

Etude Bossa

Jean-Paul Bodeveix¹ Mamoun Filali¹ Julia Lawall²
Gilles Muller³

¹IRIT

²DIKU

³EMN

Réunion CORSS
Nancy
12 septembre 2005

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

Modélisation B

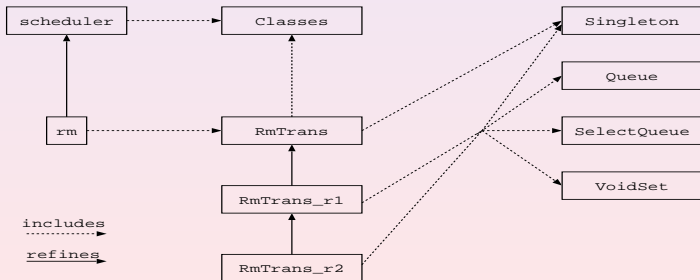
Vérification de propriétés statiques de politiques d'ordonnement

- Vérification des propriétés d'état `Singleton`.
- Respect des pré-post conditions exprimées dans les event types.

Méthode:

- Machine B associée aux event types (scheduler abstrait)
event type \Rightarrow machine abstraite
- Raffinement associé à la politique d'ordonnement.
spécification Bossa \Rightarrow raffinement

Architecture



Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B**
 - **Classes d'états**
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

Classes d'états

MACHINE Classes

SETS Process

VARIABLES

Running, Ready, Blocked, Terminated, running

INVARIANT

Running \subseteq Process

& Ready \subseteq Process

& Blocked \subseteq Process

& Terminated \subseteq Process

& running \in Process

& Running \cap Ready = \emptyset

& Running \cap Terminated = \emptyset

& Running \cap Blocked = \emptyset

& Ready \cap Terminated = \emptyset

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B**
 - Classes d'états
 - scheduler**
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

scheduler

```
MACHINE scheduler
INCLUDES Classes
OPERATIONS
...
/*
[tgt in BLOCKED] -> [tgt in TERMINATED]
*/
Process_end(tgt)  $\triangleq$ 
  PRE tgt : Process & tgt  $\in$  Blocked THEN
    CBlockedToTerminated(tgt)
  END
...
END
```

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B**
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m**
 - La politique R_m
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

RmTrans

MACHINE RmTrans

INCLUDES

Classes,

ru.Singleton(Process), /* *running state* */

rd.SelectQueue(Process,period), /* *ready state* */

yl.Singleton(Process), /* *yield state* */

bl.Queue(Process), /* *blocked state* */

ce.Queue(Process), /* *computation_ended s* */

tm.VoidSet(Process) /* *terminated state* */

INVARIANT

bl.elems \cap ce.elems = \emptyset

\wedge rd.elems \cap yl.elems = \emptyset

\wedge Running = ru.elems

\wedge Ready = rd.elems \cup yl.elems

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B**
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m**
- 4 Vérification automatique avec FMona
 - Mona et FMona
 - Traduction
 - Propriétés
- 5 Conclusion

```
On unblock.preemptive {  
  if (e.target in blocked) {  
    if ((!empty(running)) && (e.target > running))  
      running => ready;  
    }  
    e.target => ready;  
  }  
}
```

REFINEMENT rm

REFINES scheduler

INCLUDES RmTrans /* state transition machine */

VARIABLES

missed_deadlines, timer /* policy specific vari

INVARIANT

missed_deadlines : Process --> NATURAL

& timer : Process --> INTEGER

INITIALISATION

missed_deadlines := Process * {0}

|| timer := Process * {0}

OPERATIONS

```
Unblock_preemptive(tgt)  $\triangleq$   
  VAR isbk IN  
    isbk <-- RMisBlocked(tgt);  
    IF isbk = TRUE THEN  
      VAR hru IN  
        hru <-- RMhasRunning;  
        IF hru = TRUE  $\wedge$   
          period(tgt) < period(running) THEN  
            RMRunning2Ready  
          END;  
          RMBlocked2Ready(tgt)  
        END  
      END  
    END  
  END
```

Vérification automatique avec FMona

- Mona et FMona
- Propriétés vérifiées
- Exemple

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona**
 - Mona et FMona**
 - Traduction
 - Propriétés
- 5 Conclusion

Syntaxe de S1S et WS1S

Définition (Les logiques S1S et WS1S)

Soit $\{x_1, \dots, x_n\}$ un ensemble de variables du premier ordre et $\{X_1, \dots, X_n\}$ un ensemble de variables du second ordre.

- un terme t est défini inductivement par:

$t ::= 0 \mid x_i \mid s(t)s$ est le symbole successeur

- une formule f est définie inductivement par:

$f ::= t \in X_i$ appartenance

$\mid \neg f \mid f \wedge f$

$\mid \exists_1 x_i. f$ quantification du premier ordre

$\mid \exists_2 X_i. f$ quantification du second ordre (sur des ensembles)

Exemple Mona

Mona est un outil qui implante une procédure de décision de la logique WS1S.

