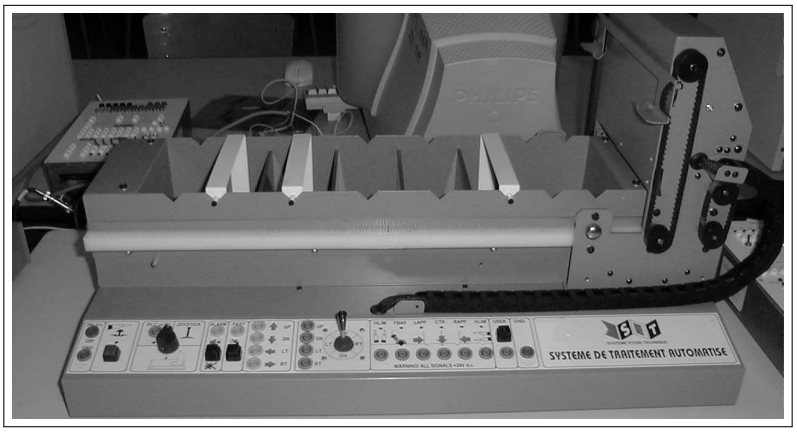
Université Paul Sabatier

Modèle Temporel avancé

- TP : Système de Traitement Automatisé -

Auteurs :
Lucien Rakotomalala
David Tocaven

Encadrant :
Pauline Ribot
Euriell Le Corronc
Michel Combacau



Faculté des Sciences et d'Ingénierie

EEA Toulouse

4 mars 2018

Table des matières

1 Modélisation et analyse de la réalisation d'une opération

Nous allons, dans un premier temps, réaliser une modélisation par réseau de Petri temporel de la réalisation d'une opération. Cette modélisation sera générique à la réalisation de toute opération O_i. Ensuite, nous réaliserons un code C qui permet d'estimer les durées des différentes opérations. Finalement, nous analyserons le réseau de Petri à l'aide de *TINA 2.8.4*.

1.1 Modèle réseau de Petri temporel d'une opération

Nous avons, pour modélisation générique d'une opération, considéré que le chariot de déplacement se trouve en bas. Voici le réseau de Petri temporel (voir figure 1.1) :

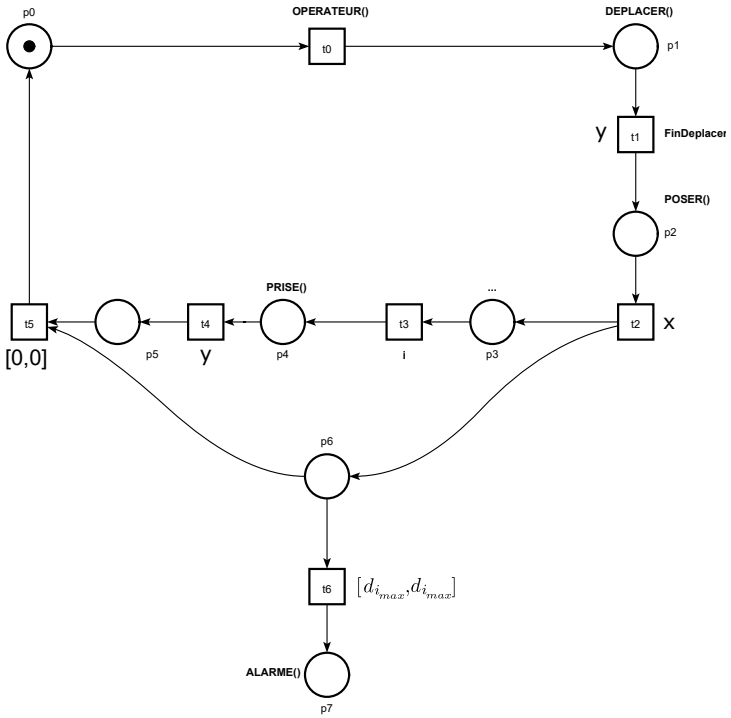


FIGURE 1.1 – Modèle réseau de Petri temporel générique d'une opération.

Nous considérerons que l'action *DEPLACER()* et l'événement *FinDeplacer* correspondent à, respectivement, *AVANCER()* et *FinAvancer* si le chariot est à droite de l'emplacement de l'opération o_i ou *RECULER()* et *FinReculer* si celui-ci est à gauche.

Le marquage initial est constitué d'un unique jeton sur la place p_0. Ce jeton, une fois que la transition t_0 est sensibilisée et tirée (pour cela il faut que l'événement *OPERATEUR()* est eu lieu), est en p_1. Le chariot se déplace tant qu'il y a un jeton en p_1. Le jeton reste en p_1 jusqu'à ce que le charriot arrive à destination, c'est-à-dire que *FinDeplacer* se déclenche. Lorsque cet événement ce produit, la transition t_1 est sensibilisée et tirée (le déplacement prend un temps y qui est représenté sur la transition t_1). Ensuite, un jeton marque la place

p_2 ce qui déclenche l'action *POSER()*. Cette action prend un temps x et celui-ci est représenté sur la transition t_2. Une fois le temps x écoulé, la transition t_2 est tirée et les places p_3 et p_6 sont marquées d'un jeton chacun.

À partir de cet état, il y a deux jetons dans le réseau : un permet de décrire le comportement du chariot et un autre, celui qui marque p_6, permet de déclencher l'alarme si la pièce qui subit l'opération o_i n'est pas reprise avant le temps maximal de l'opération. En effet, le jeton présent en p_6, au bout d'un temps $d_{i_{max}}$, va être consommé par la transition t_6 et un jeton va marquer p_7. Ceci déclenchera l'action *ALARME()*. Il faut donc que le jeton présent en p_3 arrive en p_5 en moins de $d_{i_{max}}$ unités de temps pour que l'alarme ne se déclenche pas. De cette façon, le tir de la transition t_5, qui nécessite et consomme un jeton en p_6 et un jeton en p_5, empêchera l'alarme de sonner et permettra d'effectuer une nouvelle opération (retour au marquage initial). La place p_3 à un événement "...", cela représente la possibilité d'effectuer n'importe(s) quelle(s) action(s) et de revenir à l'emplacement de l'opération o_i, de façon à ce que l'action en p_4, *PRISE()*, de durée y, permette de récupérer la pièce. La transition t_3 est marquée de la temporisation i. Celle-ci représente le temps de(s) action(s) de la place p_4 et/ou un temps d'attente afin que l'on récupère la pièce à la fin de l'opération $_i$. Ainsi, si l'on souhaite que l'alarme ne sonne pas, il faut que $i + y < d_{i_{max}}$.

1.2 Estimation de Prise/Pose et Avance/Recule

Voir annexe 3.4, page 13, avec résultats des mesures

Nous avons maintenant besoin d'identifier le temps de *AVANCER()* (égal à celui de *RECULER()*) que l'on appelait précédemment x et de *PRISE()* (équivalent à celui de *POSE()*) appelait y. Pour cela, nous avons créé, à partir d'un code fourni, un code permettant de mesurer les temps x et y. Pour mesurer le temps d'une action, nous avons stocké le temps du PC à l'instant du début de l'action, puis nous avons stocké le temps à la fin de celle-ci et avons affiché la soustraction des deux temps sur le terminal. Nous avons ainsi déterminé que :

$$x = \left[1 \quad 1\right] \text{ seconde} \tag{1.1}$$

$$y = \left[3 \quad 3\right] \text{ secondes} \tag{1.2}$$

$$\tag{1.3}$$

1.3 Analyse du modèle

Grâce aux mesures précédentes, nous avons pu remplacer x et y par des valeurs temporelles sur le modèle générique. Nous avons choisi arbitrairement les valeurs de $d_{i_{min}} = 13$ secondes et $d_{i_{max}} = 16$ secondes, respectivement le temps minimal de l'opération et le temps maximal de l'opération o_i. Ainsi, nous avons la condition suivante qui doit être respectée $i < d_{i_{max}} - y$, soit $i < 13$ secondes. Donc il "reste" moins de 13 secondes afin de réaliser d'autres opérations. Nous allons fixé une temporisation $i = \left[12 \quad 12\right]$ secondes pour étudier le réseau.

Figure 1.2, voici le nouveau réseau de Petri temporel. Une analyse à l'aide de TINA nous a permis de

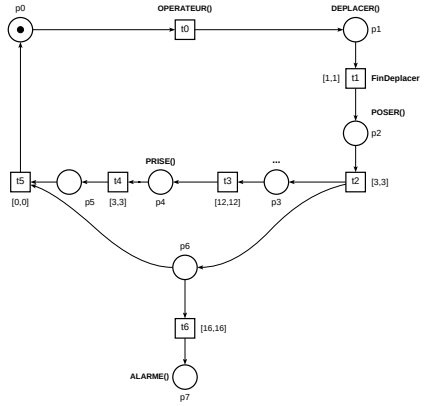


FIGURE 1.2 – Modèle réseau de Petri générique d'une opération avec les temps estimés

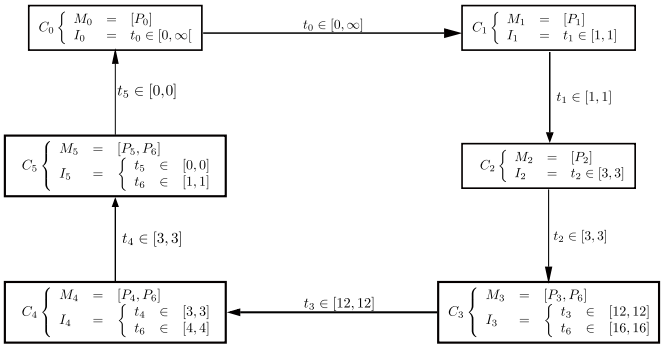


FIGURE 1.3 – Diagramme de classe.

déterminer les différentes classes du réseau. L'automate est présenté figure 1.3. Nous avons pu aussi extraire les propriétés suivantes grâce à TINA.

