Automate sur les structures temporisée

Samy Jaziri

université
PARIS-SACLAY

école
normale
supérieure
paris—saclay

Automata on Timed Structures

Thèse de doctorat
préparée à l'École Normale Supérieure Paris-Saclay
au sein du Laboratoire Spécification & Vérification

École doctorale n°580 Sciences Et Technologies De L'Information Et De La Communication (STIC)
Spécialité de doctorat: Pôle 4 - Programmation : modèles, algorithmes, langages, architecture

Présentée et soutenue à Cachan, le 24 septembre 2019, par

Samy Jaziri

Composition du jury :

Eugène ASARIN Professeur, Université Paris-Diderot	Président Rapporteur
Stavros TRIPAKIS Professeur, Northeastern University	Rapporteur
Thomas CHATAIN Maître de Conférence, ENS Paris-Saclay	Examinateur
Frédéric HERBRETEAU Maître de Conférence, Université de Bordeaux	Examinateur
Patricia BOUYER Directeur de Recherche CNRS, ENS Paris-Saclay	Encadrant
Nicolas MARKEY Directeur de Recherche CNRS, Université de Rennes	Encadrant

NNT : 2019SACLN039

Thèse de doctorat

Automata on Timed Stuctures

Doctoral Thesis

Samy Jaziri

This thesis have been supported by ERC project EQualIS

ISBN: 1234567890

À ma famille. La grande, la petite, la nadinesque, et la future.

Remerciements

Le chemin qui m'emmène aujourd'hui à écrire ces lignes est une réelle aventure de quatre ans. Probablement les quatre années les plus intenses de ma vie jusqu'ici mais de loin pas les plus simples. À de nombreuses occasions les choses auraient pu prendre une autre tournure n'eût été le soutien de plusieurs personnes.

Tout d'abord rien n'aurait été possible sans l'accompagnement de Patricia et Nicolas, mes encadrants, tout au long de ma thèse. Ce ne fut pas tout le temps facile pour eux aussi et je tiens à les remercier chaleureusement de n'avoir pas abandonné dans les moments les plus difficiles de la thèse. Ils ont accepté de se lancer avec moi dans un nouveau sujet après un an de thèse et je leur en suis reconnaissant. Je vous remercie aussi pour votre suivi régulier tout au long de mon doctorat, même après le départ de Nicolas à Rennes et même après ma prise de poste en quatrième année. Tous les doctorants n'ont pas cette chance. Par ailleurs je pense que ce n'est pas qu'une question de chance et que l'implication et l'accompagnement dont ont fait preuve mes encadrants font partie d'une politique plus globale du LSV et je veux en remercier tous ses membres ; qui travaillent chaque année à faire de ce laboratoire un espace fraternel où l'on prend soin de ses doctorants. Mes encadrants ne sont d'ailleurs pas les seuls à m'avoir accompagné dans les périodes difficiles. Je remercie en particulier Stéphane Demri pour ses conseils, sa bienveillance et son suivi ; Frederic Blanqui pour son suivi annuel de tous les doctorants et du mien en particulier ; Alain Finkel qui m'a fait prendre conscience de l'importance de prendre des vacances ; et bien d'autres qui par leurs discussions ou leurs soutiens m'ont encouragé à mener ce travail à terme, David, Sylvain, Thomas et probablement d'autres que j'oublie . . . J'espère que le déménagement à Saclay n'ébréchera pas trop ce que les différents membres du LSV ont construit à Cachan.

Et je n'aurais pas pu aller jusqu'au bout du chemin sans le dévouement d'autre membre de la communauté scientifique. Je tiens à remercier Frédéric Herbreteau et Thomas Chatain de faire partie de mon jury. Je remercie aussi mes rapporteurs pour leur flexibilité sur le calendrier et pour qui la tâche de lire ce manuscrit très technique durant l'été n'a pas dû être facile : Eugène Asarin – qui fut le premier chercheur dont je lis un papier, c'est une grande joie pour moi que vous fussiez l'un des premiers à lire mon manuscrit – et Stavros Tripakis, une partie de mon travail de thèse consista à essayer de généraliser et améliorer un de vos travaux et c'est pour moi un bonheur de pouvoir être évaluer par celui qui m'a inspiré.

Ma vie quotidienne au laboratoire (ou disons presque quotidienne) fut toujours agréable et cela fait à mon avis parti des facteurs qui aident les doctorants à s'accrocher jusqu'au bout. J'y ai en particulier rencontré des gens déterminants pour la suite de ma carrière. Je remercie d'abord Hubert Comon, pour avoir réussi à me transmettre lors de ses cours le plaisir, à la fois de l'informatique et de l'enseignement. Je remercie Claudine Picaronny, pour sa lettre de recommandation, son enseignement et ses conseils en préparation à l'agrégation (qui sont peut-être les seuls dont j'ai un bon souvenir), et le semestre de math discrètes qui fut ma première expérience d'enseignement. Je remercie David Baelde, lui aussi pour sa lettre de recommandation, pour les tapas à Madrid, et pour la précieuse expérience que j'ai eu en en-

cadrant le projet Génie Logiciel avec toi. Nous n'avons peut-être pas réussi à trouver la forme optimale d'organisation pour gérer efficacement et démocratiquement un projet informatique mais peut-être que je n'aurais pas choisi l'enseignement en IUT si je n'avais pas pris autant de plaisir dans ce cours ! Je remercie Jean Goubault, pour ses anecdotes évidemment, mais aussi pour ses conseils et pour m'avoir considéré pour faire Jury à l'agrégation d'informatique au Maroc.

Je remercie aussi vivement ceux sans qui le laboratoire se serait effondré en moins d'une semaine, et sans qui l'ambiance de travail serait probablement plus tendue. Catherine, qui s'occupait déjà de nous sauver étudiants et qui dû continuer une fois membre du laboratoire : merci de ne pas m'avoir trop engueulé quand, après m'avoir payé un billet pour New York, je me suis rendu compte que j'avais confondu la ville et l'État de New York... Imane, même si j'ai eu moins souvent affaire à toi, merci de t'occuper de toute l'administration dont on ne soupçonne même pas l'existence... Virginie, merci d'avoir engueulé chaque année le CNRS pour que je puisse recevoir mon salaire avant l'année d'après. Et merci d'avoir élevé le niveau pour les parties de babyfoot à Barbizon ! Merci à Francis pour m'avoir gentiment rappelé qu'Hugues Mandon $\neq$ Hugues Moretto ! Et aussi pour tous les conseils pour Sarah ! J'espère que tu te plais à Jussieu. Merci à Hugues pour sa bonne humeur, pour son travail, et pour avoir pris le rôle de plus gros mangeur du LSV (on a pu s'empiffrer l'esprit tranquille nous derrière comme ça !)

Je ne sais pas comment a fait le tout premier des doctorants, mais il est certain que pour ma part, sans l'aide, les conseils et l'exemple de nos prédécesseurs, la tâche aurait été plus rude encore ! Je remercie Guillaume, pour la belotte après manger, pour avoir pris au sérieux notre débat sur les légumes et les légumineuses lors du TD de complexité (les pois chiches sont des légumineuses, si c'est vrai !), pour m'avoir convaincu que les sciences sociales sont des sciences et surtout parce que sans lui, je ne me seraient probablement jamais ouvert l'esprit sur la philosophie, l'économie, l'histoire, les sciences sociales et toutes ces sous sciences ;p ! Merci à Marie pour m'avoir fait profiter de la terrasse du quatrième (depuis que tu es partie j'y ai mis rarement les pieds et c'est dommage...), pour les discussions enrichissantes, pour m'avoir fait rencontrer autant de gens intéressants et pour avoir complimenté mon topo de "logique", ça m'a fait très plaisir ! Merci à Rémi de m'avoir fait découvrir qu'il existe des centraliens sympas et intelligents ! Merci à Lucca pour les discussions sur la crypto, le monde diplomatique et le rôle des chercheurs après la révolution ! Merci aussi pour la semaine à Madrid, j'en garde un bon souvenir. Merci Jeremy pour nos discussions sur les catégories, les bisimulations, les ultrafiltres mais aussi l'actualité ! Merci pour tes conseils et ton soutien !

Les prédécesseurs sont importants, mais les camarades thésards le sont peut-être encore plus. Merci à mes camarades (éphémères) de bureau. Simon pour la batterie sur table et sol, Igorrr, les balades à la boulangerie, les discussions sur la logique et désolé pour toutes les fois où je n'étais pas là et que tu te faisais cuisiner ! Antoine pour les discussions sur Hegel et la suprématie du capitalisme, pour avoir ouvert et fermer le bureau tous les jours, pour emmener un peu de sérieux dans ce bureau et pour ne jamais avoir râlé parce que, sérieux, on l'était rarement ! Seya, même si on ne l'a pas été longtemps je crois, pour la création du contre-RU, pour les templates et la page web, pour la config i3 et pour avoir surveillé Simon en mon absence. Merci aussi à tous ceux que j'ai vus moins souvent mais avec qui on a toujours passé de bons moments. Merci Hugues pour avoir fait semblant que je ne ronflais pas à Barbizon et avoir rappelé à Charlie que ce n'était pas sympa d'organiser le Kifékoi le jour de ma soutenance ! Merci Alessio pour tes conseils pour faire une bonne bolognaise. Merci Nathan pour la plus belle barbe que j'ai pu voir jusqu'à aujourd'hui. Merci à Adrien et Marie de soutenir après moi et de ne pas me laisser être le dernier !

Maintenant que je suis prof moi aussi je pense à eux, qui ont participé à me donner goût

aux sciences, aux études et à l'enseignement. Merci à vous, M. Carcone, Mme Degorce, M. Dauvignac, Mme Grimaud, M. Roussel, pour ne citer que vous. Merci aussi à mes collègues actuels à l'IUT. Pour leur compréhension et leur accueil. Malgré ma prise de poste, vous avez facilité grandement ma tâche en rendant mes débuts à l'IUT agréables et adaptés à la charge de travail que demande la rédaction d'une thèse. Merci en particulier à Benjamin, Cécile, Chantal R., Chantal K., Hélène, Rafael, Thomas, Yacine et Xavier.

Il m'aurait été impossible de tenir le rythme et la pression de ces quatre années de thèse sans le soutien de mon entourage et à eux tous un énorme merci. Votre présence fut plus importante tous les conseils et les réunions que j'ai pu avoir. Et ceux depuis les années de prépa qui connurent leur lot de difficulté. Alors d'abord merci à Laure, Louise, Marie, Oriane, Pauline et Théo pour la belotte, les coups de pied au cul pour que je me secoue et que je bosse plus, la patinoire, la foire, le ski et tous les bons moments qui nous ont permis de surmonter les difficultés de la prépa.

Merci à tous les copains, pour tout ce qu'ils m'ont apportés durant ces quatre dernières années, et merci à eux de m'avoir encouragé à terminer ma thèse même si ça voulait dire m'éloigner. . .

Nadines, vous avez bien nadinez ces quatre dernières années et je vous nadine pour ça. Sans la nadination il est certain que j'aurais eu beaucoup de mal à nadiner cette thèse ou même à m'en nadiner aussi bien dans mes études à l'ENS ou dans la vie. Je vous nadine d'avoir bien nadiner, même quand moi je nadinais pas forcément et j'espère qu'on nadinera longtemps encore ! Même si on se nadine parfois ou qu'apparaissent au fur et à mesure des mini-nadines, vous faîtes tous un peu parti de ma vie comme de cette thèse !

Les nadines sont une grande nation, on s'y sert les coudes et on partage beaucoup de choses. Sans ce cadre de vie ça aurait été plus dur de remonter la pente quand j'étais dans les creux de la thèse. Merci à Skippy, le premier nadine (alors pas encore nadine) que j'ai connu. Merci pour le premier burger dans la chambre crous, merci de nous avoir coaché sur LoL, la recette des tomates farcies, merci pour look at my horse et pour ton coup de main dans tous les projets de première année ! Merci à Malfoy, tu fus le nadine qui partit en premier et pourtant celui qui nous influença le plus. En un an tu as fondé et soudé une nation entière. Merci pour tes histoires et ta bonne humeur ! Merci à Camille de nous avoir rappelé que le nom de Malfoy est en réalité Antoine ! Et pour les photos qui nous rappellent Jessy lors du meilleur pot que Cachan a connu. Merci à Joris, pour les what the fuckeries quotidiennes, pour avoir pu compter sur toi quand il le fallait et pour avoir fait un bout de chemin avec moi durant ces années de thèses, mais plus du côté de place d'Italie et de Presles que de Cachan. Merci à JJ aka Romaing, pour nos houleux débats sur la morale, l'exemplarité au niveau des passages piétons, la religion, le rôle de l'État, la logique vs l'algo, les probabilités et bien d'autres. Excuse-moi parfois on n'a pas fini très sereinement le débat ! Mais avec le recul, tu fais sûrement partie des gens qui m'ont donné goût à la réflexion et au débat sur ces sujets et d'autres ;) ! Merci Japan Boy, pour avoir emmené un peu de goût et de sagesse au pays des Nadines, pour ta bienveillance, pour avoir été le deuxième amateur de foot à entrer dans le groupe : on a enfin pu regarder un match grâce à toi ! Merci à Seya, pour les soirées ciné et les aprèms jeux vidéo, les génériques de GoT et les photos de salle de bain retouchés. C'était cool de s'aérer l'esprit dans les périodes difficiles de la thèse et avoir un nadine amateur de Marvel et de Smash Bros c'était salvateur ! Merci aux Présidents, de garder la nation unis et de la gérer avec générosité et attention ! Vous êtes les seuls présidents contre qui je ne prônerai pas la révolution. Long may you reign ! Merci à Président pour les cours de guitare, les botlanes Corki Leona, les parties d'EE, les rappels des deadlines, les voyages en panique à Massy, la proposition pour l'agrégation au Maroc et la fois où tu n'as pas dit à Malika que je séchais pour jouer à LoL. Merci Présidente de gérer la nation en secret toute seule parce

qu'on sait bien que Président en vrai il ne pourrait pas gérer aussi bien. Merci pour les japs où on parlait que d'informatique, pour toutes les fois où tu nous as accepté même si on clamé que la chimie c'est de la cuisine, pour les sushis maison, pour avoir accepté qu'une inconnue vienne à ton mariage, et pour avoir rapidement supporté les amis bizarres de ton mari. Merci Mini-Prez pour l'idée de la chambre de bébé qui doit être à bonne température ! Pour tes sourires aussi ! Merci Davy, pour les batailles sur Minecraft, pour nous avoir fait tenir encore un petit moment sur LoL, pour toutes les pizzas, pour les matins où c'était difficile d'aller à P7 mais qu'on s'encourageait à prendre le bus, pour le meilleur projet que je n'ai jamais rendu ! Merci de m'avoir fait découvrir le terme anarcho-capitalisme et je m'excuse qu'on n'ait pas pu en discuter plus longuement et plus sereinement à ce moment-là ! Merci pour ta générosité et tous tes coups de main ! Merci à Doc, plus vieux citoyen du pays des Nadines, d'avoir tué dans l'œuf toute compétition pour savoir qui serait le premier docteur Nadine. Merci pour les discussions en bas du I, sur l'économie, le capitalisme, la religion, et probablement d'autres sujet qui était bien plus intéressant que Frobenius-Zolotarev, sujet principal de discussion de cette époque. Merci d'avoir essayé d'élever le standing des jeux de société auxquels on joue ! Continue à militer ça payera. Longue vie à l'Internationale des Nadines.

À Echo et Alpha, je ne connais personne plus heureux de me voir après une heure de séparation que vous. Merci ça fait chaud au cœur ! Merci pour les balades qui m'ont fait sortir prendre l'air pendant la rédaction ! À Sushi, Waza et Nala, merci pour votre aide précieuse pendant la rédaction. Waza merci d'avoir fait en sorte de garder mes genoux bien chauffés et d'avoir entretenu la discussion de temps à autre. Nala merci d'avoir fait attention que je ne puisse pas rédiger plus de cinq minutes d'affiler en ramenant régulièrement ta balle et en sautant sur la souris si j'essayais de t'ignorer. Sushi merci pour les discussions philosophiques et pour avoir surveillé attentivement, couché juste au-dessus de l'ordi, que je travaillais bien sur mon manuscrit. Merci à tous les trois pour vos photos et votre aide pour les slides. Et surtout merci pour avoir empli notre foyer d'amour dans les moments difficile de la vie comme de la thèse.

À Alexis, merci de m'avoir fait découvrir l'informatique et bien d'autres choses. Merci pour nos projets, pour nos soirées EE2, pour le Futuroscope, pour les balades au Bourran, pour avoir testé le Leblabi pour moi, et pour tous les bons souvenirs, de jeunesse, de coloc et tous les autres. J'ai la chance d'avoir un ami sur qui compter depuis aussi longtemps et durant ces quatre années de thèse ça m'a fait souvent du bien !

À ma future belle-famille, Dominique, Suzanne, Simon, Coline, Louis et Auguste, merci de m'avoir accepté si vite et de m'avoir accueilli si bien ! Merci d'avoir été compréhensif les fois où je devais travailler mon manuscrit dans mon coin. Merci pour tous les moments de joie qu'on a partagés pendant la période de rédaction et qui m'ont fait du bien !

Maintenant à vous tous qui êtes là pour moi et me soutenez depuis bien plus longtemps que tous ceux cités aux dessus, un immense merci. Vous avez fait l'homme qui a écrit cette thèse, après tout, c'est un peu la vôtre aussi. À toutes mes tantes, Besma, Narjes et Nadia qui depuis mon plus jeune âge m'ont empli d'amour. À mon oncle et sa femme, Moncef et Wahiba, sur qui j'ai pu toujours compter et qui me traitent comme leur fils. À ma grand-mère, Salouha, qui, illettrée, victime du régime colonial, voit tous ses petits-enfants faire des études supérieures et parmi lesquels elle pourra compter plusieurs docteurs. À mon grand-père, Neji, que je n'ai pas connu mais dont les histoires bercent mon enfance. À mon grand-père, Guy, désolé je n'ai pas voulu faire président de la République même si j'ai fini par m'intéresser à la politique, mais je compte être docteur bientôt, j'espère que ça te conviendra tout de même ! À ma grand-mère, Josette et ma tante Céline, pour les bons souvenirs que je garde des moments passés avec vous, même s'ils ne sont sûrement pas assez nombreux. À mon frère Yassin, pour tous nos moments de jeunesse partagés ensemble, les vacances Pokémon, les étés Borderland, les

matchs, les nuits blanches, les bagarres entre nous ou contre les autres . . . À ma sœur Lilia, même si ça n'a pas été toujours facile entre nous, tu m'as apporté énormément dans la vie et tu m'as aussi aidé à un moment donné dans l'aventure qu'a été ma thèse. Pour tout ça merci. Nous avons grandi ensemble tous les trois, et les bons moments comme les moins bons font partie de ce qui m'a construit et m'a donné l'énergie de mener ces longues études et les réussir. Même si nous faisons chacun nos vies, je sais que je peux compter sur vous deux pour encore partager de nombreux moments de bonheur avec vous. À Mariam, ma petite sœur qui a bien grandi ! Merci de me sauter dans les bras à chaque fois que je reviens depuis que j'ai quitté la maison et jusqu'à aujourd'hui ! Désolé d'être parti si vite, tu as su tenir la maison pendant tout ce temps. Je suis heureux de te voir réussir dans ce que tu entreprends et de te voir devenir une jeune femme aussi talentueuse ! Bon courage pour les années à venir !

À mes parents, c'est à vous que vont mes plus grandes pensées en ce moment. Si tous ceux que j'ai cité au-dessus ont eu un rôle à jouer dans ma réussite jusqu'ici, il va sans dire que personne ne s'est plus investi, n'a plus donné et n'a plus de mérite que vous. Votre amour, votre investissement et votre éducation sont les artisans de l'homme que je suis aujourd'hui et je vous en suis reconnaissant. Même dans les pires moments de cette thèse, comme dans mes études, vous n'avez jamais perdu confiance en moi et vous m'avez soutenu de toutes les manières possibles. J'espère obtenir le titre de docteur pour le travail que j'aurais accompli les quatre dernières années. Mais vous aurez alors mérité un doctorat dans l'art d'être parent pour vos 28 dernières années de travail.

À toi, dont nos quatre ans d'histoire coïncident avec mes quatre années de thèse ; à toi qui m'as apporté du réconfort lors de la période la plus difficile de la thèse ; à toi qui m'as apporté chaque jour du soutien moral et materiel lors des longues journées et nuits de rédaction ; à toi qui m'as fait connaître le bonheur d'aimer et d'être aimé en retour et qui a tout rendu supportable ; à toi merci, merci d'avoir dit oui.

Contents

Introduction

We now live in a digital society. Some digital systems might just be helpful in everyday life : GPS-based navigation, connected home, home appliance... If they fail it usually causes in the worst case an annoying discomfort. But as we trust more and more digital systems to handle complex tasks, many lives become dependent on their safe execution. Errors in software controlling airplanes or automatic subways could, in an instant, put hundreds of people's lives at risk. Now that robots can provide surgical assistance, a bug in their execution could be lethal for the patient. Now that we trust software with more and more personal sensible data, a flaw could expose millions of users to impersonation. Now that algorithms rule the financial system, who knows if some faulty algorithm could provoke the next economic crisis... The next generation may even not be able to live without the assistance of digital systems, somehow like the previous generation couldn't imagine running a society without steam engines or electricity.

As digital systems take more power in society, they must assume more responsibility. This responsibility falls on the engineers and computer scientists who design, develop and study those systems. Digital systems execution errors might come from implementation errors – which can be detected and corrected only with the execution of the system – or from design error – which can be detected and corrected in advance by the study of models of the real system. Formal methods can be of help to detect them in both cases. For design matters, formal verification can be used to prove or disprove properties of the said design. The system is thus described as a mathematical model – such as automata and extensions thereof – in order to represent and reason about its behaviors. Various algorithmic techniques are then applied to ensure safety, liveness, or any set of properties expressed in some fixed logical language. Among them we can cite, model checking algorithms [26] [25], deductive verification [37] [31] or testing [51]. Those methods are usually costly in terms of computation, and sometimes too strong a requirement. On another hand, the implementation phase can add errors independently from the correctness of the design. For those kinds of errors, the methods described previously provide no help. Runtime verification instead aims at checking properties of a running system [40]. Runtime methods can then be used in a testing phase but also during real system execution to alert about possible errors and try to prevent them to happen. In comparison to formal verification, runtime verification can't ensure general safety but they usually can be used in a wider range of cases and perform often better. Fault diagnosis is a prominent problem in runtime verification: it consists in (deciding the existence and) building a diagnoser, whose role is to monitor real executions of a (partially-observable) system, and decide on-line whether some property holds (e.g., whether some unobservable fault has occurred) [47] [52]. The related problem of prediction, a.k.a. prognosis, (that e.g. no fault may occur in the next five steps) [34], is also of particular interest in runtime verification.

There exists a large variety of digital systems with different complexity levels and different constraints. To represent them a large variety of mathematical models have been introduced, focusing on one or another aspect of the system, depending on its nature and on the properties we aim to ensure. Real-time systems are of particular interest. These are hybrid systems

made of both physical parts and software parts. They usually obey quantitative constraints based on physical parameters such as time, temperature, pressure... Discrete models, such as finite-state transition systems, are not adequate to model such quantitative constraints. They have been extended in various ways to be used for verification on real-time systems [35] [2] [50] [3] [22] [44]. Timed systems (which involve only time as a physical parameter) have their own set of models [4],[42], [36], [2], [17]. Timed automata [4], developed at the end of the eighties, provides a convenient framework for both representing and efficiently reasoning about computer systems subject to real-time constraints. They have been restricted and extended in many ways [6] [5] [1] [19] [49]. Efficient off-line verification techniques for timed automata have been developed [12] and have been implemented in various tools such as Uppaal [11], Kronos [23] and RED [55]. The diagnosis of timed automata, however, has received less attention. A diagnoser can usually be built (for finite-state models) by determinizing a model of the system, using the powerset construction; it will keep track of all possible states that can be reached after each (observable) step of the system, thereby computing whether a fault may or must have occurred. For timed automata this problem is made difficult by the fact that timed automata can in general not be determinized [54] [33] [4]. This has been circumvented by either restricting to classes of determinizable timed automata [5], or by keeping track of all possible configurations of the automaton after a (finite) execution [53]. The latter approach is computationally very expensive, as one step consists in maintaining the set of all configurations that can be reached by following (arbitrarily long) sequences of unobservable transitions; this limits the applicability of the approach.

In this thesis, we address the problem of efficient diagnosis for timed automata. This problem has already been addressed in [18] with the restriction that the diagnoser should be himself a timed automaton. The authors prove that such a diagnoser can be constructed for some restricted classes of timed automata known to be determinizable. Then they propose a game approach to construct a diagnoser when it is possible. Our approach is different and follows the idea of [53]. We construct a diagnoser which is not necessarily a timed automaton itself but still can be computed. The challenge is then to make this diagnoser so it is computable and efficient enough to on-line monitor the system. The construction of the diagnoser is made possible by prior results of ours about the determinization of timed automata within a wider class of timed systems. This class called *automata on timed structures* is introduced the first time by us in a publication at *FORMATS 2017* conference [20] (under the name of *automata on timed domains*). We prove in this paper that timed automata fit in this class of timed models and that within this class it is always possible to construct from an automaton without silent transitions a deterministic automaton finitely representable and computable [20]. We managed to use this deterministic automaton for one-clock timed automata diagnosis and proved that it allows displacing a heavy part of the computation time as pre-computing calculations. This is the subject of our publication at *RV 18* conference [21]. We provide an implementation of this diagnoser and compare it with an implementation we've made of the diagnoser of [53] (sources accessible at `http://www.lsv.fr/~jaziri/DOTA.zip`). In this thesis, we improve the formalism proposed in [20] and propose an improved proof for the determinization results found in this paper. We discuss the various implication of those results in greater detail and try to put them in perspective with already known determinization results for subclasses of timed automata, in particular with input-determined automata. We integrate the work done [21] which now perfectly fits within the formalism of automata on timed structures.

The precise outline of this thesis is given below. This manuscript is split into three parts. Parts one and two are based on the work done in [20] and part three is based on the work done in [21].

The first part is dedicated to the introduction of the model of automata on timed structures. After some basic notions, we introduce part by part the different layers of the formalism we use in the next parts.

Chapter 1 : This chapters present some basic notions about transition systems and the associated equivalence notions. This chapter does not present original work and is inspired by common computer science knowledge. Though we took the liberty to adapt some definition to fit our needs for this thesis and we advise special care for the reader used to those notions in their more classical definition.

Chapter 2 : In this chapter we introduce *timed structures*, a formal framework to model quantitative variables.

Chapter 3 : We introduce automata on timed structures, a quantitative model which is made of both a continuous part, inherited from the timed structure and a discrete part in the form of a finite automaton build on top of this timed structure. An idea which is at the basis of hybrid automata [35]

Chapter 4 : Controls are an original idea of ours which allow to split the definition of the model and the definition of its observable actions and hence of its recognized languages. They provide a nice setting to speak about different recognized languages and better understand the case of event-clock timed automata.

The second part presents the determinization results obtained in [20] and how each already known determinization results compare to each other in the framework of automata on timed structures.

Chapter 5 : In this chapter we prove that an automaton on timed structures without silent transitions can be determinized using a powerset construction. We also show that the deterministic automata on timed structures constructed is finitely representable and computable.

Chapter 6 : This chapter is a technical interlude for the application of the results of the previous chapter.

Chapter 7 : We use then the results of last two chapters to recover some known determinization results about timed automata. We compare them to each other in our framework to get more insight about their similarities and their differences.

Chapter 8 : In this chapter we use automata on timed strutures to recover determinization results on input-determined automata and discuss, with the new point of view given by our formalism, what could make those systems determinizable.

Finally last part is dedicated to the use of our powerset construction for determinization in the context of fault-diagnosis.

Chapter 9 : In this first chapter we introduce the problem of fault diagnosis in the framework of automata on timed structures and see how the work done in part two can be of use for the construction of an efficient diagnoser for one-clock timed automata.

Chapter 10 : This chapter introduce the data structure we use to store sets of clocks in the diagnoser. We call them regular timed markings and they have the good properties of being finitely representable, of allowing efficient calculation and can be used to store information during a pre-calculus phase.

Chapter 11 : We construct in this chapter using to regular timed markings, the final diagnoser and discuss its computability.

Chapter 12 : Finally in the last chapter of this thesis we present our implementation of a diagnoser for one-clock timed automata, and compare it to an implementation of a diagnoser of [53].

Part I

Timed Systems Modeling

Chapter 1

Basic Notions on Transition Systems

Modeling Dynamic Systems

All physical systems in nature – and therefore all software dedicated to control them – are *dynamic systems*. Those systems are in evolution, which means that all the parameters describing a particular state of the system vary according to intrinsic physical laws of evolution or external interventions. To ensure, with the help of formal methods, safety properties on such systems, we need models which encompass the whole variety of the possible configurations of the system *in their evolution*.

Transition systems model the evolution of a dynamic system by giving the means to entirely describe the reachability links between all its possible configurations. In some cases, for example in the modeling of a – not too complex – software, the system can be described by a finite set of such possibilities. Finite transition systems benefits of a robust formalism and a large variety of problems about them can be algorithmically solved [8]. In other cases, like for systems with physical quantities, the system cannot be modeled with a finite transition system. Several different models can then be used depending on the needs. Infinite transition systems remain though the main formal object used to define the semantic of those models (timed transition systems [36] [2], weighted transition systems [50], stochastic transition systems [3], game structures [22],...). The theory around the infinite case is not as robust as in the finite case, but the recent work of [32] on well-structured transition systems, provide both an elegant theory to work in and algorithmic tools to effectively study them with formal methods.

In the infinite case, the one we are interested in this thesis, transition systems are still too generic to allow finite representation or implementation. That is the reason why – even though we come back to transition systems when it comes to defining a formal semantics – weaker models are usually preferred when it comes to efficiently represent a dynamic system (timed automata [4], timed Petri net [42], weighted automata [44], ...). This also explains the interest of the community in the research and computation of *similarities* between transition systems, aiming to link a transition system to a simpler or more efficient equivalent version. The most famous notion of similarity being the notion of *bisimulation equivalence*.

In this chapter, we will give a selected reminder of the formal framework we are going to work in. We present definitions and results which are not part of our contribution but have to be credited to many of our predecessors in this field. We freely adapted some of them to suit our particular needs for this thesis. First we recall some definitions and define some notations about partial functions (section 1.1) – at the basis of our definition of *bisimilation*. We recall then some definitions about transition systems (section 1.2) and define the notations used throughout this thesis. In last section 1.3 we formally introduce the notion of *functional bisimulation* which is weaker than bisimulation equivalence but adapted in the context of this

thesis.

1.1 Partial Functions

We give in this section a quick reminder of the definition of partial functions and the objects and notations, we will be using throughout this thesis.

Definition 1.1.1. *Given two sets E_1 and E_2, a partial function from E_1 to E_2, written $f : E_1 \rightharpoonup E_2$, is a relation $f \in E_1 \times E_2$ such that for all $e_1 \in E_1$ there is at most one element $e_2 \in E_2$ such that $(e_1, e_2) \in f$.*

The set of partial functions from E_1 to E_2 is written $[E_1 \rightharpoonup E_2]$.

Let $f : E_1 \rightharpoonup E_2$ be a partial function. We write

$$\mathbf{dom}(f) = \{e_1 \in E_1 \mid \exists e_2 \in E_2, (e_1, e_2) \in f\}$$

Given $e_1 \in E_1$ and $e_2 \in E_2$, we write $f(e_1) = e_2$ for $e_1 \in \mathbf{dom}(f)$ and $(e_1, e_2) \in f$.

If $g : E_1 \rightharpoonup E_2$ is another partial function and $e_1 \in E_1$, $f(e_1) = g(e_1)$ implies $e_1 \in \mathbf{dom}(f) \cap \mathbf{dom}(g)$ or $e_1 \notin \mathbf{dom}(f) \cup \mathbf{dom}(g)$. All the more, $f = g$ implies $\mathbf{dom}(f) = \mathbf{dom}(g)$.

We write

$$\mathbf{im}(f) = \{e_2 \in E_2 \mid \exists e_1 \in E_1, (e_1, e_2) \in f\}$$

We say that f is injective if and only if for all $e_1 \neq e_1' \in \mathbf{dom}(f)$, $f(e_1) \neq f(e_1')$. We say that f is surjective if and only if $\mathbf{im}(f) = E_2$. We say that f is bijective if and only if f is injective and surjective.

A total function $f \in [E_1 \rightharpoonup E_2]$ is a partial function with $\mathbf{dom}(f) = E_1$.

We will be using partial functions between two alphabets in this thesis. So we need to extend them on words. We introduce below a canonical way to extend a partial function between alphabets into a *partial morphism of free monoids*.

Fix A and B two sets. The *free monoid* of A is the set of all the sequences of 0 or more elements of A, called *words* of A. It is written A^*. ϵ_A stands for the sequence of 0 elements of A. Given two words w_1 and w_2 in A^*, we write $w_1 \cdot w_2$, called the *concatenation of w_1 and w_2* to be the sequence w_1 immediately followed by the sequence w_2.

A *partial morphism* between A^* and B^* is a partial function ϕ such that:

- $\epsilon_A \in \mathbf{dom}(\phi)$
- for all $w, w' \in A^*$, $w \cdot w' \in \mathbf{dom}(\phi)$ if and only if $w \in \mathbf{dom}(\phi)$ and $w' \in \mathbf{dom}(\phi)$.
- for all $w, w' \in A^*$, $\phi(w \cdot w') = \phi(w) \cdot \phi(w')$

Suppose now $\psi : A \rightharpoonup B$ is a partial function from A to B. ψ is naturally extended on A^* in the following way : for all $w = a_1 \dots a_n \in \mathbf{dom}(\psi)$, $\psi(w) = \psi(a_1) \cdot \dots \cdot \psi(a_n)$. $\psi : A^* \rightharpoonup B^*$ is a partial morphism between A^* and B^*.

1.2 Labeled Transition System

The idea behind transition systems is simplistic and at the same time encompasses the whole variety and complexity of dynamic systems. Each of those systems is assumed to be reducible to a set of configurations and a relation describing the law of evolution between those configurations.

Definition 1.2.1. *A* transition system T*, is a pair* $\langle C, \to \rangle$ *where* C *is a set called* configuration space *and* $\to \in \mathcal{P}(C^2)$ *is a relation on* C *called* transition relation*.*

If the configuration space of a transition system is finite, then we call it a *finite transition system*, otherwise, it is called an *infinite transition system*.

In the modelization process, the configuration space and the transition relation are related to the internal behavior of the system. An outside observer may not grasp the whole effect of a transition on the configuration space. To model this loss of information we introduce *labels*.

Definition 1.2.2. *A* labeled *transition system,* T*, is a triple* $\langle C, \to, L \rangle$ *where* C *is a set called* configuration space*,* L *is a set called* label space *and* $\to \colon L \longrightarrow \mathcal{P}(C^2)$ *maps every label in* L *to a relation on* C *called* transition relation labeled by l*.*

We say that a labeled transition system is finite if and only if *both the configuration space and the label space* are finite. Otherwise, we call it *infinite*.

For all $l \in L$ and $c, c' \in C$, we write $\xrightarrow{l}$ instead of $\to(l)$ and $c \xrightarrow{l} c'$ instead of $(c, c') \in \xrightarrow{l}$.

We recall below the definition related to labeled transition systems. Fix a transition system $T = \langle C, \to, L \rangle$. A *path*, π, of T is a sequence $\pi = c_0 \cdot l_1 \cdot c_1 \dots c_{n-1} \cdot l_n \cdot c_n \in C \times (L \times C)^*$ such that for all $1 \leq i \leq n$, $c_{i-1} \xrightarrow{l_i} c_i$. For such a path we define:

- $\mathbf{len}(\pi) = n$ is the *length* of the path
- $\mathbf{start}(\pi) = c_0$ is the *starting configuration* of π
- $\mathbf{end}(\pi) = c_n$ is the *ending configuration* of π.
- $\mathbf{w}(\pi) = l_1 \dots l_n \in L^*$ is the *word recognized* by π.

Given two configurations c and c' in C,

- For all word $w \in L^*$, $\mathbf{Path}(T, w, c, c')$ stands for the set of all paths of T starting in c, ending in c' and recognizing the word w.

  $\mathbf{Path}(T, w, c) = \bigcup_{c'' \in C} \mathbf{Path}(T, w, c, c'')$.
- $\mathbf{Path}(T, c, c') = \bigcup_{w \in L^*} \mathbf{Path}(T, w, c, c')$ stands for the set of all paths of T starting in c and ending in c'.

  $\mathbf{Path}(T, c) = \bigcup_{c'' \in C} \mathbf{Path}(T, c, c'')$.
- $\mathcal{L}(T, c, c') = \{w \in L^* \mid \mathbf{Path}(T, w, c, c') \neq \emptyset\}$ stands for the set of all words recognized by a path of $\mathbf{Path}(T, c, c')$. It is called the language of T from c to c'.
- For all $c \in C$ and for all $F \subseteq C$, we also define $\mathcal{L}(T, c, F) = \bigcup_{c' \in F} \mathcal{L}(T, c, c')$. For short we write $\mathcal{L}(T, c) = \mathcal{L}(T, c, C)$.

If $\mathbf{Path}(T, c, c')$ is not empty we say that c' *is reachable from* c and write $c \to^* c'$. If for a word $w \in L^*$, $\mathbf{Path}(T, w, c, c')$ is not empty we'll write $c \xrightarrow{w}{}^* c'$. Given $c \in C$ and $w \in L^*$, we write $\mathbf{Reach}(T, c) = \{c' \in C \mid c \to^* c'\}$ and $\mathbf{Reach}(T, w, c) = \{c' \in C \mid c \xrightarrow{w}{}^* c'\}$.

$\mathbf{Path}$ and $\mathbf{Reach}$ concentrate all the information on the behavior of the transition system whereas $\mathcal{L}$ describes the information available to an external observer. Those are the objects we are usually interested in when studying transition systems, and this thesis will not be an exception.

Finally we recall below the formal definitions of determinism and completeness.

Definition 1.2.3. *Let $T = \langle C, \to, L\rangle$ be a transition system. We say that T is deterministic if and only if for all $l \in L$, $\xrightarrow{l}$ is a partial function from C to C, i.e. for all $c \in C$, there exists at most one $c' \in C$ such that $c \xrightarrow{l} c'$.*

In a deterministic transition system, we get by induction that for all configuration c and for all word w, there is *at most* one configuration in $\mathbf{Reach}(T, w, c)$.

Definition 1.2.4. *Let $T = \langle C, \to, L\rangle$ be a transition system. We say that T is complete if and only if for all $l \in L$, for all $c \in C$, there exists at least one $c' \in C$ such that $c \xrightarrow{l} c'$.*

In a complete transition system, we get by induction that for all configuration c and for all word w, there is *at least* one configuration in $\mathbf{Reach}(T, w, c)$.

Finally a transition system $T = \langle C, \to, L\rangle$ is both deterministic and complete if and only if for all $l \in L$, $\xrightarrow{l}$ is a function from C to C. In which case for all $w \in L^*$, we can define the function $\oplus_T : C \times L^* \to C$ such that $c \oplus_T w$ is the only configuration c' such that $c \xrightarrow{w}{}^* c'$. We omit to mention the transition system, and will write only $\oplus$, if it can be easily deduced from the context.

1.3 Bisimulation on Labeled Transition Systems

Two transition systems which somehow have a *similar* set of paths or a similar language will lead to the same theoretical results, allowing us to transfer any result from one to the other and conversely. It is frequent though that one transition system has some advantage upon the other, either because it is easier to study, or because it is easier to implement, etc. Depending on our goal some transition systems can, therefore, be better than others. A big part of our work will be on finding good representations of the same object to generalize results or get efficient implementations, hence the notion of *similarity* between systems is, for us, an important notion, which lies at the core of this thesis.

A classical notion of *similarity* between transition systems is the notion of *bisimulation equivalence* which informally states that two states c and c' are equivalent if and only if they can always mimic each other in all their possible evolution.

Definition 1.3.1. *Let $T_1 = \langle C_1, \to_1, L\rangle$ and $T_2 = \langle C_2, \to_2, L\rangle$ be two labeled transition systems sharing the same label space. A* simulation *is a relation $\sim \subseteq C_1 \times C_2$ such that for all $c_1 \sim c_2$, for all $l \in L$, and for all $c_1' \in C_1$,*

$$c_1 \xrightarrow{l}_1 c_1' \implies \exists c_2' \in C_2, c_2' \sim c_1' \text{ and } c_2 \xrightarrow{l}_2 c_2'$$

A bisimulation *is a simulation $\sim \subseteq C_1 \times C_2$ such that $\sim^{-1}$ is a simulation, i.e. for all $c_1 \sim c_2$, for all $l \in L$, and for all $c_2' \in C_2$,*

$$c_2 \xrightarrow{l}_2 c_2' \implies \exists c_1' \in C_1, c_1' \sim c_2' \text{ and } c_1 \xrightarrow{l}_1 c_1'$$

To better suit our future usage of this notion we restrict our selves to a *weakened version* of the definition which enforces the relation to be a function. Definition 1.3.1 is then adapted using different label spaces and stated with a functional form.

Definition 1.3.2. *Let $T_1 = \langle C_1, \to_1, L_1 \rangle$ and $T_2 = \langle C_2, \to_2, L_2 \rangle$ be two labeled transition systems. A* functional simulation *from T_1 to T_2 is a pair of functions (ϕ, ψ) with: ψ being a partial function from L_1 to L_2, and ϕ being a partial function from C_1 to C_2 such that for all $l_1 \in L_1$, for all $c_1 \in \mathbf{dom}(\phi)$, $c_1' \in C_1$*

$$c_1 \xrightarrow{l_1}_1 c_1' \implies c_1' \in \mathbf{dom}(\phi), l_1 \in \mathbf{dom}(\psi) \text{ and } \phi(c_1) \xrightarrow{\psi(l_1)}_2 \phi(c_1')$$

A functional bisimulation *from T_1 to T_2 is a functional simulation (ϕ, ψ) such that for all $c_1 \in \mathbf{dom}(\phi)$, $c_2' \in C_2$ and $l_2 \in L_2$*

$$\phi(c_1) \xrightarrow{l_2}_2 c_2' \implies \exists c_1' \in \phi^{-1}(c_2'), \exists l_1 \in \psi^{-1}(l_2), c_1 \xrightarrow{l_1}_1 c_1'$$

It is clear by this definition, that the existence of a functional bisimulation $(\phi, \mathbf{id})$ between two transition systems $T_1 = \langle C_1, \to_1, L \rangle$ and $T_2 = \langle C_2, \to_2, L \rangle$ implies the existence of a bisimulation relation (one can easily prove that the relation $c \sim c' \iff c' = \phi(c)$ over $C_1 \times C_2$ is a bisimulation relation). However, the reverse implication is not true. Hence this idea of *weakened version*.

We introduced the possibility of having two different label spaces in definition 1.3.2 which prevents direct comparison between definition 1.3.2 and 1.3.1.

Next proposition ensure that a functional bisimulation describes what we expected from a concept of similarity between transition systems, i.e. that the existence of a bisimulation between T_1 and T_2 ensures that every behavior of T_1 is mapped to a behavior of T_2, and, conversely, every behavior of T_2 is the mapping of a behavior of T_1.

Proposition 1.3.3. *Let two transition systems $T_1 = \langle C_1, \to_1, L_1 \rangle$ and $T_2 = \langle C_2, \to_2, L_2 \rangle$ and suppose their exists a functional simulation, (ϕ, ψ) between T_1 and T_2. Then :*

- *For all $c_1 \in \mathbf{dom}(\phi)$ and $w_1 \in L_1^*$,*

$$\phi(\mathbf{Reach}(T_1, w_1, c_1)) \subseteq \mathbf{Reach}(T_2, \psi^*(w_1), \phi(c_1))$$

- *For all $c_1 \in \mathbf{dom}(\phi)$ and $c_1' \in C_1$,*

$$\psi^*(\mathcal{L}(T_1, c_1, c_1')) \subseteq \mathcal{L}(T_2, \phi(c_1), \phi(c_1'))$$

If moreover (ϕ, ψ) is a functional bisimulation, then:

- *For all $c_1 \in \mathbf{dom}(\phi)$ and $w_2 \in L_2^*$,*

$$\bigcup_{w_1 \in \psi^{*-1}(w_2)} \phi(\mathbf{Reach}(T_1, w_1, c_1)) = \mathbf{Reach}(T_2, w_2, \phi(c_1))$$

- *For all $c_1 \in \mathbf{dom}(\phi)$ and $c_2 \in C_2$,*

$$\bigcup_{c_1' \in \phi^{-1}(c_2)} \psi^*(\mathcal{L}(T_1, c_1, c_1')) = \mathcal{L}(T_2, \phi(c_1), c_2)$$

Proof.

Notice that for all $c_1 \in \mathbf{dom}(\phi)$ and $w_1 \in L_1^*$, $\mathbf{Reach}(T_1, w_1, c_1)$ is included in $\mathbf{dom}(\phi)$ by a simple induction on the size of w_1. Also for all $c_1 \in \mathbf{dom}(\phi)$ and $c_1' \in C_1$, $\mathcal{L}(T_1, c_1, c_1')$ is included in $\mathbf{dom}(\psi^*)$ by another induction on the length of the path.

Then in all cases, the proof can be easily made using an induction on the size of the words. ■

Functional bisimulation is a tool to transfer objects of T_1 into T_2. Further in this thesis, we will seek to construct from T_1 a similar transition system T_2 or conversely, given only a transformation of the configuration and label space. For now, we describe only the case where a *renaming* of the labels is given. We can then easily construct a new transition system by relabeling the transitions.

Let $T = \langle C, \to, L \rangle$ and L' a set of labels. A *labeling* ψ, of T on L' is a (total) function from L to L'. The *L'-renaming of T by* ψ is the labeled transition system $T^\psi = (C, \to_{L'}, L')$, such that for all $l' \in L'$,

$$\xrightarrow{l'}_{L'} = \{(c, c') \in C^2 \mid \exists l \in \psi^{-1}(l') \text{ such that } c \xrightarrow{l} c'\}$$

Proposition 1.3.4. *Let $T = \langle C, \to, L \rangle$, L' a set of labels and ψ a* labeling *of T on L'.*
$\Phi = (\mathbf{Id}_C, \psi)$ is a functional bisimulation from T to T^ψ.

Proof.

The proof can be easily done relating definition 1.3.2 and the definition of T^ψ. ■

We base our model on labeled transition systems, using them to define the semantics of our objects as it is usually done. As we said in the introduction though, infinite transition systems are not a model easy to handle when it comes to representation and implementation. More than a particular subclass of labeled transition systems, the model of timed structures – and of automata on timed structures – that we introduce in the following chapters is a change of view, or rather of organization, in the modelization process; which we believe is better suited for representation and implementation matters.

Chapter 2

Timed Structures

Modeling Timed Variables

Transition systems are meant to model physical systems having several – possibly an infinite number of – configurations and evolving from one configuration to an other. It is not specified within the model whether this evolution is intrinsic or due to external action. In a state the information of a discrete state (e.g. button pressed/released, robot moving forward/backward or not moving . . .) and that of a quantitative variable (e.g. temperature, speed, timer value . . .) can be both stored.

Keeping the spirit of *hybrid automata* introduced by [35] we aim at adapting transition systems to split modeling of the quantitative variables – evolving consistently to their evolution laws – and the modeling of discrete states, which might change following an external action, possibly modifying the evolution rules of the quantitative variables. We choose though not to use hybrid automata nor timed transition systems [2] [36] because we need for our work a more expressively powerful model.

Our model, that we call *automata on timed structures*, is to be given its final definition in chapter 3. In the present chapter, we begin by giving a formal framework to model quantitative variables and their evolution *only*. We call it *timed structures*. The definition of automata on timed structures will, in the end, consist in building on top of a timed structure a framework which adds discrete states and steps.

Timed structures definition is itself split into three parts corresponding to the three sections of this chapter:

Section 2.1 : the description of the intrinsic laws driving the evolution of the quantitative variables using *timed domains*.

Section 2.2 : the description of the possible effect of an external action on the quantitative variables using *updates*.
A timed domain equipped with a set of updates will be called a *timed structure*.

Section 2.3 : the description of the requirements on the values of the quantitative variables for an external action to be doable.
We define *guards* and *guard basis*. A timed structure enhanced with a guard basis is then called a *guarded timed structure*.

As an example, suppose a timed structure is meant to model the evolution of the temperature inside a room. The timed domain describes the natural evolution of the temperature. Turning on the heat would be an update that changes the evolution law of the temperature. Quickly opening a window on a windy day could be viewed as an update which decreases instantaneously the temperature without modifying its evolution law. Finally, if the heat can

only be turned on if the temperature is below 25 degrees Celsius, this could be modeled using guards.

2.1 Timed Domains

Hybrid systems confine the evolution of quantitative variables into solutions of differential equations. In our approach, we seek to allow as many behaviors as possible for the quantitative variables and lose as few expressive power as possible with regards to transition systems.

Timed transition system [36] [2] already allow modeling a very large variety of timed systems. We recall the formal definition of timed systems below.

Definition 2.1.1. *Let Σ be a finite alphabet. A Σ-timed transition system is a labeled transition system $T = \langle C, \rightarrow, \mathbb{R}_{\geq 0} \uplus \Sigma \rangle$ such that for all $d, d' \in \mathbb{R}_{\geq 0}$:*

(1) for all $c \in C$, $c \xrightarrow{0} c$

(2) for all $c \in C$, exists $c' \in C$ such that $c \xrightarrow{d} c'$

(3) for all $c, c', c'' \in C$, $c \xrightarrow{d} c'$ and $c \xrightarrow{d} c''$ implies $c' = c''$

(4) for all $c, c' \in C$, exists $c'' \in C$ such that $c \xrightarrow{d} c''$ and $c'' \xrightarrow{d'} c'$ if and only if $c \xrightarrow{d+d'} c'$.

They can model evolution laws that differential equations cannot.

Example 2.1.1. Suppose the quantitative variable we want to model is both the data of a timer (or clock) x taking values in $\mathbb{R}_{\geq 0}$ and a color c within $\{$ blue, red $\}$. The evolution law described in figure 2.1 can't be described as a differential equation.

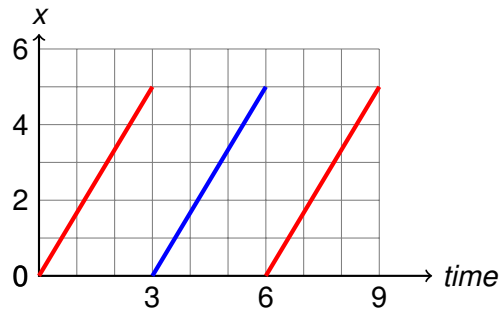


Figure 2.1: Evolution law of the variable (x, c)

However it can easily be described as the timed transition system $T = \langle \mathbb{R}_{\geq 0} \times \{$ blue, red $\}, \rightarrow, \mathbb{R}_{\geq 0} \rangle$ where, for all $d \in \mathbb{R}_{\geq 0}$, for all $(x, c) \in \mathbb{R}_{\geq 0} \times \{$ blue, red $\}$,

$$(x, c) \xrightarrow{d} (x', c') \text{ with } x' = (x + \frac{5 \times d}{3}) \mod 5 \text{ and } c' = c \text{ if } \lfloor \frac{x + \frac{5 \times d}{3}}{5} \rfloor \text{ is even, } \overline{c} \text{ otherwise}$$

Where $\overline{red} = blue$, $\overline{blue} = red$ and $\mod$ states for the classical *modulo* operator: $a \mod b = a - \lfloor \frac{a}{b} \rfloor b$.

T is a well-defined timed transition system. (1), (2) and (3) can easily be verified. Suppose $(x, c) \in \mathbb{R}_{\geq 0} \times \{$ blue, red $\}$ and $d, d' \in \mathbb{R}$.

$$\big([(x + \frac{5 \times d}{3}) \mod 5] + \frac{5 \times d'}{3}\big) \mod 5 = (x + \frac{5 \times (d + d')}{3}) \mod 5$$

which allows us to conclude that (4) is verified

We will not be using timed transition systems in this thesis but a slight variation we call *timed domains*. The easiest way to make this new model of quantitative variables as powerful as transition systems, ... is to use transition systems themselves.

The fact that timed transition systems are called *timed*, might make us say that an infinite transition system is by default *untimed*. Obviously, it doesn't mean that time does not intervene in transition systems. If so, how could one speak of evolution? It should mean that *absolute physical time value* is not used by the model, therefore that there is no *timer*. Time itself plays its role but it is not explicitly *quantified*. This is only in this sense that we shall call a transition systems *untimed*. *Timed domains* are meant, like timed transition systems, to enforce the quantification of time to take part in the model.

We will consider that a transition system is *timed* if it includes some way to *measure* time. We will restrict ourselves to *continuous time*, i.e. a measure of time expressed within the real numbers. How to include this measurement in the model? A natural approach is to consider a particular quantitative variable recording time. Let's say that a transition system $\langle V, \hookrightarrow \rangle$ models the evolution of a quantitative variable whose values are taken within the set V and such that $\hookrightarrow$ describes which value can evolve into which. We will construct a transition system of the form $\langle V \times \mathbb{R}, \hookrightarrow \rangle$. The value $(v, d) \in V \times \mathbb{R}$ in which every configuration represents both the information of the value of the quantitative variable and the point in time when the quantitative variable takes this value (absolute physical time or a timer launched at some starting point). In this setting if (v, d) evolves into (v', d') we have access to the quantity of time elapsed between those two values through the quantity $d' - d$.

However, adding a timer as we would add any other quantitative variable to our model is not what we really want. Indeed in this transition system, the quantitative variable and the timer might be interdependent. Moreover the evolution of the global system or the possibility for discrete actions might strongly depend not only on the value of the quantitative variable but also on the value of the timer. Considering the timer as any quantitative variable would mean in short that we care about its absolute value. For some models, we might indeed care. But for the notion of *timed domain* we aim to introduce, we wish to add only a way to measure time. This means that all that matters for the evolving transition is the *delay* between two related values and not the exact timestamps when they happen. Formally speaking we want that for two values v and v' in V, whatever the timestamps d and d' in $\mathbb{R}$ and the delay δ in $\mathbb{R}_{\geq 0}$, $(v, d) \hookrightarrow (v', d + \delta)$ if and only if $(v, d') \hookrightarrow (v', d' + \delta)$. Notice that with this constraint, the information of the value of the timer becomes unrelevent.

Additionally, we introduce below more constraints which implies some loss of expressive power with regards to transition systems. Still they remain in conformity with our goal to provide a framework which measures time. First we don't want time to "stop"at some point, i.e. we expect for all value v, and all time stamps $d_1 < d_2$ to have a v' such that $v, d_1 \hookrightarrow v', d_2$.

We also want time to be *directed*. Evolution have to be made continuously, in one direction. It means that for all $v \in V$ and $d, d' \in \mathbb{R}_{\geq 0}$, there exists $v' \in V$ such that $(v, d) \hookrightarrow (v', d')$ if and only if $d < d'$. The direction of time is what allows us to situate configurations using a notion of *posteriority* and *anteriority*. In formal terms it correspond to a transitivity property on the evolving transition : for all three time stamps $d_1 < d_2 < d_3$ and three values v_1, v_2, v_3, if $(v_1, d_1) \hookrightarrow (v_2, d_2) \hookrightarrow (v_3, d_3)$ then $(v_1, d_1) \hookrightarrow (v_3, d_3)$.

In the end, what we'll call a *timed domain* is a transition system enhanced with a timer whose exact value is loss and respecting the above constraints. Formally :

Definition 2.1.2. *A* timed domain *is a couple* $\langle V, \hookrightarrow \rangle$ *where* V *is a set called* value space *and* $\hookrightarrow$ *is a function from* $\mathbb{R}_{\geq 0}$ *to* $\mathcal{P}(V \times V)$ *called* delay transition*, such that*

- *for all* $v \in V$*,* $(v, v) \in \hookrightarrow(0)$
- *for all* $v \in V$ *and* $d \in \mathbb{R}_{\geq 0}$*, there exists* $v' \in V$ *such that* $(v, v') \in \hookrightarrow(d)$
- *for all* $d, d' \in \mathbb{R}_{\geq 0}$*, for all* $v, v', v'' \in V$*,*

  if $(v, v') \in \hookrightarrow(d)$ *and* $(v', v'') \in \hookrightarrow(d')$ *then* $(v, v'') \in \hookrightarrow(d + d')$

We write for all $d \in \mathbb{R}_{\geq 0}$*,* $\overset{d}{\hookrightarrow} = \hookrightarrow(d)$ *and* $v \overset{d}{\hookrightarrow} v'$ *as a shorthand for* $(v, v') \in \overset{d}{\hookrightarrow}$*.*

Remark 2.1.1. A timed domain is not a timed transition system stripped of all action transitions, because we didn't enforce time determinism (condition (3) in definition 2.1.1).

Remark 2.1.2. If $\langle V, \hookrightarrow \rangle$ is a timed domain, then $\langle V, \hookrightarrow, \mathbb{R}_{\geq 0} \rangle$ is a labeled transition system. It is complete by definition. We can export on timed domains all the notions of labeled transition systems, like paths, reachability,...

In particular, we can export determinism and speak of *deterministic timed domain*. If the timed domain is deterministic, for all $d \in \mathbb{R}_{\geq 0}$, we know that $\overset{d}{\hookrightarrow}$ is actually a *function* from V to V. In that case we'll write for all $v \in V$ and $d \in \mathbb{R}_{\geq 0}$, $v \oplus_{\langle V, \hookrightarrow \rangle} d = \overset{d}{\hookrightarrow}(v)$ or just $v \oplus d$ when the timed domain is clear from the context.

We redefine the notion of functional bisimulation on timed domains.

Definition 2.1.3. *Let* $D_1 = \langle V_1, \hookrightarrow_1 \rangle$ *and* $D_2 = \langle V_2, \hookrightarrow_2 \rangle$ *be two timed domains. A* functional bisimulation *from* D_1 *to* D_2 *is a partial function* $\phi : V_1 \rightharpoonup V_2$ *such that* $(\phi, \mathbf{id}_{\mathbb{R}_{\geq 0}})$ *is a functional bisimulation between labeled transition system from* $\langle V_1, \hookrightarrow_1, \mathbb{R}_{\geq 0} \rangle$ *to* $\langle V_2, \hookrightarrow_2, \mathbb{R}_{\geq 0} \rangle$*.*

We take some more time to introduce particular timed domain relevant in this thesis.

1. Clock Domain. This is the timed domain modeling a timer. This timer is classically called a *clock*, hence the name. Those are the clocks used in timed automata [4]. The clock can have an arbitrary real value between 0 and a maximal value $M \in \mathbb{N}$. Strictly over M, the timer is stuck on value ∞ (arbitrary value larger than M).

Formally, let $M \in \mathbb{N}$, we write $\mathbb{C}_M = [0, M] \cup \{\infty\}$. We define the delay relation as follows, for all $x \in \mathbb{C}_M$ and $d \in \mathbb{R}_{\geq 0}$:

- if $x \in [0, M]$
  - if $x + d \leq M$, $x \overset{d}{\hookrightarrow} x + d$
  - else $x \overset{d}{\hookrightarrow} \infty$
- $\infty \overset{d}{\hookrightarrow} \infty$

$(\mathbb{C}_M, \hookrightarrow)$ is a well-defined *deterministic* timed domain called the M-clock domain.

2. Perturbed-Clock Domain. This is the timed domain modeling a clock evolving at an imprecise rate. Those clocks are defined to mimic the ones used in perturbed automata [6]. Like in the clock domain, the clock can have an arbitrary real value between 0 and a maximal value $M \in \mathbb{N}$. Strictly over M, the timer is stuck on value ∞.

However, in the perturbed clock domain, clocks evolve with a non-deterministically chosen rate ranging between $1-\epsilon$ and $1+\epsilon$ for some fixed ϵ.

Formally, let $M \in \mathbb{N}$ and $\epsilon \in [0,1[$, we write $\mathbb{C}_M = [0,M] \cup \{\infty\}$. We define the delay relation as follows, for all $x \in \mathbb{C}_M$ and $d \in \mathbb{R}_{\geq 0}$:

- if $x \in [0,M]$, for all $x' \in [x+d(1-\epsilon), x+d(1+\epsilon)]$,
  - if $x' \leq M$, $x \overset{d}{\hookrightarrow}_\epsilon x'$
  - else $x \overset{d}{\hookrightarrow}_\epsilon \infty$
- $\infty \overset{d}{\hookrightarrow}_\epsilon \infty$

$(\mathbb{C}_M, \hookrightarrow_\epsilon)$ is a well-defined *non-deterministic* timed domain called the (M,ϵ)-perturbed clock domain.

3. Predicting-Clock Domain. This is a timed domain to model *countdown timers*. The timer, or clock, is considered as initially set to some real value and decreases until it reaches 0, at which point it stops. We will use those clocks to translate event-predicting timed automata [5] into our formalism.

Formally, we write $\mathbf{P}\mathbb{R}_{\geq 0} = \mathbb{R}_{\geq 0} \uplus \{\bot\}$. We define the delay relation, $\hookrightarrow_{\mathbf{P}}$ as follows, for all $x \in \mathbb{R}_{\geq 0}$ and $d \in \mathbb{R}_{\geq 0}$:

- if $x - d \geq 0$, $x \overset{d}{\hookrightarrow}_{\mathbf{P}} x-d$
- else $x \overset{d}{\hookrightarrow}_{\mathbf{P}} \bot$
- $\bot \overset{d}{\hookrightarrow}_{\mathbf{P}} \bot$

$(\mathbf{P}\mathbb{R}_{\geq 0}, \hookrightarrow_{\mathbf{P}})$ is a well-defined *deterministic* timed domain.

4. Timed-Stack Domain. This is the timed domain modeling a *timed stack*, i.e. a stack where every stack symbol is equipped with a clock evolving with time. The timed stack is inspired by [1] and [16]. Clocks are exactly the ones described in the clock domain.

Formally, let $M \in \mathbb{N}$ and Z be a finite *stack alphabet*, we write $\mathbb{C}_M = [0,M] \cup \{\infty\}$ and $\mathcal{Z}_{M,Z} = (Z \times \mathbb{C}_M)^*$. We define the delay relation as follows, for all $(z_0,x_0)\dots(z_n,x_n) \in \mathcal{Z}_{M,Z}$ and $d \in \mathbb{R}_{\geq 0}$:

$$(z_0,x_0)\dots(z_n,x_n) \overset{d}{\hookrightarrow} (z_0, x_0 \oplus_{\langle \mathbb{C}_M, \hookrightarrow \rangle} d)\dots(z_n, x_n \oplus_{\langle \mathbb{C}_M, \hookrightarrow \rangle} d)$$

where $\langle \mathbb{C}_M, \hookrightarrow \rangle$ stands for the clock domain defined above.

$(\mathcal{Z}_{M,Z}, \hookrightarrow)$ is a well-defined *deterministic* timed domain called the (M,Z)-timed stack domain.

2.2 Updates and Timed Structures

Modeling the autonomous behavior of a quantitative variable is now possible, but life (or at least verification) would be boring if we would not allow, and model external interventions. Those interventions could have artificial or natural causes, what matters is that we ideally want them to be punctual. They may have either a direct impact on the value of the quantitative variables modeled by the timed domain, an impact on the laws of evolution of those variable or both.

Formally external interventions are modeled through *updates*. Updates are functions defined on the value space, modeling the possible effects of external interventions on the values of the quantitative variables.

Definition 2.2.1. *Let $D = \langle V, \hookrightarrow \rangle$ be a timed domain. An* update set *on D is a subset of V^V.*

In this thesis, we consider that the basis of any modelization of quantitative variables is the data of a timed domain equipped with an update set. We call this combination a *timed structure*.

Definition 2.2.2. *A* timed structure *is a triple $T = \langle V, \hookrightarrow, U \rangle$ where $\langle V, \hookrightarrow \rangle$ is a timed domain and U is an update set on $\langle V, \hookrightarrow \rangle$.*

We say that a structure $\mathcal{S} = \langle V, \hookrightarrow, U \rangle$ is *deterministic* if and only if $\langle V, \hookrightarrow \rangle$ is. In that case we write for all $v \in V$ and $d \in \mathbb{R}_{\geq 0}$, $v \oplus_{\mathcal{S}} d = v \oplus_{\langle V, \hookrightarrow \rangle} d$ or just $v \oplus d$ if the timed structure is clear from the context.

We describe below some important structures for this thesis.

1. Clock Structure. This is a timed structure modeling a timer with a reset button. It is based on the clock domain $(\mathbb{C}_M, \hookrightarrow)$ equipped with two updates, one does nothing and the other resets the timer. Those updates correspond to the reset operation used in timed automata [4].

Formally, let $M \in \mathbb{N}$. Let $\mathbf{id}$ and $\mathbf{0}$ defined for all $x \in \mathbb{C}_M$ as $\mathbf{id}(x) = x$ and $\mathbf{0}(x) = 0$. $(\mathbb{C}_M, \hookrightarrow, \{\mathbf{id}, \mathbf{0}\})$ is a well-defined timed structure called the M-clock structure.

2. Perturbed-Clock Structure. This is an adaptation of the clock structure, to consider perturbed clocks.

Formally, let $M \in \mathbb{N}$ and $\epsilon \in [0, 1[$. Let $\mathbf{id}$ and $\mathbf{0}$ defined as above. $(\mathbb{C}_M, \hookrightarrow_\epsilon, \{\mathbf{id}, \mathbf{0}\})$ is a well-defined timed structure called the (M, ϵ)-perturbed clock structure.

3. Predicting-Clock Structure. This is a timed structure to model *countdown timers* which can be initialized through an update to any real value. It is based on the deterministic timed domain $(\mathbf{P}\mathbb{R}_{\geq 0}, \hookrightarrow_{\mathbf{P}})$.

We write for all $d \in \mathbb{R}_{\geq 0}$, $\mathbf{d} : \mathbf{P}\mathbb{R}_{\geq 0} \to \mathbf{P}\mathbb{R}_{\geq 0}$ the constant function mapping every elements to d. And $\perp : \mathbf{P}\mathbb{R}_{\geq 0} \to \mathbf{P}\mathbb{R}_{\geq 0}$ the constant function mapping every elements to $\perp$ Let $\mathbf{U}\mathbb{R}_{\geq 0} = \{\mathbf{d}, d \in \mathbb{R}_{\geq 0}\} \cup \{\perp\}$.

$\mathbf{P}C = (\mathbf{P}\mathbb{R}_{\geq 0}, \hookrightarrow_{\mathbf{P}}, \mathbf{U}\mathbb{R}_{\geq 0} \cup \{\mathbf{id}\})$ is a well-defined timed structure called the predicting-clock structure.

4. Timed-Stack Structure. This is a timed structure to model **push** and **pop** operations on the stack [1].

Formally, let $M \in \mathbb{N}$ and Z be a stack alphabet. Let for all $z \in Z$, $\mathbf{push}_z : \mathcal{Z}_{M,Z} \to \mathcal{Z}_{M,Z}$ mapping every stack $\overline{z} \in \mathcal{Z}_{M,Z}$ to $\mathbf{push}_z(\overline{z}) = \overline{z} \cdot (z, 0)$. Let also $\mathbf{pop} : \mathcal{Z}_{M,Z} \to \mathcal{Z}_{M,Z}$ mapping ϵ to ϵ and every stack $\overline{z} \cdot (z, x) \in \mathcal{Z}_{M,Z}$ to $\mathbf{pop}(\overline{z} \cdot (z, x)) = \overline{z}$. Let $U_{M,Z} = \{\mathbf{pop}, \mathbf{id}\} \cup \{\mathbf{push}_z, z \in Z\}$.

$\mathbf{Z}_{M,Z} = (\mathcal{Z}_{M,Z}, \hookrightarrow, U_{M,Z})$ is a well-defined timed structure called the (M, Z)-timed-stack structure.

As we already said, timed structures are at the core of our way of modeling timed systems. We didn't impose any restriction on the evolution laws of the quantitative variables we model and neither did we on the updates. This was our point of divergence with hybrid systems.

In consequence, not all timed structures can be finitely represented or effectively used. Of course for a timed structure to be effectively used we must require that all updates and all delays are computable. In this case, we say that the timed structure is *computable*, although those properties do not ensure finite representation. This is what motivates the introduction of a general notion of *guards* in the next section.

2.3 About Guards and Guarded Timed Structures

In most existing timed models, finite representation and computability of quantitative variables with infinite value space are made possible by enforcing some regularity on the behavior of the model within a finite and computable partition of the value space. We formalize below this kind of partitioning within timed structures. The object at the core of this formalization is *guards*, introduced below.

Definition 2.3.1. *Given a timed structure $\mathcal{S} = (V, \hookrightarrow, U)$ we call* guard *on $\mathcal{S}$ any subset of $V \times U$. We write $\mathcal{G}(\mathcal{S})$ for the set of guards on $\mathcal{S}$.*

We will often need to associate a guard with each pair of elements of a given set. Most of the time this set will correspond to a pair of sets, and the guard associated with a pair of states describes the prerequisites on the values to change state and which updates have then to be applied. We introduce then the convenient notion of guard function defined below.

Definition 2.3.2. *Let $\mathcal{S} = \langle S, \hookrightarrow, U\rangle$ be a timed structure and E be a finite set. A* guard function *on $\mathcal{S}$ and E is a function from $E \times E$ to $\mathcal{G}(\mathcal{S})$.*

Guards are the tool we use to partition our value space. We introduce now *guard bases*, the tool we use to enforce finite representation and computability.

Let $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure. Let G be a set of elements of $\mathcal{P}(V) \times \mathcal{P}(U)$ and g be a guard on $\mathcal{S}$.

We say that g is *(finitely) decomposable in* G if,

$$\exists n \in \mathbb{N}, \exists (\nu_i, \mu_i)_{1 \leq i \leq n} \in G^n,\ g = \bigcup_{i=1}^{n} \nu_i \times \mu_i$$

Since we will never use a decomposition other than finite we will omit to precise it. Notice that the empty guard is always decomposable on a set G.

If $\mathbf{g}$ is a guard function on $\mathcal{S}$ and some finite set E, we say that $\mathbf{g}$ is decomposable on G if for all $e, e' \in E$, $\mathbf{g}(e, e')$ is decomposable on G.

Definition 2.3.3. *We call* guard basis *on $\mathcal{S}$ any set G of elements of $\mathcal{P}(V) \times \mathcal{P}(U)$ such that for all $(\nu, \mu), (\nu', \mu') \in G$, $(\nu \times \mu) \cap (\nu' \times \mu') = \emptyset$.*

We say that G is spanning *if and only if for all $(v, u) \in V \times U$, exists $(\nu, \mu) \in G$ such that $(v, u) \in (\nu, \mu)$.*

If G is a guard basis and if a guard g can be decomposed on G, it has a unique decomposition. We write then $g = \{(\nu_1, \mu_1), \ldots, (\nu_n, \mu_n)\}$ meaning $g = \bigcup_{i=1}^{n} \nu_i \times \mu_i$ and $(\nu, \mu) \in g$ meaning $\exists 1 \leq i \leq n, \nu = \nu_i$ and $\mu = \mu_i$.