Exemple:

```
pred singleton1(var2 S) = ex1 x: S = {x};
```

```
pred singleton2(var2 S) =  
  S  $\neq$   $\emptyset$   $\wedge$  all2 T: T sub S  $\Rightarrow$  (T =  $\emptyset$  | T = S);
```

```
all2 S1, S2:  
  singleton1(S1)  $\wedge$  singleton1(S2)  $\wedge$  S1  $\cap$  S2  $\neq$   $\emptyset$   $\Rightarrow$  S1 = S2;
```

```
# ANALYSIS
```

```
# Formula is valid
```

FMona

- Interface de haut niveau pour Mona
- Définition de types complexes.
- Définition de macros d'ordre supérieur.

Exemple FMona

Mona est un outil qui implante une procédure de décision de la logique WS1S.

Exemple : Accessibilité

```
pred reachable(type State, pred(var State s) init,  
  pred(var State s,s') tr, var State s) =  
  ex array nat of State A:  
    ex nat i: A[i]=s  $\wedge$   
    init(A[0]) $\wedge$ all nat j where j<i: tr(A[j],A[j+1]);
```

```
pred always_true(type State, pred(var State s) p,  
  pred(var State s) init, pred(var State s,s') tr) =  
  all State s: reachable(init,tr,s)  $\Rightarrow$  p(s);
```

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 Vérification automatique avec FMona**
 - Mona et FMona
 - Traduction**
 - Propriétés
- 5 Conclusion

Classes d'états

```
var nat NProc; # nombre maximal de processus  
type Proc = ... NProc; # type intervalle 0..NProc-1
```

```
type StatePB = record {  
  RUNNING: set of Proc;  
  READY: set of Proc;  
  POLICY_BLOCKED: set of Proc;  
  KERNEL_BLOCKED: set of Proc;  
  KERNEL_POLICY_BLOCKED: set of Proc;  
  TERMINATED: set of Proc;  
  NOWHERE: set of Proc;  
};
```

Traduction des event types

```
block.*: [tgt in RUNNING] -> [tgt in KERNEL_BLOCKED]
```

```
pred Eblock_PB(var Proc src, tgt, var StatePB s,s') =  
  (({tgt} sub s.RUNNING)  $\wedge$  (s' = s with {  
    RUNNING := s.RUNNING \ {tgt};  
    READY := s.READY \ {tgt};  
    TERMINATED := s.TERMINATED \ {tgt};  
    POLICY_BLOCKED := s.POLICY_BLOCKED \ {tgt};  
    KERNEL_POLICY_BLOCKED := s.KERNEL_POLICY_BLOCKED \ {tgt};  
    NOWHERE := s.NOWHERE \ {tgt};  
    KERNEL_BLOCKED := s.KERNEL_BLOCKED \ {tgt}  $\cup$  {tgt};  
  }));
```

Espace concret

```
type BState = record { # on associe à chaque état Bossa un ensemble de processus  
  running: set of Proc;  
  ready: set of Proc;  
  yield: set of Proc;  
  blocked: set of Proc;  
  computation_ended: set of Proc;  
  terminated: set of Proc;  
};
```

Traduction des handlers

```
On block.* { e.target => blocked; }
```

```
pred Bblock_(var Proc src, tgt, var BState s,s') =  
preabs(BState2StatePB,Eblock_PB(src,tgt),s) ^  
(s with {  
  running := (s.running \ {tgt});  
  ready := (s.ready \ {tgt});  
  yield := (s.yield \ {tgt});  
  blocked := (s.blocked ∪ {tgt});  
  computation_ended := (s.computation_ended \ {tgt});  
  terminated := (s.terminated \ {tgt});  
} = s')
```

Outline

- 1 Objectifs
- 2 Architecture
- 3 Machines B
 - Classes d'états
 - scheduler
 - Espace d'états de la politique R_m
 - La politique R_m
- 4 **Vérification automatique avec FMona**
 - Mona et FMona
 - Traduction
 - **Propriétés**
- 5 Conclusion

Propriétés étudiées

- event types:
 - conservation du nombre de processus,
 - cardinalité de `RUNNING`,
 - respect des préconditions des événements obligatoires.
- handlers:
 - cardinalité des états singletons.
 - conformité avec les events types.
 - respect des préconditions.

Propriétés vérifiées

- au niveau B: (preuve assistée).
- au niveau FMona (prise en compte de l'accessibilité): automatique.

Remarques:

- Traitement de toutes les propriétés vérifiées par Bossa.
- Génération de code optimisé non considérée.

Conclusion

- MSO adapté à ce type de problèmes.
- Perspectives:
 - Propriétés applicatives.
 - Génération de code optimisé.
 - Génération de code certifié.