# Contribution à la reconfiguration des systèmes de production : ordonnancement par recherche d'atteignabilité

Pascale Marangé, Jean-François Pétin, Antoine Manceaux, David Gouyon

# Contribution à la reconfiguration des systèmes de production : ordonnancement par recherche d'atteignabilité

**Pascale Marangé[1] — Jean-François Pétin[1] — Antoine Manceaux[2] — David Gouyon[1]**

[1] *Centre de Recherche en Automatique de Nancy*
*UMR 7039 – Nancy-Université, CNRS*
*Faculté des Sciences et Techniques, BP 70239, Vandoeuvre-lès-Nancy,*
*{pascale.marange, jean-françois.petin, david.gouyon} @cran.uhp-nancy.fr*

[2] *Société TRANE*
*1 r. des Amériques – Z.I. de Golbey*
*88190 Golbey*

RÉSUMÉ. *Ce papier s'intéresse à la reconfiguration de la commande dans le cadre de systèmes manufacturiers. Suite à une demande de reconfiguration (variabilité des produits, aléas de production), un nouvel ordonnancement est défini pour un ensemble de machines et de produits à fabriquer, par vérification d'une propriété d'atteignabilité sur des modèles d'automates communicants. Cette approche est illustrée sur un cas d'étude fourni par la société TRANE.*

ABSTRACT. *This paper deals with control reconfiguration for manufacturing systems. Following a reconfiguration request (product variability, unexpected events in production), a new scheduling is defined for a set of machines and products to be manufactured, by verifying a reachability property on communicating automata model. This approach is illustrated on a case study provided by the company TRANE.*

MOTS-CLÉS : *Reconfiguration, ordonnancement, recherche d'atteignabilité, model-checking, automates communicants*

KEYWORDS: *Reconfiguration, scheduling, reachability analysis, model-checking, communicating automata*

## 1. Introduction

L'ordonnancement de la production consiste à définir le cheminement d'un produit à l'intérieur d'un parc machine en allouant des ressources pour réaliser les transformations à effectuer, et en définissant les dates de début et de fin de chacune des opérations. Cet ordonnancement est généralement réalisé de manière prévisionnelle en considérant un environnement stable. Cependant, pour prendre en compte la forte variabilité des produits et des aléas de fabrications (pannes machines, indisponibilité des opérateurs, …), et ainsi augmenter l'adaptabilité des systèmes de production, il est nécessaire de compléter cet ordonnancement prévisionnel par un ou plusieurs ré-ordonnancement réactif. Ce ré-ordonnancement exploite les degrés de flexibilité du système de production reposant sur la redondance fonctionnelle des ressources.

Dans ce contexte, la gamme d'un produit peut se limiter à une gamme logique représentant les séquences de transformation du produit, sous forme d'opérations à réaliser sans définition préalable d'une gamme opératoire associant les transformations aux machines. Il est alors possible de générer un ensemble de trajectoires admissibles par la gamme logique, appelé « Virtual Manufacturing Line » par (Qui, 2004). Cette génération est souvent réalisée avant implémentation en déterminant au préalable l'ensemble de toutes les trajectoires possibles grâce à différentes techniques telles que la synthèse de superviseurs (Qui, 2004 ; Gouyon *et al.*, 2007), l'utilisation d'heuristiques (Henry *et al.*, 2004), ou encore l'analyse par Réseaux de Petri (Dangoumau, 2000).

Même si ces approches ont montré leur efficacité pour la reconfiguration des séquences logiques de production, elles ne prennent que difficilement en compte leur dimension temporelle (disponibilité des machines, mode de fonctionnement des machines, aléas de production, …). L'objectif est donc, sur la base de cette gamme logique, de définir un ordonnancement réactif tenant compte de l'état du système (aléa, variabilité du produit) qui contribue à la mise en œuvre de systèmes manufacturiers reconfigurables (Mehrabi and Ulsoy, 1997, De Lamotte, 2006). Les travaux de (Behrmann *et al.,* 2005 ; Panek *et al.*, 2006 ; Subbiah and Engell, 2010) ont montré l'efficacité des automates temporisés et des méthodes de recherche d'atteignabilité pour modéliser et résoudre des problèmes d'ordonnancement sur la base des gammes opératoires. En effet, le formalisme des automates temporisés supporte une modélisation modulaire, paramétrable et instanciable, lui conférant une plus grande robustesse vis à vis des changements de configuration du système que des approches plus classiques (programmation linéaire, recuit, heuristiques, …). Ces caractéristiques correspondent aux propriétés recherchées pour une reconfiguration dynamique en réponse à des aléas, pour laquelle on ne recherchera pas systématiquement une solution optimale mais plutôt une solution rapide et admissible par rapport aux spécifications du produit.

Ce papier propose une approche d'ordonnancement basée sur une technique de recherche d'atteignabilité pour une sous-classe des automates temporisés, les automates communicants, à partir des gammes logiques et non des gammes

opératoires (Subbiah and Engell, 2010). L'allocation des opérations à réaliser sur les machines du système de production s'appuie sur une synchronisation par appel/réponse, telle que proposée par (Lemattre *et al.*, 2011), des modèles de gammes logiques et de machines.
Le papier est organisé de la manière suivante. La définition formelle du problème est présentée dans la section 2. La troisième section propose une approche par recherche d'atteignabilité pour générer un ordonnancement admissible vis à vis de la gamme logique du produit et compatible avec l'état et la disponibilité des machines. La quatrième section instancie les modèles génériques proposés en section 3 sur un cas d'étude proposé par la société TRANE, et présente les résultats obtenus, notamment en termes de performances de calcul. La cinquième section conclut ce papier et donne des perspectives de travail.

