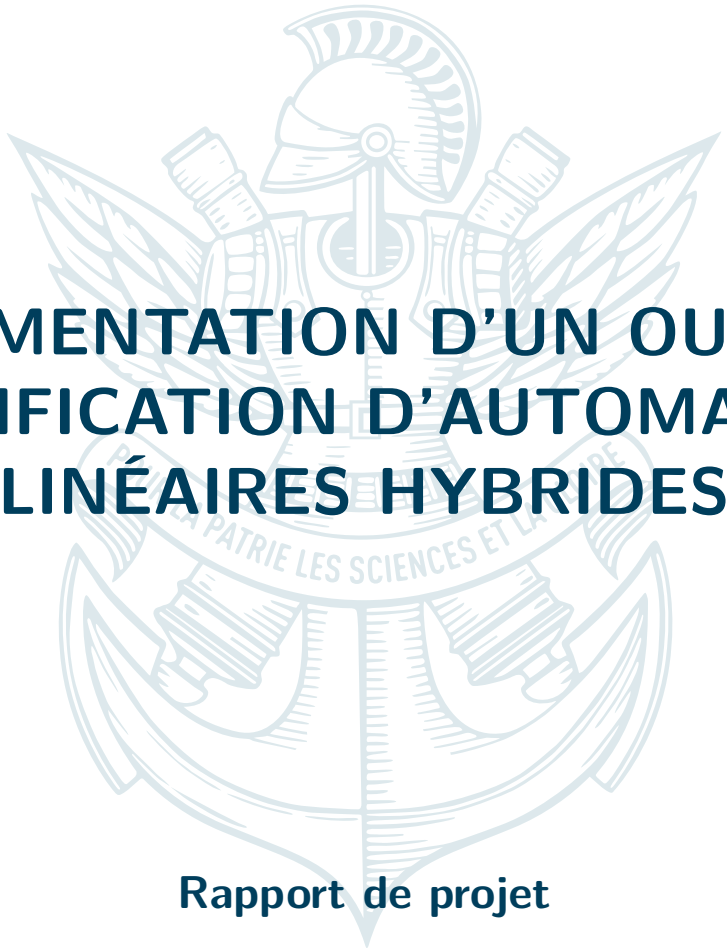

IMPLEMENTATION D'UN OUTIL DE VÉRIFICATION D'AUTOMATES LINÉAIRES HYBRIDES

Rapport de projet

5/02/18

Baptiste Pecatte & Julien Piet

Sujet et motivation Nous avons choisi d'écrire un vérifieur d'automate hybride linéaire en python. La vérification des automates hybrides reste aujourd'hui un problème dur, car il est difficile en informatique de mélanger les objets discrets, comme les transitions, et les phénomènes continus, décrits par des équations différentielles. Nous trouvons interessant l'idée d'implémenter un outil de vérification, parce que cela peut nous permettre de mieux comprendre le fonctionnement des vérifieurs existants. Nous voulions aussi avoir une approche indépendant des différentes représentations classiques des automates (UPAAL, SpaceEx, Simulink...), pour éviter de devoir intégrer notre algorithme à un logiciel existant. Nous avons choisi de représenter les automates sous forme d'objets JSON, de manière très simple, pour permettre à l'utilisateur de rapidement appréhender notre vérifieur.

Réalisation Notre algorithme peut determiner les états atteignables d'un automate hybride linéaire à partir de sa description complete sous forme de JSON, de son état initial et des plages de valeurs initiales de ses variables. Les équations différentielles considérées sont de la forme :

$$\dot{X} = AX + U(t) \tag{1}$$

$$X \in \mathbb{R}^n\,;\; A \in \mathbb{R}^{n\times n}\,;\; B \in \mathbb{R}^n\,;\; U(t) \text{ borné}\,;$$

Les gardes seront de la forme :

$$CX + Y = 0 \tag{2}$$

$$X \in \mathbb{R}^n\,;\; C \in \mathbb{R}^{n\times n}\,;\; Y \in \mathbb{R}^n\,;$$

Enfin, les mises à jour des valeurs des variables seront de la forme :

$$X = DX + W \tag{3}$$

$$X \in \mathbb{R}^n\,;\; D \in \mathbb{R}^{n\times n}\,;\; W \in \mathbb{R}^n\,;$$

Cette résolution est faite à l'aide d'une résolution approchée de l'équation différentielle par la méthode vue en cours des zonotopes. La sortie du programme est un diagramme représentant pour chaque noeud les états atteignables.

Pour le cas particulier de la dimension 1, nous avons développé un module basé sur des polygones convexes. Plus précise, cette méthode à le défaut de n'être compatible d'avec la dimension 1, car les calculs d'intersection et la représentativité des ensembles en dimension supérieure sont trop complexes. Elle se base sur des equations de la forme :

$$\dot{X} = AX + B + U(t) \tag{4}$$

$$X \in \mathbb{R}\,;\; A \in \mathbb{R}\,;\; B \in \mathbb{R}\,;\; U(t) \text{ borné}\,;$$

Les mises à jour et les gardes peuvent incorporer des composantes temporelles, et les gardes basées sur les inéquations ('<', '>') sont admises. La sortie du programme est semblable à celle des zonotopes N-dimensionnels : On y voit l'évolution temporelle pour chaque noeud de ses valeurs atteignables.

