Le développement de protocoles de communication en mode message asynchrone

G. Florin

INTRODUCTION

Le contrôle réparti

Ensemble des mécanismes offerts à un programmeur pour développer des applications réparties

Utilisation de l'interface d'un système réparti.

IPA "Interface de programmation d'applications"

API "Application Programming Interface"

Problèmes posés:

-La concurrence, la synchronisation des activités locales

-Les communications entre sites

(les interactions)

Différents styles d'interactions

Schéma de complexité croissante:

Le mode message asynchrone

Le mode rendez-vous

Le mode appel de procédure distante

Le mode mémoire virtuelle répartie

Les protocoles coopératifs

...

Construction d'API unifiant l'approche objet et les interactions:

Les systèmes d'objets répartis

Première partie

Le mode message asynchrone

Rappel de quelques points essentiels

Généralités

- Mode de base de la communication.

- Offert par le plus grand nombre des IPA réparties

- Le service comprend deux primitives principales pour communiquer et se synchroniser (couches transport).

TYPE COM_MESSAGE_ASYNCHRONE;

METHOD envoyer (id_émetteur, id_récepteur, compléments, message);
METHOD recevoir (id_émetteur, id_récepteur, compléments, message);
METHOD ...;

END COM_MESSAGE_ASYNCHRONE.

identificateurs : ports
compléments: qualités de services (selon des sémantiques multiples)
messages: zones de données.

Exemples d'IPA distribuées en mode message asynchrone

Presque toutes les interfaces des architectures "ouvertes" de réseaux

(pour presque tous les niveaux)

- **Internet, OSI, SNA**

Presque toutes les interfaces des systèmes répartis classiques,

- **Chorus , Mach, Amoeba**

De nombreux langages de programmations

- **Langages de spécifications de protocoles** (Estelle, ...)

- **Langages "acteurs" de l'intelligence artificielle distribuée** (Act1, ...)

Exceptions : les produits orientés RPC, objet, partage de mémoire.

Variantes sémantiques du mode message asynchrone

- **Propriétés de synchronisation**

La synchronisation locale

La synchronisation inter-sites.

- **Nature des entités** communicantes

point à point, diffusion

mode alternat, bidirectionnel

- **Propriétés d'ordre**.

local, global, causal

- **Propriétés de tolérance aux pannes**.

messages, sites

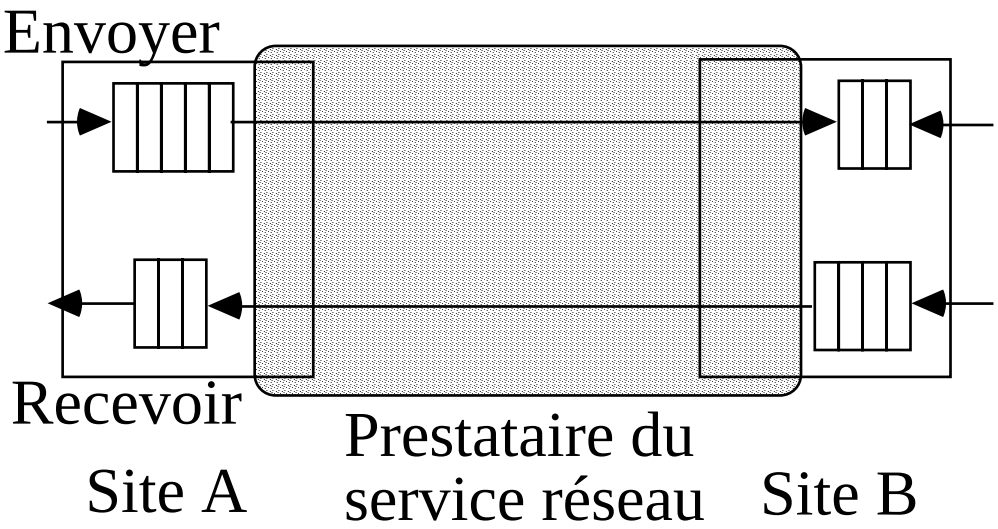
- **Propriétés temporelles**

débits, délai, variation de délai

"Cours de réseaux habituels"

Les propriétés de synchronisation

Le mode message asynchrone réalise un **"producteur-consommateur" réparti** entre un émetteur et un récepteur.



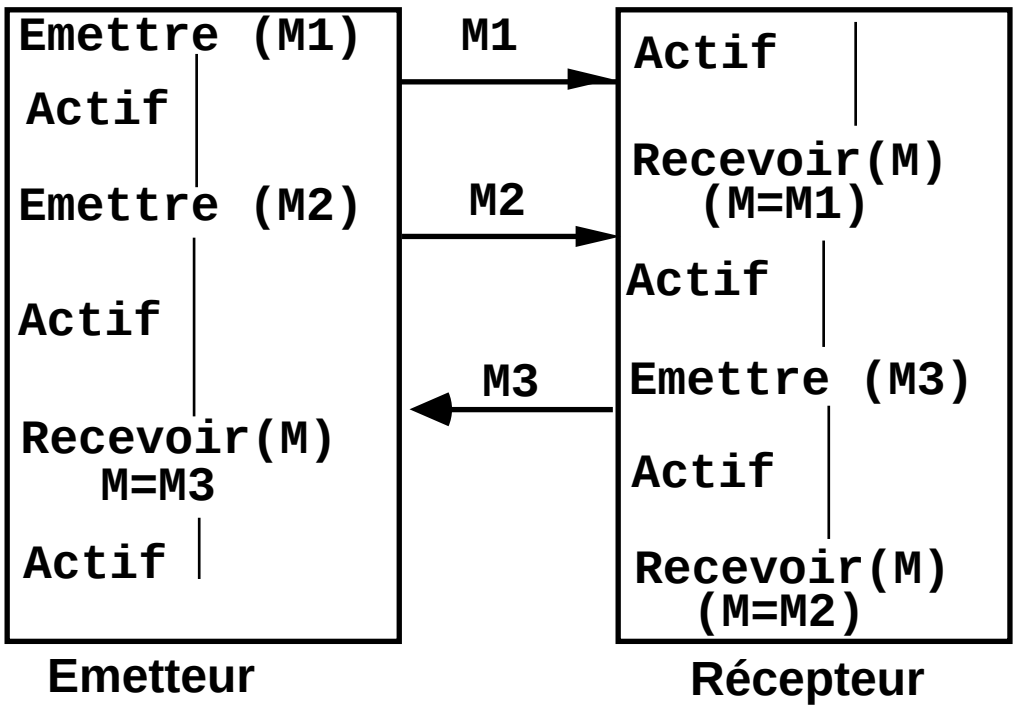
Asynchronisme entre sites distants.

