Département EEA - Faculté des Sciences et d'Ingénierie

Master STS - EEA

Électronique Électrotechnique Automatique

Spécialité ISTR

Unité d'enseignement I1

Systèmes à événements discrets, modélisation et analyse

Année 2015-2016

Travaux pratiques

Sommaire

Déroulement des séances de travaux pratiques SED

Affectation des binômes sur les manipulations de TP

La séquence des TPs est : **Ascenseur** → **BCI** → **Trains** → **Robots**

Les étudiants sont répartis en trois groupes. Le tableau suivant récapitule les manipulations que doivent préparer les binômes pour la **première séance de TP** de SED. Les personnes dont les noms n'apparaissent pas dans le tableau devez mettre vos noms dans les blancs selon vos groupes de TP disponibles sur le blog.

GROUPE 1
Dates des séances de TP SED : 16/11/15, 23/11/15, 30/11/15 et 7/12/15 de 8h à 12h.

Binôme 1	GREZIT Daphnée / IBANEZ Emmanuel	Ascenseur
Binôme 2	DOVONOU Jason / ELGOURAIN Abdellah	BCI
Binôme 3	MRANI Amine /	Trains
Binôme 4		Robots

GROUPE 2
Dates des séances de TP SED : 20/11/15, 27/11/15, 4/12/15 et 11/12/15 de 8h à 12h.

Binôme 1	BEGER Pascal / IBARZ Jean	Ascenseur
Binôme 2	KOWET Valens /	BCI
Binôme 3		Trains
Binôme 4		Robots

GROUPE 3
Dates des séances de TP SED : 17/11/14, 24/11/15, 1/12/15 et 8/12/15 de 13h30 à 17h30.

Binôme 1	SHULGA Evgeny / DATO Bruno	Ascenseur
Binôme 2	NAJAR Hédi / FAGEDET Pierre	BCI
Binôme 3	HULAUD Benjamin / EL GHZAOUI Mohamed	Trains
Binôme 4	JAILLOT Laure	Robots

Déroulement des TP

Pour chaque manipulation, des outils de génération de code appelés "moulinettes" vous sont fournis et cachent les aspects de mise en oeuvre de vos modèles de commande. Ces aspects seront traités au second semestre dans le module "Techniques de mises en oeuvre de Systèmes à événements discrets". Voir l'annexe de ce cahier de TP pour les consignes d'utilisation de ces moulinettes pour chaque manipulation.

Pour tous les TP de ce module, les parties manipulatoires seront donc constituées des tests et mise au points de vos modèles. Afin ne pas perdre de temps pendant la séance, vous devrez élaborer les modèles du procédé et des objectifs avant chaque TP.

Evaluation des TP

- Chaque binôme présentera un compte-rendu d'une manipulation tirée au sort lors d'un exposé oral qui aura lieu semaine 51. Voir la section suivante pour la préparation de ce compte-rendu oral.
- Un contrôle de TP individuel aura lieu cette même semaine 51 avec également un système de tirage au sort. Il se fera sur PC avec utilisation des logiciels DESUMA ou TINA selon le formalisme du sujet (automate ou réseau de Petri).

Compte-rendu de TP : exposé oral

Chaque binôme devra préparer un compte-rendu oral pour un sujet automate et un sujet réseau de Petri de son choix à partir des notes personnelles, des explications données par les enseignants et des impressions réalisées en séance. Ce compte-rendu oral aura un support informatique (slides en PDF) de manière à visualiser et commenter les modèles de commande obtenus mais aucune démonstration sur le matériel ne sera demandée.

Pour l'évaluation, un sujet automate ou réseau de Petri sera tiré au sort parmi les deux comptes-rendus préparés. Le binôme disposera de 20 minutes pour présenter ses résultats selon le plan proposé ci-dessous :

1. Application de la théorie
   a. Présentation des modèles du procédé et des objectifs
   b. Présentation du modèle de commande en expliquant la méthode utilisée (vue en cours et en TD)
2. Mise en pratique
   a. Critique de la commande obtenue : déterministe, oscillante, complexe, ...
   b. Amélioration et modification des modèles, explication des solutions adoptées
   c. Résultats pratiques
3. Conclusion
   a. Généralisation des concepts spécifiques au procédé étudié
   b. Retour sur la théorie du module suite à l'expérience du TP

Pour toute question d'organisation : euriell.le.corronc@laas.fr, sylvain.durola@laas.fr

MANIPULATION 1

Commande d'un ascenseur (Automates à états finis)

1.1 Objectif de la manipulation

L'objectif de cette manipulation est de commander grâce au formalisme automate, un procédé typique des systèmes à événements discrets qu'est l'ascenseur grâce à la technique de synthèse de commande vue en cours et en TD. La commande de ce système requiert de prendre en compte de nombreuses contraintes de fonctionnement. Ainsi, nous vous proposons de découper proprement ces contraintes en différents automates que vous associerez à vos modèles d'objectifs grâce à la composition parallèle. Les cahiers de charges décrivant les objectifs vont spécifier des comportements de plus en plus complexes.

1.2 Prérequis

- Automates à états finis : construction, opérations entre automates.
- Méthodes de synthèse de commande vue en cours et en TD.

1.3 Matériel utilisé

1.3.1 Système commandé

Il s'agit d'une maquette d'ascenseur à quatre étages, entraînée par deux actionneurs $Montee$ et $Descente$. La position de la cabine est repérée par quatre capteurs d'étage **ET1**, **ET2**, **ET3** et **ET4** (avec ETi = 1 cabine à l'étage i). A l'intérieur de la cabine, quatre poussoirs d'appel **AP1**, **AP2**, **AP3** et **AP4** permettent aux utilisateurs de demander le déplacement de la cabine. Un dispositif de sécurité coupe le mouvement en cours si la cabine descend au dessous de l'étage 1 ou si elle monte au dessus de l'étage 4. Six poussoirs palier **P4d**, **P3m**, **P3d**, **P2m**, **P2d**, **P1m** (**Pxd** et **Pxm** pour palierxMontee ou palierxDescente) permettent à l'utilisateur d'appeler la cabine à partir du palier de l'étage x. Un contact **PorteOuverte** passe à "1" dès qu'une des portes donnant accès à l'ascenseur est ouverte. Deux interrupteurs **Stop** et **Reprise** sont destinés, en particulier, à la gestion des urgence sur cette maquette.

La maquette dispose par ailleurs de deux afficheurs sept segments pouvant servir à afficher des informations au cours du fonctionnement (numéro de l'état actif par exemple). Les capteurs, interrupteurs et boutons poussoirs utilisés dans cette manipulation sont représentés sur la figure 1.1.

