Computation of barrier certificates for dynamical hybrids systems using interval analysis

Adel Djaballah

NNT : 2017SACLS195

Thèse de doctorat de l'Université Paris-Saclay préparée à l'Université Paris-Sud

Ecole doctorale n°580
Sciences et technologies de l'information et de la communication
Spécialité de doctorat: Automatique

par

M. Adel Djaballah

Calcul par analyse intervalle de certificats de barrière pour les systèmes dynamiques hybrides

Thèse présentée et soutenue à Palaiseau, le 03 Juillet 2017.

Composition du Jury :

M.	Antoine Girard	Directeur de recherche CNRS/L2S	(Président du jury)
M.	Nacim Ramdani	Professeur Université d'Orléans	(Rapporteur)
M.	Stefan Ratschan	Professeur associé Académie tchéque des sciences	(Rapporteur)
M.	Nacim Meslem	Maître de conférences Grenoble INP	(Examinateur)
M.	Michel Kieffer	Professeur des universités Université Paris-Sud	(Directeur de thèse)
M.	Alexandre Chapoutot	Enseignant-Chercheur ENSTA ParisTech	(Co-encadrant)

Contents

Notations

- v A scalar value.
- $\mathbf{v}$ A vector value.
- $[v]$ An interval value.
- $[\mathbf{v}]$ An interval vector value or a box.
- v_i The ith element of the vector $\mathbf{v}$.
- $[v_i]$ The ith interval of the box $[\mathbf{v}]$.
- $[\mathbf{v}]_1$, $[\mathbf{v}]_2$, ..., $[\mathbf{v}]_n$ A sequence of n interval vectors.
- $[v_i]_j$ The ith element of the jth interval vector.
- $\mathbb{R}$ The set of real numbers.
- $\mathbb{IR}$ The set of closed intervals over $\mathbb{R}$.
- $\mathcal{X}$ A set in $\mathbb{R}^n$.
- $\dot{x}$ The derivative of a function x with respect to time t.
- $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ Stands for the partial derivative of a function f with the respect of $\mathbf{x}$.
- ∇F The divergence of a differentiable vector field F.
- $\mathcal{P}(\mathcal{X})$ Stands for the power set of the set $\mathcal{X}$.
- $\langle ., . \rangle$ The dot product in $\mathbb{R}^n$.
- A^T The transpose of a matrix A.
- $[\mathbf{v}] \setminus [\mathbf{u}]$ The interval hull difference between $[\mathbf{v}]$ and $[\mathbf{u}]$.

Chapter 1

Résumé étendu

1.1 Introduction

Les besoins croissants en matière de sécurité, de protection de l'environnement et de confort en conduit à l'adoption de systèmes de contrôle automatique dans de nombreux produits industriels. Ainsi, les *smart systems* sont actuellement en développement important dans de nombreux domaines dont le transport et l'énergie. Ce type de système a pour particularité de répondre et de s'adapter automatiquement à son environnement.

L'automatisation des systèmes entraîne l'utilisation de composants électroniques contrôlés par des programmes informatiques qui deviennent de plus en plus complexes. Par ailleurs, les fonctions sont de plus en plus interconnectées. Ainsi, dans une voiture, le contrôleur de vitesse est couplé à un détecteur de collisions, qui interagit avec un système de détection de blocage des roues.

La contrepartie de cette automatisation est l'utilisation de programmes informatiques pour des tâches critiques, dont l'échec peut avoir des conséquences catastrophiques. Par exemple, le régulateur de vitesse d'une voiture a un comportement assez simple, le conducteur configure une vitesse souhaitée et le programme informatique tente de maintenir cette vitesse en contrôlant l'accélération et le freinage de la voiture. Cependant, une panne de ce programme peut entraîner un accident pouvant conduire au décès de l'automobiliste. La conception d'un tel programme qui assure un comportement sûr est difficile.

Les programmes informatiques mentionnés ci-dessus font partie de la famille de systèmes de contrôle-commande, qui permettent d'amener un système à un état souhaité. La caractéristique de ces programmes est d'interagir continuellement avec l'environnement physique, par exemple, la vitesse d'une voiture dans le cas d'un régulateur de vitesse. Cette interaction relie deux mondes qui évoluent différemment avec le temps. Le premier est l'environnement physique qui évolue de manière continue avec le temps et le deuxième est celui des programmes informatiques qui évoluent à des instants discrets. Le terme *hybride* est utilisé pour décrire l'évolution de systèmes impliquant des programmes informatique qui prennent

en considération l'environnement physique.

Pour revenir à l'exemple du système de contrôle de vitesse, l'exécution d'un tel programme est définie selon les étapes suivantes: le programme lit la vitesse actuelle de la voiture (grâce aux capteurs), prend la décision d'accélérer ou de ralentir, et envoie des commandes aux actionneurs. Ces trois étapes sont répétées à une période constante, par exemple toutes les 10ms. La conséquence de la décision d'accélérer ou ralentir entraîne une modification de la vitesse de la voiture qui peut éventuellement influencer le comportement du programme d'une manière difficilement prédictible. Le défi de la mise en œuvre de ces programmes est de prendre en compte leurs interactions avec l'environnement physique.

Les méthodes de vérification formelle peuvent être utilisées pour accroître la confiance dans les opérations effectuées par un programme interagissant avec un système physique. Ceci est obtenu en fournissant une preuve mathématique que leur comportement est satisfaisant. Elles conduisent au développement d'outils automatisés, qui vérifie le comportement des programmes informatiques. L'automatisation de la vérification permet de gérer la complexité croissante des programmes et répond ainsi aux contraintes économiques. Dans le cas de systèmes hybrides, le défi consiste à définir des outils de vérification formelle pour inclure le monde physique.

Cette thèse vise à combler en partie l'écart entre les besoins de vérification des systèmes hybrides et les méthodes formelles. Elle introduit des algorithmes permettant de prouver la sûreté de systèmes décrits par des modèles dynamiques hybrides, comportant des parties évoluant à temps continu et des parties évoluant à des instants discrets.

Ce résumé débute par un bref état de l'art présenté au paragraphe 1.2. Le paragraphe 1.3 présente une méthode de synthèse de barrières paramétriques permettant de prouver la sûreté de systèmes décrits par des modèles dynamiques à temps continu. Le paragraphe 1.4 étend ces résultats au cas de systèmes décrit par des modèles hybrides. Enfin, le paragraphe 1.5 conclut ce résumé.

1.2 Bref état de l'art

Que le système considéré soit décrit par un modèle dynamique à temps continu ou un modèle hybride, nous supposerons qu'une partie de l'espace d'état est dangereuse. Un système est dit sûr lorsque son état, lorsqu'il évolue au cours du temps, reste éloigné de la partie dangereuse de l'espace d'état. Cette propriété devra être vérifiée quelle que soit la valeur initiale de l'état appartenant à un ensemble de valeurs initiales admissibles et quelles que soient les perturbations, à conditions qu'elles restent également dans un ensemble admissible de perturbations.

La vérification des propriétés de sûreté pour les systèmes dynamiques est un domaine de recherche actif depuis plusieurs décennies. Pour prouver la sûreté d'un système hybride,

différentes approches ont été proposées. L'une de ces approches consiste à calculer explicitement une approximation extérieure du sous-espace atteignable de l'espace d'état partant d'une valeur quelconque admissible de l'état initial et cela pour toutes les perturbations admissibles. Si cette approximation extérieure n'intersecte pas la partie dangereuse de l'espace d'état, il est possible de déduire que le système est sûr.

Une autre manière de prouver la sûreté d'un système est de trouver un invariant, c'est-à-dire une partie de l'espace d'état dans laquelle l'évolution de l'état du système reste cantonnée au cours du temps. Les invariants sont très utiles pour prouver la sûreté des systèmes dynamiques car si l'intersection de l'invariant et de la partie dangereuse de l'espace d'état est vide, on peut également déduire que le système est sûr.

1.2.1 Caractérisation du sous-espace d'états atteignables

Si l'on considère des modèles dynamiques de la forme

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)), \tag{1.1}$$

où $\mathbf{x}(t)$ est l'état à l'instant t du système, le calcul de l'évolution au cours du temps du système est principalement obtenue à l'aide d'outils d'*intégration numérique garantie* en enfermant l'ensemble des valeurs possible de l'état dans des pavés ou vecteurs d'intervalles [NJC99, RA11, AdSC16].

Si l'on considère des systèmes décrits par des modèles dynamiques hybrides, différentes classes de modèles ont été considérées, ce qui a conduit à différentes techniques de calcul de l'ensemble d'états atteignables $\mathcal{R}$. Dans [ABDM00, Tiw03, GLGM06] $\mathcal{R}$ est évalué pour des modèles linéaires hybrides sur un horizon de temps fini en utilisant des ensembles tels que des polyèdres. Dans [Tiw03], cette approche est raffinée par une meilleure prise en compte des propriétés du système dynamique hybride. L'ensemble d'états atteignables est calculé dans [SSM04] pour des systèmes non-linéaires en utilisant une abstraction (approximation) à l'aide de modèles linéaires exprimés dans un nouveau système de coordonnées. Le problème d'atteignabilité pour un système non-linéaire est formulé comme un problème d'optimisation dans [CK03]. Un opérateur de Picard-Lindelöf est combiné dans [CAS12] avec des modèles de Taylor pour trouver l'ensemble d'états atteignables d'un modèle dynamique hybride.

De nombreux outils existent pour prouver la sûreté de systèmes hybrides en utilisant la caractérisation d'ensembles atteignables. Ainsi, HyTech [HHWT97], **d/dt** [ABDM00], ou SpaceEx [FLGD$^+$11] sont bien adaptés à des modèles linéaires hybrides.

1.2.2 Caractérisation d'invariant

Un ensemble de méthodes de caractérisation d'invariants a été proposé, par exemple pour des modèles avec une dynamique linéaire ou affine dans [Tiw03] ou pour des modèles dont

la dynamique est polynomiale dans [SSM04, GT08, KDSA14, YLW15].

Une méthode permettant de calculer des invariants pour des modèles dynamiques hybrides a été proposée dans [TRSS01]. L'approche proposée construit des ensembles qui sont des points fixes d'opérateurs de raffinement et d'élargissement en utilisant également des outils d'élimination de quantificateurs. Ce type d'approche peut générer des contraintes complexes et passe difficilement à l'échelle. Hsolver, proposé par [RS07] propose une approche réalisant une abstraction du modèle hybride non-linéaire à l'aide de représentation discrètes et des techniques de raffinement à l'aide de techniques de propagation de contraintes.

Une approcha alternative consiste à utiliser des fonctions de Lyapunov [GTV85]. Ainsi, [Par03] considère des fonctions de Lyapunov candidates paramétrées pour des systèmes décrits par des modèles de dynamique polynomiale formés par des polynômes de type *Sum of Squares* (SoS) et utilise des outils de programmation semi-définie (SdP) pour la synthèse des paramètres des fonctions candidates.

1.2.3 Caractérisation de barrières

Dans ce type d'approche, une fonction *barrière* est utilisée afin de définir une hypersurface dans l'espace d'état séparant l'ensemble atteignable et les parties dangereuses de l'espace d'état. Si cette hypersurface n'est traversée par aucune trajectoire de l'état, le système dynamique est sûr, voir [PJ04, Pra05, PR05, SPW12a, DGXZ17]. La principale difficulté de ce type d'approche est la synthèse de bonnes fonctions barrières.

La fonction paramétrique décrivant la barrière doit satisfaire un certain nombre de contraintes impliquant la dynamique du modèle, l'ensemble des états initiaux possibles, et l'ensemble dangereux. Ces contraintes ne sont pas convexes en général, ce qui complique la recherche de fonctions barrières satisfaisantes. Précédemment, seules des fonctions barrières polynomiales ont été considérées pour des modèle dynamiques polynomiaux.

Cette thèse considère des systèmes dynamiques relativement généraux avec des barrières paramétriques quelconques. Les solutions présentées exploitent des outils de satisfaction de contraintes sur des domaines continus et des outils issus de l'analyse par intervalles.

1.3 Sûreté des systèmes dynamiques à temps continu

Ce paragraphe résume la première contribution de cette thèse et décrit une approche permettant de concevoir des fonctions barrières afin de prouver la sûreté de systèmes décrits par un modèle dynamique à temps continu.

Considérons le système dynamique autonome perturbé à temps continu suivant

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{d}), \tag{1.2}$$

où $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ est le vecteur d'état et $\mathbf{d} \in \mathcal{D}$ est une perturbation constante et bornée. L'ensemble des états initiaux possibles à $t = 0$ est noté $\mathcal{X}_0 \subset \mathcal{X}$. Il existe un sous-ensemble dangereux $\mathcal{X}_u \subseteq \mathcal{X}$ qui ne doit pas être atteint par le système, pour tout $\mathbf{x}_0 \in \mathcal{X}_0$ et tout $\mathbf{d} \in \mathcal{D}$.

Le théorème 1 proposé par [PJ04] indique un certain nombre de conditions suffisantes pour qu'une fonction paramétrée soit une barrière valide.

Theorem 1. *Considérons le système dynamique* (1.2) *et les ensembles* $\mathcal{X}$, $\mathcal{D}$, $\mathcal{X}_0$ *et* $\mathcal{X}_u$. *S'il existe une fonction* $B : \mathcal{X} \to \mathbb{R}$ *telle que*

$$\forall \mathbf{x} \in \mathcal{X}_0, \quad B(\mathbf{x}) \leqslant 0, \tag{1.3}$$

$$\forall \mathbf{x} \in \mathcal{X}_u, \quad B(\mathbf{x}) > 0, \tag{1.4}$$

$\forall \mathbf{x} \in \mathcal{X}$, $\forall \mathbf{d} \in \mathcal{D}$,

$$B(\mathbf{x}) = 0 \implies \left\langle \frac{\partial B(\mathbf{x})}{\partial \mathbf{x}}, f(\mathbf{x}, \mathbf{d}) \right\rangle < 0, \tag{1.5}$$

alors le système décrit par le modèle (1.2) *est sûr.*

La recherche d'une barrière B est difficile car elle s'effectue dans un espace de fonctions.

1.3.1 Recherche de barrières paramétrées

L'idée de [PJ04] est de rechercher des barrières au sein d'une famille de fonctions paramétriques $B(\mathbf{x}, \mathbf{p})$ qui dépendent d'un vecteur $\mathbf{p} \in \mathcal{P} \subseteq \mathbb{R}^m$. Il suffit de trouver un vecteur $\mathbf{p}$ tel que $B(\mathbf{x}, \mathbf{p})$ satisfait (1.3)-(1.5) pour prouver la sûreté du système.

Pour faciliter la recherche, nous allons supposer en outre qu'il existe des fonctions $g_0 : \mathcal{X} \to \mathbb{R}$ et $g_u : \mathcal{X} \to \mathbb{R}$ telles que

$$\mathcal{X}_0 = \{\mathbf{x} \in \mathcal{X} \,|\, g_0(\mathbf{x}) \leqslant 0\} \tag{1.6}$$

et

$$\mathcal{X}_u = \{\mathbf{x} \in \mathcal{X} \,|\, g_u(\mathbf{x}) \leqslant 0\}. \tag{1.7}$$

Le théorème 1 peut alors être reformulé comme suit

Proposition 1. *Si* $\exists \mathbf{p} \in \mathcal{P}$ *telle que* $\forall \mathbf{x} \in \mathcal{X}$, $\forall \mathbf{d} \in \mathcal{D}$

$$\xi(\mathbf{x}, \mathbf{p}, \mathbf{d}) = (g_0(\mathbf{x}) > 0 \vee B(\mathbf{x}, \mathbf{p}) \leqslant 0) \tag{1.8}$$

$$\wedge\, (g_u(\mathbf{x}) > 0 \vee B(\mathbf{x}, \mathbf{p}) > 0) \tag{1.9}$$

$$\wedge \left(B(\mathbf{x}, \mathbf{p}) \neq 0 \vee \left\langle \frac{\partial B}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{p}), f(\mathbf{x}, \mathbf{d}) \right\rangle < 0 \right) \tag{1.10}$$

alors le système décrit par le modèle (1.2) *est sûr.*

1.3.2 Algorithme de résolution

Pour trouver une fonction barrière valide, il faut trouver un vecteur $\mathbf{p} \in \mathcal{P}$ tel que $B(\mathbf{x}, \mathbf{p})$ satisfait les conditions du théorème 1. Pour cela, l'algorithme *Computable Sufficient Conditions-Feasible Point Searcher* (CSC-FPS) [JW96] a été adapté.

On suppose pour simplifier la présentation que $\mathcal{X}$, $\mathcal{D}$, et $\mathcal{P}$ sont des pavés, *i.e.*, $\mathcal{X} = [\mathbf{x}]$, $\mathcal{D} = [\mathbf{d}]$, et $\mathcal{P} = [\mathbf{p}]$.

FPS explore le pavé de paramètres $[\mathbf{p}]$ en examinant et éliminant éventuellement des sous-pavés de $[\mathbf{p}]$. La manière dont se fait le tri est déterminée par les résultats donnés par l'algorithme CSC. Pour un pavé donné $[\mathbf{p}]_0 \subseteq [\mathbf{p}]$, CSC renvoie `Vrai` quand CSC parvient à prouver que (1.8)-(1.10) sont satisfaits pour un vecteur $\mathbf{p} \in [\mathbf{p}]_0$. CSC renvoie `Faux` quand il réussit à montrer qu'il n'existe pas de $\mathbf{p} \in [\mathbf{p}]_0$ pour lequel (1.8)-(1.10) peuvent être satisfaits. CSC renvoie `Inconnue` dans les autre cas. A chaque itération FPS tente de réduire le pavé de recherche courant dans l'espace des paramètres en utilisant des opérateurs de contraction, qui éliminent tous les paramètres qui ne satisfont pas les contraintes (1.8)-(1.10).

La figure 1.1 décrit l'algorithme FPS. La figure 1.2 donne un exemple d'algorithme CSC adapté à une contrainte spécifique. Une algorithme CSC soit être défini pour chacune des contraintes (1.8)-(1.10).

Figure 1.1: Algorithme FPS

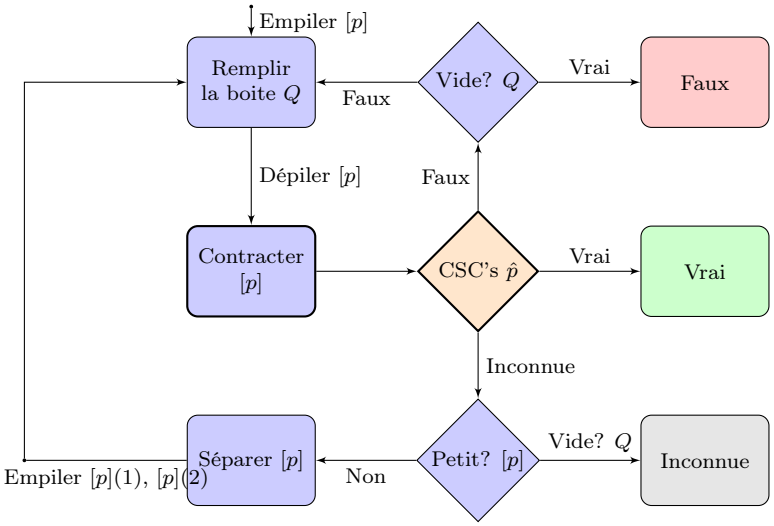
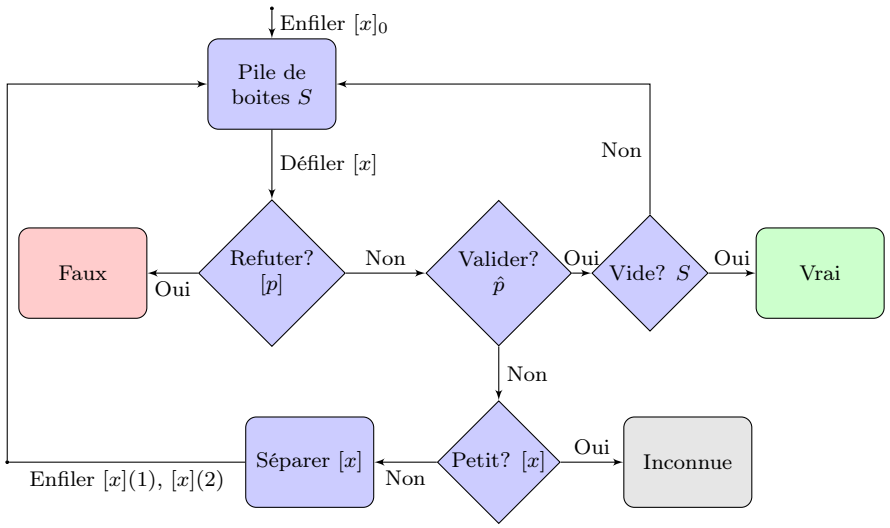


Figure 1.2: Algorithme CSC



1.3.3 Exemples

FPS et CSC ont été implantés à l'aide de la bibliothèque IBEX [ATNC12, CJ09]. La sélection du type de barrière s'effectue en considérant des polynômes de degrés croissants. Dans ce résumé, seule une partie des exemples est décrite. Pour plus de détails, voir le chapitre 4.

Le tableau 1.1 synthétise l'ensemble des résultats obtenus avec l'approche proposée et compare ces résultats avec ceux obtenus à l'aide de dReal [GKC13].

Example 1. *Considérons le système suivant*

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 \\ x_1 x_2 - 0.5 x_2^2 \end{pmatrix}$$

avec $g_0(\mathbf{x}) = (x_1 + 1.25)^2 + (x_2 - 1.25)^2 - 0.05$, $g_u(\mathbf{x}) = (x_1 + 2.5)^2 + (x_2 - 0.8)^2 - 0.05$, *et* $[\mathbf{x}] = [-10^3, 0] \times [-10^3, 10^3]$. *La fonction barrière paramétrique est définie par* $B(\mathbf{x}, \mathbf{p}) = \frac{p_1 p_2 (x_0 + p_3)}{(x_0 + p_3)^2 + p_2^2} + x_1 + p_4$.

Example 2. *Considérons le système de Lorenz décrit dans [VČ96]*

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{pmatrix} = \begin{pmatrix} 10(x_2 - x_1) \\ x_1(28 - x_3) - x_2 \\ x_1 x_2 - \frac{8}{3} x_3 \end{pmatrix}$$

avec $g_0(\mathbf{x}) = (x_1 + 14.5)^2 + (x_2 + 14.5)^2 + (x_3 - 12.5)^2 - 0.25$, $g_u(\mathbf{x}) = (x_1 + 16.5)^2 + (x_2 + 14.5)^2 + (x_3 - 2.5)^2 - 0.25$, *et* $[\mathbf{x}] = [-20, 20] \times [-20, 0] \times [-20, 20]$. *La fonction barrière paramétrique est définie par* $B(\mathbf{x}, \mathbf{p}) = p_1 x_1^2 + p_2 x_1 + p_3 x_3 + p_4$.

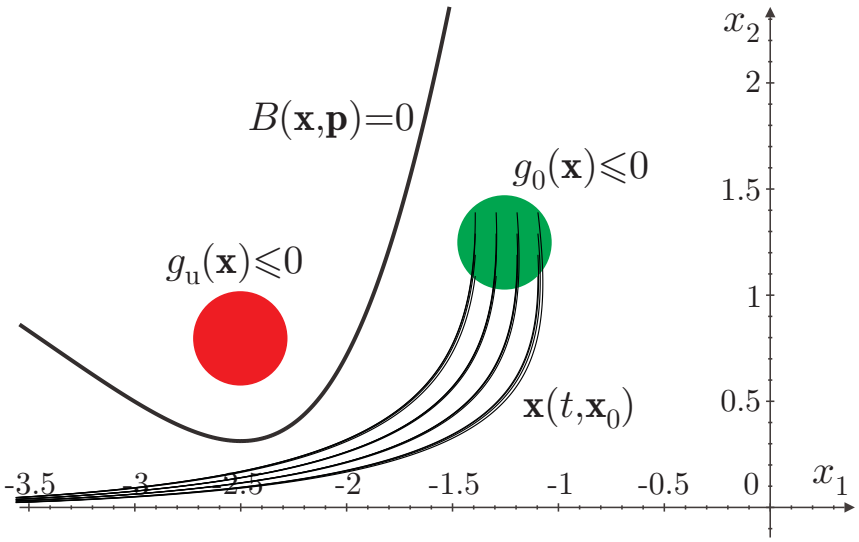


Figure 1.3: Résultat de l'exemple 1.

Table 1.1: Résultats obtenus par CSC-FPS sans et avec contracteurs, pour CSC-FPS avec contracteurs et recherche aléatoire, et pour dReal; dans les deux derniers cas, les temps minimum et maximum ainsi que le nombre de bisections pour 10 exécutions indépendantes ont été fournis

			CSC-FPS, sans rand.				CSC-FPS contr. et rand.				dReal			
			ss contr.		av. contr.		min		max		min		max	
Ex.	n	m	temps	bis.	temps	bis.	temps	bis.	temps	bis.	temps	bis.	temps	bis.
5	2	4	25.9	4520	11.7	4453	10.4	4011	13.7	5397	72.6	24962	974.6	56571
6	2	3	TO	/	1.17	159	0.76	109	1.19	173	0.02	19	0.2	180
7	2	6	857	20388	1.14	6	0.35	3	1.16	5	37.7	112	TO	/
8	2	6	189	14733	5.70	435	4.08	316	6.16	485	225.1	76	TO	/
9	2	4	TO	/	63.7	4072	58.4	3639	64.2	4290	0.04	78	0.1	106
10	3	4	115	1753	15.9	47	3.20	31	16.5	59	0.04	54	0.08	55
11	6	7	530	67600	0.14	261	0.14	281	0.14	446	0.2	88	0.2	114

Les résultats donnés dans le tableau 1.1 montrent l'importance des contracteurs qui apportent un gain sur tous les exemples considérés. L'ensemble des exemples a été également traité avec l'outil dReal [GKC13] qui est un solveur fondé sur la technique SAT-modulo-theory et [BT14] qui utilise également les outils de l'analyse par intervalles.

1.3.4 Conclusion partielle

Ce paragraphe a présenté une nouvelle méthode pour construire des fonctions barrières paramétrées pour prouver la sûreté des systèmes dynamiques non-linéaires perturbés. La recherche de fonctions barrières est formalisée dans le cadre d'un problème de satisfaction de contraintes intervalles quantifiées. L'algorithme CSC-FPS [JW96] a été étendu pour prendre en compte des opérateurs de contraction spécifiques à ce problème de synthèse de fonctions barrières. L'approche proposée peut ainsi calculer des fonctions barrières pour une large classe de systèmes dynamiques non-linéaires et perturbés.

Une approche alternative fondée sur les outils RSolver ou dReal peut être significative-

ment plus efficace pour des classes spécifiques de systèmes dynamiques, par exemple, si les fonctions paramétrées sont linéaires en les paramètres. Une combinaison de l'approche proposée dans cette thèse avec celles de RSolver ou dReal serait bénéfique pour améliorer l'efficacité globale de la caractérisation de fonctions barrières pour les systèmes dynamiques non-linéaires.

1.4 Sûreté des systèmes dynamiques hybrides

Dans cette partie, nous considérons des modèles dynamiques hybrides pour représenter le comportement de systèmes mêlant des comportements discrets et continus. Les systèmes de contrôle-commande peuvent être représentés par ce type de modèles, par exemple, les systèmes de freinage automatique des voitures ou les systèmes de commande de vol des avions de ligne. Le modèle des automates hybrides, introduit dans [ACH+95], est le modèle mathématique hybride qui est considéré dans ce paragraphe. Un automate hybride est un automate possédant un nombre fini de modes auxquels sont associés un ensemble fini de variables continues dont les évolutions sont décrites par un ensemble d'équations différentielles ordinaires.

L'objectif de ce paragraphe est d'étendre aux automates hybrides les résultats obtenus pour les systèmes dynamiques continus, exposés dans le paragraphe 1.3. Le paragraphe 1.4.1 introduit les notations et le problèmes de la sûreté des automates hybrides. Le paragraphe 1.4.2 présente le problème de sûreté à l'aide de fonctions barrières paramétrées dans le formalisme des problèmes de satisfaction de contraintes. De plus, elle introduit trois solutions possibles pour la synthèse de fonctions barrières, toutes se fondant sur l'algorithme CSC-FPS.

1.4.1 Formulation du problème

Un automate hybride $\mathcal{H}$ [ACH+95] est représenté par un tuple

$$\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho), \tag{1.11}$$

avec

- $\mathcal{X} \subseteq \mathbb{R}^n$ l'espace d'état des variables continues,
- $\mathcal{L}$ un ensemble fini de modes,
- $\mathcal{X}_0 : \mathcal{L} \to \mathcal{P}(\mathcal{X})$ est un sous-ensemble $\mathcal{X}_0(\ell) \subseteq \mathcal{X}$ contenant l'ensemble des états initiaux possibles pour chaque mode $\ell \in \mathcal{L}$,

- $\mathcal{I} : \mathcal{L} \to \mathcal{P}(\mathcal{X})$ définit un ensemble invariant $\mathcal{I}(\ell) \subseteq \mathcal{X}$ pour chaque mode $\ell \in \mathcal{L}$, par exemple, le sous ensemble dans lequel les états du système sont supposés rester quand l'automate hybride est dans le mode ℓ,
- $f_\ell : \mathcal{X} \times \mathcal{D}_\ell \to \mathbb{R}^n$ est un champ de vecteurs pour chaque mode $\ell \in \mathcal{L}$ décrivant l'évolution des variables continues quand l'automate hybride est dans le mode ℓ avec

$$\dot{\mathbf{x}} = f_\ell(\mathbf{x}, \mathbf{d}) \tag{1.12}$$

  et $\mathbf{d} \in \mathcal{D}_\ell$ est une perturbation appartenant à un ensemble compact $\mathcal{D}_\ell \subset \mathbb{R}^{n_\mathrm{d}}$
- $\Gamma : \mathcal{L} \times \mathcal{L} \to \mathcal{P}(\mathcal{X})$ définit un sous-ensemble $\Gamma(\ell, \ell') \subseteq \mathcal{X}$ pour chaque couple de modes (ℓ, ℓ') définit une transition discrète du mode ℓ au mode ℓ' quand la partie continue du systèmes atteint $\Gamma(\ell, \ell')$; $\Gamma(\ell, \ell') = \emptyset$ quand aucune transition est possible entre ℓ et ℓ',
- $\rho_{\ell,\ell'} : \Gamma(\ell, \ell') \to \mathcal{I}(\ell')$ est la fonction de remise à zéro quand une transition survient entre le mode ℓ et le mode ℓ'.

Déterminer si un automate hybride est sûr est difficile. Pour résoudre ce problème, l'approche fondée sur les fonctions barrières a été introduite dans [PJ04]. Cette méthode ne s'appuie pas sur un calcul explicite de l'ensemble des trajectoires du système mais elle a pour objectif de trouver une fonction β_ℓ qui sépare l'ensemble des trajectoires du système des zones dangereuses $\mathcal{X}_u(\ell)$.

Theorem 2 ([PJ04]). *Considérons l'automate hybride $\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho)$. Supposons qu'il existe une famille de fonctions différentiables $\beta_\ell(\mathbf{x})$, $\ell \in \mathcal{L}$ telle que, pour tout couple $(\ell, \ell') \in \mathcal{L}^2$ avec $\ell \neq \ell'$, on a*

$$\beta_\ell(\mathbf{x}) \leqslant 0 \quad \forall \mathbf{x} \in \mathcal{X}_0(\ell) \tag{1.13}$$

$$\beta_\ell(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \mathcal{X}_u(\ell) \tag{1.14}$$

$$\beta_\ell(\mathbf{x}) = 0 \implies \frac{\partial \beta_\ell(\mathbf{x})}{\partial \mathbf{x}} f_\ell(\mathbf{x}, \mathbf{d}) < 0 \quad \forall \mathbf{x} \in \mathcal{I}(\ell)\, \forall \mathbf{d} \in \mathcal{D}_\ell \tag{1.15}$$

$$\beta_\ell(\mathbf{x}) \leqslant 0 \implies \beta_{\ell'}(\rho_{\ell,\ell'}(\mathbf{x})) \leqslant 0 \quad \forall \mathbf{x} \in \Gamma(\ell, \ell') \tag{1.16}$$

alors l'automate hybride $\mathcal{H}$ est sûr.

1.4.2 Formulation des fonctions barrières paramétriques

Le théorème 2 ne fournit pas une procédure constructive pour obtenir les fonctions β_ℓ. Comme dans le cas des modèles à temps continu, nous définissons une famille de fonctions candidates paramétrées $\beta_\ell(\mathbf{x}, \mathbf{p}_\ell)$ et nous cherchons une valeur du vecteur de paramètres $\mathbf{p}_\ell$, $\ell \in \mathcal{L}$ pour laquelle les conditions du théorème 2 sont satisfaites. Si pour au moins un

mode $\ell \in \mathcal{L}$, on échoue à trouver un tel vecteur de paramètres $\mathbf{p}_\ell$, cela ne signifie pas que l'automate hybride n'est pas sûr mais que nous devons potentiellement chercher β_ℓ dans une autre famille de fonctions.