— Le RdPT (Réseau de Petri Temporisé) n'est pas vivant.
— La transition t_6 est non vivante.
— LE RdPT est borné.

2 Modélisation et analyse des premières opérations de chaque pièce

Maintenant que nous connaissons un modèle valide pour une opération ainsi que les temps nécessaires au déplacement du charriot sur l'axe vertical et horizontal, nous allons pouvoir commencer à modéliser le travail du *STA* sur deux opérations.

Nous allons, dans un premier temps, effectuer une modélisation en RdP Temporel d'une commande de deux opérations suite à quoi, nous en effectuerons une analyse grâce à une version de *TINA* identique que dans le chapitre 1. Nous utiliserons cette analyse pour déterminer les intervalles d'attentes et le meilleur ordonnancement possible pour ne pas déclencher l'alarme.

2.1 Réseau de PETRI Temporel de commande de deux opérations

A partir du modèle générique établit en 1.1, nous avons obtenu, pour la commande de opérations O_1 et O_2 le modèle RdP Temporel en figure 2.1.

Dans ce réseau, nous pouvons identifier tout d'abord la ressemblance avec le modèle générique (en figure 1.1) : les places p_4 et p_{12} sont les représentations de la place p_2 dans le modèle générique. Elles seront donc suivi, dans les modèle Temporel que nous analyseront, d'une transition qui contient le temps des opérations *Poser*. Il en va de même pour les places p_2, p_{10}, p_5 et p_{13} qui contiennent l'opération *Prendre*, elles seront suivi d'une transition contenant une intervalle de temps y.

Séparation du modèle Nous pouvons séparer ce modèle complexe en deux ensembles de places :

- l'ensemble $P_1 = \{p_2, p_3, p_4, p_5, p_6, p_7, p_{20}, p_{18}\}$ est utilisé pour emmener la pièce $p1$ du bac $e1$ (son bac initial) vers le bac $e3$ dans lequel elle subit l'opération O_1. Cet ensemble est lié avec les deux places p_{20} et p_{18} qui modélisent l'alarme liée à l'opération O_1.
- l'ensemble $P_2 = \{p_{10}, p_{11}, p_{12}p_{13}, p_{14}, p_{15}, p_{16}\}$ modélise le transport de la pièce $p2$ de $e2$ (son bac initial) vers $e3$, bac dans lequel elle subit l'opération O_2, puis du transport de $e5$ vers $e7$ (pour prévoir les prochaines opérations). L'alarme de l'opération est enclenchée par la place p_{17}, qui est liée au reste de l'ensemble P_2 par la place p_{19}.

Liaison entre les ensembles Les places p_9, p_{21} et p_{23}, places qui se situent entre les deux ensembles P_1 et P_2, sont utilisées pour effectuer les passages entre les 2 ensembles. De même, nous avons deux places p_{22} et p_{24} qui font la liaison entre les deux ensembles. Celles-ci ne servent pas à amener le charriot d'une pièce à l'autre mais à le faire attendre le temps nécessaire avant la fin d'une opération et donc la récupération d'une pièce.

Nous notons aussi les transitions t_{15} et t_{20} sont les représentations de la transition t_5 dans le modèle générique. Elles permettent dans ce contexte de synchroniser la prise de la pièce et l'arrêt du compteur de l'alarme. Comme dans le modèle générique, ces transitions ont une intervalle de temps $[0; 0]$. Cela oblige celles-ci, une fois que les jetons arrivant dans les places en amont les sensibilises, à être immédiatement tirées. Ainsi, la transition précédent l'alarme est désensibilisée immédiatement après que la pièce ait été retirer de son emplacement.

Ordonnancement Il est aussi à noter que nous avons choisi d'effectuer l'opération de la pièce $p1$ avant celle de $p2$ et cela pour des raisons temporelles. Nous avons remarqué, dans une étude préliminaire à la construction de ce réseau, que cet ordonnancement était possible alors qu'une inversion de l'ordre du traitement des pièces donne obligatoirement le retentissement d'une alarme (notamment dans la suite du TP).

Nous avons maintenant à notre disposition une commande à appliquer sur le *STA*. Toutefois, avant de passer à l'implémentation, nous allons utiliser l'outil *TINA* pour analyser le modèle ainsi obtenu.

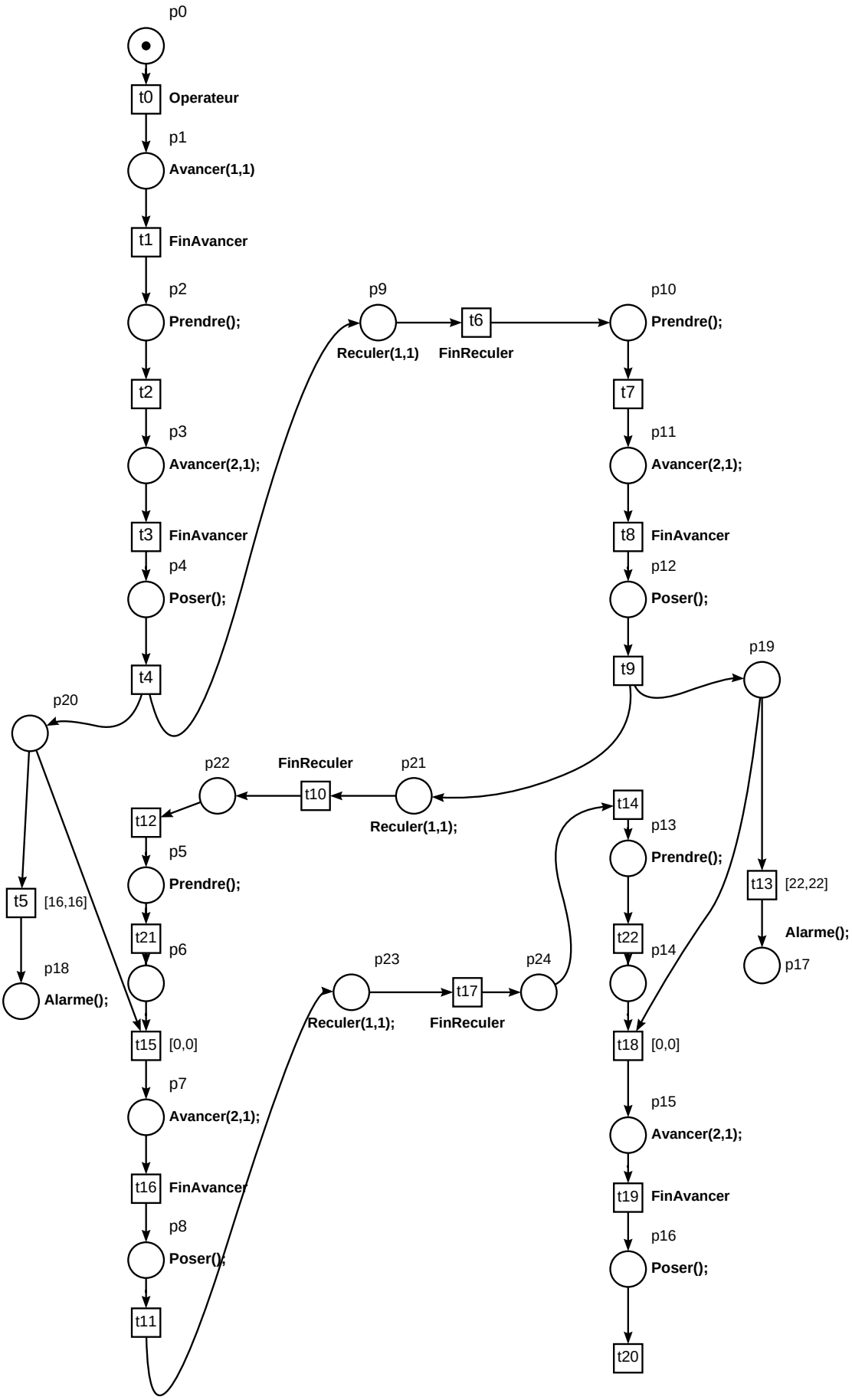


FIGURE 2.1 – Réseau de PETRI Temporel pour la commande de 2 opérations

2.2 Analyse du modèle avec TINA

L'analyse temporelle de réseau nécessite un réseau tel que nous vous présentons en figure 2.2. Dans ce type de réseau, nous avons choisi de ne plus utiliser des noms sur les évènements mais plutôt des intervalles de temps. Ces intervalles ont été établies dans la section 1.2.