## 2. Définition formelle du problème

A partir de la définition formelle proposée dans les travaux de Kanso (Kanso, 2010), nous considérons un système de production *S*= (O, *M*, *G*) défini par les opérations qu'il réalise, les machines qui le constituent, et les produits à fabriquer où :

- *O* est l'ensemble des opérations applicables aux différents produits,
- *M* est l'ensemble des machines $M_i$ = ($Odispo_i$, $E_i$) avec i $\in (0, \dots, I)$ où *I* est le nombre de machines dans le système et :
  - $Odispo_i$ est l'ensemble des couples constitués par les opérations que la machine $M_i$ est capable de réaliser et les temps d'exécution associés : $Odispo_{i_j} = (O_j, \mathrm{t}) \in O \times \mathbb{N}$ avec j $\in (0, \dots, J)$, *J* étant le nombre d'opérations réalisées par la machine $M_i$
  - $E_i$ représente la situation de la machine : {*disponible*, en traitement d'une demande, *occupée*, *en panne*}
- *G* est l'ensemble des gammes logiques associées aux produits à réaliser $G_k$ = ($P_k$, $Odem_k$, $Seq_k$) avec k $\in (0, \dots, K)$ où *K* est le nombre de produits dans le système et :
  - $P_k \in \mathbb{N}$ représente l'identifiant des produits,
  - $Odem_k$ représente l'ensemble des opérations à réaliser pour obtenir le produit fini à partir du produit initial : $Odem_k \subset O$,
  - $Seq_k$ est une relation entre les opérations :

$$Seq_k : Odem_k \times : Odem_k \to \mathcal{B}$$

$$(Odem_{k_m}, Odem_{k_n}) \mapsto \begin{cases} 1 \text{ si } Odem_{k_n} \text{ peut être} \\ \quad \text{exécutée après } Odem_{k_m} \\ 0 \text{ sinon} \end{cases}$$ [1]

Un ordonnancement consiste à associer des opérations d'une gamme logique à des machines et à définir la date de début de chaque opération. L'ordonnancement $Ordo_k$ pour un produit $P_k$ se définit donc comme un ensemble de triplets

[1] Cette définition permet d'exprimer des séquences linéaires, parallèles, cycliques, acycliques.

$(Odem_{k_q}, M_i, d)$ représentant la date d'exécution $d$ de l'opération $Odem_{k_q}$ sur la machine $M_i$. Le choix de la machine $M_i$ est contraint par la relation suivante : $\forall i, \exists j \; tel \; que \; Odispo_{i_j} = Odem_{k_q}$, c'est-à-dire qu'il faut trouver une machine $M_i$ supportant la réalisation d'une opération $Odispo_{i_j}$ correspondant à l'opération demandée par le produit $Odem_{k_q}$. L'obtention d'un ordonnancement se résume donc en deux activités :

- trouver une machine permettant de réaliser les opérations demandées dans l'ensemble des trajectoires logiques,
- définir les dates de disponibilité de ces machines.

## 3. Proposition d'une approche par recherche d'atteignabilité

### 3.1. *Principe général de l'approche*

Dans les travaux de (Subbiah and Engell, 2010), les auteurs proposent de modéliser le système de production sous la forme d'automates temporisés en considérant que les machines sont définies par deux états {occupée, libre} et que les produits sont définis par leurs gammes opératoires, c'est à dire les séquences ordonnées d'opérations à réaliser associées aux machines sur lesquelles ces opérations doivent être réalisées. Dans la gamme opératoire, un état marqué correspondant à un état d'achèvement est défini et doit être atteignable pour que le produit soit complétement fabriqué. L'obtention d'un ordonnancement consiste alors à vérifier que les états marqués de la composition entre les différents modèles de machines et celui du produit considéré sont atteignables. En cas de succès, un model-checker (Bérard *et al.,* 2001) fournit la trace permettant d'atteindre ces états, trace qui correspond en fait à l'ordonnancement recherché.

Dans la mesure où notre objectif est de faire de la reconfiguration dynamique, la machine exécutant une opération sur le produit n'est, par définition, pas connue à l'avance et dépendra de l'état du système. Pour cette raison, contrairement à l'approche de (Subbiah and Engell, 2010), nous avons choisi de définir le modèle de produit par la gamme logique (limitée à des séquences d'opérations non associées à une machine) plutôt que par la gamme opératoire. De plus pour éviter de faire la composition entre les modèles de machine et ceux de produit, ce qui conduirait à une explosion combinatoire pour des systèmes à taille industrielle tels que ceux de la TRANE qui ont motivé cette étude, nous utilisons le concept de synchronisation entre automates communicants sous la forme de mécanismes d'appel/réponse définis dans les travaux de (Lemattre *et al.*, 2011). En effet, cette technique ne définit aucune stratégie d'allocation des opérations sur les ressources et laisse le model-checker explorer l'espace d'état pour trouver une trace correspondant à une allocation admissible conduisant aux états marqués sur les gammes logiques. Cette approche s'apparente au concept de contrôle par le produit (Pétin *et al.*, 2007), où le produit est acteur dans le pilotage de sa production en sollicitant une ressource pour

exécuter une opération. Pour des raisons de simplicité liées au model-checker utilisé (UPPAAL (Behrmann *et al.*, 2002)), les états terminaux des gammes logiques seront en réalité représentés par des états bloquants et la propriété que nous chercherons à vérifier se traduit par : « il n'existe aucun état bloquant ». Si un ordonnancement est possible, cette propriété ne sera bien sûr pas vérifiée et le contre-exemple fourni par le model-checker correspondra à l'ordonnancement recherché.

