Théorie de la Modélisation et de la Simulation

Fondements formels et opérationnels de VLE

Raphaël Duboz[1] – Frédérick Garcia[2]

[1]Centre de coopération International de Recherche Agronomique pour le Développement
[2]Institut National de la Recherche Agronomique

Plan

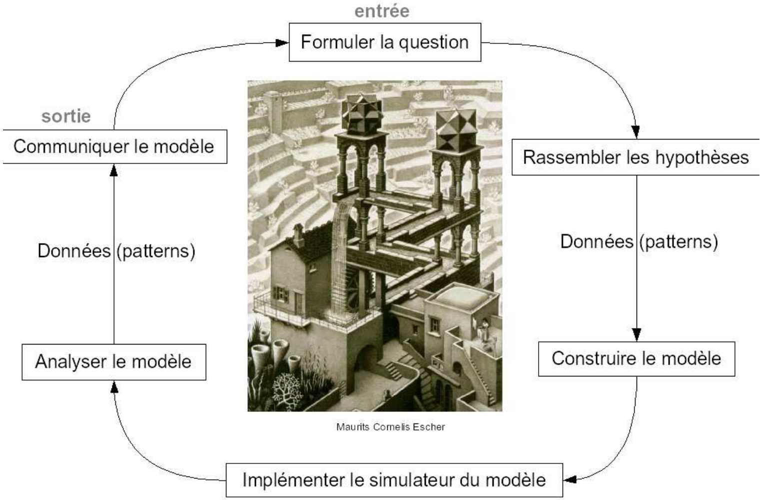
1. Modélisation et Simulation
   - La théorie de la modélisation et de la simulation de B.P. Zeigler
2. Formalismes de modélisation
   - Equations différentielles ordinaires
   - Equations différentielles partielles
   - Equations aux différences
   - Automates à états finis
   - Automates cellulaires
   - Réseaux de Pétri
   - Modèles à évènements discrets
   - Autres modèles
3. P-DEVS
4. Multi-formalisme et problématique du couplage
   - Modélisation multiple
   - Intégration opérationelle
   - Intégration formelle

Plan

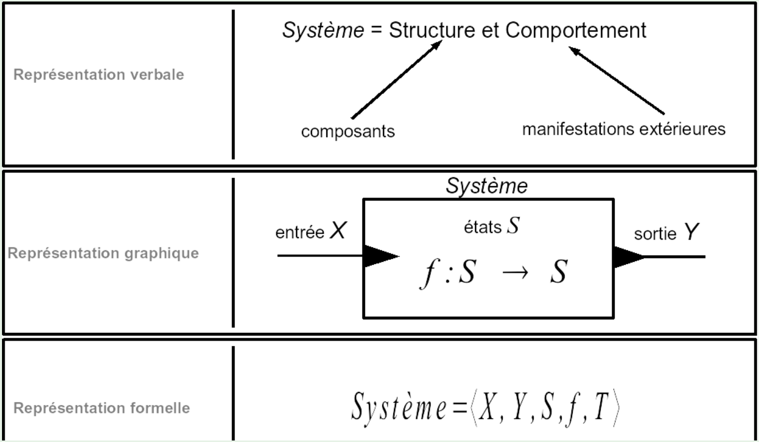
1. Modélisation et Simulation
   - La théorie de la modélisation et de la simulation de B.P. Zeigler
2. Formalismes de modélisation
   - Equations différentielles ordinaires
   - Equations différentielles partielles
   - Equations aux différences
   - Automates à états finis
   - Automates cellulaires
   - Réseaux de Pétri
   - Modèles à évènements discrets
   - Autres modèles
3. P-DEVS
4. Multi-formalisme et problématique du couplage
   - Modélisation multiple
   - Intégration opérationelle
   - Intégration formelle

Théorie M&S

Le cycle de la modélisation et de la simulation



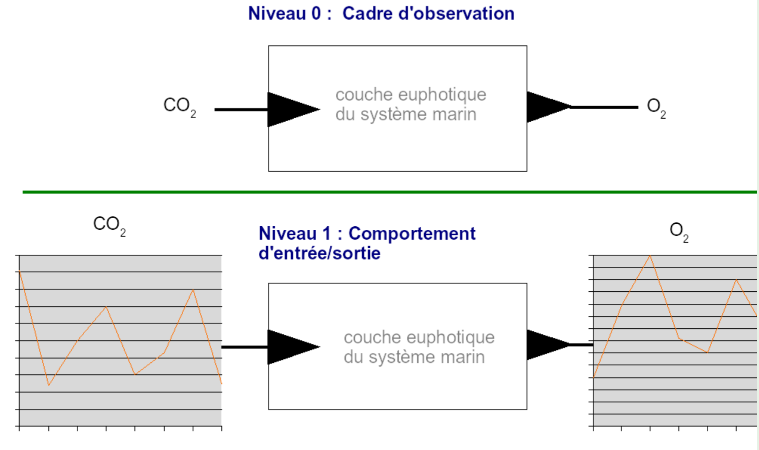
La notion de système



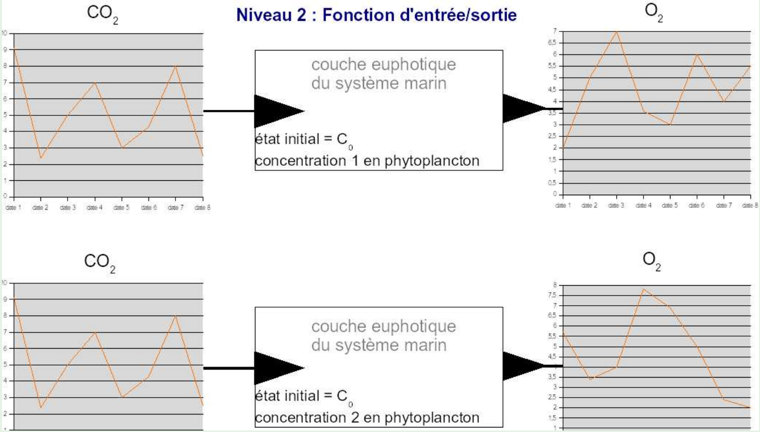
Hiérarchie de spécification

niveau	nom	que savons nous à ce niveau ?
0	cadre d'observation	quelles variables mesurer et comment les observer sur une base de temps
1	comportement d'entrée sortie	données indexées sur le temps. Pairs d'entrée – sortie
2	fonction d'entrée – sortie	connaissance de l'état initial (un entrée → une sortie)
3	transition d'états	comment les entrées affectent les états et comment les états affectent les sorties
4	composants couplés	comment les composants de niveaux inférieurs sont couplés

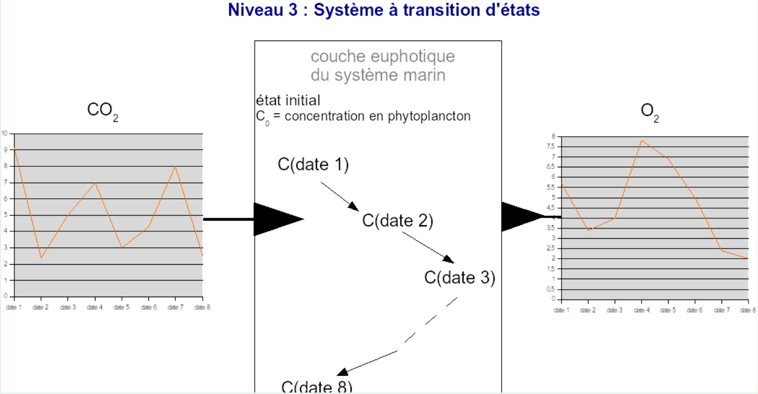
Hiérarchie de spécification



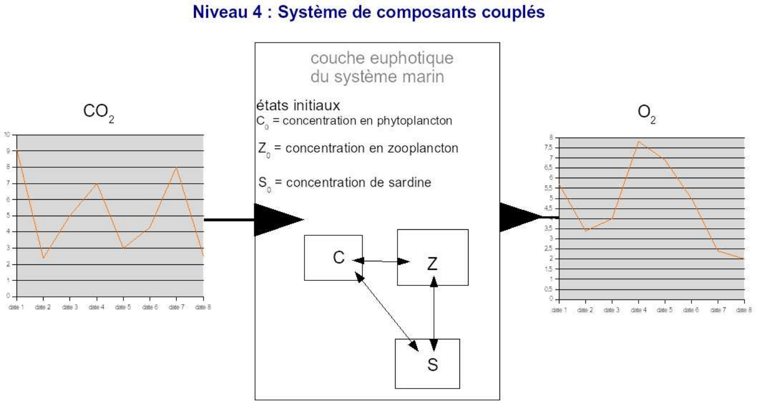
Hiérarchie de spécification



Hiérarchie de spécification

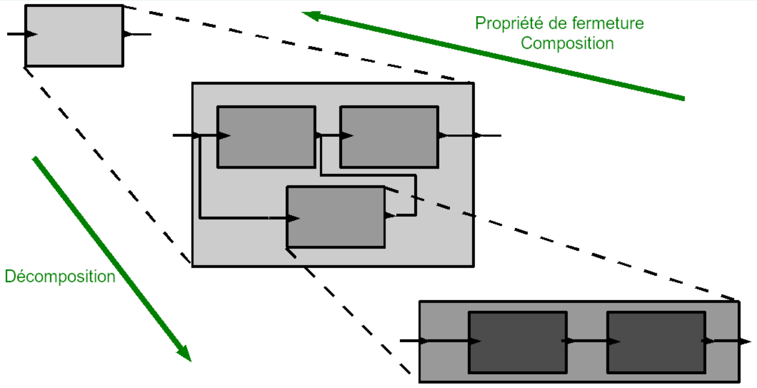


Hiérarchie de spécification



- Un système est décomposable en sous-système
- Un ensemble de système peut être vu comme un système

Composition - Décomposition



Niveaux de spécification – Niveaux d'organisation

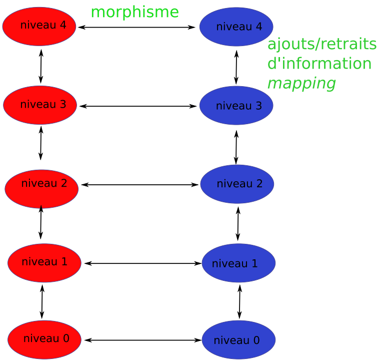
- Les niveaux de spécifications ne sont pas des niveaux d'organisation
- Ils reflètent la connaissance que nous avons du système
- Il existe néanmoins des relations
  - la décomposition mène souvent à une descente dans les niveaux d'organistion (échelles de temps et d'espace plus petites)
  - et réciproquement...

Morphisme

- Un morphisme met les éléments de deux systèmes en correspondance
- Deux systèmes sont isomorphiques au niveau 0 si nous pouvons palcer les entrées – le sorties – la base de temps en correspondance et qu'ils sont identiques

Théorie M&S

Hiérarchie de spécification



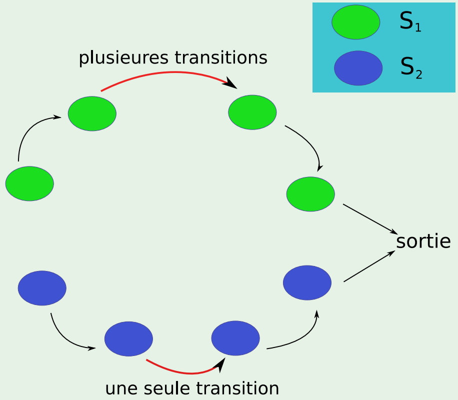
Relations entre niveaux

Un morphisme à un niveau implique un morphisme aux niveaux inférieurs

Relation de morphisme entre systèmes

niveau	nom	2 systèmes sont morphiques si :
0	cadre d'observation	mise en correspondance des entrées – sortie et de la base de temps
1	comportement d'entrée sortie	morphique au niveau 0. Les entrées sorties –sortie match une à une
2	fonction d'entrée – sortie	morphique au niveau 1 et même comportement des fonctions d'entrée sortie (un état init. donne la même sortie)
3	transition d'états	les systèmes sont homomorphiques
4	composants couplés	tous les composants sont morphiques aux niveaux inférieurs, le couplage est morphique

La relation d'homomorphisme existe au niveau 3 (transions d'états)



Les deux systèmes ont des états communs et produisent les mêmes sorties s'ils sont dans le même état au départ

Relation de morphisme entre systèmes

niveau	nom	Objets formels
0	cadre d'observation	$< T, X, Y >$
1	comportement d'entrée sortie	$< T, X, \Omega, Y, R >$
2	fonction d'entrée – sortie	$< T, X, \Omega, Y, F >$
3	système à transition d'états	$< T, X, \Omega, Y, Q, \Delta, \Lambda >$
4	système couplés	$N = < T, X_N, Y_N, D, \{M_D \mid d \in D\}, \{I_d \mid d \in D \cup N\}, \{Z_d \mid d \in D \cup N\} >$

Exemple de morphisme au niveau du cadre d'observation

Considerons :
$S_1 =< T_1, X_1, Y_1 >$ et $S_2 < T_2, X_2, Y_2 >$
et les fonctions qui relient les entrées et les sorties des deux systèmes:

$$g : (X_2, T_2) \rightarrow (X_1, T_1)$$

$$k : (Y_1, T_1) \rightarrow (Y_2, T_2)$$

si g et k sont des fonctions identitées alors S_1 et S_2 sont isomorphes[a]

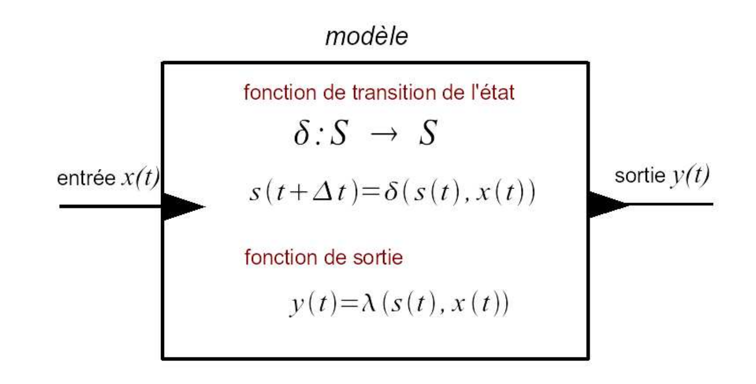
[a] À ce niveau, le morphisme rejoint la question de la validation. Un système réel ne peut pas être isomorphe d'un système artificiel conçu pour le représenter. Dans ce cas, La fonction k est toujours une statistique.

