INGENERIE DES PROTOCOLES

Refs: * "L'ingénierie", P. Augesto
Inter Editions
* "Réseaux de Pétri et System Parallèles", G. Vidal Naquet
Arnaud Colin
* "Réseaux de Pétri, Modèles fondamentaux", M. Diaz
Hermes Collection Info 2001

I_ INTRODUCTION:

(1) Protocole de communication: complexité et problèmes.

- Protocole: type de programme parallèle dont le rôle est d'acheminer des infos. Font partis des domaines algorithmiques distribuée (répartie).
Protocole se déroule sur un ensemble de machines reliées par des réseaux.
- Fonctionnalités d'un protocole (= service fourni):
  - traitements répartis:
    - * accès distants à (des traitements / des infos)
    - * calcul réparti
    - * B.D réparti.
    - * systèmes de fichiers distribués
  - communications:
    - * échanger de messages (email)
    - * échange de fichiers
    - * dialogues point à point ou multipoint

- Difficultés : concevoir des entités distribuées supportées par : * des calculateurs ≠ (performances ≠ pour le matériel et le logiciel)
  * reliées par des réseaux de types variés (vitesse, service, caractéristique).

Problème posé par l'algo distribué :

⇒ Chaque entité n'a qu'une connaissance partielle de l'état global du réseau, état global qui demeure inaccessible.

⇒ Canaux de communication non parfaitement fiables (messages perdus altérés, retardés ou dupliqués).

⇒ Vitesses respectives dépendent des processeurs et du réseau. Pas de référence temporelle globale.

Principales erreurs des algo distribués :

* Interblocages (deadlock) : Chaque entité attend un événement (ex : réception message) que le système ne peut produire → système bloqué.

* Bouclages (livelock) = cycles infinis improductifs.

* réceptions non attendues : des messages non prévus sont reçus et bloquent une entité ⇒ par propagation, risque d'interblocage.

But de l'ingénierie des protocoles :

Proposer des outils, méthodes, techniques pour :
- concevoir des protocoles
- analyser et vérifier leurs propriétés.

② Techniques de descriptions formelles:

Modèles maths qui supportent des langages infos pour spécifier, vérifier et implanter des protocoles ⇒ fournir une spécificité complète, non ambiguë, claire et transmissible.

⇒ description se fait selon un modèle ou un langage.

⇒ description simultanée ou validée pour détecter à priori les pb.

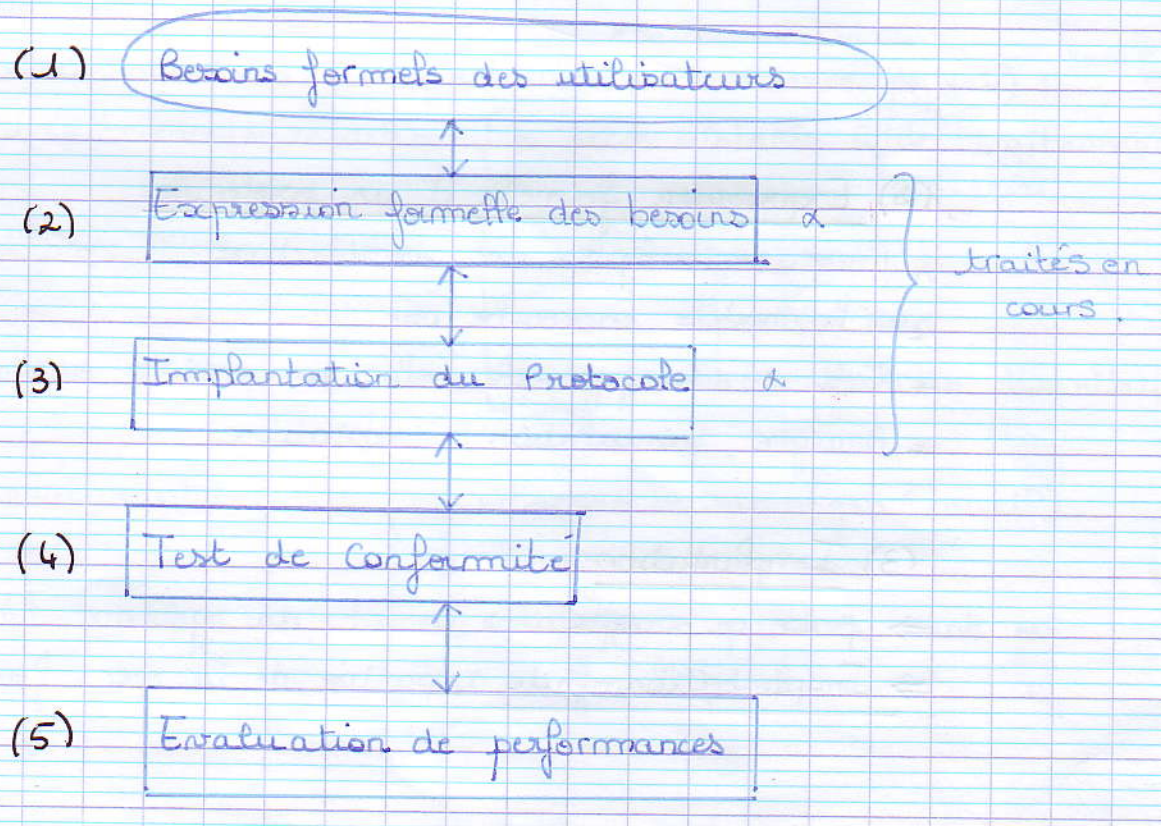
⇒ Base de test de conformité pour l'implantation.

⇒ Génération de maquettes d'implantation.

Techniques de descriptions formelles caractérisées:

* modèle math qui les supporte (⇒ analyse et vérification)
* pouvoir d'expression (capacité à écrire le pb)
* structuration des spécifications.
* Abstraction (différents niveaux de détails).

