Un processus de développement Event-B pour des applications distribuées.

B. Siala M. Tahar Bhiri J.-P. Bodeveix M. Filali

IRIT-CNRS ; Université de Sfax ; Université de Toulouse

AFADL 2016
Besançon 7-8 Juin

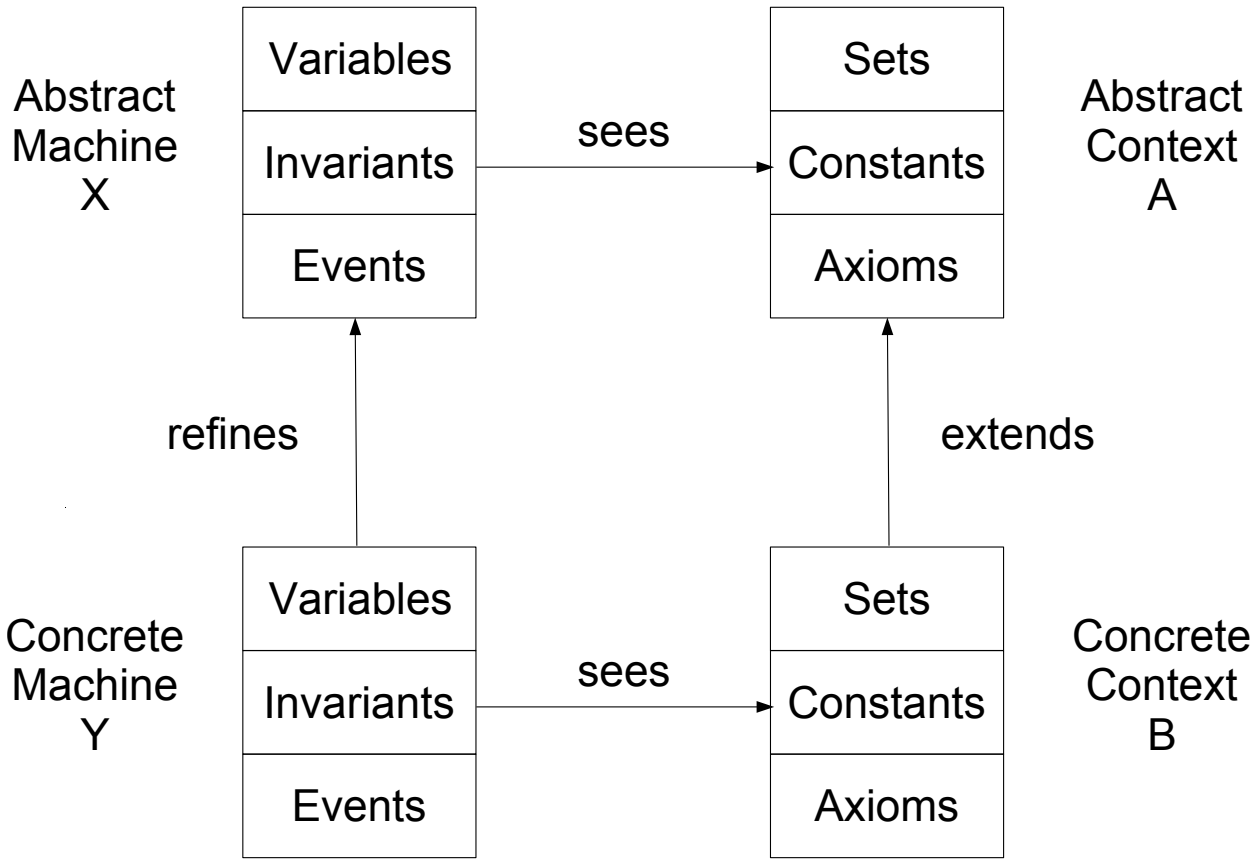
Contexte

- Le développement des systèmes distribués **sûrs** reste un défi pour les informaticiens :
  - Problèmes liés à la spécification
  - Problèmes liés à l'obtention d'un code correct
- $\rightsquigarrow$ Étude :
  - méthodologie de développement baseé sur le raffinement : Event-B.
  - langages dédiés (DSLs) : plugin Xtext.
  - plateforme d'exécution distribuée : BIP.