Example 2.3.1. In the timed automata model of [4], the term *guard* is used to describe a conjunction of inequalities that describes the prerequisite on the clocks to be able to change state. The reset or not of a clock is a piece of information given aside. For example consider that between two states of the timed automaton we can find two distinct transitions equipped respectively by the information $2 < x \leq 4, x := 0$ and $x = 1$. The guard we would define in our formalism to describe the same possibilities would be the set $g = \{(x, \mathbf{0}) \mid x \in (2, 4]\} \cup \{(1, \mathbf{id})\}$.

Moreover, in timed automata those inequalities suffer a restriction: they may involve only integers between 0 and a fixed constant, M. Guard bases are exactly the tool we introduced to be able to enforce such a restriction, in a larger variety of models than timed automata. To enforce the same kind of restriction on the clock-structure than on timed automata guards we can introduce for any integer M the M-clock guard basis.

Let $\mathcal{I}_M = \{[n, n] \mid n \in [\![1, M]\!]\} \cup \{(n, n+1) \mid n \in [\![1, M-1]\!]\} \cup \{(M, +\infty)\}$ be the set of all real open intervals bounded by two consecutive integers smaller than or equal to M and all integer singletons smaller or equal to M. We define the M-clock guard basis on the M-clock structure as

$$\mathbb{G}_M = (\mathcal{I}_M \cup \{\{\infty\}\}) \times \{\{\mathbf{id}\}, \{\mathbf{0}\}\}$$

Any combination of guards and resets used in a timed automaton of [4], can be represented by a guard of our formalism decomposable in $\mathbb{G}_M$. For example the guard g defined above can be decomposed as

$$\{\langle [1, 1], \mathbf{id}\rangle, \langle (2, 3), \{\mathbf{0}\}\rangle, \langle [3, 3], \{\mathbf{0}\}\rangle, \langle (3, 4), \{\mathbf{0}\}\rangle, \langle [4, 4], \{\mathbf{0}\}\rangle\}$$

Remark 2.3.1. It is frequent, in particular in clock structures that a guard is decomposed into several elements of the guard basis with different value part (element of $\mathcal{P}(V)$) and the same update part (element of $\mathcal{P}(U)$). In this case we use the following shorthand : instead of $\{(\nu_1, \mu_1), (\nu_2, \mu_1), (\nu_3, \mu_1), (\nu_4, \mu_2), (\nu_5, \mu_2)\}$ we write $\{(\nu_1, \nu_2, \nu_3, \mu_1), (\nu_4, \nu_5, \mu_2)\}$. For example we could rewrite the decomposition in example 2.3.1 as :

$$\{\langle [1, 1], \mathbf{id}\rangle, \langle (2, 3), [3, 3], (3, 4), [4, 4], \{\mathbf{0}\}\rangle\}$$

Example 2.3.2. Let Z be a stack alphabet, M be an integer and $\mathcal{I}_M$ be the set of all real intervals bounded by two consecutive integers smaller or equal to M. We define the (M, Z)-timed-stack guard basis on the (M, Z)-timed-stack structure as

$$\mathbb{G}_{M,Z} = \{(Z \times \mathbb{C}_M)^*\} \times (\{\mathbf{push}_z, z \in Z\} \cup \{\mathbf{id}\}) \cup \{(Z \times \mathbb{C}_M)^* \cdot (\{z\} \times I), z \in Z, I \in \mathcal{I}_M\} \times \{\mathbf{pop}\}$$

This guard basis enforces that a $\mathbf{push}$ or $\mathbf{id}$ operation can be done whatever the stack value and that a $\mathbf{pop}$ operation can have a prerequisite on the value associated with the last stack symbol, but that this prerequisite is restricted in the same way inequalities on timed automata are : involving only integer bounds.

Notice that in our formalism the update $\mathbf{pop}$ does not enforce itself any symbol on the top of the stack to be applied. It is the definition of the guard basis which allows a model respecting it to enforce a particular symbol on the top of the stack to allow a transition with a $\mathbf{pop}$ update.

When we equip a timed structure with a guard basis we try to enforce some restrictions on how transitions will be made. It will become clear how in chapter 3.

Definition 2.3.4. *Let $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure and let G be a guard basis on $\mathcal{S}$. The tuple $\mathcal{S}_G = \langle V, \hookrightarrow, U, G\rangle$, is named* guarded timed structure*.*

If $\mathcal{S}$ is a computable timed structure and G is a guard basis such that all its elements are computable and finitely representable we will say that $\mathcal{S}_G$ is computable.

Example 2.3.3. Let M be an integer. The M-clock structure guarded by the M-clock guard basis, $C_M = \langle \mathbb{C}_M, \hookrightarrow, \{\mathbf{id}, \mathbf{0}\}, \mathbb{G}_M\rangle$, is called the M-clock guarded structure and the structure we will use to define *one-clock timed automata* [4] in our framework.

Before introducing the discrete part of our modelization in the chapter 3, we present below three ways of constructing new guarded timed structures from existing ones.

1. Product. Let $\mathcal{S}_1 = \langle V_1, \hookrightarrow_1, U_1\rangle$ and $\mathcal{S}_2 = \langle V_2, \hookrightarrow_2, U_2\rangle$ be two timed structures modeling two quantitative variables. The product intuitively aims to model the evolution and updates of both quantitative variables considered as evolving in parallel and independently.

Formally we define $\mathcal{S}_1 \times \mathcal{S}_2 = \langle V_1 \times V_2, \hookrightarrow_\times, U_\times\rangle$ with:

- for all $d \in \mathbb{R}_{\geq 0}$, $\overset{d}{\hookrightarrow} = \{((v_1, v_2), (v_1', v_2')) \mid v_1 \overset{d}{\hookrightarrow}_1 v_1' \text{ and } v_2 \overset{d}{\hookrightarrow}_2 v_2'\}$
- for all $f, g \in U_1, U_2$ and $v_1, v_2 \in V_1, V_2$ we define a function $(f, g)(v_1, v_2) = (f(v_1), g(v_2))$ and write $U_\times = \{(f, g), f \in U_1, g \in U_2\}$

If G_1 and G_2 are guard basis on $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively, we write

$$G_\times = \{(\nu_1 \times \nu_2, \mu_1 \times \mu_2), (\nu_1, \mu_1) \in G_1, (\nu_2, \mu_2) \in G_2\}$$

$G_\times$ is a well-defined guard basis on $\mathcal{S}_1 \times \mathcal{S}_2$ since G_1 and G_2 are.

Given $n \in \mathbb{N}_{\geq 0}$ and a timed structure $\mathcal{S}$, we write $\mathcal{S}^n$ for the product of $\mathcal{S}$ with itself n times. If G is a guard basis on $\mathcal{S}$, we write $G^n = \{(\nu_1 \times \cdots \times \nu_n, \mu_1 \times \cdots \times \mu_n), (\nu_1, \mu_1), \ldots, (\nu_n, \mu_n) \in \mathbf{G}\}$. G^n is a guard basis on $\mathcal{S}^n$. We write then $\mathcal{S}_G^n$ for $\mathcal{S}_{G^n}^n$.

Notice finally that for all $n, m \in \mathbb{N}$, $(\mathcal{S}^n)^m = \mathcal{S}^{nm}$ and $(\mathcal{S}_G^n)^m = \mathcal{S}_G^{nm}$

Example 2.3.4. Let M, n be two integers. $\mathcal{C}_M^n$ is the (M, n)-clock guarded structure and is the structure used to define n*-clocks timed automata* [4] in our framework.

Example 2.3.5. Let M, n be two integers and Z be a stack alphabet. We write $\mathcal{Z}_{M,n,Z}$ for the product of the two guarded timed structures $\mathcal{C}_M^n$ and $(\mathcal{Z}_{M,Z})_{\mathbb{G}_{M,Z}}$. This guarded timed structure is the structure used to define *pushdown timed automata* in our framework, according to prior definition of the model found in [1] and [16].

2. E-maps. Let $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure and E be a finite set. The quantitative variable modeled by $\mathcal{S}$ is considered here as a *resource* which can be allocated, deleted, copied and stored in different spaces labeled by E. All resources evolve synchronously following the same evolution law, and updates are now a composition of updates defined in $\mathcal{S}$ and *resource operation* like the ones enumerated above.

Formally we define $[E \rightharpoonup \mathcal{S}] = \langle [E \rightharpoonup V], \hookrightarrow_E, U_E \rangle$ with :

- for all $d \in \mathbb{R}_{\geq 0}$, $\overset{d}{\hookrightarrow}_E = \{(\mathbf{v}, \mathbf{v}') \mid \mathbf{dom}(\mathbf{v}) = \mathbf{dom}(\mathbf{v}')$ and $\forall e \in \mathbf{dom}(v), \mathbf{v}(e) \overset{d}{\hookrightarrow} \mathbf{v}'(e)\}$
- For all $f \in [E \rightharpoonup (U \times E)]$ we define a function, $\mathbf{u}_f$, such that for all $\mathbf{v} \in [E \rightharpoonup V]$, $\mathbf{u}_f(\mathbf{v})$ is a partial function from E to V defined for all $e' \in \mathbf{dom}(f)$ as $\mathbf{u}_f(\mathbf{v})(e') = u(\mathbf{v}(e))$, if $f(e') = (u, e)$ and $e \in \mathbf{dom}(\mathbf{v})$.

  We write then $U_E = \{\mathbf{u}_f, f \in [E \rightharpoonup U \times E]\}$.

Let G be a guard basis on $\mathcal{S}$. Let $\mathbf{g} : E \rightharpoonup \mathcal{P}(V) \times \mathcal{P}(U)$ such that for all $e \in \mathbf{dom}(\mathbf{g})$, $\mathbf{g}(e) = (\nu_e, \mu_e^1 \cup \cdots \cup \mu_e^n)$ with for all $1 \leq i \leq n$ $(\nu_e, \mu_e^i) \in G$. Let $f \in [E \rightharpoonup U \times E]$ such that for all $e \in \mathbf{dom}(\mathbf{g})$, $\mu_e^1 \cup \cdots \cup \mu_e^n = \{u \in U \mid \exists e' \in \mathbf{dom}(f), f(e') = (u, e)\})$. Then we say that $\mathbf{u}_f, \mathbf{g}$ are compatible with G and we write,

$$G_{\mathbf{g},\mathbf{u}_f} = (\{\mathbf{v} \in \mathcal{P}([E \rightharpoonup V]), \mathbf{dom}(\mathbf{v}) = \mathbf{dom}(\mathbf{g}) \text{ and } \forall e \in \mathbf{dom}(\mathbf{v}), \mathbf{v}(e) \in \nu_e\}, \{\mathbf{u}_f\})$$

G_E is the set of all $G_{\mathbf{g},\mathbf{u}}$ with $\mathbf{g}, \mathbf{u}$ compatible with G. This is a guard basis on $[E \rightharpoonup \mathcal{S}]$ since G is a guard basis on $\mathcal{S}$. We write then $[E \rightharpoonup \mathcal{S}_G]$ for $[E \rightharpoonup \mathcal{S}]_{G_E}$.

Example 2.3.6. Let M be an integer and E be a set. $[E \rightharpoonup \mathcal{C}_M]$ is the (M, E)-enhanced clock guarded structure.

Concretely suppose $E = \{x, y, z\}$. The delay transition does what one would expect of it and increases synchronously x,y and z with time. Let's have a closer look at updates. f defines for all clock, which update to do (or if the clock will be deactivated) and on which value, among all clocks values, to do it. Suppose f maps x to $(\mathbf{0}, z)$ and y to $(\mathbf{id}, z)$ (z is not part of the domain of f). Then the update $\mathbf{u}_f$ applied on a valuation $\mathbf{v}$, defines $\mathbf{v}'$ such that x new value is 0 (the old value of z on which we apply $\mathbf{0}$), y new value is $\mathbf{v}(z)$ the old value of z and z is now inactive. This could have been symbolically summarized as $x \mapsto 0, y \mapsto z, z \mapsto \perp$.

Guards despite their complex formal definition still can be easily written in the form $z \in (1, 2), x \mapsto 0, y \mapsto z, z \mapsto \perp$.

Remark 2.3.2. In general in an *E-map* construction an update u_f can be symbolically represented as a conjunction of operations :

- $x \mapsto u(y)$ where x and y are elements of E, u is an update of the original timed structure and $f(x) = (u, y)$
- $x \mapsto \perp$, where x is an element of E which is not in the domain of f

Remark 2.3.3. Let $M \in \mathbb{N}$, $n \in \mathbb{N}_{>0}$ and E be a finite set. $[E \rightharpoonup \mathcal{C}_M^n]$ define the same guarded timed structure as $[E \times [\![1, n]\!] \rightharpoonup \mathcal{C}_M]$.

Indeed a partial function $\mathbf{x}$ from E to $\mathbb{C}_M^n$ can easily be viewed as a partial function $\mathbf{x}'$ from $E \times [\![1, n]\!]$ to $\mathbb{C}_M$ by mapping every $(e, i) \in E \times [\![1, n]\!]$ to the i^{th} element of $\mathbf{x}(e)$.

An update $\mathbf{u}_f \in U_{\times,E}$ can also be viewed as an update $\mathbf{u}'_{f'} \in U_{E \times [\![1,n]\!]}$, by defining $f'(e', i) = (u_i, (e, i))$ if $f(e') = ((u_1, \ldots, u_n), e)$.

Finally a guard $G_{\mathbf{g},\mathbf{u}_f} \in [E \rightharpoonup \mathbb{G}_{M,\times}]$ can be viewed as a guard $G'_{\mathbf{g}',\mathbf{u}'_{f'}} \in [E \times [\![1, n]\!] \rightharpoonup \mathbb{G}_M]$ with $\mathbf{g}'$ compatible with f' and $\mathbf{g}'(e, i) = (\nu_i, \mu)$ if $\mathbf{g}(e) = (\nu_1 \times \cdots \times \nu_n, \mu_1 \times \cdots \times \mu_n)$ and $\mu = \{u \in U \mid \exists (e', j) \in \mathbf{dom}(f'), f'(e', j) = (u, (e, i))\}$. $\mathbf{g}'$ verifies the hypothesis necessary so that $G'_{\mathbf{g}',\mathbf{u}'_{f'}}$ is a guard, since $\mathbf{g}$ does and because in $\mathbb{G}_M$, whatever $u \in \{\mathbf{id}, \mathbf{0}\}$, $(\nu, \{u\}) \in \mathbb{G}$.

3. E-markings. Let $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure and E be a finite set. There is less intuition behind the following construction, but roughly speaking, one could view here again the quantitative variable as a resource, with all possible operations already described for the E-map construction, but, this time, there's no limit to the number of resources which can be allocated. They are no longer stored in spaces labeled by E, but grouped into classes labeled by E. This type of structure appears naturally in the determinization process, which is the reason why we bother to introduce it here.

Formally we define $\mathbf{M}_E S = \langle \mathbf{M}_E V, \hookrightarrow_M, \mathbf{M}_E U\rangle$ with :

- $\mathbf{M}_E V = \mathcal{P}(V)^E$ is called the set of markings on E. We define a function $\mathbf{supp} : \mathbf{M}_E V \to \mathcal{P}(E)$ as, for all $\boldsymbol{\nu} \in \mathbf{M}_E V$, $\mathbf{supp}(\boldsymbol{\nu}) = \{e \in E \mid \boldsymbol{\nu}(e) \neq \emptyset\}$.
- for all $d \in \mathbb{R}_{\geq 0}$, $\overset{d}{\hookrightarrow}_M = \{(\boldsymbol{\nu}, \boldsymbol{\nu}') \in \mathbf{M}_E V \mid \forall e \in E, \boldsymbol{\nu}'(e) = \{v' \in V \mid \exists v \in \boldsymbol{\nu}(e), v \overset{d}{\hookrightarrow} v'\}\}$
- We define for all guard function $\mathbf{g}$, on $\mathcal{S}$ and E the function $\mathbf{U}_{\mathbf{g}} : \mathbf{M}_E V \to \mathbf{M}_E V$ such that:

$$\mathbf{U}_{\mathbf{g}}(\boldsymbol{\nu})(e') = \{u(v) \in V \mid \exists e \in E, \exists v \in \boldsymbol{\nu}(e), (v, u) \in \mathbf{g}(e, e')\}$$

  We write $\mathbf{M}_E U = \{\mathbf{U}_{\mathbf{g}}, \mathbf{g} \in \mathcal{G}(\mathcal{S})^{E\times E}\}$.

Suppose G is a guard basis on $\mathcal{S}$. let $\mathbf{g}$ be a guard function decomposable on G, let $\rho, \rho' \in \mathcal{P}(E)$. We define

$$G_{\rho,\mathbf{g},\rho'} = \{\boldsymbol{\nu} \in \mathbf{M}_E V \mid \mathbf{supp}(\boldsymbol{\nu}) = \rho \text{ and } \mathbf{supp}(\mathbf{U}_{\mathbf{g}}(\boldsymbol{\nu})) = \rho'\}$$

We write $\mathbf{M}_E G = \{(G_{\rho,\mathbf{g},\rho'}, \{\mathbf{U}_{\mathbf{g}}\}), \mathbf{g} \text{ a guard function decomposable on } G\}$. $\mathbf{M}_E G$ is a guard basis on $\mathbf{M}_E S$, since $\mathbf{supp}$ is a function.

We write then $\mathbf{M}_E S_G = \langle \mathbf{M}_E V, \hookrightarrow_M, \mathbf{M}_E U, \mathbf{M}_E G\rangle$.

All constructions preserve determinism of the timed structure. We claim also that product and map preserve computability. However, the computability of markings might depend on the initial timed structure we work with. We will be confronted with this problem in part III.

Chapter 3

Automata on Timed Structures

Modeling Timed Systems

Timed structures, as we have already seen, are meant to model the evolution law of some quantitative variables and the possible effect of some external actions on this variable. An automaton on a given timed structure additionally describes what are the different discrete states of the system, how and when we can evolve from one state to another, and what consequences this evolution has on the quantitative variables.

Hybrid automata [35] consist of a finite set of states and transitions which allow evolving from one state to another *instantaneously*. Those transitions are equipped with requirements on some quantitative variables. Each state determines the evolution laws of those variables by the mean of a *differential equation*. Finally, it is allowed in the model to alter the variables after a transition is applied (e.g. reinitializing it to 0). We know that all hybrid automata can't be determinized, which motivates our will to use a more powerful model when it comes to diagnosis. As we will see in chapter 5, an automaton on timed structures can be simulated by another automaton on timed structures which is deterministic and has good enough properties to be used in practice. It makes it in our opinion a better tool for diagnosis. We argue more on this matter in chapter 9.

An automaton on a given timed structure possesses a finite set of states and a set of transitions with requirements on the quantitative variables like in hybrid automata. However, the evolution laws are no longer specified in each state but are fixed by the timed structures. With updates, the timed structure also fixes what alterations are possible after a transition. Also if the timed structure is guarded, it fixes the shape of those requirements and alterations (e.g. the requirement should be in the form of an interval bounded by integers, or we cannot reset a negative clock, ...). It doesn't imply that automata on timed structures are unable to model different evolution laws in each state. If we would be to model the temperature in a room $\mathtt{t}$, we would define a timed structure which takes into account the two possible laws of evolution of $\mathtt{t}$. For example it could be $S = \langle \mathbb{R} \times \{\mathtt{on}, \mathtt{off}\}, \hookrightarrow, \{\mathbf{on}, \mathbf{off}\}\rangle$ with : (1) the updates $\mathbf{on}$ and $\mathbf{off}$ just changing the second member of the variable, e.g. $\mathbf{on}(12.5, \mathtt{off}) = (12.5, \mathtt{on})$; (2) and $\hookrightarrow$ making the temperature drop if the state of the variable if $\mathtt{off}$, e.g. $(13, \mathtt{off}) \overset{1}{\hookrightarrow} (12.5, \mathtt{off})$; and making the temperature rise if the state if $\mathtt{on}$, e.g. $(12.5, \mathtt{on}) \overset{5}{\hookrightarrow} (13, \mathtt{on})$. In an automaton on S, we must equip the update $\mathbf{on}$ on any transition modeling an external intervention which turns on the heat. The change in the law of evolution is this way prerecorded in S.

This chapter first exposes in section 3.1 the definition of automata on timed structures and guarded timed structures along with their representations. Then in section 3.2 we extend the notion of functional bisimulation on automata on timed structures and prove some technical, yet important, propositions we will use as tools in several proofs later in this thesis.

3.1 Definition and Representation

An automaton on timed structures, as its name says, is defined for a fixed timed structure. We give the formal definition of this model below. Because we are interested in determinization, and therefore in languages, we decide to explicit in the definition, initial and final states.

Definition 3.1.1. *Fix $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$, a timed structure. An automaton on $\mathcal{S}$ is a tuple $A = \langle Q, I, T, F\rangle$ where Q is a finite set of states, $I \subseteq Q \times V$ is the set of initial configurations, $T \subseteq Q \times V \times U \times Q$ is the transition relation, and $F \subseteq Q$ is the set of final states.*

We write $\mathbb{A}(\mathcal{S})$ for the set of automata over the timed structure $\mathcal{S}$.

In an automaton on a timed structure $\mathcal{S}$, Q defines all the possible (discrete) states the model can be in, $\mathcal{S}$ describes how the quantitative variables evolve independently to the state we are in (as we discussed it in the introduction) and T describes which state can be accessed from which state, with which requirement on the value of the quantitative variables and applying which update. For now, guards are not given any role. They will be treated later in this section. Finally I and F describe initial and final states, used in the definition of the language of an automaton. Notice however that I is a set of states *and values* and F is only a set of states.

As it is commonly done we define below the semantics of automata on timed structures. The data, at some point of the execution of the system, of both the discrete state of the automaton and the values of the quantitative variables is called a *configuration*. We associate to each automaton on timed structures a labeled transition system, called its *semantics*, describing how the configurations of the model evolve from one to another; in other terms which configurations are reachable from which configurations. We formalize it below.

Fix a timed structure $\mathcal{S} = (V, \hookrightarrow, U)$ and an automaton A on $\mathcal{S}$. A *configuration* of A is an element of $Q \times V$. $\mathbf{Conf}(A) = Q \times V$ is then called the *configuration space* of A. A configuration describes at a given moment in time all the relevant information describing the system we model. There may be several possible evolutions of the system from a configuration.

If there are no external interventions and we let the time flow, only the quantitative variables will evolve in the way defined by the timed structure. Formally we define for all $d \in \mathbb{R}_{\geq 0}$, the relation $\xrightarrow{d}_A$ on $\mathbf{Conf}(A) \times \mathbf{Conf}(A)$ defined as $\xrightarrow{d}_A = \{((q, v), (q, v')) \mid q \in Q, v, v' \in V \text{ and } v \overset{d}{\hookrightarrow} v'\}$.

If now we consider an external intervention, it is modeled by the effect of a transition in the automaton. Let $t = (q, v, u, q') \in T$, t means that in our model, if we are in the configuration (q, v) the transition t can be applied with as effect, a change of state – toward q' – and the application of an update – u – in such a way that we end in configuration $(q', u(v))$. Formally we define for all $t = (q, v, u, q') \in T$, the relation $\xrightarrow{t}_A$ on $\mathbf{Conf}(A) \times \mathbf{Conf}(A)$ defined as $\xrightarrow{t}_A = \{((q, v), (q', u(v)))\}$.

The *semantics* of A is then the labeled transition system $\mathbf{Sem}(A) = (\mathbf{Conf}(A), \to_A, T \uplus \mathbb{R}_{\geq 0})$ with $\to$: $T \uplus \mathbb{R}_{\geq 0} \to \mathcal{P}(\mathbf{Conf}(A) \times \mathbf{Conf}(A))$ defined as $\to(x) = \xrightarrow{x}$.

All configuration $c_i \in I$ are called *initial configurations*. A run of A is a path of T_A starting from an initial configuration. We write $\mathbf{Reach}(A)$ for $\bigcup_{c_i \in I} \mathbf{Reach}(\mathbf{Sem}(A), c_i)$, the reachable configurations following a run of A.

A configuration $(q, v) \in \mathbf{Conf}(A)$ is said to be *final* if $q \in F$. The language of A is then the set of words recognized by a path starting in an initial configuration and ending in a final configuration. Formally we write

$$\mathcal{L}(A) = \bigcup_{c_i \in C} \mathcal{L}(\mathbf{Sem}(A), c_i, F \times V)$$

Now we focus more on the *syntax* of the model and its representation. By fixing a particular timed structure $\mathcal{S}$ we obtained a set $\mathbb{A}(\mathcal{S})$ of automata. Though without some more regularity in the model there is little hope to be able to represent nor compute anything. Those guards are the tools which enforce this regularity and stand at the basis of the classification we make within $\mathbb{A}(\mathcal{S})$ and the representation we define for automata on timed structures.

Using guarded structures we impose a particular shape on the guards of the automata. That's one way to define computable classes of automata on timed structures. We expose below step by step how.

Actually the definition of an automaton on a timed structure naturally defines some guards on the timed structure, in the way describe below:

Definition 3.1.2. *Let $\mathcal{S} = \langle S, \hookrightarrow, U\rangle$ be a timed structure, $A = \langle Q, I, T, F\rangle \in \mathbb{A}(\mathcal{S})$ be a automaton over $\mathcal{S}$ and $q, q' \in Q$. We call* guard *from q to q' the guard defined by*

$$g_A(q, q') = \{(v, u) \in V \times U \ \mid \ (q, v, u, q') \in T\}$$

g_A is called the guard function of A.

Notice that a guard $g_A(q, q')$ between two states q, q' may contain some superfluous information in terms of reachability. Indeed q may not be reachable, which makes all the information provided by this guard superfluous. Or q may be reachable only with a set of values strictly smaller than the one considered in the guard. We introduce an object useful in our work which filtrates all the superfluous transitions of the model. Guards again naturally arise as the right object to do so. The guard is called *minimal* because no more information can be removed without impacting the set of reachable configurations of the model.

Definition 3.1.3. *Let $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure and $A = \langle Q, I, T, F\rangle$ be an automaton over $\mathcal{S}$. For all $q, q' \in Q$, the minimal guard from q to q', is defined as:*

$$g_A^{\mathbf{acc}}(q, q') = \{(v, u) \in V \times U \ \mid \ (q, v) \in \mathbf{Reach}(A) \text{ and } (q, v, u, q') \in T\}$$

An automaton can always be purged of its superfluous transitions by redefining its transition relation so that the new guards between states are minimal. This purge would preserve the set of reachable configuration from any configuration.

We can then define below how an automaton on a timed structure can be defined so that it respects a given guard base.

Definition 3.1.4. *Let $\mathcal{S}_G$ be a guarded timed structure and $A = \langle Q, I, T, F\rangle \in \mathbb{A}(\mathcal{S})$ be a automaton on $\mathcal{S}$. We say that A is compatible with G if g_A is decomposable in G.*

Whenever an automaton A on $\mathcal{S}$ is compatible with G, we say for short that A is an automaton on $\mathcal{S}_G$.

Let $\mathbb{A}(\mathcal{S}_G)$ be the set of all automata on $\mathcal{S}_G$.

If S_G are computable, we say that all automata $A \in \mathbb{A}(\mathcal{S}_G)$ is computable and that $\mathbb{A}(\mathcal{S}_G)$ is a computable class of automata on a timed domain.

Below we give some examples of classes of automata on guarded timed structures and in the meantime show how some classical classes we are interested in throughout this thesis, fit in our new model. To have an even better understanding of the concrete effect of guarded timed structures we advise the reader to advance pass those examples to see how we represent automata on guarded timed structures and see the example of representation we provide.

Example 3.1.1. Let $M \in \mathbb{N}$ and $n \in \mathbb{N}_{>0}$.

The class of M*-bounded* n*-clocks timed automata* [4] is the set, $\mathbf{TA}^n_M$, of all automata on $\mathcal{C}^n_M$, the (M,n)-clock guarded structure defined in example 2.3.4 and 2.3.1. Recall that this guarded structure requires for a guard to impose: one or both of the updates $\mathbf{0}$ or $\mathbf{id}$; and the values to belong to an integer bounded interval with bounds smaller than M. An automaton which guards respect the (M,n)-clock guarded structure must have a transition relation which respects the same rules as a transition relation of a timed automata of [4]. To be exact, such an automaton in our formalism would differ from a timed automaton of [4] only in the sense that it is not*labeled*. This will be fixed in chapter 4 with the introduction of controls.

Example 3.1.2. Let $M \in \mathbb{N}$ and C be a finite set of clocks.

The class of M*-bounded* C*-enhanced timed automata* is the set, $\mathbf{TA}^C_M$, of all automata on $[C \rightharpoonup \mathcal{C}_M]$, the M,C-enhanced clock guarded structure defined in example 2.3.6.

Example 3.1.3. Let $M \in \mathbb{N}$, $n \in \mathbb{N}_{>0}$ and Z a stack alphabet.

The class of M*-bounded* n*-clock* Z*-pushdown timed automata* is the set, $\mathbf{PDTA}^{\mathbf{n}}_{\mathbf{M},\mathbf{Z}}$, of all automata on $\mathcal{Z}_{M,n,Z}$ defined in example 2.3.5.

A canonic way of representing an automaton on a guarded timed structure is through what we call its *graph*. Fix a guarded timed structure $\mathcal{S}_G$ and let $A = \langle Q, I, T, F\rangle \in \mathbb{A}(\mathcal{S}_G)$. Formally we define the *graph* of A as the labeled directed multi-graph $\mathcal{G}_A = \langle Q, E_A, G\rangle$ with Q the set of nodes, G the set of labels, and $E_A \subseteq Q \times G \times Q$ the set of labeled directed edges, defined as

$$E_A = \{(q, \nu, \mu, q') \in Q \times G \times Q \mid (\nu, \mu) \in g_A(q, q')\}$$

Because A is compatible with G, we have that $\mathcal{G}_A$ is finite, which is a property we expect from a representation.

Each edge $(q, \nu, \mu, q') \in E_A$ has to be understood as: *from every configuration* (q, v) *with* $v \in \nu$ *it is possible to take a transition toward* q' *and any update within* μ *can be (non-deterministically) applied*.

Example 3.1.4. Let $\mathcal{S}_G$ be a guarded timed structure such that $G = \{(\nu_1, \mu_1), (\nu_2, \mu_2), (\nu_3, \mu_3)\}$.

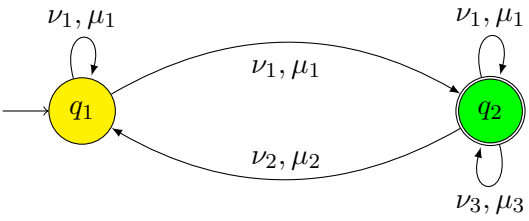


Figure 3.1: Graph of an automaton on $\mathcal{S}_G$

The automaton on $\mathcal{S}_G$ whose graph is depicted on Figure 3.1 is the automaton $A = \langle\{q_1, q_2\}, \{q_1\}, T, \{q_2\}\rangle$ where

$$\begin{aligned} T = &\{(q_1, v, u, q_1), (v, u) \in \nu_1 \times \mu_1\} \cup \\ &\{(q_1, v, u, q_2), (v, u) \in \nu_1 \times \mu_1\} \cup \\ &\{(q_2, v, u, q_2), (v, u) \in \nu_1 \times \mu_1\} \cup \\ &\{(q_2, v, u, q_2), (v, u) \in \nu_3 \times \mu_3\} \cup \\ &\{(q_2, v, u, q_1), (v, u) \in \nu_2 \times \mu_2\} \end{aligned}$$

The initial state is colored in yellow and the final state in green.

Example 3.1.5. In example 3.1.1 we formalized in our framework the class of timed automata. Consider $M = 2$, $n = 2$, $Q = \{q_1, q_2\}$, we define a 2-bounded 2-clock timed automaton A with Q as set of states. The set of transition T cannot be define freely: we imposed that the guards of A are decomposable in $\mathbb{G}_2^2$, the guard base of the $(2, 2)$-clock guarded structure on which A is defined. T must then be defined such that the guards between two states of Q require that the values of both clocks (we name them x and y) are within an integer bounded interval (c.f. example 2.3.1. Suppose we choose $T = \{(q_1, v, (\mathbf{id}, \mathbf{0}), q_2), v \in (1, +\infty) \times (1, +\infty)\times\} \cup \{(q_2, (x, 1), (\mathbf{0}, \mathbf{0}), q_2), x > 1\}$, the guards are

$$\begin{aligned} g_A(q_1, q_1) &= \emptyset \\ g_A(q_1, q_2) &= (1, +\infty) \times (1, +\infty) \times \{(\mathbf{id}, \mathbf{0})\} \\ &= \{\langle(1, 2) \times (1, 2), [2, 2] \times (1, 2), (2, +\infty) \times (1, 2), \ldots, (2, +\infty) \times (2, +\infty), \{(\mathbf{id}, \mathbf{0})\}\rangle\} \\ g_A(q_2, q_2) &= (1, +\infty) \times [1, 1] \times \{(\mathbf{0}, \mathbf{0})\} \\ &= \{\langle(1, 2) \times [1, 1], [2, 2] \times [1, 1], (2, +\infty) \times [1, 1], \{(\mathbf{0}, \mathbf{0})\}\rangle\} \end{aligned}$$

which are all decomposable on $\mathbb{G}_2^2$. Adding for example *only one* transition $(q_1, 1.5, 1, (\mathbf{0}, \mathbf{id}), q_2)$ would be impossible since A must respect the guard base. Suppose $I = \{(q_1, (0, 0))\}$ and $F = \{q_2\}$ the graph of A can be found on figure 3.2. Which exactly correspond to the timed automata in figure 3.3, represented this time with the standard notation of [4].

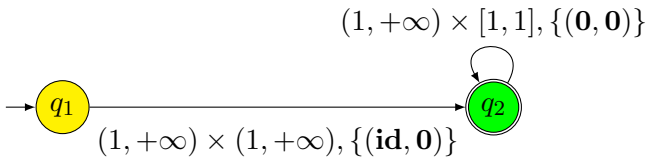


Figure 3.2: Representation of the timed automata A

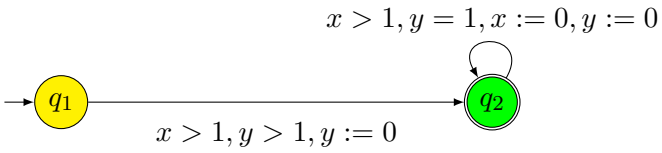


Figure 3.3: Representation of the timed automata A as in [4]

Example 3.1.6. In example 3.1.2 we formalized the class of enhanced timed automata, we give another example of how we can define and represent an automaton in this context. Consider

$M = 2$, $C = \{x, y\}$, $Q = \{q_1, q_2\}$, we define a 2-bounded C-enhanced timed automata B with Q as set of states. The set of transitions T must respect $\mathbb{G}_2^C$, the guard base of the $(2, C)$-clock guarded structure on which B is defined. We choose $T = \{(q_1, v, \mathbf{u}_{f_1}, q_2), v \in (1, +\infty) \times (1, +\infty)\times\} \cup \{(q_2, (1, y), \mathbf{u}_{f_2}, q_2), y > 1\}$, where f_1 maps x to $(\mathbf{0}, x)$ and y to $(\mathbf{id}, x)$; and f_2 maps y to $(\mathbf{0}, y)$ and x is not in its domain. We use as a shorthand $(E_1 \times E_2)^C$ when we really mean, all the partial functions on C such that whose image is included in $E_1 \times E_2$. The guards are

$$
\begin{aligned}
g_A(q_1, q_1) &= \emptyset \\
g_A(q_1, q_2) &= ((1, +\infty) \times (1, +\infty))^C \times \{\mathbf{u}_{f_1}\} \\
&= \{\langle((1, 2) \times (1, 2))^C, ([2, 2] \times (1, 2))^C, \ldots, ((2, +\infty) \times (2, +\infty))^C, \{\mathbf{u}_{f_1}\}\rangle\} \\
g_A(q_2, q_2) &= ([1, 1] \times (1, +\infty))^C \times \{\mathbf{u}_{f_2}\} \\
&= \{\langle((1, 2) \times [1, 1])^C, ([2, 2] \times [1, 1])^C, ((2, +\infty) \times [1, 1])^C, \{\mathbf{u}_{f_2}\}\rangle\}
\end{aligned}
$$

which are all decomposable on $\mathbb{G}_2^C$. Suppose $I = \{(q_1, (x, y) \mapsto (0, 0))\}$ and $F = \{q_2\}$ the graph of B can be found on figure 3.4. This representation may not seem easily understandable but with an adequate symbolic representation of $\mathbf{u}_f$ as described in 2.3.6 we come closer to a representation we understand as furnished in figure 3.5.

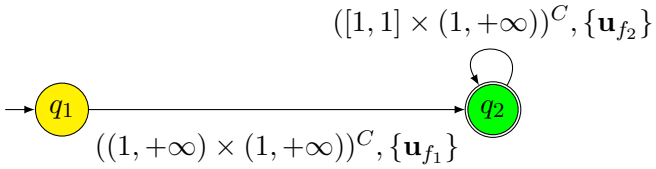


Figure 3.4: Representation of the enhanced timed automata B

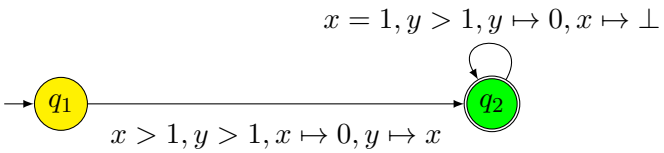


Figure 3.5: Representation of the timed automata B with symbolic notations

Keep in mind that for now, we seek to describe the complete mechanism of our system, where each effect of each evolution law is known and specified in the model. The modeling of *indistinguishable behavior*, will be introduced with *controls* in chapter 4.

3.2 Bisimulation for Automata on Timed Structures

As we argued in section 1.3, *similarities* between models have an important role to play in our work. Since we associated with each automaton on a timed structure, a semantic in the form of a labeled transition system, it seems natural to define similarity between automata by extending similarity between their semantics.

More formally we say that an automaton A_1 *simulates* an automaton A_2 if there exists a functional bisimulation from $\mathbf{Sem}(A_1)$ to $\mathbf{Sem}(A_2)$ which preserves the initial and the final configurations. Formally the functional bisimulation (ϕ, ψ) has to respect :

- $\phi(I_1) = I_2$ with I_1 and I_2 the initial states of A_1 and A_2 respectively
- for all configuration of A_1, $(q_1, v_1) \in \mathbf{dom}(\phi)$, (q_1, v_1) is a final configuration of A_1 if and only if $\phi(q_1, v_1)$ is a final configuration of A_2.

In this case, we can conclude using Proposition 1.3.3 that $\psi(\mathcal{L}(A_1)) = \mathcal{L}(A_2)$.

Remark 3.2.1. Recall that a functional bisimulation is a weaker notion than bisimulation equivalence. In particular, due to its functional form, if A_1 simulates A_2, it doesn't necessarily mean that A_2 simulates A_1.

However, to avoid going back to the semantics each time we want to speak about simulation, we avoid using directly this definition of simulation between automata and adapt the notion of functional bisimulation to be defined directly with the automata specification rather than on the semantic. This new notion of simulation is weaker than the functional bisimulation between labeled transition systems, but suits our needs better.

Recall that we adapted the notion of function bisimulation between two timed domains (c.f. section 2.1) and that it is characterized by a partial function between their value space.

Proposition 3.2.1. *Let $\mathcal{S}_1 = \langle V_1, \hookrightarrow_1, U_1 \rangle$ and $\mathcal{S}_2 = \langle V_2, \hookrightarrow_2, U_2 \rangle$ be two timed structures and $A_1 = \langle Q_1, I_1, T_1, F_1 \rangle$ and $A_2 = \langle Q_2, I_2, T_2, F_2 \rangle$ be two automata on $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively.*

Suppose there exists a triple $\Xi = (\zeta, \phi, \psi)$ such that:

(1) $\zeta : Q_1 \to Q_2$ *is a function such that* $\zeta(F_1) = F_2$

(2) $\phi = (\phi_{q_1})_{q_1 \in Q_1}$ *is a family of functional bisimulations from* $\langle V_1, \hookrightarrow_1 \rangle$ *to* $\langle V_2, \hookrightarrow_2 \rangle$ *such that*

(2.1) $\{(\zeta(q_1), \phi_{q_1}(v_1)), (q_1, v_1) \in I_1\} = I_2$.

(2.2) for all $q_1 \in Q_1$, $\{v \in V_1 \mid (q_1, v) \in \mathbf{Reach}(A_1)\} \subseteq \mathbf{dom}(\phi_{q_1})$

(3) $\psi = (\psi_{q_1, q_1'})_{q_1, q_1' \in Q_1}$ *is a family of partial functions from* $\mathbf{dom}(\phi_{q_1}) \times U_1$ *to* U_2 *such that for all* $q_1, q_1' \in Q_1$ *and for all* $(v_1, u_1) \in \mathbf{dom}(\phi_{q_1}) \times U_1$,

(3.1) $g^{\mathbf{acc}}_{A_1}(q_1, q_1') \subseteq \mathbf{dom}(\psi_{q_1, q_1'})$

(3.2) $u_1(v_1) \in \mathbf{dom}(\phi_{q_1'})$ *and* $\phi_{q_1'}(u_1(v_1)) = \psi_{q_1, q_1'}(v_1, u_1)(\phi_{q_1}(v_1))$

(3.3) $(\phi_{q_1}(v_1), \psi_{q_1, q_1'}(v_1, u_1)) \in g_{A_2}(\zeta(q_1), \zeta(q_1'))$

(4) Finally, for all $q_1, v_1 \in \mathbf{Reach}(A_1)$, *for all* $q_2' \in Q_2$, *for all* $u_2 \in U_2$, *if* $(\phi_{q_1}(v_1), u_2) \in g^{\mathbf{acc}}_{A_2}(\zeta(q_1), q_2')$ *then there exists* $q_1' \in \zeta^{-1}(q_2')$ *and* $u_1 \in U_1$ *such that*

$$\psi_{q_1, q_1'}(v_1, u_1) = u_2 \text{ and } (q_1, v_1, u_1, q_1') \in T_1$$

In this case A_1 *simulates* A_2.

Proof.

Let us define

$$\begin{array}{rccll} \Phi: & \mathbf{Conf}(A_1) & \rightharpoonup & \mathbf{Conf}(A_2) & \\ & (q_1, v_1) & \mapsto & (\zeta(q_1), \phi_{q_1}(v_1)) & \text{if } (q_1, v_1) \in \mathbf{Reach}(A_1) \end{array}$$

and let

$$\begin{array}{rccll} \Psi: & \mathbb{R}_{\geq 0} \uplus T_1 & \rightharpoonup & \mathbb{R}_{\geq 0} \uplus T_2 & \\ & d & \mapsto & d & \text{if } d \in \mathbb{R}_{\geq 0} \\ & (q_1, v_1, u_1, q_1') & \mapsto & (\zeta(q_1), \phi_{q_1}(v_1), \psi_{q_1,q_1'}(v_1, u_1), \zeta(q_1')) & \text{if } (v_1, u_1) \in g_A^{\mathbf{acc}}(q_1, q_1') \end{array}$$

Φ and Ψ are well defined, by hypothesis on the domains of ϕ_{q_1} and $\psi_{q_1,q_1'}$.
We prove that (Φ, Ψ) is a functional bisimulation from $\mathbf{Sem}(A_1)$ to $\mathbf{Sem}(A_2)$.

We prove first that (Φ, Ψ) is a functional simulation.
Let $l_1 \in \mathbb{R}_{\geq 0} \uplus T_1$, $(q_1, v_1) \in \mathbf{dom}(\Phi)$ and $(q_1', v_1') \in Q_1 \times V_1$, such that $(q_1, v_1) \xrightarrow{l_1}_1 (q_1', v_1')$.
$(q_1', v_1') \in \mathbf{Reach}(A_1) = \mathbf{dom}(\Phi)$.

- Suppose $l_1 = d \in \mathbb{R}_{\geq 0}$.
  First $l_1 \in \mathbf{dom}(\Psi)$ by definition.
  Then $q_1' = q_1$ and $v_1 \overset{d}{\hookrightarrow}_1 v_1'$.
  Therefore $\zeta(q_1) = \zeta(q_1')$ and since ϕ_{q_1} is a functional simulation of timed domains, $\phi_{q_1}(v_1) \overset{d}{\hookrightarrow}_2 \phi_{q_1}(v_1')$.
  As consequence since $\Psi(l_1) = l_1 = d$,
  $\Phi(q_1, v_1) = (\zeta(q_1), \phi_{q_1}(v_1)) \xrightarrow{\Psi(l_1)}_2 (\zeta(q_1'), \phi_{q_1'}(v_1')) = \Phi(q_1', v_1')$.

- Suppose $l_1 \in T_1$.
  Then $l_1 = (q_1, v_1, u_1, q_1')$ and $v_1' = u_1(v_1)$ by definition.
  This also means that $(v_1, u_1) \in g_A^{\mathbf{acc}}(q_1, q_1') = \mathbf{dom}(\Psi)$ since $(q_1, v_1) \in \mathbf{Reach}(A_1)$.
  $(v_1'', u_1) \in g_{A_1}^{\mathbf{acc}}(q_1'', q_1')$, therefore
  $\Psi(l_1) = (\zeta(q_1), \phi_{q_1}(v_1), \psi_{q_1,q_1'}(v_1, u_1), \zeta(q_1')) \in T_2$ by hypothesis and
  $(\zeta(q_1), \phi_{q_1}(v_1)) \xrightarrow{\Psi(l_1)}_2 (\zeta(q_1'), \psi_{q_1,q_1'}(v_1, u_1)(\phi_{q_1}(v_1)))$.
  Which is equivalent, by hypothesis, as
  $(\zeta(q_1), \phi_{q_1}(v_1)) \xrightarrow{\Psi^*(l_1)}_2 (\zeta(q_1'), \phi_{q_1'}(u_1(v_1)))$ and finally as
  $\Phi(q_1, v_1) \xrightarrow{\Psi(l_1)}_2 \Phi(q_1', v_1')$.

This proves that (Φ, Ψ) is a functional simulation from $\mathbf{Sem}(A_1)$ to $\mathbf{Sem}(A_2)$.

We prove now that (Φ, Ψ) is a functional bisimulation.
Let $l_2 \in (\mathbb{R}_{\geq 0} \uplus T_2)$, $(q_1, v_1) \in \mathbf{dom}(\Phi) = \mathbf{Reach}(A_1)$ and
$(q_2', v_2') \in Q_2 \times V_2$, such that $\Phi(q_1, v_1) \xrightarrow{l_2}_2 (q_2', v_2')$.

- Suppose $l_2 = d \in \mathbb{R}_{\geq 0}$.
  Then $q_2' = \zeta(q_1)$ and $\phi_{q_1}(v_1) \overset{d}{\hookrightarrow}_2 v_2'$.
  Therefore since ϕ_{q_1} is a functional bisimulation of timed domains,
  exists $v_1' \in \phi_{q_1}^{-1}(v_2')$ such that $v_1 \overset{d}{\hookrightarrow}_1 v_1'$.
  As consequence since $\Psi(l_2) = l_2 = d$ and $q_1, v_1 \overset{d}{\rightarrow}_1 q_1, v_1'$,
  q_1, v_1' and d are matching candidates for the required existence property.

► Suppose now $l_2 \in T_2$,
then exists $u_2 \in U_2$ such that $l_2 = (\zeta(q_1), \phi_{q_1}(v_1), u_2, q_2') \in T_2$ and $u_2(\phi_{q_1}(v_1)) = v_2'$.
Since (Φ, Ψ) is a functional simulation, according to proposition 1.3.3, $(\zeta(q_1), \phi_{q_1}(v_1)) \in \mathbf{Reach}(A_2)$.
We have then $(\phi_{q_1}(v_1), u_2) \in g_{A_2}^{\mathbf{acc}}(\zeta(q_1), q_2')$.
The end of the proof is straight-forward
using condition (3) and (4) and since $(q_1, v_1) \in \mathbf{Reach}(A_1)$.

Now it only remains to prove that Φ preserves initial and final configurations:

- $\Phi(I_1) = I_2$ by definition of Φ and according to hypothesis (2.1)
- Let $(q_1, v_1) \in \mathbf{dom}(\Phi)$, using hypothesis (1) we know that $q_1 \in F_1$ if and only if $\zeta(q_1) \in F_2$. We can easily prove from the definition of Φ that (q_1, v_1) is a final configuration of A_1 if and only $\Phi(q_1, v_1)$ is a final configuration of A_2.

In conclusion, by definition, A_1 simulates A_2. ■

A triple $\Xi = (\zeta, \phi, \psi)$, satisfying conditions (1) to (3) of proposition 3.2.1 above is called a *functional simulation* from A_1 to A_2. And indeed the first part of the proof of proposition 3.2.1 which proves that Ψ is a functional simulation does not use item (4).

If moreover Ξ satisfy (4) we call it a *functional bisimulation* from A_1 to A_2.

In several cases in this thesis we will be confronted with the task of constructing, given an automaton A on a timed structure S, an automaton A' on a different timed structure S' which simulates A. For example in chapter 6 we dedicate two sections to such proofs, the section 6.1 proves in particular that an enhanced timed automaton can always be simulated by a timed automaton.

Classically such proof is done by exhibiting the automaton A' and then proving that it simulates A, for example, we would do it using proposition 3.2.1. The next proposition provides a tool to merge those two steps of the proof, by exhibiting only a *pre-functional bisimulation* from a state of Q' and a timed structure S'. A *pre-functional bisimulation* is like a cooking recipe to construct A' with the benefits of satisfying all the good properties to be a functional bisimulation once A' constructed. Indeed after a description of the properties a pre-functional bisimulation must verify, proposition 3.2.2 provide two things :

- A definition of the automaton A' on timed structure S', based on the pre-functional bisimulation, and with Q' as set of state.
- A proof that the pre-functional bisimulation is a functional bisimulation from A' to A.

The second part is the main part of the proof of the proposition.

Proposition 3.2.2. *Let $\mathcal{S}_2 = \langle V_2, \hookrightarrow_2, U_2\rangle$ be a timed structure and $A_2 = \langle Q_2, I_2, T_2, F_2\rangle$ be an automaton over $\mathcal{S}_2$.*

Let $\mathcal{S}_1 = \langle V_1, \hookrightarrow_1, U_1\rangle$ be a timed structure and Q_1 be a finite set.

Suppose there exists a triple $\Xi = (\zeta, \phi, \psi)$ such that:

(A) $\zeta : Q_1 \to Q_2$ *is function such that* $\zeta(Q_1 = \zeta(Q_2)$ *a surjective function.*

(B) $\phi = (\phi_{q_1})_{q_1 \in Q_1}$ *is a family of functional bisimulations from* $\langle V_1, \hookrightarrow_1\rangle$ *to* $\langle V_2, \hookrightarrow_2\rangle$ *such that for all* $(q_2, v_2) \in I_2$*, there exists* $q_1 \in \zeta^{-1}(q_2)$ *such that* $v_2 \in \mathbf{im}(\phi_{q_1})$*.*

(C) $\psi = (\psi_{q_1,q_1'})_{q_1,q_1' \in Q_1}$ *is a family of partial function from* $\mathbf{dom}(\phi_{q_1}) \times U_1$ *to* U_2 *such that for all* $q_1, q_1' \in Q_1$ *and for all* $(v_1, u_1) \in \mathbf{dom}(\psi_{q_1,q_1'})$

(C.1) $u_1(v_1) \in \mathbf{dom}(\phi_{q_1'})$ *and* $\phi_{q_1'}(u_1(v_1)) = \psi_{q_1,q_1'}(v_1, u_1)(\phi_{q_1}(v_1))$

(C.2) $(\phi_{q_1}(v_1), \psi_{q_1,q_1'}(v_1, u_1)) \in g_{A_2}(\zeta(q_1), \zeta(q_1'))$

(D) for all $q_1, v_1 \in Q_1 \times \mathbf{dom}(\phi_{q_1})$*, for all* $q_2' \in Q_2$*, for all* $u_2 \in U_2$*, if* $(\phi_{q_1}(v_1), u_2) \in g_{A_2}^{\mathbf{acc}}(\zeta(q_1), q_2')$ *then exists* $q_1' \in \zeta^{-1}(q_2')$ *and* $u_1 \in U_1$ *such that*

$$\psi_{q_1,q_1'}(v_1, u_1) = u_2$$

Then $A_1 = \langle Q_1, I_1, T_1, F_1\rangle$*, where* $I_1 = \{(q_1, v_1) \in Q_1 \times V_1 \mid (\zeta(q_1), \phi_{q_1}(v_1)) \in I_2\}$*,* $F_1 = \zeta^{-1}(F_2)$ *and*

$$T_1 = \{(q_1, v_1, u_1, q_1') \in Q_1 \times V_1 \times U_1 \times Q_1 \mid (v_1, u_1) \in \mathbf{dom}(\psi_{q_1,q_1'})\}$$

is a automaton on $\mathcal{S}_1$ *which simulates* A_2*.*

Proof.

We just prove that the triple (ζ, ϕ, ψ) with $\psi = (\phi_{q_1})_{q_1 \in Q_1}$ and $\psi = (\psi_{q_1,q_1'})_{q_1,q_1' \in Q_1}$ is a functional bisimulation from A_1 to A_2 by showing it verifies every four conditions of proposition 3.2.1.

(1) Condition (1) is verified by surjectivity of ζ (A) and definition of F_1.

(2) **(2.1)** Condition (2.1) can be easily proved using surjectivity of ζ (A), hypothesis (B), and definition of I_1.

(2.2) Let $q_1 \in Q_1$ and $v_1 \in V_1$, such that $(q_1, v_1) \in \mathbf{Reach}(A_1)$.
We prove by induction on the length of the run of A_1 ending on (q_1, v_1) that $v_1 \in \mathbf{dom}(\phi_{q_1})$.

- If (q_1, v_1) is reachable with a run of length 0 then (q_1, v_1) is an initial configuration, and $v_1 \in \mathbf{dom}(\phi_{q_1})$ by definition of I_1.
- If there exists (q_1, v_1') reachable with a run of length n and $(q_1, v_1') \xrightarrow{d}_{A_1} (q_1, v_1)$ with $d \in \mathbb{R}_{\geq 0}$, then $v_1' \overset{d}{\hookrightarrow} v_1$.
  By induction hypothesis $v_1' \in \mathbf{dom}(\phi_{q_1})$ and since ϕ_{q_1} is a functional bisimulation of timed domain, $v_1 \in \mathbf{dom}(\phi_{q_1})$.
- If finally there exists (q_1', v_1') reachable with a run of length n and $(q_1', v_1') \xrightarrow{(q_1', v_1', u_1, q_1)}_1 (q_1, v_1)$, then $u_1(v_1') = v_1$ and $(v_1', u_1) \in \mathbf{dom}(\psi_{q_1',q_1})$.
  By induction hypothesis $v_1' \in \mathbf{dom}(\phi_{q_1'})$ and by hypothesis then $v_1 = u_1(v_1') \in \mathbf{dom}(\phi_{q_1})$.

(3) For all $q_1, q_1' \in Q_1$, $g_{A_1}^{\mathbf{acc}}(q_1, q_1') \subseteq g_{A_1}(q_1, q_1') = \mathbf{dom}(\psi_{q_1,q_1'})$ by definition, which prove (3.1).

(3.2) and (3.3) are enforced on $\psi_{q_1,q_1'}$ by hypothesis $(C.1)$ and $(C.2)$ respectively.

(4) Let $q_1, v_1 \in \mathbf{Reach}(A_1)$, let $q_2' \in Q_2$, let $u_2 \in U_2$.
Suppose that $(\phi_{q_1}(v_1), u_2) \in g_{A_2}^{\mathbf{acc}}(\zeta(q_1), q_2')$.
We already proved that $v_1 \in \mathbf{dom}(\phi_{q_1})$ (condition (2)).
According to hypothesis (4), we know that there exists $q_1' \in \zeta^{-1}(q_2')$ and $u_1 \in U_1$ such that $\psi_{q_1,q_1'}(v_1, u_1) = u_2$.
This implies also that $(v_1, u_1) \in \mathbf{dom}(\psi_{q_1,q_1'})$, hence that $(q_1, v_1, u_1, q_1') \in T_1$.

This concludes the proof that Ξ is a functional bisimulation from A_1 to A_2, i.e. that A_1 simulates A_2.

■

A triple $\Xi = (\zeta, \phi, \psi)$ satisfying conditions (A) to (D) is called a *pre-functional bisimulation* from $Q_1, \mathcal{S}_1$ to A_2. A_1 is then called the inverse image of A_2 and is written $\Xi^{-1}(A_2)$.

We prove below our first extension of proposition 3.2.1 to encompass preservation of guard compatibility.

Proposition 3.2.3. *Let $\mathcal{S}_2 = \langle V_2, \hookrightarrow_2, U_2 \rangle$ be a timed structure and $A_2 = \langle Q_2, I_2, T_2, F_2 \rangle$ be an automaton over $\mathcal{S}_2$.*

Let $\mathcal{S}_G = \langle V_1, \hookrightarrow_1, U_1, G \rangle$ be a guarded timed structure, Q_1 be a finite set and $\Xi = (\zeta, \phi, \psi)$ be a pre-functional bisimulation from $Q_1, \mathcal{S}_1$ to A_2 with $\phi = (\phi_{q_1})_{q_1 \in Q_1}$ and $\psi = (\psi_{q_1,q_1'})_{q_1,q_1' \in Q_1}$.

Suppose for all $q_1, q_1' \in Q_1$:

$$\mathbf{dom}(\psi_{q_1,q_1'}) \text{ is decomposable on } G$$

Then $\Xi^{-1}(A_2)$ is an automaton on $\mathcal{S}_1$, compatible with G, which simulates A_2.

Proof.

Notice just that $A_1 = \Xi^{-1}(A_2)$ is compatible with G since for all $q_1, q_1' \in Q_1$, $g_{A_1}(q_1, q_1') = \mathbf{dom}(\psi_{q_1,q_1'})$ is decomposable on G by hypothesis (c.f. proof of proposition 3.2.1). The rest is just by application of proposition 3.2.1. ■

In this case Ξ is called a *G-compatible pre-functional bisimulation* from $Q_1, \mathcal{S}_1$ to A_2.

The proof scheme we will use in this thesis to get simulations will always consist of proving that an automaton (or a class of automata) on a timed structure $\mathcal{S}$ can be simulated by an automaton (or a class of automata) on another timed structure $\mathcal{S}'$ by exhibiting a set of state Q' and a pre-functional bisimulation from Q' and S' (or an extension of it defined on proposition 3.2.3 or on other proposition in chapter 4.). This technique is used in particular in chapter 6 and 11.

Chapter 4

Controls

Determinism, Languages and Bisimulations

Automata on timed domains introduced in the last chapter are *unlabeled*: transitions have no labels (precisely they are labeled with their name). In the verification field, *labeling* is essential for language theoretical studies including determinization and simulation of formal model studies. Therefore it is often considered part of the formal model definition [4] [27] [16] [22] [13]. On another hand, labeling consists of an important aspect of the modeling process since it allows the formalization of indistinguishable and unobservable behaviors with regards to an external observer. In a formal point of view it allows the introduction *non-determinism* into the model. It then goes without saying that any work on the diagnosis of timed systems must consider labeled formal models. This is this chapter's goal. We choose however to separate the labeling part of the modeling from the behavioral part using an original tool we call *controls*.

In timed automata and more generally in timed systems we are often confronted with several notions of language (timed and untimed languages) and simulations (abstract and strong bisimulation equivalence, language equivalence) [4] [2] [13]. Even though determinism usually keeps a stable definition, *determinization* implies the preservation of languages and simulation implies the preservation of a bisimulation equivalence. We are thus confronted with a choice of language or bisimulation to preserve. Our target in this thesis is to keep our work as general as possible so it can be applied to as many timed systems as possible and encompass as many known results of determinization as possible. Results of this approach will be discussed in chapters 7 and 8. Since in the literature we encountered several choices of determinization and simulation [6] [5] [13], we decided not to settle for one or the other.

With the help of *controls*, different choices in languages or bisimulation equivalence are reduced to different labeling choices. Instead of fixing one way to label automata on timed systems on a given alphabet, we define in section 4.1 a framework in which we can define various ways of labeling them. A labeling defined through a control can be anything while it respects the underlying behavior of the automaton on which it is defined. In particular, controls allow the definition of labeling which alters delay transition information. It is rarely the case though that we take an interest in such labelings. In section 4.2 we define specific controls we call *timed controls*. Those controls produce labelings that necessarily allow complete and precise observation of time delays.

As a consequence, in this thesis, the labeling of an automaton is considered as an external parameter: it can be changed. All notions of language, determinism, equivalence... will be dependent on the control we equip on the automaton and will change with it. This might imply adjustment time and heavier notations for the reader, but we hope that as counterpart it will give new insights on the determinization results on timed systems proved until now.

4.1 General Definition

Following the discussion in the introduction, a *control* on an automaton should describe how transitions are viewed from the point-of-view of an external observer. Our choice of modeling is to give the possibility to the observer to see a whole labeled transition system describing the behavior of the observed automaton. Specifically: configurations are observable, reachability is preserved, but transitions and their labels can be altered. The formal definition is given below.

Definition 4.1.1. *Let Γ be a set of names called* control alphabet*. Let $\mathcal{S}$ be a timed structure and $A \in \mathbb{A}(\mathcal{S})$ an automaton on $\mathcal{S}$. A* Γ-control *K, on A is a labeled transition system of the form $K = (\mathbf{Conf}(A), \rightarrow_K, \Gamma)$, satisfying for all initial configuration c_i of A:*

$$\mathbf{Reach}(\mathbf{Sem}(A), c_i) = \mathbf{Reach}(K, c_i)$$

If Γ and $\mathcal{S}$ are fixed, we say for short that $\langle A, K\rangle \in \mathbb{A}(\mathcal{S}, \Gamma)$ meaning that $A \in \mathbb{A}(\mathcal{S})$ and K is a Γ-control on A. We also say that A *is equipped with* K, or that $\langle A, K\rangle$ is a Γ-controlled automaton on $\mathcal{S}$.

Remark 4.1.1. Notice that given an automaton $A = \langle Q, I, T, F\rangle$, any labeling κ of $\mathbf{Sem}(A)$ on a control alphabet Γ, defines a Γ-control, $\mathbf{Sem}(A)^\kappa$ on A. (c.f. section 1.3, proposition 1.3.4).

Because a control is a labeled transition system, every property or object defined on labeled transition systems is a property or an object on controls, and hence can be extended to controlled automata.

Definition 4.1.2. *Let Γ be a control alphabet, $\mathcal{S}$ be a timed structure and $\langle A, K\rangle \in \mathbb{A}(S, \Gamma)$ be a Γ-controlled automaton on $\mathcal{S}$. We say that A is K-deterministic (resp. K-complete) if and only if K is a deterministic (resp. complete) labeled transition system.*

Languages can also be extended to languages of controlled automata, defined below.

Definition 4.1.3. *Let Γ be a control alphabet, $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure and $\langle A, K\rangle$ a Γ-controlled automaton on $\mathcal{S}$. The language recognized by A equipped with K, written $\mathcal{L}_K(A)$ is the union of all languages recognized by K from any initial to any final configuration of A. Formally:*

$$\mathcal{L}_K(A) = \bigcup_{c_i \in I} \mathcal{L}(K, c_i, F \times V)$$

Example 4.1.1. Let $\Gamma = \{\delta, a, b\} \cup \mathbb{R}_{\geq 0}$. We expose two ways among others to equip a Γ-control on the automaton $A = \langle\{q_1, q_2\}, \{q_1\}, T, \{q_2\}\rangle$ on $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ of Figure 3.1 drawn again below.

- Let $\kappa : T \cup \mathbb{R}_{\geq 0} \rightarrow \Gamma$ defined as:
  - $\kappa((q, v, u, q)) = a$ for all $q \in \{q_1, q_2\}$, $v \in V$, $u \in U$ and $(q, v, u, q) \in T$.
  - $\kappa((q, v, u, q')) = b$ for all $q \neq q' \in \{q_1, q_2\}$, $v \in V$, $u \in U$ and $(q, v, u, q') \in T$.
  - $\kappa(d) = \delta$ for all $d \in \mathbb{R}_{\geq 0}$.

  κ is a labeling of $\mathbf{Sem}(A)$ on Γ, and $\mathbf{Sem}(A)^\kappa$ is a Γ-control on A.

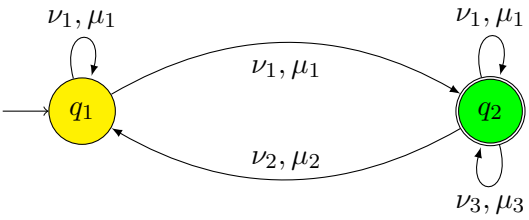


Figure 3.1: Graph of an automaton (repeated from page 26)

- Let:
  - $\xrightarrow{a}_K = \{(c, c), c \in \mathbf{Conf}(A)\}$
  - $\xrightarrow{b}_K = \{((q_1, v), (q_2, u(v))), v \in \nu_1 \text{ and } u \in \mu_1\} \cup \{((q_2, v), (q_1, u(v))), v \in \nu_2 \text{ and } u \in \mu_2\}$
  - $\xrightarrow{d}_K = \xrightarrow{d}_A$ for all $d \in \mathbb{R}_{\geq 0}$.
  - $\xrightarrow{\delta}_K = \emptyset$

  $K = (\mathbf{Conf}(A), \rightarrow_K, \Gamma)$ is a Γ-control on A. Moreover, if μ_1 and μ_2 are singletons, then A is K-deterministic.

Suppose now $\mathcal{S} = \mathcal{C}_1 = \langle \mathbb{C}_1, \hookrightarrow, \{\mathbf{id}, \mathbf{0}\} \rangle$ is the 1-clock structure, and

$$(\nu_1, \mu_1) = ([0, 0], \{\mathbf{0}\})$$
$$(\nu_2, \mu_2) = ([1, 1], \{\mathbf{0}\})$$
$$(\nu_3, \mu_3) = ([0, 1], \{\mathbf{id}\})$$

A representation is given in figure 4.2.

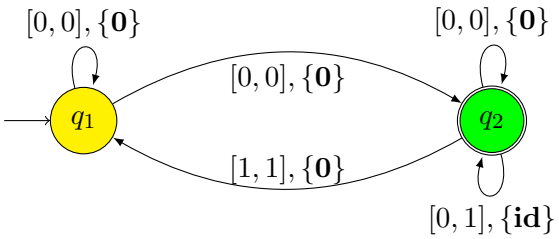


Figure 4.2: Graph of the automaton of figure 3.1 on $\mathcal{C}_1$

We have then

$$\mathcal{L}_{\mathbf{Sem}(A)^\kappa}(A) = \big((a + \delta)^* \cdot b \cdot (a + \delta)^* \cdot b\big)^* \cdot (a + \delta)^* \cdot b \cdot (a + \delta)^*$$
$$\mathcal{L}_K(A) = \big((a + 0)^* \cdot b \cdot \langle a^* \rangle_{=1} \cdot b\big)^* \cdot (a + 0)^* \cdot b \cdot (a + \mathbb{R}_{\geq 0})^*$$

where $\langle a^* \rangle_{=1}$ states for the set of all words of $(\mathbb{R}_{\geq 0} \vee a)^*$ of duration 1 (this is the sum of all real labels appearing in the word. c.f. section 4.2). The notation is inspired of timed regular expressions of [9].

We can finally adjust the concept of simulation. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two timed structures, Γ be a control alphabet, $\langle A_1, K_1\rangle \in \mathbb{A}(\mathcal{S}_1, \Gamma)$ and $\langle A_2, K_2\rangle \in \mathbb{A}(\mathcal{S}_2, \Gamma)$. We say that $\langle A_1, K_1\rangle$ simulates $\langle A_2, K_2\rangle$ if A_1 simulates A_2 (c.f. section 3.2) language and determinism are preserved, i.e. $\mathcal{L}_{K_1}(A_1) = \mathcal{L}_{K_2}(A_2)$, and if A_2 is K_2-deterministic then A_1 is K_1-deterministic.

The notion of control we just introduced is very general. For this thesis purpose, we can restrict ourselves to the study of what we call *timed controls*, defined in the next section. Timed controls encompass most of the varied notions of determinism and language we are interested in. However the notion of *untimed language* used especially for timed automata don't fit in the, to be defined, timed controls. It is covered though by the notion of control in the following way.

Definition 4.1.4. *Let Σ be a finite alphabet, $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure, $A = \langle Q, I, T, F\rangle$ an automaton on $\mathcal{S}$.*

A Σ-untimed control is a $\Sigma \uplus \{\epsilon\}$-control, $K_{\mathbf{U}}$, on A such that $K_{\mathbf{U}} = \mathbf{Sem}(A)^\kappa$ for some labeling κ from $T \uplus \mathbb{R}_{\geq 0}$ to $\Sigma \uplus \{\epsilon\}$ verifying for all $d \in \mathbb{R}_{\geq 0}$, $\kappa(d) = \epsilon$ and $\kappa(T) \subseteq \Sigma$.

Example 4.1.2. In the continuity of example 4.1.1, the automaton A on $\mathcal{S}$ the 1-clock structure, can be equipped by an untimed command.

Let $\kappa : T \cup \mathbb{R}_{\geq 0} \to \{a, b, \epsilon\}$ defined as:

- $\kappa((q, v, u, q)) = a$ for all $q \in \{q_1, q_2\}$, $v \in V$, $u \in U$ and $(q, v, u, q) \in T$.
- $\kappa((q, v, u, q')) = b$ for all $q \neq q' \in \{q_1, q_2\}$, $v \in V$, $u \in U$ and $(q, v, u, q') \in T$.
- $\kappa(d) = \epsilon$ for all $d \in \mathbb{R}_{\geq 0}$.

κ is a labeling of $\mathbf{Sem}(A)$ on $\{a, b, \epsilon\}$, and T^κ is a $\{a, b\}$-untimed control on A.

We have then $\mathcal{L}_{\mathbf{Sem}(A)^\kappa}(A) = (a^* \cdot b)^+ \cdot a^*$.

4.2 Timed Controls

Timed controls are a restricted class of controls where we allow the observer to have always access to the precise durations in the runs of the automaton. A discrete transition might be partially or completely invisible from an external observer, though if the observer can see some information about a discrete transition he can be sure it corresponds to exactly one discrete transition of the automaton. We give the formal definition of *timed controls* below, right after a formal definition of the *duration* of a word.

Given a set Γ. The duration of a word of $(\mathbb{R}_{\geq 0} \uplus \Gamma)^*$ is the sum of all real numbers appearing in the words. Formally we define by induction for all Γ the function $\delta_\Gamma : (\mathbb{R}_{\geq 0} \uplus \Gamma)^* \to \mathbb{R}_{\geq 0}$

- $\delta_\Gamma(\epsilon) = 0$
- $\delta_\Gamma(\gamma) = 0$ if $\gamma \in \Gamma$
- $\delta_\Gamma(d) = d$ if $d \in \mathbb{R}_{\geq 0}$
- $\delta_\Gamma(w \cdot w') = \delta(w) + \delta(w')$

For all $w \in (\mathbb{R}_{\geq 0} \uplus \Gamma)^*$, $\delta_\Gamma(w)$ is the duration of w.