Supposons qu'il existe pour chaque mode $\ell \in \mathcal{L}$ des fonctions $g_0 : \mathcal{L}\times\mathcal{X} \to \mathbb{R}$, $g_\mathrm{u} : \mathcal{L}\times\mathcal{X} \to \mathbb{R}$, $g_\Gamma : \mathcal{L} \times \mathcal{L} \times \mathcal{X} \to \mathbb{R}$ et $g_\mathcal{I} : \mathcal{L} \times \mathcal{X} \to \mathbb{R}$ tels que $\mathcal{X}_0(\ell) = \{\mathbf{x} \in \mathcal{X} \,|\, g_0(\ell, \mathbf{x}) \leqslant 0\}$, $\mathcal{X}_\mathrm{u}(\ell) = \{\mathbf{x} \in \mathcal{X} \,|\, g_\mathrm{u}(\ell, \mathbf{x}) \leqslant 0\}$, $\Gamma(\ell, \ell') = \{\mathbf{x} \in \mathcal{X} \,|\, g_\Gamma(\ell, \ell', \mathbf{x}) \leqslant 0\}$ et $\mathcal{I}(\ell) = \{\mathbf{x} \in \mathcal{X} \,|\, g_\mathcal{I}(\ell, \mathbf{x}) \leqslant 0\}$. Pour chaque mode, l'ensemble des perturbations $\mathcal{D}_\ell$ est représenté par un pavé $[\mathbf{d}]_\ell$. L'espace d'état $\mathcal{X}$ est également représenté par un pavé $[\mathbf{x}]$ de diamètre suffisamment grand. Pour chaque mode ℓ, supposons aussi que le pavé de recherche des paramètres $\mathbf{p}_\ell$ de $\beta_\ell(\mathbf{x}, \mathbf{p}_\ell)$ est $[\mathbf{p}]_\ell$.

Avec les notation précédentes, nous pouvons reformuler le théorème 2 de la manière suivante.

Proposition 2. *Considérons un automate hybride décrit par $\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho)$. Supposons qu'il existe une fonction différentiable $\beta_\ell(\mathbf{x}, \mathbf{p})$ qui satisfait*

$\forall \ell \in \mathcal{L}$, $\exists \mathbf{p}_\ell \in [\mathbf{p}]_\ell$, $\forall \mathbf{x} \in [\mathbf{x}]$, $\forall \mathbf{d} \in [\mathbf{d}]_\ell$

$$g_0(\ell, \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \mathbf{p}_\ell) \leqslant 0, \tag{1.17}$$

$$g_u(\ell, \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \mathbf{p}_\ell) > 0, \tag{1.18}$$

$$g_\mathcal{I}(\ell, \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \mathbf{p}_\ell) \neq 0 \vee \frac{\partial \beta_\ell(\mathbf{x}, \mathbf{p}_\ell)}{\partial \mathbf{x}} f_\ell(\mathbf{x}, \mathbf{d}) < 0, \tag{1.19}$$

et $\forall \ell' \in \mathcal{L}$, avec $\ell' \neq \ell$,

$$g_\Gamma(\ell, \ell', \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \mathbf{p}_\ell) > 0 \vee \beta_{\ell'}(\rho_{\ell,\ell'}(\mathbf{x}), \mathbf{p}_{\ell'}) \leqslant 0, \tag{1.20}$$

alors l'automate hybride $\mathcal{H}$ est sûr.

1.4.3 Algorithme de résolution

Pour prouver la sûreté d'un automate hybride $\mathcal{H}$,, une fois que la fonction barrière paramétrée $\beta_\ell(\mathbf{x}, \mathbf{p})$ a été choisie pour chaque mode $\ell \in \mathcal{L}$, on doit trouver un $\mathbf{p}_\ell \in [\mathbf{p}]_\ell$ tel que les $\beta_\ell(\mathbf{x}, \mathbf{p}_\ell)$ satisfont les conditions de la proposition 2.

Une extension de l'algorithme CSC-FPS a été présentée au paragraphe 1.4, trouver des fonctions barrières pour les automates hybrides demande de relever une nouvelle difficulté puisque les contraintes (1.20) lient les vecteurs de paramètres $\mathbf{p}_\ell$ entre eux.

Nous proposons trois méthodes pour chercher $\mathbf{p}_\ell$, $\ell \in \mathcal{L}$, afin de satisfaire les contraintes données dans la proposition 2.

1.4.4 Recherche indépendante pour chaque mode

Dans cette approche, pour chaque mode $\ell \in \mathcal{L}$, CSC-FPS est utilisé pour trouver une valeur $\widehat{\mathbf{p}}_\ell$ du vecteur de paramètres qui satisfait les contraintes (1.17), (1.18), et (1.19) mettant en jeu uniquement $\mathbf{p}_\ell$. Ensuite, il est nécessaire de vérifier que les paramètres obtenus $\widehat{\mathbf{p}}_\ell$, $\ell \in \mathcal{L}$, satisfont (1.20) pour toutes les transitions possibles. Quand cette méthode fonctionne la complexité est linéaire en le nombre de modes. Cependant, quand cette approche échoue il n'est pas évident de définir une stratégie permettant de trouver l'ensemble de paramètres qui pourraient satisfaire toutes les contraintes simultanément.

1.4.5 Recherche conjointe pour tous les modes

Dans cette approche, toutes les contraintes de la proposition 2 sont considérées simultanément. L'espace de recherche des vecteurs de paramètres est $[\mathbf{p}] = [\mathbf{p}]_0 \times ... \times [\mathbf{p}]_{|\mathcal{L}|}$. L'algorithme CSC-FPS prend en entrée la conjonction de toutes les contraintes de la proposition 5. A cause de la complexité exponentielle de l'algorithme CSC-FPS en fonction de la dimension de l'espace des paramètres, cette approche ne peut fonctionner que pour les automates hybrides qui ont un très faible nombre de modes.

1.4.6 Construction incrémentale

Cette approche considère le graphe associé à la machine à états finis sous-jacente à l'automate hybride. Ce graphe est orienté en fonction des transitions entre modes de l'automate hybride. Nous supposons que ce graphe est connexe, ce qui permet de l'explorer incrémentalement afin de limiter le nombres de contraintes à traiter simultanément.

Chaque mode ℓ est traité un par un jusqu'à ce que tous les modes soient visités. Pour chaque mode ℓ, un vecteur de paramètres valides $\widehat{\mathbf{p}}_\ell$ est calculé pour que les contraintes (1.17), (1.18), et (1.19) soient satisfaites. Les contraintes de transition (1.16) sont ensuite considérées pour le mode courant ℓ et pour tous les modes qui ont été déjà visités. Dans le cas d'une contrainte de transition qui ne peut pas être satisfaite, une procédure de *backtrack* est déclenchée. L'idée est de chercher un autre vecteur de paramètres $\widehat{\mathbf{p}}_{\ell'}$ pour un précédent mode visité ℓ' avant de revenir sur le mode ℓ.

1.4.7 Exemple

Example 3. *(Prajna) Cet exemple est inspiré de [PJ04]. L'automate hybride est composé de dynamiques en dimension* 3 *avec perturbation et est composé de* 2 *modes*

$$f_1\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}x_2\\-x+x_3\\x_1+(2x_2+3x_3)(1+x_3^2)+d\end{pmatrix}, \quad f_2\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}x_2\\-x_1+x_3\\-x_1-2x_2+3+d\end{pmatrix}$$

avec $d \in [0.1, 0.2]$

- *Conditions initiales:*

$$g_0(1, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 0.01, \qquad g_0(2, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 1$$

- *Régions dangereuses:*

$$g_u(1, \mathbf{x}) = -1, \qquad g_u(2, \mathbf{x}) = 10^4(x_1 + 5.05)^2 + x_2^2 + x_3^2 - 1$$

- *Invariants:*

$$g_{\mathcal{I}}(1, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 2, \qquad g_{\mathcal{I}}(2, \mathbf{x}) = (x_1 + 2.05)^2 + 9.5x_2^2 + 9.5x_3^2 - 9.5$$

- *Gardes:*

$$g_{\Gamma}(1, 2, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 1, \qquad g_{\Gamma}(2, 1, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 0.25$$

Les fonctions barrières paramétrées sont de la forme

$$\beta_1(\mathbf{x}, \mathbf{p}) = p_1x_1 + p_2x_2 + p_3x_3 + p_4, \qquad \beta_2(\mathbf{x}, \mathbf{p}) = p_1x_1 + p_2x_2 + p_3x_3 + p_4$$

L'espace de recherche pour les paramètres des fonctions barrières paramétrées est

$$[\mathbf{p}]_0 = [\mathbf{p}]_1 = ... = [\mathbf{p}]_{|\mathcal{L}|} = [-10, 10].$$

Table 1.2: Résultats de CFC-FPS pour les exemples de systèmes hybrides

Exemple	Dimension	Nb de modes	Tps CPU	Nb de bissections
2-TANKS	2	2	1.7s	9488
Prajna	3	2	0.2s	111
CAR	6	3	0.021s	3
Collision	3	6	0.407s	2015

Pour tous les exemples traités, l'algorithme proposé CSC-FPS-Hyb fournit de très bons résultats, voir le tableau 1.2. La cause de cette efficacité est qu'aucun exemple ne nécessite l'activation de la procédure de *backtracking*.

1.4.8 Conclusion partielle

Ce paragraphe a présenté une nouvelle méthode pour calculer des fonctions barrières pour les systèmes hybrides non-linéaires. D'une part, les systèmes hybrides sont modélisés math-

ématiquement à l'aide d'automates hybrides. D'autre part, la formulation du problème de vérification de sûreté est donnée dans le cadre des problèmes de satisfaction de contraintes quantifiées. Les contraintes sont classées en deux familles, les contraintes continues et les contraintes de transition. Les contraintes continues peuvent être résolues à l'aide de l'algorithme CSC-FPS présenté au paragraphe 1.3. Les contraintes de transitions lient les fonctions barrières de chaque mode entre elles et ne permettent par un calcul indépendant de ces fonctions barrières. Pour résoudre cette difficulté, différentes approches ont été proposées. Il est possible de calculer chaque fonction barrière indépendamment et d'essayer de prouver la satisfaction des contraintes de transition ou on peut essayer de résoudre toutes les contraintes en même temps. La première approche est adaptée pour des automates hybrides avec peu de transitions tandis que la seconde approche est adaptée pour les petits systèmes à cause de la complexité exponentielle inhérente à l'approche proposée. Une troisième approche a été proposée qui se fonde sur un parcours de graphe. Ce graphe est issu de la structure de machine à états des automates hybrides. Cette approche a de meilleures qualités pour gérer une large classe d'automates hybrides. Ce paragraphe se termine en donnant quelques résultats expérimentaux sur des exemples issus de la littérature.

1.5 Conclusion et perspectives

1.5.1 Conclusion

Cette thèse développe des outils permettant de prouver qu'un système dynamique est sûr. En supposant qu'une partie de l'espace d'état est dangereuse, un système dynamique est dit sûr lorsque son état n'atteint jamais cette partie dangereuse au cours du temps, quel que soit l'état initial appartenant à un ensemble d'états initiaux admissibles et quel que soit le niveau de perturbation restant dans un domaine admissible. Les outils proposés cherchent à établir des preuves de sûreté pour des systèmes décrits par des modèles dynamiques non-linéaires et des modèles dynamiques hybrides.

Prouver qu'un système dynamique est sûr en calculant explicitement l'ensemble des trajectoires possibles du système lorsque le modèle dynamique est non-linéaire et perturbé reste une tâche très difficile. C'est pourquoi cette thèse aborde ce problème à l'aide de fonctions barrières paramétrées. Une barrière, lorsqu'elle existe, permet de partitionner l'espace d'état et d'isoler l'ensemble des trajectoires possibles de l'état du système de la partie dangereuse de l'espace d'état. La fonction paramétrique décrivant la barrière doit satisfaire un certain nombre de contraintes impliquant la dynamique du modèle, l'ensemble des états initiaux possibles, et l'ensemble dangereux. Ces contraintes ne sont pas convexes en général, ce qui complique la recherche de fonctions barrières satisfaisantes. Précédemment, seules des fonctions barrières polynomiales ont été considérées pour des modèle dynamiques polynomiaux.

Cette thèse considère des systèmes dynamiques relativement généraux avec des barrières

paramétriques quelconques. Les solutions présentées exploitent des outils de satisfaction de contraintes sur des domaines continus et des outils issus de l'analyse par intervalles.

Dans un premier temps, cette thèse considère des systèmes dynamiques non-linéaires à temps continu. Le problème de conception d'une barrière paramétrique est formulé comme un problème de satisfaction des contraintes sur des domaines réels avec des variables quantifiées de manière existentielle et universelle. L'algorithme CSC-FPS a été adapté afin de résoudre le problème de synthèse de barrière. Cet algorithme combine une exploration de l'espace des paramètres de la barrière et une phase de vérification des propriétés de la barrière. A l'aide de contracteurs, il est possible de significativement accélérer la recherche de solutions.

Dans un second temps, ces résultats sont étendus au cas de systèmes décrits par des modèles dynamiques hybrides. La propriété de sûreté doit être prouvée lors de l'évolution à temps continu du système dynamique, mais aussi pendant les transitions du système. Ceci nécessite l'introduction de contraintes supplémentaires qui lient les fonctions barrières associées à chaque mode à temps continu entre elles. Réaliser la synthèse de toutes les fonctions barrières pour les différents modes simultanément n'est envisageable que pour des systèmes de très petite dimension avec peu de modes. Une approche séquentielle a été proposée. Les contraintes liées aux transitions sont introduites progressivement entre les modes pour lesquels une barrière a déjà été obtenue. Lorsque certaines contraintes de transition ne sont pas satisfaites, une méthode de *backtracking* doit être mise en œuvre afin de synthétiser des barrières offrant une meilleure prise en compte des contraintes de transition non satisfaites.

Ces approches ont été évaluées et comparées avec des techniques de l'état de l'art sur des systèmes décrits par des modèles à temps continu et des modèles hybrides.

1.5.2 Perspectives

Quelques pistes de recherche ont également été proposées.

En suivant les travaux [PR05], la première piste de recherche concerne l'analyse d'atteignabilité des systèmes dynamiques continus non-linéaires. Ce problème peut être considéré comme le problème dual de la preuve de sûreté des systèmes dynamiques continus [PR05]. Les idées mises en place pour la sûreté trouveraient ainsi une nouvelle application qui permettrait de prouver qu'il existe une trajectoire menant vers une région cible de l'espace d'état partant d'un espace initial donné.

La seconde piste de recherche est en lien avec la problématique du calcul de l'espace atteignable des systèmes dynamiques continus. L'idée est d'utiliser des fonctions barrières délimitant un sous-espace de l'espace d'état de volume de plus en plus petit, tout en contenant les trajectoires du système. Cette piste est une application directe de l'approche proposée dans cette thèse pour la construction des fonctions barrières.

Chapter 2

Introduction

2.1 Context

The increasing needs of safety, environmental protection, or comfort require the use of automatic control systems in many industrial products. In particular, the term *smart systems* recently emerges as a new kind of systems, developed in transport or energy domain, which can automatically respond and adapt to their environment. Basically, the automation of systems come from the wide use of electronic components controlled by computer programs. In consequence, systems become increasingly more complex. Indeed, the number of functions assumed by systems increases, and these functions are increasingly dependent on each other, for example a speed controller coupled to a frontal collision detector in cars.

The counterpart of this automation is the use of computer programs for critical tasks, for which failure can have catastrophic consequences. For example, the cruise control of a car has a fairly simple behavior, the driver intuitively set a speed, then a computer, or more precisely a piece of software, tries to maintain the vehicle speed by controlling the acceleration and brake of the car. However, a failure of this software can result in an accident, and lead to the death of motorists. The design of such software that ensure a safe behavior is challenging. Note that these security constraints can be in conflit with economic constraints which require to always go faster to reduce the time-to-market.

Computer programs mentioned above are part of the family of control-command programs, which allow to bring a system to a desired state. The characteristic of these programs is to continuously interact with its physical environment, *e.g.*, the speed of a car in the case of the cruise control system. This interaction connects two worlds that evolve differently over time. The physical environment evolves continuously in time while computer programs update its state at a given rate. The term *hybrid* is then used to describe the study of programs taking into account its physical environment. To continue with the example of the cruise control system, the execution of such a program is defined by the following steps: access the current speed of the car (through sensors), take the decision to speed up or slow,

perform the action associated with the decision (with actuators). These three steps are repeated at a constant frequency, for example every 10ms. The consequence of the decision by the program to speed up or slow down causes a change in the current speed of the car that will eventually influence next decisions. The challenge in the implementation of such programs is to understand the tight interactions between a software program and its physical environment.

Formal verification methods can be used to increase confidence in the software operations by providing mathematical proof of their behaviors. They lead to the development of automated tools, that check the behavior of computer programs. The automation of the verification allows to manage the growing complexity of control-command systems and thereby also gives an answer to economic constraints. In case of hybrid systems, the challenge is to define formal verification tools that can deal with physical parts. Indeed, formal verification methods have a long history in Computer Science where discrete mathematical objects have been considered while dealing with physical environment usually require to deal differential equations.

This thesis aims at bridging the gap between the needs of verification of hybrid systems and formal methods. In particular, the work presented in this thesis define new algorithms to prove the safety of hybrid systems.

2.2 Organization and notations

The thesis is organized as followed.

In Chapter 3, a survey of the state of the art on formal verification methods for hybrid systems is described. This survey is focused on a particular method named *barrier certificate*, as defined in [PJ04], which is the basis of new algorithms defined in this thesis. As interval analysis is the main theoretical tool used in this thesis, the second part of this chapter reviews the main features of interval analysis.

The first contribution is presented in Chapter 4. A new algorithm to compute barrier certificate for continuous-time dynamical systems is presented. After formally defining the concept of safety for continuous systems, the contribution is presented. Mainly, this contribution is two folds

- a new formulation of the barrier certificate as *quantified constraint satisfaction problem* (QCSP);
- an new algorithm to solve QCSP for barrier certificates using interval analysis tools.

The end of the chapter is dedicated to present experimental results.

The second contribution is presented in Chapter 5. A new algorithm to compute barrier certificate for hybrid automata is described. After presenting the mathematical model of

hybrid automata used to model hybrid systems and the formal definition of the safety for this kind of systems, the contribution is presented. In particular, the contribution is two folds

- a new formulation of the barrier certificate as *quantified constraint satisfaction problem* (QCSP)
- an new algorithm to solve QCSP for barrier certificates using for hybrid automata based on new methods to deal with complexity issues. In particular, as hybrid automata are graph-based representation, a combination of QCSP resolution methods and graph path is used.

The end of the chapter is dedicated to present experimental results.

In Chapter 6, the conclusion of this thesis is drawn and some perspectives of this work are provided.

2.3 List of publications

International journal Adel Djaballah, Michel Kieffer, Alexandre Chapoutot, and Olivier Bouissou. Construction of Parametric Barrier Functions for Dynamical Systems using Interval Analysis. Automatica, 78, 2017, Elsevier.

International conference with committee Olivier Bouissou, Alexandre Chapoutot, Adel Djaballah, and Michel Kieffer, Computation of parametric barrier functions for dynamical systems using interval analysis. In Proceedings of the 53rd IEEE Conference on Decision and Control, 2014.

Chapter 3

State of the art

In this thesis, one is interested in proving that dynamical systems are safe [MP90], *i.e.*, that nothing bad will happen during the evolution with time of system. For that purpose, one assumes that the state-space of the system can be partitioned into two sets, one in which the system is safe and a second in which the system is unsafe. The system is considered as safe if one is able to prove that the state evolving from any allowed initial condition, does not reach the unsafe part of the state space in finite time. More formal definitions of the safety property will be given in Chapter 4 and Chapter 5 depending of the dynamical system and of the considered model.

Different models of dynamical systems have been considered in the litterature.

Discrete-time dynamical systems are defined by discrete-time dynamics of the form

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) \quad \text{with} \quad \mathbf{x}_0 \text{ given.}$$

Continuous-time dynamical systems are defined by a continuous-time dynamics, where the evolution with time of the system is assumed to be described by a system of ordinary differential equations as

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) \quad \text{with} \quad \mathbf{x}(0) \text{ given} \tag{3.1}$$

or by a system of algebraic differential equations such as

$$f(\mathbf{x}(t), \dot{\mathbf{x}}(t)) = 0 \quad \text{with} \quad \mathbf{x}(0) \quad \text{and} \quad \dot{\mathbf{x}}(0) \text{ given.}$$

Hybrid dynamical systems are defined, in a simple form in case with two modes only, by

$$\dot{\mathbf{x}}(t) = \begin{cases} f_1(\mathbf{x}(t)) & \text{if} \quad g(\mathbf{x}(t)) \geqslant 0 \\ f_2(\mathbf{x}(t)) & \text{otherwise} \end{cases} \text{with} \quad \mathbf{x}(0) \text{ given} \ . \tag{3.2}$$

A more complete definition of hybrid dynamical systems in terms of hybrid automata is given in Chapter 5.

In this thesis, only dynamical systems of the form of (3.1) and (3.2) are considered and a review of the main methods used to prove safety properties of systems described by continuous-time dynamics and hybrid models are presented in Section 3.1.

The tools developed in this thesis to prove the safety properties of such systems will rely on interval analysis, which is used to solve constraint satisfaction problems. These two notions will thus be briefly recalled in Section 3.2.

3.1 Formal Verification of the Safety for Dynamical Systems

In the last decades several methods have been introduced to prove the safety of dynamical systems encountered in various scientific domains among Computer Science and Control Theory. In consequence, different approaches have been considered and an overview of the most important methods to prove the safety of a system is presented in this section. One focuses on two classes of approaches: *reachable set computation* techniques, presented in Section 3.1.1, and *invariant generation methods*, presented in Section 3.1.2. A focus on the method of the construction of barrier certificates is presented in Section 3.1.3 as it is the main approach to prove the safety of systems considered in this thesis.

The safety property verification may be directly introduced in the design phase of a system and of its control, instead of being verified *a posteriori*, as done in this thesis. In [Pla07, Pla10], theorem-proving approaches are employed using symbolic-numeric techniques to synthesize invariants for differential (continuous and hybrid) systems. In particular, quantifier elimination techniques are intensively used and more recently a combination with the approach presented in [KDSA14] has been defined in [AKD+15].

3.1.1 Reachable Set Computation

The main idea in the reachable set approach is to compute the set $\mathcal{R}$ of all the possible temporal evolution of the dynamical system, starting from a set of possible initial values for the state. The proof of the safety of the system is then based on a simple verification method which checks that the set $\mathcal{R}$ does not intersect unsafe regions of the state space.

The main challenge in this approach is to compute the set $\mathcal{R}$ or more precisely a "sharp" over-approximation of $\mathcal{R}$ taking into account the complexity of the dynamics of the system and the potential complex form of the set $\mathcal{R}$. Different techniques have been defined considering various set representations and different ways to propagate these sets according to the dynamics of the systems.

When considering non-linear equations of the form (3.1), the computation of the evolution with time of the state is mainly solved by methods named *validated numerical integration* considering sets of values representing by boxes [NJC99, RA11, AdSC16]. Basically, validated numerical integration methods are based on numerical integration methods involving Taylor series expansions or Runge-Kutta methods. In both cases, the numerical approximation error has to be bounded. In most of the cases, the bound is computed using interval versions of the Picard-Lindelöf operator which are also able to prove the existence and uniqueness of the solution.

When considering the case of systems described by hybrid dynamical models, different approaches have also been considered to compute efficiently the set $\mathcal{R}$. In [ABDM00, Tiw03, GLGM06] the reachable region $\mathcal{R}$ is computed for linear hybrid systems for a finite time horizon using geometric representation such as polyhedra. In [Tiw03] a refinement of the computed reachable set for linear hybrid systems is considered by taking into account the properties of linear dynamical systems. The reachable region for non-linear systems is computed in [SSM04] using an abstraction of the non-linear system by a linear system expressed in a new system of coordinates. The reachability of a non-linear system is formulated as an optimization problem in [CK03]. In [CAS12], the Picard-Lindelöf operator is combined with Taylor models to find the reachable region for non-linear hybrid systems.

Various tools exist to prove the safety of an hybrid system using reachability approaches. In [HHWT97], a parametric model checker tool called HyTech is proposed. It computes the reachable space for linear hybrid systems using polyhedral representation. In [ABDM00] the **d/dt** system was introduced that computes an approximation of the reachable space for a finite time using polyhedral representation. An extension of the Hytech tool, named PHAVer, is proposed in [Fre05]. PHAVer computes more efficiently the reachable space for linear hybrid system and also can take into account systems with inputs. For non-linear hybrid systems, [San11] proposed an abstraction into linear systems and then computes the reachable space. For some specific systems this method can be free of wrapping effect. The SpaceEx tool [FLGD$^+$11] combines the different method found to compute efficiently the reachable space for possibly large linear hybrid systems. For non-linear hybrid systems, the tool Flow* [CAS12] can be used. In [KGCC15], the tool dReach is presented that is able to compute the reachability set of non-linear hybrid systems using delta-decision SMT solvers.

The main downside of the reachability approach is the introduction of over-approximations during the computations which may lead to difficulties to decide whether the system is safe.

3.1.2 Invariant Generation Methods

An alternative way to address the safety problem is by exhibiting an invariant region in which the system remains. An invariant is a part of the state space in which the state of a

dynamical system can be proved to remain. Invariants are very useful to prove the safety of dynamical systems. If an invariant does not contain unsafe regions, then the dynamical system is safe. Methods to characterize invariants have been intensively studied for linear and polynomial dynamics but much work has still to be done for non-linear dynamical and hybrid systems.

A set of methods has been defined to compute invariants for various classes of systems, for example for linear or affine systems [Tiw03] or for polynomial systems [SSM04, GT08, KDSA14, YLW15]. These methods introduce a candidate parametric function, which parameter vector has to be adjusted to define an invariant of the considered dynamical system. Various techniques are then employed to determine satisfying parameter vectors. For example, in [SSM04], theory of ideal over polynomials and Gröbner bases are used to define constraints to be satisfied by the parameter vector to be found. These constraints are then solved numerically using tools such as those introduced in [CH91]. In [GT08], quantified polynomial constraints are introduced for the design of parameters. Then, satisfying parameter vectors are found using Farkas' Lemma and solvers from SAT-Modulo Theory [BT14]. Sum-of-Squares (SoS) polynomials are used in [KDSA14]. The design involves various system simulations and selection of candidate parameter vectors using linear programming. A final validation of the selected parameter vectors is then performed with Mathematica and using interval analysis with dReal [GKC13]. Note that our algorithm presented in Section 4.2 could also be used as validation method for the approach presented in [KDSA14]. Bilinear SoS programming is considered in [YLW15].

In [TRSS01], invariants for hybrid systems are evaluated. They are based on the construction of least and greatest fixed points refined with narrowing-widening operators and quantifier elimination. The constraints generated can be complex. The scalability is the major issue for this technique. Hsolver is proposed in [RS07] that abstracts non-linear hybrid systems into discrete interval representations and refines the model using constraint propagation techniques.

An alternative way to find such an invariant is by considering tools such as Lyapunov functions to prove the stability properties of dynamical systems [GTV85]. For example, [Par03] considers parametric functions to find a Lyapunov function for a system with polynomial dynamics formed by SoS polynomials and employs Semi-definite Programming (SdP) for the parameter synthesis. In [GJPS14], Darboux polynomials are used to design specific forms of Lyapunov functions involving rational functions, logarithmic, and exponential terms. Similar invariants have been also considered in [RMM12]. Note that in [RS10] Lyapunov functions are designed via a branch and relax approach (a standard method coming from interval analysis) and linear programming to solve the induced constraints. An extension of the Lyapunov function methods named barrier certificate is provided in [Pra06]. This approach calculates invariants for dynamical systems that verify safety properties, see Section 3.1.3.

3.1.3 Barrier Certificate Approach

Alternatively, techniques searching for *barrier certificates* aim at determining some parametric function, called *barrier*, defining an hyper-surface in the state-space which is never crossed by the dynamics of the system, see [PJ04, Pra05, PR05, SPW12a, DGXZ17]. A parameter vector of this barrier has to be found such that the barrier separates the region in which the initial state belongs from the unsafe region.

In [PJ04, Pra05, PR05], polynomial dynamics and barrier functions are considered and parameters are designed with SdP, which requires some relaxation to obtain a convex design problem. In [DGXZ17], two candidate functions are combined to define more sophisticated barriers, which parameters are again found via SdP. In [SPW12a], linear matrix inequalities and SoS are used to generate the barrier functions for hybrid dynamical systems with polynomial dynamics.

In [PJ04], the barrier certificate method is extended to polynomial hybrid systems, the computation of the barrier is done through SoS decomposition and SdP. The barrier certificate method is used in [PJP07, SPW12b, AP09] for stochastic, interconnected and biological systems. The barrier certificate concept is used in [GDT$^+$12, BMT12, RJ15] for the controller generation problem. An exponential condition is added in [KHS$^+$13, KSH$^+$14] to the set of constraints of the barrier certificate formulation. This constraints relaxes the conservativity of Prajna's theorem [PJ04] without losing the convexity of the constraints which allows them to use SdP.

To prove the safety of dynamical system $\dot{\mathbf{x}} = f(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}$ using the barrier certificate approach one has to find a function $B(\mathbf{x})$ that satisfies the constraints:

$$\forall \mathbf{x} \in \mathcal{X}_0, \quad B(\mathbf{x}) \leqslant 0, \tag{3.3}$$

$$\forall \mathbf{x} \in \mathcal{X}_u, \quad B(\mathbf{x}) > 0, \tag{3.4}$$

$\forall \mathbf{x} \in \mathcal{X}$,

$$B(\mathbf{x}) = 0 \implies \left\langle \frac{\partial B(\mathbf{x})}{\partial \mathbf{x}}, f(\mathbf{x}) \right\rangle < 0, \tag{3.5}$$

where $\mathcal{X}_0$ is the initial region or set of initial values for the state and $\mathcal{X}_u$ is the unsafe region. Construction of this barrier function can be computationally hard. For systems whose vector fields are polynomial and set descriptions are semi-algebraic [PJ04] proposed an approach based on relaxation of the constraints, the SoS decomposition and SdP.

3.1.3.1 Sum-of-Square Decomposition

Before presenting the Sum-of-Square decomposition methods for multivariate polynomials inspired by [PR05], some basic definitions on matrices are briefly recalled.

The set of square matrices of dimension n with real coefficients is denoted by $\mathbb{M}_{n\times n}$. The

transposition of a matrix $M \in \mathbb{M}_{n\times n}$ is denoted by M^T. A matrix $M \in \mathbb{M}_{n\times n}$ is symmetric if and only if $M = M^T$.

A symmetric matrix $M \in \mathbb{M}_{n\times n}$ is positive semi-definite if and only if for all non-null vectors $\mathbf{x} \in \mathbb{R}^n$ one has $\mathbf{x}^T M\mathbf{x} \geqslant 0$. A positive semi-definite matrix M will be also denoted by $M \succcurlyeq 0$.

Definition 1. *For a symmetric positive semi-definite matrix $M \in \mathbb{M}_{n\times n}$, the Choleski decomposition of M produces a lower triangular matrix L such that $M = L^T L$.*

A multivariate polynomial $p(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^n$ is a SoS polynomial if can be written as

$$p(\mathbf{x}) = \sum_{i=0}^{m} f_i^2(\mathbf{x}) \tag{3.6}$$

where $f_i(\mathbf{x})$, $i = 1, \ldots, m$, are multivariate polynomials in $\mathbf{x}$. When $p(\mathbf{x})$ is a SoS polynomial of degree $2d$, it may also be written as

$$p(\mathbf{x}) = z(\mathbf{x})^T Q z(\mathbf{x}), \tag{3.7}$$

where Q is some a symmetric positive semi-definite matrix Q and $z(\mathbf{x})$ is a vector of containing monomials in $\mathbf{x}$ of degree less than or equal to d. The relation between the two definitions 3.6 and 3.7 of SoS polynomials involves the Choleski factorization of $Q = L^T L$:

$$p(\mathbf{x}) = z(\mathbf{x})^T Q z(\mathbf{x}) = z(\mathbf{x})^T L^T L z(\mathbf{x}) = (Lz(\mathbf{x}))^T (Lz(\mathbf{x})) = \|Lz(\mathbf{x})\|_2^2 = \sum_{i=1}^{n} (Lz(\mathbf{x}))_i^2$$

where $(Lz(\mathbf{x}))_i$ is the ith element of the vector $Lz(\mathbf{x})$. In Example 4, coming from [PR05], a quartic polynomial is decomposed as a sum-of-squares.