- propagation sur le réseau

Synchronisation locale sur les tampons d'émission et de réception.

A) La synchronisation inter-sites

Aspect principal: l'asynchronisme entre l'émetteur et le récepteur



Détails du comportement

L'émetteur

. Ayant demandé une émission, **reprend la main et continue son exécution *"immédiatement"* après** la prise en compte de sa demande.

. Le message est transmis (au rythme du transport d'informations par le réseau de communication) donc de façon **asynchrone** avec le comportement émetteur**.**

Le récepteur

Ayant décidé de prendre en compte un nouveau message il acquiert un message (le premier) en instance.

-Celui qui se présente après le recevoir

-Un message qui se trouve dans une file d'attente de réception.

B) La synchronisation locale

Synchronisation sur l'interface de programmation d'application.

Primitives envoyer "bloquantes" "non bloquantes"

Le processus émetteur a fourni l'adresse d'un tampon (contenant un message) partagé avec le prestataire.

. Cas 1 :

L'émetteur reste bloqué tant que le message n'a pas été envoyé => le tampon est redevenu libre.

. Cas 2 :

L'émetteur reste bloqué tant que le message n'a pas été recopié dans une file d'attente du prestataire.

. Cas 3 :

L'émetteur reprend la main alors que le prestataire utilise le tampon. Il est averti par un signal de la fin d'utilisation => il peut le réutiliser.

Primitives recevoir "bloquantes" "non bloquantes"

Le processus récepteur a fourni l'adresse d'un tampon au prestataire

. Cas 1 : Recevoir bloquant

Le récepteur reste bloqué tant qu'un message reçu n'a pas été écrit.

. Cas 2 : Recevoir non bloquant + attente

Le récepteur continue son exécution et se bloque quand il ne peut plus ne pas disposer d'un message reçu.

. Cas 3 : Recevoir non bloquant + primitive de test de message

Le récepteur continue mais il peut savoir si un message reçu existe.

. Cas 4 : Réception conditionnelle

Le récepteur reprend toujours la main avec un message reçu ou avec un diagnostic.

L'utilisation du mode message asynchrone

En général deux aspects simultanément réalisés:

Invocation d'action distante

La nature de l'action est définie par le typage du message.

La **nature de l'action déclenchée** dépend du contexte dans lequel le message est pris en compte.

(Idée de contexte ou d'état)

("Acte de langage")

Transport d'informations dans la partie données du message

Action "d'information"

Paramètrage d'exécution distante.

Sémantique de l'activation

L'échange en mode message asynchrone permet de définir des structures de contrôle classiques en univers réparti

a) **Activation en parallèle ("fork")**

D'une activité à distance puisque en mode normal l'émetteur et le récepteur continuent leur exécution après l'interaction envoyer recevoir.

b) **Branchement inconditionnel ("goto")**

Une activité à distance s'exécute après l'activité locale si l'émetteur se suspend définitivement (wait) après l'émission.

Sémantique d'échange de données

- Le mode message permet une opération d'affectation à distance d'une variable M (de type article):

émettre (M) = écrire (M) (vers dest)

recevoir (M) = lire (M) (de emet)

. Avec possibilité de **vérification de cohérence des types** des variables (si type(M) acheminé).

. Avec **une sémantique de consistance des données entre l'émetteur et le récepteur très faible.**

Reposant sur les propriétés d'ordre, temporelles, de tolérance aux pannes spécifiées par la qualité de service.

Propriété minimale de cohérence des échanges de données par message asynchrone: la causalité

émettre/écrire (M)->recevoir/lire(M)

∃ t1 : émetteur.écrire (M)

=> ∃ t2>t1 : recepteur.lire (M)

Les messages ne remontent pas le temps

Propriétés supplémentaires de cohérence (propriétés temporelles et propriétés d'ordre).

mode rendez-vous,

mode appel de procédure

mode mémoire répartie.

Seconde partie

Spécification des applications utilisant le mode message asynchrone

Objectifs

Définir des méthodes formelles de spécification des applications réseaux en mode message asynchrone.

(FDT "Formal Description Techniques")

=> Processus **séquentiels**

=> Communicants par messages

Comportement séquentiel des processus

=> Les méthodes de spécification utilisent des **automates d'état fini.**

=> Chaque entité communicante est représentée par son automate d'état.

Interactions communication par messages

=> Les automates sont synchronisés pour la représentation des échanges de messages.

Représentation des applications: automates

Une modélisation état/transition.

États

Chaque état doit être **aisément interprétable** en terme d'évolution locale de l'application répartie.

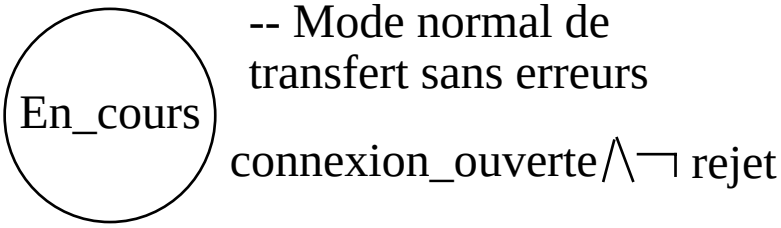
- Il représente un point d'avancement du contrôle.

- Il est défini par **un ensemble significatif de variables locales** de chaque processus. Il peut recevoir:

- un identifiant bien choisi

- un commentaire

- le prédicat sur les variables



Identification des états

Étape préliminaire importante de la spécification:

. Pas un niveau de détail trop fin (illisible, trop grand nombre d'états).

. Pas un niveau de détail trop gros
Tout est modélisé dans les transitions.

Représentation graphique de l'évolution au niveau permettant l'observation de la synchronisation.

Transitions.

Elles comportent deux mentions:

la condition déclenchante ,
l'action à réaliser.

Elles sont représentées par les arcs du graphe associé à l'automate conduisant d'un état initial à un état final.

Conditions ou gardes

Les transitions sont franchissables lorsqu'une condition booléenne ou garde est satisfaite:

. condition booléenne portant sur **les variables locales**.

. condition booléenne portant sur les **messages entrants**.

