Superviseur:Professeur-Monsieur Jean-Marie JACQUET
INFO M441 - Approche Sémantique

Protocole de la retransmission bornée pour les grands paquets de données

Présentée par: Quoc Anh LE

Lundi, Juin 02, 2008

Rappel

- Les algèbres de processus: sont un famille de langages formels permettant de modéliser les systèmes (informatiques) concurrents ou distribués.
- µCRL (micro Commen Representation Language) est un langage algébrique de processus qui est spécialement développé pour tenir compte de données lors d'étude des processus communicatifs. Il est principalement destiné pour étudier les techniques descriptives et analytiques pour les (grands) systèmes distribués.
- Coq est un système de gestion de la preuve: un fait la preuve avec Coq est mécaniquement vérifiée par la machine. En particulier, Coq permet:
  - De définir des fonctions ou des prédicats,
  - État de théorèmes mathématiques et les spécifications du logiciel,
  - De développer interactivement preuve formelles de ces théorèmes,
  - Pour vérifier ces preuves par un relativement petit de certification « noyau ».

Rappel

- Les opérateurs principales pour définir les processus finis (A, +, .) : A est un ensemble d'actions (atomic), « + » est une composition alternative et « . » est un composition séquentielle.
- Les opérateurs communicatifs ($⫫$, $\|$, $|$) pour exprimer les parallélismes.
- Deadlock et encapsulation (δ, ∂_H) pour forcer les actions atomique à la communication
- Silence et abstraction (τ , τ_I) pour les calculs internes invisibles
- Guarded linear recursion ($\langle X|E \rangle$) pour capturer les processus ordinaires

=> Grâce à eux, on peut une forme solidaire pour analyser les systèmes différents

Introduction

- Ce rapport présente la spécification formelle et la vérification du protocole BRP (Bounded Retransmission Protocol) utilisé par Philips dans l'un de ses produits.
- Un paquet est trop grand d'envoyer une seul fois, donc il doit être coupé en plusieurs parties et on envoie chaque partie tour à tour.
  - Limiter le temps d'envoi
  - Limiter le nombre de retransmission
- Les signaux sont envoyés à fin d'informer l'états en cours.
- µCRL est appliqué comme un cadre formel (µCRL est une combinaison des algèbres de processus et les types de données abstraites).
- Coq est utilisé pour vérifier la sécurité (safety), l'interblocage et le liveness

Description du comportement extérieur (1)

- Le cercle représente les transferts réussis des paquets.
- Mémoire-tampon (buffer) pour recevoir les données du client et les envoyer.
- Limiter le temps d'envoi pour chaque paquet => pas assurer les réussites
- Les paquets ne seront pas coupés ou changés de l'ordre.
- La grande donnée d'entrée au canal 1 est modelé comme une liste des petits paquets r1(d_1,…,d_n).
- Indicateur d'états sont transmis à l'envoyeur et aussi au receveur.
- Indicateurs: I_{FST}, I_{INC}, I_{OK}, I_{NOK} or I_{DK}

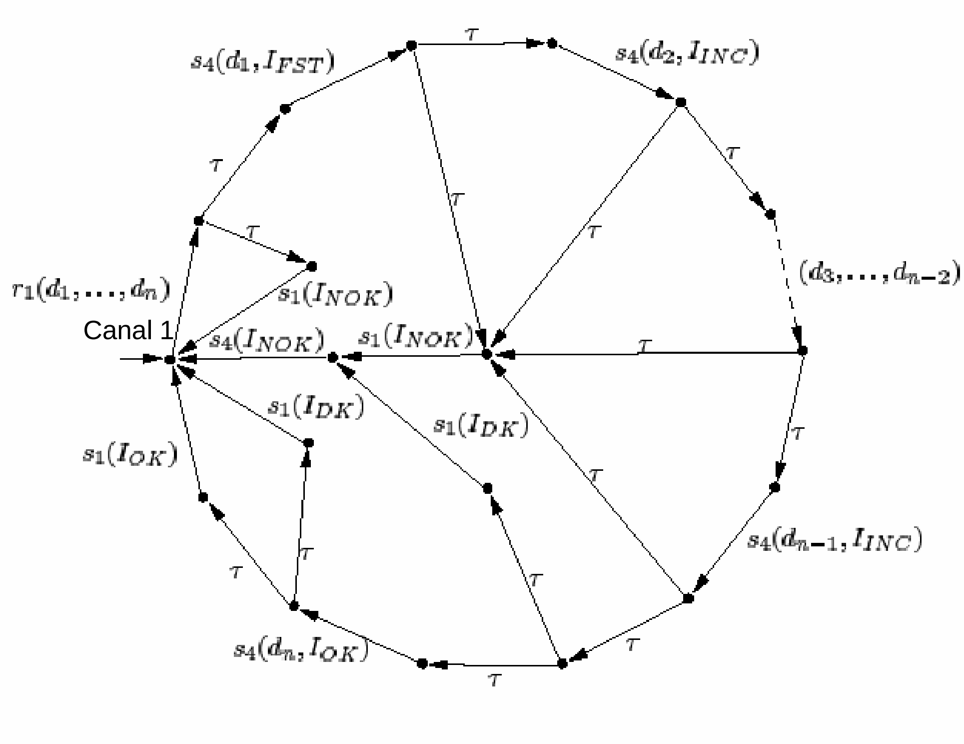

Figure 1: External behaviour of the BRP

- Il y a un cas spécial qu'un message envoyé au canal 1. Supposé que l'envoyeur envoie le dernier paquet d_n à la destinataire, mais il ne reçoit pas la confirmation, après même le nombre maximal de retransmission. Donc, il ne sait pas si le récepteur reçoit d_n ou pas. Alors, un message I_{DK} est envoyé au canal 1 après un certain temps. Donc, l'envoyeur pourra continuer transmettre la liste à venir.

