IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TD2-4/TP1-2 – Système de transitions étiquetées et pratique de FSP

Modélisation et Programmation Concurrente – ELU 512

Objectifs

L'objectif de ces quatre séances est de découvrir par la pratique les bases d'une modélisation formelle de la concurrence qu'est FSP. Nous allons :

- apprendre à modéliser un système simple,
- travailler la syntaxe de base de FSP,
- approfondir la correspondance entre les automates et la syntaxe FSP,
- composer des systèmes simples pour décrire des systèmes plus complexes.

Tout au long des séances, vous êtes invités à utiliser les notes de cours sur FSP. Les deux premières sections utilisent les notions présentées dans le premier cours et résumées dans la première section des notes de cours. La troisième section nécessite la lecture de la deuxième section des notes de cours.

Il est fortement conseillé de faire tous les exercices même s'ils n'ont pas pu être abordés en séance par manque de temps. En cas de problème, vous pouvez poser des questions sur le forum, contacter les enseignants, utiliser le monitorat informatique ou consulter la correction sur moodle.

1 La découverte de FSP (TD2)

Exercice 1 (*Feu tricolore*)

Modéliser le comportement d'un feu tricolore en fournissant :

- un programme FSP,
- l'automate associé,
- une trace.

Exercice 2 (*Machine à café*)

Modélisez le comportement d'une machine à café qui a deux boutons : un qui permet d'obtenir un café et l'autre un thé.

Exercice 3 (*Une bombe simple*)

Modéliser le comportement d'une bombe qui explose après 3 tics ou qui peut être désamorcée.

Exercice 4 (*Modélisation d'un processus de production*)

Pour produire un produit Y, trois robots sont utilisés (R1, R2 et R). Ils génèrent respectivement les composants Y1, Y2 et le produit final Y. La production suit les règles suivantes :

- R1 génère un composant Y1 et il attend que le composant soit utilisé avant d'en générer un autre,
- R2 génère le composant Y2,
- R2 a la possibilité de produire deux fois le composant avant d'attendre l'utilisation de sa production,
- R prend les composants Y1 et Y2 et les assemble pour générer le produit Y.

Inspiré de Michel Riveill.

Exercice 5 (*Sémantique de FSP*)

Donner le LTS des constructions FSP suivantes en fonction du LTS de leurs sous-éléments :

- `STOP`,
- `ERROR`,
- `a -> P`
- et `a1 -> P1 | a2 -> P2`.

2 Prise en main de l'outil LTSA (TP1)

LTSA est un outil de vérification de modèles développé à l'Imperial College de Londres, disponible de plusieurs manières :

- en tapant `ltsa.sh` dans un terminal,
- en le récupérant : `http://www.doc.ic.ac.uk/~jnm/book/ltsa/download.html` ou dans l'espace Moodle de ce module :
  - le fichier téléchargé est une archive zip, il faut le décompresser grâce à la commande `unzip ltsa.zip -d ./ltsa`;
  - lancer l'archive java (extension `.jar`) grâce à la commande `java -jar ltsa.jar`.

L'interface de LTSA est représentée dans la figure 1.

Les étapes de développement d'un modèle dans LTSA sont :

1. Édition du texte dans l'onglet d'édition (accessible via le bouton Edit) pour insérer (p.ex. par copier/coller) une spécification formelle FSP.