1.3.2 Système de commande

Le système de commande est constitué d'une carte Altera UP1 (University Program) comportant deux circuits programmables : un CPLD et un FPGA. Pour des raisons de

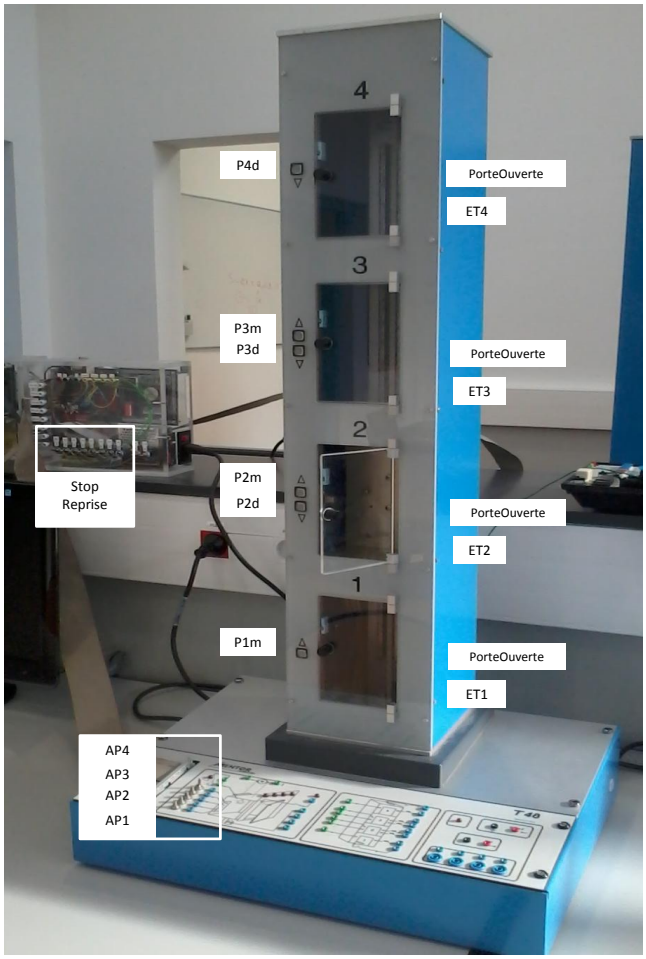


Figure 1.1: Maquette d'ascenseur

simplicité, transparentes dans ce type de manipulation, c'est le CPLD qui est utilisé comme cible d'implantation du système de commande. Une nappe de fils assure la connexion électrique entre la carte UP1 et la maquette. Le circuit dans lequel est implantée la commande à spécifier est programmé par l'intermédiaire d'un logiciel fourni par la société Altera (Quartus). Le langage d'entrée de cette spécification est VHDL.

1.4 Travail à effectuer

1.4.1 Modélisation du procédé

Déterminer les composants du procédé dont le comportement peut être indépendant des autres éléments. Vous représenterez l'ensemble des évolutions possibles de chaque composant par un automate à états finis. Le modèle complet du procédé sera obtenu en effectuant le produit parallèle des automates des différents éléments.

1.4.2 Méthodologie

Quatre étapes de modélisation vous sont proposées, chaque étape rendant la commande plus complexe que la précédente. Pour chacune d'entre elles, vous devrez :

- Proposer un modèle des objectifs de la commande ;

- Effectuer la synthèse de cette commande, c'est-à-dire calculer le produit parallèle des automates puis le restreindre aux trajectoires utiles pour les objectifs (utiliser l'outil DESUMA, cf. Annexe) ;
- Vérifier que le modèle ainsi obtenu possède bien les deux propriétés essentielles : non bloquant et pas d'instabilité incontrôlable ;
- Tester et mettre au point le modèle de commande obtenu sur la maquette (utiliser l'outil TINA, l'outil générant le code VHDL et le logiciel Quartus, cf. Annexe).

1.4.2.1 Contraintes de fonctionnement

Afin d'améliorer les performances des modèles de commandes obtenus, il vous est proposé d'utiliser les contraintes supplémentaires suivantes (chaque contrainte est représentée par un automate) :

- On ne peut prendre en compte un appel qu'en face d'un étage.
- On ne peut s'arrêter que devant un étage.
- Un mouvement doit être provoqué par un appel et tout appel doit provoquer un mouvement.

Ces contraintes sont à associer aux objectifs grâce à l'utilisation du produit parallèle.

1.4.2.2 Fonctionnement en monte-charge

Dans ce premier fonctionnement, la cabine se déplace entre les étages 1 et 2 uniquement. Chaque changement d'étage est provoqué par un appui sur le bouton poussoir **AP1**.

A l'état initial, la cabine est à l'arrêt devant l'étage 1 en attente d'un appel.

1.4.2.3 Fonctionnement en ascenseur simple

Chaque appel **APi** doit provoquer au moins un passage devant l'étage i. Tout appel doit provoquer au moins un passage devant un étage.

A l'état initial, la cabine est à l'arrêt devant l'étage 1 en attente d'un appel.

1.4.2.4 Prise en compte des arrêts d'urgence

Le système d'arrêt d'urgence est sensible à deux signaux **Stop** et **Reprise**. L'appui sur **Stop** doit figer le mouvement de la cabine. Le retour au mode automatique n'est alors possible que lorsque l'opérateur appuie sur **Reprise**.

1.4.2.5 Mémorisation des appels paliers

Les appels palier peuvent maintenant être pris en compte à n'importe quel moment. Un appel pour l'étage x est mémorisé dès que l'expression **Pxd + Pxm** devient "vraie". L'appel sera ensuite oublié dès que la porte de la cabine s'ouvre à l'étage x.

MANIPULATION 2

Commande d'un banc de contrôle industriel (Automates à états finis)

2.1 Objectif de la manipulation

Dans cette séance de travaux pratiques, on vous propose d'utiliser la théorie des langages pour développer une commande de la maquette de Banc de Contrôle Industriel (BCI). Cette maquette est une plateforme pédagogique automatisée permettant de réaliser des fonctions de tri, d'assemblage et de contrôle de pièces. La séance de TP portera plus précisément sur la fonction de contrôle et d'évacuation.

Vous aurez l'occasion de mettre en pratique la démarche systématique de modélisation vue lors du cours et des TD. Dans un premier temps, vous modéliserez le procédé seul, à partir de ses spécifications matérielles et d'une brève analyse de son fonctionnement. Ensuite, vous modéliserez les objectifs à réaliser, en vous basant sur les données des cahiers des charges proposés. Enfin, en fusionnant ces deux modèles et en simplifiant le résultat, vous obtiendrez un modèle de la commande désirée.

2.2 Prérequis

- Bases de la théorie du langage.
- Méthodes de synthèse des automates à états finis.