Description du comportement extérieur (2)

- Le comportement est modelé comme un processus défini par un système de 4 équations récursives comme le suivant:
  - $r_i(d)$ et $s_i(d)$: recevoir et envoyer le paquet d par le canal i
  - xy: composition séquentielle
  - x‹b›y: then-if-else
  - x+y: choix – composition alternative
  - τ: silence pour les calculs internes invisibles
  - Choisir une liste dans un ensemble des listes $(\sum_{d:D} x(d))$
- X1: liste l -> faire suivre à X2 dont chaque élément joint b – bit supplémentaire
  - e0 si certains éléments sont déjà envoyé avant
  - e1 si aucun élément est déjà envoyé avant
- C_{ind} calcule une indication pour le client
- I_{ind} calcule une indication pour le destinataire
- X2 = τx + τy dans lequel x,y pour perdre ou envoyer le premier élément de la liste
- head(d1,…,d_N) = d1 et tail(d1,…d_N) = (d2,…,d_N)
- Indl(l) rend e0 ou e1

$$
\begin{array}{ll}
\textbf{sort} & Ind \\
\textbf{func} & I_{FST}, I_{OK}, I_{NOK}, I_{INC}, I_{DK} : \to Ind \\
 & C_{ind} : List \to Ind \\
 & I_{ind} : Bit \times Bit \to Ind \\
 & if : \mathbf{Bool} \times Ind \times Ind \to Ind \\
\textbf{var} & l : List, i_1, i_2 : Ind \\
\textbf{rew} & C_{ind}(l) = if(eq(indl(l), e_0), I_{NOK}, I_{DK}) \\
 & I_{ind}(e_0, e_0) = I_{INC} \\
 & I_{ind}(e_0, e_1) = I_{OK} \\
 & I_{ind}(e_1, e_0) = I_{FST} \\
 & I_{ind}(e_1, e_1) = I_{OK} \\
 & if(\mathsf{t}, i_1, i_2) = i_1 \\
 & if(\mathsf{f}, i_1, i_2) = i_2 \\
\textbf{act} & r_1 : List \\
 & s_1, s_4 : Ind \\
 & s_4 : D \times Ind \\
\textbf{proc} & X_1 = \sum_{l:List} r_1(l)\, X_2(l, e_1)
\end{array}
$$

$$
\begin{array}{l}
X_2(l{:}List, b{:}Bit) = \\
\quad \tau(X_3(C_{ind}(l)) \triangleleft eq(b, e_1) \triangleright X_4(C_{ind}(l))) \\
\quad + \tau\, s_4(head(l), I_{ind}(b, indl(l))) \\
\qquad ((\tau\, X_3(I_{OK}) + \tau\, X_3(I_{DK})) \triangleleft last(l) \triangleright (\tau\, X_2(tail(l), e_0) + \tau\, X_4(I_{NOK})))
\end{array}
$$

$$X_3(c{:}Ind) = s_1(c)\, X_1$$

$$X_4(c{:}Ind) = s_1(c)\, s_4(I_{NOK})\, X_1$$

Description du protocole

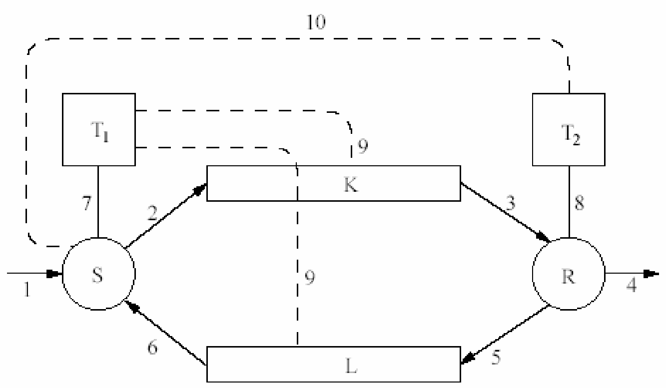

Figure 2: The structure of the BRP.

$$
\begin{array}{ll}
\textbf{sort} & TComm \\
\textbf{func} & set, reset, signal, ready, lost : TComm \\
\textbf{act} & r_2, s_2, c_2, s_3, r_3, c_3 : Bit \times Bit \times Bit \times D \\
 & r_5, s_5, c_5, r_6, s_6, c_6 \\
 & r_7, s_7, c_7, r_8, s_8, c_8, r_9, s_9, c_9, r_{10}, s_{10}, c_{10} : TComm \\
\textbf{comm} & r_2|s_2 = c_2 \quad r_5|s_5 = c_5 \quad r_7|s_7 = c_7 \quad r_9|s_9 = c_9 \\
 & r_3|s_3 = c_3 \quad r_6|s_6 = c_6 \quad s_8|r_8 = c_8 \quad r_{10}|s_{10} = c_{10}
\end{array}
$$

- Le système se compose d’un envoyeur doté un minuteur T1 et un receveur doté T2 qui échangent les paquets via les canaux K et L non déterminisme.
- T1 et T2 utilisent un ensemble de signaux (TComm) pour informer les états du système en cours à l’envoyeur et au receveur.
- Les signaux *lost* et *ready* sont transmis via les canaux 9 et 10. Ils sont considéré comme « elapse of time » et pas comme signaux physiques.
- Le *signal* est un signal de time-out
- r_i et s_i sont receveur et envoyeur respectivement via le canal i dans lequel le paquet transmis se compose de 4 parties (b,b’,b’’,d), trois premiers bits sont e_0 ou e_1 pour le but:
  - b indique si d est le premier élément de la liste
  - b’’ indique si d est le dernier élément de la liste
  - b’’ est le bit alternatif pour assurer que le paquet ne sera pas double.

