Le langage Fiacre

Frédéric Lang

joint work with
B. Berthomieu, J.-P. Bodeveix, M. Filali,
H. Garavel, F. Peres, R. Saad,
J. Stöcker, and F. Vernadat

Le langage pivot Fiacre

- Langage formel de description inspiré de nombreux travaux : théorie de la concurrence, V-Cotre, NTIF
- Défini dans le contexte de l'ACI Fiacre, du RNTL OpenEmbeDD et du projet de pôle AESE Topcased
- Description du comportement des composants de base
  - Transitions entre états du composant
  - Manipulation de données
  - Communication (messages, variables partagées)
  - Contraintes temporelles (délais, timeouts) et priorités
- Description des compositions (en cours)

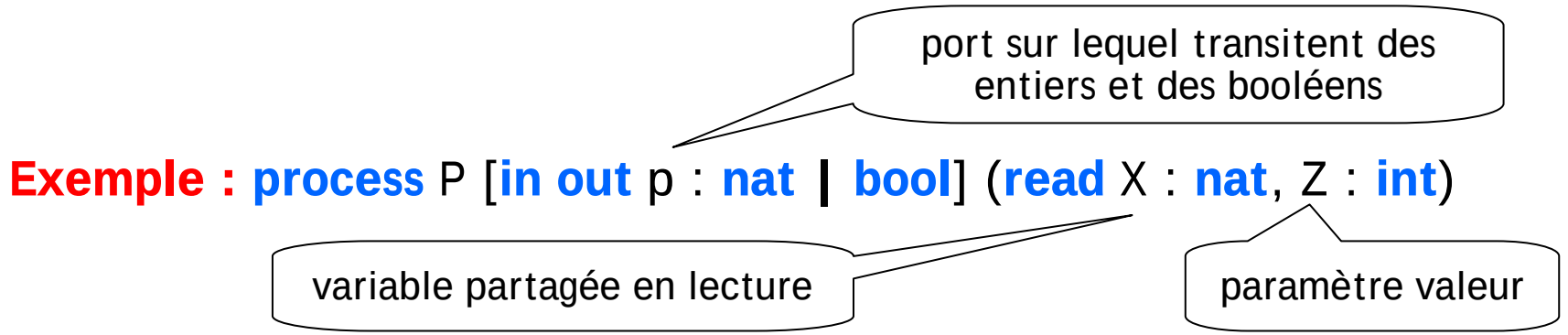
Types de données

- Types de base natifs : **int**, **nat**, **bool**

- Types construits par l'utilisateur
  - intervalles d'entiers (ex: **interval** 2..7)
  - énumérations (ex: **enum** red, green, blue **end**)
  - enregistrements (ex: **record** width, height : **nat end**)
  - tableaux de taille fixe (ex: **array of** 10 **nat**)
  - files de taille bornée (ex: **queue of** 5 **bool**)

Composant de base Fiacre

- Process paramétré par
  - des ports de communication typés et orientés (**in**, **out**)
  - des variables partagées typées avec contrôle d'accès (**read**, **write**)
  - des valeurs typées



- Comportement défini par les transitions

Transitions

- Définies par un état initial et une instruction définissant actions à effectuer et état cible

- Utilisation de structures de contrôle
  - affectations de variables déterministes (X := E)
  - affectations non-déterministes (X := **any where** E)
  - boucles conditionnelles (**while**)
  - branchements déterministes (**if-then-else**)
  - branchements non-déterministes (**select**)
  - composition séquentielle (;)
  - communication par port (émission/réception/synchro)
  - saut vers l'état suivant (**to** s)

Exemple de programme Fiacre (1/2)

```
type tcom is enum total, value end
type index is interval 0..3
type data is array of 4 nat
channel creq is tcom | tcom * index
process Card [pow_on, pow_off : none, in req : creq,
   out resp : nat] (val : data)
states off, on, send_total, send_value init on
var c : tcom, i : index, tot : nat
```

Exemple de programme Fiacre (2/2)

```
from off
   pow_on;
   to on

from on
   select
         req ?c where c = total;
         to send_total
   []
         req ?c, i where c = value;
         to send_value
   []
         pow_off;
         to off
   end
```

```
from send_total
   i := 0; tot := 0;
   while i < 4 do
         tot := tot + val[i];
         i := i + 1
   end;
   resp !tot;
   to on

from send_value
   resp !val[i];
   to on
```

Exécution des transitions

- L'exécution d'une transition est atomique
- Chaque transition définit plusieurs chemins d'exécution
  - En fonction de la valeur des variables
  - En raison du non-déterminisme
- Contrainte (vérifiée statiquement) : au plus une communication via un port sur chaque chemin d'exécution d'une transition

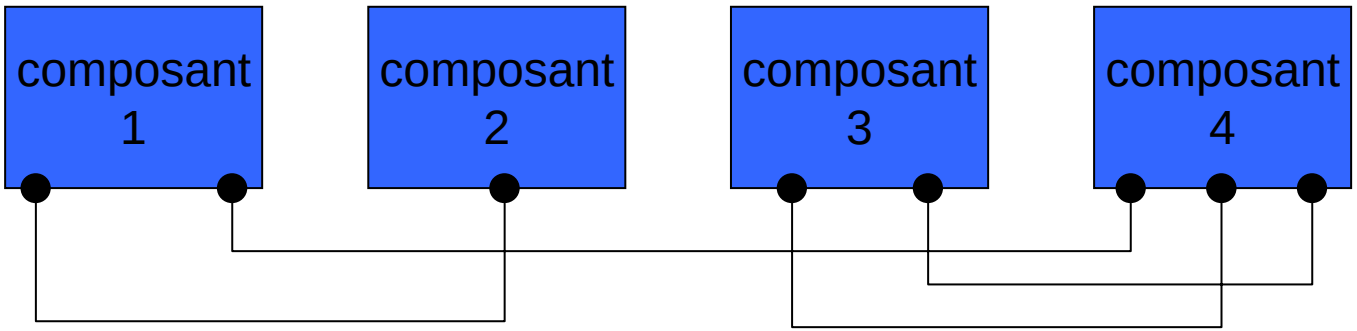
Contraintes temporelles et priorités

- Nommage des transitions et chemins dans les transitions
- Association de délais et priorités aux chemins
- **Exemple**

```
Receive : from on
            select
                req ?c where c == total; to send_total : Request
            []
                resp ?c, i where c == value; to send_value : Response
            []
                pow_off; to off : Abort
            end
priority Response > Request
delay Abort in [5, 8]
```

Composition

- Mise en parallèle d'instances des composants de base
- Définition de connexions entre les ports des composants



- (Syntaxe concrète en cours de définition)

Sémantique

- Sémantique statique (typage, bonne formation)

- Sémantique dynamique formelle
  - Nécessaire tout au long de la chaîne de transformation / vérification pour éviter les ambiguïtés
  - Associe à tout programme Fiacre un graphe d'états à transitions étiquetées par
    - des actions de communications
    - des délais temporels