République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abbes Laghrour Khenchela
Faculté des Sciences et de la Technologie
Département de Mathématiques et Informatique

Mémoire pour obtenir le diplôme de

Master en Informatique

Implémentation du Model STE

Presenté par
BEN OTHMANE Ghania

Dirigé par
M BEN OTHMANE Mohamed

Jury :

Université de Khenchela	Président
Université de Khenchela	Examinateur

Soutenu en 2017

Dédicace

A mon père Sebti

à la fleur de ma vie ma mère Souad

à mes frères : Ahmed, Sami, Moufid

aux plus proches de mon cœur mes sœurs : Djamaa, Hadjira

et à toute ma famille spécialement Dr. BEN MBARK Maya

je dédie ce mémoire.

Remerciements

Au nom d'**Allah** le plus grand,

merci lui revient de nous avoir guider vers le droit chemin, de nous avoir aidées tout au long de nos années d'étude.

Je tient à remercier vivement Mr. BEN OTHMANE Mohamed, d'avoir accepter d'être rapporteur de mon travail ainsi pour le temps et l'énergie qu'il a consacrés à ce travail, et également ses encouragements, son accompagnement, son soutien sa patience et ses conseils scientifiques et personnels durant ce travail.

Je remercie également tous mes enseignants qui mon enseignée durant ces 5^{ans} spécialement Mr. Toufik Maarouk, Mr. SIAM Abderrahime, ...

Je remercie Mr. le président du jury, également Je remercie Mrs. les membres d'avoir acceptés de participer à ce jury et d'examiner mon humble travail.

Je remercier ma famille, pour tout ce qu'ils ont fait pour moi, merci pour leur patience.

Résumé

Le sujet s'intègre dans le cadre de la plateforme MP-SAVE (Multi Paradigme - System Analysis and Verification Environemment). Dans cette plateforme, différents modèles sémantiques et de spécifications ont été déjà définis, et leurs classes sont déjà implémentées.

Le sujet consiste en la réalisation d'un compilateur pour le langage Basic-LOTOS dont le code généré sera un LTS.

Mots-Clés

Basic-LOTOS, Vraie Concurrence, Entrelacement, Full-LOTOS, ST, STE, STEMs,..

Abstract

Keywords

Basic-LOTOS, CCS, Full-LOTOS, ST, STE, STEMs,...

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction générale

1.0.1 Contexte et Problématique

1.0.2 Contributions

1.0.3 Plan du document

Ce mémoire est présenté sur 5 chapitres :

• Le **Chapitre 1** représente une introduction générale qui permet d'exposer la problématique est un plan du mémoire.

• Le**Chapitre 2** est consacré à une présentation générale , des langages de spécifications formelles : les algèbres de processus(LOTOS , CCS, SEV,...), Les Réseaux de Petri.

• Le**Chapitre 3** ce chapitre rappelle, les système de transitions et quelques-unes de ces classes et extensions.

•Le**Chapitre 4** Dans ce chapitre nous allons procéder a l'implémentation objet du model de base STE, à cette fin nous utilisons l'environnement **JAVA** que nous allons présenter dans une première section, puis nous allons présenter les algorithmes nécessaires aux opérateurs de base d'une Algèbre Basic-LOTOS. Enfin nous allons présenter quelques fiches de l'application développée.

•Le**Chapitre 5** finalement dans ce dernier chapitre on termine par une conclusion générale, perspectives.

Chapitre 2

Spécification formelle des systèmes concurrents

2.1 Introduction

Dans ce chapitre, nous présentons les principes généraux de la conception formelle des systèmes concurrents des moyens formels pour la spécification et l'analyse d'applications réparties et concurrentes.

La première section introduit le formalisme CCS et LOTOS les modèles des structures d'événements. Le second expose les réseaux de Petri.

2.2 Modèles algébriques pour la concurrence

2.2.1 Introduction aux algèbres de processus

Dans la littérature le mot processus signifie une exécution séquentielle d'instructions[Bae]. La définition pour la première fois d'une algèbre décrivant le comportement d'un système concurrent fait introduire le concept d'agents qui peuvent être compose par des opérateurs de composition séquentielle, de choix, parallèle... . Par la suite les mots processus et agent ont été utilisés indifféremment dans la littérature. Le mot algèbre dénote que nous prenons une approche algébrique/axiomatique pour la définition des comportements[T.M16].

Une algèbre de processus est un langage concis pour la description des étapes d'exécution de processus. Ce langage est constitué d'un ensemble d'opérateurs et de règles syntaxiques pour spécifier un processus utilisant

de composants simples ou atomique. Les algèbres de processus utilisant la notion d'équivalence pour montrer que deux processus ont le même comportement. Plusieurs algèbres de processus ont été utilisées pour spécifier et analyser des processus concurrents qui communiquent entre eux. Nous pouvons citer, CCS(Calculus of Communicating Systèmes) de MILNER [R.M80], ACP(Algebra of Communicating Processes) de BERGSTRA et KLOP[JJ85] et LOTOS(Langage Of Temporal Ordering Specification)[TE87]. Ces algèbres de processus citées précédemment sont des algèbres atemporelles du fait qu'elles ne permettent que de raisonner sur l'ordre d'exécution des événements et les étapes d'exécution.

2.2.2 CCS :Calculus of Comminicating Systèmes

Le formalisme CCS[R.M80] décrit les systèmes communicants comme des ensembles d'automates non-déterministes(appelés processus) qui interagissent par le biais de synchronisations. Le comportement d'un processus n'est décrit que partiellement par l'ensemble de ces traces (une trace est une suite de transitions) et la notion de bisimulation propre à cette théorie permet de raffiner la notion d'equivalence des processus.

The Calculus of Communicating Systems (or CCS) is process calculus developed by Robin Milner.

FIGURE 2.1 – Robine Milner

Dans le langage CCS, un processus peut être appelé agent. ce processus décrit à partir d'un ensemble d'actions, l'action peut être :

• Observable : représenté par les lettres a, b,...,

• Non Observable (silencieuse), noté par (τ).

Les actions constituent l'alphabet de base de processus. On suppose un ensemble dénombrable d'actions a, b, c... (input actions) chacune ayant un inverse $\overline{a}$, $\overline{b}$, $\overline{c}$...(output actions) . Le sens intuitif de la notion d'inverse et que l'action et son inverse constituent des actions pouvant se synchroniser si elles proviennent de deux processus communicants (les deux actions a et $\overline{a}$, sont complémentaires).

La syntaxe des processus CCS [R.M80] :

$$\begin{aligned} \alpha &::= a(x) \mid \overline{a}v \mid \tau \\ P, Q, R &::= 0 \mid a.P \mid P + Q \ (P|Q) \ (v\alpha)P \end{aligned}$$

Dans CCS on a trois types d'actions :

•a(x) :sur le canal a on peut recevoir n'importe quelle valeur,

• $\overline{a}$.v : émettre la valeur v sur le canal a,

• τ :action interne ou silencieuse.

Un processus P peut être soit le processus inerte 0, soit un processus préfixé par une action a, soit une composition parallèle de processus $P|Q$, ou une restriction c'est-à-dire seul le processus P a accès sur le nom x.[T.M16]

Il y a 5 opérations déférent pour la constructions d'un processus(agent) :

1-Préfixage (action prefixing) :

Si α est une action et P est un processus alors $\alpha.P$ est un processus. On peut dire que P est actif après l'affectation de α.

Le préfixe est la seule action possible dans le système :

$$\alpha.P \xrightarrow{\alpha} P$$

2-Opération alternative(choice operator) :

Si P et Q deux processus alors $P+Q$ est un processus. Il y a deux règles symétriques possibles :

$$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$$

3-Composition parallèle (parallel composition) :

P et Q deux processus, (P|Q) ainsi est un processus. La sémantique de cette opération est :

• si $P \xrightarrow{\alpha} P'$, et Q ne performe pas $\overline{\alpha}$ alors : $P|Q \xrightarrow{\alpha} P'|Q$. On note :

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

• si $Q \xrightarrow{\alpha} Q'$, et P ne performe pas $\overline{\alpha}$ alors : $P|Q \xrightarrow{\alpha} P|Q'$. On note :

$$\frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

4-Restrivtion :

Si P est un processus, L est un ensemble visible des actions, $\tau \notin L$, alors $P \setminus L$ est un processus.

Si $P \xrightarrow{\alpha} P'$ et $\alpha \notin L, \overline{\alpha} \notin L$, alors $(P \setminus L) \xrightarrow{\alpha} (P \setminus L)$. On note :

$$\frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \qquad \alpha, \overline{\alpha} \notin L$$

5-Synchronisation(communication) :

On a deux processus CCS, P et Q, le processus $P|Q$ représente un système telque :
− Les deux processus P et Q peuvent communiquer via des portes.
− P et Q peuvent procéder indépendamment.

Deux processus P, Q peuvent communiquer ssi :

$$\frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

On représente les 5 règles (opération) dans la FIGURE 2.2 :

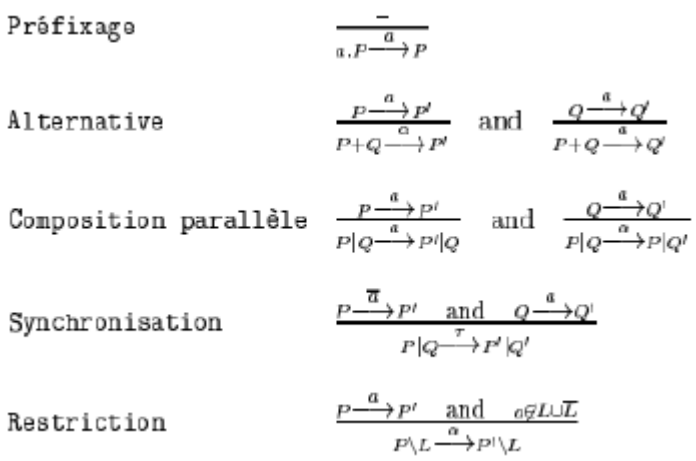

FIGURE 2.2 – Sémantique opérationnelle de CCS

2.2.3 LOTOS : Language Of Temporal Ordering Specification

LOTOS[ISO88][TE87] est une technique de description formelle, développée comme une norme internationale. LOTOS a été développé par les experts de FDTs(Formal Description Techniques) de l'ISO(International Organisation for Standardisation) durant les années 1981-1986[BE87]. L'idée de base est que LOTOS est développé pour exprimer la structure logique et temporelle de comportements. Cette technique de description est basée aussi sur la notion d'algèbre de processus.

Il y a deux volets pris en considération dans LOTOS : Données et Contrôle. Le premier volet décrit les données comme étant des types abstraits algébriques, la syntaxe et la sémantique de ces données étant exprimées dans le langage ACT-ON. la partie contrôle décrit le comportement de la spécification (Basic LOTOS), la syntaxe et la sémantique de cette partie étant largement inspirées des deux algèbres de processus CCS de MILNER (étendu par un mécanisme de synchronisation multiple hérité de CSP[C.A85] de HOARE).