Example 4. *Let $p(x, y) = 2x^4 + 2x^3y - x^2y^2 + 5y^4$. The SoS decomposition aims at rewriting p such that $p(x, y) = z(x, y)^T Q z(x, y)$ with Q symmetric and $Q \succcurlyeq 0$. If one chooses $z(x, y)$ such that $z(x, y) = (x^2, y^2, xy)$, one has*

$$\begin{aligned} p(x,y) &= \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix} \\ &= q_{11}x^4 + q_{22}y^4 + (2q_{12} + q_{33})x^2y^2 + 2q_{13}x^3y + 2q_{23}xy^3 \quad . \end{aligned}$$

By identification, the following constraints have to be solved

$$q_{11} = 2, \quad q_{22} = 5, \quad q_{13} = 1, \quad q_{23} = 0, \quad 2q_{12} + q_{33} = -1$$

such that $Q \succcurlyeq 0$. *One may choose* $q_{12} = -3$ *and* $q_{33} = 5$. *Then, using the Choleski decomposition of* Q *one has*

$$Q = L^T L \quad \textit{with} \quad L = \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$$

and

$$p(x,y) = z(x,y)^T L^T L z(x,y) = \|Lz(x,y)\|_2^2 = \frac{1}{2}(2x^2 - 3y^2 + xy)^2 + \frac{1}{2}(y^2 + 3xy)^2$$

This SoS decomposition is then used to formulate the barrier design as an SdP problem.

3.1.3.2 Semi-Definite Programming

Semi-definite programming problems generalize classical linear programming problems. Formulating a problem as an SdP [Stu02] gives access to very efficient solvers based on interior-point methods [VB96]. Two typical SdP problems are the following.

Problem 1 (SDP decision)**.** *Given* r *linear functions* $\ell_1, \ldots, \ell_r$, *find a square matrix* Q *such that*

$$\begin{aligned} \ell_i(Q) = 0, \quad \forall i = 1, \ldots, r \\ Q \succcurlyeq 0. \end{aligned}$$

Problem 2 (SDP optimization)**.** *Given* $r+1$ *linear functions* $\ell_0, \ell_1, \ldots, \ell_r$, *find a square matrix* Q *such that*

$$\begin{aligned} Q = \arg\min\nolimits_{A \in \mathbb{M}^{n\times n}} \ell_0(A) \\ \textit{subject to} \\ \ell_i(A) = 0, \quad \forall i = 1, \ldots, r \\ A \succcurlyeq 0. \end{aligned}$$

Note that SoS decomposition can be cast into Problem 1.

3.1.3.3 Computing parametric Barriers with SdP

The way a parametric barrier design problem is cast into an SdP problem is now introduced for a certain class of dynamical models.

Assume that the model of the dynamical system is $\dot{\mathbf{x}} = f(\mathbf{x})$ with $f(\mathbf{x})$ a polynomial in $\mathbf{x}$. Assume also that the sets $\mathcal{X}_0$, $\mathcal{X}_u$, and $\mathcal{X}$ are defined as a semi-algebraic sets of the form $\{\mathbf{x} \in \mathbb{R}^n \mid p(\mathbf{x}) \geqslant 0\}$ with p a polynomial function. Moreover, assume that the barrier

function B is also a polynomial function then a SdP approach can be used to compute B following a convex relaxation of (3.5). In particular, in [PJ04], (3.5) is relaxed into

$$\left\langle \frac{\partial B}{\partial \mathbf{x}}(\mathbf{x}), f(\mathbf{x}) \right\rangle < 0, \tag{3.8}$$

with the consequence of possible elimination of barrier functions that would satisfy (3.5) but not (3.8).

Once an SdP formulation has been obtained, the MATLAB toolbox SOSTOOLS [PPP02] may be used to solve it. This approach has also been defined to deal with perturbed dynamical models and with hybrid dynamical models, see [PJ04].

The contribution of this thesis is an extension of the range of problems to non-linear and non-polynomial continuous-time dynamical and hybrid systems with bounded perturbations using interval analysis. Non-polynomial parametric barrier functions may also be considered.

3.2 Interval analysis

This section briefly overviews the main tools and methods of interval analysis used in this document. These tools are used to prove the safety of systems described by continuous-time dynamical models and hybrid models. Interval analysis introduces intervals and interval vectors or boxes, as well as the notion of inclusion function, which provide an outer-approximation of the range of a function over an interval or a box. Inclusion functions will be instrumental to *numerically prove* properties of functions, such as their positivity over some box. Such properties may be proved despite the fact that computers natively provide only integer and limited accuracy floating-point arithmetic by properly rounding results.

Note that in a computer an interval is represented as a connected, closed set of $\mathbb{F}$, the set of floating-point numbers. To ease the presentation avoiding heavy notations due to rounding mode of floating-point numbers only real intervals are considered in the following.

3.2.1 Intervals and interval arithmetic

A real interval is a connected, closed subset of $\mathbb{R}$, the set of real numbers. The set of real intervals is denoted by $\mathbb{IR}$.

Definition 2. *An interval $[a, b]$ is a set of real numbers such that*

$$[a, b] = \{x \in \mathbb{R} \mid a \in \mathbb{R} \wedge b \in \mathbb{R} \wedge a \leqslant x \leqslant b\}.$$

a is the lower bound and b is the upper bound of $[a, b]$. An interval may also be denoted by $[x]$, in which case, its lower and upper bounds are usually denoted by $\underline{x}$ and $\overline{x}$.

For example, $[1, 2]$, $[-6, -3]$ and $\emptyset$ are intervals while $[2, -1]$ is not an interval.

The following properties of intervals will be useful in what follows:

- the midpoint or center of the interval $[a, b]$ is

$$\text{mid}\left([a, b]\right) = \frac{a + b}{2}, \tag{3.9}$$

- the width of the interval $[a, b]$ is

$$\text{width}\left([a, b]\right) = b - a. \tag{3.10}$$

As intervals represent sets of numbers, set operations can considered for intervals. On may evaluate, for example, the intersection of two intervals $[x]$ and $[y]$ as

$$[x] \cap [y] = \begin{cases} \left[\max(\underline{x}, \underline{y}), \min(\overline{x}, \overline{y})\right] & \text{if } \max(\underline{x}, \underline{y}) \leqslant \min(\overline{x}, \overline{y}) \\ \emptyset & \text{otherwise.} \end{cases}$$

The union of two intervals is not necessarily an interval. In what follows, the union operator between two intervals $[x]$ and $[y]$ will represents the smallest interval containing the union of $[x]$ and $[y]$ evaluated as

$$[x] \cup [y] = \left[\min(\underline{x}, \underline{y}), \max(\overline{x}, \overline{y})\right].$$

The set $\mathbb{IR}$ of real intervals can be endowed with a partial order $\subseteq$ such as

$$[x] \subseteq [y] = \begin{cases} \text{true} & \text{if } \underline{x} \geqslant \underline{y} \wedge \overline{x} \leqslant \overline{y} \\ \text{false} & \text{otherwise.} \end{cases}$$

Then, considering some set $\mathcal{X} \subseteq \mathbb{R}$, the interval hull $\square\mathcal{X}$ of $\mathcal{X}$ is the smallest interval containing $\mathcal{X}$.

Intervals may also be seen as uncertain numbers on which arithmetical operations may be performed. Considering some arithmetic operator $\star \in \{+, -, \times, \div\}$ between two real numbers, its extension to intervals is defined as

$$[x] \star [y] = \{x \star y \mid \forall x \in [x], \forall y \in [y]\}, \tag{3.11}$$

which corresponds to the range of results that may be obtained considering all possible values that may be obtained from $x \in [x]$ and $y \in [y]$. An interval containing exactly $[x] \star [y]$ may

be obtained in most of the cases,

$$
\begin{aligned}
[x] + [y] &= \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right] \\
[x] - [y] &= \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right] \\
[x] \times [y] &= \left[\min\left\{\underline{x}\underline{y}, \overline{x}\overline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}\right\}, \max\left\{\underline{x}\underline{y}, \overline{x}\overline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}\right\}\right] \\
[x] \div [y] &= \left[\frac{\underline{x}}{\overline{y}}, \frac{\overline{x}}{\underline{y}}\right] \text{ if } 0 \notin \left[\underline{y}, \overline{y}\right].
\end{aligned}
$$

In the case of a division by an interval containing 0, several approaches may be considered depending on whether $0 \in [x]$. In the remainder of this document, one considers that the result is the entire real line.

3.2.2 Interval vectors

An n-dimensional interval vector or box $[\mathbf{x}]$ is a vector with interval components

$$[\mathbf{x}] = ([x_1], \ldots, [x_n]).$$

It may also be seen as the Cartesian product of n intervals

$$[\mathbf{x}] = [x_1] \times \cdots \times [x_n].$$

The set of n-dimensional interval vectors is denoted by $\mathbb{IR}^n$.

The lower $\underline{\mathbf{x}}$ and the upper bounds $\overline{\mathbf{x}}$ of $[\mathbf{x}] \in \mathbb{IR}^n$ are denoted by

$$
\begin{aligned}
\underline{\mathbf{x}} &= (\underline{x}_1, \cdots, \underline{x}_n) \\
\overline{\mathbf{x}} &= (\overline{x}_1, \cdots, \overline{x}_n)
\end{aligned}
$$

Set operations, arithmetic operations, as well as partial order between intervals are extended to interval vectors in a component-wise manner.

The width of an interval vector $[\mathbf{x}] \in \mathbb{IR}^n$ is

$$\text{width}([\mathbf{x}]) = \max_{1 \leqslant i \leqslant n} \text{width}([x_i]).$$

3.2.3 Inclusion functions

Consider some function $f : \mathbb{R}^n \to \mathbb{R}^k$ and some interval vector $[\mathbf{x}] \in \mathbb{IR}^n$. The range $f([\mathbf{x}])$ of f over $[\mathbf{x}]$ is the set of values of $f(\mathbf{x})$ for all $\mathbf{x} \in [\mathbf{x}]$, *i.e.*, it is

$$f([\mathbf{x}]) = \{f(\mathbf{x}) \mid \mathbf{x} \in [\mathbf{x}]\}. \tag{3.12}$$

The range of elementary function over intervals may be easily obtained for monotonic functions. For example

$$\begin{aligned}\exp([x]) &= [\exp(\underline{x}), \exp(\overline{x})] \\ [x]^3 &= \left[\underline{x}^3, \overline{x}^3\right].\end{aligned}$$

For non-monotonic elementary functions, the range may still be evaluated using usually simple algorithms, for example

$$[x]^2 = \begin{cases} [0, \max\{\underline{x}^2, \overline{x}^2\}] & \text{if } 0 \in [x] \\ [\min\{\underline{x}^2, \overline{x}^2\}, \max\{\underline{x}^2, \overline{x}^2\}] & \text{otherwise.} \end{cases}$$

For more generic functions, interval analysis introduces the notion of inclusion functions which provide outer-approximations of the range of functions over interval vectors.

Definition 3. *An inclusion function* $[f] : \mathbb{IR}^n \to \mathbb{IR}^k$ *for a function* $f : \mathbb{R}^n \to \mathbb{R}^k$ *is a box-valued function which satisfies for all* $[\mathbf{x}] \in \mathbb{IR}^n$*,*

$$f([\mathbf{x}]) \subseteq [f]([\mathbf{x}]). \tag{3.13}$$

The *natural* inclusion function is the simplest to obtain. It suffices to replace all occurrences of the real variables by their interval counterpart and all arithmetic operations are evaluated using interval arithmetic. More sophisticated inclusion functions such as the centered form, or the Taylor inclusion function may also be used, see [JKDW01].

Usually, inclusion functions are *pessimistic, i.e.*, $[f]([\mathbf{x}])$ is seldom equal to $f([\mathbf{x}])$, as illustrated by Example 5.

Example 5. *Consider the function* $f(x) = x^2 - 2x - 1$ *which range has to be evaluated over the interval* $[x] = [-1, 1]$*. Considering the natural inclusion function* $[f]([x]) = [x]^2 - 2[x] - 1$ *evaluated at* $[x] = [0, 1]$*, one gets*

$$\begin{aligned}[f]([0,1]) &= [0,1]^2 - 2[0,1] - 1 \\ &= [0,1] - [0,2] - 1 \\ &= [-3, 0]\end{aligned}$$

whereas the range of $f(x)$ *over* $[x] = [0, 1]$ *can be easily shown to be* $f([0, 1]) = [-2, -1]$*.*

Even if inclusion functions are pessimistic, they may be used to prove numerically properties of functions over intervals or interval vectors. For example, consider the function f *of Example 5. From* $[f]([0, 1]) = [-3, 0]$*, one can show that* $f(x) \leqslant 0$ *for all* $x \in [0, 1]$*.*

3.2.4 Constraint Satisfaction Problems

Expressing a problem as a Constraint Satisfaction Problems (CSP) gives access to a wide variety of solution algorithms [CJ09, Rue05]. CSPs involving variables with values on continuous domains are briefly recalled in this section as they will be instrumental in the main contributions of this thesis.

3.2.4.1 Formulation of a CSP

A generic CSP $\mathcal{H} = (\mathbf{x}, [\mathbf{x}]_0, \mathcal{C})$ involving variables with values on intervals consists of

- a vector $\mathbf{x} = (x_1, \ldots, x_n)$ of variables assumed real-valued in this thesis,
- an interval vector $[\mathbf{x}]_0 = ([x_1]_0, \ldots, [x_n]_0)$ of possible values of the vector $\mathbf{x}$,
- $\mathcal{C} := \{c_1, \ldots, c_{n_\mathrm{e}+n_\mathrm{i}}\}$ is a set of equality constraints of the form

$$c_i(\mathbf{x}) \equiv f_i(\mathbf{x}) = 0 \tag{3.14}$$

  with $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, n_\mathrm{e}$ and of inequality constraints of the form

$$c_i(\mathbf{x}) \equiv g_i(\mathbf{x}) \leqslant 0 \tag{3.15}$$

  with $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = n_\mathrm{e} + 1, \ldots, n_\mathrm{e} + n_\mathrm{i}$.

In a generic CSP, all constraints in $\mathcal{C}$ have to be simultaneously satisfied. In that case, the vector $\mathbf{x}$ is solution of the CSP $\mathcal{H}$ if $\mathbf{x} \in [\mathbf{x}]_0$ and

$$(f_1(\mathbf{x}) = 0) \wedge \cdots \wedge (f_{n_\mathrm{e}}(\mathbf{x}) = 0) \wedge (g_1(\mathbf{x}) = 0) \wedge \cdots \wedge (g_{n_\mathrm{i}}(\mathbf{x}) = 0) \text{ holds true.}$$

More sophisticated quantified CSP may be considered, involving existential $\exists$ and universal $\forall$ operators. In that case, the vector $\mathbf{x}$ and the interval vector $[\mathbf{x}]_0$ may be split into two parts, $\mathbf{x} = (\mathbf{x}_\exists, \mathbf{x}_\forall)$ and $[\mathbf{x}]_0 = [\mathbf{x}]_{\exists,0} \times [\mathbf{x}]_{\forall,0}$. In that case, $\mathbf{x}$ is a solution of the $\exists\forall$-quantified CSP $\mathcal{H}$ if

$$\exists \mathbf{x}_\exists \in [\mathbf{x}]_{\exists,0} \text{ such that } \forall \mathbf{x}_\forall \in [\mathbf{x}]_{\forall,0}\,,$$

$$(f_1(\mathbf{x}) = 0) \wedge \cdots \wedge (f_{n_\mathrm{e}}(\mathbf{x}) = 0) \wedge (g_1(\mathbf{x}) \leqslant 0) \wedge \cdots \wedge (g_{n_\mathrm{i}}(\mathbf{x}) \leqslant 0) \text{ holds true.}$$

At many places of this thesis, one will consider constraints of the form

$$c_i(\mathbf{x}) \equiv h_i(\mathbf{x}) \in \mathcal{Y} \tag{3.16}$$

with $h_i : \mathbb{R}^n \to \mathbb{R}^k$ and $\mathcal{Y} \in \mathbb{R}^k$ defined as

$$\mathcal{Y} = \left\{\mathbf{x} \in \mathbb{R}^k \,|\, \gamma_j(\mathbf{x}) \leqslant 0, j = 1, \ldots, m\right\}. \tag{3.17}$$

Such constraint (3.16) may be easily rewritten as a set of inequality constraints, since

$$h_i(\mathbf{x}) \in \mathcal{Y} \text{ if and only if } (\gamma_1(h_i(\mathbf{x})) \leqslant 0) \wedge \cdots \wedge (\gamma_m(h_i(\mathbf{x})) \leqslant 0).$$

Thus, a CSP involving constraints of the form (3.16) with sets $\mathcal{Y}$ defined as (3.17), can be transformed into a generic CSP with inequality constraints defined as

$$g_1(\mathbf{x}) = \gamma_1(h_i(\mathbf{x})), \ldots, g_m(\mathbf{x}) = \gamma_m(h_i(\mathbf{x})).$$

3.2.4.2 Solution of a generic CSP involving only inequality constraints

Consider a generic CSP $\mathcal{H}_{\mathrm{i}} = (\mathbf{x}, [\mathbf{x}]_0, \mathcal{C})$ involving only inequality constraints of the form

$$c_i(\mathbf{x}) \equiv g_i(\mathbf{x}) \leqslant 0 \tag{3.18}$$

with $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, n_{\mathrm{i}}$. These constraints may be put in vector form to get

$$g(\mathbf{x}) \leqslant \mathbf{0}, \tag{3.19}$$

with

$$g(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_{n_{\mathrm{i}}}(\mathbf{x})).$$

The set-inverter via interval analysis algorithm (SIVIA) [JW93] can be used to get an inner and an outer-approximation of the set $\mathcal{S}$ of all $\mathbf{x}$ solution of $\mathcal{H}_{\mathrm{i}}$, see Algorithm 1. It involves an inclusion function $[g]$ of the function g.

Algorithm 1 SIVIA($\mathcal{H}_{\mathrm{i}}, \varepsilon$)

Require: $\mathcal{W} = \{[\mathbf{x}]_0\}, \mathcal{S}_{\mathrm{in}} = \emptyset, \mathcal{S}_{\mathrm{out}} = \emptyset, \mathcal{S}_{\mathrm{unc}} = \emptyset$
1: **while** $\mathcal{W} \neq \emptyset$ **do**
2: Pop a $[\mathbf{x}]$ out of $\mathcal{W}$
3: **if** $[g]([\mathbf{x}]) \in]-\infty, 0]^{n_{\mathrm{i}}}$ **then**
4: Push $[\mathbf{x}]$ in $\mathcal{S}_{\mathrm{in}}$
5: **else if** $[g]([\mathbf{x}]) \cap]-\infty, 0]^{n_{\mathrm{i}}} = \emptyset$ **then**
6: Push $[\mathbf{x}]$ in $\mathcal{S}_{\mathrm{out}}$
7: **else if** width($[\mathbf{x}]$) $> \varepsilon$ **then**
8: $([\mathbf{x}]_{\mathrm{l}}, [\mathbf{x}]_{\mathrm{r}}) = \mathrm{Bisect}([\mathbf{x}])$
9: Push $[\mathbf{x}]_{\mathrm{l}}$ in $\mathcal{W}$
10: Push $[\mathbf{x}]_{\mathrm{r}}$ in $\mathcal{W}$
11: **else**
12: Push $[\mathbf{x}]$ in $\mathcal{S}_{\mathrm{unc}}$
13: **end if**
14: **end while**

The result of Algorithm 1 is a paving of the initial box $[\mathbf{x}]_0$ in three subpavings, *i.e.*, three

sets of non-overlapping boxes. $\mathcal{S}_{\text{in}}$ contains all subboxes of $[\mathbf{x}]_0$ which have been proved to satisfy the constraint (3.19), $\mathcal{S}_{\text{out}}$ contains all subboxes of $[\mathbf{x}]_0$ which have been proved not to satisfy the constraint (3.19) and $\mathcal{S}_{\text{unc}}$ containing all subboxes for which SIVIA cannot decide due to a limit of the width of subboxes of $[\mathbf{x}]_0$. One has

$$\mathcal{S}_{\text{in}} \subset \mathcal{S} \subset \mathcal{S}_{\text{in}} \cup \mathcal{S}_{\text{unc}}.$$

With the parameter ε one may adjust the trade-off between computing time and accuracy of the characterization of the solution set $\mathcal{S}$.

In Algorithm 1, all boxes which have to be processed are temporarily stored in the set $\mathcal{W}$. $\mathcal{W}$ can be a stack or a queue depending on the way $[\mathbf{x}]_0$ has to be explored, *e.g.*, in depth-first or in breadth-first search.

The complexity of the SIVIA algorithm is exponential in the dimension of the vector $\mathbf{x}$. Using contractors, the practical complexity may be reduced.

3.2.4.3 Contractors

Consider some function $g : \mathbb{R}^n \to \mathbb{R}^k$ and some set $\mathcal{Y} \subset \mathbb{R}^k$ defined as (3.17).

Definition 4. *A contractor* $\mathcal{C}_c : \mathbb{IR}^n \to \mathbb{IR}^n$ *associated to the generic constraint*

$$c : g(\mathbf{x}) \in \mathcal{Y} \tag{3.20}$$

is a function taking a box $[\mathbf{x}]$ *as input and returning a box* $\mathcal{C}_c([\mathbf{x}])$ *satisfying*

$$\mathcal{C}_c([\mathbf{x}]) \subseteq [\mathbf{x}] \tag{3.21}$$

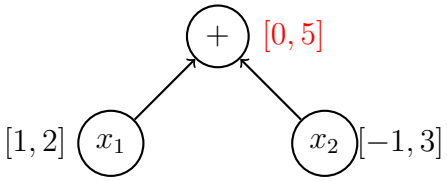
and

$$g([\mathbf{x}]) \cap \mathcal{Y} = g(\mathcal{C}_c([\mathbf{x}])) \cap \mathcal{Y}. \tag{3.22}$$

$\mathcal{C}_c$ provides a box containing the set $\{\mathbf{x} \in [\mathbf{x}] \mid g(\mathbf{x}) \in \mathcal{Y}\}$ of solutions of $g(\mathbf{x}) \in \mathcal{Y}$ included in $[\mathbf{x}]$: (3.21) ensures that the box $\mathcal{C}_c([\mathbf{x}])$ is included in $[\mathbf{x}]$ and (3.22) ensures that no solution of $g(\mathbf{x}) \in \mathcal{Y}$ in $[\mathbf{x}]$ is lost.

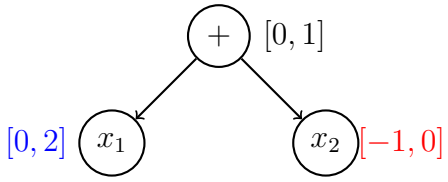
Example 6. *Consider the constraint:* $x_1 + x_2 \subseteq [y]$ *with* $[x_1] = [1, 2]$ *and* $[x_2] = [-1, 3]$*, and* $[y] = [0, 1]$*. The syntactic tree of the constraint may be used to reduce the domain for each variable.*

Step 1 *Forward propagation*



$[x_1] + [x_2] = [1, 2] + [-1, 3] = [0, 5]$
$[y] = [y] \cap [0, 5] = [0, 1]$

Step 2 *Backward propagation*



- $[x_2] = [x_2] \cap ([y] - [x_1]) = [-1, 0]$
- $[x_1] = [x_1] \cap ([y] - [x_2]) = [0, 2]$

Step 3 *(Optional) Iterative application of Step 1 and Step 2 until there is no change in the boxes for each variable.*

Various contractors have been proposed in the literature, *e.g.*, the forward-backward contractor, the contractor by parallel linearization, the Newton contractor, the Krawczyk contractor, *etc.*, see [JKDW01] for more details.

Consider now two functions $g_1 : \mathbb{R}^n \to \mathbb{R}^n$ and $g_2 : \mathbb{R}^n \to \mathbb{R}^n$, two sets $\mathcal{Y}_1 \subset \mathbb{R}^n$ and $\mathcal{Y}_2 \subset \mathbb{R}^n$, and the associated constraints $c_1 \equiv g_1([\mathbf{x}]) \subseteq \mathcal{Y}_1$ and $c_2 \equiv g_2([\mathbf{x}]) \subseteq \mathcal{Y}_2$. Assume that two contractors $\mathcal{C}_{c_1}$ and $\mathcal{C}_{c_2}$ are available for c_1 and c_2. A contractor $\mathcal{C}_{c_1 \wedge c_2}$ for the conjunction $c_1 \wedge c_2$ of c_1 and c_2 may be obtained as

$$\mathcal{C}_{c_1 \wedge c_2}([\mathbf{x}]) = \mathcal{C}_{c_1}([\mathbf{x}]) \cap \mathcal{C}_{c_2}([\mathbf{x}]), \tag{3.23}$$

or by composition of contractors

$$\mathcal{C}_{c_1 \wedge c_2}([\mathbf{x}]) = \mathcal{C}_{c_2}(\mathcal{C}_{c_1}([\mathbf{x}])). \tag{3.24}$$

A contractor $\mathcal{C}_{c_1 \vee c_2}$ for the disjunction $c_1 \vee c_2$ of c_1 and c_2 may be obtained as follows

$$\mathcal{C}_{c_1 \vee c_2}([\mathbf{x}]) = \square\{\mathcal{C}_{c_1}([\mathbf{x}]) \cup \mathcal{C}_{c_2}([\mathbf{x}])\}, \tag{3.25}$$

see [CJ09], with $\square\{\cdot\}$ the interval hull of a set.

Using a contractor $\mathcal{C}_c$ for (3.20), one is able to characterize some $[\tilde{\mathbf{x}}] \subset [\mathbf{x}]$ such that $\forall \mathbf{x} \in [\tilde{\mathbf{x}}]$, $g(\mathbf{x}) \notin \mathcal{Y}$.

Proposition 3. *Consider a box* $[\mathbf{x}]$*, the elementary constraint* (3.20)*, and the contracted box* $\mathcal{C}_c([\mathbf{x}]) \subseteq [\mathbf{x}]$*. Then,*

$$\forall \mathbf{x} \in [\mathbf{x}] \setminus \mathcal{C}_c([\mathbf{x}]), \text{ one has } g(\mathbf{x}) \notin \mathcal{Y}, \tag{3.26}$$

where $[\mathbf{x}] \setminus \mathcal{C}_c([\mathbf{x}])$ *denotes the box* $[\mathbf{x}]$ *deprived of* $\mathcal{C}_c([\mathbf{x}])$*, which is not necessarily a box.*

Proof. Consider $\mathbf{x} \in [\mathbf{x}] \setminus \mathcal{C}_c([\mathbf{x}])$ and assume that $g(\mathbf{x}) \in \mathcal{Y}$. Since $g(\mathbf{x}) \in \mathcal{Y}$ and $\mathbf{x} \in [\mathbf{x}]$, one should have $\mathbf{x} \in \mathcal{C}_c([\mathbf{x}])$, according to (3.22), which contradicts the fact that $\mathbf{x} \in [\mathbf{x}] \setminus \mathcal{C}_c([\mathbf{x}])$. □

3.2.4.4 SIVIA with contractors

Using contractors, one may adapt the SIVIA algorithm, see Algorithm 2, considering again the CSP $\mathcal{H}_i$, where a function Contract is assumed available that implements a contractor for the constraint (3.19).

Algorithm 2 SIVIA-Contract($\mathcal{H}_i, \varepsilon$)

Require: $\mathcal{W} = \{[\mathbf{x}]_0\}, \mathcal{S}_{\text{in}} = \emptyset, \mathcal{S}_{\text{out}} = \emptyset, \mathcal{S}_{\text{unc}} = \emptyset$
while $\mathcal{W} \neq \emptyset$ **do**
  Pop a $[\mathbf{x}]$ out of $\mathcal{W}$
  $[\mathbf{x}] :=$ Contract($[\mathbf{x}]$)
  if $[\mathbf{x}] \neq \emptyset$ **then**
    if $[g]([\mathbf{x}]) \in]-\infty, 0]^{n_{\text{i}}}$ **then**
      Push $[\mathbf{x}]$ in $\mathcal{S}_{\text{in}}$
    else if $[g]([\mathbf{x}]) \cap]-\infty, 0]^{n_{\text{i}}} = \emptyset$ **then**
      Push $[\mathbf{x}]$ in $\mathcal{S}_{\text{out}}$
    else if width($[\mathbf{x}]$) $> \varepsilon$ **then**
      $([\mathbf{x}]_{\text{l}}, [\mathbf{x}]_{\text{r}}) =$ Bisect($[\mathbf{x}]$)
      Push $[\mathbf{x}]_{\text{l}}$ in $\mathcal{W}$
      Push $[\mathbf{x}]_{\text{r}}$ in $\mathcal{W}$
    else
      Push $[\mathbf{x}]$ in $\mathcal{S}_{\text{unc}}$
    end if
  end if
end while

In this case, $\mathcal{S}_{\text{out}}$ contains all subboxes of $[\mathbf{x}]_0$ which have been proved not to satisfy (3.19), but some parts of $[\mathbf{x}]_0$ which are not in $\mathcal{S}$ and have been eliminated by the contractor are not in $\mathcal{S}_{\text{out}}$.

3.2.5 Tools

Several tools have been developed to solve CSPs using interval analysis. In what follows, the two main tools that have been considered in this thesis are briefly presented.

3.2.5.1 IBEX library

IBEX is open source C++ library described in [CJ09, IBE], that contains different tools to solve constraints over real domains. IBEX is built around interval and affine arithmetics. It provides solutions represented by a set of boxes. One of the main feature of IBEX is that

it can handle non-linear constraints thanks to contractors programming. IBEX implements different versions of contractors used in this thesis. One can also find tools to perform global optimization, to evaluate the solution of systems of nonlinear equations, to perform guaranteed numerical integration.

Example 7 shows that a contractor may be easily implemented from the expression of a generic constraint.

Example 7. *Consider the constraint*

$$c \equiv x_1 \exp(-x_2) \in [4, \infty[\tag{3.27}$$

and the initial box $[\mathbf{x}] = ([x_1], [x_2])^T = ([0.1, 10], [0.1, 10])^T$*. From* (3.27)*, one deduces that*

$$x_1 \in ([4, \infty[\,/\, \exp(-[x_2])) \tag{3.28}$$

Moreover, since one also has $x_1 \in [x_1]$*, one deduces that* $x_1 \in \mathcal{C}_1([\mathbf{x}])$*, with*

$$\mathcal{C}_1([\mathbf{x}]) = ([4, \infty[\,/\, \exp(-[x_2])) \cap [x_1] \tag{3.29}$$

Similarly one has $x_2 \in \mathcal{C}_2([\mathbf{x}])$*, with*

$$\mathcal{C}_2([\mathbf{x}]) = -\log([4, \infty[\,/\, [x_1]) \cap [x_2] \tag{3.30}$$

A contractor $\mathcal{C}_c$ *associated to* c *is then*

$$\mathcal{C}_c([\mathbf{x}]) = (\mathcal{C}_1([\mathbf{x}]), \mathcal{C}_2([\mathbf{x}]))^T. \tag{3.31}$$

This contractor for $[\mathbf{x}] = ([0.1, 10], [0.1, 10])^T$ *provides*

$$\mathcal{C}_1([\mathbf{x}]) = ([4.42068, 10], [0.1, 0.916291])^T.$$

This contractor corresponds to the generic forward-backward contractor, also called HC4-revise in [CJ09]. It may be simply implemented, e.g.*, using IBEX as given in Listing 3.1.*

3.2.5.2 RSolver

RSolver is numerical solver proposed by Stefan Ratschan and described in [Rat07]. RSolver aims at solving quantified inequality constraints. RSolver may address disjunction and conjunction of the constraints, as well as quantified CSPs. The output of the solver is a paving. In other terms, the solver produces a set of boxes for which the constraints are true, a set boxes where the constraints are false and a set of boxes for which nothing can be decided.

Listing 3.1: Example of IBEX contractor programming

```
#include "ibex.h"

using namespace std;
using namespace ibex;

int main() {

  unsigned int dimension = 2;
  Variable x(dimension);
  NumConstraint c1(x, x[0]*exp(-x[1]) >= 4.0);
  CtcFwdBwd contractor(c1);

  IntervalVector box(dimension);
  box[0]=Interval(0.1,10.0);
  box[1]=Interval(0.1,10.0);

  cout << "box = " << box << endl;
  contractor.contract(box);
  cout << "after contraction box = " << box << endl;

  return 0;
}
```

Example 8. *Consider the constraint*

$$\exists z \in [-2,2] \text{ such that } x^2+y^2+z^2 \leqslant 1 \quad \text{with} \quad x,y \in [-2,2]\times[-2,2]. \tag{3.32}$$

The algorithm will start by trying to prune the element of z that does not satisfy the constraint. In this case let suppose that no element could be pruned. Then the algorithm will either branch over free variables x or y, for example, it can split the interval $[-2,2]$ associated to y, or rewrite the constraint into $\exists z \in [-2,0] x^2+y^2+z^2 \leqslant 1 \vee \exists z \in [0,2] x^2+y^2+z^2 \leqslant 1$ and re-iterate. Each pruning results in boxes that satisfy the constraint and possibly boxes that does not satisfy the constraints. The iteration stops when a precision ε (provided by the user) is reached. At the end of the algorithm boxes where no conclusion can be made could remain. In Figure 3.1, the paving result produced by RSolver for Constraint (3.32) *is given for a precision $\varepsilon = 0.01$.*

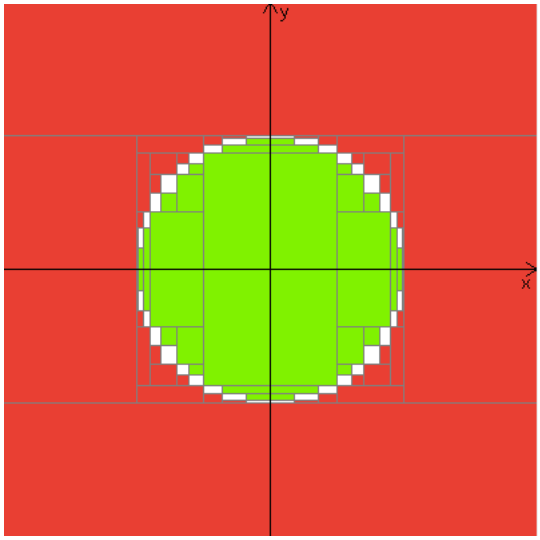


Figure 3.1: Paving result of RSolver on constraint of Example 8. In green, boxes that satisfy the constraint and, in red, boxes that do not satisfy the constraint. In white, boxes for which no conclusion can be made due to the precision ε.