2.3 Matériel utilisé

2.3.1 Système commandé

La fonction principale de la maquette de BCI est de trier deux types de pièces arrivant de manière désordonnée sur un *convoyeur*, de les assembler et de contrôler le résultat obtenu sur un *tapis roulant*. Voici les pièces que nous considérerons :

- Les pièces hautes grises, embases métalliques, que l'on assimilera à des *bouteilles*.
- Les pièces basses blanches, anneaux en matière plastique que l'on assimilera à des *bouchons*.
- Lorsqu'un anneau est correctement emboîté sur une embase, on appellera ceci un *assemblage*.

Dans cette séance, vous ne travaillerez pas explicitement sur les fonctions de tri et d'assemblage. Le guidage des bouteilles de la *zone de Tri* vers la *zone d'Assemblage* est réalisé mécaniquement, de même que l'assemblage d'un bouchon et d'une bouteille dans la *zone d'Assemblage*. Les capteurs et actionneurs qui leur sont liés sont les suivants :

- **C1**, détecteur photo-électrique de proximité, détecte toutes les pièces dans la *zone de Tri*.
- **C2**, détecteur photo-électrique de proximité, détecte uniquement les pièces hautes

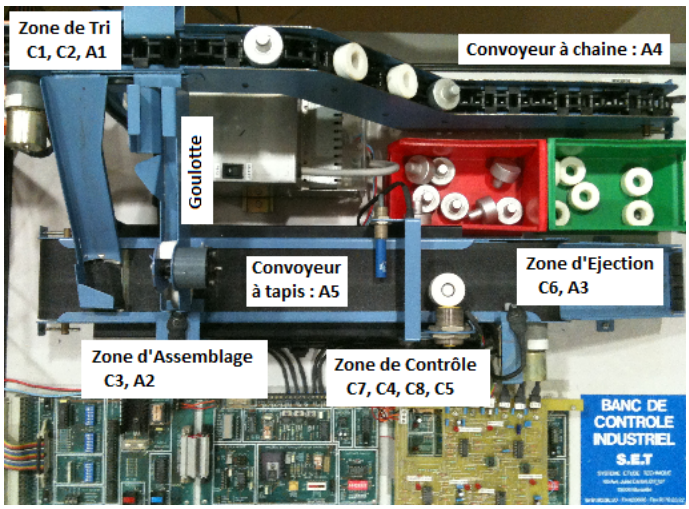


Figure 2.1: Banc de Contrôle Industriel

dans la *zone de Tri*.

- **C3**, détecteur photo-électrique de proximité, détecte un bouchon dans la *zone d'Assemblage*.
- **A1**, actionneur linéaire à solénoïde, permet d'éjecter une pièce de la *zone de Tri* vers la *goulotte*.
- **A2**, actionneur rotatif à solénoïde, permet d'admettre un bouchon de la *goulotte* dans la *zone d'Assemblage*.
- **A4** est le moteur du convoyeur à chaîne apportant des pièces dans la *zone de Tri*.

En ce qui concerne la fonction de contrôle et d'éjection, voici les capteurs et actionneurs correspondants :

- **C4**, détecteur photo-électrique par barrage, détecte toutes les pièces arrivant vers la *zone de Contrôle*.
- **C5**, détecteur photo-électrique de proximité, détecte toutes les pièces dans la *zone de Contrôle*.
- **C6**, détecteur photo-électrique de proximité, détecte toutes les pièces dans la *zone d'Éjection*.
- **C7**, détecteur de proximité inductif, détecte les bouteilles, métalliques, arrivant vers la *zone de Contrôle*.
- **C8**, détecteur de proximité capacitif placé en hauteur, détecte les pièces hautes dans la *zone de Contrôle*.
- **A3**, actionneur linéaire à solénoïde, permet d'éjecter une pièce de la *zone d'Éjection*.
- **A5** est le moteur du convoyeur à bande (tapis roulant) apportant des pièces de la *zone d'Assemblage* vers la *zone de Contrôle* et la *zone d'Éjection*.

2.3.2 Système de commande

Ce matériel sera commandé par un Automate Programmable Industriel (API) TSX. Il est relié par une première nappe de fils aux actionneurs et capteurs de la maquette, et par une seconde aux ordinateurs de l'îlot via le réseau. Dans cette manipulation, on utilisera le logiciel UNITY qui permettra de saisir un programme en langage ST et de

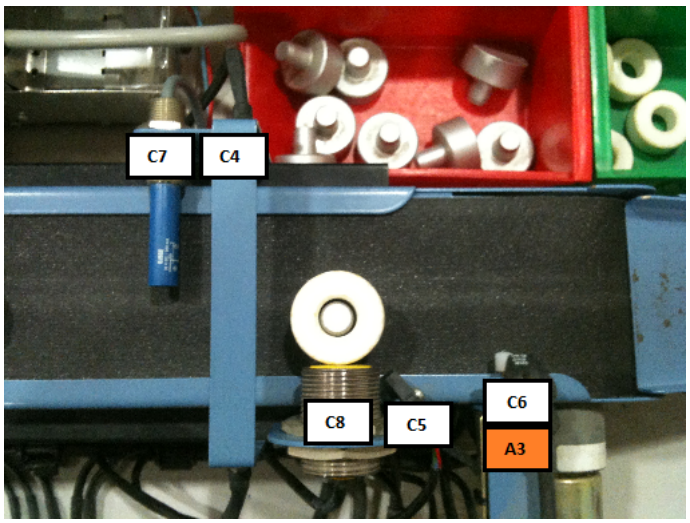


Figure 2.2: Zones de Contrôle et d'Éjection

l'envoyer sur l'automate pour exécution.

2.4 Travail à réaliser

Posons les hypothèses suivantes :

- Au maximum une pièce est présente entre les capteurs **C7** et **C6**.
- Le convoyeur à bande **A5** sera supposé actif à tout instant.

2.4.1 Analyse et modélisation du procédé

Comme les deux capteurs **C5** et **C8** sont de nature physique différente, les zones respectives dans laquelle ils peuvent détecter une pièce ne sont pas identiques. Cependant, la conception de la maquette fait que ces deux zones se chevauchent.

- Représentez sur un dessin les quatre dispositions possibles.
- Identifiez celle qui est réalisée sur la maquette proposée.

Considérons à présent les deux hypothèses énoncées plus haut :

- Ecrivez les séquences possibles pour les pièces *bouchon*, *bouteille* et *assemblage*, en prenant en compte les quatre dispositions possibles des capteurs **C5** et **C8**. Vous représenterez chaque changement d'état possible de capteur ou d'actionneur.

On suppose enfin que des pièces arrivent successivement et donc que le procédé effectue une boucle.