A timed control is obtained by a special renaming of the discrete transition. Delay transitions are not renamed, so the observer can see exactly what amount of time has elapsed. Like in a labeling (c.f. remark 4.1.1 and 1.3.4) transitions can be mapped to labels in a fixed finite alphabet Σ – this is formalized by the first item in definition 4.2.1. But contrarily to a labeling,

not all transitions are mapped to labels (which makes the class of timed control not reducible to those definable with a labeling). The transitions which are not mapped to any label are considered *silent*. Those transitions are not observable, meaning concretely that when the observer sees a delay transition, any (finite) number of those transitions may have been taken by the system – this is formalized by the second item of definition 4.2.1. Formally a timed control is characterized by a *partial function* describing how discrete transitions are labeled; silent transitions being those which are not in the domain of this partial function.

Definition 4.2.1. *Let Σ be a finite alphabet, $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure, $A = \langle Q, I, T, F\rangle$ an automaton on $\mathcal{S}$.*

A Σ-timed control *is a labeled transition system, $\mathbf{T} = \langle \mathbf{Conf}(A), \to, \Sigma \uplus \mathbb{R}_{\geq 0}\rangle$ satisfying:*

- *there exists a partial function $\kappa : T \rightharpoonup \Sigma$ such that for all $\sigma \in \Sigma$ and $c, c' \in C_A^2$,*

$$c \xrightarrow{\sigma}_{\mathbf{T}} c' \iff \exists t \in \kappa^{-1}(\sigma), c \xrightarrow{t}_A c'$$

  We write then $T_\epsilon = T \setminus \mathbf{dom}(\kappa)$ and for all $\sigma \in \Sigma$, $T_\sigma = \kappa^{-1}(\sigma)$. Any transition $t \in T_\epsilon$ is called a silent transition.

- *for all $d \in \mathbb{R}_{\geq 0}$ and $c, c' \in C_A^2$,*

$$c \xrightarrow{d}_{\mathbf{T}} c' \iff \exists w \in (T_\epsilon \uplus \mathbb{R}_{\geq 0})^*, \delta_{T_\epsilon}(w) = d \text{ and } c' \in \mathbf{Reach}(\mathbf{Sem}(A), w, c)$$

We verify that a Σ-timed control is a control in the sense of definition 4.1.1.

Proposition 4.2.2. *Let Σ be a finite alphabet, $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ be a timed structure, $A = \langle Q, I, T, F\rangle$ an automaton on $\mathcal{S}$. A* Σ-timed control *is a $\Sigma \uplus \mathbb{R}_{\geq 0}$-control on A.*

Proof.

We make sure below that a timed control is a well-defined control. Let $\mathbf{T}$ be a Σ-timed control. We prove by double inclusion that for all configuration $c_i \in I$, $\mathbf{Reach}(\mathbf{Sem}(A), c_i) = \mathbf{Reach}(\mathbf{T}, c_i)$.

($\subseteq$) Let $c \in \mathbf{Reach}(\mathbf{Sem}(A), c_i)$ with $c_i \in I$. Exists a word $w \in (T \uplus \mathbb{R}_{\geq 0})^*$ such that $c \in \mathbf{Reach}(\mathbf{Sem}(A), w, c_i)$. Let us decompose w into $w_1 \cdot t_1 \cdot w_2 \cdot \dots \cdot w_n \cdot t_n \cdot w_{n+1}$, with $w_1, \dots, w_{n+1} \in (T_\epsilon \uplus \mathbb{R}_{\geq 0})^*$ and $t_1, \dots, t_n \in \mathbf{dom}(\kappa)$. By definition

$$c \in \mathbf{Reach}(\mathbf{T}, \delta_{T_\epsilon}(w_1) \cdot \kappa(t_1) \cdot \delta_{T_\epsilon}(w_2) \cdot \dots \cdot \delta_{T_\epsilon}(w_n) \cdot \kappa(t_n) \cdot \delta_{T_\epsilon}(w_{n+1}), c_i)$$

($\supseteq$) Conversely let $c \in \mathbf{Reach}(\mathbf{T}, c_i)$ with $c_i \in I$. Exists a word $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$ such that $c \in \mathbf{Reach}(\mathbf{T}, w, c_{\text{init}})$. Let us decompose w into $d_1 \cdot \sigma_1 \cdot d_2 \cdot \dots \cdot d_n \cdot \sigma_n \cdot d_{n+1}$, with $d_1, \dots, d_{n+1} \in \mathbb{R}_{\geq 0}^*$ and $\sigma_1, \dots, \sigma_n \in \Sigma$. By definition exists $w_1, \dots, w_{n+1} \in (T_\epsilon \uplus \mathbb{R}_{\geq 0})^*$ and $t_1, \dots, t_n \in \mathbf{dom}(\kappa)$, such that for all $1 \leq i \leq n+1$, $\delta_{T_\epsilon}(w_i) = d_i$, for all $1 \leq i \leq n$, $\kappa(t_i) = \sigma_i$ and

$$c \in \mathbf{Reach}(\mathbf{Sem}(A), w_1 \cdot t_1 \cdot w_2 \cdot \dots w_n \cdot t_n \cdot w_{n+1}, c_i)$$

■

In short, for a fixed alphabet Σ, we write that $\mathbf{T}^\kappa$ is a Σ-timed control meaning that $\mathbf{T}^\kappa$ is a timed control based on the partial function κ (see remark 4.2.1 below about the ambiguity with labeling notation). We also write $\langle A, \mathbf{T}^\kappa\rangle \in \mathbb{A}(\mathcal{S}, \Sigma)$, meaning that $A \in \mathbb{A}(\mathcal{S})$ and $\mathbf{T}^\kappa$ is a Σ-timed control on A.

Remark 4.2.1. In particular if $\mathbf{dom}(\kappa) = T$ then, writing $\kappa' : \mathbb{R}_{\geq 0} \uplus T \to \mathbb{R}_{\geq 0} \uplus \Sigma$ such that κ' coincide with κ and is the identity on $\mathbb{R}_{\geq 0}$, the Σ-timed control $\mathbf{T}^\kappa$ is exactly the $(\mathbb{R}_{\geq 0} \uplus \Sigma)$-renaming, $\mathbf{Sem}(A)^{\kappa'}$, of $\mathbf{Sem}(A)$ by κ'. This justifies the choice of notation, which could seem a little bit confusing at first. In this case we say that $\langle A, \mathbf{T}^\kappa \rangle$ is *without silent transitions*.

Given $\mathbf{T}^\kappa$ being a Σ-timed control over some automaton A, we write for short A *is* κ*-deterministic* instead of A is $\mathbf{T}^\kappa$-deterministic, and $\mathcal{L}_\kappa(A)$ instead of $\mathcal{L}_{T^\kappa}(A)$. We say that π *is a run of* $\langle A, \mathbf{T}^\kappa \rangle$ any time π is a path of $\mathbf{T}^\kappa$ starting in an initial configuration of A and ending in a final configuration of A.

Remark 4.2.2. As it was implicitly done in the proof of definition 4.2.1, given a Σ-timed control κ on a automaton $A = \langle Q, I, T, F \rangle$, we can extend κ on $(T \uplus \mathbb{R}_{\geq 0})^*$ in the following way. If $w \in (T \uplus \mathbb{R}_{\geq 0})^*$ is uniquely decomposed as $w_1 \cdot t_1 \cdot w_2 \cdot \cdots \cdot w_n \cdot t_n \cdot w_{n+1}$, with $w_1, \ldots, w_{n+1} \in (T_\epsilon \uplus \mathbb{R}_{\geq 0})^*$ and $t_1, \ldots, t_n \in \mathbf{dom}(\kappa)$ we define

$$\kappa(w) = \delta_{T_\epsilon}(w_1) \cdot \kappa(t_1) \cdot \delta_{T_\epsilon}(w_2) \cdot \cdots \cdot \delta_{T_\epsilon}(w_n) \cdot \kappa(t_n) \cdot \delta_{T_\epsilon}(w_{n+1})$$

And by definition we know that if $\mathbf{Reach}(\mathbf{Sem}(A), w, c) \neq \emptyset$ then $\mathbf{Reach}(\mathbf{T}^\kappa, \kappa(w), c) \neq \emptyset$.

Example 4.2.1. In the continuity of example 4.1.1, the automaton A on $\mathcal{S}$ the 1-clock structure, can be equipped by different $\{a\}$-timed control in the following ways.

Let $\kappa : T \cup \mathbb{R}_{\geq 0} \to \{a\}$ defined as:

- $\kappa((q, v, \mathbf{id}, q)) = a$ for all $q \in \{q_1, q_2\}$, $v \in V$, $u \in U$ and $(q, v, u, q) \in T$.

κ is a partial function from T to $\{a\}$ so we can define using definition 4.2.1 the timed control T^κ on A. In this case $T_a = \{(q_2, v, \mathbf{id}, q_2), v \in [0, 1]\}$ and $T_\epsilon = \{(q, 0, \mathbf{0}, q), q \in \{q_1, q_2)\} \cup \{(q_2, 1, \mathbf{0}, q_1), (q_1, 0, \mathbf{0}, q_2)\}$

A second example could be $\kappa : T \cup \mathbb{R}_{\geq 0} \to \{a\}$ defined as:

- $\kappa((q, v, \mathbf{id}, q)) = a$ for all $q \in \{q_1, q_2\}$, $v \in V$, $u \in U$ and $(q, v, u, q) \in T$.
- $\kappa((q, v, \mathbf{id}, q')) = b$ for all $q \neq q' \in \{q_1, q_2\}$, $v \in V$, $u \in U$ and $(q, v, u, q) \in T$.

κ is a *total* function from T to $\{a\}$ so we can define using definition 4.2.1 the timed control T^κ on A. In this case $T_a = T$ and $T_\epsilon = \emptyset$. We could easily extend κ following remark 4.2.2 to transform it into a *labeling* (proposition 1.3.4). We say that $\langle A, \mathbf{T}^\kappa \rangle$ is without silent transitions;

The timed control defined below has some importance because it gives to the observer the information about the updates done at all transitions. As we will see in chapter 8, this is all that is enough to always know the values at any point of the run of the automaton, leaving only uncertain the state it is in. It is called for this reason the *full control*.

Definition 4.2.3. *Let Σ be a finite alphabet, $\mathcal{S} = \langle V, \hookrightarrow, U \rangle$ be a timed structure with U a finite set, $A = \langle Q, I, T, F \rangle$ an automaton on $\mathcal{S}$.*

A Σ-full control *is a* $([\Sigma \times U] \uplus \mathbb{R}_{\geq 0})$*-timed control, $\mathbf{T}^\kappa$, on A such that for all $t = (q, v, u, q') \in T$, exists $a \in \Sigma$ such that $\kappa(t) = (a, u)$.*

Example 4.2.2. Again in the continuity of example 4.1.1, the automaton A on $\mathcal{S}$ where $\mathcal{S}$ is the 1-clock structure, can be equipped by a full control. To point out that the control is not just timed but more specifically full, we use $\mathbf{F}$ instead of κ when we define full controls. To see a graphical representation of the automaton A equipped with $\mathbf{T}^\mathbf{F}$ see example 4.3 in section 4.3 at the end of this chapter. Let $\mathbf{F} : T \to \{a, b\}$ defined as:

- $\mathbf{F}((q, v, \mathbf{id}, q)) = (a, \mathbf{id})$ for all $q \in \{q_1, q_2\}$, $v \in V$ and $(q, v, \mathbf{id}, q) \in T$.
- $\mathbf{F}((q, v, \mathbf{0}, q)) = (a, \mathbf{0})$ for all $q \in \{q_1, q_2\}$, $v \in V$ and $(q, v, \mathbf{0}, q) \in T$.
- $\mathbf{F}((q, v, \mathbf{0}, q')) = (b, \mathbf{id})$ for all $q \neq q' \in \{q_1, q_2\}$, $v \in V$ and $(q, v, \mathbf{0}, q') \in T$.

$\mathbf{T}^{\mathbf{F}}$ is a $\{a\}$-full control on A.

We have then

$$\begin{aligned}\mathcal{L}_{\mathbf{F}}(A) \ = \big(((a, \mathbf{0}) \vee 0)^* \cdot (b, \mathbf{0}) \cdot ((a, \mathbf{0}) \vee 0 \vee (a, \mathbf{id}))^* \cdot \langle (a, \mathbf{id})^* \rangle_{=1} \cdot (b, \mathbf{0})\big)^* \cdot \\ ((a, \mathbf{0}) \vee 0)^* \cdot (b, \mathbf{0}) \cdot ((a, \mathbf{0}) \vee 0 \vee (a, \mathbf{id}))^* \cdot \langle (a, \mathbf{id})^* \rangle_{\leq 1} \cdot \mathbb{R}^*_{\geq 0}\end{aligned}$$

where $\langle a^* \rangle_{=1}$ and $\langle a^* \rangle_{\leq 1}$ states for the set of all words of $(\mathbb{R}_{\geq 0} \vee a)^*$ with duration 1 and less than 1 respectively.

4.3 Graph and Bisimulation of Timed Controlled Automata

For automata equipped with timed controls we alter the notion of guards to take in to account what is seen by the observer. Recall that we argued that guards are a way to enforce some regularity on the laws of evolution of the model. Here we are seeking to maintain this regularity from the eyes of the external observer so he too can be able to study our model and maybe compute things about it. Informally we impose now that not only the guards of the automaton respects some guard bases but also the guards obtained by grouping transitions with the same *labeling*.

Definition 4.3.1. *Let $\mathcal{S} = \langle V, \hookrightarrow, U \rangle$ be a timed structure, $A = \langle Q, I, T, F \rangle$ an automaton on $\mathcal{S}$ and $q, q' \in Q$. Let Σ be a finite alphabet, $\mathbf{T}^\kappa$ be a Σ-timed control on A and $\sigma \in \Sigma \uplus \{\epsilon\}$.*

Recall that T_σ stands for $\kappa^{-1}(\sigma)$, and T_ϵ for $T \setminus \mathbf{dom}(\kappa)$ (c.f. definition 4.2.1), We call guard of label σ from q to q' the guard defined by

$$g_\kappa(q, \sigma, q') = \{(v, u) \in V \times U \ \mid \ (q, v, u, q') \in T_\sigma{}^1\}$$

Definition 4.3.2. *Let $\mathcal{S}_G$ be a guarded timed structure, $A = \langle Q, I, T, F \rangle$ an automaton on $\mathcal{S}$. Let Σ be a finite alphabet, $\mathbf{T}^\kappa$ be a Σ-timed control on A and $\sigma \in \Sigma$.*

We say that $\mathbf{T}^\kappa$ is compatible with G if for all $q, q' \in Q$ and $\sigma \in \Sigma \uplus \{\epsilon\}$, $g_\kappa(q, \sigma, q')$ is decomposable in G.

Whenever A represents an automaton on a guarded structure with guard bases G, we say for short that $\mathbf{T}^\kappa$ is a *compatible* Σ-timed control on A, meaning that it is compatible with G.

Example 4.3.1. Let Σ be a finite alphabet, $M \in \mathbb{N}$ and $n \in \mathbb{N}_{>0}$.

We define $\Sigma\mathbf{TA}^n_M$ to be the set of all *controlled timed automata* $\langle A, \mathbf{T}^\kappa \rangle$ with $A \in \mathbf{TA}^n_M$ and $\mathbf{T}^\kappa$ a compatible Σ-timed control on A. See example 3.1.1 for the definition of timed automata.

Example 4.3.2. Let Σ be a finite alphabet, $M \in \mathbb{N}$ and C a finite set

We define $\Sigma\mathbf{TA}^C_M$ to be the set of all *controlled enhanced timed automata* $\langle A, \mathbf{T}^\kappa \rangle$ with $A \in \mathbf{TA}^C_M$ and $\mathbf{T}^\kappa$ a compatible Σ-timed control on A. See example 3.1.2 for the definition of enhanced timed automata.

[1]

The following proposition exposes the link between automata guards and control guards.

Proposition 4.3.3. *Let $\mathcal{S}_G$ be a guarded timed structure, $A = \langle Q, I, T, F\rangle$ an automaton on $\mathcal{S}_G$. Let Σ be a finite alphabet, $\mathbf{T}^\kappa$ be a Σ-timed control on A.*

Suppose $\mathbf{T}^\kappa$ is compatible with G ; then A is compatible with G.

Proof.

Just notice that since $T = \bigcup_{\sigma\in\Sigma\uplus\{\epsilon\}} T_\sigma$, for all $q, q' \in Q$,

$$\begin{aligned} g_A(q,q') &= \{(u,v) \in V \times U \mid (q,v,u,q') \in T\} \\ &= \bigcup_{\sigma\in\Sigma\uplus\{\epsilon\}} \{(v,u) \in V \times U \mid (q,v,u,q') \in T_\sigma\} \\ &= \bigcup_{\sigma\in\Sigma\uplus\{\epsilon\}} g_\kappa(q,\sigma,q') \end{aligned}$$

Since $g_\kappa(q,\sigma,q')$ is decomposable in G for all $\sigma \in \Sigma \uplus \{\epsilon\}$ then $g_A(q,q')$ is decomposable in G. ■

We can, therefore, define a canonical way of representing an automaton on a guarded timed structure equipped with a compatible timed control, in a very similar way as the definition of the graph of an automaton.

Fix a guarded timed structure $\mathcal{S}_G$ and let $A = \langle Q, I, T, F\rangle \in \mathbb{A}(\mathcal{S}_G)$. Fix Σ a finite alphabet and $\mathbf{T}^\kappa$ a compatible Σ-timed control on A. Formally we define the *graph* of $\langle A, \mathbf{T}^\kappa\rangle$ as the labeled directed multi-graph $\mathcal{G}_{A,\kappa} = \langle Q, E_{A,\kappa}, G\rangle$ with Q the set of nodes, G the set of labels, and $E_{A,\kappa} \subseteq Q \times \Sigma \times G \times Q$ the set of labeled directed edges, defined as

$$E_{A,\kappa} = \{(q,\sigma,\nu,\mu,q') \in Q \times \Sigma \times G \times Q \mid (\nu,\mu) \in g_\kappa(q,\sigma,q')\}$$

Because $\mathbf{T}^\kappa$ is compatible with G, we have that $\mathcal{G}_{A,\kappa}$ is finite.

Each edge $(q,\sigma,\nu,\mu,q') \in E_A$ has to be understood as: *from every configuration (q,v) with $v \in \nu$ it is possible to take a transition toward q'* which is labeled by σ in the control $\mathbf{T}^\kappa$ *and any update within μ can be (non-deterministically) applied*.

Example 4.3.3. Still in the continuity of example 4.1.1, recall that the automaton A on $\mathcal{S}$ where $\mathcal{S}$ is the 1-clock structure and G is as described above, has been equipped with a full control $\mathbf{T}^\mathbf{F}$ defined in example 4.2.2.

The graph of the controlled automaton $\langle A, \mathbf{T}^\mathbf{F}\rangle$ is given on figure 4.3

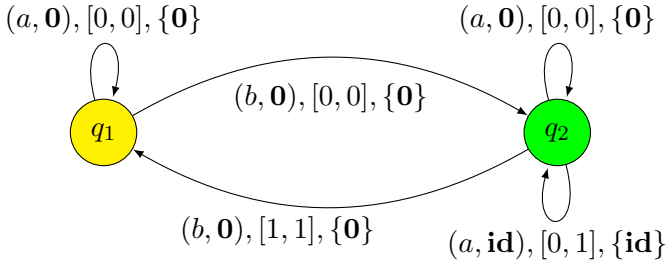


Figure 4.3: Graph of the controlled automaton $\langle A, \mathbf{T}^\mathbf{F}\rangle$

Example 4.3.4. On figure 4.4 we can find the representation, in our formalism, of the non-determinizable timed automaton (in the classical sense [4]), named $\mathcal{B}$, introduced in [4].

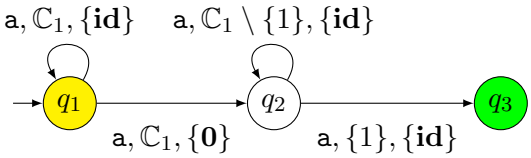


Figure 4.4: $\mathcal{B}$, a non-determinizable TA

We already introduced a concept of simulation for controlled automata in section 4.1. We end part I below with the last extension of proposition 3.2.1. Recall that this proposition states that if we are given a *pre-functional bisimulation* from a set of state Q_1 and a timed structure S_1 to an automaton A_2 on a timed structure S_2, then there exists an automaton A_1 on S_1 with, as set of states, Q_1 which simulates A_2 and we can construct it automatically from the pre-functional bisimulation expression. If now S_1 is a guarded timed structure, we already provided an extension of the proposition to construct a simulating automaton compatible with S_1. Next, we suppose that A_2 is equipped with a timed control $\mathbf{T}^{\kappa_2}$. The extension of proposition 3.2.1 we prove below, help us avoid adapting A_1, defining a timed control κ_1 and proving by hand that $\langle A_1, \mathbf{T}^{\kappa_1}\rangle$ is compatible with S_1 and that determinism is preserved. By adding some additional property to be respected by the pre-functional bisimulation we make sure that $\langle A_1, \kappa_1\rangle$ respects them automatically by construction from the expression of such pre-functional bisimulation.

Let Σ be a finite alphabet. Let $\mathcal{S}_2 = \langle V_2, \hookrightarrow_2, U_2\rangle$ be a timed structure and $A_2 = \langle Q_2, I_2, T_2, F_2\rangle$ be an automaton over $\mathcal{S}_2$. Let $\mathbf{T}^{\kappa_2}$ be a Σ-timed control on A_2. Let $\mathcal{S}_{G_1} = \langle V_1, \hookrightarrow_1, U_1, G_1\rangle$ be a guarded timed structure, Q_1 be a finite set and $\Xi = (\zeta, \phi, \psi)$ be a pre-functional bisimulation from $Q_1, \mathcal{S}_1$ to A_2, with $\phi = (\phi_{q_1})_{q_1 \in Q_1}$ and $\psi = (\psi_{q_1,q_1'})_{q_1,q_1' \in Q_1}$.

For all $q_1, q_1' \in Q_1$ and $\sigma \in \Sigma \uplus \{\epsilon\}$ we define

$$\mathbf{dom}_\sigma(\psi_{q_1,q_1'}) = \{(v_1,u_1) \in \mathbf{dom}(\psi_{q_1,q_1'}) \mid (\phi_{q_1}(v_1), \psi_{q_1,q_1'}(v_1,u_1)) \in g_{\kappa_2}(\zeta(q_1), \sigma, \zeta(q_1'))\}$$

Also if Ξ satisfies:

- for all $q_1, q_1' \in Q_1$ and $\sigma \in \Sigma \uplus \{\epsilon\}$:

  $$\mathbf{dom}_\sigma(\psi_{q_1,q_1'}) \text{ is decomposable on } G_1$$

  we say that Ξ is a Σ, G_1-*compatible pre-functional bisimulation* from $Q_1, \mathcal{S}_1$ to A_2.

- for all $q_1, v_1 \in Q_1 \times V_1$ and $\sigma \in \Sigma$, exists at most one pair $u_1, q_1' \in U_1 \times Q_1$ such that,

  $$(v_1,u_1) \in \mathbf{dom}_\sigma(\psi_{q_1,q_1'})$$

  and for all $q_1, q_1' \in Q_1$

  $$\mathbf{dom}_\epsilon(\psi_{q_1,q_1'}) = \emptyset$$

  we say that Ξ is a Σ-*deterministic pre-functional bisimulation* from $Q_1, \mathcal{S}_1$ to A_2.

Let $A_1 = \Xi^{-1}(A_2) = \langle Q_1, I_1, T_1, F_1\rangle$.
Let also

$$\begin{array}{rccll} \kappa_1 : & T_1 & \rightharpoonup & \Sigma & \\ & (q_1, v_1, u_1, q_1') & \mapsto & \sigma & \text{if } (v_1,u_1) \in \mathbf{dom}_\sigma(\psi_{q_1,q_1'}) \end{array}$$

Then $\mathbf{T}^{\kappa_1}$ is a Σ-timed control on A_1. κ_1 is then called the inverse image of κ_2 by Ξ and is written $\Xi^{-1}(\kappa_2)$.

Proposition 4.3.4. *Let Σ be a finite alphabet. Let $\mathcal{S}_2 = \langle V_2, \hookrightarrow_2, U_2\rangle$ be a timed structure and $A_2 = \langle Q_2, I_2, T_2, F_2\rangle$ be an automaton over $\mathcal{S}_2$. Let $\mathbf{T}^{\kappa_2}$ be a Σ-timed control on A_2. Let $\mathcal{S}_G = \langle V_1, \hookrightarrow_1, U_1, G\rangle$ be a guarded timed structure, Q_1 be a finite set and $\Xi = (\zeta, \phi, \psi)$ be a pre-functional bisimulation from $Q_1, \mathcal{S}_1$ to A_2, with $\phi = (\phi_{q_1})_{q_1 \in Q_1}$ and $\psi = (\psi_{q_1,q_1'})_{q_1,q_1' \in Q_1}$.*

If Ξ is a Σ, G-compatible pre-functional bisimulation from $Q_1, \mathcal{S}_1$ to A_2 then $T^{\Xi^{-1}(\kappa_2)}$ is compatible with G.

Proof.

Let $A_1 = \Xi^{-1}(A_2) = \langle Q_1, I_1, T_1, F_1\rangle$ and $\kappa_1 = \Xi^{-1}(\kappa_2)$.

Recall that for all $q_1, q_1' \in Q_1$ and for all $\sigma \in \Sigma$,

$$g_{\kappa_1}(q_1, \sigma, q_1') = \{(v_1, u_1) \in V_1 \times U_1 \mid (q_1, v_1, u_1, q_1') \in \kappa_1^{-1}(\sigma)\}$$

By definition this means that $g_{\kappa_1}(q_1, \sigma, q_1') = \mathbf{dom}_\sigma(\psi_{q_1,q_1'})$.

Also for ϵ, $g_{\kappa_1}(q_1, \epsilon, q_1') = \{(v_1, u_1) \in V_1 \times U_1 \mid (q_1, v_1, u_1, q_1') \in T_1 \setminus \mathbf{dom}(\kappa_1)\}$. Since $\mathbf{dom}(\psi_{q_1,q_1'}) = \bigcup_{\sigma \in \Sigma \uplus \{\epsilon\}} \mathbf{dom}_\sigma(\psi_{q_1,q_1'})$, by definition of $\mathbf{dom}_\sigma(\psi_{q_1,q_1'})$ and of a pre-functional bisimulation, we get that $g_{\kappa_1}(q_1, \epsilon, q_1') = \mathbf{dom}_\epsilon(\psi_{q_1,q_1'})$.

In all cases, for all $\sigma \in \Sigma \uplus \{\epsilon\}$, $g_{\kappa_1}(q_1, \sigma, q_1') = \mathbf{dom}_\sigma(\psi_{q_1,q_1'})$ is decomposable on G by hypothesis. ■

Proposition 4.3.5. *Let Σ be a finite alphabet. Let $\mathcal{S}_2 = \langle V_2, \hookrightarrow_2, U_2\rangle$ be a timed structure and $A_2 = \langle Q_2, I_2, T_2, F_2\rangle$ be an automaton over $\mathcal{S}_2$. Let $\mathbf{T}^{\kappa_2}$ be a Σ-timed control on A_2. Let $\mathcal{S}_G = \langle V_1, \hookrightarrow_1, U_1, G\rangle$ be a guarded timed structure, Q_1 be a finite set and $\Xi = (\zeta, \phi, \psi)$ be a pre-functional bisimulation from $Q_1, \mathcal{S}_1$ to A_2, with $\phi = (\phi_{q_1})_{q_1 \in Q_1}$ and $\psi = (\psi_{q_1,q_1'})_{q_1,q_1' \in Q_1}$.*

If Ξ is a Σ-deterministic pre-functional bisimulation from $Q_1, \mathcal{S}_1$ to A_2 and $\langle V_1, \hookrightarrow_1\rangle$ is deterministic*, then $\Xi^{-1}(A_2)$ is $T^{\Xi^{-1}(\kappa_2)}$ deterministic.*

Proof.

Let $A_1 = \Xi^{-1}(A_2) = \langle Q_1, I_1, T_1, F_1\rangle$ and $\kappa_1 = \Xi^{-1}(\kappa_2)$.

Let $l \in \mathbb{R}_{\geq 0} \cup \Sigma$ and let $(q_1, v_1) \in C_{A_1}$ be a configuration in $\mathbf{T}^{\kappa_1}$. We prove that exists at most one configuration $(q_1', v_1') \in C_{A_1}$ such that $(q_1, v_1) \xrightarrow{l} (q_1', v_1')$ in $\mathbf{T}^{\kappa_1}$.

Suppose $l = d \in \mathbb{R}_{\geq 0}$ and suppose $(q_1, v_1) \xrightarrow{d} (q_1', v_1')$. By definition of a Σ-timed control, $\exists w \in (T_{1/\epsilon} \uplus \mathbb{R}_{\geq 0})^*, \delta_{T_\epsilon}(w) = d$ and $(q_1', v_1') \in \mathbf{Reach}(\mathbf{Sem}(A), w, (q_1, v_1))$. But by hypothesis $T_\epsilon = \{(q, v, u, q') \in T_1 \mid (v, u) \in \mathbf{dom}_\epsilon(\psi_{q,q'})\}$ is empty. So this means that $w \in (\mathbb{R}_{\geq 0})^*$ which means, by definition of a $\mathbf{Sem}(A)$ and of a timed domain that $q_1' = q_1$ and that $v_1' = v_1 \oplus d$.

Suppose now $l = \sigma \in \Sigma$ and suppose $(q_1, v_1) \xrightarrow{\sigma} (q_1', v_1')$, then $\exists t \in \kappa_1^{-1}(\sigma), (q_1, v_1) \xrightarrow{t}_A (q_1', v_1')$. This means that exists $u_1 \in U_1$ such that $t = (q_1, v_1, u_1, q_1')$ and $v_1' = u_1(v_1)$, therefore that $(v_1, u_1) \in \mathbf{dom}_\sigma(\psi_{q_1,q_1'})$. We know that there exists at most one pair $(u_1, q_1') \in U_1 \times Q_1$ verifying that property, this proves the unicity of (q_1', v_1'). ■

This ends the definition of the model and the introduction of the theoretical tools we need for the rest of this thesis. Most important ones being :

- Guarded timed structures (chapter 2) to model quantitative variables, their evolution, updates and level of regularity with regard to those updates - $\langle V, \hookrightarrow, U, G\rangle$ where
  - $\langle V, \hookrightarrow, U\rangle$ is a timed structure which models both quantitative variables evolution (through the timed domain $\langle V, \hookrightarrow\rangle$) and quantitative variables updates (through a set of update U on V).

  - G is a guard basis which enforces some regularity on the behavior of the quantitative variable with regards to updates.

- Automata on (guarded) timed structure (chapter 3) to model discrete states and discrete actions in the model, build on top of the timed structure - $\langle Q, I, T, F \rangle$.
- Timed control (chapter 4) to allow generic modelisation of indistinguishable behavior, unobservable actions, and non-determinism - $\mathbf{T}^{\kappa}$ with κ a particular partial function on labels.
- Pre-functional bisimulation between automata on timed structures (chapter 3 and chapter 4)) to construct new equivalent automata from existing ones, on different timed structures or/and label spaces.

In the next part, we discuss the questions of determinization for our model. Notice that in the framework we just introduced, determinism is not an absolute notion but is a concept which depends on the control we equip our automaton with. That's why in some sense we will be able to prove in the next part that every automaton on any timed structure (without silent transition) is determinizable ... and in the meantime, this statement's lack of rigor could imply lots of misunderstandings ...

Part II

Determinization of Timed Systems

Chapter 5

Timed Control Determinization

A Generalized Powerset Construction

Determinization is a common problem addressed in computer science. Non-deterministic transitions are a useful tool used in modeling or theoretical proofs, but when it comes to implementation it becomes an obstacle. Therefore it is usually asked for a formal model class to be closed by determinization, meaning informally that every non-deterministic model of such class is equivalent to a deterministic one (in which sense it is equivalent, it is one of the issues of the problem).

In the most general setting, we know that every model expressible as a Turing machine is equivalent to deterministic Turing Machine [41]. However, it is often not satisfactory to have to go back to Turing machine to obtain determinism. In most cases, it is wanted that an equivalent model within a circumscribed class exists within this same class, as it is the case for finite automata [41]. In quantitative models, it is rare though that a whole class is closed by determinization. It is indeed not the case for pushdown automata [38], weighted automata [39] [43] or timed automata [4]. One of the underlying reasons for the problem is that the determinization process needs an exponential increase in the size of the model. When the model is finite it is not an intractable obstacle, but when the model is of infinite size, the exponential increase requires a configuration space of higher cardinality.

Solutions to circumvent this problem are usually found by exposing subclasses of models, either stable by determinization, either determinizable within the class – see the discussion in chapter 7. We exploit the opposite idea to propose a solution to this problem in the context of automata with timed domains: intuitively we propose for some classes of automata on timed domains a determinization process toward a *super class* of automata on timed domain. As we said it is always possible to do so if we go all the way back to Turing Machines, but our result's interest lies in the good properties verified by this superclass: simple representation and computability. Two fundamental properties to allow efficiency and complexity estimation in the context of diagnosis. The superclass considered will be constructed by expanding the timed structure using a kind of "power construction". Here the E-marking construction presented page 22 will be useful.

It remains, however, to be specified what does *equivalent* stands for in our context. Determinization classically focuses on language preservation, and this is what interests us for our purpose to build a diagnoser. In the case of timed automata, this is highlighted by the difference between untimed language, timed language and event-clock automata. Constructing a deterministic timed automaton recognizing the same untimed language than another one can be done using the region abstraction [4]. Now consider a timed automaton where each transition has a label which hardcode its update, then it can be determinized with a simple powerset

construction on discrete states (c.f. chapter 8). Event-clock automata could be viewed as timed automata with such labels. However, preserving the equivalence of timed languages cannot always be done (and it can't be decided if it can be done) [4]. In our formalism, those differences correspond to different choices of controls and it was made clear that languages depend on the control we equip the automaton we wish to determinize. We will see in chapter 8 that the choice of control has great importance, determinization of timed automata not being so difficult if they are equipped with full controls. In this chapter, we focus on determinization of automata on timed structures equipped with a Σ-*timed controls without silent transitions*, Σ being fixed. We prove that for each of them there exists a deterministic automaton on a *powerset construction of the timed structure*, also equipped by a Σ-timed structure without silent transitions. This determinized automaton, called powerset automaton, is at the basis of our diagnoser construction in chapter 9 and 11.

5.1 Determinization of Timed Controlled Automata on Timed Systems

First we formalize below what *determinization* means to us. Following the discussion in the introduction, in this thesis *determinizability* implies the construction of a new controlled automaton recognizing the same language, but does not imply that the new controlled automaton has to be on the same time structure. Recall that $\langle A, \mathbf{T}^\kappa \rangle \in \mathbb{A}(\mathcal{S}, \Sigma)$ means that A is an automaton on the timed structure $\mathcal{S}$ equipped with a Σ-timed control $\mathbf{T}^\kappa$ (c.f. page 38).

Definition 5.1.1. *Let Σ a finite alphabet, $\mathcal{S}$ a timed structure and $\langle A, \mathbf{T}^\kappa \rangle \in \mathbb{A}(\mathcal{S}, \Sigma)$.*

We say that $\langle A, \mathbf{T}^\kappa \rangle$ is Σ-determinizable if and only if exist a timed structure S' and $\langle A', \mathbf{T}^{\kappa'} \rangle \in \mathbb{A}(S', \Sigma)$ such that A' is κ'-deterministic, without silent transitions and $\mathcal{L}_\kappa(A) = \mathcal{L}_{\kappa'}(A')$.

$\langle A', \mathbf{T}^{\kappa'} \rangle$ is called a Σ-determinized automaton of $\langle A, \mathbf{T}^\kappa \rangle$.

To solve the determinization problem we introduce below a kind of *powerset construction* valid for any kind of controlled automata on any timed structure.

Fix Σ a finite alphabet, $\mathcal{S} = \langle V, \hookrightarrow, U \rangle$ a timed structure, $A = \langle Q, I, T, F \rangle$ an automaton on $\mathcal{S}$ and $\mathbf{T}^\kappa$ a Σ-timed control on A. We construct *the powerset automaton of* $\langle A, \mathbf{T}^\kappa \rangle$ on a timed structure very close to the Q-markings timed structure defined page 22. Recall that in this context we defined $\mathbf{M}_Q V = \mathcal{P}(V)^Q$.

For all $\gamma \in \Sigma \uplus \mathbb{R}_{\geq 0}$, let $\mathbf{U}_\gamma : \mathbf{M}_Q V \to \mathbf{M}_Q V$ defined for all $\boldsymbol{\nu} \in \mathbf{M}_Q V$ and for all $q' \in Q$ as

$$\mathbf{U}_\gamma(\boldsymbol{\nu})(q') = \{v' \in V \mid \exists q \in Q, \exists v \in \boldsymbol{\nu}(q),\ (q, v) \xrightarrow{\gamma}_{\mathbf{T}^\kappa} (q', v')\}$$

For all $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$ such that $w = \gamma_1 \cdot \cdots \cdot \gamma_n$, with $n \in \mathbb{N}$ and for all $1 \leq i \leq n$, $\gamma_i \in \Sigma \uplus \mathbb{R}_{\geq 0}$, we write $\mathbf{U}_w = \mathbf{U}_{\gamma_n} \circ \cdots \circ \mathbf{U}_{\gamma_1}$.

We prove below that $\mathbf{U}_\sigma \in \mathbf{M}_Q U$ for all $\sigma \in \Sigma$. Recall that $\mathbf{M}_Q U$ is the set of updates obtained in the Q-marking construction. Each update of $\mathbf{M}_Q U$ is of the form $\mathbf{U}_\mathbf{g}$ with $\mathbf{g} \in \mathcal{G}(\mathcal{S})^{E \times E}$ a guard function. $\mathbf{U}_\mathbf{g}$ maps a marking $\boldsymbol{\nu} \in \mathbf{M}_Q V$ to another marking $\boldsymbol{\nu}' \in \mathbf{M}_Q V$ which can be intuitively obtained by applying all transitions derived from $\mathbf{g}$ on $\boldsymbol{\nu}$ (for formal definition see page 22) Let for all $\sigma \in \Sigma$, $\mathbf{g}_\sigma : Q^2 \to \mathcal{G}(\mathcal{S})$ defined as $\mathbf{g}_\sigma(q, q') = g_\kappa(q, \sigma, q')$ for all $q, q' \in Q$.

Lemma 5.1.2. *For all $\sigma \in \Sigma$, $\mathbf{U}_\sigma = \mathbf{U}_{\mathbf{g}_\sigma}$.*

Proof.

Let $\boldsymbol{\nu} \in \mathbf{M}_Q V$ and for all $q' \in Q$,
Recall (p. 22) that $\mathbf{U}_{\mathbf{g}_\sigma}(\boldsymbol{\nu})(q') = \{u(v) \in V \mid \exists q \in Q, v \in \boldsymbol{\nu}(q)$ and $(v, u) \in \mathbf{g}_\sigma(q, q')\}$

$$\begin{aligned}
\mathbf{U}_\sigma(\boldsymbol{\nu})(q') &= \{v' \in V \mid \exists q \in Q, \exists v \in \boldsymbol{\nu}(q),\ (q, v) \xrightarrow{\sigma}_{\mathbf{T}^\kappa} (q', v')\} \\
&= \{v' \in V \mid \exists t \in \kappa^{-1}(\sigma), \exists q \in Q, \exists v \in \boldsymbol{\nu}(q),\ (q, v) \xrightarrow{t}_A (q', v')\} \\
&= \{u(v) \in V \mid \exists q \in Q, v \in \boldsymbol{\nu}(q) \text{ and } (q, v, u, q') \in \kappa^{-1}(\sigma)\} \\
&= \{u(v) \in V \mid \exists q \in Q, v \in \boldsymbol{\nu}(q) \text{ and } (v, u) \in g_\kappa(q, \sigma, q')\} \\
&= \{u(v) \in V \mid \exists q \in Q, v \in \boldsymbol{\nu}(q) \text{ and } (v, u) \in \mathbf{g}_\sigma(q, q')\} \\
&= \mathbf{U}_{\mathbf{g}_\sigma}(\boldsymbol{\nu})(q')
\end{aligned}$$

■

For all $d \in \mathbb{R}_{\geq 0}$, we define then

$$\overset{d}{\hookrightarrow}_{A,\kappa} = \{(\boldsymbol{\nu}, \mathbf{U}_d(\boldsymbol{\nu})), \boldsymbol{\nu} \in \mathbf{M}_Q V\}$$

and the timed structure:

$$\mathbf{D}_{A,\kappa} S = \langle \mathbf{M}_Q V, \hookrightarrow_{A,\kappa}, \mathbf{M}_Q U \rangle$$

This timed structure is heavily dependent on A. Values are markings whose support is the set of states of A and updates simulate simultaneous applications of all transitions of A for a given label. Informally, the result of the application of an update on a marking is then a new marking where each value in any state is obtained by following a transition of A from a state and a value in the original marking. It means that every state in the support of a marking obtained by successive updates is reachable in A following the same run. However, the converse is not true since the support may omit some states reachable through a sequence of silent transitions directly following the last visible action.

We introduce then a new function $\mathbf{supp}_\epsilon : \mathbf{M}_Q V \to \mathcal{P}(Q)$ mapping each marking to the set of states which are mapped to the empty set on this marking but will be mapped to a non-empty set sometime in the future. This function, even if it doesn't tell us exactly when each state is reached, computes all the possible reachable states through silent transitions. Hence the information of $\mathbf{supp}$ and $\mathbf{supp}_\epsilon$ obtained just after application of an update covers all possible reachable state of A following some run. This is useful in the context of diagnosis, where we wish to know if a faulty state can be reached without having to compute again the new support each time a delay transition is taken.

Formally $\mathbf{supp}_\epsilon$ is defined for each $\boldsymbol{\nu} \in \mathbf{M}_Q V$ as

$$\mathbf{supp}_\epsilon(\boldsymbol{\nu}) = \{q \in Q \setminus \mathbf{supp}(\boldsymbol{\nu}) \mid \exists d \in \mathbb{R}_{>0}, q \in \mathbf{supp}(\mathbf{U}_d(\boldsymbol{\nu}))\}$$

We define $\mathbf{D}_\kappa A = \langle \mathcal{P}(Q) \times \mathcal{P}(Q), \{\mathbf{D}_\kappa c_I\}, \mathbf{D}_\kappa T, \mathbf{D}_\kappa F \rangle$ on $\mathbf{D}_{A,\kappa} S$ with:

- $\mathbf{D}_\kappa c_I = ([\mathbf{supp}(\boldsymbol{\nu}_I), \mathbf{supp}_\epsilon(\boldsymbol{\nu}_I)], \boldsymbol{\nu}_I)$
  with $\boldsymbol{\nu}_I \in \mathbf{M}_Q V$ such that for all $(q, v) \in Q \times V$, $v \in \boldsymbol{\nu}_I(q) \iff (q, v) \in I$.
- $\mathbf{D}_\kappa F = \{[\rho, \rho'] \in \mathcal{P}(Q)^2 \mid \rho \cap F \neq \emptyset\}$
- $\mathbf{D}_\kappa T = \{([\rho, \rho'], \boldsymbol{\nu}, \mathbf{U}_\sigma, [\mathbf{supp}(\mathbf{U}_\sigma(\boldsymbol{\nu})), \mathbf{supp}_\epsilon(\mathbf{U}_\sigma(\boldsymbol{\nu}))]), \boldsymbol{\nu} \in \mathbf{M}_Q V, \sigma \in \Sigma, \rho, \rho' \in \mathcal{P}(Q)\}$

We also consider the Σ-timed control $\mathbf{T}^{\mathbf{D}\kappa}$ defined as for all $R_1, R_2 \in \mathcal{P}(Q)^2, \boldsymbol{\nu} \in \mathbf{M}_Q V$ and $\sigma \in \Sigma$, $\mathbf{D}\kappa((R_1, \boldsymbol{\nu}, \mathbf{U}_\sigma, R_2)) = \sigma$.

Definition 5.1.3. *$\langle \mathbf{D}_\kappa A, \mathbf{T}^{\mathbf{D}\kappa} \rangle$ constructed above is called the powerset automaton of $\langle A, \mathbf{T}^\kappa \rangle$.*

The powerset automaton of $\langle A, \mathbf{T}^\kappa \rangle$ has some important good properties described in the next proposition:

Proposition 5.1.4. *Let Σ be a finite alphabet, $\mathcal{S}$ a timed structure and $\langle A, \mathbf{T}^\kappa \rangle \in \mathbb{A}(\mathcal{S}, \Sigma)$.*

(1) $\mathbf{D}_\kappa A$ is $\mathbf{T}^{\mathbf{D}_\kappa}$-deterministic.

(2) $\mathbf{D}_\kappa A$ is $\mathbf{T}^{\mathbf{D}_\kappa}$-complete.

(3) $\langle \mathbf{D}_\kappa A, \mathbf{T}^{\mathbf{D}_\kappa} \rangle$ is without silent transitions (c.f. remark 4.2.1).

(4) For all $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$ and $(q, v) \in Q \times V$,

$$[\, q \in \mathbf{supp}(\mathbf{U}_w(\mathbf{D}_\kappa c_I)) \text{ and } v \in \mathbf{U}_w(\mathbf{D}_\kappa c_I)(q) \,] \iff \exists c_i \in I, (q, v) \in \mathbf{Reach}(\mathbf{T}^\kappa, w, c_i)$$

Proof.

Let's write $A = \langle Q, I, T, F\rangle$ and recall that $\mathbf{D}_\kappa c_I = ((\mathbf{supp}(\boldsymbol{\nu}_I), \mathbf{supp}_\epsilon(\boldsymbol{\nu}_I)), \boldsymbol{\nu}_I)$

(1) Let $(R, \boldsymbol{\nu})$ and $(R', \boldsymbol{\nu}')$ in $\mathcal{P}(Q)^2 \times \mathbf{M}_Q V$.
Let $d \in \mathbb{R}_{\geq 0}$, $(R, \boldsymbol{\nu}) \xrightarrow{d}_{\mathbf{T}^{\mathbf{D}\kappa}} (R', \boldsymbol{\nu}')$ if and only if $R' = R$ and $\boldsymbol{\nu}' = \mathbf{U}_d(\boldsymbol{\nu})$.
Let $\sigma \in \Sigma$, $(R, \boldsymbol{\nu}) \xrightarrow{\sigma}_{\mathbf{T}^{\mathbf{D}\kappa}} (R', \boldsymbol{\nu}')$ if and only if $\boldsymbol{\nu}' = \mathbf{U}_\sigma(\boldsymbol{\nu})$ and $R' = \mathbf{supp}(\boldsymbol{\nu}')$.
This proves that for all $c \in C_{\mathbf{D}_\kappa A}$, for all $\gamma \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$, there is at most one configuration c' such that $c \xrightarrow{\gamma}_{\mathbf{T}^{\mathbf{D}\kappa}} c'$.
$\mathbf{D}_\kappa A$ is then $\mathbf{D}\kappa$-deterministic.

(2) By design $\mathbf{D}_\kappa A$ is $\mathbf{D}\kappa$-complete.

(3) By definition $\langle \mathbf{D}_\kappa A, \mathbf{T}^{\mathbf{D}_\kappa}\rangle$ is without silent transitions.

(4) We write for all $w \in (\Sigma \times \mathbb{R}_{\geq 0})^*$, $\rho_w = \mathbf{supp}(\mathbf{U}_w(\mathbf{D}c_I))$ and $\boldsymbol{\nu}_w = \mathbf{U}_w(\mathbf{D}c_I)(q)$
We make the proof by induction on the length of w.
- Initialization is trivial by definition of $\boldsymbol{\nu}_I$.
- Let $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^{n+1}$, $w' \in (\Sigma \uplus \mathbb{R}_{\geq 0})^n$ and $\gamma \in (\Sigma \uplus \mathbb{R}_{\geq 0})$ such that $w = w' \cdot \gamma$.
  Let $(q, v) \in Q \times V$.

$$
\begin{aligned}
q \in \rho_w \text{ and } v \in \boldsymbol{\nu}_w(q) &\iff v \in \mathbf{U}_\gamma(\boldsymbol{\nu}_{w'})(q) \\
&\iff \exists q' \in Q, \exists v' \in \boldsymbol{\nu}_{w'}(q'), (q', v') \xrightarrow{\gamma}_{\mathbf{T}^\kappa} (q, v) \\
&\iff \exists q' \in \rho_{w'}, \exists v' \in \boldsymbol{\nu}_{w'}(q'), (q', v') \xrightarrow{\gamma}_{\mathbf{T}^\kappa} (q, v) \\
&\iff \exists c_i \in I, \exists (q', v') \in \mathbf{Reach}(\mathbf{T}^\kappa, w', c_i), \\
&\qquad (q', v') \xrightarrow{\gamma}_{\mathbf{T}^\kappa} (q, v) \quad \text{by induction hypothesis.} \\
&\iff \exists c_i \in I, (q, v) \in \mathbf{Reach}(\mathbf{T}^\kappa, w, c_i)
\end{aligned}
$$

This ends the induction. ■

However, those properties are not enough to be able to conclude that $\mathbf{D}_\kappa A$ would be a Σ-determinized of A. The problem lies in the existence of *silent transitions* in the original automaton. In fact, even if through the item (4) of property 5.1.4 just above, we can see that the powerset automaton simulates exactly the behavior of the original automaton, not all accepting runs of A correspond to accepting runs of $\mathbf{D}_\kappa A$. According to the definition of $\mathbf{D}_\kappa F$, an accepting run of A which ends by a silent transition departing from a non-final state may give a non-accepting run of $\mathbf{D}_\kappa A$. For example, the simple automaton which has to wait exactly one time unit and make a silent transition to a final state (figure 5.1), recognizes the language $\mathcal{L}_1 = \{w \in \mathbb{R}^* \mid \delta(w) \geq 1\}$ of all words of $\mathbb{R}_{>0}$ with duration greater than 1; but its powerset automaton recognizes the empty language. Indeed the initial configuration of the powerset automaton is $\langle q_i \mapsto \{0\}, q_f \mapsto \emptyset\rangle$, hence its initial state is $[\{q_i\}, \{q_f\}]$ which is not final. And since there is no visible transition in the original automaton there is no possible updates to do and therefore no transition in the powerset automaton exiting $[\{q_i\}, \{q_f\}]$.

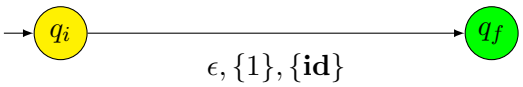


Figure 5.1: A timed automata recognizing $\mathcal{L}_1$

With the introduction of $\mathbf{supp}_\epsilon$ we have the information that the final state is reachable with a silent run, however it is not enough to make it a final state since we don't know when exactly the final state will be reachable (and we can't hard-code it in the state since it could depend on the previous date of arrival in the said state). We give below three solutions to make this powerset automaton directly a Σ-determinized automaton of A.

- We could change the definition of Σ-determinization to allow silent transitions.
- We could change the model and allow some possible changes of discrete states by the simple passage of time (which is close to the original idea of silent transitions).
- Finally we could forbid accepting runs ending by a silent transition, in which case the results of this section still holds.

This last solution circumscribes the difficulty posed by silent transitions. A difficulty which we conjecture to prevent any general construction of a Σ-determinized automaton for automata on timed domain with silent transitions.

However, the powerset automaton is still a great tool to simulate the original automaton with or without silent transitions, which is interesting in the context of diagnosis as we'll see in part III. Also, if we consider now the determinization problem on automata *without silent transitions*, the powerset automaton becomes the Σ-determinized automaton. Indeed, now $\mathbf{supp}_\epsilon$ is the empty set and the following proposition holds:

Proposition 5.1.5. *Let Σ a finite alphabet, $\mathcal{S}$ a timed structure and $\langle A, \mathbf{T}^\kappa\rangle \in \mathbb{A}(\mathcal{S}, \Sigma)$ a controlled automaton without silent transitions. Then $\mathbf{D}_{A,\kappa}S = \mathbf{M}_Q S$ and for all $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$*

$$\mathbf{Reach}(\mathbf{T}^{\mathbf{D}\kappa}, w, c_I) = \{([\mathbf{supp}(U_w(\boldsymbol{\nu}_0)), \emptyset], U_w(\boldsymbol{\nu}_0)\}$$

Proof.

From the definition of $\mathbf{U}_d$ knowing that A is without silent transition, it is clear that for all $d \in \mathbb{R}_{\geq 0}$,

$$\overset{d}{\hookrightarrow}_M = \{(\boldsymbol{\nu}, \mathbf{U}_d(\boldsymbol{\nu})), \boldsymbol{\nu} \in \mathbf{M}_Q E\} = \overset{d}{\hookrightarrow}_{A,\kappa}$$

Hence $\mathbf{D}_{A,\kappa}S = \mathbf{M}_Q S$.

The second property can then easily be proved by induction using property 5.1.4 since $\mathbf{supp}_\epsilon$ is constant equal to $\emptyset$ and for all $d \in \mathbb{R}_{\geq 0}$,and for all $\boldsymbol{\nu} \in \mathbf{M}_Q V$, $\mathbf{U}_d(\boldsymbol{\nu}) = \boldsymbol{\nu} \oplus_{\mathbf{M}_Q S} d$ so $\mathbf{supp}(\boldsymbol{\nu}) = \mathbf{supp}(\mathbf{U}_d(\boldsymbol{\nu}))$. ■

In this case, we omit the $\mathbf{supp}_\epsilon$ part in the states and consider that the state space of $\mathbf{D}_\kappa A$ is $\mathcal{P}(Q)$ instead of $\mathcal{P}(Q)^2$. We'll write for all $w \in (\Sigma \times \mathbb{R}_{\geq 0})^*$, $\rho_w = \mathbf{supp}(\mathbf{U}_w(\mathbf{D}c_I))$ and $\boldsymbol{\nu}_w = \mathbf{U}_w(\mathbf{D}c_I)(q)$, which means according to last proposition (5.1.5) and with the new notation, that for all $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$

$$\mathbf{Reach}(\mathbf{T}^{\mathbf{D}\kappa}, w, c_I) = \{(\rho_w, \boldsymbol{\nu}_w)\}$$

We are ready to state now the main result of this part: every controlled automaton *without silent transitions* is determinizable:

Theorem 5.1.6. *Let Σ a finite alphabet, $\mathcal{S}$ a timed structure and $\langle A, \mathbf{T}^\kappa\rangle \in \mathbb{A}(\mathcal{S}, \Sigma)$ without silent transitions.*

$\langle A, \mathbf{T}^\kappa\rangle$ is Σ-determinisable.

Proof.

According to proposition 5.1.4, $\mathbf{D}_\kappa A$ is $\mathbf{D}\kappa$-deterministic, and without silent transitions. We prove here that $\mathcal{L}_\kappa(A) = \mathcal{L}_{\mathbf{D}\kappa}(\mathbf{D}_\kappa A)$.

Let's write $A = \langle Q, I, T, F\rangle$ and recall that $\mathbf{D}_\kappa c_I = ((\mathbf{supp}(\boldsymbol{\nu}_I), \mathbf{supp}_\epsilon(\boldsymbol{\nu}_I)), \boldsymbol{\nu}_I)$

$$w \in \mathcal{L}_\kappa(A) \iff \exists c_i \in I, \exists q \in F, \exists v \in V, (q, v) \in \mathbf{Reach}(\mathbf{T}^\kappa, w, c_i)$$

by definition. According to proposition 5.1.4 we have then :

$$w \in \mathcal{L}_\kappa(A) \iff \exists q \in F, \exists v \in V, q \in \rho_w \text{ and } v \in \boldsymbol{\nu}_w(q)$$

But by definition of $\mathbf{supp}$ and of $\mathbf{D}_\kappa F$ we know that for all $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$,

$$\exists q \in F, q \in \rho_w \iff \rho_w \in \mathbf{D}_\kappa F$$

and $\forall q \in Q$,

$$\exists v \in \boldsymbol{\nu}_w(q) \iff q \in \mathbf{supp}(\nu_w(q)) = \rho_w$$

Hence,

$$w \in \mathcal{L}_\kappa(A) \iff \rho_w \in \mathbf{D}_\kappa F$$

Finally, according to proposition 5.1.5,

$$w \in \mathcal{L}_\kappa(A) \iff \exists \rho \in \mathbf{D}_\kappa F, \exists \boldsymbol{\nu} \in \mathbf{M}_Q V, (\rho, \boldsymbol{\nu}) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{D}\kappa}, w, c_I)$$

$$w \in \mathcal{L}_\kappa(A) \iff w \in \mathcal{L}_{\mathbf{D}\kappa}(\mathbf{D}_\kappa A)$$

which concludes the proof of the theorem. ■

Before ending this section, we suggest taking a look at the questions of representation and computation of our newly defined powerset automaton. We focus only on the case when the initial automaton A is *without silent transitions*.

If A has no silent transition we defined a lot of superfluous transition in $\mathbf{D}_\kappa A$. Indeed according to proposition 5.1.5 we know that we can consider $\mathbf{D}_\kappa T$ to be equal to:

$$\mathbf{D}_\kappa T = \{(\mathbf{supp}(\boldsymbol{\nu}), \boldsymbol{\nu}, \mathbf{U}_\sigma, \mathbf{supp}(\mathbf{U}_\sigma(\boldsymbol{\nu}))), \boldsymbol{\nu} \in \mathcal{P}(V)^Q, \sigma \in \Sigma\}$$

with no impact on the semantic of $\mathbf{D}_\kappa A$.

This being said we can prove the following proposition.

Proposition 5.1.7. *Let Σ a finite alphabet, $\mathcal{S}_G$ a guarded timed structure and $\langle A, \mathbf{T}^\kappa \rangle \in \mathbb{A}(\mathcal{S}_G, \Sigma)$ without silent transitions.*

$\langle \mathbf{D}_\kappa A, \mathbf{D}\kappa \rangle$ is compatible with $\mathbf{M}_Q G$.

Proof.

Just notice that in this context the guard of $\mathbf{D}_\kappa A$ can be written, for all $\rho, \rho' \in \mathcal{P}(Q)$ and for all $\sigma \in \Sigma$ as

$$g_{\mathbf{D}\kappa}(\rho, \sigma, \rho') = \{(\boldsymbol{\nu}, \mathbf{U}_\sigma), \boldsymbol{\nu} \in \mathbf{M}_Q V \mid \mathbf{supp}(\boldsymbol{\nu}) = \rho \text{ and } \mathbf{supp}(\mathbf{U}_\sigma(\boldsymbol{\nu})) = \rho'\}$$

Which means, if we recall that $\mathbf{U}_\sigma = \mathbf{U}_{\mathbf{g}_\sigma} \in \mathbf{M}_Q U$, that

$$g_{\mathbf{D}\kappa}(\rho, \sigma, \rho') = (G_{\rho, \mathbf{g}_\sigma, \rho'}, \mathbf{U}_{\mathbf{g}_\sigma}) \in \mathbf{M}_Q G$$

This proves the proposition. ■

We describe below how to finitely represent and compute $\boldsymbol{\nu} \in G_{\rho, \mathbf{g}_\sigma, \rho'}$ taking support on the initial automaton. Fix a guard based G such that $\langle A, \kappa \rangle$ is compatible with G.

Recall that for all $\boldsymbol{\nu} \in \mathbf{M}_Q G$, for all $\sigma \in \Sigma$ and $q' \in Q$ we have

$$\mathbf{U}_\sigma(\boldsymbol{\nu})(q') = \{u(v) \in V \mid \exists q \in Q, v \in \boldsymbol{\nu}(q) \text{ and } (v, u) \in g_\kappa(q, \gamma, q')\}$$

But if g_κ can be decomposed on G it means that for all $q, q' \in Q$ and $\sigma \in \Sigma$, there exists $\nu^1_{q,\sigma,q'}, \dots, \nu^{m_{q,\sigma,q'}}_{q,\sigma,q'}$ and $\mu^1_{q,\sigma,q'}, \dots, \mu^{m_{q,\sigma,q'}}_{q,\sigma,q'}$ such that $g_\kappa(q,\gamma,q') = \bigcup_{i=1}^{m_{q,\sigma,q'}} \nu^i_{q,\sigma,q'} \times \mu^i_{q,\sigma,q'}$. Hence if we write $Q = \{q_1, \dots, q_n\}$:

$$\begin{aligned}
\mathbf{U}_\sigma(\boldsymbol{\nu})(q') = \{u(v) \in V \mid\ & v \in \boldsymbol{\nu}(q_1) \cap \nu^1_{q_1,\sigma,q'} \text{ and } u \in \mu^1_{q_1,\sigma,q'} \\
& \textbf{or } v \in \boldsymbol{\nu}(q_1) \cap \nu^2_{q_1,\sigma,q'} \text{ and } u \in \mu^2_{q_1,\sigma,q'} \\
& \vdots \\
& \textbf{or } v \in \boldsymbol{\nu}(q_1) \cap \nu^{m_{q_1,\sigma,q'}}_{q_1,\sigma,q'} \text{ and } u \in \mu^{m_{q_1,\sigma,q'}}_{q_1,\sigma,q'} \\
& \textbf{or } v \in \boldsymbol{\nu}(q_2) \cap \nu^1_{q_2,\sigma,q'} \text{ and } u \in \mu^1_{q_2,\sigma,q'} \\
& \vdots \\
& \vdots \\
& \textbf{or } v \in \boldsymbol{\nu}(q_n) \cap \nu^{m_{q_n,\sigma,q'}}_{q_n,\sigma,q'} \text{ and } u \in \mu^{m_{q_n,\sigma,q'}}_{q_n,\sigma,q'}\}
\end{aligned}$$

This means that for all $\sigma \in \Sigma$, $\mathbf{U}_\sigma$ can be entirely described by a *finite* disjunction of rules of the type

$$\textbf{if } v \in \boldsymbol{\nu}(q) \cap \nu \textbf{ then for all } u \in \mu, u(v) \to \boldsymbol{\nu}'(q')$$

which are easily understandable in a representation, and easily computable if we can enumerate the set of $v \in \boldsymbol{\nu}(q)$ and $u \in \mu$.

Remark 5.1.1. This is exactly how we implement the update $\mathbf{U}_\sigma$ in our tool for diagnosis, **DOTA**, which we will present in part III.

If we write $\rho = \{a_1, \dots, a_{m_\rho}\}$ and $\rho' = \{b_1, \dots, b_{m'_\rho}\}$, this also means that we have for all $\boldsymbol{\nu}$ such that $\mathbf{supp}(\boldsymbol{\nu}) = \rho$:

$$\begin{aligned}
\mathbf{supp}(\mathbf{U}_\sigma(\boldsymbol{\nu})) = \rho' \iff & \forall b \in \rho', \\
& [\nu(a_1) \cap \nu^1_{a_1,\sigma,b_1} \neq \emptyset \\
& \textbf{ or } \boldsymbol{\nu}(a_1) \cap \nu^2_{a_1,\sigma,b} \neq \emptyset \\
& \vdots \\
& \textbf{ or } \boldsymbol{\nu}(a_1) \cap \nu^{m_{a_1,\sigma,b}}_{a_1,\sigma,b} \neq \emptyset \\
& \textbf{ or } \boldsymbol{\nu}(a_2) \cap \nu^1_{a_2,\sigma,b} \neq \emptyset \\
& \vdots \\
& \textbf{ or } \boldsymbol{\nu}(a_{m_\rho}) \cap \nu^{m_{a_{m_\rho},\sigma,b}}_{a_{m_\rho},\sigma,b} \neq \emptyset\ ] \\
& \textbf{and } \forall b \notin \rho', \\
& [\ \boldsymbol{\nu}(a_1) \cap \nu^1_{a_1,\sigma,b} = \emptyset \\
& \textbf{ and } \boldsymbol{\nu}(a_1) \cap \nu^2_{a_1,\sigma,b} = \emptyset \\
& \vdots \\
& \textbf{ and } \boldsymbol{\nu}(a_{m_\rho}) \cap \nu^{m_{a_{m_\rho},\sigma,b}}_{a_{m_\rho},\sigma,b} = \emptyset\ ]
\end{aligned}$$

This means that for all $\sigma \in \Sigma$, $\boldsymbol{\nu} \in G_{\rho',\mathbf{g}_\sigma,\rho'}$ can be entirely described by the combination of a simple formula describing $\mathbf{supp}(\boldsymbol{\nu}) = \rho$ and a *finite* formula with as atomic propositions

$$\boldsymbol{\nu}(q) \cap \nu \neq \emptyset \quad \text{or} \quad \nu(q) \cap \nu = \emptyset$$

which are again easily understandable in a representation, and easily computable if we can decide $\boldsymbol{\nu}(q) \cap \nu = \emptyset$.

In the next section we take a look at what this construction looks like when applied to a timed automaton.

5.2 Application to Timed Automata

Obviously timed automata as defined in this thesis (c.f. example 3.1.1) are concerned by theorem 5.1.6.

Corollary 5.2.1. *Let Σ a finite alphabet, $M \in \mathbb{N}$ and $n \in \mathbb{N}_{>0}$. All controlled timed automata $\langle A, \mathbf{T}^\kappa \rangle \in \Sigma\mathbf{TA}_M^n$ without silent transitions is Σ-determinizable.*

We propose in this section to take a look at the particular shape of the powerset automaton of a timed automaton.

Fix below a finite alphabet Σ, $M \in \mathbb{N}$, $n \in \mathbb{N}_{>0}$ and $\langle A, \mathbf{T}^\kappa \rangle \in \Sigma\mathbf{TA}_M^n$ a timed automaton without silent transition. We write $A = \langle Q, I, T, F \rangle$.

- The powerset automaton of A, $\mathbf{D}_\kappa A$ is an automaton on $\mathbf{M}_Q\mathcal{C}_M^n$. The values are maps from Q to sets of vectors of clocks which evolve all synchronously with time. This can be easily implemented by lists of arrays.
- Updates can be represented by a finite disjunction of rules of the type

  $$q \ : \ (x_1 \in I_1 \wedge x_2 \in I_2 \cdots \wedge x_n \in I_n) \implies \bigwedge_{i=0}^{m} \textit{instr}_i$$

  with $m \in \mathbb{N}$. This expression if obtained by rewriting the updates form described at the end of last section **if** $v \in \boldsymbol{\nu}(q) \cap \nu$ **then for all** $u \in \mu, u(v) \to \boldsymbol{\nu}'(q')$ in the context of timed automata. We consider "$q :$" to stand for "$\forall (x_1, \ldots, x_n) \in \boldsymbol{\nu}(q)$", $\implies$ is a shortcut for **if** ... **then** and for all $i \in [\![1, m]\!]$, *instr*$_i$ is of the form,

  $$0 \to q' \quad \text{or} \quad x_k \to q' \ \text{(for some } k \in [\![1, n]\!]\text{)}$$

  An instruction $x \to q$ means that in the resulting marking x will belong to the set associated with q.

  Those instructions can be easily computed.
- Guards can be represented by an update and a boolean formula with atomic propositions of the form

  $$q \cap I = \emptyset$$

  with q a state, I being either $\mathbb{C}_M^n$, either a product of n intervals with as bounds two consecutive integers, and $q \cap I$ stand for $\boldsymbol{\nu}(q) \cap I$.

  Those constraints can easily be effectively tested.

In conclusion powerset automata of a timed automaton can be finitely represented and are easily implementable. This makes them an interesting tool in the context of fault diagnosis and we'll dedicate the third part of this thesis to this problem.

Example 5.2.1. On figure 5.2 we give a representation of the powerset automaton of a famous non-determinizable (in the sense of [4]) timed automata, $\mathcal{B}$, described on example 4.3.4 and figure 4.4 .

For clarity sake, we omit in the representation to precise the part of the guard constraint imposing $\mathbf{supp}(\boldsymbol{\nu}) = \rho$ since it can be deduced from the source state labeled by ρ ($\forall q \in \rho, \boldsymbol{\nu}(q) \neq \emptyset$ **and** $\forall q \notin \rho, \boldsymbol{\nu}(q) = \emptyset$). We also omit to write $x \in \mathbb{C}_1 \implies \dots$ and leave it blank instead.

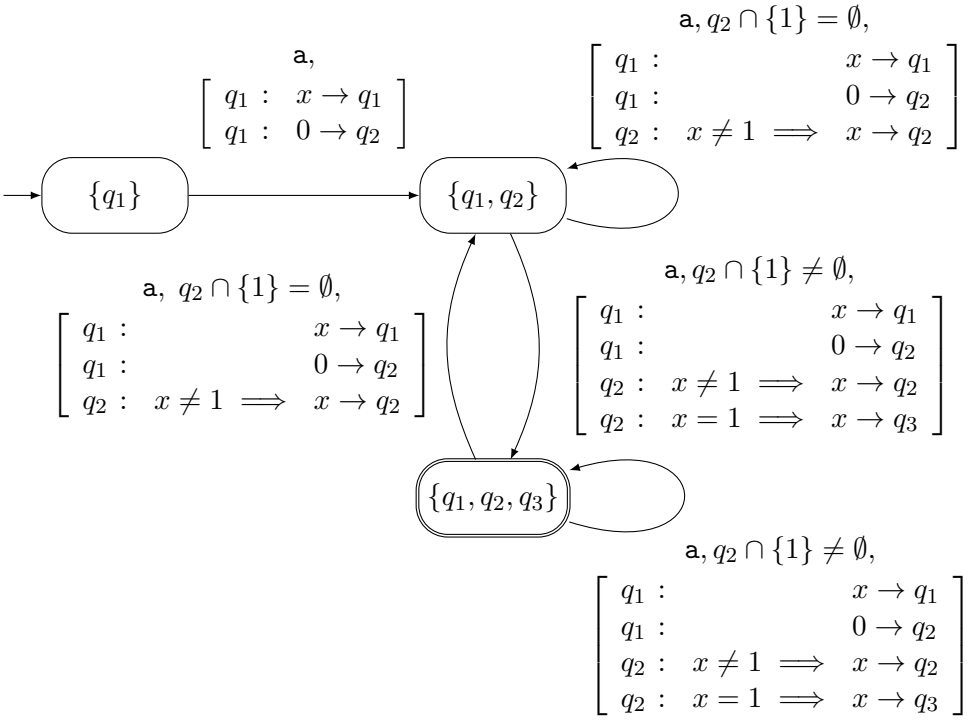


Figure 5.2: $\mathbf{D}\mathcal{B}$ - Determinized automaton of $\mathcal{B}$ of Figure 4.4

We show on figure 5.4 an example of the runs associated to $w = a \cdot 2 \cdot a \cdot 0.3 \cdot a \cdot 0.4 \cdot a \cdot 0.6 \cdot a$ executed synchronously in $\mathcal{B}$ (example 4.4) and its powerset automaton $\mathbf{D}\mathcal{B}$ (figure 5.2). The tree at the top of figure 5.4 is the possible runs of the timed automaton $\mathcal{B}$ with, in gray, delays and, in red, actions. The run below is the only run which recognizes w in $\mathbf{D}\mathcal{B}$, with delay in gray and actions in red. The cyan dotted lines show how the configurations of $\mathbf{D}\mathcal{B}$ summarize all the reachable configurations of $\mathcal{B}$ at any moment of the run.