### 3.2. *Formalisme*

Pour effectuer la modélisation des gammes logiques et des machines, nous allons utiliser des automates communicants, sous classe des automates temporisés (Alur and Dill, 1994). Un automate communicant est défini par le 6-tuplet A = (Q, E, f, $q_0$, I, h) suivant :

- *Q* est un ensemble fini d'états
- *E* est un ensemble fini de transitions
- *f* est une fonction de transition f : $Q \times E \rightarrow Q$ qui associe à une transition :
  - une garde qui est une contrainte à respecter pour passer la transition
  - une synchronisation permettant de faire évoluer deux automates A1 et A2 simultanément grâce à l'émission d'un message lors du franchissement d'une transition de A1, notée message *!*, qui provoque le franchissement d'une transition de A2 étiquetée par la réception de ce même message notée message *?*. Ce couple émission/réception associé à un message définit un canal de communication entre deux automates.
  - des mises-à-jours qui offrent la possibilité d'affecter une valeur à des variables
- q0 est l'état initial
- I est un ensemble d'invariants associés à chaque état
- *h* est une horloge utilisée pour spécifier des contraintes temporelles associées aux transitions ou des invariants associés aux états

Par la suite, les horloges ne seront pas utilisées dans nos modèles.

### 3.3. *Modèle du produit*

Le modèle du produit représente la gamme logique, c'est-à-dire la succession d'opérations à réaliser pour fabriquer un produit donné. Dans ce papier, nous avons considéré que les opérations se faisaient les unes après les autres. La Figure 1 représente le motif (4 premiers états situés au-dessus du trait horizontal) utilisé pour représenter les opérations de la gamme logique, le protocole de négociation avec une machine (émission d'une requête, attente d'une réponse) et l'attente d'un compte-rendu d'exécution. Ce motif est appliqué autant de fois qu'il y a d'opérations à

réaliser, l'état « opération terminée » d'un motif étant l'état initial du motif suivant. Le modèle complet du produit se termine par un état « produit terminé » représentant le fait que la gamme logique de fabrication est terminée, c'est-à-dire que toutes les opérations ont toutes été exécutées.

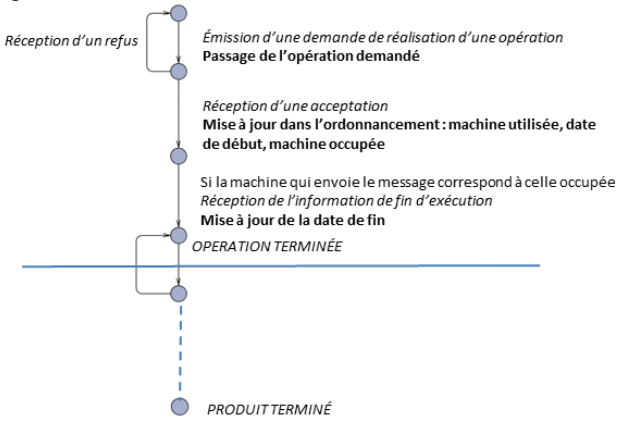


**Figure 1** *Motif du modèle de produit*

### 3.4. *Modèle des machines*

Le modèle de machine fonctionne en parallèle du modèle de produit et interagit avec celui-ci (Figure 2) par le biais de synchronisations. Les machines sont caractérisées par les opérations qu'elles peuvent réaliser, le temps d'exécution de chaque opération, et leur situation de fonctionnement (*disponible*, *en traitement d'une demande*, *occupée*, *en panne*). Les situations de fonctionnement, *disponible*, *en traitement d'une demande* et *occupée* sont prises en compte dans le modèle des machines par des états de l'automate et la situation *en panne* est prise en compte par une variable booléenne. Lors de la réception d'une demande d'exécution d'opération émise par un produit, le modèle de machine teste si la machine est disponible et si elle dispose des capacités nécessaires pour exécuter l'opération demandée.

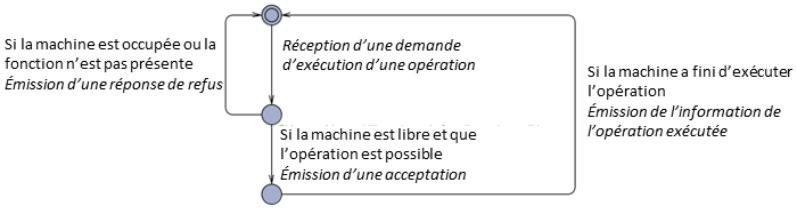


**Figure 2** *Modèle de machine*

### 3.5. *Evolution des modèles*

Les modèles de produit et de machine évoluent en parallèle de la manière suivante :