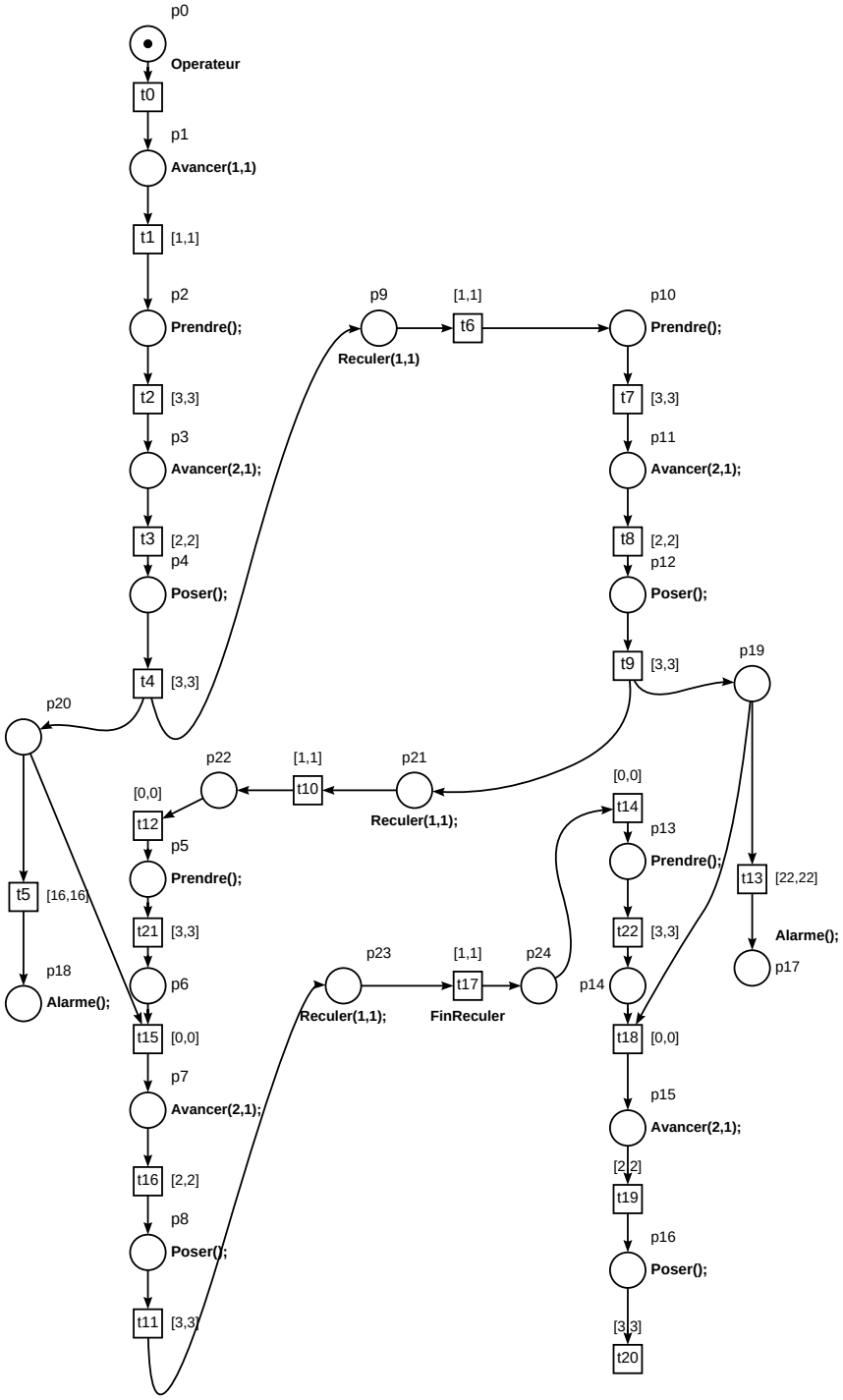


FIGURE 2.2 – Réseau de PETRI Temporel pour l'analyse de 2 opérations

Avec la version de *TINA* adéquate, nous avons réalisé la construction d'un graphe des classes d'états. Ce graphe comporte 22 classes que nous n'avons pas représenté graphiquement, cependant vous pouvez trouver le rapport d'analyse du logiciel en annexe 3.1.

Vivacité du réseau Dans cette analyse, nous nous référerons dans un premier temps à cette bonne propriété. En effet, le résultat obtenu (disponible à la ligne 244) nous indique que les 2 transitions t_5 et t_{13} ne sont jamais franchies. Ceci signifie que les places p_{18} et p_{17} ne sont jamais marquées, car le seul moyen existant pour qu'un jeton soit dans ces places est en franchissant respectivement leurs transitions. Donc, l'alarme n'est jamais activée.

Graphe des classes Une analyse des places accessibles à partir du graphe des classes nous confirme ce que la vivacité du réseau nous avait indiqué : les places p_{18} et p_{17} ne sont jamais marquées.

Cette analyse nous apporte aussi une information que nous pourrons utiliser dans la prochaine section. Il s'agit de l'intervalle temporelle restante avant le franchissement des transitions t_5 et t_{13} avec lequel il est possible de connaitre l'intervalle temporelle passée dans l'opération O_1 et O_2, respectivement. Nous pouvons observer aux lignes 153 et 194 ces intervalles, elles sont de $[3;3]$ pour t_3 et de $[9;9]$ pour t_{13}.

Intervalles de temps des opérations Intéressons nous maintenant aux temps d'exécution des opérations. En regardant les intervalles de temps que doivent respecter O_1 et O_2, nous pouvons connaitre le minimum et le maximum de temps permis par les opérations. De plus, nous avons à notre disposition le temps passé dans les opérations à l'aide des transitions t_3 et t_{13}, en regardant dans le graphe des classes les intervalles de temps durant lesquelles elles peuvent être franchies.

Donc, si l'intervalle temporelle des transitions atteint $d_{i_{max}} - d_{i_{min}}$ u.t, alors la pièce doit être sorti du bac. Nous allons donc, dans la prochaine section, faire en sorte que les dernières intervalles temporelles t_3 et t_{13} soient comprises entre : $[0; d_{i_{max}} - d_{i_{min}}]$.

2.3 Mise au point des Intervalles Temporelles

Avec la conclusion précédente, nous avons pu déterminer l'ajustement nécessaire. Toutefois, nous ne pouvons pas ajuster toutes les intervalles d'un seul coup, nous devons les placer l'une après l'autre, dans l'ordre dans lequel les opérations sont lancées.

Opération O_1 A partir du graphe des classes établit à partir du modèle 2.2, nous avons le dernier intervalle temporelle atteint par la transition t_5 à la classe 12 du graphe disponible en annexe page 17 qui est :

$$3 \leq t_5 \leq 3 \tag{2.1}$$

Nous savons que pour cette opération, la différence entre la durée maximale d_{max} et la durée minimale d_{min} d'une opération, qui est : $d_{max} - d_{min} = 16 - 15 = 1$, nous savons que l'intervalle 2.1 doit être inférieure à 1, sans atteindre 0. De cette manière, nous pouvons assurer que la pièce est restée dans l'opération suffisamment longtemps et n'a pas fait sonner l'alarme. Pour obtenir ce résultat sur t_5, nous modifions l'intervalle temporelle de la transition t_{12} de cette façon :

$$t_{12} \in [0;0] \text{ devient : } t_{12} \in [2;2] \tag{2.2}$$

Une fois cette transition modifiée, nous avons retracé un graphe des classes à partir duquel nous pourrons ajuster l'opération suivante. Une partie de l'analyse d'accessibilité est disponible en annexe, page 21.

Opération Nous pouvons observer, dans le nouveau graphe de classes, l'intervalle atteint pour la transition t_{13} à la classe 18 (ligne 142) :

$$7 \leq t_{13} \leq 7 \tag{2.3}$$

En suivant le même raisonnement que dans le paragraphe précédent, en sachant qu'ici nous avons : $d_{max} - d_{min} = 22 - 20 = 2$. Il vient donc comme intervalle temporelle sur la transition t_{14}

$$t_{14} \in [0;0] \text{ devient : } t_{14} \in [5;5] \tag{2.4}$$

Modèle fini Maintenant que les intervalles vides ont été réglées, nous pouvons vous présenter le modèle complet. Il se trouve en figure (2.3).

Nous pouvons suite à ce modèle, effectuer une analyse. Cette fois-ci, nous effectuerons un dernier graphe des classes, qui correspond au graphe utilisé pour ajuster l'opération 2, disponible en annexe en page 23. Nous pouvons voir que les places p_{17} et p_{18} n'apparaissent dans aucune classe et donc, ne sont jamais marquées. Ainsi, l'analyse du réseau montre qu'avec cette commande, l'alarme du *STA* ne va pas sonner.

Pour terminer la commande du système, nous devons enlever les transitions temporelles pour les remplacer par les signaux que nous pouvons récupérer. Nous laissons les transitions temporelles de commande, c'est-à-dire celles qui permettent d'activer les alarmes et qui permettent au charriot d'attendre les fins d'opérations.

