**Algorithme : Composition d’automates**

**Données** deux automates à états finis A1, A2.

**Résultat** un automate produit A.

**Variable**

A1 : Fichier de texte ; */\* représente les états et les transitions de l’automate A1*

 A2 : Fichier de texte ; */\* représente les états et les transitions de l’automate A2*

 A : Fichier de texte ; */\* représente les états et les transitions de l’automate A*

**Variable** i : entier ;

 **Variable** E etat-produit : Ensemble : chaîne de caractère ;

*/\* sauvegarde les états de l’automate produit*

X alph-produit : Ensemble : chaîne de caractère ;

 */\* sauvegarde les alphabets de l’automate produit*

II trans-produit : Ensemble : chaîne de caractère ;

*/\* sauvegarde les transitions de l’automate produit*

L\_ E\_trait : Vector */\*sauvegarde des états déjà traités.*

L\_ N\_etat: Vector */\*sauvegarde des nouveaux états à traités.*

List\_trans\_prod : Vector */\* contient les transitions de l’automate produit A.*

Ens\_etatf\_prod: Vector */\* contient les états finaux de l’automate produit A.*

Etat :(Si,Sj) */\* couple des états tell que Si € E1 et Sj € E2.*

**Début**

Ajouter ((S01, S02), List\_Nouv\_etat) */\* S01 état initial de A1, S02 état initial de A2*

**Tant que** List\_Nouv\_etat <> null **Faire** */\*parcourir la liste des nouveaux états*

Etat ← retirer\_premier\_element (List\_Nouv\_etat), Si ← Etat.Si, Sj ←Etat.Sj ;

Vector List\_trans1 ← Suivant (Si, A1) ;

Vector List\_trans2 ← Suivant (Sj, A2) ;

*/\* Suivant (e, A) retourne toutes les transitions ayant l’état e comme état source dans le fichier A .les transition sont de type (source, alphabet, destination).*

**Pour** chaque transition trans1= Si>alph1> dst1 ∈ List\_trans1 **Faire**

**Si** il existe une transition trans2=Sj>alph1>dest2 ∈ List\_trans2 **alors**

*/\* une transition de List\_trans2 ou on peut lire le même alphabet alph1*

ajouter ((Si,Sj),alph1 ,(dst1,dst2)), liste\_tans\_prod) ;

 supp\_element(List\_trans2,tran2)

*/\* Supprimer la transition trans2 de List\_trans2*

 **Si** (dst1, dst2) n’appartient pas (Etat\_trait U Nouv\_etat) **alors**

Ajouter ((dst1, dst2), Nouv\_etat) ;

**FSi**

**Si** (dst1∈ ens\_etf1 et dst2∈ ens\_etf2) ajouter ((dst1, dst2) ,ens\_etf) ; **FSi**

*/\* ens\_etf1 représente l’ensemble des états finaux dans A1, ens\_etf2 les états finaux dans A2, et ens\_etf dans A.*

**Sinon**

Ajouter ((Si,Sj),alph1 ,(dst1,Sj)),liste\_tans\_prod)

**Si** (dst1, Sj) n’appartient pas (Etat\_trait U Nouv\_etat) **alors**

Ajouter ((dst1, Sj), Nouv\_etat)

**Fsi**

**Si** (dst1∈ ens\_etf1 et Sj∈ ens\_etf2) ajouter ((dst1, Sj) ,ens\_etf); **FSi**

**Fsi**

**Fin Pour**

**Pour** chaque transition Sj>alph2>dst2 ∈ List\_trans2 **Faire**

Ajouter ((Si,Sj),alph2 ,(Si, dst2)),liste\_tans\_prod) ;

 **Si** (Si,dst2) n‟appartient pas (Etat\_trait U Nouv\_etat) **alors**

Ajouter ((Si, dst2) Nouv\_etat) ;

 **Finsi**

**Si** (Si∈ ens\_etf1 et dst2∈ ens\_etf2) ajouter ((Si, dst2), ens\_etf) ; FSi

**Fin Pour**

Ajouter (Etat, List\_Etat\_trait) ;

**Fin Tant que**

II trans-produit ←list\_trans\_prod ; E etat-produit← List\_Etat\_trait ;

X alph-produit ←Xalph-A1 U Xalph-A1 ; Ens\_etatf\_prod← ens\_etf ;

**Fin**.

1. ***L’exécution de l’algorithme sur un exemple :***

Soient deux services Web « réservation voiture et réservation hôtel », leurs séquences d’opérations sont représentés par les deux automates A1 et A2 définis comme suit :

A1 = { {DRV ,CND,PEC}, { q01 , q11 , q21 }, { q01 }, { q21,q11 }, T1 }

A2 = {{DRH, PEE}, {q02, q12, q22}, {q02}, {q22}, T2}

**PEC**

**CND**

**DRV**

**Services 1 : Réservation voiture**

**DRV : *Demande de réservation de voiture.***

**CND : *Confirmation de la demande.***

**PEC : *Payement en cheque***

**PEE**

**DRH**

**Services 2 : Réservation hôtel**

**DRH : *Demande réservation hôtel.***

**PEE: *Payement en espèce.***

**La résulta de service web composite :**

**PEE**

**Q21, Q12**

**Q11, Q22**

**DRH**

**Q21, Q22**

**Q01, Q22**

**Q11, Q12**

**Q21, Q02**

**Q01, Q12**

**Q01, Q02**

**Q11, Q02**

**DRH**

**DRV**

**CND**

**DRV**

**PEE**

**PEE**

**DRH**

**PEC**

**PEC**

**CND**

**PEC**

**CND**

**DRV**

**Service web composite**

**DRV : *Demande de réservation de voiture.***

**DRH : Demande réservation hôtel.**

**CND : *Confirmation de la demande.***

**PEC : *Payement en cheque.***

**PEE : *Payement en espèce.***

Soit une requête utilisateur (Demande de réservation de voiture, Demande réservation hôtel, Payement en espèce).on voit bien qu’en exécutant chaque services tout seul on ne peut pas satisfaire la requête, mais après la composition de ces deux services on obtient un service composite qui répond à cette requête comme l’indique le chemin en gras.