chapter is to present an interesting routing algorithm for sending messages to a mobile phone. In this example, we shall again encounter a tree structure as in the previous chapter (Synchronizing Processes on a Tree Structured Network), but this time the tree structure will be modified dynamically. We shall also encounter another example (besides the Bounded Re-transmission Protocol in chapter 6) where the usage of clocks will play a fundamental role. This example is taken from [1].

1 Informal Description of the Problem

A, so-called, *mobile agent* $\mathcal{M}$ is supposed to travel between various sites. Fixed agents situated in the sites in question want to establish some communications with it. To simplify matters, such communications are supposed to be unidirectional: they take the practical form of messages sent from the fixed agents to $\mathcal{M}$.

1.1 Abstract Informal Specification.

In an ideal *abstract* world, the moves of the mobile agent $\mathcal{M}$ from one site to another are instantaneous. Likewise, the knowledge by the fixed agents of the exact position of $\mathcal{M}$ is also supposed to be instantaneous. In that case, the fixed agents follow the mobile agent $\mathcal{M}$ by sending messages where it currently is. Notice that such messages are (for the moment) received immediately by $\mathcal{M}$. This is illustrated on figure 1 where the mobile agent $\mathcal{M}$ (represented by a black square) originally situated at site c, moves then successively to sites d, a, c, and b. The arrow indicates where each fixed agent are sending messages: they just follow $\mathcal{M}$ since they immediately know where it is.

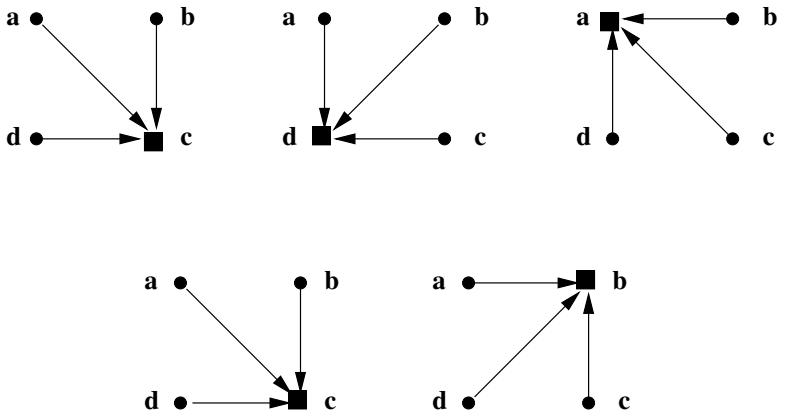


Fig. 1. First Abstraction: The Fixed Agents Always Know where the Mobile Agent is

1.2 First Informal Refinement.

In a more realistic *concrete* world, the moves of $\mathcal{M}$ from one site to another are still instantaneous, but the only site to know where $\mathcal{M}$ is, is the site that $\mathcal{M}$ has just left. The other sites are not aware of the move, they continue thus to send messages to the site where they still *believe* that $\mathcal{M}$ resides. Then it is quite possible that some messages arrive at a destination which is not currently that of $\mathcal{M}$. As a consequence, each site, besides sending its own messages, is thus also in charge of possibly *forwarding* the messages received while $\mathcal{M}$ is not present any more locally. It is quite possible that several such intermediate transmissions take place before a message eventually reaches $\mathcal{M}$. This is illustrated on figure 2.

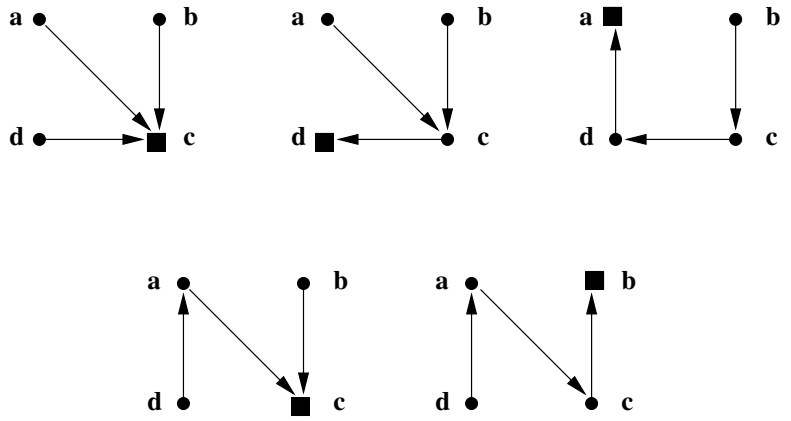


Fig. 2. First Refinement: Fixed Agents do not Know the Exact Position of the Mobile Agent

As can be seen, when $\mathcal{M}$ reaches a new site, that site destroys the previous knowledge it has concerning the location of $\mathcal{M}$. For instance, in the third snapshot, where $\mathcal{M}$ has just moved to site a coming from site d, the link between a and c that existed in the previous situation is removed. Similarly, when $\mathcal{M}$ leaves a site, that site re-actualizes its knowledge by storing the new location of $\mathcal{M}$ (again, supposed to be known instantaneously). For instance, in the fourth snapshot above, where $\mathcal{M}$ has just moved from a to c, a new link between a and c is established.