Description du protocole – L’envoyeur

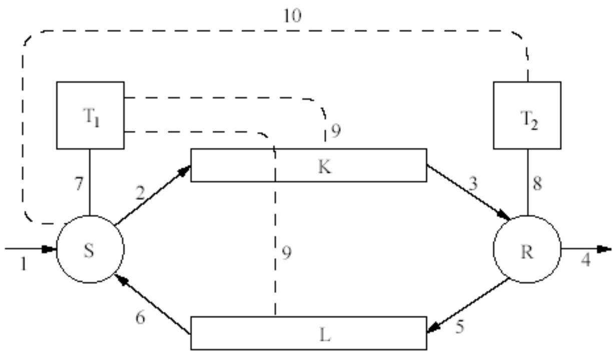

Figure 2: The structure of the BRP.

$$S(b'':Bit, max:\mathbb{N}) = \sum_{l:List} r_1(l)\, S_1(l, e_1, b'', 0, max)$$

$$S_1(l:List, b, b'':Bit, rn, max:\mathbb{N}) = s_7(set)\, s_2(b, indl(l), b'', head(l))\, S_2(l, b, b'', rn, max)$$

$$\begin{aligned} S_2(l:List, b, b'':Bit, rn, max:\mathbb{N}) = \\ & r_6\, s_7(reset)\, (s_1(I_{OK})\, S(inv(b''), max) \lhd last(l) \rhd S_1(tail(l), e_0, inv(b''), rn, max)) \\ & + r_7(signal)\, S_3(l, b, b'', rn, max, C_{ind}(l)) \end{aligned}$$

$$\begin{aligned} S_3(l:List, b, b'':Bit, rn, max:\mathbb{N}, c:Ind) = \\ & s_1(c)\, s_{10}(ready)\, r_{10}(ready)\, S(inv(b''), max) \lhd eq(rn, max) \rhd \delta \\ & + S_1(l, b, b'', s(rn), max) \lhd lt(rn, max) \rhd \delta \end{aligned}$$

- L’envoyeur lit une liste (r1) et établit le compter à zéro. Ensuit, il envoi les éléments un par un (S1). Le temps T1 est remis le compteur à zéro (s7). Après avoir envoyé, l’envoyeur attend (s2) une reconnaissance (ack) du receveur ou un time-out.
  - Si une reconnaissance arrive (r6), T1 sera remis à zéro (s7) et un signal de transmission réussi est envoyé (s1) **SI** c’est le dernier paquet de la liste, **SINON** le paquet suivant est envoyé.
  - Si time-out est rendu par T1 (r7), le paquet est retransmis **SI** le nombre de retransmission n’est pas encore dépassé le valeur maximum, alors, le compteur augmente par 1 et le temps est remis. **SINON**, la transmission est cassée et T2 expire (s10)

Description du protocole – Le receveur

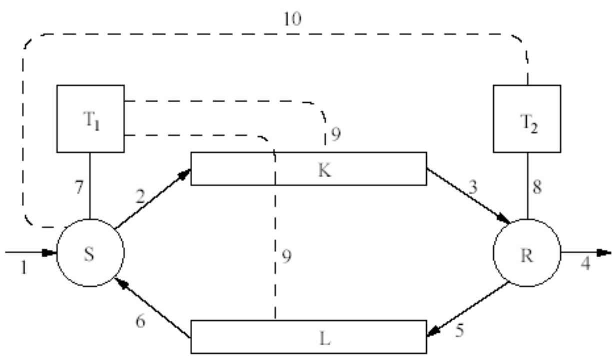

Figure 2: The structure of the BRP.

$$R = \sum_{b',b'':Bit,d:D} r_3(e_1,b',b'',d)\, R_2(b',b'',d,I_{ind}(e_1,b'))$$

$$\begin{aligned} R_1(b,b'':Bit) = &\\ &\sum_{b':Bit,d:D} \big(r_3(b,b',b'',d)\, s_8(reset)\, R_2(b',b'',d,I_{ind}(b,b'))\\ &\qquad + r_3(b',b,inv(b''),d)\, s_5\, R_1(b,b'')\big)\\ &+ r_8(signal)\,\big(s_4(I_{NOK})\, s_8(ready)\, R \triangleleft eq(b,e_0) \triangleright s_8(ready)\, R\big) \end{aligned}$$

$$R_2(b',b'':Bit,d:D,i:Ind) = s_4(d,i)\, s_8(set)\, s_5\, R_1(b',inv(b''))$$

- Le receveur attend le premier paquet r3(e1) à arriver (R). Lorsque le paquet arrive (R2), T2 est démarré (s8) et un reconnaissance est envoyé (s5). Après, il continue d'attendre les paquets suivants à arriver (R1); le valeur alternatif est stocké.
  - Le premier bit de R1 indique si le paquet précédent est le dernier ou pas. Le deuxième bit est le valeur souhaité du bit alternatif (b''=inverse(b'')). Chaque paque arrivé est reconnaît mais il est traité par le receveur si seulement le bit alternatif indique qu'il est nouveau. Dans ce cas, T2 est remis. Note que si seulement le paquet dernier est le denier paquet de la liste, un paque frais sera alors le premier de la liste suivante et un paquet répété sera encore le dernier de la vieille liste. Ça explique une double utilisation du bit b.
  - Ça continue jusqu'à T2 émit time-out (r8) lors de l'attente depuis longtemps pour une nouvel arrivée indiquant que la transmission est cessée. L'envoyeur est informé que le dernier élément n'est pas encore délivré (s4). Note que si la transmission de la liste suivante commence avant d'expiration du T2, le système du bit alternatif continue jusqu'à un échec.

Description du protocole – Le temps T1

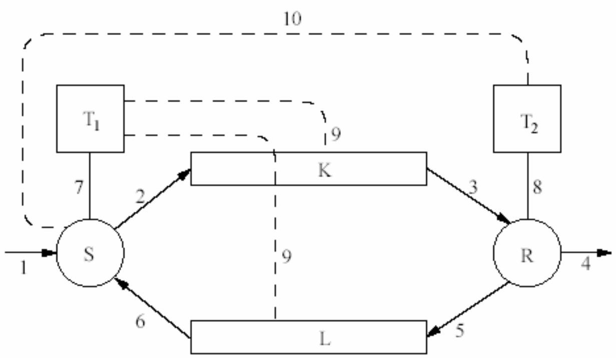

Figure 2: The structure of the BRP.

$$T_1 = r_7(set)\,(r_9(lost)\,s_7(signal) + r_7(reset))\,T_1$$

- Le système de temps T1 émit un time-out si une reconnaissance (ack) n'arrive pas à temps.
  - Il est mis chaque fois un paquet est envoyé et remis lors que ce paquet est reconnaît.
  - Pour éviter qu'un message arrive après l'expiration du temps, on demande aux canaux K et L de transmettre un signal *s9(lost)* à T1, indiquant qu'un time-out peut être apparaît. Ça suit la supposition que: le temps total pour un paquet est traversé via le canal K pour générer une reconnaissance à R et pour la transférer via L est borné par un retard fixe.