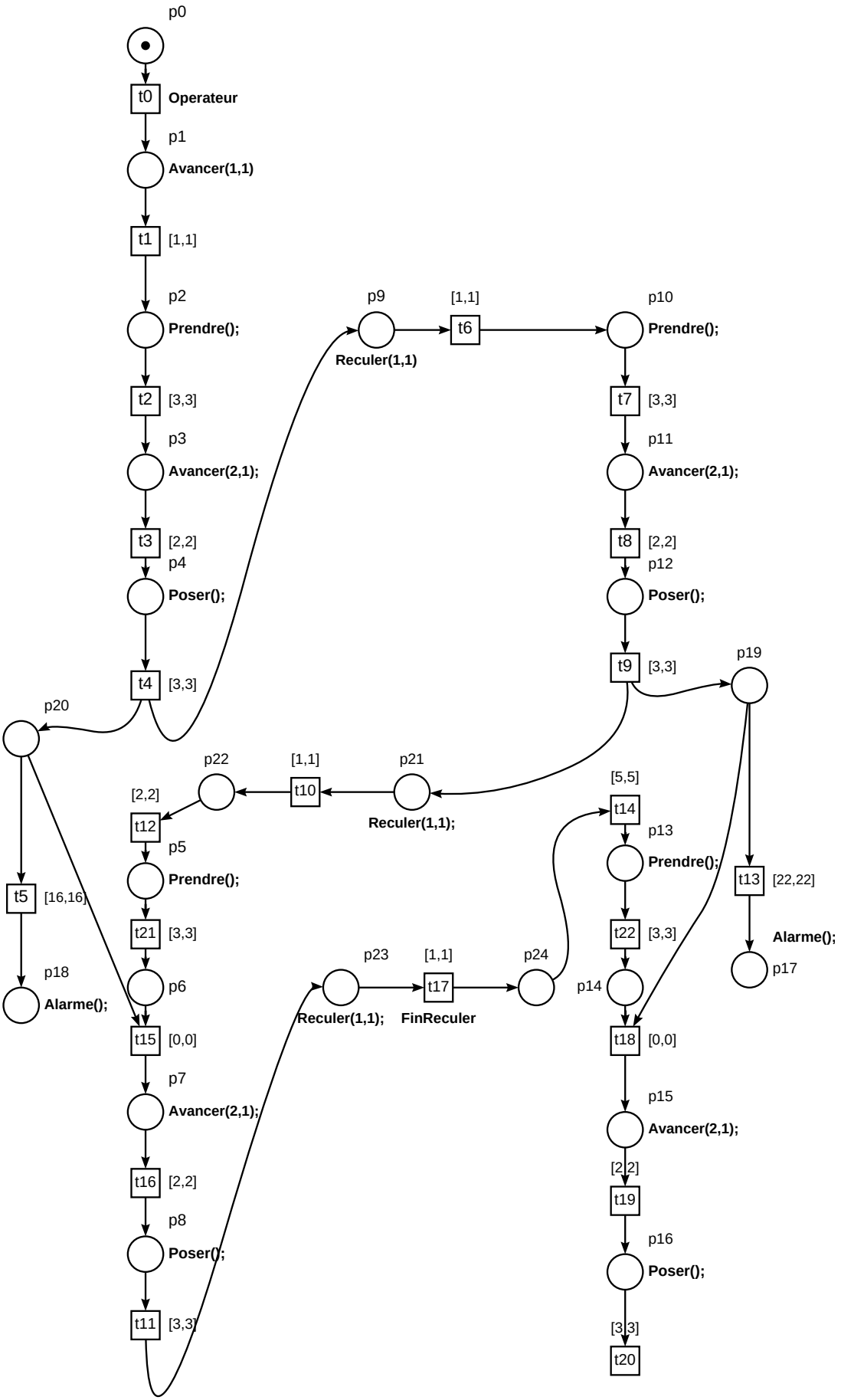


FIGURE 2.3 – Réseau de PETRI Temporel Ajusté de 2 opérations

3 Modélisation des séquences d'opérations des deux pièces

Dans cette partie, l'objet de l'étude se porte sur la réalisation de 6 opérations, 3 par pièces (les opérations o_1, o_3, o_5 pour p_1 et o_2, o_4, o_6 pour p_2). Il faut, dans un premier temps, trouver une séquence optimale telle que toutes les opérations soient effectuées en un temps minimum sans déclencher l'alarme. Ensuite, nous devons réaliser une commande en réseau de Petri temporel pour effectuer les opérations puis l'analyser à l'aide d'un graphe des classes. Dans un dernier temps, nous avons réaliser la commande en langage C.

3.1 Analyse d'ordonnancement

Nous avons décidé d'étudier l'ordonnancement optimal par un chronogramme car nous avons trouvé l'approche graphique plus intuitive au vu du faible nombre d'opérations. Nous avons réalisé cet ordonnancement sur Excel en prenant pour échelle *une case égale une seconde*. Le chronogramme (figure 3.1) est coupé en trois parties pour des raisons de visibilité : une ligne est réservée aux opérations liées à la pièce p_2, une seconde aux actions du charriot et une dernière aux opérations de p_1.

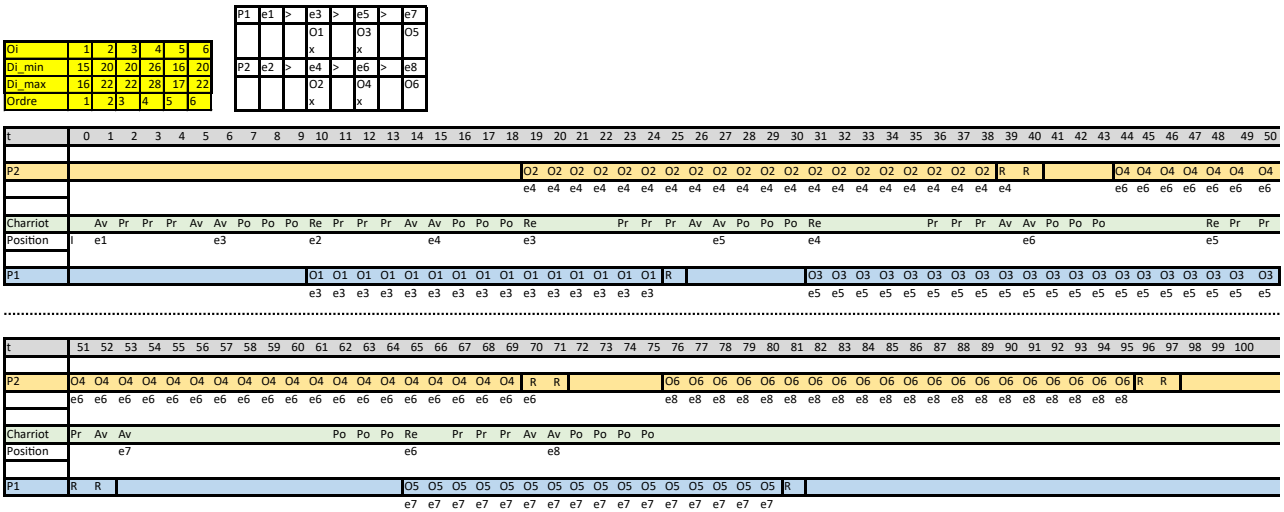


FIGURE 3.1 – Capture du tableur de calcul de l'ordonnancement.

Nous avons positionné les opérations au plus tôt, en vérifiant qu'elles n'empêchent pas de récupérer les pièces avant leurs alarmes respectives. Par exemple, l'opération o_5 pourrait commencer à partir de la 57^{e} seconde mais il serait impossible de retirer les deux pièces avant le déclenchement de l'alarme. Nous avons donc décalé o_5 de façon à ce que, ni la pose, ni la prise de la pièce p_1 ne perturbe la prise et la pose de p_2. De plus, nous prenons pour hypothèse que les deux dernières opérations o_5 et o_6 n'ont pas de date récupération maximale et peuvent rester respectivement en e_7 et en e_8 car ces deux emplacements représentant le bac de sortie. Si nous avions pris le partie de les ramener à leurs emplacements initiaux, cela n'aurait pas été possible avec l'ordonnancement actuel. Il aurait fallu organiser les opérations différemment de façon à avoir le temps de ramener les pièces au début ce qui nécessite beaucoup de temps à cause de la distance de déplacement importante.

3.2 Réseau de Petri de commande

Nous avons transformé la séquence d'actions de la ligne *Charriot* de l'ordonnancement précédent (voir figure 3.1) en un réseau de Petri temporel. Voici le réseau de Petri temporel qui contient la commande (figure 3.2) :

