Coordination et interaction dans une société d' agents autonomes

GOUAICH Abdelkader

gouaich(at)lirmm.fr

LIRMM, Montpellier

Plan

- Partie I: Environnement de déploiement d'agents autonomes -> MIC*
- Partie II: Protocoles de coordination et protocoles de conversation dans un système multi-agents
- Partie III: Intégration du modèle de coordination et MIC*

Partie I:
Environnement de déploiement d'agents autonomes MIC*

Motivation

- Ingénierie de systèmes informatiques selon les principes du paradigme multi-agents: *cohérence paradigmatique.*
- Rappel des propriétés fondamentales des agents dans le paradigme multi-agent:
  1. Autonomie;
  2. Sociabilité et interaction;
  3. Pro-activité;

Pourquoi avons-nous besoin de représenter explicitement l'environnement déploiement dans un SMA?

- (a) Garantir l'autonomie des agents:
  - Interprétation de l'autonomie:
    - Indépendance sociale (SDN) [Scc. 94]
    - Capacité d'un système à contrôler et modifier ses propres lois (internes) [Steel 95], [Castel. 95]…
    - Autonome vs. Automatique
  - Intégrité structurelle de l'agent est posée comme une condition nécessaire pour garantir l'autonomie
  - Intégrité structurelle:
    - Aucun agent (système externe) ne peut accéder, ni modifier l'état interne d'un autre agent;
  - Paradoxe! Les agents ne peuvent pas interagir
  - Pour lever ce paradoxe on introduit l'environnement de déploiement
  - L'environnement de déploiement n'est pas un agent autonome, c'est un système *automatique* qui va acheminer les interactions entre les agents.
  - Aucun agent n'a accès directement à la structure d'un autre agent => intégrité structurelle inter-agents est garantie

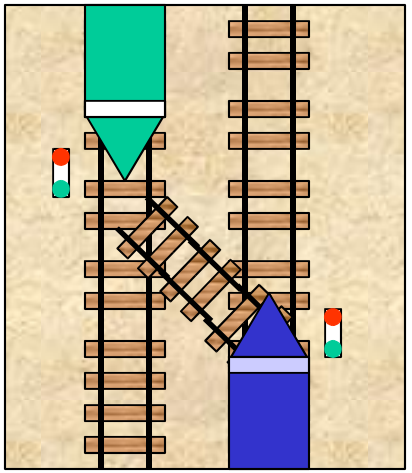
Pourquoi avons-nous besoin de représenter explicitement l'environnement déploiement dans un SMA?

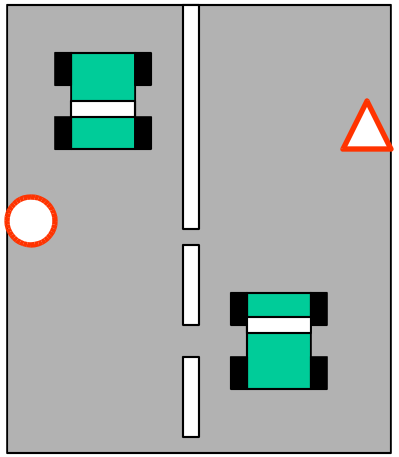
(b) Établir la responsabilité des agents autonomes:

- Les actions des agents sont considérées comme des tentatives pour changer l'univers
- Ces tentatives ne sont validées que si elles sont conformes aux lois/normes de l'environnement de déploiement

Modèle route et rail

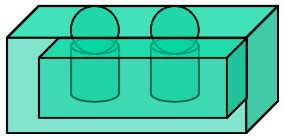
- Systèmes à entités *autonomes interactionnelles*.
- Interaction locale entre les entités
- Le système est régi par des conventions qui garantissent sa sécurité (cohérence)
- On observe des différences entre les deux systèmes:
  - Le modèle « rail » semble garantir mieux les règles de cohérence que le modèle « route »
- Pourquoi?
  - Le modèle «rail» représente explicitement les règles de cohérence comme des propriétés de l'environnement
  - Le modèle «route» délègue la gestion des règles de cohérence à des entités autonomes



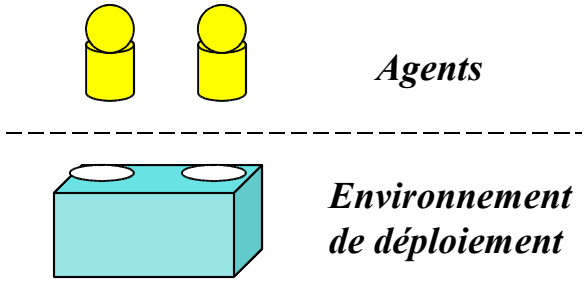


Nos objectifs…

- Séparation explicite d'un système informatique en un environnement de déploiement et des entités qui le peuplent.

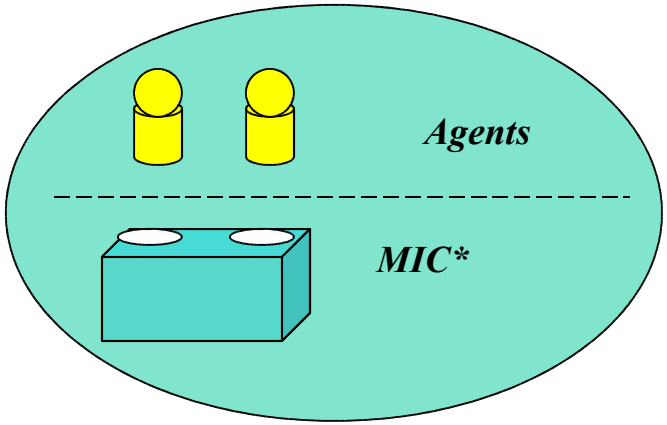


- Étudier les concepts fondamentaux de cette structure sans considérer un modèle particulier des agents



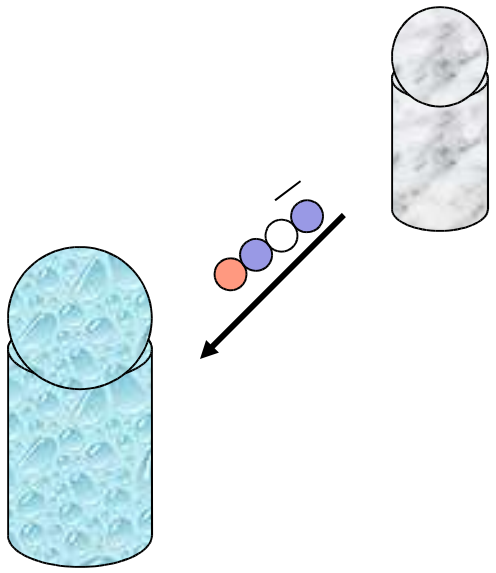
Concepts Fondamentaux de MIC*

- Description statique:
  - Objet d'interaction
  - Espace d'interaction
  - Processus de calcul (Agent)
- Dynamique de la structure:
  - Le mouvement
  - L'interaction
  - Le calcul