Description du protocole – Le temps T2

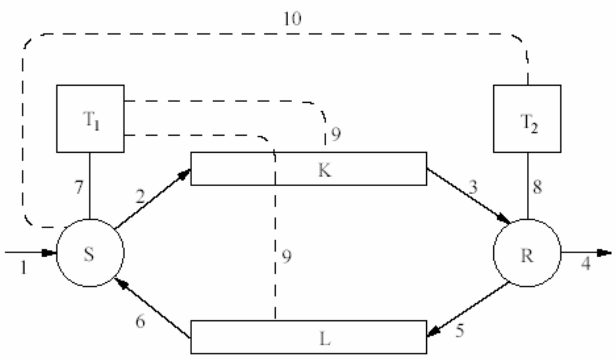

Figure 2: The structure of the BRP.

$$T_2 = (r_8(set)\,(r_{10}(ready)\,s_8(signal)\,r_8(ready)\,s_{10}(ready) + r_8(reset)) \\ + r_{10}(ready)\,s_{10}(ready))\,T_2$$

- Le système de temps T2 est remis par le receveur *r8(set)* chaque fois un nouveau paquet arrive. Il émit un time-out si la transmission d'une liste est cessée par l'envoyeur. Son retard doit surpasser le temps maximum du retard de T1. T2 est aussi modelé que l'envoyeur ne commence pas de lire et transmettre la liste suivante avant que le receveur réagit correctement l'échec. C'est nécessaire car le receveur ne change pas encore son bit alternatif, donc un nouveau paquet sera interprété comme une répétition.

- Le time-out *s8(signal)* est précédé par un signal de l'envoyeur *r10(ready)* pour assurer que la transmission de la liste en cours s'arrête. Il est suivi par *r8(ready)s10(ready)* pour empêche l'envoyeur d'envoyer une nouvelle liste trop tôt.

Description du protocole – Les intermédiaires L, K

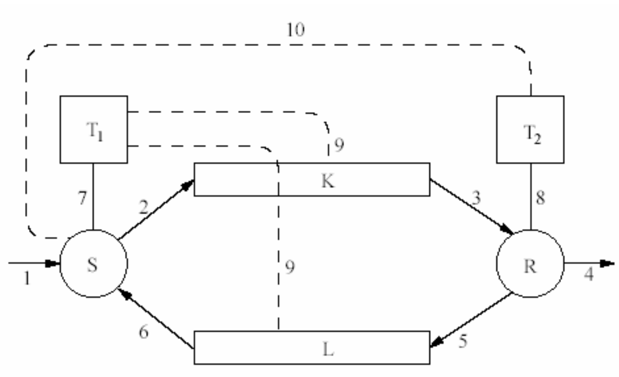

Figure 2: The structure of the BRP.

$$K = \sum_{b,b',b'':Bit,\, d:D} r_2(b,b',b'',d)\,\big(\tau\, s_3(b,b',b'',d) + \tau\, s_9(lost)\big)\, K$$
$$L = r_5\,(\tau\, s_6 + \tau\, s_9(lost))\, L$$

- K attend et recevoir un paquet au canal 2 *r2(b,b',b'',d)*, il y a alors deux cas:
  - Il transmet ce paquet au receveur via le canal 3 (r3)
  - Il perd ce paquet, après un retard il fait T1 via le canal 9 *s9(lost)* d'envoyer à l'envoyeur un time-out.
- L reçoit une reconnaissance du receveur via le canal 5 (r5). Il y a alors deux cas:
  - Il transmet ce message à l'envoyeur via le canal 6 (s6)
  - Il perd ce message, après un retard, il fait T1 via le canal 9 *s9(lost)* d'envoyer à l'envoyeur un time-out.

Description du protocole

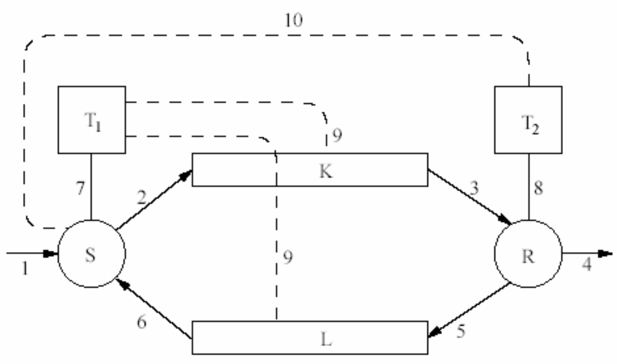

Figure 2: The structure of the BRP.

$$
\begin{array}{ll}
\textbf{sort} & TComm \\
\textbf{func} & set, reset, signal, ready, lost : TComm \\
\textbf{act} & r_2, s_2, c_2, s_3, r_3, c_3 : Bit \times Bit \times Bit \times D \\
 & r_5, s_5, c_5, r_6, s_6, c_6 \\
 & r_7, s_7, c_7, r_8, s_8, c_8, r_9, s_9, c_9, r_{10}, s_{10}, c_{10} : TComm \\
\textbf{comm} & r_2|s_2 = c_2 \quad r_5|s_5 = c_5 \quad r_7|s_7 = c_7 \quad r_9|s_9 = c_9 \\
 & r_3|s_3 = c_3 \quad r_6|s_6 = c_6 \quad s_8|r_8 = c_8 \quad r_{10}|s_{10} = c_{10}
\end{array}
$$

$$K = \textstyle\sum_{b,b',b'':Bit,d:D} r_2(b,b',b'',d)\,(\tau\, s_3(b,b',b'',d) + \tau\, s_9(lost))\,K$$
$$L = r_5\,(\tau\, s_6 + \tau\, s_9(lost))\,L$$

Protocole de la retransmission bornée pour les grands paquets de données peut être décrit comme le suivant.

