

Résumé de la thèse intitulée « Modèles et Algorithmes pour la Vérification des Systèmes Temporisés »

Patricia Bouyer

Soutenue le 5 avril 2002

Cette thèse se situe dans le cadre des études des systèmes temporisés et porte sur plusieurs aspects de leur vérification. Nous y étudions d'une part des classes de modèles pour ces systèmes, dans le but de pouvoir représenter de manière simple et concise de larges classes d'exemples réels. Par ailleurs, nous proposons des algorithmes permettant de vérifier certaines classes de modèles temporisés. Nous proposons aussi une modélisation du protocole de communication PGM (Pragmatic General Multicast), étude de cas réalisée dans le cadre du projet RNRT Calife, et nous vérifions certaines de ses propriétés.

Le modèle autour duquel se greffe cette thèse est le modèle des *automates temporisés*, introduit dans le début des années 1990 par Alur et Dill. Ce modèle est, de nos jours, énormément utilisé pour représenter des systèmes temporisés et sert de base à plusieurs outils de vérification (par exemple UPPAAL et KRONOS), utilisés tant dans le monde universitaire que dans le monde industriel.

Les automates temporisés peuvent être décrits comme des automates finis auxquels ont été rajoutées des variables, appelées *horloges*. Ces horloges évoluent au cours du temps, toutes à la même vitesse. Chaque transition d'un automate temporisé est étiquetée d'une part par l'action qui peut être effectuée, d'autre part par une contrainte d'horloges (*i.e.* une comparaison entre une horloge et une constante ou entre deux horloges) qui doit être vérifiée pour pouvoir franchir la transition, et enfin par un ensemble d'horloges qui devront être remises à zéro. La sémantique classique de ces automates est celle des *mots temporisés*. Ces derniers correspondent à une suite de couples (action,date) qui indiquent chacun quelle action est effectuée et à quelle date.

Une des grosses difficultés intrinsèques à ce modèle est due au fait que les domaines de temps couramment utilisés sont l'ensemble des réels positifs ou l'ensemble des rationnels positifs, donc des ensembles infinis et denses. Malgré cela, ce modèle peut quand même être utilisé pour la vérification, car c'est un modèle *décidable*. Mais la complexité de la procédure de vérification est beaucoup plus importante que dans le cas non temporisé (c'est un problème PSPACE-complet). Les contributions principales de cette thèse s'articulent autour des deux thèmes suivants.

Modèles temporisés. - Nous définissons une extension du modèle original des automates temporisés avec de nouvelles opérations sur les horloges, les *automates temporisés avec mises à jour*. Nous autorisons par exemple l'initialisation des horloges à une valeur quelconque plus grande qu'une constante, ou bien plus petite qu'une constante plus la valeur d'une autre horloge. Notons que l'étude de ce modèle provient d'une modélisation faite par Béatrice Bérard et Laurent Fribourg d'un algorithme de conformité, le protocole ABR. Dans cette modélisation, une mise à jour telle que décrite ci-dessus est utilisée pour modéliser une interaction avec un environnement. Nous décrivons assez finement la frontière entre les sous-classes décidables de notre modèle (c'est-à-dire les sous-classes pour lesquelles le problème du vide, fondamental en vérification, est décidable) et les sous-classes indécidables. Les sous-classes décidables ont été obtenues en restreignant l'ensemble des mises à jour que l'on peut utiliser. Plus précisément, nous associons à chaque automate temporisé avec mises à jour, un système d'inéquations linéaires (qui dépend exclusivement des mises à jour et des contraintes d'horloges utilisées). Ce système a une solution si et seulement si l'automate appartient à une sous-classe décidable. Les résultats obtenus sont parfois surprenants. Par exemple, l'utilisation de mises à jour de la forme $x := x - 1$ rend le modèle indécidable. Par contre, les mises à jour de la forme $x := x + 1$ préservent la décidabilité du modèle à condition que les seules contraintes utilisées soient des comparaisons entre une horloge et une constante. Ce modèle présente donc

une différence majeure avec les automates temporisés d'Alur et Dill : la décidabilité peut dépendre de la nature des contraintes utilisées (comparaisons entre une horloge et une constante ou comparaisons entre deux horloges).