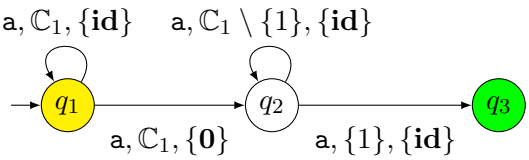


Figure 4.4: $\mathcal{B}$, a non-determinizable TA (repeated from page 42)

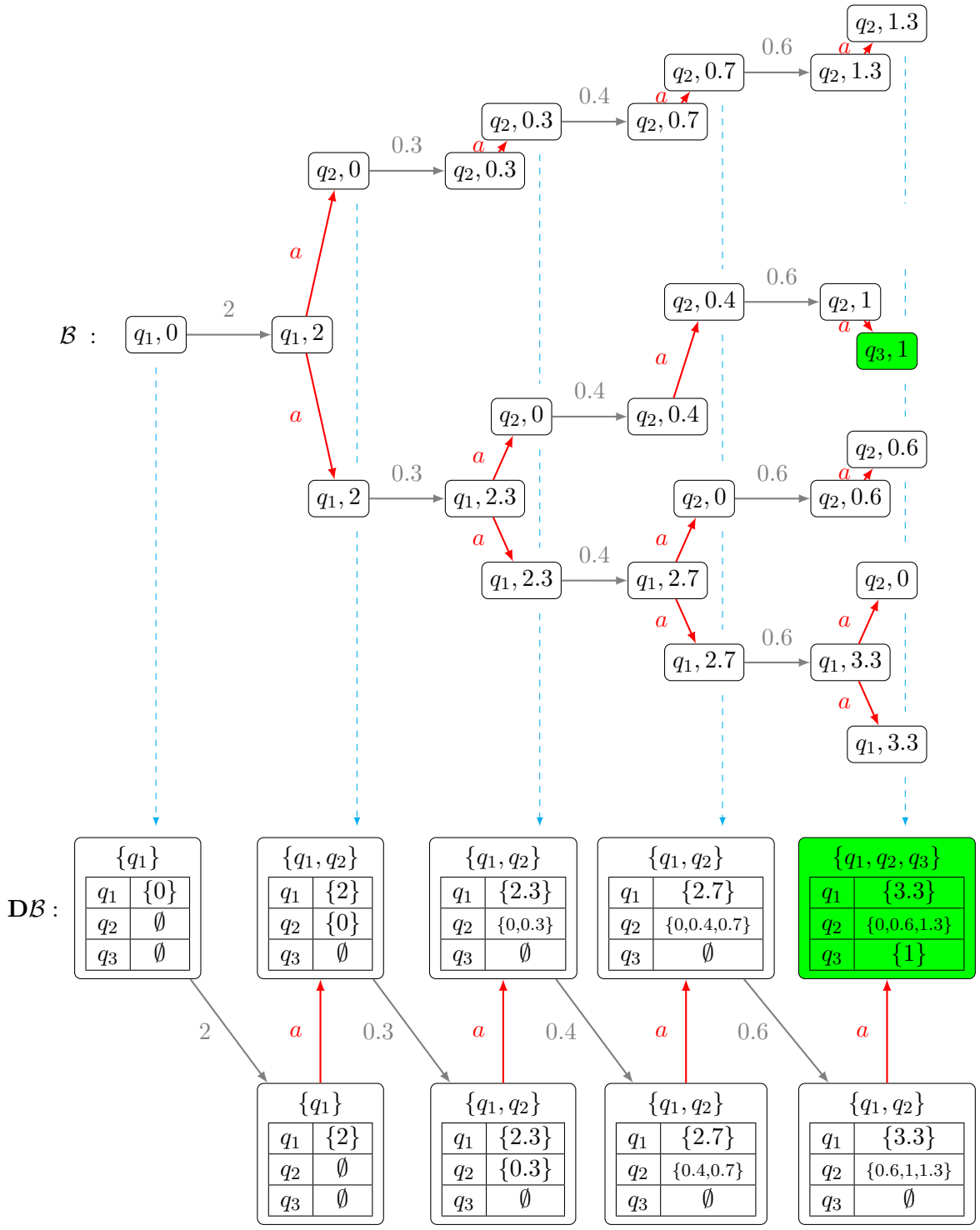


Figure 5.4: Simulation of $a \cdot 2 \cdot a \cdot 0.3 \cdot a \cdot 0.4 \cdot a \cdot 0.6 \cdot a$

Chapter 6

Timed Structure Equivalences

A Step Toward Strong Determinization

The last chapter introduced a determinization construction for all automata on timed domains equipped with a timed control without silent transitions. The determinized automaton is however constructed on a different timed structure obtained by an E-marking construction (c.f. page 22), a kind of powerset construction. In the next chapter, we will prove that this construction can also be used to study the more classical problem of determinization within the same class, i.e. in our framework: without changing the timed structure. We will use our technique to prove again the determinization results one can find in the literature and study how they relate to each other.

In this overview of our work, this chapter consists of a theoretical interlude between those two chapters. Its goal is to introduce two *equivalences* between timed structures which build an important bridge between the clock structure (c.f. example 2.3.4 page 20) and the Q-marking construction of the clock structure, i.e. a bridge between the timed structure used to model a timed automaton and the timed structure model used in its powerset automata. This bridge will help us expose some sufficient conditions allowing to construct from the powerset automaton of a timed automaton, an equivalent deterministic timed automaton, i.e. to construct thanks to our powerset construction the determinized timed automaton of a timed automaton.

Those equivalences are intuitive but their proofs require heavy technicalities, therefore we chose to detail them in a separate chapter. We dedicate one section for each equivalence.

Section 6.1 : The first equivalence is similar to the equivalence result presented in [19] between updatable timed automata and timed automata. It proves that an enhanced timed automata without silent transitions can be simulated by a timed automata without silent transitions. Recall that an enhanced timed automaton is defined as an automaton on the enhanced clock timed structure where values are partial functions and updates can reset, disable or swap clocks (c.f. example 2.3.6 page 21).

Section 6.2 : The second equivalence closes the gap between an enhanced clock structure and the timed structure of the powerset automaton of a timed automaton. Precisely, given any timed structure S, we prove the equivalence between an automaton on the E-marking construction of S bounded by some integer n – the notion of bounded automata will be defined in the section – and an automaton on the $E \times [\![1, n]\!]$ -map construction of S (c.f page 20).

Putting together those two equivalences draws an equivalence between bounded powerset automata of timed automata without silent transitions and timed automata without silent transitions.

6.1 From Product to Maps

Let us study the particular shape of updates in the enhanced clock structure. We suppose $E = \{x_1, \ldots, x_n\}$.

Recall that an update $\mathbf{u}_f$ is based on a partial function f from E to $\{\mathbf{id}, \mathbf{0}\} \times E$. Let $\mathbf{x}$ be a value in $[E \rightharpoonup \mathbb{C}_M]$, let $\mathbf{y} = \mathbf{u}_f(\mathbf{x})$ and let $e' \in E$. $\mathbf{y}(e') = u(\mathbf{x}(e))$ with $\mathbf{u}_f(e') = (u, e)$ and $e \in \mathbf{dom}(\mathbf{x})$ by definition. So if $u = \mathbf{id}$, then $\mathbf{y}(e') = \mathbf{x}(e)$ and if $u = \mathbf{0}$, then $\mathbf{y}(e') = 0$. It's easy to see that the effect of $\mathbf{u}_f$ on a value $\mathbf{x}$ can be intuitively represented by a combination of operations of the type $x_i := 0$ (reset of a clock, x_i is mapped to 0 by $\mathbf{u}_f(\mathbf{x})$) or $x_i := x_j$ (x_i is mapped to the previous value of x_j $\mathbf{u}_f(\mathbf{x})$).

With this notice, the idea of the proof is to simulate a configuration $(q, \mathbf{x})$ of an enhanced timed automaton by a configuration $((q, f), \overrightarrow{y})$ of a timed automaton such that f is partial function describing where the value of a clock x_i of $\mathbf{x}$ can be found within the clocks in $\overrightarrow{y}$.

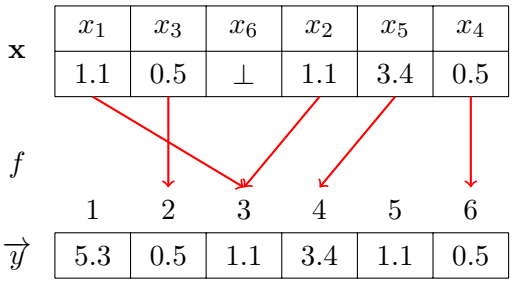


Figure 6.1: How $\mathbf{x}$ is recovered from a partial function f and a vector $\vec{y}$

Then an update $\mathbf{u}_f$ in the enhanced timed automaton is simulated by an update $\overrightarrow{u}$ depending on f and the adequate choice of a function f' for the destination state. They are chosen such that all operations $x_i := 0$ of $\mathbf{u}$ are simulated by the reset of one clock y_0 in $\overrightarrow{y}$ and the fact that f' picks the value of x_i on y_0; all operations $x_i := x_j$ enforce that the clock y giving its value to x_j must not be reset by $\overrightarrow{u}$ and f' picks the value of x_i on y.

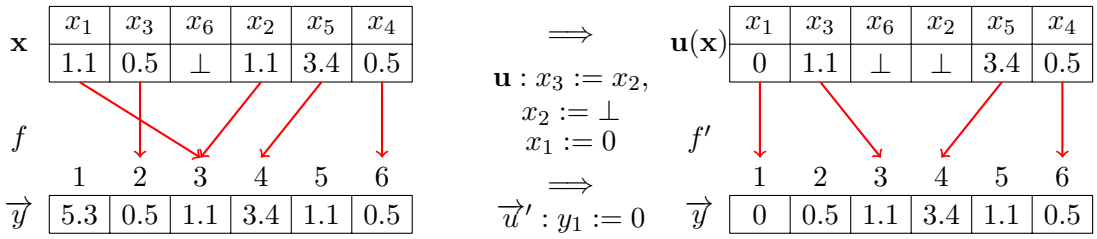


Figure 6.2: How $\mathbf{u}$ is simulated by constructing $\vec{u}'$ and f'

Formally we provide below the formal statement and proof of the result.

Theorem 6.1.1. *Let Σ be a finite alphabet, $M \in \mathbb{N}$ and C be a finite set.*

For all controlled enhanced timed automata $\langle A_2, \mathbf{T}^{\kappa_2} \rangle \in \Sigma\mathbf{TA}_M^C$ without silent transitions, there exists a controlled timed automata $\langle A_1, \mathbf{T}^{\kappa_1} \rangle \in \Sigma\mathbf{TA}_M^{|C|}$ without silent transition which simulates $\langle A_2, \mathbf{T}^{\kappa_2} \rangle$.

Proof.

Let's write $A_2 = \langle Q_2, I_2, T_2, F_2 \rangle$ and $n = |C|$. Let $\mathbf{ind}$ be any bijection from C to $[\![1, n]\!]$. Let $Q_1 = Q_2 \times [C \rightharpoonup [\![1, n]\!]]$, we are going to construct $\Xi = (\zeta, \phi, \psi)$ candidate to be a $\Sigma, \mathbb{G}_M^n$-compatible, pre-functional bisimulation from $Q_1, \mathcal{C}_M^n$ to A_2, which is Σ-deterministic if $\langle A_2, \kappa_2 \rangle$ is.

- Let $\zeta = Q_1 \to Q_2$ be the projection of Q_1 on Q_2. For all $(q, f) \in Q_1$, $\zeta(q, f) = q$.
  The following lemma is trivial:

Lemma 6.1.2. *ζ is surjective.*

- For all $(q, f) \in Q_1$, let $\phi_{(q,f)} : \mathbb{C}_M^n \to \mathbb{C}_M^C$ mapping any $(x_1, \ldots, x_n)$ to

$$\phi_{(q,f)}(x_1, \ldots, x_n) : C \rightharpoonup \mathbb{C}_M$$
$$e \mapsto x_{f(e)} \text{ if } e \in \mathbf{dom}(f)$$

Lemma 6.1.3. *For all $(q, f) \in Q_1$, ϕ_{q_1} is a functional bisimulation of timed domains. For all $q_2, \mathbf{x}_2 \in I_2$, there exists $f \in [C \rightharpoonup [\![1, n]\!]]$, there exists $\overrightarrow{x}_1 \in \mathbb{C}_M^n$, such that $\phi_{(q_2,f)}(\vec{x}_1) = \mathbf{x}_2$.*

Proof.

- Let $(q, f) \in Q_1$. $\mathbb{C}_M^n$ and $\mathbb{C}_M^C$ are both deterministic and trivially for all $d \in \mathbb{R}_{\geq 0}$,

  $$\phi_{(q,f)}((x_1, \ldots, x_n) \oplus d) = \phi_{(q,f)}((x_1, \ldots, x_n)) \oplus d$$

  so $\phi_{(q,f)}$ is a functional bisimulation of timed domains.

- Let now $q_2, \mathbf{x}_2 \in I_2$.
  Let $f : C \rightharpoonup [\![1, n]\!], e \mapsto \mathbf{ind}(e)$ if $e \in \mathbf{dom}(\mathbf{x}_2)$.
  Let $\overrightarrow{x}_1 = (x_1, \ldots, x_n)$ with for all $1 \leq i \leq n$:
  - $x_i = \mathbf{x}_2(\mathbf{ind}^{-1}(i))$ if $\mathbf{ind}^{-1}(i) \in \mathbf{dom}(\mathbf{x}_2)$
  - $x_i = 0$ otherwise.

  Then $\phi_{(q_2,f)}(\overrightarrow{x}_1) = \mathbf{x}_2$.

■

We need to go through several steps before defining ψ. The first of which being the introduction of a family ψ^0 of functions at the core of the definition of ψ.

- For all $f, f' \in [C \rightharpoonup [\![1, n]\!]]$:
  If $\mathbf{dom}(f) \neq \emptyset$ then we write $m_f = \mathbf{ind}^{-1}(\min(\mathbf{ind}(\mathbf{dom}(f))))$.
  m_f is a clock of $\mathbf{dom}(f)$ mapped to the smallest element of $[\![1, n]\!]$.
  We also fix a partial function $\mathbf{pick}_{f,f'} : C \rightharpoonup C$ such that:
  - $\mathbf{dom}(\mathbf{pick}_{f,f'}) = \mathbf{dom}(f') \cap f'^{-1}(\mathbf{im}(f))$
  - for all $e \in \mathbf{dom}(\mathbf{pick}_{f,f'})$, $\mathbf{pick}_{f,f'}(e) \in f^{-1}(f'(e))$.

  We call such function the *picking function on f and f'*.
  This function returns, given a clock e mapped to i by f', another clock e' mapped to the same number i by f'.

- Now for all $q_1 = (q, f), q_1' = (q', f') \in Q_1$:
  Let $\psi^0_{q_1,q_1'} : \{\mathbf{id}, \mathbf{0}\}^n \rightharpoonup [C \rightharpoonup \{\mathbf{id}, \mathbf{0}\} \times C]$ such that $\mathbf{dom}(\psi^0_{q_1,q_1'}) = \emptyset$ if $\mathbf{dom}(f) = \emptyset$, otherwise:

  $$\mathbf{dom}(\psi^0_{q_1,q_1'}) = \{(u_1, \dots, u_n) \in \{\mathbf{id}, \mathbf{0}\}^n \mid \forall 1 \leq i \leq n, i \notin \mathbf{im}(f) \implies u_i = \mathbf{0}\}$$

  and defined for all $(u_1, \dots, u_n) \in \mathbf{dom}(\psi^0_{q_1,q_1'})$ by:

  $$\begin{aligned} \psi^0_{q_1,q_1'}(u_1, \dots, u_n) : C &\rightharpoonup \{\mathbf{id}, \mathbf{0}\} \times C \\ e &\mapsto (\mathbf{id}, \mathbf{pick}_{f,f'}(e)) && \text{if } u_{f'(e)} = \mathbf{id} \\ e &\mapsto (\mathbf{0}, m_f) && \text{if } u_{f'(e)} = \mathbf{0} \text{ and } \mathbf{dom}(f) \neq \emptyset \end{aligned}$$

  Notice that if $u_{f'(e)} = \mathbf{id}$ implies $e \in \mathbf{dom}(\mathbf{pick}_{f,f'})$ by definition of $\mathbf{dom}(\psi^0_{q_1,q_1'})$.
  For short we write for all $\overrightarrow{u} \in \mathbf{dom}(\psi^0_{q_1,q_1'})$, $\mathbf{u}_{q_1,q_1',\overrightarrow{u}}$ for the update of U_C associated to the partial function $\psi^0_{q_1,q_1'}(\overrightarrow{u})$.

Lemma 6.1.4. *For all $q_1 = (q, f), q_1' = (q', f') \in Q_1$, for all $\overrightarrow{x} \in \mathbb{C}^n_M$ and for all $\overrightarrow{u} \in \mathbf{dom}(\psi^0_{q_1,q_1'})$, it holds that*

$$\phi_{q_1'}(\overrightarrow{u}(\overrightarrow{x})) = \mathbf{u}_{q_1,q_1',\overrightarrow{u}}(\phi_{q_1}(\overrightarrow{x}))$$

Proof.

- If $\mathbf{dom}(f) = \emptyset$, $\mathbf{dom}(\psi^0_{q_1,q_1'}) = \emptyset$ and the result is trivially true.

- Assume $\mathbf{dom}(f) \neq \emptyset$.
  Let's write $\overrightarrow{x} = (x_1, \dots, x_n)$ and $\overrightarrow{u} = (u_1, \dots, u_n)$.
  Let $e \in C$.
  - Suppose $u_{f'(e)} = \mathbf{id}$,

    $$\begin{aligned} \phi_{q_1'}(\overrightarrow{u}(\overrightarrow{x}))(e) &= u_{f'(e)}(x_{f'(e)}) = x_{f'(e)} \\ &= x_{f(\mathbf{pick}_{f,f'}(e))} = u_{f'(e)}(x_{f(\mathbf{pick}_{f,f'}(e))}) \\ &= \mathbf{u}_{q_1,q_1',\overrightarrow{u}}(\phi_{q_1}(\overrightarrow{x}))(e) \end{aligned}$$

  - Suppose $u_{f'(e)} = \mathbf{0}$

    $$\begin{aligned} \phi_{q_1'}(\overrightarrow{u}(\overrightarrow{x}))(e) &= u_{f'(e)}(x_{f'(e)}) \\ &= 0 \\ &= u_{f'(e)}(m_f) \\ &= \mathbf{u}_{q_1,q_1',\overrightarrow{u}}(\phi_{q_1}(\overrightarrow{x}))(e) \end{aligned}$$

  - Suppose finally $e \notin \mathbf{dom}(f')$.
    $\phi_{q_1}(\overrightarrow{u}(\overrightarrow{x}))(e)$ is not defined and $\mathbf{u}_{q_1,q_1',\overrightarrow{u}}(\phi_{q_1}(\overrightarrow{x}))(e)$ is not defined either.

■

- Let now $q_1 = (q, f) \in Q_1$, $q' \in Q_2$ and $g \in [C \rightharpoonup \{\mathbf{id}, \mathbf{0}\} \times C]$:
  - We define, for all $1 \leq i \leq n$,

$$u_i = \begin{cases} \mathbf{0} & \text{if } i \notin \mathbf{im}(f) \\ \mathbf{0} & \text{if } i \in \mathbf{im}(f) \text{ and } \forall e' \in f^{-1}(i), (\mathbf{id}, e') \notin \mathbf{im}(g) \\ \mathbf{id} & \text{otherwise} \end{cases}$$

    and $\overrightarrow{u}_{q_1,q',g} = (u_1, \ldots, u_n)$.

    Let $i_0 = \min_{1 \leq i \leq n}(u_i = \mathbf{0})$ if it exists. Notice that:

    If there exists $e \in \mathbf{dom}(g)$ such that $g(e) = (\mathbf{0}, e')$ with $e' \in \mathbf{dom}(f)$ and $u_{f(e')} = \mathbf{id}$, then i_0 is well-defined.

    Indeed, exists $(\mathbf{id}, e'') \in \mathbf{im}(g)$ such that $f(e'') = f(e')$, so:
    - If $e'' \neq e'$, then f is not injective and $\mathbf{im}(f) \neq [\![1, n]\!]$, which ensures the existence of i_0.
    - If $e'' = e'$, then there exists $e^{(3)} \in C$ such that neither $(\mathbf{id}, e^{(3)})$ nor $(\mathbf{0}, e^{(3)})$ are in $\mathbf{im}(g)$, by a cardinality argument.
      - If $e^{(3)} \notin \mathbf{dom}(f)$ then $\mathbf{im}(f) \neq [\![1, n]\!]$ which ensures the existence of i_0.
      - Else if $f^{-1}(f(e^{(3)})$ is not reduced to $e^{(3)}$, then f is not injective and $\mathbf{im}(f) \neq [\![1, n]\!]$ which ensures the existence of i_0.
      - Finally if $f^{-1}(f(e^{(3)})$ is reduced to $e^{(3)}$, the property:
        $\forall e^{(4)} \in f^{-1}(f(e^{(3)}), (e^{(4)}, \mathbf{id}) \notin u_2(C)$
        is verified and $u_{f(e^{(3)})} = \mathbf{0}$, which ensures the existence of i_0.

    ■

  - We define $f'_{q_1,q',g} : C \rightharpoonup [\![1, n]\!]$ as follows:
    - if $e \in \mathbf{dom}(g)$ and $g(e) = (\mathbf{id}, e')$ with $e' \in \mathbf{dom}(f)$:
      $f'_{q_1,q',g}(e) = f(e')$
    - if $e \in \mathbf{dom}(g)$, $g(e) = (\mathbf{0}, e')$ with $e' \in \mathbf{dom}(f)$ and $u_{f(e')} = \mathbf{0}$:
      $f'_{q_1,q',g}(e) = f(e')$
    - if $e \in \mathbf{dom}(g)$, $g(e) = (\mathbf{0}, e')$ with $e' \in \mathbf{dom}(f)$ and $u_{f(e')} = \mathbf{id}$:
      $f'_{q_1,q',g}(e) = i_0$

    We write $q_{q_1,q',g} = (q', f_{q_1,q',g})$.

    Finally we write $\mathbf{u}_{q_1,q',g} = \mathbf{u}_{q_1,q_{q_1,q',g},\overrightarrow{u}_{q_1,q',g}}$.

Lemma 6.1.5. *For all $\overrightarrow{x} \in \mathbb{C}_M^n$, $\mathbf{u}_{q_1,q',g}(\phi_{q_1}(\overrightarrow{x})) = \mathbf{u}_g(\phi_{q_1}(\overrightarrow{x}))$.*

Proof.

Let $e \in C$ and let's write $\overrightarrow{x} = (x_1, \ldots, x_n) \in \mathbb{C}_M^n$.
Let $\mathbf{x} = \phi_{(q,f)}(\overrightarrow{x})$.

- Suppose $\mathbf{dom}(f) = \emptyset$.
  Then $\mathbf{dom}(f'q_1, q', g) = \emptyset$ and $\mathbf{u}_{q_1,q',g}(\mathbf{x})(e)$ is not defined, neither is $\mathbf{u}_g(\mathbf{x})(e)$.
- Suppose now $\mathbf{dom}(f) \neq \emptyset$.
  - Suppose $e \in \mathbf{dom}(g)$ and $g(e) = (u, e')$ with $u \in \{\mathbf{id}, \mathbf{0}\}$ and $e' \notin \mathbf{dom}(f)$.
    Then $\mathbf{u}_g(\mathbf{x})(e)$ is not defined.
    Also $e \notin \mathbf{dom}(f'q_1, q', g)$ and $\mathbf{u}_{q_1,q',g}(\mathbf{x})(e)$ is not defined either.
  - Suppose $e \in \mathbf{dom}(g)$ and $g(e) = (\mathbf{id}, e')$ with $e' \in \mathbf{dom}(f)$.
    Then $f'_{q_1,q',g}(e) = f(e')$ and $u_{f'_{q_1,q',g}(e)} = \mathbf{id}$.
    $$\begin{aligned}\mathbf{u}_{q_1,q',g}(\mathbf{x})(e) &= \mathbf{x}(\mathbf{pick}_{f,f'_{q_1,q',g}}(e)) \\ &= x_{f(\mathbf{pick}_{f,f'_{q_1,q',g}}(e))} \\ &= x_{f'_{q_1,q',g}(e)} \\ &= x_{f(e')} = \mathbf{x}(e') = \mathbf{u}_g(\mathbf{x})(e)\end{aligned}$$
  - Suppose $e \in \mathbf{dom}(g)$ and $g(e) = (\mathbf{0}, e')$ with $e' \in \mathbf{dom}(f)$.
    - Suppose $u_{f(e')} = \mathbf{0}$.
      Then $f'_{q_1,q',g}(e) = f(e')$ and $u_{f'_{q_1,q',g}(e)} = \mathbf{0}$.
      $\mathbf{u}_{q_1,q',g}(\mathbf{x})(e) = 0 = \mathbf{u}_g(\mathbf{x})(e)$
    - Suppose $u_{f(e')} = \mathbf{id}$.
      Then $f'_{q_1,q',g}(e) = i_0$ and $u_{f'_{q_1,q',g}(e)} = u_{i_0} = \mathbf{0}$.
      $\mathbf{u}_{q_1,q',g}(\mathbf{x})(e) = 0 = \mathbf{u}_g(\mathbf{x})(e)$

■

For all $q_1 = (q, f) \in Q_1$ and $G_{\mathbf{g},\mathbf{u}_g} \in \mathbb{G}_{M,C}$ a guard on $\mathcal{C}_M^C$, we say that f and $\mathbf{g}$ are *coherent* if and only if:

- $\mathbf{dom}(\mathbf{g}) = \mathbf{dom}(f)$
- for all $e, e', \in \mathbf{dom}(f)$, if $f(e) = f(e')$ then $\nu_e = \nu'_e$, where $\mathbf{g}(e) = (\nu_e, \mu_e)$ and $\mathbf{g}(e') = \nu'_e, \mu'_e$

Let again $q_1 = (q, f) \in Q_1$, $q' \in Q_2$ and $g \in [C \rightharpoonup \{\mathbf{id}, \mathbf{0}\} \times C]$.
Let also $G_{\mathbf{g},\mathbf{u}_g} = (\boldsymbol{\nu}, \{\mathbf{u}_g\}) \in \mathbb{G}_{M,C}$ such that f and $\mathbf{g}$ are coherent.

Let $\boldsymbol{\nu}_{q_1,q',g,\mathbf{g}} = I_1 \times \cdots \times I_n$ with for all $1 \leq i \leq n$:

- If $i \notin \mathbf{im}(f)$ then $I_i = \mathbb{C}_M$
- Else $I_i = \nu_e$ given any $e \in f^{-1}(i)$ with $\mathbf{g}(e) = (\nu_e, \mu_e)$, which is well-defined since, by coherence of f and $\mathbf{g}$, the choice of the inverse image of i by f does not matter.

Lemma 6.1.6. $(\phi_{q_1}^{-1}(\boldsymbol{\nu}), \{\overrightarrow{u}_{q_1,q',g}\}) = (\boldsymbol{\nu}_{q_1,q',g,\mathbf{g})}, \{\overrightarrow{u}_{q_1,q',g}\})$ *is decomposable on* $\mathbb{G}_M^n$

Proof.

Obviously $(\boldsymbol{\nu}_{q_1,q',g,\mathbf{g}}, \{\overrightarrow{u}_{q_1,q',g}\})$ is decomposable in $\mathbb{G}_M^n$.

- By definition of ϕ_{q_1} and $G_{\mathbf{g},\mathbf{u}_g}$, for all $\overrightarrow{x} \in \boldsymbol{\nu}_{q_1,q',g,\mathbf{g}}$, $\phi_{q_1}(\overrightarrow{x}) \in \boldsymbol{\nu}$.
- Let now $\overrightarrow{x} = (x_1, \ldots, x_n) \in \phi_{q_1}^{-1}(\boldsymbol{\nu})$, then for all $1 \leq i \leq n$:
  - If $i \notin \mathbf{im}(f)$, then $x_i \in I_i = \mathbb{C}_M$.
  - If $i \in \mathbf{im}(f)$, then $x_i = x_{f(e)} = \phi_{q_1}(\overrightarrow{x})(e)$ for some $e \in f^{-1}(i)$.
    Since $\mathbf{dom}(f) = \mathbf{dom}(\mathbf{g})$, $e \in \mathbf{dom}(\mathbf{g})$ and
    since $\phi_{q_1}(\overrightarrow{x}) \in \boldsymbol{\nu}$, $x_{f(e)} \in \nu_e = I_i$ with $\mathbf{g}(e) = (\nu_e, \mu_e)$.

  So finally $\overrightarrow{x} \in \boldsymbol{\nu}_{q,f,q',g,\mathbf{g}}$.

By double inclusion we proved that

$$(\phi_{q_1}^{-1}(\boldsymbol{\nu}), \{\overrightarrow{u}_{q_1,q',g}\}) = (\boldsymbol{\nu}_{q_1,q',g,\mathbf{g})}, \{\overrightarrow{u}_{q_1,q',g}\}) \text{ is decomposable on } \mathbb{G}_M^n$$

■

We are ready now to define ψ. Let $q_1 = (q, f), q_1' = (q', f') \in Q_1$ and $\sigma \in \Sigma$.

$g_{\kappa_2}(q, \sigma, q')$ is decomposable on $\mathbb{G}_{M,C}$.
Hence there exists $G_{\mathbf{g}_1,u_{g_1}}, \ldots, G_{\mathbf{g}_n,u_{g_n}} \in \mathbb{G}_{M,C}$ such that $g_{\kappa_2}(q, \sigma, q') = \{G_{\mathbf{g}_1,u_{g_1}}, \ldots, G_{\mathbf{g}_n,u_{g_n}}\}$.

- We define $D_{q_1,q_1',\sigma}$ as the set of pairs $(\overrightarrow{x}, \overrightarrow{u}) \in \mathbb{C}_M^n \times \{\mathbf{id}, \mathbf{0}\}^n$ for which there exists $1 \leq i \leq n$ such that :
  - $\overrightarrow{x} \in \boldsymbol{\nu}_{q_1,q',g_i,\mathbf{g}_i}$
  - $\overrightarrow{u} = \overrightarrow{u}_{q_1,q',g_i}$
  - $\mathbf{g}_i$ and f are coherent
  - $f' = f'_{q_1,q',g_i}$
- We also define $D_{q_1,q_1'}$ as the set of pairs $(\overrightarrow{x}, \overrightarrow{u}) \in \mathbb{C}_M^n \times \{\mathbf{id}, \mathbf{0}\}^n$ for which there exists $\sigma \in \Sigma$ such that $(\overrightarrow{x}, \overrightarrow{u}) \in D_{q_1,q_1',\sigma}$.
- We can define then:

$$\begin{aligned} \psi_{q_1,q_1'} : \mathbb{C}_M^n \times \{\mathbf{id}, \mathbf{0}\}^n &\rightharpoonup \{\mathbf{id}, \mathbf{0}\}_C \\ (\overrightarrow{x}, \overrightarrow{u}) &\mapsto \mathbf{u}_g \text{ if } (\overrightarrow{x}, \overrightarrow{u}) \in D_{q_1,q_1'} \text{ and } \overrightarrow{u} = \overrightarrow{u}_{q_1,q',g} \end{aligned}$$

Let's prove below that $\Xi = (\zeta, \phi, \psi)$ with $\phi = (\phi_{q_1})_{q_1 \in Q_1}$ and $\psi = (\psi_{q_1,q_1'})_{q_1,q_1' \in Q_1}$ is a pre-functional bisimulation from Q_1,$\mathcal{C}_M^n$ to A_2.

We check it verifies conditions (A) to (D) of proposition 3.2.2.

(A/B) This is lemma 6.1.2 and 6.1.3.

(C) For all $q_1, q_1' \in Q_1$:

(C.1) According to lemma 6.1.4 and lemma 6.1.5 for all $(\overrightarrow{x}, \overrightarrow{u}) \in D_{q_1,q_1'}$ such that $\overrightarrow{u} = \overrightarrow{u}_{q_1,q',g}$:

$$\phi_{q_1}(\overrightarrow{u}(\overrightarrow{x})) = \mathbf{u}_{q_1,q_1',\overrightarrow{u}}(\phi_{q_1}(\overrightarrow{x})) = \mathbf{u}_g(\phi_{q_1}(\overrightarrow{x})) = \psi_{q_1,q_1'}(\overrightarrow{x}, \overrightarrow{u})(\phi_{q_1}(\overrightarrow{x}))$$

(C.2) By definition of $D_{q_1,q_1'}$, we also know that there exists $\sigma \in \Sigma$,

$$(\phi_{q_1}(\overrightarrow{x}), \psi_{q_1,q_1'}(\overrightarrow{x}, \overrightarrow{u})) \in g_{\kappa_2}(q, \sigma, q') \subseteq g_{A_2}(q, q')$$

(D) Let now $q_1 = (q, f) \in Q_1$, $\overrightarrow{x} \in \mathbf{dom}(\phi_{q_1})$, $q' \in Q_2$ and $g \in [C \rightharpoonup \{\mathbf{id}, \mathbf{0}\} \times C]$.
Suppose $(\phi_{q_1}(\overrightarrow{x}), \mathbf{u}_g) \in g^{\mathbf{acc}}_{A_2}(q, q')$.

- We show below that $(\overrightarrow{x}, \overrightarrow{u}_{q_1,q',g}) \in D_{q_1,q'_1}$.
  Let $G_{\mathbf{g},\mathbf{u}_g} = (\boldsymbol{\nu}, \{u_g\}) \in g_{\kappa_2}(q, \kappa_2(q, \phi_{q_1}(\overrightarrow{x}), \mathbf{u}_g, q'), q')$ such that $\phi_{q_1}(\overrightarrow{x}) \in \boldsymbol{\nu}$.
  Such $G_{\mathbf{g},\mathbf{u}_g}$ exists since $(\phi_1(\overrightarrow{x}), \mathbf{u}_g) \in g^{\mathbf{acc}}_{A_2}(q, q')$.
  $\mathbf{dom}(\mathbf{g}) = \mathbf{dom}(\phi_{q_1}(\overrightarrow{x})) = f$ by definition.
  Let $e, e' \in \mathbf{dom}(f)$.
  Suppose $f(e) = f(e')$.
  Then $\phi_{q_1}(\overrightarrow{x})(e) = x_{f(e)} = x_{f(e')} = \phi_{q_1}(\overrightarrow{x})(e')$, and if we write,
  $g(e) = (\nu_e, \mu_e)$ and $g(e') = (\nu'_e, \mu'_e)$, this means that $x_{f(e)} \in \nu_e \cap \nu'_e$.
  Since $\mathbb{G}_M$ is a guard base, $\nu_e \cap \nu_{e'} \neq \emptyset$ implies $\nu_e \times \{\mathbf{id}\} \cap \nu_{e'} \times \{\mathbf{id}\} \neq \emptyset$
  which means that $\nu_e = \nu'_e$ are that $\mathbf{g}$ and f are coherent.
  This allows us to conclude that:
  $(\overrightarrow{x}, \overrightarrow{u}_{q_1,q',g}) \in \boldsymbol{\nu}_{q_1,q',g,\mathbf{g}} \times \{\overrightarrow{u}_{q_1,q',g}\} \subseteq D_{q_1,q'_1}$.

  By definition then, $\psi_{q_1,(q',f'_{q_1,q',g})}(\overrightarrow{x}, \overrightarrow{u}_{q_1,q',g}) = \mathbf{u}_g$.

According to proposition 3.2.2 then, Ξ is a pre-functional bisimulation from $Q_1, \mathcal{C}^n_M$ to A_2.

- Moreover, we constructed ψ such that for all $q_1 = (q, f), q'_1 = (q', f') \in Q_1$, for all $\sigma \in \Sigma$, if $\mathbf{G}_{q_1,q'_1,\sigma} = \{G_{\mathbf{g},\mathbf{u}_g} \in g_{\kappa_2}(q, \sigma, q') \mid \mathbf{g} \text{ and } f \text{ are coherent and } f' = f'_{q_1,q',g)}\}$, then:

$$\mathbf{dom}_\sigma(\psi_{q_1,q'_1}) = D_{q_1,q'_1,\sigma} = \bigcup_{G_{\mathbf{g},\mathbf{u}_g} \in \mathbf{G}_{q_1,q'_1,\sigma}} \boldsymbol{\nu}_{q_1,q',g,\mathbf{g}} \times \{\overrightarrow{u}_{q_1,q',g)}\}$$

  is decomposable on $\mathbb{G}^n_M$ by definition.
  Hence Ξ is a $\Sigma, \mathbb{G}^n_M$-compatible pre-functional bisimulation from $Q_1, \mathcal{C}^n_M$ to A_2.

- Suppose finally that A_2 is κ_2-deterministic.
  - Let $q_1, q'_1 \in Q_1$.
    By design $\mathbf{dom}_\epsilon(\psi_{q_1,q'_1}) = \emptyset$.
  - Let now $(q_1, \overrightarrow{x}) \in Q_1 \times \mathbb{C}^n_M$ and $\sigma \in \Sigma$.
    Suppose there exists $\overrightarrow{u} \in \{\mathbf{id}, \mathbf{0}\}^n$ and $q'_1 = (q', f') \in Q_1$ such that $(\overrightarrow{x}, \overrightarrow{u}) \in \mathbf{dom}_\sigma(\psi_{q_1,q'_1})$.
    Then $\mathbf{u}_g = \psi_{q_1,q'_1}(\overrightarrow{x}, \overrightarrow{u})$ is the unique update and q' is the unique state such that
    $(\phi_{q_1}(\overrightarrow{x}), \mathbf{u}_g) \in g_{\kappa_2}(q, \sigma, q')$ by κ_2-determinism of A_2.
    Let $G_{\mathbf{g},\mathbf{u}_g}$ be the unique element of $g_{\kappa_2}(q, \sigma, q')$ such that $(\phi_{q_1}(\overrightarrow{x}), \mathbf{u}_g) \in G_{\mathbf{g},\mathbf{u}_g}$.
    It exists since T^{κ_2} is compatible with the guard base $\{\mathbf{id}, \mathbf{0}\}_C$.
    Necessarily then $f' = f'_{q_1,q',g}$ and $\overrightarrow{u} = \overrightarrow{u}_{q_1,q',g}$ by definition of $\mathbf{dom}_\sigma(\psi_{q_1,q'_1})$.

    Finally notice that $(\mathbb{C}^n_M, \hookrightarrow)$ is deterministic so, *in fine*
    Ξ is a Σ-deterministic, $\Sigma, \mathbb{G}^n_M$-compatible pre-functional bisimulation
    from $Q_1, \mathcal{C}^n_M$ to A_2.

By application of proposition 4.3.5 and 4.3.4, there exists a controlled timed automata $\langle A_1, \mathbf{T}^{\kappa_1}\rangle \in \Sigma\mathbf{TA}^{|C|}_M$ which simulates simulates $\langle A_2, \mathbf{T}^{\kappa_2}\rangle$.
By definition of $\Xi^{-1}(\kappa_2)$, since $\mathbf{dom}_\epsilon(\psi_{q_1,q'_1}) = \emptyset$, $\langle A_1, \mathbf{T}^{\kappa_1}\rangle$ is without silent transitions
■

For examples of the simulations of enhanced timed automata by timed automata and for a simpler proof of this result made in the classical framework of timed automata, we advise the reader to see the work of [19]. The goal of this section is to formally ensure that we recover their result in our new framework.

6.2 From Maps to Bounded Markings

The idea of the proof is to simulate a configuration $(q, \boldsymbol{\nu})$ of an automaton on $\mathbf{M}_E S$, by a configuration $(q, \mathbf{x})$ of a automaton on $[E \times [\![1, n]\!] \rightharpoonup S]$ where n is the bound on the size of the sets associated to each elements of e. All elements in $\boldsymbol{\nu}(e)$ are distributed within the indexes $(e, 1)$ to (e, n) in $\mathbf{x}$.

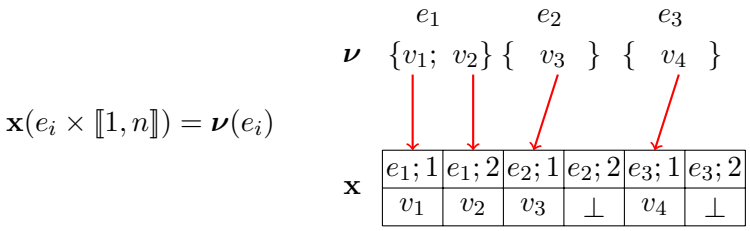


Figure 6.3: How marking $\boldsymbol{\nu}$ bounded by 2 is recovered from a partial function $\mathbf{x}$

Indirectly an update in $\mathbf{M}_E U$ repeatedly select in a set $\boldsymbol{\nu}(e)$ all values within some set (defined by the guard function), apply to them all the update in U in another set (also defined by the guard function) and finally put them in another set $\boldsymbol{\nu}(e')$. If there is no more than n of such update in U to be applied we can track one by one which update is applied to which value in which set and put in what set, and be able to reconstruct an update in U_E with that information.

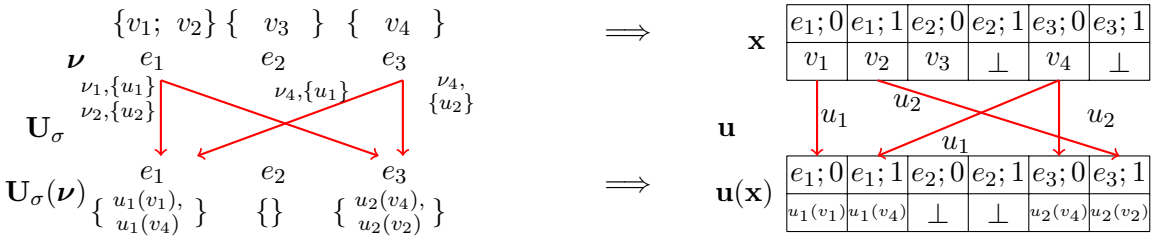


Figure 6.4: How $\mathbf{U}_\sigma$ is simulated by constructing $\mathbf{u}$

We have to begin with some definitions to formally state the hypotheses an automaton has to satisfy.

A marking $\boldsymbol{\nu} \in \mathbf{M}_E V$ is said to be *bounded* by an integer $n \in \mathbb{N}$ if for all $e \in E$, $|\boldsymbol{\nu}(e)| \leq n$. We write $\mathbf{M}_E V_{\leq n}$ for the set of markings bounded by n. We say that a set of markings is bounded by n if all of its elements are bounded by n. Given an automaton $A = \langle Q, I, T, F \rangle \in \mathbb{A}(\mathbf{M}_E S)$, we say that A is *bounded by* n if $\{\boldsymbol{\nu} \in \mathbf{M}_E V \mid \exists q \in Q, (q, \boldsymbol{\nu}) \in \mathbf{Reach}(A)\}$ is bounded by n.

Given an automaton $A = \langle Q, I, T, F \rangle \in \mathbb{A}(\mathbf{M}_E S)$, we say that A is *strongly bounded* by n if for all $(q, \boldsymbol{\nu}) \in I$, $\boldsymbol{\nu} \in \mathbf{M}_E V_{\leq n}$ and for all $q, q' \in Q$, for all $(\boldsymbol{\nu}, \mathbf{U_g}) \in g_A^{\mathbf{acc}}(q, q')$, for all $e' \in E$,

$$|\{(v, u) \in V \times U \mid \exists e \in E, v \in \boldsymbol{\nu}(e) \text{ and } (v, u) \in \mathbf{g}(e, e')\}| \leq n$$

We can easily prove by induction that an automaton strongly-bounded by n is bounded by n.

Let, for some finite alphabet Σ, $\langle A, \kappa\rangle \in \mathbb{A}(\mathbf{M}_E S, \Sigma)$, we say that $\langle A, \mathbf{T}^\kappa\rangle$ is bounded (*resp.* strongly bounded) by an integer n if A is bounded (*resp.* strongly bounded).

Finally we say that a guard base G is *strongly disjoint* if and only if for all $(\nu, \mu), (\nu', \mu') \in G$, $\nu = \nu'$ or $\nu \cap \nu' = \emptyset$.

Theorem 6.2.1. *Let Σ be a finite alphabet and $n \in \mathbb{N}$. Let $\mathcal{S}_G$ be a guarded deterministic timed structure, with G* finite*,* spanning *(definition 2.3.3) and* strongly disjoint*. Let $\langle A_2, \mathbf{T}_2^\kappa\rangle \in \mathbb{A}(\mathbf{M}_E S_G, \Sigma)$, strongly bounded by n and without silent transitions.*

There exists a deterministic controlled automaton $\langle A_1, \mathbf{T}_1^\kappa\rangle \in \mathbb{A}(\mathcal{S}_G^{E\times[\![1,n]\!]}, \Sigma)$ without silent transitions which simulates $\langle A_2, \mathbf{T}_2^\kappa\rangle$.

Proof.

Let's write $A_2 = \langle Q_2, I_2, T_2, F_2\rangle$ and $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$.
Let $Q_1 = Q_2 = Q$, we are going to construct $\Xi = (\zeta, \phi, \psi)$ candidate to be a $\Sigma, [E \times [\![1, n]\!] \rightharpoonup G]$-compatible, Σ-deterministic pre-functional bisimulation from $Q_1, \mathcal{S}^{E\times[\![1,n]\!]}$ to A_2.

- Let $\zeta = \mathbf{id}_Q$. Trivially

Lemma 6.2.2. *ζ is surjective.*

- For all $q \in Q_1$, let $\phi_q : [E \times [\![1, n]\!] \rightharpoonup V] \to \mathbf{M}_E V$ mapping any $\mathbf{x}$ to

$$\begin{aligned}\phi_q(\mathbf{x}) : E &\rightharpoonup \mathcal{P}(V)\\ e &\mapsto \mathbf{x}(\{e\} \times [\![1, n]\!])\end{aligned}$$

We clearly have $\phi_q(V^{E\times[\![1,n]\!]}) = \mathbf{M}_E V_{\leq n}$.

Lemma 6.2.3. *For all $q \in Q$, ϕ_q is a functional bisimulation of timed domains. For all $q_2, \boldsymbol{\nu}_2 \in I_2$, exists $\mathbf{x}_1 \in [E \times [\![1, n]\!] \rightharpoonup V]$ such that $\phi_q(\mathbf{x}_1) = \boldsymbol{\nu}_2$.*

Proof.

- Let $q \in Q$. Since $\mathcal{S}$ is deterministic, both $[E \rightharpoonup \mathcal{S}]$ and $\mathbf{M}_E S$ are deterministic and for all $d \in \mathbb{R}_{\geq 0}$,

$$\phi_q(\mathbf{x} \oplus d) = \phi_q(\mathbf{x}) \oplus d$$

  by definition. Hence ϕ_q is a functional bisimulation of timed domains.

- Let now $(q_2, \boldsymbol{\nu}_2 \in I_2)$.
  We choose $\mathbf{x}_1$ within $\phi_q^{-1}(\boldsymbol{\nu}_2)$, and $q_1 = q_2$.

■

We need to prove several intermediate results before being able to define ϕ. First, we'll try to outline the guards which will enter in the definition of the domain of ψ. Then we'll discuss how to construct an update in U_E which simulates an update $\mathbf{U}_\mathbf{g}$.

► Fix a guard $(G_{\rho,\mathbf{g},\rho'}, \mathbf{U_g}) \in \mathbf{M}_E G$ and let's write for all $e, e' \in E$,

$$\mathbf{g}(e,e') = \bigcup_{k=1}^{m_{e,e'}} \nu^k_{e,e'} \times \mu^k_{e,e'}$$

such that for all $e, e' \in E$, for all $k \neq j \in [\![1, m_{e,e'}]\!]$,

- $\nu^k_{e,e'} \neq \nu^j_{e,e'}$
- $\mu^k_{e,e'} = \mu_1 \cup \cdots \cup \mu_l$ with for all $1 \leq i \leq l$, $(\nu^k_{e,e'}, \mu_i) \in G$.

Let $q, q' \in Q$ and $\mathbf{x} \in \phi_q^{-1}(G_{\rho,\mathbf{g},\rho'})$.
We write $\boldsymbol{\nu} = \phi_q(\mathbf{x})$.
Suppose $(\boldsymbol{\nu}, \mathbf{U_g}) \in g^{\mathbf{acc}}_{A_2}(q, q')$, we say that $\mathbf{x}$ is *ϕ_q-accessible*.

► By hypothesis we know since A_2 is strongly bounded by n that for all $e' \in E$, $|F_{\mathbf{x},e'}| \leq n$, where $F_{\mathbf{x},e'} = \{(v,u) \in V \times U \mid \exists e \in E, v \in \boldsymbol{\nu}(e) \text{ and } (v,u) \in \mathbf{g}(e,e')\}$.
Let for all $e' \in E$, $\mathbf{ind}_{\mathbf{x},e'}$ be an injection from $F_{\mathbf{x},e'}$ to $[\![1,n]\!]$.
We define for all $e' \in E$, $f^0_{\mathbf{x},e'} : F_{\mathbf{x},e'} \to E \times [\![1,n]\!]$ such that
for all $(v,u) \in F_{e'}$, $\mathbf{x}(f^0_{\mathbf{x},e'}(v,u)) = v$.
Such function $f^0_{\mathbf{x},e'}$ exists by definition of ϕ_q.
We define then $f_{\mathbf{x}} : E \times [\![1,n]\!] \rightharpoonup U \times E \times [\![1,n]\!]$ such that
$\mathbf{dom}(f_{\mathbf{x}}) = \{(e',j) \in E \times [\![1,n]\!] \mid j \in \mathbf{im}(\mathbf{ind}_{\mathbf{x},e'})\}$ and
for all $(e',j) \in \mathbf{dom}(f_{\mathbf{x}})$, $f_{\mathbf{x}}(e',j) = (u, f^0_{\mathbf{x},e'}(v,u))$ with $\mathbf{ind}_{\mathbf{x},e'}(v,u) = j$.

Because G is a spanning guard base, we know that exists $\mathbf{h_x}$ such that $\mathbf{u}_{f_\mathbf{x}}, \mathbf{h_x}$ is compatible with G and $(\mathbf{x}, \mathbf{u}_{f_\mathbf{x}}) \in G_{\mathbf{h_x}, \mathbf{u}_{f_\mathbf{x}}} \times \{u_{f_\mathbf{x}}\}$.

Let's write for all $(e,i) \in E \times [\![1,n]\!]$,

$$\mathbf{h_x}(e,i) = (\nu_{e,i}, \mu_{e,i})$$

Lemma 6.2.4. *For all* $\mathbf{x}$*,* ϕ_q*-accessible, for all* $\mathbf{y} \in G_{\mathbf{h_x},\mathbf{u}_{f_\mathbf{x}}}$

$$\phi_q(\mathbf{y}) \in G_{\rho,\mathbf{g},\rho'} \text{ and } \phi_{q'}(\mathbf{u}_{f_\mathbf{x}}(\mathbf{y})) = \mathbf{U_g}(\phi_q(\mathbf{y}))$$

Proof.

Let $\mathbf{x}$, ϕ_q-accessible and $\mathbf{y} \in G_{\mathbf{h_x},\mathbf{u}_{f_\mathbf{x}}}$.

- For all $e' \in E$, by definition :
  $\phi_{q'}(u_{f_\mathbf{x}}(\mathbf{y}))(e') = \{\mathbf{u_x}(\mathbf{y})(e', j), j \in [\![1, n]\!]\}$ and
  $\mathbf{U_g}(\phi_{q'}(\mathbf{y}))(e') = \{u(v), (u, v) \in F_{\mathbf{y},e'}\}$.
- We prove below that for all $e' \in E$,
  $\{\mathbf{u_x}(\mathbf{y})(e', j), j \in [\![1, n]\!]\} = \{u(v), (u, v) \in F_{\mathbf{y},e'}\}$.
  Let $e' \in E$.
  - Let $j \in [\![1, n]\!]$ and suppose $(e', j) \in \mathbf{dom}(\mathbf{u_x}(\mathbf{y}))$.
    Then $(e', j) \in \mathbf{dom}(f_\mathbf{x})$ and $f_\mathbf{x}(e', j) = (u, f^0_{\mathbf{x},e'}(u, v))$ with
    $\mathbf{ind}^{-1}_{\mathbf{x},e'}(j) = (u, v)$, by definition.
    Let $(e, i) = f^0_{\mathbf{x},e'}(u, v)$.
    By definition $v = \mathbf{x}(e, i) \in \nu_{e,i}$ and exists $k \in [\![1, m_{e,e'}]\!]$ such that
    $(v, u) \in \nu^k_{e,e'} \times \mu^k_{e,e'}$.
    Since G is strongly disjoint this means that $\nu_{e,i} = \nu^k_{e,e'}$.
    We know by hypothesis on $\mathbf{y}$ that $\mathbf{y}(e, i) \in \nu_{e,i}$.
    Hence $(\mathbf{y}(e, i), u) \in \nu^k_{e,e'} \times \mu^k_{e,e'}$ and $(\mathbf{y}(e, i), u) \in F_{\mathbf{y},e'}$.
    Since $\mathbf{u_x}(\mathbf{y})(e', j) = u(\mathbf{y}(f^0_{\mathbf{x},e'}(u, v))) = u(\mathbf{y}(e, i))$,
    $\mathbf{u_x}(\mathbf{y})(e', j) \in \{u(v), (u, v) \in F_{\mathbf{y},e'}\}$.
  - Let $(u, v) \in F_{\mathbf{y},e'}$.
    By definition exists $(e, i) \in E \times [\![1, n]\!]$ such that $\mathbf{y}(e, i) = v$.
    This means that $v \in \nu_{e,i}$.
    Exists also $k \in [\![1, m_{e,e'}]\!]$ such that $(v, u) \in \nu^k_{e,e'} \times \mu^k_{e,e'}$.
    Since G is strongly disjoint this means that $\nu_{e,i} = \nu^k_{e,e'}$.
    Let $v' = \mathbf{x}(e, i)$, well-defined since $\mathbf{dom}(\mathbf{x}) = \mathbf{dom}(\mathbf{h_x}) = \mathbf{dom}(\mathbf{y})$.
    $(v', u) \in \nu_{e,i} \times \mu^k_{e,e'} = \nu^k_{e,e'} \times \mu^k_{e,e'}$.
    This means that $(v', u) \in F_{\mathbf{x},e'}$, let $j = \mathbf{ind}_{\mathbf{x},e'}(v', u))$.
    $f_\mathbf{x}(e', j) = (u, f^0_{\mathbf{x},e'}(v', u)) = (u, (e, i))$.
    So $u(v) = u(\mathbf{y}(e, i)) = \mathbf{u}_{f_\mathbf{x}}(\mathbf{y})(e', j) \in \{\mathbf{u_x}(\mathbf{y})(e', j), j \in [\![1, n]\!]\}$.

  This proves that $\phi_{q'}(\mathbf{u}_{f_\mathbf{x}}(\mathbf{y})) = \mathbf{U_g}(\phi_q(\mathbf{y}))$.

- Notice that since $\mathbf{dom}(\mathbf{x}) = \mathbf{dom}(\mathbf{h}_x) = \mathbf{dom}(\mathbf{y})$,
  by definition of ϕ_q, $\mathbf{supp}(\phi_q(\mathbf{y})) = \mathbf{supp}(\phi_q(\mathbf{x})) = \rho$.
  And according to what we just proved:

$$\begin{aligned}\mathbf{supp}(\mathbf{U_g}(\phi_q(\mathbf{y}))) &= \mathbf{supp}(\phi_{q'}(\mathbf{u}_{f_\mathbf{x}}(\mathbf{y}))) \\ &= \mathbf{supp}(\phi_{q'}(\mathbf{u}_{f_\mathbf{x}}(\mathbf{y}))) \\ &= \{e' \in E, \exists j \in [\![1, n]\!], (e', j) \in = \mathbf{dom}(f_\mathbf{x})\} \\ &= \mathbf{supp}(\phi_{q'}(\mathbf{u}_{f_\mathbf{x}}(\mathbf{x}))) \\ &= \rho'\end{aligned}$$

  So $\phi_q(\mathbf{y}) \in G_{\rho,\mathbf{g},\rho'}$.

■

We are ready now to define ψ. Let $q, q' \in Q$ and $\sigma \in \Sigma$.

► We define:

$$D_{q,\sigma,q'} = \bigcup_{(G_{\rho,\mathbf{g},\rho'},\mathbf{U}_\sigma)\in g_{\kappa_2}(q,\sigma,q')} \quad \bigcup_{\mathbf{x}\in\phi_q^{-1}(G_{\rho,\mathbf{g},\rho'}) \mid (\phi_q(\mathbf{x}),\mathbf{U_g})\in g_{A_2}^{\mathbf{acc}}(q,q')} G_{\mathbf{h_x},\mathbf{u}_{f_\mathbf{x}}} \times \{\mathbf{u}_{f_\mathbf{x}}\}$$

and

$$D_{q,q'} = \bigcup_{\sigma\in\Sigma} D_{q,\sigma,q'}$$

► We define then:

$$\psi_{q,q'} : [E \times [\![1,n]\!] \rightharpoonup V] \times U_E \rightharpoonup \mathbf{M}_E U$$
$$(\mathbf{x}, \mathbf{u}_{f_\mathbf{x}}) \mapsto \mathbf{U_g} \text{ if } (\mathbf{x}, \mathbf{u}_{f_\mathbf{x}}) \in D_{q,\sigma,q'} \text{ and } (\mathbf{x}, \mathbf{u}_{f_\mathbf{x}}) \in G_{\mathbf{h_x},\mathbf{u}_{f_\mathbf{x}}} \times \{\mathbf{u}_{f_{fx}}\}$$

Let's prove below that $\Xi = (\zeta, \phi, \psi)$ with $\phi = (\phi_q)_{q\in Q}$ and $\psi = (\psi_{q,q'})_{q,q'\in Q_1}$ is a pre-functional bisimulation from Q,$[E[\![1,n]\!] \rightharpoonup S]$ to A_2.

We check it verifies conditions (A) to (D) of proposition 3.2.2.

(A) This is lemma 6.2.2.

(B) This is lemma 6.2.3.

(C) Let $q, q' \in Q_1$, let $(\mathbf{y}, \mathbf{u}) \in \mathbf{dom}(\psi_{q,q'})$.

(C.1) We know that exists $\sigma \in \Sigma$, $(G_{\rho,\mathbf{g},\rho'}, U_\mathbf{g}) \in g_{\kappa_2}(q,\sigma,q')$, and $\mathbf{x} \in \phi_q^{-1}(G_{\rho,\mathbf{g},\rho'})$ such that $(\phi_q(\mathbf{x}), \mathbf{U_g}) \in g_{A_2}^{\mathbf{acc}}(q,q')$ and $(\mathbf{y}, \mathbf{u}) \in G_{\mathbf{h_x},\mathbf{u}_{f_\mathbf{x}}} \times \{\mathbf{u}_{f_\mathbf{x}}\}$. Then $\mathbf{u} = \mathbf{u}_{f_\mathbf{x}}$, $\psi_{q,q'}(\mathbf{x}, \mathbf{u}) = \mathbf{U_g}$ and according to lemma 6.2.4:

$$\phi_{q'}(\mathbf{u})(\mathbf{y})) = \psi_{q,q'}(\mathbf{x}, \mathbf{u})(\phi_q(\mathbf{y}))$$

(C.2) According to lemma 6.2.4, we know that

$$(\phi_q(\mathbf{y}), U_\mathbf{g}) \in G_{\rho,\mathbf{g},\rho'} \times \{U_\mathbf{g}\} \subseteq g_{\kappa_2}(q,\sigma,q') \subseteq g_{A_2}(q,q')$$

which is enough to conclude.

(D) Let now $q \in Q$, $\mathbf{x} \in \mathbf{dom}(\phi_q)$, $q' \in Q_2$ and $\mathbf{U_g} \in \mathbf{M}_E U$.
Suppose $(\phi_q(\mathbf{x}), \mathbf{U_g}) \in g_{A_2}^{\mathbf{acc}}(q,q')$.
Because $\mathbf{T}_2^\kappa$ is compatible with $\mathbf{M}_E G$, we know that
exists $\sigma \in \Sigma$ and $(G_{\rho,\mathbf{g},\rho'}, \mathbf{U_g}) \in g_{\kappa_2}(q,\sigma,q')$ such that $(\phi_q(\mathbf{x}), \mathbf{U_g}) \in (G_{\rho,\mathbf{g},rho'}, \mathbf{U_g})$.
By definition of $D_{q,\sigma,q'}$, this implies that $G_{\mathbf{h_x},\mathbf{u}_{f_\mathbf{x}}} \subseteq \mathbf{dom}(\phi_{q,q'})$,
which means that $\psi_{q,q'}(\mathbf{x}, \mathbf{u}_{f_\mathbf{x}}) = \mathbf{U_g}$.

According to proposition 3.2.2 then, Ξ is a pre-functional bisimulation from $Q_1, \mathcal{C}_M^n$ to A_2.

- Moreover, we constructed ψ such that for all $q, q' \in Q$, for all $\sigma \in \Sigma$, $\mathbf{dom}_\sigma(\psi_{q,q'}) = D_{q,\sigma,q'}$ is a union of elements of $[E \rightharpoonup G]$.
  We just have to notice that G being finite by hypothesis, $[E \rightharpoonup G]$ is finite and $D_{q,\sigma,q'}$ is then a finite union of elements of $[E \rightharpoonup G]$.
  So $\mathbf{dom}_\sigma(\psi_{q,q'})$ is decomposable on $[E \rightharpoonup G]$,
  Hence Ξ is a $\Sigma, [E \rightharpoonup G]$-compatible pre-functional bisimulation from $Q, [E \rightharpoonup G]$ to A_2.

- Suppose finally that A_2 is κ_2-deterministic.
  - Let $q, q' \in Q_1$.
    By design $\mathbf{dom}_\epsilon(\psi_{q,q'}) = \emptyset$.
  - Let now $(q, \mathbf{x}) \in Q \times [E \rightharpoonup V]$ and $\sigma \in \Sigma$.
    Suppose exists $\mathbf{u} \in U_E$ and $q' \in Q$ such that $(\mathbf{x}, \mathbf{u}) \in \mathbf{dom}_\sigma(\psi_{q,q'})$.
    Then $\mathbf{U_g} = \psi_{q,q'}(\mathbf{x}, \mathbf{u})$ is the unique update and q' is the unique state such that $(\phi_q(\mathbf{x}), \mathbf{U_g}) \in g_{\kappa_2}(q, \sigma, q')$ by κ_2-determinism of A_2.
    By definition then $\mathbf{u}$ have to be equal to $\mathbf{u}_{f_\mathbf{x}}$.
    And $(\mathbf{u}_{f_\mathbf{x}}, q')$ is the unique pair such that $(\mathbf{x}, \mathbf{u}_{f_\mathbf{x}}) \in \mathbf{dom}_\sigma(()\psi_{q,q'})$.

    Finally notice that by hypothesis $\mathcal{S}$ is deterministic so, *in fine*
    Ξ is a Σ-deterministic, $\Sigma, \mathbb{G}_M^n$-compatible pre-functional bisimulation
    from $Q_1, \mathcal{C}_M^n$ to A_2.

By application of proposition 4.3.5 and 4.3.4, exists a controlled timed automata $\langle A_1, \mathbf{T}^{\kappa_1} \rangle \in \mathbb{A}([E \rightharpoonup S], \Sigma)$ which simulates simulates $\langle A_2, \mathbf{T}^{\kappa_2} \rangle$.
By definition of $\Xi^{-1}(\kappa_2)$, since $\mathbf{dom}_\epsilon(\psi_{q_1,q_1'}) = \emptyset$, $\langle A_1, \mathbf{T}^{\kappa_1} \rangle$ is without silent transitions ■

We give below an example of the construction of an enhanced timed automata simulating the powerset automaton of the timed automata $\mathcal{A}$ of figure 6.5 below.

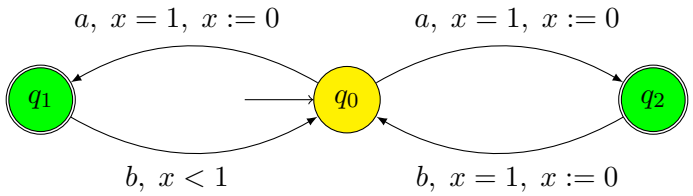
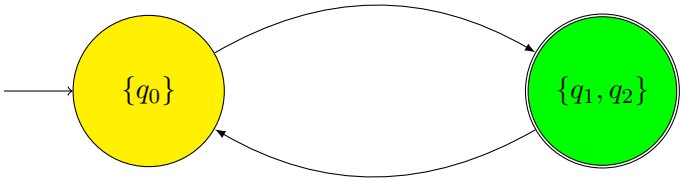


Figure 6.5: Timed Automaton $\mathcal{A}$

The graph of the powerset automaton of $\mathcal{A}$ constructed following the method of chapter 5 is given in figure 6.6 below.

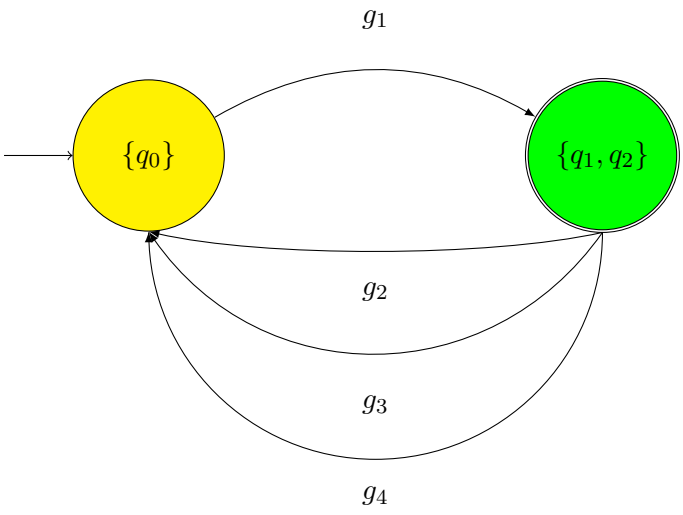
This powerset automaton is strongly bounded by 2, therefore we can apply our construction to create a deterministic enhanced timed automaton which simulates the powerset automaton of $\mathcal{A}$, hence simulates $\mathcal{A}$. We give a representation of the enhanced automaton obtained by this technique in figure 6.7 below. According to the construction we've described in this section, the enhanced timed automaton we get has six clocks, $x^1_{q_0}, x^2_{q_0}, x^1_{q_1}, x^2_{q_1}, x^1_{q_2}$ and $x^2_{q_2}$. The initial configuration is $(q_0, [x^1_{q_0} \mapsto 0])$ (meaning that all other clocks are undefined).

$$\mathtt{a}, q_0 \cap \{1\} \neq \emptyset$$
$$\left[\begin{array}{lllll} q_0: & x=1 & \Longrightarrow & 0 \to q_1 \\ q_1: & x=1 & \Longrightarrow & 0 \to q_2 \end{array}\right]$$



$$\mathtt{b}, q1 \cap [0,1[\cup q_2 \cap \{1\} \neq \emptyset$$
$$\left[\begin{array}{lllll} q_1: & x<1 & \Longrightarrow & x \to q_0 \\ q_2: & x=1 & \Longrightarrow & 0 \to q_0 \end{array}\right]$$

Figure 6.6: Powerset Automaton of $\mathcal{A}$



$g_1:$ $\mathtt{a},\ x^1_{q_0} = 1 \vee x^2_{q_0} = 1, x^1_{q_0} := \bot, x^2_{q_0} := \bot, x^1_{q_1} := 0, x^2_{q_2} := 0$

$g_2:$ $\mathtt{b},\ x^1_{q_1} < 1 \wedge x^1_{q_2} \neq 1, x^1_{q_1} := \bot, x^1_{q_2} := \bot, x^1_{q_0} := x^1_{q_1}$

$g_3:$ $\mathtt{b},\ x^1_{q_1} >= 1 \wedge x^1_{q_2} = 1, x^1_{q_1} := \bot, x^1_{q_2} := \bot, x^1_{q_0} := 0$

$g_4:$ $\mathtt{b},\ x^1_{q_1} < 1 \wedge x^1_{q_2} = 1, x^1_{q_1} := \bot, x^1_{q_2} := \bot, x^1_{q_0} := x^1_{q_1}, x^2_{q_0} := 0$

Figure 6.7: Deterministic Enhanced Timed Automaton Simulating $\mathcal{A}$

Chapter 7

Application to Classes of Timed Automata

Recovering Determinizable Classes

In chapter 5 we already recalled that a formal model is usually called determinizable if it is closed by determinization. When this property is proved to be false the considered solution is to expose subclasses of the said models, either stable by determinization, or determinizable within the class.

For quantitative systems, which are rarely closed by determinization, there has been a certain amount of work done to find determinizable subclasses. *Visibly pushdown automata* are proved to be stable by determinization [7]. There are semi-algorithms which provide when they terminate, a determinized weighted automata [39] [43]. For timed automata, several subclasses have been proved to be stable w.r.t. determinization [5] [45] ; also several algorithm allow determinization of some timed automata which respects given properties [10] [15] [48].

Our results on the determinization of automata on timed systems exposed on chapter 5 and not only useful for diagnoser construction but can also be used in the search of classes closed by determinization. We show this can be done in the last two chapters of this part of the thesis. Chapter 8 will focus on the classes we could classify as *input determined*, whereas this chapter recovers the other determinization results known about timed automata :

Section 7.1 : First we prove that for a timed automaton which admits a bounded powerset automaton, we can construct an equivalent deterministic timed automaton. This result is to be put in relation with the work done in [10].

Section 7.2 : In the second section we discuss the application to our results to strongly non-Zeno timed automata [10], 0-bounded timed automata [45], integer reset timed automata [49] and perturbed timed automata [6]. We also study a new class of timed automata where at some point different clock values do not imply different behaviors. We call them finally imprecise timed automata.

Thanks to our more general new approach we can put those results in perspective and sketch a common property for all those classes. Our conclusion on this matter is that, even if the variety, not only of classes but also of determinization notions considered, harden the definition of a clear and formal common property, it appears that we can establish the link, for an automaton on a timed structure, between the existence of a determinized automaton on the same time structure, and the ability to bound in some sense its powerset automaton. The intuitive idea we emitted above relies on the similarity we found in the proofs in our framework of those different results.

7.1 Application to Timed Automata

We've already discussed how theorem 5.1.6 can be set for timed automaton *without silent transitions* and said a few words about the powerset automaton of a timed automaton. In this section, we look at how theorems of chapter 6 can be applied to *strongly determinize* a timed automaton (i.e. construct a deterministic timed automaton which simulates the original).

Theorem 7.1.1. *Let Σ be a finite alphabet and $M, B, n \in \mathbb{N}$ with $n > 0$. Let $\langle A, \mathbf{T}^\kappa \rangle \in \Sigma\mathbf{TA}_M^n$ without silent transitions, such that $A = \langle Q, I, T, F \rangle$.*

If $\mathbf{D}_\kappa A$ is bounded by B then there exists $\langle A', \mathbf{T}^{\kappa'} \rangle \in \Sigma\mathbf{TA}_M^{2nB|Q|^2}$ deterministic and without silent transitions such that $\mathcal{L}_{\kappa'}(A') = \mathcal{L}_\kappa(A)$.