La technique de description formelle LOTOS est un langage dont l'objectif est d'assurer la production de spécifications standards pour les systèmes de norme OSI (Open System Interconnections).

En général, on peut représenter un processus par plusieurs sous-processus,

ces processus peuvent interagir entre eux à travers des ports de communications (points d'interaction), ces interactions sont réalisées par des événements (des unités de synchronisations).Un événement est une action qui peut être observable ou non observable par son environnement.

Dans LOTOS la représentation des processus et de la forme suivante : le processus peuvent être considérés comme des boites noires qui ont des ports (les points d'interaction) pour communiquer de rendez-vous (voir FIGURE 2.3)

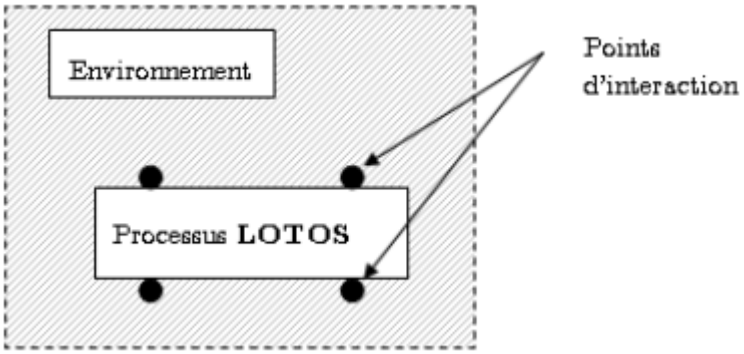

FIGURE 2.3 – Representation abstrait d'un processus LOTOS

Présentation de Basic LOTOS

Basic-LOTOS est un sous ensemble de LOTOS où les processus interagissent entre eux par synchronisation pur, sans échange de valeurs[T.M16].En Basic-LOTOS les actions sont identiques aux points d'interaction (portes de synchronisation) des processus.

Opérateurs du langage :

Soient P et Q deux processus et un sous ensemble de portes $L \subseteq G$:

• Inaction *Stop* : est une expression qui représente un processus inactif (Processus de base n'interagissant pas avec son environnement).

• Terminaison avec succès *exit* : est un processus qui se terminer, et qui se transformer en le processus de base.

• Préfixage $a;P$: représente le processus qui réalise l'action a et qui se comporte ensuite comme P.

• Intériorisation $hide \;\; L \;\; in \;\; P$: tout les interactions sur les portes de L sont rendues invisibles (cachées) à l'environnement de P .

• Choix : l'expression $P[]Q$ représente le processus qui se comporte (transforme) soit comme P ou comme Q suivant l'environnement.

• Composition parallèle : l'exprission $P|[]|Q$ représente la composition parallèle de P et Q (dans le cas générale) avec synchronisation sur les portes qui sont dans L. Il y a deux cas particuliers de composition parallèle :

$*P|||Q$: représente le cas où l'ensemble des portes de synchronisation L est vide (exécution en parallèle sans synchronisation).

$*P||Q$: qui représent que l'exécution de P et Q est en parallèle et se synchronisent sur chaque porte visible.

• Composition séquentielle (Séquencement de processus) : l'expression $P >> Q$ dénote un processus qui dès que P s'est terminé avec sucés, se comporte comme Q.

• Interruption (Préemption) $P[> Q$: tant que P ne s'est pas terminé avec seccés, peut être interrompu par Q.

Syntaxe de Bsic-LOTOS

• Soit P l'ensemble des identifiants (variables)de processus.

• Soit $X \in P$.

• Soit G l'ensemble des noms de ports définissable (ensemble des actions observables en Basic-LOTOS).

• Soit $g, g_1, ..., g_n \in G$.

• Soit L un sous ensemble quelconque de G (pouvent être vide) noté $L = g_1, ..., g_n$.

• l'action interne est désignée par i.

La syntaxe formalle du Basic-LOTOS est donnée par :

$$
\begin{array}{c}
P ::= \text{stop} \\
|\text{exit} \\
|P[L] \\
|g; P \\
|i; P \\
|P[]P \\
|P|[L]|P \\
|\ \text{hide } L \text{ in } P \\
|P >> P \\
|P[> P
\end{array}
$$

Sémantique opérationnelle de Basic-LOTOS

Nous donnons dans ce qui suit le Tableau 2.1(page 10) qui représente la sémantique opérationnelle de Basic-LOTOS.

Les notations suivantes seront utilisées :

•δ est l'action de terminaison de Basic-LOTOS.

•$G^i = G \cup \{i\}, G^\delta, G \cup \{i, \delta\}$.

•$P \xrightarrow{g} P'$, signifier que le processus P peut réalise l'action g et se compote comme P'.

$\text{exit} \xrightarrow{\delta} \text{stop}$
$g; P \xrightarrow{g} P \; g \in G$
$i; P \xrightarrow{i} P$
$\frac{P \xrightarrow{g} P'}{P[]Q \xrightarrow{g} P'} \; (g \in G^{i,\delta})$
$\frac{P \xrightarrow{g} P' Q \xrightarrow{g} Q'}{P\|[L]\|Q \xrightarrow{g} P'\|[L]\|Q'} \; (g \in L \cup \delta)$ $\frac{P \xrightarrow{g} P'}{P\|[L]\|Q \xrightarrow{g} P'\|[L]\|Q} \; (g \in G^i \setminus \{\delta\})$
$\frac{P \xrightarrow{g} P' \; (g \in G^{i,\delta} \setminus \{L\})}{hide \; L \in P \xrightarrow{g} hide \; L \in P'}$ $\frac{P \xrightarrow{g} P' \; (g \in L)}{hide \in P \xrightarrow{i} hide \; L \in P'}$
$\frac{P \xrightarrow{g} P'}{P >> Q \xrightarrow{g} P' >> Q} \; (g \in G^i)$ $\frac{P \xrightarrow{\delta} P'}{P >> Q \xrightarrow{i} Q}$
$\frac{P \xrightarrow{g} P'}{P [> Q \xrightarrow{g} P' [> Q} \; (g \in G^i)$ $\frac{Q \xrightarrow{g} Q'}{P [> Q \xrightarrow{g} Q'} \; (g \in G^{i,\delta})$ $\frac{P \xrightarrow{\delta} P'}{P [> Q \xrightarrow{\delta} Q})$
$\frac{PX[g_1/g_{1'} \ldots g_n/g_{n'}] \xrightarrow{g} Q'}{X[g_1 \ldots g_n] = PX} X[g_1 \ldots g_n] \xrightarrow{g} Q'$ $\frac{P \xrightarrow{g} P' \; (\phi = [g_1/g_{1'} \ldots g_n/g_{n'}])}{P_\phi \overset{\phi\{g\}}{\rightarrow} P'_\phi \; (g \in G^{i,\delta})}$

TABLE 2.1 – Sémantique opérationnelle de Basic-LOTOS

Présentation de full LOTOS

Full LOTOS (ou simplement LOTOS), est Basic-LOTOS étendu avec la possibilité d'échanger des valeurs lors des synchronisations entre processus. Contrairement à Basic-LOTOS, en full LOTOS une action n'est pas simplement une porte de synchronisation, mais une porte plus des données échangées lors de la synchronisation.

Les structures de données et les opérations associées sont définies par le langage Act-One qui formalise la spécification de types abstraits de données, il définit les propriétés essentielles des données et les opérations qu'une implémentation correcte du type doit assurer.

les valeurs en LOTOS peuvent être :

• Echangées entre processus lors d'une synchronisation,

• Employées dans les prédicats de garde des processus,

• Utilisées comme paramètres pour la définition des processus et pour l'instanciation de valeurs,

• Associées à l'opérateur de choix généralisé,

• Exportées lors d'une terminaison.

Avec l'ajout des données dans la spécification, les actions deviennent des entités qui adjoignent à une porte trois composantes de base, comme cela est illustré dans le tableau 1.3. Un action peut :

• Offrir une valeur x (!x),

• Accepter une valeur y(?y :type),

• Incorporer des prédicats qui conditionnent l'acceptation de valeurs.

P1	**P2**	**Condition de synchronisation**	**Type d'interaction**	**Résultat**
$g!v_1$	$g!v_2$	valeur (v_1) = valeur (v_2)	concordance des valeurs	synchronisation
$g!v$	$g?x:T$	$v \in T$	passage de valeur	après synchronisation, $x = valeur\,(v)$
$g?x:T$	$g?y:U$	$T = U$	choix aléatoire d'une valeur	après synchronisation $x = y = v$ avec $v \epsilon T\,(ouv \in U)$

TABLE 2.2 – Règles de synchronisation en LOTOS avec données

Le Tableau 2.2, illustre les nouvelles règles en LOTOS avec données de deux processus P1 et P2 composés par une synchronisation sur la porte g. La règle veut qu'après une synchronisation, les actions impliquées dans cette synchronisation doivent offrir des variables de même type et de même valeur.

Le format de définition d'un processus

$$Proess < processus - identifier >< pramater - list >:=< behaviour - expression > EndProcProc$$

Où :

$< processus - identifier >$: Identifie le nom de processus.

$< pramater - list >$: Qualifie la liste des portes par les quelles les événements peuvent apparaitre.

$< behaviour - expression >$: Définie le comportement observable (expression de comportement du processus).

Les principaux opérateurs LOTOS avec utilisation de données :

• composition séquentielle $exit\ (v_1, ..., v_m)\ >>\ accept\ x_1 : T_1, ..., x_m : T_m i P$: le processus qui a terminé avec succès transmit des valeurs $v_1, ..., v_m$ au processus suivant qui peut les consulter à travers les variables $x_1 : T_1, ..., x_m : T_m$ respectivement.

• choix généralisé $choice\ x : Type[]P(x)$: si P dépend des variables x, l'opérateur offre le choix entre les processus pour toutes les valeurs possibles de $x \in Type$.

• Déclaration de variable $let\ x : Type = v\ in\ P$ réalise l'instanciation de la variable x avec la valeur v dans P.

• Prédicat de garde $[pred] - | > P$: le processus aura le comportement de$P$ si pred et vrai, sinon il devient $Stop$.

x correspond à un nom de variable, $Type$ à un type de données et un prédicat de garde($pred$ à une expression logique), $v_1, ..., v_m$ une liste de valeurs.

Sémantique opérationnelle de LOTOS

La sémantique opérationnelle de LOTOS est donner dans le Tableau 2.3.

