LPT : Little Parametric Tool, outil pour la validation d'une borne temporelle paramétrée

KAREN GODARY [1]

[1] Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
161 rue Ada 34392 Montpellier Cedex 05, France

karen.godary@lirmm.fr

Résumé – Cet article s'intéresse au problème de la validation formelle de propriétés temporelles quantitatives et paramétrées pour les systèmes à évènements discrets. Le model checking est une technique d'analyse exhaustive de l'espace des états d'un système modélisé dans un lange formel. Il permet la vérification de propriétés de logique temporelle quantitative. Cependant, l'introduction de paramètres entraine une complexité trop grande de l'analyse, en particulier sur un modèle exprimé en réseaux de Petri temporels. Cet article propose donc une méthode alternative et présente un outil : LPT (Little Parametric Tool) permettant la validation d'une borne temporelle paramétrée. Avec LPT, le modèle du système est analysé et la valeur de la borne maximale est recherchée par l'outil.

Mots-clés – Validation, model checking, réseaux de Petri temporels, analyse paramétrée, systèmes à évènements discrets

I. INTRODUCTION

Ces travaux se situent dans le cadre de la validation formelle de contraintes temporelles quantitatives et paramétrées pour des systèmes à évènements discrets. Lors de la conception de ces systèmes, une première étape consiste en leur modélisation et leur validation. Dans ce cadre il est souvent nécessaire de vérifier des propriétés temporelles de type quantitatif, telles que par exemple des durées d'exécution. De plus, il serait très utile de pouvoir utiliser des paramètres lors du processus d'analyse. Par exemple, pour vérifier qu'une action a une durée maximale d'exécution, mais sans connaitre cette durée au préalable. Mais l'introduction de paramètres au sein des modèles ou des propriétés vérifiées sur ces modèles est difficile et entraîne une grande complexité des méthodes de validation.

Cet article présente une alternative simple à ce problème, permettant la validation paramétrée dans le cas particulier de la vérification d'une durée maximale d'exécution entre deux évènements. L'outil proposé (LPT, Little Parametric Tool) permet de paramétrer la durée d'exécution entre deux transitions d'un modèle exprimé en réseaux de Petri temporels. Cet outil s'applique à la validation par model checking du modèle, et permet d'extraire la borne maximale du temps d'exécution de façon automatique. Du point de vue complexité, cette technique est très proche de la complexité d'une analyse classique (non paramétrée), et cela malgré l'introduction du paramètre.

II. CONTEXTE

Cet article propose de prendre comme exemple le cadre des systèmes distribués communicants, et plus exactement les travaux exposés dans [1], qui présentent la validation formelle d'un nouveau protocole MAC appelé STIMAP (Sliding Time Interval based Medium Access Protocol).

A. Contexte applicatif

Le protocole STIMAP a été conçu pour prendre en charge le niveau communication d'une architecture de stimulation électrique neurale distribuée[1], architecture dédiée à la stimulation artificielle du système nerveux de personnes handicapées. La stimulation électrique fonctionnelle peut ainsi être appliquée pour la stimulation cardiaque ou auditive, mais aussi pour des applications plus complexes comme la réhabilitation de mouvements. Le système devient alors un ensemble d'unités de stimulation implantées dans le corps humain. Ces unités doivent communiquer pour répartir la fonction de stimulation, pour partager les données et pour se synchroniser. Dans ce cadre applicatif, il est évident que la communication est un des éléments critiques de l'architecture. Il ne pourra par exemple pas être toléré un retard dans les transmissions.

B. Architecture cible, niveau communication

Le système à valider est donc une architecture distribuée sur un médium partagé. L'accès au medium est basé sur deux concepts : un mécanisme maître-esclave et une attribution au médium en fonction de priorités statiques. La communication dans STIMAP s'effectue suivant deux possibilités : soit le maître ne donne la parole qu'à un seul nœud esclave, ce qui ne pose alors pas de problème d'accès au médium; soit le maître interroge un groupe de nœuds qui devront tous lui répondre. Dans ce cas, un nœud esclave devra attendre son tour en fonction de sa priorité, attribuée de façon statique (pour plus de détails sur le protocole STIMAP, voir [1] ou [2]). L'important ici est que ce protocole garanti aux nœuds un accès au médium déterministe : leur durée d'attente pour accéder au médium est donc censée être bornée. Cette garantie est capitale dans le contexte applicatif décrit précédemment, et c'est donc une propriété prioritaire à valider.