- Le modèle de produit émet une demande d'exécution d'une opération $Odem_{k_q}$ à une machine $M_i$
- Deux contraintes sont alors testées dans le modèle de produit : la disponibilité de la machine et sa capacité à exécuter l'opération demandée. Ces deux contraintes sont définies dans une garde sous la forme : $Odem_{k_q} == Odispo_{i_j}.E_i == disponible$. Si cette garde est vérifiée, l'opération $Odem_{k_q}$peut être exécutée sur la machine $M_j$, et que le produit est accepté par la machine :
  - Dans le cas d'une acceptation, la machine $M_j$ devient occupée, émet un message d'acceptation au produit $P_k$ et passe dans un état représentant l'exécution de l'opération $Odem_{k_q}$. Le modèle de produit attend alors la réception du message de fin d'exécution émis par la machine $M_i$ et la date de fin est mise à jour en tenant compte du temps d'exécution de la machine,
  - Dans le cas contraire, la machine $M_j$ émet un message de refus au produit $P_k$ et celui-ci fait une nouvelle demande

Il faut noter que la requête n'est pas adressée explicitement à une machine particulière puisque c'est la trace d'exécution du model-checker qui fournira le résultat d'allocation d'une opération à une machine. En revanche, le compte-rendu d'exécution est, quant à lui, associé à la machine ayant acceptée la requête. La section 4 détaillera les messages et gardes associées dans le formalisme supporté par le model-checker UPPAAL.

### 3.6. *Instanciation et implémentation des modèles*

Le système se modélise par un ensemble d'automates communicants qui comporte *n* produits et *m* machines, c'est-à-dire :

- *n* instances du modèle de produit (Figure 1)
- *m* instances du modèle de machine (Figure 2)

L'évolution des modèles se fait par synchronisation d'envoi et de réception de messages (demande d'exécution, acceptation, refus, fin d'exécution). Pour synchroniser les évolutions des instances $n_i$ et $m_j$ au cours d'un cycle de négociation (demande d'exécution d'opération par un produit et attente d'une réponse d'acceptation ou de refus par une machine) et ainsi interdire l'évolution des autres modèles de produit tant que le produit n'a pas reçu sa réponse nous avons utilisé des sémaphores représentés par des expressions logiques dans les gardes. L'instanciation des modèles et les mécanismes de synchronisation sont détaillés sur un cas d'étude dans les sections 4.1 et 4.2.

L'implémentation des modèles de produit et de machine se fait par la traduction des gardes en équations logiques, des synchronisations et des mises-à-jour des figures 1 et 2.

### 3.7. *Vérification par model-checking*

Afin de générer un nouvel ordonnancement, nous utilisons un model-checker qui explore l'espace d'états pour déterminer s'il existe un chemin permettant de fabriquer tous les produits. Pour cela, nous définissons une propriété stipulant qu'il n'existe aucun état de blocage. En langage CTL, cette propriété s'écrit :

*A[]not deadlock*

Ce blocage signifie que les modèles de produit ne peuvent plus évoluer, c'est à dire qu'ils ont tous atteint l'état « produit terminé » et donc qu'il existe un ordonnancement possible. Dans ce cas, cette propriété ne pourra pas être vérifiée. Le contre-exemple fourni par le model-checker, sous la forme d'une trace d'exécution conduisant à l'état de blocage, correspondra à une solution possible d'ordonnancement.

### 3.8. *Utilisations pour définir un ordonnancement*

L'utilisation de cette approche pour définir un ordonnancement peut se faire :

- dans le cadre d'une reconfiguration suite à un aléa ; dans ce cas, ce n'est pas l'optimalité de la solution qui est intéressante mais l'obtention d'une solution rapidement. La partie 4.3 présente les temps de calcul de notre approche en fonction du nombre de machines et de produits.
- par anticipation ; dans ce cas, l'obtention de la solution optimale peut se faire par recherche du plus court chemin qui satisfasse la propriété. Malheureusement, cette technique se heurte aux mêmes difficultés que celles rencontrées dans les méthodes classiques d'ordonnancement (temps de calculs, explosion combinatoire). Des solutions basées sur la réduction de l'espace d'état ou des heuristiques existent pour réduire le problème d'explosion combinatoire du model-checking et pourraient être utilisées. Une alternative consiste à ne pas proposer qu'une seule solution mais plusieurs parmi lesquelles un gestionnaire de production pourrait sélectionner celles qui répondent le mieux à ses critères d'ordonnancement. Cet ensemble de solutions peut être facilement obtenu par itération, en intégrant à la propriété vérifiée une composante interdisant certaines des allocations (opération / machine) déjà obtenues dans une solution précédente.

## 4. Cas d'étude

Cette partie présente l'instanciation des modèles de produits et de machines sur un cas d'étude extrait d'un système de production de l'entreprise TRANE. Ce système, inclus dans un processus de fabrication d'unité de climatisation, présente trois principales phases : découpe, pliage (au cours duquel 3 opérations doivent être réalisées) et peinture. La reconfiguration dynamique intervient dans deux cas :

- pour des raisons d'optimisation du rendement matière, les ordres de fabrication sont découpés en lots variables dont l'ordonnancement doit être redéfini de manière dynamique pour lancer la phase de pliage sur quatre presses plieuses (machines 0 à 3 du tableau 1) pouvant chacune exécuter 3 opérations distinctes parmi les 5 possibles sur la phase de pliage (tableau 1)
- en cas de défaillance sur une des 4 presses plieuses.