$I := \{c_2, c_3, c_5, c_6, c_7, c_8, c_9, c_{10}\}$ and $H := \{r_2, s_2, r_3, s_3, r_5, s_5, r_6, s_6, r_7, s_7, r_8, s_8, r_9, s_9, r_{10}, s_{10}\}$.

$\textbf{proc} \quad BRP(max{:}\mathbb{N}) = \tau_I \partial_H (T_1 \parallel S(e_0, max) \parallel K \parallel L \parallel R \parallel T_2)$

La démonstration de correction

- L'objectif: BRP satisfait tous les comportements extérieurs?
- Techniques utilisées de vérification:
  - Expansion: Afin de calculer les transitions initiales des processus t1 ‖t2 ‖t3 … il est suffisant de calculer les transitions initiales des arguments t1, t2, t3, …en utilisant les axioms
  - Si un processus t ne peut exécuter aucun actions de l'ensemble H, alors on a: $\partial_H(t) = t$
  - Bisimulation de branche:

$$\text{if } p \stackrel{a}{\Rightarrow} q \text{ and } p \stackrel{b}{\Rightarrow} r \text{ with } a \neq b, \text{ then } q \stackrel{b}{\Rightarrow} q' \text{ and } r \stackrel{a}{\Rightarrow} r' \text{ with } q' \underline{\leftrightarrow}_b r';$$
$$\text{if } p \stackrel{a}{\Rightarrow} q \text{ and } p \Rightarrow r, \text{ then } r \stackrel{a}{\Rightarrow} r' \text{ with } q \underline{\leftrightarrow}_b r'.$$

  - Toutes les équations gardées ont une seule solution => donc, il faut transformer les équations à la forme gardée
- Théorème: Pour tout *max*: *N*, on a X_1 = BRP(*max*)
- Tout d'abord, on donne les calculs qui sont nécessaires pour la démonstration sous les lemmes

$$Z_1(b'', max) = \sum_{l:List} r_1(l)\, \tau_I \partial_H(T_1 \parallel S_1(l, e_1, b'', 0, max) \parallel K \parallel L \parallel R \parallel T_2)$$
$$Z_1'(b'', max) = \sum_{l:List} r_1(l)\, \tau_I \partial_H(T_1 \parallel S_1(l, e_1, b'', 0, max) \parallel K \parallel L \parallel R_1(e_1, b'') \parallel T_2')$$

- Supposé que on a deux situations Z_1 et Z_1' dans lesquelles le receveur ne connaît pas le bit alternatif pour le suivant lorsque:
  - Après avoir démarré le protocole.
  - Après la terminaison à cause d'échec.
- Alors, on doit montrer ces deux situations mènent une solution unique

La démonstration

- $Z_2,\ldots,Z_4''$ décrivent les comportements extérieurs du protocole. Ce sont les équations récursives comme le suivant:

$$
\begin{aligned}
&Z_2(l, b'', max) = \\
&\qquad \big(\tau\, Z_4(I_{DK}, b'', max) + \tau\, Z_3(l, b'', max)\big) \\
&\qquad \lhd last(l) \rhd \\
&\qquad \big(\tau\, Z_4(I_{NOK}, b'', max) \\
&\qquad\quad + \tau\, s_4(head(l), I_{FST})\,(\tau\, Z_2'(tail(l), e_0, inv(b''), max) + \tau\, Z_4''(I_{NOK}, b'', max))\big)
\end{aligned}
$$

$$
\begin{aligned}
&Z_2'(l, b, b'', max) = \\
&\qquad \big(\tau\,(Z_4(I_{DK}, b'', max) \lhd eq(b, e_1) \rhd Z_4''(I_{DK}, b'', max)) + \tau\, Z_3(l, b'', max)\big) \\
&\qquad \lhd last(l) \rhd \\
&\qquad \big(\tau\,(Z_4(I_{NOK}, b'', max) \lhd eq(b, e_1) \rhd Z_4''(I_{NOK}, b'', max)) \\
&\qquad\quad + \tau\, s_4(head(l), I_{ind}(b, e_0))\,(\tau\, Z_2'(tail(l), e_0, inv(b''), max) + \tau\, Z_4''(I_{NOK}, b'', max))\big)
\end{aligned}
$$

$$Z_3(l, b'', max) = s_4(head(l), I_{OK})\,(\tau\, Z_4'(I_{OK}, b'', max) + \tau\, Z_4(I_{DK}, b'', max))$$

$$
\begin{aligned}
Z_4(c, b'', max) &= s_1(c)\, Z_1(inv(b''), max) \\
Z_4'(c, b'', max) &= s_1(c)\, Z_1'(inv(b''), max) \\
Z_4''(c, b'', max) &= s_1(c)\, s_4(I_{NOK})\, Z_1(inv(b''), max)
\end{aligned}
$$

La démonstration – Le lemme

Define the auxiliary processes

$$K'(b,b',b'',d) := (\tau\, s_3(b,b',b'',d) + \tau\, s_9(lost))\, K$$
$$L' := (\tau\, s_6 + \tau\, s_9(lost))\, L$$
$$T_2' := (r_{10}(ready)\, s_8(signal)\, r_8(ready)\, s_{10}(ready) + r_8(reset))\, T_2$$
$$T_1' := (r_9(lost)\, s_7(signal) + r_7(reset))\, T_1$$

Lemma 5.1. *For all* $b, \bar{b}, b'', b''' : Bit,\ max, rn : \mathbb{N},\ i, c : Ind$ *with* $lt(rn, max)$ or $eq(rn, max)$, we find