For the proof of this theorem, we advise the reader to get familiar again with the definitions given in section 6.2. In particular those of *spanning guards* page 19, of *strongly disjoint guards* page 67 and of *strong boundedness* page 66.

Proof.

Recall that $A \in \mathbb{A}(\mathcal{C}_M^n)$.
This implies that $\mathbf{D}_\kappa A$ is in $\mathbb{A}(\mathbf{M}_Q \mathcal{C}_M^n)$ by theorem 5.1.6.
$\mathbb{G}_M^n$ is finite, spanning and strongly disjoint by definition.

- We prove below that $\mathbf{D}_\kappa A$ is strongly bounded by $2|Q|B$.
  By hypothesis, $\mathbf{D}_\kappa A$ is bounded by B.
  We are sure then that for all $(\rho, \boldsymbol{\nu}) \in \mathbf{D}_\kappa I$, $\boldsymbol{\nu} \in \mathbf{M}_Q(\mathbb{C}_M^n)_{\leq 2|Q|B}$.
  Consider now $\rho, \rho' \in \mathcal{P}(Q)$, $(\boldsymbol{\nu}, \mathbf{U}_\sigma) \in g^{\mathbf{acc}}_{\mathbf{D}_\kappa A}(\rho, \rho')$ and $q' \in Q$.
  $|\{(v,u) \in \mathbb{C}_M^n \times \{\mathbf{id}, \mathbf{0}\} | \exists q \in Q, v \in \boldsymbol{\nu}(q) \text{ and } (v,u) \in g_{\mathbf{D}\kappa}(q, \sigma, q')\}|$
  $\leq |\{(v,u) \in \mathbb{C}_M^n \times \{\mathbf{id}, \mathbf{0}\} | \exists q \in Q, v \in \boldsymbol{\nu}(q)\}|$
  $\leq |\{(v,u) \in \boldsymbol{\nu}(q) \times \{\mathbf{id}, \mathbf{0}\}, q \in Q\}$
  $\leq 2\Sigma_{q \in Q} |\boldsymbol{\nu}(q)|$
  $\leq 2|Q||B|$ by hypothesis, because, since $\boldsymbol{\nu}$ is reachable, $\boldsymbol{\nu}$ is bounded.
  Which proves that $\mathbf{D}_\kappa A$ is strongly bounded by $2|Q|B$.
  According to theorem 6.2.1, there exist $\langle A'', \kappa'' \rangle \in \mathbb{A}((\mathcal{C}_M^n)^{Q \times [\![1,2|Q|B]\!]}, \Sigma)$
  without silent transitions which simulates $\langle \mathbf{D}_\kappa A, \mathbf{T}^{\mathbf{D}\kappa} \rangle$.
  This means that A'' is κ''-deterministic and that $\mathcal{L}_{\kappa'}(A'') = \mathcal{L}_{\mathbf{D}\kappa}(\mathbf{D}_\kappa A) = \mathcal{L}_\kappa(A)$.
  As noticed in remark 2.3.3, it is equivalent to say that $A'' \in \mathbb{A}(\mathcal{C}_M^{Q \times [\![1,2|Q|B]\!] \times [\![1,n]\!]})$.
  In other words $\langle A'', \mathbf{T}^{\kappa''} \rangle \in \Sigma\mathbf{TA}_M^{Q \times [\![1,2|Q|B]\!] \times [\![1,n]\!]}$.
  Applying theorem 6.1.1, we get the existence of $\langle A', \mathbf{T}^{\kappa'} \rangle \in \Sigma\mathbf{TA}_M^{|Q| \times 2|Q|B \times n})$
  without silent transitions which simulates $\langle A'', \mathbf{T}^{\kappa''} \rangle$.
  This means that A' is κ'-deterministic and that $\mathcal{L}_{\kappa'}(A') = \mathcal{L}_{\kappa''}(A'') = \mathcal{L}_\kappa(A)$.

■

This last theorem is in close relation to the result of [10] . We find in both results and proofs, this idea that the set of different values that can be reached following one run has to be bounded. In [10] this allows the authors to fix a finite set of clocks and, by permutation of the clocks, to always be able to represent all possible clocks reached with only a finite set. In our work boundedness allows us to simulate the powerset automaton with a automaton using maps between finite sets of clocks.

7.2 About Other Determinizable Classes

We do not know if all timed automata verifying hypotheses of theorem 7.1.1 satisfy the hypothesis required in [10] and conversely. We didn't try to compare them because it would take a lot of effort to make the work of [10] enter in our formalism to be able to formally compare the hypotheses.

However, we think the statement of theorem 7.1.1 might provide us easier hypotheses to verify on a timed automaton.

Strongly non-Zeno timed automata. As an example consider all n-clock timed automaton bounded by M such that on every transitions we have as constraint a conjunction with the constraint $z >= \delta$ and among the updates $z := 0$, for some special clock z and some value $\delta > 0$. We know that there will never be more than $\lceil \frac{M+1}{\delta} \rceil$ clocks with different values since the distance between their values have to be greater than δ and one cannot fit more than $\lceil \frac{M+1}{\delta} \rceil$ different values in $\mathbb{C}_M = [0; M] \cup \{\infty\}$. This means that the powerset automaton of such timed automata will be bounded by $\lceil \frac{M+1}{\delta} \rceil$ and thus that theorem 7.1.1 is applicable.

This class of timed automata is therefore strongly determinizable – determinizable in the classical sense. This result was already deduced in [10].

0-bounded timed automata. Another example is the case of the class described in [45] of timed automata with constant only 0. By the choice of modeling the set of clock values by $\mathbb{C}_0 = [0; 0] \uplus \{\infty\}$ we get immediately that the powerset automaton of a timed automaton with constant only 0 is bounded by 2. Theorem 7.1.1 is applicable and this class of timed automata is strongly determinizable.

Integer Reset Timed Automata. Finally, we can also obtain strong determinization of the class of integer reset timed automata (IRTA) [49] with this reasoning. Indeed in an M-bounded n-clock IRTA, there can be a reset update on a clock if and only if there is at least one constraint of the guard of the type $x = k$ for some clock x and some integer $k \leq M$. We can prove by a simple induction on the size of the run, that any time a clock is reset along a run of a duration d, d have to be in $\mathbb{N}$ so the valuation we take the transition on maps every clock to an integer.

In the powerset automaton, this also means that a reset can be simulated only after a integer duration, where all values in the markings map every clock to an integer. One can easily notice that the only way to increase the number of different valuations appearing in a reachable marking of the powerset automaton is to simulate at least one reset (while keeping the initial value we reset). This means that each time we are in the situation to add a value in a marking, all other values are within $(\llbracket 1, M \rrbracket \cup \{\infty\})^n$. Hence the size of such marking can't exceed $(M + 2)^{n|Q|}$. Since this is the only way to increase the size of a marking, this means that the power set automaton of an M-bounded n-clock IRTA is bounded by $(M + 2)^{n|Q|}$ and theorem 7.1.1 is applicable which makes this class of timed automata strongly determinizable.

Finally imprecise timed automata. Theorem 7.1.1 is a sufficient condition of strong determinization but we know that this is not a necessary condition. We give a counter example in figure 7.1 and 7.2 of a timed automaton which is strongly determinizable but does not directly satisfies the hypotheses of theorem 7.1.1. Indeed the automaton in figure 7.1 recognizes the language of all timed sequences of a satifying : exists one a such that no a is done exactly 1 time unit after. The powerset automaton of this timed automaton can reach any configuration $(\{q_1, q_2, q_3\}, \boldsymbol{\nu})$ where $\boldsymbol{\nu}(q_1) = \{d\}$, $\boldsymbol{\nu}(q_2)$ is a finite set included in $\mathbb{C}_1 \setminus \{1\}$ and $\boldsymbol{\nu}(q_3) = \boldsymbol{\nu}(q_2)$.

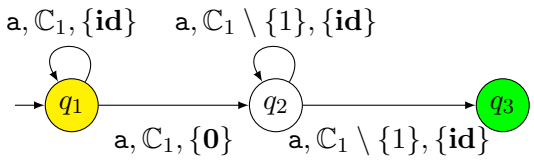


Figure 7.1: A determinizable TA ...

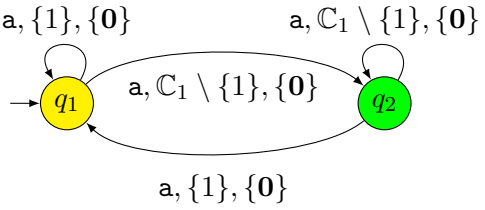


Figure 7.2: ... and its determinized TA.

Even though the marking of q_2 have no bound, the only important information to do an action a is knowing that there exists at least one clock which is not equal to 1 in q_2. This can be intuitively achieved by only keeping the information of two different clocks in q_2 (such that if one is equal to 1 we know the other one is not). Hence even if we may have an unbounded amount of information stored in q_2, most of it is superfluous and is not needed in the construction of a determinized automaton.

This scenario may concern a whole class of timed automata we call *finally imprecise timed automata*.

Definition 7.2.1. *Let Σ be a finite alphabet, $M \in \mathbb{N}$ and $n \in \mathbb{N}_{>0}$. Let $\langle A, \mathbf{T}^\kappa \rangle \in \Sigma\mathbf{TA}^n_M$ be a timed automaton such that $A = \langle Q, I, T, F \rangle$.*

We say that $q \in Q$ is imprecise *if for all $\overrightarrow{x_1}, \overrightarrow{x_2} \in \mathcal{I}^n_M$ [1] the languages recognized in A from $(q, \overrightarrow{x_1})$ and $(q, \overrightarrow{x_2})$ are the same, i.e.*

$$\mathcal{L}(\mathbf{T}^\kappa, (q, \overrightarrow{x_1}), F \times \mathbb{C}^n_M) = \mathcal{L}(\mathbf{T}^\kappa, (q, \overrightarrow{x_2}), F \times \mathbb{C}^n_M)$$

The principle of an imprecise state is that it doesn't matter if you reach it with some value or another. This means that all the variety of values we can reach the state with can be ignored and reduced to just one value per guards, cutting, in fact, a lot of possibilities. We ensure then that all reachable states are imprecise; we could bound from this point all the possibilities to the number of states times the number of guards, making the class strongly determinizable without effectively rendering the powerset automaton bounded.

Definition 7.2.2. *Let Σ be a finite alphabet, $M \in \mathbb{N}$ and $n \in \mathbb{N}_{>0}$. We say that a timed automaton $\langle A, \mathbf{T}^\kappa \rangle \in \Sigma\mathbf{TA}^n_M$ is* finally imprecise *if there exists an integer m such that all states reached with a run of length greater than m is imprecise.*

In the context of [20] we proved that this kind of automata can be strongly determinized. In the setting of this thesis, the proof of strong determinizability of the class of finally imprecise timed automata could be made by an adaptation of the proof of theorem 6.2.1. We just give some hint below to avoid an additional technical construction of a functional bisimulation.

One just need, when constructing the update $\mathbf{u}_{f_\mathbf{x}}$ associated to $(\phi_q(\mathbf{x}), \mathbf{U_g})$ to pick only one element by (product of) interval to be updated and mapped to the resulting partial function. In the setting of the proof of theorem 6.2.1, if there exist $q \in Q$ and $k \in [\![1, m_{e,e'}]\!]$ such that v and v' are in $\boldsymbol{\nu}(e) \cap \boldsymbol{\nu}^k_{e,e'}$, when constructing $\mathbf{u}_{f_\mathbf{x}}$ we only implement the update made on v and forget about those made on v'. This way we can remain in a finite setting without loss of expressiveness.

[1] $\mathcal{I}_M$ is the set of all real intervals bounded by two consecutive integers c.f. 2.3.1

1-bounded 1-clock perturbed timed automata. At last, we say a few words about the work of [6] on one-clock perturbed timed automata. We've defined on page 17 the perturbed clock structure. A one-clock perturbed timed automaton can be easily defined in our formalism as an automaton on the perturbed clock structure, guarded by $\mathbb{G}_M$.

Our first remark is that the result of determinization proposed in [6] differs from all the results we discussed above. As discussed in the introduction of chapter 5, determinization usually stands for the construction, for any automaton of a *particular class*, of a deterministic automaton *within the same class*, which recognizes the same language. In [6] the authors prove that for any *one-clock perturbed timed automata*, one can construct a deterministic *timed automaton* which recognizes the same language, which are two models of distinct classes. On one side, a timed automata is just a particular case of perturbed timed automata; however, we can see in our formalism that the nature of the timed structure both models are defined on are really different. This is the fundamental reason we can't obtain determinizability of perturbed timed automata as a consequence of the work we've done in this part.

The perturbed clock structure is a non-deterministic timed structure contrarily to the clock structure. It doesn't prevent us to construct the powerset automaton of a perturbed automaton – and we will take a look to its particular shape below – but it prevents the application of theorem 5.1.6. Also, we draw the attention of the reader to the fact that in the construction of [6]; the guards of the final timed automaton are modified to allow constant of the type $1-\epsilon$ and $1+\epsilon$. In our formalism, this means a change of guarded timed structure.

This doesn't mean that the result of [6] can't be proved in our formalism but it means the proof is more about proving that a class of automata can be simulated by another one. Actually, we made the proof of this result in [20] with a slightly different formal context and results. In the formalism we used in this thesis we could again recover the proof but it would mean coming back nearly to a raw proof of bisimulation between two transition systems, and hence in some sense re-doing the proof of [6] . Indeed even the tool of functional bisimulation is too high level to be used to prove the equivalence between the different models.

Still, our construction is not entirely useless for this case. We take a look below to the powerset automaton of a one-clock perturbed timed automaton, and see how it helps us understand the fundamental reason which makes the class of perturbed timed automata translatable into timed automata.

If we fix $\langle A, \kappa\rangle$ a one-clock perturbed timed automaton, the configurations of its powerset automaton $\langle \mathbf{D}_\kappa A, \mathbf{T}^{\mathbf{D}\kappa}\rangle$ are a combination of a set of states and a marking on the states. The marking maps each state to a set of values. Because the perturbed clock structure is non-deterministic and because a clock can be perturbed by any δ between $1-\epsilon$ and $1+\epsilon$, the first thing to notice about those markings – which we recover in the proof of [6] – is that they map each state to a *finite union of intervals*.

This makes those sets uncountable at first glance. But if we manage to represent an interval by two values (representing in some way their lower and upper bound), we would just have to prove that the number of disjoint intervals is bounded. We could hence represent those unions by a finite number of pairs of values. If we restrict our selves to a $(1,\epsilon)$-perturbed clock structure (as they do in [6]) we can split $\mathbb{C}_1$ into the intervals

$$\mathcal{G}_\alpha = \{\{0\}, (0,\alpha), \{\alpha\}, (\alpha, 2\alpha), \ldots, (\lfloor\frac{1}{\alpha}\rfloor\alpha, 1), \{1\}, \{\infty\}\}$$

with $\alpha = \frac{\epsilon}{1+\epsilon}$. If the marking is reachable we can prove, because all intervals are of the form $[x(1-\epsilon), x(1+\epsilon)]$ that in the set associated to a state q,

- All intervals I such that $I \cap \left([\frac{1}{2}, 1] \cup \{\infty\}\right) \neq \emptyset$ can't be included in an interval $J \in \mathcal{G}_\alpha$.
- From all disjoint intervals I_1, I_2 included in some interval $J \in \mathcal{G}_\alpha$, if I is the convex hull of $I_1 \cup I_2$, A recognizes the same words from any configuration (q, x) with $x \in I$. This is a consequence of the fact that a guard constraint is either $\{0\}$, $(0, 1)$ or $\{1\}$, and by the time any value of I_1 or I_2 contains the value 1, they would have merged and would have generated the same values as I (c.f. figure 7.3). [20]

With those last remarks we could consider an automaton equivalent to $\mathbf{D}_\kappa A$, where every update merges intervals in the same $J \in \mathcal{G}_\alpha$. In such an automaton there would be no more than $|\mathcal{G}_\alpha|$ disjoint intervals as components of a set in the markings.

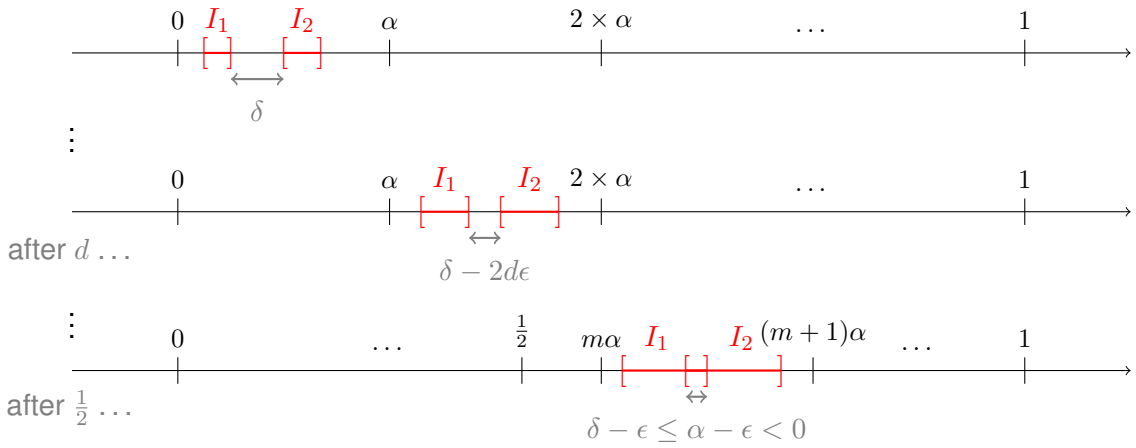


Figure 7.3: Evolution of the possible values of a perturbed clock

Proving that this last automaton is similar to the powerset automaton requires some work. Proving that this automaton is similar to an enhanced timed automaton requires some more work. This could be done with techniques very close to what is done in [20] . Though we can already see with the remarks above that the core of the proof is again that we could in some sense have a bounded representation of the powerset automaton.

Chapter 8

Full Control Determinization

Input-Determinacy Revisited

Input-determinacy is the property satisfied by systems in which for every *accepted word* it is possible to know the value of the quantitative variables at each moment of the run whatever the sequence of visited states. This property was first observed for event-clock automata [5] and was given a logical formalization by [29]. Contrarily to deterministic systems, along the run one may face a non-deterministic choice – and hence all words are not necessarily recognized by only one run –, but when one reaches a final state, it disambiguates the taken path and one can be sure of which run has been followed. The best example of such behavior is that of event-predicting clocks: such clocks store the time before the next instance of a particular event, they are non-deterministically chosen, and one knows their exact value only when the event is actually taken [5].

Section 8.1 : Input-determinacy property is verified by any automaton on a timed structure equipped with a *full control*. In this section we prove actually a stronger result which implies both input-determinacy property and stability with regards to full control determinization.

Section 8.2 : In this section we study results on input-determined systems determinization to sketch – thanks to our new framework – how full control determinization and input-determined systems relate. We can't provide formal conclusive results but we managed to establish a link between some input-determined system's determinization and the existence of an isomorphism between full control and timed control language.
The problems we focus on in section 8.2 are those of event-recording and event-predicting clocks [5], integer reset automata [49] [16], visibly pushdown automata [7] and their extension to dense time [1] [16].

We believe that those techniques could provide a general method to obtain determinization of input-determined systems for free.

8.1 Full Controlled ATS are always Strongly Determinizable

The fundamental property of a full controlled automaton on a deterministic timed structure is that the observer can know at any point of execution what is exactly the value of the quantitative variables. This is not without reminding us the notion of *input-determinacy* introduced in [24] .

Indeed recall that in a full control, the alphabet is the combination of a label *and an update*. Hence the observer knows the value, v, of the quantitative variable at some point:

- if he observes a delay d then he can deduce the reached value by computing $v \oplus d$,
- if he observes a discrete action (a, u) then he can deduce the value reached by computing $u(v)$.

Formally this is expressed in the following proposition.

Proposition 8.1.1. *Let Σ be a finite alphabet, $\mathcal{S}_G = \langle V, \hookrightarrow, U, G\rangle$ a guarded deterministic timed structure where U is finite and $A = \langle Q, I, T, F\rangle \in \mathbb{A}(\mathcal{S}_G)$ equipped with a compatible Σ-full control $\mathbf{T}^{\mathbf{F}}$.*

For all $c_i = (q_i, v_i) \in I$ and $w \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^*$ there exist at most one $v \in V$ such that

$$\exists q \in Q, (q, v) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, c_i)$$

Proof.

Let $c_i = (q_i, v_i) \in I$ and $w \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^*$
We make this proof by induction on the size of w.
- Initialization is obvious since $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, \epsilon, c_i) = \{c_i\}$
- Suppose we proved the property for all $w \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^n$ and let $w \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^{n+1}$.
  - Suppose $w = w_0 \cdot d$ with $w_0 \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^n$ and $d \in \mathbb{R}_{\geq 0}$.
    If $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w_0, c_i) = \emptyset$ then $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, c_i) = \emptyset$.
    Else we know that $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w_0, c_i) = \rho \times \{v_0\}$ with $v_0 \in V$ and $\rho \subseteq Q$.
    In this case $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, c_i) = \rho \times \{v_0 \oplus d\}$ by definition of $\mathbf{T}^{\mathbf{F}}$ and because $\mathcal{S}_G$ is deterministic.
  - Suppose $w = w_0 \cdot (a, u)$ with $w_0 \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^n$ and $(a, u) \in \Sigma \times U$.
    If $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w_0, c_i) = \emptyset$ then $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, c_i) = \emptyset$.
    Else we know that $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w_0, c_i) = \rho \times \{v_0\}$ with $v_0 \in V$ and $\rho \subseteq Q$.
    In this case $\mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, c_i) = \rho' \times \{u(v_0)\}$ with
    $\rho' = \{q' \in Q \mid \exists q \in \rho, (q, v_0, u, q') \in \mathbf{T}_{(a,u)}\}$
    by definition of $\mathbf{T}^{\mathbf{F}}$ and because $\mathcal{S}_G$ is deterministic.
  In all cases the property is verified at rank $n + 1$.
This ends the proof by induction of the proposition.

■

The only data undetermined is then which discrete state we are in. But to palliate to this last indeterminacy we can always make a standard powerset construction on the states.

From this observation and proposition 8.1.1 we get the core of the proof of the theorem below.

Theorem 8.1.2. *Let Σ a finite alphabet, $\mathcal{S}_G$ a guarded deterministic timed structure where U is finite and $A \in \mathbb{A}(\mathcal{S}_G)$ equipped with a compatible Σ-full control $\mathbf{T}^{\mathbf{F}}$.*

There exist $A' \in \mathbb{A}(\mathcal{S}_G)$ and $\mathbf{T}^{\mathbf{F}'}$ a compatible Σ-full control on A' such that A' is $\mathbf{F}'$-deterministic and $\mathcal{L}_{\mathbf{F}'}(A') = \mathcal{L}_{\mathbf{F}}(A)$.

Proof.

Let's write $A = \langle Q, I, T, F\rangle$ and $\mathcal{S}_G = \langle V, \hookrightarrow, U, G\rangle$.
We define $A_{\mathbf{d}} = \langle \mathcal{P}(Q), I_{\mathbf{d}}, T_{\mathbf{d}}, F_{\mathbf{d}}\rangle$ such that

- $I_{\mathbf{d}} = \{(\rho, v) \in \mathcal{P}(Q) \times V \mid \rho = \{q \in Q \mid (q, v) \in I\}\}$
- For all $(a, u) \in \Sigma \times U$,
  $T_{\mathbf{d},(a,u)} = \{(\rho, v, u, \rho') \in \mathcal{P}(Q) \times V \times U \times \mathcal{P}(Q) \mid$
  $\rho' = \{q' \in Q \mid \exists q \in \rho, (q, v, u, q') \in \mathbf{T}_{(a,u)}\}\}$
  And then $T_{\mathbf{d}} = \bigcup_{(a,u)\in\Sigma\times U} T_{\mathbf{d},(a,u)}$
- $F_{\mathbf{d}} = \{\rho \in \mathcal{P}(Q) \mid \rho \cap F \neq \emptyset\}$

We define also $\mathbf{F_d} : T \to \Sigma \times U$ such that for all $(a, u) \in \Sigma \times U$,
$t \in T_{\mathbf{d},(a,u)}$ implies $\mathbf{F_d}(t) = (a, u)$.
$\mathbf{F_d}$ is a well-defined full control on $\mathbf{A_d}$.

Let $(a, u) \in \Sigma \times U$.
We prove that for all $\rho, \rho' \in \mathcal{P}(Q)$, $g_{\mathbf{F_d}}(\rho, (a, u), \rho')$ is decomposable in G.

$$\begin{aligned} g_{\mathbf{F_d}}(\rho, (a, u), \rho') &= \{(v, u) \in V \times U \mid \rho' = \{q' \in Q \mid \exists q \in \rho, (q, v, u, q') \in \mathbf{T}_{(a,u)}\}\} \\ &= \{(v, u) \in V \times U \mid \rho' = \{q' \in Q \mid \exists q \in \rho, (v, u) \in g_{\mathbf{F}}(q, (a, u), q')\}\} \end{aligned}$$

and because G is a guard base and $\mathbf{T^F}$ is compatible with G:

$$g_{\mathbf{F_d}}(\rho, (a, u), \rho') = \bigcup_{(\nu,\mu)\in G \mid \rho'=\{q'\in Q \mid \exists q\in\rho, (\nu,\mu)\in g_{\mathbf{F}}(q,(a,u),q')\}} \nu \times \mu$$

Since $g_{\mathbf{F}}(q, (a, u), q')$ is decomposable on G, the decomposition above is finite and $g_{\mathbf{F_d}}(\rho, (a, u), \rho')$ is decomposable on G.

We proved that $\langle A_{\mathbf{d}}, \mathbf{T}^{\mathbf{F_d}}\rangle$ is in $\mathbb{A}(\mathcal{S}_G, \Sigma)$, we prove now that:
$A_{\mathbf{d}}$ is $\mathbf{F_d}$-deterministic and $\mathbf{F_d}$-complete.

Indeed let (ρ, v) be a configuration in $\mathbf{T}^{\mathbf{F}'}$.

- Let $d \in \mathbb{R}_{\geq 0}$.
  Because $\mathcal{S}_G$ is deterministic, the only reachable configuration by a transition d is $(\rho, v \oplus d)$.
- Let $(a, u) \in \Sigma \times U$.
  There exist at most one transition in $t = T_{(a,u)}$ applicable on (ρ, v) and this is (ρ, v, u, ρ') with $\rho' = \{q' \in Q \mid \exists q \in \rho, (q, v, u, q') \in \mathbf{T}_{(a,u)}\}$ by definition of $T_{\mathbf{d}}$ and $\mathbf{F_d}$.
  So the only reachable configuration by a transition (a, u) is $(\rho', u(v))$

Finally there is exactly one reachable configuration from (ρ, v)
Which proves the property.

■

We proved that $\langle A_{\mathbf{d}}, \mathbf{T}^{\mathbf{F_d}}\rangle$ is a good candidate to be the determinized of $\langle A, \mathbf{T^F}\rangle$.
In order to prove the language equality we need an intermediate result.
We prove now that for all $w \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^*$, by induction that:

(1) For all $(\rho_i, v_i) \in I_{\mathbf{d}}$, for all $(\rho, v) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F_d}}, w, (\rho_i, v_i))$, for all $q \in \rho$, exists $q_i \in \rho_i$ such that $(q, v) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, (q_i, v_i))$.

- Initialization is directly obtained by definition of $I_{\mathbf{d}}$.
- Suppose we proved the property for all words of length $n \in \mathbb{N}$.
  Let $w \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^{n+1}$.
  - Suppose $w = w_0 \cdot d$ with $w_0 \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^n$ and $d \in \mathbb{R}_{\geq 0}$.
    Let $(\rho_i, v_i) \in I_{\mathbf{d}}$, $(\rho, v) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F_d}}, w, (\rho_i, v_i))$, and $q \in \rho$.
    It means that there exist $(\rho', v') \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F_d}}, w_0, (\rho_i, v_i))$ such that $(\rho', v') \xrightarrow{d}_{\mathbf{T}^{\mathbf{F_d}}} (\rho, v)$.
    So $\rho' = \rho$ and $v = v' \oplus d$.
    By induction hypothesis, there exist $q_i \in \rho_i$ such that for all $q' \in \rho$, $(q', v') \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w_0, (q_i, v_i))$.
    Hence in particular $(q, v') \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w_0, (q_i, v_i))$,
    and $(q, v' \oplus d) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, (q_i, v_i))$.
  - Suppose $w = w_0 \cdot (a, u)$ with $w_0 \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^n$ and $(a, u) \in \Sigma \times U$.
    Let $(\rho_i, v_i) \in I_{\mathbf{d}}$, $(\rho, v) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F_d}}, w, (\rho_i, v_i))$, and $q \in \rho$.
    It means that there exist $(\rho', v') \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F_d}}, w_0, (\rho_i, v_i))$ such that $(\rho', v') \xrightarrow{(a,u)}_{\mathbf{T}^{\mathbf{F_d}}} (\rho, v)$.
    So $\rho = \{q' \in Q \mid \exists q'' \in \rho', (q'', v, u, q') \in \mathbf{T}_{(a,u)}\}$ and $v = u(v')$.
    Therefore there exist $q' \in \rho'$ such that $(q', v', u, q) \in \mathbf{T}_{(a,u)}$.
    By induction hypothesis, there exist $q_i \in \rho_i$ such that $(q', v') \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w_0, (q_i, v_i))$.
    Hence $(q, u(v')) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, (q_i, v_i))$.

  This proves property (1) by induction.

■

and that:

(2) For all $(q_i, v_i) \in I$, for all $(q, v) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F}}, w, (q_i, v_i))$, exists $\rho_i, \rho \in \mathcal{P}(Q)$ such that $q_i \in \rho_i$, $q \in \rho$ and $(\rho, v) \in \mathbf{Reach}(\mathbf{T}^{\mathbf{F_d}}, w, (\rho_i, v_i))$.

- Initialization is directly obtained by definition of $I_{\mathbf{d}}$.
- Suppose we proved the property for all words of length $n \in \mathbb{N}$.
  Let $w \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^{n+1}$.
  - Suppose $w = w_0 \cdot d$ with $w_0 \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^n$ and $d \in \mathbb{R}_{\geq 0}$.
    Let $(q_i, v_i) \in I$, and $(q, v) \in \mathbf{Reach}(\mathbf{T^F}, w, (q_i, v_i))$
    It means that there exist $(q', v') \in \mathbf{Reach}(\mathbf{T^F}, w_0, (q_i, v_i))$ such that $(q', v') \xrightarrow{d}_{\mathbf{T^F}} (q, v)$.
    So $q' = q$ and $v = v' \oplus d$.
    By induction hypothesis, there exist $\rho_i, \rho \in \mathcal{P}(Q)$ such that $q_i \in \rho_i$, $q \in \rho$ and $(\rho, v') \in \mathbf{Reach}(\mathbf{T^{F_d}}, w_0, (\rho_i, v_i))$.
    Hence this means that $(\rho, v' \oplus d) \in \mathbf{Reach}(\mathbf{T^{F_d}}, w, (\rho_i, v_i))$.
  - Suppose $w = w_0 \cdot (a, u)$ with $w_0 \in [(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^n$ and $(a, u) \in \Sigma \times U$.
    Let $(q_i, v_i) \in I$, and $(q, v) \in \mathbf{Reach}(\mathbf{T^F}, w, (q_i, v_i))$
    It means that there exist $(q', v') \in \mathbf{Reach}(\mathbf{T^F}, w_0, (q_i, v_i))$ such that $(q', v') \xrightarrow{(a,u)}_{\mathbf{T^F}} (q, v)$.
    So $v = u(v')$ and $(q', v', u, q) \in T_{(a,u)}$.
    By induction hypothesis, there exist $\rho_i, \rho' \in \mathcal{P}(Q)$ such that $q_i \in \rho_i$, $q' \in \rho'$ and $(\rho', v') \in \mathbf{Reach}(\mathbf{T^{F_d}}, w_0, (\rho_i, v_i))$.
    Let $\rho = \{l \in Q \mid \exists l' \in \rho', (l', v, u, l) \in T_{(a,u)}\}$.
    $q \in \rho$ and $(\rho', v', u, \rho) \in T_{\mathbf{d}}$ by definition.
    Hence $(\rho, u(v')) \in \mathbf{Reach}(\mathbf{T^{F_d}}, w, (\rho_i, v_i))$.

  This proves property (2) by induction.

■

We can now prove that $\mathcal{L}_{\mathbf{F}}(A) = \mathcal{L}_{\mathbf{F_d}}(A_{\mathbf{d}})$.
We proceed by double inclusion.

- Let $w \in \mathcal{L}_{\mathbf{F}}(A)$ and π a run recognizing w in $\langle A, \mathbf{T^F} \rangle$.
  Let $c_i = \mathbf{start}(\pi) = (q_i, v_i) \in I$ and $(q, v) = \mathbf{end}(\pi) \in F \times V$.
  According to (2), there exist $\rho_i, \rho \in \mathcal{P}(Q)$ such that $q_i \in \rho_i, q \in \rho$ and $(\rho, v) \in \mathbf{Reach}(\mathbf{T^{F_d}}, w, (\rho_i, v_i))$.
  Since $q \in F$ and $q \in \rho$, then $\rho \in F_{\mathbf{d}}$.
  Which means that $w \in \mathcal{L}_{\mathbf{F_d}}(A_{\mathbf{d}})$.
- Let $w \in \mathcal{L}_{\mathbf{F_d}}(A_{\mathbf{d}})$ and π a run recognizing w in $\langle A_{\mathbf{d}}, \mathbf{T^{F_d}} \rangle$.
  Let $c_i = \mathbf{start}(\pi) = (\rho_i, v_i) \in I_{\mathbf{d}}$ and $(\rho, v) = \mathbf{end}(\pi) \in F_{\mathbf{d}} \times V$.
  Let $q \in \rho \cap F_{\mathbf{d}}$.
  According to (1), there exist $q_i \in \rho_i$ such that $(q, v) \in \mathbf{Reach}(\mathbf{T^F}, w, (q_i, v_i))$.
  Since $q \in F$ and $q_i, v_i \in I$ by definition of $I_{\mathbf{d}}$,
  $w \in \mathcal{L}_{\mathbf{F}}(A)$.

Which proves the language equality and ends the proof.

■

This theorem (8.1.2) provides us a way to determinize an automaton, with regard to a full control, *within the same timed structure*. Even though the full control does not seem to be an interesting restriction – because it gives a lot of information to the observer by giving him the update – we show in the next section how it can be used to recover some determinization results about input-determined models.

8.2 Application to Input-Determined Models

In this section, we show how we can retrieve some results about input-determined timed automata by using full control determinization. As we noticed in section 8.1, an automaton controlled by a full control satisfies a kind of input-determinacy property exposed in proposition 8.1.1. We wondered if, conversely, there is a link between input-determinacy and full controls.

We don't claim we have an answer to this question, but we show that this idea seems to be confirmed by the studies we make below on event-clock timed automata [5] , strict integer reset timed automata [49] [16] and dense-timed integer reset visibly pushdown automata. [1] [16]

Those studies are based on the results we expose below which transfer properties expressed about full-controls into properties about timed controls. Fix Σ a finite alphabet, $\mathcal{S}_G$ a guarded deterministic timed structure with U finite and $A = \langle Q, \{c_i\}, T, F\rangle \in \mathbb{A}(\mathcal{S}_G)$ equipped with a compatible Σ-full control $\mathbf{T}^{\mathbf{F}}$. Notice that we imposed here that A have only one initial state. We can construct from $\mathbf{F}$ a Σ-timed control $\Pi(\mathbf{F})$ defined for all $t \in T$ as $\Pi(\mathbf{F})(t) = p(\mathbf{F}(t))$ where $p : \Sigma \times U \to \Sigma$ is the projection on Σ of $\Sigma \times U$. $\mathbf{T}_{\Pi(\mathbf{F})}$ is a compatible Σ-timed control on A. Let's write p^* for the extension of p on $[(\Sigma \times U) \uplus \mathbb{R}_{\geq 0}]^*$ where we consider $p(d) = d$ for all $d \in \mathbb{R}_{\geq 0}$. Notice that conversely, it is easy to construct from any Σ-timed control $\mathbf{T}^{\kappa}$ on A, a compatible Σ-full control $\mathbf{T}^{\Pi^{-1}(\kappa)}$, defined for all $t = (q, v, u, q') \in T$ by $\Pi^{-1}(\kappa)(t) = (\kappa(t), u)$.

Corollary 8.2.1. *There exist $A' \in \mathbb{A}(\mathcal{S}_G)$ and $\mathbf{T}^{\mathbf{F}'}$ a compatible Σ-full control such that A' is $\mathbf{F}'$-deterministic and $\mathcal{L}_{\Pi(\mathbf{F}')}(A') = \mathcal{L}_{\Pi(\mathbf{F})}(A)$.*

Proof.

By application of theorem 8.1.2 we know that there exist $A' \in \mathbb{A}(\mathcal{S}_G)$ and $\mathbf{T}^{\mathbf{F}'}$ a compatible Σ-full control such that A' is $\mathbf{F}'$-deterministic and $\mathcal{L}_{\mathbf{F}'}(A') = \mathcal{L}_{\mathbf{F}}(A)$.

We get then

$$\mathcal{L}_{\Pi(\mathbf{F})}(A) = p^*(\mathcal{L}_{\mathbf{F}}(A)) = p^*(\mathcal{L}_{\mathbf{F}'}(A)) = \mathcal{L}_{\Pi(\mathbf{F}')}(A)$$

■

Below we state a corollary which allows us to recover one main property of input-determined systems.

Corollary 8.2.2. *Suppose p^* is a bijection from $\mathcal{L}_{\mathbf{F}}(A)$ to $\mathcal{L}_{\Pi(\mathbf{F})}(A)$*

There exist $A' \in \mathbb{A}(\mathcal{S}_G)$ and $\mathbf{T}^{\mathbf{F}'}$ a compatible Σ-full control such that A' is $\mathbf{F}'$-deterministic, $\mathcal{L}_{\Pi(\mathbf{F}')}(A') = \mathcal{L}_{\Pi(\mathbf{F})}(A)$ and for all $w \in \mathcal{L}_{\Pi(\mathbf{F}')}(A')$ there exist exactly one run π of $\langle A', \mathbf{T}^{\Pi(\mathbf{F}')}\rangle$ such that $\mathbf{w}(\pi) = w$.

Proof.

By application of corollary 8.2.1 we know that there exist $A' \in \mathbb{A}(\mathcal{S}_G)$ and $\mathbf{T}^{\mathbf{F}'}$ a compatible Σ-full control such that A' is $\mathbf{F}'$-deterministic and $\mathcal{L}_{\Pi(\mathbf{F}')}(A') = \mathcal{L}_{\Pi(\mathbf{F})}(A)$.

Let $w \in \mathcal{L}_{\Pi(\mathbf{F}')}(A')$ and $w_f = {p^*}^{-1}(w)$.
w_f is unique and well-defined because p^* is a bijection.
Let π_f be a run of $\langle A', \mathbf{T}^{\mathbf{F}'}\rangle$ such that $\mathbf{w}(\pi_f) = w$.
π_f is unique because $\langle A', \mathbf{T}^{\mathbf{F}'}\rangle$ is deterministic and A' has only one initial configuration, (given that A has only one initial configuration, c.f. proof of theorem 8.1.2).

We write $\pi_f = c_0 \xrightarrow{\gamma_1} c_1 \xrightarrow{\gamma_2} \dots \xrightarrow{\gamma_n} c_n$.
Of course, $\pi = c_0 \xrightarrow{p^*(\gamma_1)} c_1 \xrightarrow{p^*(\gamma_2)} \dots \xrightarrow{p^*(\gamma_n)} c_n$ is a run of $\langle A', \mathbf{T}^{\Pi(\mathbf{F}')}\rangle$ such that $\mathbf{w}(\pi) = w$.

Let π' a run of $\langle A', \mathbf{T}^{\Pi(\mathbf{F}')}\rangle$ such that $\mathbf{w}(\pi) = w$.
We write $\pi = c_0 \xrightarrow{\gamma'_1} c_1 \xrightarrow{\gamma'_2} \dots \xrightarrow{\gamma'_n} c_n$.
For all $1 \leq i \leq n$:
we define $\gamma_i = d$ if $\gamma'_i = d$;
else if $\gamma_i = a \in \Sigma$, we know by definition of $\Pi(\mathbf{F}')$ that there exist $u \in U$ such that
$c_{i-1} \xrightarrow{(a,u)} c_i$ in $\mathbf{T}^{\mathbf{F}'}$; we define then $\gamma_i = (a, u)$.
This means that $\pi'_f = c_0 \xrightarrow{\gamma_1} c_1 \xrightarrow{\gamma_2} \dots \xrightarrow{\gamma_n} c_n$ is a run of $\langle A', \mathbf{T}^{\mathbf{F}'}\rangle$ such that $p^*(\mathbf{w}(\pi'_f)) = w$.
By injectivity of p^* and determinism of $\langle A', \mathbf{T}^{\mathbf{F}'}\rangle$,
this means that $p'_f = p_f$ and thus that $\pi' = \pi$.

This proves existence and unicity of π.

■

Finally in some very particular case corresponding to a labeling which pairs each label with exactly one update being two by two distinct, we get a strong determinizability result:

Corollary 8.2.3. *Suppose p is a bijection from $\mathbf{im}(\mathbf{F})$ to $\mathbf{im}(\Pi(\mathbf{F}))$.*
There exist $A' \in \mathbb{A}(\mathcal{S}_G)$ and $\mathbf{T}^{\mathbf{F}'}$ a compatible Σ-full control such that A' is $\Pi(\mathbf{F}')$-deterministic, $\mathcal{L}_{\Pi(\mathbf{F}')}(A') = \mathcal{L}_{\Pi(\mathbf{F})}(A)$.

Proof.

By application of corollary 8.2.1 we know that there exist $A' \in \mathbb{A}(\mathcal{S}_G)$ and $\mathbf{T}^{\mathbf{F}'}$ a compatible Σ-full control such that A' is $\mathbf{F}'$-deterministic and $\mathcal{L}_{\Pi(\mathbf{F}')}(A') = \mathcal{L}_{\Pi(\mathbf{F})}(A)$.

We proves that A' is actually $\Pi(\mathbf{F}')$-deterministic.
- Let $(q, v) \in C_{A'}$ and $d \in \mathbb{R}_{\geq 0}$.
  Because $\mathcal{S}_G$ is deterministic, the only reachable configuration by a transition d is $(q, v \oplus d)$.
- Let $a \in \Sigma$.
  If $a \notin \mathbf{im}(\Pi(\mathbf{F}))$, then there is no transition a from c.
  Otherwise, if $a \in \mathbf{im}(\Pi(\mathbf{F}))$, then let $(a, u) \in \mathbf{im}(\mathbf{F})$.
  Such element exists by hypothesis on p.
  Suppose $c \xrightarrow{a} c'$ in $\mathbf{T}^{\Pi(\mathbf{F}')}$, this means that there exist $(a, u') \in \mathbf{im}(\mathbf{F})$ such that $c \xrightarrow{(a,u')} c'$ in $\mathbf{T}^{\mathbf{F}'}$ by definition of $\Pi(\mathbf{F}')$.
  By injectivity of p though, it means that $u = u'$ and $c \xrightarrow{(a,u)} c'$.
  Finally by determinism of $\langle A', \mathbf{T}^{\mathbf{F}'}\rangle$, c' is then unique.

This proves that A' is $\Pi(\mathbf{F}')$-deterministic.

■

Event-recording automata. Let Σ be a finite alphabet and $M \in \mathbb{N}$. An *M-bounded event-recording automaton on Σ* as defined in [5] is a special M-bounded $|\Sigma|$-clocks timed automata on Σ without silent transition, having one clock for each element of Σ being reset each time, and only when, the corresponding action is taken.

Formally speaking, fix $\mathbf{ind} : \Sigma \to [\![1, |\Sigma|]\!]$ any bijection, and write for all $\sigma \in \Sigma$,

$$\mathbf{0}_\sigma = (\underbrace{\mathbf{id}, \dots, \mathbf{id}}_{\mathbf{ind}(\sigma)-1}, \mathbf{0}, \underbrace{\mathbf{id}, \dots, \mathbf{id}}_{|\Sigma|-\mathbf{ind}(\sigma)})$$

$\langle A, \mathbf{T}^\kappa \rangle$ with $A = \langle Q, \{(q_i, \overrightarrow{0})\}, T, F \rangle$ is an M-bounded event-recording automata on Σ, if and only if for all $q, q' \in Q$ and for all $\sigma \in \Sigma$, there exist $\nu \subseteq \mathbb{C}_M^{|\Sigma|}$ such that

$$g_\kappa(q, \sigma, q') = \nu \times \{\mathbf{0}_\sigma\}$$

With this constraint is is clear that for every M-bounded event-recording timed automata on Σ, $\langle A, \mathbf{T}^\kappa \rangle$, p – the projection of $\Sigma \times \{\mathbf{id}, \mathbf{0}\}$ on Σ – is a bijection from $\mathbf{im}(\Pi^{-1}(\kappa))$ to $\mathbf{im}(\kappa)$.

Indeed $\mathbf{im}(\kappa)$ corresponds to all the actions, a, attributed to a transition, and we know that then those transitions have as only possible update, $\mathbf{0}_\sigma$. So $(a, \mathbf{0}_\sigma)$ is in $\mathbf{im}(\Pi^{-1}(\kappa))$, because it is attributed to those same transition in the full control $\Pi^{-1}(\kappa)$ (by definition of it). And the only possible inverse image of a by p becomes $(a, \mathbf{0}_\sigma)$.

This means that corollary 8.2.3 is applicable and that the class of event-recording automata is strongly determinizable.

Event-predicting automata. Let Σ be a finite alphabet and $M \in \mathbb{N}$. An *event-predicting automaton on* Σ as defined in [5] one clock for any action in Σ. is an automaton on the predicting clock structure (c.f. page 17). At each transition, a guard is enforcing the clock associated with the label of this transition to be 0 and non-deterministically updating the clock to the delay until the next occurrence of this letter using an update in $\mathbf{U}\mathbb{R}_{\geq 0}$.

Recall that $\mathbf{U}\mathbb{R}_{\geq 0} = \{\mathbf{d}, d \in \mathbb{R}_{\geq 0}\}$. Formally fix $\mathbf{ind} : \Sigma \to [\![1, |\Sigma|]\!]$ any bijection, we define

$$\mathbf{P}\mathbb{G}_M = \mathcal{I}_M{}^{1} \cup \{\bot\} \times \mathbf{U}\mathbb{R}_{\geq 0}$$

we also define for all $\sigma \in \Sigma$.

$$G_\sigma = \{\big([I_1, \dots, I_{\mathbf{ind}(\sigma)-1}, [0;0], I_{\mathbf{ind}(\sigma)+1}, \dots, I_{|\Sigma|-\mathbf{ind}(\sigma)}], \\ [\underbrace{\{\mathbf{id}\}, \dots, \{\mathbf{id}\}}_{\mathbf{ind}(\sigma)-1}, \mathbf{U}\mathbb{R}_{\geq 0}, \underbrace{\{\mathbf{id}\}, \dots, \{\mathbf{id}\}}_{|\Sigma|-\mathbf{ind}(\sigma)}] \big), I_1, \dots, I_{|\Sigma|} \in \mathcal{I}_M\}$$

G_σ describes all guards imposing $x_\sigma = 0$ and updating only x_σ to a random value, where x_σ stand for the clock associated to σ.

An M, Σ-event-predicting automaton, is:

- An automaton $A = \langle Q, \{(q_i, \overrightarrow{0})\}, T, F \rangle$ on the guarded structure $\mathbf{P}C^{|\Sigma|}_{\mathbf{P}\mathbb{G}_M}$. The initial state puts all clock value to $\bot$. They will have to be initialized.
- Equipped with a $\Sigma \uplus \{\#\}$-timed control κ without silent transitions. $\#$ is the starting symbol which role is to initialize all values.
- Satisfying for all $q \in Q \setminus F$, for all $\sigma \in \Sigma$, $g_\kappa(q_i, \sigma, q') = \emptyset$ and

  $$g_\kappa(q_i, \#, q') = (\overrightarrow{\{0\}}, \mathbf{U}\mathbb{R}_{\geq 0}{}^{|\Sigma|})$$

  for all $q \in Q \setminus \{q_i\}$, for all $q' \in F$, $g_\kappa(q, \#, q') = \emptyset$ and for all $\sigma \in \Sigma$,

  $$g_\kappa(q, \sigma, q') = \big([\underbrace{\{\bot\}, \dots, \{\bot\}}_{\mathbf{ind}(\sigma)-1}, [0;0], \underbrace{\{\bot\}, \dots, \{\bot\}}_{|\Sigma|-\mathbf{ind}(\sigma)}], \quad [\underbrace{\{\mathbf{id}\}, \dots, \{\mathbf{id}\}}_{\mathbf{ind}(\sigma)-1}, \bot, \underbrace{\{\mathbf{id}\}, \dots, \{\mathbf{id}\}}_{|\Sigma|-\mathbf{ind}(\sigma)}] \big)$$

[1] The set of real intervals with two consecutive integers as bounds (c.f example 3.1.1)

and finally for all $q, q' \in Q \setminus F \cup \{q_i\}$, $g_\kappa(q, \#, q') = \emptyset$ and for all $\sigma \in \Sigma$, there exist $G_1, \ldots, G_m \in G_\sigma$ such that

$$g_\kappa(q, \sigma, q') = \bigcup_{i=1}^{m} G_i$$

With this setting, because an accepting run has to end with value $\overrightarrow{\perp}$ and each transition has to be made when the corresponding clock is equal to 0, one can proves that p^* – the extension of the projection of $(\Sigma \cup \{\#\}) \times (\mathbf{U}\mathbb{R}_{\geq 0} \cup \{\mathbf{id}\})$ on $\Sigma \cup \{\#\}$ – is a bijection from $\mathcal{L}_{\Pi^{-1}(\kappa)}(A)$ to $\mathcal{L}_\kappa(A)$. In fact, given an accepted word, we can easily retrieve which sequence of updates have been made along the run. To do so we compute at each action along the run the time elapsed since the previous occurrence of this same action.

We can therefore apply corollary 8.2.2 and get for each event-predicting automaton an automaton recognizing the same κ language and such that each accepted word is recognized by exactly one run. Notice that this new automaton *is not κ-deterministic*, it is only $\Pi^{-1}(\kappa)$-deterministic. In fact, the difference is that, even if one accepted word corresponds exactly to one run, it does not mean that from each configuration of the automaton there is only one configuration reachable per action. This is the acceptance condition that enforces the unicity of the accepting run, and not the definition of the transition relation.

We could observe a similar thing for event-clock automaton as defined in [5] , which are a combination of event-recording and event-predicting clocks. In our formalism it would be modeled by an automaton on $(\mathcal{C}_M \times \mathbf{P}C)^{|\Sigma|}$.

Strict Integer Reset Timed Automata. Let Σ be a finite alphabet and $M \in \mathbb{N}$. A *strict integer reset timed automaton on* Σ as defined in [16] is an integer reset timed automaton as defined in section 7.2 with only one clock, reset each time a transition is made on an integer timestamp.

We already proved that such an automaton is strongly determinizable w.r.t. to timed controls in section 7.2. We could prove it using full controls. One can prove, as it is kind of done in [16] that hypothesis of corollary 8.2.2 are verified and thus that we can construct for any strictIRTA $\langle A, \kappa \rangle$ a strictIRTA $\langle A', \kappa' \rangle$ such that every accepted word is recognized exactly by one run. We can't use directly corollary 8.2.3 to get strong determinizability, but from $\langle A', \kappa' \rangle$, which we know is $\Pi^{-1}(\kappa)$-deterministic we would just have to notice that for all configuration (q, x) the update is entirely determined by the fact that $x \in \mathbb{N}$ (in which case the update of all transition leaving have to be $\mathbf{0}$), to be able to prove that A' is κ'-deterministic.

Visibly Timed Pushdown Automata Visibly Pushdown Automata as introduced in [7] define a determinizable class of pushdown automata. The idea is close to the idea of event-clock automata: associating labels to particular updates. In visibly pushdown automata we split actions between popping and pushing actions. In [16], visibly pushdown automata are extended with ages for the stack symbols and additional clocks that can be reset only on integer timestamps.

We give here as an example of how our work can extend to quantitative models (i.e. not pure clock models), a modelization of some altered, simpler, versions of those models and how we could prove it has the input-determinacy property.

Formally fix $M \in \mathbb{N}$, and Z be a stack alphabet, recall that $U_{M,Z} = \{\mathbf{pop}, \mathbf{id}\} \cup \{\mathbf{push}_z, z \in Z\}$ and that $\mathbf{Z}_{M,Z}$ stands for the M, Z-timed stack structure.

An M, Z, Σ-visibly pushdown timed automata, is:

- An automaton $A = \langle Q, \{(q_i, \epsilon)\}, T, F\rangle$ on the guarded structure $\mathbf{Z}_{M,Z}$. The initial state require all clocks value to 0 and the stack to be empty.
- Equipped with a $\Sigma_{\mathbf{pop}} \cup \Sigma_{\mathbf{push}} \cup \Sigma$-timed control κ without silent transitions. $\Sigma_{\mathbf{pop}}$ corresponds to the labels of transitions with a $\mathbf{pop}$ update, $\Sigma_{\mathbf{push}}$ corresponds to the labels of transitions with a $\mathbf{push}$ update, and Σ corresponds to the labels of transitions with an $\mathbf{id}$ update.
- Verifying for all $q, q' \in Q$ and for all $\sigma \in \Sigma_{\mathbf{pop}}$, there exist $\nu \subseteq \mathcal{Z}_{m,Z}$ such that

  $$g_\kappa(q, \sigma, q') = \nu \times \{\mathbf{pop}\}$$

  and for all $\sigma \in \Sigma_{\mathbf{push}}$, there exist $\nu \subseteq \mathcal{Z}_{m,Z}$ and $z \in Z$ such that

  $$g_\kappa(q, \sigma, q') = \nu \times \{\mathbf{push}_z\}$$

  and for all $\sigma \in \Sigma$, there exist $\nu \subseteq \mathcal{Z}_{m,Z}$ such that

  $$g_\kappa(q, \sigma, q') = \nu \times \{\mathbf{id}\}$$

If we consider Z to be reduced to one element $Z = \{z\}$, then it is easy to see that the hypothesizes of corollary 8.2.3 are satisfied and that such an automaton can be strongly determinized.

In the case of multiple stack symbols, suppose we impose that all final states are reached with an empty stack. Although given an accepted word, we can track back what are the possible sequences of $\mathbf{push}$ and $\mathbf{pop}$ updates which leads to an empty stack, the automaton does not directly verify the hypothesis of corollary 8.2.2. This is because two different $\mathbf{push}$ and $\mathbf{pop}$ sequences can lead to the same stack value. We would need a prior transformation of the automaton, as they do in [7], to merge the different possibilities into one and then we could conclude, first using corollary 8.2.2 to construct an automaton with an input-determinacy property, which recognizes the same language.

Finally as the last example, suppose we extend this model to work on the guarded time structure $\mathcal{Z}_{M,1,Z}$ and impose that all final states are reached with an empty stack *and* that the clock is reset at all integer time stamp like in strictIRTA. In [16] they prove this model has the input-determinacy property, even though they prove that, given an accepted word, only the clock value is deductible at any point of the run. In our setting input-determinacy would require both the clock *and the stack* value to be deductible. As was the case in previous examples, the automaton does not directly satisfy the hypothesis of corollary 8.2.2 – for the same reason. We think that the same kind of construction as above would allow us to construct an automaton with an input-determinacy property, which recognizes the same language.

Those studies make us think that the notion of controls might help us better understand the link between input-determinacy and determinization, and more generally determinization results in their different shapes. We hope those remarks could be useful to unify the different results about determinization, for input-determined systems and the others.

Part III

Diagnosis of Timed Automata

Chapter 9

Diagnosis for Timed Automata

Overview of the problem

In this part, we address the problem of diagnosis for timed automata as an application of the framework of automata on timed structures. Diagnosis is an on-the-fly verification method. When one cannot ensure safety properties using model-checking or want to execute a non-safe system, it can still observe a live execution of its system and try to predict if and when a fault can occur so it can prevent it from happening.

A diagnoser [47] is a machine that automatically performs those observations and predictions. It can be constructed based on formal modeling of the system and ran in parallel to the system, observing each *observable* actions and providing predictions about fault occurrences if it can. . .

Indeed it is not always possible to construct such a machine. To be able to exactly state if a fault is going to occur or not, the model of the system must not have ambiguous behavior. However, even if ambiguous behaviors are inherent to the model, it is still possible to construct a diagnoser if we consider a third status which says that *a fault may occur*, without being able to state nor safety nor fault. The diagnoser is asked to be computable and efficient so it can quickly simulate and compute the status of the system. That's one reason why in the literature a diagnoser is usually sought within the same class as the model (to ensure computability and easy efficiency comparison).

Theoretically, the disambiguation of the system is achieved using determinization and silent transition elimination, even if it is not always sufficient. Like many other models, as already mentioned, timed automata are not stable by determinization [4] or by silent transition elimination [14], which prevents the automatic construction for any timed automata of a timed automata diagnoser. To circumvent this problem, in [18] the authors first address the problem on determinizable classes of timed automata. They also provide a game approach to decide if a timed automaton is diagnosable which can be put in close relationship with the game approach used in [15] to decide if a timed automaton with silent transitions can be determinized. Whereas in [53] the preferred solution is to construct a diagnoser outside the class of timed automata which can still be computed.

Our approach falls within the second type of solution. Using automata on timed structures we use the power-construction of the clock domain to construct a diagnoser which belongs to a super-class of timed automata. We will in particular need to address the problem of silent transition elimination and solve it by considering again a super-class of timed automata. A similar idea was exploited in [28] in a different framework, leading to different constructions.

This chapter's goal is to formally translate the problem of diagnosis in our framework and introduce what is our contribution to the problem in comparison to the work done in [53].

9.1 General Context

In this section, we take some time to introduce the context of the work. We'll recall some definitions and results to situate our work in what was already done. We will also explain what exactly enters in our formalism and what does not. Indeed the work of [53] considers timed automata with state invariants, and because we chose not to model them some results become irrelevant in our framework. Also because we are not interested in languages in this part, final states are not relevant and we will usually define the set of final states as being the empty set.

Fix for all this section a finite alphabet Σ and a Σ-timed automaton $A_\kappa = \langle A, \mathbf{T}^\kappa \rangle$. We write $A = \langle Q, I, T, F \rangle$. We recall that κ can be naturally extended on words of $(T \uplus \mathbb{R}_{\geq 0})^*$ in the way exposed in remark 4.2.2.

We suppose that there exist a set $T_f \subseteq T$ modeling a set of *faulty transitions*. A *faulty path of* A_κ is a path π of T_A such that a transition in T_f appears in $\mathbf{w}(\pi)$.

Example 9.1.1. Fix $\Sigma = \{a\}$ and consider for example the 1-clock 1-bounded timed automaton $\langle B, \eta \rangle \in \Sigma\mathbf{TA}_1^1$ described below.

$$B = (\{q_1, q_2, q_f\}, \{(q_1, 0)\}, T_B, \emptyset)$$

such that

$$\begin{aligned} g_\eta(q_1, a, q_1) &= (0,1) \times \{\mathbf{0}\} \\ g_\eta(q_1, a, q_2) &= (0,1) \times \{\mathbf{0}\} \\ g_\eta(q_2, a, q_1) &= (0,1) \times \{\mathbf{0}\} \\ g_\eta(q_2, \epsilon, q_f) &= \{1, \infty\} \times \{\mathbf{id}\} \\ g_\eta(q_f, a, q_2) &= \{1, \infty\} \times \{\mathbf{0}\} \end{aligned}$$

with unspecifieds guard implicitly defined as empty. Its canonical graph is given in figure 9.1.

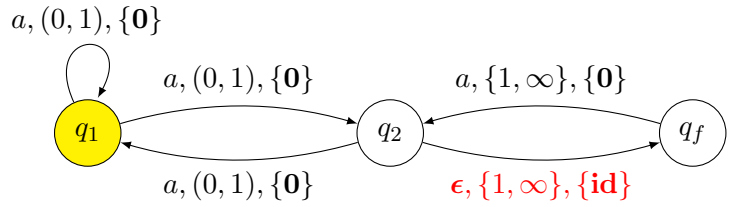


Figure 9.1: A one-clock timed automaton $\mathcal{B}$ with a faulty transition

If we consider that $T_f = \{(q_2, v, \{\mathbf{id}\}, q_f), v \in \{1, \infty\}\}$ then the run

$$\pi_f = (q_1, 0) \xrightarrow{0.4} (q_1, 0.4) \xrightarrow{(q_1, 0.4, \mathbf{0}, q_1)} (q_1, 0) \xrightarrow{(q_1, 0, \mathbf{0}, q_2)} (q_2, 0) \xrightarrow{2} (q_2, \infty) \xrightarrow{(q_2, \infty, \mathbf{id}, q_f)} (q_f, \infty)$$

is a faulty path.

Diagnosability is defined in [53] as the ability to distinguish faulty paths from non-faulty paths based on the observable actions of those paths. In particular, this means that a timed automaton will not be diagnosable if we can find two paths, one faulty, the other not, which involves the same observable behavior. In this case, it would be indeed impossible to distinguish the two paths and to state with absolute certainty that we just observed a faulty path or on the contrary a non-faulty behavior. Moreover, the author of [53] defines a notion of Δ-diagnosability

which is the ability to distinguish faulty paths from non-faulty *within Δ time units after the occurrence of the faulty transition*. This notion of Δ-diagnosability is not relevant for our settings because we didn't model state invariants. Because we can always indefinitely delay in any state of the automaton, only observable actions can ensure to a diagnoser whether the automaton performed a faulty action or not. Therefore to translate the notion of diagnosability in our formalism, we would need to introduce *diagnosability after n step* which corresponds to n observable actions. This is not the subject of our work so we chose not to explore those questions.

Δ-diagnosability is decidable on timed automata as it is proved in [53]. In this thesis, we don't work on diagnosability checks and hence won't claim any result on decidability of diagnosability after n steps.

We focus, in this part of the thesis, only on the construction of a diagnoser. More precisely we work on the construction of a *state estimator* or *observer* for A_κ.

A diagnoser as defined in [53] is a *deterministic (computable) machine* which is allowed to observe all observable actions of A_κ and to react to it. It must announce a fault *only if* the path in A is faulty and never stop announcing it. Such a machine can be constructed as explained in [53]. An observer is also a *deterministic (computable) machine* which is allowed to observe all observable actions of A_κ and to react to it. It keeps track of all the reachable configurations of A_κ. As it said in [53] [30] a diagnoser and an observer are similar objects with a different goal. An observer usually computes a state estimation. From this observer, we could obtain a diagnoser by keeping track of the possible transitions made to reach the states (if we know that the automaton is diagnosable after n-steps we only have to store n transitions).

In the next section, we will impose a context where our observer can be easily used as a diagnoser, and describe how.

Example 9.1.2. Consider the automaton $\langle B, \eta\rangle$ defined in example 9.1.1. We show below what could be the outcome of an observer $\mathbf{O}_\eta B$ and diagnoser $\mathcal{D}_\eta B$ on a faulty path in $\langle B, \eta\rangle$.

$$
\begin{array}{lccccccccc}
B: & (q_1, 0.4) & \xrightarrow{a} & (q_2, 0) & \xrightarrow{2} & (q_2, \infty) & \xrightarrow{\epsilon} (q_f, \infty) & \xrightarrow{a} & (q_2, 0) \\
\mathbf{O}_\eta B: & (q_1, 0.4) & \xrightarrow{a} & \{(q_1, 0), (q_2, 0)\} & \xrightarrow{2} & \{q_1, q_2, q_f\} \times \{\infty\} & & \xrightarrow{a} & \{(q_2, 0)\} \\
\mathcal{D}_\eta B: & \textbf{safe} & \xrightarrow{a} & \textbf{safe} & \xrightarrow{2} & \textbf{maybe faulty} & & \xrightarrow{a} & \textbf{faulty}
\end{array}
$$

The diagnoser of [53] works by keeping track of pairs of states and polyhedra and simulating all possible reachable configurations after each observable transition and after each fixed discrete time step of value `TO`. We claim, and show in the next section, that the powerset automaton of A_κ is an observer of A_κ, very similar to the machine constructed in [53].

9.2 Diagnosis with Automata On Timed Structures

Fix for all this section a finite alphabet Σ, $\mathcal{S} = \langle V, \hookrightarrow, U\rangle$ a timed structure and fix $A_\kappa = \langle A, \mathbf{T}^\kappa\rangle \in \mathbb{A}(\mathcal{S}, \Sigma)$, a controlled automaton on $\mathcal{S}$. We write $A = \langle Q, I, T, F\rangle$. We recall that the powerset automaton of A_κ is written $\langle \mathbf{D}_\kappa A, \mathbf{T}^{\mathbf{D}\kappa}\rangle$ and is an automaton on the timed structure $\mathbf{D}_{\kappa,A} S = \langle \mathbf{M}_Q V, \hookrightarrow_{A,\kappa}, \mathbf{M}_Q U\rangle$ defined as the power construction (c.f section 5.1) of S. Recall that we defined for all $\boldsymbol{\nu} \in \mathbf{M}_Q V$, $\mathbf{supp}_\epsilon(\boldsymbol{\nu})$ to be the set of states, q, of A such that $\boldsymbol{\nu}$ maps it to an empty set but after some delay d, $\mathbf{U}_d(\boldsymbol{\nu})$ no longer maps it to an empty set. Recall

also that we extended update $\mathbf{U}_\gamma$ with $\gamma \in \Sigma \uplus \mathbb{R}_{\geq 0}$ into function $\mathbf{U}_w$ for all $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$ by sequentially composing the updates.

Let us define the function $\mathbf{conf}$ from $\mathbf{M}_Q V$ to $\mathcal{P}(C_A)$ as follow:

$$\mathbf{conf}(\boldsymbol{\nu}) = \{(q, v) \in C_A \mid q \in Q, v \in \boldsymbol{\nu}(q)\}$$

We define also $\mathbf{conf}_\epsilon$ from $\mathbf{M}_Q V$ to $\mathcal{P}(C_A)$:

$$\mathbf{conf}_\epsilon(\boldsymbol{\nu}) = \{(q, v) \in C_A \mid q \in Q, \exists d \in \mathbb{R}_{\geq 0}, v \in \mathbf{U}_d(\boldsymbol{\nu})(q)\}$$

By definition $\mathbf{supp}(\boldsymbol{\nu}) = \{q \in Q \mid \exists v \in V, (q, v) \in \mathbf{conf}(\boldsymbol{\nu})\}$ and $\mathbf{supp}_\epsilon(\boldsymbol{\nu}) = \{q \in Q \mid \exists v \in V, (q, v) \in \mathbf{conf}_\epsilon(\boldsymbol{\nu})\}$. We obtain directly from proposition 5.1.4 that for all $w \in (\Sigma \uplus \mathbb{R}_{\geq 0})^*$,

- $\mathbf{conf}(\mathbf{U}_w(\mathbf{D}_\kappa c_I)) = \bigcup_{c_i \in I} \mathbf{Reach}(\mathbf{T}^\kappa, w, c_i)$
- $\mathbf{conf}_\epsilon(\mathbf{U}_w(\mathbf{D}_\kappa c_I)) = \bigcup_{d \in \mathbb{R}_{\geq 0}} \bigcup_{c_i \in I} \mathbf{Reach}(\mathbf{T}^\kappa, w \cdot d, c_i)$

We also know that $\mathbf{D}_\kappa A$ is deterministic according to the same proposition 5.1.4. This exactly means that $\mathbf{D}_\kappa A$ can be used as an observer by simply converting its configurations into sets of configurations of A using $\mathbf{conf}$. It has an interest if and only if $\mathbf{D}_\kappa A$ can be computed.

We explore below how the powerset automaton can be turned into a diagnoser. This is possible if we consider, as it is done in [53], the following hypotheses:

- The set of states of A_κ is divided into faulty and non faulty state, say Q_f and Q_s respectively.
- In A_κ, all states reachable from a faulty states are faulty.
- A path of T_A starting in an initial configuration is faulty if and only if it ends in a faulty state.

As this is explained in [53], it is easy to transform an automaton to an automaton satisfying the above condition. We show on example 9.2.1 how to transform $\langle \mathcal{B}, \mathbf{T}^\eta \rangle$ defined in example 9.1.1.

Example 9.2.1. We duplicate q_1 and q_2 to define a faulty version of them, q_1^f and q_2^f. The transitions $(q_f, v, \mathbf{0}, q_2)$ with $v \in \{1, \infty\}$ are then transformed into $(q_f, v, \mathbf{0}, q_2^f)$. And all transitions between q_1 and q_2 are duplicated on q_1^f and q_2^f. We get the automaton describe in figure 9.2 where red states are faulty states.

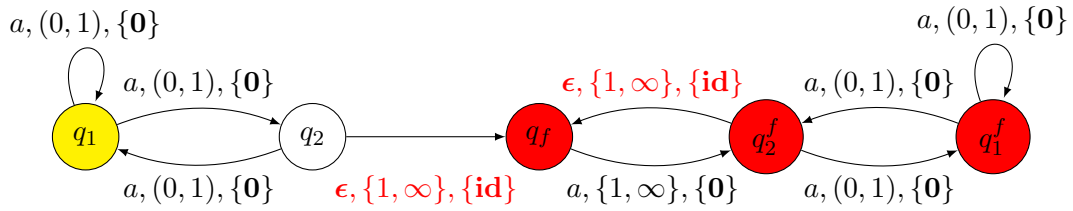


Figure 9.2: A one-clock timed automata $\mathcal{B}$ with a faulty transition

Provided that A_κ satisfies those hypotheses we can deport the detection of a faulty transition onto the detection of faulty states. Now a path is faulty if and only if it ends in a faulty state. So, given that we have an observer, if we observe a set of configurations, call it E, three cases are possible :

- $E \cap (Q_f \times V) = \emptyset$, in which case we are sure the path is non-faulty. We say that E is *safe*.
- $E \subseteq (Q_f \times V)$, in which case we are sure the path is faulty. We say that E is *faulty*.
- $E \cap (Q_f \times V) \neq \emptyset$ and $E \not\subseteq (Q_f \times V)$, in which case we don't know for sure if the path is faulty. We say that E is *maybe faulty*.

Hence if we define the function $\mathbf{diag}$ from $\mathbf{M}_Q V$ to $\mathcal{P}(C_A)$ as follow:

$$\mathbf{diag}(\boldsymbol{\nu}) = \begin{cases} \textbf{safe} & \text{if } \mathbf{conf}(\boldsymbol{\nu}) \text{ is safe} \\ \textbf{maybe faulty} & \text{if } \mathbf{conf}(\boldsymbol{\nu}) \text{ is maybe safe} \\ \textbf{faulty} & \text{if } \mathbf{conf}(\boldsymbol{\nu}) \text{ is faulty} \end{cases}$$

we can use $\mathbf{D}_\kappa A$ as a diagnoser by simply converting its configurations using $\mathbf{diag}$. And again it can be used as such if and only if it can be computed.