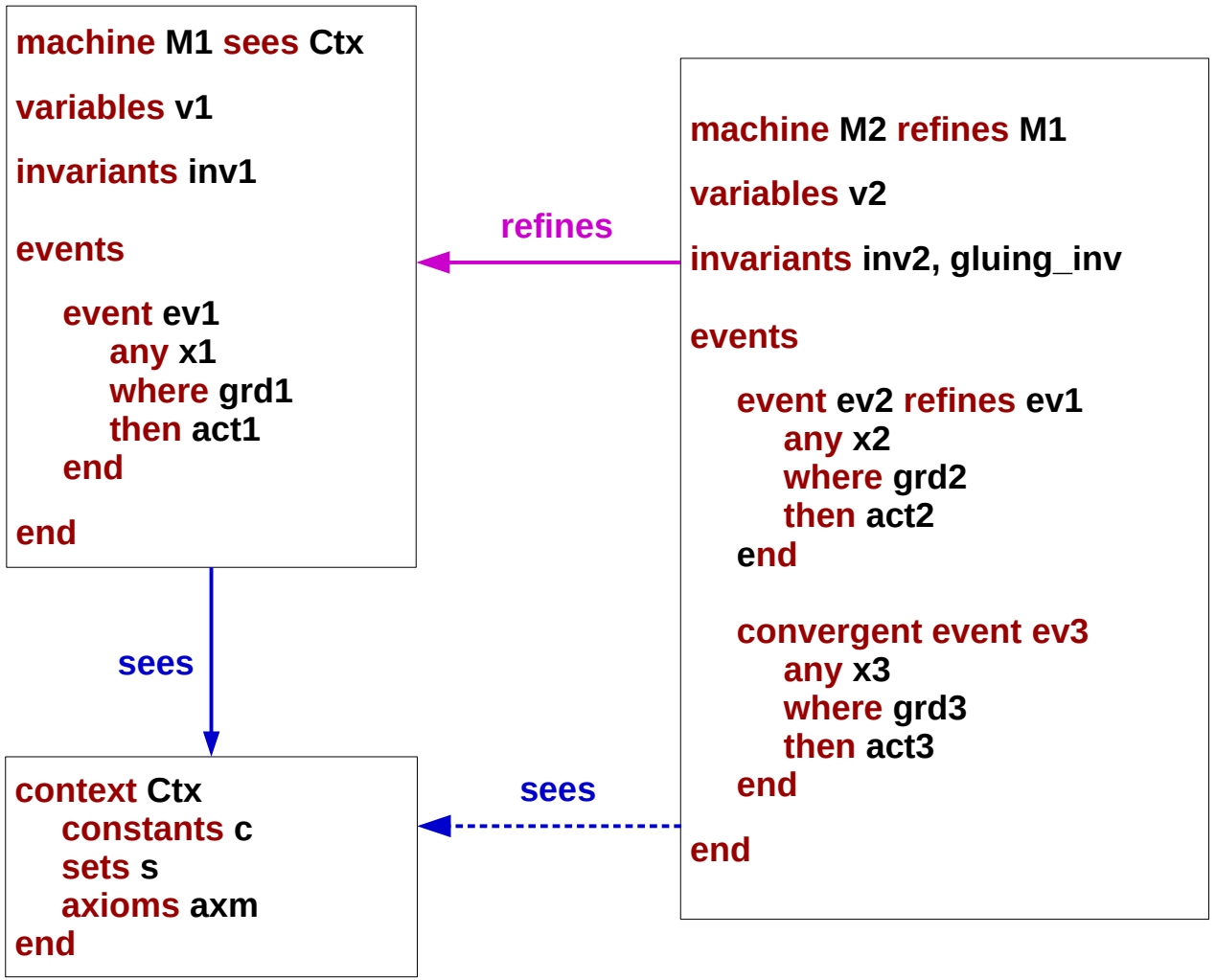
Plan

1. Méthode Event-B.
2. Langage BIP.
3. Processus de développement
   - Étape de fragmentation.
   - Étape de distribution.
   - Étape de génération de code.
4. Conclusion.