Nous avons considéré, pour la phase de pliage, des produits définis par une gamme linéaire composée de 3 opérations. Dix types de produits différents, dont les gammes sont données dans le tableau 2, ont été considérés dans notre cas d'étude.

| Machines | Marche (1 : marche, 0 : panne) | Occupée (1 : occupee, 0 : disponible) | Fonction disponible | | | Temps de fabrication | | |
|---|---|---|---|---|---|---|---|---|
| Machine0 | 1 | 0 | 1 | 2 | 3 | 2 | 1 | 3 |
| Machine1 | 0 | 0 | 2 | 3 | 5 | 5 | 2 | 1 |
| Machine2 | 1 | 0 | 4 | 2 | 5 | 2 | 2 | 4 |
| Machine3 | 1 | 0 | 4 | 1 | 3 | 1 | 1 | 2 |

**Tableau 1.** Machines, capabilité (opérations réalisables) et temps d'exécution

| Produits | Légende | Gamme à réaliser | | | Produits | Légende | Gamme à réaliser | | |
|---|---|---|---|---|---|---|---|---|---|
| Produit0 | | 1 | 2 | 3 | Produit5 | | 4 | 3 | 5 |
| Produit1 | | 2 | 3 | 5 | Produit6 | | 2 | 1 | 5 |
| Produit2 | | 4 | 2 | 5 | Produit7 | | 3 | 1 | 3 |
| Produit3 | | 4 | 1 | 3 | Produit8 | | 5 | 2 | 1 |
| Produit4 | | 1 | 5 | 3 | Produit9 | | 2 | 4 | 5 |

**Tableau 2.** Gammes logiques des 10 produits retenus pour le cas d'étude.

### 4.1. *Modèle du produit*

A partir des données du cas d'étude proposé par la société TRANE, le modèle de la Figure 3 utilise le motif de la Figure 1 pour représenter la gamme logique pour fabriquer les différents produits.

Les modèles des différents produits sont identiques au niveau de leur structure (c'est-à-dire trois opérations à exécuter l'une après l'autre pour chacun des produits) seuls le type de opération et l'ordre d'exécution changent.

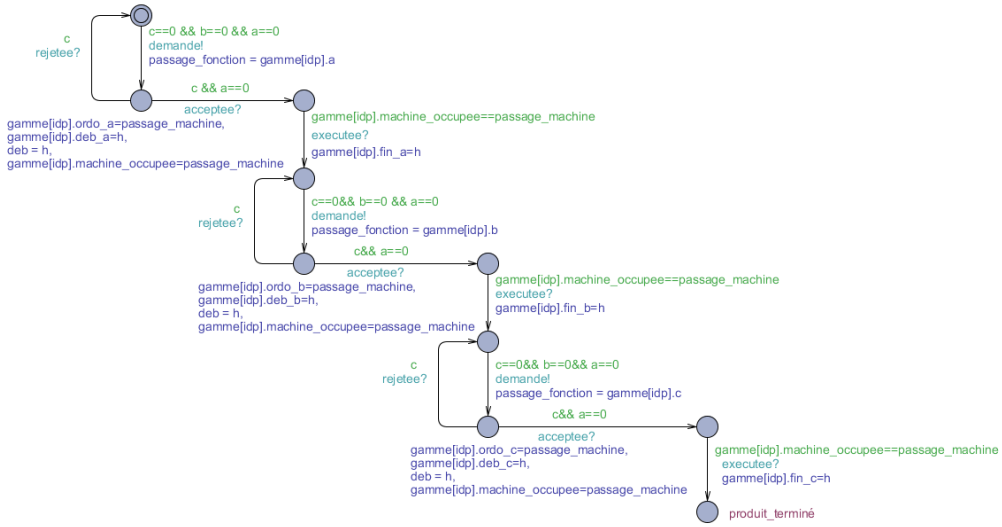


**Figure 3** *Instanciation du modèle du produit*

Le modèle du produit est instancié par l'indice *idp* ∈ *Nb_produit*. L'instanciation se fait en définissant la structure suivante : *struct { int a, b, c, deb_a, deb_b, deb_c, fin_a, fin_b, fin_c, ordo_a, ordo_b, ordo_c, machine_occupee; } gamme[Nb_produit]* où :

- *a*, *b*, *c* représentent les variables désignant les opérations à réaliser
- *deb_a*, *deb_b*, *deb_c* représentent la date de début de l'exécution de l'opération
- *fin_a*, *fin_b*, *fin_c* représentent la date de fin de l'exécution de l'opération
- *ordo_a*, *ordo_b*, *ordo_c* représentent la machine où a été exécutée l'opération

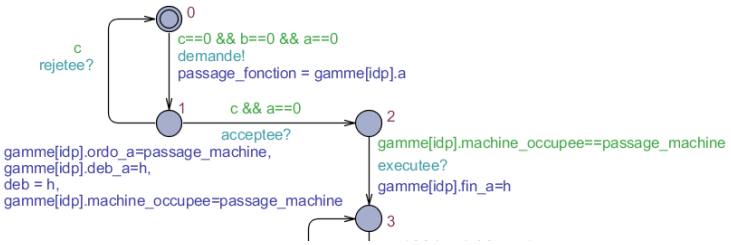


**Figure 4** *Motif instancié du modèle du produit*

Le modèle du produit est composé d'une succession de trois motifs (pour les trois opérations) à quatre états (Figure 4) répété trois fois pour les trois opérations

composant la gamme de production :

- un état {0, 3, 6} où le produit peut demander à une machine l'exécution d'une opération,
- un état {1, 4, 7} où le produit attend la réponse de la machine. Soit cette demande est refusée et le produit revient dans l'état de demande pour demander à une autre machine,
- un état {2, 5, 8} où le produit est sur la machine pour l'exécution de l'opération, lorsque la demande a été acceptée,
- un état {3, 6, produit_termine} où l'opération a été exécutée.

