N° d'ordre : 4182 ANNÉE 2010

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Benoît Delahaye

préparée à l'unité de recherche S4 - IRISA
Institut de Recherche en Informatique et Systèmes aléatoires
IFSIC

Modular Specification and Compositional Analysis of Stochastic Systems

Thèse soutenue à Rennes
le 08 octobre 2010

devant le jury composé de :

Jean-Pierre Banâtre
Prof. Univ. Rennes 1/président

Joost-Pieter Katoen
Prof. RWTH Aachen Univ./rapporteur

Kim G. Larsen
Prof. Aalborg Univ/examinateur

Sergio Bovelli
Dr. EADS Innovation Works/examinateur

Axel Legay
CR INRIA-Rennes/examinateur

Benoît Caillaud
CR INRIA-Rennes/directeur de thèse

Remerciements

Parce que cette thèse ne serait rien sans toutes ces personnes qui m'ont accompagnées lors de cette longue aventure, je tiens à écrire ces quelques lignes pour les remercier.

Je remercie tout d'abord mes deux rapporteurs de thèse, monsieur Joost-Pieter Katoen et madame Marta Kwiatkowska d'avoir pris le temps de lire en détails mes travaux, de les corriger, de les commenter et surtout de les comprendre.

Je remercie également les membres de mon jury de m'avoir fait l'honneur de leur présence. Merci à Jean-Pierre Banâtre qui a su dégager quelques heures de son emploi du temps extrêmement chargé pour assister à ma soutenance. Merci à Sergio Bovelli d'avoir fait le déplacement depuis Munich pour écouter mes travaux dont seule une partie était proche de ses domaines d'intérêt. Un énorme merci à Kim G. Larsen qui, en plus d'avoir fait le déplacement depuis Aalborg pour assister à ma soutenance, a aussi lu en détails, commenté et corrigé mon manuscrit. Merci aussi de m'avoir accueilli à Aalborg pendant ma thèse et de m'avoir accordé autant de temps. Ce séjour m'a été extrêmement profitable de par le dynamisme et la qualité de la recherche qui se fait au CISS et j'espère sincèrement que cette collaboration durera encore longtemps. Merci à Benoît Caillaud, mon directeur de thèse, pour la confiance et l'encadrement qu'il m'a apporté pendant ces années. Bien entendu, un merci particulier à Axel Legay qui a su me pousser, me motiver et m'accompagner pendant cette "dernière année" de thèse. Tu m'as montré un autre visage de la recherche et bien qu'on ait eu nos désaccords et autres "coups de gueule", je sais que tout ce que tu m'as demandé était dans mon intérêt et je t'en remercie.

Merci aussi à tous mes autres coauteurs qui ont participé, de près ou de loin, à tous les travaux présentés dans cette thèse: Andrzej Wasowski, Mikkel L. Pedersen, Ananda Basu, Marius Bozga, Saddek Bensalem...

Il va sans dire que je remercie tout le personnel de l'IRISA et particulièrement des équipes S4, DISTRIBCOM et VERTECS que j'ai beaucoup côtoyées. Je me rends compte combien les conditions de travail sont agréables à l'IRISA de par la compétence et la qualité du personnel qui y travaille. En particulier, un grand merci aux "filles de la cafet", Pierrette, Laurence et les autres, qui nous accueillent tous les jours avec le sourire.

Je tiens aussi à remercier les différents professeurs de l'ENS Cachan Antenne de Bretagne qui m'ont dans un premier temps suivi et encadré pendant mes études, puis écouté et accompagné pendant mon monitorat: Luc Bougé, Claude Jard, David Cachera, David Pichardie, Anne Bouillard, Dominique Lavenier.... Ces expériences enrichissantes m'ont beaucoup apporté. Dans le même élan, je remercie tous les moniteurs de l'ENS avec qui j'ai pu travailler à un moment ou à un autre: Ludo, Fanfoué, Kevin, Romain, Christophe...

Merci à tous mes amis “de Rennes”, qui m’ont supporté dans mes pires moments de “gronchonnerie”, qui m’ont accompagné à la bavette le midi et avec qui j’ai fait tant de pauses café: Fanfoué, Jérémy, J.P., Romaric, Patoche et j’en passe. C’est toujours un plaisir que de passer un moment avec vous pour sortir la tête du boulot. Merci aussi à ceux que je vois moins souvent, mais avec qui j’ai passé tant de bonnes soirées pendant ma thèse et mes études: Ludo et Christelle, Mickael, Yves-Pol, Jean-Baka, Romaric, Pierre, Fanch, Sophie, MLP, Lima, Jon, Arnaud et Alina, Marie, Benjamin...

Merci à tous mes amis “de Bordeaux”, qui, même si je ne les vois qu’à l’occasion, comptent énormément pour moi: Anne, Axelle, Pierrot, Seb et Anne-élise, Aurélie, Mathieu, Rémi, Marie et Bitou, Sam et Jeannick, Marie Anne et Gaetan, Aurel, Manue, Damien, William et Delphine... Sachez que si les occasions sont rares, elles n’en sont que plus appréciées !

Merci aux amis de Maurice que je vois encore plus rarement mais à qui je pense toujours: Murvin, Loic, Amrish, Nicolas, Roshni, Stephanie, Pramil...

Merci à ma famille qui est toujours là pour moi. Merci à mes parents qui s’intéressent à ce que je fais même quand ça n’est pas leur tasse de thé. Merci à Françoise, Jean-George, Plume et Mme Carda qui font toujours le déplacement pour me voir quand je passe à la maison. Merci à tous mes autres oncles et tantes, cousins et cousines que je vois si peux mais que j’aimerai voir plus...