1. $Z_1(b'', max) = \tau_I \partial_H(T_1 \parallel S(b'', max) \parallel K \parallel L \parallel R \parallel T_2)$
2. $Z_1'(b'', max) = \tau_I \partial_H(T_1 \parallel S(b'', max) \parallel K \parallel L \parallel R_1(e_1, b'') \parallel T_2')$
3. $Z_4(c, b'', max) = \tau_I \partial_H(T_1 \parallel S_3(l, b, b'', max, max, c) \parallel K \parallel L \parallel R \parallel T_2)$
4. $Z_4''(c, b'', max) = \tau_I \partial_H(T_1 \parallel S_3(l, \bar{b}, b'', max, max, c) \parallel K \parallel L \parallel R_1(e_0, b''') \parallel T_2')$
5. $Z_4(c, b'', max) = \tau_I \partial_H(T_1 \parallel S_3(l, \bar{b}, b'', max, max, c) \parallel K \parallel L \parallel R_1(e_1, b''') \parallel T_2')$
6. $Z_4'(c, b'', max) = \tau_I \partial_H(T_1 \parallel s_1(c)\, S(inv(b''), max) \parallel K \parallel L \parallel R_1(e_1, inv(b'')) \parallel T_2')$

- 1,2 sont justifiés grâce à l'expansion à droit pour les processus parallèles. Il est lassé passer parce qu'il est complètement standard.
- 3,4,5 sont justifié grâce à 1 et à l'expansion à droit.
- 6 est justifié grâce à 2 et l'expansion à droit.

La démonstration – Le lemme

7. $last(l) = \mathsf{f} \rightarrow$
$$\tau\,(\tau\, Z_2'(tail(l), e_0, inv(b''), max) + \tau\, Z_4''(I_{NOK}, b'', max)) =$$
$$\tau\,\tau_I\partial_H(T_1' \parallel S_2(l, b, b'', rn, max) \parallel K \parallel L' \parallel R_1(e_0, inv(b'')) \parallel T_2')$$

8. $last(l) = \mathsf{t} \rightarrow$
$$\tau\,(\tau\, Z_4'(I_{OK}, b'', max) + \tau\, Z_4(I_{DK}, b'', max)) =$$
$$\tau\,\tau_I\partial_H(T_1' \parallel S_2(l, b, b'', rn, max) \parallel K \parallel L' \parallel R_1(e_1, inv(b'')) \parallel T_2')$$

9. $last(l) = \mathsf{t} \rightarrow$
$$Z_3(l, b'', max) = \tau_I\partial_H(T_1' \parallel S_2(l, b, b'', rn, max) \parallel K \parallel L \parallel R_2(indl(l), b'', head(l), I_{OK}) \parallel T_2)$$

10. $last(l) = \mathsf{t} \rightarrow$
$$\tau\,(\tau\, Z_4'(I_{OK}, b'', max) + \tau\, Z_4(I_{DK}, b'', max)) =$$
$$\tau\,\tau_I\partial_H(T_1' \parallel S_2(l, b, b'', rn, max) \parallel K'(b, indl(l), b'', head(l)) \parallel L \parallel R_1(e_1, inv(b'')) \parallel T_2')$$

11. $last(l) = \mathsf{t} \rightarrow$
$$\tau\,(\tau\, Z_4'(I_{OK}, b'', max) + \tau\, Z_4(I_{DK}, b'', max)) =$$
$$\tau_I\partial_H(T_1 \parallel S_1(l, b, b'', rn, max) \parallel K \parallel L \parallel R_1(e_1, inv(b'')) \parallel T_2')$$

12. $last(l) = \mathsf{f} \rightarrow$
$$s_4(d, i)\,(\tau\, Z_2'(tail(l), e_0, inv(b''), max) + \tau\, Z_4''(I_{NOK}, b'', max)) =$$
$$\tau_I\partial_H(T_1' \parallel S_2(l, b, b'', rn, max) \parallel K \parallel L \parallel R_2(e_0, b'', d, i) \parallel T_2)$$

- 8,10,11: Tout d'abord, on prouve 8 grâce à 5,6. Ensuit, à partir de 8 et 5, on prouve 10, après 11 est une conséquence de 10. En fait, 8 utilise 6 et l'hypothèse inductive pour 11. Enfin, 9 est déduit de 8.

La démonstration – Le lemme

13. $last(l) = \mathsf{f} \rightarrow$
$\tau(\tau\, Z_2'(tail(l), e_0, inv(b''), max) + \tau\, Z_4''(I_{NOK}, b'', max)) =$
$\tau\, \tau_I \partial_H(T_1' \,\|\, S_2(l, b, b'', rn, max) \,\|\, K'(b, indl(l), b'', head(l)) \,\|\, L \,\|\, R_1(e_0, inv(b'')) \,\|\, T_2')$

14. $last(l) = \mathsf{f} \rightarrow$
$\tau(\tau\, Z_2'(tail(l), e_0, inv(b''), max) + \tau\, Z_4''(I_{NOK}, b'', max)) =$
$\tau_I \partial_H(T_1 \,\|\, S_1(l, b, b'', rn, max) \,\|\, K \,\|\, L \,\|\, R_1(e_0, inv(b'')) \,\|\, T_2')$

15. $\tau\, Z_2(l, b'', max) = \tau\, \tau_I \partial_H(T_1' \,\|\, S_2(l, e_1, b'', rn, max) \,\|\, K'(e_1, indl(l), b'', head(l)) \,\|\, L \,\|\, R \,\|\, T_2)$

16. $\tau\, Z_2'(l, b, b'', max) =$
$\tau\, \tau_I \partial_H(T_1' \,\|\, S_2(l, b, b'', rn, max) \,\|\, K'(b, indl(l), b'', head(l)) \,\|\, L \,\|\, R_1(b, b'') \,\|\, T_2')$

17. $\tau\, Z_2(l, b'', max) = \tau_I \partial_H(T_1 \,\|\, S_1(l, e_1, b'', rn, max) \,\|\, K \,\|\, L \,\|\, R \,\|\, T_2)$

18. $\tau\, Z_2'(l, b, b'', max) = \tau\, \tau_I \partial_H(T_1 \,\|\, S_1(l, b, b'', rn, max) \,\|\, K \,\|\, L \,\|\, R_1(b, b'') \,\|\, T_2')$

- 7, 12, 13, 14, 16, 18: pour le max, rn et l fixes, 7=>12, 13=>14 et 16=>18 peuvent être justifiés grâce à l'expansion.
- Si l = vide, alors 7 et 13 sont vrais car *last(vide)* = *vrai*. Pour *rn=max*, 7 entraîne 16.15
- 15,17: 15 entraîne 17, pour 15, on utilise 3.