. condition booléenne portant sur des d'événements internes (**horloges**)

Action

- C'est l'opération réalisée lors du franchissement de la transition.

les traitements à réaliser lors du passage à l'état suivant

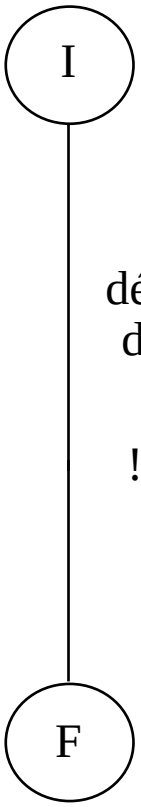
affectation de variables,

envoi de messages.

- Le détail des actions ne doit pas être trop important sinon le modèle est illisible

Sinon renvoyer à des textes annexés à la spécification.

Représentation des transitions.



Prédicat définissant l'état

Etat de contrôle antécédent
Etat initial

?condition: prédicat définissant les conditions de déclenchement de la transition

Partie condition ou garde

! action:comportement associé à la transition

Partie action de la transition

Prédicat définisant les connaissances après la transition

Etat de contrôle conséquent
Progression de l'algorithme

Les commandes d'échanges

- Notation des opérations envoyer, recevoir

<émission> ::= <dest> **!** <message>

<réception> ::= <source> **?** <message>

Désignation

La commande d'émission doit désigner un destinataire (P2).

Ex : Dans le processus P1 :P2!M1

La commande de réception (exécutée par un processus P2) doit évoquer une source P1.

Ex : Dans le processus P2 :P1?M1

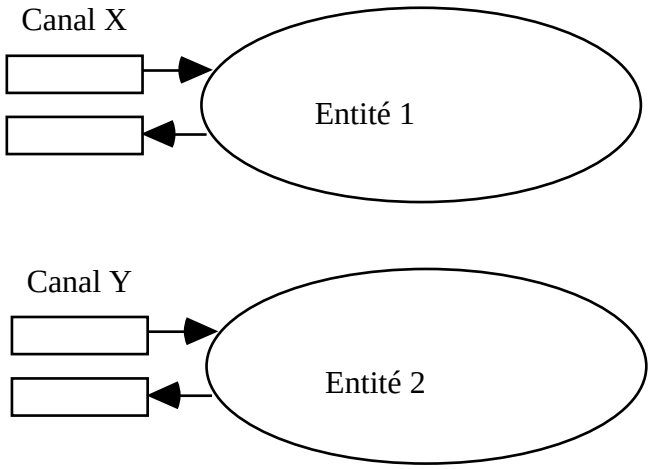
Remarque: Dans le cas d'une communication point à point absence d'ambiguïté => pas de désignation.

Utilisation de canaux

- Très souvent la désignation se fait au moyen de canaux (terminologie des langages de description de protocoles) qui sont ensuite connectés pour réaliser une architecture.

Ex : Émission sur canalX : canalX!M

Réception sur canalY : canalY?M



La définition d'architecture consiste à spécifier ensuite une liaison entre canalX et canalY.

Typage

Il doit y avoir correspondance de type entre le message reçu (la variable de réception) et le message émis (la valeur émise).

Données d'un message =

variable de type article.

Type[message_émis]=

Type [message_reçu]

Si le type d'un message reçu n'est pas prévu à la réception:

Erreur: "réception non spécifiée".

Sémantique de l'alternative constituée par l'existence de plusieurs transitions en un état

- Évaluation des gardes

. Une garde est **franchissable** si elle est constituée par **une expression booléenne** portant sur des variables locales à **valeur vrai**.

. Une garde est **franchissable** s'il s'agit d'une **commande d'échange** satisfaite : en fait une réception dont le processus source à fait parvenir un message du type attendu qui se trouve en tête de la file d'attente.

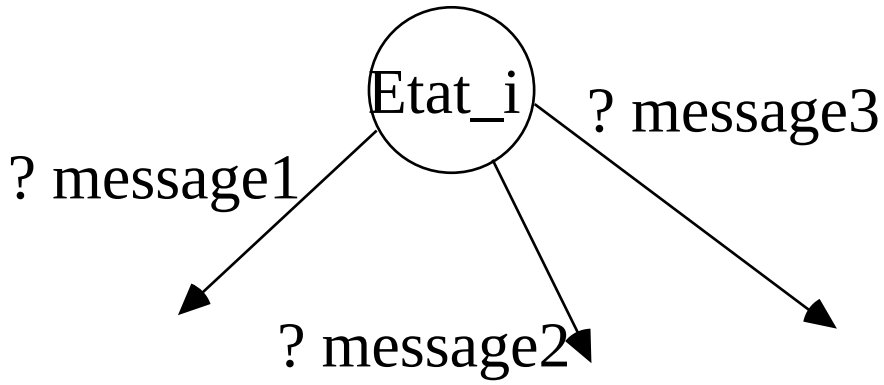
. Une garde **combinaison des deux cas précédents.**

Alternative en un état

Indéterminisme de comportement

Indéterminisme aspect fondamental du réseau.

- en un même état plusieurs messages (ou événements) peuvent arriver.
- l'un des messages est placé en tête de la file de réception.
- ce sont les aléas de fonctionnement des matériels et des logiciels, les choix effectués qui conduisent souvent l'interclassement dans la file d'entrée.



Alternative en un état

Aspect séquentiel des automates

- On choisit **l'une des transitions** ayant sa condition booléenne satisfaite (sa garde ouverte).

Indéterminisme de l'évaluation

- Si plusieurs conditions sont simultanément vérifiées l'une **quelconque des** transitions peut-être sélectionnée