- Il y a une distinction claire entre modèle et simulateur
- Les deux peuvent s'écrire comme des systèmes
- Il est possible d'établir des relation d'équivalence entre eux (morphismes)

La notion d'erreur introduite par le simulateur n'est pas toujours simple à formuler.
sauf dans le cas d'un intégrateur où elle est peut souvent l'être (analyse numérique)

- La séquence $s_0, s_1, ...s_n$ est appelée trajectoire d'état
- La séquence $x_0, x_1, ...x_n$ est appelée trajectoire d'entrée
- La séquence $y_0, y_1, ...y_n$ est appelée trajectoire de sortie

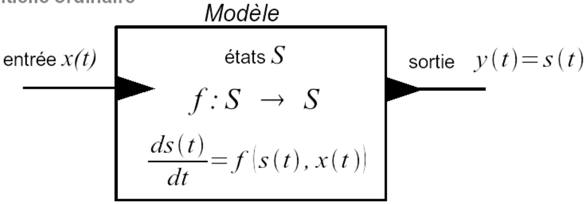
modèle temps discret



Simulateur

$t = t_0$
$s = s_0$
tant que $t <= t_f$ faire
  $y(t) = \lambda(s(t), x(t))$
  $s(t + \Delta t) = \delta(s(t), x(t), \Delta t)$
fin tant que

Équation différentielle ordinaire



Système couplé d'équations différentielles ordinaires

$$\frac{ds_1(t)}{dt} = f_1(s_1(t), s_2(t), \ldots, s_n(t), \; x_1(t), x_2(t), \ldots, x_n(t))$$

$$\frac{ds_2(t)}{dt} = f_2(s_1(t), s_2(t), \ldots, s_n(t), \; x_1(t), x_2(t), \ldots, x_n(t))$$

$$\ldots$$

$$\frac{ds_n(t)}{dt} = f_n(s_1(t), s_2(t), \ldots, s_n(t), \; x_1(t), x_2(t), \ldots, x_n(t))$$

- Si possible, l'équation différentielle est résolue de façon analytique
  - Impossible dans la grande majorité des modèles
  - Sinon, on intègre l'équation par résolution numérique

Simulateur

l'idée sous jacente pour un intégrateur parfait :

$$\frac{ds(t)}{dt} = \lim_{\Delta t \to 0} \frac{s(t + \Delta t) - s(t)}{\Delta t}$$

Donc pour un interval de temps très petit :

$$s(t + \Delta t) \approx \Delta t \frac{ds(t)}{\Delta t} + s(t)$$

$$s(t + \Delta t) \approx \Delta t[f(x(t), s(t))] + s(t)$$

Plan

1. Modélisation et Simulation
   - La théorie de la modélisation et de la simulation de B.P. Zeigler
2. Formalismes de modélisation
   - Equations différentielles ordinaires
   - Equations différentielles partielles
   - Equations aux différences
   - Automates à états finis
   - Automates cellulaires
   - Réseaux de Pétri
   - Modèles à évènements discrets
   - Autres modèles
3. P-DEVS
4. Multi-formalisme et problématique du couplage
   - Modélisation multiple
   - Intégration opérationelle
   - Intégration formelle

Etat continu, Espace discret, Temps continu

E.D.O d'ordre n

$$F(x, x', \ldots, x^{(n)}, t) = 0$$

avec F continue.

Exemple : le pendule simple

$$mL^2\ddot{\theta} + b\dot{\theta} + mgLsin(\theta) = c_e(t)$$

Résoudre une E.D.O : trouver une fonction $x(t)$ vérifiant $F(x(t), x'(t), \ldots, x^{(n)}(t), t) = 0$ sur le domaine de t.

Une E.D.O est sous forme résolue si

$$x^{(n)} = f(x, x', \ldots, x^{(n-1)}, t) \quad (1)$$

avec f continue.
Sous certaines hypothèses de régularité, le théorème de Cauchy-Lipschitz établit que pour (1)

$$x(t_0) = x_0,\ x'(t_0) = x_1, \ldots, x^{(n-1)}(t_0) = x_{n-1}$$

il existe une (unique) solution satisfaisant (1) autour de t_0.

E.D.O du 1er ordre

$$F(x, x', t) = 0$$

F peut être vectoriel, systèmes d'équations différentielles couplées

$$x^{(n)} = f(x, x', \dots, x^{(n-1)}, t) \quad (1)$$

équivalent à

$$y' = g(y, t), \text{ avec } y = \begin{bmatrix} x \\ \dot{x} \\ \dots \\ x^{(n-1)} \end{bmatrix}.$$

Exemple : le pendule simple

$$\begin{aligned} \dot{\theta} &= \omega \\ \dot{\omega} &= -\frac{g}{L} sin(\theta) - \frac{b}{mL^2}\omega + \frac{1}{mL^2} c_e(t) \end{aligned}$$

Pour une E.D.O du 1er ordre avec C.I.

$$\dot{x} = f(x(t), t), \quad x(a) = x_0$$

- développement de Taylor :

$$x(t+h) = x(t) + h\dot{x}(t) + \frac{h^2}{2}\ddot{x}(t) + \frac{h^3}{3!}x^{(3)}(t) + \cdots$$

  Au premier ordre : méthode d'Euler, avec pas fixe $h = \frac{(t_f - t_0)}{N}$.
- Runge-Kutta : évaluation itérative de $f()$ en plusieurs points sur l'intervalle $[t_n, t_{n+1}]$.

Exemple : Runge-Kutta ordre 2

$$x_{n+1} = x_n + \frac{h_n}{4}\left[f(x_n, t_n) + 3f\left(x_n + \frac{2}{3}h_n f(x_n, t_n), t_n + \frac{2}{3}h_n\right)\right]$$

- Pas adaptatifs

Changement discontinu de l'état, changement de la dynamique au passage d'une frontière

- détection du changement de signe d'une fonction $h(x)$
- recherche itérative du zéro de $h()$.

Il est difficile de coupler dynamique continue et dynamique stochastique.

Exemple : le mouvement brownien

Etat continu, Espace continu, Temps continu

On parle d'EDP dans le cas de fonctions qui dépendent de plusieurs variables, une EDP exprimant un lien entre les dérivées partielles d'une fonction.

$$F(\dots, \frac{\partial^{i+j} u(x,t)}{\partial x^i \partial t^j}, \dots) = 0$$

C.I. fonctionnelles.

Exemple : l'équation de la chaleur

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2}$$

avec $u(x,0) = h(x)$ condition de temperature initiale, et par exemple $u(0,t) = u(L,t) = 0$.

On parle plutôt de résolution.

Différentes méthodes qui passent par la discrétisation en x et et en t de l'EDP (volumes finis, éléments finis, etc.).

Etat continu, Espace discret, Temps discret

$$x_n = f(x_{n-1}, x_{n-2}, \ldots, x_{n-m}, n)$$

avec C.I. $x_0, x_{-1}, \ldots, x_{-m+1} \in \boldsymbol{R}$.

Se ramène à une E.D. d'ordre 1

$$y_n = g(y_{n-1}, n) \text{ avec } y_n = \begin{bmatrix} x_n \\ \cdots \\ x_{n-m+1} \end{bmatrix}.$$

E.D. d'ordre 1 monotone : $y_n = f(y_{n-1})$.

Cas particulier de l'équation précédente

$$x_n^i = f_i(x_{n-1}^1, \ldots, x_{n-1}^{i-1}, x_{n-1}^i, x_{n-1}^{i+1}, \ldots, x_{n-1}^p)$$

avec x^i variables d'état spatialisées.

Toujours exactement simulable si x_n reste dans le domaine de $f()$.

Phénomènes de cycles limites, d'oscillation, de chaos.

Rendre stochastique $x_n = f(x_{n-1}, x_{n-2}, \ldots, x_{n-m}, n)$
Modélisation d'un processus Markovien d'ordre m

$$dx_n = F(x_{n-1}, x_{n-2}, \ldots, x_{n-m}, n)$$

avec F fonction de distribution sur x_n.

Une approche très simple consiste à ajouter à $f()$ une variable aléatoire :

$$x_n = f(x_{n-1}, x_{n-2}, \ldots, x_{n-m}, n, \theta_n)$$

avec θ_n processus aléatoire.

Egalement possibilité de tirer aléatoirement les C.I. $x_0, x_{-1}, \ldots, x_{-m+1} \in \boldsymbol{R}$.

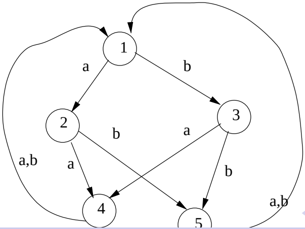
Etat discret, Espace discret, Temps discret

$$A = \langle X, S, s_i, Q, \Delta \rangle$$

avec S l'ensemble des états s du système, s_i l'état initial, Q les état terminaux, X l'ensemble des symboles x possibles en entrée et Δ la fonction de transition :

$$s(n) = \Delta(x(n-1), s(n-1))$$

Exemple avec $S = \{1, 2, 3, 4, 5\}$, $A = \{a, b\}$



La simulation est simple, de type équations aux différence pour une séquence de symbole d'entrée donnée.

Comme pour les équations aux différence, les AEF se prêtent bien à l'analyse théorique de leurs propriétés.

Rendre stochastique la fonction de transition $s(n) = \Delta(x(n-1), s(n-1))$.

On représente Δ par une chaîne de Markov sur $s_n \in S$ contrôlée par $x_n \in X$, selon

$$P(s(n) \mid s(n-1), x(n-1))$$

Etat discret, Espace discret, Temps discret

Automate à état fini particulier avec représentation spatialisée de l'état.

$$RdP = \langle G, \S, \Delta \rangle$$

avec

- G une grille infinie régulière sur $\boldsymbol{R}^d$, $d = 1, 2, 3$ qui porte les *cellules*,
- S l'espace d'état en chaque cellule,
- Δ la fonction de transition locale de chaque cellule

$$s_i(n) = \Delta_i(s_{j_1}(n-1), \ldots, s_{j_k}(n-1)), \Delta \quad j_k \text{ voisins de } i$$

La fonction de transition Δ peut être spatialement *homogène* ou non, dépendate du temps ou non, synchrone ou asynchrone, déterministe ou stochastique.

Méthode classique d'analyse des automates cellulaires.

Nécessité de fixer les condtion aux limites car simulation d'une grille finie.

Simulation à évènements discrets pour les automates temporels asynchrones.

Etat discret, Espace discret, Temps discret

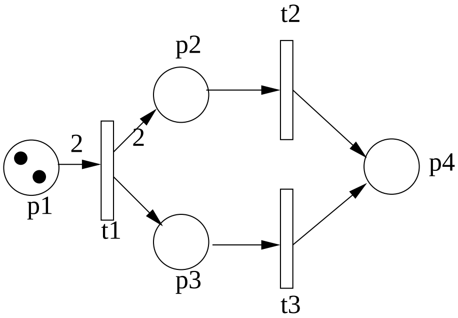
Automate à état fini particulier, avec représentation factorisée de l'état et de la fonction de transition.

$$RdP = \langle P, T, I, O, M_0$$

avec

- P l'ensemble des *places*,
- T l'ensemble des *transitions*,
- I et O l'ensembles des arcs valués reliant places et transitions,
- M_0 le *marquage* initial.

La dynamique d'un RdP est modélisée par le *tirage* des transitions dont les places en amont sont remplies, amenant une modification du marquage M.



t2
p2
2
2
p1
t1
p4
p3
t3

Les réseaux de Pétri bien adaptés pour modéliser des systèmes dynamiques présentant des propriétés telles que :

- execution séquentielle
- conflit
- concurrence
- synchronisation
- ressources contraintes

Utilisation principale des RdP
Vérification de propriétés formelles vérifiées par le réseau :

- atteignabilité
- réseau sain
- réseau vivant

Autre approche pour étudier un RdP.