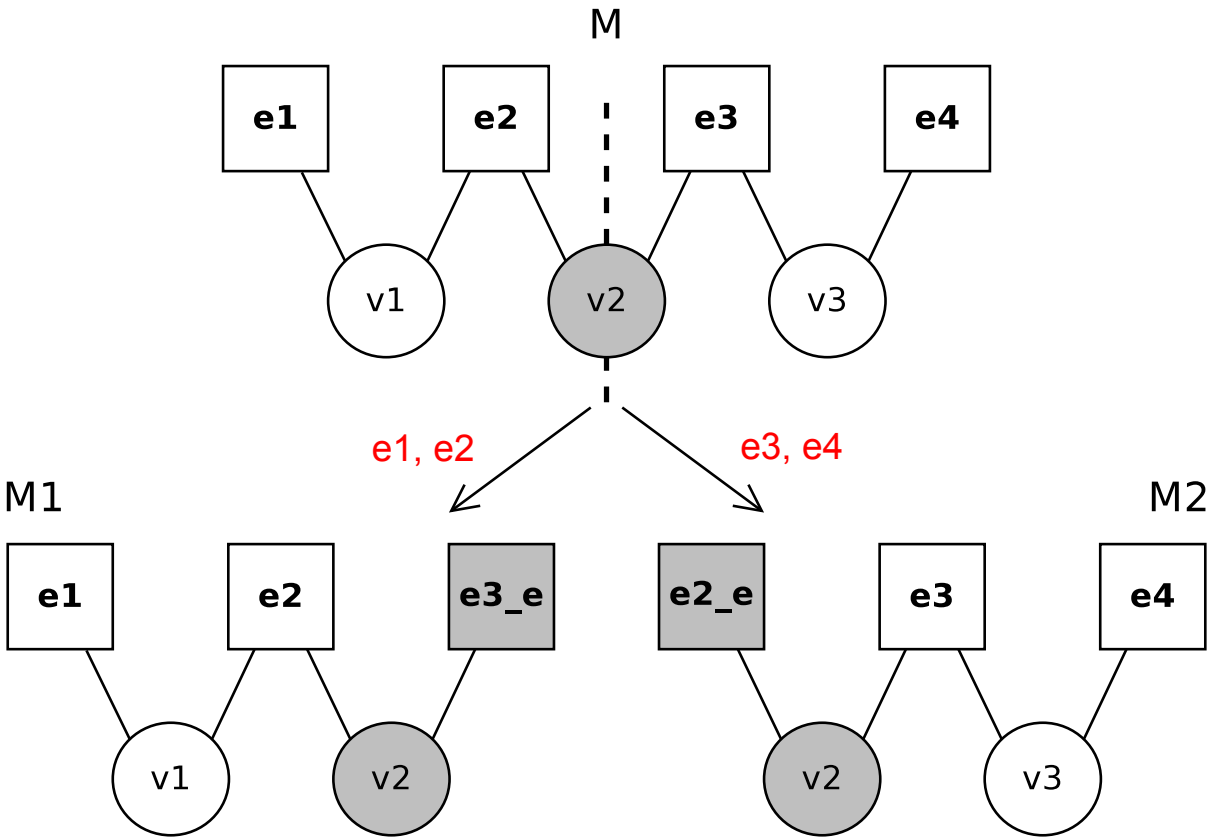
Méthode Event-B



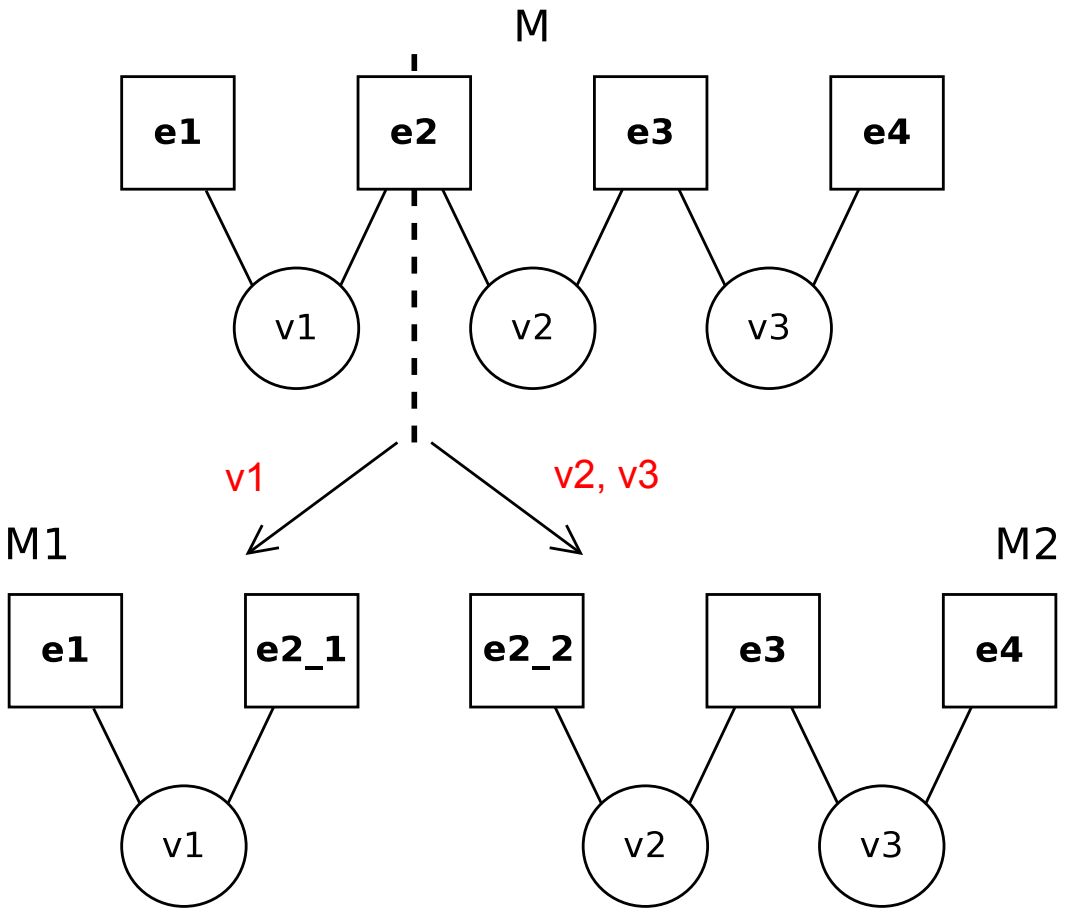
Raffinement : simulation faible



Décomposition par variable partagée



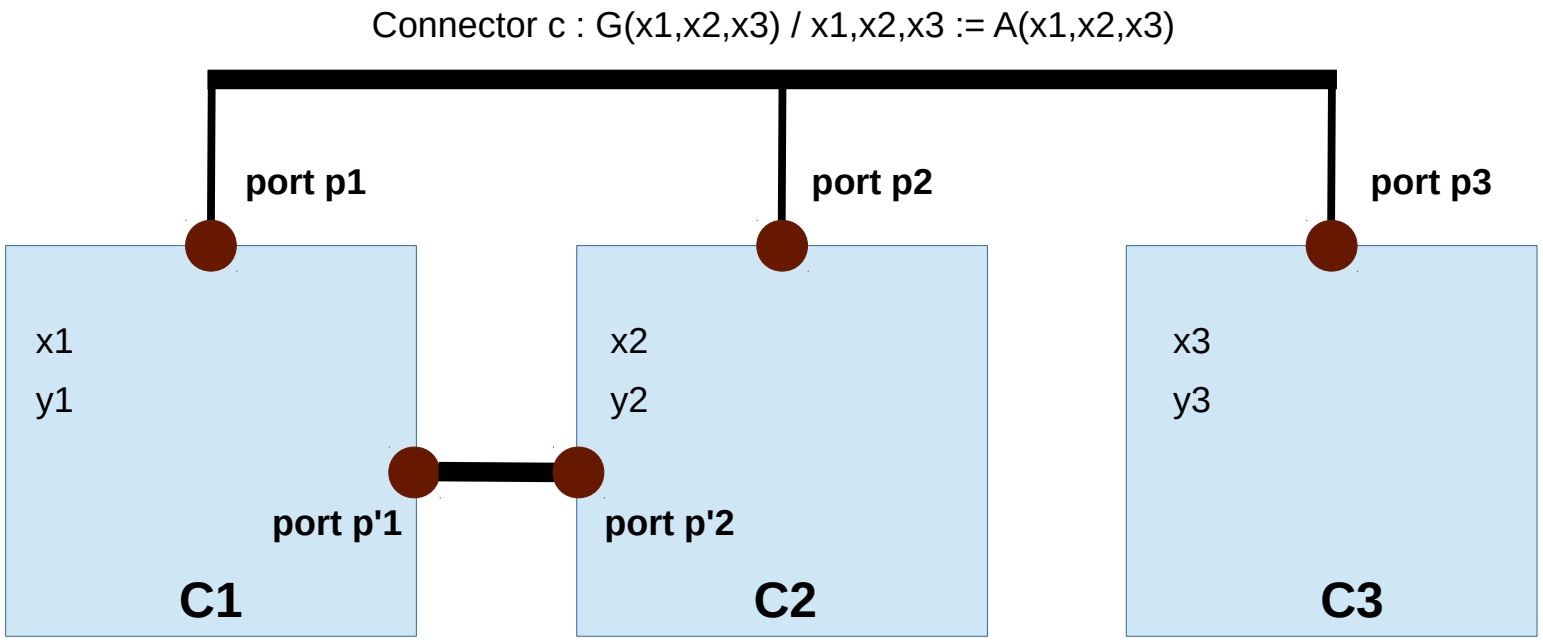
Décomposition par événement partagé



Langage BIP

- Langage à base de composants (composants, ports, connecteurs n-aire [calcul + transfert de données])
- Plateforme d'exécution distribuée
- Détection d'interblocage et Vérification

Synchronisation BIP



on p1
provided G1(x1,y1)
do x1,y1:= A1(x1,y1)