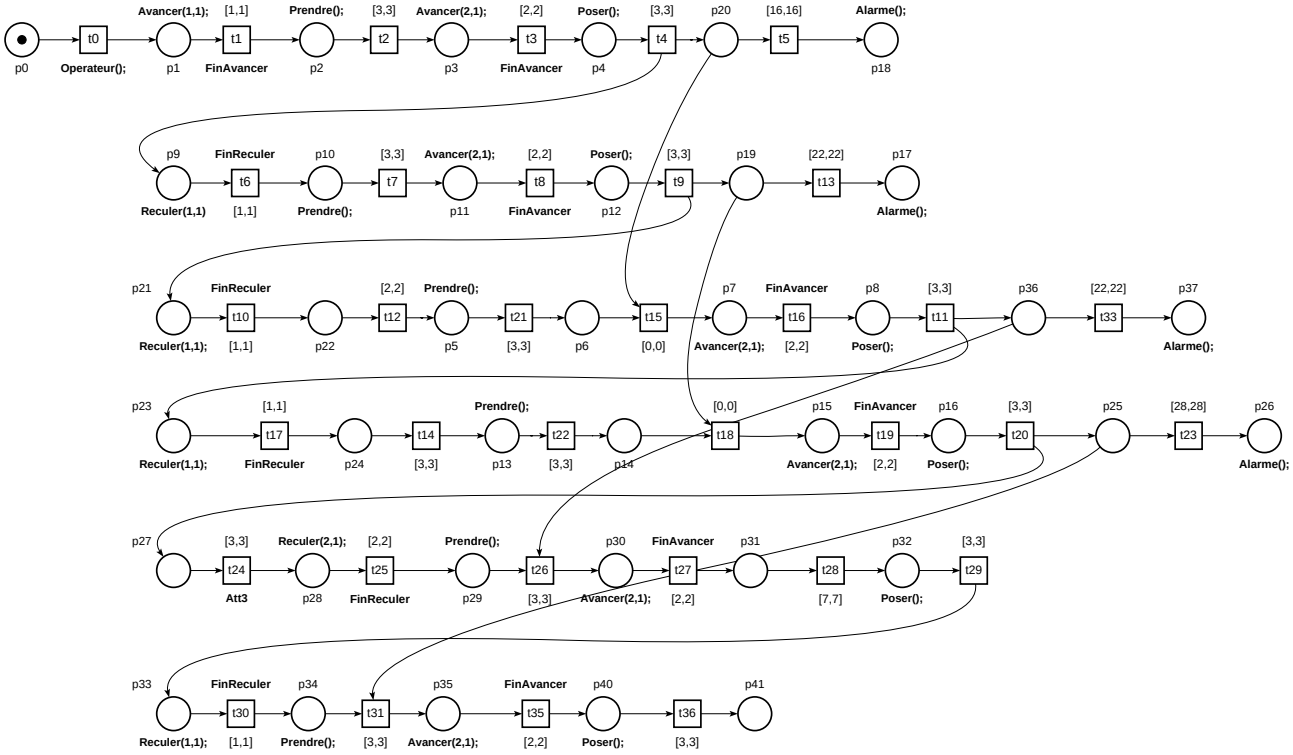


FIGURE 3.2 – Modèle réseau de Petri temporel de commande du STA.

Le réseau de Petri Temporel de commande ci-dessus est basé sur le modèle de l'exercice précédent auquel il a été ajouté la séquence pour les 4 autres opérations avec les déclenchement d'alarmes nécessaires. Nous avons représenté une ligne par lancement d'opération. Comme expliqué précédemment, nous avons choisi de ne pas considérer les temporisations des opérations o_5 et o_6, c'est pourquoi les deux dernières lignes du réseau ne contiennent pas de vérification que la pièce est retirée avant le temps maximal de chaque opération. Les transitions t_7, t_{14} et t_{28} contiennent des temporisations afin que le charriot attende de façon à intervenir au bon moment.

3.3 Analyse par graphe de classes

Afin d'étudier la commande, nous avons ajouté au modèle de commande précédent (figure 3.2) des temporisations représentants les temps des déplacements (*Avancer()* et *Reculer()*) et les temps de manipulation de pièce (*Prise()* et *Pose()*). Nous obtenons le modèle réseau de Petri temporel présenté figure 3.3.

Nous avons fait une analyse de ce réseau par la méthode du graphe des classes à l'aide de *TINA*. Cela donne un graphe à 40 classes. Le résultat de l'analyse est disponible en annexe page 27. Nous avons étudié le graphe afin de vérifier que les places activant les alarmes ne soient jamais marquées. Il s'agit des places p_{18} pour l'opération o_1, p_{17} pour l'opération o_2, p_{37} pour l'opération o_3 et p_{26} pour l'opération o_4. Aucune des places déclenchant l'alarme n'apparait dans une classe du graphe, donc notre commande ne commet pas d'erreur d'ordonnancement.

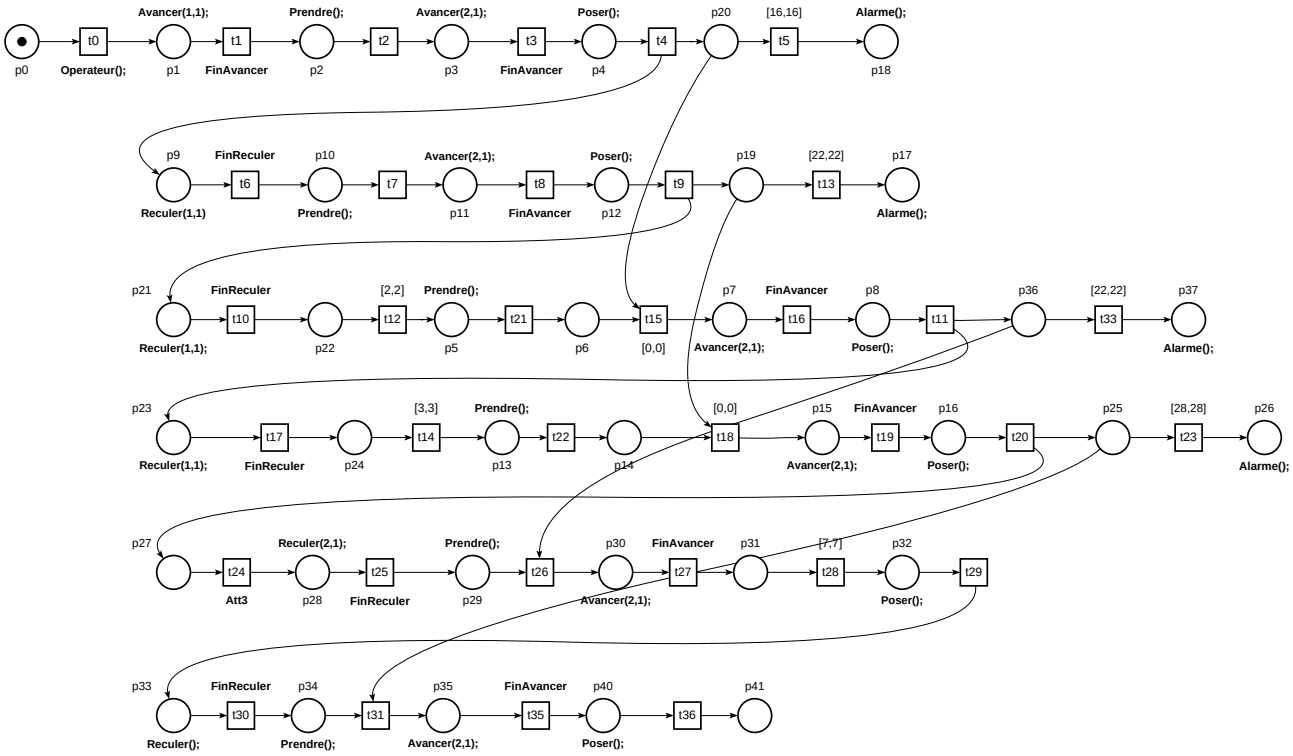


FIGURE 3.3 – Modèle réseau de Petri temporel de d'analyse de la commande du STA.

3.4 Implémentation

Nous n'avons pas eu le temps de réaliser l'implémentation en C de notre modèle de commande. Si nous avions eu le temps, nous aurions fait une implémentation de même type que celle proposé en annexe de l'énoncé. Nous aurions créé 3 temporisations pour les transitions t_7, t_{14} et t_{28} afin que la commande attende et d'autres pour les transitions t_5, t_{13}, t_{33} et t_{23} nécessaires au déclenchement éventuel de l'alarme.

Annexes

Annexe 1 - Mesures des temps du Système physique

3.1 Mesure du temps de déplacement et de saisie/dépôt d'une pièce

Dans le code suivant, vous trouverez le code des blocs FMG utilisé pour mesurer le temps de déplacement d'une case et de la prise du pièce. Nous avons pour cela implémenté la construction d'un Réseau de PETRI à 3 places : p_0 attend un appui sur "Opérateur", p_1 avance d'une case et p_2 prend une pièce.

Mesure de x, le temps pour avancer d'un emplacement et de y, le temps pour poser ou prendre une pièce.

```
while (1)
    {
      /**********************/
      /* Lecture des entrees */
      /**********************/

      appG = entree(APPG);
      ctr = entree(CTR);
      appD = entree(APPD);
      presence = entree(PRESENCE);
      lim_hor = entree(LIM_HOR);
      lim_ver = entree(LIM_VER);
      operateur = entree(OPERATEUR);


// ------------------------------------------------------------------------
  fintempo = difftime(time(NULL),tempo1);
// ------------------------------------------------------------------------

     /* allongement du cycle programme */
     // usleep(50);

// ------------------------------------------------------------------------

    /* blocs F */   // Description des transitions possibles

  if(p[0]==1){
    printf("p0 \n");
    if(operateur==1){  // marquage initial, attente d'un appui sur operateur
      ps[0]--;
      ps[1]++;
      ti = time(NULL);  // Init dela mesure
    }
  }

  if(p[1]==1){
    if(FinAvancer==1){  // Fin de l'avance d'une case
        ps[1]--;
      ps[2]++;
      tf = time(NULL);
```

```
        // Affichage du temps mis pour parcourir une case
        printf("Avance : %f \n",difftime(tf,ti));
        // Remise a zero de la machine a etats definie dans la fonction Avancer
        Avancer(0,0);
        ti = time(NULL);
      }
    }
    if(p[2]==1){  // Fin de la prise d'une piece
      ps[2]--;
      ps[3]++;
      tf = time(NULL);
      printf("Prendre : %f\n",difftime(tf,ti));

    }
      for(i=0;i<NBPLACES;i++){  // et actualisation des etats presents
          p[i] = ps[i];
    }
/ ------------------------------------------------------------------------------

        /*************************/
        /* Ecriture des sorties  */
        /*************************/

// ------------------------------------------------------------------------------

     /* blocs G */  // gestion des sorties en fonction du marquage mis a jour

      if(p[1]==1){
      Avancer(1,1);
      }

      if(p[2]==1){
      Prendre();
    }

// ------------------------------------------------------------------------------

        sortie(V_ACC,0);
       // sortie(HAUT, haut); // actions haut/bas gerees dans les fonctions
       // sortie(BAS,bas);    // Prendre et Poser ci-dessous
        sortie(GAUCHE,gauche);
        sortie(DROITE,droite);
        sortie(ALARME,0);
```

