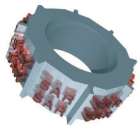
Master UPMC Sciences et Technologies, mention Informatique
Spécialité Systèmes et Applications Réparties

Réalisation Assistée d'Applications Réparties



Examen

B.Bérard, F. Kordon & G. Lasnier

17 Novembre 2010

Durée : 2h

Les téléphones portables doivent être éteints et dans les sacs

Chacune des parties de ce sujet doit être traitée sur une copie différente

Avant-propos

Toutes vos réponses doivent être claires et justifiées (une réponse non justifiée peut être considérée comme fausse). Les barêmes associés aux questions sont indicatifs.
Ne confondez pas « bonne réponse » avec « longue réponse » ;-)

Partie I
Prototypage, génération automatique de programmes (10 points)

1 Sur le cours

Question I.1 (1 point) Dans l'article de Darren Corfer, Citez deux langages de spécification mis en avant dans sa chaîne de traduction.

Question I.2 (2 points) Présentez brièvement (15 lignes max) quels sont les différents avantages que l'on est en droit d'attendre d'une démarche de développement par prototypage (ou dirigée par les modèles) en supposant qu'elle soit bien outillée.

2 Génération de programmes

Nous souhaitons générer un programme associé au réseau de Petri de la figure 1. L'invocation d'un outil de calcul des propriétés structurelles nous ramène le calcul des invariants suivants :

F_1 $i+h+g=cst$
F_2 $h+g+f=cst$
F_3 $k+j+i=cst$
F_4 $k+j+f=cst$
F_5 $d+c+b+a=cst$
F_6 $p+e+h+2\times a=cst$
F_7 $d+l+2\times m+k=cst$
F_8 $p+o+n+m=cst$

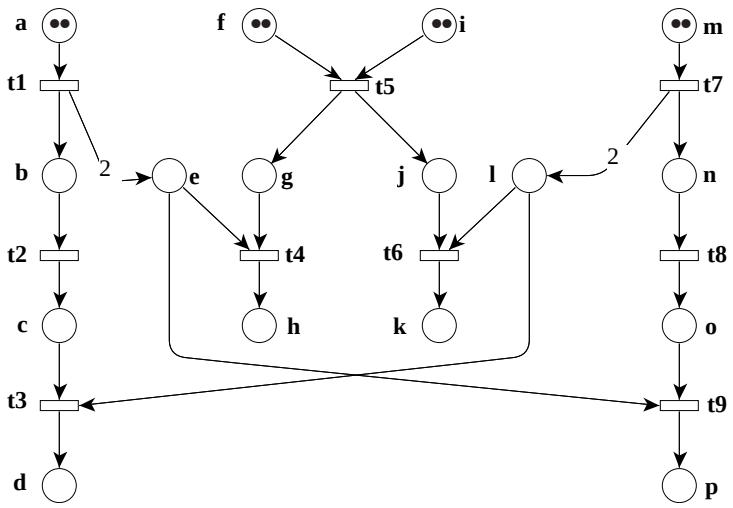


FIGURE 1 – Le réseau de Petri à étudier.

Question I.3 (1 point) Une analyse préliminaire de ces invariants nous permet-elle d'en suprimer certains ? Justifiez votre réponse

Question I.4 (1 point) Peut-on proposer un (ou plusieurs) partitionnement(s) du modèle ? expliquez brièvement pourquoi et explicitez ce (ces) partitionnement(s).

Question I.5 (1 point) Que constatez vous entre certaines décompositions ?

Question I.6 (1 point) Indiquez tous les objets de génération associés au(x) partitionnement(s) que vous proposez.

Question I.7 (1 point) Pouvez-vous indiquer combien de threads aura l'implémentation de cette spécification (vous devez le faire pour chaque décomposition que vous aurez trouvée) ?