$\text{exit}(v_1, ..., v_m) \overset{\delta!(valeur(v_1)...valeur(v_n))}{\rightarrow} \text{Stop}$
$g\,!v; P \overset{g:(valeur(v))}{\rightarrow} P\ (g \in G)$ $g\,?:y; P \overset{g?(p)}{\rightarrow} let\ x:T=p\ in\ P(g \in G)(p \in T)$
$\dfrac{P \overset{g}{\rightarrow} P'}{let\ x{:}T{=}v\ in\ P \overset{g}{\rightarrow} let\ x{:}T{=}v\ in\ P'}$ $\dfrac{[v/x]P \overset{g}{\rightarrow} P'}{let\ x{:}T{=}v\ in\ P \overset{g}{\rightarrow} P'}$
$\dfrac{P \overset{g}{\rightarrow} P'\ \ valeur(v){=}true}{[v]{->}P \overset{g}{\rightarrow} P'}$
$\dfrac{P_X[g_1/g'_1...g_n/g'_n](valeur(v_1)/y_1...valeur(v_n)/y_n) \overset{g}{\rightarrow} Q'X[g_1...g_n](y_1{:}T_1...y_n{:}T_n){=}P_X}{X[g_1...g_n](valeur(v_1)...valeur(v_n)) \overset{g}{\rightarrow} Q'}$

TABLE 2.3 – Sémantique opérationnelle de LOTOS

2.2.4 Les modèles des structures d'événements(SE)

Les modèles des structures d'événements sont définis à partir d'un ensemble d'événements et de relation entre ces événements.

Evénement

Chaque événement est étiqueté par le nom d'une action, et représente une occurrence de cette action à un instant donné. De ce ait, un événement apparaît au plus une fois dans une structure d'événements.

les événements sont notés par $e_1, e_2, ...$, et les noms des actions (étiquettes) sont notés par $a, b, ...$ De façon schématique, les événements seront représenté par des points, leur nom étant écrit justes à côté.

Relation entre événements

On distingue trois types de relations, qui sont respectivement :

- La relation de **précédence** (relation de **causalité**).
- La relation de **conflit**.
- La relation de **d'indépendance** (relation de **parallélisme**).

Relation de précédence : cette relation définit un ordre partiel des événements, elle concrétisée par le mécanisme des horloges logique. La relation de causalité est une relation binaire noté "$\leq$", elle représentée par une flèche. Dans l'exemple $a\bullet \longrightarrow \bullet b$ (notée aussi $e_1 \longrightarrow^{\bullet} e_2$), l'occurrence de a précédé

l'occurrence de b, on dit aussi que l'occurrence da a cause l'occurrence de b. Donc si une exécution contient l'occurrence de b, elle contient nécessairement l'occurrence de a.

Relation de conflit : est une relation binaire symétrique notée $\sharp$ et représenté par un segment de droite en pointilles. L'exemple $a \bullet \cdots \bullet b$ représente un comportement dans lequel les occurrences des actions a et b sont mutuellement exclusives.

Relation de d'indépendance : c'est une relation binaire symétrique. Cette relation est simplement indiquée par l'absence de causalité (précédence) et de conflit entre événements. $a \bullet \quad \bullet b$ est une exemple qui signifie que les deux actions a et b son indépendants (concurrents ou non comparables). Si les deux actions sont sensibilisées, elles peuvent alors s'exécuter dans n'importe quel ordre ou simultanément.

Les exemples illustrant l'interruption de quelques expressions de comportement à la LOTOS en termes de structures d'événements donnée dans la FIGURE 3.4

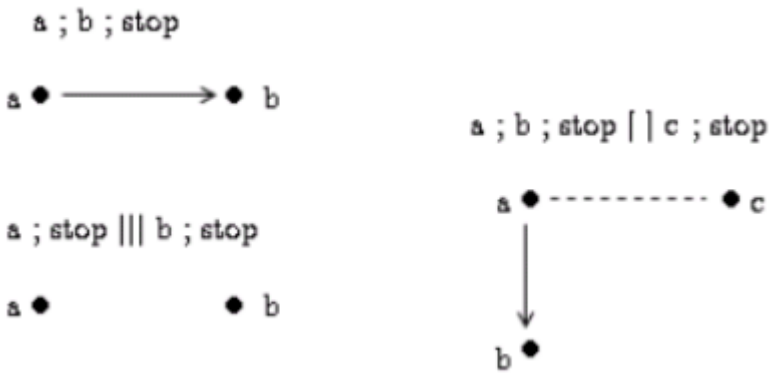

FIGURE 2.4 – De Basic LOTOS aux structures d'événements

Il y a différents types de structures d'événements (le lecteur est invité à voir [R.L92])

Structures d'événements Primaires(SEP)

Définition : Une SEP étiquetée est un quadruplet $\epsilon = (E, \sharp, \leq, l')$ avec :

• E est l'ensemble des événements.

• $\sharp \subseteq E \times E$ (relation de causalité) est une relation irréflexive et symétrique.

• $\leq \subseteq E \times E$ (relation de causalité) est un ordre partiel.

• $l : E \longmapsto Act$ est la fonction (relation) d'étiquetage qui satisfait :

1. finitude des causes : $\forall e \in E$, l'ensemble$\{e' \in E : e' \leq e\}$ est fini.

2. héritage du conflit : $\forall e, e', e" \in E, e\sharp e' \wedge e' \leq e" \Longrightarrow e\sharp e"$.

La propriété de l'héritage du conflit implique que si deux événements sont en conflit alors tous les successeurs de l'un des événements (par la relation de précédence) sont en conflit avec les successeurs de l'autre événement. Donc un événement n'est sensibilisé que par un seul et un seul ensemble d'événements [G.W89].

Structures d'événements à flux (SEF)

Définition : Une SEF étiquetée est un quadruplet $\epsilon = (E, \sharp, <, l)$ avec :

• E est un ensemble dénombrable d'événements.

• $\sharp \subseteq E \times E \; E \times E$ (relation de conflit) est une relation symétrique.

• $< \subseteq E \times E$(relation de flux) est une relation irréflexive.

• $l : E \longmapsto Act$ est la fonction d'étiquetage.

Structures d'événements stables (SES) Les structures d'événement stables sont définies par Winskel dans [Win82][G.W89].

Définition : une SES est un quadruplet $\epsilon = (E, \sharp, ++, l)$ avec :

- E est un ensemble d'événements.

- $\sharp \subseteq E \times E$ (relation de conflit).

- $++ \longmapsto \subseteq 2^{E \times E}$ est la relation de sensibilisation.

- $l : E \longmapsto Act$ est la relation d'étiquetage qui satisfait :

  1.cohérence : $(F++e \Longrightarrow F \cup \{e\})$ ne contient pas de conflits,

  2.stabilité : $(F++e \wedge G++e) \Longrightarrow (FG$ contient de conflits ou bien $F = G)$.

Structures d'événements à "Bundle" (SEB)

Un bundle est un couple (X, e) avec $X \subseteq E, e \in E$, on note également $Xt \longmapsto e$, X est l'ensemble des conditions de causalité de e. On dit qu'un événement e est sensibilise si pour tout bundle $Xt \longmapsto e$ il existe $e' \in X$ qui est déjà produit.

Définition : Une SEB est un quadruplet $\epsilon = (E, ++, \longmapsto, l)$ avec :

- E est un ensemble dénombrable d'événements.

- $++ \subseteq E \times E$est une relation irréflexive et symétrique.

- $\longmapsto \subseteq 2^{E \times E}$ est l'ensemble des bundles.

- $l : E \longrightarrow Act$ est la fonction d'étiquetage qui satisfait :

  1.$X \longmapsto e \Longrightarrow \forall e_1, e_2 \in X, (e_1 <> e_2 \Longrightarrow e_1 ++ e_2)$.

2.3 Les modèles basés réseaux de Petri

Les réseaux de Ptri ont été créés par Carl ADAM PETRI en 1962 afin de modéliser la composition entre automates. Le 1^{er} modèle, appelé Réseaux de Petri Conditions-Evenements (RdP CE) basé (reposait) sur l'utilisation des valeurs booléennes, vrai (true)et faux (false). Le 2^{eme} modèle, appelé Réseaux de Petri Places-Transitions (RdP PT) généralisent les RdP CE (Petri

Conditions-Evenements) en manipulant des valeurs entières.

2.3.1 Les familles de RdP

Il y a trois grandes familles peuvent distinguer dans la FIGURE 2.5 :

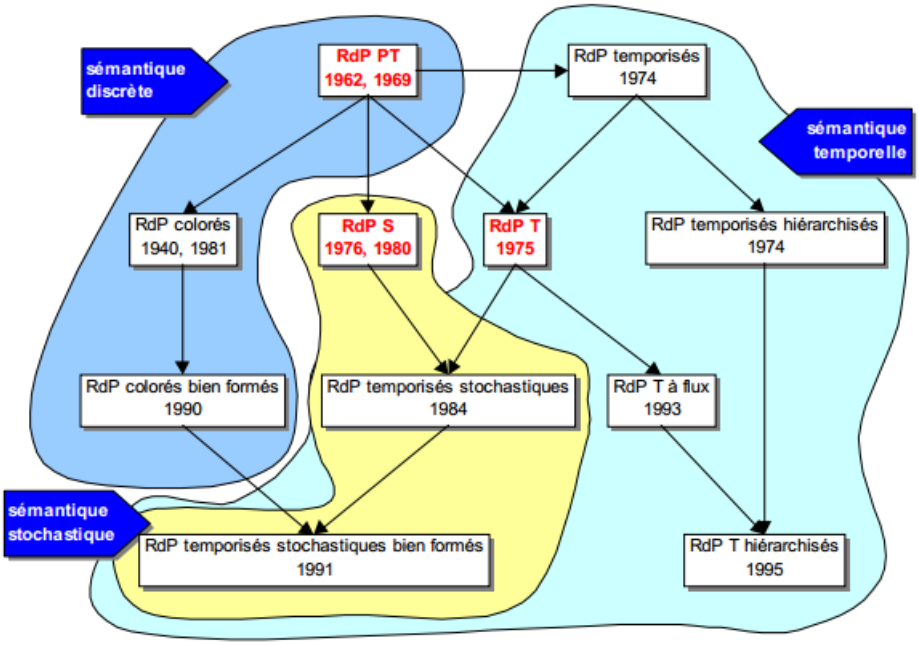

FIGURE 2.5 – Les familles de RdP

Les trois grandes familles correspondant à trois domaines sémantiques :

• ***La sémantique discrète (RdR PT et ses dérivés)*** : pour les comportements qui peuvent se représenter par des graphes finis ou dénombrables d'états,

• ***La sémantique stochastique (RdP S et ses dérivés)*** : pour les comportements qui incluent des distributions de franchissement et conduisent à des processus stochastique (chaîn de Markov, ...etc),

• ***La sémantique continue ou temporelle*** : pour les comportements qui nécessitent la pris en considération explicite d'un temps dense.

2.3.2 Concepts et définitions

Réseau de Petri(RdP), est un graphe orienté comprenent deux types de sommets :

∗ Les places (noté graphiquement par des cercles),

∗ Les Transitions (notes graphiquement par des barres ou des rectangles).

