# Le langage pivot asynchrone Fiacre

**Frédéric Lang**

travail commun avec
B. Berthomieu, J.-P. Bodeveix, M. Filali,
H. Garavel, F. Peres, R. Saad,
J. Stöcker, and F. Vernadat


# Fiacre

(Meta)-modeleur

Langages de Modélisation

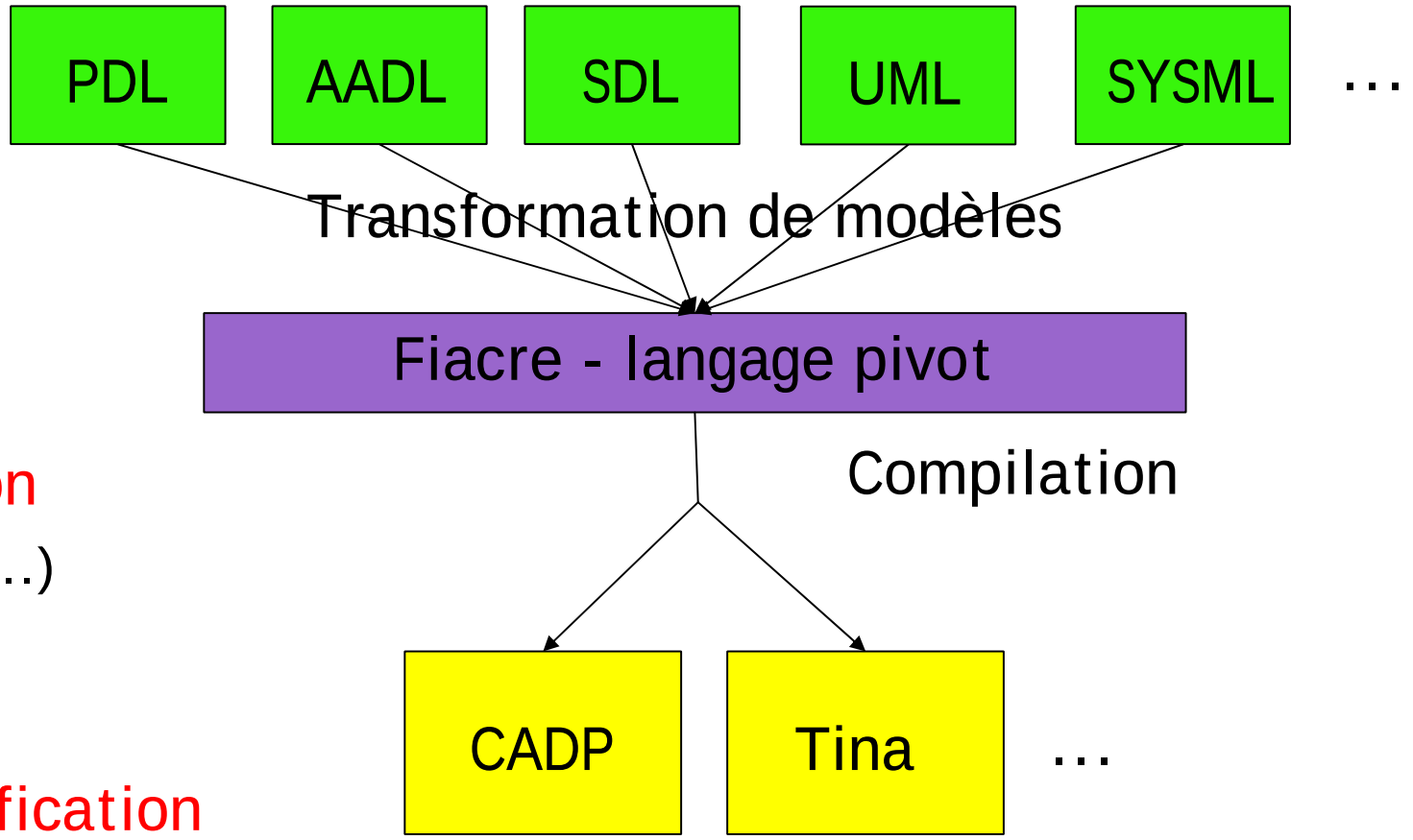
Editeurs

Moteurs de transformation

(ATL,KERMETA,...)