[1] cf. le projet DEMAR, www-sop.inria.fr/demar/

III. PROCESSUS DE VALIDATION

A. Technique d'analyse

La validation formelle du protocole STIMAP a été effectuée par model checking [21]. Cette méthode consiste dans un premier temps à modéliser le système en langage formel de type machine à états (réseaux de Petri, automates), puis à construire l'espace d'états du modèle : le graphe d'analyse. Il s'agit enfin de parcourir exhaustivement l'ensemble des états de ce graphe en vérifiant la propriété désirée.

Le langage formel utilisé pour la modélisation de STIMAP sont les réseaux de Petri temporels (RdPT) [3]. C'est un formalisme adapté à la modélisation des systèmes à évènements discrets, systèmes concurrents communicants et synchronisés. Les résultats de validation de cet article ont été obtenus à l'aide de l'outil de modélisation et d'analyse de réseaux de Petri temporels Tina [4], qui implémente efficacement la construction du graphe des classes [5] (graphe d'analyse typique des RdPT).

B. Propriété(s) à vérifier

Il s'agit ici de la validation d'une borne temporelle d'exécution : la durée maximale pouvant s'écouler entre deux évènements. Pour un nœud donné, la propriété à valider pour l'accès au medium peut s'exprimer ainsi :

Prop1 : "A partir de l'instant ou le nœud veut émettre sa trame, et jusqu'à l'instant ou ce nœud émet réellement, il ne doit pas s'écouler plus d'une durée maximale."

La validation du protocole STIMAP a été effectuée en vérifiant pour chacun des nœuds branchés sur le réseau son temps maximal d'attente pour l'accès au médium. La validation totale du mécanisme d'accès au medium de STIMAP consiste donc en la vérification de la Prop1 pour tous les nœuds du système.

C. Technique de l'observateur

Prop1 est une propriété de type vivacité (notion de temps quantitatif). Ces propriétés ne sont pas vérifiables sur un graphe des classes classique, qui ne conserve pas les informations de dates absolues des horloges. On utilise alors un observateur (ou chien de garde) qui permet de modéliser la propriété et de transformer le problème en une vérification de l'accessibilité d'un état spécifique.

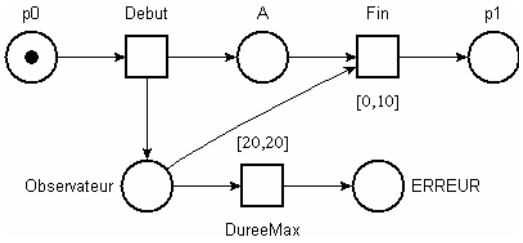

Figure 1 : Observateur

La figure 1 représente le principe de base d'un observateur permettant la vérification de la durée maximale d'une action A, représentée par la place `A`. La transition `Debut` correspond au lancement de l'action. Le tir de cette transition génère un jeton dans une place appelée `Observateur`, ce qui sensibilise la transition `DureeMax` et déclenche le décompte de son intervalle de tir `[20,20]`. Ici, seule la borne maximale nous intéresse, l'intervalle est donc représenté par une durée. La place `Observateur` sera démarquée soit lors du tir de la transition `Fin`, c'est-à-dire lorsque A se termine, soit lorsque la durée de l'intervalle de tir de la transition `DureeMax` est écoulée avant la fin de A : `DureeMax` est alors tirée et la place `ERREUR` est marquée. Le marquage de cette place signifie donc que le temps d'exécution de l'action A est plus long que la durée 20. La vérification de la durée maximale d'exécution de l'action A devient donc un problème d'accessibilité de la place `ERREUR`.

D. Borne(s) paramétrée(s)

L'exemple présenté ci-dessus part de l'hypothèse que la borne maximale associée à la transition `DureeMax` est connue. Or l'expérience montre que même si la connaissance approfondie du système permet d'obtenir un ordre d'idée de cette borne maximale, il est rare d'en connaître la valeur avec exactitude dans des systèmes complexes (voir [6] pour un exemple de borne complexe). Il serait donc intéressant de pouvoir effectuer une analyse du modèle en paramétrant l'intervalle de la transition `DureeMax`, afin que ce soit l'analyse qui permette d'obtenir cette borne maximale.

IV. VALIDATION QUANTITATIVE ET PARAMETREE : ETAT DE L'ART

La vérification de propriétés quantitatives est une problématique résolue dans le domaine de l'analyse des modèles exprimés en automates temporisés (AT) [7]. Ce même article [7] montre la décidabilité du model checking de propriétés TCTL (logique temporelle quantitative) sur les AT, et cette méthode est implémentée dans plusieurs outils d'analyse comme par exemple UPPAAL [9].

