# Synthèse de contraintes de conception à partir de réseaux de Petri temporels paramétrés

**Louis-Marie Traonouez, David Delfieu et Olivier H. Roux**

*IRCCyN, 1 rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France*
*02-40-37-69-16*
*Louis-Marie.Traonouez@irccyn.ec-nantes.fr*

RÉSUMÉ. *Un système temps réel est décomposé en composants appelés* modules. *L'objectif est de garantir des contraintes temporelles projetées sur ces modules sous la forme de contraintes paramétrées. Pour cela un ensemble de* contraintes de conception *est calculé. Le modèle sur lequel est basé cette approche est celui des réseaux de Petri temporels paramétrés, traduits en formules de Logique Linéaire.*

ABSTRACT. *A real time system is decomposed in components called* modules. *The objective is to guarantee temporal constraints which are projected on these modules on a parametrized constraint form. To achieve this, a set of* conception constraints *is computed. The model on which is based this approach is a parametrized time Petri net, translated in Linear Logic formulae.*

MOTS-CLÉS : *Réseaux de Petri temporels, paramètres, logique linéaire, contraintes temporelles, conception des systèmes temps réel.*

KEYWORDS: *Time Petri nets, parameters, Linear Logic, timed constraints, real-time systems design.*

## 1. Introduction

Le cahier des charges d'une application informatique est en général formulé en langage naturel. Il est donc soumis à l'interprétation des rédacteurs de la spécification, ce qui est source d'erreurs. Par ailleurs, pour des programmes d'une taille raisonnable, il est impossible, pour le concepteur, d'appréhender toutes les interactions entre les différents composants. Les méthodes formelles visent à offrir un cadre mathématique permettant de décrire de manière précise et stricte les systèmes et programmes que nous voulons construire. Dans ce cadre mathématique, le système est décrit par un système de transitions étiquetées (ou par un modèle permettant d'abstraire un tel système comme les automates, les réseaux de Petri, les algèbres de processus...), et la spécification de la correction du système est décrite par des propriétés qui peuvent être exprimées sous la forme d'observateurs du modèle ou dans une logique particulière telle que les logiques temporelles.

Parmi les méthodes formelles, le model-checking est une procédure automatique qui permet de vérifier qu'un modèle d'un système $S$ satisfait une spécification décrite par une propriété $\varphi$.

*Conception d'un Système Temps Réel*

Dans le cycle de développement d'un système, les étapes de spécification et de conception architecturale amènent à réaliser une décomposition du système en sous-ensembles (modules ou objets) devant être réalisés séparément puis composés (assemblés). Cette démarche peut également faire appel à la notion d'objet hiérarchique.

Cette décomposition est généralement réalisée à partir des contraintes fonctionnelles du système. Elle permet de définir l'architecture logicielle du système qui, projetée sur l'architecture matérielle, fournira l'architecture opérationnelle. Cependant, certaines spécifications non fonctionnelles telles que les spécifications temporelles du système global ne peuvent pas être partitionnées en un ensemble de spécifications à appliquer à chaque module de cette décomposition.

La solution qui consiste alors à projeter les propriétés temporelles $\varphi$ du système sur les modules de l'architecture opérationnelle, pose le problème majeur de la compositionnalité, c'est à dire : le fait que chaque module vérifie la projection de $\varphi$ garantit-il que le système complet vérifiera $\varphi$ ? On peut traiter ce problème en effectuant une projection *pessimiste* et non optimale afin de garantir la compositionnalité mais au prix de contraintes trop strictes pouvant rendre difficile la réalisation (l'implémentation) des modules.

Nous proposons de faire une projection paramétrée des propriétés sur les modules. En plus de son architecture fonctionnelle, un module dispose alors des informations suivantes :

– les contraintes fonctionnelles ou non qui lui sont propres, donc sans paramètre ;

– les contraintes temporelles qu'il partage avec d'autres modules, donc paramétrées.

Notre méthode consiste à prendre l'ensemble de ces informations pour déterminer les contraintes paramétrées aux frontières de chaque module. Elles seront appelées *contraintes de conception*. Cette démarche présente les avantages suivants :

– Chaque module dispose de toutes les contraintes qu'il doit vérifier.

– À partir de l'ensemble des contraintes générées, on peut savoir en utilisant un solveur si une solution existe ou non et reprendre la conception en amont si cela est nécessaire.

– Dans la phase d'assemblage, les modules partageant le même paramètre sont d'abord regroupés afin de vérifier la (ou les) propriété(s) associée(s) à ce paramètre. Si la propriété n'est pas vérifiée, le problème est traité sur cet ensemble de modules et non sur le système complet.

*Exemple*

Illustrons cette démarche sur un exemple théorique très simple. Nous considérons un système $S$, dont la décomposition fonctionnelle est une séquence de deux modules $A$ et $B$ : $S = A; B$. Le module $B$ est lui-même décomposé en une séquence de deux modules : $B = B_1; B_2$. De plus, la durée d'exécution du module $B_1$ est imposée : $dB_1 = 2\ ms$.

Une contrainte temporelle est fixée sur ce système : la durée d'exécution totale du système doit être inférieure à $10\ ms$.

La méthode que nous proposons, permet à partir de la projection de cette contrainte sur les modules de déterminer la contrainte de conception : $dA + dB_2 \leq 8$, où $dA$ et $dB_2$ représentent respectivement les temps d'exécution des modules $A$ et $B_2$. Cette contrainte fournit un guide aux concepteurs du système pour développer les modules $A$ et $B_2$. En s'assurant qu'elle est vérifiée, ils garantissent que la propriété temporelle initiale (durée d'exécution inférieure à $10\ ms$) est vérifiée.