Intuitively for the moment, we can figure out that the communication channels are dynamically modified while maintaining a *tree structure* whose root is the actual site of $\mathcal{M}$. This is illustrated on figure 3 where we have reordered the sites to show more clearly the tree structure. Each site is then indirectly *connected* to the site of $\mathcal{M}$ and there exists *no cycles* that might put some forwarded messages in an endless loop.

1.3 Second Informal Refinement: a Problem.

In a still more realistic world, the moves of $\mathcal{M}$ between sites are not instantaneous anymore. In fact, when $\mathcal{M}$ leaves a site, it does not know necessarily where it is going. Only when $\mathcal{M}$ arrives at its destination, is it able to send a *service message* to its previous site in order to inform it of its present location. Of course, the service message in question does not itself travel instantaneously. Communication messages are thus still forwarded from sites to sites, but that forwarding might be *suspended* in some sites, which $\mathcal{M}$ has left in the past, until such sites receive service messages informing them of the "present" location of $\mathcal{M}$ (present, however, when the service message was sent, maybe not any more when it is received).

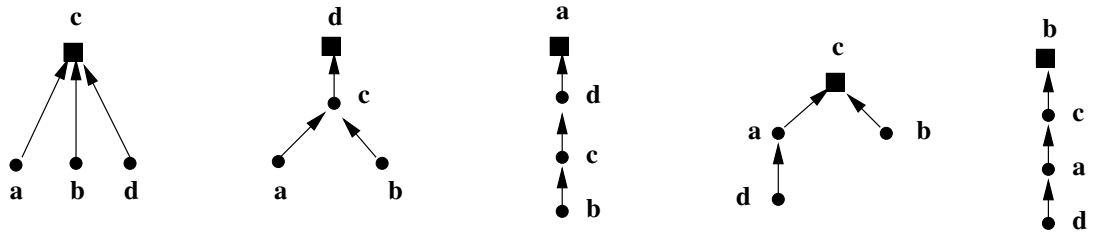


Fig. 3. The Tree Structure

We have no control over the relative speed of the service messages: some of them can reach their destination quite quickly, while some others might take more time (but we suppose that they will eventually arrive at their destination). On figure 4, we have put some dashed lines to indicate that the corresponding service messages have not yet arrived: notice that service messages following the dashed lines circulate in a direction which is the opposite of that followed by the communication messages that will be established upon reception of the service message. In fact, when a service message arrives at its destination, the corresponding dashed line is transformed in a "plain" line going in the opposite direction.

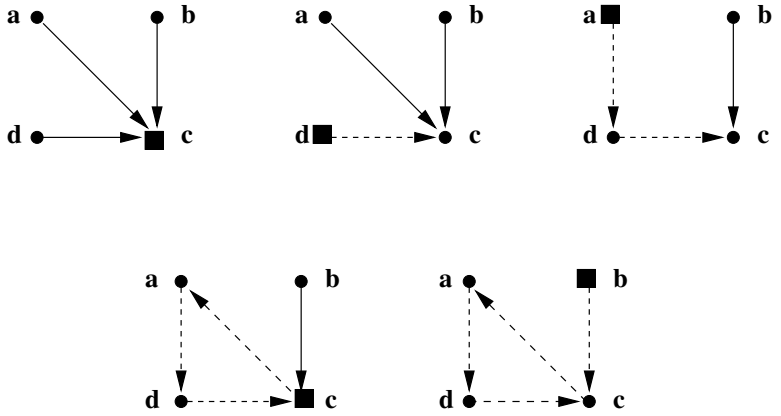


Fig. 4. Second Refinement: The Mobile Agent sends a Service Message with its new Position

On figure 4, we have shown a series of snapshots where all service messages are pending. Notice that the situation pictured in the last snapshot contains a potential problem. This is because site c is expected to receive two service messages, one from d and another one from b. As a matter of fact, a site might expect as many service messages as there has been past visits of that site by the mobile agent $\mathcal{M}$.

On figure 5, we show various snapshots corresponding to the arrival of some service messages. The last snapshot shows a situation were all service messages have reached their destination except the ones, $sm1$ and $sm2$ supposed to reach site c form d and b respectively.

If the service message $sm1$ between d and c is very late (arriving after the service message $sm2$ between b and c although sent before $sm2$), then, upon arrival, $sm1$ may have the disastrous effect of: (1) isolating completely site b, and (2) forming a cycle within which communication messages may circulate for ever. This is illustrated figure 6 where the two snapshots shows the arrival of service message $sm2$ followed by that of service message $sm1$.

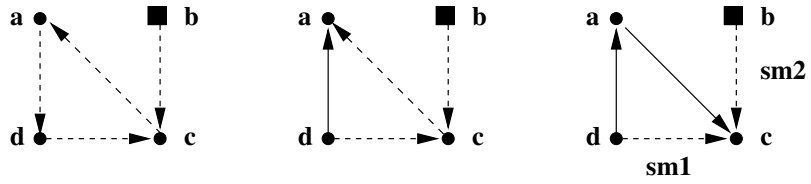


Fig. 5. Service Messages Messages sm2 from b to c and sm1 from d to c have not yet arrived

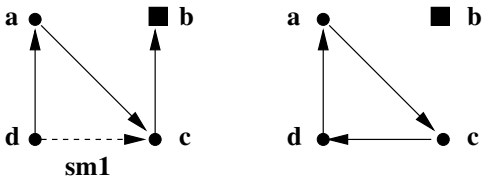


Fig. 6. Arrival of Service Message sm2 before sm1

This failure is due to the fact that site c is mislead by service message $sm1$ from site d. In fact, message $sm1$ should have been *discarded* by site b when sending service message $sm2$ to c. But how can site b know about the existence of such a pending service message $sm1$ whose destination is also c?