Progression du travail

Nous avons commencé par une implémentation d'un solver à une dimension basé sur la méthode des hypercubes (encadrement des valeurs permises par des hypercubes). Si cette méthode était rapide à implémenter, son efficacité n'était pas suffisante. L'étape suivante a été de remplacer ces carrés dans le plan par des quadrilatères, qui permettaient une sur-approximation minimale grâce à un calcul de la tangente à la courbe : Pour relier deux états atteignables à des instants t et $t + d$, il fallait calculer l'equation de la tangente à la courbe pour englober juste l'essentiel.

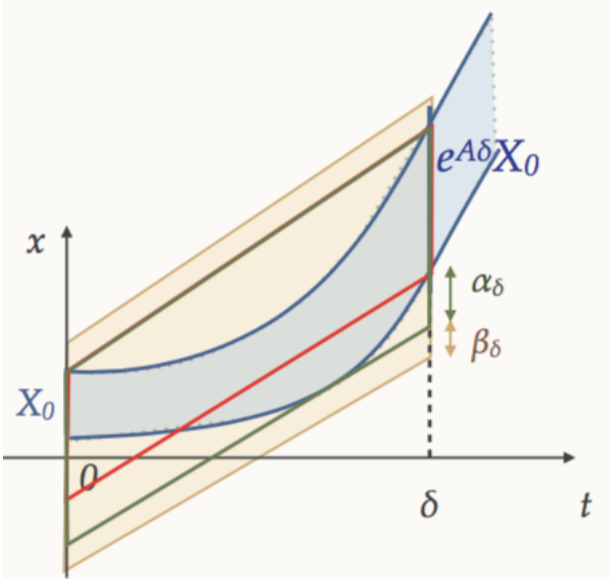

Figure 1 – Calcul de tangente, comme vu en cours

Nous avons introduit le temps en tant de variable dans ce calcul, grâce à la manipulation de matrices 2×2, afin de pouvoir manipuler des équations temporelles. Ensuite, nous avons introduit les notions de gardes grâce à un algorithme d'intersection entre la garde, un hyperplan affine, et chaque polygone. L'object généré étant un polygone, il est alors possible de réitérer l'algorithme de résolution d'équation sur un autre noeud pour continuer les approximations d'atteignabilité. Voici le rendu final de cette méthode, sur un automate à deux états :

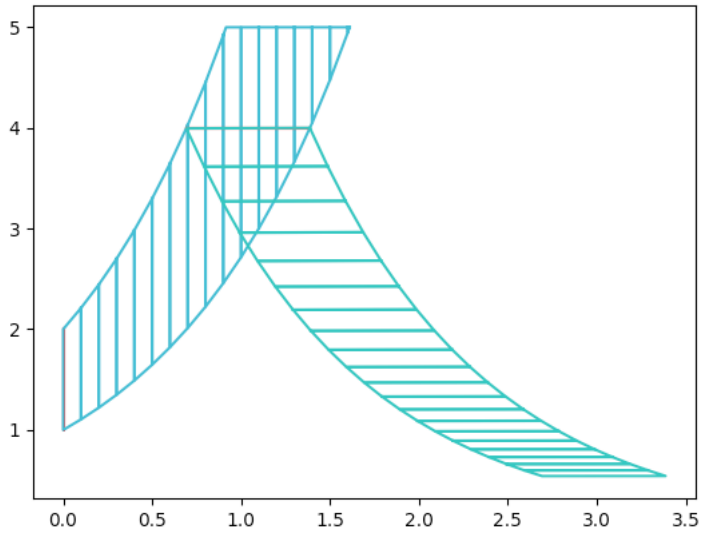

Figure 2 – Sortie de l'algorithme pour la dimension 1

Ensuite, nous avons voulu développer un algorithme général en N-Dimensions à l'aide de la représentation matricielle d'un polyèdre convexe général. Malheureusement, après implémentation de cette classe, nous nous sommes rendus comptes de la difficulté de calculer l'intersection de deux polyèdres convexes, ce qui a rendu cette méthode inutilisable. Néanmoins, nous avons codé une méthode de résolution à l'aide d'hypercubes, qui même si très inefficace, fonctionnait. Le souci d'une telle méthode est qu'elle n'est difficilement scalable à des dimensions supérieures à 1, car l'application de la matrice de transformation d'un état t à $t+d$ peut engendrer des rotations, voire toutes les applications orthogonales du groupe $O(n)$. Les hypercubes ne s'adaptent bien qu'aux translations et dilatations, mais englobent de manière bien trop approximative les transformations plus subtiles.