Ils sont reliés par des arcs orientés (des flèches qui joignent les places aux transitions et les transitions aux places). Un arc relie soit une place à une transitions, soit une transition à une place **jamais** une place à une place ou une transition à une transition. Ils consistent ainsi des jetons (distribution de jetons dans les places).

On a un exemple sur la représentations de RdP dans la FIGURE2.6 :

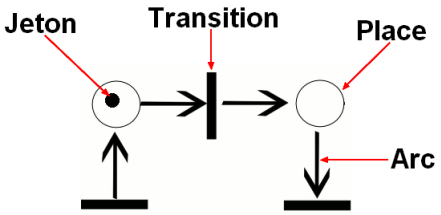

FIGURE 2.6 – représentations graphique de RdP

On peut maintenant donner une définition plus formelle des RdPs.

Définition : Un réseau de Petri est un quadruplet(P, T, F, W) tel que :

• P : est l'ensemble de places,

• T : est l'ensemble de transition,

- F : est une relation de flot avec $\subseteq (P \times T) \cup (T \times P)$,

- $W : ((P \times T) \cup (T \times P)) \longrightarrow N = \{0, 1, 2, 3, ...\}$: est la fonction de poids.

A chaque place et transitions, un nom peut être associé par exemple sur le RdP de la FIGURE 2.7, les places sont nommées P_1, P_2, P_3 et les transitions t_1, t_2, t_3, t_4. $t_1$est reliée a P_1 par un arc orienté de t_1 vers P_1, on dit que P_1 est en sortie de t_1. P_1 est reliée a t_2 par un arc orienté de P_1 vers t_2, on dit que t_2 est en sortie de P_1. De même on peut dire que P_1 est en entrée de t_2. La place P_2 est en entrée de t_1.

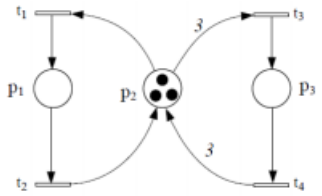

FIGURE 2.7 – Exemple de RdP

2.3.3 Cas particuliers

Transition source

Définition : Une transition t d'un réseau de Petri est une transition source si et seulement si $^\bullet$t est l'ensemble vide.

Une transition source n'a pas de place d'entrée de la transition.

FIGURE 2.8 – Transition source

Transition puits

Définition : Une transition t d'un réseau de Petri est une transition puits si et seulement si $t^\bullet$ est l'ensemble vide.

Une transition puits n'a pas de place en sortie de la transition.

FIGURE 2.9 – Transition puis

2.3.4 Noeds d'entrée et de sortie

• L'ensemble des transition d'entrée d'une place $p \in P$ de réseaux de Petri est l'ensemble des transitions t telles que $(t,p) \in F :^\bullet p = \{t, (t,p) \in F\}$.

• L'ensemble des transition de sortie d'une place $p \in P$ de réseaux de Petri est l'ensemble des transitions t telles que $(p,t) \in F : p^\bullet = \{t, (p,t) \in F\}$.

• L'ensemble des places d'entrée d'une transition $t \in T$ de réseaux de Petri est l'ensemble des places p telles que $(p,t) \in F :^\bullet t = \{p, (p,t) \in F\}$.

• L'ensemble des places de sortie d'une transition $t \in T$ de réseaux de Petri est l'ensemble des places p telles que $(t,p) \in F : t^\bullet = \{p, (t,p) \in F\}$.

2.3.5 Boucle élémentaire

Définition : Une boucle élémentaire est définie par la présence d'une place p et d'une transition t telles que $p \in^\bullet t$ et $p \in t^\bullet$

2.3.6 Comportement dynamique d'un réseau de Petri

Le comportement dynamique d'un réseau de Petri c'est-à-dire son évolution au cours du temps et de l'occurrence des événements (on dit sa sémantique opérationnelle) est spécifiée par les définitions suivantes[T.M16].

Définition : Le marquage d'un réseaux de Petri(P, T, F, W) est défini par une fonction $M : P \longrightarrow N$. Un marquage est généralement représenté par la déposition de jetons dans les places. Le nombre 0 correspond à l'absence de jeton, c'est-à-dire à une place vide.

Définition : un réseau de Petri marqué (P, T, F, W, M_0) est un réseau de Petri (P, T, F, W) avec un marquage initial M_0.

On appelle marquage M d'un Réseau de Petri le vecteur du nombre de marques dans chaque place : la i^{eme} composante correspond au nombre de marques dans la i^{eme} place. On appelle marquage initial, noté $M-0$, le marquage à l'instant initial $(t = 0)$.

2.3.7 Transition validée

Définition Une transition est dite validée si toutes les places en amont (c'est-à-dire en entrée) de celle-ci possèdent au moins une marque. Une transition source est par définition toujours validée.

2.3.8 Franchissement ou tir de transition

Si la transition est validée, on peut effectuer le franchissement de cette transition : on dit alors que la transition est franchissable. Le franchissement consiste à :

- retirer une marque dans chacune des places en entrée de la transition,
- ajouter une marque à chacune des places en sortie.

2.4 Conclusion

Nous avons introduit dans ce chapitre, une présentation générale sur des modèles de spécification formelle, nous avons présenté : les modèles des algèbres de processus comme CCS et les techniques de description formelle (Basic-LOTOS, full LOTOS ou simplement LOTOS), les modèles des structures d'événements les (SEP, SEF, SES...) et les réseaux de Petri.

Dans le chapitre suivant nous allons d'avoir les systèmes de transitions et nous formalise le modèle de système de transitions étiquetées maximales qui basé sur les systèmes de transitions étiquetés , et autre modèles basé système de transitions étiquetées.

Chapitre 3

Systèmes de transitions

3.1 Introduction

Le comportement d'un système réel peut être représenté par un système de transitions sous certaine hypothèse d'abstraction. Ce système e transition est un ensemble d'états muni d'une relation d'accessibilité entre états et d'une relation d'étiquetage des transitions[T.M16]. Les transitions sont décorées par des étiquettes représentant les actions exécutées, ces étiquettes indiquent le comportement produit durant la transition. Lorsque l'on se trouve dans l'un des états du système de transitions, il est possible de changer d'état en effectuant une action qui étiquette l'une des transitions sortantes de cet état, tandis que les nouds représentent les états du système.

3.2 Systèmes de transitions

Les systèmes de transitions (appelé aussi les automates) sont les principaux modèles utilisés pour spécifier le comportement de systèmes réactifs et/ou temps réel. Ils offrent des mécanismes qui sont à la fois formels et simple à comprendre et à utiliser (surtout lorsqu'ils utilisés sous forme graphique).

Définition : Un système de transitions est un triplé $< S, T, R >$:

- S : ensemble d'états. Peut être fini ou infini,
- $T \subseteq S$: ensemble des transitions,
- $R \subseteq S \times S$: relation (de transitions) entre paires d'états.

$(s, s_0) \in R$ signifie qu'il existe une transition faisant passer le système de l'état s à l'état s_0.

Dans la FIGURE 3.1 on à un exemple sur un système de transition :

$$S = \{s_0, s_1, s_2, s_3, s_4\}$$
$$I = \{s_0\}$$
$$R = \{(s_0, s_0), (s_0, s_1), (s_0, s_2), (s_2, s_3), (s_3, s_4), (s_4, s_3)\}$$

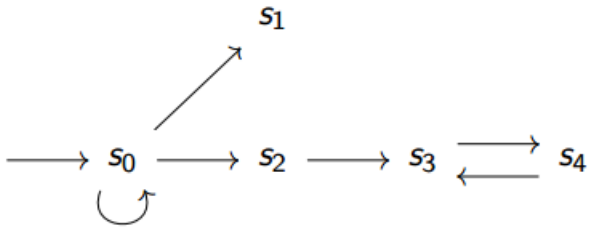

FIGURE 3.1 – Exemple d'un système de transitions

Un système de transitions paramétré en ajoutant des paramètres aux états et au transitions.[Rab04]

Définition : Un système de transitions paramétré par (X, Y) est un système de transitions $< S, I, R >$ les sous-ensembles S_x de S paramètres d'états) et T_y de T (paramètres des transitions) sont définis pour tout $x \in \mathbf{X}$ et tout $y \in \mathbf{Y}$. Où :

- $\mathbf{X} = x_1, ..., x_n$ est un ensemble fini de noms de paramètres d'états,
- $\mathbf{Y} = y_1, ..., y_n$ est un ensemble fini de noms de paramètres de transitions.

Séquence

Définition : Soit S un ensemble :

S^* = L'esemble des séquences finies sur S.

S^w = L'ensemble des séquences infinies sur S.

σ_i = Le i^{eme} (à partir de 0) élément d'une séquence σ.

Conventions de représentation :

• Une séquence s et notée sous la forme : $< s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 ..... >$.

• $\langle\rangle$: La séquene vide.

Pour une Séquence finie σ :

• σ^* =L'ensemble des séquences finies produites par la répétition arbitraire de σ.

• $\sigma^+ = \sigma^* \{\langle\rangle\}$.

• σ^w =La Séquence infinie produite par la répétition infinie de σ

Traces finis

Définition : Soit $\langle S, I, R\rangle$un Système de transitions.
On appelle **trace finie** une Séquence finie $\sigma \in S^*$ telle que :

• $\sigma = \langle s_0 \rightarrow s_1 \rightarrow ..... \rightarrow s_{n-1} \rightarrow s_n\rangle$,

• $\forall i \in [0..n[: (s_i, si + 1) \in R$.

Définition : Soit $\langle S, I, R\rangle$un Système de transitions. Une trace Finnie $\langle s_0 \rightarrow s_1 \rightarrow ..... \rightarrow s_{n-1} \rightarrow s_n\rangle \in S^*$ est **maximale** ssi il n'existe pas d'états successeur à S_n.

$\forall s \in S : (s_n, s) \notin R$

Traces infinies

Definition : Soit $\langle S, I, R\rangle$ un Système de transitions et $s_0 \in S$. On appelle **trace infinie** à partir de s_0 un élément $tr \in S^w$ telle que :

- $tr = \langle s_0 \to s_1 \to s_2 \to .....\rangle$,

- $\forall i \in N : (s_i, si + 1) \in R$.

Définition : Soit $\langle S, I, R\rangle$ un Système de transitions et $s_0 \in S$.

Traces (s) = L'ensemble des traces infinies ou finies **maximales** Commençant à l'etat s

Exécutions

Définition : Soit $\langle S, I, R\rangle$ un Système de transitions. Une **exécution** $\sigma = \langle s_0 \to .....\rangle$ est une trace infinie ou finie **maximal** telle que $s_0 \in I$.

Exec (S) =L'ensemble des exécutions de S.