*Contribution et plan de l'article*

Le travail décrit dans cet article consiste, à partir du modèle formel d'un module et de ses contraintes paramétrées, à calculer un ensemble de contraintes appelées *contraintes de conception* du module. Le système complet peut alors être vérifié par compositionnalité : le système complet vérifiera ses contraintes temporelles si l'ensemble des contraintes de conception des modules est vérifié. La section 2 décrit la méthode et l'algorithme permettant de déterminer les contraintes de conception d'un module. Cette méthode est ensuite illustrée de manière détaillée dans la section 3 sur un exemple extrait (sans modification) d'une application industrielle qui nous a été fournie par un grand groupe Européen du temps réel embarqué (que nous n'avons pas eu l'autorisation de nommer). Enfin, nous terminons par une conclusion.

## 2. Méthode de synthèse des contraintes de conception

### 2.1. *Introduction*

La synthèse de *contraintes de conception* que nous proposons s'inscrit dans une méthode générale que nous décrivons succinctement ci-après.

#### 2.1.1. *Présentation de la méthode générale*

Le préambule à cette méthode est la projection des contraintes temporelles sur les modules de l'architecture opérationnelle. Les contraintes de conception sont alors déterminées sur chacun des modules, et ces derniers sont ensuite assemblés. Ce processus est itéré jusqu'à l'obtention des contraintes de conception du système complet. Reprenons l'exemple précédent pour illustrer ces deux phases.

**Projection des contraintes temporelles :** la première étape consiste à projeter la contrainte globale sur les modules. Formalisons ce problème : notons $\theta_d(.)$ (respectivement $\theta_f(.)$) les dates de début (respectivement de fin) d'exécution d'un module. La propriété à vérifier s'écrit : $\theta_f(S) - \theta_d(S) \leq 10$. La projection de cette propriété sur les modules $A$ et $B$ est $\theta_f(B) - \theta_d(A) \leq 10$, ce qui introduit deux paramètres communs $\theta_f(B)$ et $\theta_d(A)$. Cette contrainte projetée sur $B_1$ et $B_2$ devient $\theta_f(B_2) - \theta_d(A) \leq 10$.

**Assemblage des modules :** À partir de ces projections l'algorithme 1 présenté dans la suite de cet article, permet de remonter les contraintes aux frontières des modules. Sur le module $B_2$, sachant que $\theta_f(B_2) = \theta_d(B_2) + dB_2$, une nouvelle propriété à vérifier $\theta_d(B_2) + dB_2 - \theta_d(A) \leq 10$ est identifiée (les frontières du module $B_2$ sont les deux paramètres $\theta_d(B_2)$ et $dB_2$). Les modules sont alors assemblés en identifiant les paramètres communs : $\theta_d(B_2) = \theta_f(B_1)$.

Le processus est alors itéré sur $B_1$. On obtient : $\theta_d(B_1) + dB_2 - \theta_d(A) \leq 8$. Puis sur $A$, en faisant la liaison de $B_1$ vers $A$ : $\theta_d(B_1) = \theta_d(B) = \theta_f(A)$. On obtient finalement la contrainte de conception : $dA + dB_2 \leq 8$.

#### 2.1.2. *Principe de détermination des contraintes de conception*

Au sein de la méthode générale précédemment décrite, l'essentiel de notre contribution est la détermination des contraintes de conception d'un module ou d'un ensemble de modules. Détaillons maintenant le principe.

La décomposition du système a permis de définir des modules paramétrés, sur lesquels des contraintes fonctionnelles et temporelles sont fixées. À partir de ces spécifications, nous avons développé un algorithme qui permet de déterminer les contraintes de conception d'un module. Le principe général de cette méthode est décrit sur le schéma de la figure 1.

Nous considérons le modèle formel d'un module, défini par un réseaux de Petri temporel paramétré, sur lequel il nous faut vérifier les contraintes temporelles. Pour cela nous analysons le RdP à l'aide d'un processus de preuve en logique linéaire afin de déterminer un ensemble $(I)$ d'inéquations portant sur les dates de tir des transitions de ce réseau. Par ailleurs, nous traduisons les contraintes à vérifier en un autre ensemble $(\phi)$ d'inéquations. À l'aide de ces deux ensembles, l'algorithme 1 développé permet de déterminer un ensemble $(\psi)$ d'inéquations, portant sur les paramètres du module, qui est suffisant pour vérifier les contraintes temporelles.

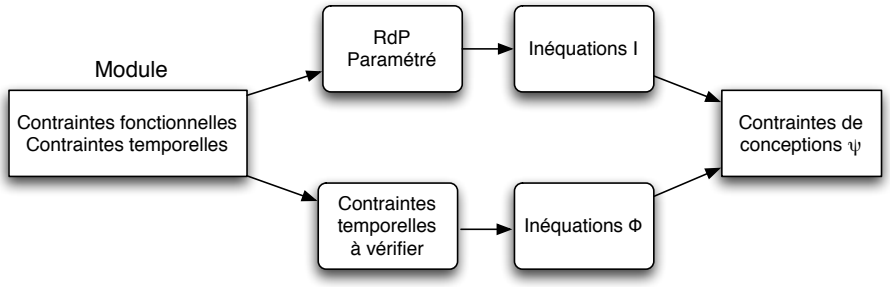


**Figure 1.** *Méthode de synthèse de contraintes de conception*

### 2.2. *Réseaux de Petri T-temporels paramétrés*

Nous définissons les RdP T-temporels paramétrés, en considérant un ensemble de *paramètres* $Q = \{q_1, \ldots, q_n\}$. Nous définissons une valuation de ces paramètres comme une fonction $\nu : Q \to \mathbb{R}^+$.