Chapter 4

Barrier functions for continuous-time dynamical systems

This chapter provides a new algorithm to design barrier functions aiming at assessing the safety of a continuous system, [PJ04, Pra06] a barrier function partitions the state space and isolates an unsafe region from the part of the state space containing the initial region. In [PJ04] polynomial barriers are considered for polynomial systems and semi-definite programming is used to find satisfying barrier functions. The work presented in this chapter aims at extending the class of problems that may be considered to non-polynomial systems and to non-polynomial barriers.

First, the design of a barrier function is formulated as a quantified constraints satisfaction problem (QCSP) [BG00, Rat06]. Second, interval analysis is then used to find the parameters of a barrier function such that the QCSP is satisfied. More specifically, the algorithm presented in [JW96] for robust controller design is adapted and supplemented with some of the pruning schemes found in [CJ09] to solve the QCSP associated to the barrier function design.

The chapter is organized as follows. Section 4.1 defines the notion of safety and describes how barrier functions can be used to prove safety. Section 4.2 formulates the design of barrier functions as a QCSP and presents the framework developed to solve the QCSP. Design examples are presented in Section 4.3. Section 4.4 concludes the chapter.

4.1 Safety for continuous-time systems

This section recalls the safety characterization introduced in [PJ04] for continuous-time systems using barrier functions.

Consider the autonomous continuous-time perturbed dynamical system

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{d}), \tag{4.1}$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state vector and $\mathbf{d} \in \mathcal{D}$ is a constant and bounded disturbance. The set of possible initial states at $t = 0$ is denoted $\mathcal{X}_0 \subset \mathcal{X}$. There is some unsafe subset $\mathcal{X}_u \subseteq \mathcal{X}$ that shall not be reached by the system, given any $\mathbf{x}_0 \in \mathcal{X}_0$ at time $t = 0$ and any $\mathbf{d} \in \mathcal{D}$. We assume that classical hypotheses (see, *e.g.,* [BC63]) on f are satisfied so that (4.1) has a unique solution $\mathbf{x}(t, \mathbf{x}_0, \mathbf{d}) \in \mathcal{X}$ for any given initial value $\mathbf{x}_0 \in \mathcal{X}_0$ at time $t = 0$ and any $\mathbf{d} \in \mathcal{D}$.

Definition 5. *The dynamical system* (4.1) *is* safe *if* $\forall \mathbf{x}_0 \in \mathcal{X}_0$, $\forall \mathbf{d} \in \mathcal{D}$ *and* $\forall t \geqslant 0$, $\mathbf{x}(t, \mathbf{x}_0, \mathbf{d}) \notin \mathcal{X}_u$.

4.1.1 Barrier certificates

A way to prove that (4.1) is safe is by the barrier certificate approach introduced in [PJ04]. A barrier is a differentiable function $B : \mathcal{X} \to \mathbb{R}$ that partitions the state space $\mathcal{X}$ into $\mathcal{X}_-$ where $B(\mathbf{x}) \leqslant 0$ and $\mathcal{X}_+$ where $B(\mathbf{x}) > 0$ such that $\mathcal{X}_0 \subseteq \mathcal{X}_-$ and $\mathcal{X}_u \subseteq \mathcal{X}_+$. Moreover, B has to be such that $\forall \mathbf{x}_0 \in \mathcal{X}_0$, $\forall \mathbf{d} \in \mathcal{D}$, $\forall t \geqslant 0$, $B(\mathbf{x}(t, \mathbf{x}_0, \mathbf{d})) \leqslant 0$.

Proving that $B(\mathbf{x}(t, \mathbf{x}_0, \mathbf{d})) \leqslant 0$ requires an evaluation of the solution of (4.1) for all $\mathbf{x}_0 \in \mathcal{X}_0$ and $\mathbf{d} \in \mathcal{D}$. Alternatively, [PJ04] provides some sufficient conditions a barrier function has to satisfy to prove the safety of a dynamical system, see Theorem 3.

Theorem 3. *Consider the dynamical system* (4.1) *and the sets* $\mathcal{X}$, $\mathcal{D}$, $\mathcal{X}_0$ *and* $\mathcal{X}_u$. *If there exists a function* $B : \mathcal{X} \to \mathbb{R}$ *such that*

$$\forall \mathbf{x} \in \mathcal{X}_0, \quad B(\mathbf{x}) \leqslant 0, \tag{4.2}$$

$$\forall \mathbf{x} \in \mathcal{X}_u, \quad B(\mathbf{x}) > 0, \tag{4.3}$$

$\forall \mathbf{x} \in \mathcal{X}$, $\forall \mathbf{d} \in \mathcal{D}$,

$$B(\mathbf{x}) = 0 \implies \left\langle \frac{\partial B(\mathbf{x})}{\partial \mathbf{x}}, f(\mathbf{x}, \mathbf{d}) \right\rangle < 0, \tag{4.4}$$

then (4.1) *is safe.*

In (4.4) $\langle ., . \rangle$ stands for the dot product in $\mathbb{R}^n$. In Theorem 3, (4.2) and (4.3) ensure that $\mathcal{X}_0 \subseteq \mathcal{X}_-$, and $\mathcal{X}_u \subseteq \mathcal{X}_+$, while (4.4) states that if $\mathbf{x}$ is on the border between $\mathcal{X}_-$ and $\mathcal{X}_+$ (*i.e.*, $B(\mathbf{x}) = 0$), then the dynamics f pushes the state back in $\mathcal{X}_-$ whatever the value of the disturbance $\mathbf{d}$.

4.1.2 Parametric barrier functions

The search for a barrier B is challenging since it is over a functional space. As in [PJ04], one considers here barriers belonging to a family of parametric functions (or templates) $B(\mathbf{x}, \mathbf{p})$

depending on a parameter vector $\mathbf{p} \in \mathcal{P} \subseteq \mathbb{R}^m$. Then one may search for some parameter value $\mathbf{p}$ such that $B(\mathbf{x}, \mathbf{p})$ satisfies (4.2)-(4.4).

If there is no $\mathbf{p} \in \mathcal{P}$ such that $B(\mathbf{x}, \mathbf{p})$ satisfies (4.2)-(4.4), this does not mean that the system is not safe: other structures of functions $B(\mathbf{x}, \mathbf{p})$ could provide a barrier certificate.

In what follows, one assumes that the structure of a parametric barrier has been chosen our aim is to determine whether there exists satisfying barrier function with that structure. The choice of the structure is discussed in Section 4.3 and 4.4.

4.2 Constraint satisfaction problem

This section presents an approach to find a barrier function that fulfills the constraints of Theorem 3. These constraints are first reformulated to cast the design of a barrier function as a quantified constraint satisfaction problem (QCSP).

Assume that there exist some functions $g_0 : \mathcal{X} \to \mathbb{R}$ and $g_u : \mathcal{X} \to \mathbb{R}$, such that

$$\mathcal{X}_0 = \{\mathbf{x} \in \mathcal{X} \,|\, g_0(\mathbf{x}) \leqslant 0\} \tag{4.5}$$

and

$$\mathcal{X}_u = \{\mathbf{x} \in \mathcal{X} \,|\, g_u(\mathbf{x}) \leqslant 0\}. \tag{4.6}$$

Theorem 3 may be reformulated as follows.

Proposition 4. *If* $\exists \mathbf{p} \in \mathcal{P}$ *such that* $\forall \mathbf{x} \in \mathcal{X}$, $\forall \mathbf{d} \in \mathcal{D}$

$$\begin{aligned} \xi(\mathbf{x}, \mathbf{p}, \mathbf{d}) = {} & (g_0(\mathbf{x}) > 0 \vee B(\mathbf{x}, \mathbf{p}) \leqslant 0) && (4.7)\\ & \wedge (g_u(\mathbf{x}) > 0 \vee B(\mathbf{x}, \mathbf{p}) > 0) && (4.8)\\ & \wedge \left(B(\mathbf{x}, \mathbf{p}) \neq 0 \vee \left\langle \frac{\partial B}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{p}), f(\mathbf{x}, \mathbf{d}) \right\rangle < 0 \right) && (4.9) \end{aligned}$$

holds true, then the dynamical system (4.1) *is safe.*

Proof. The first component of $\xi(\mathbf{x}, \mathbf{p}, \mathbf{d})$,

$$\xi_0(\mathbf{x}, \mathbf{p}) = (g_0(\mathbf{x}) > 0 \vee B(\mathbf{x}, \mathbf{p}) \leqslant 0) \tag{4.10}$$

may be rewritten as

$$\xi_0(\mathbf{x}, \mathbf{p}) = (g_0(\mathbf{x}) \leqslant 0 \implies B(\mathbf{x}, \mathbf{p}) \leqslant 0),$$

see, *e.g.*, [Gal86]. If $\xi_0(\mathbf{x}, \mathbf{p})$ holds true for some $\mathbf{p} \in \mathcal{P}$ and $\mathbf{x} \in \mathcal{X}$, then one has either $\mathbf{x} \in \mathcal{X}_0$ and $B(\mathbf{x}, \mathbf{p}) \leqslant 0$, or $\mathbf{x} \notin \mathcal{X}_0$. In both cases, (4.2) is satisfied. Similarly the second

component of $\xi(\mathbf{x}, \mathbf{p}, \mathbf{d})$ representing (4.3) may be rewritten as

$$\xi_{\mathrm{u}}(\mathbf{x}, \mathbf{p}) = (g_{\mathrm{u}}(\mathbf{x}) \leqslant 0 \implies B(\mathbf{x}, \mathbf{p}) > 0)\,. \tag{4.11}$$

If $\xi_{\mathrm{u}}(\mathbf{x}, \mathbf{p})$ holds true for some $\mathbf{p} \in \mathcal{P}$ and $\mathbf{x} \in \mathcal{X}$, then one has either $\mathbf{x} \in \mathcal{X}_{\mathrm{u}}$ and $B(\mathbf{x}, \mathbf{p}) \leqslant 0$, or $\mathbf{x} \notin \mathcal{X}_{\mathrm{u}}$. In both cases, (4.3) is satisfied. Now, one may rewrite the last component of $t(\mathbf{x}, \mathbf{p}, \mathbf{d})$,

$$\xi_b(\mathbf{x}, \mathbf{p}, \mathbf{d}) = \left(B(\mathbf{x}, \mathbf{p}) \neq 0 \vee \left\langle \frac{\partial B}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{p}), f(\mathbf{x}, \mathbf{d}) \right\rangle < 0\right) \tag{4.12}$$

as

$$\xi_b(\mathbf{x}, \mathbf{p}, \mathbf{d}) = \left(B(\mathbf{x}, \mathbf{p}) = 0 \implies \left\langle \frac{\partial B}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{p}), f(\mathbf{x}, \mathbf{d}) \right\rangle < 0\right), \tag{4.13}$$

which corresponds to (4.4). If $\exists \mathbf{p} \in \mathcal{P}$ such that $\forall \mathbf{x} \in \mathcal{X}$, $\forall \mathbf{d} \in \mathcal{D}$, $\xi(\mathbf{x}, \mathbf{p}, \mathbf{d})$ holds true, then the conditions of Theorem 3 are satisfied and (4.1) is safe. □

In [PJ04], (4.9) is relaxed into

$$\xi_{\mathrm{b}}(\mathbf{x}, \mathbf{p}, \mathbf{d}) = \left(\left\langle \frac{\partial B}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{p}), f(\mathbf{x}, \mathbf{d}) \right\rangle < 0\right), \tag{4.14}$$

with the consequence of possible elimination of barrier functions that would satisfy (4.9) for some $\mathbf{p}$ but not (4.14). Our aim in this thesis is to design barrier functions without resorting to this relaxation by considering methods from interval analysis [JKDW01] which allow to consider strongly nonlinear dynamics and barrier functions.

4.2.1 Solving the constraints

To find a valid barrier function one needs to find some $\mathbf{p} \in \mathcal{P}$ such that $B(\mathbf{x}, \mathbf{p})$ satisfies the conditions of Proposition 4. For that purpose, the *Computable Sufficient Conditions-Feasible Point Searcher* (CSC-FPS) algorithm [JW96] is adapted.

In what follows, we assume that $\mathcal{X}$, $\mathcal{D}$, and $\mathcal{P}$ are boxes, *i.e.*, $\mathcal{X} = [\mathbf{x}]$, $\mathcal{D} = [\mathbf{d}]$, and $\mathcal{P} = [\mathbf{p}]$. CSC-FPS may also be applied when $\mathcal{X}$, $\mathcal{D}$, and $\mathcal{P}$ consist of a union of non-overlapping boxes.

Consider some function $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ and some box $[\mathbf{z}] \in \mathbb{IR}^k$. CSC-FPS is designed to determine whether

$$\exists \mathbf{p} \in [\mathbf{p}]\,,\ \forall \mathbf{x} \in [\mathbf{x}]\,,\ g(\mathbf{x}, \mathbf{p}) \in [\mathbf{z}] \tag{4.15}$$

and to provide some satisfying $\mathbf{p}$. We extend CSC-FPS to handle conjunctions and disjunctions of constraints and supplement it with efficient pruning techniques involving contractors provided by interval analysis [JKDW01].

FPS branches over the parameter search box $[\mathbf{p}]$. Branching is performed based on the results provided by CSC. For a given box $[\mathbf{p}]_0 \subseteq [\mathbf{p}]$, CSC returns `true` when it manages to

prove that (4.15) is satisfied for some $\mathbf{p} \in [\mathbf{p}]_0$. CSC returns `false` when it is able to show that there is no $\mathbf{p} \in [\mathbf{p}]_0$ satisfying (4.15). CSC returns `unknown` in the other cases.

In Proposition 4, $\xi(\mathbf{x}, \mathbf{p}, \mathbf{d})$ consists of the conjunction of three terms of the form

$$\tau(\mathbf{x}, \mathbf{p}, \mathbf{d}) = (u(\mathbf{x}, \mathbf{p}) \in \mathcal{A}) \vee (v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \in \mathcal{B}). \tag{4.16}$$

For $\xi_0(\mathbf{x}, \mathbf{p})$, defined in (4.7),

$$\mathcal{A} =]0, +\infty] \text{ and } \mathcal{B} =]-\infty, 0]; \tag{4.17}$$

for $\xi_u(\mathbf{x}, \mathbf{p})$, defined in (4.8),

$$\mathcal{A} =]0, +\infty] \text{ and } \mathcal{B} =]0, +\infty[; \tag{4.18}$$

for $\xi_b(\mathbf{x}, \mathbf{p}, \mathbf{d})$, defined in (4.9),

$$\mathcal{A} =]-\infty, 0[\cup]0, +\infty[\text{ and } \mathcal{B} =]-\infty, 0[. \tag{4.19}$$

To illustrate the main ideas of CSC-FPS combined with contractors, one focuses on the generic QCSP

$$\exists \mathbf{p} \in [\mathbf{p}], \forall \mathbf{x} \in [\mathbf{x}], \forall \mathbf{d} \in [\mathbf{d}], \tau(\mathbf{x}, \mathbf{p}, \mathbf{d}) \text{ holds true.} \tag{4.20}$$

Finding a solution for such QCSP involves three steps: validation, reduction of the parameter and state spaces, and bisection.

4.2.1.1 Validation

In the validation step, one tries to prove that some vector $\mathbf{p} \in [\mathbf{p}]$ is such that $\forall \mathbf{x} \in [\mathbf{x}]$, $\forall \mathbf{d} \in [\mathbf{d}]$, $\tau(\mathbf{x}, \mathbf{p}, \mathbf{d})$ holds true. By definition of $\tau(\mathbf{x}, \mathbf{p}, \mathbf{d})$, one has to prove that

$$\exists \mathbf{p} \in [\mathbf{p}], \forall \mathbf{x} \in [\mathbf{x}], \forall \mathbf{d} \in [\mathbf{d}], (u(\mathbf{x}, \mathbf{p}) \in \mathcal{A}) \vee (v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \in \mathcal{B}). \tag{4.21}$$

For that purpose, one chooses some arbitrary $\mathbf{p} \in [\mathbf{p}]$ and evaluates the set of values $u([\mathbf{x}], \mathbf{p}) = \{u(\mathbf{x}, \mathbf{p}) \mid \mathbf{x} \in [\mathbf{x}]\}$ and $v([\mathbf{x}], \mathbf{p}, [\mathbf{d}]) = \{v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \mid \mathbf{x} \in [\mathbf{x}], \mathbf{d} \in [\mathbf{d}]\}$ taken by $u(\mathbf{x}, \mathbf{p})$ and $v(\mathbf{x}, \mathbf{p}, \mathbf{d})$ for all $\mathbf{x} \in [\mathbf{x}]$ and for all $\mathbf{d} \in [\mathbf{d}]$. Outer-approximations of $u([\mathbf{x}], \mathbf{p})$ and $v([\mathbf{x}], \mathbf{p}, [\mathbf{d}])$ are easily obtained using *inclusion functions* provided by interval analysis see section 3.2.

Using inclusion functions, one may evaluate whether

$$[u]([\mathbf{x}], \mathbf{p}) \subseteq \mathcal{A} \quad \text{or} \quad [v]([\mathbf{x}], \mathbf{p}, [\mathbf{d}]) \subseteq \mathcal{B} \quad \text{holds true}$$

for the various $\mathcal{A}$ and $\mathcal{B}$ defined in (4.17), (4.18), and (4.19).

Different choices can be considered for $\mathbf{p}$: one can take a random point in $[\mathbf{p}]$, the middle, or one of the edges of $[\mathbf{p}]$. Here, only the middle of $[\mathbf{p}]$ is considered.

4.2.1.2 Reduction of the parameter and state spaces

To facilitate the search for $\mathbf{p} \in [\mathbf{p}]$ one may first eliminate parts of $[\mathbf{p}]$ which may be proved not to contain any $\mathbf{p}$ satisfying (4.21). The elimination process can be done by evaluation or by using contractors [CJ09], see also section 3.2.

Evaluation Considering the negation of (4.21) one deduces that a box $[\mathbf{p}]$ can be eliminated, *i.e.*, shown not to contain any $\mathbf{p}$ satisfying (4.16), if

$$\forall \mathbf{p} \in [\mathbf{p}], \exists \mathbf{x} \in [\mathbf{x}], \exists \mathbf{d} \in [\mathbf{d}], u(\mathbf{x}, \mathbf{p}) \notin \mathcal{A} \wedge v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \notin \mathcal{B}, \tag{4.22}$$

If, using again inclusion functions, one is able to prove that

$$\exists \mathbf{x} \in [\mathbf{x}], \exists \mathbf{d} \in [\mathbf{d}], [u](\mathbf{x}, [\mathbf{p}]) \cap \mathcal{A} = \emptyset \wedge [v](\mathbf{x}, [\mathbf{p}], \mathbf{d}) \cap \mathcal{B} = \emptyset \tag{4.23}$$

then (4.22) holds true and one can thus eliminate $[\mathbf{p}]$.

In general, $[u](\mathbf{x}, [\mathbf{p}])$ and $[v](\mathbf{x}, [\mathbf{p}], \mathbf{d})$ are not degenerated intervals, *i.e.*, are not reduced to a real value. When $\mathcal{A}$ and $\mathcal{B}$ are half-lines, as is the case for (4.7) and (4.8), one may be able to show that (4.23) holds true. Nevertheless, this will be impossible to show for (4.9) since in this case $\mathcal{A} = \mathbb{R} \setminus \{0\}$ and $[u](\mathbf{x}, [\mathbf{p}])$ is not a degenerated interval. To show that $[u](\mathbf{x}, [\mathbf{p}]) \cap \mathcal{A} = \emptyset$, one needs to have $[u](\mathbf{x}, [\mathbf{p}]) = [0, 0]$, which is impossible in general.

Proposition 3 in section 3.2 can be used to eliminate $[\mathbf{p}]$ or a part of $[\mathbf{p}]$ for which it is not possible to find any $\mathbf{p}$ satisfying (4.21). Consider the constraint

$$\tau \equiv (u(\mathbf{x}, \mathbf{p}) \in \mathcal{A}) \vee (v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \in \mathcal{B}) \tag{4.24}$$

and a contractor $\mathcal{C}_\tau$ for this constraint. It involves elementary contractors for the components of the disjunction in (4.24), combined as in (3.25). For the boxes $[\mathbf{x}]$, $[\mathbf{p}]$, and $[\mathbf{d}]$, one gets

$$([\mathbf{x}]', [\mathbf{p}]', [\mathbf{d}]') = \mathcal{C}_\tau([\mathbf{x}], [\mathbf{p}], [\mathbf{d}]), \tag{4.25}$$

where $[\mathbf{x}]'$, $[\mathbf{p}]'$, and $[\mathbf{d}]'$ are the contracted boxes. Three cases may then be considered.

1. If $[\mathbf{p}] \setminus [\mathbf{p}]' \neq \emptyset$, then

$$\forall \mathbf{p} \in [\mathbf{p}] \setminus [\mathbf{p}]', \forall \mathbf{x} \in [\mathbf{x}], \forall \mathbf{d} \in [\mathbf{d}], u(\mathbf{x}, \mathbf{p}) \notin \mathcal{A} \wedge v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \notin \mathcal{B}, \tag{4.26}$$

   and there is no $\mathbf{p} \in [\mathbf{p}] \setminus [\mathbf{p}]'$ such that (4.24) holds true for all $\mathbf{x} \in [\mathbf{x}]$ and for all $\mathbf{d} \in [\mathbf{d}]$. Consequently, the search space for $\mathbf{p}$ can be reduced to $[\mathbf{p}]'$, see Figure 4.1 (a).

2. If $[\mathbf{x}] \setminus [\mathbf{x}]' \neq \emptyset$ then, from Proposition 3 in Section 3.2, one has

$$\forall \mathbf{p} \in [\mathbf{p}], \forall \mathbf{x} \in [\mathbf{x}] \setminus [\mathbf{x}]', \forall \mathbf{d} \in [\mathbf{d}], u(\mathbf{x}, \mathbf{p}) \notin \mathcal{A} \wedge v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \notin \mathcal{B}, \tag{4.27}$$

and there is no $\mathbf{p} \in [\mathbf{p}]$ such that (4.24) holds true for all $\mathbf{x} \in [\mathbf{x}]$, see Figure 4.1 (b).

3. If $[\mathbf{d}] \setminus [\mathbf{d}]' \neq \emptyset$, then

$$\forall \mathbf{p} \in [\mathbf{p}], \forall \mathbf{x} \in [\mathbf{x}], \forall \mathbf{d} \in [\mathbf{d}] \setminus [\mathbf{d}]', u(\mathbf{x}, \mathbf{p}) \notin \mathcal{A} \wedge v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \notin \mathcal{B}, \tag{4.28}$$

and there is no $\mathbf{p} \in [\mathbf{p}]$ such that (4.24) holds true for all $\mathbf{d} \in [\mathbf{d}]$, see Figure 4.1 (b).

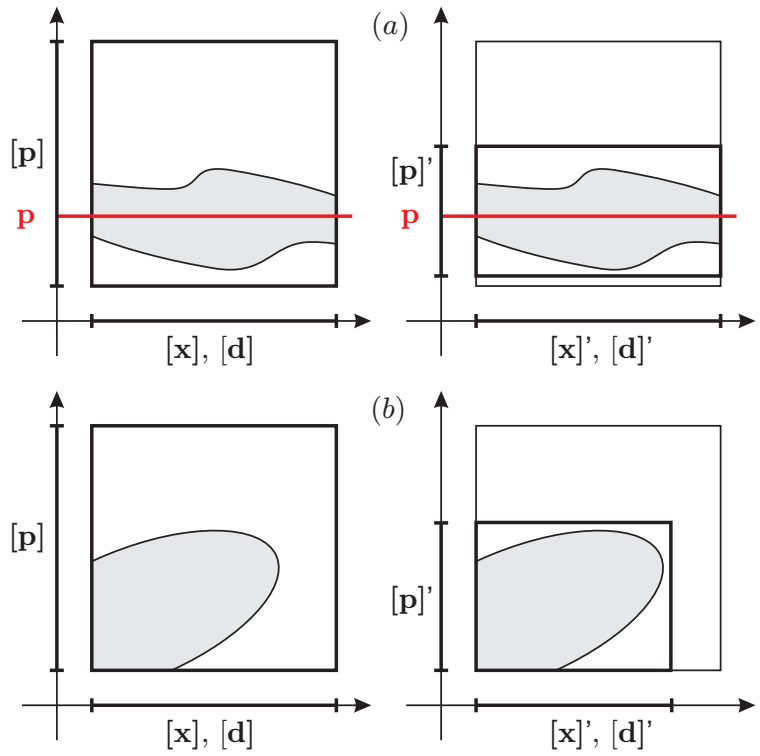


Figure 4.1: Contractions using $\mathcal{C}_\tau$; the set for which (4.24) is satisfied is in gray; (a) $[\mathbf{p}]\setminus[\mathbf{p}]' \neq \emptyset$ and the search space for satisfying $\mathbf{p}$ can be reduced to $[\mathbf{p}]'$; (b) $[\mathbf{x}]' \neq [\mathbf{x}]$ or $[\mathbf{d}]' \neq [\mathbf{d}]$, it is thus not possible to find some $\mathbf{p} \in [\mathbf{p}]$ such that (4.24) is satisfied for all $\mathbf{x} \in [\mathbf{x}]$ and all $\mathbf{d} \in [\mathbf{d}]$.

One can reduce the size of the sets for the state $\mathbf{x}$ and the disturbance $\mathbf{d}$ on which (4.24) has to be verified using the contraction on the negation of this constraint. Consider the negation $\overline{\tau}$ of τ

$$\overline{\tau} \equiv \left(u(\mathbf{x}, \mathbf{p}) \in \overline{\mathcal{A}}\right) \wedge \left(v(\mathbf{x}, \mathbf{p}, \mathbf{d}) \in \overline{\mathcal{B}}\right) \tag{4.29}$$

and a contractor $\mathcal{C}_{\overline{\tau}}$ for this constraint. Assume that after applying $\mathcal{C}_{\overline{\tau}}$ for the boxes $[\mathbf{x}]$, $[\mathbf{p}]$, and $[\mathbf{d}]$, one gets

$$([\mathbf{x}]", [\mathbf{p}]", [\mathbf{d}]") = \mathcal{C}_{\overline{\tau}}([\mathbf{x}], [\mathbf{p}], [\mathbf{d}]). \tag{4.30}$$

From Proposition 3, one knows that

$$\forall\,(\mathbf{x},\mathbf{p},\mathbf{d}) \in ([\mathbf{x}] \times [\mathbf{p}] \times [\mathbf{d}]) \setminus ([\mathbf{x}]'' \times [\mathbf{p}]'' \times [\mathbf{d}]''),\, u(\mathbf{x},\mathbf{p}) \in \mathcal{A} \vee v(\mathbf{x},\mathbf{p},\mathbf{d}) \in \mathcal{B}. \quad (4.31)$$

Indeed, if $[\mathbf{p}]'' = [\mathbf{p}]$, one can focus on the search for some $\mathbf{p} \in [\mathbf{p}]$ satisfying (4.24) by considering only $[\mathbf{x}]'' \times [\mathbf{d}]''$, since for all $(\mathbf{x},\mathbf{d}) \in ([\mathbf{x}] \times [\mathbf{d}]) \setminus ([\mathbf{x}]'' \times [\mathbf{d}]'')$, (4.24) is satisfied for all $\mathbf{p} \in [\mathbf{p}]$, see Figure 4.2 (a).

Now, if $[\mathbf{p}]'' \neq [\mathbf{p}]$, then any $\mathbf{p} \in [\mathbf{p}] \setminus [\mathbf{p}]''$ will satisfy (4.24) for all $(\mathbf{x},\mathbf{d}) \in ([\mathbf{x}] \times [\mathbf{d}])$, see Figure 4.2 (b) satisfying parameter vectors can then be obtained within $[\mathbf{p}] \setminus [\mathbf{p}]''$.

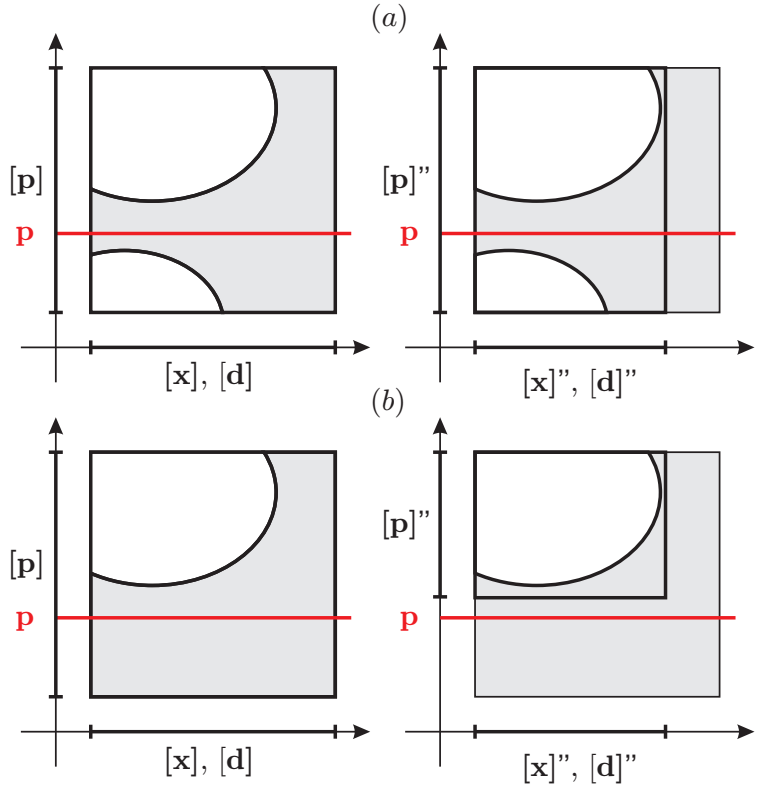


Figure 4.2: Contractions using $\mathcal{C}_{\overline{\tau}}$; the set for which (4.29) is satisfied is in white; figures on the right column have the original bases, the one on the left column the contracted bases. (a) $[\mathbf{x}]'' \neq [\mathbf{x}]$ and/or $[\mathbf{d}]'' \neq [\mathbf{d}]$ and one has only to find some suitable $\mathbf{p} \in [\mathbf{p}]$ such that (4.24) is satisfied for all $\mathbf{x} \in [\mathbf{x}]''$ and all $\mathbf{d} \in [\mathbf{d}]''$; (b) $[\mathbf{p}]'' \neq [\mathbf{p}]$, one may choose any $\mathbf{p} \in [\mathbf{p}] \setminus [\mathbf{p}]''$ (for example the value of $\mathbf{p}$ indicated in red) and (4.24) will hold true for all $(\mathbf{x},\mathbf{d}) \in ([\mathbf{x}] \times [\mathbf{d}])$.

4.2.1.3 Bisection

One is unable to decide whether some $\mathbf{p} \in [\mathbf{p}]$ satisfies (4.21) when

$$[u]([\mathbf{x}],\mathbf{p}) \cap \mathcal{A} \neq \emptyset \text{ and } [u]([\mathbf{x}],\mathbf{p}) \not\subseteq \mathcal{A} \quad (4.32)$$

or when

$$[v]([\mathbf{x}], \mathbf{p}, [\mathbf{d}]) \cap \mathcal{B} \neq \emptyset \text{ and } [v]([\mathbf{x}], \mathbf{p}, [\mathbf{d}]) \not\subseteq \mathcal{B}. \tag{4.33}$$

This situation occurs in two cases. First, when the selected $\mathbf{p}$ does not satisfy (4.21) for all $\mathbf{x} \in [\mathbf{x}]$ and for all $\mathbf{d} \in [\mathbf{d}]$. Second, when inclusion functions introduce some *pessimism*, *i.e.*, they provide an over-approximation of the range of functions over intervals see Section 3.2.