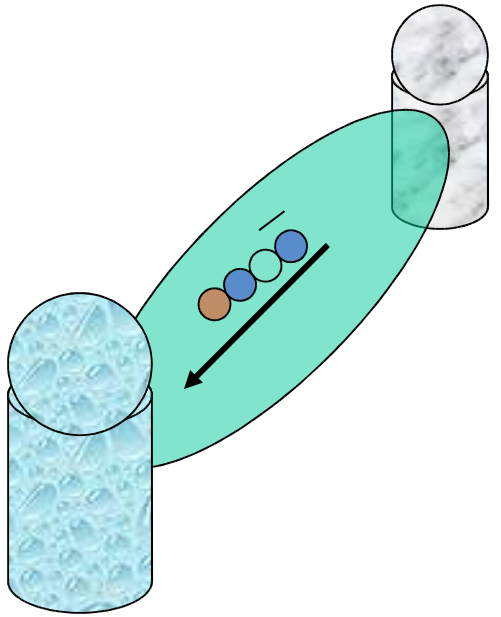
Objet d'interaction

- Objets d'interaction:
  - L'interaction est conduite à travers des objets d'interaction $o \in O$.
  - $(O,+)$ possède une structure de groupe commutatif.
  - Les objets d'interaction peuvent être composés: $x + y$.
  - $0 \in O$, L'objet d'interaction neutre.
  - Pour $x \in O$, $-x \in O$.



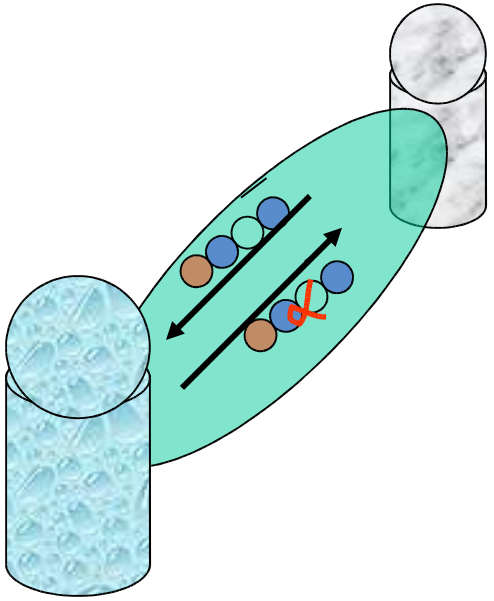
Espace d'interaction

- Espace d'interaction:
  - Abstraction qui définit une localisation/contexte où différentes entités interagissent.
  - Entité active, qui régit les règles d'interaction entre les différentes entités.
    - Par exemple: L'espace d'interaction peut altérer les objets d'interaction émis par les agents.



Processus

- Processus (Agent):
  - Entité autonome qui perçoit et réagit aux interactions externes par un calcul local et l'émission de nouveaux objets d'interaction.
  - Les interactions de l'agent ne sont effectives que si elles sont en adéquation avec les règles de l'espace d'interaction.



La structure statique de MIC*

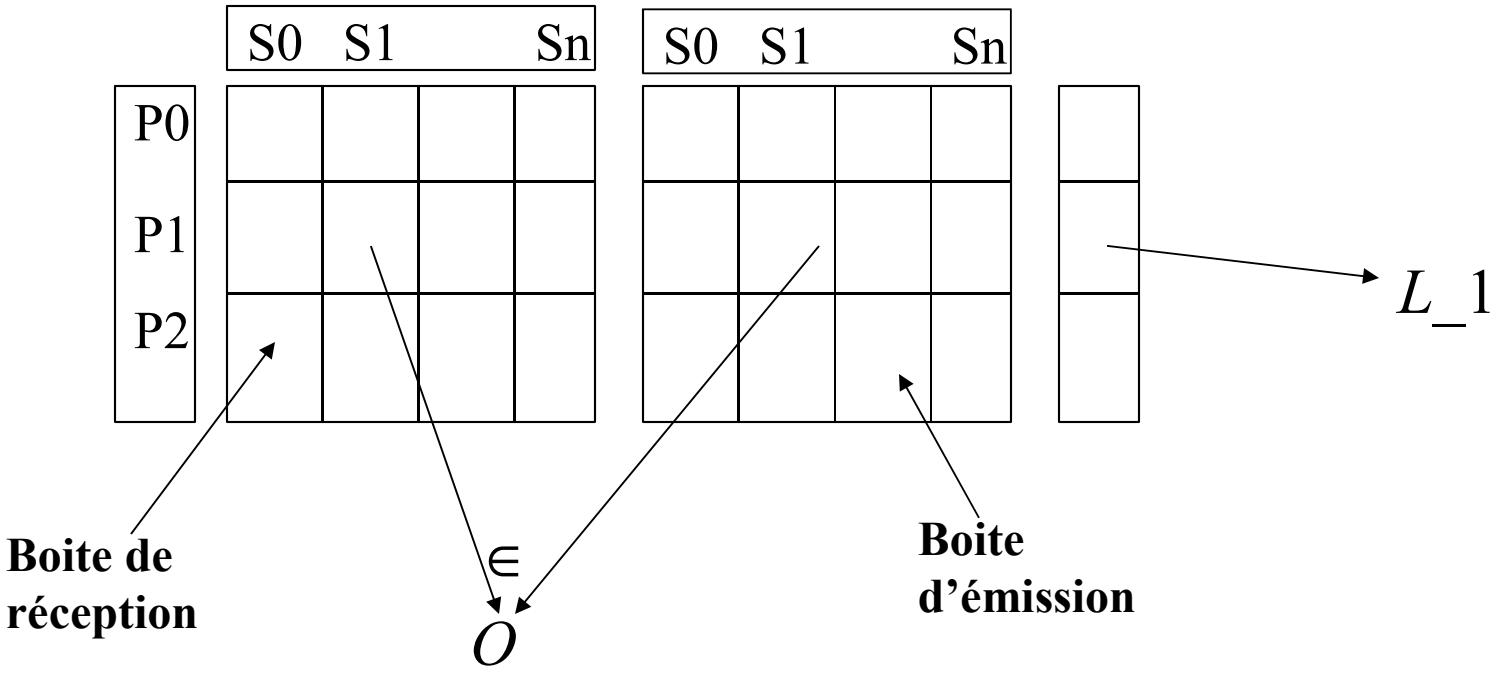
- $(O, +)$ un groupe abélien qui représente les objets d'interaction.
- *I, J* deux ensembles représentant les processus et les espaces d'interaction.
- $(L\_i)i \in I$ une famille d'ensembles pointés où *L_i* représente les termes du langage interne du processus *i*.
- Soit $O^{(I \times J)}$ l'ensemble des familles $(a_{i,j})_{(i,j) \in I \times J}$ éléments de O .
- Soit ℓ l'ensemble des familles $(m_i)_{i \in I}$ avec m_i élément de *L_i*.
- Un environnement MIC* est défini comme suit:

$$\underbrace{O^{(I \times J)}}_{\text{Matrice de Perception}} \times \underbrace{O^{(I \times J)}}_{\text{Matrice d'Émission}} \times \underbrace{\ell}_{\text{Vecteur des Mémoires}}$$