- Trouvez l'expression régulière correspondante.
- Tracez le graphe de l'automate équivalent.
- Pour des états finaux correspondant à l'éjection ou l'admission (non éjection) d'une pièce reconnue (*bouchon*, *bouteille* ou *assemblage*) et un état initial correspondant à un tapis vide, quel est le langage marqué reconnu par l'automate ?
- Pour des états finaux correspondant uniquement à l'admission (non éjection) d'une pièce reconnue (*bouchon*, *bouteille* ou *assemblage*) et un état initial correspon-

dant à un tapis vide, quel est le langage marqué reconnu par l'automate ?

2.4.2 Modélisation des objectifs

2.4.2.1 Premier cahier des charges

Réalisez sous la forme d'une automate la modélisation des objectifs correspondant au cahier des charges suivant :

- On admet tous les *assemblages*.
- On éjecte toutes les pièces qui ne sont pas des *assemblages*.
- Le fonctionnement est répétitif.

2.4.2.2 Second cahier des charges

Réalisez sous la forme d'un automate la modélisation des objectifs correspondant au cahier des charges suivant :

- On admet un *bouchon* puis une *bouteille*.
- On éjecte toutes les pièces qui ne correspondent pas à cette séquence.
- Le fonctionnement est répétitif.

2.4.3 Produit parallèle des deux modèles

Pour chacun des deux cahiers des charges, calculez le produit parallèle de l'automate des objectifs de commande avec celui du procédé. Pour ce faire, vous utiliserez le logiciel DESUMA (cf. Annexe).

2.4.4 Simplification

Modifier chacun des deux automates obtenus lors du produit parallèle pour n'obtenir que les trajectoires utiles pour les objectifs.

2.4.5 Tests et mise au point

Réaliser en séance de TP les tests et mise au point des modèles obtenus. Pour ce faire, vous utiliserez le logiciel TINA, l'outil générant du code ST et le logiciel Unity (cf. Annexe).

2.4.6 Nouvelles hypothèses de modélisation

On vous propose maintenant d'étudier le cas où la première hypothèse (au maximum une pièce est présente entre les capteurs **C7** et **C6**) n'est pas vérifiée.

Posons de nouvelles hypothèses remplaçant celle devenue caduque :

- Au maximum deux pièces sont présentes entre les capteurs **C7** et **C6**.
- Au maximum une pièce est présente devant le capteur **C7**.
- Au maximum une pièce est présente devant le capteur **C4**.
- Au maximum une pièce est présente devant les capteurs **C5** ou **C8**.
- Au maximum une pièce est présente devant le capteur **C6**.

- Les pièces avancent à la même vitesse sur le tapis et ne peuvent donc pas se doubler.

2.4.6.1 Analyse

Dans le travail demandé précédemment, vous avez identifié trois mots correspondants aux pièces *bouchon*, *bouteille* et *assemblage* passant devant les capteurs **C7**, **C4**,**C5** et **C8**. Découpez les trois mots *bouchon*, *bouteille* et *assemblage* en trois syllabes :

- S_1 (signature laissée devant le capteur **C7**),
- S_2 (signature laissée devant le capteur **C4**),
- S_3 (signature laissée devant les capteurs **C5** et **C8**).
- Identifiez aussi deux mots correspondant aux actions *éjecter la pièce* et *admettre la pièce*. Ils forment la quatrième syllabe S_4.

Lorsque deux pièces sont présentes entre les capteurs **C7** et **C6**, on peut considérer que deux mots sont en train d'être lus. Les quatre autres hypothèses proposées permettent en réalité de ne pas mélanger ces deux mots : une lettre lue ne peut correspondre qu'à une unique syllabe d'un unique mot.

- Dénombrez et identifiez à tout instant les situations de lecture de syllabe en cours.
- En appelant l_i une des lettres composant la syllabe S_i et f_i une lettre la finissant, tracez schématiquement l'évolution de ces situations de lecture en fonction des lettres lues.

2.4.6.2 Modélisation du procédé

La modélisation du procédé, dans ce cas, est loin d'être simple et fera intervenir plusieurs dizaines d'états. On vous demande d'esquisser l'automate correspondant : prenez une grande feuille et ne paniquez pas.

MANIPULATION 3

Commande d'un système de transport (Réseaux de Petri)

3.1 Objectif de la manipulation

L'objectif de cette étude est de développer une commande du système d'approvisionnement en matières premières constitué de wagonnets navigant entre des postes de chargement et de déchargement. La spécification de ce système de commande sera effectuée par réseau de Petri. Chaque cahier des charges vous demandera de réaliser une commande de plus en plus complète ce qui vous obligera à prendre en compte des ressources partagées et donc des contraintes supplémentaires. Les bonnes propriétés des réseaux de commande obtenus seront également analysées à l'aide du logiciel TINA.

3.2 Prérequis

- Modélisation par des réseaux de Petri interprétés
- Composition de modèles réseaux de Petri par la méthode de fusion des transitions
- Etude des propriétés d'un réseau de Petri à partir d'une analyse structurelle

3.3 Matériel utilisé

3.3.1 Système commandé

La maquette est composée de 4 voies A, B, C et D reliées par 4 aiguillages et 2 croisement-aiguillages. La disposition des voies empruntées par les wagonnets est donnée sur la figure 3.1

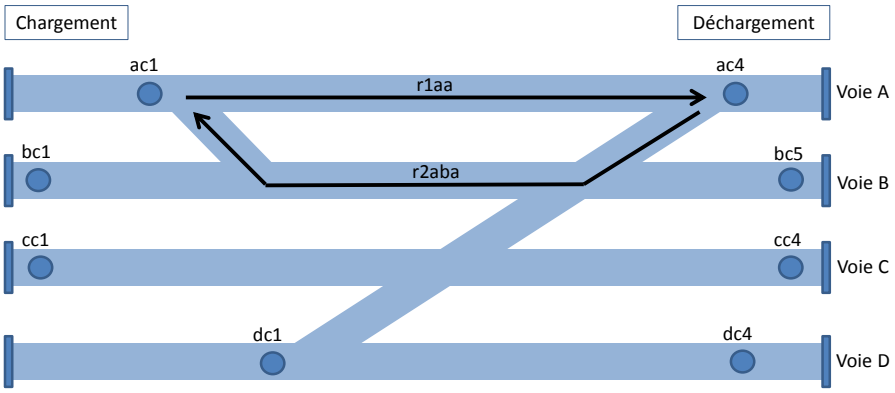


Figure 3.1: Maquette du système de transport

Des postes de chargement se trouvent aux extrémités gauche des voies et des postes de déchargement se trouvent aux extrémités droite des voies. La position des wagonnets à ces postes est détectable par un ensemble de capteurs (**ac1** à **dc4**).