1.4 Third Informal Refinement: the Solution.

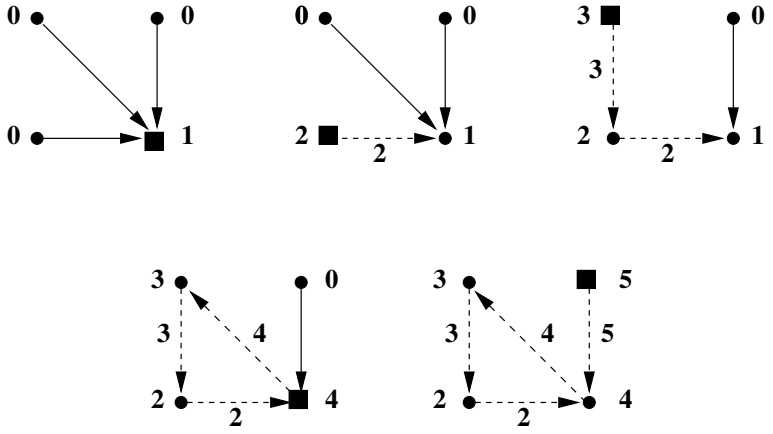


Fig. 7. Introducing the clock

The purpose of the distributed routing algorithm presented here and developed by L. Moreau in [1] is precisely to solve the potential problem we discovered in previous section. The idea is to have the mobile agent $\mathcal{M}$ travelling with a *logical clock* which is incremented each time it arrives to a new site. Upon arrival in a site, the value of the logical clock is stored (after being incremented). It thus records the time of the last visit of the mobile $\mathcal{M}$ to this site. When $\mathcal{M}$ sends its service message to its previous site, it

stamps that message with the new time (the one that has just been incremented and recorded in the new site). This is illustrated on figure 7 where the local time stored in each site can be seen next to each of them. The stamp value on the service messages are shown next to the middle of the corresponding arrow (they are all equal as expected to the value of the clock stored at the origin of the message).

As a consequence, a new service message is stamped with a value that is certainly greater than that recorded in its destination. When the service message arrives, it is filtered: if the stamp is smaller than or equal to the local time of the destination then it is discarded because it is clearly a late service message. If the stamp of a service message is greater than the local time, then the message is accepted and, simultaneously, the local clock of the site is updated with the value of the stamp travelling with the service message. Thanks to this, the message could not be used a second time (in case of misbehavior of the network). Figure 8 shows the series of situations corresponding to the arrival of the service messages. We have decorated these situations with the clocks and the stamps.

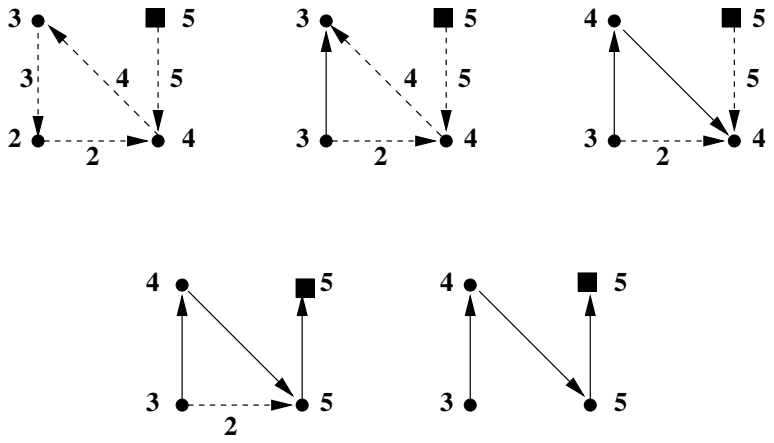


Fig. 8.

As can be seen, the last service message is discarded because its stamp value, 2, is smaller than the local clock, 5, at destination. We have reversed from the potential failure presented earlier. This is due to the presence of the clock and stamps, and of the particular adopted strategy.

The system we have described seems to work, at least according to our informal explanations. But it is certainly important to develop it formally so as to be sure that it behaves in a correct fashion. This is the purpose of the coming sections.

2 Initial Model

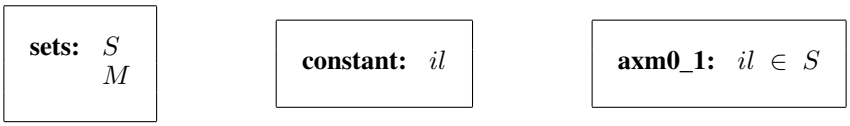
Now, we have enough information to start the formal construction of this routing protocol. The first question that we must ask ourself with an example such as this one (and many others) is that concerning the *level of description* we have to start from. It is out of the question to start from the final solution, because then nothing really can be proved. We must start from an abstract enough level, which must be pretty obvious (and where no technical difficulties exist yet, in particular those dealing with time and distances), so that the proposed solution can be proved to indeed solve the problem that has been informally described.