**Définition 1 (Réseau de Petri T-temporel paramétré)**
*Un réseau de Petri T-temporel, paramétré par un ensemble de paramètres $Q$, est un 7-uplet $(P,\ T,\ {}^\bullet(.),\ (.)^\bullet,\ \alpha,\ \beta,\ M_0)$ où :*

– $P$ *est un ensemble fini de* places*,*

– $T$ *est un ensemble fini de* transitions*, avec* $P \cap T = \emptyset$*,*

– ${}^\bullet(.) \in (\mathbb{N}^P)^T$ *est la* fonction d'incidence amont*,*

– $(.)^\bullet \in (\mathbb{N}^P)^T$ *est la* fonction d'incidence aval*,*

– $M_0 \in \mathbb{N}^P$ *est le* marquage initial*,*

– $\alpha \in (\mathbb{R}^+ \cup Q)^T$ *et* $\beta \in (\mathbb{R}^+ \cup Q \cup \{\infty\})^T$ *sont les fonctions donnant pour chaque transition, respectivement son instant de tir* au plus tôt *et* au plus tard *(avec* $\alpha \leq \beta$*).*

La sémantique concrête d'un RdPT paramétré $N$, définie pour une valuation $\nu$, et notée $[N]_\nu$, est une sémantique de type monoserveur ou une transition sensibilisée ne

peut pas dépasser sa borne temporelle maximale (sémantique forte) [BER 91]. Pour rendre possible le processus de compositionnalité nous restreignons notre démarche au cadre de réseaux de Petri T-temporels saufs [MER 74].

### 2.3. *Traduction du réseau en Logique Linéaire*

La suite de la démarche est de traduire le réseau en Logique Linéaire [GIR 87]. L'analyse d'un RdPT est traditionnellement un calcul exploratoire de type graphe des classes [BER 91] ou graphe des zones [GAR 03]. En logique linéaire, nous réalisons cette analyse grâce à un processus de preuve de séquents de logique linéaire (un séquent correspondant à un scénario). Nous nous basons pour cela sur des travaux montrant une équivalence entre les séquents de logique linéaire et les scénarios dans les réseaux de Petri [GIR 97, PRA 99].

**Définition 2 (Scénario)** *Un* scénario *est un ensemble de franchissements de transition (évènements) qui permet de passer d'un marquage initial $M$ à un marquage final $M'$ [PRA 99].*

Tout comme les techniques de dépliage des RdP [MCM 95, CHA 06], l'analyse de scénarios par la logique linéaire est une approche « évènements » de model-checking, à opposer aux approches « états » citées précédemment. Un *évènement* $e$ représente le tir d'une transition $t$ du réseau à une date de tir symbolique $\theta(e)$. On note $\Theta$ l'ensemble des dates de tir d'un scénario. Ainsi, un évènement résout tout conflit qui implique la transition. La notion de scénario se rapproche donc de celle de processus dans les dépliages. Les scénarios considérés dans cet article seront finis.

Une méthode d'analyse, en sémantique forte, de scénarios exprimés en logique linéaire, a été développée à partir de ces travaux [DEL 07]. Elle produit in fine, un système d'inéquations linéaires portant sur les dates de tir $\theta(e)$ de chaque évènement $e \in E$ et sur les paramètres $q$, que nous appellerons *domaine de tir*. La construction d'un préfixe fini du dépliage du RdPT, comme décrit dans [CHA 06], produit des résultats similaires. C'est sur ce système d'inéquations que nous appliquons l'algorithme présenté par la suite, afin de déterminer les contraintes de conception. Au final, les contraintes de conception sur le module seront la conjonction de celles trouvées pour chacun des scénarios étudiés.

### 2.4. *Domaine de tir d'un scénario et contraintes temporelles*

**Définition 3 (Domaine de tir d'un scénario)** *Le* domaine de tir *d'un scénario $E$ est un ensemble $I(\Theta, Q)$ d'inéquations, qui donne pour chaque évènement $e \in E$ une ou plusieurs bornes minimales et maximales pour sa date de tir $\theta(e)$. Dans sa forme normale, un domaine de tir est une expression booléenne sous forme conjonctive, et les inéquations $i$, en conjonction ou en disjonction, ont la forme suivante :*

- $\theta(e_k) + \alpha(t) \leq \theta(e)$, *pour la date de tir au plus tôt,*
- $\theta(e) \leq \theta(e_l) + \beta(t)$, *pour la date de tir au plus tard,*

*avec* $e_k$ *et* $e_l$ *deux évènements de* $E$ *distincts de* $e$*, et* $t$ *une transition de* $T$*.*

Le domaine de tir $(I)$ est donc une conjonction d'expressions booléennes. Pour la plupart, ces expressions sont des inéquations, cependant certaines peuvent être des disjonctions d'inéquations. En effet, soit un évènement $e_t$ représentant le tir d'une transition $t$, cet évènement peut dépendre de jetons produits par les évènements $e_k, \ldots, e_l$, dont les dates de productions sont indépendantes, et l'on aura donc une inéquation du type :

$$\theta(e_t) \leq max(\theta(e_k), \ldots, \theta(e_l)) + \beta(t)$$

L'opérateur $max(.)$ peut alors être décomposé en une disjonction d'inéquations :

$$\theta(e_t) \leq \theta(e_k) + \beta(t) \vee \cdots \vee \theta(e_t) \leq \theta(e_l) + \beta(t)$$

*Contraintes temporelles*

Comme évoqué dans la partie inférieure de la figure 1, un cahier des charges impose sur le système modélisé des contraintes temporelles à vérifier. Ces contraintes sont projetées sur les modules, et traduites en inéquations. Nous possédons donc un second système $(\phi)$ d'inéquations, sous forme conjonctive, correspondant aux propriétés temporelles à vérifier. Ces inéquations seront mises en forme de telle sorte que leur deux membres soient une somme entre des dates de tir $\theta(e)$, des paramètres $q$ et des constantes.