③ Démarches générales de conception:



(1) Descriptions informelles ou semi-formelles:

- langage naturel avec qques notations spécifiques
- description des primitives de services.
- diagrammes ou tables d'états
- diagrammes temporels

tables d'état d'une entité:

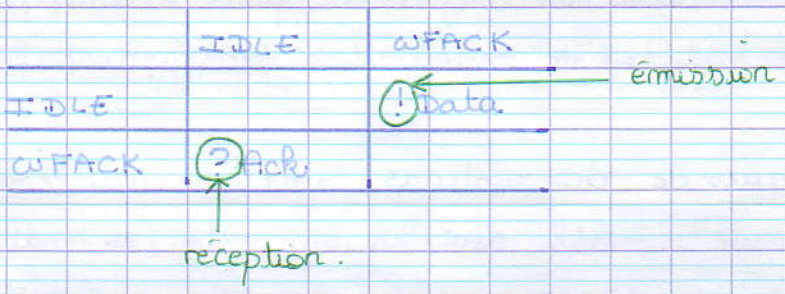
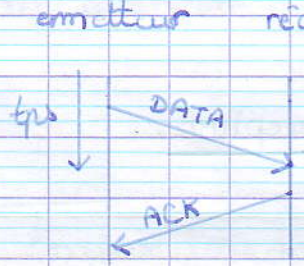


diagramme temporel:



(2) Expression formelle d'un protocole:

* Prise en compte une architecture de spécification basée sur le modèle en couche osi
* Écriture du protocole selon un modèle ou langage
* Analyse, Simulation, Vérification

(3) Implantation:

⇒ prise en compte les caract. du système hôte.
⇒ Implantation faite à partir de la spec. formelle.

(4) Test de Conformité :
L'implantation vérifie les caractéristiques attendues.

(5) Evaluation de performance :
Performance de l'implantation.

④ Remarque :
Technique de descriptions formelles : 2 points de vue :
- Génie logiciel : cycle de développement de protocoles.
- Modèles formels : Modélisation, Analyse, Preuves.

II_ MODELES POUR SPECIFICATIONS FORMELLES :

2 types : - orientés description propriétés { ↑ preuves ; ↓ implantations
- orientés description du comportement des entités { → preuves ; ↑ implantations

① Logique :
≠ logiques : prédicats, 1er ordre, temporelles, modèles.
Ex : (1er ordre) : Au plus un programme Pi peut exécuter à la fois la partie section critique.
$\exists ! P_j \in \mathcal{P}$, section critique (P_j)
Tout message doit être émis avant d'être reçu $\forall m$,
date_émission (m) < date_réception (m)
⇒ spécifications abstraites, très peu orientées description des comportements des entités.
⇒ mal adapté à la conception.

② Automates à états finis (machines à états)

→ Modèle le plus simple de description.

- Déf des entités communiquantes
- Chaque entité a son comportement décrit par 1 automate.

Machine à états: Sextuplet $\langle Q, q_0, I, O, A, T \rangle$

Q: ensemble fini d'états

q_0: Etat initial

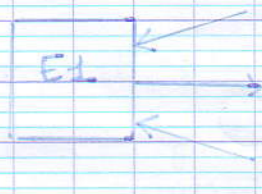
I: ensemble fini d'états d'entrée (exc: réception d'1 msg)

O: ——— de sortie (exc: émission ———)

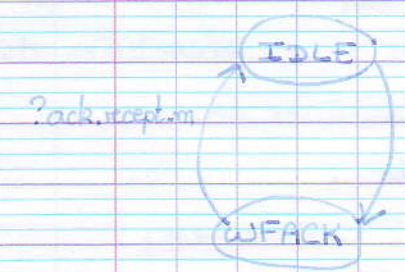
A: fonction de création d'état de sortie ($Q \times I \to O$)

T: fonct° de transition d'état ($Q \times I \to Q$)

Ex: Entité E1 :
- qui attend 1 ordre d'envoi env_m
- à la réception, elle émet le msg m
- attend que l'entité paire ait reçu m (réception de ack_m)



ⓐ machine à états de E1 :



événements → séparation → action

SOUS-FORME DE TABLEAU :

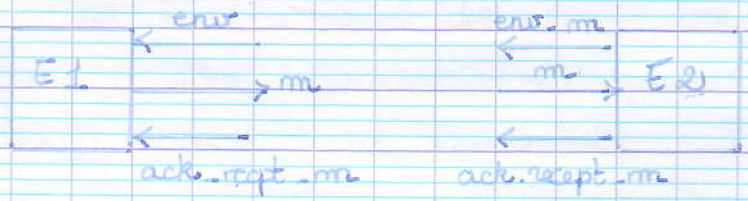
	IDLE	WFACK
IDLE		?env_m
WFACK	?ack_recept_m	

⑥ Machines à états étendues :

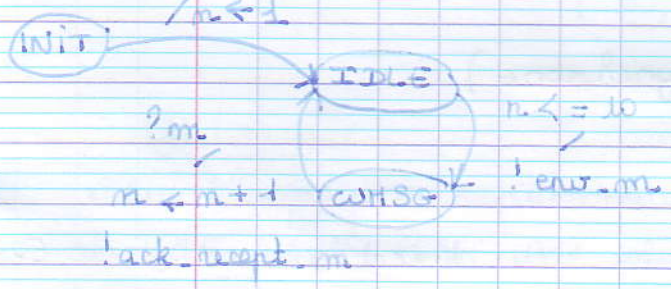
→ Extension avec ajout de :

- variables locales aux entités.
- prédicats (fonctions booléennes) renvoient V/F suivant certaines variables.
- messages typés avec plusieurs champs possibles.
- délais de tir de transition.
- priorités.

Ex- Entité pairé E2 qui demande à E1 10 messages.



machines à états étendues de E2 :



< condition > / < action >

Si condition est vraie, alors la transition est tirable (activable)

Si l'on tire la transition, alors la partie < action > est réalisée.

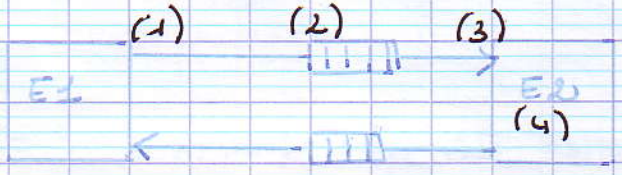
⑦ Connexion de machines à état :

↳ moyens de communication entre entités.

Asynchrone: 2 files d'attente de FIFO en général de capacités infinies.