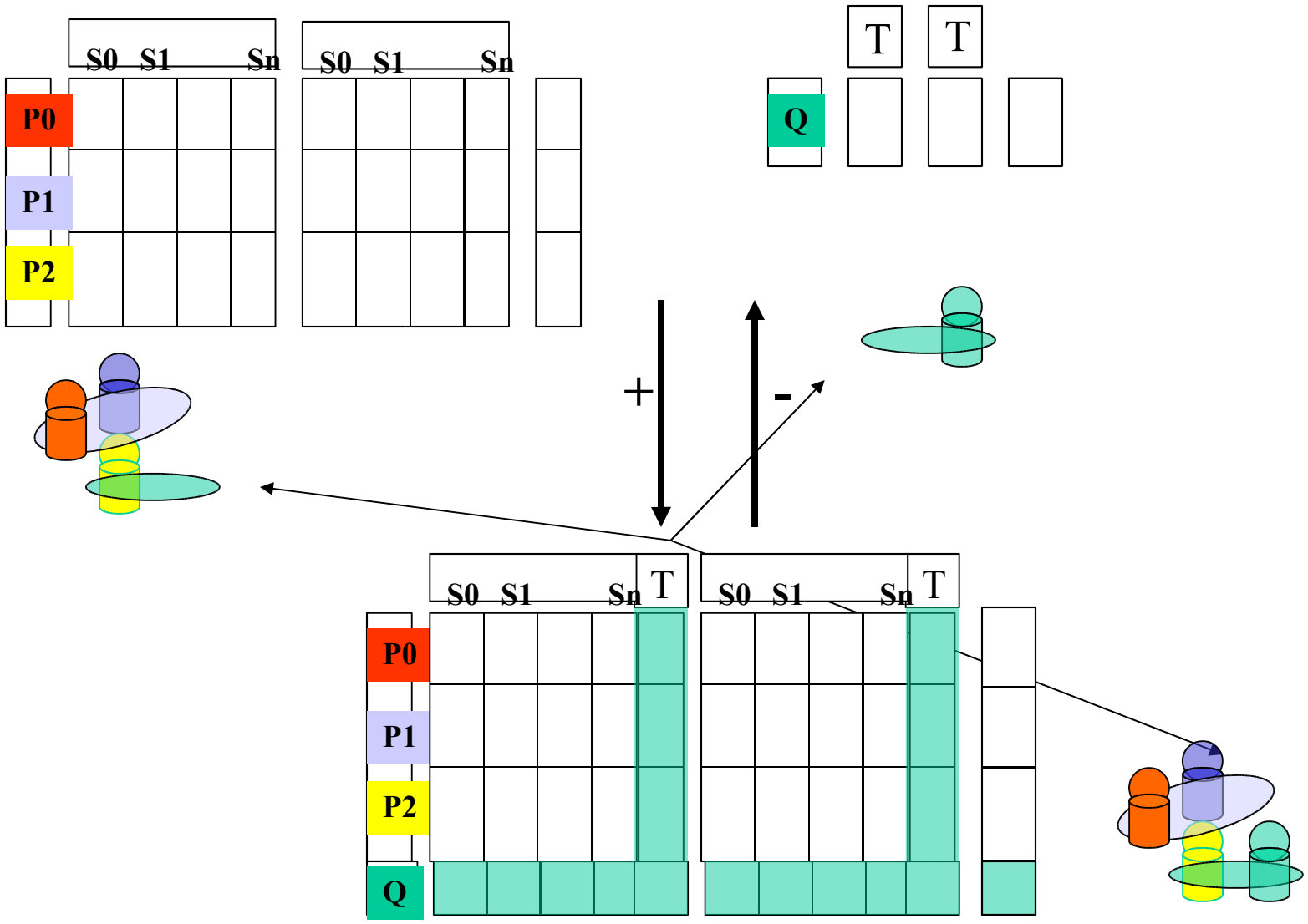
Matrices associées à MIC*

$$\underbrace{O^{(I\times J)}}\times\underbrace{O^{(I\times J)}}\times\underbrace{\ell}$$

Matrice de Perception *Matrice d'Émission* *Vecteur des Mémoires*

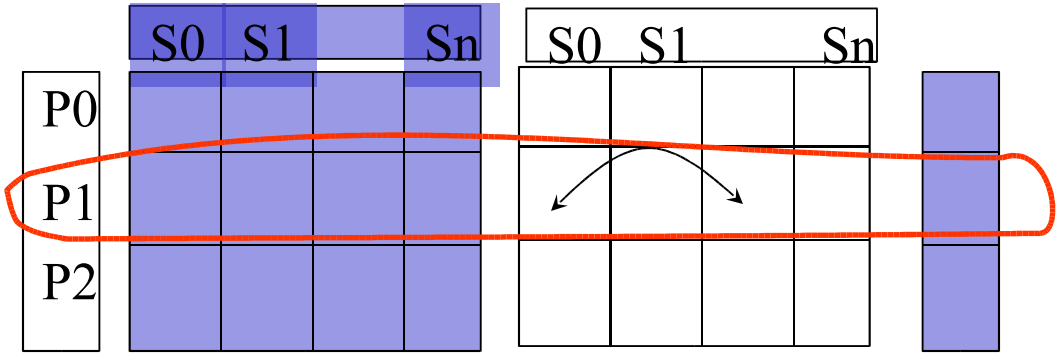


Composition des termes MIC*



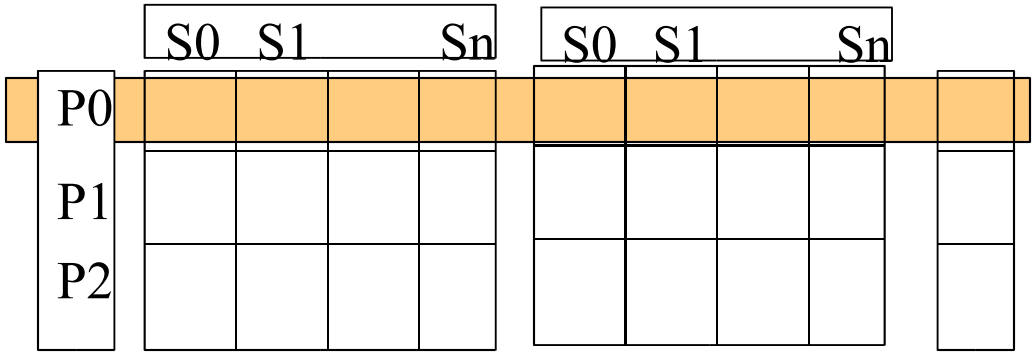
Règles d'évolution: Le mouvement

- Le mouvement est une application μ qui vérifie les propriétés suivantes:
  - μ ne modifie pas les boites de réception ni la mémoire.
  - μ conserve globalement les boites d'émission d'un processus donné.
  - Les boites d'émission après mouvement d'un processus donné ne dépendent que des boites d'émission avant mouvement du même processus.