**Définition 4 (Contraintes temporelles projetées)** *Étant donné un réseau de Petri* $N$*, les contraintes temporelles projetées sur* $N$ *sont un système d'inéquations* $\phi(\Theta, Q)$ *sous forme normale conjonctive, dont les opérandes sont des dates de tir* $\theta(t) \in \Theta$ *des transitions* $t \in T$*, des paramètres* $q \in Q$ *et des constantes numériques.*

**Définition 5 (Contraintes de conception)** *Étant donné un réseau de Petri* $N$*, un scénario* $E$ *de* $N$ *et son domaine de tir* $I$*, des contraintes temporelles* $\phi$ *projetées sur* $N$*, les contraintes de conception de* $N$ *sont un système d'inéquations* $\psi(Q)$*, sous forme normale conjonctive, dont les opérandes sont uniquement des paramètres* $q \in Q$ *et des constantes numériques, et tel que :* $I \wedge \psi \Rightarrow \phi$*.*

### 2.5. *Algorithme de génération des contraintes de conception*

*Notations :*

Soit $X(V)$ un ensemble d'inéquations portant sur des variables $V$. Notons $[X(V)]_v$ la substitution dans $X(V)$ des variables $V$ par leur valuation $v$.

Soit $P_W$ la projection sur l'ensemble des variables $W$, définie par :

$$P_W(X(V,W)) = \{w \in W \mid \exists v \in V,\ [X(V,W)]_{(v,w)}\}$$

**Théorème 1 (Problème)** *À partir du domaine de tir $I(\Theta, Q)$ d'un scénario et des contraintes temporelles projetées $\phi(\Theta, Q)$, on détermine les contraintes de conception $\psi(Q)$, ne dépendant que des paramètres Q, et telles que $I(\Theta, Q) \wedge \psi(Q) \Rightarrow \phi(\Theta, Q)$. Soit $P_Q$ la projection sur l'ensemble des paramètres Q, la condition $\neg P_Q(I(\Theta, Q) \wedge \neg\phi(\Theta, Q))$ est nécessaire et suffisante pour établir $\psi(Q)$. On a donc :*

$$\psi(Q) = \neg P_Q(I(\Theta, Q) \wedge \neg\phi(\Theta, Q))$$

*Preuve de la correction :*

Pour toute valuation $\nu \in \mathbb{R}^{+Q}$ des paramètres vérifiant $[\psi(Q)]_\nu$, il faut vérifier que toute solution $\sigma \in \mathbb{R}^{\Theta}$ vérifie $[I(\Theta, Q)]_{(\sigma,\nu)} \Rightarrow [\phi(\Theta, Q)]_{(\sigma,\nu)}$.

Puisque $[\psi(Q)]_\nu$, on a $[\neg P_Q(I(\Theta, Q) \wedge \neg\phi(\Theta, Q))]_\nu$. Si $\nu$ n'est pas dans la projection, alors $\forall\sigma \in \mathbb{R}^{\Theta}$, $[\neg(I(\Theta, Q) \wedge \neg\phi(\Theta, Q))]_{(\sigma,\nu)}$.

D'où en développant : $\neg[I(\Theta, Q)]_{(\sigma,\nu)} \vee [\phi(\Theta, Q)]_{(\sigma,\nu)}$, ce qui est équivalent à : $[I(\Theta, Q)]_{(\sigma,\nu)} \Rightarrow [\phi(\Theta, Q)]_{(\sigma,\nu)}$.

*Preuve de la complétude :*

Raisonnons par l'absurde en considérant une valuation $\nu \in \mathbb{R}^{+Q}$ qui ne vérifie pas $\psi(Q)$, mais qui est solution du **Problème**, c'est à dire telle que $\forall\sigma \in \mathbb{R}^{\Theta}$, $[I(\Theta, Q)]_{(\sigma,\nu)} \Rightarrow [\phi(\Theta, Q)]_{(\sigma,\nu)}$.

Puisque $\nu$ ne vérifie pas $\psi(Q)$, $\nu$ vérifie $P_Q(I(\Theta, Q) \wedge \neg\phi(\Theta, Q))$. Si $\nu$ est dans la projection sur $Q$, cela signifie qu'il existe une valuation $\sigma' \in \mathbb{R}^{\Theta}$ telle que $[I(\Theta, Q) \wedge \neg\phi(\Theta, Q)]_{(\sigma',\nu)}$.

Ceci peut s'écrire sous la forme : $\neg(\neg[I(\Theta, Q)]_{(\sigma',\nu)} \vee [\phi(\Theta, Q)]_{(\sigma',\nu)})$, ce qui est encore équivalent à : $\neg([I(\Theta, Q)]_{(\sigma',\nu)} \Rightarrow [\phi(\Theta, Q)]_{(\sigma',\nu)})$, d'où la contradiction avec les hypothèses initiales. Les contraintes $\psi$ calculées délimitent donc tout l'ensemble des solutions au problème.