L'évolution d'un état à l'autre se fait par les transitions suivantes :

- de l'état 0 à l'état 1 : le modèle fait une demande d'exécution à une machine via le canal de communication par le message *demande !* et mémorise l'opération demandée *gamme[idp].a*
- de l'état 1 à l'état 0 : le modèle attend la réception d'un message *rejetee ?,* si l'opération *gamme [idp].a* ne peut pas être exécutée
- de l'état 1 à l'état 2 : le modèle attend la réception d'un message *acceptee ?* si l'opération *gamme[idp].a* peut être réalisée par la machine, et lors du franchissement de la transition, la machine exécutant l'opération *gamme[idp].ordo_a*, la date de début d'exécution de l'opération *gamme[idp].deb_a*, et la machine actuellement occupée par le produit *gamme [idp].machine occupee* sont mémorisées.
- de l'état 2 à l'état 3 : le modèle attend la réception du message de fin d'exécution de l'opération *executee ?*, à condition que la machine émettant le message soit bien celle sur laquelle le produit est, ceci étant testé par la garde *gamme [idp].machine_occupee==passage_machine*.

A ce modèle, ainsi qu'au modèle de la machine, les variables *a*, *c*, *b*, sont ajoutées pour donner des priorités dans l'ordre d'exécution des transitions.

### 4.2. *Modèle de la machine*

Le modèle proposé en Figure 2 est instancié par rapport au cas d'étude de la TRANE pour obtenir le modèle de la Figure 5. Les machines, et donc les modèles les representant, sont distinctes les unes des autres par les opérations qu'elles peuvent réaliser, et le temps d'exécution de celles-ci. Les différentes machines sont identifiées par l'indice *idm* ∈ *Nb_Machine*, et sont définies par la structure suivante : *struct { bool marche, occupee; int a, b, c, tps_fab_a, tps_fab_b, tps_fab_c, tps_fab_M, deb;} machine[Nb_machine]* où :

- *marche* représente l'état de fonctionnement (0 en panne, 1 sinon),
- *occupé* représente la présence ou non d'un produit sur la machine,
- *a, b, c* représentent les opérations pouvant être réalisées sur la machine,
- *tps_fab_a, tps_fab_b, tps_fab_c* représente les temps de fabrication des différentes opérations,
- *tps_fab_M* représente le temps de fabrication en cours d'exécution
- *deb* représente la date de début de l'exécution.

Nous pouvons remarquer qu'il y a 2 états supplémentaires par rapport à la Figure 2, (états 12 et 14). L'ajout de ces états et des tags {a, b, c} permettent de garantir la synchronisation d'évolution entre les « bons » modèles produit/machine et ainsi la « bonne » affectation des variable indicée par *idm*.

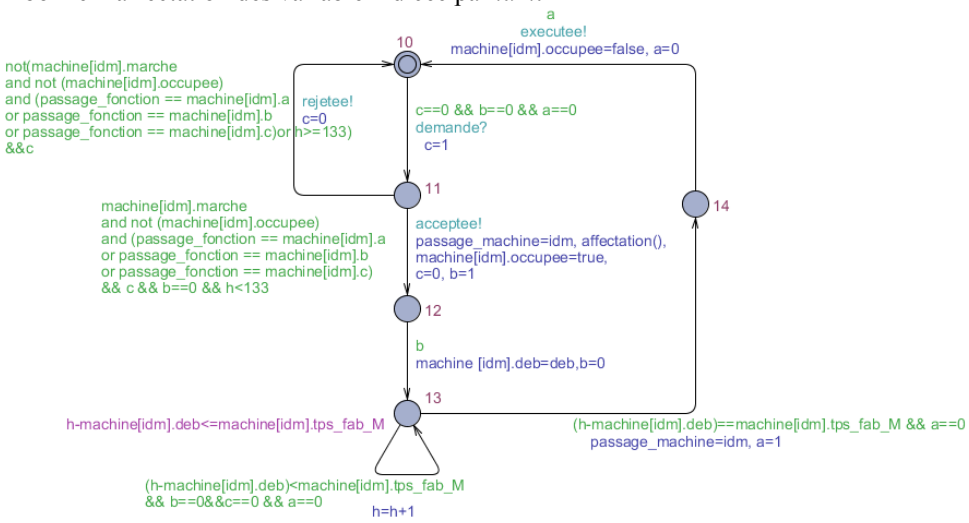


**Figure 5** *Instanciation du modèle de la machine*

Une machine peut être dans cinq états : l'état 10 représente l'état d'attente d'une requête d'un produit, l'état 11 est un état de réponse ; l'état 12 permet la mise à jour du début de fabrication, l'état 13 représente l'état d'exécution de l'opération, l'état 14 représente la fin d'exécution de l'opération.
L'état 13 peut être actif seulement le temps de l'exécution de l'opération. Pour cela, un invariant a été défini pour représenter le fait que le temps passé dans l'état (*h-machine[idm].deb*) doit être toujours inférieur ou égal à *machine[idm].tps_fab_M*). Le temps est modélisé par l'intermédiaire d'un compteur global à tous les modèles du système et est noté *h*.
Le passage d'un état à l'autre se fait de la manière suivante :