Nous étudions par ailleurs le pouvoir d'expression de ce modèle, en nous attachant plus particulièrement aux classes décidables (les sous-classes indécidables ayant déjà été montrées équivalentes aux machines de Turing). Nous montrons alors que les classes décidables des automates temporisés avec mises à jour ne sont pas plus expressives que les automates temporisés, mais par contre, elles permettent de représenter de manière beaucoup plus compacte de larges classes de systèmes temporisés. Cette concision est souvent un facteur déterminant pour vérifier des systèmes réels en raison de la complexité de la vérification des systèmes temporisés.

La classe des langages formels reconnaissables peut être caractérisée de diverses manières : par les automates finis, les expressions rationnelles, la logique monadique du second ordre, des logiques temporelles, les monoïdes finis... Toutes ces caractérisations ne constituent pas seulement une des bases de l'informatique théorique, mais forment également les fondements de la théorie de la vérification des systèmes non temporisés. Il est ainsi naturel de tenter de bâtir un socle théorique équivalent pour l'étude des systèmes temporisés.

- Parmi les équivalences dans le cas non temporisé évoquées ci-dessus, le théorème de Kleene est fondamental. Il montre que le formalisme des automates finis est équivalent à celui des langages rationnels (*i.e.* les langages obtenus à partir des mots de longueur finie en utilisant les opérations d'union finie, de concaténation et d'itération finie). Cherchant à obtenir un tel théorème dans le cadre temporisé, la première chose à faire est de définir une concaténation sur les mots temporisés. Plusieurs concaténations peuvent être définies de manière naturelle, elles ont été étudiées par Asarin, Caspi et Maler et leur ont permis de proposer un théorème de Kleene. Cependant, celui-ci nécessite l'utilisation des opérations d'intersection et surtout de renommage, rendant ainsi le résultat assez éloigné du cadre non temporisé.

Pour résoudre ce problème et ainsi se passer de cette opération de renommage, nous proposons une nouvelle sémantique pour les automates temporisés, les *langages d'horloges*. Un mot d'horloges contient plus d'informations qu'un mot temporisé. En plus du nom de l'action et de la date à laquelle cette action est effectuée, il contient la valeur de toutes les horloges au moment où l'action est effectuée. Grâce à cette nouvelle sémantique, nous obtenons un théorème "à la Kleene/Büchi", très proche de celui existant dans le cadre des langages formels. Un langage d'horloges est reconnu par un automate temporisé si et seulement s'il est définissable à partir d'un nombre fini d'objets de base en utilisant un nombre fini d'opérateurs d'union, concaténation et itérations finie ou infinie.

- Une des caractérisations les plus élégantes des langages formels est celle, de nature purement algébrique, basée sur les monoïdes finis. Cette caractérisation est à l'origine d'algorithmes simples et surprenants pour la vérification. Dans le cadre temporisé, il n'existait pas, à notre connaissance, de travaux convaincants sur le sujet. La difficulté majeure est qu'il est nécessaire d'inventer un moyen de reconnaissance plus complexe que dans le cadre non temporisé. En effet, si l'on utilise simplement un morphisme dans un monoïde fini, le langage temporisé simple $\{(a, t)(a, t + 1) \mid t > 0\}$ ne serait pas reconnu, ce qui n'est pas très naturel. Pour obtenir une caractérisation intéressante dans le cadre temporisé, nous sommes amenés à généraliser les langages temporisés et à définir les *langages de données*. Un ensemble de données est un ensemble quelconque, fini ou infini, qui peut être un domaine de temps ou toute autre chose. Un mot de données est une suite de couples (action, donnée) alors qu'un langage de données est un ensemble de tels mots de données. Nous proposons un formalisme purement algébrique basé lui aussi sur les monoïdes finis pour reconnaître les langages de données. Ce formalisme utilise, pour effectuer ses calculs dans le monoïde, une mémoire auxiliaire bornée qui lui permet de stocker des données.

En outre, nous définissons un modèle d'*automates de données*, ces automates sont munis de registres (qui servent de mémoire auxiliaire) et peuvent ranger les données lues dans ces registres. Nous montrons alors que ce modèle d'automates est équivalent au formalisme algébrique défini auparavant. Notons que la restriction de nos formalismes à un ensemble de données à un seul élément correspond exactement au cas des langages formels.

Nous montrons aussi que, nous restreignant à un domaine de temps, les automates de données sont plus puissants que les automates temporisés d'Alur et Dill. Nous décrivons aussi une condition sous laquelle le

modèle des automates de données est un modèle décidable. Cette condition est, en général, assez simple à calculer.