*Principe de l'algorithme*

L'algorithme implémente la solution du **Problème** qui est $\psi(Q) = \neg P_Q(I(\Theta, Q) \wedge \neg\phi(\Theta, Q))$. La projection $P_Q$ est réalisée en éliminant successivement par déduction (principe de Fourier-Motzkin) dans $I(\Theta, Q) \wedge \neg\phi(\Theta, Q)$ toutes les variables $\Theta$. Finalement, on détermine la solution en prenant la négation du résultat.

*Notations utilisées dans l'algorithme*

Dans cet algorithme, les fonctions $l_d()$ et $l_g()$ fournissent respectivement le membre droit et le membre gauche d'une inéquation. Toutes les inéquations $i$ traitées sont de la forme : $i = (l_g(i) \sim l_d(i))$ avec $\sim \in \{\leq, <\}$.

$\overline{\psi} \leftarrow \emptyset$
$\overline{\phi} \leftarrow non(\phi)$
**Pour Chaque** $\varphi \in \overline{\phi}$ **Faire**
    Soit $\overline{\psi_k}$ un système d'inéquations sous forme disjonctive
    $\overline{\psi_k} \leftarrow \varphi$
    **Tant que** $\exists i \in \overline{\psi_k}$, $\exists e_t \in E, \theta(e_t) \in l_d(i) \cup l_g(i)$ *(avec t "maximum")* **Faire**
        **Si** $\theta(e_t) \in l_d(i) \cap l_g(i)$ **Alors**
            Supprimer $\theta(e_t)$ de $i$
        **Sinon Si** $\theta(e_t) \in l_d(i)$ **Alors**
            Soit $M \subset (I)$ l'ensemble des inéquations telles que :
            $\forall j \in M, l_g(j) = \theta(e_t)$
            $M$ *est une forme conjonctive des valeurs maximales de* $\theta(e_t)$
            **Pour Chaque** $j \in M$ **Faire**
                $h \leftarrow i[l_d(j)/\theta(e_t)]$
                $M \leftarrow M[h/j]$
            **Fin Pour**
            $\overline{\psi_k} \leftarrow \overline{\psi_k}[M/i]$
            $\overline{\psi_k} \leftarrow dij(\overline{\psi_k})$
        **Sinon**
            Soit $M \subset (I)$ l'ensemble des inéquations telles que :
            $\forall j \in M, l_d(j) = \theta(e_t)$
            $M$ *est une forme conjonctive des valeurs minimales de* $\theta(e_t)$
            **Pour Chaque** $j \in M$ **Faire**
                $h \leftarrow i[l_g(j)/\theta(e_t)]$
                $M \leftarrow M[k/j]$
            **Fin Pour**
            $\overline{\psi_k} \leftarrow \overline{\psi_k}[M/i]$
            $\overline{\psi_k} \leftarrow dij(\overline{\psi_k})$
        **Fin Si**
    **Fin Tant que**
    $\overline{\psi} \leftarrow \overline{\psi} \vee \overline{\psi_k}$
**Fin Pour**
Retourner $non(\overline{\psi})$

**Algorithme 1** : Génération des contraintes de conception

On considère de plus, des fonctions de substitution : $F[t/x]$ qui signifie que l'on substitue la variable $x$ par $t$ dans l'expression $F$.

Enfin, on se donne deux fonctions de manipulation d'expressions logiques. $non(.)$ qui donne la négation d'un ensemble d'inéquations, en inversant le sens des inéquations et en remplaçant les conjonctions (respectivement les disjonctions) par des disjonctions (respectivement des conjonctions) ; et $dij(.)$ qui met un système d'inéquations sous une forme disjonctive.

*Remarque*

Dans les inéquations des contraintes, les dates des évènements doivent être remplacées dans l'ordre inverse d'apparition dans le scénario, afin que toutes les éliminations possibles soient effectuées (premier cas de l'alternative). Cet ordre est l'ordre partiel connu entre les évènements du scénario. Si l'on note les évènements $e_t$ selon cet ordre, il s'agit alors d'éliminer en priorité celui dont $t$ est le plus élevé.

Nous illustrons dans la section suivante cette méthode et cet algorithme sur un cas d'étude.

## 3. Cas d'étude

Ce cas d'étude a été extrait, sans modification, d'une application industrielle représentant un système transformationnel, impliquant des tâches $A$, $B$, $C$ qui échangent des données par le biais de deux canaux de communication $i_1$ et $i_2$. Ce système est soumis à des contraintes temporelles portant sur les durées d'exécution des tâches, les délais de transmissions ainsi que sur les occurrences des évènements.

### 3.1. *Modélisation informelle*

Le cahier des charges a été donné par l'industriel sous la forme d'une machine à état, présentée sur la figure 2, et d'un ensemble de contraintes temporelles portant sur les durées de séjours dans les états ainsi que sur certaines durées de transmissions. Certaines durées sont connues d'autres non.

Sont connus, le temps d'exécution de la tâche $A$, qui varie entre 2 et $5\ ms$, le temps d'exécution de la tâche $B$, qui est de $10ms$, le temps d'exécution de la tâche $C$, qui est de $8\ ms$, et le délai de transmission de $i_1$, qui est de $20\ ms$.