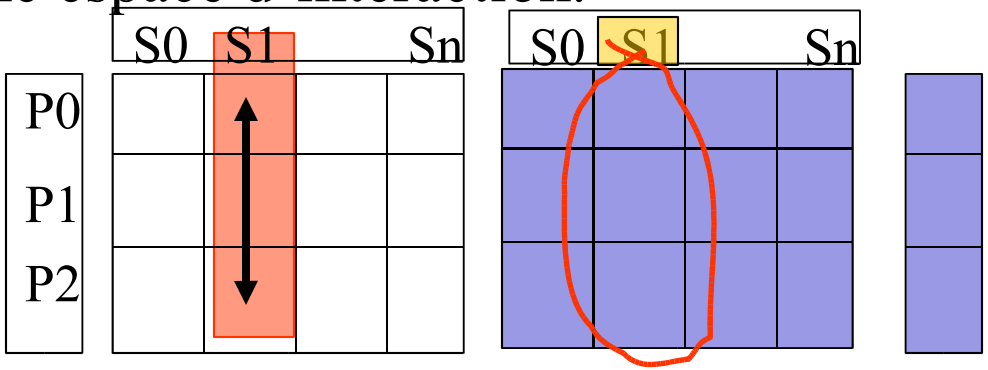
Règles d'évolution: Le Calcul

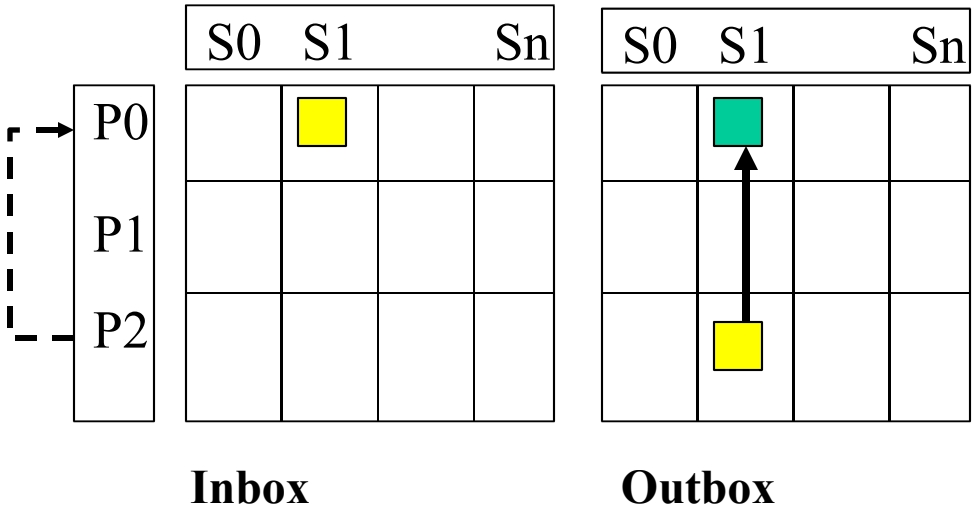
- Le calcul est une application γ qui vérifie les propriétés suivantes:
  - γ consomme les objets d'interaction présent dans le vecteur de réception d'un processus.
  - Les valeurs des boites d'émission et de la mémoire après calcul d'un processus donné ne dépendent que des valeurs des boites d'émission, de réception et de la mémoire avant calcul de *ce processus*.



Règles d'évolution: L'interaction

- L'interaction est une application φ qui vérifie les propriétés suivantes:
  - φ ne modifie pas les boites d'émission ni la mémoire.
  - si un processus n'est pas présent dans un espace d'interaction il ne peut interagir.
  - Les boites de réception après interaction d'un processus donné dans un espace d'interaction fixé ne dépendent que de leurs valeurs avant l'interaction et des boites d'émission des processus dans le même espace d'interaction.



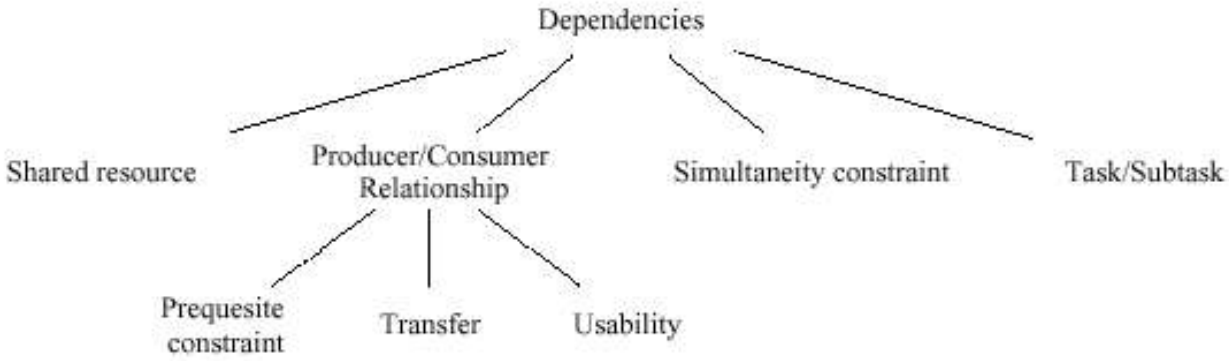


Fonction/règle d'interaction

Comment contrôler les protocoles d'interaction (normes sur la coordination) par l'environnement de déploiement?

Partie II:
Protocoles de coordination et protocoles de conversation dans un système multi-agents

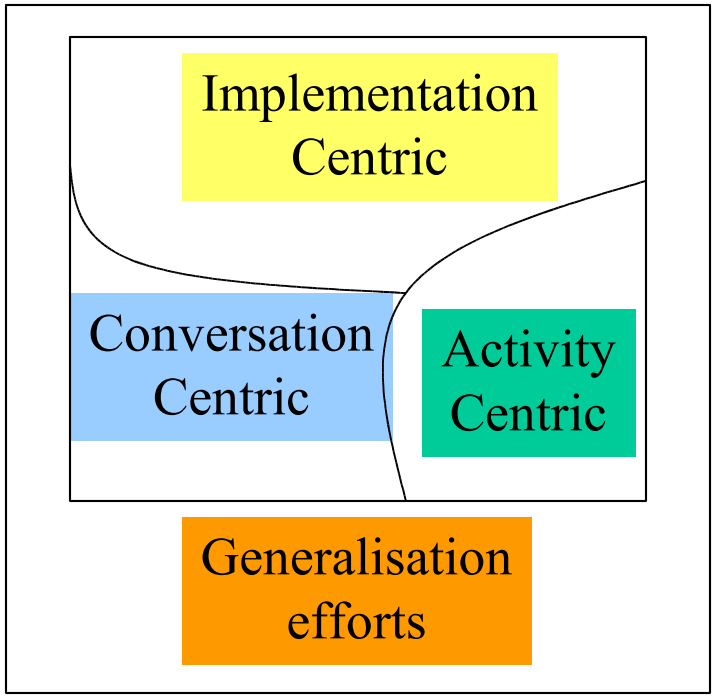
Prérequis: What is coordination?