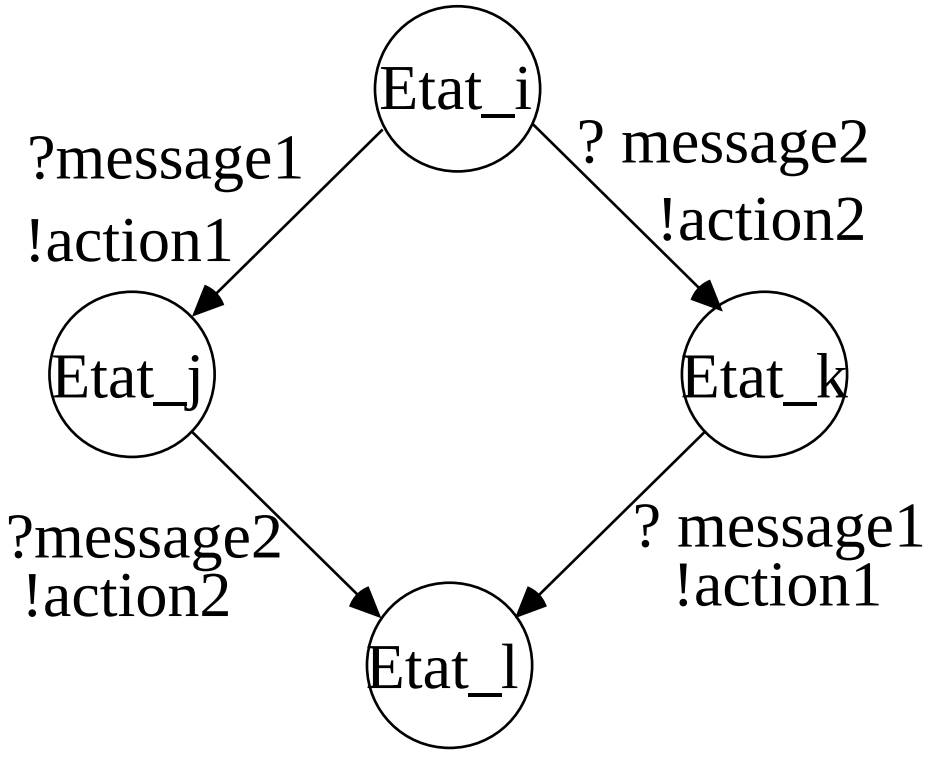
- **La liste des commandes** associée à la garde est **exécutée**.

La modélisation doit tenir compte de cette caractéristique majeure.

Schémas persistants

Si deux messages peuvent être reçu en un état et que leur réception ne modifie pas les conditions de fonctionnement ultérieures

=> leur réception doit continuer à être prévue.



Alternative en un état

Sémantique en cas d'attente

- Aucune garde n'est franchissable mais il existe des gardes avec condition de réception pouvant être satisfaite par une émission d'un site distant.

=> **Attente que l'une d'elles soit satisfaite.**

- Aucune garde n'est franchissable et elles sont toutes booléennes.

=> Fin du programme
Probable **erreur d'exécution.**

- Aucune garde ne pourra plus jamais être franchie (certaines sont des attentes de message).

=> Fin du programme
Erreur: interblocage de message.

Différents automates

Le service

Définit les échanges entre un prestataire (fournisseur) et un utilisateur d'un service.

. Établissement de **la liste des unités de service** (primitives, SDU) échangés entre le prestataire et l'utilisateur (niveau n et niveau n+1).

. **Définition des enchaînements autorisés** de primitives sous la forme d'un automate d'état.

=> Toutes les successions légales qu'un observateur de l'interface n n+1 peut observer.

=> Deux points de vue duaux:

- le prestataire du service
- l'utilisateur du service

Exemple de comportements autorisés dans une entité de liaison: éléments de connexion.

Éléments du service de connexion avec accord confirmé

Connect.request

Connect.indication

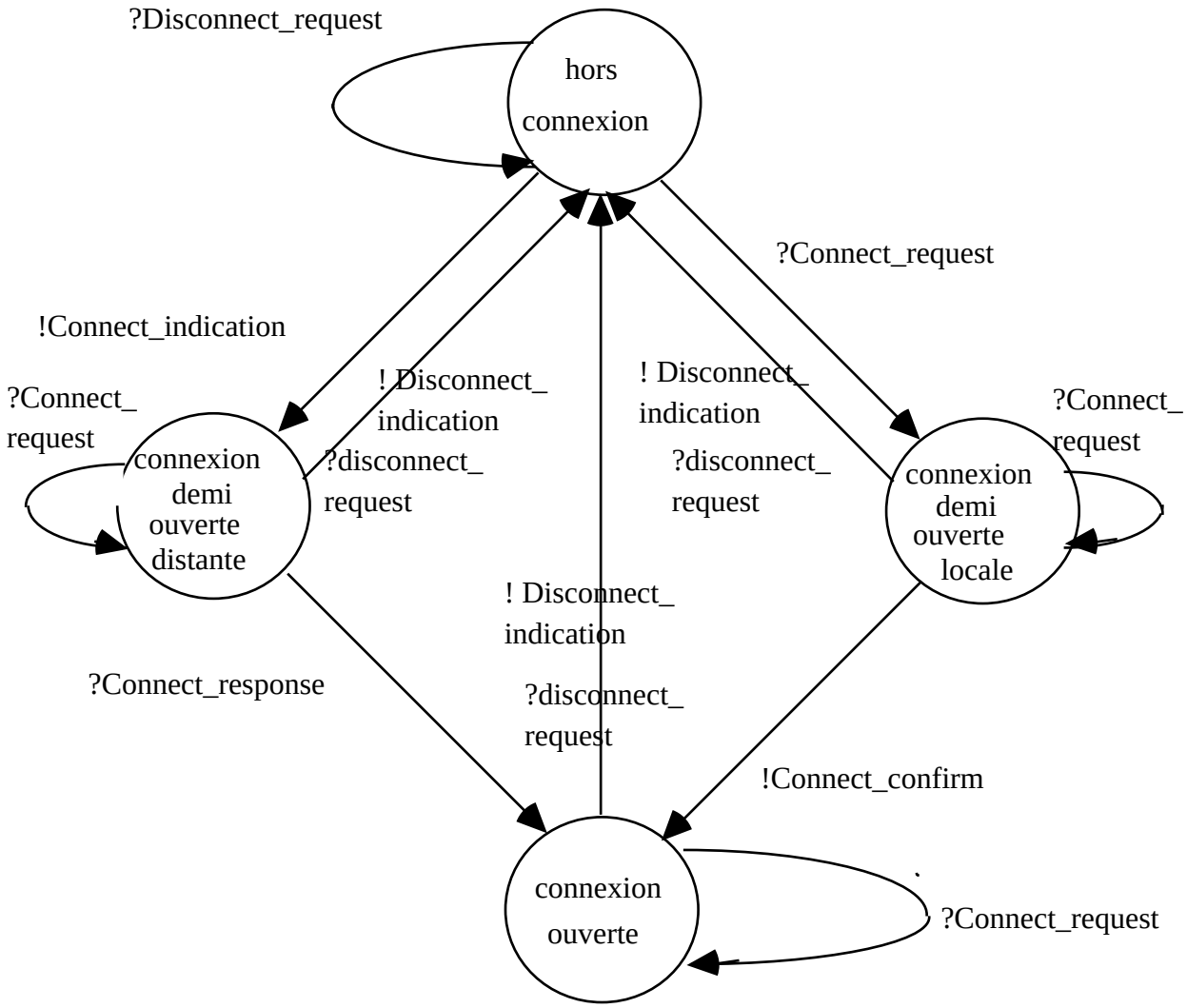
Connect.response (cas d'accord)

ou Disconnect.request (cas de rejet)

Connect.confirmation (accord)

ou Disconnect.indication (rejet)

Automate du service



Le protocole

Définit les échanges entre deux prestataires de même niveau.