Les wagonnets se déplacent d'un poste de chargement à un poste de déchargement à l'aide d'une commande **r1x** où **x** représente les noms des voies empruntées. Par exemple, deux commandes permettent qu'un wagonnet se déplace de **ac1** en **ac4**, **r1aa** permet d'effectuer un trajet direct uniquement sur la voie A et **r1aba** emprunte également la voie B. Sur la maquette, il existe donc 12 trajets possibles pour se déplacer d'un poste de chargement à un poste de déchargement : **r1aa**, **r1aba**, **r1ab**, **r1bb**, **r1ba**, **r1cc**, **r1cb**, **r1cba**, **r1dd**, **r1dc**, **r1dcb** et **r1dcba**.

Les wagonnents se déplacent d'un poste de déchargement à un poste de chargement à l'aide d'une commande **r2y** où **y** représente les noms des voies empruntées pour le trajet de retour. Par exemple, la commande **r2bcd** permet le déplacement d'un wagonnet de **bc5** en **dc1**. Il existe également 12 trajets possibles pour le retour d'un wagonnet à un poste de chargement : **r2aa**, **r2aba**, **r2ab**, **r2abc**, **r2abcd**, **r2bb**, **r2ba**, **r2bc**, **r2bcd**, **r2cc**, **r2cd** et **r2dd**.

L'alimentation des différents cantons et aiguillages de la maquette est commandée directement à partir des trajets sélectionnés.

Un interrupteur **bp1** est également disponible sur la maquette.

3.3.2 Système de commande

Le système de commande est constitué d'un ordinateur PC fonctionnant sous linux et contenant des cartes d'entrée/sortie adaptées pour contrôler les déplacements des wagonnets sur les différentes voies. Le langage d'entrée du système de commande est le langage C.

3.4 Travail à effectuer

3.4.1 Cahier des charges et méthodologie

Durant les différentes phases de la manipulation, nous nous intéresserons d'abord aux voies A et B, puis aux voies A, B et C. Le but est de commander des wagonnets pour acheminer des produits d'une zone de chargement vers une de zone de déchargement. Après déchargement, chaque wagonnet doit revenir à un poste de chargement (pas forcément celui de départ). Le départ de chaque cycle de chargement/déchargement/retour en poste de chargement est déclenché par l'utilisateur en appuyant le bouton poussoir **bp1**.

La synthèse de la commande désirée est décomposée pour être réalisée progressivement, chaque étape s'appuie sur la précédente. Pour chacune de ces étapes, vous devrez :

- Proposer un modèle du procédé afin de représenter toutes les évolutions potentielles du système ;
- Proposer un modèle des objectifs de la commande ;
- Effectuer la synthèse de cette commande, c'est-à-dire réaliser la fusion des modèles du procédé et des objectifs sans oublier de le restreindre aux trajectoires utiles pour les objectifs (utiliser le logiciel TINA, cf. Annexe) ;
- Effectuer sous TINA une analyse structurelle du modèle de commande obtenu afin de conclure quant aux bonnes propriétés du réseau (vivant, non-bloquant, ré-initialisable) ;
- Tester et mettre au point le modèle de commande obtenu sur la maquette (utiliser

le logiciel TINA, l'outil générant du code C et le terminal du PC, cf. Annexe).

3.4.2 Commande d'un wagonnet w1 sur les voies A et B

Un seul wagonnet est présent dans ce procédé et il ne peut naviguer que sur les voies A et B.

L'objectif est de diriger le wagonnet w1 intialement situé au poste de chargement de la voie A vers le poste de déchargement de la voie B et de le faire revenir ensuite au poste de chargement de la voie B. Le trajet emprunté par w1 sera laissé au choix du concepteur.

3.4.3 Commande des wagonnets w1 et w2 sur les voies A et B

On considère maintenant qu'un second wagonnet w2 peut circuler sur les voies A et B dans le procédé. Afin d'éviter toute collision entre les deux wagonnets w1 et w2, il faut empêcher qu'ils empruntent simultanément les mêmes voies. Cela peut se modéliser grace à des contraintes liées au partage des ressources.

Le premier objectif est que les wagonnets w1 et w2, respectivement positionnés aux postes de chargement de la voie A et de la voie B, doivent effectuer un déchargement puis revenir un à poste de déchargement.

Un second objectif demande maintenant à ce que les deux wagonnets échangent leur position initiale après avoir effectuer le déchargement décrit dans le premier objectif.

3.4.4 Commande des wagonnets w1 et w2 sur les voies A, B et C

La voie C de la maquette est maintenant considérée dans le procédé.

L'objectif est que les wagonnets w1 et w2, respectivement positionnés aux postes de chargement de la voie A et de la voie C, doivent se déplacer en **bc1** et **ac1**.

MANIPULATION 4

Commande d'une station de tri robotisée (Réseaux de Petri)

4.1 Objectif de la manipulation

Cette manipulation a pour but d'utiliser les réseaux de Petri afin de synthétiser une commande d'une station de tri robotisée en s'appuyant sur la méthode vue en cours et en TD. La commande à réaliser étant très complète car faisant appel à différents éléments indépendants, nous vous proposons de modéliser le procédé global par une approche modulaire de chaque élément du système. La commande globale sera alors réalisée en fusionnant les différentes commandes obtenues pour chacun de ses éléments. Vous rechercherez également les bonnes propriétés des modèles réseaux de Petri développés à l'aide de l'outil TINA.

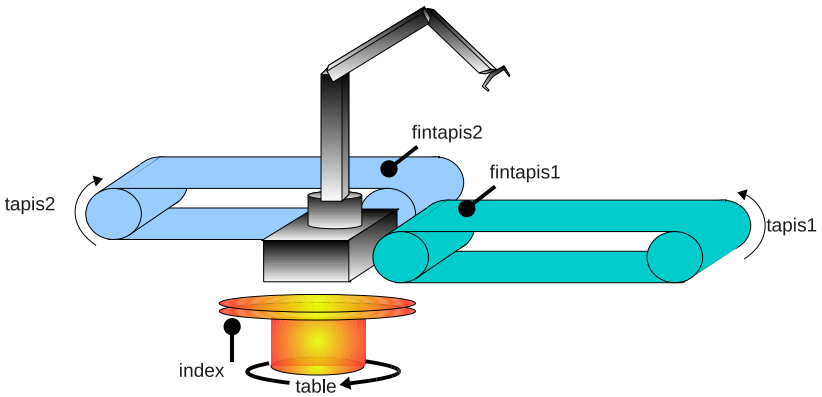
4.2 Prérequis

- Modélisation par des réseaux de Petri interprétés
- Composition de modèles réseaux de Petri par la méthode de fusion des transitions
- Etude des propriétés d'un réseau de Petri à partir d'une analyse structurelle

4.3 Matériel utilisé

4.3.1 Système commandé

Le système à commander est composé d'un robot, de deux tapis convoyeurs, d'un poste d'emballage et d'une table d'indexage. Les éléments sont disposés de sorte que le robot puisse accéder aux tapis, au poste d'emballage et à la table.