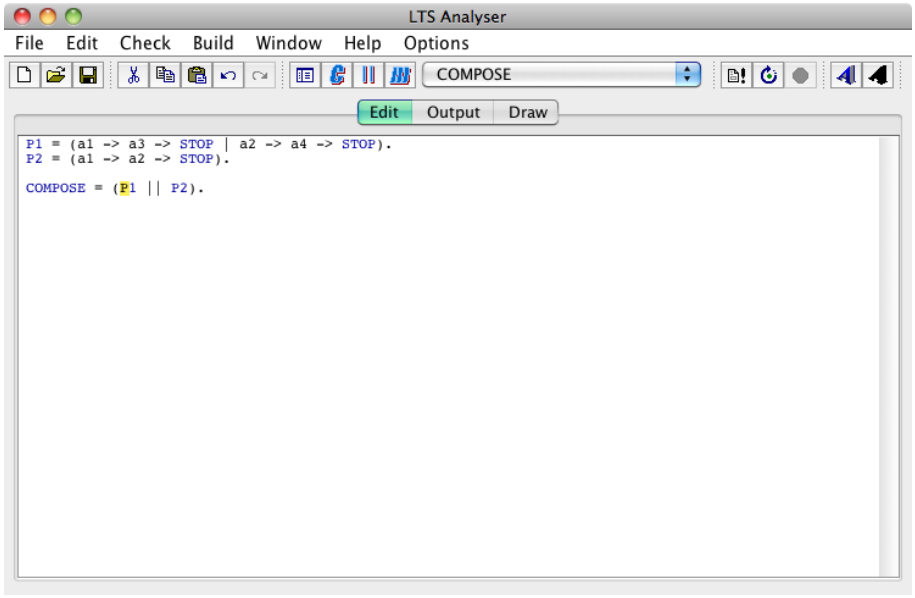


FIGURE 1 – L'interface graphique de l'outil LTSA.

2. Vérification syntaxique de votre spécification (bouton Parse,). Le résultat des différentes opérations peut être consulté dans l'onglet Output.
3. Compilation pour créer les automates des processus (bouton Compile,). Vous remarquerez que la compilation lance automatiquement la vérification syntaxique si vous ne l'avez pas faite.
4. Visualisation des automates générés automatiquement dans l'onglet Draw. Des boutons de redimensionnement (et) permettent de zoomer/dézoomer certaines parties.
5. Composition éventuelle à l'aide du bouton Compose ( ) permettant de générer un unique LTS correspondant au produit des automates des différents processus.
6. Analyses de propriétés (par exemple Safety, Progress, Reachability) sont disponibles dans le menu Check. En cas de détection de problèmes fonctionnels dans une spécification, une trace est fournie. Dans un premier temps, nous n'utiliserons pas ces fonctionnalités.
7. Exécution du modèle avec l'action Run du menu Check, qui ouvre une fenêtre Animator. La fenêtre permet de retracer visuellement, que ce soit en exécution totale ou en pas à pas (bouton Step). Dans le deuxième cas, à chaque pas, vous sélectionnez une action à réaliser. Vous pouvez voir en instantané sur les automates des processus les actions étiquetées qui passent en rouge.

Vous allez pouvoir prendre en main cet outil au travers des exercices qui suivent.

Exercice 6 (*Radio FM*)

Une radio portable FM a trois boutons. Un bouton `on`/`off` qui allume et éteint l'appareil. Le choix des fréquences radios s'effectue au travers de 2 boutons : `scan` et `reset`. Quand on allume la radio ou que l'on appuie sur le bouton `reset`, la radio est ramenée sur la fréquence de base de la bande FM (88MHz). Quand on appuie sur le bouton `scan`, la radio recherche la prochaine fréquence d'émission en montant en fréquence jusqu'à s'ajuster sur une station ou à atteindre la fin de la bande FM. Si la radio est déjà ajustée et que l'on appuie à nouveau sur le bouton `scan`,

le même processus est appliqué jusqu'à atteindre la prochaine station sur la bande FM.

▷ **Question 6.1 :**
En utilisant l'alphabet {`on`, `off`, `scan` `reset`, `ajuster`, `fin`}, modéliser la radio FM sous forme d'un processus FSP.

▷ **Question 6.2 :**
Construire un radio réveil en composant ce processus avec une horloge et visualiser le comportement résultant.