On a une (seule et unique) exécution vide $\langle\rangle$ ssi $I = \phi$.

Définition : Pour σ une exécution, un **préfixe d'exécution** est une trace finie $\sigma_p = \langle s_0 \to .....s_n\rangle$ telle que $s_0 \in Iet\sigma = \sigma_p \to .....s_n$ est appelé l'état final de ce préfixe.

Dans la FIGURE 3.2 on a un exemple sur les traces :

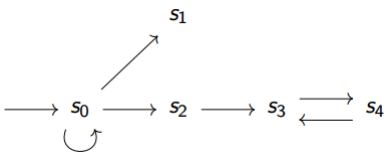

FIGURE 3.2 – Exemple sur les traces

On a $s_0 \to s_0 \to s_2 \to s_3$ est une trace **finite non maximal**.
Traces$(s_1) = \langle s_1\rangle$.
Traces$(s_2) = \langle s_2 \to (s_3 \to s_4)^w\rangle$.
Traces$(s_3) = \langle (s_3 \to s_4)^w\rangle$.
Traces$(s_0) = \langle s_0^w\rangle, \langle s_0^+ \to s_1\rangle, \langle s_0^+ \to s_2 \to (s_3 \to s_4)^w\rangle$.
Traces$(S) = Traces(s_0)$.

3.3 Systèmes de transitions étiquetées

Un système de transitions étiquetées est un graphe dont les arêtes sont étiquetées par une action appartenant à un alphabet.[Jea06]

Les systèmes de transitions étiquetés sont le modèle de référence utilisé pour représenter et comparer des comportements. Ils sont connus dans la littérature sous le nom de LTS (Labeled Transition Systèmes).[R.M76]

Définition1 : Un système de transitions étiquetées par un alphabet A est un 5-typle [Rab04] :

$$A =< S, T, \alpha, \beta, \lambda >$$

Où :

- S est un ensemble fini ou infini d'états,
- T est un ensemble fini ou infini de transitions,
- α et β sont deux mapping de T vers S qui associe à tout transitions t de T deux états $\alpha(t)$ et $\beta(t)$, respectivement la source et la cible de la transition t,
- λ est un mapping de T vers A associant chaque transitions t à son étiquète $\lambda(t)$.

Définition2 : Un système de transitions étiqueté est un quintuplet $\langle S, I, R, L, Etiq \rangle$:

- S : ensemble d'états.
- $I \subseteq s$: ensemble des états initiaux.
- $R \subseteq S \times S$: relation de transitions entre paires d'états.
- L : ensemble d'étiquettes.
- $Etiqu$: fonction qui associe un étiquette à chaque transition : $Etiq \in R \rightarrow L$.

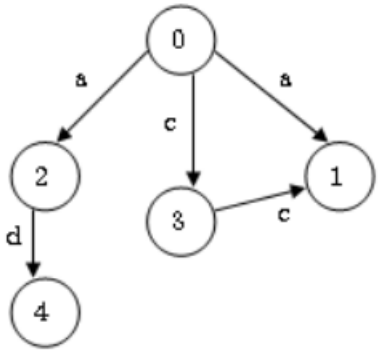

FIGURE 3.3 – représentation graphique d'un système de transition étiqueté

La FIGURE 3.3 représenter un exemple d'un STE. Une transitions ayant pour origine l'état 0, pour but l'état 1 est étiquetée par un symbole a est notée $0 \xrightarrow{a} 1$. Le symbole a représente l'action exécutée durant la transition. Nous notons τ l'action interne ou silencieuse (muette). $0 \xrightarrow{\tau^*} 1$ est un chemin d'un nombre fini de transitions silencieuse, éventuellement aucune.

La notation $0 \stackrel{a}{\Longrightarrow} 1$ pour $a \in$ l'ensemble d'actions équivalant à $0 \xrightarrow{\tau}{}^* \xrightarrow{a} 1$ [Jea06].

l'équivalence entre les systèmes de transitions avec/sans étiquettes

Une système de transitions étiqueté $\langle S, I, R, L, Etiq\rangle$ est équivalent au système sans étiquette $\langle S', I', R'\rangle$défini par :

- $S' = (L \cup \epsilon) \times S$
- $I' = \epsilon \times I$.
- $\{(\langle I, S\rangle), (\langle I', S'\rangle) | (s, s') \in R \wedge I' = etiq(s, s')\}$.

Une transition $s_1 \xrightarrow{a} s_2$ deviant $\langle -, s_1\rangle \rightarrow \langle a, s_2\rangle$, où "-" est n'importe quell etiquette.

La FIGURE 3.4 représente un exemple sur l'équivalence entre les systèmes de transition avec/sans étiquette :

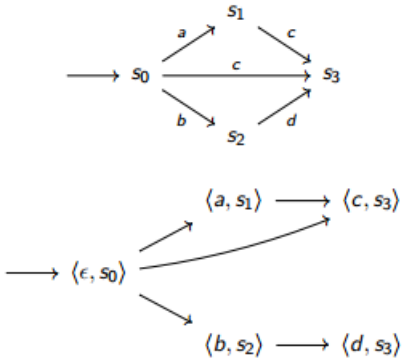

FIGURE 3.4 – l'équivalence entre les systèmes de transition avec/sans étiquette

3.4 Les Systèmes de Transitions Étiquetés Basés sur La Maximalité (STEMs)

Le modèle des systèmes de transitions étiquetées maximales sont des systèmes basé sur les systèmes de transition étiquetées. Les STEMs ne sont rient d'autre qu'une forme plus générale d'arbres maximaux, dans le sens où la structure pourra contenir des cycles.

3.4.1 Le principe du STEMs

Dans un système de transition étiqueté maximale, les transitions sont des événements qui représentent le début d'exécution des actions. La transition est étiquetée par le nome de l'action qui lui correspond. Un état est étiqueté par l'ensemble des actions maximales qui sont les actions en cours d'exécutions. Le début de chaque action est identifier à artire d'un ensemble dénombrable de noms d'éventements M. Il y a plusieurs actions qui ante le même nome peuvent s'exécuter en parallèle (autoconcurrence).

3.4.2 Les arbres maximaux

Dans l'approche basée sur la maximalité, les transitions sont des événements qui ne représente que le début de l'exécution des actions. En consé-

quence, l'exécution concurrente de plusieurs actions devient possible, 'est-à-dire que l'on peut distinguer exécutions séquentielles et exécutions parallèles d'actions.

Or, étant donné que plusieurs actions qui ont le même nom peuvent s'exécuter en parallèle, nous associons pour distinguer les exécutions de chacune des actions, un identificateur à chaque début d'exécution d'action, c'est-à-dire à l'action ou encoure à l'événement associé.

Dans une état, un événement est dit maximal s'il correspond au début de l'exécution d'une action qui peut éventuellement être toujours en train de s'exécuter dans cet état.

L'application de la sémantique de maximalité à des modèles de Petri[R.D92], et plus récemment sur les algèbres de processus[JD95].

Définition : M étant un ensemble dénombrable de noms d'événements, un arbre maximal de support M est un quintuple(T, l, μ, ξ, ψ) avec :

$*(T, l)$ un arbre étiqueté de support L (La définition des arbres étiquetés dans [Sai96]),

$*\mu : T \to 2^M_{fn}$ est une fonction qui associe à chaque transition l'ensemble (fini) des noms des événements correspondant aux actions qui ont commencé leur exécution et dont la terminaison sensibilise cette transition, cet ensemble correspond aux causes direct de la transition,

$*\psi : T \to 2^M_{fn}$ est une fonction qui associe à chaque noud l'ensemble (fini) des noms des événements maximaux présents au niveau de ce noeud,

$*\xi : \Theta(T) \to M$ est une fonction qui associe à chaque transition le nom de l'événement qui identifie son occurrence.

Les résultats attendu du modèle des arbres maximaux est de pouvoir exprimer le comportement des systèmes concurrents, c'est-à-dire représenter les différents états du système, et déterminer pour chacun de ces états les actions qui sont en train de s'exécuter en parallèle.

Définition : M étant un ensemble dénombrable des noms des événements, un système de transitions étiquetées maximale(STEM) de support M est un quintuplait$(\Omega, A, \mu, \xi, \psi)$ avec :

$*\Omega :< S, T, R >$ est un système de transitions,

$* < \Omega, A >$ est un système de transitions étiquetées par un alphabet A.

3.5 Les systèmes de transitions floue

Les systèmes flous sont introduite par Zadeh en 1965, les systèmes flous sont considérant comme des systèmes de transitions flous ou simplement (FTS). Une[Moh12]des directions de recherches principales dans les systèmes flous est de les considérés comme une généralisation des automates non déterministes ou de réseaux de Petri et de les examinés avec le même cadre conceptuel comme les systèmes classiques.

Les systèmes de transitions flou ou simplement FTS (Fuzzy Transition Systems) sont un modèle basé systèmes de transitions étiquetées. Un FTS représenter par un graphe équilibré peut être avec une infinité de sommets ou d'arcs. Un FTS est caractérisé par un ensemble d'états et un ensemble de transitions flous.

Définition : Un système de transitions flous (FTS) est un quadruplet $S = (S, A, \delta, s_0)$ où :

- S est un ensemble fini ou infini d'états,

- A est un ensemble fini ou infini d'étiquettes,

- δ est la relation de transitions floue, de $S \times A$ vers $\boldsymbol{F}(S)$, ou de la façon équivalente une fonction multi-valuée de $S \times A$ vers S,

- $s_0 \in S$ est l'état initial.

On peut dire qu'un FTS est fini si S et A sont fini (les deux), sinon on dit que le FTS est infini. Selon le langage en question, les étiquettes peuvent représenter différents choses.

$\delta(s, \alpha)(s') > 0$ signifie qu'il existe un transition de s vers s' avec l'étiquette $\delta(s, \alpha)(s')$. Un FTS est un généralisation des système de transition étiqueté qui a 0 et 1 comme degrés de possibilités, toute fonction de transition floue

$\delta : S \times A \to F(s)$ ave une relation floue $\to_\delta$: $S \times A \times S \to [0,1]$.[Moh12]

3.5.1 Bisimulation

La bisimulation est une relation binaire $R \subseteq S_1 \times S_2$, noté "$\sim$". On peut dire ainsi que la bisimulation est une relation binaire R sur l'espace d'états de l'automate subjacent.

La bisimulation est une relation binaire entre deux systèmes d'évènement discrets comme les algèbres de processus, les réseaux de Petri ou les modèles d'automates, associant des systèmes qui se comportent de la même manière dans le sens où un système simule un autre et vice versa.[Moh12]

La bisimulation est une relation d'équivalence :

- Reflexive : $S \sim S \forall S$,
- Symétrique : $S_1 \sim S_2 \Rightarrow S_2 \sim S_1$,
- Transitive : Si $S_1 \sim S_2$ et $S_2 \sim S_3$ alors $S_1 \sim S_3$.