Par ailleurs, ce système doit satisfaire certaines contraintes :

– Le vieillissement maximum de $i_1$ en entrée de $B$ (relativement à la date de réception de $eA$) doit être de $30\ ms$ ;

– le vieillissement maximum de $i_2$ en entrée de $B$ (relativement à la date de réception de $eC$) doit être de $50\ ms$ ;

– $i1$ et $i2$ en entrée de $B$ ne doivent pas être reçues avec plus de $20\ ms$ d'écart ;

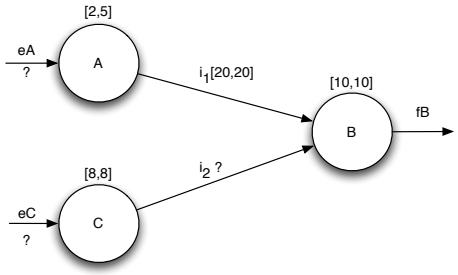


**Figure 2.** *Cahier des charges*

– le temps de traversée maximum du système est de $80\ ms$.

Le délai de transmission de $i_2$ n'est par contre pas connu. Nous allons donc dans cette étude considérer deux paramètres $i_2m$ et $i_2M$ représentant respectivement la borne minimale et la borne maximale de ce délai de transmission.

### 3.2. *Modélisation de l'application sous forme de réseau de Petri*

On cherche dans un premier temps à modéliser de façon formelle cette machine à états par un réseau de Petri T-temporel paramétré. On peut relever dans ce cahier des charges les caractéristiques suivantes :

– Certaines données temporelles ne sont pas connues. On aura donc au niveau des transitions des valeurs constantes et des valeurs paramétrées.

– Les tâches seront modélisées par le triplet $(transition, place, transition)$ pour représenter l'entrée dans la tâche, sa durée et sa fin. Par exemple la tâche $A$ sera modélisée par $(eA, A, fA)$

– La fin de transmissions des donnés $i_1$ et $i_2$ amène le contrôle du réseau dans les états $i_1enB$ et $i_2enB$. Ces états sont nécessaires pour modéliser le fait que la tâche $B$ peut ne pas les prendre en compte immédiatement comme l'indique les contraintes exprimées sur le vieillissement de ces données.

Le cahier des charges peut donc être modélisé par le réseau de la figure 3. Le scénario étudié dans cet exemple est constitué d'une occurrence de tir de chaque transition du réseau.

Le but de cette étude est de déterminer les conditions sur les paramètres du système, qui permettent d'assurer que les contraintes précédentes sont nécessairement vérifiées. Les paramètres du système sont $i_2m$ et $i_2M$, ainsi que les dates $\theta(eA)$ et $\theta(eC)$ de réception des données $eA$ et $eC$ en entrée de $A$ et de $C$ (c'est à dire les dates de tir des transitions associées, $eA$ et $eC$). Étant donné que les contraintes sont relatives aux dates de réception de $eA$ et de $eC$, on cherchera uniquement des conditions sur la différence $\theta(eA) - \theta(eC)$.

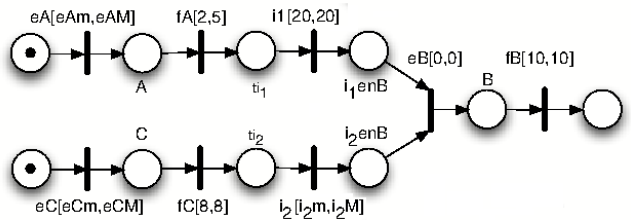


**Figure 3.** *Modélisation formelle*

### 3.3. *Domaine de tir et contraintes temporelles*

Par un processus d'analyse du réseau de Petri utilisant la logique linéaire, nous déterminons pour chaque transition, les bornes minimales et maximales de sa date de tir. Nous déterminons ainsi les dates de tir $\theta(fA)$, $\theta(i_1)$, $\theta(fC)$, $\theta(i_2)$, $\theta(eB)$ et $\theta(fB)$, des transitions du réseau de Petri de la figure 3, en fonction des paramètres $i_2m$, $i_2M$, $\theta(eA)$ et $\theta(eC)$, ce qui forme le système d'inéquations $(I)$ sous forme conjonctive :

$$\theta(eA) + 2 \leq \theta(fA) \leq \theta(eA) + 5 \quad (1)$$

$$\theta(fA) + 20 \leq \theta(i_1) \leq \theta(fA) + 20 \quad (2)$$

$$\theta(eC) + 8 \leq \theta(fC) \leq \theta(eC) + 8 \quad (3)$$

$$\theta(fC) + i_2m \leq \theta(i_2) \leq \theta(fC) + i_2M \quad (4)$$

$$\theta(i_1) \leq \theta(eB) \quad (5)$$

$$\theta(i_2) \leq \theta(eB) \quad (6)$$

$$\theta(eB) \leq \theta(i_1) \quad \vee \quad \theta(eB) \leq \theta(i_2) \quad (7)$$

$$\theta(eB) + 10 \leq \theta(fB) \leq \theta(eB) + 10 \quad (8)$$

Nous devons ensuite interpréter les contraintes du problème et les traduire également en inéquations. La première contrainte $C1$ s'interprète ainsi : la donnée issue de $eA$ doit être inférieure (relativement à l'origine temporelle $\theta(eA)$) à 30 $ms$ lorsqu'elle est utilisée par $B$. Cette donnée est utilisée dès que l'autre donnée issue de $eC$ est arrivée, c'est à dire dès que la transition $eB$ est franchie. Nous en déduisons l'inéquation suivante :

$$\theta(eB) \leq \theta(eA) + 30$$

De même pour la contrainte $C2$, concernant le vieillissement maximal de i2 :