Exercice 7 (*Le télérupteur*)

Un télérupteur est un type de relais électromécanique commandé par des impulsions électriques. Il permet l'alimentation d'un circuit (par exemple l'éclairage d'un escalier) par plusieurs boutons poussoirs. Ce dispositif de commande d'éclairage remplace avantageusement le va-et-vient car il peut comporter un nombre illimité de boutons de commande.

Un télérupteur a un fonctionnement similaire à une « bascule ». Une impulsion (pression sur un bouton poussoir) déclenche le télérupteur, ce dernier ferme le circuit jusqu'à ce qu'une nouvelle impulsion ouvre à nouveau le circuit.

▷ **Question 7.1 :**
Modéliser le système télérupteur composé d'un bouton poussoir et d'un relais basculant dans les positions `on`/`off`.

▷ **Question 7.2 :**
Connecter plusieurs boutons poussoirs au même télérupteur.

Exercice 8 (*La bombe*)

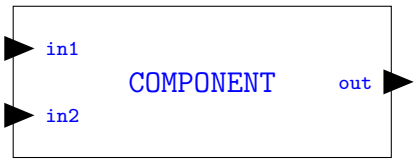
Modéliser le comportement d'une bombe qui explose après N tics ou qui peut être désamorcée.

3 Décomposition système (TD3)

Un système va être modélisé par une hiérarchie de modèles (des automates). Cette décomposition d'un sous-système se fait jusqu'à ce que le système soit suffisamment simple pour que décrire son automate soit facile. Ensuite, chaque modèle obtenu est composé pour construire le modèle final du système.

Pour rendre aisée la re-composition, l'idée est d'exprimer clairement les entrées et sorties de chaque composant ainsi que la façon dont les entrées d'un composant sont fournies par l'entrée ou la sortie d'un autre composant.

Pour cela, nous allons utiliser un formalisme graphique simple de boîtes et traits. Un composant est représenté par une boîte avec en partie gauche ses entrées (des actions) et en partie droite ses sorties (des actions également) comme ci-dessous.

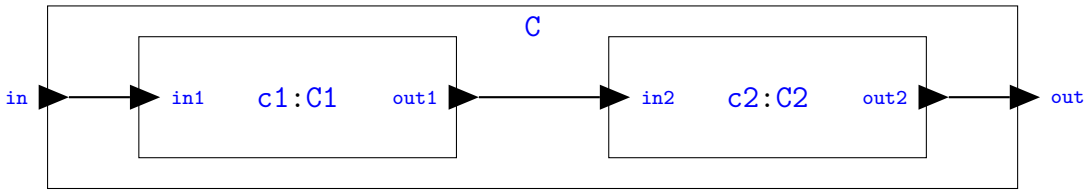


Exercice 9 (*Additionneur*)

▷ **Question 9.1 :**
Modéliser un composant additionneur et son comportement.

▷ **Question 9.2 :**
Utiliser cet additionneur pour calculer la somme de 1 et 2.

Plusieurs instances de composants peuvent être composés et regroupés en une nouvelle instance de composant comme ci-dessous. Suivant ce que l'on appelle un *schéma d'architecture*.



Cela correspond en terme de FSP à disposer des automates de `C1` et `C2` et les composer pour produire l'automate de `C`. Chaque connection donne lieu à un renommage. Les entrées sorties du composant englobant doivent être conservées (partie gauche du renommage). Pour les connections « internes », la convention adoptée ci-dessous est de conserver l'action de sortie.

```
C1 = ...
C2 = ...
||C = (c1:C1 || c2:C2) / { c1.out1 / c2.in2, in / c1.in1, out / c2.out2 }.
```

L'automate obtenu pour `C` laisse apparaître l'action `c1.out1`. Si l'on souhaite le cacher, on peut utiliser la commande `@ {...}` qui transforme toutes les actions ne figurant pas dans l'ensemble d'actions fournit en action silencieuse (τ). On peut alors minimiser l'automate, ce qui supprime les actions silencieuses. Ainsi, par exemple avec les automates `C1` et `C2` qui suivent, on obtient l'automate du haut de la figure 2 qui est minimisé en celui du bas de la même figure.

```
C1 = (in1 -> out1 -> C1).
C2 = (in2 -> in2 -> out2 -> C2).
||C = (c1:C1 || c2:C2) / { c1.out1 / c2.in2, in / c1.in1, out / c2.out2 } @ {in,out}.
```