Simulation de type AEF + choix non-déterministes.

Enrichissement du pouvoir d'expression des RdP au détriment de l'analyse formelle

- RdP colorés jetons et places distingués
- RdP temporels Assignation de valeurs temporels aux transitions, places, arcs
  Etat discret, Espace discret, Temps continu
  - deterministic Times Petri nets (DTPN)
  - stochastic timed Petri nets (STPN)

  Simulation à évènements discrets.

Etat discret, Espace discret, Temps continu

Changement de l'état du système (*event*) à des dates discrètes quelconques sur R^+.

Un modèle à évènements discrets pour pouvoir :

- programmer / annuler des évènements
- décrire le changement d'état associé à un évènement

Par rapport aux AEF, l'accent est mis sur la modélisation de la dynamique des évènements (messages X).

La simulation d'un modèle à évènements discrets consiste à gérer une liste courante d'évènements programmés, et, successivement :

- faire avancer le temps jusqu'au au prochain évènement programmé
- exécuter l'évènement : modifier l'état et la liste d'évènements

Generalized Semi-Markov Processes : formalisme mathématique de description de modèles à évènements discrets stochastiques

Modélisation à base d'horloge associées aux évènements

- très utilisé pour prouver des propriétés de modèles
- adaptable à la problématique de décision optimale

Autres modèles

Il existe au delà des principaux formalismes présentés ici de nombreux autres modèles qui s'y rattachent plus ou moins

- *System Dynamics* de Forrester : modèles de type systèmes d'EDO modélisant des stocks et flux Cf. Vensim, ModelMaker, etc.
- *Dynamic Systems* : Issu de l'Automatique. Modèles de type équation différentielles algébriques modélisant intégration, dérivation, filtrage, retard, etc. sur des variables physiques continues. Cf. Matlab Simulink, Scilab, etc.
- Modèles d'agents, systèmes multi-agents. Généralise les automates cellulaires, avec des cellules qui seraient de type modèles à évènements discrets ... . Cf. Swarm, etc.
- Modèles décisionnels : plans d'actions, ordonnancement, plans réactifs, etc.

Plan

Un formalisme systémique de modélisation :

- événements discrets
- ensembles, états, fonctions de transition d'états
- approche modulaire et hiérarchique

DEVS à été initié par B. P. Zeigler en 1976 :

- +50 simulateurs DEVS (libres/payants, spécialisés etc.)
- Monde : États-Unis (Univ. Arizona), Canada (Univ. Mc Gill), Portugal (Univ. Calombra), Corée du Sud (Univ. Hang Kong), Allemagne (Univ. Rostock), . . .
- France : Calais (Univ ulco), Clermont-Ferrand (Isima), en Corse (Univ. de Corse), Marseille (Univ Aix 3), Montpellier (Cirad), . . .

Un formalisme systémique de modélisation :

- événements discrets
- ensembles, états, fonctions de transition d'états
- approche modulaire et hiérarchique

DEVS à été initié par B. P. Zeigler en 1976 :

- +50 simulateurs DEVS (libres/payants, spécialisés etc.)
- Monde : États-Unis (Univ. Arizona), Canada (Univ. Mc Gill), Portugal (Univ. Calombra), Corée du Sud (Univ. Hang Kong), Allemagne (Univ. Rostock), . . .
- France : Calais (Univ ulco), Clermont-Ferrand (Isima), en Corse (Univ. de Corse), Marseille (Univ Aix 3), Montpellier (Cirad), . . .

DEVS [Zeigler, 1976] :

- un formalisme de haut niveau (généralité)

  $\rightarrow$ DEVS « encapsule » les formalismes

- propose un cadre formel pour le couplage de modèles

  $\rightarrow$ Propriété de fermeture sous couplage

- utilise le paradigme des événements discrets

  $\rightarrow$ Les événements pilotent la simulation

- intègre un cadre opérationnel : les simulateurs abstraits

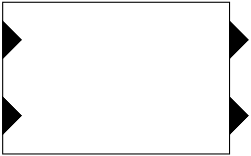
  $\rightarrow$ Un ensemble d'algorithmes

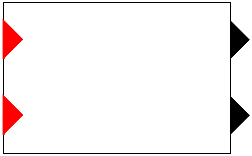
- Définition du modèle atomique :

* Nous considérons ici la varsion parallèle de DEVS, P-DEVS. Quand nous dirons DEVS par la suite, il s'agira en fait de P-DEVS

$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda \rangle$$

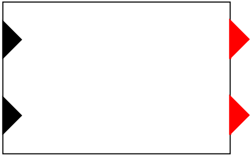
  - Une structure mathématique composée de variables et de fonctions
  - Une représentation graphique :





$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda \rangle$$

X : l'ensemble des ports d'entrée et des valeurs attachées.



$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda \rangle$$

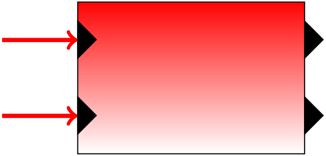
Y : l'ensemble des ports de sortie et des valeurs attachées.

$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda \rangle$$

S : l'ensemble des états du système.
À un instant t, $Q = \{(s, e) \mid s \in S, 0 < e < ta(s)\}$ est l'état total du système où e est le temps passé dans l'état.

DEVS, Discrete Event System Specification

Description du modèle atomique

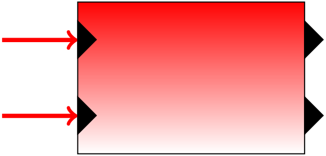


$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda \rangle$$

δ_{ext} : fonction de transition externe : $\delta_{ext} : Q \times X^b \rightarrow S$
représente les réponses du système aux événements externes où X^b est un ensemble de bag dans X

$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda \rangle$$

δ_{int} : fonction de transition interne : $\delta_{int} : S \rightarrow S$
représente les évolutions autonomes.

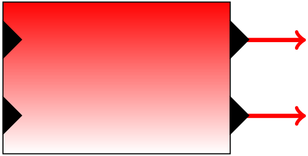


$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda \rangle$$

δ_{con} : fonction de conflict : δ_{con} : $Q \times X^b \rightarrow S$
représente les réponses du système si des événements externes arrivent et que $e = ta(s)$
$\delta_{con}(s, x) = \delta_{ext}(\delta_{int}(s), 0, x)$ ou $\delta_{int}(\delta_{ext}(s, ta(s), x))$

$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, ta, \lambda \rangle$$

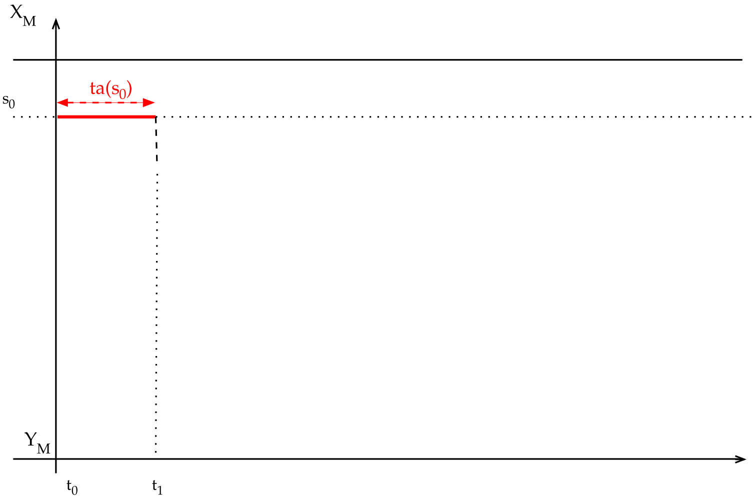
$ta(s)$: le temps pendant lequel le modèle reste dans l'état S (hors événements externes)



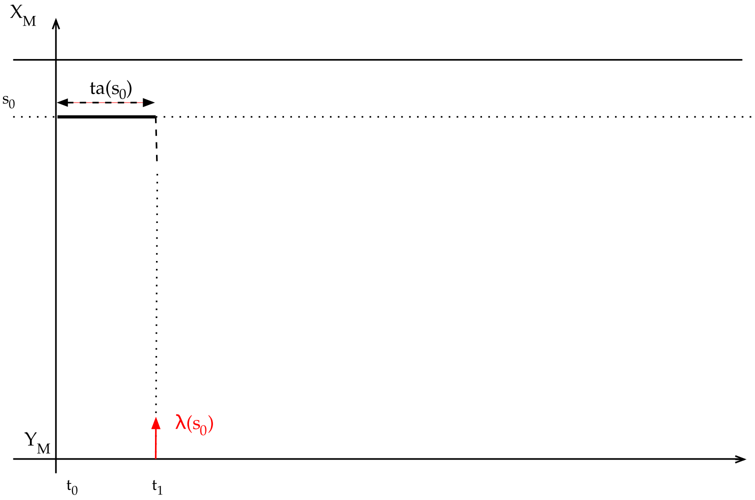
$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, ta, \delta_{con}, \lambda \rangle$$

λ : la fonction de sortie : $\lambda : S \rightarrow Y^b$ représente les influences externes.

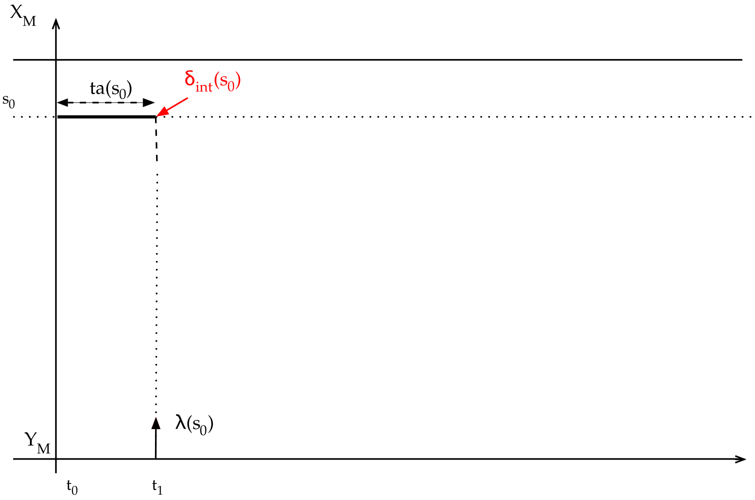
X_M
Y_M
t_0



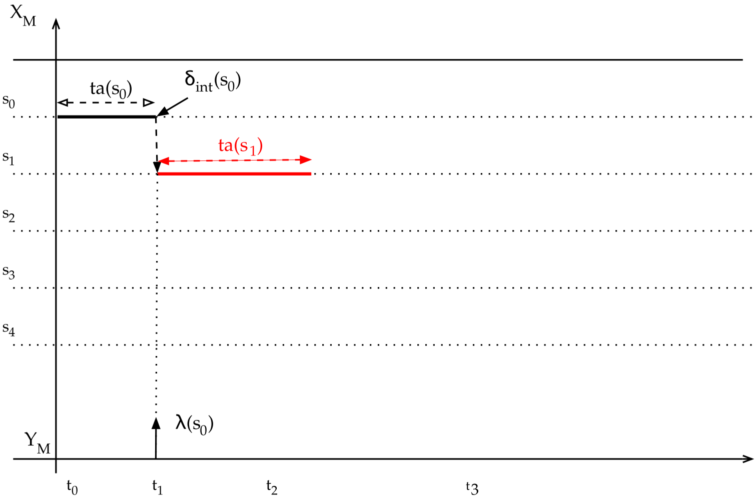
X_M
$ta(s_0)$
s_0
Y_M
t_0
t_1



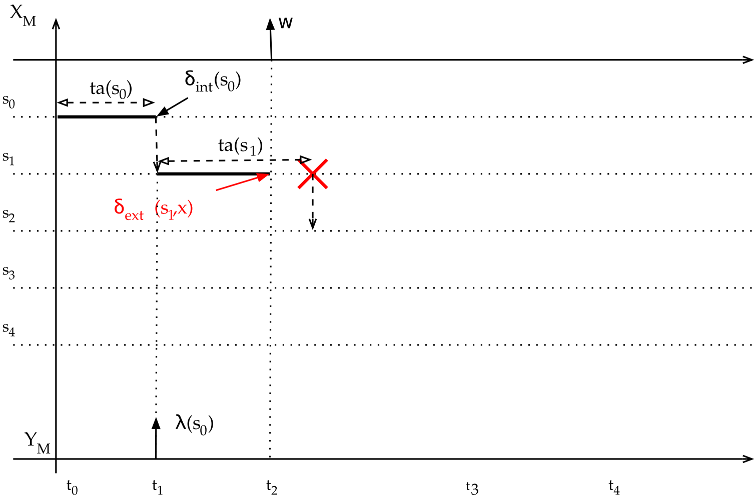
X_M
$ta(s_0)$
s_0
$\lambda(s_0)$
Y_M
t_0
t_1