$$\theta(eB) \leq \theta(eC) + 50$$

La contrainte $C3$ spécifie que les deux données $i_1$ et $i_2$, en entrée de B, ne doivent pas arriver avec plus de 20 $ms$ d'écart. Nous en déduisons la contrainte suivante $|\theta(i_1) - \theta(i_2)| \leq 20$, ce qui peut encore s'écrire :

$$-20 \leq \theta(i_1) - \theta(i_2) \leq 20$$

Enfin, la contrainte $C4$ spécifie que le temps maximum de traversée du système ne doit pas excéder 80 $ms$, ce qui s'écrit : $\theta(fB) \leq min(\theta(eA), \theta(eC)) + 80$. L'opérateur $min$ peut être supprimé, créant ainsi une conjonction de deux inéquations :

$$\theta(fB) \leq \theta(eA) + 80$$
$$\theta(fB) \leq \theta(eC) + 80$$

Les inéquations précédentes forme le système d'inéquations $(\phi)$ des contraintes temporelles, qui est également sous forme conjonctive.

### 3.4. *Détermination des contraintes de conception*

A partir des deux systèmes d'inéquations précédents, nous allons déterminer les contraintes de conception que les paramètres du système devront vérifier, afin que les contraintes initiales soient nécessairement vérifiées. Appliquons pour cela l'algorithme 1

*Contrainte* $C1$

$$\varphi_1 = \theta(eB) \leq \theta(eA) + 30 \qquad \longrightarrow \qquad \overline{\varphi_1} = \theta(eA) + 30 < \theta(eB)$$

A l'aide des inéquations 1,2,3,4 et 7, on en déduit :

$$(I) \wedge \overline{\varphi_1} \Rightarrow \underbrace{\theta(eA) + 30 < \theta(eA) + 25 \quad \vee \quad \theta(eA) + 30) < \theta(eC) + i_2M + 8}_{\overline{\psi_1}}$$

D'où la contrainte de conception :

$$\psi_1 = \quad \theta(eA) + 25 \leq \theta(eA) + 30 \quad \wedge \quad \theta(eC) + i_2M + 8 \leq \theta(eA) + 30$$

La première est automatiquement vérifiée, la seule contrainte de conception à prendre en compte est donc :

$$\boxed{i_2M - 22 \leq \theta(eA) - \theta(eC)} \tag{9}$$

*Contrainte* $C2$

Pour l'inéquation $\varphi_2 = \theta(eB) \leq \theta(eC) + 50$, nous utilisons la même méthode et nous obtenons les contraintes de conception suivantes :

$$\boxed{\theta(eA) - \theta(eC) \leq 25} \tag{10}$$

$$\boxed{i_2M \leq 42} \tag{11}$$

*Contrainte* $C3$

Pour la contrainte C3, deux inéquations sont à vérifiées :

$$\varphi_3 = \theta(i_2) \leq \theta(i_1) + 20 \quad \text{et} \quad \varphi_4 = \theta(i_1) \leq \theta(i_2) + 20$$

Nous calculons $\overline{\varphi_3}$ et $\overline{\varphi_4}$, et à l'aide des inéquations 1, 2, 3 et 4, nous en déduisons :

$$\overline{\psi_3} = \quad \theta(eA) + 42 < \theta(eC) + i2M + 8$$

et :

$$\overline{\psi_4} = \quad \theta(eC) + i2m + 28 < \theta(eA) + 25$$

D'où, en prenant la négation, les contraintes de conception :

$$\boxed{i_2M - 34 \leq \theta(eA) - \theta(eC)} \tag{12}$$

$$\boxed{\theta(eA) - \theta(eC) \leq i_2m + 3} \tag{13}$$

*Contrainte* $C4$

Deux inéquations à vérifier :

$$\varphi_5 = \theta(fB) \leq \theta(eA) + 80 \quad \text{et} \quad \varphi_6 = \theta(fB) \leq \theta(eC) + 80$$

$$(I) \wedge \overline{\varphi_5} \Rightarrow \underbrace{\theta(eA) + 80 < \theta(eA) + 35 \quad \vee \quad \theta(eA) + 80 < \theta(eC) + i_2M + 18}_{\overline{\psi_5}}$$

$$(I) \wedge \overline{\varphi_6} \Rightarrow \underbrace{\theta(eC) + 80 < \theta(eA) + 35 \quad \vee \quad \theta(eC) + 80 < \theta(eC) + i_2M + 18}_{\overline{\psi_6}}$$

Nous déterminons alors $\psi_5$ et $\psi_6$, et nous extrayons les trois contraintes de conception suivantes :

$$\boxed{i_2M - 62 \leq \theta(e_A) - \theta(e_C)} \tag{14}$$

$$\boxed{\theta(eA) - \theta(eC) \leq 45} \tag{15}$$

$$\boxed{i_2M \leq 62} \tag{16}$$

### 3.5. *Résultats et interprétation*

Les calculs précédents nous fournissent l'ensemble des contraintes de conception que les paramètres du systèmes doivent vérifier. Dans ce cas d'étude, cet ensemble de contraintes est uniquement conjonctif. Nous observons que certaines de ces

contraintes sont redondantes. Nous ne gardons que les plus restrictives, c'est à dire les inéquations 9, 10, 11 et 13 :

$$i_2M - 22 \leq \theta(eA) - \theta(eC)$$

$$\theta(eA) - \theta(eC) \leq 25$$

$$i_2M \leq 42$$

$$\theta(eA) - \theta(eC) \leq i_2m + 3$$

En les regroupant, nous obtenons au final les deux contraintes de conception suivantes :

$$\boxed{i_2M - 22 \leq \theta(eA) - \theta(eC) \leq min(25, i_2m + 3)} \quad (17)$$

$$\boxed{i_2M \leq 42} \quad (18)$$

Elles s'ajoutent à la contrainte de conception déjà connue $0 \leq i_2m \leq i_2M$.