- Malone and Crowson [mal94] define coordination as managing *dependencies* among activities
- Common dependencies among activities are:
  - Shared resource
  - Producer/Consumer
    - Prerequisite
    - Transfer
    - Usability
  - Simultaneity constraint
  - Task/Sub-Task

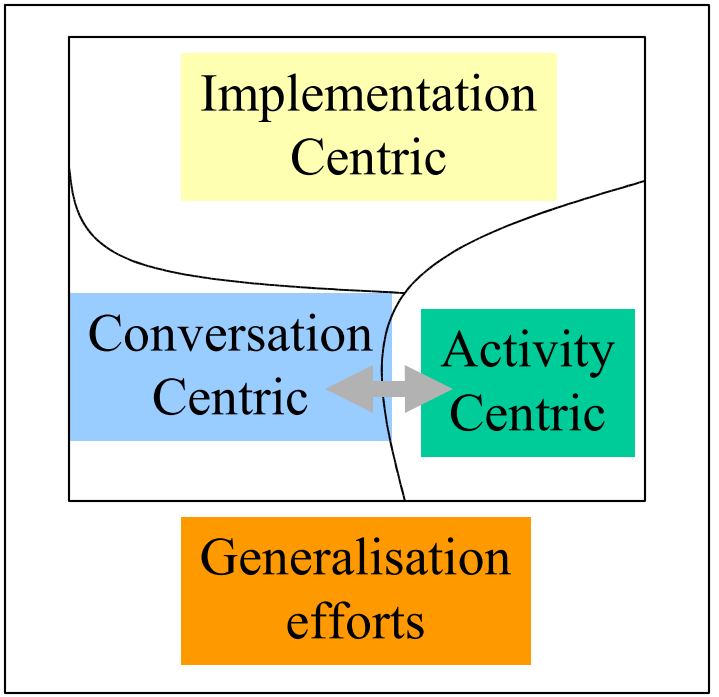
Backgrounds: What is coordination?

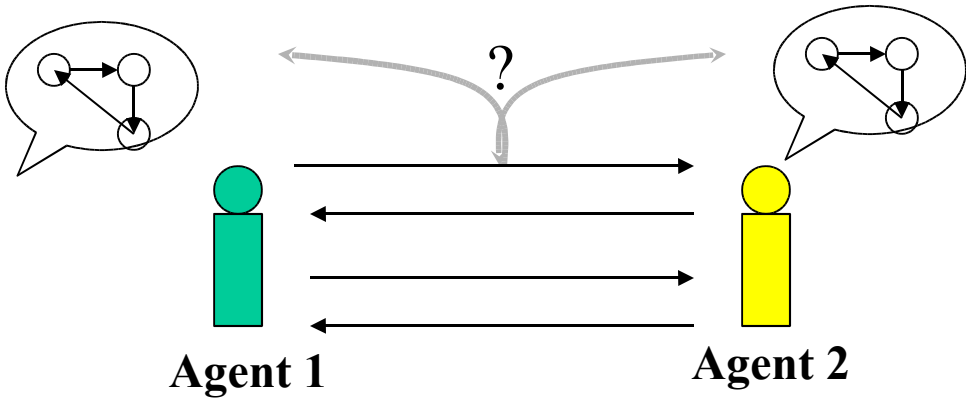
- 1) Focus is made on activity aspect of coordination:
  - What activity to execute?
  - When an activity should be executed?
  - Model to coordinate distributed tasks such as: Statecharts, Flowcharts, Process algebra, Lotos, SDL, Estelle …
- 2) Conversational Aspects of coordination
  - What is the structure of the conversation among the coordinating entities?
  - FSM, Petri-Nets, State Transition Diagrams
- 3) Implementation centric works
  - How to implement distributed software systems where software components coordinate their actions.
  - For instance, Linda-family coordination mediums



Problem statement

- What is the link between the conversation protocol and the coordination protocol ?
- To answer this question, the following methodology has been adopted:
  - Step 1: Starting from Malone and Crowson works a simple coordination model has been proposed
  - Step 2: Starting from this coordination model, the conversation protocol is formally induced as valid sequences of messages recognised by a defined grammar.





Modélisation

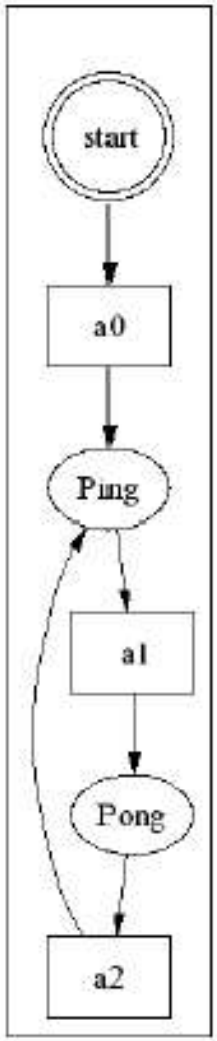
4.1.1 Dependency-based coordination model structure

Definition 4.1.1. The dependency-based coordination model is a four tuple $G = (A, R, C, P)$, $A = \{a_0, a_1, ..., a_n\}$ is a finite set of activities nodes ($n \geq 0$), $R = \{r_0, r_1, ..., r_m\}$ is a finite set of resources nodes ($m \geq 0$). $R \cap A = \emptyset$. $C : A \rightarrow \mathcal{P}(R)$ is defined as the consumption function, a mapping from activities nodes to a set of resources nodes; $P : A \rightarrow \mathcal{P}(R)$ is defined as the production function, a mapping from activities nodes to a set of resources nodes.

Definition 4.1.2. The digraph $DG = (V', A')$ associated to a dependency-based coordination graph $G = (A, R, C, P)$ is defined as follows:

$$
\begin{aligned}
V' &= A \cup R \\
\forall \quad & a \in A, \forall r \in C(a) : (r, a) \in A \\
\forall \quad & a \in A, \forall r \in P(a) : (a, r) \in A
\end{aligned}
$$

$$
\begin{aligned}
G_{\text{ping-pong}} &= (A, R, P, C) \\
A &= \{a_0, a_1, a_2\} \\
R &= \{\text{'start'}, \text{'ping'}, \text{'pong'}\} \\
P &: \begin{cases} P(a_0) &= \{\text{'ping'}\} \\ P(a_1) &= \{\text{'pong'}\} \\ P(a_2) &= \{\text{'ping'}\} \end{cases} \\
C &: \begin{cases} C(a_0) &= \{\text{'start'}\} \\ C(a_1) &= \{\text{'ping'}\} \\ C(a_2) &= \{\text{'pong'}\} \end{cases}
\end{aligned}
$$