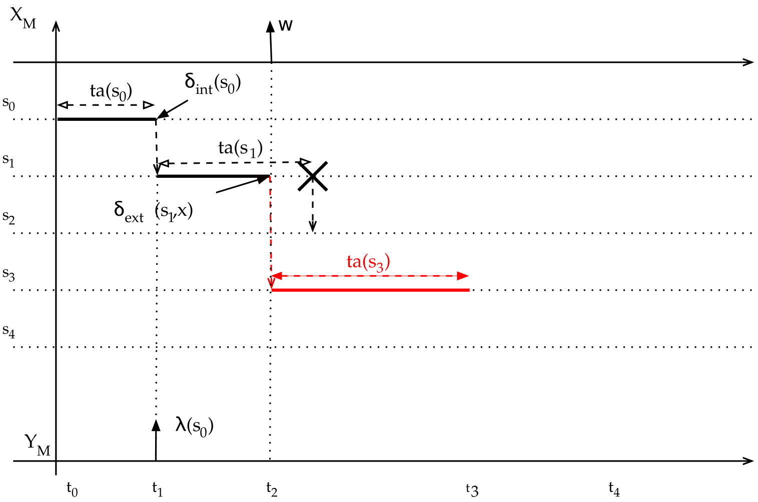
X_M
s_0
$ta(s_0)$
$\delta_{int}(s_0)$
$\lambda(s_0)$
Y_M
t_0
t_1



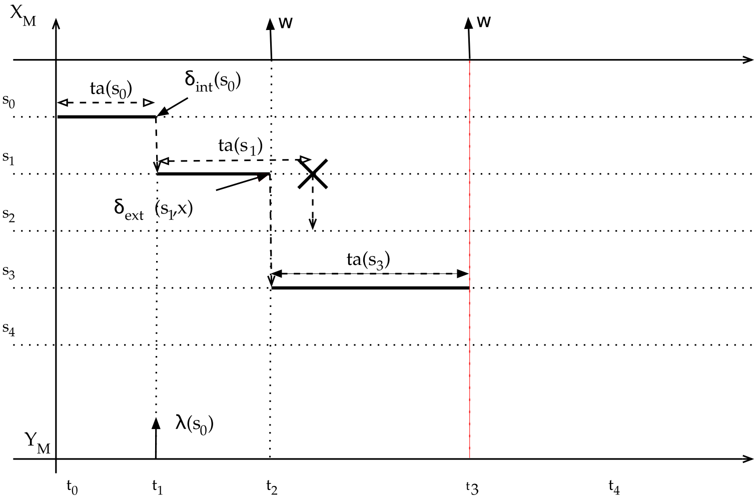
X_M
s_0
s_1
s_2
s_3
s_4
$ta(s_0)$
$\delta_{int}(s_0)$
$ta(s_1)$
$\lambda(s_0)$
Y_M
t_0
t_1
t_2
t_3



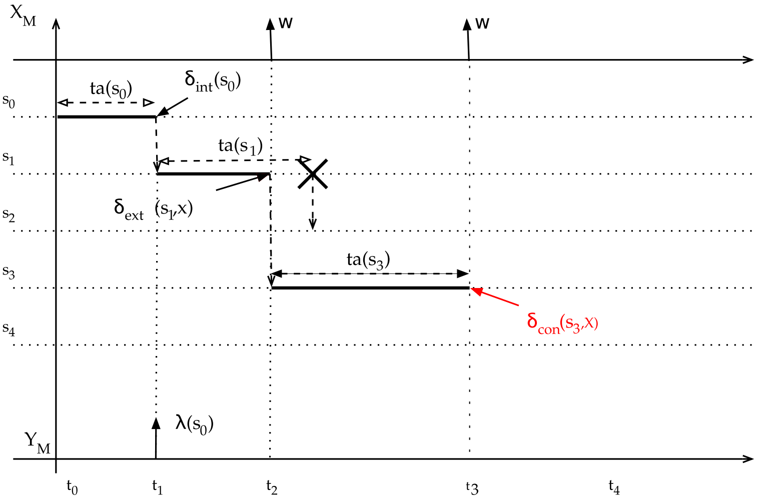
X_M
w
$\delta_{int}(s_0)$
$ta(s_0)$
s_0
$ta(s_1)$
s_1
δ_{ext} (s_1,x)
s_2
s_3
s_4
$\lambda(s_0)$
Y_M
t_0
t_1
t_2
t_3
t_4



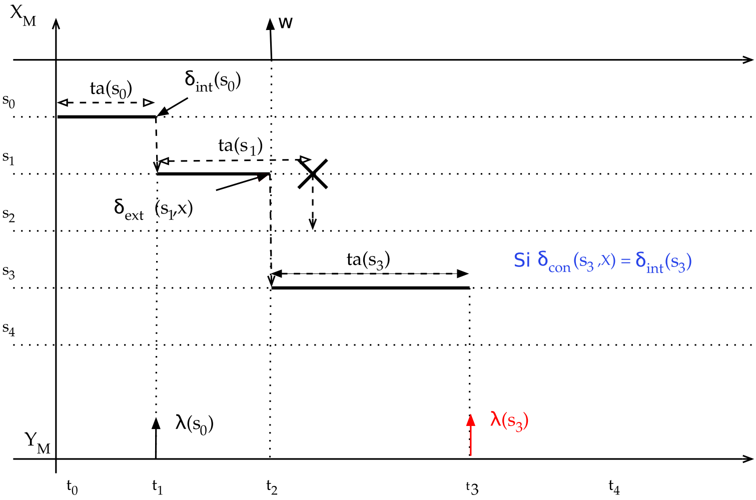
X_M
w
$ta(s_0)$
$\delta_{int}(s_0)$
s_0
$ta(s_1)$
s_1
δ_{ext} (s_1,x)
s_2
$ta(s_3)$
s_3
s_4
$\lambda(s_0)$
Y_M
t_0
t_1
t_2
t_3
t_4



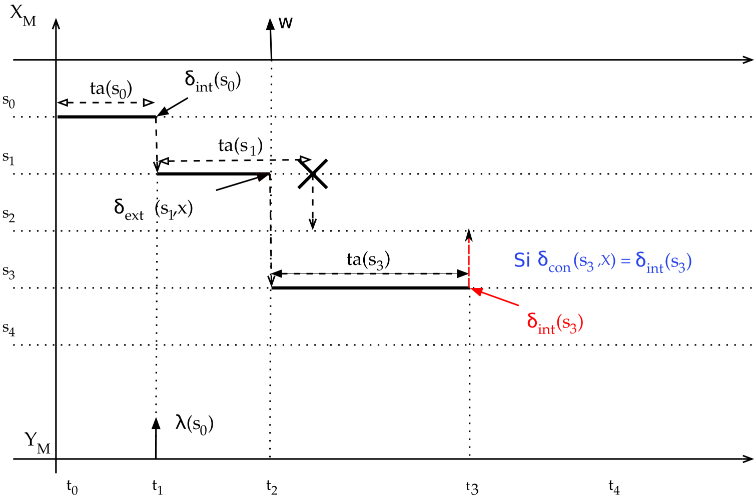
X_M
w
w
$ta(s_0)$
$\delta_{int}(s_0)$
s_0
$ta(s_1)$
s_1
s_2
δ_{ext} (s_1,x)
$ta(s_3)$
s_3
s_4
$\lambda(s_0)$
Y_M
t_0
t_1
t_2
t_3
t_4

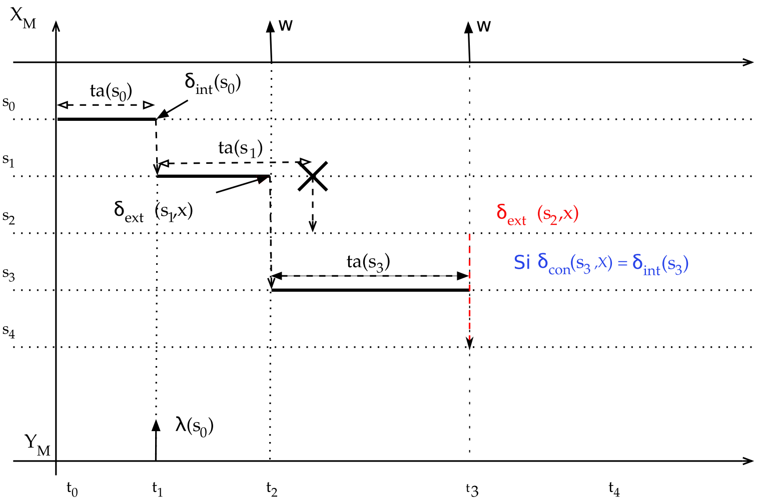


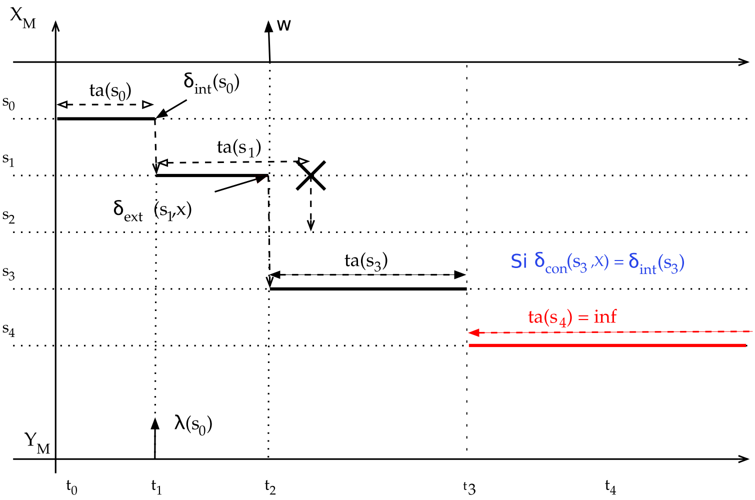
X_M
w
w
ta(s_0)
δ_int(s_0)
s_0
ta(s_1)
s_1
s_2
δ_ext (s_1,x)
ta(s_3)
s_3
δ_con(s_3,x)
s_4
λ(s_0)
Y_M
t_0
t_1
t_2
t_3
t_4



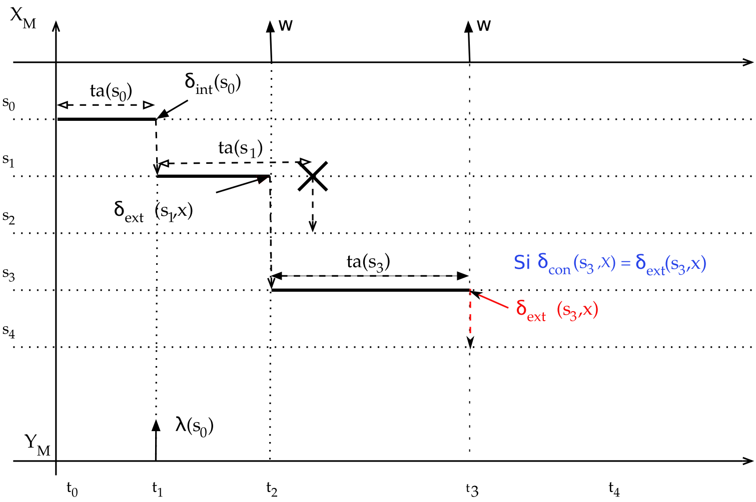
X_M
w
s_0
$ta(s_0)$
$\delta_{int}(s_0)$
s_1
$ta(s_1)$
s_2
$\delta_{ext}\ (s_1,x)$
s_3
$ta(s_3)$
Si $\delta_{con}(s_3, X) = \delta_{int}(s_3)$
s_4
$\lambda(s_0)$
$\lambda(s_3)$
Y_M
t_0
t_1
t_2
t_3
t_4


X_M
w
s_0
s_1
s_2
s_3
s_4
$ta(s_0)$
$\delta_{int}(s_0)$
$ta(s_1)$
$\delta_{ext}\ (s_1,x)$
$ta(s_3)$
Si $\delta_{con}(s_3,X) = \delta_{int}(s_3)$
$\delta_{int}(s_3)$
$\lambda(s_0)$
Y_M
t_0
t_1
t_2
t_3
t_4


X_M
w
w
s_0
$ta(s_0)$
$\delta_{int}(s_0)$
s_1
$ta(s_1)$
s_2
δ_{ext} (s_1,x)
δ_{ext} (s_2,x)
s_3
$ta(s_3)$
Si $\delta_{con}(s_3,x) = \delta_{int}(s_3)$
s_4
$\lambda(s_0)$
Y_M
t_0
t_1
t_2
t_3
t_4



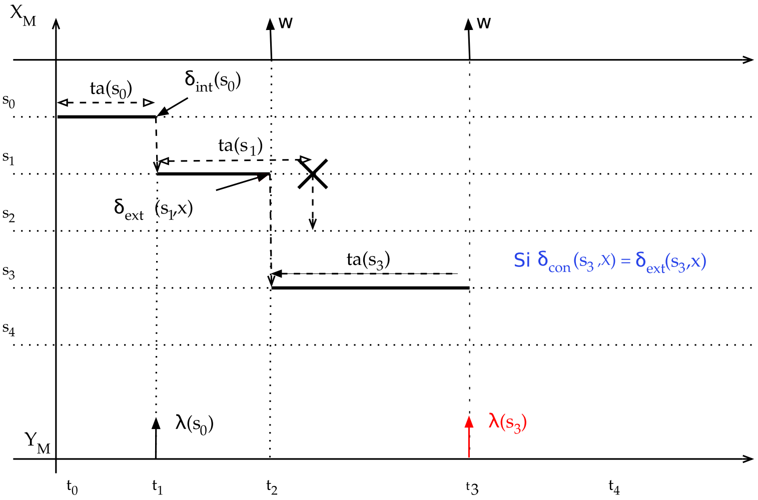
X_M
w
$ta(s_0)$
$\delta_{int}(s_0)$
s_0
s_1
$ta(s_1)$
s_2
$\delta_{ext}\ (s_1,x)$
s_3
$ta(s_3)$
Si $\delta_{con}(s_3, x) = \delta_{int}(s_3)$
s_4
$ta(s_4) = inf$
$\lambda(s_0)$
Y_M
t_0
t_1
t_2
t_3
t_4