*Interprétation des résultats*

Le cahier des charges du système étudié spécifie des contraintes temporelles sur le système. Nous avons traduit ces contraintes en inéquations $(\phi)$ sur les dates de tir des transitions du réseaux de Petri modélisant le système. Par une analyse du réseau de Petri, nous avons déterminé le domaine de tir de ces transitions, c'est à dire un ensemble $(I)$ d'inéquations, décrivant le fonctionnement du réseau. A partir de ces deux ensembles d'inéquations, nous avons déduit une nouvel ensemble $(\psi)$ d'inéquations, en faisant remonter les contraintes $(\phi)$ vers les interfaces du système, c'est à dire en les exprimant uniquement en fonction des paramètres du système. Cet ensemble d'inéquations $(\psi)$ ne porte en effet, que sur les quatre paramètres définis dans le système. Il permet, si ces inéquations sont vérifiées, d'assurer que les contraintes $(\phi)$ seront vérifiées et donc que le cahier des charges sera respecté.

Si l'on étudie ces inéquations, on s'aperçoit tout d'abord que l'on possède des bornes minimales et maximales pour le paramètre $i_2M$ (borne de tir maximale pour la transition $i_2$), qui doit donc être compris entre $[0, 42]$. Une fois ce premier paramètre choisi, l'autre borne $i_2m$ peut être choisie dans l'intervalle $[0, i_2M]$.

Les bornes de la transition $i_2$ sont alors spécifiées. La contrainte 17 permet de caractériser les entrées du système, en spécifiant l'écart minimum et maximum possible entre la réception des deux données d'entrée $eA$ et $eC$.

## 4. Conclusion

Dans le cycle de développement d'un système temps réel, les étapes de décomposition architecturale rendent délicate (voir impossible) l'utilisation des méthodes de

model-checking classique, qui ne peuvent s'appliquer qu'au niveau du système global. De plus, les approches model-checking sont souvent mises à mal par l'explosion combinatoire qui se produit lorsque l'on veut calculer l'espace d'état du système dans sa globalité.

Aussi, nous proposons une méthode s'appliquant de manière paramétrée à un niveau modulaire de l'architecture opérationnelle d'un système temps réel. Cette méthode présente l'avantage de permettre la prise en compte d'un cahier des charges incomplètement défini nécessitant l'emploi de paramètres. Les contraintes temporelles du système sont projetées sur les modules (décrits par des réseaux de Petri T-temporels paramétrés) sous la forme de contraintes paramétrées. La méthode d'analyse du module est basée sur les scénarios et la logique linéaire. Elle permet de manipuler de façon symbolique un réseau de Petri T-temporel et de prendre en compte naturellement des variables temporelles paramétrées. La méthode s'applique sur un ensemble initial d'inéquations appelé domaine de tir qui est ensuite réduit par un solveur symbolique. La suite naturelle de ces travaux est d'étendre les méthodes classiques de model-checking symbolique des réseaux de Petri temporels, basées sur le calcul du graphe des zones ou du graphe des classes d'états, afin de prendre en compte également des paramètres.

## 5. Bibliographie

[BER 91] BERTHOMIEU B., DIAZ M., « Modeling and Verification of Time Dependent Systems Using Time Petri Nets », *IEEE Trans. Softw. Eng.*, vol. 17, n° 3, 1991, p. 259–273, IEEE Press.

[CHA 06] CHATAIN T., JARD C., « Complete Finite Prefixes of Symbolic Unfoldings of Safe Time Petri Nets », *ICATPN*, vol. 4024 de *LNCS*, june 2006, p. 125-145.

[DEL 07] DELFIEU D., SOGBOHOSSOU M., TRAONOUEZ L.-M., REVOL S., « Parameterized study of a Time Petri Net », *Cybernetics and Information Technologies, Systems and Applications : CITSA 2007*, Orlando, Florida, USA, July 2007.

[GAR 03] GARDEY G., ROUX O., ROUX O., « Using Zone Graph Method for Computing the State Space of a Time Petri Net », *First International Workshop on Formal Modelling and Analysis of Timed Systems FORMATS 2003*, Marseille, France, 2003.

[GIR 87] GIRARD J. Y., « Linear logic », *Theorical Computer Science*, vol. 50, 1987.

[GIR 97] GIRAULT F., « Formalisation en logique linéaire du fonctionnement des réseaux de Petri », PhD thesis, Université Paul Sabatier, Toulouse, France, 1997.

[MCM 95] MCMILLAN K. L., « A technique of state space search based on unfolding », *Form. Methods Syst. Des.*, vol. 6, n° 1, 1995, p. 45–65, Kluwer Academic Publishers.

[MER 74] MERLIN P., « A Study of the Recoverability of Computer Systems », PhD thesis, University of California, Irvine, 1974.

[PRA 99] PRADIN-CHÉZALVIEL B., VALETTE R., KÜNZLE L. A., « Formalisation de scénarios, réseaux de Petri et logique linéaire », *Journées Formalisation des Activités Concurrentes FAC'99*, CERT-IRIT-LAAS, Toulouse, 1999.