L'introduction de paramètres est par contre encore un domaine ouvert. Alur et Dill dans [10] ont montré que l'introduction de paramètres au sein même du modèle entraîne une analyse indécidable. Une solution pour atteindre la décidabilité est de considérer un sous-ensemble des automates temporisés paramétrés [10] ou un sous ensemble de la logique TCTL paramétrée [11]. Ce n'est qu'assez récemment qu'une solution décidable a été proposée par Raskin et Bruyère [12] pour le model checking de propriétés de logique TCTL paramétrée sur des automates temporisés. Ces mêmes auteurs ont ensuite tenté d'introduire des paramètres dans le modèle mais ont montré que l'analyse devient rapidement indécidable sauf sous certaines hypothèses [13]. Ces travaux ne sont pas encore implémentés dans des outils d'analyse. Dans ce domaine, à ma connaissance seul [14] propose une implémentation logiciel (extension de l'outil UPPAAL). Ces travaux proposent une analyse des automates temporisés paramétrés mais qui n'est pas décidable dans certains cas, et identifie une sous-classe des automates temporisés paramétrés pour laquelle l'analyse est décidable.

Si l'on s'intéresse au formalisme des réseaux de Petri temporels (RdPT), le problème est encore plus complexe. Le premier problème réside dans l'analyse quantitative des RdPT. La méthode traditionnelle de construction de l'espace des états de ce modèle est la méthode du graphe des classes d'états [5]. Mais ce graphe ne conserve pas les informations de dates absolues des horloges, ce qui empêche l'analyse des propriétés quantitatives. C'est pour cette raison qu'une première approche a été de proposer des transformations des RdPT en AT [15, 16] afin d'utiliser les outils de ce dernier formalisme. Mais la transformation automatique est souvent basée sur une multiplication des horloges (un automate par transition du RdPT initial) ce qui entraine une grande complexité de l'analyse de l'AT résultant. Une autre solution, proposée dans [17], réside

dans une nouvelle technique de construction de l'espace d'état des RdPT : le graphe de zones, adaptation du graphe des régions utilisé pour les AT [7]. Ces travaux permettant l'analyse quantitative ont récemment été implémentés dans l'outil Roméo [19].

Réside encore le problème de l'introduction de paramètres dans le processus d'analyse des RdPT. Une approche, proposée dans [22] et [18], transforme le problème d'analyse en la vérification d'un ensemble d'inéquations représentant des contraintes de conception. Cette solution permet d'effectuer une analyse paramétrée mais en reportant le problème sur chacun des modules d'un système, sans plus considérer le système global. Il n'existe à priori pas d'autres travaux aboutis associant la dimension temporelle et l'introduction de paramètres pour l'analyse formelle des réseaux de Petri.

Ainsi, l'analyse de propriétés temporelles quantitatives et paramétrées reste un problème, en particulier pour les RdP. L'introduction de paramètres entraine une complexité qui pour l'instant ne s'accorde pas aux limites des techniques et des outils d'analyse existants.

Cet article propose donc de contourner le problème en représentant les propriétés quantitatives par un observateur, et en remplaçant l'introduction de paramètres par l'exécution multiple de l'analyse avec des valeurs de bornes différentes. Si la méthode n'est pas nouvelle, cet article propose un outil permettant d'obtenir automatiquement une telle borne à partir d'un modèle en RdPT, sans entrainer une augmentation problématique de la complexité du processus d'analyse.

V. DESCRIPTION DE NOTRE APPROCHE

A. *Description du logiciel*

Le logiciel LPT (Little Parametric Tool) proposé est dans sa première version adapté à l'outil de modélisation et d'analyse des réseaux de Petri temporels Tina [4]. LPT s'applique sur un RdPT initial modélisé avec l'outil Tina et génère un ensemble de fichiers respectant la syntaxe de cet outil. Une version compatible avec l'outil Romeo [19], autre outil d'analyse des réseaux de Petri temporels, est en cours. La figure 2 montre les interactions de LPT et ses entrées / sorties. L'outil LPT a été codé en langage C et est exécutable en ligne de commande. Il est pour l'instant disponible en téléchargement libre [20].