La démonstration

- A partir des lemmes 17 et 18, on a

$$Z_1(b'', max) = \sum_{l:List} r_1(l)\, Z_2(l, b'', max)$$
$$Z_1'(b'', max) = \sum_{l:List} r_1(l)\, Z_2'(l, e_1, b'', max)$$

- Et on a aussi

$$\begin{aligned}
Z_2(l, b'', max) = {} & \\
& (\tau\, Z_4(I_{DK}, b'', max) + \tau\, Z_3(l, b'', max)) \\
& \triangleleft last(l) \triangleright \\
& (\tau\, Z_4(I_{NOK}, b'', max) \\
& \;\; + \tau\, s_4(head(l), I_{FST})\, (\tau\, Z_2'(tail(l), e_0, inv(b''), max) + \tau\, Z_4''(I_{NOK}, b'', max)))
\end{aligned}$$

$$\begin{aligned}
Z_2'(l, b, b'', max) = {} & \\
& (\tau\, (Z_4(I_{DK}, b'', max) \triangleleft eq(b, e_1) \triangleright Z_4''(I_{DK}, b'', max)) + \tau\, Z_3(l, b'', max)) \\
& \triangleleft last(l) \triangleright \\
& (\tau\, (Z_4(I_{NOK}, b'', max) \triangleleft eq(b, e_1) \triangleright Z_4''(I_{NOK}, b'', max)) \\
& \;\; + \tau\, s_4(head(l), I_{ind}(b, e_0))\, (\tau\, Z_2'(tail(l), e_0, inv(b''), max) + \tau\, Z_4''(I_{NOK}, b'', max)))
\end{aligned}$$

$$Z_3(l, b'', max) = s_4(head(l), I_{OK})\, (\tau\, Z_4'(I_{OK}, b'', max) + \tau\, Z_4(I_{DK}, b'', max))$$

$$\begin{aligned}
Z_4(c, b'', max) &= s_1(c)\, Z_1(inv(b''), max) \\
Z_4'(c, b'', max) &= s_1(c)\, Z_1'(inv(b''), max) \\
Z_4''(c, b'', max) &= s_1(c)\, s_4(I_{NOK})\, Z_1(inv(b''), max)
\end{aligned}$$

- C’est clair qu’elles sont sous formes une spécification récursive gardée qui a une solution unique et Z_1, …, Z_4 ’’ forment la solution. Donc, BRP(*max*) = $Z_1(e_0, max)$

Vérifié par Coq V5.8.2

- Grâce au lemme 1, on a

$$\sum_{l:List} r_1(l)\,\tau_I\partial_H(T_1 \parallel S_1(l, e_1, b'', 0, max) \parallel K \parallel L \parallel R \parallel T_2) = \tau_I\partial_H(T_1 \parallel S(b'', max) \parallel K \parallel L \parallel R \parallel T_2),$$

- Sous la langue vulgaire, le preuve de l'équation ci-dessus est décrit comme le suivant:

```
Goal (b'':Bit)(max:Nat)
  <proc>(sum List [l:List] (seq (ia List r1 l) (hide IL (enc HL
          (mer T1 (mer (S1 l e1 b'' o max) (mer K (mer L (mer R T2))))))))
       =(hide IL (enc HL (mer T1 (mer (S b'' max) (mer K (mer L (mer R T2))))))).
Intros.
Pattern 2 T1; Rewrite <- hnf_T1. Rewrite <- Proc_S. ...
Load exp_tac.
Rewrite -> hnf_T1; Rewrite -> Proc_S; ...
Rewrite <- S_Lmer.
Load hide_strip.
Load equal_tac.
Save Exp_1.
```

- Cependant, on ne peut pas mécaniser tout calcul algèbre. Donc, on doit ajouter certains caractéristiques manqués du système Coq:
  - Metavariable
  - Full second order matching
  - Definition unfolding mechanism
  - Extensible vernacular language

Conclusion

- Ce protocole est déjà appliqué aux produits de Philip donc, sa correction est justifiée.
- La description est très concis (environ une page)
- La démonstration est justifiée par les opérateurs standards
- Trois caractéristiques importants: le sécurité, l'interblocage et le liveness sont vérifiés par Coq

Annexe

sort	$\mathbf{Bool}$
func	$\mathsf{f}, \mathsf{t} :\longrightarrow \mathbf{Bool}$
	$\wedge : \mathbf{Bool} \times \mathbf{Bool} \rightarrow \mathbf{Bool}$
var	$b : \mathbf{Bool}$
rew	$\mathsf{t} \wedge b = b$
	$\mathsf{f} \wedge b = \mathsf{f}$
sort	$D, List$
func	$d_0 :\longrightarrow D$
	$if : \mathbf{Bool} \times D \times D \rightarrow D$
	$eq : D \times D \rightarrow \mathbf{Bool}$
	$empty :\longrightarrow List$
	$add : D \times List \rightarrow List$
	$head : List \rightarrow D$
	$tail : List \rightarrow List$
	$last : List \rightarrow \mathbf{Bool}$
	$indl : List \rightarrow Bit$
var	$d, d_1, d_2 :\longrightarrow D$
	$l :\longrightarrow List$
rew	$head(empty) = d_0$
	$head(add(d, l)) = d$
	$tail(empty) = empty$
	$tail(add(d, l)) = l$
	$last(empty) = \mathsf{t}$
	$last(add(d, empty)) = \mathsf{t}$
	$last(add(d_1, add(d_2, l))) = \mathsf{f}$
	$indl(empty) = e_1$
	$indl(add(d, empty)) = e_1$
	$indl(add(d_1, add(d_2, l))) = e_0$
	$if(\mathsf{t}, d_1, d_2) = d_1$
	$if(\mathsf{f}, d_1, d_2) = d_2$
	$eq(d, d) = \mathsf{t}$
	$if(eq(d_1, d_2), d_1, d_2) = d_2$