Résultat obtenu Nous avons obtenu, à l'aide de ce code, l'affichage suivant :

- "Avance : 1.0"
- "Prendre : 3.0"

3.2 Mesure du temps dedéplacement de bout en bout

Mesure du temps de traversé de bout en bout.

```
        /* Lecture des entrees */
        /***********************/

        appG = entree(APPG);
        ctr = entree(CTR);
        appD = entree(APPD);
        presence = entree(PRESENCE);
```

```
      lim_hor = entree(LIM_HOR);
      lim_ver = entree(LIM_VER);
      operateur = entree(OPERATEUR);
// ------------------------------------------------------------------------
  fintempo = difftime(time(NULL),tempo1);
// ------------------------------------------------------------------------

     /* allongement du cycle programme */
     // usleep(50);
// ------------------------------------------------------------------------

    /* blocs F */   // Description des transitions possibles
  if(p[0]==1){
    printf("p0 \n");
    if(operateur==1){  // marquage initial, attente d'un appui sur operateur
      ps[0]--;
      ps[1]++;
      ti = time(NULL);  // Init dela mesure
    }
  }

  if(p[1]==1){
    printf("p1 \n");     // FinAvancer est a 1 lorsque le capteur ctr est a 1
    if(FinAvancer==1){  // et qu'on a parcouru toutes les encoches demandees
        ps[1]--;         // Exemple: 3 encoches parcourues si Avancer(2,1)
      ps[2]++;
      tf = time(NULL);
      // Affichage du temps mis pour parcourir une case
      printf("Avance : %f \n",difftime(tf,ti));
      Avancer(0,0); // Remise a zero de la machine a etats definie dans la
          fonction Avancer
    }
  }
    for(i=0;i<NBPLACES;i++){  // et actualisation des etats presents
        p[i] = ps[i];
  }
// ------------------------------------------------------------------------

      /**************************/
      /* Ecriture des sorties   */
      /**************************/

// ------------------------------------------------------------------------

   /* blocs G */  // gestion des sorties en fonction du marquage mis a jour

  if(p[1]==1){
        Avancer(8,1);
    }

// ------------------------------------------------------------------------

      sortie(V_ACC,0);
     // sortie(HAUT, haut); // actions haut/bas gerees dans les fonctions
     // sortie(BAS,bas);    // Prendre et Poser ci-dessous
      sortie(GAUCHE,gauche);
      sortie(DROITE,droite);
      sortie(ALARME,0);
  /**COMMENTAIRE eTUDIANT*/
  //printf ("haut: %d -- bas : %d -- gauche : %d -- droite : %d \n",haut,bas,
     gauche, droite);
```

```
}

        sortie(V_ACC, 0);
```

Résultat obtenu Pour compléter l'analyse des temps de parcours, nous avons étoffé nos résultats avec une dernière mesure : une mesure du point de départ du chariot jusqu'à la fin du rail. Cette dernière mesure nous a semblé utile car nous avons remarqué que l'espacement des bacs n'était pas tout à fait identique. Avec le code disponible ci-dessus, nous obtenons un temps de traversé de bout en bout de $t = 9$, soit une seconde de plus. Étant donné que nous ne pouvons pas utiliser de décimale dans les intervalles temporelles du RdP, nous ne pouvons pas exploiter cette légère différence avec le temps $t = 8$ attendu.

Annexe 2 - Analyse TINA

3.1 Analyse d'accessibilité du modèle 2 opérations

Analyse d'accessibilité du modèle 2 opérations

```
Tina version 2.8.4 -- 10/27/06 -- LAAS/CNRS

mode -W

INPUT NET ---------------------------------------------------------

parsed net {reseau-AnalyseIII-2}

25 places, 23 transitions

net {reseau-AnalyseIII-2}
tr t0 : Operateur p0 -> p1
tr t1 [1,1] p1 -> p2
tr t10 [1,1] p21 -> p22
tr t11 [3,3] p8 -> p23
tr t12 [0,0] p22 -> p5
tr t13 [22,22] p19 -> p17
tr t14 [0,0] p24 -> p13
tr t15 [0,0] p20 p6 -> p7
tr t16 [2,2] p7 -> p8
tr t17 : FinReculer [1,1] p23 -> p24
tr t18 [0,0] p14 p19 -> p15
tr t19 [2,2] p15 -> p16
tr t2 [3,3] p2 -> p3
tr t20 [3,3] p16 ->
tr t21 [3,3] p5 -> p6
tr t22 [3,3] p13 -> p14
tr t3 [2,2] p3 -> p4
tr t4 [3,3] p4 -> p20 p9
tr t5 [16,16] p20 -> p18
tr t6 [1,1] p9 -> p10
tr t7 [3,3] p10 -> p11
tr t8 [2,2] p11 -> p12
tr t9 [3,3] p12 -> p19 p21
pl p0 (1)
pl p1 : {Avancer(1,1)}
pl p10 : {Prendre();}
pl p11 : {Avancer(2,1);}
pl p12 : {Poser();}
pl p13 : {Prendre();}
pl p15 : {Avancer(2,1);}
pl p16 : {Poser();}
pl p17 : {Alarme();}
pl p18 : {Alarme();}
pl p2 : {Prendre();}
pl p21 : {Reculer(1,1);}
pl p23 : {Reculer(1,1);}
pl p3 : {Avancer(2,1);}
pl p4 : {Poser();}
pl p5 : {Prendre();}
pl p7 : {Avancer(2,1);}
pl p8 : {Poser();}
pl p9 : {Reculer(1,1)}

0.000s

REACHABILITY ANALYSIS ---------------------------------------------
```

```
bounded

22 classe(s), 21 transition(s)

CLASSES:

class 0
    marking
        p0
    domain
        0 <= t0

class 1
    marking
        p1
    domain
        1 <= t1 <= 1

class 2
    marking
        p2
    domain
        3 <= t2 <= 3

class 3
    marking
        p3
    domain
        2 <= t3 <= 2

class 4
    marking
        p4
    domain
        3 <= t4 <= 3

class 5
    marking
        p20 p9
    domain
        16 <= t5 <= 16
        1 <= t6 <= 1

class 6
    marking
        p10 p20
    domain
        15 <= t5 <= 15
        3 <= t7 <= 3

class 7
    marking
        p11 p20
    domain
        12 <= t5 <= 12
        2 <= t8 <= 2

class 8
    marking
        p12 p20
    domain
        10 <= t5 <= 10
        3 <= t9 <= 3

class 9
    marking
        p19 p20 p21
    domain
        1 <= t10 <= 1
        22 <= t13 <= 22
        7 <= t5 <= 7

class 10
    marking
        p19 p20 p22
```

```
        domain
            0 <= t12 <= 0
            21 <= t13 <= 21
            6 <= t5 <= 6

    class 11
        marking
            p19 p20 p5
        domain
            21 <= t13 <= 21
            3 <= t21 <= 3
            6 <= t5 <= 6

    class 12
        marking
            p19 p20 p6
        domain
            18 <= t13 <= 18
            0 <= t15 <= 0
            3 <= t5 <= 3

    class 13
        marking
            p19 p7
        domain
            18 <= t13 <= 18
            2 <= t16 <= 2

    class 14
        marking
            p19 p8
        domain
            3 <= t11 <= 3
            16 <= t13 <= 16

    class 15
        marking
            p19 p23
        domain
            13 <= t13 <= 13
            1 <= t17 <= 1

    class 16
        marking
            p19 p24
        domain
            12 <= t13 <= 12
            0 <= t14 <= 0

    class 17
        marking
            p13 p19
        domain
            12 <= t13 <= 12
            3 <= t22 <= 3

    class 18
        marking
            p14 p19
        domain
            9 <= t13 <= 9
            0 <= t18 <= 0

    class 19
        marking
            p15
        domain
            2 <= t19 <= 2

    class 20
        marking
            p16
        domain
            3 <= t20 <= 3

    class 21
```

```
    marking

    domain


215 REACHABILITY GRAPH:

    0 -> t0 in [0,w[/1
    1 -> t1 in [1,1]/2
    2 -> t2 in [3,3]/3
    3 -> t3 in [2,2]/4
    4 -> t4 in [3,3]/5
    5 -> t6 in [1,1]/6
    6 -> t7 in [3,3]/7
    7 -> t8 in [2,2]/8
225 8 -> t9 in [3,3]/9
    9 -> t10 in [1,1]/10
    10 -> t12 in [0,0]/11
    11 -> t21 in [3,3]/12
    12 -> t15 in [0,0]/13
    13 -> t16 in [2,2]/14
    14 -> t11 in [3,3]/15
    15 -> t17 in [1,1]/16
    16 -> t14 in [0,0]/17
    17 -> t22 in [3,3]/18
235 18 -> t18 in [0,0]/19
    19 -> t19 in [2,2]/20
    20 -> t20 in [3,3]/21
    21 ->

    0.000s

    LIVENESS ANALYSIS ----------------------------------------------------------

    not live
245
    1 dead classe(s), 1 live classe(s)
    2 dead transition(s), 0 live transition(s)

    dead classe(s): 21

    dead transition(s): t5 t13

    STRONG CONNECTED COMPONENTS:

255 21 : 0
    20 : 1
    19 : 2
    18 : 3
    17 : 4
    16 : 5
    15 : 6
    14 : 7
    13 : 8
    12 : 9
265 11 : 10
    10 : 11
    9 : 12
    8 : 13
    7 : 14
    6 : 15
    5 : 16
    4 : 17
    3 : 18
    2 : 19
275 1 : 20
    0 : 21

    SCC GRAPH:

    21 -> t0/20
    20 -> t1/19
    19 -> t2/18
    18 -> t3/17
    17 -> t4/16
285 16 -> t6/15
```

```
15 -> t7/14
14 -> t8/13
13 -> t9/12
12 -> t10/11
11 -> t12/10
10 -> t21/9
9 -> t15/8
8 -> t16/7
7 -> t11/6
6 -> t17/5
5 -> t14/4
4 -> t22/3
3 -> t18/2
2 -> t19/1
1 -> t20/0
0 ->

0.000s

ANALYSIS COMPLETED ------------------------------------------------------
```