We can even obtain more information and predict if and when a configuration can turn into *maybe faulty* sometime in the future providing that we do not observe any action.

We define also $\mathbf{conf}_\epsilon$ from $\mathbf{M}_Q V$ to $\mathcal{P}(C_A)$:

$$\mathbf{diag}_\epsilon(\boldsymbol{\nu}) = \begin{cases} \textbf{always safe if silent} & \text{if } \mathbf{conf}_\epsilon(\boldsymbol{\nu}) \text{ is safe} \\ \textbf{soon maybe faulty if silent} & \text{if } \mathbf{conf}_\epsilon(\boldsymbol{\nu}) \text{ is maybe safe} \\ \textbf{always faulty if silent} & \text{if } \mathbf{conf}_\epsilon(\boldsymbol{\nu}) \text{ is faulty} \end{cases}$$

Remark 9.2.1. Suppose $\mathbf{diag}_\epsilon(\boldsymbol{\nu}) =$ **soon maybe faulty if silent**, we can actually compute when exactly the configuration will become **soon maybe faulty if silent** by computing

$$\inf\{d \in \mathbb{R}_{\geq 0} \mid \mathbf{conf}_\epsilon(\mathbf{U}_d(\boldsymbol{\nu})) \text{ is } \textbf{soon maybe faulty if silent}\}$$

Notice that the function $\mathbf{conf}$ and $\mathbf{diag}$ are based on *markings* and never take in account the state reached by the powerset automaton. This is mainly because, as you recall from section 5.1, the powerset automaton of an automaton with silent transitions cannot store statically in its state the information of the reachable states of A_κ. A state of the powerset automaton stores two informations: first the states of A_κ reached directly after the observed action, and then the states of A_κ not reached yet but reachable through an unobservable sequence of action. Though it does not know when exactly the not-yet reached states will be, indeed, reached. Still we could provide just from observing the states of $\mathbf{D}_\kappa A$ the information that a state is **soon maybe faulty** thanks to $\mathbf{supp}_\epsilon$; we would just not be able to tell exactly when the runs become **maybe faulty** given only the current state information.

In conclusion $\mathbf{D}_\kappa A$ can be easily translated into an observer or a diagnoser. We didn't impose any restriction on the timed structure until now so the method can be easily adapted for several models. The only, but nonetheless important point which makes $\mathbf{D}_\kappa A$ not directly and not always usable is that it may not be necessarily computed.

As discussed in section 5.1, the updates $\mathbf{U}_\sigma$ can be computed, given that A_κ is compatible with a computable guard base, that all updates are computable and that we can enumerate $\boldsymbol{\nu}$. Still to be able to compute $\mathbf{U}_d$ we need to be able to compute the reachable sets of A_κ reduced to its silent part (keeping only the silent transitions).

In the case of timed automata, all those hypotheses are satisfied and thus we can recover the result of [53] that it is possible to construct a diagnoser for timed automata. The three last chapters of this thesis are about how we try to take advantage of the particular shape of the powerset automaton of a (one-clock) timed automaton to (try to) make, by using pre-computations, a diagnoser more efficient than the one proposed in [53]. We choose to focus on the one-clock case to avoid the combinatorial explosion of the configuration space of n-clock timed automata both in the definition of the data structure we need and in the reasoning we will do on those structures. We discuss in the conclusion leads to extend our work to n-clock timed automata.

Our approach is summarized in figure 9.3. We know how to define $\mathbf{U}_d$ which takes a marking $\boldsymbol{\nu}$ and simulates a delay and a finite sequence of silent transitions to construct as result another marking $\boldsymbol{\nu}'$ (top arrow of figure 9.3). In this diagram $\boldsymbol{\nu}$ is difficult to represent and $\mathbf{U}_d$ is difficult to compute. We propose then a simpler representation of $\boldsymbol{\nu}$ into a *timed interval marking* represented by $\mathbf{I}$ and $\mathbf{I}'$ in figure 9.3 (chapter 10). This representation will also allow simpler simulation of time delays, converting it into a closure operator involving set operations ($\boldsymbol{\epsilon}$ in figure 9.3) and additions over the reals (chapter 11) ($+d$ in figure 9.3). Our work will hence consist of defining what the dotted arrows of figure 9.3 do.

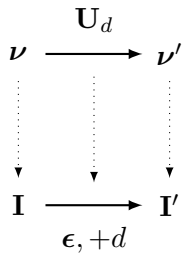


Figure 9.3: Work Plan

Chapter 10

Timed Set Algebra

A Framework for Precomputation

In this section, we focus on defining what will be the shape of $\mathbf{I}$ and $\mathbf{I}'$ in our work plan (c.f. figure 10.1. We said before that we sought to find a finite representation for markings that allow efficient computation of delays and action transitions. We propose *timed sets* and dedicate all this chapter to their definition, representation and their basic algebraic operations. Which makes it rather technical.

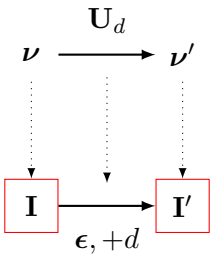


Figure 10.1: Work Plan - Timed Sets

Consider $M \in \mathbb{N}$ and $\boldsymbol{\nu}$ a value of the powerset automaton of some M-bounded *one-clock timed automata*. The motivation to introduce timed sets comes directly from our goal of finding a representation of a marking $\boldsymbol{\nu}$. Recall that $\boldsymbol{\nu}$ maps each state to a set of values that will evolve synchronously with time. The major complication, due to silent transition simulation, is that after some delay some new value can appear. This will create a new value of $\boldsymbol{\nu}'$ which can be very different from $\boldsymbol{\nu}$ and which is not easily deducible from $\boldsymbol{\nu}$. So the idea is the following: we try to concentrate all the information of the future values reachable (by a silent run) from $\boldsymbol{\nu}$ into a single object. The simplest object which does that is the function $f_{\boldsymbol{\nu}} : \mathbb{R}_{\geq 0} \to \mathbf{M}_E\mathbb{C}_M$ defined for all $d \in \mathbb{R}_{\geq 0}$ as $f_{\boldsymbol{\nu}}(d) = \mathbf{U}_d(\boldsymbol{\nu})$.

Based on this intuition we begin with this function (defined on the reals instead of $\mathbb{C}_M$) and construct simpler computable objects, from the atomic and simple timed sets (c.f. section 10.1) to the regular timed markings (c.f. section 10.3). Regular timed marking is the adapted data structure used in elements $\mathbf{I}$ and $\mathbf{I}'$ of figure 10.1. We give them a finite representation and describe how to compute basic set operations on them. We need before that to introduce the notion of regular union of intervals in section 10.2.

How to go from a marking of the powerset automaton to a regular timed marking will be the question addressed and solved in chapter 11.

10.1 Timed Sets

Recall that we restricted ourselves to one-clock timed automata. Let us fix $M \in \mathbb{N}$ for all this section.

Remark 10.1.1. We should work then in $\mathbb{C}_M = [0, M] \cup \{\infty\}$. For simplicity sake we will however work here in $\mathbb{R}_{\geq 0}$. In the end we will view elements of $\mathbb{R}_{\geq 0}$ as element of $\mathbb{C}_M$ considering the following transformation on $\mathbb{R}_{\geq 0}$:

$$\mathbf{p}_M(d) = \begin{cases} d & \text{if } d \in [0, M] \\ \infty & \text{if } d \in]M, +\infty[\end{cases}$$

For all $d_0, d \in \mathbb{R}_{\geq 0}$, recall that $\oplus_{\langle \mathbb{C}_M, \hookrightarrow \rangle}$ stands for the deterministic delay transition between elements of $\mathbb{C}_M$ (simple real value addition, with rounding to ∞ when going over M). Notice that $\mathbf{p}_M(d_0 + d) = \mathbf{p}_M(d_0) \oplus_{\langle \mathbb{C}_M, \hookrightarrow \rangle} d$.

Example 10.1.1. As an example, supposing that $M = 12$,

- $\mathbf{p}_M(144) = \infty$,
- $\mathbf{p}_M([6, 21[) = [6, 12] \cup \{\infty\}$,
- $\mathbf{p}_M(]12, 16]) = \{\infty\}$.

We need to introduce some notations we'll use for all this part. For any set $E \subseteq \mathbb{R}$, and any real $d \in \mathbb{R}$, we recall that

$$E + d = \{x + d, x \in E\}$$

For $r \in [\![0, M]\!]$, we define the following intervals of $\mathbb{R}$:

$$\updownarrow r = [r; +\infty) \qquad \uparrow r = (r; +\infty) \qquad \downarrow r = (-\infty; r) \qquad \updownarrow r = (-\infty; r].$$

We let $\widehat{\mathbb{C}}_M = \{\updownarrow r, \uparrow r \mid r \in [\![0, M]\!]\}$; in the sequel, elements of $\widehat{\mathbb{C}}_M$ are denoted with $\widehat{r}$. Similarly, we let $\underset{\sim}{\mathbb{C}}_M = \{\updownarrow r, \downarrow r \mid r \in [\![0, M]\!]\} \cup \{\downarrow +\infty\}$, with $\downarrow +\infty = \mathbb{R}$, and use notation $\underset{\sim}{r}$ for intervals in $\underset{\sim}{\mathbb{C}}_M$. The elements of $\widehat{\mathbb{C}}_M$ can be (totally) ordered using inclusion: we write $\widehat{r} \prec \widehat{r}'$ whenever $\widehat{r}' \subset \widehat{r}$ (so that $r < r'$ entails $\uparrow r \prec \uparrow r'$). $\widehat{\mathbb{C}}_M$ and $\underset{\sim}{\mathbb{C}}_M$ are stable by union and intersection and $\widehat{r} \cap \widehat{r}' = \max(\widehat{r}, \widehat{r}')$, $\widehat{r} \cup \widehat{r}' = \min(\widehat{r}, \widehat{r}')$, $\underset{\sim}{r} \cap \underset{\sim}{r}' = \min(\underset{\sim}{r}, \underset{\sim}{r}')$ and $\underset{\sim}{r} \cup \underset{\sim}{r}' = \max(\underset{\sim}{r}, \underset{\sim}{r}')$.

As just discussed in the introduction, a timed set is hence a function mapping every delay $d \in \mathbb{R}_{\geq 0}$ to the *value* (being a set) the timed set will have after d time units. Also in our context values can appear and grow but they can't disappear, decrease or remains constant. Formally,

Definition 10.1.1. *A timed set is a function from $\mathbb{R}_{\geq 0}$ to $\mathcal{P}(\mathbb{R}_{\geq 0})$ such that for all $d, d' \in \mathbb{R}_{\geq 0}$, $f(d) + d' \subseteq f(d + d')$.*

The set of timed sets is written $\mathcal{F}_{\mathbb{R}_{\geq 0}}$.

If $f \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$, $f(0)$ can be understood as the initial state of the set and $f(d)$ as its states after d times units.

Example 10.1.2. The function which maps every $d \in \mathbb{R}_{\geq 0}$ to $[d, d+3[$ represent a set of possible value of a clock evolving from $[0, 3[$ to $[12, 15[$ after 12 time units, $[144, 147[$ after 144 time units, etc.

We extend below set theoretic operations on timed sets. Let $f, f' \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$. For all $d \in \mathbb{R}_{\geq 0}$,

- $(f \cup f')(d) = f(d) \cup f'(d)$
- $(f \cap f')(d) = f(d) \cap f'(d)$
- $\overline{f}(d) = \overline{f(d)}$

with $\overline{E} = \mathbb{R}_{\geq 0} \setminus E$ for all subset E of $\mathbb{R}_{\geq 0}$.

Proposition 10.1.2. *For all $f, f' \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$, $f \cup f'$ and $f \cap f'$ are timed sets.*

Proof.

This is a simple consequence of the fact that for all $E, F \subseteq \mathbb{R}_{\geq 0}$ and $d \in \mathbb{R}_{\geq 0}$, $(E \cup F) + d = (E + d) \cup (F + d)$ and $(E \cap F) + d = (E + d) \cap (F + d)$. ■

Notice that in the other hand $\overline{f}$ is not a timed set because values which were appearing in f now are disappearing in $\overline{f}$. For example let f which maps every real $d \in \mathbb{R}_{\geq 0}$ to $f(d) = [0, d+3]$. $\overline{f}(0) + 1 = (-\infty, 1) \cup [4, +\infty)$ wich is not included in $\overline{f}(1) = (-\infty, 0) \cup [4, +\infty)$.

We define also for all family $(f_i)_{i \in \mathbb{N}} \in \mathcal{F}^{\mathbb{N}}_{\mathbb{R}_{\geq 0}}$, for all $d \in \mathbb{R}_{\geq 0}$,

$$(\bigcup_{i \in \mathbb{N}} f_i)(d) = \bigcup_{i \in \mathbb{N}} f_i(d)$$

Proposition 10.1.3. *For all family $(f_i)_{i \in \mathbb{N}} \in \mathcal{F}^{\mathbb{N}}_{\mathbb{R}_{\geq 0}}$, $\bigcup_{i \in \mathbb{N}} f_i$ is a timed set.*

Proof.

This is a consequence again of the fact that for all family $(E_i)_{i \in \mathbb{N}} \in \mathcal{P}(\mathbb{R}_{\geq 0})^{\mathbb{N}}$ and $d \in \mathbb{R}_{\geq 0}$, $(\bigcup_{i \in \mathbb{N}} E_i) + d = \bigcup_{i \in \mathbb{N}} (E_i + d)$. ■

Finally we define an operation $f + d$ between a timed set f and a positive real $d \in \mathbb{R}_{\geq 0}$ which is not the natural extension of the set operation $E + d$: for all $d' \in \mathbb{R}_{\geq 0}$,

$$(f + d)(d') = f(d + d')$$

This operation simulates a delay d on the time sets in such way that $(f + d)(0)$ takes the value of $f(d)$.

Example 10.1.3. For $f \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$ mapping $d' \in \mathbb{R}_{\geq 0}$ to $[d, d + 3[$. $(f + 12)(0) = [12, 15[$ and $(f + 12)(132) = [144, 147[$, etc.

Proposition 10.1.4. *For all $f \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$ and $d \in \mathbb{R}_{\geq 0}$, $f + d$ is a timed set.*

Proof.

By definition, since for all $d', d'' \in \mathbb{R}_{\geq 0}$, $f(d + d') + d'' \subseteq f(d + d' + d'')$. ■

We extend set inclusion. Let $f, f' \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$:

$$f \subseteq f' \iff \forall d \in \mathbb{R}_{\geq 0}, f(d) \subseteq f'(d)$$

with as consequence $f = f' \iff (f \subseteq f'$ and $f' \subseteq f)$. Other classical propositions can be retrieved such as $f \subseteq f' \implies f \cup f' = f'$, $f_1 \subseteq f'_1$ and $f_2 \subseteq f'_2 \implies f_1 \cup f_2 \subseteq f'_1 \cup f'_2$ or $f = f' \implies f \cup f'' = f' \cup f'' \ldots$

The timed set at the basis of all our representations of markings, is a timed set where values appear always with a fixed value. It will not be a restriction in the future to impose that those values are taken within $[\![0, M]\!]$.

Definition 10.1.5. *An* atomic timed set *is a timed set f for which exists $E \subseteq \mathbb{R}$ and $\widehat{r} \in \widehat{\mathbb{C}}_M$ such that for all $d \in \mathbb{R}_{\geq 0}$*

$$f(d) = (E + d) \cap \widehat{r}$$

f is then written $(E; \widehat{r})$

In an atomic timed set $(E, \widehat{r})$, E is called the *dynamic* part of the timed set and $\widehat{r}$ is called the *static* part. The reason of those appellations can be intuited on figure 10.2.

Remark 10.1.2. Notice that this writing for an atomic timed set is not always *unique*. Indeed $([12, 13], \uparrow 12)$ and $([12, 13], \uparrow 0)$ are two different writings for the same timed set. However if $E \not\subseteq \widehat{r}$, in this case there exists only one writing for $(E; \widehat{r})$.

Notice also that E can contain negative values. Because $\widehat{r} \subseteq \mathbb{R}_{\geq 0}$, f is a well-defined timed set.

Example 10.1.4. Figure 10.2 displays an example of an atomic timed set $f = (E; \uparrow 1)$, with $E = [-3; -1] \cup [0; 2]$. The picture displays the sets $f(0) = [1; 2]$ and $f(3) = [1; 2] \cup [3; 5]$.

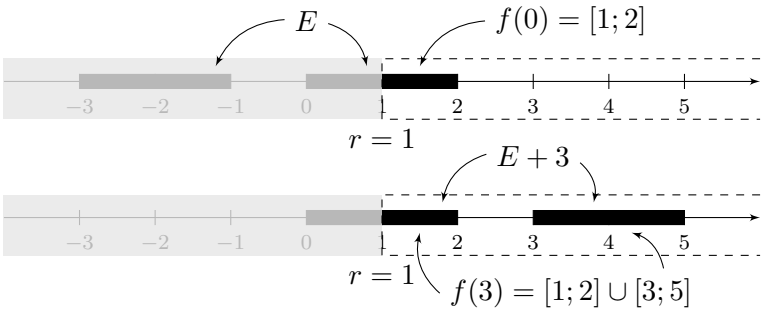


Figure 10.2: Example of an atomic timed set $f = ([-3; -1] \cup [0; 2]; \uparrow 1)$.

In the following proposition, we prove some rewriting properties along with the stability of atomic timed sets with regards to addition by a positive real. Recall that $(E; \widehat{r})$ represents a function so equality stands for function equality.

Proposition 10.1.6. *Let $E, F \subseteq \mathbb{R}$, $\widehat{r}, \widehat{r}' \in \widehat{\mathbb{C}}_M$ and $d \in \mathbb{R}_{\geq 0}$.*

- $(E; \widehat{r}) \cup (F; \widehat{r}) = (E \cup F; \widehat{r})$
- $(E; \widehat{r}) \cup (E; \widehat{r}') = (E; \min(\widehat{r}, \widehat{r}'))$
- $(E; \widehat{r}) + d = (E + d; \widehat{r})$

Proof.

By definition, using set theoretic operation factorization and by noticing that for all $E, F \subseteq \mathbb{R}_{\geq 0}$ and $d \in \mathbb{R}_{\geq 0}$, $(E \cup F) + d = (E + d) \cup (F + d)$ and $(E \cap F) + d = (E + d) \cap (F + d)$. ■

We also state a stability result for countable unions,

Proposition 10.1.7. *Let* $(E_i)_{i\in\mathbb{N}}\mathcal{P}(\mathbb{R})^{\mathbb{N}}$ *and* $\widehat{r} \in \widehat{\mathbb{C}}_M$ *.*

$$\bigcup_{i\in\mathbb{N}}(E_i;\widehat{r}) = (\bigcup_{i\in\mathbb{N}} E_i;\widehat{r})$$

Proof.

By definition again, using set theoretic operation factorization. ■

Definition 10.1.8. *A finite union of atomic timed sets is called a* simple timed set*.*

We will prove in chapter 11 that values of the powerset automaton of a one-clock timed automaton can be represented by simple timed sets. That's the reason why we focus, in the rest of this chapter, only on those particular timed sets.

Example 10.1.5. The simple timed set $([-3,1]\cup[1,2];\uparrow 1)\cup([-1.5,-0.5]\cup[0,0.5];\uparrow 2)$ is represented through its graph in figure 10.5. The x-axis represents the delay d to wait and the y-axis the set of value contained in $f(d)$. We highlight in particular $f(0) = [1,2]$ and $f(3) = [1,2.5]\cup[3,3.5]\cup[4,5]$.

This graph can be obtain by union of the graphs of $([-3,1]\cup[1,2];\uparrow 1)$ in figure 10.3 and $([-1.5,-0.5]\cup[0,0.5];\widehat{2})$ in figure 10.4. Their graphs are obtained by intersection of the gray zone, corresponding to the static part of the atomic timed sets (resp $\widehat{1}$ and $\widehat{2}$) and the red zone corresponding to the dynamic part of the atomic timed sets evolving with time (at absciss d we respresent respectively $[-3,-1]\cup[1,2]+d$ and $[-1.5,-0.5]\cup[0,0.5]+d$).

A simple timed set can be represented by a function from $\widehat{\mathbb{C}}_M$ to $\mathcal{P}(\mathbb{R})$, called *representation of the timed set* in the following way: let $\iota : \widehat{\mathbb{C}}_M \to \mathcal{P}(\mathbb{R})$ be such a function, then ι is a representation of the simple timed set f if and only if

$$f_\iota = \bigcup_{\widehat{r}\in\widehat{\mathbb{C}}_M} (\iota(\widehat{r});\widehat{r})$$

On the other hand if $f \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$ can be written $\bigcup_{i=0}^{m}(E_i;\widehat{r}_i)$, we can define for all $\widehat{r} \in \widehat{\mathbb{C}}_M$, and according to proposition 10.1.6 $f_{\iota_f} = f$ and ι_f is a

$$\iota_f(\widehat{r}) = \bigcup_{i\in[\![0,m]\!] \mid \widehat{r}_i=\widehat{r}} E_i$$

representation of f. However, such a function does not characterize f, in the sense that several different functions can define the same simple timed set.

Example 10.1.6. Fix $M = 1$. The three following writings represent the same simple timed function.

- $([-2,0] ;\updownarrow 0) \cup ([0,0] ;\uparrow 0) \cup ([0,3] ;\updownarrow 1) \cup (\emptyset ;\uparrow 1)$
- $([-2,0]\cup[1,3] ;\updownarrow 0) \cup (\emptyset ;\uparrow 0) \cup (]0,1[;\updownarrow 1) \cup (\emptyset ;\uparrow 1)$
- $([-2,0]\cup[1,3] ;\updownarrow 0) \cup ([-2,0]\cup[1,3] ;\uparrow 0) \cup ([-2,3] ;\updownarrow 1) \cup ([-2,3] ;\uparrow 1)$

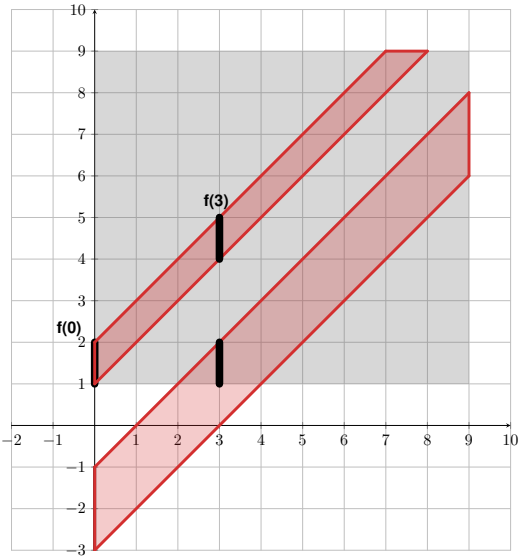


Figure 10.3: $f = ([-3, 1] \cup [1, 2]; \uparrow 1)$

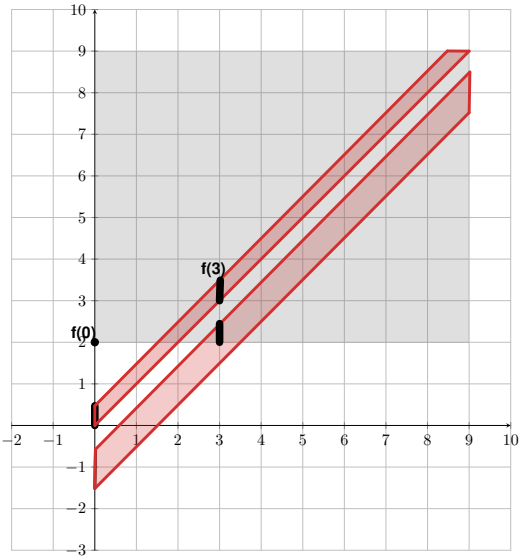


Figure 10.4: $f = ([-1.5, 0.5] \cup [0, 0.5]; \uparrow 2)$

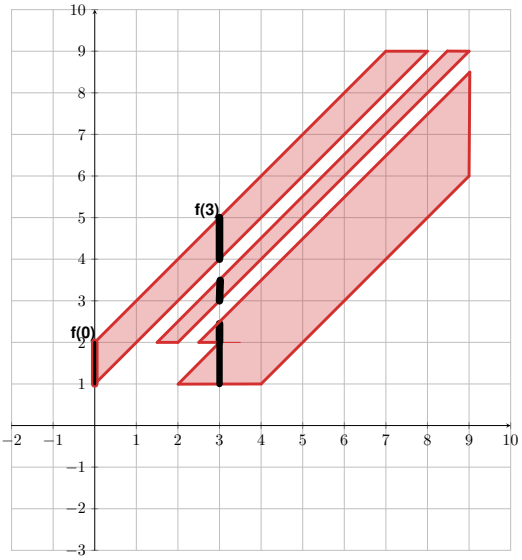


Figure 10.5: $f = ([-3, 1] \cup [1, 2]; \uparrow 1) \cup ([-1.5, -0.5] \cup [0, 0.5]; \uparrow 2)$

In the first writing of example 10.1.6 there is redundant information. Indeed $f(1) = [0,1] \cup [1,1] \cup [1,4] \cup \emptyset]$ which provides in three different ways that 1 is in the set : $([0,3]; \updownarrow 1)$ states that a value 1 will appear in exactly 1 time unit; but $([0,0]; \uparrow 0)$ states that a value 0 has just appeared and this value will be equal to 1 in 1 time unit, ... This representation is therefore not *minimal*.

We can remove redundant information by imposing that given a representation ι of a simple timed set, for all $\widehat{r} \neq \widehat{r}' \in \widehat{\mathbb{C}}_M$, $\iota(\widehat{r}) \cap \iota(\widehat{r}') = \emptyset$. As it is done in the second writing. We say that such a representation is *disjoint*. However a disjoint representation still does not characterize a simple timed set, since, for example in example 10.1.6, the information that $[1,3]$ is in the timed set from the beginning is here given by $([-2,0] \cup [1,3]; \updownarrow 0)$ but could be given by $(]0,3]; \updownarrow 1)$ if we choose to.

The last writing takes the opposite point of view and gathers all information available by imposing that for all $\widehat{r} \prec \widehat{r}' \in \widehat{\mathbb{C}}_M$, $\iota(\widehat{r}) \subseteq \iota(\widehat{r}')$. A representation satisfying such a hypothesis is called *complete*. It is still not a characterizing representation for the same reason as for the disjoint representations.

We could make both the disjoint and complete representations characterizing by imposing the following additional hypothesis, called slicing hypothesis, on the representation: for all $\widehat{r} \in \widehat{\mathbb{C}}_M \setminus \{\updownarrow 0\}$, $\iota(\widehat{r}) \cap \widehat{r} = \emptyset$. We could prove then that a simple timed set is characterized by a sliced disjoint (or complete) representation, called then *disjoint (or complete) normal form*, and that this normal form is computable. It is not interesting for the goal we have of building a diagnoser, hence we will save the reader from additional technical proof. It could be done by examining exactly what happens when time flows on the appearing points of the timed set (0,0^+,...,M,M^+, with r^+ meaning an infinitesimal time after r).

Given two arbitrary representations ι, ι' of two simple timed sets $f, f' \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$ we can easily construct the representation of $f \cup f'$ (which is obviously a simple timed set) in the following way: for all $\widehat{r} \in \widehat{\mathbb{C}}_M$ we define,

$$(\iota \cup \iota')(\widehat{r}) = \iota(\widehat{r}) \cup \iota'(\widehat{r}')$$

Proposition 10.1.9. *Let f, f' be two simple timed sets and ι, ι' be two representations of f. Then $(\iota \cup \iota')$ is a representation of $f \cup f'$.*

Proof.

By writing down the definition of a representation and according to proposition 10.1.6,

$$f \cup f' = \big[\bigcup_{\widehat{r} \in \widehat{\mathbb{C}}_M} (\iota(\widehat{r}); \widehat{r}) \big] \cup \big[\bigcup_{\widehat{r} \in \widehat{\mathbb{C}}_M} (\iota'(\widehat{r}); \widehat{r}) \big] = \bigcup_{\widehat{r} \in \widehat{\mathbb{C}}_M} (\iota(\widehat{r}) \cup \iota'(\widehat{r}); \widehat{r}) = \big[\bigcup_{\widehat{r} \in \widehat{\mathbb{C}}_M} ((\iota \cup \iota')(\widehat{r}); \widehat{r}) \big]$$

■

We can extend this result on countable unions of simple timed sets and prove that simple timed sets are stable by countable unions. Given $(f_i)_{i \in \mathbb{N}}$ be a family of simple timed sets and $(\iota_i)_{i \in \mathbb{N}}$ be a family of representations such that for all $i \in \mathbb{N}$, ι_i is a representation of f_i we can construct the representation of $\bigcup_{i \in \mathbb{N}} f_i$ in the following way: for all $\widehat{r} \in \widehat{\mathbb{C}}_M$ we define,

$$(\bigcup_{i \in \mathbb{N}} \iota_i)(\widehat{r}) = \bigcup_{i \in \mathbb{N}} \iota_i(\widehat{r})$$

Proposition 10.1.10. *Let $(f_i)_{i \in \mathbb{N}}$ be a family of simple timed sets and $(\iota_i)_{i \in \mathbb{N}}$ be a family of representations such that for all $i \in \mathbb{N}$, ι_i is a representation of f_i.*

Then $\bigcup_{i \in \mathbb{N}} \iota_i$ is a representation of $\bigcup_{i \in \mathbb{N}} f_i$.

Proof.

By writing down the definition of a representation and according to proposition 10.1.7. ■

Remark 10.1.3. Besides the use we make of representation in the proofs of chapter 11, representation are an especially important object because they represent exactly the way we encoded simple timed sets in our prototype **DOTA**. Simple timed sets are represented as arrays of length $|\widehat{\mathbb{C}}_M|$ where each cell contain $\iota(\widehat{r}_i)$ for some representation ι and where $\widehat{\mathbb{C}}_M$ is linearly ordered and $\widehat{r}_i$ is the i^{th} element of $\widehat{\mathbb{C}}_M$. ι is computed by defining a base representation of an atomic timed set and computing unions of representation following proposition 10.1.6 when needed.

Remark 10.1.4. The definition of atomic and simple timed sets is dependent on the integer M. We will always make sure in the rest of this thesis that M is fixed, before speaking of such timed sets. We will then implicitly mean that those atomic or simple timed sets depend on this fixed integer M.

We imposed some regularity on the appearing value of a simple timed set, by working on its *static* parts and imposing that they have to be within a finite set of values. We focus, in the next section on imposing some regularity on the moment in time they will appear.

10.2 Regular Timed Interval

Let us fix $M \in \mathbb{N}$ again for all this section.

Giving some continuity to the moment in time a value appears in an atomic timed set is possible using restrictions on its *dynamic* part. For example for an atomic timed set $f = ([1,2];\uparrow 3)$ we know that the value 3 is going to appear in exactly 1 time unit, will continue to appear during 1 time unit and will appear for the last time in 2 time unit.

Considering finite unions of intervals as dynamic parts for atomic timed sets is a good way to give the continuity we can observe in the values of the powerset automaton of a one-clock timed automaton. They also have the important property of being finitely and easily representable, and to allow efficient membership test, intersection and union computation.

Definition 10.2.1. *A* timed interval *is an atomic timed set* $(E;\widehat{r})$ *such that* E *is an interval of* $\mathbb{R}$.

Notice that this definition is valid because given an atomic timed set, all its possible writings have the same dynamic part (c.f. lemma 10.1.6).

A finite union of timed intervals is then a simple timed set and for any representation ι of a finite union of timed intervals, for all $\widehat{r} \in \widehat{\mathbb{C}}_M$, $\iota(\widehat{r})$ *is a finite union of intervals of* $\mathbb{R}$.

However finite unions of timed intervals are not enough to represent all the possible values a powerset automaton can take as we show in example 10.2.1.

Example 10.2.1. Consider the silent timed automaton of figure 10.6. In its powerset automaton, the initial marking $\boldsymbol{\nu}_I$ maps q_0 to $\{0\}$ and q_1 to $\emptyset$. We represent as an atomic timed set (in figure 10.6) the evolution of $\boldsymbol{\nu}_I(q_1)$ with time, let's write for all $d \in \mathbb{R}_{\geq 0}$ $f_I(d) = \mathbf{U}_d(\boldsymbol{\nu}_I)(q_1)$.

First notice that $f_I(0)$ is the closure by silent runs of $\boldsymbol{\nu}_I$, and because we can instantly make one, $f_I(0) = \{0\}$. Then, this value grows with time, but even if we can still take a silent transition for 1 time unit (strictly) we never add another possible value (since we cannot take the transition from q_1 to q_0), so $f_I(d) = \{d\}$ for $d \in [0,1)$. We have to wait 1 time unit exactly,

at which point a branching is possible: we can stay in q_1 or make two silent transitions, adding a new possible value 0 for q_1. Hence $f_I(1) = \{1, 0\}$. Again we find ourselves in the same situation and can wait for 1 time units at which point we'll add another 0 as possible value for q_1, therefore having $f_I(2) = \{2, 1, 0\}\ldots$

Such a timed state cannot be represented by a finite union of timed intervals, because finite unions of intervals of $\mathbb{R}$ must have an infimum, and once this infimum value has grown past M we can be sure no other value is going to appear. We need an infinite (countable) union of intervals of $\mathbb{R}$ to represent the timed set. In our case, we can represent f_I as the following simple timed set

$$f_I = (\bigcup_{n\in\mathbb{N}} [-n, -n]; \updownarrow 0)$$

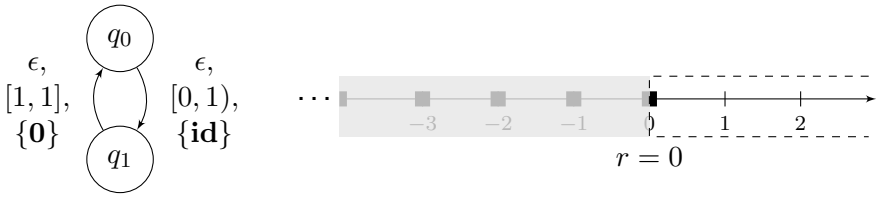


Figure 10.6: A silent timed automaton

To palliate this problem, one solution is to consider countable unions of timed intervals (as it is done in example 10.2.1). As we have seen in section 10.1, countable unions of timed interval are still simple timed sets. For any representation ι of a countable union of timed interval, for all $\widehat{r} \in \widehat{\mathbb{C}}_M$, $\iota(\widehat{r})$ *is a countable union of intervals of* $\mathbb{R}$.

Still, with countable unions of intervals, we introduce another problem which is that we can't always finitely represent such unions and test membership or compute unions and intersections. That's why we need to introduce a special class of countable unions of (real) intervals, which is both: wide enough to encompass all the possible shapes of simple timed set issued from the powerset automaton values representation (as we will prove in chapter 11) *and* finitely representable and allowing efficient computation. We name this class *regular unions of intervals*.

Definition 10.2.2. *A regular union of intervals is a subset of* $\mathbb{R}$ *of the form*

$$E = I \cup \bigcup_{k\geq m} J + k \cdot p$$

where I *and* J *are finite unions of interval and* m *and* p *are non-negative integers, and such that* $J \subseteq [0, p[$.

E *is then written* (I, J, p, m).

In a regular union of intervals (I, J, p, m), I is called the *base interval*, J is called the *repeated interval*, p is called the *period* and m the *offset*.

Remark 10.2.1. A finite union of interval is a regular union of interval represented by $(I, \emptyset, 0, 0)$.

Remark 10.2.2. The representation (I, J, p, m) for E is not *unique*, for several reasons. We could, for example, integrate into the base the first repetition of J or duplicate J and double the period, ...

We explore below the different behaviors of regular unions of intervals through union, intersection, and addition of a real number. Recall that given two sets, $E - F$ stands for the set of all $x - f$ with $x \in E$ and $f \in F$.

Proposition 10.2.3. *Let $E = (I, J, p, m)$ and $E' = (I', J', p', m')$ be two regular unions of intervals, K be an interval with a finite upper bound, and $d \in \mathbb{R}$. Then $E \cup E'$, $E \cap E'$, $E - K$ and $E + d$ are regular unions of intervals. $E \cap K$ is a finite union of intervals.*

Proof.

Recall that we have :

$$E = I \cup \bigcup_{k=m}^{+\infty} J + k \cdot p \qquad\qquad E' = I' \cup \bigcup_{k=m'}^{+\infty} J' + k' \cdot p'.$$

Let $q = \mathbf{lcm}(p, p')$ and $a, a' \in \mathbb{N}$ such that $q = a \cdot p$ and $q = a' \cdot p'$. We let also $b, b' \in \mathbb{N}$, respectively the quotient of the Euclidian division of m by a and of m' by a'. Finally let $c = \max(b + 1, b' + 1)$ Then we can write

$$E = \left(I \cup \bigcup_{k=m}^{c \cdot a - 1} J + k \cdot p \right) \cup \bigcup_{k=c}^{+\infty} \left[\left(\bigcup_{r=0}^{a-1} J + r \cdot p \right) + k \cdot a \cdot p \right].$$

and

$$E' = \left(I' \cup \bigcup_{k=m'}^{c \cdot a' - 1} J' + k \cdot p' \right) \cup \bigcup_{k=c}^{+\infty} \left[\left(\bigcup_{r=0}^{a'-1} J' + r \cdot p' \right) + k \cdot a' \cdot p' \right].$$

Let's write

$$I_1 = I \cup \bigcup_{k=m}^{c \cdot a - 1} J + k \cdot p \qquad\qquad I'_1 = I' \cup \bigcup_{k=m'}^{c \cdot a' - 1} J' + k \cdot p'$$

$$J_1 = \bigcup_{r=0}^{a-1} J + r \cdot p \qquad\qquad J'_1 = \bigcup_{r=0}^{a'-1} J' + r \cdot p'$$

Since $J \subseteq [0, p)$, we have $J_1 \subseteq [0, a \cdot p) = [0, q)$, and similarly for J'_1.

- We focus here on $E \cup E'$. Taking the union of the equalities above, we get

$$E \cup E' = I_1 \cup I'_1 \cup \bigcup_{k=c}^{+\infty} \left[J_1 \cup J'_1 + k \cdot q \right].$$

  Since $J_1 \cup J'_1 \subseteq [0, q)$, $E \cup E'$ is a regular union of intervals represented by $(I_1 \cup I'_1, J_1 \cup J'_1, c, q)$.

- We focus now on $E \cap K$. Let $l = \sup K \in \mathbb{R}$ since K as an upper bound. For all $k > \lceil \frac{l}{p} \rceil$, $J + kp \subseteq (l, +\infty)$. Hence we can write,

$$E \cap K = (I \cap K) \cup \bigcup_{k=m}^{\lceil \frac{l}{p} \rceil} (J + k \cdot p) \cap K$$

  which is a finite union of interval.

- We focus here on $E \cap E'$. As seen just before, any intersection between a regular union of intervals and a finite union of intervals creates a finite union of intervals. By distributivity there exists then a finite union of intervals, I'', such that we can write

$$E \cap E' = I'' \cup \left(\bigcup_{k=c}^{+\infty} [J_1 + k \cdot q] \cap \bigcup_{k=c}^{+\infty} \left[J_1' + k \cdot q \right] \right)$$

But because for all $k \geq c$, $J_1 + kq \subseteq [kq, kq+1)$ and $J_1' + kq \subseteq [kq, kq+1)$, we can write

$$E \cap E' = I'' \cup \bigcup_{k=c}^{+\infty} \left[J_1 \cap J_1' + k \cdot q \right]$$

Since $J_1 \cap J_1' \subseteq [0, q)$, $E \cap E'$ is a regular union of intervals represented by $(I'', J_1 \cap J_1', c, q)$.

- Finally to compute $E + d$, let $n = \lfloor \frac{d}{p} \rfloor$ (possibly negative) and $r = d - np < p$. Let $K_1 = J + r \cap [0, p)$ and $K_2 = J + r \cap [p, p+1)$. We know for sure that $J + r = K_1 \cup K_2$. We write then

$$\begin{aligned}
E + d &= I + d \ \cup \bigcup_{k=m}^{+\infty} J + kp + d \\
&= I + d \ \cup \bigcup_{k=m}^{+\infty} J + kp + np + r \\
&= I + d \ \cup \bigcup_{k=m}^{\max(m,n)-1} (J + r) + (k + n)p \ \cup \bigcup_{k=\max(m,n)+n}^{+\infty} (J + r) + kp \\
&= I'' \cup \bigcup_{k=\max(m,n)+n}^{+\infty} K_1 + kp \ \cup \bigcup_{k=\max(m,n)+n}^{+\infty} K_2 + kp
\end{aligned}$$

with $I'' = I + d \ \cup \bigcup_{k=m}^{\max(m,n)-1} (J + r) + (k + n)p$. I'' have been introduced to be sure that the first index of the infinite repetition part is a natural number. We define then $I^{(3)} = I'' \cup K_1 + (\max(m, n) + n)p$. Then :

$$\begin{aligned}
E + d &= I^{(3)} \cup \bigcup_{k=\max(m,n)+n+1}^{+\infty} K_1 + kp \ \cup \bigcup_{k=\max(m,n)+n+1}^{+\infty} (K_2 - p) + kp \\
&= I^{(3)} \cup \bigcup_{k=\max(m,n)+n+1}^{+\infty} (K_1 \cup K_2 - p) + kp
\end{aligned}$$

with $K_1 \cup K_2 - p \subseteq [0, p)$. Which means that $E + d$ is a regular union of intervals represented by $(I^{(3)}, K_1 \cup K_2 - p, \max(m, n) + n + 1, p)$.

■

Remark 10.2.3. We will need the properties just proved in proposition 10.2.3 for the proofs in chapter 11. But more than its theoretic interest, this proof describes exactly how we will handle regular unions of intervals in **DOTA**. Regular unions of intervals are represented as a tuple of

four elements exactly as it is done in this section. The union is computed by first rewriting E and E' with the same period and then by computing the union of the bases and repeated parts – as it is done in the proof of proposition 10.2.3 exactly.

We also need to compute intersections of regular unions of intervals and finite unions of intervals in **DOTA**. It is done in the same way we have proved that such an intersection builds a finite union of intervals: by reducing the infinite repetition of J into a finite one.

In **DOTA** however we do not compute the addition by a real ($E+d$) as it is done in proposition 10.2.3. Comforted by the fact that $E + d$ is indeed a regular union of intervals, we only store at each point the accumulated delays which have been elapsed since the last observable action. We noticed indeed that every time we need to access to some information on the regular union of intervals – for example to compute the set of reached or soon to be reached states of the observed timed automaton – we do first some intersection of a finite union of intervals with an upper bound, say K. Because $(E + d) \cap K = (E \cap K - d) + d$ we can avoid making heavy computations of $E + d$ and instead make two simple computation of an addition of two finite unions of intervals with a real.

We prove one intermediate result, implemented in **DOTA**, and useful in the proof of the two succeeding propositions.

Lemma 10.2.4. *Let I and J be finite unions of interval and m and p be two integers and let*

$$E = I \cup \bigcup_{k \geq m} J + k \cdot p$$

Suppose there exists $b \in \mathbb{N}$ such that $J \subseteq [-b, b)$, then E is a regular union of interval.

Proof.

Though J is not included in $[0, p)$, it is not difficult to adapt the repeated part so it fits in $[0, p)$. We show how below.
First consider $J' = J + (\lfloor \frac{b}{p} \rfloor + 1)p$, we know have $J' \subseteq [0, 2b + 1)$ and $E = I \cup \bigcup_{k=m+\lfloor \frac{b}{p} \rfloor+1} J' + k \cdot p$.

Let $N = \lceil \frac{2b+1}{p} \rceil$ and

$$J'' = \bigcup_{i=0}^{N-1} J' - i \cdot p \cap [0, p)$$

by definition $J'' \subseteq [0, p)$.

$$\begin{aligned} E &= I \cup \bigcup_{k \in \mathbb{N}} J' + k \cdot p \\ &= I \cup \bigcup_{l \in \mathbb{N}} \left(\bigcup_{k \in \mathbb{N}} J' + k \cdot p \right) \cap [l \cdot p, (l+1) \cdot p) \\ &= I \cup \left(\left(\bigcup_{k \in \mathbb{N}} J' + k \cdot p \right) \cap [0, N \cdot p) \right) \cup \bigcup_{l=N}^{+\infty} \left(\bigcup_{k \in \mathbb{N}} J' + k \cdot p \right) \cap [l \cdot p, (l+1) \cdot p) \end{aligned}$$

but since $J' \subseteq [0, N \cdot p)$ it means that for all $l, k \in \mathbb{N}$,

$$J' + k \cdot p \cap [l \cdot p, (l+1) \cdot p) \neq \emptyset \iff k < l + 1 \text{ and } k > l - N$$

Hence, if we write $I' = I \cup \big((\bigcup_{k\in\mathbb{N}} J' + k\cdot p) \cap [0, N\cdot p)\big)$,

$$\begin{aligned}
E &= I' \cup \bigcup_{l=N}^{+\infty} \left(\bigcup_{k=l-N+1}^{l} J' + k\cdot p \right) \cap [l\cdot p, (l+1)\cdot p) \\
&= I' \cup \bigcup_{l=N}^{+\infty} \left(\bigcup_{k=0}^{N-1} J' + (k+l-N+1)\cdot p \right) \cap [l\cdot p, (l+1)\cdot p) \\
&= I' \cup \bigcup_{l=N}^{+\infty} \bigcup_{k=0}^{N-1} \big(J' + (k+l-N+1)\cdot p \cap [l\cdot p, (l+1)\cdot p) \big) \\
&= I' \cup \bigcup_{l=N}^{+\infty} \bigcup_{k=0}^{N-1} \big(J' + (k-N+1)\cdot p + l\cdot p \cap (\ [0,p) + l\cdot p) \big) \\
&= I' \cup \bigcup_{l=N}^{+\infty} \bigcup_{k=0}^{N-1} \big(J' + (k-N+1)\cdot p \cap\ [0,p) \big) + l\cdot p \\
&= I' \cup \bigcup_{l=N}^{+\infty} \left(\bigcup_{k=0}^{N-1} J' - k\cdot p \cap\ [0,p) \right) + l\cdot p \\
&= I' \cup \bigcup_{l=N}^{+\infty} J'' + l\cdot p
\end{aligned}$$

In conclusion, E is a regular union of intervals represented by (I', J'', N, p). ■

Remark 10.2.4. This adaptation of the repeating interval is implemented in **DOTA** and in the precomputation process.

The following proposition is again a stability property useful in chapter 11. It is a direct consequence of lemma 10.2.4.

Proposition 10.2.5. *For all regular union of intervals E, and for all bounded interval K, $I - K$ is a regular union of intervals.*

Proof.

Let $E = I \cup \bigcup_{k=m}^{+\infty} J - kp$.

$$E - J = (I - K) \cup (\bigcup_{k=m}^{+\infty} J - K + kp)$$

is a regular union of timed interval since $J - K \subseteq [-\sup K, p - \inf K]$ and according to lemma 10.2.4. ■

Below we prove a property of regular unions of intervals at the heart of our future proof that reachable values in a powerset automaton can be represented by regular unions of intervals.

Proposition 10.2.6. *Let I, J be two finite unions of intervals such that J has a lower bound, $n \in \mathbb{N}$ and $p_1, \ldots, p_n$ be n strictly positive integers.*

$$E = I \cup \bigcup_{k_1,\ldots,k_n \in \mathbb{N}^n} J + k_1\cdot p_1 + k_2\cdot p_2 + \ldots k_n \cdot p_n$$

is a regular union of interval.

Proof.

First suppose that $\sup J = +\infty$, then $E = I \cup J$ is a finite union of interval. Suppose now that $\sup J = j_+ \in \mathbb{R}_{\geq 0}$.

Let $p = \mathbf{lcm}_{1 \leq i \leq n}(p_i)$. For all $1 \leq i \leq n$, let's define $a_i = \frac{p}{p_i} \in \mathbb{N}$.

$$
\begin{aligned}
E &= I \cup \bigcup_{q_1,\dots,q_n \in \mathbb{N}, r_1,\dots,r_n \in [\![1,a_1]\!] \times \cdots \times [\![1,a_n]\!]} J + (q_1 \cdot a_1 + r_1)p_1 + \cdots + (q_n \cdot a_n + r_n) \cdot p_n \\
&= I \cup \bigcup_{q_1,\dots,q_n \in \mathbb{N}} \bigcup_{r_1,\dots,r_n \in [\![1,a_1]\!] \times \cdots \times [\![1,a_n]\!]} J + (q_1 \cdot a_1 + r_1)p_1 + \cdots + (q_n \cdot a_n + r_n) \cdot p_n \\
&= I \cup \bigcup_{q_1,\dots,q_n \in \mathbb{N}} \left(\bigcup_{r_1,\dots,r_n \in [\![1,a_1]\!] \times \cdots \times [\![1,a_n]\!]} J + r_1 \cdot p_1 + \cdots + r_n \cdot p_n \right) + (q_1 + \cdots + q_n) \cdot p \\
&= I \cup \bigcup_{k \in \mathbb{N}} J' + k \cdot p
\end{aligned}
$$

with $J' = \left(\bigcup_{r_1,\dots,r_n \in [\![1,a_1]\!] \times \cdots \times [\![1,a_n]\!]} J + r_1 \cdot p_1 + \cdots + r_n \cdot p_n \right)$. Let $j'_+ = j_+ + \Sigma_{i=1}^{n}(a_i - 1) \cdot p_i$ and $N = \lceil \frac{j'_+}{p} \rceil$. We have then $J' \subseteq [0, N \cdot p)$. E is then a regular union of intervals according to lemma 10.2.4. ■

Remark 10.2.5. Again, this proof gives with the theoretical result, the algorithm to convert the expression given in the statement of the proposition – which is the archetype of the expression we have to work with at some point during the precomputation in **DOTA** – into a regular union of intervals. We implemented this conversion in **DOTA** exactly in the way it is depicted in the proof of 10.2.6

Regular unions of intervals can store an infinite number of possible values of a clock while being finitely representable and so that we can efficiently compute unions, additions by a real, and even complex repetitions with multiple periods. They will be useful to the representation of the values reached by the powerset automaton, we will use them particularly in the form of *timed regular intervals* defined below.

Definition 10.2.7. *A* timed regular interval *is an atomic timed set* $(-E; \widehat{r})$ *with* E *being a regular union of intervals.*

Remark 10.2.6. Notice that technically, the dynamic part of a timed regular interval is not a regular union of intervals but *the opposite of a regular union of intervals*.

As one can expect we allow finite unions of timed regular intervals and call them *regular timed sets*. In particular, regular timed sets are simple timed sets.

Regular timed sets are at the heart of the representation of a marking $\boldsymbol{\nu}$ reachable in the powerset automaton, but they are not a marking themselves. We use them to define *regular timed markings* in section 10.3, which are exactly the object we use to represent reachable markings.

Remark 10.2.7. Following remark 10.1.4 the definition of regular timed sets is also dependent on the integer M. We will use the same precaution as for atomic and simple timed sets.

10.3 Regular Timed Markings

Let us fix M again for all this section. We also fix a finite set Q, which can be considered as the set of state of a one-clock timed automaton.

Timed markings are functions from Q to the set of timed sets. Intuitively, a timed marking $\mathbf{m}$, maps each state, q, to a timed set representing the evolution over time of the possible values that can take the clock *while being in* q. We write $\mathbb{M}_Q$ for the set of timed markings.

Example 10.3.1. For example, if we work with the 2-bounded one-clock timed automaton of figure 10.7 we can represent the evolution of the possible values of x by the two timed sets on the right. The higher time set represent the possible values of x when we are in q_0, that is 0 (initial value), 0 in 2 time units (after one cycle), 0 in 4 time units (after two cycles), etc. The lower time set represent the possible values of x when we are in q_1, that is 1 in 1 time unit (first time we take the transition), 1 in 3 time units (after one cycle and taking the transition from q_0 to q_1), etc.

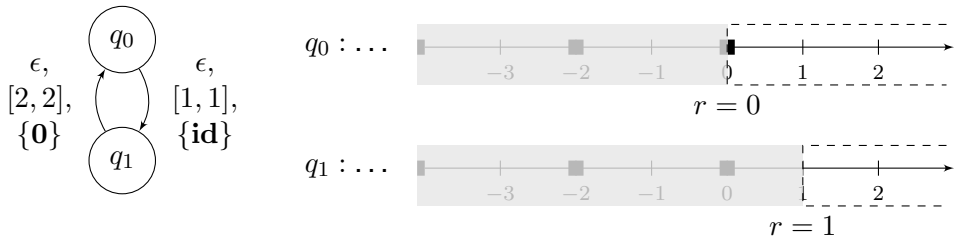


Figure 10.7: Another silent timed automaton

As we discussed it in the introduction of this chapter we can intuitively associate with each marking a timed marking – and conversely. We can't formally make a bijection between the two spaces for two reasons. First because a marking $\boldsymbol{\nu}$ is different from $\mathbf{U}_0(\boldsymbol{\nu})$ which applies all possible silent transitions on the marking and possibly adds new possible values of the clock. Then because a marking takes value in $\mathcal{P}(\mathbb{C}_1)$ and a timed marking $\mathcal{P}(\mathbb{R}_{\geq 0})$.

However, this won't bother us for this work. We will show how the correspondence is made in chapter 11. In that chapter we will also prove that, in fact, markings obtained by execution of the powerset automaton of a one-clock automaton can be put in correspondence to a *regular timed markings* defined below.

Definition 10.3.1. *Regular timed markings are function from Q to $\mathcal{P}(\mathbb{R})$, mapping each state in Q to a* regular timed set.

We can also define a *simple timed marking* as a function from Q to $\mathcal{P}(\mathbb{R})$, mapping each state in Q to a *simple timed set*. A regular timed marking is then a simple timed marking;

Timed markings (and then regular timed markings) behave, like timed sets, well with regards to set theoretic operations. We can extend them on timed markings as follows: let $\mathbf{m}$ and $\mathbf{m}'$ be two timed markings, $d \in \mathbb{R}_{\geq 0}$, for all $q \in Q$,

- $(\mathbf{m} \cup \mathbf{m}')(q) = \mathbf{m}(q) \cup \mathbf{m}'(q)$
- $(\mathbf{m} \cap \mathbf{m}')(q) = \mathbf{m}(q) \cup \mathbf{m}'(q)$
- $(\mathbf{m} + d)(q) = \mathbf{m}(q) + d$

According to proposition 10.1.2 and 10.1.4, those operations define new timed markings.

Given a family $(\mathbf{m}_i)_{i\in\mathbb{N}}$ of timed markings, we also define for all $q \in Q$:

$$(\bigcup_{i\in\mathbb{N}} \mathbf{m}_i)(q) = \bigcup_{i\in\mathbb{N}} \mathbf{m}_i(q)$$

According to proposition 10.1.3, the countable union of timed markings are timed markings.

We can also extend the notion of representation of simple timed sets to simple timed markings. A *representation, $\boldsymbol{\iota}$, of a simple timed marking*, $\mathbf{m}$, is a function from $Q \times \widehat{\mathbb{C}}_M$ to $\mathcal{P}(\mathbb{R})$ such that for all $q \in Q$,

$$\mathbf{m}(q) = \bigcup_{\widehat{r}\in\widehat{\mathbb{C}}_M} (\boldsymbol{\iota}_{\mathbf{m}}(q,\widehat{r});\widehat{r})$$

Conversely, if $\mathbf{m}$ is a simple timed marking we can define for all $q \in Q$ and for all $\widehat{r} \in \widehat{\mathbb{C}}_M$,

$$\boldsymbol{\iota}_{\mathbf{m}}(q,\widehat{r}) = \iota_{\mathbf{m}(q)}(\widehat{r})$$

which is a representation of $\mathbf{m}$.

Finally, we introduce an atomic decomposition for timed markings. An *atomic marking* , $\mathbf{m}$, is a timed marking such that there exists $q \in Q$ such that $\mathbf{m}(q)$ is an atomic timed set, $(E;\widehat{r})$ and for all $q' \in Q \setminus \{q\}$ $\mathbf{m}(q')$ is the empty timed set (mapping all $d \in \mathbb{R}_{\geq 0}$ to $\emptyset$). Such a marking is written $\mathbf{m}_{q,(E;\widehat{r})}$.

An atomic marking $\mathbf{m}_{q,(E;\widehat{r})}$ is a *regular atomic marking* if and only if E is a regular union of intervals.

Every simple timed marking can be decomposed as a finite union of simple timed markings.

Proposition 10.3.2. *Let $\mathbf{m}$ be a simple timed marking. There exists a finite number of pairs of states and atomic timed sets, $(q_1,(E_1;\widehat{r}_1)),\ldots,(q_n,(E_n;\widehat{r}_n))$, such that*

$$\mathbf{m} = \bigcup_{i=1}^{n} \mathbf{m}_{q_i,(E_i;\widehat{r}_i)}$$

Moreover, if $\mathbf{m}$ is regular then for all $i \in [\![1,n]\!]$, $(E_i;\widehat{r}_i)$ can be chosen to be regular ; conversely if for all $i \in [\![1,n]\!]$, $(E_i;\widehat{r}_i)$ is regular then $\mathbf{m}$ is regular too.

Proof.

Just recall that for all $q \in Q$, $\mathbf{m}(q) = \bigcup_{\widehat{r}\in\widehat{\mathbb{C}}_M}(\boldsymbol{\iota}_{\mathbf{m}}(q,\widehat{r});\widehat{r})$. Hence we can write

$$\mathbf{m} = \bigcup_{q\in Q} \bigcup_{\widehat{r}\in\widehat{\mathbb{C}}_M} \mathbf{m}_{q,(\boldsymbol{\iota}_{\mathbf{m}}(q,\widehat{r});\widehat{r})}$$

▶ If $\mathbf{m}$ is regular then we can choose a representation of $\mathbf{m}$ such that for all $q \in Q$ and $\widehat{r} \in \widehat{\mathbb{C}}_M$, $\boldsymbol{\iota}_{\mathbf{m}}(q,\widehat{r})$ is a regular union of intervals.

▶ If conversely for all $i \in [\![1,n]\!]$, E_i is a regular union of intervals, then by proposition 10.2.3, $\mathbf{m}$ is regular. ■

Remark 10.3.1. Following remarks 10.1.4 and 10.2.7 the definition of regular timed markings is also dependent on the integer M and the set Q. We will use the same precaution as for regular timed sets.

In chapter 11 we will construct a diagnoser working on timed markings, similar to the power set automaton. We will focus first on the definition of this automaton and the proof it simulates the powerset automaton, and in a second time we will focus on proving that this automaton works actually on regular timed markings and expose how updates can be efficiently computed.

Chapter 11

Precomputing with Timed Sets

Theoretical and Algorithmic Aspects

Now that we introduced regular timed markings and defined the nature of I and I' in our work plan (figure 11.1), we focus on describing the transformation from ν to I, i.e. from markings to regular timed markings, and the translation of update $\mathbf{U}_d$. The difficulty is that our translation of a marking into a regular timed marking does not immediately allow the simulation of $\mathbf{U}_d$ by a simple addition $+d$. First, the regular timed marking needs to be *closed under silent transitions*. To do so we need to introduce an operator ϵ which completes the regular timed markings adding all possible future values accessible through silent runs. Once completed using ϵ a delay can be simulated on the regular timed marking by simple addition. This is proved in theorem 11.1.7, whose conclusion ensures the coherence of our work plan scheme in figure 11.1 : the translation of $\mathbf{U}_d(\nu)$ is equal to the translation of ν closed by ϵ to which d is added.

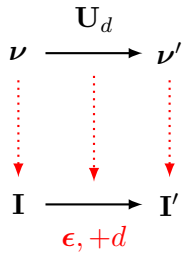


Figure 11.1: Work Plan - ϵ-closure

This allows us to construct a diagnoser on a new timed structure based on regular timed markings. This is theorem 11.1.9, which is the important result of section 11.1.

At this point, the diagnoser would have the good properties of using finite representation for values (regular timed markings) and efficient ways of computing action transition using set operations. Delay transitions could be efficiently computed using simple addition if we can efficiently compute the closure operation ϵ.

Section 11.2 is dedicated to the rather technical proof of theorem 11.2.1 that indeed ϵ :

- maps a regular timed marking to another regular timed marking,
- and as importantly can be efficiently computed using precomputed tables of regular timed markings.

11.1 Timed Marking Diagnoser

Fix in all this section $M \in \mathbb{N}$ and Σ a finite alphabet. We also fix $\langle A, \kappa \rangle \in \Sigma\mathbf{TA}^1_M$ an M-bounded one-clock Σ-timed automaton. We write $A = \langle Q, I, T, F \rangle$.

In this section, we construct an efficient diagnoser similar to the powerset automaton. This construction is made by putting in evidence a functional bisimulation from *the timed marking structure* – defined below – to the $(M, 1)$-clock structure.

To define the *timed marking structure* we need to define a delay transition and a set of updates which will simulate respectively the effects of $\mathbf{U}_d$ with $d \in \mathbb{R}_{\geq 0}$ and $\mathbf{U}_\sigma$ with $\sigma \in \Sigma$.

▶ First we extend $\mathbf{p}_M$, the projection from $\mathbb{R}_{\geq 0}$ to $\mathbb{C}_1$ defined in chapter 10, on timed markings in the following way. For all $\mathbf{m} \in \mathbb{M}_Q$, for all $q' \in Q$ we define

$$\mathbf{p}_M(\mathbf{m}) : q \mapsto \mathbf{p}_M(\mathbf{m}(q)(0))$$

▶ Then we extend $\mathbf{U}_\sigma$ over $\mathbb{M}_Q$. We define then for all $\sigma \in \Sigma$, the function $\mathbf{U}^{\mathbf{T}}_\sigma$ such that for all $\mathbf{m} \in \mathbb{M}_Q$, for all $q' \in Q$:

$$\mathbf{U}^{\mathbf{T}}_\sigma(\mathbf{m})(q') = (\{u(v) \in \mathbb{R}_{\geq 0} \mid \exists q \in Q, \exists v \in \mathbf{m}(q)(0), (\mathbf{p}_M(v), u) \in g_\kappa(q, \sigma, q')\}; \widehat{0})$$

which corresponds to the set of valuations reached from m after an action transition labeled by σ.

Lemma 11.1.1. *For all timed marking* $\mathbf{m}$, $\mathbf{p}_M(\mathbf{U}^{\mathbf{T}}_\sigma(\mathbf{m})) = \mathbf{U}_\sigma(\mathbf{p}_M(\mathbf{m}))$.

Proof.

Let $\mathbf{m}$ be a timed marking and $q' \in Q$. Notice that, because $\mathbb{R}_{\geq 0}$ is stable with regards to one-clock timed automata updates $\mathbf{id}$ and $\mathbf{0}$,

$$\mathbf{U}^{\mathbf{T}}_\sigma(\mathbf{m})(q')(0) = \{u(v) \in \mathbb{R}_{\geq 0} \mid \exists q \in Q, \exists v \in \mathbf{m}(q)(0), (\mathbf{p}_M(v), u) \in g_\kappa(q, \sigma, q')\}$$

Hence,

$$\begin{aligned}\mathbf{p}_M(\mathbf{U}^{\mathbf{T}}_\sigma(\mathbf{m}))(q') &= \mathbf{p}_M(\{u(v) \in \mathbb{R}_{\geq 0} \mid \exists q \in Q, \exists v \in \mathbf{m}(q)(0), (\mathbf{p}_M(v), u) \in g_\kappa(q, \sigma, q')\}) \\ &= \{\mathbf{p}_M(u(v)) \in \mathbb{R}_{\geq 0} \mid \exists q \in Q, \exists v \in \mathbf{m}(q)(0), (\mathbf{p}_M(v), u) \in g_\kappa(q, \sigma, q')\}\end{aligned}$$

Notice then that for all $v \in \mathbb{R}_{\geq 0}$, $\mathbf{p}_M(\mathbf{id}(v)) = \mathbf{id}(\mathbf{p}_M(v)) = \mathbf{p}_M(v)$ and $\mathbf{p}_M(\mathbf{0}(v)) = \mathbf{0}(\mathbf{p}_M(v)) = 0$. Thus we can write,

$$\begin{aligned}\mathbf{p}_M(\mathbf{U}^{\mathbf{T}}_\sigma(\mathbf{m}))(q') &= \{u(\mathbf{p}_M(v)) \in \mathbb{R}_{\geq 0} \mid \exists q \in Q, \exists v \in \mathbf{m}(q)(0), (\mathbf{p}_M(v), u) \in g_\kappa(q, \sigma, q')\} \\ &= \{u(v) \in \mathbb{C}_1 \mid \exists q \in Q, \exists v \in \mathbf{p}_M(\mathbf{m})(q), (v, u) \in g_\kappa(q, \sigma, q')\} \\ &= \mathbf{U}_\sigma(\mathbf{p}_M(\mathbf{m}))(q')\end{aligned}$$

■

▶ Finally we define the delay transitions. The delay transition definition needs more work and some new object definitions. The idea is the following. We ideally want to have $\mathbf{m} \overset{d}{\hookrightarrow} \mathbf{m}+d$. However, this would not be taking in account for the possible silent transition like it is done with $\mathbf{U}_d$. Indeed consider the automaton of example 10.3.1, the initial marking would map q_0 to $\{0\}$ and q_1 to $\emptyset$ and we can't simulate a delay 3 by just adding 3 to the marking (c.f. example 10.3.1). As it is done in example 10.3.1 we first need to *close* the initial marking by silent transitions,

so it maps q_0 to $(\bigcup_{n\geq 0} -2n; \uparrow 0)$ and q_1 to $(\bigcup_{n\geq 0} -2n; \uparrow 1)$. Then a delay can be simulated by a simple addition on the marking.

We introduce then a *closure operator*, $\boldsymbol{\epsilon}$, in such a way that we will be able to write $\mathbf{m} \overset{d}{\hookrightarrow} \boldsymbol{\epsilon}(\mathbf{m}) + d$. With this, we could reduce the computation of delays to, first, a computation of $\boldsymbol{\epsilon}(\mathbf{m})$ and then only a simple addition by a real number. We will discuss that later.

To compute the closure of a timed marking through silent transitions we will need to work on the *silent component of* A, i.e. the component of A restricted to its silent transitions. We introduce then an M-bounded one-clock timed automaton, A_ϵ, representing this silent component of A.

Recall that T_ϵ represents $T \setminus \mathbf{dom}(\kappa)$, the set of silent transitions of A (c.f. definition 4.2.1). Formally we define

$$A_\epsilon = \langle Q, I, T_\epsilon, F \rangle$$

We know that $\langle A, \kappa \rangle$ is compatible with $\mathbb{G}_1$. By definition it implies that for all $q, q' \in Q$, $g_{A_\epsilon}(q, q') = g_\kappa(q, \epsilon, q')$ is decomposable on $\mathbb{G}_1$. So A_ϵ is compatible with $\mathbb{G}_1$.

We can then write for all $q, q' \in Q$, $g_{A_\epsilon}(q, q') = \bigcup_{i=1}^{n_{\mathbf{id},q,q'}} I_i^{\mathbf{id},q,q'} \times \{\mathbf{id}\} \cup \bigcup_{j=1}^{n_{\mathbf{0},q,q'}} I_j^{\mathbf{0},q,q'} \times \{\mathbf{0}\}$ with for all $i \in [\![1, n_{\mathbf{id},q,q'}]\!]$, $I_i^{\mathbf{id},q,q'}$ are real intervals bounded by two consecutive integers (c.f. example 2.3.1) pairwise disjoints; and similarly for all $j \in [\![1, n_{\mathbf{0},q,q'}]\!]$, $I_i^{\mathbf{0},q,q'}$ are real intervals bounded by two consecutive integers (c.f. example 2.3.1) paiwise disjoints – by definition of $\mathbb{G}_1$.

We define two alphabets

$$\Sigma_\epsilon^{\mathbf{id}} = \{t_i^{\mathbf{id},q,q'}, q, q' \in Q, i \in [\![1, n_{\mathbf{id},q,q'}]\!]\} \text{ and } \Sigma_\epsilon^{\mathbf{0}} = \{t_i^{\mathbf{0},q,q'}, q, q' \in Q, i \in [\![1, n_{\mathbf{0},q,q'}]\!]\}$$

and a timed control $\mathbf{T}^{\kappa_\epsilon}$ such that for all $(q, v, u, q') \in T_\epsilon$ – we know that there exists a unique index $i \in [\![1, n_{u,q,q'}]\!]$ such that $v \in I_i^{u,q,q'}$ and – we define

$$\kappa_\epsilon(q, v, u, q') = t_i^{u,q,q'}$$

Therefore, if we write $\Sigma_\epsilon = \Sigma_\epsilon^{\mathbf{id}} \cup \Sigma_\epsilon^{\mathbf{0}}$, $\langle A_\epsilon, \kappa_\epsilon \rangle \in \Sigma_\epsilon \mathbf{TA}_M^1$ is a $M, 1$-clock Σ_ϵ-controlled timed automaton *without silent transitions* since we labeled each silent transition with a unique name.

Finally notice that for all $u \in \{\mathbf{0}, \mathbf{id}\}$, $q, q' \in Q$ and $i \in [\![1, n_{u,q,q'}]\!]$, since $I_i^{u,q,q'}$ is bounded by two consecutive (or equal) positive integers smaller or equal to M (as defined in 2.3.1) there exists two uniques intervals $\widehat{b}_i^{u,q,q'} \in \widehat{\mathbb{C}}_M$ and $\underset{\smile}{b}_i^{u,q,q'} \in \underset{\smile}{\mathbb{C}}_M$ such that

$$\mathbf{p}_M^{-1}(I_i^{u,q,q'}) = \widehat{b}_i^{u,q,q'} \cap \underset{\smile}{b}_i^{u,q,q'}$$

Unicity is obtained by definition of $\widehat{\mathbb{C}}_M$ and $\underset{\smile}{\mathbb{C}}_M$, since the bounds are within $[\![1, M]\!]$, with only one way to represent $\{\infty\}$ by using $\uparrow M \cap \downarrow +\infty$. For simplicity sake, we will often use the following notation in the rest of this thesis: given $t \in \Sigma_\epsilon$, we write $\widehat{t}$ and $\underset{\smile}{t}$ for the intervals $\widehat{b}_i^{u,q,q'}$ and $\underset{\smile}{b}_i^{u,q,q'}$ such that $t = t_i^{u,q,q'}$.

The general idea to compute the closure is first to see how we could compute the closure of a timed marking by *one* silent transition (adding all possible values taken by the clock after taking this particular silent transition any time in the future); then we extend this closure on a finite sequence of silent transitions by sequentially applying the closure by one silent transition; and finally we compute the countable union of all the closures by sequences of silent transitions. To compute the closure by one silent transition we will proceed by defining first the closure on an atomic timed set, extend it on simple timed sets and then on simple timed markings.

To compute the closure by a silent transition *with a reset* (labeled in Σ_ϵ^0), we need to introduce a particular set operation $\bowtie$, defined for all subsets E, F of $\mathbb{R}$ as:

$$E \bowtie F = (E - F) \cap \mathbb{R}_{\leq 0}$$

We also say that $E \leq F$ if and only if for all $e \in E$ and $f \in F$, $e \leq f$. The following lemma holds:

Lemma 11.1.2.