. Établissement de la liste des unités de protocole (messages, PDU) échangés entre deux niveaux n.

. Définition des suites d'échanges qu'un observateur de la voie de communication entre les deux niveaux n peut observer.

=> Toutes les successions légales d'éléments de protocole observables sur la voie logique de communication n <--> n.

Éléments du protocole de liaison en phase de connexion

Demande d'ouverture

SABM

"Set Asynchronous Balanced Mode"

Demande de fermeture

DISC

"Disconnect"

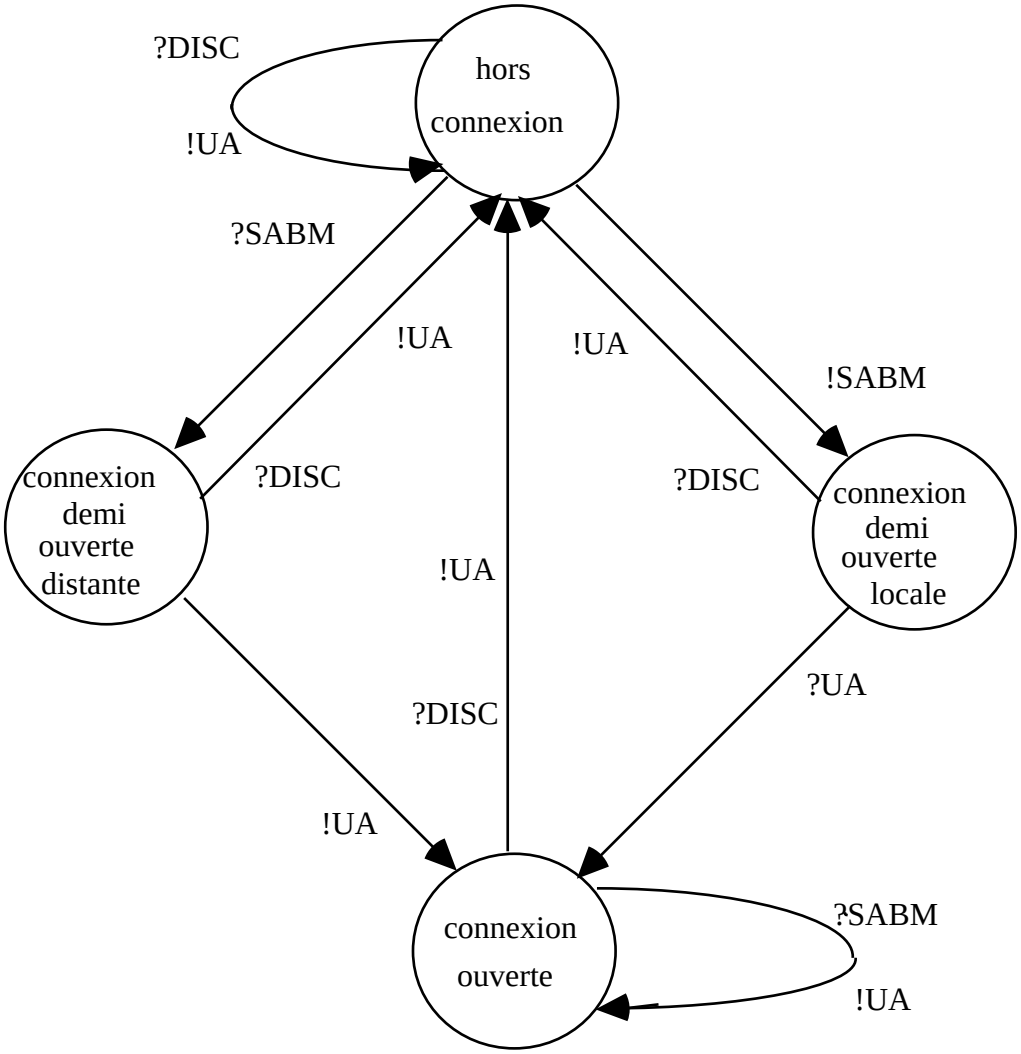
Accusé de réception non numéroté

UA

"Unnumbered Acknowledge"