To address both cases, one may perform bisections of $[\mathbf{x}] \times [\mathbf{d}]$ and try to verify (4.21) on the resulting sub-boxes for the *same* $\mathbf{p}$. Bisection allows to isolate subsets of $[\mathbf{x}] \times [\mathbf{d}]$ on which one may show that (4.22) holds true. Bisections also reduce pessimism, and may thus facilitate the verification of (4.21). The bisection of $[\mathbf{x}] \times [\mathbf{d}]$ is performed within CSC as long as the width of the bisected boxes are larger than some $\varepsilon_{\mathrm{x}} > 0$. When CSC is unable to prove (4.21) or (4.22) and when all bisected boxes are smaller than ε_{x}, CSC returns `unknown`.

FPS performs similar bisections on $[\mathbf{p}]$ and stops when the width of all bisected boxes are smaller than $\varepsilon_{\mathrm{p}} > 0$.

4.2.1.4 Composition of constraints

To prove the safety of the dynamical system (4.1), Proposition 4 shows that one has to find some $\mathbf{p} \in [\mathbf{p}]$ such that $\forall \mathbf{x} \in [\mathbf{x}]$, $\forall \mathbf{d} \in [\mathbf{d}]$, $\xi(\mathbf{x}, \mathbf{p}, \mathbf{d})$ holds true. Since $\xi(\mathbf{x}, \mathbf{p}, \mathbf{d})$ consists of the conjunction of three elementary constraints of the form (4.20), validation requires the verification of (4.21) for the *same* $\mathbf{p}$ considering (4.7), (4.8), and (4.9) *simultaneously*. Invalidation may be performed as soon as one is able to prove that one of the constraints (4.7), (4.8), or (4.9) does not hold true using (4.22). Contraction may benefit from the conjunction or disjunction of these constraints, as introduced in (3.24) and (3.25).

4.2.2 CSC-FPS algorithms with contractors

The CSC-FPS algorithm, presented in [JW96] is supplemented with the contractors introduced in Section 4.2.1 to improve its efficiency. No specific contractor is indicated in this section. Nevertheless, the forward-backward contractor used in the experimental part provides good results and is provided directly from the explicit expression of the constraint using tools such as IBEX.

FPS, described in Algorithm 3, searches for some *satisfying* $\mathbf{p} \in [\mathbf{p}]$, *i.e.*, some $\mathbf{p} \in [\mathbf{p}]$ such that $\xi(\mathbf{x}, \mathbf{p}, \mathbf{d})$ introduced in Proposition 4 holds true for all $\mathbf{x} \in [\mathbf{x}]$ and all $\mathbf{d} \in [\mathbf{d}]$. This may require to bisect $[\mathbf{p}]$ into sub-boxes stored in a queue $\mathcal{Q}$, which initial content is $[\mathbf{p}]$.

A sub-box $[\mathbf{p}]_0 \subseteq [\mathbf{p}]$ is extracted from $\mathcal{Q}$ at Line 5. A reduction of $[\mathbf{p}]_0$ is then performed at Line 6 to eliminate values of $\mathbf{p} \in [\mathbf{p}]_0$ which cannot be satisfying. To facilitate contraction *e.g.*, with the classical forward-backward contractor, specific $\mathbf{x} \in [\mathbf{x}]$ are chosen; here only the midpoint $\mathrm{mid}([\mathbf{x}])$ of $[\mathbf{x}]$ is considered.

Algorithm 3 FPS

1: **procedure** FPS(ξ_0, ξ_u, ξ_b, $[\mathbf{p}]$,$[\mathbf{x}]$, $[\mathbf{d}]$) ▷ ξ_0, ξ_u, ξ_b from (4.7), (4.8) and (4.9)
2: queue $\mathcal{Q}$:= $[\mathbf{p}]$
3: flag := `true`
4: **while** $\mathcal{Q} \neq \emptyset$ **do**
5: $[\mathbf{p}]_0$:= dequeue($\mathcal{Q}$) ▷ Reduction of the parameter space using (3.24), (4.25), (4.26)
6: $[\mathbf{p}]'_0 := \mathcal{C}_{\xi_0 \wedge \xi_u}(\text{mid}([\mathbf{x}]), [\mathbf{p}]_0)$ ▷ When $[\mathbf{p}]'_0 = \emptyset$, there is no satisfying $\mathbf{p} \in [\mathbf{p}]_0$
7: **if** $[\mathbf{p}]'_0 = \emptyset$ **then**
8: continue
9: **end if**
10: r_0:= CSC(ξ_0,$[\mathbf{p}]'_0$, $[\mathbf{x}]$,$[\mathbf{d}]$) ▷ Call CSC for each constraint
11: r_u:= CSC(ξ_u,$[\mathbf{p}]'_0$, $[\mathbf{x}]$,$[\mathbf{d}]$)
12: r_b:= CSC(ξ_b,$[\mathbf{p}]'_0$, $[\mathbf{x}]$,$[\mathbf{d}]$)
13: **if** (r_0=`true`)$\wedge$(r_u=`true`)$\wedge$(r_b=`true`) **then**
14: return(`true`,mid($[\mathbf{p}]'_0$) ▷ m($[\mathbf{p}]'_0$) is satisfying if all CSCs hold true
15: **end if**
16: **if** (r_0=`false`)$\vee$(r_u=`false`)$\vee$(r_b=`false`) **then**
17: continue ▷ no solution in $[\mathbf{p}]'_0$ if one CSC holds false
18: **end if**
19: **if** width($[\mathbf{p}]'_0$) $\geq \varepsilon_{\text{p}}$ **then**
20: $([\mathbf{p}]_1, [\mathbf{p}]_2)$:=bisect($[\mathbf{p}]'_0$) ▷ no conclusion for $[\mathbf{p}]'_0$ and large enough to be bisected
21: enqueue($[\mathbf{p}]_1$) in $\mathcal{Q}$
22: enqueue($[\mathbf{p}]_2$) in $\mathcal{Q}$
23: **else**
24: flag:=`unknown` ▷ no decision could be made for $[\mathbf{p}]'_0$ and thus bisect
25: **end if**
26: **end while**
27: **if** flag=`unknown` **then**
28: return(`unknown`,$\emptyset$) ▷ a small $[\mathbf{p}]'_0$ was not explored, impossible to state whether there is no valid solution in $[\mathbf{p}]$
29: **end if**
30: return(`false`,$\emptyset$) ▷ no valid solution in $[\mathbf{p}]$
31: **end procedure**

At Line 7 if $[\mathbf{p}]'_0$ is empty, the next box in $\mathcal{Q}$ has to be explored. Otherwise, at Line 10-12 CSC is called for each constraint t_0, t_u, and t_b to verify whether $\text{m}\left([\mathbf{p}]'_0\right)$, the midpoint of $[\mathbf{p}]'_0$, is satisfying. When all calls of CSC return `true` at Line 13, a barrier function with parameter $\text{m}\left([\mathbf{p}]'_0\right)$ is found. When one of the calls of CSC returns `false` at Line 16, $[\mathbf{p}]'_0$ is proved not to contain any satisfying $\mathbf{p}$. In all other cases, when width($[\mathbf{p}]'_0$), the width of $[\mathbf{p}]'_0$, is larger than ε_{p}, $[\mathbf{p}]'_0$ is bisected and the resulting sub-boxes are stored in $\mathcal{Q}$, see Lines 19-22. When $[\mathbf{p}]'_0$ is too small, even if one was not able to decide whether it contains a satisfying $\mathbf{p}$, it is not further considered to ensure termination of FPS in finite time. The price to be paid in such situations is the impossibility to state whether the initial box $[\mathbf{p}]$ contains some satisfying $\mathbf{p}$. This is done by setting the flag to `unknown` at Line 24. Finally, when $\mathcal{Q} = \emptyset$, no satisfying $\mathbf{p}$ has been found. Whether $[\mathbf{p}]$ may however contain some satisfying $\mathbf{p}$ in an

Algorithm 4 CSC

1: **procedure** CSC(τ,$[\mathbf{p}]_0$,$[\mathbf{x}]$,$[\mathbf{d}]$) ▷ τ is of the form (4.16)
2: stack $S := [\mathbf{x}] \times [\mathbf{d}]$
3: flag:=`true`
4: **while** $S \neq \emptyset$ **do**
5: $[\mathbf{x}]_0 \times [\mathbf{d}]_0$:=pop(S)
6: **if** $[u]([\mathbf{x}]_0, \text{mid}([\mathbf{p}]_0)) \subseteq \mathcal{A} \vee [v]([\mathbf{x}]_0, \text{mid}([\mathbf{p}]_0), [\mathbf{d}]_0) \subseteq \mathcal{B}$ **then**
7: continue ▷ Validation of mid($[\mathbf{p}]_0$) using(4.21)
8: **end if**
9: **if** $[u]\left(\text{m}\left([\mathbf{x}]_0\right), [\mathbf{p}]_0\right) \cap \mathcal{A} = \emptyset \wedge [v]\left(\text{m}\left([\mathbf{x}]_0\right), [\mathbf{p}]_0, \text{m}\left([\mathbf{d}]_0\right)\right) \cap \mathcal{B} = \emptyset$ **then**
10: return(`false`) ▷ Elimination of $[\mathbf{p}]_0$ using (4.22)
11: **end if**
12: $([\mathbf{x}]_0', [\mathbf{p}]_0', [\mathbf{d}]_0') := \mathcal{C}_\tau([\mathbf{x}]_0, [\mathbf{p}]_0, [\mathbf{d}]_0)$ ▷ Reduction of $[\mathbf{p}]_0$ using (4.25)
13: **if** $[\mathbf{x}]_0' \neq [\mathbf{x}]_0 \vee [\mathbf{d}]_0' \neq [\mathbf{d}]_0$ **then**
14: return(`false`) ▷ Prove with Proposition 3 that $[\mathbf{p}]_0$ does not contain any satisfying $\mathbf{p}$
15: **end if**
16: ▷ $[\mathbf{x}]_0 \times [\mathbf{d}]_0$ undetermined and large enough to be bisected
17: **if** $(\text{width}([\mathbf{x}]_0 \times [\mathbf{d}]_0) \geq \varepsilon_{\text{x}})$ **then**
18: $([\mathbf{x}]_0'', [\mathbf{d}]_0'') := \mathcal{C}_{\overline{\tau}}([\mathbf{x}]_0, \text{mid}([\mathbf{p}]_0), [\mathbf{d}]_0)$ ▷ Reduction of the state space using (4.30), (4.31)
19: $([\mathbf{x}]_1 \times [\mathbf{d}]_1, [\mathbf{x}]_2 \times [\mathbf{d}]_2)$:=bisect($[\mathbf{x}]_0'' \times [\mathbf{d}]_0''$) ▷ Bisection
20: push($[\mathbf{x}]_1 \times [\mathbf{d}]_1$) in S
21: push($[\mathbf{x}]_2 \times [\mathbf{d}]_2$)in S
22: **else**
23: flag:=`unknown` ▷ $[\mathbf{x}]_0 \times [\mathbf{d}]_0$ too small to be further explored
24: **end if**
25: **end while**
26: return(flag)
27: **end procedure**

unexplored subbox $[\mathbf{p}]_0'$ depends on the value of flag.

CSC, described in Algorithm 4, determines whether $\text{m}([\mathbf{p}])$ satisfies (4.16) by trying to show that $\tau\left(\mathbf{x}, \text{m}([\mathbf{p}]), \mathbf{d}\right)$ holds true for all $\mathbf{x} \in [\mathbf{x}]$ and all $\mathbf{d} \in [\mathbf{d}]$. Alternatively, CSC may prove that there is no $\mathbf{p} \in [\mathbf{p}]$ satisfying (4.16) for all $\mathbf{x} \in [\mathbf{x}]$ and all $\mathbf{d} \in [\mathbf{d}]$.

For that purpose, due to the pessimism of inclusion functions, it may be necessary to bisect $[\mathbf{x}] \times [\mathbf{d}]$ in sub-boxes stored in a stack $\mathcal{S}$ initialized with $[\mathbf{x}] \times [\mathbf{d}]$.

For each sub-box $[\mathbf{x}]_0 \times [\mathbf{d}]_0 \subseteq [\mathbf{x}] \times [\mathbf{d}]$, CSC determines at Line 7 whether mid($[\mathbf{p}]_0$) is satisfying. CSC tries to prove that $[\mathbf{p}]_0$ does not contain any satisfying $\mathbf{p}$. This is done at Line 10, one applies (4.23) considering some $(\mathbf{x}, \mathbf{d}) \in [\mathbf{x}]_0 \times [\mathbf{d}]_0$, here taken as the midpoint of $[\mathbf{x}]_0 \times [\mathbf{d}]_0$. Second, one applies Proposition 3 at Line 12 using the result of a contractor for τ, as described in (4.27) and (4.28).

When one is not able to conclude and provided that $\text{width}([\mathbf{x}]_0 \times [\mathbf{d}]_0)$ is larger than ε_{x}, some parts of $[\mathbf{x}]_0 \times [\mathbf{d}]_0$ for which mid($[\mathbf{p}]_0$) is satisfying are removed at Line 18, *e.g.*, using the forward-backward contractor, before performing a bisection and storing the resulting

sub-boxes in $\mathcal{S}$. When width($[\mathbf{x}]_0 \times [\mathbf{d}]_0$) is less than ε_x to ensure completion of CSC in a finite time, $[\mathbf{x}]_0 \times [\mathbf{d}]_0$ is not further explored. The price to be paid is an impossibility to determine whether mid($[\mathbf{p}]_0$) is satisfying for all $(\mathbf{x}, \mathbf{d}) \in [\mathbf{x}]_0 \times [\mathbf{d}]_0$. This is indicated by setting flag to `unknown` at Line 23. Nevertheless one may still prove that one $[\mathbf{p}]_0$ does not contain any satisfying $\mathbf{p}$ considering other sub-boxes of $[\mathbf{x}] \times [\mathbf{d}]$.

4.3 Experimental results

This section presents experiments for the characterization of barrier functions. The considered dynamical systems are described first before providing numerical results, comparison of different approaches, and discussions.

4.3.1 Dynamical system descriptions

For the following examples, one provides the dynamics of the system, the constraints g_0 and g_u for the definition of the sets $\mathcal{X}_0$ and $\mathcal{X}_u$, the state space $[\mathbf{x}]$, and the parametric expression of the barrier function. In all cases, the barrier function parameter search space is chosen as $[\mathbf{p}] = [-10, 10]^m$ where m is the number of parameters.

Example 9. *Consider the system*

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 \\ x_1 x_2 - 0.5 x_2^2 \end{pmatrix}$$

with $g_0(\mathbf{x}) = (x_1+1.25)^2+(x_2-1.25)^2-0.05$, $g_u(\mathbf{x}) = (x_1+2.5)^2+(x_2-0.8)^2-0.05$, *and* $[\mathbf{x}] = [-10^3, 0] \times [-10^3, 10^3]$. *The parametric barrier function is* $B(\mathbf{x}, \mathbf{p}) = \frac{p_1 p_2 (x_0+p_3)}{(x_0+p_3)^2+p_2^2} + x_1 + p_4$.

Example 10. *Consider the system from [AKP11]*

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + x_1 x_2 \\ -x_2 \end{pmatrix}$$

with $g_0(\mathbf{x}) = (x_1-1.125)^2+(x_2-0.625)^2-0.0125$, $g_u(\mathbf{x}) = (x_1-0.875)^2+(x_2-0.125)^2-0.0125$, *and* $[\mathbf{x}] = [-100, 100] \times [-100, 100]$. *The parametric barrier function used is* $B(\mathbf{x}, \mathbf{p}) = \ln(p_1 x_1) - \ln(x_2) + p_2 x_2 + p_3$.

Example 11. *Consider the system from [PP05]*

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{x1+x2}{\sqrt{1+(x1+x2)^2}} \end{pmatrix}$$

with $g_0(\mathbf{x}) = {x_1}^2 + {x_2}^2 - 0.5$, $g_u(\mathbf{x}) = (x_1 - 3.5)^2 + (x_2 - 0.5)^2 - 0.5$, *and*$[\mathbf{x}] = [-10^3, 10^3] \times [-100, 100]$. *A quadratic parametric barrier function is chosen* $B(\mathbf{x}, \mathbf{p}) = p_1 x_1^2 + p_2 x_2^2 + p_3 x_1 x_2 + p_4 x_1 + p_5 x_2 + p_6$.

Example 12. *Consider the disturbed system from [PJ04]*

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -x_1 + \frac{d}{3}x_1^3 - x_2 \end{pmatrix}$$

with $g_0(\mathbf{x}) = (x_1 - 1.5)^2 + {x_2}^2 - 0.25$, $g_u(\mathbf{x}) = (x_1 + 0.8)^2 + (x_2 + 1)^2 - 0.25$, $[\mathbf{x}] = [-100, 100] \times [-10, 10]$, *and* $d \in [0.9, 1.1]$. *The parametric barrier function* $B(\mathbf{x}, \mathbf{p}) = p_1 x_1^2 + p_2 x_2^2 + p_3 x_1 x_2 + p_4 x_1 + p_5 x_2 + p_6$ *is considered.*

Example 13. *Consider the system with a limit cycle*

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 + (1 - x_1^2 - x_2^2)x_1 + \ln(x_1^2 + 1) \\ -x_1 + (1 - x_1^2 - x_2^2)x_2 + \ln(x_2^2 + 1) \end{pmatrix}$$

with $g_0(\mathbf{x}) = (x_1 - 1)^2 + (x_2 + 1.5)^2 - 0.05$, $g_u(\mathbf{x}) = (x_1 + 0.6) + (x_2 - 1)^2 - 0.05$, *and* $[\mathbf{x}] = [-10^3, 10^3] \times [-10^3, 10^3]$. *The parametric barrier function used is* $B(\mathbf{x}, \mathbf{p}) = \left(\frac{x_1 + p_1}{p_2}\right)^2 + \left(\frac{x_2 + p_3}{p_4}\right)^2 - 1$.

Example 14. *Consider the Lorenz system from [VČ96]*

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{pmatrix} = \begin{pmatrix} 10(x_2 - x_1) \\ x_1(28 - x_3) - x_2 \\ x_1 x_2 - \frac{8}{3}x_3 \end{pmatrix}$$

with $g_0(\mathbf{x}) = (x_1 + 14.5)^2 + (x_2 + 14.5)^2 + (x_3 - 12.5)^2 - 0.25$, $g_u(\mathbf{x}) = (x_1 + 16.5)^2 + (x_2 + 14.5)^2 + (x_3 - 2.5)^2 - 0.25$, *and* $[\mathbf{x}] = [-20, 20] \times [-20, 0] \times [-20, 20]$. *The considered parametric barrier function is* $B(\mathbf{x}, \mathbf{p}) = p_1 x_1^2 + p_2 x_1 + p_3 x_3 + p_4$.

Example 15. *Consider the system from [PP02]*

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{pmatrix} = \begin{pmatrix} -x_1 + 4x_2 - 6x_3 x_4 \\ -x_1 - x_2 + x_5^3 \\ x_1 x_4 - x_3 + x_4 x_6 \\ x_1 x_3 + x_3 x_6 - x_4^3 \\ -2x_2^3 - x_5 + x_6 \\ -3x_3 x_4 - x_5^3 - x_6 \end{pmatrix}$$

with

$$g_0(\mathbf{x}) = (x_1 - 3.05)^2 + (x_2 - 3.05)^2 + (x_3 - 3.05)^2 + (x_4 - 3.05)^2 + (x_5 - 3.05)^2 + (x_6 - 3.05)^2 - 0.0001,$$
$$g_u(\mathbf{x}) = (x_1 - 7.05)^2 + (x_2 - 3.05)^2 + (x_3 - 7.05)^2 + (x_4 - 7.05)^2 + (x_5 - 7.05)^2 + (x_6 - 7.05)^2 - 0.0001.$$

The considered parametric barrier function is

$$B(\mathbf{x}, \mathbf{p}) = p_1 x_1^2 + p_2 x_2^4 + p_3 x_3^2 + p_4 x_4^2 + p_5 x_5^4 + p_6 x_6^2 + p_7.$$

$[\mathbf{x}] = [0, 10] \times [0, 10] \times [2, 10] \times [0, 10] \times [0, 10] \times [0, 10]$

4.3.2 Experimental conditions and results

CSC-FPS, presented in Section 4.2, has been implemented using the IBEX library [ATNC12, CJ09]. The selection of candidate barrier functions is performed choosing polynomials with increasing degree, except for Examples 9, 10, and 12, where parametric functions taken from [mat15, AKP11] are considered.

For each example, the computing time to get a valid barrier function and the number of bisections of the search box $[\mathbf{p}]$ are provided. Table 4.1 summarizes the results for the versions of CSC-FPS with and without contractors. As in [JW96], we choose $\varepsilon_{\mathrm{x}} = 10^{-1}$ and $\varepsilon_{\mathrm{p}} = 10^{-5}$. CSC is very intensively called. Further reducing $\varepsilon_{\mathrm{x}} = 10^{-1}$ improves the performance of CSC which may less frequently conclude unknown, but the price to be paid is an increase of the complexity cost. Computations were done on an Intel core 1.7 GHz processor with 8 GB of RAM. If after 30 minutes of computations no valid barrier function has been found, the search is stopped. This is denoted by T.O. (time out) in Table 4.1. Moreover, for Examples 9, 12, and 13, graphical representation of the computed barrier functions are provided. In Figures 4.3, 4.4, and 4.5, $\mathcal{X}_0$ is in green, $\mathcal{X}_u$ is in red, the bold line is the barrier function and some trajectories starting from $\mathcal{X}_0$ are also represented.

The results in Table 4.1 show the importance of contractors which are beneficial in all cases. Thanks to contractors, valid barrier functions were obtained for all examples, which is not the case employing the original version of CSC-FPS proposed in [JW96]. In theory, both FPS and CSC are of exponential complexity in the dimension m of the parameter space and n of the state space. In practice, contractors allow, on the considered examples, to get solutions in a reasonable amount of time, even for systems with a larger number of states and parameters.

The search for barrier functions using the relaxed version (4.14) of the constraint (4.12) as in [PJ04], has also been performed using the version of our algorithms with contractors

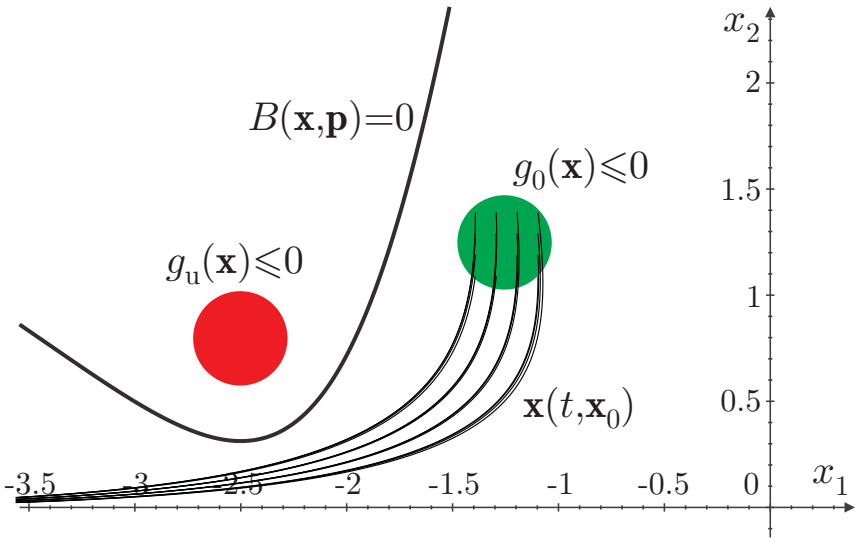


Figure 4.3: Results for Example 9.

considering the same parametric barrier functions. We were not able to find a valid barrier function in less than 30 minutes for most of the examples. This shows the detrimental effect of the relaxation (4.14) on the search technique.

Some examples were also addressed using RSolver, which is a tool to solve some classes of QCSP [Rat06]. Nevertheless, this requires some modifications of the examples, since RSolver does not address dynamics or constraints involving divisions [Rat07]. Moreover, the type of constraints processed by RSolver, for problems such as (4.20), allows only parametric barrier functions that are linear in the parameters. Outer-approximating boxes for the initial and the unsafe regions $\mathcal{X}_0$, $\mathcal{X}_u$ defined by g_0 and g_u are given to Rsolver. As a consequence, Examples 9, 10 and 11 could not be considered and only Examples 12, 13, and 14 were tested, Rsolver was able to find a satisfying barrier function only for 14 in less than 1s. RSolver was unable to find a solution for Examples 12 and 13. Rsolver is well designed for problems with parametric barriers linear in the parameters, but has difficulties for non-linear problem and non-convex constraints such as (4.12).

The set of examples were also produced with dReal [GKC13], a SAT-modulo-theory solver [BT14] based on interval analysis to handle first-order logic formulas over the real numbers. In the tested version (3.15.11.03) of dReal, the support for QCSP such as (4.20) does not benefit from contractors. A randomization heuristic is used in the search of satisfying parameter vectors. This leads to results obtained in a amount of time varying with the runs. A precision parameter of 0.001 (a tentative to have a fair comparison with respect to values of ε_{X} and ε_{P} used in our algorithm) was employed. Due to the non-deterministic behavior of dReal, each example has been run 10 times and we report minimal and maximal values for the execution time and the number of bisections. When no solution is found in less than 30 minutes, T.O. is indicated.

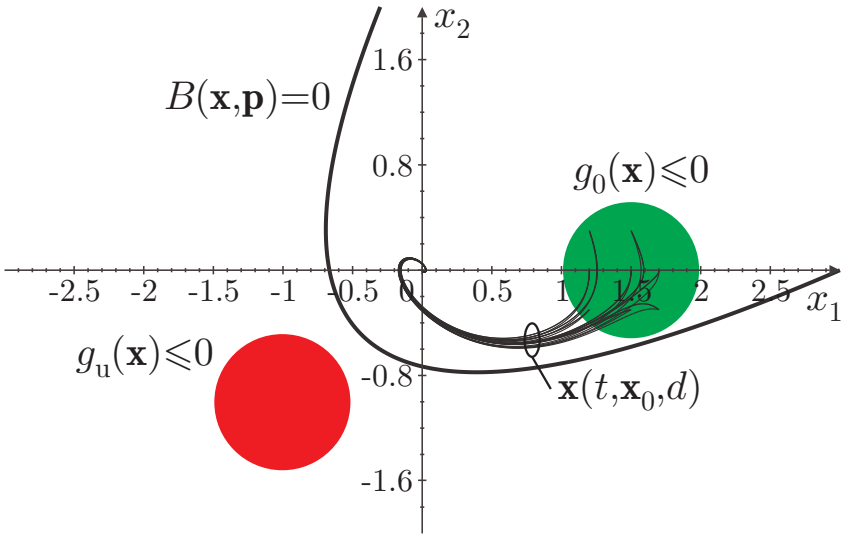


Figure 4.4: Results for Example 12 with various values of the disturbance d.

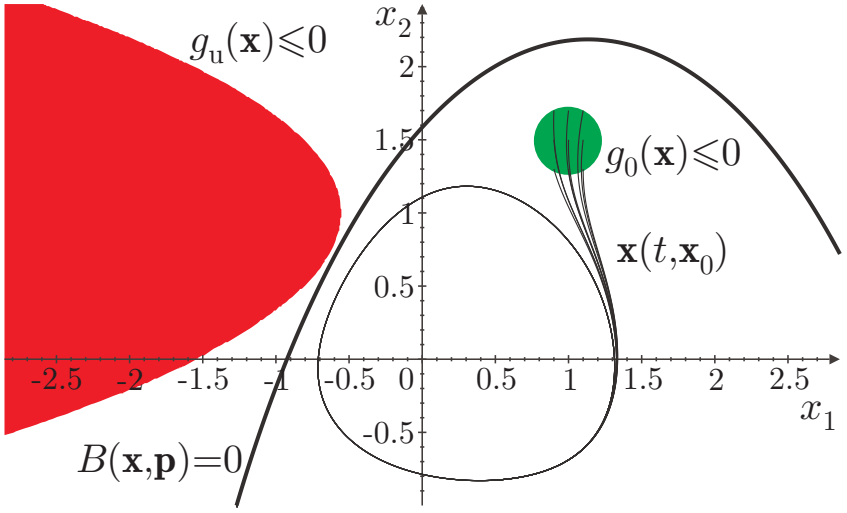


Figure 4.5: Results for Example 13.

Table 4.1 shows that dReal can handle all considered problems despite the absence of contractors. In particular, one observes that randomization speed up the search for valid parameters, as in the case of Examples 2, 5, 6 and 7. Nevertheless, as a consequence, the overall algorithm has a varying execution time. The absence of contractors may increase the parameter search time, see Examples 1, 3, and 4. Combining randomized methods and contractor programming may be beneficial to efficiently solve QCSP.

4.4 Partial conclusion

This section presents a new method to find parametric barrier functions for nonlinear continuous-time perturbed dynamical systems. The proposed technique addresses the design of

Table 4.1: Results for CSC-FPS without and with contractors, for CSC-FPS with contractors and randomization, and for dReal; for the two last cases, the minimum and maximum times and number of bisections over 10 independent runs are provided

			CSC-FPS, wo. rand.				CSC-FPS w. contr. and rand.				dReal			
			wo. contr.		w. contr.		min		max		min		max	
Ex.	n	m	time	bis.	time	bis.	time	bis.	time	bis.	time	bis.	time	bis.
5	2	4	25.9	4520	11.7	4453	10.4	4011	13.7	5397	72.6	24962	974.6	56571
6	2	3	TO	/	1.17	159	0.76	109	1.19	173	0.02	19	0.2	180
7	2	6	857	20388	1.14	6	0.35	3	1.16	5	37.7	112	TO	/
8	2	6	189	14733	5.70	435	4.08	316	6.16	485	225.1	76	TO	/
9	2	4	TO	/	63.7	4072	58.4	3639	64.2	4290	0.04	78	0.1	106
10	3	4	115	1753	15.9	47	3.20	31	16.5	59	0.04	54	0.08	55
11	6	7	530	67600	0.14	261	0.14	281	0.14	446	0.2	88	0.2	114

quite general barrier functions for continuous-time systems described by nonlinear dynamics that are also general. The search for barrier functions is formulated as an interval quantified constraint satisfaction problem. A branch-and-prune algorithm proposed in [JW96] has been supplemented with contractors to address this problem. Contractors are instrumental in solving problems with large number of parameters. The proposed approach can thus find barrier functions for a large class of possibly perturbed dynamical systems.

Alternative techniques based on RSolver or dReal may be significantly more efficient for some specific classes of problems where the parameters appear linearly in the parametric barrier functions. A combination of RSolver or dReal and our approach may be useful to improve the global efficiency of barrier function characterization.

Future work will be dedicated to the search for the class of parametric barrier functions to consider. This may be done by exploring a library of candidate barrier functions. In our approach rejection of a candidate function occurs mainly after a timeout. Even if contractors aiming at eliminating some parts of the parameter space were defined, their efficiency is limited. Better contractors for that purpose may be very helpful.

Chapter 5

Barrier functions for hybrid dynamical systems

5.1 Introduction

Hybrid dynamical models are used to represent systems with both digital and analog components. Many computer-controlled systems can be described by hybrid models, *e.g.*, the antilock braking system in a car or the automatic pilot in a plane. Hybrid automata, presented in [ACH+95], constitute the class of hybrid dynamical models considered in this chapter. A hybrid automaton consists of a finite state machine (FSM) with a finite set of continuous variables whose evolution is described by a set of ordinary differential equations.

The aim of this chapter is to extend the results obtained in Chapter 4 to systems described by hybrid dynamical models which are hybrid automata. Section 5.2 introduces some notations and formulates the barrier design problem for hybrid automata. Section 5.3 formulates the parametric barrier design problem as a constraint satisfaction problem and describes three possible design solutions involving CSC-FPS. Section 5.4 describes the incremental design solution, which is better suited for models with several locations. Section 5.5 provides some experimental results, before drawing some conclusions in Section 5.6.