- de l'état 10 à l'état 11 : le modèle attend la requête d'un produit par la reception du message *demande ?,*
- de l'état 11 à l'état 10 : si l'opération demandée n'est pas présente sur la machine (*fonction_demandee==machine[idm].a or fonction_demandee==machine[idm].b or fonction_demandee==machine[idm].c*), si la machine est occupée (*machine[idm].occupee*), ou si la machine est en panne (*not(machine[idm].marche)*), alors le modèle de la machine émet le message *rejetee !* pour permettre l'évolution du modèle du produit,
- de l'état 11 à l'état 12 : si l'opération demandée est présente, la machine est disponible et n'est pas en panne, alors le modèle de la machine émet *acceptee !*, mémorise la machine pour le modèle produit (passage_machine

= idm), rend indisponible la machine (machine[idm].occupee=1) et met à jour le temps de fabrication de la machine en fonction de l'opération à exécuter (affectation()),

- de l'état 12 à l'état 13 : permet de mémoriser la date de début d'exécution dans la machine,
- de l'état 13 à l'état 13 : permet l'incrémentation du temps d'une manière globale à toutes les machines, tant que celui-ci est inférieur au temps de fabrication (*h-machine[idm].deb <=machine[idm].tps_fab_M).* L'état 13 est le seul état qui consomme du temps,
- de l'état 13 à l'état 14 : lorsque la garde (*h-machine[idm].deb ==machine[idm].tps_fab_M)* est vraie, cette transition est franchie et la machine qui va générer le message *executee ?* est mémorisée dans la transition suivante de l'état 14 à l'état 10.

### 4.3. *Résultats*

La figure 6 présente l'ordonnancement prévisionnel défini pour les 10 produits tableau 2. Nous avons considéré un ré-ordonnancement des opérations de la phase de pliage suite à la défaillance de la machine 1.

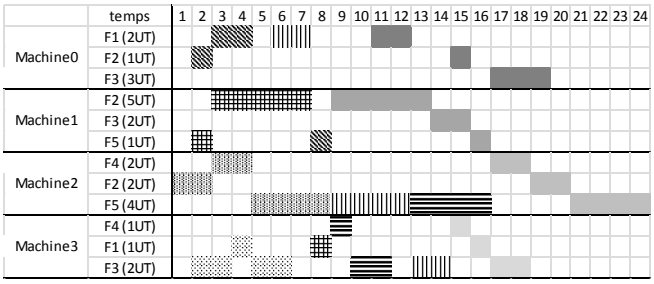


**Figure 6** *Ordonnancement prévisionnel avec toutes les machines disponibles*

Nous cherchons alors à obtenir un nouvel ordonnancement dans lequel la machine 1 est passée dans un état de panne (passage à 0 de la variable *machine[1].marche*). La propriété que l'on cherche à vérifier stipule qu'il n'existe aucun état de blocage correspondant à l'atteinte des états « produit terminé » pour chacune des 10 instances du modèle produit. L'échec de la vérification réalisée sur le model-checker UPPAAL génère un contre-exemple, à partir duquel un nouvel ordonnancement est définit avec une durée h = 34 unités de temps (Figure 7).

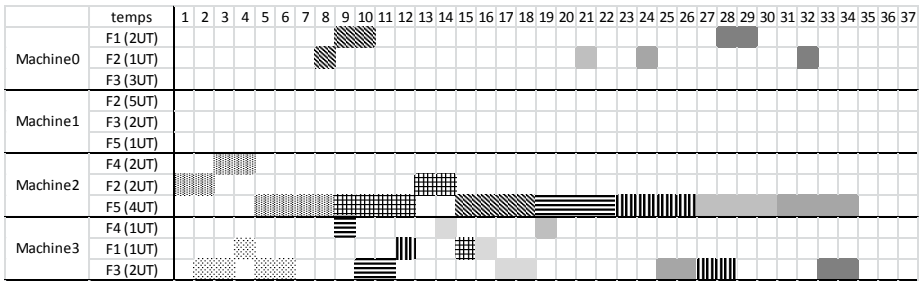


**Figure 7** *Nouvel ordonnancement obtenu par recherche d'atteignabilité*

Le model-checker UPPAAL retourne une reconfiguration possible, dont on ne peut garantir l'optimalité, même si, dans les cas étudiés, la solution obtenue s'est trouvée être comparable aux solutions obtenues avec les outils d'ordonnancement classiques utilisés à la TRANE. En revanche, l'intérêt réside dans le fait de pouvoir proposer très rapidement et en ligne une solution admissible, et ainsi minimiser le temps d'attente nécessaire à la détermination d'une nouvelle configuration.
Pour mieux conforter ce constat, les performances temporelles de notre approche ont été évaluées, à titre indicatif et sur un ordinateur personnel de performance moyenne, en faisant varier le nombre de produits et de machines. Le tableau 3 donne, à titre d'information, le temps de calcul en fonction du nombre de produits et de machines, l'exemple n'ayant pas été traité avec d'autres algorithmes d'ordonnancement.

| Nb de produits | Nb de machines | Temps de calcul (en s) | Nb de produits | Nb de machines | Temps de calcul (en s) |
|---|---|---|---|---|---|
| 30 | 5 | 0.78 | 90 | 10 | 52.36 |
| 60 | 5 | 4.43 | 180 | 10 | 500,47 |
| 90 | 5 | 14.85 | 210 | 10 | 928.15 |
| 120 | 5 | 37.09 | 30 | 20 | 5.9 |
| 150 | 5 | 82.70 | 60 | 20 | 41.92 |
| 180 | 5 | 153.53 | 90 | 20 | 136,58 |
| 210 | 5 | 292.11 | 180 | 20 | 1345.29 |
| 240 | 5 | 479.10 | 30 | 40 | 21.02 |
| 30 | 10 | 2.52 | 60 | 40 | 155.71 |
| 60 | 10 | 16.18 | 90 | 40 | 521,13 |

Tableau 3. Temps de calcul pour différentes configurations produit / machine.