Nous nous sommes donc tournés vers un papier sur les zonotopes [1] pour implémenter cette méthode en dimension N. Les zonotopes sont une classe de polygone ayant des propriétés particulières. Moins précises que les enveloppes convexes quand il s'agit d'approximer un ensemble quelconque, elles ont l'avantage d'être très facile à stocker, et d'avoir des opérations faciles à calculer (somme de Minkowski, et application linéaire). En utilisant la somme de Minkowski, on peut obtenir simplement une approximation des valeurs prises par l'équation entre 0 et t, puis on peut obtenir en temps linéaire et en espace constant les valeurs prises par l'équation pour tout les intervals successifs $[nt; n(t+1)]$.

Une fois que nous sommes capables d'obtenir une sur-estimation de l'ensemble des valeurs prises par l'équation différentielles, pour un zonotope donné initial, il nous faut alors déterminer quand les gardes sont prises. Il n'est pas facile de calculer l'intersection entre un hyperplan et un zonotope. En revanche, il est facile de déterminer si l'intersection est non-nulle. On peut alors sur-approximer l'intersection d'un zonotope et d'un hyperplan au zonotope lui-même. De cette façon, nous obtenons une méthode d'analyse n-dimensionnelle, beaucoup plus précise que les hypercubes, mais ayant un coût de calcul relativement faible.

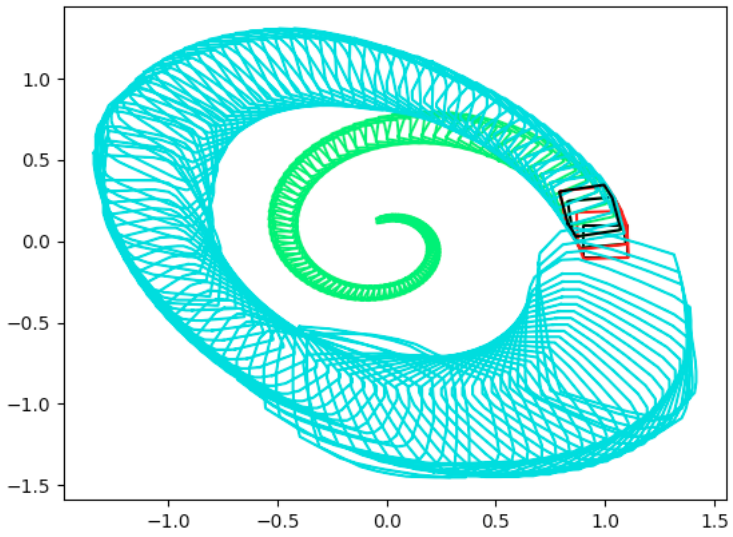

Figure 3 – Sortie de l'algorithme utilisant les zonotopes

Enfin, nous avons créé un parseur pour automatiquement générer des automates à partir de leur description textuelle, et une interface générique pour représenter chaque classe, comme le solveur, les noeuds, les gardes, ou encore l'algorithme d'itération. Cela nous permet d'implémenter à la fois la version complète N-dimensionnelle ainsi que la version 1-dimensionnelle en manipulant des classes ayant le même prototype.

Répartition du travail Nous avons travaillé à deux sur ce projet. Baptiste s'est occupé du parseur, et de l'implémentation de la méthode des zonotopes. Julien s'est occupé de la résolution à une dimension, et de l'approximation des résolutions N-dimensionnelles avec des hypercubes. Enfin, l'interface de résolution générale est le fruit d'une reflexion partagée, et a été codé par Baptiste.

Conclusion Si sur le papier le calcul des états atteignables semble simple, nous avons découvert que sa traduction en informatique est difficile en raison des problèmes de représentation. Même si les calculs sont assez simples, il est difficile de trouver une représentation numérique d'un ensemble continu N-dimensionnel qui soit précise, suffisamment complexe pour pouvoir d'adapter à tous les problèmes, et qui gère facilement les intersections et mises à jour. C'est un domaine qui n'est pas encore très documenté, et il reste beaucoup à y découvrir. Des méthodes plus précises existent (utilisation de polytopes et de fonctions supports [3], mélange polytope/zonotope [2]), mais nous n'avons pas pu les implémenter faute de temps.

RÉFÉRENCES

[1] Antoine Girard
Reachability of uncertain linear systems using zonotopes.
http://www-ljk.imag.fr/membres/Antoine.Girard/Publications/hscc2005.pdf

[2] Matthias Althoff, Olaf Stursberg and Martin Buss
Computing Reachable Sets of Hybrid Systems Using a Combination of Zonotopes and Polytopes.
https://mediatum.ub.tum.de/doc/1287515/893367.pdf

[3] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, Oded Maler
SpaceEx : Scalable Verification of Hybrid Systems.
http://spaceex.imag.fr/sites/default/files/paper_55.pdf