Automate du protocole de connexion

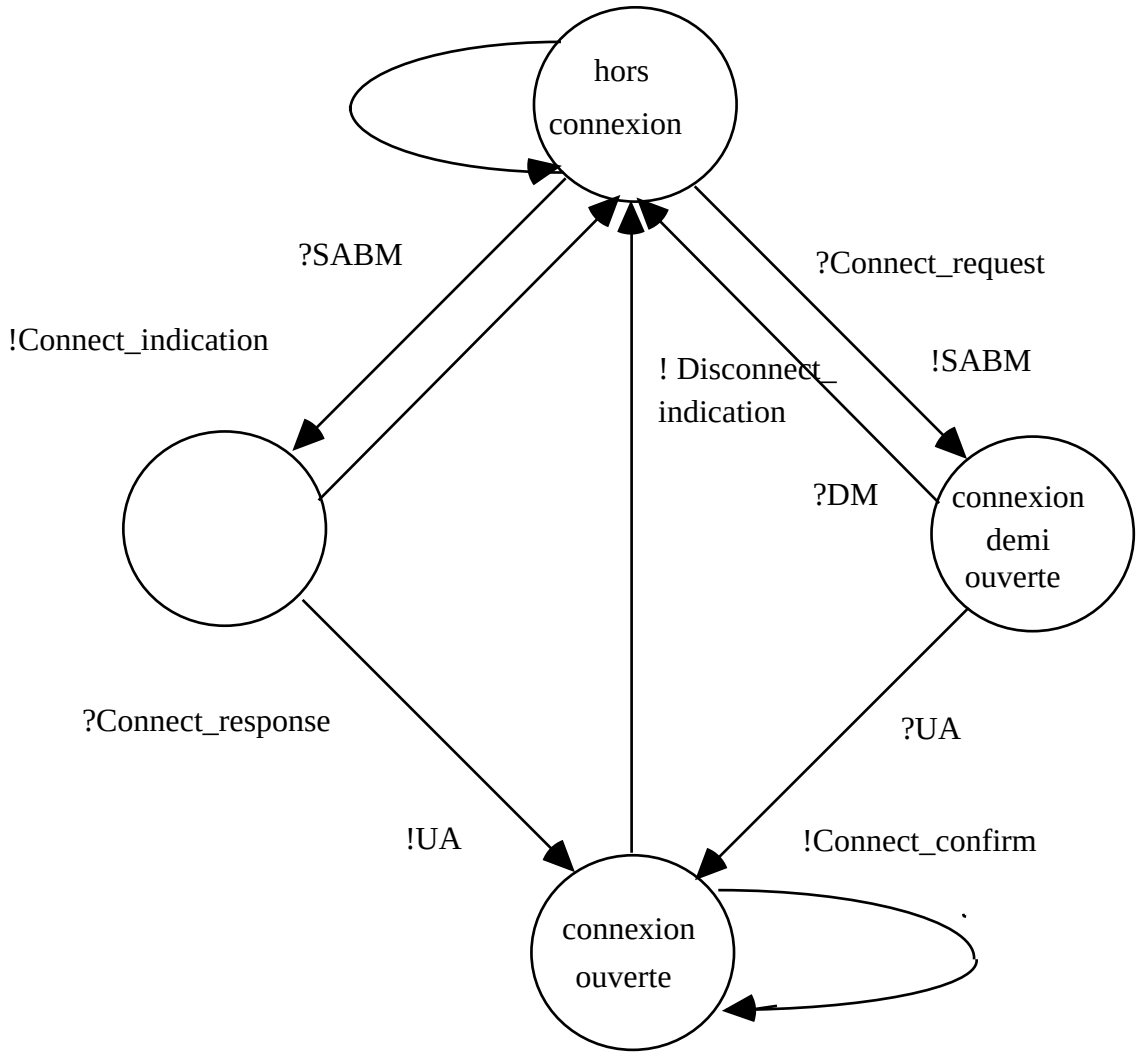
Vision partielle



Automate complet

. Réunion des deux automates de service et de protocole

Vision partielle

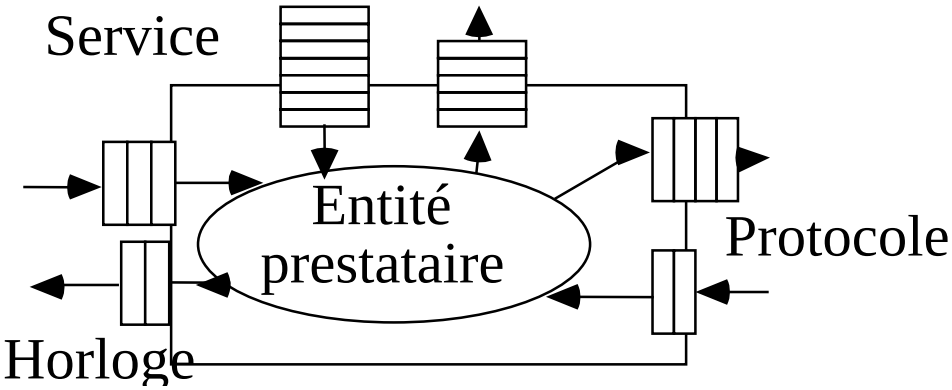


Gestion des files d'interactions

Événements systèmes locaux: retombée signaux d'horloge

Événements distants :
arrivée d'éléments protocolaires

Événements requêtes de service:
arrivée d'éléments de service



Validation des spécifications

Problèmes de constructions

Réceptions non spécifiées

Dans un état un message peut se présenter dont le cas n'a pas été prévu

Interblocages

Dans un état un message (ou une configuration) est attendu qui ne peut jamais se présenter.

Plus généralement

Définition d'assertions de bon fonctionnement sur le modèle à automates communicants.

- assertions portant sur des variables d'état (booléennes)

- assertions portant sur des trajectoires (logique temporelle)

Méthodes de validation systématiques employées

Simulation comportementale

Exécution en **simulation** du comportement décrit par les automates: une partie seulement des comportements sont couverts.

Détection d'erreurs et correction (technique s'apparentant au test du modèle).

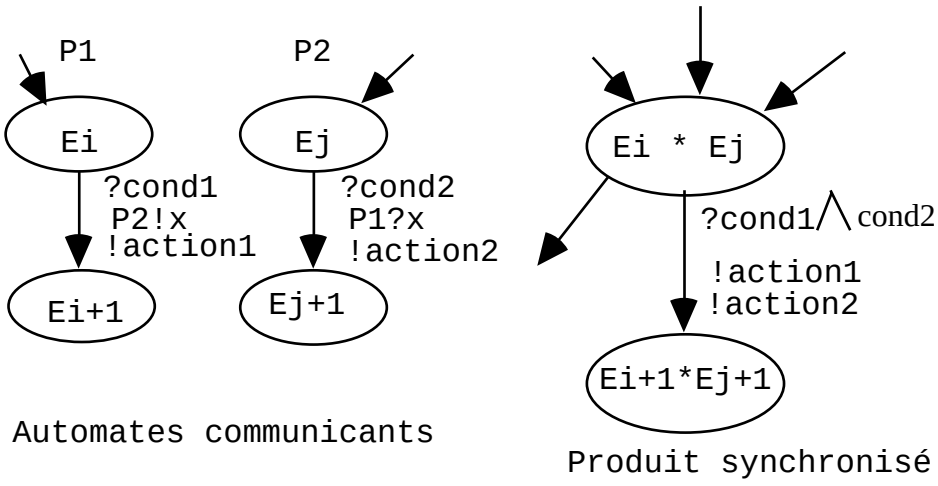
Difficulté majeure : On ne connaît pas la couverture du test (le taux d'erreurs résiduelles après le test).

Exemple d'outil: Estelle- VEDA

Analyse exhaustive.

- L'application est définie par un ensemble d'automates communicants

- Réalisation du produit des automates (produit "synchronisé" tenant compte des communications).



- Obtention du graphe d'accessibilité du système complet.

- Vérification d'assertions sur le graphe complet.