Décomposition du graphe de coordination en ‘rôles’

Definition 4.1.3. Let $G = (A, R, C, P)$, be a coordination model, let $D = \{d_0, d_1, ..., d_j\}, j \geq 0$ be the finite set of roles. A role decomposition associated to the coordination model G is a mapping $d : D \rightarrow \mathcal{P}(A)$ that associates to each role $d \in D$ a set of activities such that:

$$\begin{aligned} \forall x, y \in D \times D : x &\neq y \implies d(x) \cap d(y) = \emptyset \\ \bigcup_{x \in D} d(x) &= A \end{aligned}$$

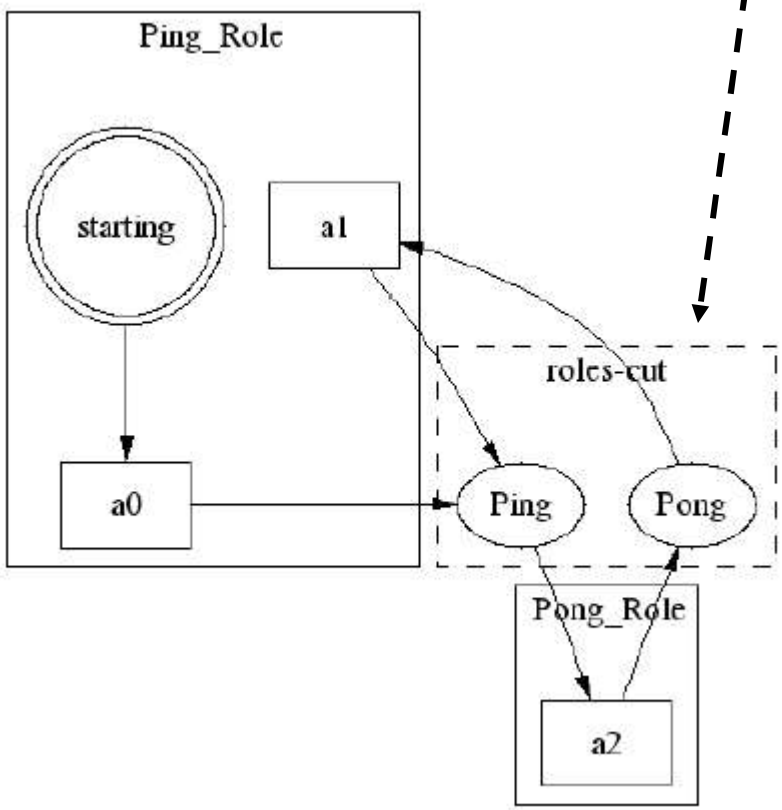
Definition 4.1.4. The roles-cut of a dependency-based coordination model $G = (A, R, P, C)$, is defined as the minimal set of resources $c \subset R$ that if all the resources $x \in c$ were removed along with there related production and consumption relations the G-associated digraph would have more connected components than the original digraph.

Structure de la jointure entre les rôles: exemple

$$D = \{\text{'Ping Role'}, \text{'Pong Role'}\}$$

$$d : \begin{cases} d(\text{'Ping Role'}) = \{a_0, a_1\} \\ d(\text{'Pong Role'}) = \{a_2\} \end{cases}$$

La partie observable du processus de coordination



Grammaire des ressources : pré-requis

- Ensemble utilitaire:

Definition 4.2.1. Let R be a set; the market set R' associated to R is defined as follows:

$$R' = R \times \{\top, \bot\}$$

. R' is partitioned in two disjoints sets $R' = R^{\top} \cup R^{\bot}$, where:

$$\begin{aligned} R^{\bot} &= \{(r, \bot), r \in R\} \\ R^{\top} &= \{(r, \top), r \in R\} \end{aligned}$$

To simplify the notations, elements $(r, \top) \in R'$ are simply written as r and elements $(r, \bot)$ are represented as $\bar{r}$.
$|| : R' \to R$ is a function that removes the tag from a tagged element: $|(r, .)| \mapsto r$.

Grammaire des ressources : définition générale

Definition 4.2.2. Let R be the set of resources, R' its associated marked set and $\sigma \subseteq \mathcal{P}(R) \times \mathcal{P}(R)$ the set of simplification rules. The grammar of the resource patterns generated by σ is given by the rewriting relation $\xrightarrow{\sigma}: R'^* \times R'^*$ defined as:

$$U_0 r_1 U_1 ... r_i U_i r_{i+1} U_{i+1} ... r_n U_n \xrightarrow{\sigma} U_0 \bar{r_1} U_1 \bar{r_2} U_2 ... \bar{r_i} U_i U_{i+1} ... U_n$$

$$\Leftrightarrow$$

$$(\{|r_1|, .., |r_i|\}, \{|r_{i+1}|, .., |r_n|\}) \in \sigma \quad \wedge \quad \forall k \in [1..i], r_k \notin R^{\perp}$$

$(r_i)_{i \in [1..n]} \in R'$ are tagged resources and $(U_j)_{j \in [0..n]} \in R'^*$ are sequences, that can be empty, of marked resources.

$(\{x_p\}_{p \in [0..n]}, \{x_p\}_{p \in [n+1..m]}) \in \sigma$

L'interprétation de cette règle est que l'ensemble des ressources à gauche produisent l'ensemble des ressources à droite, ou l'ensemble des ressources à droite consomment les ressources à gauche.

Grammaire des ressources : construction de l'ensemble des règles à partir du graphe de coordination

- L'ensemble des règles de réduction (production) dérive directement de la structure du graphe de coordination comme suit:

$$G = (A, R, P, C)$$

$$\forall a \in A, (C(a), P(a)) \in \sigma_G$$

Grammaire des ressources : enfin, la définition finale…

Finally, the set of all resource patterns generated by a particular coordination model and a given axiom is defined as:

Definition 4.2.3. Let $G = (A, R, P, C)$ be the coordination model; resource patterns set $\text{r\_patterns}_{(G,x_0)}$ for a dependency-based coordination model G and the axiom $x_0 \in R^*$ is defined as follows:

$$\forall u \in R^*, u \in \text{r\_patterns}_{(G,x_0)} \iff \exists u' \in R'^* : u' \overset{\sigma_G}{\rightsquigarrow} x_0 \wedge |u'| = u$$

Grammaire des ressources observables: exemple

$$\sigma_{G_{\text{ping-pong}}} = \{(\{\text{'start'}\}, \{\text{'ping'}\}), (\{\text{'ping'}\}, \{\text{'pong'}\})\}$$

Pattern	$? \in \text{r\_patterns}_{(G_{\text{ping-pong}}, \text{start})}$	Reduction path
start.ping	yes	start.ping $\xrightarrow{\sigma}$ start
start.ping.pong	yes	start.ping.pong $\xrightarrow{\sigma}$ start.ping $\xrightarrow{\sigma}$ start
start.ping.pong.ping	yes	start.ping.pong.ping $\xrightarrow{\sigma}$ start.ping.pong $\xrightarrow{\sigma}$ start.ping $\xrightarrow{\sigma}$ start
start.pong	no	start.pong (cannot be simplified)
start.ping.ping	no	start.ping.ping $\xrightarrow{\sigma}$ start.ping (cannot be simplified)