Le fonctionnement de ces éléments est le suivant.

Le robot : son rôle, dans cette manipulation, consiste à déplacer des pièces entre les deux tapis, la table et le poste d'assemblage. A cet effet, il peut tourner sur sa base (actionneurs $gauche$ et $droite$) entre deux capteurs de surcourse (scd et scg) entre lesquels quatre capteurs de position correspondant aux éléments sur lesquels le robot peut être amené à saisir ou déposer des pièces ($surt1$, $surt2$, $surtable$ et $surassemblage$). La prise (resp. la pose) d'une pièce à la position où se trouve le robot est déclenchée par l'actionneur $prise$ (resp. $pose$), la fin de cette opération considérée comme atomique dans cette manipulation est signalée par le capteur $finprise$ (resp. $finpose$).

Les tapis : permettent d'acheminer des pièces dans une direction unique matérialisée sur la figure par une flèche. L'avancement du tapis 1 (reps. tapis 2) est provoqué par l'actionneur $tapis1$ (reps.$tapis2$). Un capteur de présence permet de détecter l'arrivée d'une pièce dans la zone accessible par le robot. Ce capteur $fintapis1$ (resp. $fintapis2$) est électromécanique et fonctionne en logique négative (enclenché, le capteur est à 0) et c'est la pièce acheminée par le tapis qui l'enclenche. Pour cette raison, la position du robot ($surt1$ ou $surt2$) correspond à la position de la pièce sur le tapis au moment ou celle-ci relâche le capteur de présence (sans cette précaution, le robot heurterait le capteur en venant saisir la pièce).

La table : il s'agit d'une table circulaire pouvant tourner sur son axe (actionneur $table$). Un capteur nommé $indextable$ est enclenché tous les huitièmes de tour. Dans cette manipulation nous considèrerons que cette table peut occuper huit positions différentes repérées par l'apparition du front montant sur le capteur $indextable$.

Pour les besoins de la manipulation, nous disposons également de sept interrupteurs/poussoirs (a, b, c, d, e, f, g) sur un boîtier d'interface standard, ainsi que 8 diodes électroluminescentes peuvent être allumées ou éteintes grâce aux actionneurs $led1$ à $led8$.

4.3.2 Système de commande

Le système de commande est constitué d'un ordinateur PC fonctionnant sous linux RT. Une carte d'entrée/sortie spécialisée permet de commander le robot. Un logiciel est fourni avec cette carte pour engendrer les mouvements composés du robot (la prise et la pose d'une pièce).

4.4 Travail à effectuer

4.4.1 Cahier des charges et méthodologie

On souhaite obtenir le fonctionnement suivant : des pièces sont acheminées par les deux tapis et doivent être déposées sur la table circulaire qui effectue une rotation d'un quart de tour à chaque fois qu'une pièce est posée. La table a donc une capacité de 4 pièces. Un opérateur peut enlever une à une les pièces de la table circulaire en respectant l'ordre dans lequel elles ont été déposées sur la table. Chaque fois qu'il enlève une pièce de la table, l'opérateur le signale en appuyant sur le poussoir f.

Plutôt que d'aborder l'ensemble du problème de commande frontalement, nous vous proposons une démarche progressive de modélisation et de génération de la commande via le formalisme réseau de Petri. Cette démarche va tirer profit de la relative indépendance des éléments contituant cette maquette. Ainsi, pour chaque élément,

vous devrez :

- Proposer un modèle du procédé de l'élément ;
- Proposer un modèle des objectifs de la commande ;
- Effectuer la synthèse de cette commande, c'est-à-dire réaliser la fusion des modèles du procédé et des objectifs sans oublier de le restreindre aux trajectoires utiles pour les objectifs (utiliser le logiciel TINA, cf. Annexe);
- Effectuer sous TINA une analyse structurelle du modèle de commande obtenu afin de rechercher en particulier les invariants de transitions correspondant au fonctionnement cyclique décrit par les objectifs de la commande ;
- Tester et mettre au point le modèle de commande obtenu sur la maquette (utiliser le logiciel TINA, l'outil générant du code C et le terminal du PC, cf. Annexe).

N'oubliez pas de réaliser des modèles pour les procédés des boutons poussoirs, ils vous seront utiles lors des différentes commandes.

4.4.2 La commande des tapis

Objectif : l'appui sur un poussoir a démarre le tapis et dès qu'une pièce est détectée à la position de saisie par le robot, le tapis doit être arrêté. Le cycle est relancé dès que le poussoir a est à nouveau enclenché.

4.4.3 La commande du robot

Concernant le procédé du robot, noter que la rotation de la base et les mouvements du bras (prise et pose) sont totalement indépendants. Mais vous n'envisagerez pas le cas où deux ordres contradictoires sont engagés simultanément (droite et gauche) et (prise et pose).

Objectif 1 : la prise d'une pièce sur un des tapis est déclenchée par un appui sur le poussoir b (tapis 1) et c (tapis 2). L'autorisation de déposer la pièce sur la table est donnée par un appui sur le poussoir d. Les actions de prise et de pose d'une pièce doivent être effectuées lorsque le robot est à l'arrêt à la position correspondante.

Objectif 2 : modéliser l'alternance de la prise des pièces sur les tapis.

4.4.4 La commande de la table d'indexage

Objectif : la pose d'une pièce est signalée par une action sur le poussoir e, la table doit effectuer un quart de tour immédiatement après le dépôt d'une pièce. Remarque : le dépôt d'une pièce sur la table ne peut se faire que si celle-ci est à l'arrêt.

4.4.5 Commande globale de l'ensemble de la maquette

1. Compte tenu des trois réseaux de Petri de commande que vous avez obtenus aux question précédentes, proposez une démarche pour obtenir le réseau de commande global dans lequel les actions sur les poussoirs b, c, d et e sont remplacés par des éléments structurels reliant les trois réseaux de Petri.
2. Saisir l'ensemble avec le logiciel TINA et expliquez pourquoi ce système de commande doit posséder toutes les bonnes propriétés. Montrer ensuite formellement qu'il les possède bien.
3. Montrer également par l'analyse structurelle que certaines propriétés liées au cahier des charges sont bien présentes dans ce modèle de commande. Il convient au

préalable d'indentifier ces propriétés et d'expliciter comment elles doivent se traduire dans le réseau de Petri.