3.2 Graphe des classes de Mise au point des Intervallles Temporelles

Opération O_1

Graphe des classes TINA pour mise au point de l'intervalle temporelle de l'opération 1

```
Tina version 2.8.4 -- 01/05/16 -- LAAS/CNRS

mode -W

REACHABILITY ANALYSIS ---------------------------------------------------

bounded

22 classe(s), 21 transition(s)

CLASSES:

class 0
    marking
        p0
    domain
        0 <= t0

class 1
    marking
        p1
    domain
        1 <= t1 <= 1

class 2
    marking
        p2
    domain
        3 <= t2 <= 3

class 3
    marking
        p3
    domain
        2 <= t3 <= 2

class 4
    marking
        p4
    domain
        3 <= t4 <= 3

class 5
    marking
        p20 p9
    domain
        16 <= t5 <= 16
        1 <= t6 <= 1
```

```
    class 6
        marking
52          p10 p20
        domain
            15 <= t5 <= 15
            3 <= t7 <= 3

    class 7
        marking
            p11 p20
        domain
            12 <= t5 <= 12
62          2 <= t8 <= 2

    class 8
        marking
            p12 p20
        domain
            10 <= t5 <= 10
            3 <= t9 <= 3

    class 9
72      marking
            p19 p20 p21
        domain
            1 <= t10 <= 1
            22 <= t13 <= 22
            7 <= t5 <= 7

    class 10
        marking
            p19 p20 p22
82      domain
            2 <= t12 <= 2
            21 <= t13 <= 21
            6 <= t5 <= 6

    class 11
        marking
            p19 p20 p5
        domain
            19 <= t13 <= 19
92          3 <= t21 <= 3
            4 <= t5 <= 4

    class 12
        marking
            p19 p20 p6
        domain
            16 <= t13 <= 16
            0 <= t15 <= 0
            1 <= t5 <= 1
102
    class 13
        marking
            p19 p7
        domain
            16 <= t13 <= 16
            2 <= t16 <= 2

    class 14
        marking
112         p19 p8
        domain
            3 <= t11 <= 3
            14 <= t13 <= 14

    class 15
        marking
            p19 p23
        domain
            11 <= t13 <= 11
122         1 <= t17 <= 1

    class 16
```

```
        marking
            p19 p24
        domain
            10 <= t13 <= 10
            0 <= t14 <= 0

    class 17
        marking
            p13 p19
        domain
            10 <= t13 <= 10
            3 <= t22 <= 3

    class 18
        marking
            p14 p19
        domain
            7 <= t13 <= 7
            0 <= t18 <= 0

    class 19
        marking
            p15
        domain
            2 <= t19 <= 2

    class 20
        marking
            p16
        domain
            3 <= t20 <= 3

    class 21
        marking

        domain


    REACHABILITY GRAPH:

    0 -> t0 in [0,w[/1
    1 -> t1 in [1,1]/2
    2 -> t2 in [3,3]/3
    3 -> t3 in [2,2]/4
    4 -> t4 in [3,3]/5
    5 -> t6 in [1,1]/6
    6 -> t7 in [3,3]/7
    7 -> t8 in [2,2]/8
    8 -> t9 in [3,3]/9
    9 -> t10 in [1,1]/10
    10 -> t12 in [2,2]/11
    11 -> t21 in [3,3]/12
    12 -> t15 in [0,0]/13
    13 -> t16 in [2,2]/14
    14 -> t11 in [3,3]/15
    15 -> t17 in [1,1]/16
    16 -> t14 in [0,0]/17
    17 -> t22 in [3,3]/18
    18 -> t18 in [0,0]/19
    19 -> t19 in [2,2]/20
    20 -> t20 in [3,3]/21
    21 ->

    0.000s
```

Opération O_2

Graphe des classes TINA pour mise au point de l'intervalle temporelle de l'opération 2

```
    Tina version 3.4.4 -- 01/05/16 -- LAAS/CNRS
2
    mode -W

    REACHABILITY ANALYSIS ----------------------------------------------

    bounded
```

```
22 classe(s), 21 transition(s)

CLASSES:
12
class 0
    marking
        p0
    domain
        0 <= t0

class 1
    marking
        p1
22  domain
        1 <= t1 <= 1

class 2
    marking
        p2
    domain
        3 <= t2 <= 3

class 3
32  marking
        p3
    domain
        2 <= t3 <= 2

class 4
    marking
        p4
    domain
        3 <= t4 <= 3
42
class 5
    marking
        p20 p9
    domain
        16 <= t5 <= 16
        1 <= t6 <= 1

class 6
    marking
52      p10 p20
    domain
        15 <= t5 <= 15
        3 <= t7 <= 3

class 7
    marking
        p11 p20
    domain
        12 <= t5 <= 12
62      2 <= t8 <= 2

class 8
    marking
        p12 p20
    domain
        10 <= t5 <= 10
        3 <= t9 <= 3

class 9
72  marking
        p19 p20 p21
    domain
        1 <= t10 <= 1
        22 <= t13 <= 22
        7 <= t5 <= 7

class 10
    marking
        p19 p20 p22
82  domain
        2 <= t12 <= 2
```

```
            21 <= t13 <= 21
            6 <= t5 <= 6

    class 11
        marking
            p19 p20 p5
        domain
            19 <= t13 <= 19
92          3 <= t21 <= 3
            4 <= t5 <= 4

    class 12
        marking
            p19 p20 p6
        domain
            16 <= t13 <= 16
            0 <= t15 <= 0
            1 <= t5 <= 1
102
    class 13
        marking
            p19 p7
        domain
            16 <= t13 <= 16
            2 <= t16 <= 2

    class 14
        marking
112         p19 p8
        domain
            3 <= t11 <= 3
            14 <= t13 <= 14

    class 15
        marking
            p19 p23
        domain
            11 <= t13 <= 11
122         1 <= t17 <= 1

    class 16
        marking
            p19 p24
        domain
            10 <= t13 <= 10
            5 <= t14 <= 5

    class 17
132     marking
            p13 p19
        domain
            5 <= t13 <= 5
            3 <= t22 <= 3

    class 18
        marking
            p14 p19
        domain
142         2 <= t13 <= 2
            0 <= t18 <= 0

    class 19
        marking
            p15
        domain
            2 <= t19 <= 2

    class 20
152     marking
            p16
        domain
            3 <= t20 <= 3

    class 21
        marking
```

domain

REACHABILITY GRAPH:

```
0 -> t0 in [0,w[/1
1 -> t1 in [1,1]/2
2 -> t2 in [3,3]/3
3 -> t3 in [2,2]/4
4 -> t4 in [3,3]/5
5 -> t6 in [1,1]/6
6 -> t7 in [3,3]/7
7 -> t8 in [2,2]/8
8 -> t9 in [3,3]/9
9 -> t10 in [1,1]/10
10 -> t12 in [2,2]/11
11 -> t21 in [3,3]/12
12 -> t15 in [0,0]/13
13 -> t16 in [2,2]/14
14 -> t11 in [3,3]/15
15 -> t17 in [1,1]/16
16 -> t14 in [5,5]/17
17 -> t22 in [3,3]/18
18 -> t18 in [0,0]/19
19 -> t19 in [2,2]/20
20 -> t20 in [3,3]/21
21 ->

0.000s
```