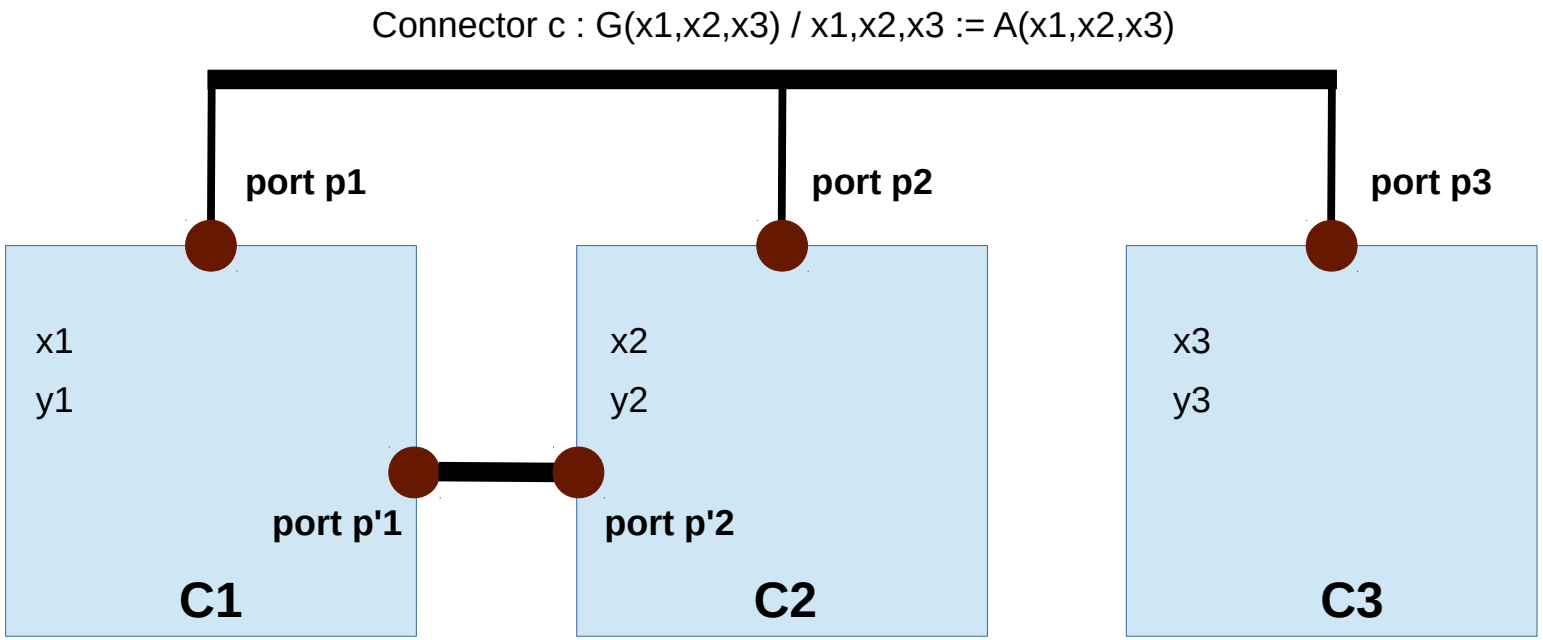
on p2
provided G2(x2,y2)
do x2,y2:= A2(x2,y2)

on p3
provided G3(x3,y3)
do x3,y3:= A3(x3,y3)

on p'1
provided G'1(x1,y1)
do x1,y1:= A'1(x1,y1)

on p'2
provided G'2(y2)
do x2,y2:= A'2(x2,y2)

Synchronisation BIP



on p1
provided G1(x1,y1)
do x1,y1:= A1(x1,y1)
1

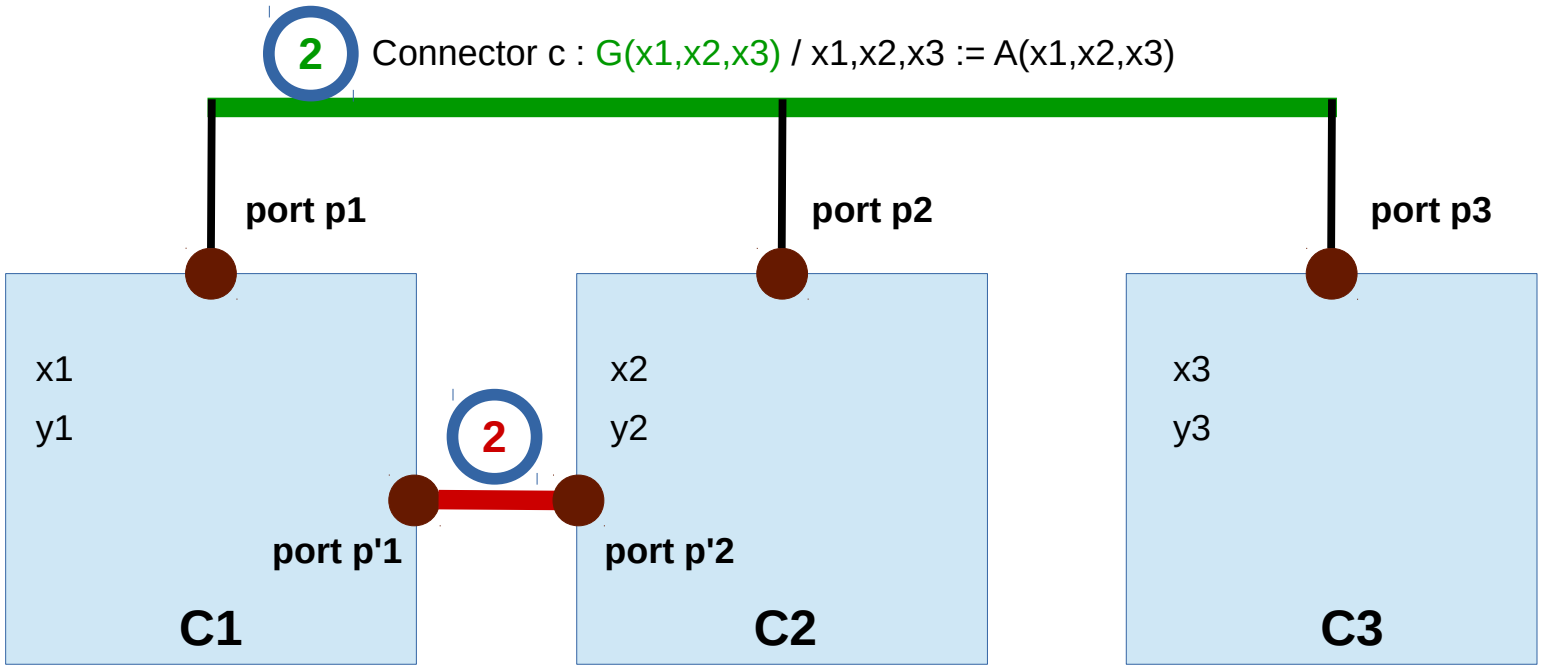
on p2
provided G2(x2,y2)
do x2,y2:= A2(x2,y2)
1'

on p3
provided G3(x3,y3)
do x3,y3:= A3(x3,y3)
1''

on p'1
provided G'1(x1,y1)
do x1,y1:= A'1(x1,y1)
1

on p'2
provided G'2(y2)
do x2,y2:= A'2(x2,y2)
1'

Synchronisation BIP



on p1
provided G1(x1,y1) (1)
do x1,y1:= A1(x1,y1)