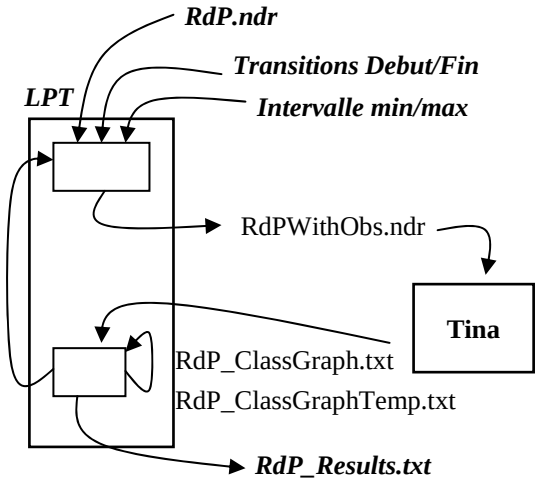

Figure 2 : E/S du logiciel LPT

Le logiciel LPT prend en paramètres le nom du fichier comportant le réseau de Petri initial (`RdP.ndr`), les noms des deux transitions (`Debut` et `Fin`) entre lesquelles la borne du pire temps d'exécution va être vérifiée/cherchée, et un intervalle dans lequel doit se situer cette borne (borne `min` et borne `max` de l'intervalle). En l'absence de ces deux derniers paramètres, l'intervalle par défaut sera [0,5000]. Le fichier de sortie final est le fichier `RdP_Results.txt`, dans lequel sont indiqués les résultats importants de la validation comme la valeur de la borne ou la place mémoire (taille du graphe d'analyse) et la durée de l'analyse. Ce fichier contient également les résultats des analyses intermédiaires nécessaires à la recherche de la borne.

B. *Algorithme de recherche de la borne paramétrée*

La recherche de la borne paramétrée peut s'apparenter à une problématique de recherche d'un nombre dans un intervalle trié. La recherche dans LPT s'effectue donc par un algorithme classique par dichotomie, algorithme connu pour son efficacité (complexité logarithmique) [23] pour ce type de problème. Cet algorithme, donné de façon simplifiée figure 3, effectue la recherche de la borne maximale à l'intérieur de l'intervalle donné en paramètre.

Chaque itération de la boucle de recherche représente l'analyse du modèle initial pour une valeur de la borne donnée. Si l'on se reporte à la figure 1, une itération représente la vérification pour une valeur donnée de la durée associée à la transition `DureeMax`. A chaque itération, le modèle initial est tout d'abord modifié pour ajouter de façon automatique l'observateur entre les deux transitions `Debut` et `Fin`. La valeur de la durée associée à la transition `DureeMax` dépend de "l'indice d'itération" de l'algorithme dichotomique. Le nouveau modèle RdP obtenu (`RdPWithObs.ndr`) est un fichier de format compatible avec l'outil Tina. La seconde étape consiste alors à construire le graphe des classes de ce nouveau modèle. On utilise pour cela l'outil Tina. La commande de construction du graphe est appelée automatiquement à l'intérieur du programme LPT : l'utilisateur n'a pas à le faire lui-même. Le résultat obtenu, le fichier texte `RdPWithObs_ClassGraph.txt`, est ensuite analysé pour conclure si la place `ERREUR` est atteignable pour la valeur de `DureeMax` testée. Ce fichier texte contient en effet l'ensemble des états atteignables du système, il suffit donc de vérifier si la chaîne de caractères "ERROR" est présente dans ce fichier. Il est toutefois nécessaire avant cette étape de modifier le fichier `RdPWithObs_ClassGraph.txt`. En effet, il contient dans une première partie la description textuelle du RdP : il faut donc supprimer cette partie pour ne conserver que le graphe des classes lui-même.

```
while (NonTrouvee && IntervalleNonNul)
{
      Modifier_RdP();
      Construire_GrapheClasses();
      Rechercher_Etat_ERREUR();

      Intervalle = Intervalle/2;
}
```

Figure 3 : Algorithme de recherche de la borne

En fonction de la réponse de cette recherche d'atteignabilité, et des résultats des itérations précédentes de l'algorithme de recherche par dichotomie, l'indice d'itération est modifié c'est-à-dire que l'intervalle de recherche est réduit de moitié, et ceci jusqu'à trouver la borne maximale ou jusqu'à ce que l'intervalle

ai été entièrement testé (dans ce cas la borne maximale n'appartient pas à l'intervalle initial donné en paramètre et n'a pas été trouvée).

VI. VALIDATION PARAMETREE DE STIMAP

Cette section illustre l'utilisation de LPT pour la validation du protocole STIMAP. Le modèle complet du protocole ne sera pas présenté ici, pour des raisons de place et de complexité. De plus, les détails de modélisation du fonctionnement interne des nœuds ne sont pas indispensables à la compréhension de la méthode de validation et de l'utilisation de LPT.