- $E \bowtie F = -\{d \in \mathbb{R}_{\geq 0} \mid (E+d) \cap F \neq \emptyset\} = -\{d \in \mathbb{R}_{\geq 0} \mid E \cap (F-d) \neq \emptyset\}$
- *if* $E \leq F$ *then* $E \bowtie F = E - F$
- *if* $E > F$ *then* $E \bowtie F = \emptyset$.

Proof.

▶ By a simple rewriting,

$$\begin{aligned} E \bowtie F &= (E - F) \cap \mathbb{R}_{\leq 0} \\ &= \{e - f \in \mathbb{R}_{\leq 0} \mid e \in E, f \in F\} \\ &= -\{f - e \in \mathbb{R}_{\geq 0} \mid f \in F, e \in E\} \\ &= -\{d \in \mathbb{R}_{\geq 0} \mid \exists f \in F, e \in E, d = f - e\} \\ &= -\{d \in \mathbb{R}_{\geq 0} \mid \exists e \in E, d + e \in F\} \\ &= -\{d \in \mathbb{R}_{\geq 0} \mid \exists x \in E + d, x \in F\} \\ &= -\{d \in \mathbb{R}_{\geq 0} \mid \exists x \in E + d \cap F\} \\ &= -\{d \in \mathbb{R}_{\geq 0} \mid E + d \cap F \neq \emptyset\} \end{aligned}$$

▶ if $E \leq F$ then for all $e \in E$, $f \in F$, $e - f \in \mathbb{R}_{\leq 0}$ so $E - F \cap \mathbb{R}_{\leq 0} = E - F$.
▶ if $E > F$ then for all $e \in E$, $f \in F$, $e - f \in \mathbb{R}_{>0}$ so $E - F \cap \mathbb{R}_{\leq 0} = \emptyset$. ■

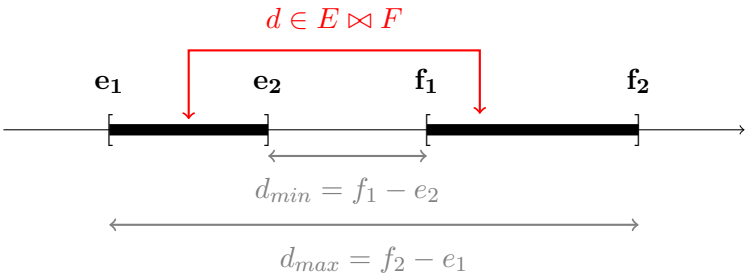


Figure 11.2: Illustration of lemma 11.1.2

So in particular if $E \subseteq \mathbb{R}_{\leq 0}$ and $F \subseteq \mathbb{R}_{\geq 0}$ then $E \bowtie F = E - F$. So in particular for any three subsets, E, F of $\mathbb{R}$ and G of $\mathbb{R}_{\geq 0}$, $(E \bowtie F) \bowtie G = (E \bowtie F) - G$. Notice moreover than if E and F are real intervals then $E \bowtie F$ is a real interval.

We define then the closure of an atomic timed set by a silent transition. Let $(E; \widehat{r})$ be an atomic timed set and $t \in \Sigma_\epsilon$.

$$\boldsymbol{\epsilon}((E;\widehat{r}),t) = \begin{cases} (\emptyset; \uparrow 0) & \text{if } \widehat{r} \cap \underset{\smile}{t} = \emptyset \\ (E \cap \underset{\smile}{t}; \max(\widehat{r}, \widehat{t})) & \text{if } \widehat{r} \cap \underset{\smile}{t} \neq \emptyset \text{ and } t \in \Sigma_\epsilon^{\mathbf{id}} \\ (E \bowtie (\widehat{r} \cap \underset{\smile}{t} \cap \widehat{t}); \uparrow 0) & \text{if } \widehat{r} \cap \underset{\smile}{t} \neq \emptyset \text{ and } t \in \Sigma_\epsilon^{\mathbf{0}} \end{cases}$$

$\boldsymbol{\epsilon}$ is well-defined, i.e. the definition does not depend on the representation. Indeed, suppose an atomic timed set, f, has two equivalent and *different* representations $(E,\widehat{r})$ and $(F,\widehat{s})$, we already noticed in remark 10.1.2 that $E \subseteq \widehat{r}$ and $F \subseteq \widehat{s}$. Hence $E = f(0) = F$.

► Suppose $\widehat{r} \cap \underset{\smile}{t} = \emptyset$ and $\widehat{s} \cap \underset{\smile}{t} \neq \emptyset$ (or conversely), then $F = E > \underset{\smile}{t}$ and $\boldsymbol{\epsilon}((F,\widehat{s}),t) = (\emptyset;\uparrow 0)$ whatever t (using lemma 11.1.2 in the case $t \in \Sigma_\epsilon^{\mathbf{0}}$).

► Suppose $\widehat{r} \cap \underset{\smile}{t} \neq \emptyset$, $\widehat{s} \cap \underset{\smile}{t} \neq \emptyset$ and $t \in \Sigma_\epsilon^{\mathbf{id}}$. If $\widehat{r} > \widehat{t}$ then $E \cap \underset{\smile}{t} \subseteq \widehat{r} = \max(\widehat{r},\widehat{t})$ and $F \cap \widehat{t} = E \cap \underset{\smile}{t} \subseteq \widehat{s} \cap \widehat{t}$. Hence $(E \cap \underset{\smile}{t}; \max(\widehat{r},\widehat{t}))$ and $(F \cap \underset{\smile}{t}; \max(\widehat{s},\widehat{t}))$ are two representations of the same timed set.

Else if $\widehat{r} \leq \widehat{t}$ and $\widehat{s} \leq t$ then $\max(\widehat{r},\widehat{t}) = \widehat{t} = \max(\widehat{s},\widehat{t})$ and $\boldsymbol{\epsilon}((F,\widehat{s}),t) = \boldsymbol{\epsilon}((E,\widehat{r}),t)$.

► Suppose $\widehat{r} \cap \underset{\smile}{t} \neq \emptyset$, $\widehat{s} \cap \underset{\smile}{t} \neq \emptyset$ and $t \in \Sigma_\epsilon^{\mathbf{0}}$. Notice that since $E \subseteq \widehat{r}$, for all $d \in \mathbb{R}_{\geq 0}$ $E+d \subseteq \widehat{r}$ and $\{d \in \mathbb{R}_{\geq 0} \mid E+d \cap \widehat{r} \cap \widehat{t} \cap \underset{\smile}{t} \neq \emptyset\}$ is equal to the set $\{d \in \mathbb{R}_{\geq 0} \mid E+d \cap \widehat{t} \cap \underset{\smile}{t} \neq \emptyset\}$. Similarly for F, so according to lemma 11.1.2 $E \bowtie (\widehat{r} \cap \widehat{t} \cap \underset{\smile}{t}) = F \bowtie (\widehat{r} \cap \widehat{t} \cap \underset{\smile}{t})$ and $\boldsymbol{\epsilon}((F,\widehat{s}),t) = \boldsymbol{\epsilon}((E,\widehat{r}),t)$.

This proves that $\boldsymbol{\epsilon}$ is well-defined. Notice also that $\boldsymbol{\epsilon}$ maps an atomic timed marking to an atomic timed marking.

We prove in the following lemma that it is *sound*, i.e. that it really computes values accessible through the given silent transition.

Lemma 11.1.3. *Let f be an atomic timed set, $q,q' \in Q$, $u \in \{\mathbf{id},\mathbf{0}\}$, $i \in [\![1, n_{u,q,q'}]\!]$ and $t = t_i^{u,q,q'} \in \Sigma_\epsilon$.*

For all $d \in \mathbb{R}_{\geq 0}$ and $x \in \mathbb{R}_{\geq 0}$,

$$x \in \boldsymbol{\epsilon}(f,t)(d) \iff \exists d_0 \in [0,d], \exists y \in f(d_0), (q, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (l, u(\mathbf{p}_M(y))) \text{ and } u(y)+d-d_0 = x$$

Proof.

Let's write $f = (E;\widehat{r})$. Let $x,d \in \mathbb{R}_{\geq 0}$. We make the proof by case disjunction.

► Suppose $E = \emptyset$, then $f = (\emptyset;\uparrow 0)$.
$\boldsymbol{\epsilon}(f,t) = (\emptyset;\uparrow 0)$ in all cases.
The property is trivially true.

► Suppose now $E \neq \emptyset$. We distinguish three cases for e,
  ► If $\widehat{r} \cap \underset{\smile}{t} = \emptyset$.
  $\boldsymbol{\epsilon}(f,t)(d) = (\emptyset;\uparrow 0)(d) = \emptyset$.
  For all $d_0 \in [0,d]$, for all $y \in f(d_0) \subseteq \widehat{r}$, $y \notin \underset{\smile}{t}$
  So $y \notin \widehat{t} \cap \underset{\smile}{t}$ which means that $(q, \mathbf{p}_M(y), u, l) \notin \kappa_\epsilon^{-1}(t)$
  And that $(q, \mathbf{p}_M(y)) \overset{t}{\not\to} (q', u(\mathbf{p}_M(y)))$.
  The property is then trivially true.

  ► $\widehat{r} \cap \underset{\smile}{t} \neq \emptyset$ and $u = \mathbf{id}$.
  $\widehat{r} < \nu_+$ and $\mathbf{op} = \mathbf{id}$. Let $d \in \mathbb{R}_{\geq 0}$.
  $x \in \boldsymbol{\epsilon}(f,e)(d) \iff x \in (E \cap \underset{\smile}{t} + d) \cap \max(\widehat{r},\widehat{t})$
  $\iff x - d \in E \cap \underset{\smile}{t}$ and $x \in \max(\widehat{r},\nu_-)$
  We prove that this last statement is equivalent to
  $\exists d_0 \in [0,d], \exists y \in (E+d_0) \cap \widehat{r} \cap \widehat{t} \cap \underset{\smile}{t}, y + d - d_0 = x$
  ► Suppose $x - d \in E \cap \underset{\smile}{t}$ and $x \in \max(\widehat{r},\widehat{t})$.
  We already know that $\max(\widehat{r},\widehat{t}) \cap \underset{\smile}{t} \neq \emptyset$.

We know also have that $x - d \in \underset{\smile}{t}$ so $\uparrow\!(x-d) \cap \underset{\smile}{t} \neq \emptyset$ and
$\max(\widehat{r}, \uparrow\!(x-d), \widehat{t}) \cap \underset{\smile}{t} \neq \emptyset$.
Moreover $x \in \max(\widehat{r}, \uparrow\!(x-d), \widehat{t})$ which is equivalent as saying that
$\max(\widehat{r}, \uparrow\!(x-d), \widehat{t}) \cap \downarrow\!x \neq \emptyset$.
This implies that $\max(\widehat{r}, \uparrow\!(x-d), \widehat{t}) \cap \downarrow\!x \cap \underset{\smile}{t} \neq \emptyset$.
Let then $y \in \max(\widehat{r}, \uparrow\!(x-d), \widehat{t}) \cap \downarrow\!x \cap \underset{\smile}{t}$.
Let $d_0 = y - x + d$.
$y \in \downarrow\!x \cap \uparrow\!(x-d)$, which means that $x - d \leq y \leq x$ and that $d_0 \in [0, d]$.
$x - d = y - d_0 \in E$ by hypothesis, so $y \in E + d_0$.
We can conclude that $y \in (E + d_0) \cap \widehat{r} \cap \widehat{t} \cap \underset{\smile}{t}$.
This proves one way of the equivalence.

- Suppose now $\exists d_0 \in [0, d], \exists y \in (E + d_0) \cap \widehat{r} \cap \widehat{t} \cap \underset{\smile}{t}, y + d - d_0 = x$.
  Then $x - d = y - d_0 \in (E + d_0) \cap \underset{\smile}{t} - d_0 \subseteq E \cap \underset{\smile}{t}$ and
  $y \in \widehat{r} \cap \widehat{t} = \max(\widehat{r}, \widehat{t})$
  so, since $d - d_0 \geq 0, x = y + d - d_0 \in \max(\widehat{r}, \widehat{t})$
  which proves the other way of the equivalence.

So,

$$\begin{aligned} x \in \boldsymbol{\epsilon}(f, e)(d) &\iff \exists d_0 \in [0, d], \exists y \in (E + d_0) \cap \widehat{r} \cap \widehat{t} \cap \underset{\smile}{t}, y + d - d_0 = x \\ &\iff \exists d_0 \in [0, d], \exists y \in (E + d_0) \cap \widehat{r}, \kappa_\epsilon(q, \mathbf{p}_M(y), \mathbf{id}, q') = t \\ &\qquad \text{and } y + d - d_0 = x \end{aligned}$$

The left-to-right way of the equivalence above is proved by definition of $\widehat{t}$ and $\underset{\smile}{t}$.
Finally,

$$\begin{aligned} x \in \boldsymbol{\epsilon}(f, e)(d) &\iff \exists d_0 \in [0, d], \exists y \in f(d_0), (q, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (l, \mathbf{p}_M(y)) \\ &\qquad \text{and } y + d - d_0 = x \end{aligned}$$

- $\widehat{r} \cap \underset{\smile}{t} \neq \emptyset$ and $u = \mathbf{0}$.

$$\begin{aligned} x \in \boldsymbol{\epsilon}(f, t)(d) &\iff x \in (E \bowtie (\widehat{t} \cap \widehat{r} \cap \underset{\smile}{t}) + d) \cap \mathbb{R}_{\geq 0} \\ &\iff x - d \in E \bowtie (\widehat{r} \cap \widehat{t} \cap \underset{\smile}{t}) \text{ and } x \in \mathbb{R}_{\geq 0} \\ &\iff d - x \in \mathbb{R}_{\geq 0}, x \in \mathbb{R}_{\geq 0} \text{ and } (E + d - x) \cap \widehat{r} \cap \widehat{t} \cap \underset{\smile}{t} \neq \emptyset \end{aligned}$$

According to lemma 11.1.2

$$\begin{aligned} &\iff \exists d_0 \in [0, d], d - d_0 = x, \exists y \in (E + d_0) \cap \widehat{r} \cap \widehat{t} \cap \underset{\smile}{t} \\ &\iff \exists d_0 \in [0, d], \exists y \in (E + d_0) \cap \widehat{r}, (q, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (l, 0) \text{ and } d - d_0 = x \end{aligned}$$

The left-to-right way of the equivalence above is proved by definition of $\widehat{t}$ and $\underset{\smile}{t}$.
Finally,

$$\iff \exists d_0 \in [0, d], \exists y \in f(d_0), (q, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (l, 0) \text{ and } d - d_0 = x$$

■

We extend first the closure operator on *sequences of silent transitions*, in the following way: let f be an atomic timed set and $t_1, t_2 \ldots, t_n \in \Sigma_\epsilon$,

$$\boldsymbol{\epsilon}(f, t_1 \cdot t_2 \cdot \cdots \cdot t_n) = \boldsymbol{\epsilon}(\ldots \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(f, t_1), t_2) \ldots)$$

We extend lemma 11.1.3. We say that a sequence of transitions $w = t_1 \cdot t_2 \cdot \cdots \cdot t_n \in \Sigma_\epsilon^*$, *starts* in a state q if $t_1 = t_i^{u,q,l}$ for some $u \in \{\mathbf{id}, \mathbf{0}\}, l \in Q$ and $i \in [\![1, n_{u,q,l}]\!]$. Analogously we say that it *ends* in a state q' if $t_n = t_i^{u,l,q'}$ for some $u \in \{\mathbf{id}, \mathbf{0}\}, l \in Q$ and $i \in [\![1, n_{u,l,q'}]\!]$. The empty sequence ϵ start and ends in every state. We write for all $q, q' \in Q$, $\Sigma_\epsilon^{q,q'}$ for the set of words in Σ_ϵ^* starting in q and ending in q'.

We also write for all words w' in $(\Sigma_\epsilon \cup \mathbb{R}_{\geq 0})^*$, $\mathbf{u}(w')$ for the projection of w' on Σ_ϵ^* (i.e. all letters $w' \in \mathbb{R}_{\geq 0}$ are mapped to ϵ leaving only the sequence of transitions concatenated in the same order).

Lemma 11.1.4. *Let $f = (E; \widehat{r})$ be an atomic timed set with $E \subseteq \mathbb{R}_{\geq 0}$, $q, q' \in Q$ and $w \in \Sigma_\epsilon^{q,q'}$ starting in q and ending in q'. For all $d \in \mathbb{R}_{\geq 0}$ and $x \in \mathbb{R}_{\geq 0}$,*

$$\begin{aligned} x \in \boldsymbol{\epsilon}(f, w)(d) \iff & \exists d_0 \in [0, d], \exists y \in f(d_0), \exists w' \in \mathbf{u}^{-1}(w), \\ & (q', \mathbf{p}_M(x)) \in \mathbf{Reach}(T^{\kappa_\epsilon}, w', (q, \mathbf{p}_M(y))) \text{ and } \delta_{\Sigma_\epsilon}(w') = d - d_0 \end{aligned}$$

where $\delta_{\Sigma_\epsilon}(w)$ is the duration of w as defined in section 4.2.

Proof.

We prove the proposition by induction on $w \in \Sigma_\epsilon^*$.

- Suppose $w = \epsilon$ and let $d \in \mathbb{R}_{\geq 0}$ and $x \in \mathbb{R}_{\geq 0}$.
  $q = q'$ and $x \in f(d) \iff \exists y \in E, y + d = x$ and $y + d \in \widehat{r}$
  $\iff \exists d_0 \in [0; d], \exists y \in E, y + d_0 \in \widehat{r}$ and $(q, \mathbf{p}_M(y)) \xrightarrow{d-d_0} q, \mathbf{p}_M(x)$
  $\iff \exists y \in f(d_0), (q, \mathbf{p}_M(x)) \in \mathbf{Reach}(\mathbf{T}^{\kappa_\epsilon}, d - d_0, (q, \mathbf{p}_M(y)))$
  We can then easily conclude that the property is true at rank 0.

- Suppose we proved the property for all word w of size n.
  Let $w = w_n \cdot t$ of size $n + 1$ with $t = t_i^{u,l,q'}$,
  starting in q and ending in q'.
  Let $d \in \mathbb{R}_{\geq 0}$ and $x \in \mathbb{R}_{\geq 0}$.
  $x \in \boldsymbol{\epsilon}(f, w_n \cdot t)(d) \iff x \in \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(f, w_n), t)(d)$
  According to lemma 11.1.3,
  $$\begin{aligned} x \in \boldsymbol{\epsilon}(f, w_n \cdot t)(d) \iff & \exists d_0 \in [0, d], \exists y \in \boldsymbol{\epsilon}(f, w_n)(d_0), \\ & (l, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (q', u(\mathbf{p}_M(y))) \text{ and } u(y) + d - d_0 = x \\ \iff & \exists d_0 \in [0, d], \exists y \in \boldsymbol{\epsilon}(f, w_n)(d_0), (l, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (q', u(\mathbf{p}_M(y))) \\ & \text{and } \mathbf{p}_M(u(y)) \oplus_{\langle \mathbb{C}_1, \hookrightarrow \rangle} (d - d_0) = \mathbf{p}_M(x) \\ \iff & \exists d_0 \in [0, d], \exists y \in \boldsymbol{\epsilon}(f, w_n)(d_0), \\ & (l, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (q', u(\mathbf{p}_M(y))) \xrightarrow{d-d_0}_{\mathbf{T}^{\kappa_\epsilon}} (q', \mathbf{p}_M(x)) \end{aligned}$$
  By induction hypothesis,
  $$\begin{aligned} x \in \boldsymbol{\epsilon}(f, w_n \cdot t)(d) \iff & \exists d_0 \in [0, d], \exists y \in \mathbb{R}_{\geq 0}, \exists d_1 \in [0, d_0], \exists z \in f(d_1), \exists w'_n \in \mathbf{u}^{-1}(w_n), \\ & (l, \mathbf{p}_M(y)) \in \mathbf{Reach}(\mathbf{T}^{\kappa_\epsilon}, w'_n, (q, \mathbf{p}_M(z))), \delta_{\Sigma_\epsilon}(w'_n) = d_0 - d_1 \text{ and} \\ & (l, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (q', u(\mathbf{p}_M(y))) \xrightarrow{d-d_0}_{\mathbf{T}^{\kappa_\epsilon}} (q', \mathbf{p}_M(x)) \\ \iff & \exists d_0 \in [0, d], \exists d_1 \in [0, d_0], \exists z \in f(d_1), \exists w'_n \in \mathbf{u}^{-1}(w_n), \\ & (q', \mathbf{p}_M(x)) \in \mathbf{Reach}(\mathbf{T}^{\kappa_\epsilon}, w'_n \cdot t \cdot (d - d_0), (q, \mathbf{p}_M(z))) \\ & \text{and } \delta_{\Sigma_\epsilon}(w'_n) = d_0 - d_1 \\ \iff & \exists d_1 \in [0, d], \exists z \in f(d_1), \exists w' \in \mathbf{u}^{-1}(w_n \cdot t), \\ & (q', \mathbf{p}_M(x)) \in \mathbf{Reach}(\mathbf{T}^{\kappa_\epsilon}, w', (q, \mathbf{p}_M(z))) \text{ and } \delta_{\Sigma_\epsilon}(w') = d - d_1 \end{aligned}$$
  Which concludes the proof of the induction case.

And concludes the proof by induction of the equivalence.

■

An important corollary of this result is exposed below:

Corollary 11.1.1. *Let $f = (E; \widehat{r})$ be an atomic timed set with $E \subseteq \mathbb{R}_{\geq 0}$, $d, d' \in \mathbb{R}_{\geq 0}$ and $w, w' \in \Sigma_\epsilon^*$ such that w starts in q and ends in l, and w' start in l and ends in q':*

$$\boldsymbol{\epsilon}(\boldsymbol{\epsilon}(f, w) + d, w')(d') = \boldsymbol{\epsilon}(f, w \cdot w')(d + d')$$

Proof.

By application of lemma 11.1.4, we can write the following sequence of equivalences. Let $x \in \mathbb{R}_{\geq 0}$,

$$
\begin{aligned}
x \in & \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(f,w)+d,w')(d') \\
& \iff \exists d_0 \in [0,d'], \exists y \in (\boldsymbol{\epsilon}(f,w)+d)(d_0), \exists w_1' \in \mathbf{u}^{-1}(w'), \\
& \quad (q', \mathbf{p}_M(x)) \in \mathbf{Reach}(T^{\kappa_\epsilon}, w_1', (l, \mathbf{p}_M(y))) \text{ and } \delta_{\Sigma_\epsilon}(w_1') = d' - d_0 \\
& \iff \exists d_0 \in [0,d'], \exists y \in \boldsymbol{\epsilon}(f,w)(d+d_0), \exists w_1' \in \mathbf{u}^{-1}(w'), \\
& \quad (q', \mathbf{p}_M(x)) \in \mathbf{Reach}(T^{\kappa_\epsilon}, w_1', (l, \mathbf{p}_M(y))) \text{ and } \delta_{\Sigma_\epsilon}(w_1') = d' - d_0 \\
& \iff \exists d_0 \in [0,d'], \exists y \in \mathbb{R}_{\geq 0}, \exists w_1' \in \mathbf{u}^{-1}(w'), \exists d_1 \in [0, d+d_0], \exists z \in f(d_1), \exists w_1 \in \mathbf{u}^{-1}(w), \\
& \quad (q', \mathbf{p}_M(x)) \in \mathbf{Reach}(T^{\kappa_\epsilon}, w', (l, \mathbf{p}_M(y))) \text{ and } \delta_{\Sigma_\epsilon}(w') = d' - d_0 \text{ and} \\
& \quad (l, \mathbf{p}_M(y)) \in \mathbf{Reach}(T^{\kappa_\epsilon}, w, (q, \mathbf{p}_M(z))) \text{ and } \delta_{\Sigma_\epsilon}(w') = d + d_0 - d_1 \\
& \iff \exists d_1 \in [0, d+d'], \exists w_2 \in \mathbf{u}^{-1}(w \cdot w'), \exists z \in f(d_1), \\
& \quad (q', \mathbf{p}_M(x)) \in \mathbf{Reach}(T^{\kappa_\epsilon}, w', (q, \mathbf{p}_M(z))) \text{ and } \delta_{\Sigma_\epsilon}(w') = d' + d - d_1 \\
& \iff x \in \boldsymbol{\epsilon}(f, w \cdot w')(d'+d)
\end{aligned}
$$

■

Then we can extend it on simple timed sets such that, for all $w \in \Sigma_\epsilon^*$ and $f = f_1 \cup f_2 \cup \cdots \cup f_n$ where $f_1, f_2, \ldots, f_n$ are atomic timed sets, we have:

$$\boldsymbol{\epsilon}(f,w) = \bigcup_{i=1}^{n} \boldsymbol{\epsilon}(f_i, w)$$

We prove below that this extension is well-defined. Indeed we already showed in example 10.1.6 that the decompositions of a simple timed set into a union of atomic timed sets is not unique. We take then two different decompositions and prove that the computation of the closure from either of the two decomposition leads to the same timed set.

Let $w \in \Sigma_\epsilon^*$ and $f = f_1 \cup f_2 \cup \cdots \cup f_n = f_1' \cup f_2' \cup \cdots \cup f_k'$ with $f_1, f_2, \ldots, f_n, f_1', f_2', \ldots, f_n'$ atomic timed sets. Let $d, x \in \mathbb{R}_{\geq 0}$.

$$x \in \boldsymbol{\epsilon}(f_1 \cup f_2 \cup \cdots \cup f_n, w)(d) \iff \exists i \in [\![1,n]\!], x \in \boldsymbol{\epsilon}(f_i, w)(d)$$

Therefore by lemma 11.1.3, exists $i \in [\![1,n]\!]$, $d_0 \in [0,d]$, and $y \in f_i(d_0)$ such that

$$(q, \mathbf{p}_M(y)) \xrightarrow{t}_{\mathbf{T}^{\kappa_\epsilon}} (l, u(\mathbf{p}_M(y))) \text{ and } u(y) + d - d_0 = x$$

But because $f_1 \cup f_2 \cup \cdots \cup f_n = f_1' \cup f_2' \cup \cdots \cup f_k'$, we know that there exists $j \in [\![1,k]\!]$ such that $y \in f_j'(d_0)$. This means, still according to lemma 11.1.3, that $x \in \boldsymbol{\epsilon}(f_j', w)(d)$. In consequence $x \in \boldsymbol{\epsilon}(f_1' \cup f_2' \cup \cdots \cup f_n', w)(d)$.

This ensures that $\boldsymbol{\epsilon}$ is well defined on simple timed sets. Notice that if we would have formalized a normal form for simple timed sets, we could have proved that this definition is correct without going back to the timed automaton. Also $\boldsymbol{\epsilon}$ maps any simple timed set to a simple timed set.

We also extend $\boldsymbol{\epsilon}$ to countable unions of simple timed sets. Let $(f_i)_{i \in \mathbb{N}}$ be such a family, and let $w \in \Sigma_\epsilon^*$, we define

$$\boldsymbol{\epsilon}(\bigcup_{i \in \mathbb{N}} f_i, w) = \bigcup_{i \in \mathbb{N}} \boldsymbol{\epsilon}(f_i, w)$$

$\boldsymbol{\epsilon}$ is well-defined on countable unions of simple timed sets. The proof of this statement can be easily obtain by the same way we proved the well-definition of $\boldsymbol{\epsilon}$ on simple timed sets.

The next step is to extend inductively this closure operator on markings in such a way that for all simple timed markings $\mathbf{m}$, $\boldsymbol{\epsilon}(\mathbf{m}, \epsilon) = \mathbf{m}$ and for all sequence of silent transition $w \in \Sigma_\epsilon^*$ and all silent transition $t = t_i^{u,q,q'} \in \Sigma_\epsilon$ with $u \in \{\mathbf{id}, \mathbf{0}\}$, $q, q' \in Q$ and $i \in [\![1, n_{u,q,q'}]\!]$:

$$\begin{array}{lll} \boldsymbol{\epsilon}(\mathbf{m}, w \cdot t) : & Q \to \mathbf{T}\mathbb{R}_{\geq 0} & \\ & p \mapsto (\emptyset; \uparrow 0) & \text{if } p \neq q' \\ & q' \mapsto \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}, w)(q), t) & \text{otherwise} \end{array}$$

Notice that $\boldsymbol{\epsilon}$ maps any simple timed marking to a simple timed marking. We say that a sequence of silent transitions w is *well-formed* if for all decomposition $w = w_1 \cdot w_1'$ of w, w_1 ends on the state w_1' starts with. For the next proposition recall the definition of atomic markings page 111.

Proposition 11.1.5. *Let $w \in \Sigma_\epsilon^*$,*

- *For all markings $\mathbf{m}_1$ and $\mathbf{m}_2$, $\boldsymbol{\epsilon}(\mathbf{m}_1 \cup \mathbf{m}_2, w) = \boldsymbol{\epsilon}(\mathbf{m}_1, w) \cup \boldsymbol{\epsilon}(\mathbf{m}_2, w)$*
- *For all family of markings $(\mathbf{m}_i)_{i \in \mathbb{N}}$, $\boldsymbol{\epsilon}(\bigcup_{i \in \mathbb{N}} \mathbf{m}_i, w) = \bigcup_{i \in \mathbb{N}} \boldsymbol{\epsilon}(\mathbf{m}_i, w)$.*
- *For all atomic timed set f, if w starts in a state q and ends in a state q' and is well formed,*

$$\boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) = \mathbf{m}_{q', \boldsymbol{\epsilon}(f,w)}$$

and for all state $q_0 \neq q$, $\boldsymbol{\epsilon}(\mathbf{m}_{q_0,f}, w)$ is the empty marking (mapping every state to $(\emptyset; \uparrow 0)$ and written $\emptyset$). Moreover is w is not well-formed, for all $q \in Q$,

$$\boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) = \emptyset$$

Proof.

- By definition of the union between timed markings and $\boldsymbol{\epsilon}$.
- By definition of $\boldsymbol{\epsilon}$ on countable union of simple timed sets.
- By an induction on the size of w, simply using the definition of $\boldsymbol{\epsilon}$.

■

Example 11.1.1. On figure 11.3 we recover the one-clock timed automaton of example 10.3.1 and show how we progressively compute the closure of an initial marking, $\mathbf{m}_I$, which maps q_0 to $([0,0]; \uparrow 0)$ and q_1 to $(\emptyset; \uparrow 0)$. We rename the silent transition t_1 and t_2 as it is explained before defining $\boldsymbol{\epsilon}$ and define $w_0 = \epsilon$, $w_1 = t_1$, $w_2 = t_1 \cdot t_2$, $w_3 = t_1 \cdot t_2 \cdot t_1$. Each red arrow tells us when the value is added (after which additional transition in the word). Indeed $\boldsymbol{\epsilon}(\mathbf{m}_I, t_1)$ is an operation which adds $([0,0]; \uparrow 1)$ in q_1, $\boldsymbol{\epsilon}(\mathbf{m}_I, t_1 \cdot t_2)$ adds $([0,0] \bowtie [2,2]; \uparrow 0) = ([-2,-2]; \uparrow 0)$ to q_0 and finally, $\boldsymbol{\epsilon}(\mathbf{m}_I, t_1 \cdot t_2 \cdot t_3)$ adds $([-2,-2]; \uparrow 1)$ to q_1. And we could go on like that, adding each alternation of t_1 and t_2. All other words add nothing more, because two identical transitions in a raw adds nothing to the marking by definition. In the ends to obtain the closure we had in example 10.3.1. We must take the countable union of all closure by a word with alternating transitions.

We say that a marking $\mathbf{m}$ is *basic* if for all $q \in Q$, $\mathbf{m}(q)$ is an atomic set which can be represented as $(E_q, \uparrow 0)$ with $E_q \subseteq \mathbb{R}_{\geq 0}$. A basic marking is therefore a simple timed marking. We write $\mathbb{M}_Q^0$ for the set of basic markings.

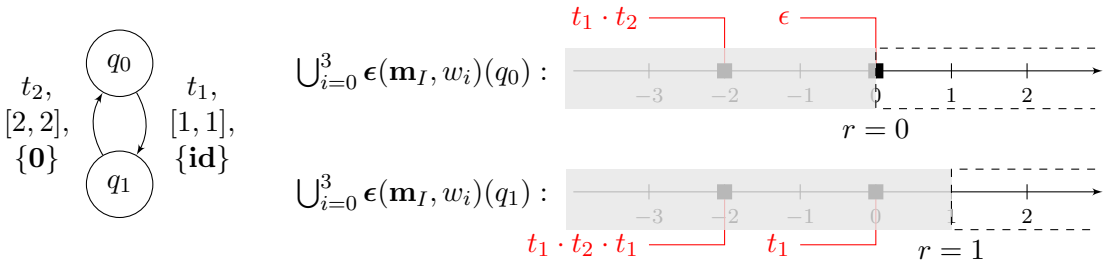


Figure 11.3: Closure by a sequences of transition, $w_0 = \epsilon$, $w_1 = t_1$, $w_2 = t_1 \cdot t_2$, $w_3 = t_1 \cdot t_2 \cdot t_1$

We finally extend the closure operator on languages, such that for all $\mathcal{L} \subseteq \Sigma_\epsilon^*$ and all simple timed markings $\mathbf{m}$,

$$\boldsymbol{\epsilon}(\mathbf{m}, \mathcal{L}) = \bigcup_{w \in \mathcal{L}} \boldsymbol{\epsilon}(\mathbf{m}, w)$$

and then we can define *the ϵ-closure of a timed marking* as

$$\boldsymbol{\epsilon}(\mathbf{m}) = \boldsymbol{\epsilon}(\mathbf{m}, \Sigma_\epsilon^*)$$

Remark 11.1.1. Notice that, as a corollary of proposition 11.1.5, we have that for all atomic marking $\mathbf{m}_{q,f}$, and for all $w, w' \in \Sigma_\epsilon^*$ such that w starts in q, ends in l and w' starts in $l' \neq l$ and ends in q',

$$\boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w \cdot w') = \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w), w') = \boldsymbol{\epsilon}(\mathbf{m}_{l,\boldsymbol{\epsilon}(f,w)}, w') = \emptyset$$

According to proposition 10.3.2, this also holds for any timed marking. And the same reasoning would prove that for all $d, d' \in \mathbb{R}_{\geq 0}$,

$$\boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) + d, w') + d' = \boldsymbol{\epsilon}(\mathbf{m}_{l,\boldsymbol{\epsilon}(f,w)+d}, w') + d' = \emptyset$$

Hence this proposition:

Proposition 11.1.6. *Let $\mathbf{m}$ be a simple timed marking, and let $d, d' \in \mathbb{R}_{\geq 0}$,*

$$\boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}) + d) + d' = \boldsymbol{\epsilon}(\mathbf{m}) + d + d'$$

Proof.

> Notice that $\Sigma_\epsilon = \bigcup_{q,q' \in Q} \Sigma_\epsilon^{q,q'}$.

Let $q \in Q$,

$$\begin{aligned}(\boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}) + d) + d')(q) &= \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}) + d)(q) + d' \\ &= \bigcup_{w \in \Sigma_\epsilon^*} \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}) + d, w)(q) + d' \\ &= \bigcup_{w \in \Sigma_\epsilon^*} \boldsymbol{\epsilon}(\bigcup_{w' \in \Sigma_\epsilon^*} \boldsymbol{\epsilon}(\mathbf{m}, w') + d, w)) + d' \\ &= \bigcup_{w \in \Sigma_\epsilon^*} \bigcup_{w' \in \Sigma_\epsilon^*} \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}, w') + d, w)) + d' \\ &= \bigcup_{w, w' \in \Sigma_\epsilon^*} \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}, w') + d, w)) + d'\end{aligned}$$

According to the remark 11.1.1 we can eliminate all combinations of words such that the second does not start in the same state the first ended. By definition of $\boldsymbol{\epsilon}$ we can also eliminate the second words which does not end in q.

$$(\boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}) + d) + d')(q) = \bigcup_{q_0, l} \bigcup_{w \in \Sigma_\epsilon^{q_0, l}} \bigcup_{w' \in \Sigma_\epsilon^{l, q}} \boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}, w') + d, w)) + d'$$

We can easily extend corollary 11.1.1 on atomic marking, since for all states l, l', atomic timed set f and well-formed word $w_1 \in \Sigma_\epsilon^{l,l'}$, $\boldsymbol{\epsilon}(\mathbf{m}_{l,f}, w) + d = \mathbf{m}_{l', \boldsymbol{\epsilon}(f,w)+d}$. Thus we can extend corollary 11.1.1 to simple timed markings with proposition 10.3.2, and again using remark 11.1.1,

$$\begin{aligned}(\boldsymbol{\epsilon}(\boldsymbol{\epsilon}(\mathbf{m}) + d) + d')(q) &= \bigcup_{q_0, l} \bigcup_{w \in \Sigma_\epsilon^{q_0, l}} \bigcup_{w' \in \Sigma_\epsilon^{l, q}} \boldsymbol{\epsilon}(\mathbf{m}, w' \cdot w) + d + d' \\ &= \bigcup_{w, w' \in \Sigma_\epsilon^*} \boldsymbol{\epsilon}(\mathbf{m}, w' \cdot w) + d + d' \\ &= \bigcup_{w \in \Sigma_\epsilon^*} \boldsymbol{\epsilon}(\mathbf{m}, w) + d + d' \\ &= \boldsymbol{\epsilon}(\mathbf{m}) + d + d'\end{aligned}$$

■

The fundamental theorem about ϵ-closure of timed markings is the following one:

Theorem 11.1.7. *For all basic marking* $\mathbf{m}$*, for all* $d \in \mathbb{R}_{\geq 0}$*,*

$$\mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}) + d) = \mathbf{U}_d(\mathbf{p}_M(\mathbf{m}))$$

Proof.

Since $\mathbf{m}$ is basic, there exists for all state $q \in Q$, a set $E_q \subseteq \mathbb{R}_{\geq 0}$ such that

$$\mathbf{m} = \bigcup_{i=1}^{n} \mathbf{m}_{q,(E_q; \uparrow 0)}$$

Let $x \in \mathbb{C}_M$, and $q' \in Q$. We can then write by definition and distributivity of $\boldsymbol{\epsilon}$,

$$x \in \mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}) + d) \iff \exists q \in Q, \exists w \in \Sigma_\epsilon^*, x \in \mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}_{q,(E_q,\uparrow 0)}, w) + d)(q')$$

According to proposition 11.1.5, and because

$$\mathbf{p}_M(\mathbf{m}_{q',\boldsymbol{\epsilon}((E_q,\uparrow 0),w)} + d)(q') = \mathbf{p}_M([\boldsymbol{\epsilon}((E_q, \uparrow 0), w)) + d](0)) = \mathbf{p}_M(\boldsymbol{\epsilon}((E_q, \uparrow 0), w))(d))$$

we can write

$$x \in \mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}) + d)(q') \iff \exists q \in Q, \exists w \in \Sigma_\epsilon^{q,q'}, \exists x' \in \mathbf{p}_M(\boldsymbol{\epsilon}(E_q, \uparrow 0), w)(d)), \mathbf{p}_M(x') = x$$

By application of lemma 11.1.4 we have,

$$\begin{gathered} x \in \mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}) + d)(q') \\ \Updownarrow \\ \exists q \in Q, \exists w \in \Sigma_\epsilon^{q,q'}, \exists x' \in \mathbb{R}_{\geq 0}, \exists d_0 \in [0, d], \exists y \in E_q + d_0, \exists w' \in \mathbf{u}^{-1}(w), \\ (q', \mathbf{p}_M(x)') \in \mathbf{Reach}(T^{\kappa_\epsilon}, w', (q, \mathbf{p}_M(y))), \mathbf{p}_M(x') = x \text{ and } \delta_{\Sigma_\epsilon} = d - d_0 \end{gathered}$$

By definition of a timed control,

$$\begin{aligned} x \in \mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}) + d)(q') &\iff \exists q \in Q, \exists d_0 \in [0, d], \exists y \in E_q + d_0, (q, \mathbf{p}_M(y)) \xrightarrow{d-d_0}_{\mathbf{T}^{\kappa_\epsilon}} (q', x) \\ &\iff \exists q \in Q, \exists y \in E_q, \\ &\qquad (q, \mathbf{p}_M(y)) \xrightarrow{d_0}_{\mathbf{T}^{\kappa_\epsilon}} (q, \mathbf{p}_M(y + d_0)) \xrightarrow{d-d_0}_{\mathbf{T}^{\kappa_\epsilon}} (q', x) \\ &\iff \exists q \in Q, \exists y \in \mathbf{p}_M(E_q), (q, y) \xrightarrow{d}_{\mathbf{T}^{\kappa_\epsilon}} (q', x) \end{aligned}$$

Because $\mathbf{m}$ is a basic marking we know that $\mathbf{p}_M(\mathbf{m})(q) = \mathbf{m}(q)(0) = E_q \cap \uparrow 0 = E_q$ so we can write by definition of $\mathbf{U}_d$:

$$\begin{aligned} x \in \mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}) + d)(q') &\iff \exists q \in Q, \exists y \in \mathbf{p}_M(\mathbf{m})(q), (q, y) \xrightarrow{d}_{\mathbf{T}^{\kappa_\epsilon}} (q', x) \\ &\iff x \in \mathbf{U}_d(\mathbf{p}_M(\mathbf{m}))(q') \end{aligned}$$

This stands for any $x \in \mathbb{C}_M$ and any $q' \in Q$, so it holds that

$$\mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}) + d) = \mathbf{U}_d(\mathbf{p}_M(\mathbf{m}))$$

■

► We have enough results to define an adequate timed structure we will call the *timed marking structure* and construct a pre-functional bisimulation from this structure to $\langle A, \kappa \rangle$.

The timed marking structure is then the timed structure $\mathcal{M}_Q = \langle \mathbb{M}_Q, \hookrightarrow_{\mathbb{M}}, U_{\mathbb{M}} \rangle$, where for all $d \in \mathbb{R}_{\geq 0}$, $\overset{d}{\hookrightarrow}_{\mathbb{M}} \{(\mathbf{m}, \boldsymbol{\epsilon}(\mathbf{m}) + d), \mathbf{m} \in \mathbb{M}_Q\}$ and $U_{\mathbb{M}} = \{U_\sigma^{\mathbf{T}}, \sigma \in \Sigma\}$. $\langle \mathbb{M} \rangle \hookrightarrow_{\mathbb{M}}$ is well-defined according to proposition 11.1.6. Notice that $\mathcal{M}_Q$ is a *deterministic* timed structure.

We extend first $\mathbf{supp}$ and $\mathbf{supp}_\epsilon$ on timed marking such that for all $\mathbf{m} \in \mathbb{M}_Q$,

$$\begin{aligned} \mathbf{supp}(\mathbf{m}) &= \{q \in Q \mid \mathbf{m}(q)(0) \neq \emptyset\} \\ \mathbf{supp}_\epsilon(\mathbf{m}) &= \{q \in Q \mid \mathbf{m}(q)(0) = \emptyset \text{ and } \exists d \in \mathbb{R}_{\geq 0}, \boldsymbol{\epsilon}(\mathbf{m})(q)(d) \neq \emptyset\} \end{aligned}$$

Remark 11.1.2. It is easy to compute $\mathbf{supp}(\mathbf{m})$ and $\mathbf{supp}_\epsilon(\mathbf{m})$. For the first one we check for all q if $\mathbf{m}(q) = \emptyset$, if not we check if $\mathbf{m}(0)$ is empty by computing a sequence of intersection (for each atomic timed set of a representation of $\mathbf{m}$). For $\mathbf{supp}_\epsilon$ we just check if $\mathbf{m}(q) = \emptyset$, if not we check if it is in $\mathbf{supp}(\mathbf{m})$.

Moreover, we can compute for each $q \in \mathbf{supp}_\epsilon(\mathbf{m})$ the exact moment when q will be reached. In order to do so, we take a representation of $\mathbf{m}(q) = (E_1, \widehat{r}'_1 \cup \cdots \cup (E_n; \widehat{r}_n)$ and compute $\min_{1\leq i\leq n}(r_i - \sup E_i)$. We can also easily compute a boolean to know if the time to wait is strict or not.

We implemented this last feature in **DOTA** so that the diagnoser not only points out that a possible faulty state is reachable, but tells us also when exactly it will be reached.

We consider:

- $\zeta = \mathbf{id}_{\mathcal{P}(Q)^2}$,
- for all $\rho \in \mathcal{P}(Q)^2$, $\mathbf{dom}(\phi_\rho) = \mathbb{M}_Q^0 \cup \{\boldsymbol{\epsilon}(\mathbf{m}) + d, \mathbf{m} \in \mathbb{M}_Q^0, d \in \mathbb{R}_{\geq 0}\}$, which is closed under $\boldsymbol{\epsilon}$ according to proposition 11.1.6. On this domain $\phi_\rho = \mathbf{p}_M$
- for all $\rho \in \mathcal{P}(Q)^2, \xi_1, \xi'_1 \in \mathcal{P}(Q)$, $\psi_{\rho,[\xi_1,\xi'_1]} : \mathbb{M}_Q \times U_{\mathbb{M}} \to \mathbf{M}_Q U$ maps every couple $(\mathbf{m}, \mathbf{U}^{\mathbf{T}}_\sigma)$ with $\mathbf{m} \in \mathbf{dom}(\phi_\rho)$, $\mathbf{supp}(\mathbf{U}^{\mathbf{T}}_\sigma(\mathbf{m})) = \xi_1$, $\mathbf{supp}_\epsilon(\mathbf{U}^{\mathbf{T}}_\sigma(\mathbf{m})) = \xi'_1$, and $\sigma \in \Sigma$ to $\mathbf{U}_\sigma$.

Lemma 11.1.8. *The triple $\Phi = (\zeta, (\phi_\rho)_{\rho\in\mathcal{P}(Q)^2}, (\psi_{\rho,\rho_1})_{\rho,\rho'\in\mathcal{P}(Q)^2})$ is a Σ-deterministic pre-functional bisimulation from $\mathcal{P}(Q)^2, \mathcal{M}_Q$ to $\langle A, \kappa\rangle$.*

Proof.

We prove that all hypotheses of proposition 3.2.2 are verified so we can apply it.

(A) $\mathbf{id}_{\mathcal{P}(Q)^2}$ is surjective

(B) Let $\rho \in \mathcal{P}(Q)^2$. We prove that ϕ_ρ is a functional bisimulation from $\langle \mathbb{M}_Q, \hookrightarrow_{\mathbb{M}}\rangle$ and $\langle \mathbf{M}_Q V, \hookrightarrow_M\rangle$. Recall that $\mathbf{dom}(\phi_\rho) = \mathbb{M}_Q^0 \cup \{\boldsymbol{\epsilon}(\mathbf{m}) + d, \mathbf{m} \in \mathbb{M}_Q^0, d \in \mathbb{R}_{\geq 0}\}$.

- Let $\mathbf{m} = \boldsymbol{\epsilon}(\mathbf{m}_0) + d_0 \in \mathbf{dom}(\phi_\rho)$, with $\mathbf{m}_0 \in \mathbb{M}_Q^0$, $\mathbf{m}' \in \mathbb{M}_Q$ and $d \in \mathbb{R}_{\geq 0}$.
  Suppose $\mathbf{m} \overset{d}{\hookrightarrow}_{\mathbb{M}} \mathbf{m}'$.
  $\mathbf{m}' = \boldsymbol{\epsilon}(\mathbf{m}) + d = \boldsymbol{\epsilon}(\mathbf{m}_0) + d + d_0 \in \mathbf{dom}(\phi_\rho)$ by proposition 11.1.6.
  According to proposition 11.1.7,
  $\mathbf{U}_d(\mathbf{p}_M(\mathbf{m})) = \mathbf{U}_d(\mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}_0) + d_0)) = \mathbf{U}_{d+d_0}(\mathbf{p}_M(\mathbf{m}_0)))$
  $= \mathbf{p}_M(\boldsymbol{\epsilon}(\mathbf{m}_0) + d + d_0)) = \mathbf{p}_M(\mathbf{m}')$
  So $\mathbf{p}_M(\mathbf{m}) \overset{d}{\hookrightarrow}_M \mathbf{p}_M(\mathbf{m}')$ This means that $\mathbf{p}_M$ is a functional simulation.

- Let $\mathbf{m} = \boldsymbol{\epsilon}(\mathbf{m}_0) + d_0 \in \mathbf{dom}(\phi_\rho)$, with $\mathbf{m}_0 \in \mathbb{M}_Q^0$ and $\boldsymbol{\nu} \in$
  Suppose $\mathbf{p}_M(\mathbf{m}) \overset{d}{\hookrightarrow}_M \boldsymbol{\nu}$.
  $\boldsymbol{\nu} = \mathbf{U}_d(\mathbf{p}_M(\mathbf{m})) = \mathbf{U}_{d+d_0}(\mathbf{p}_M(\mathbf{m}_0))$ by definition.
  According to proposition 11.1.7,
  $\mathbf{m} \overset{d}{\hookrightarrow}_{\mathbb{M}} \boldsymbol{\epsilon}(\mathbf{m}_0) + d + d_0 = \boldsymbol{\epsilon}(\mathbf{m}) + d \in \mathbf{p}_M^{-1}(\mathbf{U}_d(\mathbf{p}_M(\mathbf{m}))) = \mathbf{p}_M^{-1}(\boldsymbol{\nu})$
  This means that $\mathbf{p}_M$ is a functional bisimulation.

In conclusion for all $\rho \in \mathcal{P}(Q)^2$, ϕ_ρ is a functional bisimulation. Let then $\mathbf{m}_I$ a timed marking defined for all $q \in Q$ as $\mathbf{m}_I(q) = (\boldsymbol{\nu}_I(q); \uparrow 0)$ with $\boldsymbol{\nu}_I$ the initial marking defined in chapter 5 section 5.1. We have $\mathbf{m}_I \in \mathbb{M}_Q^0$ and $\mathbf{p}_M(\mathbf{m}_I) = \boldsymbol{\nu}_I$ which allows us to conclude this item.

(C) Let $\rho \in \mathcal{P}(Q)^2$, $\xi_1, \xi'_1 \in \mathcal{P}(Q)$, $\mathbf{m} \in \mathbb{M}_Q$ and $\sigma \in \Sigma$.

(C.1) $\mathbf{U}_\sigma^{\mathbf{T}}(\mathbf{m}) \in \mathbb{M}_Q^0$ by definition. Moreover, for all $q' \in Q$,

$$\begin{aligned}\mathbf{p}_M(\mathbf{U}_\sigma^{\mathbf{T}}(\mathbf{m}))(q') &= \{\mathbf{p}_M(u(v)) \mid \exists q \in Q, \exists v \in \mathbf{m}(q)(0), (\mathbf{p}_M(v), u) \in g_\kappa(q, \sigma, q')\} \\ &= \{u(\mathbf{p}_M(v)) \mid \exists q \in Q, \exists v \in \mathbf{m}(q)(0), (\mathbf{p}_M(v), u) \in g_\kappa(q, \sigma, q')\} \\ &= \{u(v) \mid \exists q \in Q, \exists v \in \mathbf{p}_M(\mathbf{m}(q)(0)), (v, u) \in g_\kappa(q, \sigma, q')\} \\ &= \mathbf{U}_\sigma(\mathbf{p}_M(\mathbf{m}))\end{aligned}$$

Which allows us to conclude this item.

(C.2) We know that $\xi_1 = \mathbf{supp}(\mathbf{U}_\sigma^{\mathbf{T}}(\mathbf{m}))$ and $\xi_1' = \mathbf{supp}_\epsilon(\mathbf{U}_\sigma^{\mathbf{T}}(\mathbf{m}))$ by definition of $\mathbf{dom}(\psi_{\rho,[\xi_1,\xi_1']})$. But we can easily see that

$$\mathbf{supp}(\mathbf{U}_\sigma(\mathbf{p}_M(\mathbf{m}))) = \mathbf{supp}(\mathbf{p}_M(\mathbf{U}_\sigma^{\mathbf{T}}(\mathbf{m}))) = \xi_1$$

and

$$\mathbf{supp}_\epsilon(\mathbf{U}_\sigma(\mathbf{p}_M(\mathbf{m}))) = \mathbf{supp}_\epsilon(\mathbf{p}_M(\mathbf{U}_\sigma^{\mathbf{T}}(\mathbf{m}))) = \xi_1'$$

Therefore $(\mathbf{p}_M(\mathbf{m}), \mathbf{U}_\sigma) \in g_A(\rho, [\xi_1, \xi_1'])$. Which conclude this item.

(D) Let $\rho \in \mathcal{P}(Q)^2$, $\xi_1, \xi_1' \in \mathcal{P}(Q)$, $\mathbf{m} \in \mathbf{dom}(\phi_\rho)$ and $\sigma \in \Sigma$.

Suppose $(\mathbf{p}_M(\mathbf{m}), \mathbf{U}_\sigma) \in g_A(\rho, [\xi_1, \xi_1'])$.

Then $\xi_1 = \mathbf{supp}(\mathbf{U}_\sigma(\mathbf{p}_M(\mathbf{m}))) = \mathbf{supp}(\mathbf{U}^{\mathbf{T}}(\mathbf{m}))$ and $\xi_1 = \mathbf{supp}_\epsilon(\mathbf{U}_\sigma(\mathbf{p}_M(\mathbf{m}))) = \mathbf{supp}_\epsilon(\mathbf{U}^{\mathbf{T}}(\mathbf{m}))$, so $(\mathbf{m}, \mathbf{U}_\sigma^{\mathbf{T}}) \in \mathbf{dom}(\psi_{\rho,[\xi_1,\xi_1']})$ and $\psi_{\rho,[\xi_1,\xi_1']}(\mathbf{m}, \mathbf{U}_\sigma^{\mathbf{T}}) = \mathbf{U}_\sigma$. This concludes this item.

Φ is, therefore, a pre-functional bisimulation.

Notice also that for all $\rho, \rho' \in \mathcal{P}(Q)^2$, $\mathbf{dom}_\epsilon(\psi_{\rho,\rho'}) = \emptyset$ and for all $\sigma \in \Sigma$ and $\mathbf{m} \in \mathbb{M}_Q$, there exists at most one pair, $(\mathbf{m}, \mathbf{U}_\sigma)$, in $\mathbf{dom}_\sigma(\psi_{\rho,\rho'})$ by definition. This implies that Φ is a Σ-deterministic pre-functional bisimulation.

■

The direct conclusion of this lemma is that we can construct an automaton on $\mathcal{M}_Q$ which is $\Phi^{-1}(\mathbf{D}\kappa)$-deterministic and which simulates the powerset automaton of A.

Theorem 11.1.9. *The powerset automaton of $\langle A, \kappa \rangle$ can be simulated by a deterministic controlled automaton on $\mathcal{M}_Q$.*

Proof.

Considering $\langle \Phi^{-1}(\mathbf{D}_\kappa A), \Phi^{-1}(\mathbf{D}\kappa) \rangle$, the proof is a consequence of lemma 11.1.8, proposition 3.2.2 and proposition 4.3.5. ■

We will write $\mathbf{M}_\kappa A = \Phi^{-1}(\mathbf{D}_\kappa A)$ and $\mathbf{M}\kappa = \Phi^{-1}(\mathbf{D}\kappa)$. $\mathbf{M}_\kappa A$ action transitions can be easily computed with the techniques described in chapter 5 section 5.2, since $\mathbf{U}_\sigma^{\mathbf{T}}$ can be computed exactly like $\mathbf{U}_\sigma$. We also discussed in remark 11.1.2 how to compute the current state from the current value. For now this is the only easy part to compute. Indeed to compute delay transitions we need an efficient computation of $\boldsymbol{\epsilon}$. Moreover we are not even sure we can finitely represent the configurations of $\mathbf{M}_\kappa A$ since we only know that they are simple timed sets (which are not by essence representable).

We dedicate the next section to the proof, first that the reachable timed marking of $\mathbf{M}_\kappa A$ are *regular* (and therefore finitely representable), and then that $\boldsymbol{\epsilon}$ can be computed by a finite number of unions between precomputed regular timed markings.

11.2 Finite Representation of the closure

Fix again in all this section $M \in \mathbb{N}$ and Σ a finite alphabet. Fix $\langle A, \kappa \rangle \in \Sigma\mathbf{TA}^1_M$ a M-bounded one-clock Σ-timed automaton. We write $A = \langle Q, I, T, F \rangle$. *We suppose moreover in this section that* $I = \{(q_0, 0)\}$ *for some* $q_0 \in Q$*.*

One implication of proposition 11.1.6 is that if several delays are done sequentially, we do not need to compute again the closure of the marking. Thus, throughout an execution, we need to compute the closure only once after each action transition. This also means that we just have to focus, for computation and representation, on the closure of a *basic marking*.

To prove that all reached values in $\mathbf{M}_\kappa A$ are regular timed markings we need to proceed by induction. The initial case is easy since $\mathbf{m}_I$ is a basic marking by hypothesis on I.

Remark 11.2.1. The hypothesis we fixed in the introduction is stronger than needed. I just has to be regular enough so that for each state, the set of associated initial values form an interval. Although we supposed we were in the case depicted in the introduction in the implementation of **DOTA**.

In case we take an action transition, $\sigma \in \Sigma$, in $\mathbf{M}_\kappa A$, it is easy to see that if $\mathbf{m}$ is a regular timed marking, then $\mathbf{U}^{\mathbf{T}}_\sigma(\mathbf{m})$ is a regular union of intervals. By definition of $\mathbf{U}^{\mathbf{T}}_\sigma$, because for all $q \in Q$, $\mathbf{m}(q)(0)$ is a finite union of intervals.

The technical part is to prove the following theorem:

Theorem 11.2.1. *For all basic marking* $\mathbf{m}_0$*,* $\boldsymbol{\epsilon}(\mathbf{m}_0)$ *is a regular timed marking and can be effectively computed.*

This whole section is dedicated to the proof of this theorem. The proof itself describes a efficient computation technique for $\boldsymbol{\epsilon}$ using precomputed regular unions of interval. We will discuss at the end of the section what exactly can be precomputed.

► Let $\mathbf{m}$ be a *basic marking* such that for all q, $\mathbf{m}(q) = (E_q; \uparrow 0)$ with $E_q \subseteq \mathbb{R}_{\geq 0}$.

We can write $\mathbf{m} = \bigcup_{q \in Q} \mathbf{m}_{q,(E_q;\uparrow 0)}$. Notice then that according to proposition 11.1.5, for all $q \in Q$,

$$\boldsymbol{\epsilon}(\mathbf{m}_0) = \bigcup_{w \in \Sigma_\epsilon^*} \bigcup_{q \in Q} \boldsymbol{\epsilon}(\mathbf{m}_{q,(E_q;\uparrow 0)}, w) = \bigcup_{q \in Q} \bigcup_{w \in \Sigma_\epsilon^*} \boldsymbol{\epsilon}(\mathbf{m}_{q,(E_q;\uparrow 0)}, w) = \bigcup_{q \in Q} \boldsymbol{\epsilon}(\mathbf{m}_{q,(E_q;\uparrow 0)})$$

In consequence, it is sufficient to prove the theorem for all basic atomic marking $\mathbf{m}_{q,(E;\uparrow 0)}$, to get the theorem for all basic markings. Notice that a regular atomic marking $\mathbf{m}_{q,(E;\uparrow 0)}$ is necessarily such that E is a finite union of intervals (according to the definition of a basic marking and of a regular timed marking).

► Recall that we defined in section 11.2, $\langle A_\epsilon, \kappa_\epsilon \rangle$ to be the component of A restricted to its silent transitions, with $A_\epsilon = \langle Q, I, T_\epsilon, F \rangle$.

We define for all $q, q' \in Q$, $\mathcal{L}_\epsilon(q, q')$ to be the projection on Σ_ϵ of $\mathcal{L}(T_A, q \times \mathbb{C}_M, \{q'\} \times \mathbb{C}_M)$ (usually named the *untimed language of* A_ϵ from q to q'). Notice that since delays are not taken in account through the projection, it is similar to consider the projection of $\mathcal{L}(T_A, q \times \mathbb{C}_M, \{q'\} \times \mathbb{C}_M)$ or of $\mathcal{L}(T_A, (q, 0), \{q'\} \times \mathbb{C}_M)$. Formally:

$$\mathcal{L}_\epsilon(q, q') = \mathbf{u}(\mathcal{L}(T_A, (q, 0), \{q'\} \times \mathbb{C}_M)) = \mathbf{u}(\mathcal{L}(T_A, q \times \mathbb{C}_M, \{q'\} \times \mathbb{C}_M))$$

Let $q \in Q$ and $w \in \Sigma_\epsilon^*$. Let $\mathbf{m}_{q,f}$ be a non empty basic atomic marking. We already know from proposition 11.1.5 that if w does not start on q or is not well formed, then $\boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) = \emptyset$. On the contrary if w starts on q and is well formed, let's say w ends on q', then $\boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) = \mathbf{m}_{q',\boldsymbol{\epsilon}(f,w)}$. One consequence of lemma 11.1.4 is that actually $\boldsymbol{\epsilon}(f, w)$ is not empty if and only if $w \in \mathcal{L}_\epsilon(q, q')$. In summary,

$$\boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) \neq \emptyset \iff \exists q' \in Q, w \in \mathcal{L}_\epsilon(q, q')$$

This means that for all basic marking $\mathbf{m}_{q,f}$,

$$\begin{aligned}
\boldsymbol{\epsilon}(\mathbf{m}_{q,f}) &= \bigcup_{\Sigma_\epsilon^*} \boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) \\
&= \bigcup_{q' \in Q} \bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) \\
&= \bigcup_{q' \in Q} \bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \boldsymbol{\epsilon}(\mathbf{m}_{q,f}, w) \\
&= \bigcup_{q' \in Q} \bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \mathbf{m}_{q',\boldsymbol{\epsilon}(f,w)}
\end{aligned}$$

according to proposition 11.1.5.

This means that for all $q' \in Q$,

$$\boldsymbol{\epsilon}(\mathbf{m}_{q,f})(q') = \bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \boldsymbol{\epsilon}(f, w)$$

As we said previously f have to be of the form $(E; \uparrow 0)$ with E a finite union of interval included in $\mathbb{R}_{\geq 0}$. Then we reduced the proof of theorem 11.2.1 to the proof of the following lemma.

Lemma 11.2.2. *For all $E \subseteq \mathbb{R}_{\geq 0}$ a finite union of interval and $q, q' \in Q$,*

$$\bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \boldsymbol{\epsilon}((E; \uparrow 0), w) \text{ is a regular timed set.}$$

► In order to prove lemma 11.2.2, we split the operator $\boldsymbol{\epsilon}$ on words into two operators $\boldsymbol{\epsilon}_\mathtt{d}$ and $\boldsymbol{\epsilon}_\mathtt{s}$ computing respectively the effect of $\boldsymbol{\epsilon}$ on the dynamic part and the static part of an atomic timed set. Those two operators can be independently defined by induction on Σ_ϵ^* in the following way. For all $\widehat{r} \in \widehat{\mathbb{C}}_M$ and $t \in \Sigma_\epsilon$, we write $\mathbf{I}(\widehat{r}, t)$ for the interval $\widehat{r} \cap \widehat{t} \cap \underset{\sim}{t}$. let $w \in \Sigma_\epsilon^*$ and $t \in \Sigma_\epsilon^u$ for $u \in \{\mathbf{id}, \mathbf{0}\}$, with $w \cdot t \in \mathcal{L}_\epsilon$, and $(E; \widehat{r})$ be an atomic timed set,

$\boldsymbol{\epsilon}_\mathtt{d}(E, \widehat{r}, \epsilon) = E$	$\boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, \epsilon) = \widehat{r}$	
$\boldsymbol{\epsilon}_\mathtt{d}(E, \widehat{r}, w \cdot t) = \emptyset$	$\boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, w \cdot t) = \emptyset$	if $\boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, w) \cap \underset{\sim}{t} = \emptyset$
$\boldsymbol{\epsilon}_\mathtt{d}(E, \widehat{r}, w \cdot t) = \boldsymbol{\epsilon}_\mathtt{d}(E, \widehat{r}, w) \cap \underset{\sim}{t}$	$\boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, w \cdot t) = \boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, w) \cap \widehat{t}$	else if $t \in \Sigma_\epsilon^{\mathbf{id}}$
$\boldsymbol{\epsilon}_\mathtt{d}(E, \widehat{r}, w \cdot t) = \boldsymbol{\epsilon}_\mathtt{d}(E, \widehat{r}, w) \bowtie \mathbf{I}(\boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, w), t)$	$\boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, w \cdot t) = \uparrow 0$	else if $t \in \Sigma_\epsilon^{\mathbf{0}}$

Notice that because we suppose $w \cdot t \in \mathcal{L}_\epsilon$, it is impossible that $\boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, w) \cap \underset{\sim}{t} = \emptyset$.

We can easily prove by induction that for all atomic marking $(E; \widehat{r})$, and all word $w \in \mathcal{L}_\epsilon$, $\boldsymbol{\epsilon}((E; \widehat{r}), w) = (\boldsymbol{\epsilon}_\mathtt{d}(E, \widehat{r}, w); \boldsymbol{\epsilon}_\mathtt{s}(\widehat{r}, w))$. We recover this way all good properties of $\boldsymbol{\epsilon}$: for all atomic timed set $(E; \widehat{r})$ and $w \in \mathcal{L}_\epsilon$,

- $w = u \cdot v \implies \boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, w) = \boldsymbol{\epsilon}_{\mathtt{s}}(\boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, u), v)$
- $w = u \cdot v \implies \boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, w) = \boldsymbol{\epsilon}_{\mathtt{d}}(\boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, u), \boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, u), v)$.
- $\boldsymbol{\epsilon}_{\mathtt{s}}(\uparrow\!0, w) \neq \emptyset$
- $\boldsymbol{\epsilon}_{\mathtt{d}}([0,0], \uparrow\!0, w) \neq \emptyset$

Some other properties can be easily obtained by induction on w using the trivial property that for any two sets F and G, $F \bowtie G \subseteq \mathbb{R}_{\leq 0}$ so for all $\widehat{r}' \in \widehat{\mathbb{C}}_M$ and $\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M$, $F \bowtie G \cap \widehat{r}' = \emptyset$ and $F \bowtie G \cap \underset{\sim}{s}' = F \bowtie G$.

- $w = t_1 \cdot \dots \cdot t_n \in \Sigma_\epsilon^{\mathbf{id}*} \implies \boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, w) = \widehat{r} \cap \bigcap_{i=1}^n \widehat{t}$
- $w = t_1 \cdot \dots \cdot t_n \in \Sigma_\epsilon^{\mathbf{id}*} \implies \boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, w) = E \cap \bigcap_{i=1}^n \underset{\sim}{t_i}$
- $w \in \Sigma_\epsilon^* \cdot \Sigma_\epsilon^{\mathbf{0}} \implies \boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, w) = \uparrow\!0$
- $w = \tau \cdot u \in \Sigma_\epsilon^{\mathbf{0}} \cdot \Sigma_\epsilon^{\mathbf{id}*} \implies \boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, w) = \boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, \tau)$

Finally $\boldsymbol{\epsilon}_{\mathtt{d}}$ and $\boldsymbol{\epsilon}_{\mathtt{s}}$ *inherits all distributive properties of* $\boldsymbol{\epsilon}$ on unions and countable unions of atomic timed sets.

Below we prove one crucial property, which needs additional notation definitions. We define a mapping $\mathbf{J}_{\widehat{r}} : \Sigma_\epsilon^* \to \mathbb{N}^{\widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}}$ that counts the number of occurrences of certain timing constraints at resetting transitions along a path: precisely, it is defined inductively as follows (where $\Cup$ represents addition of an element to a multiset):

$$\begin{aligned}
\mathbf{J}_{\widehat{r}}(\epsilon) &= \{0\}^{\widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}} \\
\mathbf{J}_{\widehat{r}}(w \cdot t) &= \mathbf{J}_{\widehat{r}}(w) \Cup \{(\boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, w), t)\} && \text{if } t \in \Sigma_\epsilon^{\mathbf{0}} \\
\mathbf{J}_{\widehat{r}}(w \cdot e) &= \mathbf{J}_{\widehat{r}}(w) && \text{if } e \in \Sigma_\epsilon^{\mathbf{id}}.
\end{aligned}$$

Lemma 11.2.3. *Let $(E; \widehat{r})$ be an atomic timed set with $E \subseteq \mathbb{R}_{\leq 0}$, and $w \in \Sigma_\epsilon^*$. Then*

$$\boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w) = E - \sum_{(\widehat{s}, t) \in \widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}} \mathbf{J}_{\widehat{r}}(w)(\widehat{s}, t) \times \mathbf{I}(\widehat{s}, t) \subseteq \mathbb{R}_{\leq 0}.$$

Proof.

The proof is done by induction on w.

- The result is straightforward for $w = \epsilon$.
- Now, assume the result holds for some $w \in \Sigma_\epsilon^*$, and consider $w' = w \cdot t$.
  - Suppose $t \in \Sigma_\epsilon^{\mathbf{id}}$.
    Then $\boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w \cdot t) = \boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w) \cap \underset{\sim}{t}$ by definition.
    Since $\boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w) \subseteq \mathbb{R}_{\leq 0} \subseteq \underset{\sim}{t}$, we have $\boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w \cdot t) = \boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w)$.
    Since $\mathbf{J}_{\widehat{r}}(w \cdot t) = \mathbf{J}_{\widehat{r}}(w)$ when $t \in \Sigma_\epsilon^{\mathbf{id}}$, our result follows.
  - Suppose $t \in \Sigma_\epsilon^{\mathbf{0}}$.
    $\boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w \cdot t) = \boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w) \bowtie \mathbf{I}(\boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, w)), t)$.
    Since $\boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w) \subseteq \mathbb{R}_{\leq 0}$ and $\widehat{r} \subseteq \mathbb{R}_{\geq 0}$, lemma 11.1.2 entails
    $\boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w \cdot t) = \boldsymbol{\epsilon}_{\mathtt{d}}((E; \widehat{r}), w) - \mathbf{I}(\boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, w)), t)$.
    This precisely corresponds to the effect of adding $\{(\boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{r}, w), t)\}$ into $\mathbf{J}_{\widehat{r}}(w)$.

This concludes the proof by induction of this lemma.

■

Now we split $\boldsymbol{\epsilon}$, let $E \subseteq \mathbb{R}_{\geq 0}$ a finite union of intervals and $q, q' \in Q$, we can write:

$$\bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \boldsymbol{\epsilon}((E; \uparrow 0), w) = \bigcup_{w \in \mathcal{L}_\epsilon(q,q')} (\boldsymbol{\epsilon}_{\mathtt{d}}(E, \uparrow 0, w); \boldsymbol{\epsilon}_{\mathtt{s}}(\uparrow 0, w))$$

For all $w \in \mathcal{L}_\epsilon(q, q')$, the function $\iota_w : \widehat{\mathbb{C}}_M \to \mathcal{P}(\mathbb{R}_{\geq 0})$ mapping $\boldsymbol{\epsilon}_{\mathtt{s}}(\uparrow 0, w)$ to $\boldsymbol{\epsilon}_{\mathtt{d}}(E, \uparrow 0, w)$ and all other $\widehat{r} \in \widehat{\mathbb{C}}_M$ to $\emptyset$, is representations for $(\boldsymbol{\epsilon}_{\mathtt{d}}(E, \uparrow 0); \boldsymbol{\epsilon}_{\mathtt{s}}(\uparrow 0, w))$. Hence according to proposition 10.1.10, $\bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \iota_w$ is a representation of $\bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \boldsymbol{\epsilon}((E; \uparrow 0), w)$.