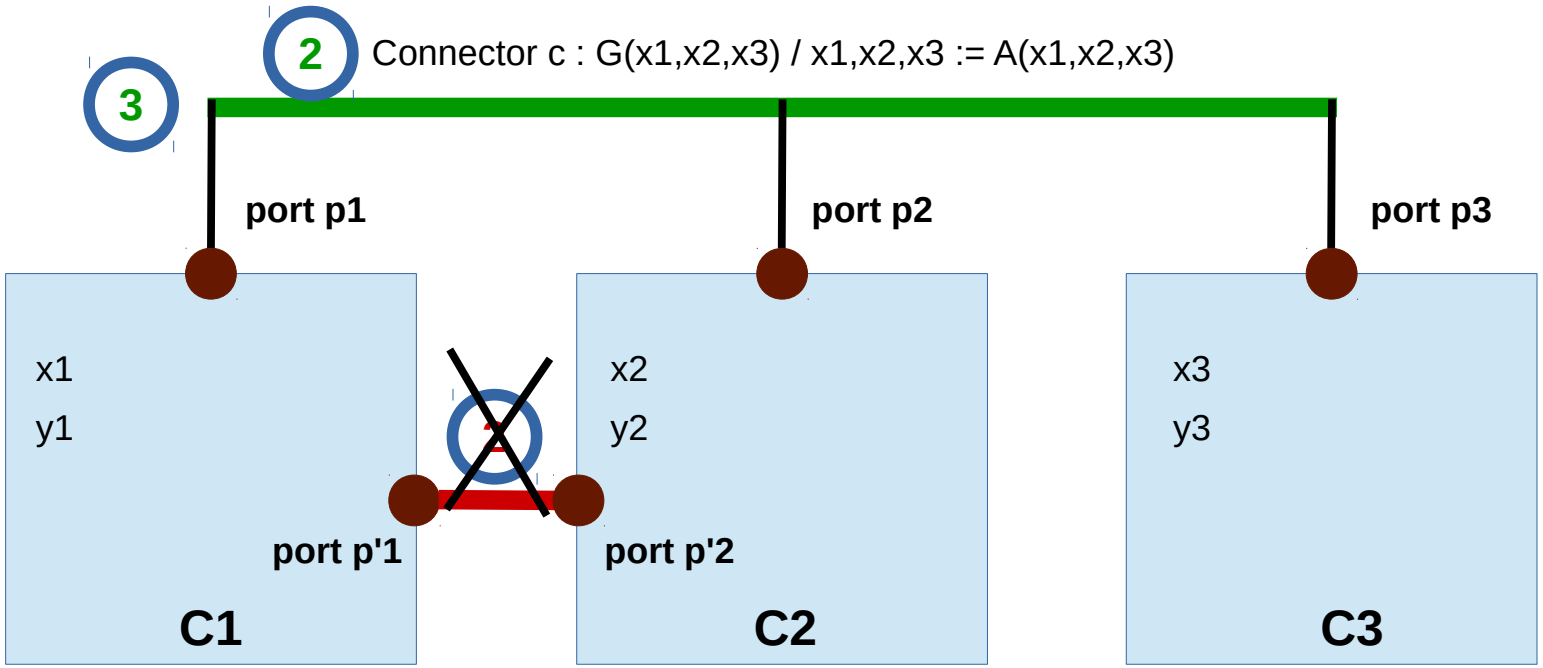
on p2
provided G2(x2,y2) (1')
do x2,y2:= A2(x2,y2)

on p3
provided G3(x3,y3) (1'')
do x3,y3:= A3(x3,y3)

on p'1
provided G'1(x1,y1) (1)
do x1,y1:= A'1(x1,y1)

on p'2
provided G'2(y2) (1')
do x2,y2:= A'2(x2,y2)

Synchronisation BIP



on p1
provided G1(x1,y1) (1)
do x1,y1:= A1(x1,y1)

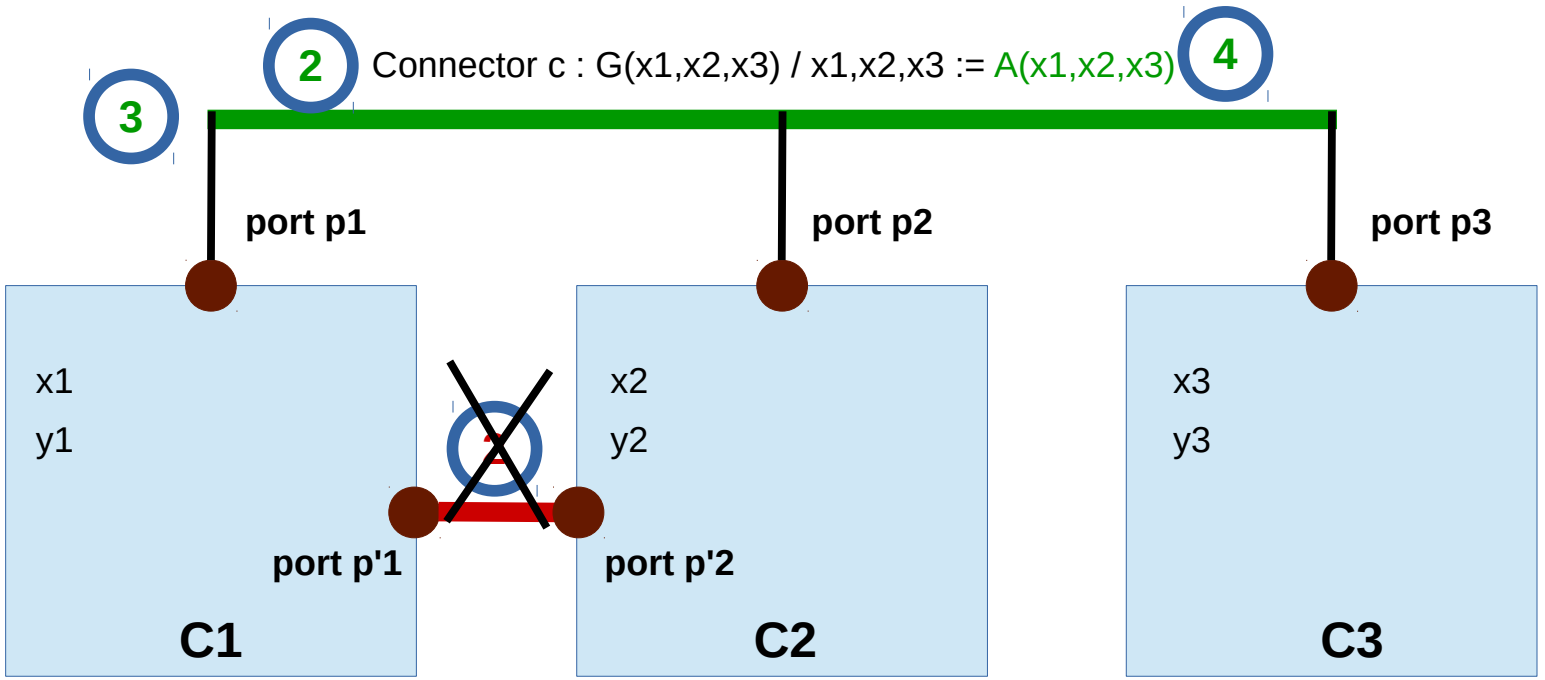
on p2
provided G2(x2,y2) (1')
do x2,y2:= A2(x2,y2)

on p3
provided G3(x3,y3) (1'')
do x3,y3:= A3(x3,y3)

on p'1
provided G'1(x1,y1) (1)
do x1,y1:= A'1(x1,y1)

on p'2
provided G'2(y2) (1')
do x2,y2:= A'2(x2,y2)