In the present case, we are going to start from the second abstract informal level described in section 1.2, where the exact position of the mobile is only known by its previous site. This level is quite simple:

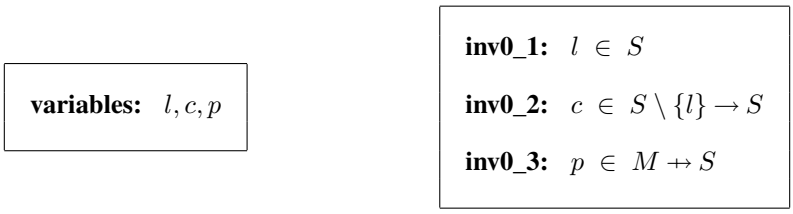
the communication channels, as we have said informally, form a tree structure (an invariant that owes to be proved, of course), the moves of the mobile are timeless, and finally the knowledge concerning the new location of the mobile is instantaneously communicated to its previous site (no service message thus). Further refinements will introduce more realistic constraints.

2.1 The State

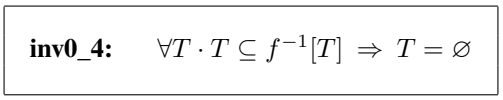
We have two carrier sets S and M: the set S denotes the set of sites and M the set of communication messages. To have a carrier set representing the set of communication messages is really a useful abstraction since we are not interested in the contents of these messages: we can then suppose that they are all distinct. Initially the mobile is at some *initial location* denoted by the constant il (**axm0_1**):



We have three variables in our initial model: l, c, and p. The variable l denotes the actual location of the mobile agent (**inv0_1**). The variable c denotes the dynamically changing communication channels between sites: it is a total function from sites to sites (**inv0_2**). Notice that this function is obviously not meaningful at l. Finally the variable p denotes the pool of messages that are waiting to be forwarded on each site: clearly a given message is at most in one site at a time, so that p is a partial function from messages to sites (**inv0_3**). Formally:



It remains now for us to formalize the tree structure of the communication channels. The root of the tree is l and the parent function is the function c. We shall use exactly the same formulation as the one introduced in chapter 9 (section 7.7), namely **inv0_4**:



We can see the difference with the previous chapter: in this one, this statement is an invariant, not an axiom.

2.2 The Events

We have four events besides event init which shows that all nodes are pointing to the initial position of the mobile $\mathcal{M}$. Event rcv_agt corresponds to the mobile moving instantaneously from the site l to another one (different from l). Event snd_msg corresponds to a new communication message m sent from one site s to the mobile. Event fwd_msg corresponds to a communication message m being forwarded from one site s to another one by means of the corresponding channel (note that this transfer is also instantaneous for the moment). And finally event dlv_msg corresponds to a communication message m being delivered to the mobile. Here are these events:

init
$l := il$
$c := (S \setminus \{l\}) \times \{il\}$
$p := \varnothing$

rcv_agt
any s **where**
$s \in S \setminus \{l\}$
then
$l := s$
$c := (\{s\} \lhd c) \cup \{l \mapsto s\}$
end

snd_msg
any s, m **where**
$s \in S$
$m \in M \setminus \mathrm{dom}(p)$
then
$p(m) := s$
end

fwd_msg
any m **where**
$m \in \mathrm{dom}(p)$
$p(m) \neq l$
then
$p(m) := c(p(m))$
end

dlv_msg
any m **where**
$m \in \mathrm{dom}(p)$
$p(m) = l$
then
$p := \{m\} \lhd p$
end

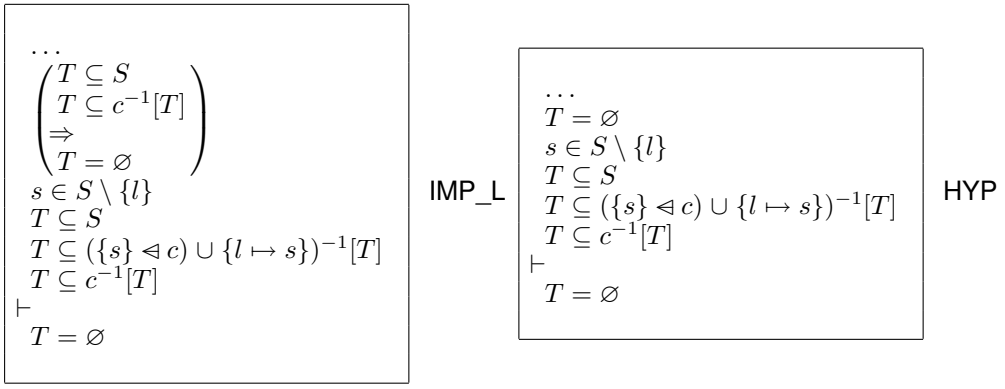
2.3 The Proofs

The only interesting proof at this level is that of the preservation of invariant **inv0_4** by event rcv_agt. Here is what we have to prove:

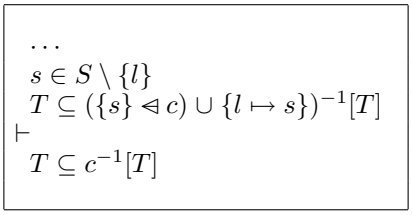
$$\begin{array}{ll} \ldots & \\ \text{Invariant } \textbf{inv0\_4} & \\ \text{Guard of rcv\_agt} & \\ \vdash & \\ \text{Modified Invariant } \textbf{inv0\_4} & \end{array} \qquad \boxed{\begin{array}{l} \ldots \\ \left(\begin{array}{l} \forall T \cdot \ T \subseteq S \\ \qquad\quad T \subseteq c^{-1}[T] \\ \qquad\quad \Rightarrow \\ \qquad\quad T = \varnothing \end{array}\right) \\ s \in S \setminus \{l\} \\ \vdash \\ \left(\begin{array}{l} (\forall T \cdot \ T \subseteq S \\ \qquad\quad T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\ \qquad\quad \Rightarrow \\ \qquad\quad T = \varnothing \end{array}\right) \end{array}}$$

We can remove the universal quantification in the consequent of this implication by using rules ALL_R and IMP_R. The proof proceeds then as follows:

$$\boxed{\begin{array}{l} \ldots \\ \left(\begin{array}{l} \forall T \cdot \ T \subseteq S \\ \qquad\quad T \subseteq c^{-1}[T] \\ \qquad\quad \Rightarrow \\ \qquad\quad T = \varnothing \end{array}\right) \\ s \in S \setminus \{l\} \\ T \subseteq S \\ T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\ \vdash \\ T = \varnothing \end{array}} \quad \text{ALL\_L} \quad \boxed{\begin{array}{l} \ldots \\ \left(\begin{array}{l} T \subseteq S \\ T \subseteq c^{-1}[T] \\ \Rightarrow \\ T = \varnothing \end{array}\right) \\ s \in S \setminus \{l\} \\ T \subseteq S \\ T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T] \\ \vdash \\ T = \varnothing \end{array}} \quad \text{SET} \ \ldots$$



The key of this proof is the following lemma whose proof is sketched below.



Consider two cases successively $s \notin T$ and $s \in T$. In the first case, $T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T]$ reduces to $T \subseteq (c^{-1} \rhd \{s\})[T]$, hence $T \subseteq c^{-1}[T]$. In the second case, $T \subseteq (\{s\} \lhd c) \cup \{l \mapsto s\})^{-1}[T]$ reduces to $T \subseteq (c^{-1} \rhd \{s\})[T] \cup \{l\}$ which contradicts $s \in T$ since $s \notin (c^{-1} \rhd \{s\})[T]$ and $s \neq l$.

3 First Refinement

In this first refinement, we are more concrete. The movements of the mobile will not be instantaneous any more, it will be made in two steps: first the mobile leaves the site l (new event leave_agt), and then the mobile arrives at a new site s different from l (old event rcv_agt).

In that second case, the knowledge by the previous site l of the new position s of the mobile will not be instantaneous any more as was the case in the initial model. In fact, as we said in section 1.3, the mobile will send a *service message* to its previous site s, in order for that site to update its forward pointer to the new site of the mobile. But during the travelling delay of the mobile and then the transmission delay of the service message, site s cannot transmit any forward message because it does not know where the agent is. Site s only knows that it is expecting a service message. The site corresponding to the previous position of the mobile $\mathcal{M}$ will eventually receive the service message (new event rcv_srv).

3.1 The State

Clearly, the channel structure in this new refined model is not in phase with that of the previous model where it was modified when the mobile agent arrived at its new destination: this was done by event rcv_agt (in the previous model, we had no travelling time and no transmission of a service message). In this refinement, we have thus to add a new variable, d, denoting this new channel structure (**inv1_1**), which will only be updated as a consequence of the new event rcv_srv.

We have a new variable, a, denoting the service channel containing the service messages mentioned in the informal description. It is a partial function from sites to sites (**inv1_2**), more precisely, from the site

where the mobile was before moving to the site *where it currently is*. This function contains the *future* of the communication channel. We shall make precise in what follows the reason why a is such a *function*, which is far from being obvious a priori: we shall see that, in this abstraction, this channel has a rather magic behavior.

The next invariant, **inv1_3**, establishes the connection between the abstract communication channel c and its concrete counterpart d. It says that the abstract channel c corresponds to the concrete one, d, *overridden* by the service channel a, where we have removed a potential link from l corresponding to a service message still travelling to l. Formally:

variables:	l
	p
	d
	a
	da

inv1_1:	$d \in S \setminus \{l\} \nrightarrow S$
inv1_2:	$a \in S \nrightarrow S$
inv1_3:	$c = d \Leftarrow (\{l\} \lhd a)$

We introduce another variable da which is a subset of sites (**inv1_4**). Variable da records the sites at which the current forwarding direction for information messages is not meaningful any more. This is because the mobile has left these sites while none of them has received yet the expected service message. For these reasons, the sites of da cannot forward any messages. It is claimed in the next invariant (**inv1_5**) that the domain of the acknowledgment message a extended with the singleton $\{l\}$ is equal to the set da also extended with the singleton {l}:

inv1_4:	$da \subseteq S$
inv1_5:	$\mathrm{dom}(a) \cup \{l\} = da \cup \{l\}$

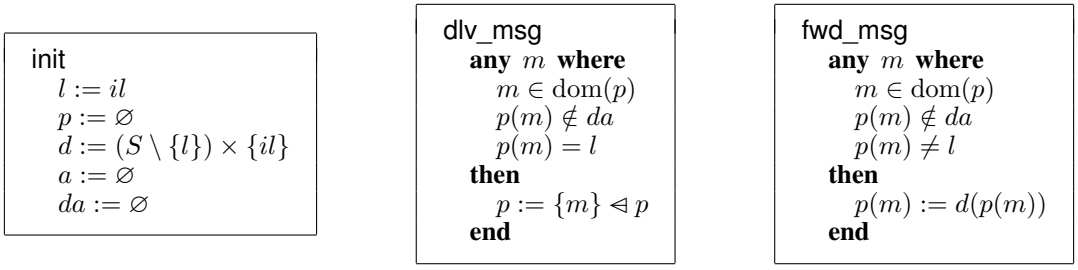
Invariant **inv1_5** might seem a bit strange: why not having just $\mathrm{dom}(a) = da$? This is due to two reasons, which are the following:

1. Sometimes variable l does not denote the site where $\mathcal{M}$ is. More precisely, when $\mathcal{M}$ is travelling from l, then l might not be yet in the domain of a (the service message to l has not yet been issued) whereas, as we shall see in event leave_agt, l is in da because site l must not forward any message since it does know where $\mathcal{M}$ is. So, in that case, we have $\mathrm{dom}(a) \cup \{l\} = da$ and we have $l \in da$ and might have $l \notin \mathrm{dom}(a)$.

2. Conversely, when the travelling $\mathcal{M}$ just arrives at its new destination s (which will become thus the new l), then the new l is removed from da since the new l is now the new site of $\mathcal{M}$, hence the new l should not forward messages anymore but delivers them directly to $\mathcal{M}$. But, at the same time, a late service message to the new l might still be travelling, hence we might have the new l being still a member of $dom(a)$. So, in that case, we have $\mathrm{dom}(a) = da \cup \{l\}$ and we have $l \notin da$ and might have $l \in dom(a)$.

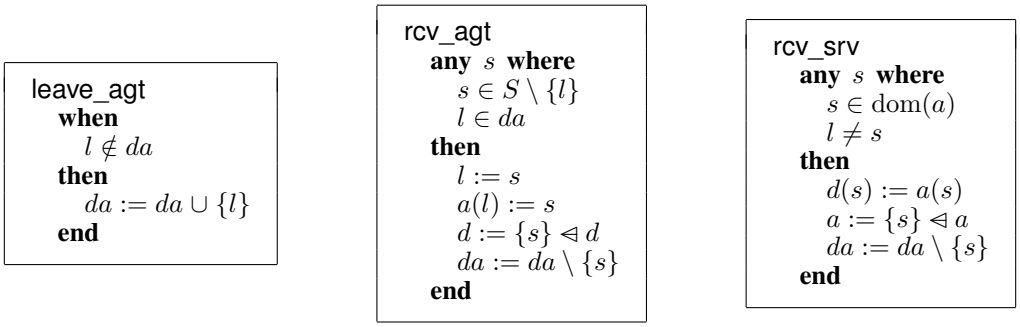
3.2 The Events

The various concrete events are very close to their abstraction. As event snd_msg does not change, we have not copied it in what follows. Here are the first ones:



Event fwd_msg has a stronger guard than its abstraction. More precisely, a message m whose site is $p(m)$ can only be forwarded if this site is not in da, since then the destination is not known, and also different from l, since then it can be delivered directly by event dlv_msg.

We have a new event called leave_agt. It corresponds to the mobile leaving its site l. Of course, it can only happen when l is not in da since otherwise the mobile would have been already in transit. As can be seen, site l is now expecting a service message (assignment $da := da \cup \{l\}$).



Event rcv_agt has a very interesting behavior. We note that when putting a new message $\{l \mapsto s\}$ in the service channel a (this message goes from the new site s of the mobile to its previous site l), we magically *remove the previous pair (if any) whose first component was* l (this corresponds to the assignment $a(l) := s$). In other words, we clean the channel by removing pending service messages to l, which might have not been delivered yet at l. In this way, there is at most one service message pointing to a given site, and there is thus no risk of having a pending message arriving late (that is, after a more recent one) and having the kind of bad effect we have described in section 1.3. Of course, this is quite magic for the moment. What we only wanted to express at this level is the intended behavior of the channel. It will remain, of course, for us to implement this magic behavior, which is another matter: this will be the business of the next refinement.

Finally, we have a new event, rcv_srv, corresponding to the reception by a site s of a service message informing s of the new location $a(s)$ of the mobile at the time this service message was sent. Notice that after we have just said, the move from s to $a(s)$ is indeed the most recent move done by the mobile from s since all other pending service messages to s have been discarded by event rcv_agt. The communication channel is updated and the service message removed from the service channel.

3.3 The Proofs

The proofs are left to the reader

4 Second Refinement

In the next refinement, we shall implement the magic service channel of the previous abstraction. This is the heart of the development.