Annexe 3 - Analyse TINA

3.1 Analyse d'accessibilité du modèle 3 opérations

Analyse d'accessibilité du modèle 3 opérations

```
Tina version 2.8.4 -- 10/27/06 -- LAAS/CNRS
2
mode -W

INPUT NET ---------------------------------------------------------

parsed net {reseau_Analyse--III-3}

40 places, 35 transitions

net {reseau_Analyse--III-3}
12 tr t0 : {Operateur();} p0 -> p1
tr t1 : FinAvancer [1,1] p1 -> p2
tr t10 : FinReculer [1,1] p21 -> p22
tr t11 [3,3] p8 -> p23 p36
tr t12 [2,2] p22 -> p5
tr t13 [22,22] p19 -> p17
tr t14 [3,3] p24 -> p13
tr t15 [0,0] p20 p6 -> p7
tr t16 : FinAvancer [2,2] p7 -> p8
tr t17 : FinReculer [1,1] p23 -> p24
22 tr t18 [0,0] p14 p19 -> p15
tr t19 : FinAvancer [2,2] p15 -> p16
tr t2 [3,3] p2 -> p3
tr t20 [3,3] p16 -> p25 p27
tr t21 [3,3] p5 -> p6
tr t22 [3,3] p13 -> p14
tr t23 [28,28] p25 -> p26
tr t24 : Att3 [3,3] p27 -> p28
tr t25 : FinReculer [2,2] p28 -> p29
tr t26 [3,3] p29 p36 -> p30
32 tr t27 : FinAvancer [2,2] p30 -> p31
tr t28 [7,7] p31 -> p32
tr t29 [3,3] p32 -> p33
tr t3 : FinAvancer [2,2] p3 -> p4
tr t30 : FinReculer [1,1] p33 -> p34
tr t31 [3,3] p25 p34 -> p35
tr t33 [22,22] p36 -> p37
tr t35 : FinAvancer [2,2] p35 -> p40
tr t36 [3,3] p40 -> p41
tr t4 [3,3] p4 -> p20 p9
42 tr t5 [16,16] p20 -> p18
tr t6 : FinReculer [1,1] p9 -> p10
tr t7 [3,3] p10 -> p11
tr t8 : FinAvancer [2,2] p11 -> p12
tr t9 [3,3] p12 -> p19 p21
pl p0 (1)
pl p1 : {Avancer(1,1);}
pl p10 : {Prendre();}
pl p11 : {Avancer(2,1);}
pl p12 : {Poser();}
52 pl p13 : {Prendre();}
pl p15 : {Avancer(2,1);}
pl p16 : {Poser();}
pl p17 : {Alarme();}
pl p18 : {Alarme();}
pl p2 : {Prendre();}
```

```
pl p21 : {Reculer(1,1);}
pl p23 : {Reculer(1,1);}
pl p26 : {Alarme();}
pl p28 : {Reculer(2,1);}
pl p29 : {Prendre();}
pl p3 : {Avancer(2,1);}
pl p30 : {Avancer(2,1);}
pl p32 : {Poser();}
pl p33 : {Reculer(1,1);}
pl p34 : {Prendre();}
pl p35 : {Avancer(2,1);}
pl p37 : {Alarme();}
pl p4 : {Poser();}
pl p40 : {Poser();}
pl p5 : {Prendre();}
pl p7 : {Avancer(2,1);}
pl p8 : {Poser();}
pl p9 : {Reculer(1,1)}

0.000s

REACHABILITY ANALYSIS ----------------------------------------------

bounded

32 classe(s), 31 transition(s)

CLASSES:

class 0
    marking
        p0
    domain
        0 <= t0

class 1
    marking
        p1
    domain
        1 <= t1 <= 1

class 2
    marking
        p2
    domain
        3 <= t2 <= 3

class 3
    marking
        p3
    domain
        2 <= t3 <= 2

class 4
    marking
        p4
    domain
        3 <= t4 <= 3

class 5
    marking
        p20 p9
    domain
        16 <= t5 <= 16
        1 <= t6 <= 1

class 6
    marking
        p10 p20
    domain
        15 <= t5 <= 15
        3 <= t7 <= 3

class 7
    marking
        p11 p20
```

```
        domain
            12 <= t5 <= 12
            2 <= t8 <= 2

    class 8
        marking
            p12 p20
        domain
            10 <= t5 <= 10
            3 <= t9 <= 3

    class 9
        marking
            p19 p20 p21
        domain
            1 <= t10 <= 1
            22 <= t13 <= 22
            7 <= t5 <= 7

    class 10
        marking
            p19 p20 p22
        domain
            2 <= t12 <= 2
            21 <= t13 <= 21
            6 <= t5 <= 6

    class 11
        marking
            p19 p20 p5
        domain
            19 <= t13 <= 19
            3 <= t21 <= 3
            4 <= t5 <= 4

    class 12
        marking
            p19 p20 p6
        domain
            16 <= t13 <= 16
            0 <= t15 <= 0
            1 <= t5 <= 1

    class 13
        marking
            p19 p7
        domain
            16 <= t13 <= 16
            2 <= t16 <= 2

    class 14
        marking
            p19 p8
        domain
            3 <= t11 <= 3
            14 <= t13 <= 14

    class 15
        marking
            p19 p23 p36
        domain
            11 <= t13 <= 11
            1 <= t17 <= 1
            22 <= t33 <= 22

    class 16
        marking
            p19 p24 p36
        domain
            10 <= t13 <= 10
            3 <= t14 <= 3
            21 <= t33 <= 21

    class 17
        marking
            p13 p19 p36
```

```
        domain
            7 <= t13 <= 7
            3 <= t22 <= 3
            18 <= t33 <= 18

    class 18
        marking
            p14 p19 p36
        domain
            4 <= t13 <= 4
            0 <= t18 <= 0
            15 <= t33 <= 15

    class 19
        marking
            p15 p36
        domain
            2 <= t19 <= 2
            15 <= t33 <= 15

    class 20
        marking
            p16 p36
        domain
            3 <= t20 <= 3
            13 <= t33 <= 13

    class 21
        marking
            p25 p27 p36
        domain
            28 <= t23 <= 28
            3 <= t24 <= 3
            10 <= t33 <= 10

    class 22
        marking
            p25 p28 p36
        domain
            25 <= t23 <= 25
            2 <= t25 <= 2
            7 <= t33 <= 7

    class 23
        marking
            p25 p29 p36
        domain
            23 <= t23 <= 23
            3 <= t26 <= 3
            5 <= t33 <= 5

    class 24
        marking
            p25 p30
        domain
            20 <= t23 <= 20
            2 <= t27 <= 2

    class 25
        marking
            p25 p31
        domain
            18 <= t23 <= 18
            7 <= t28 <= 7

    class 26
        marking
            p25 p32
        domain
            11 <= t23 <= 11
            3 <= t29 <= 3

    class 27
        marking
            p25 p33
        domain
```

```
            8 <= t23 <= 8
            1 <= t30 <= 1

    class 28
        marking
            p25 p34
        domain
            7 <= t23 <= 7
            3 <= t31 <= 3

    class 29
        marking
            p35
        domain
            2 <= t35 <= 2

    class 30
        marking
            p40
        domain
            3 <= t36 <= 3

    class 31
        marking
            p41
        domain

    REACHABILITY GRAPH:

    0 -> t0 in [0,w[/1
    1 -> t1 in [1,1]/2
    2 -> t2 in [3,3]/3
    3 -> t3 in [2,2]/4
    4 -> t4 in [3,3]/5
    5 -> t6 in [1,1]/6
    6 -> t7 in [3,3]/7
    7 -> t8 in [2,2]/8
    8 -> t9 in [3,3]/9
    9 -> t10 in [1,1]/10
    10 -> t12 in [2,2]/11
    11 -> t21 in [3,3]/12
    12 -> t15 in [0,0]/13
    13 -> t16 in [2,2]/14
    14 -> t11 in [3,3]/15
    15 -> t17 in [1,1]/16
    16 -> t14 in [3,3]/17
    17 -> t22 in [3,3]/18
    18 -> t18 in [0,0]/19
    19 -> t19 in [2,2]/20
    20 -> t20 in [3,3]/21
    21 -> t24 in [3,3]/22
    22 -> t25 in [2,2]/23
    23 -> t26 in [3,3]/24
    24 -> t27 in [2,2]/25
    25 -> t28 in [7,7]/26
    26 -> t29 in [3,3]/27
    27 -> t30 in [1,1]/28
    28 -> t31 in [3,3]/29
    29 -> t35 in [2,2]/30
    30 -> t36 in [3,3]/31
    31 ->

    0.000s

    LIVENESS ANALYSIS ----------------------------------------------------

    not live

    1 dead classe(s), 1 live classe(s)
    4 dead transition(s), 0 live transition(s)

    dead classe(s): 31

    dead transition(s): t5 t33 t23 t13
```

```
STRONG CONNECTED COMPONENTS:

31 : 0
30 : 1
29 : 2
28 : 3
27 : 4
26 : 5
25 : 6
24 : 7
23 : 8
22 : 9
21 : 10
20 : 11
19 : 12
18 : 13
17 : 14
16 : 15
15 : 16
14 : 17
13 : 18
12 : 19
11 : 20
10 : 21
9 : 22
8 : 23
7 : 24
6 : 25
5 : 26
4 : 27
3 : 28
2 : 29
1 : 30
0 : 31

SCC GRAPH:

31 -> t0/30
30 -> t1/29
29 -> t2/28
28 -> t3/27
27 -> t4/26
26 -> t6/25
25 -> t7/24
24 -> t8/23
23 -> t9/22
22 -> t10/21
21 -> t12/20
20 -> t21/19
19 -> t15/18
18 -> t16/17
17 -> t11/16
16 -> t17/15
15 -> t14/14
14 -> t22/13
13 -> t18/12
12 -> t19/11
11 -> t20/10
10 -> t24/9
9 -> t25/8
8 -> t26/7
7 -> t27/6
6 -> t28/5
5 -> t29/4
4 -> t30/3
3 -> t31/2
2 -> t35/1
1 -> t36/0
0 ->

0.000s

ANALYSIS COMPLETED ----------------------------------------------------------
```