Synchronisation BIP



on p1
provided G1(x1,y1) (1)
do x1,y1:= A1(x1,y1)

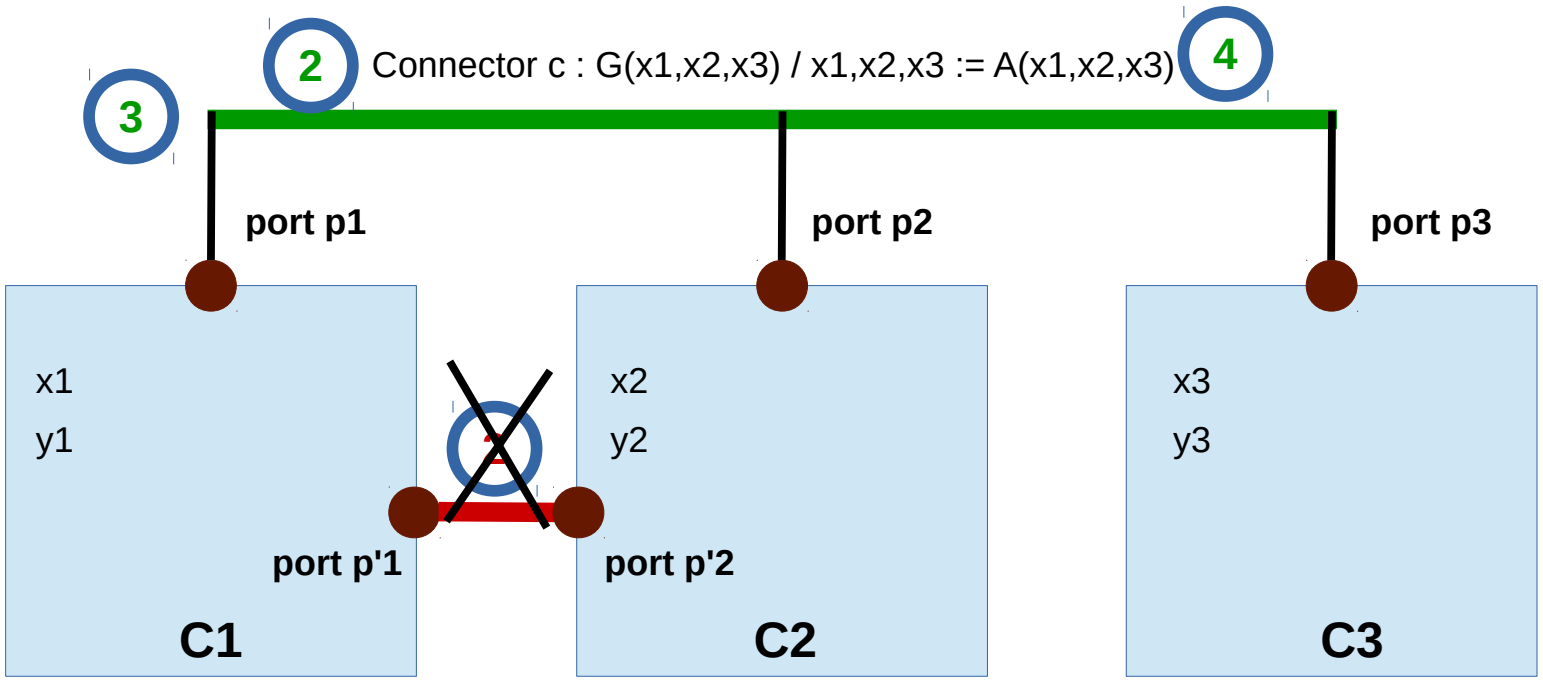
on p2
provided G2(x2,y2) (1')
do x2,y2:= A2(x2,y2)

on p3
provided G3(x3,y3) (1'')
do x3,y3:= A3(x3,y3)

on p'1
provided G'1(x1,y1) (1)
do x1,y1:= A'1(x1,y1)

on p'2
provided G'2(y2) (1')
do x2,y2:= A'2(x2,y2)

Synchronisation BIP



on p1
provided G1(x1,y1)
do x1,y1:= A1(x1,y1)

1
5

on p2
provided G2(x2,y2)
do x2,y2:= A2(x2,y2)

1'
5'

on p3
provided G3(x3,y3)
do x3,y3:= A3(x3,y3)

1''
5''

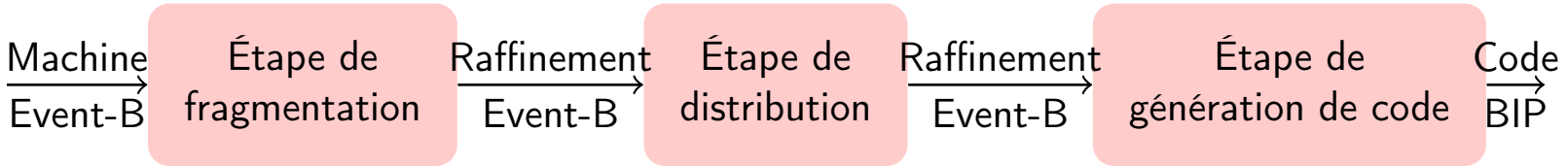
on p'1
provided G'1(x1,y1)
do x1,y1:= A'1(x1,y1)

1

on p'2
provided G'2(y2)
do x2,y2:= A'2(x2,y2)

1'