4. Tester et mettre au point le programme de commande globale sur la maquette.

Annexe de travaux pratiques de SED

1 Récupérer les outils et fichiers nécessaires

L'ensemble des outils, projets et fichiers nécessaires pour chaque manipulation se trouve dans un répertoire commun situé dans

- *home/partage/commun/M1_ISTR_SED* sous Linux,
- *I1/Agamemnon/commun/M1_ISTR_SED* sous Windows (accessible en cliquant sur *Outils/Connecter un lecteur réseau*).

Ce répertoire contient quatre sous-dossiers ASCENSEUR, BCI, ROBOTS, TRAINS-2, correspondant à chacune des manipulations. **Recopier l'ensemble des fichiers dans votre répertoire personnel.**

Le logiciel TINA sous Linux peut récupéré dans */misc*.

2 Synthèse de commande pour les automates avec DESUMA

Pour les TP utilisant le formalisme automate (ascenseur et BCI), la syntèse de commande, c'est-à-dire le produit parallèle entre vos modèles du procédé et de l'objectif, se fait avec le logiciel DESUMA.

Dans DESUMA, le répertoire de travail peut être modifié en cliquant sur *View/Options*.

Il existe deux possibilités pour saisir un automate. Vous pouvez définir chaque état/transition de l'automate par une ligne de code dans le champ 'COMMAND':

- pour définir un état : `s nom_etat`
- pour définir une transition : `trans etat_origine etat_direction nom_evt`

Pour définir l'état initial et les états finaux (marqués) de l'automate, il vous suffit de cocher les cases *Initial* et *Marked* pour ces états dans la fenêtre à droite du graphe (voir figure 4.1). Il est également possible de rentrer toutes les commandes qui décrivent l'automate dans un script, par exemple *autom1.txt* :

```
s etat1 -I
s etat2
s etat3 -M
trans etat1 etat2 a
trans etat1 etat2 b
trans etat2 etat3 c
```

Le script *autom1.txt* peut ensuite être importé sous DESUMA en cliquant sur l'icône en haut à droite (*Batch Window*) puis sur *Open/autom1.txt/Submit*.

Ensuite, vous pouvez réaliser le produit parallèle de plusieurs automates. Pour cela, vous devez avoir un fichier *.fsm* par automate. Puis, cliquez sur *UMDES/Manipulation/Parallel Composition* pour faire apparaître la liste des automates ouverts et sélectionner ceux que vous souhaitez composer. L'automate résultant de cette composition apparaît lorsqu'on

clique ensuite sur *Cancel*.
Pour des problèmes d'affichage, il est recommandé de fermer la fenêtre de droite (liste des états/transitions) avant d'effectuer le calcul du produit d'automates.

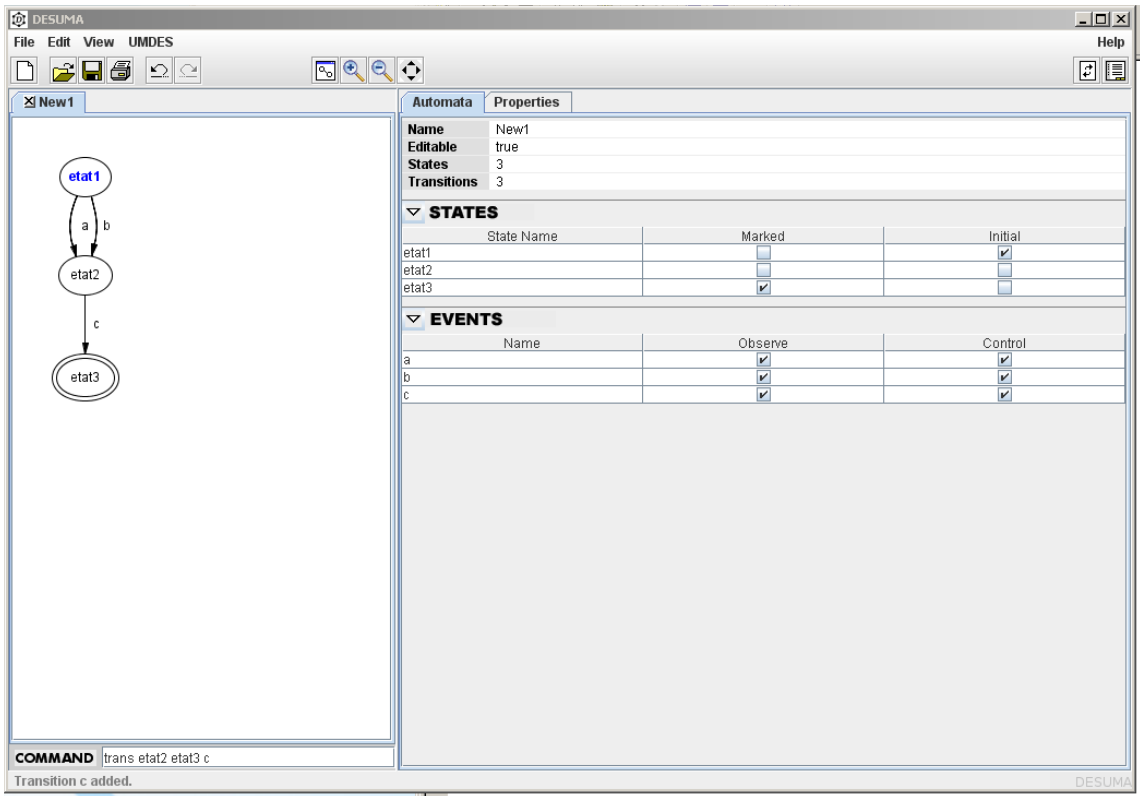


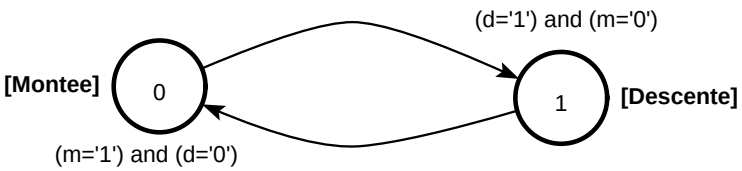
Figure 4.1: Capture d'écran du logiciel DESUMA

3 Saisie sous TINA, fusion des modèles sous TINA et génération automatique de code

3.1 Commande d'un ascenseur

Le dossier ASCENSEUR contient un outil de génération de code VHDL (**moulinette_VHDL**) ainsi que l'ensemble des fichers permettant de prédéfinir un projet (**Ascenseur.qpf**) sous le logiciel Quartus. **Ces fichiers ne doivent pas être modifiés.**