A. Modèle simplifié

Le modèle global est composé de plusieurs nœuds répartis autour du médium de communication. Ces nœuds représentent des nœuds esclaves. Le maître par contre n'est pas modélisé explicitement; seuls sont représentés les instants d'émission et de réception de ses trames. La figure 4 est un modèle très simplifié (le modèle réel comporte plusieurs centaines de places, arcs et transitions) du fonctionnement de STIMAP, avec seulement deux nœuds et le médium de communication. Le modèle du nœud N1 est situé en haut, le nœud N2 en bas (leurs comportements sont symétriques), et le modèle du médium est situé au milieu. Le modèle du médium est composé de deux parties distinctes. Ce modèle simplifié va permettre de comprendre le principe du protocole et de sa validation avec LPT.

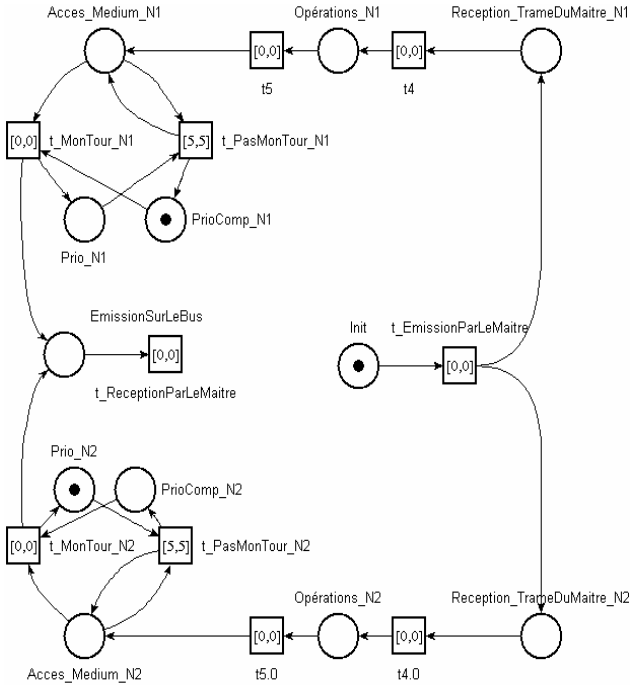

Figure 4 : modèle schématisé pour 2 nœuds

B. Fonctionnement

Dans le mode de communication "par groupe", le maître émet une trame qui permettra de donner la parole à un groupe de nœuds. Cette émission est représentée par le tir de la transition `t_EmissionParLeMaitre`. Puis pour chacun des nœuds esclaves, la place `Reception_TrameDuMaitre_Ni` représente la réception par le nœud Ni de cette trame. On peut remarquer que dans ce modèle, la réception est supposée instantanée : le temps de propagation est négligé, l'intervalle de tir de la transition `t_EmissionParLeMaitre` est nul. Les opérations effectuées par les nœuds sur cette trame ne sont pas détaillées (et elles sont représentées comme instantanées sur le modèle). Par contre, la place `Acces_Medium_Ni` représente l'état du nœud Ni lorsqu'il est prêt à émettre. Dans cet état, le nœud doit alors attendre son tour suivant sa priorité, mécanisme schématisé ici par deux transitions : `t_PasMonTour_Ni` qui signifie que le nœud Ni doit encore attendre, et `t_MonTour_Ni` qui signifie qu'il peut émettre sa trame sur le bus. Le tir de l'une ou l'autre de ces transitions dépend du marquage des places `Prio_Ni` et `PrioComp_Ni` qui représentent les priorités statiques des nœuds. Pour le nœud N1, on suppose qu'il peut immédiatement accéder au médium : la place `PrioComp_N1` est déjà marquée. Par contre pour le nœud N2, il lui faudra attendre une période de temps avant d'émettre. Cette période est un paramètre statique de STIMAP. Ainsi, lorsque N2 arrive dans l'état `Acces_Medium_N2`, la transition `t_MonTour_N2` n'est pas tirable. Par contre, `t_PasMonTour_N2` est sensibilisée : il faut attendre 5 unités de temps pour pouvoir la tirer. Après le tir de cette transition, le marquage des places `Prio_N2` et `PrioComp_N2` est inversé et la transition `t_MonTour_N2` devient tirable. Le nœud N2 peut donc émettre à l'instant 5, c'est-à-dire une période après l'émission du nœud N1. Lors de l'émission, le tir de la transition `t_MonTour_Ni` entraîne le marquage de la place `EmissionSurLeBus`, ce qui représente l'émission de la trame du nœud sur le médium. Cette place est partagée par tous les nœuds, comme le médium l'est; ainsi, si les deux nœuds émettent en même temps, il y aura deux jetons dans cette place. Enfin, la transition `t_ReceptionParLeMaître` représente la réception par le maître de la trame émise par le nœud esclave.