Outre ces formalismes algébriques et à base d'automates, nous proposons un langage logique, basé sur la logique monadique du second ordre, qui permet d'exprimer exactement tous les langages acceptés par les automates de données, ce qui renforce l'intérêt de ces langages de données.

Algorithmes et étude de cas. - Nous prolongeons ensuite les travaux faits sur les automates temporisés avec mises à jour. Pour les automates temporisés classiques, l'algorithme basé sur les régions, qui a permis à Alur et Dill de montrer la décidabilité du modèle, n'est pas implémenté dans les outils comme UPPAAL et KRONOS. En effet, il n'y a pas de structure de données bien adaptée à la manipulation des régions. L'algorithme implémenté calcule par une analyse en avant une surapproximation de l'ensemble des états accessibles. Nous avons étendu cet algorithme aux automates temporisés avec mises à jour et avons étudié en détail sa correction, obtenant ainsi comme cas particulier, la correction de l'algorithme implémenté dans UPPAAL et KRONOS (il faut noter que les preuves publiées de cet algorithme sont toutes, à notre connaissance, incomplètes). Nous avons aussi montré comment la structure de données des DBMs, celle utilisée dans UPPAAL et KRONOS, peut être utilisée pour calculer les opérations plus compliquées qui apparaissent dans notre algorithme. Ainsi, il devrait être facile de rajouter le modèle des automates temporisés aux outils existants.

- Les propriétés d'accessibilité sont les propriétés les plus simples à vérifier pour de nombreux systèmes. Il est alors naturel de vouloir transformer, *via* une réduction adéquate, la vérification de propriétés plus générales en la vérification de telles propriétés d'accessibilité. La réduction que nous étudions est la suivante. Pour chaque propriété ϕ , nous construisons un "observateur" appelé *automate de test* et que nous notons T_ϕ . Cet automate vérifie alors que si A est un automate temporisé,

$$A \text{ vérifie } \phi \iff A \parallel T_\phi \text{ ne permet pas d'atteindre un état rejetant de } T_\phi$$

Ainsi, pour tester si A vérifie ϕ , il suffit de pouvoir tester l'accessibilité de certains états du système formé de la mise en parallèle de l'automate A avec l'observateur T_ϕ . Nous définissons un fragment d'un langage proche d'une logique modale temporisée et nous proposons un algorithme qui permet de construire, de manière syntaxique, pour chaque formule ϕ de notre langage, un automate de test T_ϕ comme décrit ci-dessus. Nous montrons alors que ce fragment de langage logique caractérise complètement l'ensemble des propriétés dont la vérification peut se réduire de la manière décrite ci-dessus à un test d'accessibilité.

- Nous proposons ensuite une modélisation du protocole de communication de France Télécom R&D appelé PGM (*Pragmatic General Multicast*). C'est un protocole multicast de transmission de données sur un réseau non fiable. Les sources envoient des données dans le réseau aux différents destinataires. Le réseau pouvant perdre des données, certains destinataires ne les reçoivent pas toutes. Ils peuvent s'en rendre compte car, régulièrement, la source envoie sur le réseau des paquets indiquant quels sont les noms des données qui ont été envoyées. Lorsqu'un destinataire se rend compte qu'il n'a pas une donnée qu'il aurait dû recevoir, il envoie un message d'erreur à la source. Lorsque la source reçoit un de ces messages d'erreur, elle peut envoyer une réparation si elle a encore cette donnée en mémoire. Le but est alors d'étudier la correction de ce protocole, à savoir, est-ce que chaque destinataire reçoit chaque donnée ? Si ce n'est pas le cas, la source se rend-elle toujours compte qu'une donnée a été définitivement perdue ?

Le protocole que nous venons de décrire de manière très simplifiée, est en fait donné par une spécification de 115 pages. Nous proposons une modélisation de ce protocole et nous décrivons son implémentation dans l'outil de vérification UPPAAL. Nous vérifions ensuite certaines propriétés du protocole en utilisant le module de vérification d'UPPAAL. Cette étude nous permet de mettre à jour deux problèmes dans ce protocole.

Dans cette thèse, nous nous sommes donc intéressés à différents aspects de la vérification des systèmes temporisés, de la définition et l'étude de modèles pour représenter les systèmes temporisés à une étude de cas réel en passant par l'étude d'algorithmes et de structures de données adaptés aux systèmes temporisés.