Ainsi, pour construire le modèle d'un système, nous allons procéder par décomposition puis composition :

1. identifier les actions d'entrée / sortie du système ;
2. décomposer le système en un ensemble de sous-systèmes
3. pour chaque sous-système ;
   — s'il est simple, décrire en FSP son comportement (enchaînement des actions) de manière incrémentale ;
   — s'il est complexe, réappliquer la décomposition (1 à 3) ;
4. produire la composition (avec les renommages nécessaires et éventuellement en cachant les actions internes).

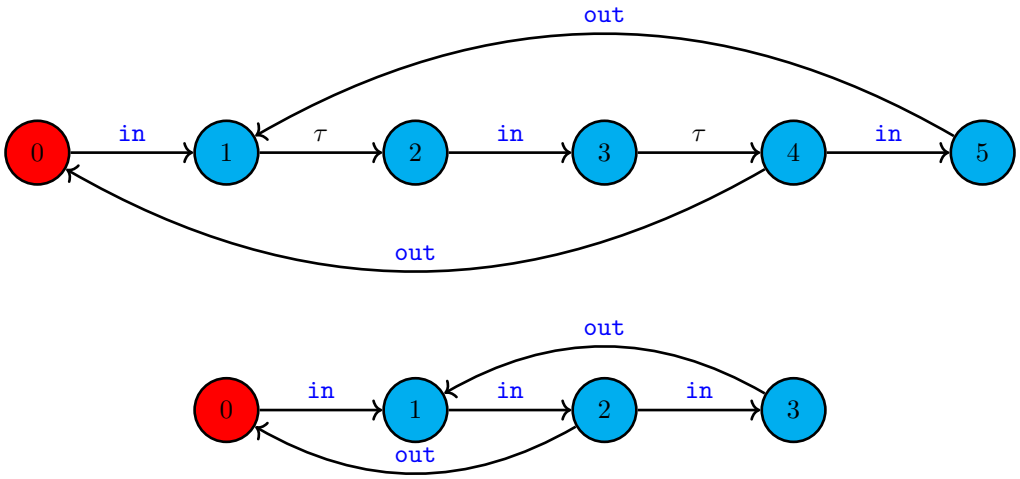


FIGURE 2 – Automate de composition / décomposition

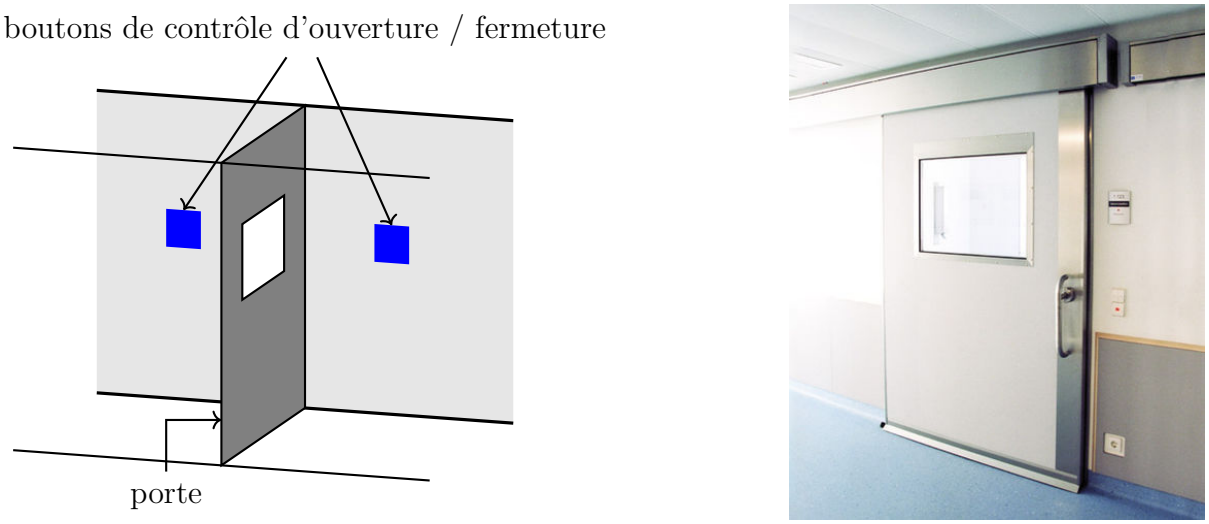


FIGURE 3 – Porte coulissante à commande déportée

Exercice 10 (*Un porte coulissante à commande déportée*)

Le but de cet exercice est de modéliser une porte dont l'ouverture et la fermeture se font par l'intermédiaire de boutons installés de part et d'autres de la porte, en général sur des murs. La figure 3 contient une image d'une telle porte et un schéma du système à modéliser. Les boutons sont des boutons-poussoirs et il n'ont donc qu'une seule action `press`. Lorsqu'un utilisateur presse un des deux boutons, la porte s'ouvre si elle est fermée et se ferme si elle est ouverte.

▷ **Question 10.1 (*Porte simple*) :**
Proposer une décomposition du système en éléments plus simples. Donner le schéma d'architecture et le modèle FSP de ce système en décrivant chacun des sous-systèmes proposés. Enfin, donner un scénario d'utilisation du système ainsi que la trace qui correspond.

Nous allons reprendre notre système de porte et le modifier pour ajouter une temporisation. Il

faut faire en sorte qu'après un certain temps, une porte ouverte se ferme automatiquement [1].

▷ **Question 10.2 (*Fermeture automatique*) :**