X_M
w
w
s_0
s_1
s_2
s_3
s_4
$ta(s_0)$
$\delta_{int}(s_0)$
$ta(s_1)$
$\delta_{ext}\ (s_1,x)$
$ta(s_3)$
Si $\delta_{con}(s_3,X) = \delta_{ext}(s_3,x)$
$\delta_{ext}\ (s_3,x)$
$\lambda(s_0)$
Y_M
t_0
t_1
t_2
t_3
t_4

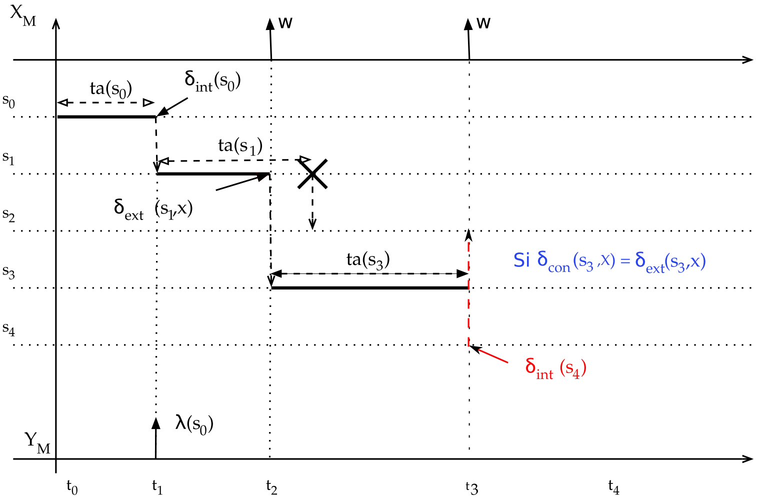
P-DEVS

Dynamique des états



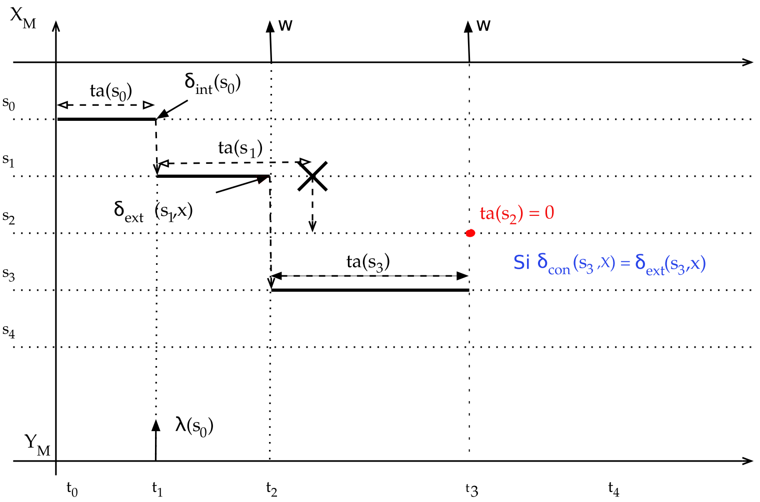
P-DEVS

Dynamique des états



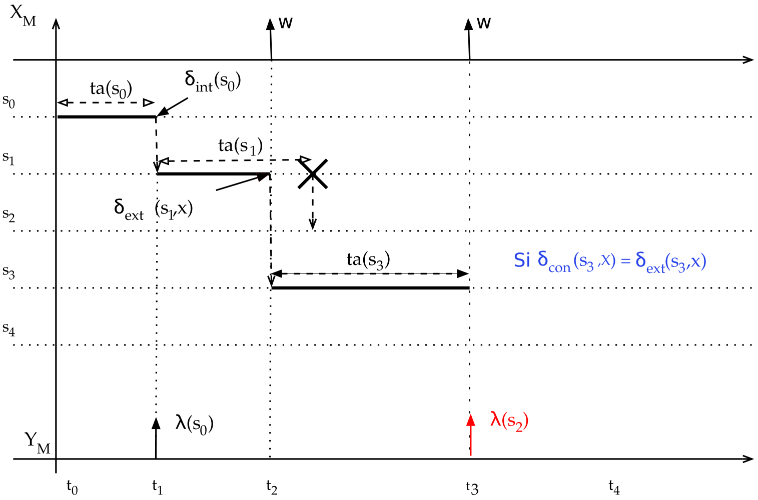
P-DEVS

Dynamique des états



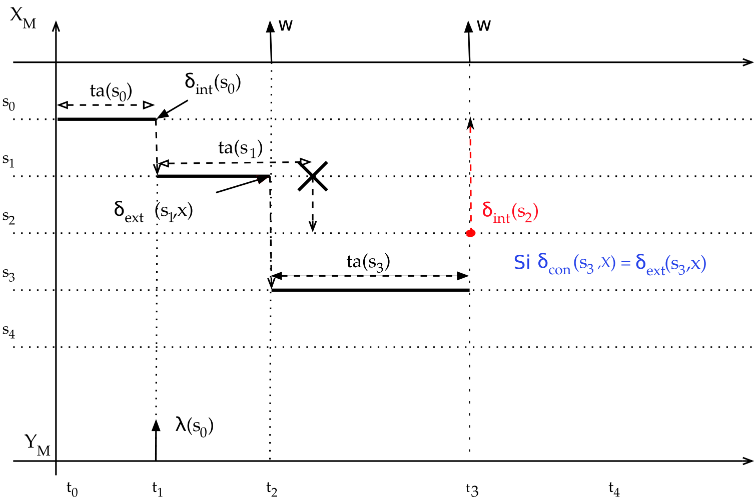
P-DEVS

Dynamique des états



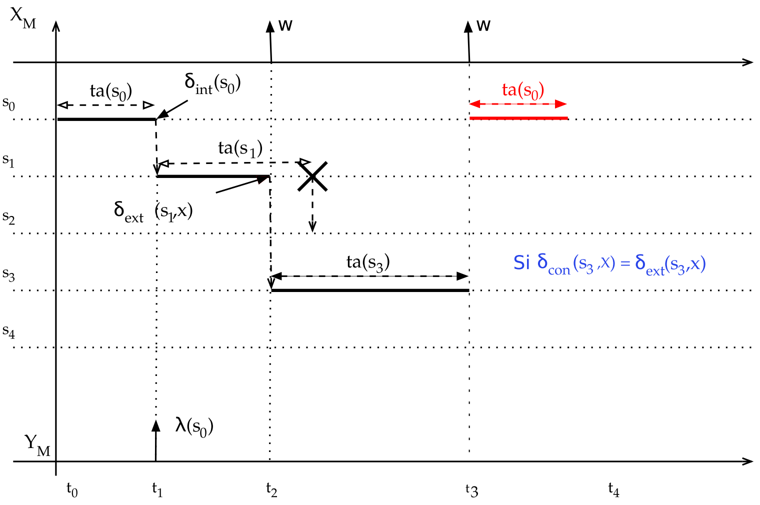
P-DEVS

Dynamique des états



P-DEVS

Dynamique des états



Variables

```
parent // coordinateur parent
tl     // durée depuis le dernier évènement
tn     // durée jusqu'au prochain évènement
DEVS   // modèle DEVS associé avec l'état total (s,e)
y      // bag de sortie
e      // le temps passé dans l'état courant s
```

Fonction

```
quand reçoit i-message (i,t) à la date t alors: // fonction init.
  tl <- t - e
  tn <- tl + ta(s)
```

Variables

```
parent // coordinateur parent
tl     // durée depuis le dernier évènement
tn     // durée jusqu'au prochain évènement
DEVS   // modèle DEVS associé avec l'état total (s,e)
y      // bag de sortie
e      // le temps passé dans l'état courant s
```

Fonction

```
quand reçoit *-message (*,t) à la date t alors: // transition interne
  si t = tn alors:
    y <- lambda(s)
    envoie y-message (y, t) au coordinateur parent
```

Variables

```
parent // coordinateur parent
tl     // durée depuis le dernier évènement
tn     // durée jusqu'au prochain évènement
DEVS   // modèle DEVS associé avec l'état total (s,e)
y      // bag de sortie
e      // le temps passé dans l'état courant s
```

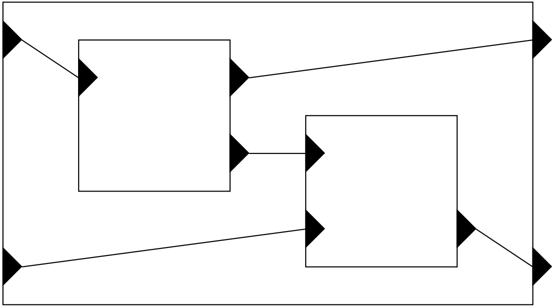
Fonction

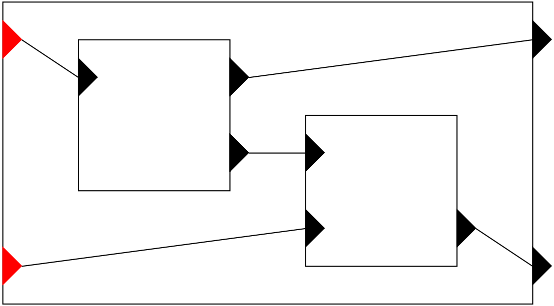
```
quand recçoit x-message (x, t) alors:
  si (x est vide  et  t = tn) alors s = delta_int(s) // trans. interne
  sinon si (t = tn) alors s = delta_con(s, x)      // fonc. conflict
        sinon si (tl <= t <= tn)
// trans. externe
          e  <- t - tl
          s  <- delta_ext(s,e,x)
          tl <- t
          tn <- tl + ta(s)
```

- Définition du modèle couplé (ou réseau de modèles) :

$$N = \langle X, Y, D, EIC, EOC, IC \rangle$$

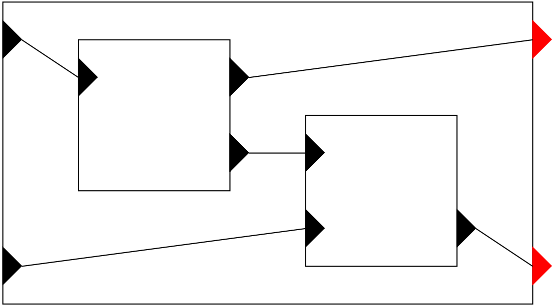
  - Une structure mathématique composée uniquement de variables
  - Une représentation graphique :





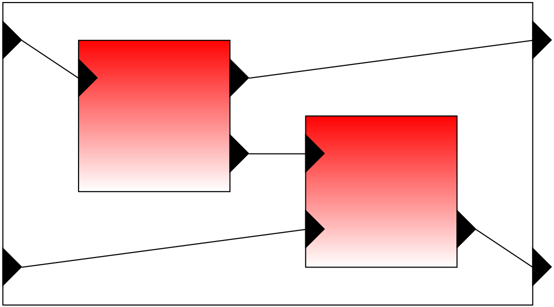
$$N = \langle X, Y, D, EIC, EOC, IC \rangle$$

X l'ensemble des ports d'entrée et des valeurs associées.



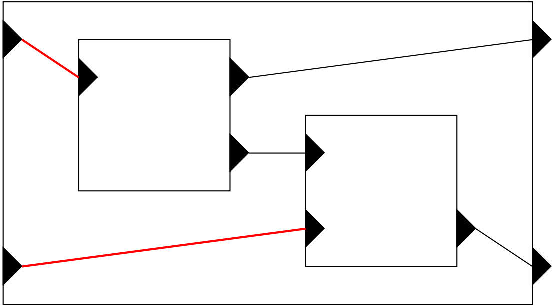
$$N = \langle X, Y, D, EIC, EOC, IC \rangle$$

Y l'ensemble des ports de sortie et des valeurs associées.



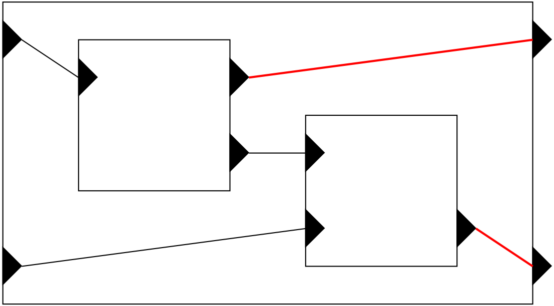
$$N = \langle X, Y, D, EIC, EOC, IC \rangle$$

D l'ensemble des identifiants des sous-modèles avec : $\{M_d | d \in D\}$.