5.2 Notations and Problem Formulation

5.2.1 Hybrid Automata

A hybrid automaton, as introduced in [ACH+95], consists of the tuple

$$\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho), \tag{5.1}$$

where

- $\mathcal{X} \subseteq \mathbb{R}^n$ is the continuous state space,
- $\mathcal{L}$ is a finite set of locations,
- $\mathcal{X}_0 : \mathcal{L} \to \mathcal{P}(\mathcal{X})$ provides a subset $\mathcal{X}_0(\ell) \subseteq \mathcal{X}$ containing the set of possible initial state values for each location $\ell \in \mathcal{L}$,
- $\mathcal{I} : \mathcal{L} \to \mathcal{P}(\mathcal{X})$ provides the invariant set $\mathcal{I}(\ell) \subseteq \mathcal{X}$ for each $\ell \in \mathcal{L}$, *i.e.*, the subset in which the state of the system is supposed to remain when the hybrid automata is in location ℓ,
- $f_\ell : \mathcal{X} \times \mathcal{D}_\ell \to \mathbb{R}^n$ is a vector field for each $\ell \in \mathcal{L}$ describing the continuous-time state equation of the hybrid model in location ℓ with

$$\dot{\mathbf{x}} = f_\ell(\mathbf{x}, \mathbf{d}) \tag{5.2}$$

  and $\mathbf{d} \in \mathcal{D}_\ell$ a disturbance belonging to the compact set $\mathcal{D}_\ell \subset \mathbb{R}^{n_\mathrm{d}}$,
- $\Gamma : \mathcal{L} \times \mathcal{L} \to \mathcal{P}(\mathcal{X})$ provides the subsets $\Gamma(\ell, \ell') \subseteq \mathcal{X}$ for each pair of locations (ℓ, ℓ') leading to a transition of the discrete part of the state from location ℓ to location ℓ' when the continuous part of the dynamics reaches $\Gamma(\ell, \ell')$; $\Gamma(\ell, \ell') = \emptyset$ when no transition is possible from ℓ to ℓ',
- $\rho_{\ell,\ell'} : \Gamma(\ell, \ell') \to \mathcal{I}(\ell')$ is the reset map when a transition occurs from location ℓ to location ℓ'.

The state $\mathbf{x}_\mathrm{H} = (\ell, \mathbf{x})$ of the hybrid automaton $\mathcal{H}$ consists of the pair where $\ell \in \mathcal{L}$ and $\mathbf{x} \in \mathcal{X}$. When the location of the hybrid automaton is ℓ, the continuous part of the model evolves according to (5.2) until $\mathbf{x}(t) \in \Gamma(\ell, \ell')$ for some ℓ' at time t_i. In that case, a transition occurs at time t_i and the new location is ℓ'. The continuous part of the state at time t_i is initialized at $\mathbf{x}(t_i) = \lim_{t \to t_i^-} \rho_{\ell,\ell'}(\mathbf{x}(t))$. For each location ℓ, one assumes that $\mathbf{d}$ is bounded. Moreover, the value of $\mathbf{d}$ is assumed to change only when the location changes. Thus $\mathbf{d}(t)$ is a piecewise constant right-continuous function.

Let $\mathcal{X}_\mathrm{u} : \mathcal{L} \to \mathcal{P}(\mathcal{X})$ be a set-valued map providing the *unsafe* subset $\mathcal{X}_\mathrm{u}(\ell) \subseteq \mathcal{X}$ of each location $\ell \in \mathcal{L}$. $\mathcal{X}_\mathrm{u}(\ell)$ is the part of the continuous state space which should not be reached by the hybrid system when it is in location ℓ. When there is no unsafe subset in location ℓ, $\mathcal{X}_\mathrm{u}(\ell) = \emptyset$.

The following two-tank example, taken from [RS07] illustrates the various components of an hybrid automaton.

Example 16. *(2-TANK) The hybrid model of a 2-tank system has a continuous component of the state-space of dimension $n = 2$. It consists of 2 locations. The flow for each location*

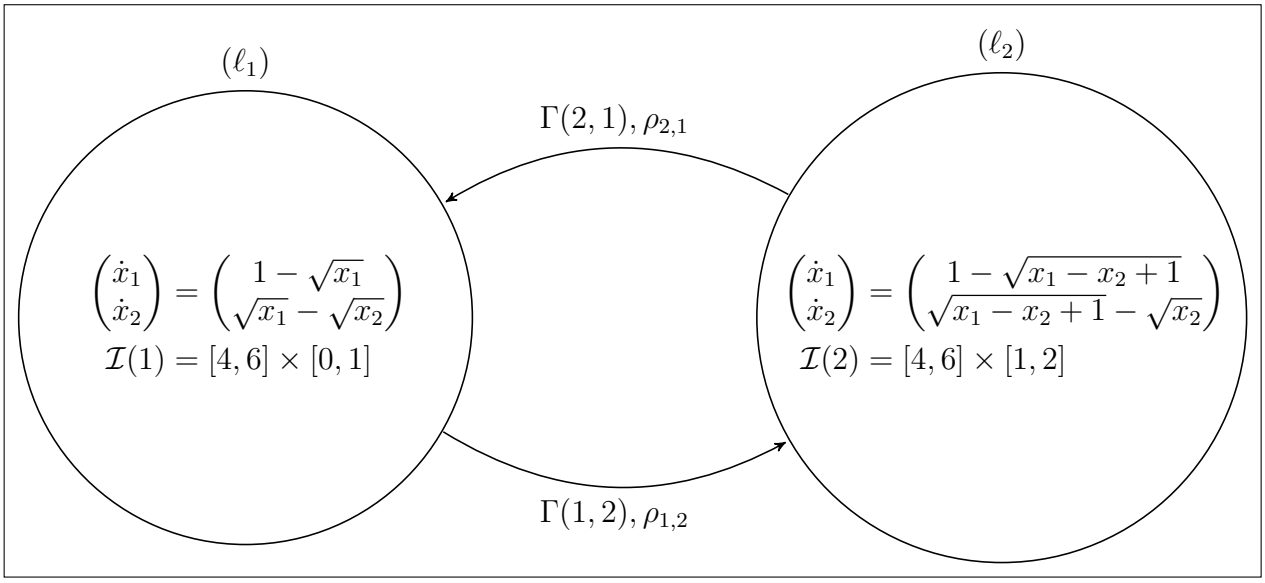


Figure 5.1: Hybrid automata of the 2 Tank

is described by

$$f_1\begin{pmatrix}x_1\\ x_2\end{pmatrix} = \begin{pmatrix}1-\sqrt{x_1}\\ \sqrt{x_1}-\sqrt{x_2}\end{pmatrix}, \qquad f_2\begin{pmatrix}x_1\\ x_2\end{pmatrix} = \begin{pmatrix}1-\sqrt{x_1-x_2+1}\\ \sqrt{x_1-x_2+1}-\sqrt{x_2}\end{pmatrix}$$

The other parts of the hybrid automaton are:

- *Initial conditions:* $\mathcal{X}_0(1) = [5.25, 5.75] \times [0, 0.5]$ *and* $\mathcal{X}_0(2) = [4,6] \times [1,1]$
- *Unsafe regions:* $\mathcal{X}_u(1) = [4, 4.5] \times [0, 0.5]$ *and* $\mathcal{X}_u(2) = \emptyset$
- *Invariants:* $\mathcal{I}(1) = [4,6] \times [0,1]$ *and* $\mathcal{I}(2) = [4,6] \times [1,2]$
- *Guards and resets*
  - $\Gamma(1,2) = [4,6] \times [0,99,1]$ *and* $\rho_{1,2}\begin{pmatrix}x_1\\ x_2\end{pmatrix} = \begin{pmatrix}x_1\\ 1\end{pmatrix}$
  - $\Gamma(2,1) = \emptyset$ *and* $\rho_{2,1}\begin{pmatrix}x_1\\ x_2\end{pmatrix} = \begin{pmatrix}x_1\\ x_2\end{pmatrix}$

A trajectory of the hybrid automata $\mathcal{H}$ represents a time evolution of the hybrid state $\mathbf{x}_{\mathrm{H}} = (\ell, \mathbf{x})$ over some time interval $[0, T]$, where $T > 0$ is potentially infinite. Let $\Pi(\ell_0)$ be the set of trajectories of $\mathcal{H}$ starting from some initial location ℓ_0 and with initial value $\mathbf{x}_0 \in \mathcal{X}_0(\ell_0)$ of the continuous part of the state. The set of all possible trajectories is

$$\Pi = \bigcup_{\ell \in \mathcal{L}} \Pi(\ell). \tag{5.3}$$

Assume that for each trajectory $\mathbf{x}_{\mathrm{H}} \in \Pi$, there exists a countable set of time instants $0 = t_0 < t_1 < t_2 < \dots$ such that

- in all time intervals $[t_i, t_{i+1}[$ between two transitions, the continuous part $\mathbf{x}$ of the hybrid state $\mathbf{x}_{\mathrm{H}}$ evolves according to

$$\dot{\mathbf{x}} = f_{\ell(t_i)}(\mathbf{x}, \mathbf{d}(t_i)) \tag{5.4}$$

  for some $\mathbf{d}(t_i) \in \mathcal{D}$, with $\mathbf{x}(t_i) \in \mathcal{I}(\ell(t_i))$ given by (5.6) and $\ell(t) = \ell(t_i)$.

- at all time instants t_i, one has

$$\lim_{t \to t_i^-} \mathbf{x}(t) \in \mathcal{I}(\ell(t)) \cap \Gamma(\ell(t), \ell(t_i)) \tag{5.5}$$

  and

$$\mathbf{x}(t_i) = \lim_{t \to t_i^-} \rho_{\ell(t),\ell(t_i)}(\mathbf{x}(t)). \tag{5.6}$$

Definition 6. *The hybrid automaton $\mathcal{H}$ is safe if for all trajectories $\mathbf{x}_H \in \Pi$, there is no $t \in [0, T]$ such that*

$$\mathbf{x}(t) \in \mathcal{X}_u(\ell(t)). \tag{5.7}$$

Proving (5.7) is hard since it has to be verified for all the trajectories $\mathbf{x}_{\mathrm{H}}(t)$, whatever the initial conditions $\ell_0 \in \mathcal{L}$ and $\mathbf{x}_0 \in \mathcal{X}_0(\ell_0)$ and the realizations of the noise $\mathbf{d}(t)$.

5.2.2 Barrier Functions

Determining whether a hybrid automaton is safe is difficult. To address this problem, one considers, as in Chapter 4, the barrier certificate approach introduced in [PJ04]. This method does not require the explicit computation of the set Π. This approach aims at finding, for every location $\ell \in \mathcal{L}$, a barrier function β_ℓ that separates the set of trajectories Π from the unsafe regions $\mathcal{X}_u(\ell)$.

Theorem 4 ([PJ04])**.** *Consider the hybrid automaton $\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho)$. Assume that there exist a family of differentiable functions $\beta_\ell(\mathbf{x})$, $\ell \in \mathcal{L}$ such that, for all pairs $(\ell, \ell') \in \mathcal{L}^2$ with $\ell \neq \ell'$, one has*

$$\beta_\ell(\mathbf{x}) \leqslant 0 \quad \forall \mathbf{x} \in \mathcal{X}_0(\ell) \tag{5.8}$$

$$\beta_\ell(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \mathcal{X}_u(\ell) \tag{5.9}$$

$$\beta_\ell(\mathbf{x}) = 0 \implies \frac{\partial \beta_\ell(\mathbf{x})}{\partial \mathbf{x}} f_\ell(\mathbf{x}, \mathbf{d}) < 0 \quad \forall \mathbf{x} \in \mathcal{I}(\ell)\, \forall \mathbf{d} \in \mathcal{D}_\ell \tag{5.10}$$

$$\beta_\ell(\mathbf{x}) \leqslant 0 \implies \beta_{\ell'}(\rho_{\ell,\ell'}(\mathbf{x})) \leqslant 0 \quad \forall \mathbf{x} \in \Gamma(\ell, \ell') \tag{5.11}$$

then the system $\mathcal{H}$ is safe.

A hybrid automaton is safe when one is able to exhibit barrier functions $\beta_\ell(\mathbf{x})$, for $\ell \in \mathcal{L}$ satisfying the various properties introduced in Theorem 4. According to (5.8), β_ℓ has to be negative for all $\mathbf{x} \in \mathcal{X}_0(\ell)$. From (5.10), β_ℓ has to remain negative during a continuous evolution since the Lie derivative of β_ℓ is negative. Moreover, if (5.11) is satisfied, then β_ℓ remains negative when a transition occurs. As a consequence, all trajectories included in the set $\mathcal{I}(\ell)$ are such that $\beta_\ell(\mathbf{x}) \leqslant 0$, $\ell \in \mathcal{L}$. Finally since, according to (5.9), β_ℓ is positive for all $x \in \mathcal{X}_{\mathbf{u}}(\ell)$, no trajectory will reach an unsafe region.

5.3 Characterization of parametric barrier functions

Theorem 4 is not constructive. The search of the functions β_ℓ can thus be challenging since it is on a functional space. To address this problem, one can define parametric functions $\beta_\ell(\mathbf{x}, \mathbf{p}_\ell)$ and search for some value of the parameter vectors $\mathbf{p}_\ell$, $\ell \in \mathcal{L}$ satisfying Theorem 4. If for some $\ell \in \mathcal{L}$, one fails to find such $\mathbf{p}_\ell$, this does not mean that the system is not safe one has to look for other parametric function β_ℓ.

In the following, an approach to find barrier functions that satisfy Theorem 4 is presented. As in Chapter 4, it is formulated as a constraints satisfaction problem and solved using tools from interval analysis. Nevertheless, the fact that here hybrid models have to be considered requires several adaptations of the approach proposed in Chapter 4.

5.3.1 Constraint satisfaction problem

Assume that there exists for each location $\ell \in \mathcal{L}$ some functions $g_0 : \mathcal{L} \times \mathcal{X} \to \mathbb{R}$, $g_{\mathrm{u}} : \mathcal{L} \times \mathcal{X} \to \mathbb{R}$, $g_\Gamma : \mathcal{L} \times \mathcal{L} \times \mathcal{X} \to \mathbb{R}$ and $g_{\mathcal{I}} : \mathcal{L} \times \mathcal{X} \to \mathbb{R}$ such that

$$\begin{aligned}
\mathcal{X}_0(\ell) &= \{\mathbf{x} \in \mathcal{X} \mid g_0(\ell, \mathbf{x}) \leqslant 0\},\\
\mathcal{X}_{\mathrm{u}}(\ell) &= \{\mathbf{x} \in \mathcal{X} \mid g_{\mathrm{u}}(\ell, \mathbf{x}) \leqslant 0\},\\
\Gamma(\ell, \ell') &= \{\mathbf{x} \in \mathcal{X} \mid g_\Gamma(\ell, \ell', \mathbf{x}) \leqslant 0\},\\
\mathcal{I}(\ell) &= \{\mathbf{x} \in \mathcal{X} \mid g_{\mathcal{I}}(\ell, \mathbf{x}) \leqslant 0\}.
\end{aligned}$$

For each location the set of perturbations $\mathcal{D}_\ell$ is represented by a box $[\mathbf{d}]_\ell$. The state space $\mathcal{X}$ is also represented by some sufficiently large $[\mathbf{x}]$. For each location ℓ, assume that the search box for the parameter $\mathbf{p}_\ell$ of $\beta_\ell(\mathbf{x}, \mathbf{p}_\ell)$ is $[\mathbf{p}]_\ell$.

With the previous notations, one may reformulate Theorem 4 as follows.

Proposition 5. *Consider a hybrid system described by $\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho)$. Assume there exists a differentiable function $\beta_\ell(\mathbf{x}, \mathbf{p})$ which satisfies*

$\forall \ell \in \mathcal{L}$, $\exists \mathbf{p}_\ell \in [\mathbf{p}]_\ell$, $\forall \mathbf{x} \in [\mathbf{x}]$, $\forall \mathbf{d} \in [\mathbf{d}]_\ell$

$$g_0(\ell, \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \mathbf{p}_\ell) \leqslant 0, \tag{5.12}$$

$$g_u(\ell, \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \mathbf{p}_\ell) > 0, \tag{5.13}$$

$$g_{\mathcal{I}}(\ell, \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \mathbf{p}_\ell) \neq 0 \vee \frac{\partial \beta_\ell(\mathbf{x}, \mathbf{p}_\ell)}{\partial \mathbf{x}} f_\ell(\mathbf{x}, \mathbf{d}) < 0, \tag{5.14}$$

and $\forall \ell' \in \mathcal{L}$, *with* $\ell' \neq \ell$,

$$g_\Gamma(\ell, \ell', \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \mathbf{p}_\ell) > 0 \vee \beta_{\ell'}(\rho_{\ell,\ell'}(\mathbf{x}), \mathbf{p}_{\ell'}) \leqslant 0, \tag{5.15}$$

then the system $\mathcal{H}$ *is safe.*

Proposition 5 provides a set of constraints that have to be satisfied by the parametric barrier functions to ensure the safety of $\mathcal{H}$.

Proof. For Constraints (5.12) to (5.14), see Chapter 4, page 49. Constraint (5.15) may be rewritten as

$$g_\Gamma(\ell, \ell', \mathbf{x}) > 0 \implies (\beta_\ell(\mathbf{x}, \mathbf{p}_\ell) \leq 0 \implies \beta_{\ell'}(\rho_{\ell,\ell'}(\mathbf{x}), \mathbf{p}_{\ell'}) \leq 0) \quad \forall x \in [\mathbf{x}]$$

Since $\Gamma(\ell, \ell') = \{\mathbf{x} \in \mathcal{X} \,|\, g_\Gamma(\ell, \ell', \mathbf{x}) \leq 0\}$ and $\Gamma(\ell, \ell') \subseteq [\mathbf{x}]$, then Constraint (5.15) is equivalent to (5.11). □

5.3.2 CSC-FPS

To prove the safety of the hybrid model $\mathcal{H}$, once parametric barrier functions $\beta_\ell(\mathbf{x}, \mathbf{p})$ have been chosen for each $\ell \in \mathcal{L}$, one has to find some $\mathbf{p}_\ell \in [\mathbf{p}]_\ell$ such that $\beta_\ell(\mathbf{x}, \mathbf{p}_\ell)$ satisfies the conditions of Proposition 5.

In Chapter 4, the *Computable Sufficient Conditions-Feasible Point Searcher* (CSC-FPS) algorithm introduced in [JW96] has been adapted and supplemented with contractors [CJ09] to solve quantified constraints of the form

$$\exists \mathbf{p} \in [\mathbf{p}], \forall \mathbf{x} \in [\mathbf{x}], \forall \mathbf{d} \in [\mathbf{d}], \quad \tau = \begin{cases} g_0(\mathbf{x}) > 0 \vee \beta(\mathbf{x}, \mathbf{p}) \leqslant 0 \\ g_{\mathrm{u}}(\mathbf{x}) > 0 \vee \beta(\mathbf{x}, \mathbf{p}) > 0 \\ \beta(\mathbf{x}, \mathbf{p}) \neq 0 \vee \frac{\partial \beta(\mathbf{x}, \mathbf{p})}{\partial x} f(\mathbf{x}, \mathbf{d}) < 0 \end{cases} \tag{5.16}$$

In Chapter 4, the CSC-FPS algorithm takes as input the constraint τ defined in (5.16), the boxes $[\mathbf{x}]$ and $[\mathbf{d}]$ and a search queue $\mathcal{Q}$ of boxes in the parameter space containing initially

only $[\mathbf{p}]$. CSC-FPS returns a vector $\widehat{\mathbf{p}}$, solution to (5.16) as well as the updated content of the search queue once $\widehat{\mathbf{p}}$ has been found. One gets thus

$$(\eta, \widehat{\mathbf{p}}, \mathcal{Q}') = \text{CSC-FPS}\,(\tau, \mathcal{Q}, [\mathbf{x}], [\mathbf{d}], \varepsilon_{\text{F}}, \varepsilon_{\text{C}},)\,. \tag{5.17}$$

The flag η indicates whether a satisfying $\mathbf{p}$ has been found. When η holds true, CSC-FPS has been able to find a value $\widehat{\mathbf{p}}$ satisfying (5.16). The list $\mathcal{Q}'$ contains boxes included in $[\mathbf{p}]$ which were not yet explored by CSC-FPS. When η holds false, one is able to prove that there is no $\mathbf{p} \in [\mathbf{p}]$ such that (5.16) is satisfied. In this case, $\mathcal{Q}'$ is empty. When CSC-FPS is unable to provide any satisfying solution, without being able to prove that there is no solution in $[\mathbf{p}]$, η is set to unknown, with an empty $\mathcal{Q}'$. ε_{F} and ε_{C} are two precision parameters that limit the search complexity.

CSC-FPS consists of two procedures, namely CSC and FPS. The CSC procedure aims at verifying that the constraints (5.16) are satisfied for some $\mathbf{p}$ chosen in a *finite* set of candidate values from some $[\mathbf{p}]_0 \subset [\mathbf{p}]$, for all values of $\mathbf{x} \in [\mathbf{x}]$ and $\mathbf{d} \in [\mathbf{d}]$. It may be written as

$$(\eta_{\text{C}}, \widehat{\mathbf{p}}) = \text{CSC}\,(\tau, [\mathbf{x}], [\mathbf{d}]\,, [\mathbf{p}]_0\,, \varepsilon_{\text{C}})\,. \tag{5.18}$$

When η_{C} holds true, then $\widehat{\mathbf{p}} \in [\mathbf{p}]_0$ satisfies (5.16). If η_{C} holds false, one has been able to prove that there is no $\widehat{\mathbf{p}} \in [\mathbf{p}]_0$ such that (5.16) is satisfied. Finally when η_{C} is set to unknown, no conclusion can be made. The parameter ε_{C} determines the maximum width of boxes bisected in CSC.

FPS explores the search list $\mathcal{Q}$ in the parameter space. $\mathcal{Q}$ contains initially only $[\mathbf{p}]$. For each box $[\mathbf{p}]_0$ taken out of $\mathcal{Q}$, FPS tries to eliminate parts of $[\mathbf{p}]_0$ which can be proved not to contain any value $\mathbf{p}$ satisfying (5.16). Then, if the reduced box $[\mathbf{p}]'_0 \subset [\mathbf{p}]_0$ is not empty, FPS calls CSC on $[\mathbf{p}]'_0$. Three cases are considered, depending on the value of η_{C} provided by CSC. If η_{C} holds false, $[\mathbf{p}]'_0$ is discarded and FPS continues testing other boxes in $\mathcal{Q}$. If η_{C} holds true, a solution $\widehat{\mathbf{p}}$ has been found. This solution is provided by the CSC-FPS algorithm. If η_{C} is unknown and provided that the width of $[\mathbf{p}]'_0$ is larger than some precision parameter ε_{p}, then $[\mathbf{p}]'_0$ is bisected and the two resulting boxes are put in $\mathcal{Q}$. If $[\mathbf{p}]'_0$ is too small, it is discarded. If such situation occurs, η cannot hold false anymore. If η_{C} is false then the box $[\mathbf{p}]'_0$ is no more considered and the search continues on the remaining boxes in $\mathcal{Q}$.

Extending CSC-FPS to the design of barrier functions for hybrid systems is challenging due to the constraints (5.15) which relates several $\mathbf{p}_\ell$ simultaneously.

Three methods are considered for the search for $\mathbf{p}_\ell$, $\ell \in \mathcal{L}$ that satisfy the constraints of Proposition 5.

5.3.3 Independent design for each location

With this approach, for each location $\ell \in \mathcal{L}$, CSC-FPS is used to find a value $\widehat{\mathbf{p}}_\ell$ of the vector of parameters that satisfies the constraints (5.12), (5.13), and (5.14) involving $\mathbf{p}_\ell$ only. Then, one has to verify that the obtained parameters $\widehat{\mathbf{p}}_\ell$, $\ell \in \mathcal{L}$, satisfy (5.15) for all possible transitions.

The main difference with the CSC-FPS found in Chapter 4 is that it has to take into account conjunctions of constraints of the form $\bigvee_{i=1}^{n} u(\ell, \mathbf{x}, \mathbf{p})$. Once $\widehat{\mathbf{p}}_\ell$ has been obtained for all $\ell \in \mathcal{L}$, one has to check the validity of the constraints

$$\forall \mathbf{x} \in [\mathbf{x}] \quad \tau_{\ell,\ell'} = g_\Gamma(\ell, \ell', \mathbf{x}) > 0 \vee \beta_\ell(\mathbf{x}, \widehat{\mathbf{p}}_\ell) > 0 \vee \beta_{\ell'}(\rho_{\ell,\ell'}(\mathbf{x}), \widehat{\mathbf{p}}_{\ell'}) \leqslant 0 \tag{5.19}$$

$\forall (\ell, \ell') \in \mathcal{L} \times \mathcal{L}$, $(\ell \neq \ell')$ with the help of CSC. If the constraints (5.19) are all satisfied, then one has found satisfying $\widehat{\mathbf{p}}_\ell$, $\ell \in \mathcal{L}$, and the system $\mathcal{H}$ is proven to be safe. When this method works the complexity is linear in the number of locations. Nevertheless, when it fails there is no obvious strategy to find other sets of parameters that could satisfy all the constraints simultaneously.

5.3.4 Joint design for all locations

In this approach all the constraints of Proposition 5 are considered simultaneously. The search space for the parameter vectors is $[\mathbf{p}] = [\mathbf{p}]_0 \times ... \times [\mathbf{p}]_{|\mathcal{L}|}$. CSC-FPS takes as input the conjunction of all constraints of Proposition 5. Due to the exponential complexity of CSC-FPS in the dimension of the parameter space, this approach can only be used for hybrid systems with a very small number of locations.

5.3.5 Incremental design

This approach considers the graph with nodes representing the locations and oriented edges representing the possible transitions between locations of the hybrid automaton. This graph is assumed to be connected and is explored in an incremental way.

Each location ℓ is treated one by one until all locations have been visited. For each location ℓ, a valid vector of parameters $\widehat{\mathbf{p}}_\ell$ is searched for the constraints (5.12), (5.13), and (5.14). The transition constraints (5.11) are then considered for the current location ℓ and all the previously explored locations. In case a transition constraint cannot be satisfied, a backtracking procedure is engaged. The idea is to search for an other valid vector of parameters $\widehat{\mathbf{p}}_{\ell'}$ for a previously visited location ℓ' before going back to Location ℓ. A complete description of this approach is given in Section 5.4.

5.4 Incremental design algorithm: CSC-FPS-Hyb

The incremental design algorithm CSC-FPS-Hyb manages a list $\mathcal{V}$ of locations which have already been visited. It also manages a working list $\mathcal{W}$ of pairs $(\widehat{\mathbf{p}}_\ell, \mathcal{Q}_\ell)$ containing the already found valid parameter vectors and for each parameter vector, the part of the search space remaining to explore. Since $\mathbf{p}_\ell$ has been proved to satisfy constraints associated to transitions between location in $\mathcal{V}$ only, if there exists a transition to a location $\ell' \notin \mathcal{V}$ such that $\mathbf{p}_\ell$ is not satisfying, one may avoid exploring the whole search space for the parameters and continue a search in the set of not explored parameters, stored in $\mathcal{Q}_\ell$.

5.4.1 Generic algorithm

For $\ell = 1$, CSC-FPS-Hyb tries to find a valid $\widehat{\mathbf{p}}_1$ satisfying the constraints (5.12), (5.13), and (5.14). If no satisfying parameters can be found then the algorithm stops. Otherwise, the pair $(\widehat{\mathbf{p}}_\mathbf{1}, \mathcal{Q}_\mathbf{1})$ is added to the working list $\mathcal{W}$ containing the valid parameters and the parts of the search space remaining to explore. The location $\ell = 1$ is added to the list of visited location $\mathcal{V}$ for which a satisfying parameter vector has been found. The algorithm continues with the unexplored location with the smallest index

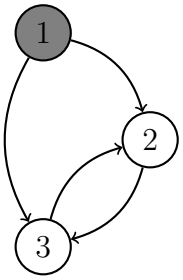
$$\underline{\ell} = \min\left(\left(\bigcup_{\ell\in\mathcal{V}} \mathrm{succ}\,(\ell) \cup \left(\bigcup_{\ell\in\mathcal{V}} \mathrm{pred}\,(\ell)\right) \setminus \mathcal{V}\right)\right), \tag{5.20}$$

where $\mathrm{pred}\,(\ell)$ and $\mathrm{succ}\,(\ell)$ represent respectively the sets of locations from which ℓ may be reached and the set of locations that may be reached from ℓ, both in one transition. Since the graph is connected, $\underline{\ell}$ exists until all locations have been explored.

The algorithm considers now the location $\underline{\ell}$. CSC-FPS-Hyb tries to find a valid $\widehat{p}_{\underline{\ell}} \in \mathcal{Q}^0_\ell$ satisfying the constraints (5.12), (5.13), and (5.14). Moreover, additionally $\widehat{\mathbf{p}}_{\underline{\ell}}$ has to satisfy (5.11) for all $(\underline{\ell}, \ell')$ and for all $(\ell', \underline{\ell})$ with $\ell' \in \mathcal{V}$. For that purpose, CSC-FPS-Hyb takes as input the values of the parameters $\widehat{\mathbf{p}}_j$, $j \in \mathcal{V}$ found for the previously explored locations.

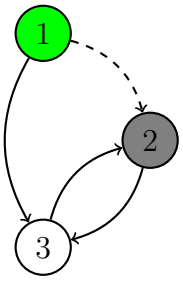
If a satisfying value $\widehat{\mathbf{p}}_{\underline{\ell}}$ has been found, the algorithm updates $\mathcal{V}$ and $\mathcal{W}$ and continues with the new location provided by (5.20) until $\mathcal{V} = \mathcal{L}$. Figures 5.2 and 5.3 describe this behavior.

Figures 5.2 and 5.3 describe the iterations of the CSC-FPS-Hyb for an hybrid system with three locations ℓ_1, ℓ_2, and ℓ_3. The algorithm is initialized with the location $\underline{\ell} = 1$ and searches for a parameter $\mathbf{p}$ in the initial queue $\mathcal{Q}^0_{\underline{1}}$ that satisfies the continuous constraints (5.12), (5.13), and (5.14). Once the parameter $\widehat{\mathbf{p}}_1$ is found it is added to the working list $\mathcal{W}$ with the remaining queue $\mathcal{Q}_1$ of non explored parameter. Location 1 is then added the list $\mathcal{V}$. Then CSC-FPS-Hyb picks the smallest index in succ(1) and pred(1), which is $\underline{\ell} = 2$ and searches for a parameter $\mathbf{p}$ in the initial queue $\mathcal{Q}^0_2$ that satisfies the continuous constraints (5.12), (5.13), and (5.14) and the transition constrains (5.11) for the pairs of locations $(1, 2)$



$\mathcal{V} = \emptyset$
$\mathcal{W} = \emptyset$
$\underline{\ell} = 1$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}^0_{\underline{\ell}}$

Consider Location 1 without considering transitions to find $\widehat{\mathbf{p}}_1$



$\mathcal{V} = \{1\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1)\}$
$\underline{\ell} = 2$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}^0_{\underline{\ell}}$

Consider Location 2 and consider the transition $(1, 2)$ to find $\widehat{\mathbf{p}}_2$

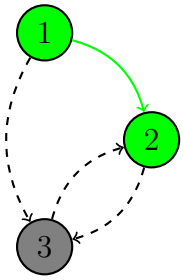
Figure 5.2: First two steps of the exploration of the graph associated to the hybrid automaton by the CSC-FPS-Hyb algorithm when no backtrack occurs.

and $(2, 1)$. Once the parameter $\widehat{\mathbf{p}}_2$ is found the working list $\mathcal{W}$ and the list $\mathcal{V}$ are updated. Finally the last location $\underline{\ell} = 3$ is visited, the algorithm searches for a parameter $\mathbf{p}$ in the initial queue $\mathcal{Q}^0_3$ that satisfies the continuous constraints (5.12), (5.13), and (5.14) and the transition constrains (5.11) for the pairs of locations $(1, 3)$, $(3, 1)$, $(2, 3)$, and $(3, 2)$. Once the parameter $\widehat{\mathbf{p}}_3$ is found the algorithm stops and returns the valid parameters.

If no satisfying value $\widehat{\mathbf{p}}_{\underline{\ell}}$ can been found, two cases have to be considered. First, there is no $\widehat{\mathbf{p}}_{\underline{\ell}}$ in $\mathcal{Q}_{\underline{\ell}}$ satisfying the constraints (5.12), (5.13), and (5.14), the algorithm stops. Second, for all $\widehat{\mathbf{p}}_{\underline{\ell}}$ in $\mathcal{Q}_{\underline{\ell}}$ satisfying the constraints (5.12), (5.13), and (5.14) there is at least one location $\ell' \in \mathcal{V}$ for which one of the Constraint (5.11) associated to the transitions $(\underline{\ell}, \ell')$ or $(\ell', \underline{\ell})$ is not satisfied. The value of the parameter $\widehat{\mathbf{p}}_{\ell'}$ found for location ℓ' does not allow the satisfaction of Constraint (5.11) associated to the transition $(\underline{\ell}, \ell')$ or $(\ell', \underline{\ell})$. An other value for $\widehat{\mathbf{p}}_{\ell'}$ has to be found to address this issue. Let $\mathcal{S}_{\underline{\ell}} \subset \mathcal{Q}_{\underline{\ell}}$ be the set of all $\widehat{\mathbf{p}}_{\underline{\ell}}$ in $\mathcal{Q}_{\underline{\ell}}$ satisfying the constraints (5.12), (5.13), and (5.14). Let $\mathcal{U} \subseteq \mathcal{V}$ be the set of location indexes ℓ such that there exists $\widehat{\mathbf{p}}$ in $\mathcal{S}_{\underline{\ell}}$ for which $\widehat{\mathbf{p}}$ does not satisfy (5.11) associated to the transition $(\ell, \underline{\ell})$ or $(\underline{\ell}, \ell)$.