Proposer une décomposition du système en éléments plus simples. Donner le modèle FSP de ce système en décrivant chacun des sous-systèmes proposés. Enfin, donner un scénario d'utilisation du système ainsi que la trace qui correspond.

4 Vers des plus gros modèles (TD4)

Exercice 11 (*Un sas de décontamination*)

Pour des raisons de sûreté, on a parfois besoin de combiner deux portes comme celles de l'exercice précédent pour proposer un sas. Un tel sas présente une contrainte de sûreté supplémentaire, une porte ne peut s'ouvrir que si la deuxième est fermée.

▷ **En réutilisant le système produit dans l'exercice précédent question 1, proposer un modèle FSP qui respecte la contrainte de sûreté.**

Exercice 12 (*Retour sur la machine à café*)

Maintenant, la machine à café se voit enrichie des caractéristiques suivantes :
- le prix d'une boisson est 30 centimes,
- les pièces acceptées sont 10, 20 et 50 centimes,
- la machine rend la monnaie.

▷ **Modéliser un tel système en FSP.**

Exercice 13 (*Des utilisateurs pour la machine à café*)

Ajouter quelques utilisateurs à la machine à café. Par exemple :
- Toto qui n'a que des pièces de 50
- Tutu qui utilise uniquement des pièces de 10
- Tata qui utilise alternativement des pièces de 10 et de 20

5 Mise en pratique (TP2)

Exercice 14 (*La centrale de réservation*)

Modéliser une centrale de réservation de siège de spectacle. Cette centrale est reliée à un ensemble de terminaux de réservation. Sur un de ces terminaux, un employé peut consulter l'état d'un siège qui peut être libre ou occupé. Un client peut réserver un siège qui est libre, l'employé entre alors le numéro du siège et imprime un ticket. Un siège ne peut être réservé deux fois et le système doit garantir cette propriété. Inspiré de [M&K] exercice 4.3.

1. Il faudrait ajouter un capteur pour éviter d'écraser des utilisateurs mais nous ignorons ce problème ici.