4.1 The State

We now have a clock k travelling with the agent (**inv2_1**). We also have in each site a variable, t, recording the time of the last visit of the mobile in the corresponding site (**inv2_2**). Formally:

variables: $\ldots$, b, k, t

inv2_1: $k \in \mathbb{N}$

inv2_2: $t \in S \to \mathbb{N}$

The new service channel b replacing a has now a structure far richer than its abstraction. It may contain several stamped messages to the same site s. It is formalized as indicated in the following example:

$$s \mapsto \{3 \mapsto s1, 5 \mapsto s2, 9 \mapsto s3, \ldots\}$$

It means that there has been a message $s \mapsto s1$ emitted at time 3, a message $s \mapsto s2$ emitted at time 5, a message $s \mapsto s3$ emitted at time 9, etc. The channel b is thus typed as follows:

inv2_3: $b \in S \to (\mathbb{N} \nrightarrow S)$

Next comes the invariant **inv2_4** connecting the abstract service channel a and the concrete one b: the service message to s in the abstract channel a corresponds, among all service messages to s in the concrete channel b, to the one with the *greatest time* (the most recent one):

inv2_4: $\forall s \cdot \; s \in \mathrm{dom}(a) \Rightarrow \mathrm{dom}(b(s)) \neq \varnothing \; a(s) = b(s)(\max(\mathrm{dom}(b(s))))$

When the time of the last visit $t(s)$ of the recipient s of a service message is strictly smaller than the maximum time of the pending service messages for that recipient, that is $\max(\mathrm{dom}(b(s)))$, then the recipient s in question is indeed expecting a real message as in the abstraction. This is formalized in the following invariant **inv2_5**:

inv2_5: $\forall s \cdot \; s \in S \; \mathrm{dom}(b(s)) \neq \varnothing \; t(s) < \max(\mathrm{dom}(b(s))) \Rightarrow s \in \mathrm{dom}(a)$

But the problem, of course, is that, *a priori*, the recipient does not know that it is indeed receiving the maximum in question. This difficulty will be circumvented below in the last invariant **inv2_9**. Invariant **inv2_5** allows us to prove the guard strengthening of event rcv_srv, with the help of invariant **inv2_9** below.

We now have three more invariants concerned with the time of the last visit t and the clock k: (1) the times in the pending service messages are never bigger than the clock (**inv2_6**), (2) the time of the last visit is equal to the clock at the site of the Mobile (**inv2_7**), and (3) in other sites, the time of the last visit is at most equal to the clock (**inv2_8**):

inv2_6: $\forall s \cdot s \in S \ \wedge \ \mathrm{dom}(b(s)) \neq \varnothing \ \Rightarrow \ \max(\mathrm{dom}(b(s))) \leq k$

inv2_7: $t(l) = k$

inv2_8: $\forall s \cdot s \in S \setminus \{l\} \ \Rightarrow \ t(s) \leq k$

Now comes at last the *key invariant* **inv2_9**. When the recipient s of a service message receives a message with a time n that is strictly greater than its own time of the last visit $t(s)$ then it can be *absolutely certain* that it is indeed receiving the message with the greatest time, therefore the same message as in the abstraction according to invariant **inv2_4**.

inv2_9: $\forall s, n \cdot \ s \in S \ \wedge \ n \in \mathrm{dom}(b(s)) \ \wedge \ t(s) < n \ \Rightarrow \ n = \max(\mathrm{dom}(b(s)))$

This invariant is far from being completely intuitive. The informal explanation is a follows. If several service messages are expected at a site s, then it means that the mobile has visited s several times. And on each such visit it has updated the time of last visit of s with the most recent value of the clock. Upon leaving site s it has sent to s (when arriving at its new location) a service message with a stamp value which is one more than that of the time of last visit of s. So, during its *last* visit to s, which has certainly taken place *after* the sending (not necessarily the receiving) of the previous service messages to s, the updated value of the time of last visit of s is then certainly greater than that of the stamp of any pending service messages to s. As a consequence, when the mobile leaves s again for the last time, it sends (upon arrival at its new location) yet another service message, which is then *the only one* with a stamp greater than the value of the time of the last visit at s. *All this, clearly, needs confirmation from a formal proof.* Thanks to this invariant, we can implement the magic abstract channel a with the concrete channel b.

4.2 The Events

Here is first the last version of event init:

init
$l := il$
$p := \varnothing$
$d := (S \setminus \{l\}) \times \{il\}$
$b := S \times \{\varnothing\}$
$da := \varnothing$
$k := 1$
$t := S \times \{0\} \ \lhd\!\!+ \{il \mapsto 1\}$

Next comes event rcv_agt together with its previous abstract version:

(abstract-)rcv_agt
any s **where**
$s \in S \setminus \{l\}$
$l \in da$
then
$l := s$
$a(l) := s$
$d := \{s\} \lhd d$
$da := da \setminus \{s\}$
end

(concrete-)rcv_agt
any s **where**
$s \in S \setminus \{l\}$
$l \in da$
then
$l := s$
$t(s) := k + 1$
$k := k + 1$
$b(l)(k + 1) := s$
$d := \{s\} \lhd d$
$da := da \setminus \{s\}$
end

Notice the incrementation of the clock k and the storing of it in $t(s)$. And now we propose event rcv_srv, again together with its previous abstract version:

(abstract-)rcv_srv
any s **where**
$s \in \mathrm{dom}(a)$
$l \neq s$
then
$d(s) := a(s)$
$a := \{s\} \lhd a$
$da := da \setminus \{s\}$
end

(concrete-)rcv_srv
any s, n **where**
$s \in S$
$n \in \mathrm{dom}(b(s))$
$t(s) < n$
then
$d(s) := b(s)(n)$
$t(s) := n$
$da := da \setminus \{s\}$
end