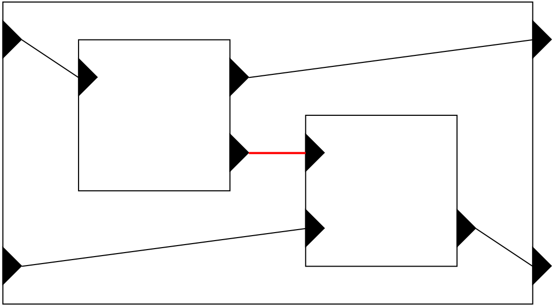
$$N = \langle X, Y, D, EIC, EOC, IC \rangle$$

EIC l'ensemble des connexions d'entrée.



$$N = \langle X, Y, D, EIC, EOC, IC \rangle$$

EOC l'ensemble des connexions de sortie.

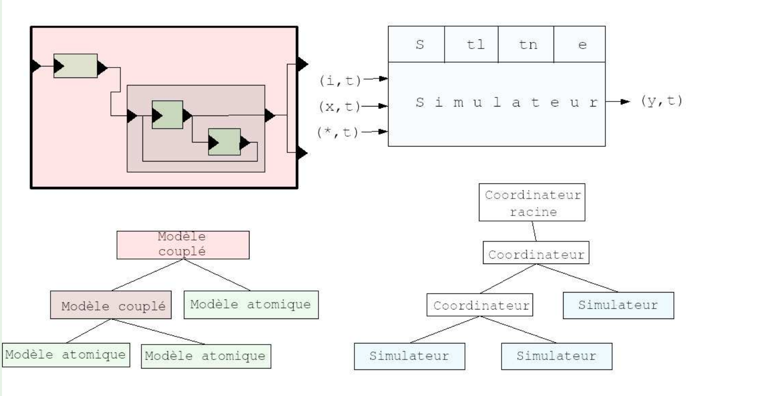


$$N = \langle X, Y, D, EIC, EOC, IC \rangle$$

IC l'ensemble des connexions internes.

Composition hiérarchique de modèles

Relation modèle formel – simulateur abstrait



Simulateur abstrait

Variables

```
N =< X, Y, D, (M_d), (I_d), (Z_d) > // le modèle couplé associé
// avec I_d ⊆ D ∪ N l'ensemble des influencers de d
// et Z_d : ×_{i∈I_d} YX_i → XY_d ⊆ D ∪ N
//       l'esemble des ports interfaces pour d
//       où:
//           XY_i = X_i si i = N
//           XY_i = Y_i si i ≠ N
//           XY_d = Y_d si d = N
//           XY_d = X_d si d ≠ N
```

Simulateur abstrait

Variables

```
parent                      // coordinateur parent
tl          // durée depuis le dernier évènement
tn          // durée avant le prochain évènement
event_list // liste des élements (d, tn_d) triés par par tn_d
IMM          // Les enfants imminents
mail          // bag de sortie
y_parent          // bag de sortie pour le parent
(y_d)          // ensemble bags de sortie pour les enfants
```

Simulateur abstrait

Initialisation

quand reçoit i-message (i,t):
  $\forall d \in D$ faire:
    envoie i-message à d
  trie la liste des évènements en fonction de $tn_d \forall d \in D$
  tl = $max\{tl_d \mid d \in D\}$
  tn = $min\{tn_d \mid d \in D\}$

Fonctions de sortie imminentes chez les composants

quand reçoit *-message (*,t):
  IMM = $\{d \mid (d, tn_d) \in (event\_list \wedge tn_d = tn)\}$
  $\forall r \in$ IMM faire
    envoie *-message (*,t) à r

Transitions imminentes chez les composants

```
quand reçoit x-message (x,t):
  receveurs = {r | r ∈ children, N ∈ I_r, Z_{N,r}(x) ≠ ∅}

  ∀r ∈ receveurs faire :
    envoie x-message (Z_{N,r}(x), t) à r

  ∀r ∈ IMM ∧ ∀r ∉ receveurs faire :
    envoie x-message (∅, t) à r

  trie event_list en fonction de t_n
  tl = t
  tn = min{tn_d | d ∈ D
```

Fonctions de sortie imminentes chez le modèle couplé

quand reçoit y-message (y_d,t):
  si $d \neq dernier_d \in IMM$ alors: // récup. des sorties
    add $y_d \rightarrow mail$
    marque d comme étant à traiter

  sinon :
    // traitement des sorties du modèle couplé
    $y_{parent} = \emptyset$
    $\forall d \in I_N \wedge d$ à traité faire :
      si $Z_{d,N}(y_d) \neq \emptyset$ alors :
        add $y_d \rightarrow y_{parent}$
    envoie y-message (y_{parent}, t) au parent

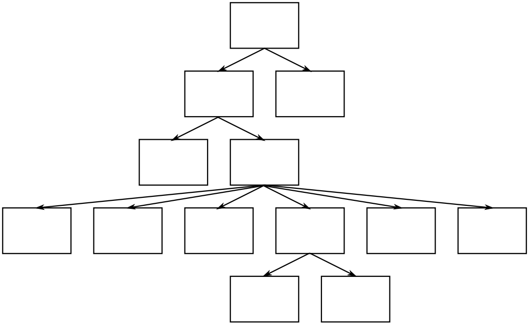
Simulateur abstrait

Fonctions de sortie imminentes chez le modèle couplé

```
quand reçoit y-message (y_d,t):
    ...
  sinon :
    // traitement des connexions internes
    ∀r | d ∈ I_r ∧ d à traiter ∧Z_{d,r}(y_d) ≠ ∅ faire :
      ∀d ∈ I_r ∧ d à traiter ∧Z_{d,r}(y_d) ≠ ∅ faire :
        add Z_{d,ri}(y_d) → y_r
    envoie x-messages (y_r, t) à r

    ∀r ∈ IMM ∧ y_r = ∅
      envoie x-message(∅, t) à r
    tl = t
    tn = min{tn_d | d ∈ D}
```

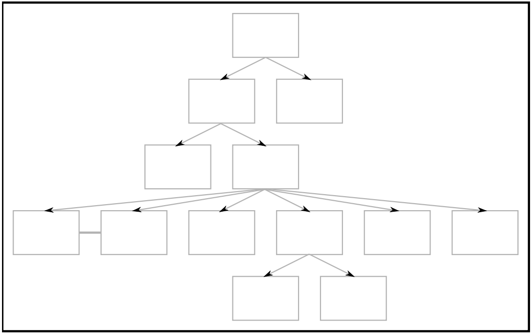
Composition hiérarchique de modèles



Propriété de fermeture sous couplage

Garantie qu'un modèle DEVS couplé est rigoureusement équivalent à un modèle DEVS atomique.

Composition hiérarchique de modèles



Propriété de fermeture sous couplage

Garantie qu'un modèle DEVS couplé est rigoureusement équivalent à un modèle DEVS atomique.

Propriété de fermeture sous couplage

Objectif

Montrer que :

$$N = < X, Y, D, (M_d), (I_d), (Z_d) >$$

peut être écrit sour la forme d'un modèle DEVS atomique :

$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, ta, \delta_{con}, \lambda \rangle$$

Méthode

Écrire le modèle résultant du modèle couplé en définissant les fonctions de transition, la fonction d'avancement du temps, la fonction de sortie et l'ensemble des états.

Propriété de fermeture sous couplage

- À un instant donné, en considérant qu'il n'y a pas d'évènement en entrée du modèle couplé, nous pouvons écrire:
  - L'état du modèle résultant est :

$$S = \times_{d \in D} Q_d$$

  - La fonction d'avancement du temps est:

$$ta(s) = min\{\sigma_d \mid d \in D\}, s \in S \wedge \sigma_d = ta(s_d) - e_d$$

  - La fonction de sortie est l'ensemble des fonctions de sortie du bag imminent:

$$\lambda(s) = \{Z_{d,N}(\lambda_d(s_d)) \mid d \in IMM(s) \wedge d \in I_N\}$$

  - De même, à partir de l'ensemble des composant imminents, on définit les fonctions de transition internes, externes et de conflict...

Les extensions de DEVS

Les extensions de DEVS permettent d'étendre formellement la spécification DEVS : nouveaux types modèles.

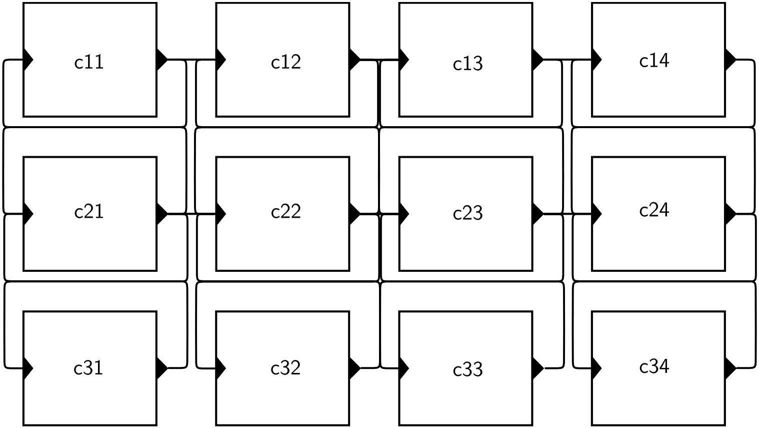
Quelques exemples pour les modèles :

Cell-DEVS

- l'état du modèle S contient son état et celui de ces voisins
- une diffusion instantanée λ ou avec délais δ_{int} des modifications ou perturbations aux voisins δ_{ext}
- la cellule est dupliquée pour former un automate cellulaire.

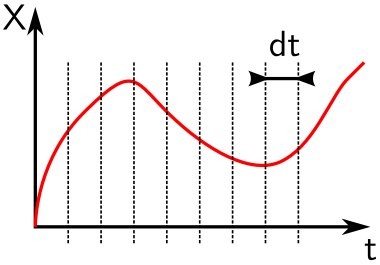
Les extensions de DEVS

Graphe de couplage d'un CellDevs

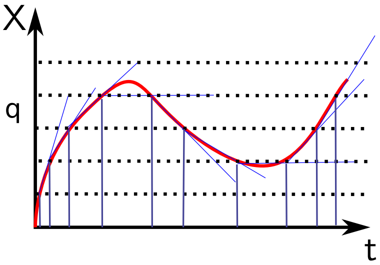


Comment résoudre les équations différentielles ordinaires :

- DESS [Zeigler et al., 2000] : discrétisation de l'espace du temps :
  - *ta* la constante du pas d'intégration
  - δ_{int} le moteur d'intégration (Euler, Runge Kutta, etc.).



- QSS [Kofman and Junco, 2001] : discrétisation de l'espace des valeurs :
  - δ_{int} calcul de la pente (pour QSS1)
  - *ta* calcul de la date de dépassement du seuil δ_q.

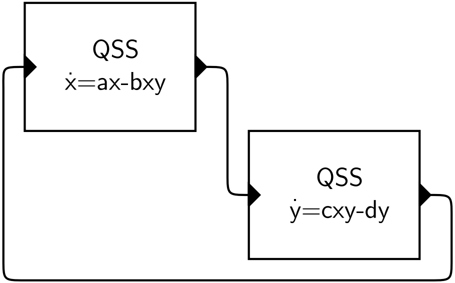


DEVS s'appuie, entre autre, sur deux propriétés très importantes :

- le couplage de modèles : un modèle atomique peut être combiné à un autre modèle pour former le couplage de modèle.
- la fermeture sous couplage : un modèle couplé possède les mêmes propriétés qu'un modèle atomique.

Développement d'un système d'équation

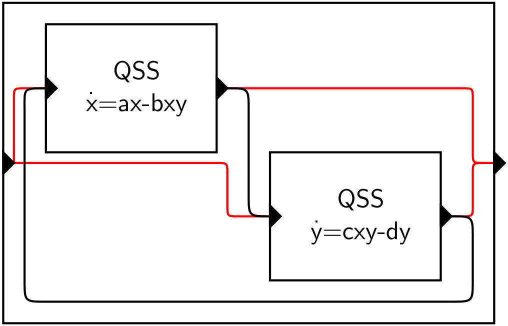
Exemple avec le modèle de Lotka-Voltera couplé à un modèle de décision



Deux modèles QSS avec une équation chacun : construction d'un système d'équations par le couplage.

Développement d'un système d'équation

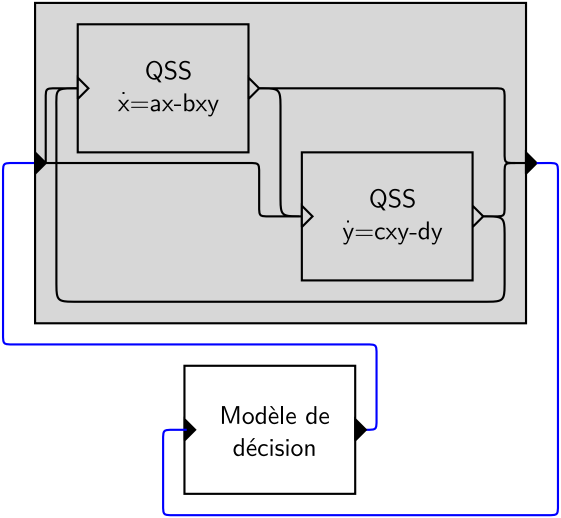
Exemple avec le modèle de Lotka-Voltera couplé à un modèle de décision



Le couplage peut former un nouveau modèle dont seuls les ports d'entrée et de sortie permettent la communication.