Canal bidirectionnel: 2 files, FIFOs connectées à 2 entités

Com asynchrone: l'émission de msg n'est pas synchro avec sa réception (réception indépendante de l'émission, a lieu après)



Etapes: 1) Avant la communication

2) Emission d'1 mess de la FIFO

3) Lecture du msg par le récepteur E2 lorsqu'il est en tête de FIFO.

4) Après communication.

((2) et (3) ⇒ com asynchrone)

Rem: Evolution des 2 machines non synchronisée (comportement de E1 sans lien direct avec celui de E2)

Synchrone: (rendez-vous)

- la transmission d'1 msg se fait en synchronisant son émission et sa réception.

- La communication n'a lieu que lorsque le récepteur est prêt à recevoir le message
- pas d'état transitoire entre émission et réception.
- Evolution synchronisée du comportement des 2 machines

→ Les 2 machines changent d'état "en même tps"

Etapes : 1) Avant le RDV

2) Emission de m, avec réception de m et réalisat° des actions associées pour E2 (RDV) + changement d'état synchronisé des machines.

3) Après RDV.

Rem : • Le RDV peut être mis à l'utilisation d'1 FIFO vide.

• 7 RDV bidirectionnels, avec des RDV à N entités.

3) Réseaux de Pétri (RdP)

Extension des machines à états, proposée par C.A Pétri (62)

a) Rdp places / transitions :

Quadruplet $Q = \langle P, T, Pré, Post \rangle$

P : ensemble fini de places

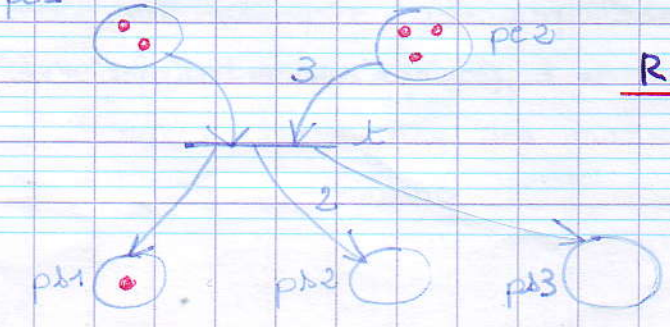
T : ——— de transitions

Pré : $P \times T \to N$ (entiers). Fonct° de valuation d'incidence Avant } Structure du RdP

Post : $T \times P \to N$ ——— arrière }

Graphique : RdP = graphe orienté : * noeuds = places

* barres = transitions



Rem : une flèche sans numéro correspond à 1.

Marquage du RdP :

Ajout d'1 fonction M : P → N

- Nombre de jetons contenus ds chaque place.

But : rendre le réseau actif pr représenter 1 comportement.

< R, M > = RdP marqué.

M∅ : Marquage initial.

Ex : Marquage en rouge.

P : { pe1, pe2, ps1, ps2, ps3 }

T : { t }

Pré (pe1, t) = 1 (1 flèche sans n° = équivaut à 1)

Pré (pe2, t) = 3

Pré (ps1, t) = pré (ps2, t) = pré (ps3, t) = 0

(pas de flèche de psi vers t)

Post (ps1, t) = 1

Post (ps2, t) = 2

Post (ps3, t) = 1

Post (pe1, t) = Post (pe2, t) = 0 (pas de flèche t → psi)

M (pe1) = 2

M (pe2) = 3 M = Marquage

M (ps1) = 1 (nbre de jetons ds la place)

M (ps2) = 0

M (ps3) = 0

Tir d'1 transition :

"t" est franchissable (tirable) ssi : le nbre de jetons des places d'entrée pei >= poids des arcs des places d'entrée pei → t, (donné par les fct° Pré)

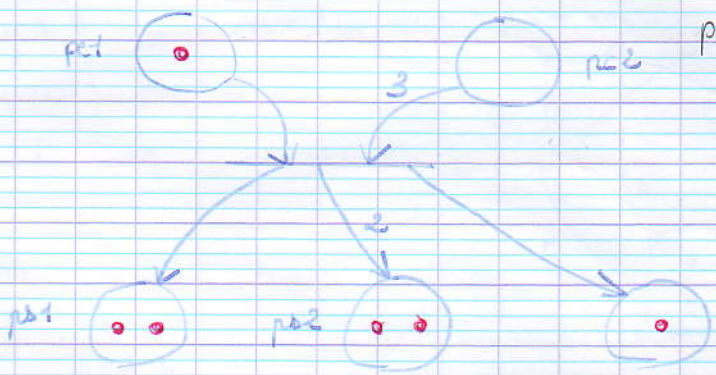
Action de tir :

→ Pr chaque place d'entrée $Pe_i \to t$ on enlève le nbre de jétons donné par $Pré(Pe_i, t)$.

→ Pr chaque place de sortie $t \to Ps_j$, on ajoute le nbre de jétons donné pour $Post(Ps_j, t)$.

Ex: tir de t.

⚠ : Le nombre de jetons n'est pas conservatif. (Nbre global peut varier)



→ le tir d'1 transition modifie le marquage du RdP lors t tirée, on a :

$$\forall p_i \in P,\ M'(p_i) = M(p_i) - Pré(p_i, t) + Post(p_i, t)$$

- $M'(p_i)$: nouveau marquage
- $M(p_i)$: ancien marquage
- $Pré(p_i, t)$: jetons consommés par le tir de t
- $Post(p_i, t)$: jetons produits par le tir de t.

Rem : * le tir d'1 transition est atomique (indivisible)

x On ne peut tirer qu'une transition à la fois (si plusieurs tirables, on en choisit une).

Représentation d'1 RdP sous forme matricielle (taille & Dim)

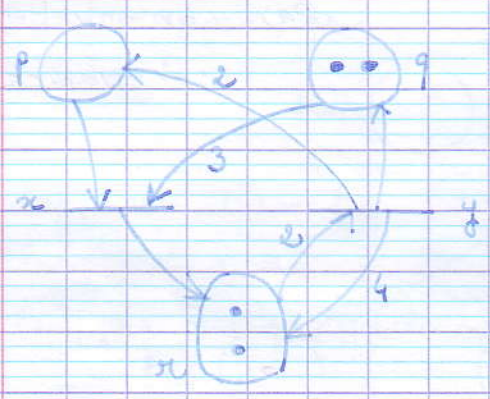
→ le calcul matriciel représente l'évolution du RdP

* Numéroter les places P et les transitions T.
* P seront les lignes des matrices
* T _____ colonnes _____

* pré(p, t) : matrice à coeffs dans $\mathbb{N}$.
* post(p, t) : ——————————
* C = Post − Pré matrice d'incidence du RdP.