C. Résultats de validation

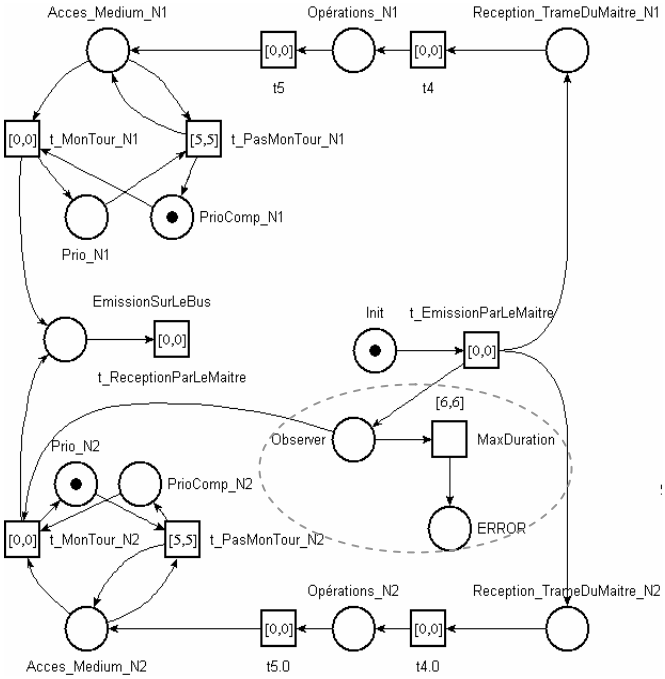

Figure 5 : Observateur rajouté par LPT

LPT recherche alors la borne maximale entre les deux transitions données. Si l'on a par exemple entré en paramètre l'intervalle [0,20], c'est-à-dire que la borne doit se trouver dans cet intervalle, on obtient alors le fichier de résultats donné figure 6. La valeur de la borne maximale est égale à 6 ; en effet le nœud N2 peut émettre à l'instant 5 : la transition `t_MonTour_N2` est tirable. Mais si l'intervalle de la transition `MaxDuration` est égal à 5, la transition `MaxDuration` est également tirable : donc à l'instant 5 l'état `ERROR` est encore

atteignable. La borne supérieure pour laquelle l'état ERROR n'est jamais atteint est donc la valeur 6. Ce fichier résultat contient également les détails de l'algorithme de dichotomie : cinq analyses ont été nécessaires pour trouver la borne à partir de l'intervalle initial [0,20].

```
------ RECHERCHE PAR DICHOTOMIE ------
---------------------------------------
RdP INITIAL : STIMAPSchematise.ndr
ANALYSE ENTRE LES TRANSITIONS
t_EmissionParLeMaitre et t_MonTour_N2
---------------------------------------
---------------------------------------
Valeur de la durée max = 10 =>
L'état ERROR n'est pas atteignable
Valeur de la durée max = 4 =>
L'état ERROR est atteignable
Valeur de la durée max = 7 =>
L'état ERROR n'est pas atteignable
Valeur de la durée max = 5 =>
L'état ERROR est atteignable
Valeur de la durée max = 6 =>
L'état ERROR n'est pas atteignable
---------------------------------------
---------------------------------------
LA VALEUR DE LA BORNE EST : 6
---------------------------------------
---------------------------------------
TAILLE DU GRAPHE : 19 classe(s), 26
transition(s)
TEMPS DE CALCUL : 0.3 seconds
```

Figure 6 : Fichier de résultats

VII. PERFORMANCES

Cette partie propose quelques mesures de performances de l'analyse afin de montrer l'intérêt de l'outil LPT. Elles concernent la validation du protocole STIMAP (cf partie III), pour une architecture comportant 5 nœuds. C'est donc la même analyse que décrite dans la partie précédente, mais sur un modèle réel de STIMAP. La taille du modèle RdPT utilisé pour ces résultats est de 149 places, 146 transitions et 559 arcs. Les résultats concernent la vérification de la propriété Prop1 pour le nœud 1.

A. Performances en place mémoire

La taille du graphe des classes obtenu est de :

- 8661 classes et 35295 transitions pour une analyse sans observateur,
- 8647 classes et 35270 transitions avec l'observateur, si la borne n'est pas atteinte (place erreur jamais marquée),
- 8949 classes et 36060 transitions si la borne est atteinte.