Grammaire des ressources concrètes observables:

- Distinction entre les ressources abstraites définies dans le graphe de coordination et les ressources concrètes qui circulent entre les entités
- Le lien se fait par des fonctions de reconnaissance et un mapping des ressources abstraites

Definition 4.3.1. Let $G = (A, R, P, C)$ be a coordination model and $\mathcal{O}$ the set of actual resources. A resource recognition mapping κ of the coordination model G is a function from the set of resources to the set of control functions or a null value:

$$\kappa : R \to (\mathcal{O} \to \{\top, \bot\}) \cup \{0\}$$

Grammaire des ressources concrètes observables:

The recognition of an actual resources sequence $m = m_0 m_1 ... m_n \in \mathcal{O}^*$ according to a resource node pattern $p = r_0 r_1 ... r_m \in R^*$ and a resource recognition mapping κ, is performed as follows:

1. each resource node in the resource pattern p is replaced by its associated recognition function defined in κ. This produces a sequence of recognition functions $f = f_0 f_1 ... f_j \in ((\mathcal{O} \rightarrow \{\top, \bot\}) \cup \{0\})^*$.

2. this sequence of function is normalized by removing all zero (or null) elements; the normalized sequence is noted $f' = ||f||$.

3. we assume that the length of f' is greater than the length of the actual resources sequence m; if not the resource pattern is considered as too small to recognize the actual resources sequence.
4. the recognition function sequence 'is applied' to the sequence of actual resources m. This is done by replacing each i-th element of m (m_i) by the application of i-th function of f' to m_i $(f'_i(m_i))$. So, the result of this operation is a sequence of Boolean values.

5. if the resulting sequence contains only $\top$ values, this means that the actual resources sequence has been *validated* by the resource pattern and the recognition mapping κ; otherwise, it is invalid.

Grammaire des ressources concrètes observables: exemple

Pattern = *start.ping.pong.ping.pong*

$$\mathcal{O} = \{0, 1\}$$
$$s = 0101$$

$$\kappa : \begin{cases} \kappa(\text{start}) & = 0 \\ \kappa(\text{ping}) & = \lambda(x) : (x = 0) \\ \kappa(\text{pong}) & = \lambda(x) : (x = 1) \end{cases}$$

1. each element of r is replaced by its associated recognition function:

$$\begin{aligned} f &= \kappa(start).\kappa(ping).\kappa(pong).\kappa(ping).\kappa(pong) \\ &= 0.\lambda(x) : (x = 0).\lambda(x) : (x = 1).\lambda(x) : (x = 0).\lambda(x) : (x = 1) \end{aligned}$$

2. normalization of the recognition functions sequence: this is done by removing all zero elements from the sequence of functions:

$$||f|| = \lambda(x) : (x = 0).\lambda(x) : (x = 1).\lambda(x) : (x = 0).\lambda(x) : (x = 1)$$

3. apply the sequence of recognition functions to the sequence of actual objects:

$$\begin{aligned} & \lambda(x) : x = 0)(0).\lambda(x) : (x = 1)(1).\lambda(x) : (x = 0)(0).\lambda(x) : (x = 1)(1) \\ = \; & (0 = 0).(1 = 1).(0 = 0).(1 = 1) \\ = \; & \top.\top.\top.\top \end{aligned}$$

4. we can conclude that the sequence s have been validated by the pattern r. Furthermore, since the pattern r belongs to $\text{r\_patterns}_{(G_{\text{ping-pong}}, \text{start})}$, then we can conclude that sequence of actual resources s belongs to the conversation protocol defined by this coordination model.

Grammaire des ressources concrètes observables: définition finale..

Definition 4.3.2. Let $\mathcal{O}$ be the set of interaction objects; $G = (A, R, P, C)$ a coordination model associated to the resource recognition mapping κ and a roles-cut d such as the mapping κ satisfies the constraints of the roles-cut d. The conversation protocol generated by G and the axiom $x_0 \in R^*$ is defined as the set of all sequences of interaction objects that are validated by the resource patterns of G according to the axiom x_0:

$$\forall c \in \mathcal{O}^*, c \in \text{conversations}_{(G,\kappa,d,x_0)} \iff \exists p \in \text{r\_patterns}_{(G,x_0)} : \text{validate}(p, c, \kappa)$$

Comment reconnaître des séquences de ressources concrètes : Introduction informelle des réseaux de Petri à pile

- On utilise une extension des réseaux de Petri, les réseaux de Petri à pile Queue: Petri Nets (QPN)

- Structure d'un QPN :
  - Une pile
  - Jetons
  - Transitions classiques
  - Transitions qui consomment des objets de la pile

Définition formelle des QPN

4.4.4 Queue Petri-Net structure:

Definition 4.4.1. Let $\mathcal{O}$ be the set of interaction objects; $\mathcal{O}^*$ the set of ordered sequences of interaction objects; the Queue Petri net structure is a six tuple $C = \{P, T, I, O, F, q\}$, $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places $n \geq 0$ and $T = \{t_1, t_2, ..., t_m\}$ is a finite set of transitions $m \geq 0$. $P \cap T = \emptyset$. $I : T \rightarrow P^\infty$ is defined as the input function, a mapping from transitions to bags of places and $O : T \rightarrow P^\infty$ is defined as the output function, a mapping from transitions to bags of places; $F : T \rightarrow \{0\} \cup (\mathcal{O} \rightarrow \mathbb{B})$, is defined as the message control function, a mapping from transitions to control functions (defined between message sequences and Boolean values) or zero; $q \in \mathcal{O}^*$ is defined as the message queue: an ordered sequence of elements belonging to $\mathcal{O}$.

Règles d'exécution des QPNs

Definition 4.4.2. Transition $t_j \in T$ in a marked Queue Petri net $C = (P, T, I, O, F, q)$ with marking μ is enabled iff:

1. for all $p_i \in P$:
$$\mu(p_i) \geq \#(p_i, I(t_j))$$
2. if $F(t_j) \neq 0$ then:
$$F(t_j)(front(q)) = \top$$

Definition 4.4.3. Let t_j be a fireable transition in a marked Queue Petri $C = (P, T, I, O, F, q)$ net with marking μ. The firing of this transition results in a new marking μ' defined as follows:

$$\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j))$$

and a new message queue q' defined as follows:

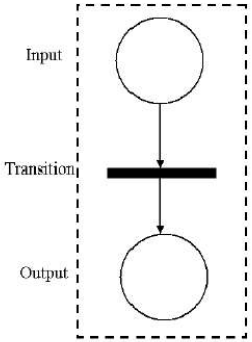
$$q' = \begin{cases} pop(q) & \text{, if } F(t_j) \neq 0 \\ q & \text{, if } F(t_j) = 0 \end{cases}$$