Processus de développement

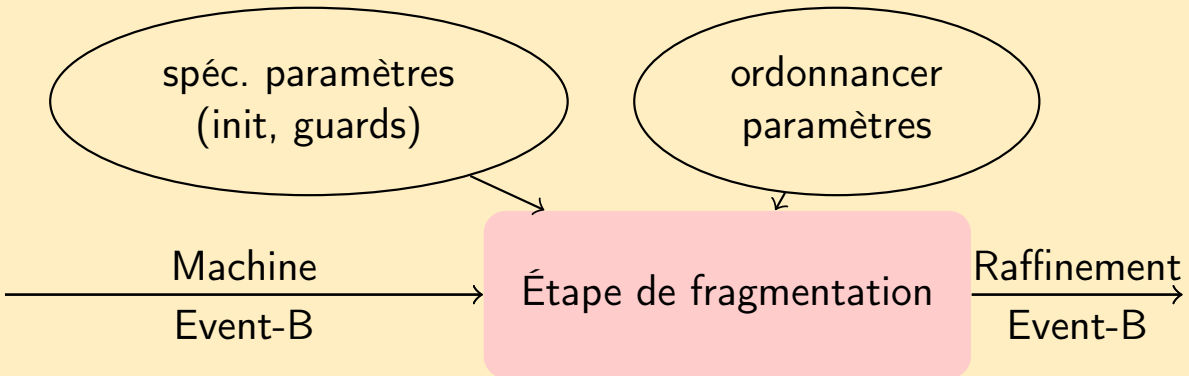


Étapes de transformation

- Étape de fragmentation : répartir le non-déterminisme.
- Étape de distribution : définition des composants et distribution
- Étape de génération de code : d'Event-B vers BIP

Étape de fragmentation

La transformation fragmentation



Langage dédié

```
event ev
      when p1 ... pn parameter p init v with g1 ... gm
      when ...parameter ...
```

Traitement (I)

```
machine input_machine
....
  event ev any p  ... where @gi gi then ... end
...
end
```

```
splitting generated
refines input_machine
events
  event ev
    parameter p init ev_p with gi
end
```

```
machine generated refines input_machine
variables
 ev_p  ev_p_computed // temoin et statut pour p de ev
invariants
  @ev_gi ev_p_computed ⇒ gi // ou p est remplace par ev_p

  ...
end
```

Traitement (II)

```
machine generated refines input_machine
  ...
events
  event INITIALISATION extends INITIALISATION
  then
    @ev_p ev_p := v
    @ev_p_comp ev_p_computed := FALSE
  end

  convergent event compute_ev_p // calcule le param. p de ev
  any p where
   @gi gi // garde specifiant p
   @pi ev_pi_computed = TRUE //parameters dont p depend
                    //ont ete calcules
   @p ev_p_computed = FALSE // p reste a calculer
  then
    @a ev_p := p
    @computed ev_p_computed := TRUE // decroissance du variant
  end
  ...
end
```

Traitement (III)

```
machine generated refines input_machine
  ...
variant  // compteur des parametres restant a calculer
  {FALSE ↦ 1, TRUE ↦ 0}(ev_p_computed) + ...
events
  ...
  event ev refines ev
  when @p_comp ev_p_computed = TRUE
  with
    @p p = ev_p // le parametere p de l'evenement abstrait
         //est raffine en ev_p
  then
    // remplacer p par ev_p dans les actions
      //de l'evenement raffine
  end
end
```

Étape de distribution

- variables de la machine $\mapsto$ sous-composants.
- gardes d'événement $\mapsto$ sous-composants.

Langage dédié

components C_1 ...C_n
mappings
variables vi ...vk $\mapsto$ C_i
...
variables vj $\mapsto$ C_m
...
guards gl $\mapsto$ C_n
...

Distribution : 2 phases

- La phase de réplication : reproduit les variables sur les composants afin de permettre un accès local à des variables distantes (raffinement)
- La phase de projection : répartir la machine sur l'ensemble des sous-composants.

Phase 1 : réplication

Deux problèmes :

- accès distant dans les gardes
  - $\rightarrow$ création de copies + événement **convergent** de synchronisation.
- accès distant dans les actions
  - $\rightarrow$ ajout d'un paramètre à l'événement.

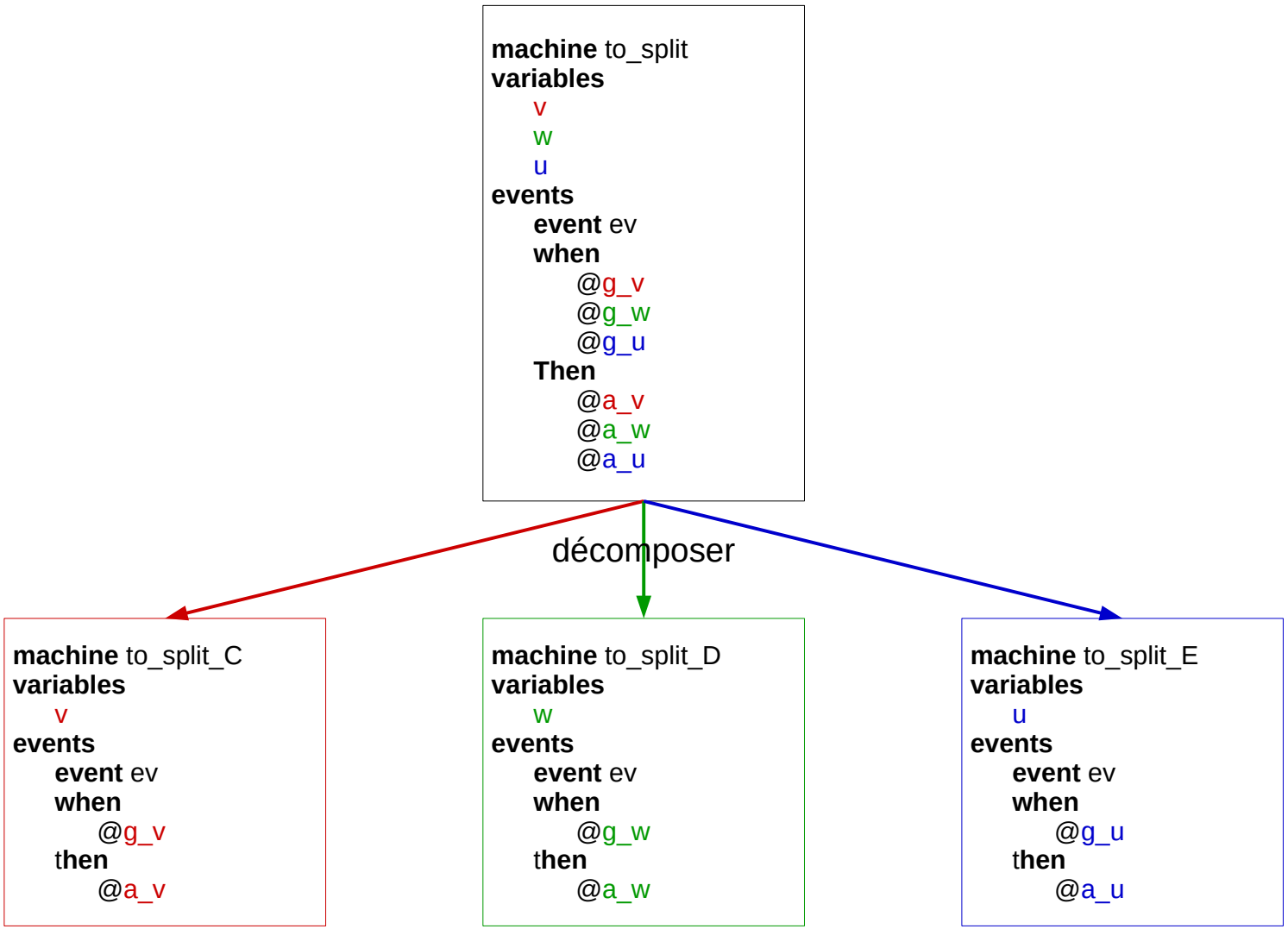
Phase 1 : réplication (garde complexe)

```
machine input_machine
variables
  v  //sur C
  w  //sur D
  u  //sur E
events
  event ev
  when
    @g v>0  // sur D
  then
    @a w := w+v+u  //sur D
  end
end
```

```
machine generated refines input_machine
variables
  v  //sur C
  v_fresh  //sur C
  v_copy  //sur D
  w  //sur D
  u  //sur E
events
  convergent event share_v  //partagé par C et D
  any local_v where
    @g v_fresh = FALSE  //sur C
    @l local_v = v  //sur C
  then
    @to_D v_copy := local_v  //sur D
    @done v_fresh := TRUE  //sur C
  end

  event ev
  when
    @g v_copy>0  //sur D
  then
    @a w := w+v_copy+u  //sur D
  end
end
```