On dit que deux systèmes sont bisimulaire ($S_1 \sim S_2$) si le changement de chacun s'accordent avec l'autre.

Définition1 Soit $S = (S, A, \delta, s_0)$ un FTS. Si R est une relation d'équivalence sur S, alors les assertions suivantes sont équivalentes :

1.R est une bisimulation sur S,

2. Pour chaque $(s,t) \in R, \delta(s,a)(C) = \delta(t,a)(C)$ tient pour tout $a \in A$ et $C \in S/R$.

Définition2 Soit $S_1 = (S_1, A, \delta_1, s_{01})$ et $S_2 = (S_2, A, \delta_2, s_{02})$ deux $FTSs$. Une relation $R \subseteq = S_1 \times S_2$ est dite bisimulation entre S_1 et S_2 si pour tous $(s,t) \in R$ et $\alpha \in A$:

1.$s \xrightarrow{\alpha} \mu$ dans S_1 implique $t \xrightarrow{\alpha} \eta$ dans S_2 pour certains $\eta \in F(S2)$ tel que $\mu(U) = \eta(V)$pour chaque paire R-corrélationnelle (U, V),

2.$t \xrightarrow{\alpha} \eta$ dans S_2 implique $s \xrightarrow{\alpha} \mu$ dans S_1 pour certains $\mu \in F(S1)$ tel que $\mu(U) = \eta(V)$ pour chaque paire R-corrélationnelle (U, V).

Dans les systèmes flous, on peut dire que $S_1 \sim S_2$ quand il existe la même action avec la même maximum de possibilité dans chaque paire corrélationnelles [Moh12].

3.6 Les systèmes de transitions étiquetés flous

Toutes les définitions données dans cette section sont prises du travail[Moh12].
Fuzzy labeled Transitions System (FLTS) : On va définir tout d'abord la notion de l'actions flous, une notion clé pour les FLTSs.
Les actions sont sujettent au raffinement, car elles sont partiellement définies. On peut distinguer les actions non-définis $Act_i = \{\perp\}$ et les actions totalement définies (celles ayant un degré de définition égale à un), formellement :

Définition1 L'ensemble des actions flous Act est définit par $Act = \cup_{i=0}^{n} Act_i$ tel que :

$*$ $act_0 = \{\perp\}$,

$*$ $\forall i, j \in \{0, n\}$ et $i < j \mid \kappa_z(Act_i) < \kappa_z(Act_j)$,

$*$ $\kappa_z(Act_i)$ représenter la valeur floue des actions dans Act_i,

$*$ $\kappa_z(Act_n) = 1$.

Définition2 Un système de transition floue (FLTS) est un structure $A = (S, s_0, \kappa_z, \kappa_A)$ tel que :

$*$ S : est une ensemble des états,

$*$ s_0 : est l'état initial,

$*$ $T \subseteq S \times S$: est l'ensemble du transitions,

$*$ κ_z : $Act \longrightarrow [0, 1]$: est une fonction de floue qui associer a chaque ensemble des actions Act_i une valeur dans $[0, 1]$,

$*$ $\kappa_A : T \longrightarrow Act \cup \{\perp\}$: est une fonction de floue qui associer a chaque transition un actions dans $Act \cup \{\perp\}$.

3.7 conclusion

Nous avons présenté dans ce chapitre un survol des modèles de spécification dans l'esprit des systèmes de transitions.

Nous avons abordé les systèmes de transitions étiquetés et les systèmes de transitions étiquetés maximales, on parle sure les systèmes de transitions flous STFs et la bisimulation.

Dans le chapitre suivant, nous allons détailler notre cas d'étude.

Chapitre 4

Implémentation

4.1 Introduction

Ce chapitre est consacré à la réalisation et la mis en oeuvre de notre application qui (**"Implémentation du Model STEs"**), qui consiste en une première phase pour la réalisation d'un compilateur de spécifications LOTOS vers des STEs.

Dans une première section nous allons présenter le langage et les outils utilisés. Dans une deuxième section nous allons présenter les définitions inductives des certains operateurs du langage LOTOS afin de les utiliser comme algorithmes a la réalisation de l'application. Dans une troisième section nous allons présenter notre application en quelques fiches.

4.2 Le langage utilisés et l'outile utilisé

4.2.1 JAVA

JAVA est un langage de programmation moderne développé par Sun Microsystems en 1991 (aujourd'hui racheté par Oracle), dans le cadre d'un projet de développement de logiciel pour des appareils électroniques de grand consommation : télévision, magnétoscopes, ...[LL97].

Java est un langage de programmation orienté objet (POO) qui s'organise en :

- Packages

- Classe
  - Propriétés
  - Méthodes.

4.2.2 Eclipse IDE

Eclipse IDE est un environnement de développement libre permettant de créer des programmes dans de nombreux langages de programmation (Java, C++, PHP.... C'est l'outil que nous allons utiliser pour programmer .

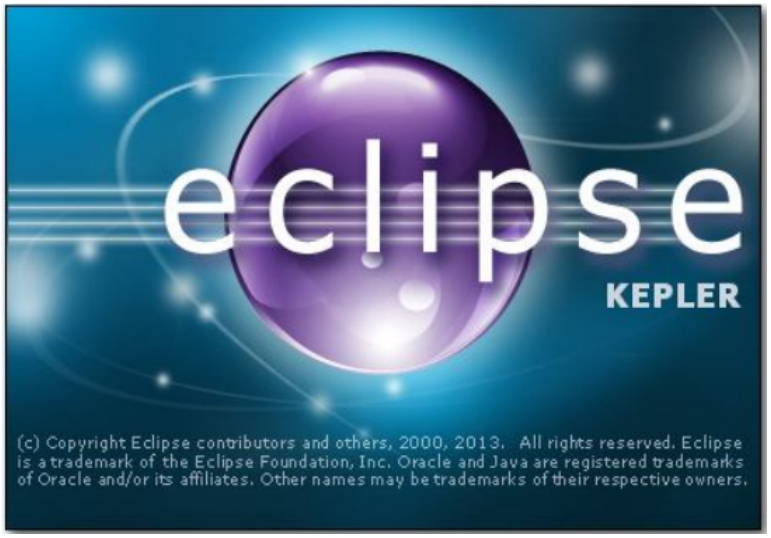

FIGURE 4.1 – eclipse

.

Ecran de bien venue

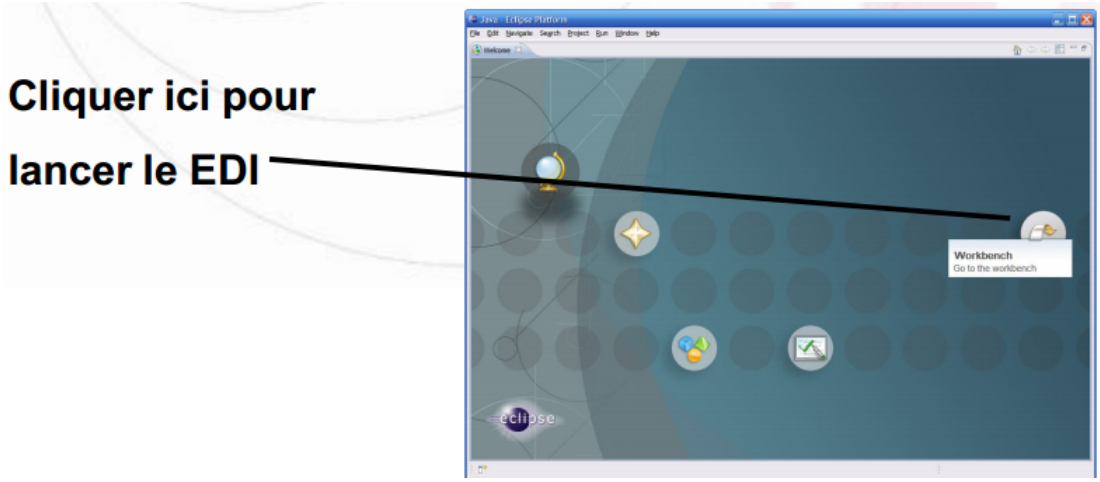

FIGURE 4.2 – Ecran de bien venue dans eclipse

.

EDI

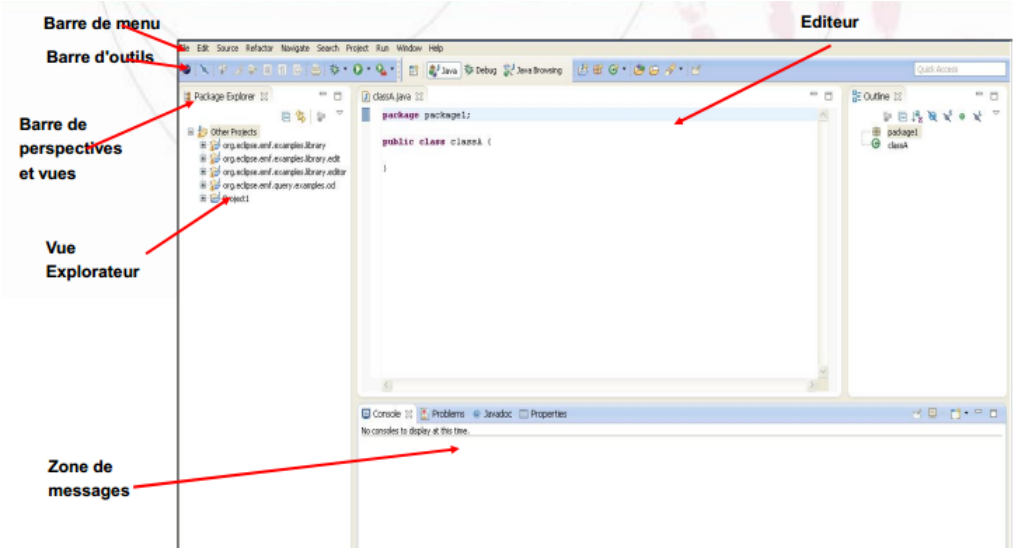

FIGURE 4.3 – Workbench

.

Nouveau projet

Pour crée un nouveau projet on va clique sur :

File ⟶ New ⟶ Java Project.