Let us introduce for all $\widehat{s}, \widehat{s}' \in \widehat{\mathbb{C}}_M$ and for all language $\mathcal{L} \subseteq \mathcal{L}_\epsilon$,

$$\mathcal{L}_{\widehat{s}}^{\widehat{s}'} = \{w \in \mathcal{L}, \boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{s}, w) = \widehat{s}'\}$$

Notice that for all $\widehat{r} \in \widehat{\mathbb{C}}_M$, we can then write by definition :

$$(\bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \iota_w)(\widehat{r}) = \bigcup_{w \in \mathcal{L}_\epsilon(q,q')} \iota_w(\widehat{r}) = \bigcup_{w \in [\mathcal{L}_\epsilon(q,q')]_{\uparrow 0}^{\widehat{r}}} \boldsymbol{\epsilon}_{\mathtt{d}}(E, \uparrow 0, w)$$

We write for all set $E \subseteq \mathbb{R}$, for all $\widehat{r} \in \widehat{\mathbb{C}}_M$ and for all $\mathcal{L} \subseteq \Sigma_\epsilon^*$, $\boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, \mathcal{L}) = \bigcup_{w \in \mathcal{L}} \boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, w)$. The following lemma will entail Lemma 11.2.2:

Lemma 11.2.4. *Let $E \subseteq \mathbb{R}_{\geq 0}$ a finite union of intervals, $\widehat{r}, \widehat{r}' \in \widehat{\mathbb{C}}_M$ and $q, q' \in Q$,*

$$-\boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, [\mathcal{L}_\epsilon(q, q')]_{\widehat{r}}^{\widehat{r}'}) \text{ is a regular union of intervals.}$$

Proof.

$\boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, [\mathcal{L}_\epsilon(q, q')]_{\widehat{r}}^{\widehat{r}'})$, will be denoted η for the sake of readability.
We define $A_\epsilon(q, q') = (Q, \{(q, 0)\}, T_\epsilon, \{q'\})$ to be a version of A_ϵ with one initial configuration $(q, 0)$ and only one final state q'. We have then $\mathcal{L}_\epsilon(q, q') = \mathbf{u}(\mathcal{L}(A_\epsilon(q, q')))$ is the untimed language of a one-clock timed automaton and is hence regular [4].

► We want to construct a timed automaton whose untimed language will be $[\mathcal{L}_\epsilon(q, q')]_{\widehat{r}}^{\widehat{r}'}$. In order to do so, we decorate in $A_\epsilon(q, q')$ all states with elements of $\widehat{\mathbb{C}}_M$ to keep track of value of $\boldsymbol{\epsilon}_{\mathtt{s}}$ along a run.

$$[A_\epsilon(q, q')]_{\widehat{r}}^{\widehat{r}'} = (Q \times \widehat{\mathbb{C}}_M, \{(q, \widehat{r}, 0)\}, T_\epsilon^{\boldsymbol{\epsilon}_{\mathtt{s}}}, \{(q', \widehat{r}')\})$$

where transitions are adapted such that from a state $(l, \widehat{s})$ we reach a state $(l', \widehat{s}')$ if and only if l' is reachable from l through a transition t in A_ϵ and $\widehat{s}' = \boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{s}, t)$. Formally,

$$T_\epsilon^{\boldsymbol{\epsilon}_{\mathtt{s}}} = \{([l, \widehat{s}], v, u, [l', \widehat{s}']) \in Q \times \widehat{\mathbb{C}}_M \times \mathbb{C}_M \times \{\mathbf{id}, \mathbf{0}\} \times Q \times \widehat{\mathbb{C}}_M \mid \exists t \in \Sigma_\epsilon, (l, v, u, l') \in \kappa_\epsilon^{-1}(t) \text{ and } \widehat{s}' = \boldsymbol{\epsilon}_{\mathtt{s}}(\widehat{s}, t)\}$$

We equip $[A_\epsilon(q, q')]_{\widehat{r}}^{\widehat{r}'}$ with a $(\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M)$-timed control $\kappa_\epsilon^{\boldsymbol{\epsilon}_{\mathtt{s}}}$ without silent transitions, defined for all $t_0 = ([l, \widehat{s}], v, u, [l', \widehat{s}']) \in T_\epsilon^{\boldsymbol{\epsilon}_{\mathtt{s}}}(t)$ as

$$\kappa_\epsilon^{\boldsymbol{\epsilon}_{\mathtt{s}}}(t_0) = (\widehat{s}, \kappa_\epsilon(l, v, u, l'), \widehat{s}')$$

Let $\mathbf{p}_\epsilon$ be the projection on Σ_ϵ^* of $(\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M)^*$. Formally $\mathbf{p}_\epsilon$ is the free monoïd extension of the projection $\mathbf{p}_\epsilon : \widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M \to \Sigma_\epsilon$ mapping any $(\widehat{s}, t, \widehat{s}')$ to t.

By construction for all $l \in Q$, $\widehat{s} \in \widehat{\mathbb{C}}_M$ and for all word $w' \in (\mathbb{R}_{\geq 0} \cup (\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M))^*$,

$$w' \in \mathcal{L}(T^{\kappa_\epsilon^{\boldsymbol{\epsilon_s}}}, (q, \widehat{r}, 0), \{(l', \widehat{s})\} \times \mathbb{C}_M) \implies \boldsymbol{\epsilon_s}(\widehat{r}, \mathbf{p}_\epsilon(\mathbf{u}(w'))) = \widehat{s}$$

Moreover for all $l \in Q$, $\widehat{s} \in \widehat{\mathbb{C}}_M$ and for all words $w \in [\mathcal{L}_\epsilon(q, l)]_{\widehat{r}}^{\widehat{s}}$, exists a unique word $w_1 \in (\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M)$ such that

$$\mathbf{p}_\epsilon(w_1) = w \text{ and } w_1 \in \mathbf{u}(\mathcal{L}(T^{\kappa_\epsilon^{\boldsymbol{\epsilon_s}}}, (q, \widehat{r}, 0), \{(l', \widehat{s})\} \times \mathbb{C}_M))$$

Let's write for all $l, l' \in Q$ and $\widehat{s}$, $\widehat{s}' \in \widehat{\mathbb{C}}_M$, $\mathcal{L}_\epsilon^{\boldsymbol{\epsilon_s}}(l, \widehat{s}, l', \widehat{s}') = \mathbf{u}(\mathcal{L}([A_\epsilon(l, l')]_{\widehat{s}}^{\widehat{s}'}))$. We can conclude that for all $\widehat{s} \in \widehat{\mathbb{C}}_M$, $\mathbf{p}_\epsilon$ is a bijection from $\mathcal{L}_\epsilon^{\boldsymbol{\epsilon_s}}(q, \widehat{r}, q', \widehat{s})$ to $[\mathcal{L}_\epsilon(q, q')]_{\widehat{r}}^{\widehat{s}}$.

► We now decompose η, in two parts: the first one computing the closure by all paths which never resets the clock, and the second considering at least one reset. In order to do that let us define for all $\widehat{s} \in \widehat{\mathbb{C}}_M$ and $\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M$,

$$W_{\widehat{s}, \underset{\sim}{s}'}^{\mathbf{id}} = \{w = (\widehat{r}_1, t_1, \widehat{r}'_1) \cdot (\widehat{r}_2, t_2, \widehat{r}'_2) \cdots (\widehat{r}_n, t_n, \widehat{r}'_n) \mid \widehat{r}'_n = \widehat{s} \text{ and } \min_{1 \leq i \leq n} t_i = \underset{\sim}{s}'\}.$$

We decompose $\mathcal{L}_\epsilon^{\boldsymbol{\epsilon_s}}(q, \widehat{r}, q', \widehat{r}')$ as the union of all runs that do not contain resetting transitions :

$$\bigcup_{\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M} \mathcal{L}_\epsilon^{\boldsymbol{\epsilon_s}}(q, \widehat{r}, q', \widehat{r}') \cap W_{\widehat{r}', \underset{\sim}{s}'}^{\mathbf{id}}$$

and all paths containing at least one resetting transition labeled $(\widehat{s}, t, \updownarrow 0)$ (on the the right-hand side of the intersection) :

$$\bigcup_{\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M} \bigcup_{(\widehat{s}, t) \in \widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}} \mathcal{L}_\epsilon^{\boldsymbol{\epsilon_s}}(q, \widehat{r}, q', \widehat{r}') \cap \left[W_{\widehat{s}, \underset{\sim}{s}'}^{\mathbf{id}} \times \{(\widehat{s}, t, \updownarrow 0)\} \times (\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M)^* \right]$$

Using this decomposition, we get

$$\eta = \bigcup_{\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M} \bigcup_{w \in \mathcal{L}_\epsilon^{\boldsymbol{\epsilon_s}}(q, \widehat{r}, q', \widehat{r}') \cap W_{\widehat{r}', \underset{\sim}{s}'}^{\mathbf{id}}} \boldsymbol{\epsilon_d}((E, \widehat{r}), \mathbf{p}_\epsilon(w)) \cup$$
$$\bigcup_{\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M} \bigcup_{(\widehat{s}, t) \in \widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}} \bigcup_{w \in \mathcal{L}_\epsilon^{\boldsymbol{\epsilon_s}}(q, \widehat{r}, q', \widehat{r}') \cap W_{\widehat{r}', \underset{\sim}{s}'}^{\mathbf{id}} \times \{(\widehat{s}, t, \updownarrow 0)\} \times (\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M)^*} \boldsymbol{\epsilon_d}((E, \widehat{r}), \mathbf{p}_\epsilon(w))$$

► In the first part, $\mathbf{p}_\epsilon(w)$ contains only non-resetting transitions, by definition of $W_{\widehat{r}', \underset{\sim}{s}'}^{\mathbf{id}}$, so that for all $\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M$, we have $\boldsymbol{\epsilon_d}(E, \widehat{r}, \mathbf{p}_\epsilon(w)) = E \cap \underset{\sim}{s}'$. This part is then a finite union of regular intervals, which we will write μ :

$$\mu = \bigcup_{\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M} \bigcup_{\mathcal{L}_\epsilon^{\boldsymbol{\epsilon_s}}(q, \widehat{r}, q', \widehat{r}') \cap W_{\widehat{r}', \underset{\sim}{s}'}^{\mathbf{id}} \neq \emptyset} E \cap \underset{\sim}{s}'$$

► As for the second part, decomposing $w \in W^{\mathbf{id}}_{\widehat{r}',\underset{\sim}{s}'} \times \{(\widehat{s}, t, \uparrow 0)\} \times (\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M)^*$ as $u \cdot (\widehat{s}, t, \uparrow 0) \cdot v$, we have

$$\boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, \mathbf{p}_\epsilon(w)) = \boldsymbol{\epsilon}_{\mathtt{d}}[\boldsymbol{\epsilon}_{\mathtt{d}}(\boldsymbol{\epsilon}_{\mathtt{d}}[E, \widehat{r}, \mathbf{p}_\epsilon(u)], \widehat{s}, t), \uparrow 0, \mathbf{p}_\epsilon(v)].$$

Since $\mathbf{p}_\epsilon(u)$ contains non-resetting transitions, by definition of $W^{\mathbf{id}}_{\widehat{r}',\underset{\sim}{s}'}$, we have like before $\boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, \mathbf{p}_\epsilon(u)) = E \cap \underset{\sim}{s}'$, and since t is a resetting transition, we get

$$\boldsymbol{\epsilon}_{\mathtt{d}}(E, \widehat{r}, \mathbf{p}_\epsilon(w)) = \boldsymbol{\epsilon}_{\mathtt{d}}([E \cap \underset{\sim}{s}'] \bowtie \mathbf{I}(\widehat{s}, t), \uparrow 0, \mathbf{p}_\epsilon(v)).$$

Let's write then for all $\widehat{s} \in \widehat{\mathbb{C}}_M$, $\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M$ and $t \in \Sigma^{\mathbf{0}}_\epsilon$,

$$\mathbf{L}(\widehat{s}, \underset{\sim}{s}, t) = (W^{\mathbf{id}}_{\widehat{s},\underset{\sim}{s}'} \times \{(\widehat{s}, t, \uparrow 0)\}) \backslash \mathcal{L}^{\boldsymbol{\epsilon}_{\mathtt{s}}}_\epsilon(q, \widehat{r}, q', \widehat{r}')$$

to be the left-quotient of $\mathcal{L}^{\boldsymbol{\epsilon}_{\mathtt{s}}}_\epsilon(q, \widehat{r}, q', \widehat{r}')$ by $W^{\mathbf{id}}_{\widehat{s},\underset{\sim}{s}'} \times \{(\widehat{s}, t, \uparrow 0)\}$, i.e.

$$\mathbf{L}(\widehat{s}, \underset{\sim}{s}, t) = \{v \in (\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M)^* \mid \exists u \in W^{\mathbf{id}}_{\widehat{s},\underset{\sim}{s}'} \times \{(\widehat{s}, t, \uparrow 0)\}, u \cdot v \in \mathcal{L}^{\boldsymbol{\epsilon}_{\mathtt{s}}}_\epsilon(q, \widehat{r}, q', \widehat{r}')\}$$

Notice that $\mathbf{L}(\widehat{s}, \underset{\sim}{s}, t)$ is a regular language since it is the left quotient of a language by a regular language. The second part of ν can be written

$$\xi = \bigcup_{\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M} \bigcup_{(\widehat{s},t) \in \widehat{\mathbb{C}}_M \times \Sigma^{\mathbf{0}}_\epsilon} \boldsymbol{\epsilon}_{\mathtt{d}}\left((E \cap \underset{\sim}{s}') \bowtie \mathbf{I}(\widehat{s}, t), \uparrow 0, \mathbf{p}_\epsilon(\mathbf{L}(\widehat{s}, \underset{\sim}{s}, t))\right)$$

so that we can write $\eta = \mu \cup \xi$, and it is left to prove that ξ is a regular union of interval.

► Let's write for all $\widehat{s} \in \widehat{\mathbb{C}}_M$, $\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M$ and $t \in \Sigma^{\mathbf{0}}_\epsilon$,

$$\xi(\widehat{s}, \underset{\sim}{s}', t) = \boldsymbol{\epsilon}_{\mathtt{d}}\left((E \cap \underset{\sim}{s}') \bowtie \mathbf{I}(\widehat{s}, t), \uparrow 0, \mathbf{p}_\epsilon(\mathbf{L}(\widehat{s}, \underset{\sim}{s}, t))\right)$$

We prove that for all $\widehat{s} \in \widehat{\mathbb{C}}_M$, $\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M$ and $t \in \Sigma^{\mathbf{0}}_\epsilon$, $-\xi(\widehat{s}, \widehat{s}', t)$ is a regular union of interval, we will then be able to conclude using proposition 10.2.3.
For all $\widehat{s} \in \widehat{\mathbb{C}}_M$, $\underset{\sim}{s}' \in \underset{\sim}{\mathbb{C}}_M$ and $t \in \Sigma^{\mathbf{0}}_\epsilon$, we write $E_{t,\widehat{s},\underset{\sim}{s}'} = (E \cap \underset{\sim}{s}') \bowtie \mathbf{I}(\widehat{s}, t) \subseteq \mathbb{R}_{\leq 0}$. From Lemma 11.2.3, we derive for all $w \in \mathbf{L}(\widehat{s}, \underset{\sim}{s}', t)$,

$$\boldsymbol{\epsilon}_{\mathtt{d}}((E_{t,\widehat{s},\underset{\sim}{s}'}, \uparrow 0), \mathbf{p}_\epsilon(w)) = E_{t,\widehat{s},\underset{\sim}{s}'} - \sum_{(\widehat{s}'',t') \in \widehat{\mathbb{C}}_M \times \Sigma^{\mathbf{0}}_\epsilon} \mathbf{J}_{\uparrow 0}(\mathbf{p}_\epsilon(w))(\widehat{s}'', t') \times \mathbf{I}(\widehat{s}'', t').$$

► We introduce the notation $\mathbf{Par} : (\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \widehat{\mathbb{C}}_M)^* \to ((\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \widehat{\mathbb{C}}_M) \to \mathbb{N})$ for the Parikh image of a word in $(\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \widehat{\mathbb{C}}_M)^*$ (a function which associate to each word a function counting the number of occurence of each letters).
Let $w \in \mathbf{L}(\widehat{s}, \underset{\sim}{s}', t)$. By definition of $\mathbf{J}_{\uparrow 0}$, for all $\widehat{s}''$ and $t' \in \Sigma_\epsilon^{\mathbf{0}}$, $\mathbf{J}_{\uparrow 0}(\mathbf{p}_\epsilon(w))(\widehat{s}'', t')$ correspond to the number of prefixes $\mathbf{p}_\epsilon(w') \cdot t'$ of $\mathbf{p}_\epsilon(w)$ such that $\boldsymbol{\epsilon}_{\mathtt{s}}(\uparrow 0, \mathbf{p}_\epsilon(w')) = \widehat{s}''$. Let $w' \cdot t_0$ be the unique inverse image of $\mathbf{p}_\epsilon(w') \cdot t'$ by $\mathbf{p}_\epsilon()$. By construction of $[A_\epsilon(q, q')]_{\widehat{r}}^{\widehat{r}'}$, $w' \cdot t_0$ is in the untimed language of $[A_\epsilon(q, q')]_{\widehat{r}}^{\widehat{r}'}$, $\boldsymbol{\epsilon}_{\mathtt{s}}(\uparrow 0, \mathbf{p}_\epsilon(w')) = \widehat{s}''$ if and only if $t_0 = (\widehat{s}'', t', \uparrow 0)$. This means that $\mathbf{J}_{\uparrow 0}(\mathbf{p}_\epsilon(w))(\widehat{s}'', t')$ is actually equal to the number of occurrence of $(\widehat{s}'', t', \uparrow 0)$ is w. Formally speaking, for all word $w \in \mathbf{L}(\widehat{s}, \underset{\sim}{s}', t)$,

$$\mathbf{J}_{\uparrow 0}(\mathbf{p}_\epsilon(w))(\widehat{s}'', t') = \mathbf{Par}(w)(\widehat{s}'', t', \uparrow 0)$$

► By Parikh's theorem [46], we know that $\mathbf{Par}(\mathbf{L}(\widehat{s}, \underset{\sim}{s}', t))$ is a *semi-linear set* and can be written

$$\mathbf{Par}(\mathbf{L}(\widehat{s}, \underset{\sim}{s}', t)) = \bigcup_{i=1}^{c} (\mathbf{Par}_0^i + \sum_{j=1}^{d_i} \mathbf{Par}_j^i \times \mathbb{N})$$

where $\mathbf{Par}_j^i \in \mathbb{N}^{\widehat{\mathbb{C}}_M \times \Sigma_\epsilon \times \widehat{\mathbb{C}}_M}$ for all $1 \leq i \leq c$ and $0 \leq j \leq d_i$.
Hence for all word $w \in \mathbf{L}(\widehat{s}, \underset{\sim}{s}', t)$, exists $i \in [\![1, c]\!]$ and $n_1^i, \ldots, n_{d_i}^i \in \mathbb{N}$ such that $\mathbf{Par}(w) = \mathbf{Par}_0^i + \sum_{j=1}^{d_i} n_j^i \mathbf{Par}_j^i$, and we can then write,

$$\boldsymbol{\epsilon}_{\mathtt{d}}((E_{t,\widehat{s},\underset{\sim}{s}'}, \uparrow 0), \mathbf{p}_\epsilon(w)) = E_{t,\widehat{s},\underset{\sim}{s}'} - \sum_{(\widehat{s}'', t') \in \widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}} (\mathbf{Par}_0^i + \sum_{j=1}^{d_i} n_j^i \mathbf{Par}_j^i)(\widehat{s}'', t', \uparrow 0) \times \mathbf{I}(\widehat{s}'', t').$$

and conversely any $1 \leq i \leq c$ and $n_1^i, \ldots, n_{d_i}^i \in \mathbb{N}$ can be related to a word.

► Consequences of the discussions above is that we can write

$$\begin{aligned}
\xi(\widehat{s}, \underset{\sim}{s}', t) &= \boldsymbol{\epsilon}_{\mathtt{d}} \left((E \cap \underset{\sim}{s}') \bowtie \mathbf{I}(\widehat{s}, t), \uparrow 0, \mathbf{p}_\epsilon(\mathbf{L}(\widehat{s}, \underset{\sim}{s}, t)) \right) \\
&= \bigcup_{w \in \mathbf{L}(\widehat{s}, \underset{\sim}{s}, t)} \boldsymbol{\epsilon}_{\mathtt{d}} \left((E \cap \underset{\sim}{s}') \bowtie \mathbf{I}(\widehat{s}, t), \uparrow 0, \mathbf{p}_\epsilon(w) \right) \\
&= \bigcup_{i=1}^{c} \bigcup_{n_1, \ldots, n_{d_i} \in \mathbb{N}} E_{t,\widehat{s},\underset{\sim}{s}'} - \sum_{(\widehat{s}'', t') \in \widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}} (\mathbf{Par}_0^i + \sum_{j=1}^{d_i} n_j^i \mathbf{Par}_j^i)(\widehat{s}'', t') \times \mathbf{I}(\widehat{s}'', t') \\
&= E_{t,\widehat{s},\underset{\sim}{s}'} - \bigcup_{i=1}^{c} \Big(\sum_{(\widehat{s}'', t') \in \widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}} \mathbf{Par}_0^i(\widehat{s}'', t', \uparrow 0) \times \mathbf{I}(\widehat{s}'', t') + \\
&\qquad \bigcup_{n_1, \ldots, n_{d_i} \in \mathbb{N}} \sum_{j=1}^{d_i} \sum_{(\widehat{s}'', t') \in \widehat{\mathbb{C}}_M \times \Sigma_\epsilon^{\mathbf{0}}} n_j \cdot \mathbf{Par}_j^i(\widehat{s}'', t', \uparrow 0) \times \mathbf{I}(\widehat{s}'', t') \Big)
\end{aligned}$$

We write $\Gamma_{t',\widehat{s},\underset{\sim}{s'}}$ for the second term, so that $\xi(\widehat{s},\underset{\sim}{s'},t) = E_{t,\widehat{s},\underset{\sim}{s'}} - \Gamma_{t,\widehat{s},\underset{\sim}{s'}}$. For $1 \leq i \leq c$ and $0 \leq j \leq d_i$, we let

$$K^i_j = \sum_{(\widehat{s''},t') \in \widehat{\mathbb{C}}_M \times \Sigma^{\mathbf{0}}_\epsilon} \mathbf{Par}^i_j(\widehat{s''},t',\updownarrow 0) \times \mathbf{I}(\widehat{s''},t').$$

► First assume that for some i_0 and j_0, it holds $K^{i_0}_{j_0}$ has strictly positive length, λ. Let $L_n = K^{i_0}_0 + n \cdot K^{i_0}_{j_0}$ for any $n \in \mathbb{N}$. Then clearly $L_n \subseteq \Gamma_{t,\widehat{s},\underset{\sim}{s'}}$ for all n. We write λ_n for the length of L_n. We prove that exists $\alpha \in \mathbb{R}_{\geq 0}$ such that $(\alpha, +\infty) \subseteq \bigcup_{n\in\mathbb{N}} L_n \subseteq \Gamma_{t,\widehat{\mathbb{C}}_M s,\underset{\sim}{s}}$.
Notice that for all $n \in \mathbb{N}$,

$$\inf L_{n+1} - \sup L_n = \inf L_n + \inf K^{i_0}_{j_0} - \sup L_n = \inf K^{i_0}_{j_0} - \lambda_n = \inf K^{i_0}_{j_0} - \lambda_n$$

Therefore, since $\lambda_n = \lambda_0 + n\lambda$ and $\lambda > 0$, if $N = \lfloor \frac{\inf K^{i_0}_{j_0} - \lambda_0}{\lambda} \rfloor + 1$, for all $n \geq N$, $L^-_{n+1} < L^+_n$. This means that for all $n \geq N$, L_n and L_{n+1} overlap, thus that

$$\bigcup_{n \geq N} L_n = (\inf L_N, +\infty)$$

Hence, fixing $\alpha = \inf L_N$ we get $(\alpha, +\infty) \subseteq \Gamma_{t,\widehat{s},\underset{\sim}{s'}}$ As consequences we prove below that $\Gamma_{t,\widehat{s},\underset{\sim}{s'}}$ is a finite union of intervals.
Let $1 \leq i \leq c$. We write $\Gamma_{t,\widehat{s},\underset{\sim}{s'}}(i) = K^i_0 + \bigcup_{n_1,\ldots,n_{d_i}\in\mathbb{N}} \sum_{j=1}^{d_i} n_j \times K^i_j$.
Let $1 \leq j \leq d_i$,

► If $K^i_j = [0,0]$ we define $m^i_j = 0$. For all $n \geq m^i_j$, $nK^i_j \cup (\alpha, +\infty)$ is constant, equal to $[0,0] \cup (\alpha, +\infty)$, and is an interval.
► Else if $K^i_j = [0,b]$ or $[0,b)$ with $b > 0$, then let $m^i_j = \lfloor \frac{\alpha}{b} \rfloor + 1$. We then have $\sup m_i \times K^i_j = m_i \times b > \alpha$ and for all $n \geq m^i_j$, $nK^i_j \cup (\alpha, +\infty)$ is constant, equal to $[0, +\infty)$, and is an interval.
► Finally if $\inf K^i_j = a > 0$, then let $m^i_j = \lfloor \frac{\alpha}{a} \rfloor + 1$. We then have $\inf m_i \times K^i_j = m_i \times a > \alpha$ and for all $n \geq m^i_j$, $nK^i_j \cup (\alpha, +\infty)$ is constant, equal to $(\alpha, +\infty)$, and is an interval.

In conclusion, if $\leq_\times$ stands for the product order on $\mathbb{N}^{d_i}$, for all $1 \leq i \leq c$, for all $(n_1,\ldots,n_{d_i}) \geq_\times (m^i_1,\ldots,m^i_{d_i})$, $\sum_{j=1}^{d_i} n_j \times K^i_j$ is constant and is an interval. This means that $\bigcup_{n_1,\ldots,n_{d_i} \geq (m^i_1,\ldots,m^i_{d_i})} n_j \times K^i_j$ is an interval and that,

$$\Gamma_{t,\widehat{s},\underset{\sim}{s'}}(i) = K^i_0 + \Big(\bigcup_{n_1,\ldots,n_{d_i} <_\times (m^i_1,\ldots,m^i_{d_i})} \sum_{j=1}^{d_i} n_j \times K^i_j \Big) \cup \Big(\bigcup_{n_1,\ldots,n_{d_i} \geq_\times (m^i_1,\ldots,m^i_{d_i})} \sum_{j=1}^{d_i} n_j \times K^i_j \Big)$$

is a finite union of intervals, with as direct consequence that $\Gamma_{t,\widehat{s},\underset{\sim}{s'}}$ is a finite union of intervals.

► We now assume that for all i and j, exists a^i_j. Such that $K^i_j = [a^i_j, a^i_j]$. Notice that by definition of $\widehat{\mathbb{C}}_M$, $\underbrace{\mathbb{C}}_M$ and $\mathbf{I}$, $a^i_j \in \mathbb{N}$.
We write again,

$$\Gamma_{t,\widehat{s},\underbrace{s}'}(i) = K^i_0 + \bigcup_{n_1,\dots,n_{d_i}\in\mathbb{N}} \sum_{j=1}^{d_i} n_j \times K^i_j$$

$$= K^i_0 + \bigcup_{n_1,\dots,n_{d_i}\in\mathbb{N}} \sum_{j=1}^{d_i} n_j \times [a^i_j, a^i_j] \quad = \bigcup_{n_1,\dots,n_{d_i}\in\mathbb{N}} K^i_0 + \sum_{j=1}^{d_i} n_j \times [a^i_j, a^i_j]$$

According to proposition 10.2.6, $\Gamma_{t,\widehat{s},\underbrace{s}'}(i)$ is then a regular union of intervals, and according to proposition 10.2.3 $\Gamma_{t,\widehat{s},\underbrace{s}'}$ is a regular union of intervals.

This proves that for all $\widehat{s} \in \widehat{\mathbb{C}}_M$, $\underbrace{s}' \in \underbrace{\mathbb{C}}_M$ and $t \in \Sigma^{\mathbf{0}}_\epsilon$, $\Gamma_{\widehat{s},\underbrace{s}',t'}$ is a regular union of intervals. And according finally to proposition 10.2.5 $\Gamma_{\widehat{s},\underbrace{s}',t'} - E_{t,\widehat{s},\underbrace{s}'} = -\xi(\widehat{s},\underbrace{s}',t')$ is a regular union of intervals. Which allows us to conclude that $-\eta = -\mu \cup -\xi$ is a regular interval.

■

Remark 11.2.2. Lemma 11.2.4 could be easily generalized to any regular language $\mathcal{L} \subseteq \mathcal{L}_\epsilon$ but for clarity sake we considered only the special case we are interested in.

In conclusion since $-\boldsymbol{\epsilon}_{\mathsf{d}}(E, \widehat{r}, [\mathcal{L}_\epsilon(q,q')]^{\widehat{r}'}_{\widehat{r}})$ is a regular union of interval, we can conclude that $\boldsymbol{\epsilon}((E;\uparrow 0), \mathcal{L}_\epsilon(q,q')$ is a regular timed set and $\boldsymbol{\epsilon}(\mathbf{m}_0)$ is a regular timed marking.

Efficiency: Moreover the proof as it is done in this section describes a computation method for $\boldsymbol{\epsilon}(\mathbf{m}_0)$.

- $\mathbf{m}_0$ can be effectively decomposed as a union of atomic markings $\mathbf{m}_{q,(E;\uparrow 0)}$
- For all $q' \in Q$, $\boldsymbol{\epsilon}(\mathbf{m}_{q,(E;\uparrow 0)})(q')$ can be computed by using its representation $\iota_{q,q',E}$.
- For all $\widehat{r}$, $\iota_{q,q',E}(\widehat{r})$ is equal to $\boldsymbol{\epsilon}_{\mathsf{d}}(E, \uparrow 0, [\mathcal{L}_\epsilon(q,q')]^{\widehat{r}}_{\uparrow 0})$.
- Using the notation of lemma 11.2.4, $\boldsymbol{\epsilon}_{\mathsf{d}}(E, \uparrow 0, [\mathcal{L}_\epsilon(q,q')]^{\widehat{r}}_{\uparrow 0})$ can be computed by computing two sets μ and ξ:
  - $\mu = \bigcup_{\underbrace{s}\in\underbrace{\mathbb{C}}_M} \bigcup_{\mathcal{L}^{\boldsymbol{\epsilon}_{\mathsf{s}}}_\epsilon(q,\uparrow 0,q',\widehat{r})\cap W^{\mathbf{id}}_{\widehat{r},\underbrace{s}} \neq \emptyset} E \cap \underbrace{s}$ can be effectively computed because the emptiness of the intersection of two regular languages is decidable.
    Remark 11.2.3. Notice that it is possible to precompute for all $q, q' \in Q$ and all $\widehat{r} \in \widehat{\mathbb{C}}_M$ the list of all $\underbrace{s} \in \underbrace{\mathbb{C}}_M$ such that $\mathcal{L}^{\boldsymbol{\epsilon}_{\mathsf{s}}}_\epsilon(q, \uparrow 0, q', \widehat{r}) \cap W^{\mathbf{id}}_{\widehat{r},\underbrace{s}} \neq \emptyset$
  - ξ can be computed as a (negation of) union of regular interval $\xi(\widehat{s}, \widehat{s}', t)$, for any $\widehat{s} \in \widehat{\mathbb{C}}_M$, $\underbrace{s}' \in \underbrace{\mathbb{C}}_M$ and $t \in \Sigma^{\mathbf{0}}_\epsilon$. Each of those regular union intervals can be computed by using $E \cap \underbrace{s}' \bowtie \mathbf{I}(\widehat{s}, t)$, and subtracting to it a regular union of interval $\Gamma_{t,\widehat{s},\underbrace{s}'}$. This last regular union of intervals is obtained by computing the Parikh image of the regular language $\mathcal{L}^{\boldsymbol{\epsilon}_{\mathsf{s}}}_\epsilon(q, \uparrow 0, q', \widehat{r})$ (which can be done by adapting the method of McNaughton and Yamada to compute the regular language of a finite automaton, to work on the region automaton of the timed automaton $[A_\epsilon(q,q')]^{\widehat{r}}_{\uparrow 0}$ and compute the Parikh image instead). From the Parikh image we either compute a bound after which we do not need to add intervals (the set of all m^i_j) and just compute a finite

union of intervals and add an unbounded interval $(\alpha, +\infty)$; or we use proposition 10.2.6 to compute a regular union of intervals. In each case the method can be effectively implemented.

Remark 11.2.4. Moreover, in each case we can make the computation independently of the value of E and hence precompute $\Gamma_{t,\widehat{s},\underset{\sim}{s'}}$ for all $q, q' \in Q$, $\widehat{r}, \widehat{s} \in \widehat{\mathbb{C}}_M$, $\underset{\sim}{s'} \in \underset{\sim}{\mathbb{C}}_M$ and $t \in \Sigma_\epsilon^{\mathbf{0}}$.

We implemented a prototype **DOTA**using the diagnoser on $\mathcal{M}_Q$ we defined in section 11.2 and this computation method proved in this section and summarized just above. In chapter 12 we discuss our implementation techniques and compare the efficiency of **DOTA**with regards to an implementation of the diagnoser defined by Tripakis in [53].

Chapter 12

DOTA : Diagnozer for One-clock Timed Automata

Comparing Two Diagnosis Methods

We implemented both the diagnoser we constructed in the last chapters and the diagnoser proposed in [53] since we couldn't find an implementation of it. It also had the advantage of making a comparison of their efficiency easier. Our approach has only been developed on one-clock timed automata so we could only compare the one-clock timed automata version of the diagnoser of [53] which could, however, be constructed in the case of timed automata with multiple clocks and state invariants.

We focused on comparing the computation time of action and delay transition performed by each diagnoser, with the idea that, thanks to precalculus, our diagnoser would perform better. We didn't manage yet to confirm this idea, nor with a formal study of the complexity of action computation in our diagnoser, nor with experimentation. However, we propose some ideas and conjecture for the formal complexity of our diagnoser and still present some concrete comparison of efficiency between our implementation of both our diagnoser and the diagnoser of [53]. The only thing we can conclude for sure regarding all tests we have done is that our diagnoser performs better with regard to delay transitions.

The chapter is organized as follow :

Section 12.1 : In this section we put side by side our construction of a diagnoser and the construction proposed in [53]. We point out the similarities and differences in our constructions, discuss the limitations of both constructions and try to theoretically evaluate their performance.

Section 12.2 : We describe in this section our implementations of the diagnosers. We propose both the general scheme of the organization of the program and details on implementation and complexity of the different bricks used to compute and execute the diagnoser.

Section 12.3 : Finally we discuss the performance of both diagnosers and provide some partial results on action transition, delay transition and pre-computational time cost.

The source code of our implementations of both diagnosers is made in Python3 and can be accessed at `http://www.lsv.fr/~jaziri/DOTA.zip`. More instructions can be found in the README file.

12.1 Comparison of the approaches

We outline the main differences between the diagnoser proposed by Tripakis [53] and the diagnoser constructed in chapter 11.

In the approach of [53], the set of possible current configurations is stored as a clock marking. If an action σ occurs after some delay d, the diagnoser computes the set of all possible configurations reached after a delay d (possibly following silent transitions) and applies from the resulting markings the set of all available transitions labeled σ. This amounts to computing the functions $\mathbf{U}_d$ and $\mathbf{U}_\sigma$ at each observation. There is also a timeout, which makes the diagnoser update the marking (with $\mathbf{U}_d$) regularly if no action is observed. The computation of $\mathbf{U}_d$ is heavily used and has to be performed very efficiently so that the diagnoser can be used at runtime. Both $\mathbf{U}_\sigma$ and $\mathbf{U}_d$ are computed by computing the set of all reachable configurations of the timed automaton with the help of DBMs.

In our approach, we use *timed markings* to store sets of possible configurations. Given a timed marking, when an action σ is observed after some delay d, we can easily compute the set of configurations reachable after delay d, and have to apply $\mathbf{U}_\sigma^{\mathbf{T}}$ and recompute the silent closure, ϵ. Following section 11.1, $\mathbf{U}_\sigma^{\mathbf{T}}$ can be computed like $\mathbf{U}_\sigma$, i.e. as a series of set operations on intervals as we described it in section 5.2. The silent closure, ϵ, can be computed as a series of unions of subtractions between an interval and regular unions of intervals (see section 11.2). Those regular unions can be precomputed as we observed in remark 11.2.4; while this may require exponential time and space to compute and store, this makes the simulation of delay transitions very efficient. To be exact, in our prototype **DOTA**the delays are not even computed. They are only stored and considered only before any application on $\mathbf{U}_\sigma^{\mathbf{T}}$ following the discussion we had in remark 10.2.3.

We haven't made a thorough study of the complexity of our operator $\mathbf{U}_\sigma^{\mathbf{T}}$ and ϵ. Although we believe our formalization of $\mathbf{U}_\sigma$ and its description as a series of set operations allows (considering theoretical complexity) a faster computation in our approach than in the approach of [53], though in [53] the approach is more general, since it considers an arbitrary number of clock and state invariants. Because we moved in precomputation the computation of the Parikh image and the language emptiness check (hence all exploration of the region automaton of the timed automaton is done before the simulation), we also conjecture that ϵ can be (again considering theoretical complexity) computed in polynomial time with regards to the size of the (silent part of the) timed automaton. We believe this would depend on the size of the regular timed interval generated in the precomputation.

However, we also believe that the precomputation time we require in our approach, and the complex regular timed interval generated can be a heavy burden in cases the exploration of the timed automaton is not too costly.

Finally, our implementation allows recovering easily at any point of the computation the set of reachable states in the future and *the exact time remaining before it will be reached*.

In our implementation of the diagnoser of [53], we wanted to compare the efficiency only of the transition computations ($\mathbf{U}_\sigma$ and $\mathbf{U}_d$), and not any efficiency difference which would come from the representation of the configurations. That's the reason why we chose to represent the configuration with *basic timed markings* and not with lists of pairs of states and zones like it is done in [53]. The computation of $\mathbf{U}_\sigma$ is done by exploring all possible transitions labeled σ and store the effect of each transition in the resulting marking. The computation of $\mathbf{U}_d$ is done by exploring all possible silent paths of length less than d and add its effect in the resulting marking.

In the next section we give some explanation of how the code of **DOTA**is organized.

12.2 Implementation

Our implementation is written in Python3.

As we discussed in section 12.1, both diagnosers are implemented as automata over timed domains, where the timed domain is the set of timed markings. The only difference lies in the functions computing the action and the delay transitions

As a consequence, both implementations benefit from the data structure we chose for representing timed intervals, which allows us to compute basic operations in linear time. Also, both structures use the same reachability graphs for either computing the sets of reachable configurations or the Parikh images.

TILib The library `TILib` contains all objects and functions which are related to timed set and marking algebra. It is separated into three layers represented by three classes (and hence three files).

• **MILib** The class `MILib` implements finite unions of intervals and basic set operations on them. A finite union of intervals is represented by a list of elements of $\widehat{\mathbb{C}}_M$ such that two consecutive elements $\widehat{r}$ and $\widehat{r}'$ defined the interval $\widehat{r} \cap \underset{\sim}{s}$ with $\underset{\sim}{s}$ being the complementary of $\widehat{r}'$ in $\mathbb{R}$. Intersection of two finite union of intervals, I and J is computed in $\mathcal{O}(|J| \log |I| + (|J|-1)|I|)$. If J is an interval, which is the main use of intersection, we compute then $I \cap J$ in $\mathcal{O}(\log |I|)$. Same way addition of $I + J$ is computed in $\mathcal{O}(|J||I| + (|J|-1)|I|)$ and in $\mathcal{O}(|I|)$ in case J is an interval which is again the main case. Union is computed in $\mathcal{O}(|I| + |J|)$.

• **RUILib** The class `RUILib` implements regular union of intervals and basic set operation on them. All implementations of set operations are based on the results of section 10.2 (c.f. remark 10.2.3) The intersection of a regular interval $E = (I, J, p, m)$ and an interval K is made with complexity $\mathcal{O}(|I| + \lceil \frac{\inf K + m}{p} \rceil |J|)$. Adapting a repeating interval to get a well formed regular interval (c.f. remark 10.2.4) is made in $\mathcal{O}(|I| + 2\lceil \frac{\sup J - \inf J}{p} \rceil |J|)$. Addition between E and K is in $\mathcal{O}(2|I| + 2(\lceil \frac{\sup J - \inf J}{p} \rceil + 1)|J|$. Union between two regular intervals $E_1 = (I_1, J_1, m_1, p_1)$ and $E_1 = (I_2, J_2, m_2, p_2)$ is computed in $\mathcal{O}(|I_1| + |I_2| + |J_1| + |J_2|)$.

• **TIMLib** The file `TIMLib` contains two classes, one for finite unions of regular timed intervals and one for regular timed marking.

► The class `FUTInterval` implements finite unions of timed interval as their representation. The information stored is then an array mapping any $\widehat{\mathbb{C}}_M$ to a regular union of interval. Let's consider the size of such timed set f as $|f|$ being bounded by M times the largest regular union of timed interval in its representation. The computation time of its intersection with a guard $f \cap \widehat{r} \cap \underset{\sim}{r}'$ can be bounded by $\mathcal{O}(M \log |f|)$. The size of the union of f and f' can be overestimated to $\mathcal{O}(|f| + |f'|)$.

► The class `TIMarking` implements regular timed markings as array mapping each state to a `FUTInterval`. We implemented a function that returns the set of reached and reachable states and which provide the exact time remaining before the reachable (but not reached) states will be reached.

ATDLib The library `ATDLib` collects all objects and functions related to the simulation of automata on timed domains and the computation of diagnosers. The general idea of the simulation is described in figure 12.1.

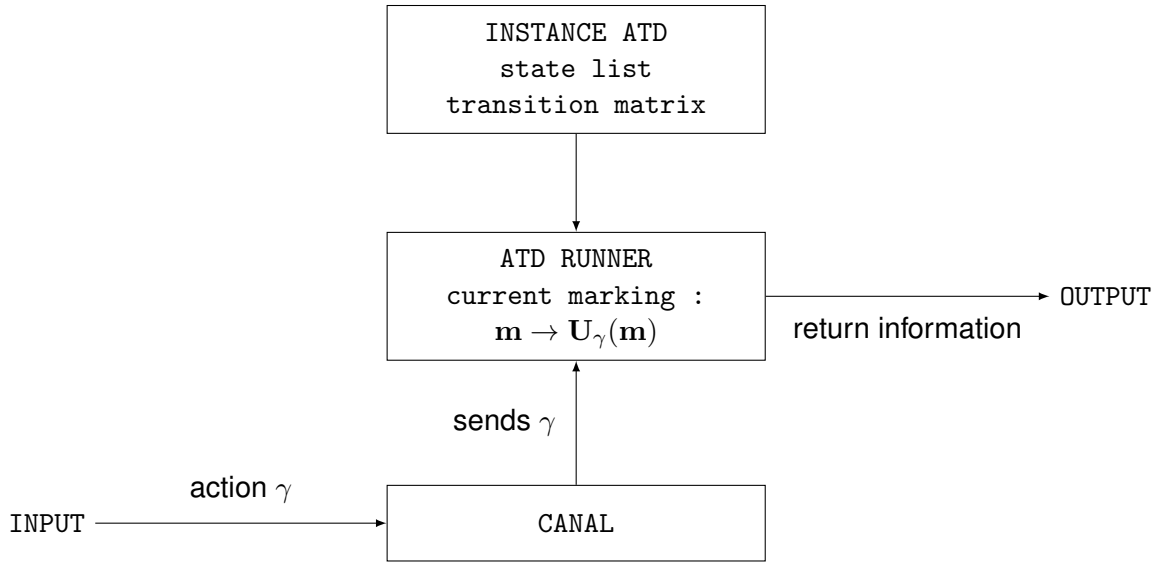


Figure 12.1: General functioning of an automaton simulation

• **Canal** The file `Canal` contains *canal object*, which, in several ways, have a function to get an action as input and send it to an instance of an `ATDRunner`. The input can be given through several forms. We can ask a user to enter a transition label or a delay (`STDINCanal`), we can ask only the transitions label and automatically input a delay at a regular pace (`AutoDelayCanal`) or we can let another part of the program enter the input (`STRCanal`).

• **ATDRunner** The class `ATDRunner` is an interface that simulates an automaton on a timed domain while listening to a canal. It is provided initially with a *atd object*, being an instance of any class of automata on timed domain. Then it initializes an initial marking based on the ATD object. It owns then two methods to simulate an action transition or a delay transition by calling the corresponding method of the initial automaton.

• **OTaumata** We can find in this library three classes of automata on timed domains. The class `OTAutomata` implements one clock timed automata. It owns a method that computes the accessibility graph of its silent part and a method `__call__` which computes the effect of any transition of this automaton on any regular timed marking. We also provide a function to load an `OTAutomata` from a file.

• **DiagAutomata** The class `DiagOTA` implements our diagnoser. It has to be initialized with two tables of precomputed information necessary to the computation of the closure as described in remark 11.2.3 and 11.2.4. It owns like `OTAutomata` a `__call__` method which computes the effect of any transition of the diagnoser on any timed marking. We recall that delays are not really computed but only stored as future padding to be applied while computing the next action transition. We additionally find two functions that compute from any object `OTAutomata` the two tables mentioned above. Finally, we can find a function which creates

from an object `OTAutomata` an object `DiagOTA`, its diagnoser. This function first precomputes the tables needed by the `DiagOTA` object and then initialize and return it.

• **TripakisDiagAutomata** The class `TripakisDOTA` implements the diagnoser of [53]. It does not require anything more than the transition function of the one-clock timed automaton it is meant to diagnose. It has as all ATD objects a method `__call__` which computes the effect of any transition of the diagnoser of any timed marking. Its implementation differs from the one in the class `DiagOTA` in ways discussed in section 12.1.

To diagnose a one-clock timed automaton by, say, our diagnoser, we combine then the modules in the way described in figure 12.2. The two `ATDRunners` for the timed automata and its diagnoser communicate through a communication canal implemented by a `STRCanal`. In the end, the diagnoser returns the set of reached states and the set of reachable states along with the delay before they will be reached.

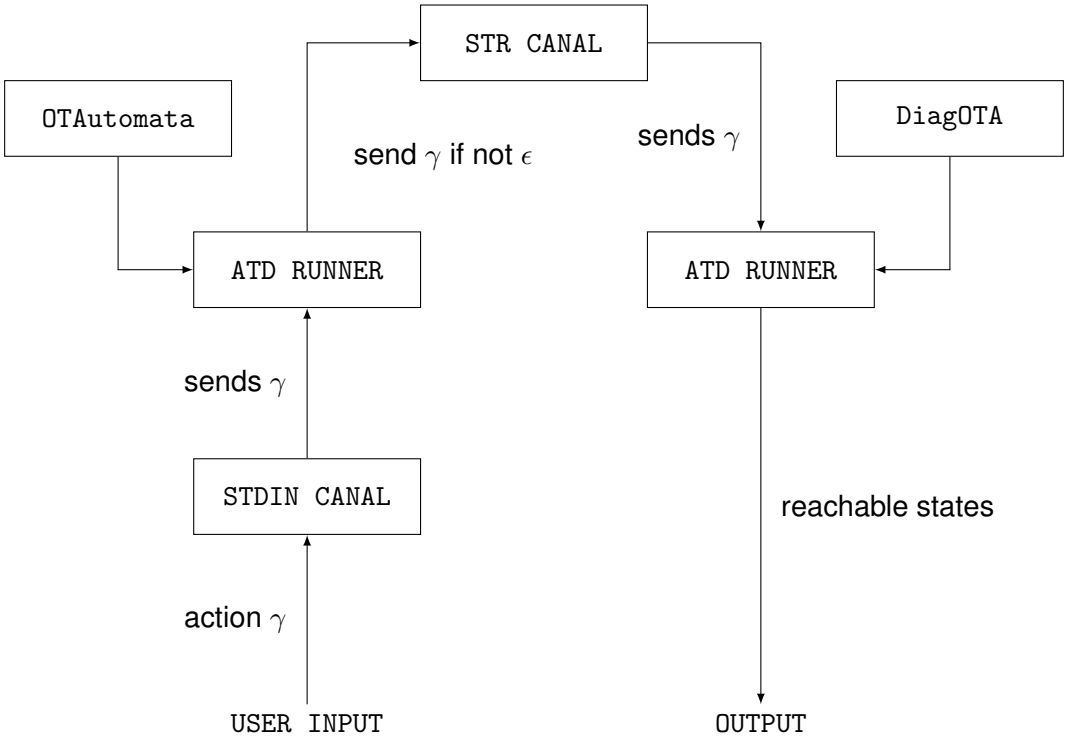


Figure 12.2: Implementation of One-Clock Timed Automata Diganosis

Other files have been implemented for test generation and execution. The interested reader can find more information on how to use them in the `README` file provided with the source code.

12.3 Results

Table 12.3 reports on the performances of both implementations on a small set of (randomly generated) examples. Those examples are given in Appendix A and are distributed with our prototype.

In Table 12.3, we give the important characteristics of each automaton (number of states and silent transitions), the amount of precomputation time used by our approach, and the

	Example2	Example3	Example4	Example6	Example8
#State/#Silent Trans	3/6	4/6	4/7	7/10	7/5
Precomputation Time	173.25s	0.38s	791.06s	11.01s	4.96s
Actions `DiagOTA`	0.014s	0.019s	0.029s	0.17s	0.15s
Actions `TripakisDOTA`	0.020s	0.078s	0.049s	0.26s	0.042
Ratio (actions)	0.73	0.25	0.59	0.64	3.71
Delays `DiagOTA`	0.000012s	0.0000011s	0.000011s	0.000011s	0.000012s
Delays `TripakisDOTA`	0.032s	0.057s	0.049s	0.30s	0.033s
Ratio (delays)	0.0004	0.0002	0.0002	0.00003	0.0004

Figure 12.3: Bench for 5 examples over 400 runs with 10 to 20 actions

average time (over 400 random runs) used in the two approaches to simulate action- and delay transitions.

As could be expected, our approach outperforms the approach of [53] on delay transitions by several orders of magnitude in all cases. The performances of both approaches are comparable when simulating action transitions.

The precomputation phase of our approach is intrinsically very expensive. In our examples, it takes from less than a second to more than 13 minutes, and it remains to be understood which factors make this precomputation phase more or less difficult. We may also refine our implementation of the computation of Parikh images, both on the algorithmic and encoding aspects, which is heavily used in the precomputation phase. We could then make better experiments by increasing the number of states and silent transitions.

We may also enhance our algorithm to generate more interesting samples of timed automata. We implemented a generator which can forbid automaton with cycles or allow only cycles where all guards are punctual. We did not, for now, get conclusive results, and we think it will be hard to before solving the two issues we spoke about above, about Parikh image computation and timed automata random generation.

Conclusion

In this thesis, we introduced an original framework to model quantitative and in particular timed systems. This framework, *automata on timed structures*, proved itself useful to discuss the representation and computability of formal models. With *controls* we could introduce and study classical language theoretic questions as well as diagnosis and control questions. Using automata on timed structures we proved that it is always possible, for any system without silent transition, to construct a language equivalent deterministic system, easily *representable and computable* using elements of the initial systems (i.e. not simply by modeling the system into a Turing machine and determinizing it). We applied this method on timed automata to construct a deterministic *powerset automaton* which can be easily computed from the timed automaton. We also recovered several determinization results about timed automata and highlighted some similarities between them but also fundamental differences which tell us more about the nature of each result. We finally used this powerset automaton to construct a diagnoser for one-clock timed automata. We implemented this diagnoser and a second diagnoser based on the construction of [53]. The comparison between the two implementations have not given conclusive results yet, even if we can already see that delay transitions are computed significantly faster in our approach.

Our implementation only works on *one-clock* timed automata contrarily to the diagnoser proposed in [53] which can handle timed automata with several clocks and with state invariants. Extending our construction to diagnose timed automata with several clocks is the next topic we are going to work on. We believe it can be done be founding an adequate multi-dimensional extension of regular timed intervals, probably by considering zones instead of intervals (which are one-dimensional zones) and defining regular timed zones. The implementation could then be efficiently done using difference bound matrix [2]. The extension of the diagnoser to handle state invariants could then be an additional question of interest.

We believe automata on timed domains could also be useful to get more insight on other topics like bisimulation equivalence or logic characterization. We could integrate those notions in our framework, making them dependent on the control equipped to the automaton. We could then study how they fit in and what this new point of view can bring to the understanding of those results.

We could also hope that the framework of automata on timed domain could give new insight about the characterization of determinizable timed automata, even if we have no certainty it would bring anything new and we know that this characterization would not be a decidable property since determinization of timed automata is not decidable [4].

We hope that this new framework would be helpful for others for applications and in domains we haven't considered yet.

Bibliography

[1] Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. Dense-timed pushdown automata. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, pages 35–44. IEEE Computer Society, 2012.

[2] Luca Aceto, Anna Ingólfsdóttir, Kim Guldstrand Larsen, and Jiri Srba. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, 2007.

[3] Luca de Alfaro. Stochastic transition systems. In *Proceedings of the 9th International Conference on Concurrency Theory*, CONCUR '98, pages 423–438, Berlin, Heidelberg, 1998. Springer-Verlag.

[4] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, apr 1994.

[5] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. A determinizable class of timed automata. In *CAV – Computer Aided Verification*, pages 1–13. Springer, Berlin, Heidelberg, 1994.

[6] Rajeev Alur, Salvatore La Torre, and Parthasarathy Madhusudan. Perturbed timed automata. In *International Workshop on Hybrid Systems: Computation and Control*, pages 70–85. Springer, 2005.

[7] Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In *STOC – Symposium on Theory of computing*, page 202, New York, New York, USA, 2004. ACM Press.

[8] André Arnold. *Finite Transition Systems: Semantics of Communicating Systems*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1994.

[9] Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *Journal of the ACM*, 49(2):172–206, mar 2002.

[10] Christel Baier, Nathalie Bertrand, Patricia Bouyer, and Thomas Brihaye. When Are Timed Automata Determinizable? In *ICALP – Automata, Languages, and Programming*, number April, pages 43–54. Springer Berlin Heidelberg, 2009.

[11] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. 2006.

[12] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Advanced Course on Petri Nets*, pages 87–124. Springer, 2003.

[13] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H Roux. Comparison of the expressiveness of timed automata and time petri nets. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 211–225. Springer, 2005.

[14] Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the Expressive Power of Silent Transitions in Timed Automata. *Fundamenta Informaticae*, 36(2,3):145–182, jan 1998.

[15] Nathalie Bertrand, Amélie Stainer, Thierry Jéron, and Moez Krichen. A game approach to determinize timed automata. *Formal Methods in System Design*, 46(1):42–80, feb 2015.

[16] Devendra Bhave and Shibashis Guha. Adding dense-timed stack to integer reset timed automata. In *International Workshop on Reachability Problems*, pages 9–25. Springer, 2017.

[17] Mikołaj Bojańczyk and Sławomir Lasota. A Machine-Independent Characterization of Timed Languages. In *ICALP – Automata, Languages, and Programming*, pages 92–103. Springer Berlin Heidelberg, 2012.

[18] Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault Diagnosis Using Timed Automata. In *FoSSaCS – Foundations of software science and computational structures*, pages 219–233. Springer, Berlin, Heidelberg, 2005.

[19] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2-3):291–345, aug 2004.

[20] Patricia Bouyer, Samy Jaziri, and Nicolas Markey. On the determinization of timed systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 25–41. Springer, 2017.

[21] Patricia Bouyer, Samy Jaziri, and Nicolas Markey. Efficient timed diagnosis using automata with timed domains. In *RV 2018 - 18th International Conference on Runtime Verification*, pages 1–26, Limassol, Cyprus, November 2018.

[22] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust Weighted Timed Automata and Games. In *FORMATS – Formal Modeling and Analysis of Timed Systems*, volume 8053 LNCS, pages 31–46. Springer Berlin Heidelberg, 2013.

[23] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 298–302. Springer, 1998.

[24] Fabrice Chevalier, Deepak D'Souza, and Pavithra Prabhakar. On Continuous Timed Automata with Input-Determined Guards. In *FSTTCS – Foundations of Software Technology and Theoretical Computer Science*, pages 369–380, jan 2006.

[25] Edmund M Clarke, Thomas A Henzinger, Helmut Veith, and Roderick P Bloem. *Handbook of model checking*. Springer, 2016.

[26] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.

[27] Lorenzo Clemente and Sławomir Lasota. Timed pushdown automata revisited. In *LICS – Logic in Computer Science*, pages 738–749. IEEE, mar 2015.

[28] Cătălin Dima and Ruggero Lanotte. Removing all silent transitions from timed automata. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 118–132. Springer, 2009.

[29] Deepak D'souza. A Logical Characterisation of Event Clock Automata. *International Journal of Foundations of Computer Science*, 14(04):625–639, aug 2003.

[30] Eric Fabre. Diagnosis and automata. *Lecture Notes in Control and Information Sciences*, 433:85–106, 01 2013.

[31] Jean-Christophe Filliâtre. Deductive software verification, 2011.

[32] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1):63 – 92, 2001. ISS.

[33] Olivier Finkel. Undecidable Problems About Timed Automata. In *FORMATS – Formal Modeling and Analysis of Timed Systems*, pages 187–199. Springer Berlin Heidelberg, 2006.

[34] Sahika Genc and Stéphane Lafortune. Predictability of event occurrences in partially-observed discrete-event systems. *Automatica*, 45(2):301–311, 2009.

[35] Thomas A Henzinger. The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, pages 265–292. Springer, 2000.

[36] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. Timed transition systems. In *Proceedings of the Real-Time: Theory in Practice, REX Workshop*, pages 226–251, Berlin, Heidelberg, 1992. Springer-Verlag.

[37] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[38] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[39] Daniel Kirsten and Ina Mäurer. On the determinization of weighted automata. *Journal of Automata, Languages and combinatorics*, 10(2/3):287–312, 2005.

[40] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009.

[41] Harry R Lewis and Christos H Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall PTR, 1997.

[42] Philip Merlin and David Farber. Recoverability of communication protocols–implications of a theoretical study. *IEEE transactions on Communications*, 24(9):1036–1043, 1976.

[43] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997.

[44] Mehryar Mohri. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer, 2009.

[45] Joël Ouaknine and James Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Logic in Computer Science, 2004. Proceedings of the 19th Annual IEEE Symposium on*, pages 54–63. IEEE, 2004.

[46] Rohit J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, October 1966.

[47] Meera Sampath, Raja Sengupta, Stephane Lafortune, Kasim Sinnamohideen, and Demosthenis C Teneketzis. Failure diagnosis using discrete-event models. *IEEE transactions on control systems technology*, 4(2):105–124, 1996.

[48] Amélie Stainer. *Contribution to the Verification of Timed Automata: Determinization, Quantitative Verification and Reachability in Networks of Automata*. Theses, Université Rennes 1, 2013.

[49] P. Vijay Suman, Paritosh K. Pandya, Shankara Narayanan Krishna, Lakshmi Manasa, Shankara Narayanan Krishna, P. Vijay Suman, Paritosh K. Pandya, Shankara Narayanan Krishna, and Lakshmi Manasa. Timed Automata with Integer Resets: Language Inclusion and Expressiveness. In *FORMATS – Formal Modeling and Analysis of Timed Systems*, volume 5215 LNCS, pages 78–92, Berlin, Heidelberg, jan 2008. Springer Berlin Heidelberg.

[50] Claus Thrane, Uli Fahrenberg, and Kim G Larsen. Quantitative analysis of weighted transition systems. *The Journal of Logic and Algebraic Programming*, 79(7):689–703, 2010.

[51] Jan Tretmans. Conformance testing with labeled transition systems: Implementation relations and test generation. *Computer networks and ISDN systems*, 29(1):49–79, 1996.

[52] Stavros Tripakis. Description and schedulability analysis of the software architecture of an automated vehicle control system. In *International Workshop on Embedded Software*, pages 123–137. Springer, 2002.

[53] Stavros Tripakis. Fault Diagnosis for Timed Automata. In *FTRTFT – Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 205–221. Springer, Berlin, Heidelberg, 2002.

[54] Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Information Processing Letters*, 99(6):222–226, sep 2006.

[55] Farn Wang. Redlib for the formal verification of embedded systems. In *Leveraging Applications of Formal Methods, Verification and Validation, 2006. ISoLA 2006. Second International Symposium on*, pages 341–346. IEEE, 2006.

Appendices

Appendix A

Examples definition

A.1 Example2

```
# States. Size : 3
q0;q1;q2
# Transitions
q0;[0,2];0;q0;a
q0;[1,1];0;q1;a
q1;[1,2];0;q1;a
q1;[2,2];0;q2;a
q2;[0,inf[;0;q0;a
q0;]0,2];0;q0;b
q0;[0,inf[;0;q1;b
q1;]1,2];0;q0;b
q1;[0,2[;0;q1;b
q2;[0,0];0;q1;b
q2;]0,1];0;q2;b
q0;]2,inf[;0;q0;e
q0;[1,2[;0;q1;e
q0;[0,2[;1;q2;e
q1;]0,2];0;q0;e
q1;]2,inf[;0;q1;e
q2;[1,1];0;q0;e
```

A.2 Example3

```
# States. Size : 4
q0;q1;q2;q3
# Transitions
q0;]1,2];0;q1;a
q0;[2,2];1;q3;a
q1;[0,1];0;q1;a
q2;]1,2[;1;q1;a
q2;]0,2];0;q2;a
q2;[1,2];0;q3;a
q3;[1,1];1;q2;a
```

```
q0;[2,inf[;0;q0;b
q0;[2,2];1;q2;b
q2;[0,0];0;q3;b
q3;]1,2];1;q0;b
q3;]0,inf[;1;q1;b
q3;[0,0];1;q3;b
q0;[0,inf[;1;q0;e
q0;[2,inf[;1;q1;e
q0;[0,1[;1;q3;e
q1;]0,2];1;q1;e
q1;]0,inf[;1;q2;e
q2;[1,inf[;1;q0;e
```

A.3 Example4

```
# States. Size : 4
q0;q1;q2;q3
# Transitions
q0;[0,2[;0;q0;a
q0;]0,2[;1;q1;a
q0;]1,inf[;1;q2;a
q0;]1,2[;0;q3;a
q1;]1,2];0;q0;a
q1;[0,1];1;q1;a
q1;[0,2[;0;q2;a
q1;]0,1[;0;q3;a
q2;[1,inf[;1;q0;a
q2;[1,2[;0;q3;a
q3;[2,inf[;0;q0;a
q3;[1,2];0;q2;a
q0;]1,2];1;q1;b
q1;[0,1[;1;q2;b
q1;[0,2];0;q3;b
q2;[0,0];1;q0;b
q2;[0,0];0;q2;b
q3;[0,2[;1;q0;b
q3;]1,2];1;q2;b
q3;[0,2];1;q3;b
q0;]0,inf[;1;q0;e
q0;]0,2];1;q1;e
q0;[1,1];0;q2;e
q0;]2,inf[;0;q3;e
q1;[0,inf[;1;q0;e
q1;[0,2[;0;q1;e
q2;[0,inf[;0;q0;e
```

A.4 Example6

```
# States. Size : 7
q0;q1;q2;q3;q4;q5;q6
# Transitions
q0;]0,1[;1;q0;a
q0;]0,2];1;q4;a
q0;[2,2];1;q6;a
q1;[0,0];1;q0;a
q1;[1,2];0;q4;a
q1;[0,0];0;q5;a
q2;[1,2];1;q1;a
q2;[0,3[;0;q2;a
q2;]0,2[;0;q5;a
q3;[0,inf[;1;q0;a
q3;]0,3];1;q4;a
q4;[3,3];1;q1;a
q4;]1,2];0;q3;a
q4;[0,2[;0;q4;a
q4;[2,3];0;q6;a
q5;]0,2[;0;q0;a
q5;]1,3[;0;q3;a
q5;[0,3[;1;q6;a
q6;]1,2[;1;q0;a
q6;[3,inf[;0;q1;a
q6;[1,2];1;q2;a
q6;]0,3[;0;q3;a
q6;[3,3];1;q5;a
q0;]2,inf[;1;q3;b
q0;]0,2[;0;q4;b
q0;[0,2];1;q5;b
q1;[1,3];0;q0;b
q1;[3,inf[;0;q2;b
q1;[0,1];1;q6;b
q2;[0,1[;0;q0;b
q2;[2,3[;0;q1;b
q3;[0,2];0;q3;b
q3;]0,2[;1;q4;b
q3;]2,3[;1;q6;b
q4;[3,inf[;1;q2;b
q4;[0,3];1;q6;b
q5;[0,0];0;q0;b
q5;[1,1];1;q2;b
q5;]1,3[;1;q4;b
q5;[3,3];1;q6;b
q6;[1,3];1;q1;b
q6;]1,2];1;q3;b
q0;[1,inf[;1;q0;e
q0;[0,0];1;q2;e
```

```
q0;[1,2];0;q4;e
q1;]0,3];1;q0;e
q1;]2,3[;0;q1;e
```

A.5 Example8

```
# States. Size : 7
q0;q1;q2;q3;q4;q5;q6
# Transitions
q0;[2,2];0;q0;a
q0;[0,3[;1;q1;a
q0;[2,2];1;q3;a
q0;[1,3];1;q5;a
q1;]0,3];1;q0;a
q1;]0,2[;0;q5;a
q1;]2,3];1;q6;a
q2;]3,inf[;1;q1;a
q2;[2,2];1;q2;a
q2;]0,3[;1;q4;a
q2;]1,3[;1;q5;a
q3;]1,3[;0;q1;a
q3;]1,2[;0;q3;a
q4;[1,3];0;q0;a
q4;[0,3];0;q2;a
q4;[0,3[;1;q3;a
q4;]2,3];0;q4;a
q4;[0,3[;0;q5;a
q4;[2,inf[;0;q6;a
q5;[2,inf[;1;q0;a
q5;[0,0];1;q1;a
q5;]1,2[;1;q2;a
q5;]0,1];1;q5;a
q6;]1,2[;1;q1;a
q6;]1,2[;1;q3;a
q6;[0,1];0;q5;a
q6;]2,3[;0;q6;a
q0;[3,3];1;q2;b
q0;]2,3[;0;q4;b
q0;]2,3[;0;q5;b
q0;]2,3];1;q6;b
q1;]1,inf[;1;q4;b
q1;]0,2];0;q6;b
q2;[1,inf[;1;q0;b
q2;]3,inf[;0;q1;b
q2;[1,3[;1;q4;b
q2;[0,3[;0;q5;b
q3;[1,3[;1;q0;b
q3;]3,inf[;1;q2;b
q3;[2,inf[;1;q4;b
```

```
q4;]0,2[;1;q1;b
q4;]0,inf[;1;q2;b
q4;]0,2[;1;q3;b
q4;[1,3[;1;q4;b
q4;]3,inf[;0;q5;b
q5;[0,2];1;q1;b
q6;]1,3];1;q3;b
q6;[1,2[;1;q4;b
q6;]1,2];0;q5;b
q6;[0,1];1;q6;b
q0;[0,3[;1;q1;e
q0;]1,3];1;q3;e
q0;]0,3[;0;q5;e
q0;]3,inf[;1;q6;e
q1;]1,2[;0;q0;e
q1;]2,3];0;q2;e
q1;[1,2[;1;q3;e
q1;[0,1[;0;q5;e
q1;]0,3];1;q6;e
q2;[1,inf[;1;q0;e
```

Titre: Automates sur les structures temporisées

Mots clés: Automates, Vérification, Déterminisation, Diagnostic

Résumé:
Les systèmes digitaux jouent un rôle croissant dans le bon fonctionnement de notre société. On confie aujourd'hui des tâches importantes à des algorithmes. Déjà largement utilisés dans des domaines aussi délicat que le transport, la chirurgie ou l'économie, il est aujourd'hui de plus en plus question de faire de la place aux systèmes digitaux dans les domaines sociaux et politiques : vote électronique, algorithmes de sélection, profilage électoral... Pour les tâches confiées à des algorithmes, la responsabilité est déplacées de l'exécutant vers les concepteurs, développeurs et testeurs de ces algorithmes. Il incombe aussi aux chercheurs qui étudient ces algorithmes de proposer des techniques de vérifications fiable qui pourront être utilisées à tous les niveaux : conception, développement et test. Les méthodes de vérifications formelles donnent des outils mathématiques pour prévenir des erreurs à chaque niveaux. Parmi elle, le diagnostic d'erreur consiste en la création d'un diagnostiqueur basé sur un modèle formel du système à vérifier.

Pour les systèmes modélisés par des automates temporisés, il n'est pas toujours possible de construire un diagnostiqueur sous la forme d'un autre automate temporisé. En effet les automates temporisés, introduits par [4] dans les années 90 et largement étudiés et utilisés depuis pour modéliser des systèmes avec contraintes temporelles, ne sont pas déterminisable. Une machine plus puissante qu'un automate temporisé peut cependant être utilisée pour construire le diagnostiqueur d'un automate temporisé comme le montre [53]. L'aboutissement de ce travail de thèse est la construction automatique d'un diagnostiqueur pour les automates temporisés à une horloge. Ce diagnostiqueur, dans le même esprit que celui de [53], est une machine plus puissante qu'un automate temporisé. Cette thèse présente un nouveau cadre formel, étudie le problème de la déterminisation dans ce cadre et utilise les résultats obtenus pour construire un diagnostiqueur pour les automates temporisés à une horloge. Cette technique est implémentée dans un outils, **DOTA**, et comparée à la machine construite par [53].

Title: Automata on Timed Structures

Keywords: Automata, Verification, Determinization,Diagnosis

Abstract:
Digital system are now part of our society. They are used in a wide range of domains and in particular they have to handle delicate tasks. Already used in domains such as transportation, surgery or economy, we speak now of using digital systems for social or political matters : electronic vote, selection algorithms, electoral profiling... For task handled by algorithm, the responsibility is moved from the executioner to the designer, developer and tester of those algorithms. It is also the responsibility of computer scientists who study those algorithms to propose reliable techniques of verification which will be applicable in the design, the development or the testing phase. Formal verification methods provide mathematical tools to prevent executions error in all phases. Among them, fault-diagnosis consist on the construction of a diagnoser based on a formal model of the system we aim to check. For systems modeled by timed automata, it is not always possible to construct a timed automaton to diagnose it. Indeed timed automata, introduce in the nineties by [4] and widely studied and used since to model timed systems, are not determinizable. A machine, more powerful than a timed automaton, can still be used to construct the diagnoser of a timed automaton as it is done in [53]. This thesis work aim at constructing a diagnoser for any one-clock timed automata. This diagnoser is constructed with the help of a machine more powerful than timed automata, following the idea of [53]. In this thesis we introduce a formal framework, study determinization in this framework and use those results to construct a one-clock timed automata diagnozer. This technique is implemented in a tool, **DOTA**, and is compared to the technique used in [53].

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