Commande d'une séquence de traitement de pièces Modélisation par RdP temporel (STA)

I Objectif de la séance de TP

Dans cette séance de travaux pratiques, on vous propose de mettre en œuvre une commande permettant de réaliser une séquence d'opérations pour le traitement de deux pièces. Vous aurez l'occasion de modéliser cette séquence par réseau de Petri temporel, d'analyser votre modèle par la méthode du graphe des classes vue en Cours et en séance de Travaux Dirigés afin de vérifier s'il correspond bien aux spécifications. Votre commande sera ensuite mise en œuvre en langage C.

Un compte-rendu doit être rédigé par chaque binôme. Ce compte-rendu doit présenter vos solutions selon votre compréhension du sujet. Vous y joindrez les modèles RdP saisis sous le logiciel TINA, les graphes des classes correspondants et vos interprétations. Le code en langage C que vous écrivez afin de mettre en œuvre ces solutions doit également être indiqué.

II Description du matériel utilisé : le STA

La maquette du système de traitement automatisé (STA) permet de simuler des fonctions de fabrication de circuits imprimés, de traitement chimique ou de nettoyage de pièces.

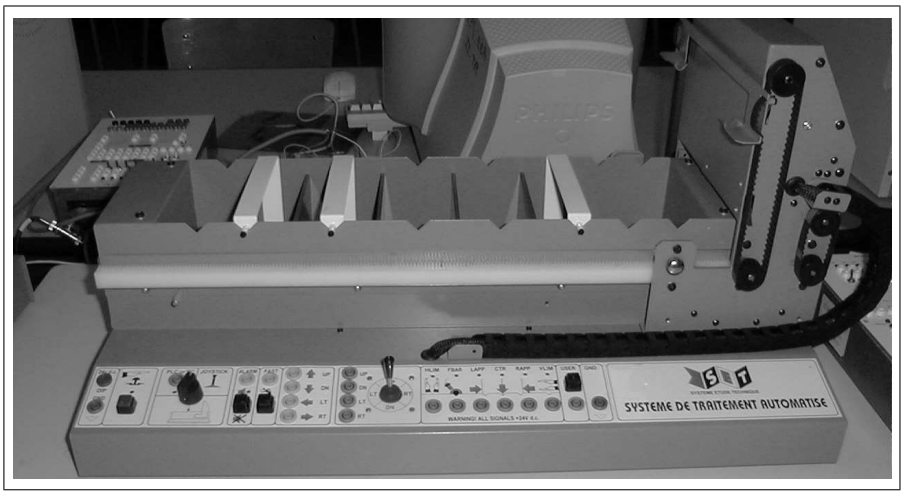

Figure 1: Système de traitement automatisé

Cette maquette est composée de 5 bacs (voir photographie et maquette en salle) ayant chacun au moins un emplacement pour poser le support transporté par le chariot mobile :

- le bac de droite, dit d'entrée possède trois emplacements $e1$, $e2$ et $e3$,
- celui de gauche, dit de sortie, deux emplacements $e7$ et $e8$,
- les bacs intermédiaires possèdent trois emplacement $e4$, $e5$, $e6$ et sont positionnés respectivement du bac d'entrée vers le bac de sortie (de droite à gauche).

Sur la photographie présentée les supports sont respectivement dans le bac d' entrée (au troisième emplacement), dans le bac 3 et dans le bac de sortie.
Pour déplacer les supports, un chariot se déplace horizontalement pour amener celui-ci au-dessus du bac désiré et verticalement pour prendre ou déposer le support. Les capteurs fournissant les entrées du système de commande sont :

- deux capteurs de sur-course fixés sur le chariot; ils définissent les limites de déplacements horizontaux et verticaux : signaux **LIMHOR** et **LIMVER**. Le sur-course horizontal à droite et à gauche active **LIMHOR**, le sur-course vertical en bas et en haut active **LIMVER**.
- trois capteurs permettent de déterminer la position du chariot par rapport à un emplacement pouvant recevoir un support. Pour chaque position, 3 signaux sont délivrés :
  - approche gauche = **APPG**,
  - centre = **CTR**,
  - approche droite = **APPD**.
- un dernier capteur situé sur le transporteur détermine la présence d'un support, signal : **PRESENCE**. Le signal **PRESENCE** est valide lorsque **APPG** est vrai.
- un poussoir **OPERATEUR** permet une intervention de l' utilisateur.

Les actionneurs que le système de commande doit piloter sont:

- vitesse accélérée : **V_ACC**
- commande vers le haut : **HAUT**
- commande vers le bas : **BAS**
- commande vers la gauche : **GAUCHE**
- commande vers la droite : **DROITE**
- alarme constitué d'une LED et d'un signal sonore (qui peut être coupé) : **ALARME**.