Ex: $P = \{p, q, r\}$ $\quad T = \{x, y\}$

Pré :
$$\begin{array}{c} \\ p \\ q \\ r \end{array}\begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{pmatrix} 1 & 0 \\ 3 & 0 \\ 0 & 2 \end{pmatrix} \end{array}$$



Post :
$$\begin{array}{c} \\ p \\ q \\ r \end{array}\begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{pmatrix} 0 & 2 \\ 0 & 1 \\ 1 & 4 \end{pmatrix} \end{array}$$

M = Vecteur de marquage = (nbre de jetons dans les places)
$$\begin{array}{c} \\ p \\ q \\ r \end{array}\begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}$$

$$C = \text{Post} - \text{Pré} = \begin{pmatrix} -1 & 2 \\ -3 & 1 \\ 1 & 2 \end{pmatrix}$$

tx : {Vecteur caractéristique de la transition x
{ttes ses composantes seront nulles, sauf celles de x mise à 1.

$$tx = \begin{array}{c} \\ x \\ y \end{array}\begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad ty = \begin{array}{c} \\ x \\ y \end{array}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- ti : tirable ssi ttes les valeurs de M sont >= à celles de la colonne qui correspond à ti ds la matrice Pré.
M >> Pré * ti (isole la i-ème colonne)

- tir de ti : * soustraire les valeurs de la colonne de ti de Pré à M
* ajoute —————— Post à M

$M' = M - Pré * ti + Post * ti$

$M' = M + (Post - Pré) * ti$

$M' = M + C * ti$

Rem: Représentation matricielle va servir pour l'analyse des propriétés du RdP.

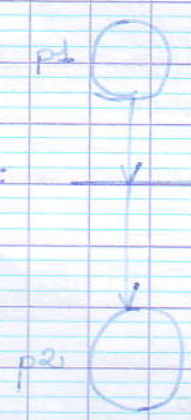
RdP étiquetés:

RdP : objet non typé

Étiquettes : donnent 1 signification aux transitions

1 même étiquette est réutilisée si elle correspond au même événement.

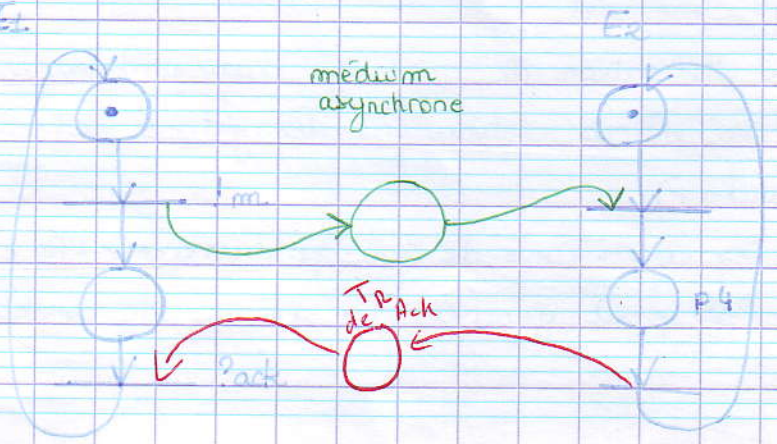
Ex:



Ex de comportement décrit en RdP:

Entité E1 qui émet 1 message m, puis attend son aquittement ack

Entité E2 qui attend 1 message m, puis émet 1 ack.



b) Interconnexion de RdP:

- comportement de chaque entité représenté par 1 RdP.
- comportement des moyens de communication pr relier les entités défini par 1 autre RdP.

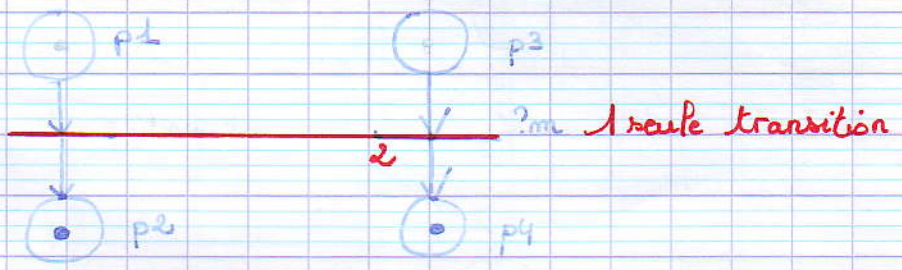
Communication asynchrone:

- place partagée entre les 2 entités.
- permet de modéliser un médium de capacité infinie

Communication synchrone:

- Faites par fusion de transition.

Ex: E1 E2



c) Principales extensions des RdP:

RdP = facile à utiliser.

Inconvénient: bcp de places et de transitions pour modéliser 1 système réel.

Système réel:

→ Utilisation de modèles plus compacts et plus puissants:

→ les places contiennent des jetons identifiés par 1 numéro (couleur)

→ les conditions de Pré portent sur le nbre de jetons

→ Post peut générer des jetons de couleurs ≠.

Rem: - directement équivalent à des Rdp classiques par pliage / dépliage.

- regroupent souvent des entités avec des comportements identiques.

Rdp prédicats / transitions (Genrich)

→ constantes manipulées par des jetons.
→ chaque jeton est 1 n-uplet (+ variables structurées)
→ les conditions de Pré sont fixées par les valeurs et profils de jetons.
→ le tir des transitions utilise l'unification (= lien entre 1 profil de jeton et 1 jeton réel).

Rdp à objets (Silberton - Blanc)

→ les jetons sont en fait des objets au sens informatique (liste de valeurs + comportements)

Rdp temporels

- 1 placé ≡ déroulement d'1 activité.
Intervalle de tps de chaque place ≡ tps de déroulement de l'activité.

Rdp temporisés

Intervalles de tps associés aux transitions (intervalle de tir t)

Rdp stochastiques

Densités de pb à chaque transition.