Développement d'un système d'équation

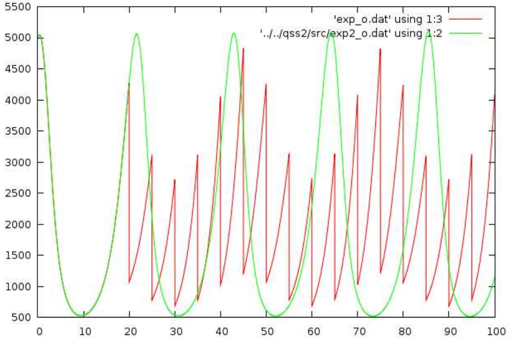
Exemple avec le modèle de Lotka-Voltera couplé à un modèle de décision



Un modèle de décision peut venir se coupler sur ce modèle

Développement d'un système d'équation

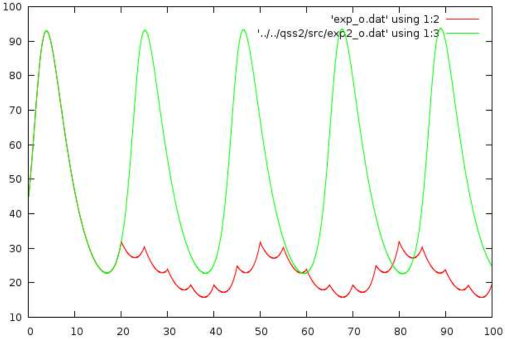
Exemple avec le modèle de Lotka-Volterra couplé à un modèle de décision



1. Vert, variable X (proies) sans perturbation.
2. Rouge, variable X avec une perturbation commençant quand les proies dépassent le seuil de 2000 puis toutes les 5 ut. (-75% de X).

Développement d'un système d'équation

Exemple avec le modèle de Lotka-Volterra couplé à un modèle de décision



1. Vert, variable Y sans perturbation.
2. Rouge, variable Y lorsque la variable X répercute ses modifications.

- DEVS possède une structure statique : modèles couplés/atomiques
  - $\longrightarrow$ offrir une structure dynamique permet de répondre à la diversité des modèles
- La spécification choisie : *Dynamic Structure DEVS* [Barros, 1995].
  - $\longrightarrow$ ajoute un modèle atomique au réseau de modèles : exécutif
  - $\longrightarrow$ le modèle exécutif possède le contrôle complet de la structure du modèle couplé :
    - ajouts et suppressions de modèles, de connexions et/ou de ports.

Réseau de modèles : déportent les connexions dans un modèle dit *exécutif* :

$$DSDEVN_N = \langle X_N, Y_N, \chi, M_\chi \rangle$$

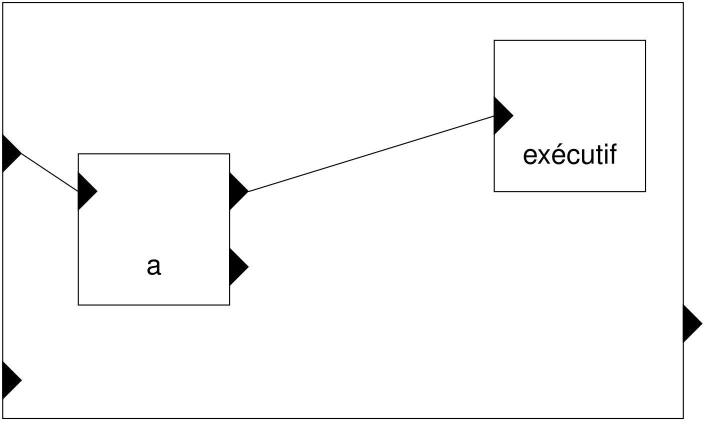
où :

- X_N et Y_N sont les ports d'entrée et de sortie du réseau
- χ le nom du modèle *exécutif*
- M_χ le modèle *exécutif*

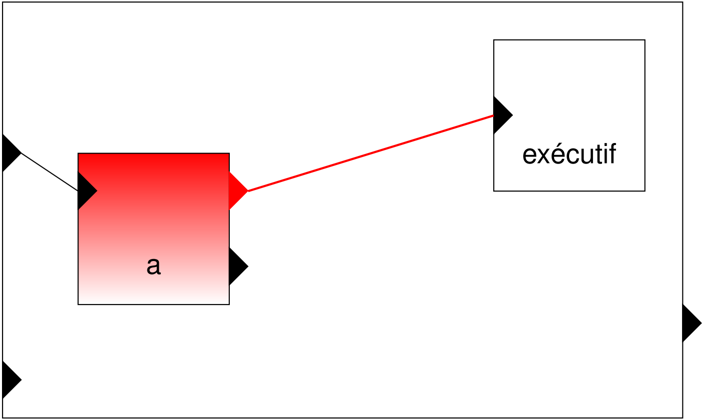
$$M_\chi = \langle X_\chi, Y_\chi, S_\chi, \delta_{int}^\chi, \delta_{ext}^\chi, ta_\chi, \lambda_\chi, \Sigma^*, \gamma \rangle$$

Avec :

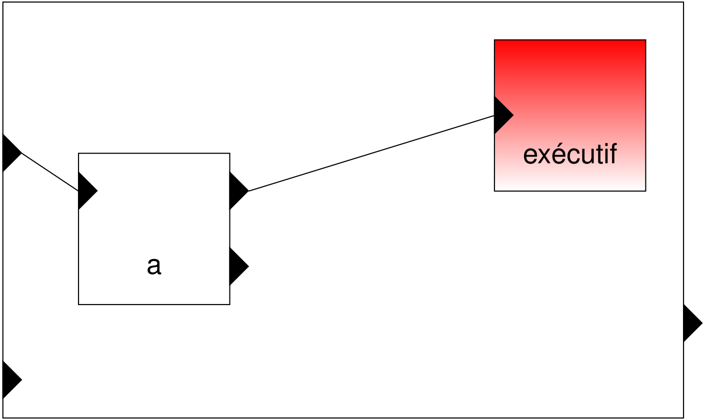
$$\begin{cases} \gamma : S_\chi \rightarrow \Sigma^* \text{ la fonction de structure} \\ \Sigma* : \text{ l'ensemble des structures} \\ \Sigma_\alpha \in \Sigma^* : \\ \qquad \Sigma_\alpha = \langle D, EIC, EOC, IC \rangle \end{cases}$$

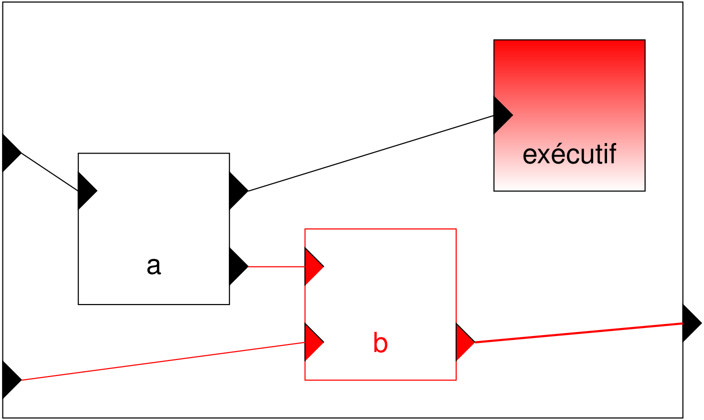


exécutif
a



exécutif
a


exécutif
a



exécutif
a
b

Les extensions de DEVS

Qss1, Qss2 et Qss3 Moteur d'intégration d'équations différentielles

Cell-Qss Moteur d'intégration d'équations différentielles spatialisées

Réseau de Petri Traduction (*mapping*) des RdP en DEVS

Automate à état finis Traduction des FSA en DEVS

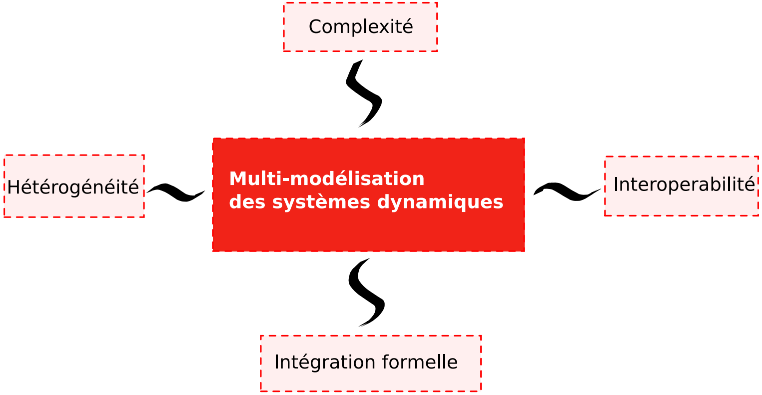
etc.

GSMP et GSMDP Processus de Markov généralisé, en cours par Emmanuel Rachelson

Plan

Modélisation multiple

Définitions



Modélisation multiple

Définitions

- Un multi-modèle est un modèle qui rassemble plusieurs paradigmes et/ ou formalisme dans sa réalisation.

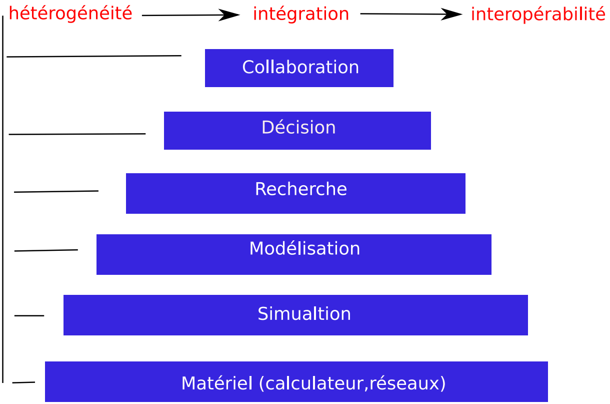
  $\Rightarrow$ Augmentation de la puissance descriptive du modèle.

- La multi-modélisation est l'ensemble des concepts, outils et techniques de construction de multi-modèles.

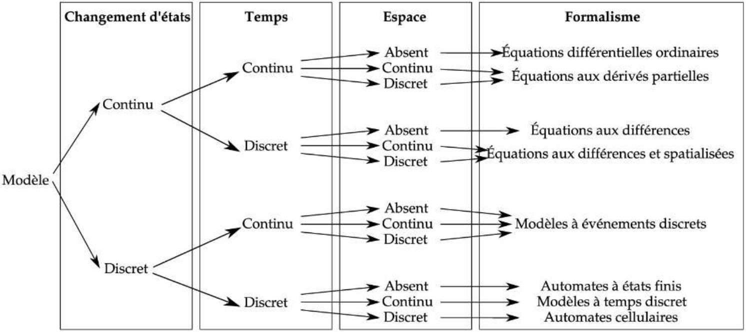
Les précurseurs

- [Ören, 1989] Dynamics Templates and Semantic Rules for Simulation Advisters and Certifiers.
- [Fishwick and Zeigler, 1992] - A multi-model methodology for qualitative model engineering.

Cadre conceptuel de la modélisation et de la simulation [Zeigler and Sarjoughian, 2000]



États, temps, espace [Ramat, 2003]



Trois directions orthogonales

- L'intégration de plusieurs formalismes dans un nouveau
  - ⇒ [Vangheluwe, 2000] (Differential Algebraic Equations),
  - ⇒ [Zeigler et al., 2000] DEV&DESS,
  - ⇒ [Barros, 2003] (HFSS) pour l'intégration de systèmes discrets et continus.
- La spécification des sous systèmes dans un formalisme unique
  - ⇒ Réecriture de tous les sous systèmes dans le même formalisme.
- L'approche de co-simulation. Tous les sous modèles ont leur propres simulateurs. La difficulté et de les coupler (solution HLA par exemple).