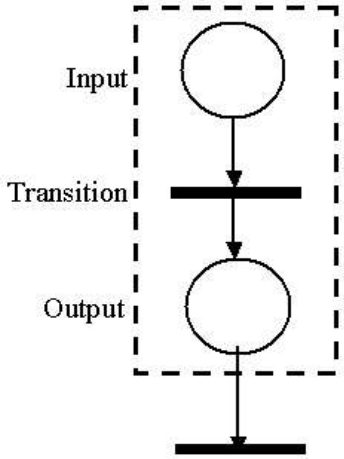
When there are no fireable transitions the QPN is said to be *dead.*

Definition 4.4.4. *The Queue Petri Net $C = (P, T, I, O, F, q)$ with a marking μ is said to have recognised its sequence message queue q if there is an execution path that leads to $C \rightsquigarrow C' = (P, T, I, O, F, q')$ with a marking μ' and where q' is an empty queue.*

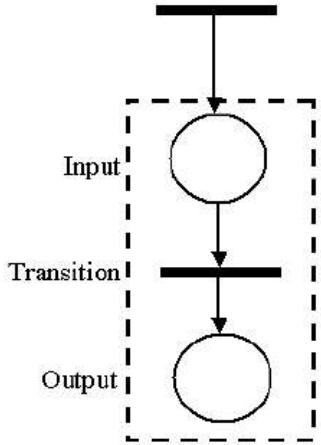
Construction du QPN à partir du graphe de coordination: pré-requis

- Structure de bloc dans un réseau de Petri:





Relation de production



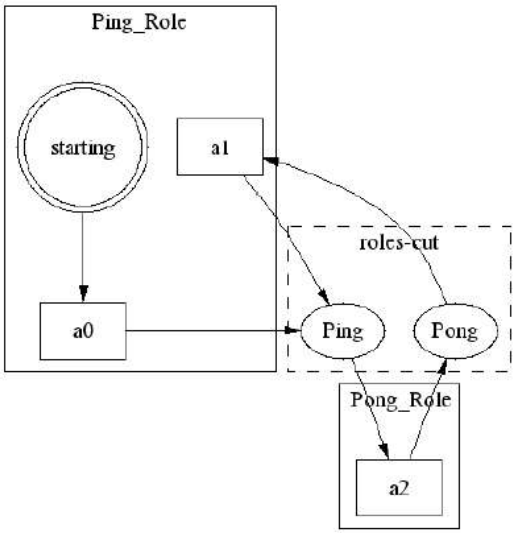
Relation de consommation

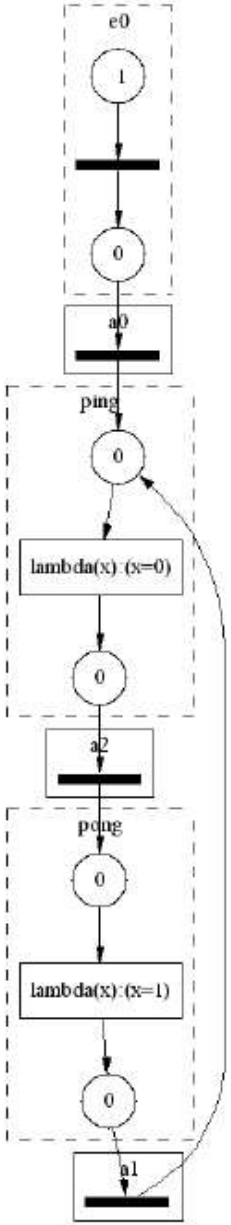
Construction du QPN à partir du graphe de coordination:algorithme de génération

Algorithm 1 Building the structure of the QPN from the coordination model

1: **function** FROMCOORDINATIONMODEL2QPN(cm,x_0,d,κ) ▷
   cm represents the coordination model structure, x_0 the axiom of the coordination model, d is the set of resources found in the roles-cut and κ represents a given recognition functions mapping.
Require: satisfies_roles-cut_constraints(κ,d)
2: $resource2bloc \leftarrow$ new_map() ▷ This is a mapping that links each resource node to its bloc.
3: $qpn \leftarrow$ new_QPN() ▷ This is the QPN that is returned by this function.
4: **for all** $r \in cm$.getResourceNodes() **do**
5: $b \leftarrow$ new_bloc(r)
6: **if** $r \in x_0$ **then**
7: b.getInputPlace().setMarking(1)
8: **else**
9: b.getInputPlace().setMarking(0)
10: **end if**
11: **if** $r \in d$ **then**
12: b.getTransition().setRecoFunction($\kappa(r)$)
13: **else**
14: b.getInputPlace().setRecoFunction(0)
15: **end if**
16: $resource2bloc[r] \leftarrow b$
17: qpn.addBloc(b)
18: **end for**
19: **for all** $a \in cm$.getActivityNodes() **do**
20: $t \leftarrow$ new_transition(a)
21: qpn.addSimpleTransition(t)
22: **for all** $r \in cm$.getProducedResources(a) **do** qpn.makeBlocProductionLink($resource2bloc[r]$,t)
23: **end for**
24: **for all** $r \in cm$.getConsumedResources(a) **do** qpn.makeBlocConsumptionLink($resource2bloc[r]$,t)
25: **end for**
26: **end for**
27: **return** qpn
28: **end function**

Construction du QPN à partir du graphe de coordination: exemple





Validation des conversations des agents par l'observation des objets d interaction.

- On utilise le QPN généré pour un protocole coordination
- Il faut attention aux conflits:
  - Conflit sur les jetons (alternative dans les réseaux de Petri classiques)
  - Conflit sur la pile quand plusieurs fonctions de reconnaissance reconnaissent le premier message de la pile
- Solution: duplication du QPN
- Une conversation est valide, si au moins un QPN vide sa pile
- Une conversation est invalide si tout les QPN sont morts
- Une conversation invalide => Protocole de coordination non respecté
- Limitation: le QPN généré ne doit pas comporter de cycle libre: cycle ne contenant aucune transition qui consomme un message de la pile

Partie III:
Intégration du modèle de coordination et MIC*

Encore une 1 heure ?…

Conclusion

- Environnement de déploiement et son importance pour l ingénierie des SMA
- Unification des protocoles d'interaction et protocoles de coordination:
  - Dans les SMA, un protocole d'interaction (conversation) n'est que la partie visible d'un processus de coordination
- Validation des protocoles de coordination par observation des messages échangés sans contredire l'autonomie des agents
- L'intégration des deux parties se fait par cadre d'ingénierie social
  - Modèle social : basé sur le modèle AGR + Protocole de coordination
  - MIC* représente dans ce contexte l'environnement social qui implémente et garantisse les normes établies sur la coordination.

Réalisations

- MIC*, projet sur sourceforge, http://mic.sourceforge.net
- Coordination framework, gamme d'outils de développement en C++
  - Formalisme XML
  - Outil de génération automatique du ProtocolController en C++