sort	Bit
func	$e_0, e_1 :\longrightarrow Bit$
	$inv : Bit \rightarrow Bit$
	$if : \mathbf{Bool} \times Bit \times Bit \rightarrow Bit$
var	$b, b_1, b_2 : Bit$
rew	$inv(e_0) = e_1$
	$inv(e_1) = e_0$
	$if(\mathsf{t}, b_1, b_2) = b_1$
	$if(\mathsf{f}, b_1, b_2) = b_2$
	$if(eq(b_1, b_2), b_1, b_2) = b_2$
	$eq(b, inv(b)) = \mathsf{f}$
	$eq(b, b) = \mathsf{t}$
sort	$\mathbb{N}$
func	$0 :\longrightarrow \mathbb{N}$
	$s, pred : \mathbb{N} \rightarrow \mathbb{N}$
	$eq : \mathbb{N} \times \mathbb{N} \rightarrow \mathbf{Bool}$
	$lt : \mathbb{N} \times \mathbb{N} \rightarrow \mathbf{Bool}$
	$minus : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$
var	$n, n_1, n_2 :\longrightarrow \mathbb{N}$
rew	$eq(0, 0) = \mathsf{t}$
	$eq(0, s(n)) = \mathsf{f}$
	$eq(s(n), 0) = \mathsf{f}$
	$eq(s(n_1), s(n_2)) = eq(n_1, n_2)$
	$lt(0, s(n)) = \mathsf{t}$
	$lt(n, 0) = \mathsf{f}$
	$lt(s(n_1), s(n_2)) = lt(n_1, n_2)$
	$pred(0) = 0$
	$pred(s(n)) = n$
	$minus(n, 0) = n$
	$minus(n_1, s(n_2)) = pred(minus(n_1, n_2))$

Table 1: Specification of standard data types used in the BRP

Annexe

A1	$x + y = y + x$	SUM1	$\Sigma_{d:D} x = x$
A2	$x + (y + z) = (x + y) + z$	SUM3	$\Sigma_{d:D}\ p(d) = \Sigma_{d:D}\ p(d) + p(e)$
A3	$x + x = x$	SUM4	$\Sigma_{d:D}(p(d) + q(d)) = \Sigma_{d:D}\ p(d) + \Sigma_{d:D}\ q(d)$
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$	SUM5	$\Sigma_{d:D}(p(d) \cdot x) = (\Sigma_{d:D}\ p(d)) \cdot x$
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	SUM11	$(\forall d\ p(d) = q(d)) \to \Sigma_{d:D}\ p(d) = \Sigma_{d:D}\ q(d)$
A6	$x + \delta = x$		
		Bool1	$\neg(\mathsf{t} = \mathsf{f})$
		Bool2	$\neg(b = \mathsf{t}) \to b = \mathsf{f}$
B1	$x \cdot \tau = x$	C1	$x \lhd \mathsf{t} \rhd y = x$
B2	$z \cdot (\tau \cdot (x + y) + x) = z \cdot (x + y)$	C2	$x \lhd \mathsf{f} \rhd y = y$

Table 2: pCRL axioms

SUM6	$\Sigma_{d:D}(p(d) \mathbin{\lfloor\!\lfloor} z) = (\Sigma_{d:D}\ p(d)) \mathbin{\lfloor\!\lfloor} z$	CF	$a(d) \mid b(e) = \begin{cases} \gamma(a,b)(d) & \text{if } d = e \text{ and } \gamma(a,b) \text{ defined} \\ \delta & \text{otherwise} \end{cases}$
SUM7	$\Sigma_{d:D}(p(d) \mid z) = (\Sigma_{d:D}\ p(d)) \mid z$		
SUM8	$\Sigma_{d:D}(\partial_H(p(d))) = \partial_H(\Sigma_{d:D}\ p(d))$		
SUM9	$\Sigma_{d:D}(\tau_I(p(d))) = \tau_I(\Sigma_{d:D}\ p(d))$		
SUM10	$\Sigma_{d:D}(\rho_R(p(d))) = \rho_R(\Sigma_{d:D}\ p(d))$	CD1	$\delta \mid x = \delta$
		CD2	$x \mid \delta = \delta$
CM1	$x \parallel y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$	CT1	$\tau \mid x = \delta$
CM2	$c \mathbin{\lfloor\!\lfloor} x = c \cdot x$	CT2	$x \mid \tau = \delta$
CM3	$c \cdot x \mathbin{\lfloor\!\lfloor} y = c \cdot (x \parallel y)$		
CM4	$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$	DD	$\partial_H(\delta) = \delta$
CM5	$c \cdot x \mid c' = (c \mid c') \cdot x$	DT	$\partial_H(\tau) = \tau$
CM6	$c \mid c' \cdot x = (c \mid c') \cdot x$	D1	$\partial_H(a(d)) = a(d)$ if $a \notin H$
CM7	$c \cdot x \mid c' \cdot y = (c \mid c') \cdot (x \parallel y)$	D2	$\partial_H(a(d)) = \delta$ if $a \in H$
CM8	$(x + y) \mid z = x \mid z + y \mid z$	D3	$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$
CM9	$x \mid (y + z) = x \mid y + x \mid z$	D4	$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$

Table 3: Primary μCRL axioms

TID	$\tau_I(\delta) = \delta$	
TIT	$\tau_I(\tau) = \tau$	
TI1	$\tau_I(a(d)) = a(d)$	if $a \notin I$
TI2	$\tau_I(a(d)) = \tau$	if $a \in I$
TI3	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$	
TI4	$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$	

Table 4: Secondary μCRL axioms

SC1	$x \mathbin{\lfloor\!\lfloor} (y \parallel z) = (x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z$
SC2	$x \mathbin{\lfloor\!\lfloor} \delta = x\,\delta$
SC3	$x \mid y = y \mid x$
SC4	$(x \mid y) \mid z = x \mid (y \mid z)$
SC5	$(x \mid y) \mathbin{\lfloor\!\lfloor} z = x \mid (y \mathbin{\lfloor\!\lfloor} z)$
Handshaking	$(x \mid y) \mid z = \delta$

Table 5: Standard Concurrency and Handshaking