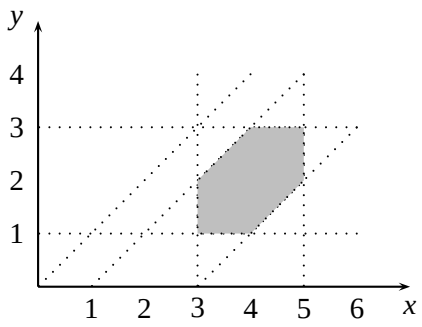
Question I.8 (1 point) Si on considère une stratégie de gestion ≪hibryde≫ des ressources, quel partitionnement proposez-vous ? justifiez brièvement votre réponse.

Question I.9 (1 point) Quel placement des entités du code généré suggérez-vous sur une machine quadri-processeurs ?

Partie II
Sémantique d'exécution des applications réparties (6 points)

3 Automates temporisés

Question II.1 Pour l'ensemble d'horloges $X = \{x, y\}$, on considère la zone Z dessinée ci-dessous.



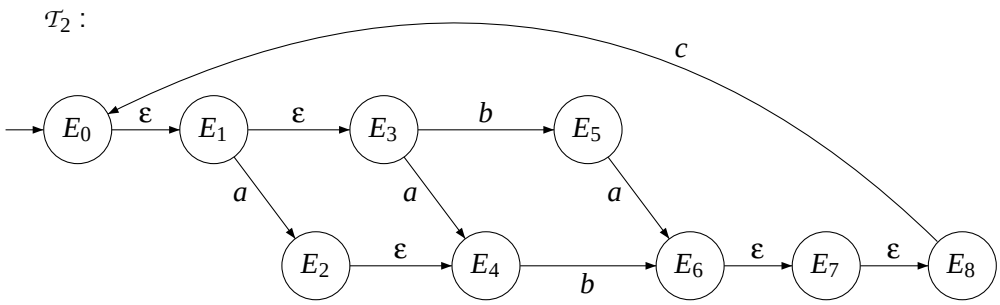
1. Donner une conjonction de contraintes qui la définit. Donner les contraintes définissant le futur de Z.
2. Construire un automate temporisé $\mathcal{A}$ avec la propriété suivante : $\mathcal{A}$ comporte un état q pour lequel l'ensemble des valuations v obtenues en arrivant dans q est exactement Z.

4 Algèbres de processus

Question II.2.
On considère les processus définis dans l'algèbre de processus du cours par :

$$F_1 \stackrel{def}{=} a.F_2 + b.F_3 \qquad F_2 \stackrel{def}{=} b.F_4 \qquad F_3 \stackrel{def}{=} a.F_4 \qquad F_4 \stackrel{def}{=} a.F_1$$

1. Dessiner le système de transition $\mathcal{T}_1$ associé.
2. Comparer les systèmes $\mathcal{T}_1$ et $\mathcal{T}_2$, où $\mathcal{T}_2$ est le système représenté ci-dessous.



Partie III
Modélisation en AADL (4 points)

5 Questions de cours

Question III.1 (0,5 point) Qu'est ce qu'un langage de description d'architecture (ADL) ?

Question III.2 (0,5 point) Que peut on faire avec le langage AADL ?