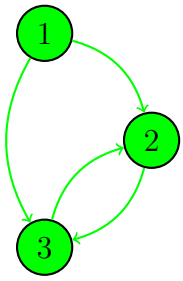
CSC-FPS-Hyb picks one $\ell \in \mathcal{U}$, for example, it can be the location with the smallest index. This location ℓ cannot be considered as validated any more and has to be removed from the list $\mathcal{V}$. One also removes the pair $(\widehat{\mathbf{p}}_\ell, \mathcal{Q}_\ell)$ from the working list $\mathcal{W}$. The search for a valid parameter $\mathbf{p}_\ell$ is done starting from $\mathcal{Q}_\ell$. Figures 5.4 and 5.5 describe such backtracking behavior.

In Figures 5.4 and 5.5 CSC-FPS-Hyb fails at finding a parameter $\widehat{\mathbf{p}}_3$. It backtracks to the location $\underline{\ell} = 1$ and searches for a parameter $\mathbf{p}$ in the queue $\mathcal{Q}_1$ that satisfies the continuous



$\mathcal{V} = \{1, 2\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1), (\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\}$
$\underline{\ell} = 3$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}^0_{\underline{\ell}}$

Consider Location 3 and consider the transitions $(2, 3)$, $(3, 2)$ and $(1, 3)$ to find $\widehat{\mathbf{p}}_3$



$\mathcal{V} = \{1, 2, 3\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1), (\widehat{\mathbf{p}}_2, \mathcal{Q}_2), (\widehat{\mathbf{p}}_3, \mathcal{Q}_3)\}$
$\underline{\ell} = 3$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}_{\underline{\ell}}$

End of the algorithm all locations and transitions have been considered

Figure 5.3: Last steps of the exploration of the graph associated to the hybrid automaton by the CSC-FPS-Hyb algorithm when no backtrack occurs.

constraints (5.12), (5.13), and (5.14) as well as the transition constrains (5.11) for the pair of locations $(1, 2)$ and $(2, 1)$. Once the parameter $\widehat{\mathbf{p}}_1$ is found the algorithm goes back to Location $\underline{\ell} = 3$ and tries to find a parameter $\mathbf{p}$ in the initial queue $\mathcal{Q}^0_3$ that satisfies the continuous constraints (5.12), (5.13), and (5.14) and the transition constraints (5.11) for the pairs of locations $(1, 3)$, $(3, 1)$, $(2, 3)$, and $(3, 2)$.

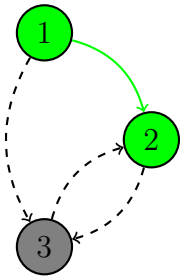
Note that, CSC-FPS-Hyb can manage a cascade of backtracking.

Figure 5.6 and 5.7 exhibit the case of multiple backtracks. If the algorithm fails to find a valid parameter $\widehat{\mathbf{p}}_3$ for the location $\underline{\ell} = 3$ and backtracks to $\underline{\ell} = 1$ and fails to find a parameter $\widehat{\mathbf{p}}_1$ for $\underline{\ell} = 1$ the algorithm then backtracks to the location $\underline{\ell} = 2$ and tries to find a parameter $\mathbf{p}$ in the queue $\mathcal{Q}_2$ that satisfies (5.12), (5.13), and (5.14). Then, once $\widehat{\mathbf{p}}_2$ is found CSC-FPS-Hyb goes back to $\underline{\ell} = 1$ and searches for a parameter $\mathbf{p}$ in the initial queue $\mathcal{Q}^0_1$ that satisfies the continuous constraints (5.12), (5.13), and (5.14) as well as the transition constraints (5.11) for the pairs of locations $(1, 2)$ and $(2, 1)$ and finally after finding $\widehat{\mathbf{p}}_1$ the algorithm searches for $\mathbf{p}$ in the initial queue $\mathcal{Q}^0_3$ that satisfies the continuous constraints (5.12), (5.13), and (5.14) as well as the transition constraints (5.11) for the pairs of locations $(1, 3)$, $(3, 1)$, $(2, 3)$ and $(3, 2)$ and terminates once it has found $\widehat{\mathbf{p}}_3$.

Finally, Algorithm 5 summarizes the incremental design of barrier functions performed by CSC-FPS-Hyb as previously presented.

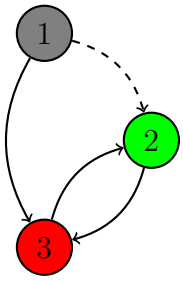
Algorithm 5 CSC-FPS-Hyb

function CSC-FPS-HYB($\mathcal{H}$, $[\mathbf{x}]$)
  ▷ Initialization
  $\mathcal{V} := \emptyset$ ▷ Locations for which valid $\widehat{\mathbf{p}}$ have been found
  $\mathcal{W} := \emptyset$ ▷ List of pairs $(\widehat{\mathbf{p}}_\ell, \mathcal{Q}_\ell)$
  for $\ell := 1$ to $|\mathcal{L}|$ **do** ▷ Initial search space for parameters for each location
    $\mathcal{Q}_\ell^0 = \{[\mathbf{p}]_\ell\}$
  end for
  $\underline{\ell} := 1$ ▷ Initial location
  $\mathcal{Q}_{\underline{\ell}} := \mathcal{Q}_{\underline{\ell}}^0$
  ▷ Main loop
  while $\mathcal{V} \neq \mathcal{L}$ **do**
    repeat ▷ CSC-FPS applied to (5.12), (5.13), and (5.14)
      $(\eta, \widehat{\mathbf{p}}_{\underline{\ell}}, \mathcal{Q}_{\underline{\ell}}) = \text{CSC-FPS}(\mathcal{Q}_{\underline{\ell}}, [\mathbf{x}], [\mathbf{d}]_{\underline{\ell}}, \mathcal{W})$
      if $\eta \neq 1$ **then**
        return failure ▷ Unsatisfiable problem or maximal precision has been reached
      else
        $\mathcal{U} = \emptyset$
        for all $\ell \in \mathcal{V}$ **do** ▷ CSC applied to (5.11) considering the transitions $(\ell, \underline{\ell})$ or $(\underline{\ell}, \ell)$
          $\eta_{\text{C}} = \text{CSC}([\mathbf{x}], \widehat{\mathbf{p}}_{\underline{\ell}}, \widehat{\mathbf{p}}_\ell)$
          if $\eta_{\text{C}} \neq 1$ **then**
            $\mathcal{U} = \mathcal{U} \cup \{\ell\}$ ▷ Constraint on transition $(\ell, \underline{\ell})$ or $(\underline{\ell}, \ell)$ is not satisfied
          end if
        end for
      end if
    until $\mathcal{U} = \emptyset \vee \mathcal{Q}_{\underline{\ell}} = \emptyset$
    if $\mathcal{U} = \emptyset$ **then** ▷ Everything is fine, go to the next location
      $\mathcal{V} = \mathcal{V} \cup \{\underline{\ell}\}$
      $\mathcal{W} = \mathcal{W} \cup \{(\widehat{\mathbf{p}}_{\underline{\ell}}, \mathcal{Q}_{\underline{\ell}})\}$
      $\underline{\ell}$ is defined as (5.20)
    else ▷ Location $\underline{\ell}$ cannot be proved valid, backtracking
      $\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}_{\underline{\ell}}^0$
      $\underline{\ell} = \min(\mathcal{U})$
      $\mathcal{V} = \mathcal{V} \setminus \{\underline{\ell}\}$
      $\mathcal{W} = \mathcal{W} \setminus \{(\widehat{\mathbf{p}}_{\underline{\ell}}, \mathcal{Q}_{\underline{\ell}})\}$
    end if
  end while
  return $\mathcal{W}$ ▷ All constraints for $\mathcal{H}$ are satisfied
end function



$\mathcal{V} = \{1, 2\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1), (\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\}$
$\underline{\ell} = 3$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}^0_{\underline{\ell}}$

Consider Location 3 and consider the transitions $(2, 3)$, $(3, 2)$ and $(1, 3)$ to find $\widehat{\mathbf{p}}_3$



$\mathcal{V} = \{2\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\}$
$\underline{\ell} = 1$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}_1$

Search of $\widehat{\mathbf{p}}_3$ failed so backtracking to Location 1 considering only transition $(1, 2)$ and starting the search for $\widehat{\mathbf{p}}_1$ with the search space $\mathcal{Q}_1$

Figure 5.4: CSC-FPS-Hyb when a backtracking has to be performed

5.5 Experimental results

Several hybrid dynamical models are now considered to test the efficiency of the proposed CSC-FPS-Hyb algorithm.

5.5.1 Hybrid dynamical system descriptions

Beside the 2-TANK example taken from [RS07] and already described in Section 5.2.1, several examples are considered in what follows, some of which account for the presence of state perturbations.

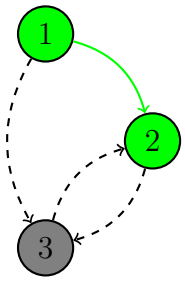
Example 17. *(Prajna) This example is inspired from [PJ04]. It is a 3-dimensional example with 2 locations and a state perturbation:*

$$f_1\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ -x + x_3 \\ x_1 + (2x_2 + 3x_3)(1 + x_3^2) + d \end{pmatrix}, \quad f_2\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ -x_1 + x_3 \\ -x_1 - 2x_2 + 3 + d \end{pmatrix}$$

with $d \in [0.1, 0.2]$

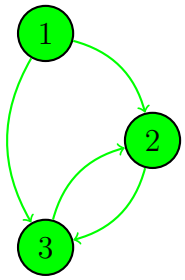
- *Initial conditions:*

$$g_0(1, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 0.01, \qquad g_0(2, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 1$$



$\mathcal{V} = \{1, 2\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1)\,(\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\}$
$\underline{\ell} = 3$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}^0_{\underline{\ell}}$

Search for a new $\widehat{\mathbf{p}}_1$ succeed so go back to Location 3 considering all the transitions $(2,3)$, $(3,2)$ and $(1,3)$ and starting the search for $\widehat{\mathbf{p}}_3$ with the initial search space $\mathcal{Q}^0_3$



$\mathcal{V} = \{1, 2, 3\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1)\,, (\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\,, (\widehat{\mathbf{p}}_3, \mathcal{Q}_3)\}$
$\underline{\ell} = 3$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}_{\underline{\ell}}$

End of the algorithm all locations and transitions have been considered

Figure 5.5: Last steps of CSC-FPS-Hyb when a single backtracking has to be performed

- *Unsafe regions:*

$$g_u(1, \mathbf{x}) = -1, \qquad g_u(2, \mathbf{x}) = 10^4(x_1 + 5.05)^2 + x_2^2 + x_3^2 - 1$$

- *Invariants:*

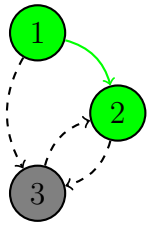
$$g_{\mathcal{I}}(1, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 2, \qquad g_{\mathcal{I}}(2, \mathbf{x}) = (x_1 + 2.05)^2 + 9.5x_2^2 + 9.5x_3^2 - 9.5$$

- *Guards:*

$$g_\Gamma(1, 2, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 1, \qquad g_\Gamma(2, 1, \mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 0.25$$

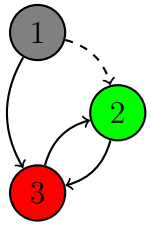
The considered parametric barrier functions are

$$\beta_1(\mathbf{x}, \mathbf{p}) = p_1x_1 + p_2x_2 + p_3x_3 + p_4, \qquad \beta_2(\mathbf{x}, \mathbf{p}) = p_1x_1 + p_2x_2 + p_3x_3 + p_4$$



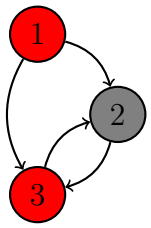
$\mathcal{V} = \{1, 2\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1), (\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\}$
$\underline{\ell} = 3$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}^0_{\underline{\ell}}$

Consider Location 3 and consider the transitions $(2, 3)$, $(3, 2)$ and $(1, 3)$ to find $\widehat{\mathbf{p}}_3$



$\mathcal{V} = \{2\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\}$
$\underline{\ell} = 1$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}_{\ell_1}$

Search of $\widehat{\mathbf{p}}_3$ failed so backtracking to Location 1 considering only transition $(1, 2)$ and starting the search for $\widehat{\mathbf{p}}_1$ with the search space $\mathcal{Q}_1$



$\mathcal{V} = \emptyset$
$\mathcal{W} = \emptyset$
$\underline{\ell} = 2$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}_2$

Search of a new $\widehat{\mathbf{p}}_1$ also failed so backtracking to Location 2 considering no transition and starting the search for $\widehat{\mathbf{p}}_2$ with the search space $\mathcal{Q}_2$

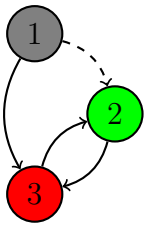
Figure 5.6: CSC-FPS-Hyb algorithm in case of a cascade of backtracking

Example 18. *(CAR) This example is taken from [CAS12]. It has a state-space of dimension 6 with three locations describing a simplified motion of a car:*

$$f_1\begin{pmatrix} x \\ y \\ v \\ c \\ s \\ o \end{pmatrix} = \begin{pmatrix} vc \\ vs \\ -0.05 \\ ov^2 s \\ -ov^2 c \\ -0.1 \end{pmatrix}, \quad f_2\begin{pmatrix} x \\ y \\ v \\ c \\ s \\ o \end{pmatrix} = \begin{pmatrix} vc \\ vs \\ 0 \\ ov^2 s \\ -ov^2 c \\ 0 \end{pmatrix}, \quad f_3\begin{pmatrix} x \\ y \\ v \\ c \\ s \\ o \end{pmatrix} = \begin{pmatrix} vc \\ vs \\ 0.05 \\ ov^2 s \\ -ov^2 c \\ 0.1 \end{pmatrix}$$

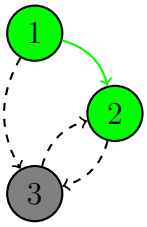
- *Initial conditions:*

$$\begin{aligned} g_0(1, \mathbf{x}) &= (x - 1.1)^2 + (y - 1.1)^2 + 200(v - 0.805)^2 + 200(c - 0.705)^2 + \\ &\quad 200(s - 0.705)^2 + 16(o - 0.025)^2 - 0.01 \\ g_0(2, \mathbf{x}) &= (x - 1.1)^2 + (y - 1.1)^2 + 200(v - 0.805)^2 + 200(c - 0.705)^2 \\ &\quad + 4(s - 0.75)^2 + 16(o - 0.025)^2 - 0.01 \\ g_0(3, \mathbf{x}) &= (x - 1.1)^2 + (y - 1.1)^2 + 200(v - 0.805)^2 + 200(c - 0.705)^2 \\ &\quad + (s - 0.9)^2 + 16(o - 0.025)^2 - 0.01 \end{aligned}$$



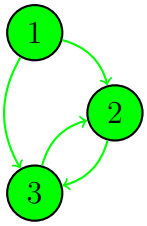
$\mathcal{V} = \{2\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\}$
$\underline{\ell} = 1$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}^0_{\underline{\ell}}$

Search of a new $\widehat{\mathbf{p}}_2$ succeeded, so Location 1 is now considered taking into account transition $(1, 2)$ and starting the search for $\widehat{\mathbf{p}}_1$ with the initial search space $\mathcal{Q}^0_1$



$\mathcal{V} = \{1, 2\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1)\,(\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\}$
$\underline{\ell} = 3$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}^0_{\underline{\ell}}$

Search for a new $\widehat{\mathbf{p}}_1$ succeeded, so go back to Location 3 considering all the transitions $(2, 3)$, $(3, 2)$ and $(1, 3)$ and starting the search for $\widehat{\mathbf{p}}_3$ with the initial search space $\mathcal{Q}^0_3$



$\mathcal{V} = \{1, 2, 3\}$
$\mathcal{W} = \{(\widehat{\mathbf{p}}_1, \mathcal{Q}_1)\,, (\widehat{\mathbf{p}}_2, \mathcal{Q}_2)\,, (\widehat{\mathbf{p}}_3, \mathcal{Q}_3)\}$
$\underline{\ell} = 3$
$\mathcal{Q}_{\underline{\ell}} = \mathcal{Q}_{\underline{\ell}}$

End of the algorithm all locations and transitions have been considered.

Figure 5.7: Last steps of CSC-FPS-Hyb algorithm with a cascade of backtracking

- *Unsafe regions:*

$$\begin{aligned} g_u(1, \mathbf{x}) = g_u(2, \mathbf{x}) &= (x - 6.5)^2 + (y - 2.5)^2 + 10^5(v - 0.805)^2 \\ &\quad + 10^5(c - 0.705)^2 + 100(s - 0.75)^2 + 400(o - 0.025)^2 - 0.25 \\ g_u(3, \mathbf{x}) &= (x - 6.5)^2 + (y - 2.5)^2 + 10^5(v - 0.805)^2 \\ &\quad + 10^5(c - 0.705)^2 + 50(s - 0.75)^2 + 400(o - 0.025)^2 - 0.25 \end{aligned}$$

- *Invariants:*

$$\begin{aligned} g_{\mathcal{I}}(1, \mathbf{x}) &= (x - 4)^2 + 9(y - 2)^2 + 10^6(v - 0.805)^2 + 10^6(c - 0.705)^2 \\ &\quad + 10^6(s - 0.705)^2 + 15 \cdot 10^3(o - 0.025)^2 - 9 \\ g_{\mathcal{I}}(2, \mathbf{x}) &= (x - 4)^2 + 9(y - 2)^2 + 10^6(v - 0.805)^2 + 10^6(c - 0.705)^2 \\ &\quad + 2 \cdot 10^3(s - 0.725)^2 + 15 \cdot 10^3(o - 0.025)^2 - 9 \\ g_{\mathcal{I}}(3, \mathbf{x}) &= (x - 4)^2 + 9(y - 2)^2 + 10^6(v - 0.805)^2 + 10^6(c - 0.705)^2 \\ &\quad + 10^3(s - 0.725)^2 + 15 \cdot 10^3(o - 0.025)^2 - 9 \end{aligned}$$

- *Guards:*

$$\begin{aligned}
g_\Gamma(1,2,\mathbf{x}) &= (x-1.1)^2 + (y-1.1)^2 + 200(v-0.805)^2 + 200(c-0.705)^2 \\
&\quad + 4(s-0.75)^2 + 16(o-0.025)^2 - 0.01 \\
g_\Gamma(2,1,\mathbf{x}) &= (x-1.1)^2 + (y-1.1)^2 + 200(v-0.805)^2 + 200(c-0.705)^2 \\
&\quad + 16(s-0.675)^2 + 16(o-0.025)^2 - 0.01 \\
g_\Gamma(2,3,\mathbf{x}) &= (x-1.1)^2 + (y-1.1)^2 + 200(v-0.805)^2 + 200(c-0.705)^2 \\
&\quad + (s-0.9)^2 + 16(o-0.025)^2 - 0.01 \\
g_\Gamma(3,2,\mathbf{x}) &= (x-1.1)^2 + (y-1.1)^2 + 200(v-0.805)^2 + 200(c-0.705)^2 \\
&\quad + 16(s-0.725)^2 + 16(o-0.025)^2 - 0.01 \\
g_\Gamma(1,3,\mathbf{x}) &= -1 \\
g_\Gamma(3,1,\mathbf{x}) &= -1
\end{aligned}$$

The considered parametric barrier functions are

$$\beta_1(\mathbf{x},\mathbf{p}) = \beta_2(\mathbf{x},\mathbf{p}) = \beta_3(\mathbf{x},\mathbf{p}) = p_1x + p_2y + x_3v + p_4c + p_4s + p_5o,$$

Example 19. *(Collision) This example, found in [CFH+03], has dimension of the state space equal to 3 and has 6 locations. It describes a collision avoidance system. We replaced the location* in canal *which described the unsafe location to an unsafe region in our problem. To note we took linear template for all location except for the Location* 3 *where quadratic template was chosen.*

$$f_1\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}-2\sin(x_2)\\0\\0\end{pmatrix}, \quad f_2\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}-2\sin(x_2)\\\frac{\pi}{4}\\1\end{pmatrix}, \quad f_3\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}-2\sin(x_2)\\-\frac{\pi}{4}\\1\end{pmatrix}$$

$$f_4\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}-2\sin(x_2)\\-\frac{\pi}{4}\\-2\end{pmatrix}, \quad f_5\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}-2\sin(x_2)\\\frac{\pi}{4}\\-2\end{pmatrix}, \quad f_6\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix} = \begin{pmatrix}-2\sin(x_2)\\0\\0\end{pmatrix}$$

- *Initial conditions:*

$$\begin{aligned}
&g_0(1,\mathbf{x}) = x_1^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1, && g_0(2,\mathbf{x}) = 10^6(x_1-1)^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1 \\
&g_0(3,\mathbf{x}) = 10^6(x_1+1)^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1, && g_0(4,\mathbf{x}) = 10^6(x_1-1)^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1 \\
&g_0(5,\mathbf{x}) = 10^6(x_1+1)^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1, && g_0(6,\mathbf{x}) = x_1^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1
\end{aligned}$$

- *Unsafe regions:*

$$g_u(1,\mathbf{x}) = 10^6(x_1+2)^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1, \qquad g_u(2,\mathbf{x}) = 10^6(x_1+2)^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1$$
$$g_0(3,\mathbf{x}) = 10^6(x_1+2)^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1, \qquad g_u(4,\mathbf{x}) = 10^6(x_1+2)^2 + \frac{\pi}{2}x_2^2 + \frac{(x_3-2)^2}{2} - 1$$
$$g_u(5,\mathbf{x}) = 10^6(x_1+2)^2 + \frac{\pi}{2}x_2^2 + \frac{(x_3-2)^2}{2} - 1, \quad g_u(6,\mathbf{x}) = 10^6(x_1+2)^2 + \frac{\pi}{2}x_2^2 + \frac{(x_3-2)^2}{2} - 1$$

- *Invariants:*

$$g_{\mathcal{I}}(1,\mathbf{x}) = x_1^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1, \qquad g_{\mathcal{I}}(2,\mathbf{x}) = (x_1-2.5)^2 + \pi x_2^2 + 10^6x_3^2 - 2.2$$
$$g_{\mathcal{I}}(3,\mathbf{x}) = (x_1+0.5)^2 + \pi x_2^2 + 10^6x_3^2 - 2.2, \quad g_{\mathcal{I}}(4,\mathbf{x}) = x_1^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1$$
$$g_{\mathcal{I}}(5,\mathbf{x}) = x_1^2 + \frac{\pi}{2}x_2^2 + \frac{(x_3-2)^2}{4} - 1, \qquad g_{\mathcal{I}}(6,\mathbf{x}) = x_1^2 + \frac{\pi}{2}x_2^2 + \frac{(x_3-2)^2}{4} - 1$$

- *Guards:*

$$g_\Gamma(1,2,\mathbf{x}) = -x_1 + 10^6x_3^2 + 1, \qquad g_\Gamma(1,3,\mathbf{x}) = x_1 + 10^6x_3^2 + 1$$
$$g_\Gamma(2,4,\mathbf{x}) = -x_1 + 10^6x_3^2 + 1, \qquad g_\Gamma(3,5,\mathbf{x}) = x_1 + 10^6x_3^2 + 1$$
$$g_\Gamma(4,3,\mathbf{x}) = x_1 + \frac{(x_3-2)^2}{4} + 1, \qquad g_\Gamma(4,6,\mathbf{x}) = x_1^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1$$
$$g_\Gamma(5,2,\mathbf{x}) = -x_1 + \frac{(x_3-2)^2}{4} + 1, \qquad g_\Gamma(5,6,\mathbf{x}) = x_1^2 + \frac{\pi}{2}x_2^2 + 10^6x_3^2 - 1$$

The considered parametric barrier functions are

$$\beta_1(\mathbf{x},\mathbf{p}) = \beta_2(\mathbf{x},\mathbf{p}) = \beta_4(\mathbf{x},\mathbf{p}) = \beta_5(\mathbf{x},\mathbf{p}) = \beta_6(\mathbf{x},\mathbf{p}) = p_1x_1 + p_2x_2 + p_3x_3 + p_4,$$

$$\beta_3(\mathbf{x},\mathbf{p}) = p_1x_1^2 + p_2x_2^2 + p_3x_1 + p_4x_2 + p_5x_3 + p_6,$$

5.5.2 Experimental conditions and results

The search space for the parameter vectors is set for all examples as $[\mathbf{p}]_0 = [\mathbf{p}]_1 = ... = [\mathbf{p}]_{|\mathcal{L}|} = [-10, 10]$.

For all examples, CSC-FPS-Hyb provides very good results. This is mainly due to the absence of backtracking in the considered examples.

Table 5.1: Results for the hybrid systems

Example	Dimension	Nb of locations	Comput. time	Nb of bisections
2-TANKS	2	2	1.7s	9488
Prajna	3	2	11s	111
CAR	6	3	0.021s	3
Collision	3	6	0.407s	2015

5.5.2.1 The resulting barrier functions

For the exemple Prajna the algorithm return

$$\beta_1(\mathbf{x},\mathbf{p}) = -5, \qquad \beta_2(\mathbf{x},\mathbf{p}) = -2.99505x_1 - 1.25x_2 - 2.5x_3 - 7.5,$$

For the exemple 2TANK the algorithm return

$$\beta_1(\mathbf{x},\mathbf{p}) = -0.3125x_2^2 - 1.46242x_1 - 0.9375x_2 + 7.20355, \qquad \beta_2(\mathbf{x},\mathbf{p}) = -3.38983x,$$

For the exemple CAR the algorithm return

$$\beta_1(\mathbf{x},\mathbf{p}) = 3.32308x - 5y, \qquad \beta_2(\mathbf{x},\mathbf{p}) = 3.25385x - 5y,$$
$$\beta_3(\mathbf{x},\mathbf{p}) = 3.1x - 5y,$$

For the exemple Collision the algorithm return

$$\begin{aligned}
\beta_1(\mathbf{x},\mathbf{p}) &= -3.75x_1 - 1.25x_2 - 2.5x_3 - 6.3385,\\
\beta_2(\mathbf{x},\mathbf{p}) &= -2.5x_1,\\
\beta_3(\mathbf{x},\mathbf{p}) &= 1.77462x_1^2 + 2.5x_2^2 + -2.66193x_1 + 2.5x_2 - 7.5x_3 - 7.94653,\\
\beta_4(\mathbf{x},\mathbf{p}) &= -5x_1 + 5x_2 + 5x_3,\\
\beta_1(\mathbf{x},\mathbf{p}) &= -5x_1 + 5x_2 + 5x_3,\\
\beta_2(\mathbf{x},\mathbf{p}) &= -5x_1 + 5x_2 + 5x_3,
\end{aligned}$$

5.6 Partial conclusions

This chapter presents a new method to compute barriers functions for systems described by non-linear hybrid models consisting of hybrid automata. The formulation of the barrier functions for such model are given as a quantified constraint satisfaction problem. The constraints can be classified in two categories, continuous constraints and transition constraints. The continuous constraints can be solved using a slightly adapted version of the CSC-FPS

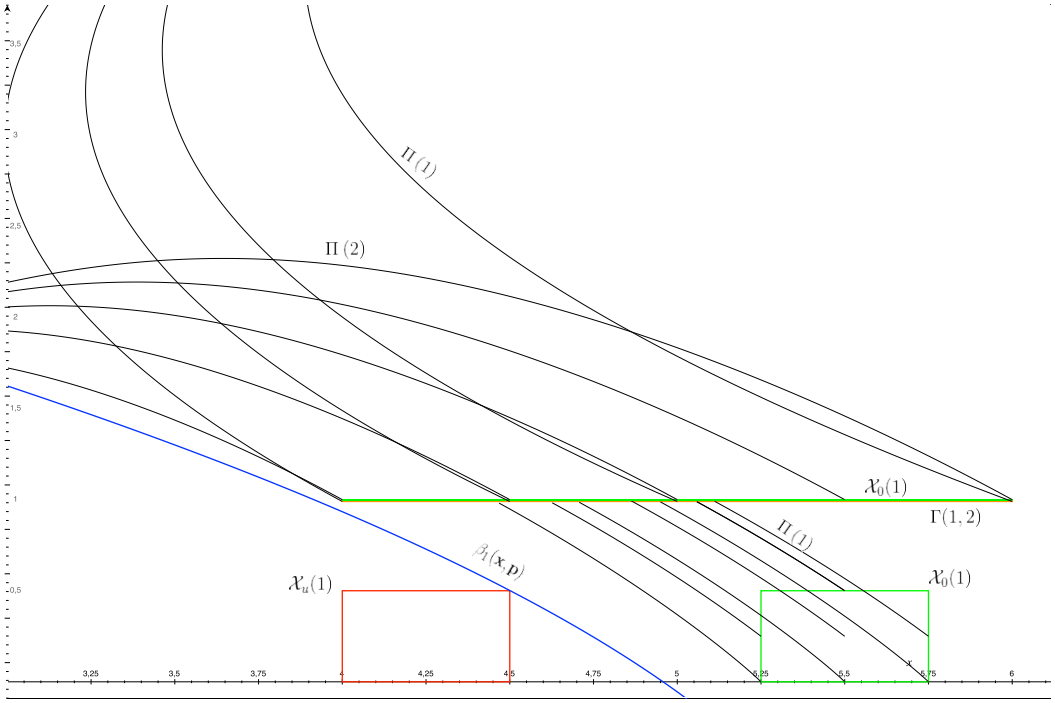


Figure 5.8: Results for Example 2TANK.

algorithm presented in Chapter 4. The transition constraints bind the barrier function together and do not allow to compute the barrier functions independently. To address this issue different methods for computing the barrier functions for hybrid models are proposed. One can either compute each barrier independently and try to prove the satisfaction of the transition constraints, or one can try to solve all the constraints in a bulk. The first method is suited for systems with a low number of transition constraints. The second method is better suited to small systems due to the high complexity required to solve all the constraints simultaneously. The third incremental design method is based on the exploration of the graph that describes the location and the transition of the hybrid system. This method may address more general forms of hybrid models. This chapter ends with a description of the CSC-FPS-Hyb algorithm and provides some results on examples taken from the literature.

Chapter 6

Conclusions and future research directions

This chapter provides an overview of the ideas presented in the thesis and presents some future research directions.

6.1 Conclusion

This thesis addresses the problem of proving the safety of non-linear dynamical systems and hybrid dynamical systems. A system is said to be safe if all trajectories of its state do not reach an unsafe region. Proving the safety of system by explicitly computing all its trajectories when its dynamic is non-linear or when its behavior is described by an hybrid model with non-linear dynamics remains a challenging task.

To prove the safety of a system, the barrier function approach is considered. A barrier function partitions the state space of the system and has to satisfy some constraints to isolate the trajectories of the system starting from any possible initial values of the state and the unsafe part of the state space. The set of constraints, which have to be satisfied by a barrier function are usually non-convex, rendering the search of satisfying barrier functions hard. Previously, only polynomial barrier functions were taken in consideration and for systems with polynomial dynamics.

This thesis considers relatively general dynamical systems with generic non-linear barrier function. The solutions presented are based on template barrier functions and interval analysis.

The first part of the thesis is focused on non-linear dynamical systems, which may be seen as hybrid systems with a single location. The barrier function design problem is first formulated as a constraint satisfaction problem that can be solved using tools from interval analysis. This formulation allows one to prove the safety of a nonlinear dynamical system by finding the parameters of a template barrier function such that all constraints are satisfied.

An algorithm based on the developed algorithm [JW96] proposed by Jaulin and Walter in the 90s was proposed. The main idea of the algorithm is to explore all the parameter space and try at each step to see whether the constraints are valid or not. Such algorithm has a double exponential complexity in the worst case, so contractors were added to the CSC-FPS algorithm to prune parts of search space and facilitate the search of a satisfying candidate barrier function.

The second part of the thesis is dedicated to the design of barrier functions for systems described by hybrid dynamical models. Safety properties have to be proven during the continuous-time evolution of the system, but also during transitions of the system. This leads to additional constraints that have to be satisfied by candidate barrier functions. With the CSC-FPS algorithm for the continuous part, we have only to prove that the constraints associated to transitions of the hybrid system are satisfied. A transition constraint links two barrier functions for two different locations of the hybrid system. The consequence of this propriety is that one cannot simply compute barrier functions independently for each location. Solving all the constraints simultaneously to find all the barrier functions is usually computationally intractable. In the proposed approach, the CSC-FPS-Hyb algorithm explores all the locations sequentially. Transition constraints are introduced progressively between the already explored locations. Backtracking to previous location is considered when transition constraints are not satisfied.

The CSC-FPS and CSC-FPS-Hyb algorithm have been tested on several examples taken from the state-of-the-art, showing their efficiency. Contractors were instrumental in most of the cases to reduce the computational complexity.

6.2 Future work

This section provides some future research directions. Most are relatively direct applications of some ideas presented in this thesis.