## 5. Conclusion et perspectives

Dans ce papier, nous avons proposé une approche d'ordonnancement adaptée à la reconfiguration dynamique des systèmes de production. Cette approche repose sur une modélisation à base d'automates communicants du parc machine et des gammes

logiques des produits à fabriquer et utilise la recherche d'atteignabilité pour générer une trace d'exécution correspond à l'ordonnancement de la production recherché. Plusieurs perspectives à ces travaux sont envisageables.

La première concerne la prise en compte d'un modèle plus fin des états de fonctionnement des machines, notamment en intégrant explicitement dans le modèle machine :

- une contraintes sur les taux d'occupation qui devrait permettre de diminuer la taille de l'espace d'états exploré et d'introduire un critère supplémentaire d'ordonnancement,
- un état de panne qui devrait permettre de prendre en compte la notion de mode au sein du processus de reconfiguration. La prise en compte probabiliste des taux de défaillances des machines constituerait incontestablement un plus pour proposer des ordonnancements anticipant les défaillances probables dans un horizon temporel pour répondre à un niveau de confiance donné.

Le second axe de recherche à développer concerne la définition de points de reconfiguration plus réalistes. En effet, lors de la recherche d'une reconfiguration, nous avons considéré qu'aucune des gammes de produits n'était commencée et que les machines n'étaient pas déjà occupées par un produit en cours de fabrication. La prise en compte des gammes en cours d'exécution peut se faire facilement sur nos modèles en ajoutant un état d'initialisation conduisant directement à l'état en cours.

Enfin, pour une meilleure prise en compte de la variabilité des produits, il devrait être possible de définir un modèle générique de produit se limitant à émettre toutes les demandes d'opérations et en faisant supporter les contraintes de précédence d'une gamme logique (Marangé *et al.*, 2010) par les gardes du modèle de machines.

## 6. Bibliographie

Alur R., Dill D., « A Theory of Timed Automata », *Theoretical Computer Science*, vol. 126, n°2, p. 183–235, 1994.

Behrmann G., Brinksma E., Hendriks M., Mader A., «Production scheduling by reachability analysis - a case study », *Parallel and Distributed Processing Symposium*, International, 3 : pp140–147, 2005.

Behrmann G., Bengtsson J., David A., Larsen K., Pettersson P.,Yi W., «UPPAAL implementation secrets », *Proc. of 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems* (FTRTFT), Oldenburg, Germany, september 2002.

Bérard B., Bidoit M., Finkel A., Laroussinie F., Petit A., Petrucci L., Schnoebelen P., *Systems and Software Verification*, Springer, 2001.

Dangoumau N., Contribution à la Gestion des Modes des Systèmes Automatisés de Production, Thèse de doctorat, Université des Sciences et Technologies de Lille, 2000

De Lamotte F., Proposition d'une approche haut niveau pour la conception, l'analyse et l'implantation des systèmes reconfigurables, thèse de doctorat de l'Université Paul

Sabatier de Toulouse, 2006

Gouyon D., Pétin J.-F., Morel G., « A product-driven reconfigurable control for shop floor systems », *Studies in Informatics and Control* 16, 1, 2007

Henry S., Deschamps E., Zamai E., Jacomino, « Control law synthesis algorithm for discrete event systems », *IFAC conference on Mangement and Control of Production and Logistics*, 2004

Kanso M., Contribution à la construction des configurations des systèmes manufacturiers : une approche basée sur la décision multicritères, thèse de doctorat de l'université de Bretagne sud, 2010

Lemattre T., Denis B., Faure J-M., Pétin J-F., Salaün P., « Aide à la conception d'architecture opérationnelles de commandes de systèmes critiques par analyse d'atteignabilité », *4èmes Journées Doctorales / Journées Nationales MACS*, 9-10 juin 2011, Marseille (à paraître)

Marangé P., Gouyon D., Pétin J.-F., Gellot F., « Formalisation et vérification de contraintes fonctionnelles dans le cadre d'un contrôle par le produit », *6ème Conférence Internationale Francophone d'Automatique*, 2010, Nancy

Mehrabi, M. G., Ulsoy, A. G, « State-of-the-Art in Reconfigurable Manufacturing Systems », Report #2, Vol. I and Vol. II, Engineering Research Center for Reconfigurable Machining Systems (ERCIRMS), The University of Michigan, Ann Arbor, MI., 1997

Panek S., Engell S., Stursberg O., « Scheduling and Planning with Timed automata », *16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering*, published by Elsevier B.V., 2006

Pétin J-F., Gouyon, D. Morel G., « Supervisory synthesis for product-driven automation and its application to a flexible assembly cell », *Control Engineering Practice*, 15, pp. 595-614, 2007.

Qiu R.G., Joshi S., McDonnell P.. « An approach to regulating machine sharing in reconfigurable back-end semiconductor manufacturing », *Journal of Intelligent Manufacturing*, volume 15, pages 579-591, 2004

Subbiah S., Engell S., « Short-Term Scheduling of Multi-Product Batch Plants with Sequence-Dependent Changeovers Using Timed Automata Models », *20th European Symposium on Computer Aided Process Engineering*, volume 28, pp 1201–1206, 2010.