6 Les navettes (3 points)

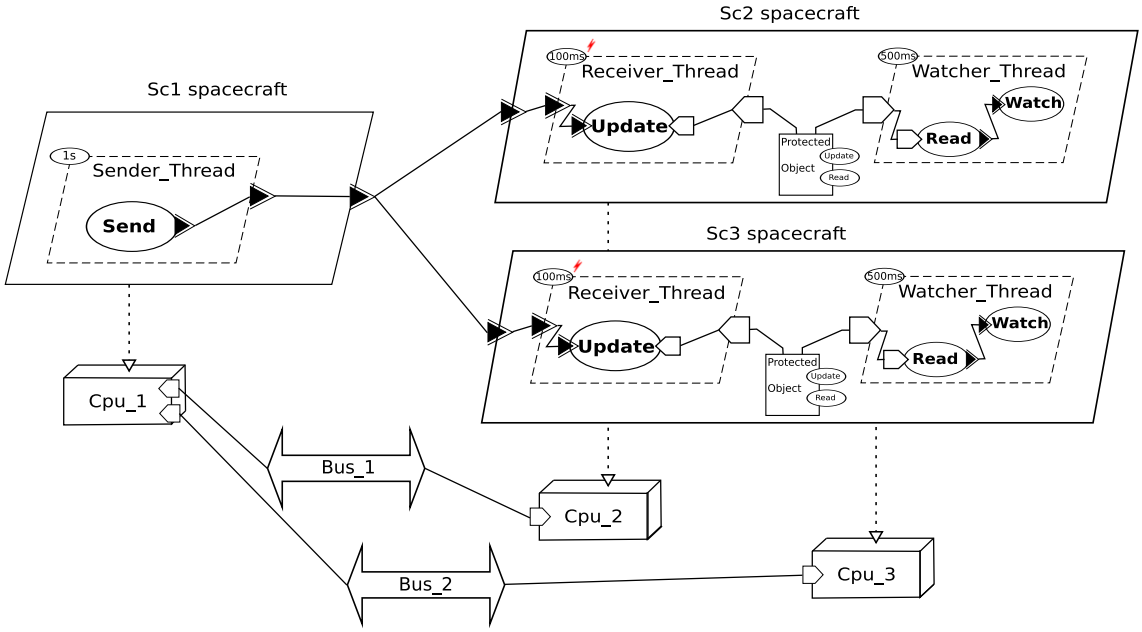


FIGURE 2 – Cas détude MPC : représentation graphique

La figure 2 décrit le cas d'étude "*Multi-Platform Cooperation*" (MPC) dans sa représentation graphique AADL.
Le système contient trois navettes spatiales ayant des rôles différents. SC_1 est la navette leader qui contient un thread périodique (`Sender_Thread`, période = 1s) qui envoie sa position à SC_2 et SC_3 qui le suivent. Ils reçoivent la position envoyée par SC_1 à l'aide d'un thread sporadique (`Receiver_Thread`), mettent à jour leur position respective et sauvegarde cette position dans un objet protégé local. Un second thread (`Watcher_Thread`, période = 500ms) lit périodiquement les données à partir de l'objet protégé et les affiche.
Les listing 1, 2, 3 et 4 représentent notre cas d'étude dans sa version AADL textuelle.

```
---------------
-- Processor --
---------------

processor Simple_CPU
features
  ETH_1 : requires bus access Ethernet_Bus;
end Simple_CPU;
```

```
processor Complex_CPU
features
  ETH_2 : requires bus access Ethernet_Bus;
  ETH_3 : requires bus access Ethernet_Bus;
end Complex_CPU;

processor implementation Simple_CPU.P4
end Simple_CPU.P4;

processor implementation Complex_CPU.P4
end Complex_CPU.P4;

---------
-- Bus --
---------

bus Ethernet_Bus
end Ethernet_Bus;

-----------------
-- Subprograms --
-----------------

subprogram Update
features
  Update_Value : in parameter Record_Type.Impl;
  This       : requires data access Protected_Type.Impl;
end Update;
-- Updates the protected local object

subprogram Read
features
  Read_Value : out parameter Record_Type.Impl;
  This     : requires data access Protected_Type.Impl;
end Read;
-- Reads the value of the local Object

subprogram Observe_Object
features
  Data_Source : out parameter Record_Type.Impl;
end Observe_Object;
-- Sends a new value of the local object to a remote node

subprogram Watch_Object_Value
features
  Read_Value : in parameter Record_Type.Impl;
end Watch_Object_Value;
```

Listing 1 – Cas d'étude MPC: composants matériels et sous-programmes

```
-------------
-- Threads --
-------------

-- Sender

thread Sender_Thread
features
  Data_Source : out event data port Record_Type.Impl;
properties
```

```
  -- A COMPLETER

end Sender_Thread;

thread implementation Sender_Thread.Impl
calls {
  Send : subprogram Observe_Object;
};
connections
  parameter Send.Data_Source -> Data_Source;
end Sender_Thread.Impl;

-- Receiver

thread Receiver_Thread
features
  Protected_Local : requires data access Protected_Type.Impl;
  Data_Sink     : in event data port Record_Type.Impl;
properties

  -- A COMPLETER

end Receiver_Thread;

thread implementation Receiver_Thread.Impl
calls {
  Update : subprogram Protected_Type.Update;
};
connections
  data access Protected_Local -> Update.This;
  parameter Data_Sink -> Update.Update_Value;
end Receiver_Thread.Impl;

-- Watcher

thread Watcher_Thread
features
  Protected_Local : requires data access Protected_Type.Impl;
properties

  -- A COMPLETER

end Watcher_Thread;

thread implementation Watcher_Thread.Impl
calls {
  Read  : subprogram Protected_Type.Read;
  Watch : subprogram Watch_Object_Value;
};
connections
  data access Protected_Local -> Read.This;
  parameter Read.Read_Value -> Watch.Read_Value;
end Watcher_Thread.Impl;
```