Ces chiffres montrent que dans ce cas, l'ajout de l'observateur n'augmente que très peu la complexité de l'analyse du point de vue utilisation mémoire. Cette augmentation est liée à l'ajout de la composante "quantitative" de la propriété à vérifier. Par contre, lors de l'ajout de la dimension paramétrée, notre méthode n'entraine aucune complexité mémoire supplémentaire. En effet, dans l'algorithme de dichotomie le graphe des classes est recalculé à chaque itération, mais c'est le même graphe à chaque fois, et c'est le même graphe que lors d'une analyse directe avec l'outil Tina. Seul l'espace mémoire nécessaire à la gestion de l'algorithme de dichotomie est ajouté, ce qui est négligeable par rapport à la complexité du graphe des classes.

B. Performances en temps de calcul

Quand à la complexité temporelle, les 3 analyses précédentes ont été effectuées avec l'outil Tina en moins d'1s. Le tableau 1 donne les performances de l'analyse avec l'outil LPT (remarque : la borne ici est de 8). Ce tableau montre que l'augmentation du temps de calcul, même si elle est effective, est loin d'être problématique : la recherche de la borne maximale ne prend que 20.4s pour un intervalle de [0,5000].

Outil	Intervalle	Temps (s)
LPT	[0,5000]	20.4
LPT	[0,1000]	16.9
LPT	[0,500]	14.2
LPT	[0,100]	9.7
LPT	[0,50]	7.5
LPT	[0,20]	5.1
LPT	[7,8]	2.6

Tableau 1 : temps d'analyse avec LPT

Ainsi ces résultats confirment la faible complexité de l'utilisation de l'outil LPT pour la vérification paramétrée de bornes temporelles quantitatives. L'espace mémoire nécessaire pour l'analyse paramétrée est le même que pour une analyse quantitative normale, seul le temps d'analyse augmente mais de façon raisonnable. De plus, les limitations réelles des outils d'analyse par model checking se situent principalement du point de vue de l'espace mémoire. Dans ce type d'analyse, il est nécessaire de construire et de stocker l'ensemble de l'espace des états du système, c'est-à-dire qu'il faut stocker tous les états du graphe d'analyse. C'est donc l'espace mémoire nécessaire à ce stockage qui est l'élément principal de la limitation des processus d'analyse.

VIII. CONCLUSION

Notre approche permet l'introduction d'un paramètre dans le processus d'analyse des contraintes temporelles d'un système, sans en augmenter la complexité. L'analyse présentée permet de vérifier et même d'obtenir la durée maximale d'exécution entre deux évènements sans en connaître la valeur à priori. La technique proposée ne représente pas une avancée du point de vue des techniques d'analyse formelle. Elle offre cependant un grand intérêt d'un point de vue pratique, en permettant l'utilisation d'un paramètre sans entraîner d'augmentation de la complexité de l'analyse d'un point de vue utilisation mémoire. Le but de cet article est de présenter cette technique et de mettre à disposition des chercheurs intéressés le logiciel LPT (LittleParametricTool) développé au LIRMM.

Cet outil est dans son état actuel compatible avec l'outil de modélisation et d'analyse de réseaux de Petri temporels Tina. Il serait évidemment intéressant d'en étendre la portée à d'autres outils d'analyse usuellement utilisés pour la vérification formelle de propriétés, que ce soit pour les réseaux de Petri temporels comme pour les automates temporisés. Une version compatible avec l'outil Roméo est en cours de développement. Cette version pourrait permettre des résultats intéressants étant donné que l'outil Roméo implémente un algorithme de construction du graphe de zones. Cette méthode permet la vérification directe de propriétés quantitatives, sans passer par l'ajout d'un observateur spécifique. Une extension intéressante serait également de

pouvoir vérifier la propriété d'accessibilité à la volée. Cette technique de model checking à la volée permet dans certains cas de ne pas construire la totalité du graphe d'analyse. Cette extension est plus délicate à réaliser car elle implique l'intégration de LPT au sein même du code des outils d'analyse.

La problématique abordée dans cet article peut être considérée dans un contexte plus large de réflexion sur l'accessibilité des techniques de validation pour la conception et l'analyse des systèmes réels. Trop souvent, les possibilités des méthodes théoriques sont inexploitables dans un contexte industriel de part leur complexité, la nouveauté de leur point de vue, le manque d'outils performants ou le manque de communication entre les spécialistes de la validation et les concepteurs de systèmes. Des progrès peuvent être faits dans ce contexte, en essayant d'associer la puissance des méthodes de validation aux contraintes réelles de conception des systèmes et aux caractéristiques de leurs contextes d'utilisation, afin de fournir un ensemble de méthodes et d'outils directement utilisables pour la validation des systèmes réels et industriels.