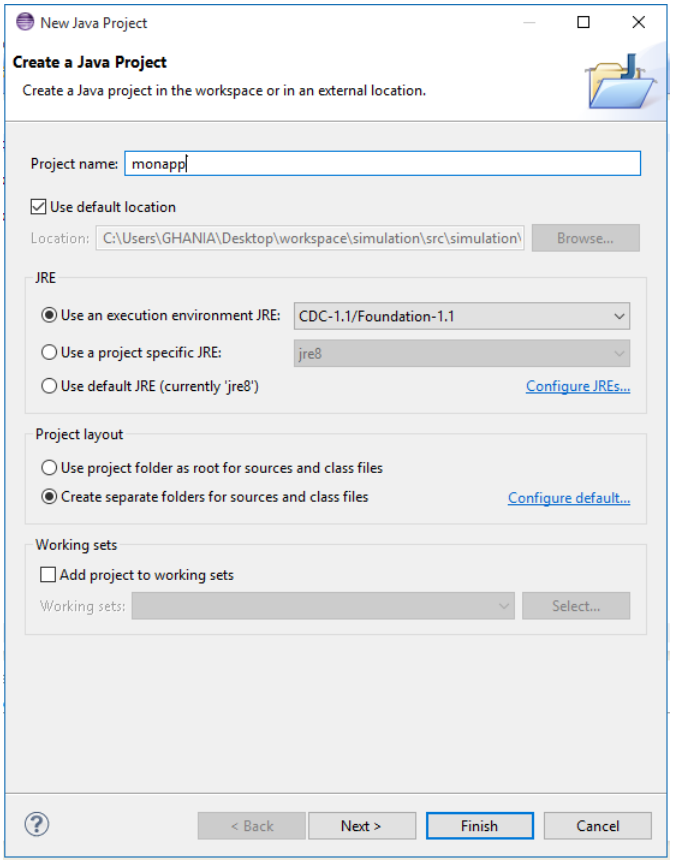

FIGURE 4.4 – nouvelle projet

.

Nouveau classe

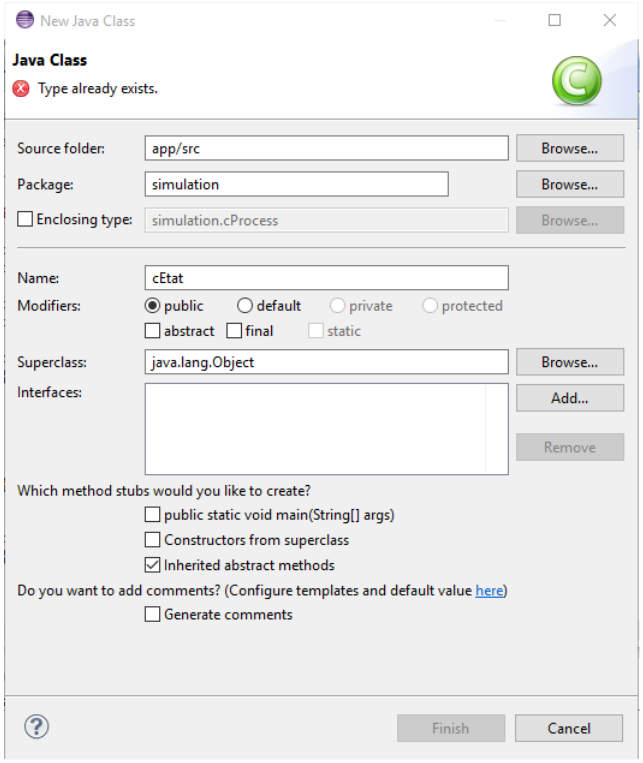

FIGURE 4.5 – la creation de la classe cEtat

.

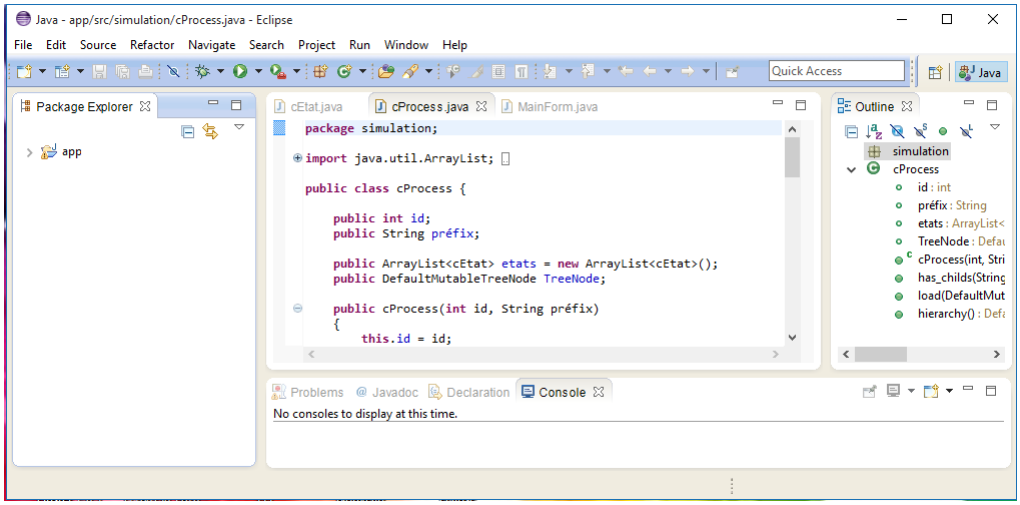

FIGURE 4.6 – la classe cProcess

4.3 Définition des opérateur de base LOTOS

4.3.1 Préfixage par action

Une opération de base celle qui introduit une nouvelle transition au commencement d'un processus. Etant donnée un STE A, l'opérateur de préfixage $a; A$, introduit une transition avec l'étiquette a a partir d'un nouveau état initial à l'état initial intérieur (l'état initial du STEs), concrètement d'un tel opérateur est donnée par :

Définition [MS17] Soit $a \in G$ et $A = (S, l, s_0, T)$ un système de transitions étiquetées de support G. Le préfixage d'un STE A par l'action a, est noté $a; A$, il définis par $A' = (S', l', s'_0, T')$ tel que $s'_0 \notin S$ avec :

1. $T' = T \cup \left\{ s'_0 \xrightarrow{a} s_0 \right\}$ et $G' = \{a\} \cup G$,

2. $S' = S \cup \{s'_0\}$,

3. $l'(s'_0, s0) = a$ et $l'(t) = l(t) \forall t \in T$.

4.3.2 La somme

Dans les algèbres de processus les chois indéterministes permet ou plusieurs processus alternatifs, premièrement on va définir la somme de deux processus en suite définir la somme généraliser.

Considérant que les états d'un processus sont disjoints, sinon on doit procédé a un prétraitement pour renommer les états d'un processus.

Définition1 [MS17] Soit $A_1 = (s^1, l^1, s_0^1, T^1)$ et $A_2 = (s^2, l^2, s_0^2, T^2)$ deux STEs, la somme de $A_1 + A_2$ est définis par :

1. $s_0 = (s_0^1, s_0^2)$,

2. $S = (S^1 - \{s_0^1\}) \cup (S^2 - \{s_0^2\})$,

3. $T = T^1 \cup T^2 - (T_1^1 \cup T_2^2) \cup T'$ avec :

   * $T_1^1 = \{t \in T^1 \mid \alpha(t) = s_0^1\}$;

   * $T_2^2 = \{t \in T^2 \mid \alpha(t) = s_0^2\}$;

   * $T' = (s_0, s) | \exists t \in T_1^1 \cup T_2^2$ avec $\beta(t) = s$.

4. $l(t) = \begin{cases} l^1(t) & if \quad t \in T^1 \\ l^2(t) & if \quad t \in T^2 \\ l^1(s_0^1, s) & if \quad s \in S^1 \quad and \quad (s_0, s) \in T' \\ l^2(s_0^2, s) & if \quad s \in S^2 \quad and \quad (s_0, s) \in T'. \end{cases}$

4.3.3 La généralisation de la somme

Définition1 [MS17] Soit $A_i = (S^i, l^i, s_0^i, T^i)$ un ensemble de STEs indexés par $i \in I$, la somme $\sum_{i \in I}(A_i)$ est définir par :

1. $s_0 = \prod_{i \in I} s_0^i = (s_0^1, s_0^2, ..., s_0^i, ...)$,

2. $S = \cup_{i \in I}(S^i - \{s_0^i\})$,

3. $T = \cup_{i \in I}(T^i) - \cup_{i \in I}(T_i^i) \cup T'$ tel que :

   $T_i^i = \{t \in T^i \mid \alpha(t) s_0^i\}$;

   $T' = (s_0, s) \mid \exists t \in \cup_{i \in I}() T_i^i$ avec $\beta(t) = s$.

4. $l(t) = \begin{cases} l^i(t) & if \quad t \in T^i \\ l^i(s_0^i, s) & if \quad s \in S^i \quad and \quad (s_0, s) \in T'. \end{cases}$.

4.4 Les interfaces de l'application

4.4.1 Système de transition "l'interface principale"

Au lancement de notre application, l'interface principale apparaitra et toutes les fonctionnalités y sont :

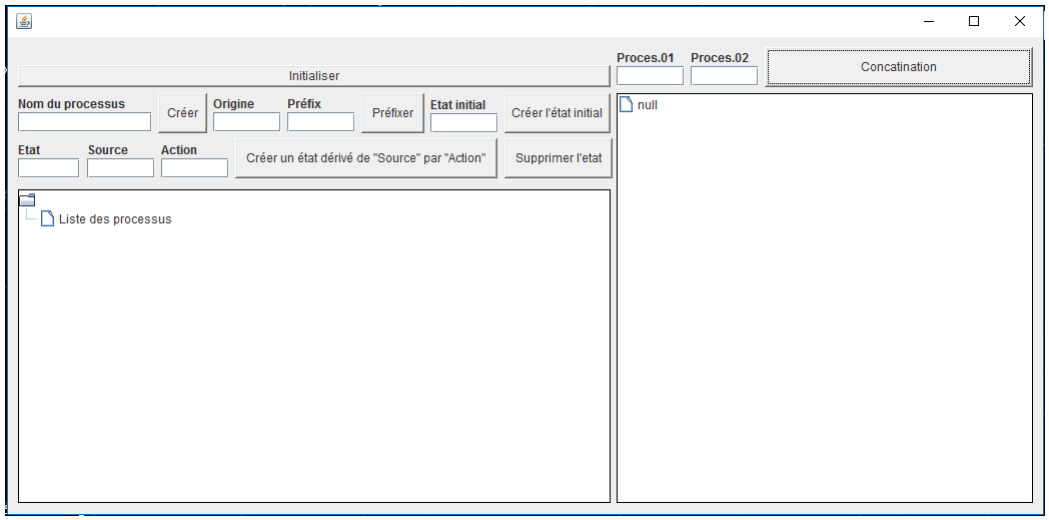

FIGURE 4.7 – Système de transition

.

4.4.2 création d'un processus

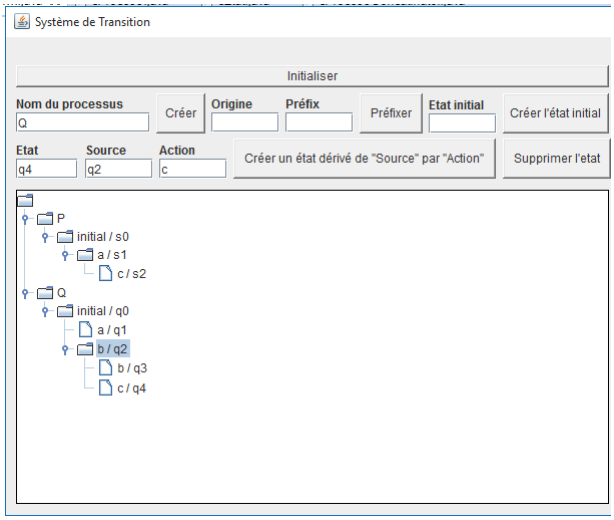

FIGURE 4.8 – création des processus P et Q

.