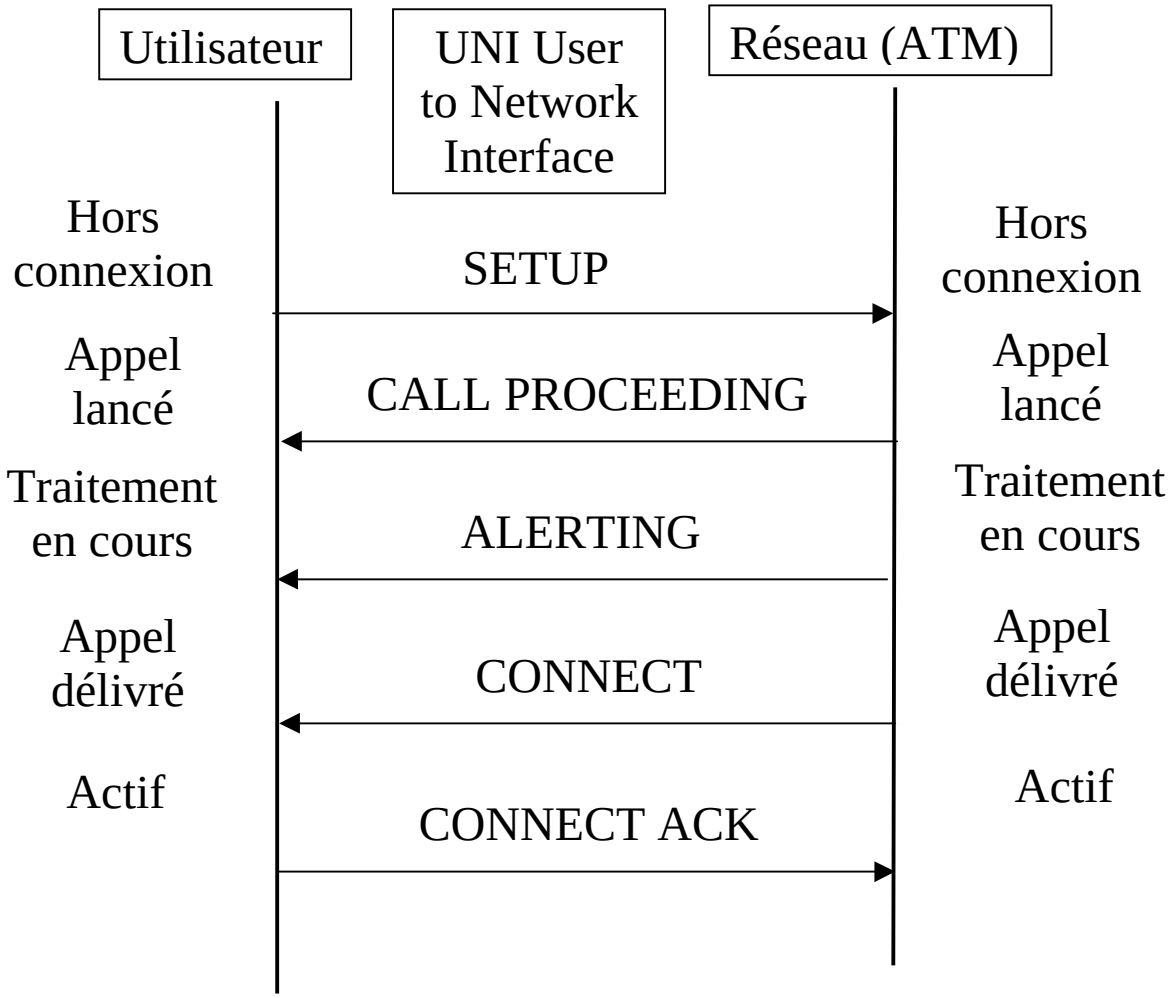
Difficulté majeure

Explosion combinatoire de la taille des espaces d'état.

Normes dans le domaine des réseaux

MSC ' Message State Charts ' ITU-T Z120

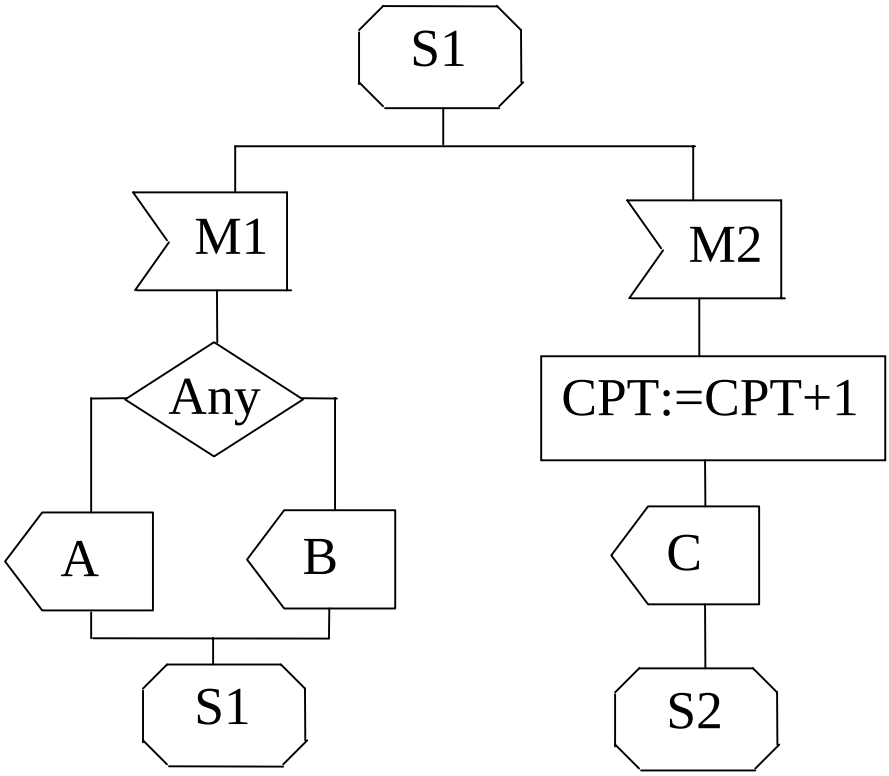
- Scénario d'échange de messages pour décrire des protocoles réseaux (ordre des messages, utilisation des temporisateurs …)



SDL ' Specification and Description Language ' ITU-T Z100

Notation graphique (et aussi langage) pour décrire des entités réseaux sous la forme d'automates synchronisés.

Un_processus_SDL



Commentaire: Recevoir M1 ou M2, si M1 envoyer A ou B de façon indéterministe. Si M2 incrémenter un compteur et envoyer C.

TTCN ' Testing and Test Control Notation ' ITU-T Z140

Une notation graphique langage pour décrire des séquences de test d'une entité réseau.