Les prises et poses de supports se font simplement à partir des actionneurs **HAUT**, **BAS**, **GAUCHE** et **DROITE**. Après avoir (éventuellement) détecté la présence d'un support - **APPG** et **PRESENCE** actifs-, le chariot sera amené sur l'emplacement du support (capteur **CTR**), la commande **HAUT** permet de saisir le support, **GAUCHE** (ou **DROITE**) permettent d'amener le support à la verticale d'un emplacement (capteur **CTR**) où l'action **BAS** repose le support sur un emplacement; **DROITE** ou **GAUCHE** permettent alors de se dégager (voir figure 2).

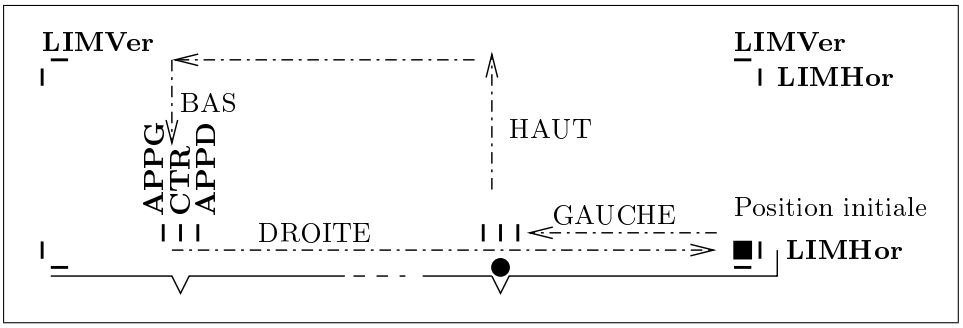

Figure 2: Prise et pose : capteurs et actionneurs mis en jeu

Les supports à traiter sont mis en place manuellement; l'opérateur peut avertir le système de commande de leur présence par une action sur le poussoir **OPERATEUR**.

Le calculateur de type PC est connecté par un réseau de communication à une carte d'interfaçage reliée aux capteurs et actionneurs de la maquette. Une bibliothèque est fournie pour gérer tous les problèmes de lecture des valeurs de capteurs et d'écriture des valeurs des sorties sur les actionneurs. La commande doit être spécifiée en langage C.

III Travail à réaliser

Deux pièces se trouvent respectivement aux emplacements 1 et 2 dans le bac d'entrée. Elles doivent subir chacune trois opérations de trempage.

- La pièce $p1$ doit tout d'abord subir l'opération $o1$ à l'emplacement $e3$, l'opération $o3$ à l'emplacement $e5$ puis l'opération $o5$ à l'emplacement $e7$.
- La pièce $p2$ doit d'abord subir l'opération $o2$ à l'emplacement $e4$, l'opération $o4$ à l'emplacement $e6$ puis l'opération $o6$ à l'emplacement $e8$.

Une durée incertaine $d_i = [d_{i_{min}}, d_{i_{max}}]$ est associée à chaque opération o_i et représente la durée d'opération avec une tolérance associée. Lorsque cette durée a fini de s'écouler, une alarme est déclenchée. Le chariot mobile permet d'amener les pièces aux différents emplacements afin qu'elles subissent les opérations souhaitées. En particulier, lorsqu'une opération o_i est en cours, le chariot devra se déplacer au bon emplacement et retirer la pièce avant que l'alarme ne se déclenche. Le chariot se déplace d'un emplacement en x unité de temps.

III.1 Modélisation et analyse de la réalisation d'une opération

On cherche à représenter de manière générique la réalisation d'une opération o_i en tenant compte de sa durée incertaine d_i, de la durée x de déplacement du chariot, du temps de prise/pose d'une pièce que l'on notera y.

1. Pour une valeur d'opération d_i choisie arbitrairement, donner le réseau de Petri temporel représentant la réalisation d'une opération o_i pour une pièce. Mettre en œuvre une commande en langage C afin de déterminer les durées x et y sur la maquette (se reporter à l'annexe IV.2).
2. Analyser ce réseau sous le logiciel TINA par la méthode du graphe des classes (se reporter à l'annexe IV.1).

III.2 Modélisation et analyse des premières opérations de chaque pièce

La durée incertaine associée à la première opération o_1 que doit subir la pièce p_1 est $d_1 = [15, 16]$. La durée incertaine associée à l'opération o_2 que doit subir la pièce p_2 est $d_2 = [20, 22]$.

1. Représenter par un réseau de Petri temporel la commande permettant d'effectuer les premières opérations o_1 et o_2 des deux pièces en un minimum de temps.
2. Analyser le modèle proposé sous le logiciel TINA par la méthode du graphe des classes.
3. Ajuster les intervalles temporels associés aux transitions du modèle RdP afin de respecter la durée d'opération et d'éviter de déclencher les alarmes de fin d'opération.