IX. REFERENCES

[1] K. Godary, D. Andreu and G. Souquet. Sliding Time Interval based MAC Protocol and its Temporal Validation. In 7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems (FET'07), Toulouse, France, November 2007.

[2] D. Guiraud, D. Andreu, G. Cathebras, Y. Bertrand, S. Bernard, J. Galy and J.D. Techer (2004). Dispositif de répartition de courant entre des cathodes d'une électrode multipolaire, notamment d'un implant, Patent Number FR0409351 2004-09-0.

[3] P.M. Merlin, A study of the recoverability of omputing systems. Ph.D. thesis, 1974, Department of Information and Computer Science, University of California, Irvine, CA.

[4] B. Berthomieu, F. Vernadat, Time Petri Nets Analysis with TINA, Proceedings of the 3rd international conference on the Quantitative Evaluation of Systems, pp 123 – 124, 2006, http://www.laas.fr/Tina/

[5] B. Berthomieu and M. Diaz, Modeling and verification of time dependent systems using time Petri nets. IEEE transactions on software engineering, 1991, 17,3 (March), 259-273.

[6] K. Godary, I. Augé-Blum and A. Mignotte. Temporal Bounds for TTA: Validation. In IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES'04). Toulouse, France, August 2004

[7] R. Alur and DR. Dill, A theory of timed automata, Theoretical Computer Science, 126(2):183-235, 1994.

[8] T.A. Henzinger, X. Nicollin, J. Sifakis and S. Yovine, Symbolic model checking for real time systems. Information and Computation, 111(2):193-244, June 1994.

[9] K. G. Larsen, P. Pettersson and W. Yi, UPPAAL in a Nutshell, Journal of Software Tools for Technology Transfer, 1(1/2):134-152, October 1997. www.uppaal.com

[10] R. Alur, TA Henzinger and MY. Vardi. Parametric real-time reasoning. In Proceedings of the 25th Annual Symposium on Theory of Computing, pp 592-601. 1993.

[11] F. Wang and PA. Hsiung. Parametric analysis of computer systems. In Algebraic Methodology and Software Technology, pp 539-553, 1997.

[12] V. Bruyère, E. Dall'Olio and JF. Raskin. Durations,Parametric Model Checking in timed Automata with Presburger Arithmetic. In Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'03), LNCS 2607, Berlin, 2003.

[13] V. Bruyère and JF. Raskin. Real-Time Model-Checking: Parameters Everywhere. In journal Logical Methods in Computer Science, vol. 3(1:7), February 2007.

[14] T. Hune, J. Romijn, M. Stoelinga and F. Vaandrager. Linear parametric model checking of timed automata. Journal of Logic and Algebraic Programming, vol. 52-53, pages 183-220, 2002.

[15] AT Sava, Sur la synthèse de la commande des systèmes à évènements discrets temporisés, PhD thesis, Institut National polytechnique de Grenoble, Grenoble, France, November 2001.

[16] F. Cassez and OH. Roux, Structural translation from Time Petri Nets to Timed Automata, The journal of Systems and Software, 79(10):1456-1468, 2006.

[17] G. Gardey, OH. Roux and O. Roux, A zone-based method for computing the state space of a time Petri net. In Formal Modeling and Analysis of Timed Systems (FORMAT'03), vol 2791 of LNCS, pp 246-259, Marseille, France, September 2003.

[18] H. Boucheneb, H. Rakay, and OH. Roux. Réseaux de Petri à arcs temporels généralisés aux sémantiques faible et forte. In 6ieme Colloque Francophone sur la Modélisation des Systèmes Réactifs, (MSR'07), Lyon, France, October 2007

[19] G. Gardey, D. Lime M. Magnini and OH Roux. Roméo: A Tool for Analyzing Time Petri Nets Analysis. In 17th International Conference on Computer Aided Verification (CAV'05), Lecture Notes in Computer Science, Edinburgh, Scotland, UK, July 2005. Springer.

[20] LPT : www.lirmm.fr/~godary/Tools/LPT/

[21] J. Sifakis - A unified approach for studying the properties of transition systems, Theoretical Computer Science, Vol. 18, 1982

[22] D. Delfieu, M. Sogbohossou, LM. Traonouez and S. Revol. Parameterized study of a Time Petri Net, Cybernetics and Information Technologies, Systems and Applications : CITSA 2007, Orlando, Florida, USA, July 2007.

[23] T. Cormen, C. Leiserson, R. Rivest and C. Stein. Introduction à l'algorithmique. Dunod, Collection Sciences Sup, 2002, 2e édition, ISBN : 9782100039227