Cas_test_1

!M2

?C	pass
?OTHERWISE	fail

Commentaire : Pour tester l'entité SDL précédente, émettre M2, la réception de C est alors correcte (le test continue), toute autre réception est une erreur.

Normes dans le domaine UML 'Unified Modelling Language'

MSC ' Message Sequence Charts '

- Même forme que les 'Message State Charts'.

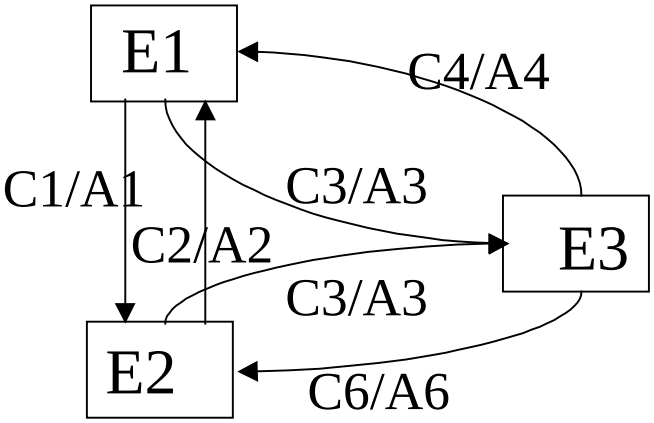
- Utilisation très fréquente pour décrire les interfaces de service sous forme de cas d'utilisations ('use cases')

UML : Harel Statecharts

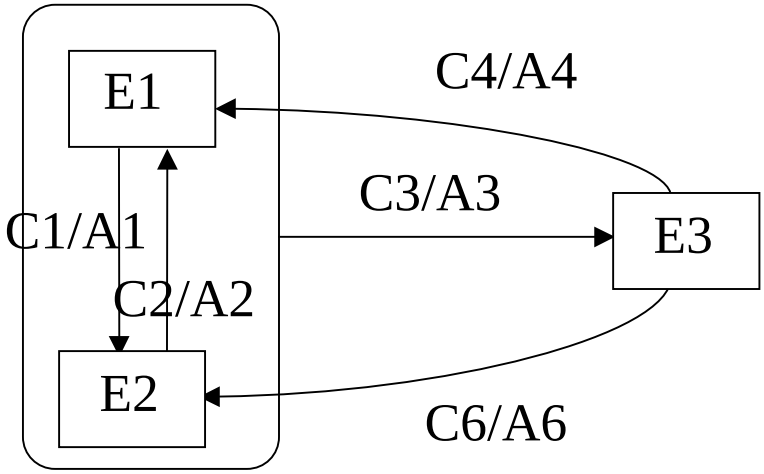
Une modélisation état/transition d'un système avec beaucoup de possibilités (complexité):

. regroupement d'état:modélisation hiérarchique avec gestion des événements externes.

. concurrence (opérateurs, fork, merge)



Un modèle état transition (conditions/actions)



Le même modèle en statecharts de Harel.

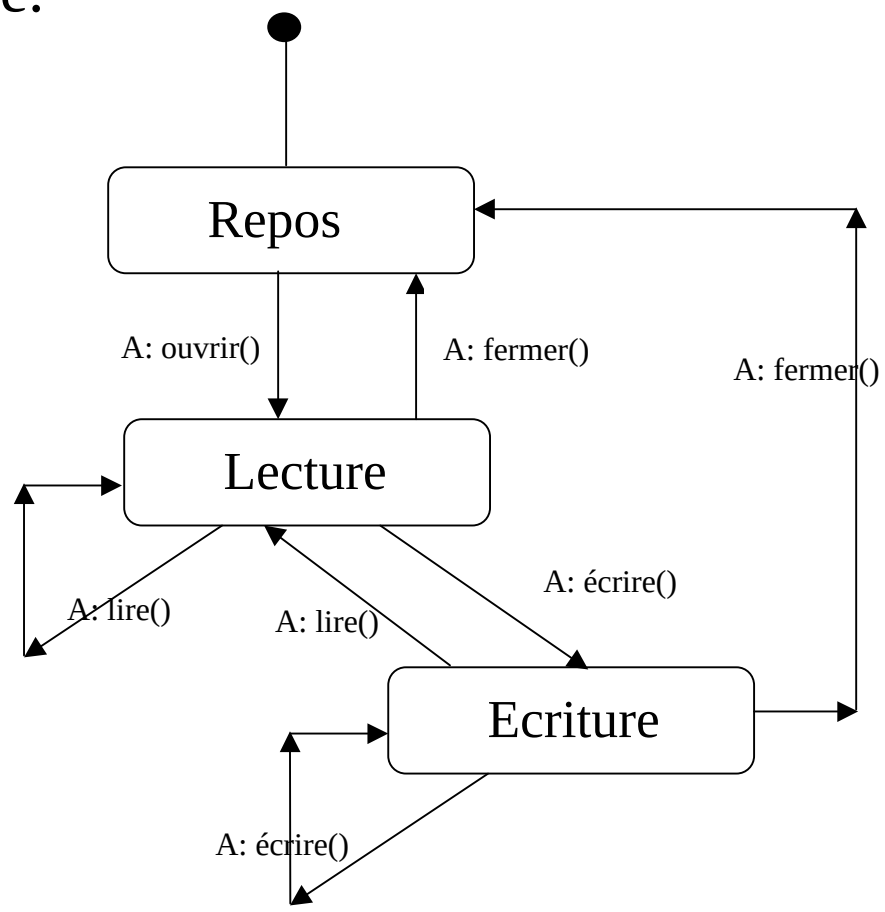
'UML Statecharts' : une variante de Harel.

UML : PSM 'Protocol State Machines'

Objectif : Modéliser le comportement collectif des méthodes d'un objet (en fait modéliser le service).

. un ensemble d'états.

. un ensemble de transitions étiquetées avec une expression booléenne (garde) et une invocation de méthode.



Autres diagrammes UML : 'activity', 'action semantics', SDL.

Conclusion : mode message asynchrone

- C'est le mode le plus basique

Comparable à l'assembleur

Réalisation d'instructions

d'affectation, de branchement.

- Le moins contraignant il permet aux utilisateurs par des échanges successifs, la construction de tout type de protocole.

- L'utilisateur n'a pas en général envie d'être obligé de construire ses propres outils d'où:

Le besoin d'autres schémas prédéfinis plus complexes.

L'enrichissement du mode message en termes de qualité de service par des couches successives.

- Ce mode est encore **le mode privilégié des interfaces réseaux.**