Phase 2 : Projection[SPHB11]

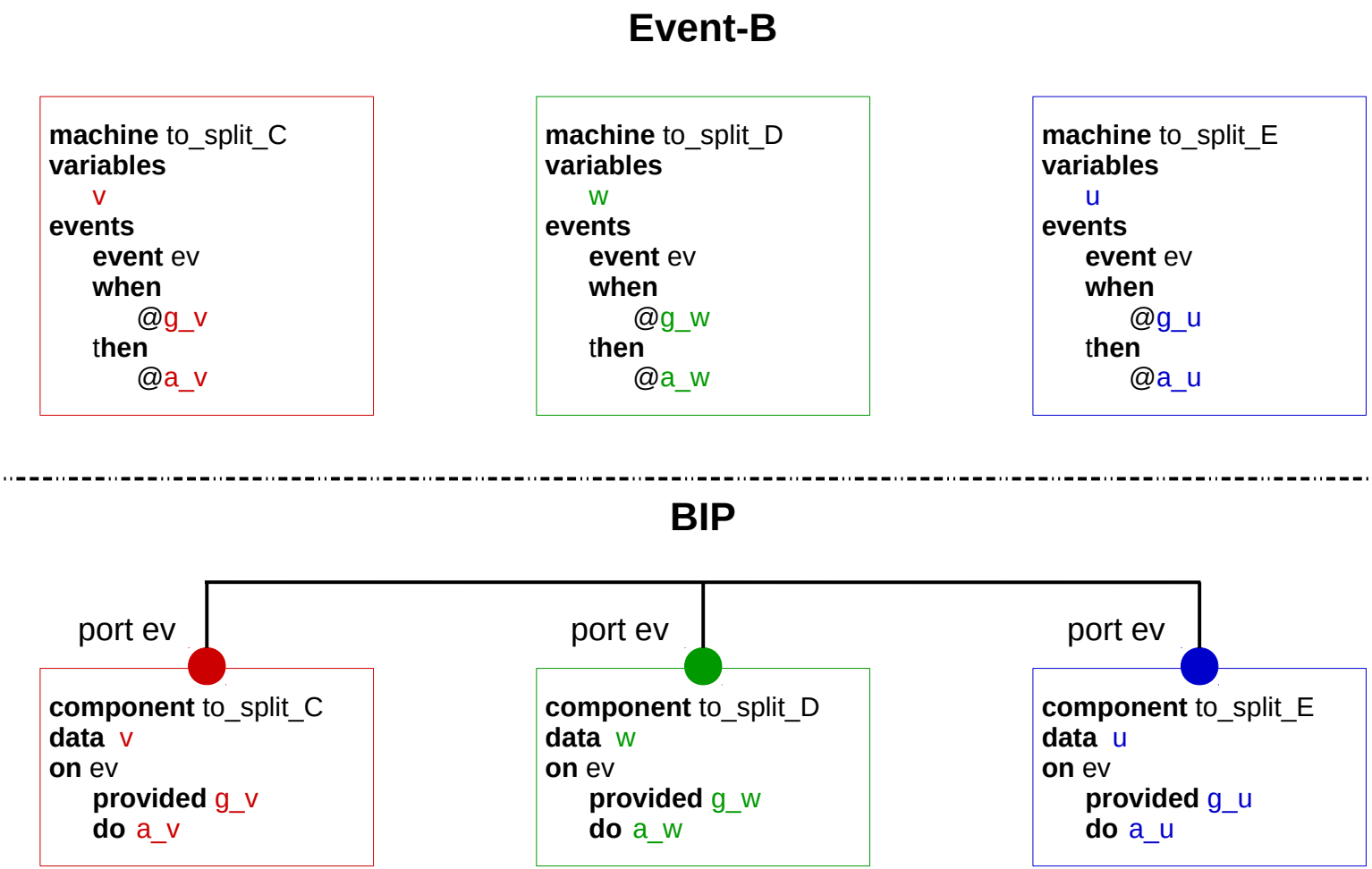


Génération de code

- Génération des composants BIP.
  - machine Event-B $\mapsto$ composant BIP.
  - événement Event-B $\mapsto$un type de port BIP.
- Événement accédant à des variables distantes.

  $\mapsto$ connecteur BIP simple (uniquement .transfert de données).
- Génération d'un composant composite.

Génération de code



Conclusion et perspectives

- Méthode de développement de systèmes distribués **corrects par construction** :
  - Framework Eclipse $\rightsquigarrow$ Framework Rodin.
  - Fragmentation, Distribution $\rightsquigarrow$ plugins Rodin
  - Génération d'une squelette de code BIP $\rightsquigarrow$ génération d'un code complet.
- Améliorer l'outillage de ce processus.
- Finaliser le générateur de code BIP.
- Prouver la correction du processus.

Merci pour votre attention

Questions ?

Références

Renato Silva, Carine Pascal, Thai Son Hoang, and Michael Butler.
Decomposition tool for Event-B.
Software : Practice and Experience, 41(2) :199–208, 2011.