Compilateurs

Outils de vérification



# Fiacre

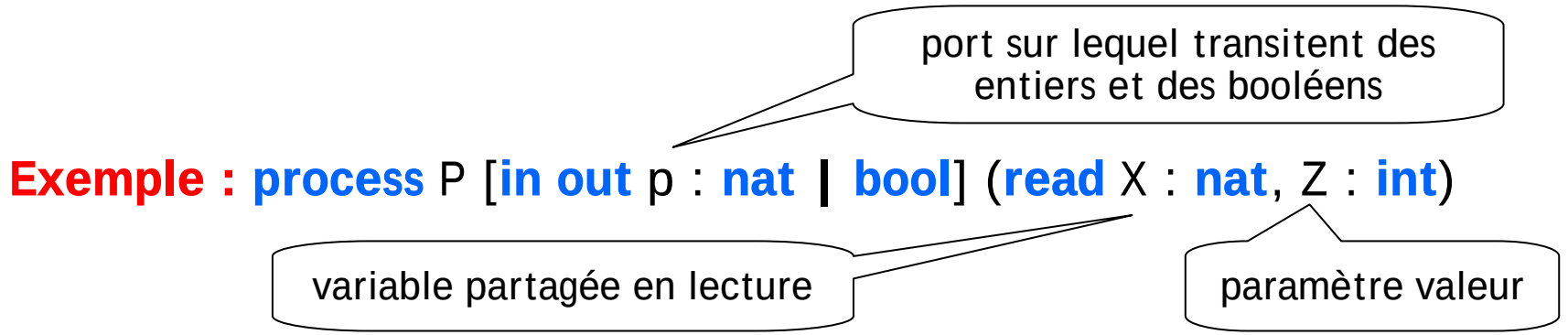
- Langage formel
- Inspiré de la théorie de la concurrence, V-Cotre, NTIF, E-LOTOS
- Description des composants de base
  - Transitions entre états du composant
  - Manipulation de données
  - Communication (messages, variables partagées)
- Description des compositions
  - Mise en parallèle des composants
  - Contraintes temporelles et priorités

# 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
- Définition structurée
  - 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 (;)
  - événement de communication (émission/réception/synchronisation sur un port)
  - saut vers l'état suivant (**to** s)

# Exemple (1/2)

```
type command is enum get_sum, get_value end

type index is interval 0..3

type data is array of 4 nat

channel request is command | command * index
```

type réservé aux ports (permet les unions de types)

# Exemple (2/2)

```
process Proc [in req : request, out resp : nat] (val : data)
states ready, send_sum, send_value init ready
var c : command, i : index, sum : nat

from ready
   select
        req ?c, i where c = get_value;
        to send_value
   []
        req ?c where c = get_sum;
        to send_sum
   end

from send_value
   resp !val[i];
   to ready

from send_sum
   i := 0; sum := 0;
   while i < 4 do
        sum := sum + val[i];
        i := i + 1
   end;
   resp !sum;
   to ready
```

# Exécution des transitions

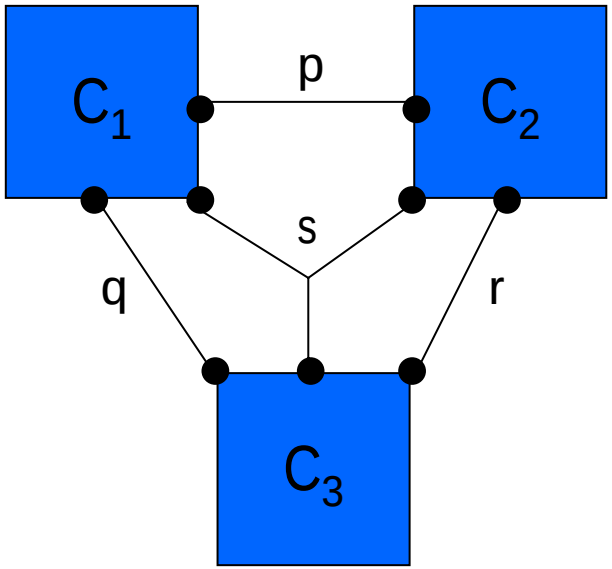
- Toute transition est atomique
- Chaque transition définit plusieurs chemins d'exécution
  - En fonction de la valeur des variables
  - En raison du non-déterminisme
- Contrainte (statique) : au plus un événement de communication sur chaque chemin d'exécution

# Composition

- Définition de composants complexes (**component**) paramétrés 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 :
  - La mise en parallèle d'instances des composants (hiérarchique)
  - Des priorités entre actions
  - Des contraintes temporelles sur les communications

# Mise en parallèle

- Les composants mis en parallèle sont précédés d'une "interface" listant des ports de communication
- Tous les composants (2 ou plus) dont les interfaces partagent un même port doivent se synchroniser
- Exemple :