Trouver des moyens techniques et algorithmiques de coupler des modèles hétérogènes

- Trouver des moyens techniques

  $\Rightarrow$ Problèmes d'architecture matérielle et logicielle

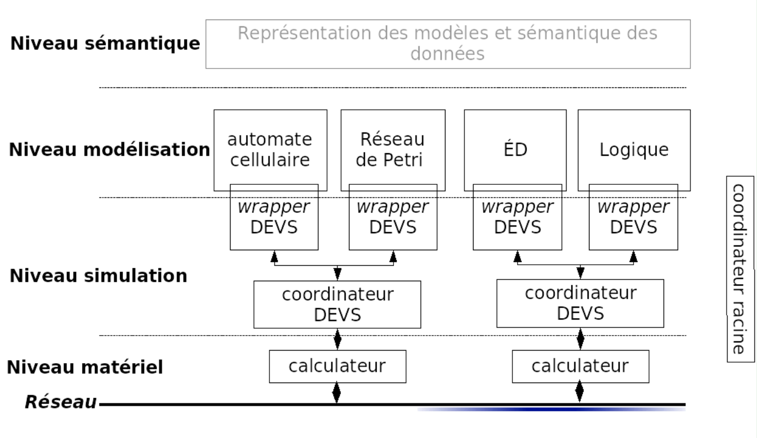
  $\Rightarrow$ Problèmes des langages de programmation différents

- Intégrer les représentations de la dynamique des sous systèmes

  $\Rightarrow$ Intégration des notions de temps, d'espace, d'états et de transition [Quesnel et al., 2004]

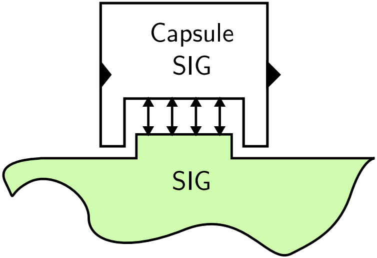
Besoin de définir un bus logiciel commun

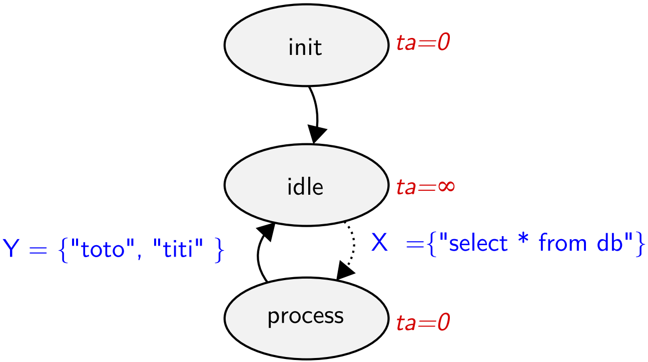
Schéma conceptuel du bus DEVS (modifié d'après [Kim and Kim, 1998])



Les capsules ou *wrappers*:

- sont des modèles DEVS classiques
- font le lien entre un simulateur DEVS et un programme existant (traduction des notions de temps, d'états et de transitions).
- l'état d'un *wrapper* est l'union de son état et du composant encapsulé

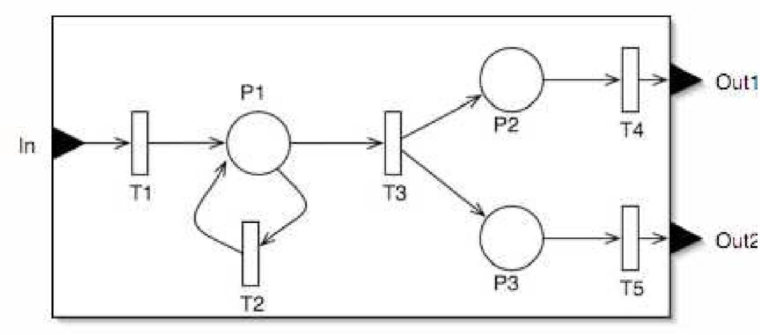




Graphe d'états d'un *wrapper* autour d'une base de données SQL.

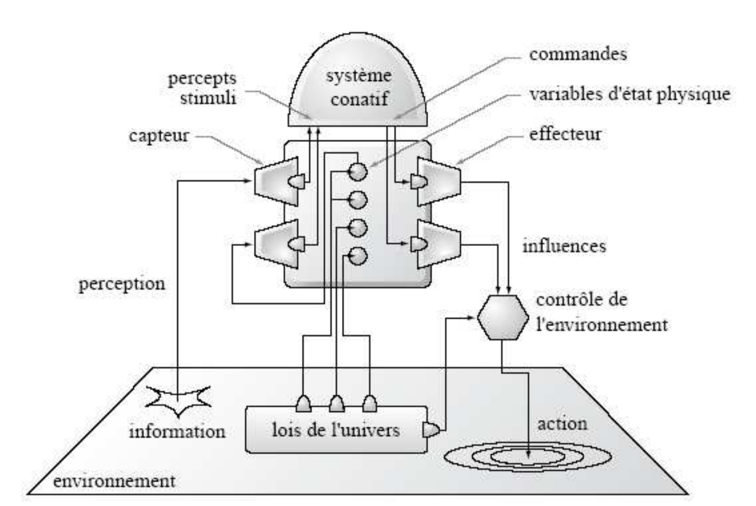
Intégration opérationelle

Exemple : wrapping d'un réseau de Petri



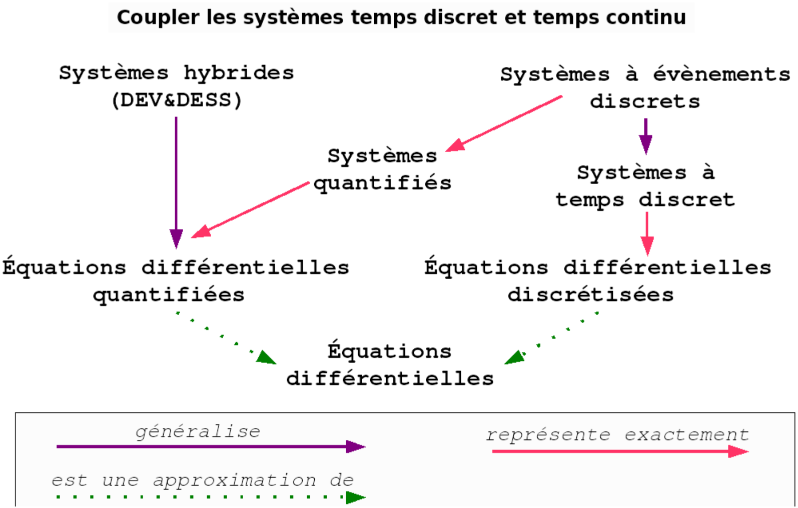
- Traduction :
  - Événement d'entrée en jetons (fonction de transition externe)
  - Fonction de transition interne (évolution du marquage)
  - Fonction d'avancement du temps → introduction du temps ?
  - Fonction sortie en fonction du marquage

Les agents dans DEVS [Duboz et al., 2004, Duboz et al., 2006]



Intégration formelle

Vers du "tout en DEVS"



Intégration formelle

mapping : Définition

- Reformalisation d'un système d'écriture dans un autre
- Préservation de la dynamque des états – transitions
- Homomorphisme (niveau 3)
- Pas forcement d'isomorphisme au niveau du modèle couplé

Il s'agit essentiellement de traduire une sémantique particulière dans DEVS

Exemples de reformalisation

- Les automates à états finis
- Les automates cellulaires
- Les systèmes à temps discrets
- Les chaînes de Markov temporisées

Exemples de reformalisation

Les réseaux de Petri [Jacques and Wainer, 2002]

- Bien adapté pour la simulation de la concurence des évènements
- Trois éléments : les places, les transitions et les arcs
- Une place à 0 ou + entrées et sorties
- Une transition peut avoir 0 ou + entrées et sortie (si 0 sortie alors c'est un puit)
- Le mouvements des jetons (ressource par exemple) décrit la dynamique du système sur les places (lieux par exemple)
- Une transition est permise si au moins un jeton est présent sur une des ses places connectée en entrée
- Quand une transition est déclanchée, elle retire un jeton de chaque place connectée en entrée et en dépose un sur chaque place connectée en sortie.

une place peut être formalisée par:

$$M =< X, S, Y, \delta_{int}, \delta_{ext}, D, \lambda >$$

$$X = \{IN \in N^+\}$$

$$Y = \{OUT \in N^+\}$$

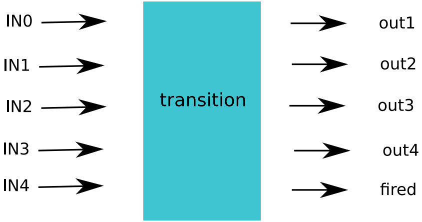
$$S = \{tokens \in N_0^+\} \cup \{id \in N^+\}$$

tokens est le nombre de jetons
id est l'identifiant de la place

Exemples de reformalisation

Les réseaux de Petri

```
dext (s,e,x) {
retrieve id and number of tokens from message
case id = 0 /* generic message */
increment tokens
    hold in active 0 /* to advertise the number
    of tokens */
case id != 0 /* specific message */
    id matches id of this place?
       no: disregard the message
       yes: decrement tokens by the number of
tokens specified if there are enough. Otherwise
throw an exception.
    hold in active 0 /* to advertise the number
    of tokens */
}
dint (s) {
}
lambda(s) {
combine id and tokens state variables in one message
and send on the out port.
}
```



une transition peut être formalisée par:

$$M = < X, S, Y, \delta_{int}, \delta_{ext}, D, \lambda >$$

$$X = \{IN0 \in N^+, IN1 \in N^+, IN2 \in N^+, IN3 \in N^+, IN4 \in N^+\}$$

$$Y = \{OUT1 = 1, OUT2 = 2, OUT3 = 3, FIRED \in N^+\}$$

$$S = \{inputs \in N_0^+\} \cup \{enable \in bool\}$$

inputs nombre de place en entrée
enable transition possible ou non

- *IN*1 combien de jeton dans la place ?
- *IN*2 idem mais modélise une double connexion
- *IN*3 idem mais modélise une triple connexion
- *IN*4 idem mais modélise une quadruple connexion
- *IN*0 inibiteur (pas de jeton sur la place)
- *OUT*1 donne un jeton
- *OUT*2 donne 2 jetons
- *OUT*3 donne 3 jetons
- *OUT*4 donne 4 jetons
- *fired* suprime les jetons de la place en entrée

Exemples de reformalisation

Les réseaux de Petri

```
dext (s,e,x) {
case port
    in0: set arcwidth (temp var) to 0.
    in1: set arcwidth (temp var) to 1.
    in2: set arcwidth (temp var) to 2.
    in3: set arcwidth (temp var) to 3.
    in4: set arcwidth (temp var) to 4.
- extract id (place of origin) and number of tokens
from the message.
First message we get from this id?
yes: increment inputs.
no: continue
- save id, arcwidth and number of tokens in database.
- scan the entire database to determine if all input
places have enough tokens to enable the transition.
```

```
    transition is enabled ?
        yes: set enabled to true
        hold in (active, random (0 â 60 sec))
        no: set enabled to false
        passivate
}
dint (s) {
    inputs = 0?
        yes: /* transition is a source */
        hold in (active, random (0 â 60 sec))
        no: passivate
}
lambda(s) {
- send 1 on out1 port.
- send 2 on out2 port.
- send 3 on out3 port.
- send 4 on out4 port.
- go through the database and send a message to
every input place via the fired port.
}
```

Points importants non abordés ici

- Universalité de DEVS
- Validation
- Vérification
- Approximation (savoir vivre avec l'erreur)
- Les cadres expérimentaux

Barros, F. J. (1995).
Dynamic structure discrete event system specification: A new modeling and simulation formalism for dynamic structure systems.
In *Proceedings of the 1995 Winter Simulation Conference*, pages 781–785.

Barros, F. J. (2003).
Dynamic structure multiparadigm modeling and simulation.
ACM Transactions on Modeling and Computer Simulation, 13(3):259–275.

Duboz, R., Ramat, É., and Quesnel, G. (2004).
Systèmes multi-agents et théorie de la modélisation et de la simulation : une analogie opérationnelle.
In Boissier, O. and Guessoum, Z., editors, *Actes des douzièmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA) - Systèmes multi-agents défis scientifiques et nouveaux usages*, Paris.

Duboz, R., Versmisse, D., Quesnel, G., Muzzy, A., and Ramat, É. (2006).

Specification of dynamic structure discrete event multiagent systems.
In *Conference proceedings of Agent Directed Simulation (Spring Simulation Multiconference)*, Huntsville, Alabama, USA.

Fishwick, P. A. and Zeigler, B. P. (1992).
A multi-model methodology for qualitative model engineering.
ACM transaction on Modeling and Simulation, 2(1):52–81.

Jacques, C. and Wainer, G. A. (2002).
Using the CD++ DEVS tookit to develop petri nets.
In *SCS Conference*.

Kim, J. Y. and Kim, T. G. (1998).
A heterogeneous simulation framework based on the DEVS bus and the High Level Architecture.
In *Winter Simulation Conference*, Washington, DC.

Kofman, E. and Junco, S. (2001).
Quantized State Systems. a DEVS Approach for Continuous Systems Simulation.
In *Transactions of SCS.*, volume 18, pages 123–132.

Ören, T. (1989).
Knoledge Based Simulation: Methodology and Application, chapter Dynamics Templates and Semantic Rules for Simulation Advisters and Certifiers.
Berlin Springer Verlag.

Quesnel, G., Duboz, R., and Ramat, É. (2004).
DEVS wrapping: A study case.
In *Proceedings of Conference on Conceptual Modeling and Simulation 2004*, pages 374–382, Genoa, Italy.

Ramat, É. (2003).
Contributions à la modélisation et à la simulation des systèmes complexes.
Habilitation à diriger des recherches.

Vangheluwe, H. (2000).
DEVS as a common denominator for hybrid systems modelling.

In Varga, A., editor, *IEEE International Symposium on Computer-Aided Control System Design*, pages 129–134, Anchorage, Alaska. IEEE Computer Society Press.

Zeigler, B. and Sarjoughian, H. S. (2000).
Creating distributed simulation using devs m& s environment.
In *Proceedings of the 2000 Winter Simulation Conference*.

Zeigler, B. P. (1976).
Theory of Modeling and Simulation.
Wiley Interscience.

Zeigler, B. P., Kim, D., and Praehofer, H. (2000).
Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.
Academic Press.

Some slides comme from G. Quesnel. Thanks to him.

Licence