Listing 2 – Cas d'étude MPC: composants matériels et sous-programmes

```
---------------
-- Processes --
---------------

-- Sender
```

```
process Sender_Process
features
  Data_Source : out event data port Record_Type.Impl;
end Sender_Process;

process implementation Sender_Process.Impl
subcomponents
  Sender : thread Sender_Thread.Impl;
connections

   -- A COMPLETER
  event data port A COMPLETER -> A COMPLETER;
end Sender_Process.Impl;

-- Receiver

process Receiver_Process
features
  Data_Sink : in event data port Record_Type.Impl;
end Receiver_Process;

process implementation Receiver_Process.Impl
subcomponents
  Local_Object : data Protected_Type.Impl;
  Receiver     : thread Receiver_Thread.Impl;
  Watcher      : thread Watcher_Thread.Impl;

connections
  data access Local_Object -> Receiver.Protected_Local;
  data access Local_Object -> Watcher.Protected_Local;

  -- A COMPLETER
  event data port A COMPLETER -> A COMPLETER;
end Receiver_Process.Impl;
```

Listing 3 – Cas d'étude MPC: composants processes

Question III.3 (0,75 point) Avec toutes les informations mises à votre disposition complétez la spécification des threads `Sender_Thread`, `Receiver_Thread` et `Watcher_Thread` (types de threads, période, deadline, etc).

Question III.4 (0,75 point) En analysant la figure 2 et à partir des descriptions des composants (listings 1, 2 et 3) complétez la spécification des connexions process-thread et thread-process.

Le listing 4 décrit le système global représentant le cas d'étude MPC.

```
-- System --
system MPC
end MPC;

system implementation MPC.Impl
subcomponents
  SC_1 : -- A COMPLETER
  SC_2 : -- A COMPLETER
  SC_3 : -- A COMPLETER

  -- The CPUs
  CPU_SC_1 : processor Complex_CPU.P4;
```

```
  CPU_SC_2 : processor Simple_CPU.P4;
  CPU_SC_3 : processor Simple_CPU.P4;

  -- The Ethernet buses
  ETH_1_2 : bus Ethernet_Bus;
  ETH_1_3 : bus Ethernet_Bus;
connections
  bus access ETH_1_2 -> CPU_SC_1.ETH_2;
  bus access ETH_1_2 -> CPU_SC_2.ETH_1;
  bus access ETH_1_3 -> CPU_SC_1.ETH_3;
  bus access ETH_1_3 -> CPU_SC_3.ETH_1;

  -- SC1 to SC2 : A COMPLETER
  event data port A COMPLETER -> A COMPLETER
    {Actual_Connection_Binding => (reference ETH_1_2);};

  -- SC1 to SC3 : A COMPLETER
  event data port A COMPLETER -> A COMPLETER
    {Actual_Connection_Binding => (reference ETH_1_3);};
properties
  -- A COMPLETER
  Actual_Processor_Binding => reference A COMPLETER applies to A COMPLETER;
  Actual_Processor_Binding => reference A COMPLETER applies to A COMPLETER;
  Actual_Processor_Binding => reference A COMPLETER applies to A COMPLETER;
end MPC.Impl;
```

Listing 4 – Cas d'étude MPC: composant systeme

Question III.5 (1 point) Complétez la spécification du système global en y ajoutant les composants manquants décrits dans la figure 2.

Question III.6 (0,5 point) Que permet d'exprimer la propriété *Actual_Processor_Binding* ?