$C_1$, $C_2$, $C_3$ : instances ou mises en parallèle de composants

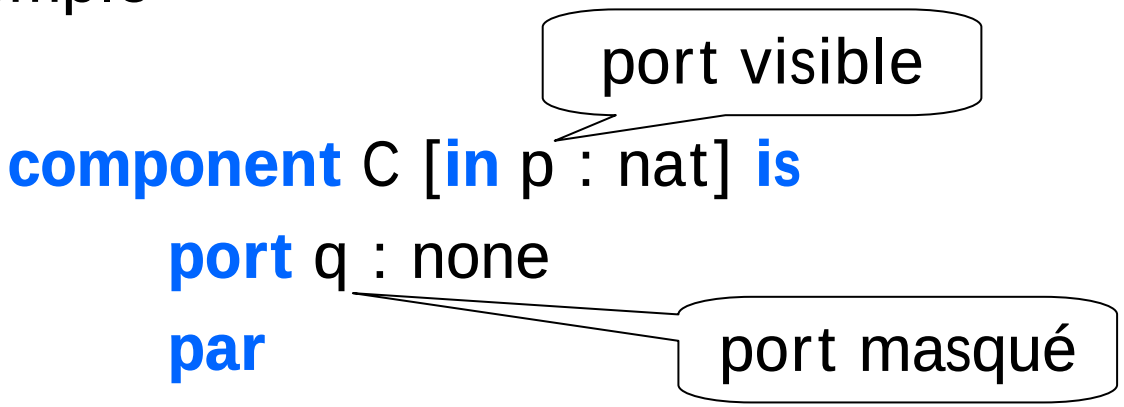
**par**

$p, q, s \rightarrow C_1$
$p, r, s \rightarrow C_2$
$q, r, s \rightarrow C_3$

**end**

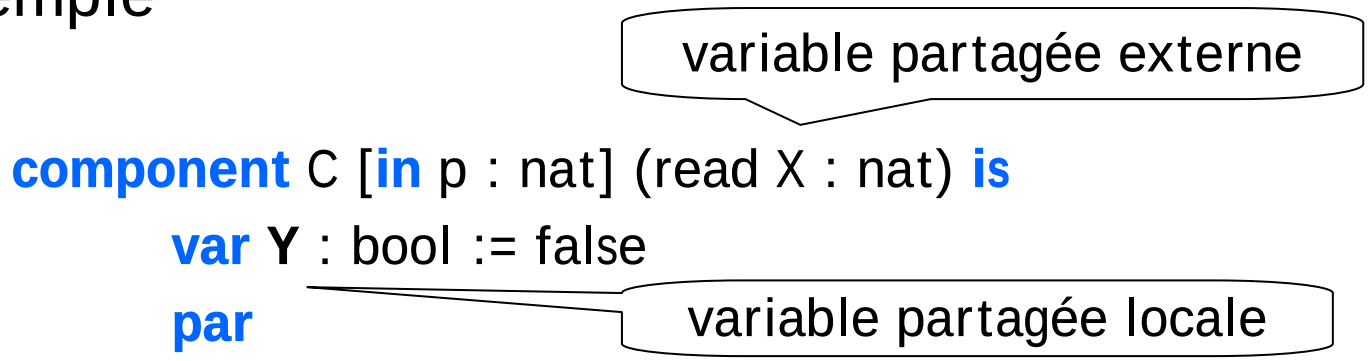
# Ports locaux

- Un port passé comme paramètre formel du composant est instanciable (renommage) et visible
- Un port peut être local au composant : il est *masqué*, ses communications sont *inobservables*
- Exemple



# Variables partagées locales

- Une variable partagée passée comme paramètre formel du composant est instanciable et externe au composant
- Une variable peut être locale au composant : elle ne peut être accédée que par les sous-composants
- Exemple



# Priorités

- Fiacre permet d'associer des priorités aux ports
- Si 2 événements sont possibles simultanément, celui qui a lieu sur le port le plus prioritaire inhibe l'autre
- Exemple

```
component C [in p : nat] is
        port q : none
        priority p > q
        par
                q → C1 [p, q]
                q → C2 [p, q]
        end
```

# Contraintes temporelles

- Fiacre permet d'associer un intervalle temporel (ouvert ou fermé) à un port local
- Exemple

```
component C is
        port q : none in ]2.7, 3.8]
        par
                q → C1 [q]
                q → C2 [q]
        end
```

  - Un temps strictement supérieur à 2.7 unités doit s'écouler à partir du moment où $C_1$ et $C_2$ sont prêts à communiquer sur q
  - La communication sur q doit avoir eu lieu au plus tard 3.8 unités de temps après que $C_1$ et $C_2$ sont prêts à communiquer sur q

# Bilan

- Travail (presque) achevé
  - Syntaxe et sémantique formelle du langage complet
- Travail en cours
  - Définition d'un meta-modèle Fiacre (C-S)
  - Transformation de SDL vers Fiacre (C-S)
- Travail futur
  - Compilation vers les outils (CADP, TINA)