4.4.3 création d'un état initial

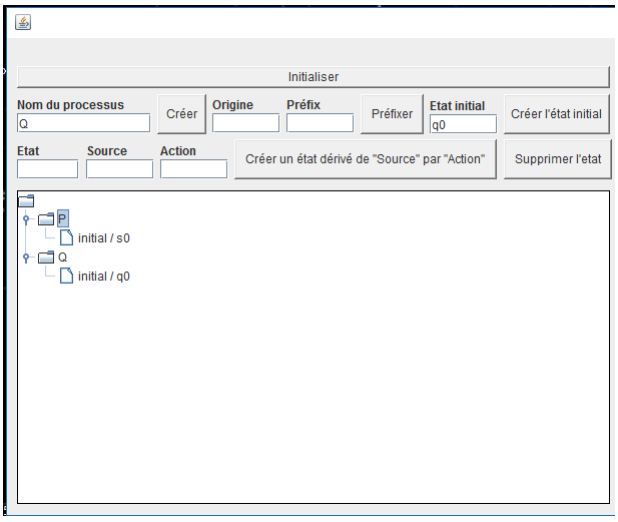

FIGURE 4.9 – Création d'un état initial

.

4.4.4 Opération du préfixage par une action

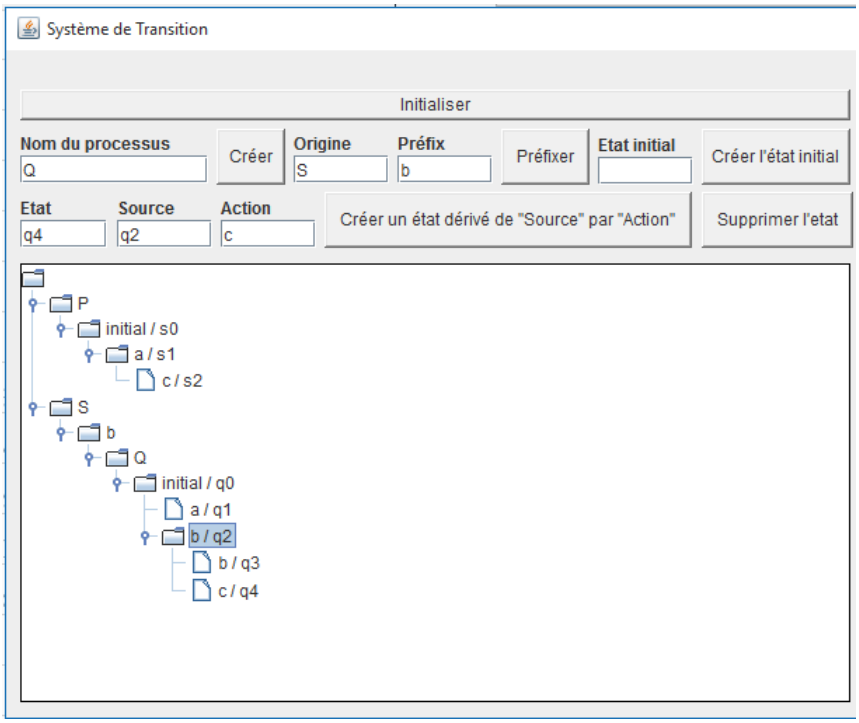

FIGURE 4.10 – préfixage du P par l'action a

.

4.4.5 La somme

.

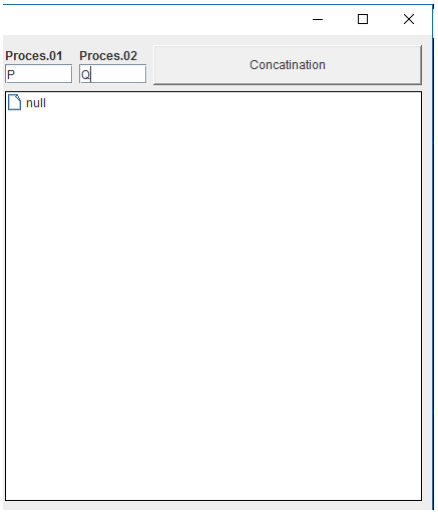

FIGURE 4.11 – la création de deux processus a sommé

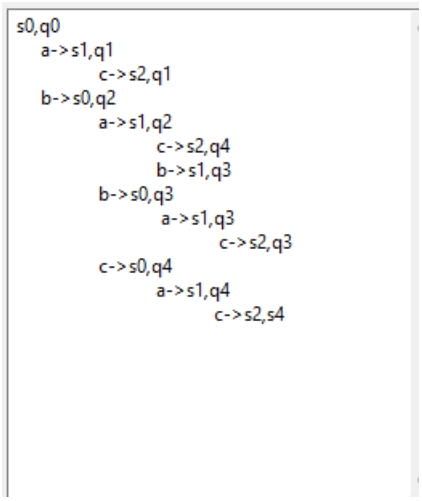

FIGURE 4.12 – Resultat de la somme

4.5 Conclusion

Dans cette dernière partie de notre projet, nous avons présenté différents outils pour le développement de notre application, ainsi que ses interfaces essentielles, tout on passant par les definitions inductives nécessaires, qui servent comme des esquisses d'algorithmes récursives.

Conclusion et perspectives

Au cours de ce mémoire nous avons présenté les différentes étapes qui ont permis la réalisation d'un compilateur de génération des MLTS à partir de spécification LOTOS. Nous avons mis comme objectif préalable de ce mémoire, la vue globale non exhaustive sur l'état de l'art de ce domaine, et on donnant la syntaxe du langage Basic-LOTOS.

Notre mémoire comprend une étude théorique et une autre pratique. Dans le premier chapitre, on a défini le contexte de le mémoire, on a exposé la problématique, de la vérification et montrer l'organisation du document. Dans la deuxième chapitre nous avons présenté des généralités sur les méthodes formelles, nous avons mis en évidence les algèbres de processus, les modèles sémantique pour la vraie concurrence. Dans la troisième partie nous avons présenté les ST et quelque extension. Dans le chapitre quatre a été consacré à l'aspect réalisation de notre outil. Enfin, nous avons conclu par les étapes de la réalisation de le compilateur en expriment les outils de développements utilisés suivis d'une description de notre travail.

Ainsi, il est important de mentionner que nous avons rencontré certaines difficultés de notre travail. En revanche, ce projet a fait l'objet d'une expérience intéressante qui nous a permis d'améliorer nos connaissance ces et nous compétences dans la programmations.

L'objectif du travail a été de définir un compilateur du langage LOTOS vers le modèle des STEs à des fins de vérification, mais nous avons réduit le travail aux opérateurs de bases suscités pour des raisons purement hors de notre volonté, le temps aussi était un facteur primordiale du fait que nous nous pouvions jamais dans une période ci-restreinte atteindre un objectif aussi large qu'il soit.

Perspectives

Le travail présenté dans ce document peut être continué dans plusieurs directions :

Bibliographie

[Bae] •. •, •.

[BE87] Bologonesi and E.Brinksma. *Introduction to the ISO Specification langage.* LOTOS, Computer Networks and ISDN Systems, Vol.14, Pages 25-59, 1987.

[C.A85] C.A.R.Hoare. *Communicating Sequential Processes.* Prentice - Hall, 1985.

[G.W89] G.Winskel. *An Introduction to Event Structures*, volume 354 of LNCS. Pages 364 - 397. Spring - Verlag, 1989.

[ISO88] ISO/IEC. *LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour.* International Standard 8807, International Organisation of Standardization - information Processing Systems - Open Systems Interconnection, Genève, Septembre 1988.

[JD95] J.P.Courtiat and D.E.Saidouni. *Relating Maximality - Based Semantics to Action Refinement in Process Algebra.* In D. Hogrefe and S. Leue Editor, pages 293 - 308, ormal Description Techniques(FORTE'94), Chapman and Hall, 1995.

[Jea06] Jean.Charles.ROGER. *Exploitation de contextes et d'observateurs pour la validation formelle de modèles.* PhD thesis, Rennes 1, 2006.

[JJ85] J.A.Bergstra and J.W.Klop. *Algebra of Communicating Processes with Abstraction.* TCS, 1985.

[LL97] Laura LEMAY and Charles L.PERKINS. *Apprenez Java en 21 Jours.* Simon and Schuster Macmillan, 19, rue Michal le Comte 750003 PARIS, 1997.

[Moh12] KHERGAG Mohamed. Systèmes de transition flous. Technical report, Laboratoire MISC, UNiversité Abdalhamid MEHRI, Constantine, 2012.

[MS17] BEN OTHMANE Mohamed and D.E Saidouni. Cpos characterization of lts and mlts models. Technical report, Université Abbes

Laghrour Khenchela Laboratoire MISC, UNiversité Abdalhamid MEHRI, Constantine, 2017.

[Rab04] Rabéa.AMEUR.BOULIFA. *Génération de Modèles Comportementaux des Application Réparties.* PhD thesis, Université de Nice-Sophia Antipolis, 2004.

[R.D92] R.Devillers. *Maximality Presvation and the ST-idea for Action Refinement, Advances in Petri Nets.* In G, Rozenberg Editors LNCS Vol. 609, pages 108 - 151, Springer-Verlage, 1992.

[R.L92] R.Langerak. *Transformations and Semantics for LOTOS.* PhD thesis, University of Twente, Netherland, 1992.

[R.M76] R.M.Keller. *Formal Verification of Parallel Programs Commun.* ACM, July 1976.

[R.M80] R.Milner. *Acalculus of Communicating Systèms.* Lecture Notes in Computer Science, 92, 1980.

[Sai96] D-E. Saidouni. *Sémantique de Maximalité : Application au Raffinement d'Actions dans LOTOS.* PhD thesis, LAAS-CNRS, 7, Avenue du Colonel Roche, 31077 Toulouse Cedex, France, 1996.

[TE87] T.Bologensi and E.Brinksma. *Introduction to the ISO Specification Langguage LOTOS.* Computer Networks and ISDN Systems, 14 :25-59, 1987.

[T.M16] T.M.Maarouk. Modèles de spécification des systèmes distribués. Technical report, Notes de Cours, 2016.

[Win82] Winskel. *Event Structure Semantics for CCS and Related Languages.* In ICALP 82, volume 140 of LNCS, pages 561 - 576. Springer-Verlage, 1982.