First one considers the problem of analyzing the reachability problem for a continuous-time dynamical system. In this problem, one considers a given target set $\mathcal{T}$ and tries to determine whether at least one point of $\mathcal{T}$ may be reached by the dynamical system from a set $\mathcal{X}_0$ of initial conditions. This problem can be seen as the dual problem of the safety analysis problem as stated in [PR05]. Some ideas used to prove the safety may be used in the reachability analysis context.

A second closely-related problem that may be considered is to find the smallest region where all trajectories remain for an unbounded time starting from a set $\mathcal{X}_0$ of initial conditions. This problem may be addressed by a direct application of the barrier function design approach.

6.2.1 Reachability for dynamical systems

Barrier functions have been used to prove that all trajectories of some dynamical system starting from an initial region cannot reach an unsafe region. If a valid barrier function is found, then the system is safe, but if no barrier function can be found in a timely manner, one cannot decide whether the system is safe or not.

In [PR05], a dual formulation for the barrier function theorem, which can be used to prove the safety of a dynamical system, is provided. It allows one to prove the reachability of a given region of the state space. A potential function has to satisfy some constraints to show the reachability property. If these constraints are satisfied, one may prove that the dynamical system reaches a target region starting form some initial region of the state space. This may be very useful to show that a system is not safe.

In what follows, one considers the usual topology of $\mathbb{R}^n$. The closure and the interior of a set $\mathcal{X} \subset \mathbb{R}^n$ are respectively denoted $\mathrm{cl}\,(\mathcal{X})$ and $\mathrm{int}\,(\mathcal{X})$.

The interior of a set $\mathcal{X} \subset \mathbb{R}^n$ is defined as the set of all $\mathbf{x} \in \mathcal{X}$ such that there exists some positive real number $\varepsilon(\mathbf{x}) > 0$ and an open ball $\mathcal{B}(\mathbf{x}, \varepsilon(\mathbf{x}))$ centered in $\mathbf{x}$ and of radius $\varepsilon(\mathbf{x})$ such that $\mathcal{B}(\mathbf{x}, \varepsilon(\mathbf{x})) \subset \mathcal{X}$.

The closure of a set $\mathcal{X} \subset \mathbb{R}^n$ is defined as the set of all $\mathbf{x} \in \mathbb{R}^n$ such there exists a sequence $\{a_n\}$ of elements of $\mathcal{X}$ converging to $\mathbf{x}$.

Theorem 5 ([PR05])**.** *Consider the dynamical system $\dot{\mathbf{x}} = f(\mathbf{x})$. Let $\mathcal{X} \subset \mathbb{R}^n$ be the state space, $\mathcal{X}_0 \subset \mathcal{X}$ be the set of possible initial states, $\mathcal{X}_u \subset \mathcal{X}$ the unsafe set, and $\mathcal{X}_r \subset \mathcal{X}$ the set one has to determine whether it can be reached.*

If there exist a function $B : \mathcal{X} \to \mathbb{R}$ such that

$$\forall \mathbf{x} \in \mathcal{X}_0, \quad B(\mathbf{x}) \leqslant 0, \tag{6.1}$$

$$\forall \mathbf{x} \in \mathcal{X}_u, \quad B(\mathbf{x}) > 0, \tag{6.2}$$

$\forall \mathbf{x} \in \mathcal{X}$,

$$B(\mathbf{x}) = 0 \implies \frac{\partial B(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}) < 0, \tag{6.3}$$

then the system is safe.

If there exists a function ρ satisfying

$$\int_{\mathcal{X}_0} \rho(\mathbf{x}) d\mathbf{x} > 0, \tag{6.4}$$

$$\rho(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in cl(\partial\mathcal{X} \setminus \partial\mathcal{X}_r), \tag{6.5}$$

$$\nabla(\rho f)(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in cl(\mathcal{X} \setminus \mathcal{X}_r), \tag{6.6}$$

then there exists a trajectory of the system that reaches $\mathcal{X}_r$ from $\mathcal{X}_0$.

where for $\rho : \mathbb{R}^n \to \mathbb{R}$ one has

$$\nabla(\rho) = \frac{\partial \rho}{\partial x_1}(\mathbf{x}) + ... + \frac{\partial \rho}{\partial x_n}(\mathbf{x}).$$

6.2.1.1 Template approach

To facilitate the search for a ρ function, one considers again parametric templates for the reachability function of the form $\rho(\mathbf{x}, \mathbf{p})$ with $\mathbf{p} \in \mathcal{P}$ a vector of parameters. Then, one has to search for parameter vectors that satisfy the constraints of Theorem 5.

Proposition 6. *Consider the dynamical system $\dot{\mathbf{x}} = f(\mathbf{x})$. Let $\mathcal{X} \subset \mathbb{R}^n$ be the state space, $\mathcal{X}_0 \subset \mathcal{X}$ be the set of possible initial states, and $\mathcal{X}_r \subset \mathcal{X}$ the set one has to determine whether it can be reached. If there exist a function ρ for which $\exists \mathbf{p} \in \mathcal{P}$ such that*

$$\int_{\mathcal{X}_0} \rho(\mathbf{x}, \mathbf{p}) d\mathbf{x} > 0, \tag{6.7}$$

$$\forall \mathbf{x} \in cl(\partial\mathcal{X} \setminus \partial\mathcal{X}_r) \quad \rho(\mathbf{x}, \mathbf{p}) < 0, \tag{6.8}$$

$$\forall \mathbf{x} \in cl(\mathcal{X} \setminus \mathcal{X}_r) \quad \rho(\mathbf{x}, \mathbf{p}) = 0 \implies \nabla.(\rho f)(\mathbf{x}, \mathbf{p}) > 0, \tag{6.9}$$

then there exists a trajectory of the system that reaches $\mathcal{X}_r$ from $\mathcal{X}_0$

6.2.1.2 Interval formulation

In what follows, one assumes that all sets are boxes. This approach can easily be extended to sets defined as unions of non-overlapping boxes.

Proposition 7. *Assume that $\mathcal{X} = [\mathbf{x}]$, $\mathcal{X}_0 = [\mathbf{x}]_0$, $\mathcal{X}_r = [\mathbf{x}]_r$, and $\mathcal{P} = [\mathbf{p}]$. If there exist a function ρ satisfying $\exists \mathbf{p} \in [\mathbf{p}]$*

$$\int_{[\mathbf{x}]_0} \rho(\mathbf{x}, \mathbf{p}) d\mathbf{x} > 0, \tag{6.10}$$

$$\forall \mathbf{x} \in cl(\partial([\mathbf{x}]) \setminus \partial([\mathbf{x}]_r)) \quad \rho(\mathbf{x}, \mathbf{p}) < 0, \tag{6.11}$$

$$\forall \mathbf{x} \in [\mathbf{x}] \setminus [\mathbf{x}]_r \quad \rho(\mathbf{x}, \mathbf{p}) = 0 \implies \nabla(\rho f)(\mathbf{x}, \mathbf{p}) > 0, \tag{6.12}$$

then there exist a trajectory of the system that reaches $[\mathbf{x}]_r$ from $[\mathbf{x}]_0$.

6.2.1.3 Resolution of the constraints

To find a satisfying parametric function ρ, one needs to find some $\mathbf{p} \in [\mathbf{P}]$ such that the constraints of Proposition 7 are satisfied. An adaptation of the CSC-FPS algorithm is necessary to handle such constraints. For the constraint (6.10), one may consider bounding the

integral using interval analysis. For the constraint (6.11) implying the closure of the intersection of box boundaries, one may, for example, consider degenerate thin boxes containing these boundaries.

Alternatively [PR05] proposed another formulation of the reachability problem.

Theorem 6 ([PR05])**.** *Consider the dynamical system* $\dot{\mathbf{x}} = f(\mathbf{x})$*. Let* $\mathcal{X} \subset \mathbb{R}^n$ *be the state space,* $\mathcal{X}_0 \subset \mathcal{X}$ *be the set of possible initial states,* $\mathcal{X}_u \subset \mathcal{X}$ *the unsafe set, and* $\mathcal{X}_r \subset \mathcal{X}$ *the set one has to determine whether it can be reached.*

If $\mathcal{X}_0$ *has non-empty interior there exists a function* B *satisfying*

$$\int_{\mathcal{X}_0} B(\mathbf{x}) d\mathbf{x} \leq 0, \tag{6.13}$$

$$B(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in cl(\partial\mathcal{X} \setminus \partial\mathcal{X}_r), \tag{6.14}$$

$$\frac{\partial B}{\partial \mathbf{x}}(\mathbf{x}) f(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in cl(\mathcal{X} \setminus \mathcal{X}_r), \tag{6.15}$$

then there exists a trajectory of the system that reaches $\mathcal{X}_r$ *from* $\mathcal{X}_0$*.*

If there exists a set $\tilde{\mathcal{X}}_0 \subseteq \mathcal{X}_0$ *and* $\tilde{\mathcal{X}} \subseteq \mathcal{X}$ *and a function* ρ *satisfying*

$$\rho(\mathbf{x}) \geq 0 \quad \forall \mathbf{x} \in \tilde{\mathcal{X}}_0 \tag{6.16}$$

$$\rho(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in \mathcal{X}_u \tag{6.17}$$

$$\nabla(\rho f)(\mathbf{x}) \leq 0 \quad \forall \mathbf{x} \in \tilde{\mathcal{X}} \tag{6.18}$$

then system is safe. $\mathcal{X}_r$ *from* $\mathcal{X}_0$*.*

It may be easier to find the function ρ defined in Theorem 6 using the CSC-FPS algorithm found in Chapter 4.

6.2.2 Smallest invariant set

6.2.2.1 Invariants set

A consequence of the barrier function theorem is the possibility to evaluate invariants.

Definition 7. *Consider the hybrid system* $\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho)$ *as defined in Section 5.2.1. An invariant* $\mathcal{S}(\ell)$ *for Location* $\ell \in \mathcal{L}$ *is a set that contains all the trajectories* $\Pi(\ell)$ *as introduced in* (5.3) *for some initial set* $\mathcal{X}_0(\ell)$*.*

For a hybrid system $\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho)$ and the unsafe sets $\mathcal{X}_u(\ell)$, $\forall \ell \in \mathcal{L}$, if one has found barrier functions $\beta_\ell(\mathbf{x})$, $\forall \ell \in \mathcal{L}$, then the sets $\mathcal{S}(\ell) = \{\beta_\ell(\mathbf{x}) \leqslant 0 \mid \mathbf{x} \in \mathcal{X}\}$ are invariant sets since all the trajectories of the hybrid system are contained in $\mathcal{S}(\ell)$.

6.2.2.2 Smallest Invariant Set

In many studies [TRSS01] the aim is to determine the smallest invariant for a system since it will enclose more tightly the trajectories of the system. The idea of the considered approach is to first calculate barrier functions to find the invariant sets. Once found, the algorithm may be called iteratively using as input the invariant set found in the previous iteration.

For hybrid system $\mathcal{H} = (\mathcal{X}, \mathcal{L}, \mathcal{X}_0, \mathcal{I}, f, \Gamma, \rho)$, in the first iteration one calls the algorithm described in Chapter 5 with an arbitrary unsafe region chosen in state space $\mathcal{X}$. As an output, one gets the barrier functions $\beta_\ell(\mathbf{x})$. The direct result is that the system $\mathcal{H}$ trajectories are contained in the set $\mathcal{S}(\ell) = \{\beta_\ell(\mathbf{x}) \leqslant 0 \ \mathbf{x} \in \mathcal{X}\}$ thus the states space function can be update $g_{\mathcal{I}}(\ell, \mathbf{x}) = \beta_\ell(\mathbf{x})$. The process remain the same for the next iteration, each time, an arbitrary unsafe region is taken from the state space and the state space is update each time with the new barrier functions. At the end one gets the following invariant $\mathcal{S}(\ell) = \{\beta_\ell(\mathbf{x})_1 \leqslant 0 \wedge ... \wedge \beta_\ell(\mathbf{x})_n \ \mathbf{x} \in \mathcal{X}\}$ with $\beta_\ell(\mathbf{x})_i$ the barrier obtained at the iteration i. The limitation for this approach is the way to chose the unsafe region for each iteration.

Example 20. *Consider the Van der Pol dynamical system described by the following equations*

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 \\ x_1 x_2 - 0.5 x_2^2 \end{pmatrix} \tag{6.19}$$

and the initial region is $\mathcal{X}_0 = \{\mathbf{x} \mid g_0(\mathbf{x}) \leqslant 0\}$ *such that* $g_0(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 2)^2 - 0.1$ *and the consider parametric barrier* $\beta(\mathbf{x}) = (\frac{x_1 + p_1}{p_2})^2 + (\frac{x_2 + p_3}{p_4})^2 - 1$. *For such system the smallest invariant method is used, in the following figures a representation is given that shows the method used for 4 iterations.*

The choice of unsafe regions remain an open problem. In the example 20 the strategy was to take the unsafe region in each quadrant of the state space, aiming to maximize the reduction of the invariant set size. The program was stopped after 5 minute of unsuccessful computation and a new unsafe region was considered.

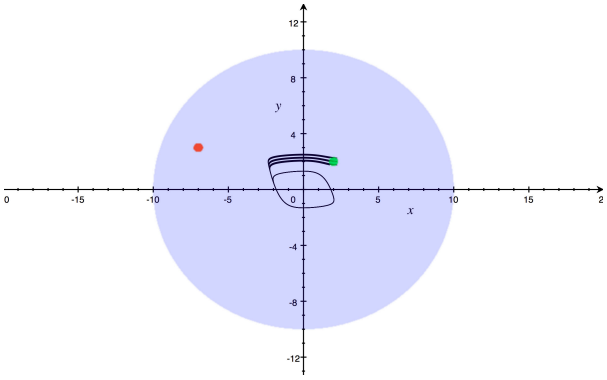


Figure 6.1: Iteration 0

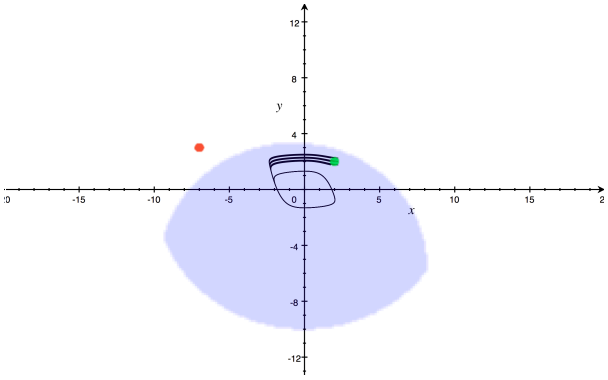


Figure 6.2: Iteration 1

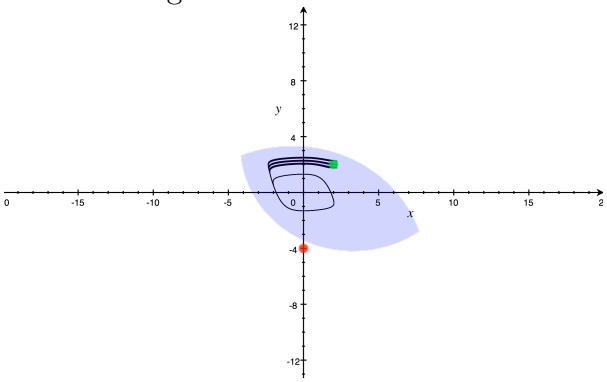


Figure 6.3: Iteration 2

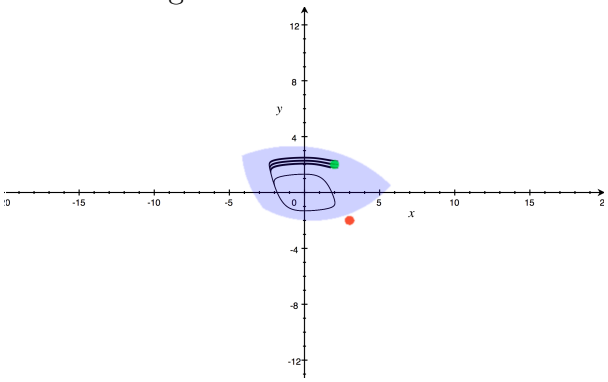


Figure 6.4: Iteration 3

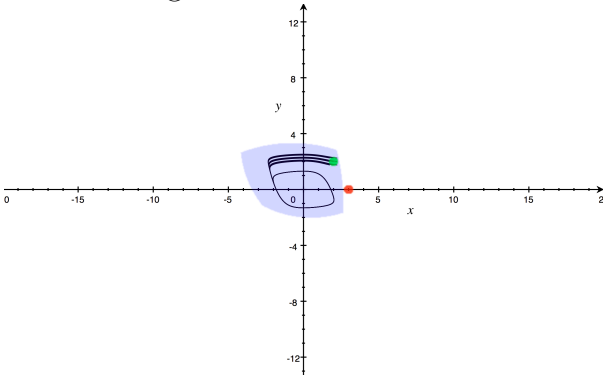


Figure 6.5: Iteration 4

Figure 6.6: In the figures the green circle represents the initial region, the red circles the unsafe region for each iteration, the light blue region the current invariant for the iteration, and the black line a simulation of the trajectories of the Van der Pol dynamic for discrete time instants and finite time $t = 0..10$.

Bibliography

[ABDM00] Eugene Asarin, Olivier Bournez, Thao Dang, and Oded Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *Proceedings of Hybrid Systems: Computation and Control*, pages 20–31. Springer, 2000.

[ACH+95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[AdSC16] Julien Alexandre dit Sandretto and Alexandre Chapoutot. Validated Explicit and Implicit Runge-Kutta Methods. *Reliable Computing*, 22, 2016.

[AKD+15] Nikos Aréchiga, James Kapinski, Jyotirmoy V. Deshmukh, André Platzer, and Bruce Krogh. Forward invariant cuts to simplify proofs of safety. In *Proceedings of IEEE Conference on Embedded Software*, pages 227–236, 2015.

[AKP11] Amir Ali Ahmadi, Miroslav Krstic, and Pablo A. Parrilo. A globally asymptotically stable polynomial vector field with no polynomial Lyapunov function. In *Proceedings of IEEE Conference on Decision and Control and European Control Conference*, pages 7579–7580, 2011.

[AP09] James Anderson and Antonis Papachristodoulou. On validation and invalidation of biological models. *BMC bioinformatics*, 10(1):132, 2009.

[ATNC12] Ignacio Araya, Gilles Trombettoni, Bertrand Neveu, and Gilles Chabert. Upper bounding in inner regions for global optimization under inequality constraints. *Journal of Global Optimization*, 60(2):145–164, 2012.

[BC63] Richard E. Bellman and Kenneth L. Cooke. Differential-difference equations. *RAND Corporation*, 1963.

[BG00] Frédéric Benhamou and Frédéric Goualard. Universally quantified interval constraints. In *Principles and Practice of Constraint Programming*, volume 1894 of *LNCS*, pages 67–82. Springer, 2000.

[BMT12] Andrew J. Barry, Anirudha Majumdar, and Russ Tedrake. Safety verification of reactive controllers for uav flight in cluttered environments using barrier certificates. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 484–490, 2012.

[BT14] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*. Springer, 2014.

[CAS12] Xin Chen, Erika Abrahám, and Sriram Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Proceedings of IEEE Real-Time Systems Symposium*, pages 183–192, 2012.

[CFH+03] Edmund Clarke, Ansgar Fehnker, Zhi Han, Bruce Krogh, Olaf Stursberg, and Michael Theobald. Verification of hybrid systems based on counterexample-guided abstraction refinement. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems*, pages 192–207. Springer, 2003.

[CH91] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, 1991.

[CJ09] Gilles Chabert and Luc Jaulin. Contractor programming. *Artificial Intelligence*, 173(11):1079–1100, 2009.

[CK03] Alongkrit Chutinan and Bruce H Krogh. Computational techniques for hybrid system verification. *Transaction on Automatic Control*, 48(1):64–75, 2003.

[DGXZ17] Liyun Dai, Ting Gan, Bican Xia, and Naijun Zhan. Barrier certificates revisited. *Journal of Symbolic Computation*, 80, Part 1:62 – 86, 2017.

[FLGD+11] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Proceedings of Computer aided verification*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011.

[Fre05] Goran Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *Proceedings of Hybrid Systems: Computation and Control*, pages 258–273. Springer, 2005.

[Gal86] J. H. Gallier. Logic for computer science. *Longman Higher Education*, 1986.

[GDT+12] Elena Glassman, Alexis Lussier Desbiens, Mark Tobenkin, Mark Cutkosky, and Russ Tedrake. Region of attraction estimation for a perching aircraft: A Lyapunov method exploiting barrier certificates. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2235–2242, 2012.

[GJPS14] Éric Goubault, Jacques-Henri Jourdan, Sylvie Putot, and Sriram Sankaranarayanan. Finding non-polynomial positive invariants and Lyapunov functions for polynomial systems through Darboux polynomials. In *Proceedings of IEEE American Control Conference*, pages 3571–3578, 2014.

[GKC13] Sicun Gao, Soonho Kong, and Edmund M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *Proceedings of Conference on Automated Deduction*, volume 7898 of *LNCS*, pages 208–214. Springer, 2013.

[GLGM06] Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Proceedings of Hybrid Systems: Computation and Control*, pages 257–271. Springer, 2006.

[GT08] Sumit Gulwani and Ashish Tiwari. Constraint-based approach for analysis of hybrid systems. In *Proceedings of Computer aided verification*, volume 5123 of *LNCS*, pages 190–203. Springer, 2008.

[GTV85] Roberto Genesio, Michele Tartaglia, and Antonio Vicino. On the estimation of asymptotic stability regions: State of the art and new proposals. *Transaction on Automatic Control*, 30(8):747–755, 1985.

[HHWT97] Thomas A Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: A model checker for hybrid systems. In *Proceedings of Computer aided verification*, volume 1254 of *LNCS*, pages 460–463. Springer, 1997.

[IBE] IBEX library. http://www.ibex-lib.org.

[JKDW01] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Éric Walter. *Applied Interval Analysis*. Springer, 2001.

[JW93] Luc Jaulin and Éric Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053 – 1064, 1993.

[JW96] Luc Jaulin and Éric Walter. Guaranteed tuning, with application to robust control and motion planning. *Automatica*, 32(8):1217–1221, 1996.

[KDSA14] James Kapinski, Jyotirmoy V. Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. Simulation-guided Lyapunov analysis for hybrid dynamical systems.

In *Proceedings of Hybrid Systems: Computation and Control*, pages 133–142. ACM, 2014.

[KGCC15] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dReach: δ-reachability analysis for hybrid systems. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *LNCS*, pages 200–205. Springer, 2015.

[KHS+13] Hui Kong, Fei He, Xiaoyu Song, William NN Hung, and Ming Gu. Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In *Proceedings of Computer aided verification*, volume 8044 of *LNCS*, pages 242–257. Springer, 2013.

[KSH+14] Hui Kong, Xiaoyu Song, Dong Han, Ming Gu, and Jiaguang Sun. A new barrier certificate for safety verification of hybrid systems. *Computer Journal*, 57(7):1033–1045, 2014.

[mat15] Mathcurve. http://www.mathcurve.com/, 2015.

[MP90] Zohar Manna and Amir Pnueli. A hierarchy of temporal properties. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 377–410, 1990.

[NJC99] Nedialko S. Nedialkov, Kenneth R. Jackson, and George F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21 – 68, 1999.

[Par03] Pablo A. Parrilo. Semidefinite programming relaxations for semi-algebraic problems. *Mathematical programming*, 96(2):293–320, 2003.

[PJ04] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Proceedings of Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

[PJP07] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[Pla07] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In *Proceedings of TABLEAUX*, volume 4548 of *LNCS*, pages 216–232. Springer, 2007.

[Pla10] André Platzer. *Logic analysis of hybrid systems*. Springer-Verlag, 2010.

[PP02] Antonis Papachristodoulou and Stephen Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In *Proceedings of IEEE Conference on Decision and Control*, volume 3, pages 3482–3487, 2002.

[PP05] Antonis Papachristodoulou and Stephen Prajna. Analysis of non-polynomial systems using the SOS decomposition. In *Proceedings of Positive Polynomials in Control*, pages 23–43. Springer, 2005.

[PPP02] Stephen Prajna, Antonis Papachristodoulou, and Pablo A. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver. In *Proceedings of IEEE Conference on Decision and Control*, volume 1, pages 741–746, 2002.

[PR05] Stephen Prajna and Anders Rantzer. Primal–dual tests for safety and reachability. In *Proceedings of Hybrid Systems: Control and Computation*, volume 3414 of *LNCS*, pages 542–556. Springer, 2005.

[Pra05] Stephen Prajna. *Optimization-Based Methods for Nonlinear and Hybrid Systems Verification*. PhD thesis, California Institute of Technology, Pasadena, California, 2005.

[Pra06] Stephen Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126, 2006.

[RA11] Andreas Rauh and Ekaterina Auer. Verified simulation of ODEs and DAEs in ValEncIA-IVP. *Reliable Computing*, 15(4):370–381, 2011.

[Rat06] Stefan Ratschan. Efficient solving of quantified inequality constraints over the real numbers. *ACM Transaction on Computational Logic*, 7(4):723–748, 2006.

[Rat07] Stefan Ratschan. Rsolver user manual. http://rsolver.sourceforge.net, 2007.

[RJ15] Muhammad Zakiyullah Romdlony and Bayu Jayawardhana. On the construction of control Lyapunov-barrier function. In *Proceedings of Benelux Meeting on Systems and Control*, page 22, 2015.

[RMM12] Rachid Rebiha, Nadir Matringe, and Arnaldo Vieira Moura. Transcendental inductive invariants generation for non-linear differential and hybrid systems. In *Proceedings of Hybrid Systems: Computation and Control*, pages 25–34. ACM, 2012.

[RS07] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Transacion on Embedded Computing Systems*, 6(1), 2007.

[RS10] Stefan Ratschan and Zhikun She. Providing a basin of attraction to a target region by computation of Lyapunov-like functions. *SIAM Journal on Control and Optimization*, 48(7):4377–4394, 2010.

[Rue05] Michel Rueher. Solving continuous constraint systems. In *Proceedings of International Conference on Computer Graphics and Artificial Intelligence*, pages 35–55, 2005.

[San11] Sriram Sankaranarayanan. Automatic abstraction of non-linear systems using change of bases transformations. In *Proceedings of Hybrid Systems: Computation and Control*, pages 143–152. ACM, 2011.

[SPW12a] Christoffer Sloth, George J. Pappas, and Rafael Wisniewski. Compositional safety analysis using barrier certificates. In *Proceedings of Hybrid Systems: Computation and Control*, pages 15–24. ACM, 2012.

[SPW12b] Christoffer Sloth, George J. Pappas, and Rafael Wisniewski. Compositional safety analysis using barrier certificates. In *Proceedings of Hybrid Systems: Computation and Control*, pages 15–24. ACM, 2012.

[SSM04] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Constructing invariants for hybrid systems. In *Proceedings of Hybrid Systems: Computation and Control*, pages 539–554. Springer, 2004.

[Stu02] Bernd Sturmfels. *Solving Systems of Polynomial Equations*. Number 97 in Conference Board of the Mathematical Sciences Regional Confe. Conference Board of the Mathematical Sciences, 2002.

[Tiw03] Ashish Tiwari. Approximate reachability for linear systems. In *Proceedings of Hybrid Systems: Computation and Control*, pages 514–525. Springer, 2003.

[TRSS01] Ashish Tiwari, Harald Rueß, Hassen Saïdi, and Natarajan Shankar. A technique for invariant generation. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems*, pages 113–127. Springer, 2001.

[VB96] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[VČ96] Antonin Vaněček and Sergej Čelikovskỳ. *Control systems: from linear analysis to synthesis of chaos*. Prentice Hall, 1996.

[YLW15] Zhengfeng Yang, Wang Lin, and Min Wu. Exact safety verification of hybrid systems based on bilinear SoS representation. *Transaction on Embedded Computing Systems*, 14(1):1–19, 2015.

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title : Proving the safety of dynamical systems using interval analysis

Keywords : Interval analysis, constraint satisfaction problems, Lyapunov functions

Abstract : This thesis addresses the problem of proving the safety of systems described by non-linear dynamical models and hybrid dynamical models. A system is said to be safe if all trajectories of its state do not reach an unsafe region. Proving the safety of systems by explicitly computing all its trajectories when its dynamic is non-linear or when its behavior is described by an hybrid model with non-linear dynamics remains a challenging task.

This thesis considers the barrier function approach to prove the safety of a system. A barrier function, when it exists, partitions the state space and isolate the trajectories of the system starting from any possible initial values of the state and the unsafe part of the state space. The set of constraints, which have to be satisfied by a barrier function are usually non-convex, rendering the search of satisfying barrier functions hard. Previously, only polynomial barrier functions were taken in consideration and for systems with polynomial dynamics.

This thesis considers relatively general dynamical systems with generic non-linear barrier functions. The solutions presented are based on template barrier functions, constraint satisfaction problems, and interval analysis.

The first part of the thesis focuses on non-linear dynamical systems. The barrier function design problem is formulated as a constraint satisfaction problem that can be solved using tools from interval analysis. This formulation allows one to prove the safety of a non-linear dynamical system by finding the parameters of a template barrier function such that all constraints are satisfied using the FPS-CSC algorithm, which has been adapted and supplemented with contractors to improve its efficiency.

The second part of the thesis is dedicated to the design of barrier functions for systems described by hybrid dynamical models. Safety properties have to be proven during the continuous-time evolution of the system, but also during transitions. This leads to additional constraints that have to be satisfied by candidate barrier functions. Solving all the constraints simultaneously to find all the barrier functions is usually computationally intractable. In the proposed approach, the algorithm explores all the locations sequentially. Transition constraints are introduced progressively between the already explored locations. Backtracking to previous location is considered when transition constraints are not satisfied.

The efficiency of the proposed approaches has been compared with state-of-the-art solutions.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Calcul par analyse intervalle de certificats de barrière pour les systèmes dynamiques hybrides

Keywords : Analyse par intervalles, satisfaction de contraintes, fonctions de Lyapunov

Résumé: Cette thèse développe des outils permettant de prouver qu'un système dynamique est sûr. En supposant qu'une partie de l'espace d'état est dangereuse, un système dynamique est dit sûr lorsque son état n'atteint jamais cette partie dangereuse au cours du temps, quel que soit l'état initial appartenant à un ensemble d'états initiaux admissibles et quel que soit le niveau de perturbation restant dans un domaine admissible. Les outils proposés cherchent à établir des preuves de sûreté pour des systèmes décrits par des modèles dynamiques non-linéaires et des modèles dynamiques hybrides.
Prouver qu'un système dynamique est sûr en calculant explicitement l'ensemble des trajectoires possibles du système lorsque le modèle dynamique est non-linéaire et perturbé reste une tâche très difficile. C'est pourquoi cette thèse aborde ce problème à l'aide de fonctions barrières paramétrées. Une barrière, lorsqu'elle existe, permet de partitionner l'espace d'état et d'isoler l'ensemble des trajectoires possibles de l'état du système de la partie dangereuse de l'espace d'état. La fonction paramétrique décrivant la barrière doit satisfaire un certain nombre de contraintes impliquant la dynamique du modèle, l'ensemble des états initiaux possibles, et l'ensemble dangereux. Ces contraintes ne sont pas convexes en général, ce qui complique la recherche de fonctions barrières satisfaisantes. Précédemment, seules des fonctions barrières polynomiales ont été considérées pour des modèles dynamiques polynomiaux.
Cette thèse considère des systèmes dynamiques relativement généraux avec des barrières paramétriques quelconques. Les solutions présentées exploitent des outils de satisfaction de contraintes sur des domaines continus et des outils issus de l'analyse par intervalles.
Dans un premier temps, cette thèse considère des systèmes dynamiques non-linéaires à temps continu. Le problème de conception d'une barrière paramétrique est formulé comme un problème de satisfaction des contraintes sur des domaines réels avec des variables quantifiées de manière existentielle et universelle. L'algorithme CSC-FPS a été adapté afin de résoudre le problème de synthèse de barrière. Cet algorithme combine une exploration de l'espace des paramètres de la barrière et une phase de vérification des propriétés de la barrière. à l'aide de contracteurs, il est possible de significativement accélérer la recherche de solutions.
Dans un second temps, ces résultats sont étendus au cas de systèmes décrits par des modèles dynamiques hybrides. La propriété de sûreté doit être prouvée lors de l'évolution à temps continu du système dynamique, mais aussi pendant les transitions du système. Ceci nécessite l'introduction de contraintes supplémentaires qui lient les fonctions barrières associées à chaque mode à temps continu entre elles. Réaliser la synthèse de toutes les fonctions barrières pour les différents modes simultanément n'est envisageable que pour des systèmes de très petite dimension avec peu de modes. Une approche séquentielle a été proposée. Les contraintes liées aux transitions sont introduites progressivement entre les modes pour lesquels une barrière a déjà été obtenue. Lorsque certaines contraintes de transition ne sont pas satisfaites, une méthode de *backtracking* doit être mise en œuvre afin de synthétiser des barrières offrant une meilleure prise en compte des contraintes de transition non satisfaites.
Ces approches ont été évaluées et comparées avec des techniques de l'état de l'art sur des systèmes décrits par des modèles à temps continu et des modèles hybrides.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France