4) Algebre des Processus

Proposé par Robin MILNER avec le calcul des systèmes communiquants (CCS)

But : Représenter 1 Système sous forme de process :

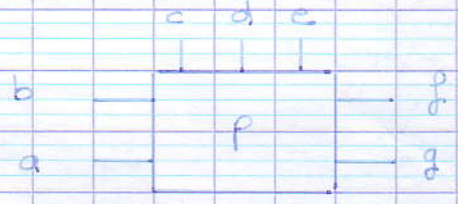
- possède 1 comportement
- réalise des actions de com. avec d'autre process

Fournir des modèles d'interaction et de déroulement des process par des synchronisation

Principales opérations (basées sur le langage basic LOTOS).

process : Boîte noire qui intéragit avec son environnement via des portes.

ex :



process P avec portes a à f.

Comportement d'1 process :

- action de base.
- et/ou . sous-process P' inclus dans P.

Actions de bases :

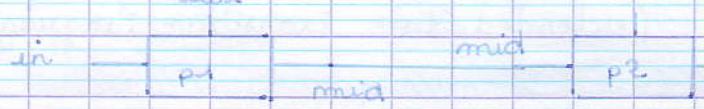
- STOP process inactif (fin du process)
- Action a réalisée a ; B (action a puis devient le process B)
- Choix B1 [] B2 (soit B1, soit B2)
- répétition (récursion) B ← a ; B.

Synchronisation de process :

- faites sur des portes communes, par RDV.

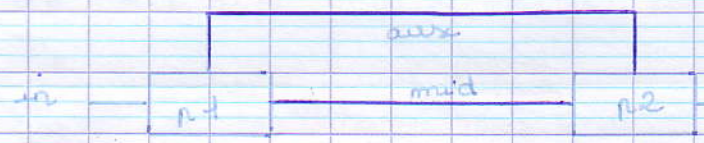
p1 |[mid]| p2.

Action mid faite par P1 se synchronise avec celle de P2



Action mid faite par P1 synchronisée avec celles de P2.

Synchro totale : toutes les portes de même nom, entre 2 process (ici mid et aux) p1 || p2.



Entrelacement : p1 ||| p2

Aucune porte mise en commun entre P1 et P2.

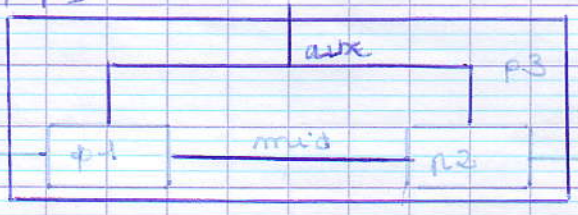
⇒ comportement de P1 indépendant de celui de P2.

Masquage de porte :

1 porte entre 2 process devient interne à 1 3ème process.

Hide mid in P3

Ex :



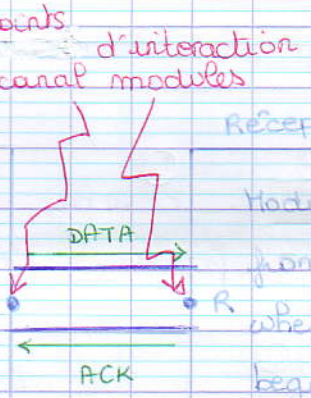
Remq: Tout évènement sera détectée depuis l'extérieur de ES
⇒ 1 action "i" inobservable.
⇒ ≠ niveaux d'abstract° des comportements.
⇒ les portes servent dans des extensions du langage,
à transmettre des valeurs (Full LOTOS)

III - Principaux langages formels de description de protocoles:

1. Estelle: (extended State Transition Language) (ISO IS 9074)

- machines à états communiquants étendues.
- types de données et fonctions écrites en PASCAL
- communications: canaux FIFOS.
- structuration: - modules génériques instanciables dynamiquement.
  - 1 module ≡ 1 machine à état
  - 1 module peut contenir des sous-modules.

Exc:




```
Module Emetteur                    Module Récepteur
FROM IDLE TO WFACK                 from IDLE TO IDLE
begin                              when R.DATA(...)
  output S.DATA(...)               begin
end;                                 stocke_parametres(...)
from WFACK to IDLE                   output R.ACK
when S.ACK                         end;
begin
end;
```

S et R: points d'intéraction utilisés par les modules, pr communiquer avec l'extérieur (sont connectés au canal).

Environnement support: ESTIM (proto projet SEMOS)
ASA +
Estelle Development Toolset (INT)

2) SDL:

Specification and Description Language (UIT-T Z.100 à Z.106)

* Machines à états communiquent par FIFOS
* Types de données écrits en Act One.
* Syntaxe Graphique Normalisée (+)

Environnement support: - Ellis (F Tel R&D)
- ITAU G2 (ITAU)

3) LOTOS

Language of Temporal Ordering Specification (ISO IS 8807).

- Basé sur calcul des systèmes communiquants.
- Communication par RDV.
- Types de données en Act One.

Environment Support: LOTOS (Projet LOTOSPHERE)
TORTLE (proto CNRS)

4) Rdf:

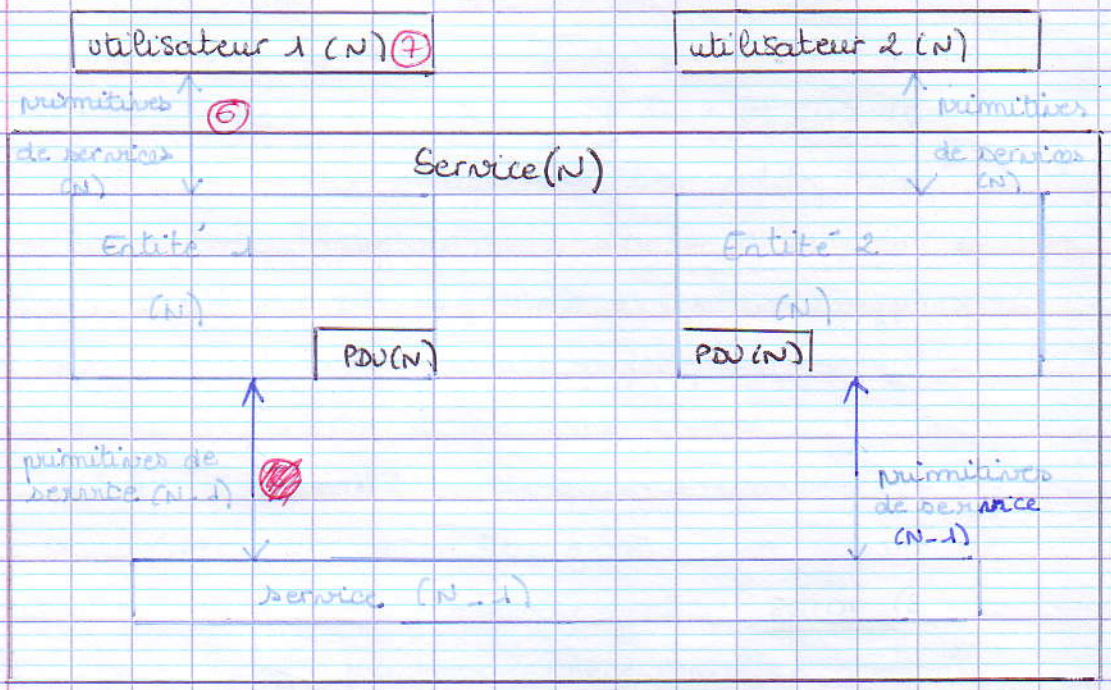
Nombreux éditeurs, simulateurs, vérificateurs, greffons, PIPN (proto CNRS), Mozilla - Rdf (Société ixi).

Rem: Tous ces environnements supportent simulation et/ou vérification et/ou test et/ou implantation des spécifications écrites.

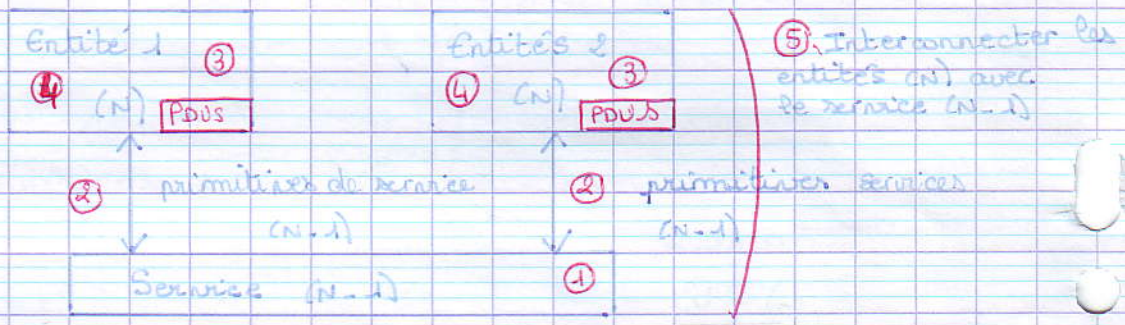
Je t'aime

IV - Conception de Protocoles :

1) Contexte : Modèle en couches OSI de l'ISO :



2) Architecture pour l'analyse (modèle à 2 niveaux) :



① - A partir des propriétés informelles du service (N-1), décrire 1 module qui traduit ces propriétés (ex : Service non fiable défini par 2 transitions, 1 qui achemine et 1 qui perd les messages).

② - Décrire les primitives de service (N-1) utilisées par les entités supérieures.

③ - Décrire les PDUs échangés entre entités paires.

④ - Formuler le comportement local de chaque entité (N).

⇒ la mise au point et la vérification se font en étudiant le comportement global du système interconnecté.

3) Architecture à 3 niveaux pour l'implantation :

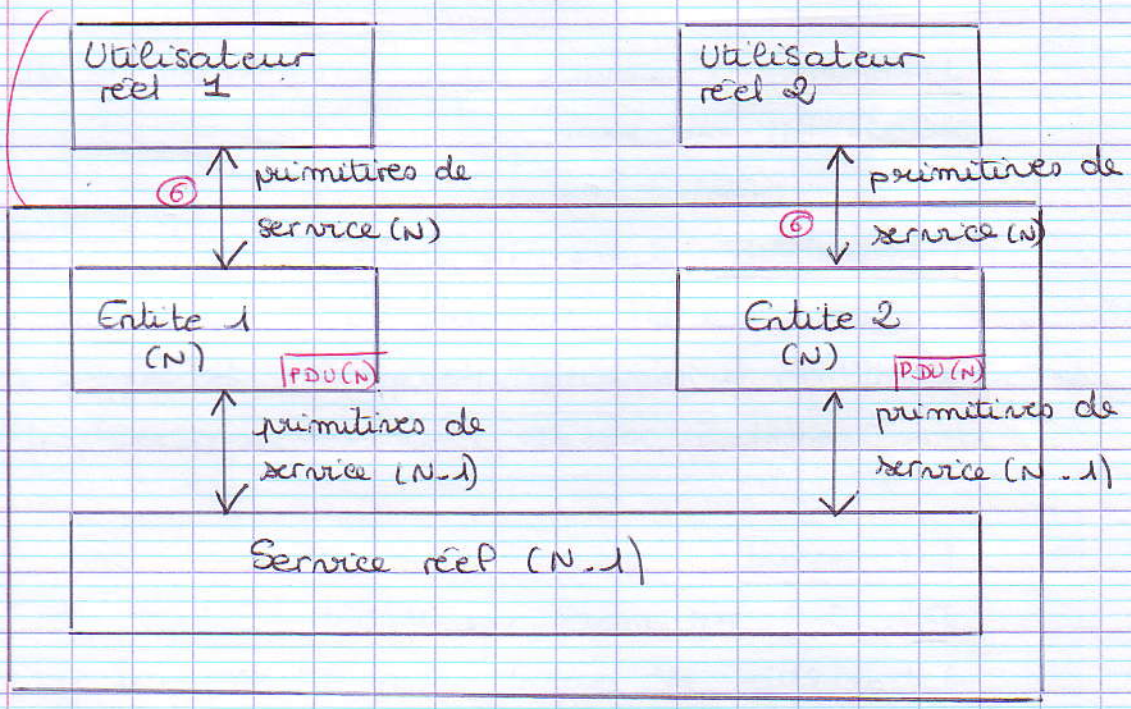
⑥ Ajout de primitives de service (N)

⑦ Ajout de modules qui représentent le comportement d'utilisateurs réels des services (N).