III.3 Modélisation des séquences d'opérations des deux pièces

La séquence complète d'opérations pour chaque pièce doit maintenant être considérée. Les durées d'opération et les tolérances associées sont les suivantes : $d_3 = [20, 22]$, $d_4 = [26, 28]$, $d_5 = [16, 17]$, $d_6 = [20, 22]$. L'objectif est de modéliser par réseau de Petri temporel la séquence complète d'opérations des deux pièces $p1$ et $p2$ afin de minimiser le temps de cycle global et d'éviter de déclencher les alarmes associées aux différentes opérations.

1. Analyser la séquence possible d'opérations en représentant les contraintes temporelles dues au chariot (temps de déplacement, temps de prise/pose) et aux différentes opérations des deux pièces. Cette analyse pourra se faire mathématiquement à l'aide d'un système d'inéquations ou graphiquement avec un chronogramme. Afin d'éviter que les alarmes se déclenchent, il est possible de retarder certaines opérations ou d'interrompre le déplacement du chariot.
2. Donner le réseau de Petri temporel représentant la commande globale des deux pièces. Ajuster les intervalles de sensibilisation des transitions à l'aide de l'analyse temporelle effectuée à la question précédente.
3. Vérifier que les alarmes de fin d'opérations ne se déclenchent pas en effectuant l'analyse du modèle sous TINA à l'aide du graphe des classes.
4. Implémenter votre commande en langage C et vérifier le bon fonctionnement du système de commande. Se reporter à l'annexe IV.2 pour la mise en œuvre d'un exemple de réseau de Petri sur la maquette étudiée.

IV Annexe

Consignes pour démarrer :

Ne pas oublier de démarrer l'ordinateur sous Fedora version 2.6.5 pour avoir accès aux bibliothèques qui sont sur le serveur. Copier ensuite tous les fichiers nécessaires de */home/partage/commun/M2_ISTR/MTA-RDP/manip-STA* dans votre répertoire de travail.

IV.1 Saisie et analyse d'un modèle Réseau de Petri sous le logiciel TINA

Récupérer le logiciel TINA sous Linux dans *usr/local/tina-3.4.4*.
Démarrer le logiciel par l'icône "nd".
Sauvegarder un modèle ".ndr" dans votre répertoire de travail.
Les trois boutons "gauche", "centre" et "droit" de la souris peuvent être utilisés pour saisir les éléments du réseau de Petri :

- centre : crée une place,
- ctrl + centre : crée une transition,
- centre sur un élément : débute un arc qui doit être étiré jusqu'à l'élément destination en maintenant appuyé,
- gauche : sélectionne un objet,
- droit sur un objet : permet d'éditer les propriétés de l'objet (nom, poids, marquage, intervalle temporel, etc.)

Lorsque vous avez saisi le réseau, sauver et effectuer une analyse par énumération des marquages en calculant le graphe des classes (*tools/reachability analysis*, choisir l'option "*State Classes Preserving*" et le format "*verbose*"). Analyser le fichier texte produit et conclure quant aux bonnes propriétés du réseau. Pour visualiser le graphe des classes sous forme d'automate, le recalculer en choisissant le format "*lts (.aut)*". Cliquer droit sur la fenêtre ouverte "open file in nd" puis dans la nouvelle fenêtre, choisir l'option d'affichage dans "*file/draw*".

IV.2 Mise en œuvre d'une commande RdP du STA en langage C

Pour mettre en œuvre la commande RdP désirée vous pouvez vous inspirer du code donné pour l'exemple ci-dessous. Une commande de la maquette STA (*sta-macsed_rdp.c*) est donnée sur la figure 3. La mise en œuvre de cette commande utilise la méthode de description des transitions.

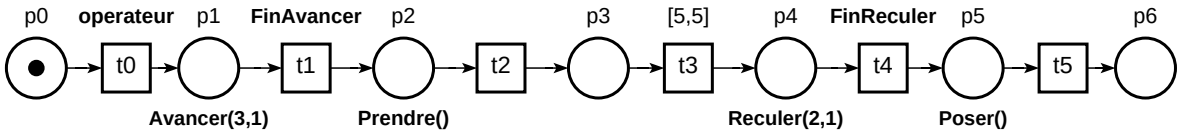

Figure 3: Exemple de commande RdP du STA

Les lignes de code propres à cette commande sont entourées de lignes en commentaires dans le programme. Les fonctions du squelette (Avancer, Prendre, Reculer et Poser) sont reprises pour l'illustration. Pour la temporisation, un objet de type *time_t* (disponible dans la librairie *time.h* qui est appelée dans *entreesortie_sta.h*) ainsi que la fonction *difftime* sont utilisés. Il est bien sûr possible d'utiliser la structure temporisateur prédéfinie.