We again copy below **inv2_5** and **inv2_9** in order to show how part of guard strengthening, namely $s \in \mathrm{dom}(a)$, can be proved:

inv2_5: $\forall s \cdot \begin{pmatrix} s \in S \\ \mathrm{dom}(b(s)) \neq \varnothing \\ t(s) < \max(\mathrm{dom}(b(s))) \\ \Rightarrow \\ s \in \mathrm{dom}(a) \end{pmatrix}$

inv2_9: $\forall s, n \cdot \begin{pmatrix} s \in S \\ n \in \mathrm{dom}(b(s)) \\ t(s) < n \\ \Rightarrow \\ n = \max(\mathrm{dom}(b(s))) \end{pmatrix}$

In fact, putting together **inv2_9** and **inv2_5**, we easily obtain the following theorem:

thm2_1: $\forall s, n \cdot \; s \in S \;\; n \in \mathrm{dom}(b(s)) \;\; t(s) < n \;\; \Rightarrow \;\; s \in \mathrm{dom}(a)$

The second part of guard strengthening, namely $l \neq s$, can be proved according to **inv2_7**, **inv2_6** and again **inv2_9**. The proof is by contradiction: we suppose $l = s$ and derive a contradiction.

Note that, to simplify matters, we do not clean the channel b in event rcv_srv. As a matter of fact, *it is not necessary*. Since the abstract channel was cleaned (the refinement is correct), this means that the message will not be accepted another time. This is because of the updating of the time of last visit ($t(s) := n$). This gives us a cleaning effect.

4.3 The Proofs

Proofs are left to the reader.

5 Third Refinement: Data Refinement

In this refinement, we transform the set da into a boolean function. In fact we then localize this information in each site.

5.1 The State

We introduce the variable dab (**inv3_1**) replacing abstract variables da. Invariants **inv3_2** defines the boolean function as the characteristic function of the corresponding set.

variables: $\ldots \;\; dab$

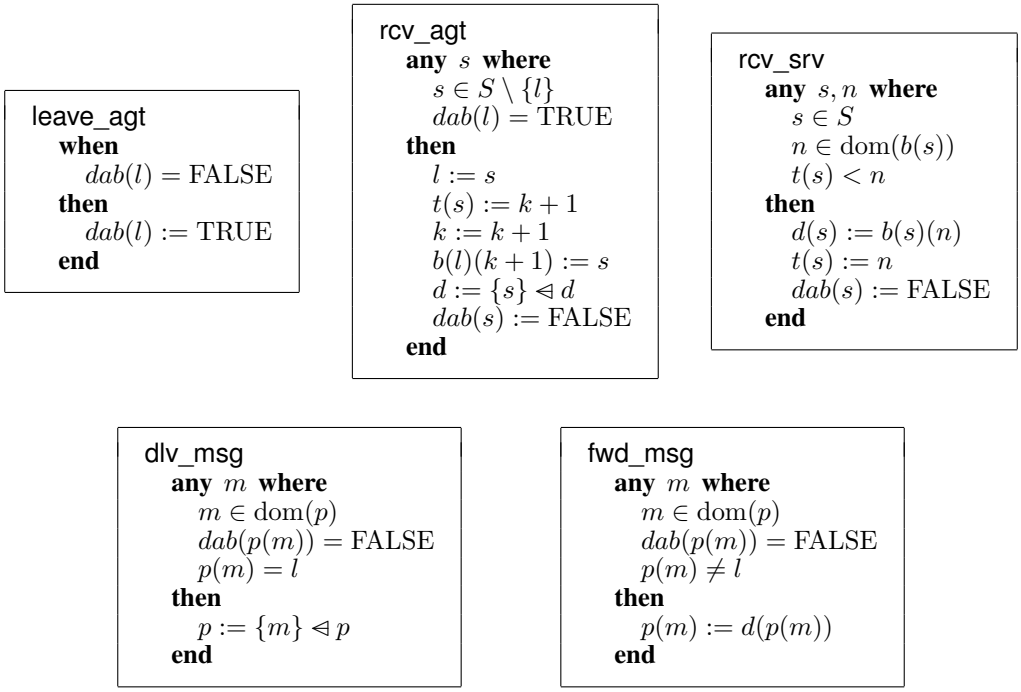
inv3_1: $dab \in S \rightarrow \mathrm{BOOL}$

inv3_2: $\forall x \cdot x \in S \Rightarrow (x \in da \Leftrightarrow dab(x) = \mathrm{TRUE})$

5.2 The Events

The events are now refined in a straightforward way as follows (note how the function dab is initialized):

init
$l := il$
$p := \varnothing$
$d := (S \setminus \{l\}) \times \{il\}$
$b := S \times \{\varnothing\}$
$dab := S \times \{\mathrm{FALSE}\}$
$k := 1$
$t := S \times \{0\} \; \lhd\!\!+ \; \{il \mapsto 1\}$



5.3 The Proofs

Proofs are left to the reader.

6 Fourth Refinement

There is one more refinement where we implement the effective migration of the forwarded communication messages. We leave it as an exercise to the reader to develop this refinement.

References

1. L. Moreau. *Distributed Directory Service and Message Routers for Mobile Agent*. Science of Computer Programming 39(2-3):249-272, 2001.