- Votre modèle de commande résultant du produit parallèle (obtenu via DESUMA, cf. section 2) doit être saisi sous le TINA et enregistrer dans ce même répertoire ASCENSEUR sous le nom **commande.adr**. Les expressions portant sur les entrées/sorties du système qui sont associées aux transitions et aux états du modèle devront respecter la syntaxe suivante :



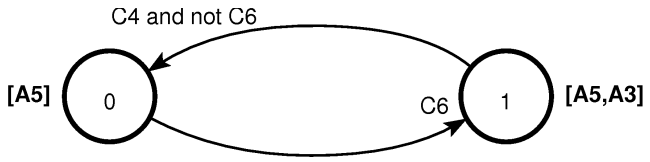
- Pour générer le fichier **commande.vhd** correspondant à votre modèle de commande en langage VHDL, il vous suffit de cliquer sur l'exécutable **moulinette_VHDL**.
- Le projet **Ascenseur.qpf** devra ensuite être ouvert sous Quartus. Il vous faut compiler et générer l'ensemble du projet avant de lancer la programmation. Après avoir vérifié le fonctionnement de votre modèle sur le procédé, vous fermerez l'interface de programmation sans sauver.

  *Remarque : l'interrupteur E_0 (27) correspond à l'entrée **init**, l'interrupteur E_1 (28) à l'entrée **Arret** et l'interrupteur E_2 (29) à l'entrée **Reprise**.*

3.2 Commande d'un BCI

Le dossier BCI contient un outil de génération de code ST (**moulinette_ST**) ainsi qu'un projet **modele_BCI.stu** prédéfini sous le logiciel Unity. **Ces fichiers ne doivent pas être modifiés.**

- Votre modèle de commande résultant du produit parallèle (obtenu via DESUMA, cf. section 2) doit être saisi sous le TINA et enregistrer dans ce même répertoire BCI sous le nom **commande.adr**. Les expressions portant sur les entrées/sorties du système qui sont associées aux transitions et aux états du modèle devront respecter la syntaxe suivante :



- Pour générer le fichier **commande.xst** correspondant à votre modèle de commande en langage ST, il vous suffit de cliquer sur l'exécutable **moulinette_ST**.
- Le projet **modele_BCI.stu** doit ensuite être ouvert sous Unity. Il vous faut créer une nouvelle section et importer votre fichier **commande.xst**. Le projet doit ensuite être généré. Pour exécuter votre programme de commande, se connecter à l'automate (`menu/AP`), ensuite choisir transférer programme du PC vers l'automate (si la génération du projet n'est pas faite elle vous sera imposée ici), et ensuite lancer l'exécution (`Run`).

3.3 Commande d'une station de tri robotisée

Pour cette manipulation, vous n'oublierez pas de démarrer votre ordinateur sous RTAI !

Le dossier ROBOTS contient un outil de génération de code **moulinetteRDP** en langage C.

- Vos modèles du procédé et des objectifs doivent être saisis sous TINA afin d'obtenir votre modèle de commande par la méthode de fusion des transitions (sans oublier de réaliser la restriction aux trajectoires utiles pour l'objectif).
- Le modèle de commande ainsi obtenu doit être enregistré dans ce même répertoire ROBOTS. Les expressions portant sur les entrées/sorties du système qui sont

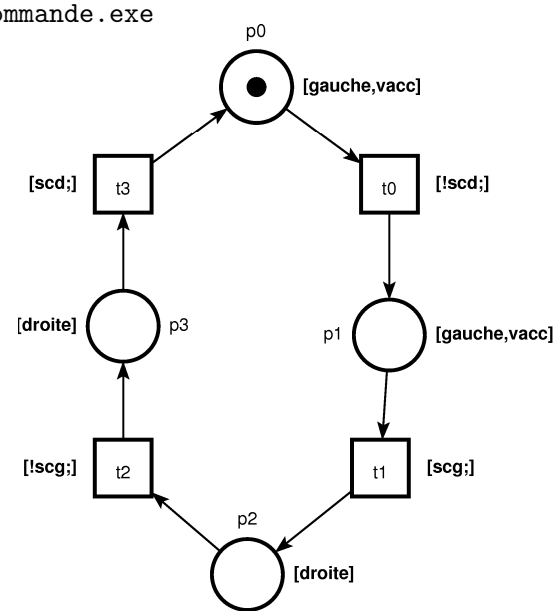
associées aux transitions et aux places du modèle devront respecter la syntaxe décrite sur la figure ci-dessous.

- Pour générer le fichier **commande.c** correspondant à votre modèle de commande en langage C, il vous faut exécuter la "moulinette RDP" :

```
>> ./moulinetteRDP.exe
Entrer le nom du fichier :  nomfichiersansextension
```

- Le fichier **commande.c** doit ensuite être compilé puis exécuté avec les commandes suivantes:

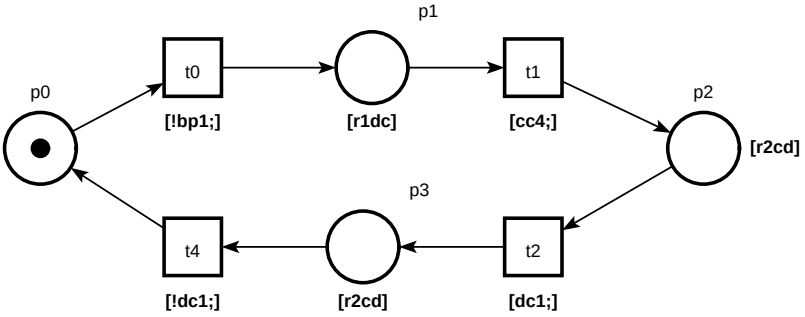
```
>> make nom_pgm=commande
>> sudo run commande.exe
```



3.4 Commande d'un système de transport

Le dossier TRAINS-2 contient un outil de génération de code **moulinetteRDP** en langage C.

- Vos modèles du procédé et des objectifs doivent être saisis sous TINA afin d'obtenir votre modèle de commande par la méthode de fusion des transitions (sans oublier de réaliser la restriction aux trajectoires utiles pour l'objectif).
- Le modèle de commande ainsi obtenu doit être enregistré dans ce même répertoire TRAINS. Les expressions portant sur les entrées/sorties du système qui sont associées aux transitions et aux places du modèle devront respecter la syntaxe décrite sur la figure ci-dessous.



- Pour générer le fichier **commande.c** correspondant à votre modèle de commande en langage C, il vous faut exécuter la "moulinette RDP" :

```
>> ./moulinetteRDP.exe
Entrer le nom du fichier : nomfichiersansextension
```

- Le fichier **commande.c** doit ensuite être compilé puis exécuté avec les commandes suivantes:

```
>> gcc -Wall commande.c -o commande -ltrain -lpci_dask
>> ./commande
```