⑧ Interconnecter les utilisateurs au système

Dans ce cas, on étudie l'ens. du système comportant les 3 niveaux.

Comment obtenir l'implantation réelle



L'implantation se fait en remplaçant le service simulé (N-1) par le service réel et en fournissant les primitives de service (N) à des utilisateurs réels.

I - Vérification :

Rem : plutôt orienté à base de RdP.

Vérification = analyse du système pour sa validation.

(propriétés spécifiées ≡ propriétés attendues)

1) Vérification des propriétés :

2 types :

propriétés générales :

doivent être satisfaite par tous les systèmes.

RdP borné :

1 RdP borné ssi, pour tout marquage accessible M, le marquage de chaque place est borné supérieurement.

$\forall p \in P, \exists B \in \mathbb{N}, \forall M \in A(M\emptyset), M(p) \leq B$.

théorème : l'ensemble des marquages d'1 RdP est fini ssi le RdP est borné.

Interprétation : l'ens. des états d'1 système représenté par 1 RdP borné est fini.

Si RdP non borné, infinité possible de jetons dans 1 place, débordement de capacité, système non réalisable.

RdP vivant :

Def : Soit $M\emptyset$ marquage initial

1 transition est vivant pour $M\emptyset$ ssi, quel que soit le marquage M atteint, il est tjrs possible d'atteindre 1 marquage M' qui sensibilise t à nouveau $\forall M \in A(M\emptyset)$, $\exists M' \in A(M)$ tq M' sensibilise t.

Interprétation : RdP vivant ⇒ pas de blocage du système.
RdP non vivant ⇒ Blocage partiel : Au moins
aucune transitions restent tirables, mais d'autres ne le
sont plus, à partir de certains états du système.

Blocage total (deadlock) : plus aucune transition tirable.
Système entièrement bloqué ⇒ erreur.

RdP réinitialisable :
le système repasse par son état initial M_0.
$\forall M \in A(M_0), M_0 \in A(M)$.

Interprétation : Fonctionnement cyclique du système.

→ propriétés générales { - borne
- vivacité
- réinitialisation

2). Propriétés spécifiques.

* propriétés que doivent satisfaire le système réalisé.
* pour 1 protocole

- échange à l'alternat
- accès unique à des ressources partagées.
- exclusion mutuelle...

c). Analyse du RdP:

3 méthodes:
- réduction du RdP
- Analyse du graphe des marquages
- recherche d'invariants.

a)- Réduction:

→ Réduire la taille du RdP en gardant certaines de ses propriétés.

→ Suppression de places et de transitions selon certaines règles.

Pb: perte de signification du RdP réduit

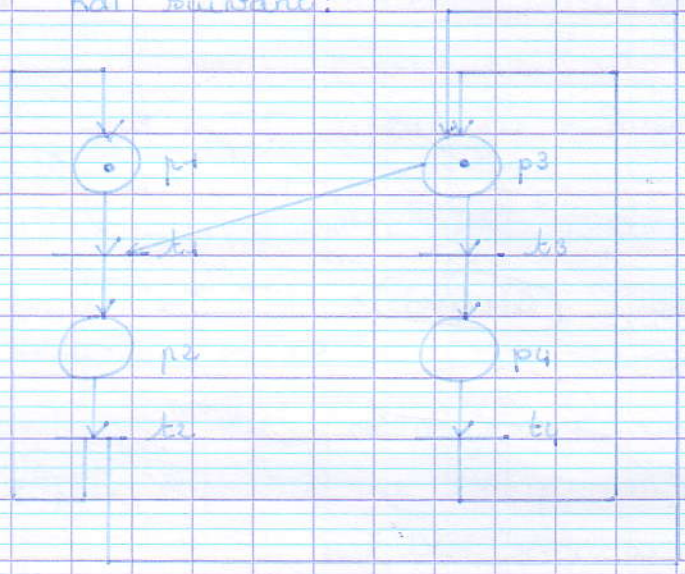
b)- Graphe des marquages accessibles (ou Graphe d'accessibilité).

Analyse du RdP par examen de tous ses états.

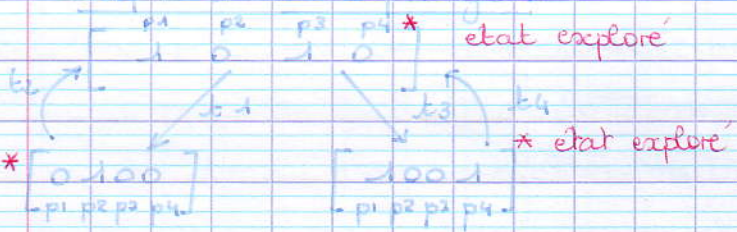
Déf: le graphe représente toutes les situations possibles pour l'évolution du système. Définit tous les marquages possibles et toutes les séquences de tirs réalisables.

RdP		Graphe des marquages
marquage courant	⟺	1 noeud du graphe
tir de 1 transition t	⟺	1 flèche étiquetée par t du graphe
$A(M_0)$ (ensemble des marquages accessibles)	⟺	ensemble des noeuds du graphe

Exc: RdP suivant:

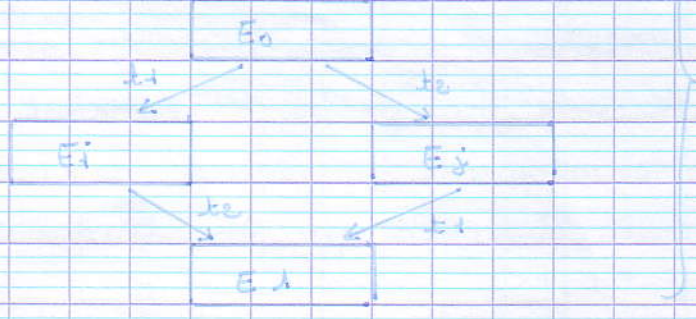


Graphe des marquages:

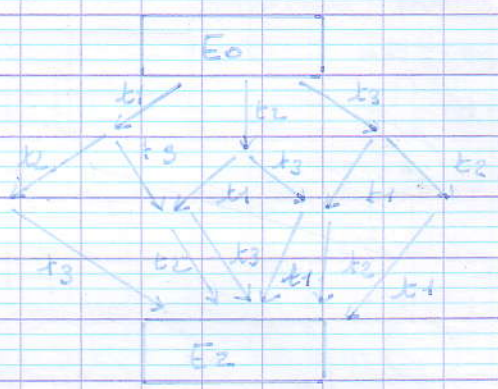


Rem: La sémantique utilisée pour représenter le parallélisme dans le graphe des marquages est celle de l'entrelacement ⇒ Deux comportements indépendants (pouvant se dérouler en parallèle) sont représentés "en quinconces" dans le graphe des marquages. t1 // t2

graphe des marquages:



$t_3 // t_2 // t_3$



Rem: pour la spécification faite avec des machines à états communiquant par FIFOs, le graphe d'accessibilité s'obtient comme suit :

1 état = { valeur courante de l'état de chaque machine, valeurs courantes de ses variables, contenu de chaque FIFO }

transition = Idem RdP.

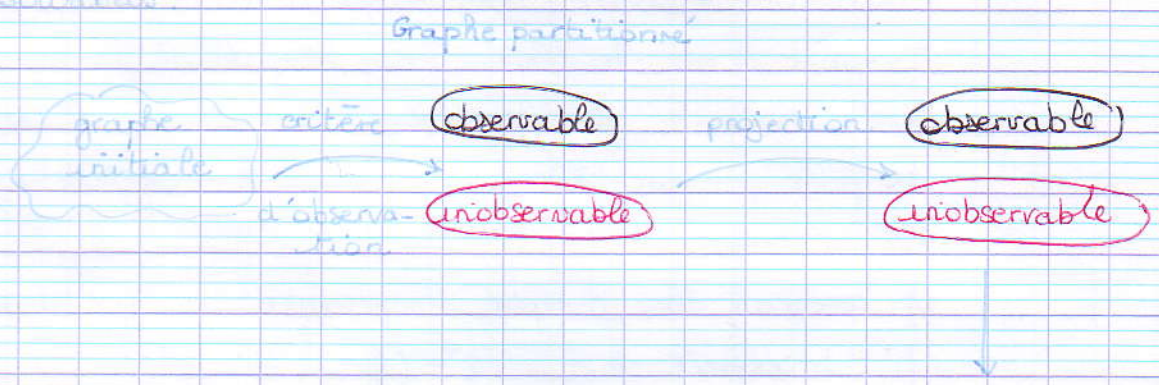
* la simulation est équivalente à un parcours aléatoire dans le graphe des marquages.

2) - Analyse du graphe d'accessibilité directe:

Etude de la structure du graphe, de ses chemins, des états puits (états sans transition suivante), analyse des séquences de transition, étude des propriétés à l'aide de la logique.

projection : Graphe d'accessibilité très gros.
idée : en observer une partie en donnant certains critères d'observation.
Ex: On ne veut observer que les transitions qui contiennent "t_1".
. Partition du graphe en 1 partie observable (que l'on va étudier) et 1 partie interne (inobservable).

- Vérification par abstraction : utilisation d'une projection (équivalence observationnelle) qui va réduire le graphe qui ne gardera que les relations entre parties observables.



Remq: Approche par abstraction très utilisée pour étudier le service rendu (on ne conserve que les transitions qui ont un effet vers les couches supérieures, et on masque les transitions liées aux mécanismes protocolaires)

→ Approche par abstraction très utilisée en algèbre de processus.

d). Analyse structurelle (invariants):

→ Utilisation de la représentation matricielle des RdP et de l'algèbre linéaire pour déduire des propriétés du système.

Equation fondamentale:

s: suite de transitions par affiles.

$\vec{s}$: Vecteur d'occurence de s = chaque composante i de $\vec{s}$ donne le nbre de fois que la transition t_i intervient dans s.

$$M \xrightarrow{s} M'$$

En utilisant les propriétés d'associativité et de commutativité du calcul matriciel,

on obtient :

$$\boxed{M' = M + C * \vec{s}}$$

$M_1 = M + C * t_1$
$M_2 = M_1 + C * t_1$
$\vdots$
$M' = M_{i-1} + C * t_{i-1}$

Invariant de place (composante conservative) :
Trouver l'ens. de places pour lesquelles la somme des jetons reste constante au cours de l'évolution du système
⇒ pour trouver des propriétés qui se conservent dans le système.

f : vecteur dont chaque composante désigne 1 place ($f = \begin{pmatrix} f_1 \\ \vdots \\ f_p \end{pmatrix}$)
f^T : transposée de f $\quad f^T = (f_1, \dots, f_p)$
En appliquant la relation fondamentale multipliée par f^T, on obtient :

$$f^T * M' = f^T * M + f^T * C * \vec{s}$$

Si f^T est 1 composante conservative, alors, au cours du tps, on a : $f^T * M' = f^T * M$.

ce qui implique : $f^T * C * \vec{s} = 0$
on a pris $\vec{s}$ suite de transitions, mon nulle :

$$\Rightarrow \boxed{f^T * C = 0}$$

Relation à appliquer pour trouver les invariants.

calcul de l'invariant :

$$f^T * M' = f^T * M = \boxed{f^T * M_0 = \text{cste}}$$ valeur de l'invariant

Déf : $\|f\|$: support de f = ens. des composants non nuls de f.

Rem : f doit être mon nul pour désigner 1 invariant de place.
f : appelé 1 semi-flot.

Invariant de transition (composantes stationnaires):
Déterminer des cycles répétitifs de transitions $\vec{\sigma}$ (sigma barre) = Vecteur dont chaque composante est 1 transition.

Relation fondamentale: $M' = M + C * \vec{\sigma}$
$\vec{\sigma}$ désigne 1 composante stationnaire : $M = M'$
ce qui implique : $\boxed{C * \vec{\sigma} = 0}$ (Relation pour trouver 1 cycle)

Rem: $\vec{\sigma}$ est non nul.

propriétés:
- 1 de place qui appartient à 1 composante conservative est bornée.
- RdP conservatif ssi toutes places sont parties d'1 composante conservative.
- 1 RdP conservatif est borné.
- 1 RdP vivant est stationnaire (& réciproque fausse)

Rem: l'analyse structurelle évite de construire le graphe des marquages