M2Pro : Génie Logiciel

Winter Trimester 2009/2010

Mounir Lallali, Prof. Burkhart Wolff
Parc Orsay Université
4, rue Jacques Monod
Building H / Room 012

TD 4

Date : 13.10.2009, Durée : 3 heures

Exercice 1
Récapitulation de tout ce qu'il faut ajouter, jusqu'à présent, dans le type `MsgStat`

```
type MsgStat is Boolean, GPEaddr, NaturalNumber, KindOfMsg, MsgID
    sorts Msg
    opns
        setDest : Msg, GPEaddr -> Msg
        getDest : Msg -> GPEaddr
        setSeq : Msg Nat -> Msg
        getSeq : Msg -> Nat
        setCoord : Msg, GPEaddr -> Msg
        getCoord : Msg -> GPEaddr
        setKind : Msg Kind -> Msg
        getKind : Msg -> Kind
        ...
        getID : Msg -> msgID
    eqns ofsort GPEaddr
        getDest(setDest(M, GAD)) = GAD;
        getCoord(setCoord(M, GAD)) = GAD;
    eqns ofsort Nat
        getSeq(setSeq(M, N)) = N;
    eqns ofsort Kind
        getKind(setKind(M, K)) = K;
    eqns ofsort msgID
        getID(M) = consID(getCoord(M), getSeq(M));
        ...
endtype
```

```
type KindOfMsg is Boolean
    sorts Kind
    opns
        NEW, OK, NOK, COMMIT, ABORT, QUERY -> Kind
        _eq_ , _neq_ : Kind, Kind -> Bool
    eqns ofsort Bool
        K eq K = true;
        NEW eq OK = false;
        ...
        K neq K1 = not (K eq K1) ;
endtype
```

```
type MsgId is Boolean, GPEaddr, NaturalNumber
    sorts msgID
    opns
        consMID : GPEaddr, Nat -> msgID
        getC : msgID -> GPEaddr
        getS : msgID -> Nat
        _eq_ , _neq_ : Kind, Kind -> Bool
        eqns ofsort GPEaddr
        getC(consMID (GAD, N)) = GAD;
    eqns ofsort Nat
        getS (consMID(GAD, N)) = N;
    eqns ofsort Bool
        Mid1 eq Mid2 = (getC(Mid1) eq getC(Mid2))
                        and (getS(Mid1) eq getS(Mid2);
        Mid1 neq Mid2 = not (Mid1 eq Mid2);
endtype
```

Rappel : Cas d'une nouvelle diffusion à faire, FromUser ?msg :Msg ; ...
Pour cela, il faut commencer à spécifier le type `MsgDict`.

```
(fromUser?msg:Msg;
    let newM: Msg = setKind(setCoord(msg, moi), NEW) in
        writePending! ...;
        Broadcast [toNet] (newM, lesAutres) >>
        GroupProtocol [ToUser, FromUser, FromNet, ToNet, ReadPending,
WritePending, ReadOutcome, WriteOutcome,...] (moi, lesAutres)
```

Sur `WritePending`, il faut émettre l'ancienne valeur du dictionnaire des messages *pendants* à laquelle on ajoute le nouveau message. Attention, il faut préciser qu'il n'y a eu aucun vote...
Il faut à ajouter `MsgDict` une liste de votes... On simplifie en se contentant de lister `lesAutres` qui doivent voter. Quand un message `OK` arrive, on retire son origine de la liste.
On peut alors compléter la demande d'une nouvelle diffusion :

```
(fromUser?msg:Msg;
    let newM: Msg = setKind(setCoord(msg, moi), NEW) in
        writePending! ajout(pending, getID(newM), newM, lesAutres);
        Broadcast [toNet] (newM, lesAutres) >>
        GroupProtocol [ToUser, FromUser, FromNet, ToNet, ReadPending,
           WritePending, ReadOutcome,
WriteOutcome, ...] (moi, lesAutres)
```

NB : il y a plusieurs manières de spécifier `MsgDict`. On a choisi un dictionnaire indexé par l'identité des messages...

```
type MsgDictwithVotes is Boolean, NaturalNumber, MsgStat, MsgID
    sorts MsgDictV
    opns
        emptyD -> MsgDictV
        ajout: MsgDictV, MsgID, Msg, ListGPEaddr -> MsgDictV
        rechMsg : MsgDictV, MsgID -> Msg
        rechVotes : MsgDictV, MsgID -> ListGPEaddr
        majVotes : MsgDictV, MsgID, ListGPEaddr -> MsgDictV
        suppr : MsgDictV, MsgID -> MsgDictV
        isinMDV : MsgDictV, MsgID -> Bool
    eqns ofsort ...
  ...
endtype
```

Question 1
Dans le scénario ci-dessous, un message au réseau est suivi de la liste de ses estinataires : par exemple le premier message `NEW,m,(GPE1, GPE2)` est envoyé par `GPE3` au réseau avec pour destinataires `GPE1` et `GPE2`. Puis le message `NEW`, m est envoyé par le réseau à `GPE1`.

Remarque : contrairement aux diagrammes simplifiés vus au débuts du cours sur le protocole `2 Phase Commit` on détaille ici le fait que les `GPE` passent obligatoirement par le réseau pour communiquer.

Dans le protocole `2 Phase Commit` étudié en cours, donner la suite de ce scénario

- dans le cas où il n'y a aucune panne
- dans le cas ou `GPE1` tombe en panne après avoir envoyé `OK` à `GPE3`, mais avant de l'avoir fait pour `GPE2`, et redémarre après un temps suffisant pour que `GPE2` ait envoyé un abort et `GPE3` un Commit
- dans le cas où `GPE1` ne tombe pas en panne, mais `GPE3` le fait avant d'avoir reçu les `OK` de `GPE1` et `GPE2`

Les scénarios doivent être complets : les messages m et m' doivent avoir fait l'objet soit d'un `COMMIT`, soit d'un `ABORT` chez tous leurs destinataires

Question 2 Dans le processus `GroupProtocol`, compléter le cas où on reçoit un message de sorte `NEW`.

Question 3 Dans le processus `GroupProtocol`, compléter le cas où on reçoit un message de sorte `COMMIT`, ou un message de sorte `ABORT`.

Question 4 Compléter les cas `OK` et `NOK`.

Question 5 Réfléchir au cas `QUERY`.