Merci à la famille Guibert qui m’accueille comme un roi à chaque fois que je vais au Mans.

Finalement, merci merci merci Flo d’être là pour moi chaque jour. Merci de me comprendre, de me supporter, de me motiver et surtout de me rendre heureux. Chaque minute passée avec toi n’est que bonheur et j’espère que ça continuera encore longtemps.

Contents

Preamble **i**

Résumé de la thèse :
Spécification Modulaire et Analyse Compositionnelle de Systèmes Stochastiques **iii**
0.1 Introduction . . . iii
0.1.1 Contexte . . . iii
0.1.2 Contributions et plan de la thèse . . . v
0.2 Chaînes de Markov à Intervalles . . . ix
0.3 Chaînes de Markov à Contraintes . . . xi
0.4 Contrats (Probabilistes) . . . xii
0.5 Abstraction Stochastique et Model-Checking d'un Système Hétérogène de grande taille . . . xiv

1 Introduction **1**

2 Interval Markov Chains **7**
2.1 Introduction . . . 7
2.2 Background . . . 8
2.3 Refinement Relations . . . 12
2.3.1 Weak and Strong Refinement . . . 13
2.3.2 Granularity . . . 13
2.3.3 Deciding Thorough Refinement . . . 14
2.4 Determinism . . . 21
2.5 Common Implementation and Consistency . . . 23
2.6 Conclusion and Related Work . . . 28

3 Constraint Markov Chains **31**
3.1 Introduction . . . 31
3.2 Constraint Markov Chains . . . 32
3.3 Consistency, Refinement and Conjunction . . . 38
3.3.1 Consistency . . . 39
3.3.2 Refinement . . . 41
3.3.3 Conjunction . . . 46
3.4 Compositional Reasoning . . . 48
3.4.1 Independent parallel composition . . . 49
3.4.2 Synchronization . . . 51
3.4.3 Comparison of conjunction and parallel composition . . . 53

3.5 Disjunction and Universality . . . 54
3.5.1 On the Existence of a Disjunction of CMCs . . . 54
3.5.2 The Universality Problem for CMCs . . . 55
3.6 Deterministic CMCs . . . 57
3.7 Polynomial CMCs . . . 64
3.8 On the relation with Probabilistic Automata . . . 65
3.8.1 Reduction from Simulation . . . 66
3.8.2 Encoding Probabilistic Simulation . . . 68
3.9 Related Work and Concluding Remarks . . . 71

4 Probabilistic contracts: a compositional reasoning methodology for the design of stochastic systems 73
4.1 Introduction . . . 73
4.2 Preliminaries . . . 74
4.3 Non-Probabilistic Contracts . . . 76
4.3.1 Contracts . . . 76
4.3.2 Compositional reasoning . . . 78
4.3.3 Compositional Verification . . . 79
4.3.4 Effective algorithms/representations . . . 83
4.4 Probabilistic Contracts . . . 84
4.4.1 Probabilistic contracts . . . 87
4.4.2 Operations on probabilistic contracts and Compositional reasoning . . . 87
4.4.3 Effective algorithms/representations . . . 99
4.5 Some Related Work . . . 101
4.6 Achievements and Future Work . . . 102

5 Statistical Abstraction and Model-Checking of Large Heterogeneous Systems 105
5.1 Introduction . . . 105
5.2 An Overview of Statistical Model Checking . . . 107
5.2.1 Qualitative Answer using Statistical Model Checking . . . 108
5.2.2 Quantitative Answer using Statistical Model Checking . . . 109
5.2.3 Playing with Statistical Model Checking Algorithms . . . 109
5.3 Validation Method and the BIP Toolset . . . 109
5.3.1 Validation Method: Stochastic Abstraction . . . 109
5.3.2 An Overview of BIP . . . 110
5.4 Case Study: Heterogeneous Communication System . . . 112
5.4.1 Server . . . 113
5.4.2 Network Access Controller (NAC) . . . 114
5.4.3 Device . . . 116
5.4.4 Complexity of the modeling . . . 116
5.5 Experiments on the HCS . . . 116
5.5.1 The Precision Time Protocol IEEE 1588 . . . 116
5.5.2 Parametric Precision Estimation for PTP . . . 119
5.5.3 Model Simulations . . . 121
5.6 Experiments on Precision Estimation for PTP . . . 121
5.6.1 Property 1: Synchronization . . . 123

5.6.2 Property 2: Average failure 128
5.6.3 Clock Drift 129
5.7 Another case study: the AFDX Network 131
5.8 Achievements and Future work 134

6 Conclusion **137**

List of Figures

1 IMCs S_1 et S_2 illustrant la non-clôture par conjonction. vi
2 Modèle de l'exemple HCS. Les NACs effectuent l'ordonnancement des différents messages échangés entre le serveur et les différents dispositifs représentés par les rectangles. viii
3 Exemples de chaînes de Markov et de chaînes de Markov à intervalles. x
4 Deux spécifications (CMCs) et deux implémentations (MC) d'un relais optique. xii
5 Un exemple de contrats et de propriété de fiabilité. Pour chaque contrat, la lettre V représente l'ensemble des variables, la lettre A représente les hypothèses et la lettre G représente les garanties. xiv
6 Abstraction stochastique du protocole PTP entre le serveur et un composant. . . xvi

1.1 IMCs showing non-closure under conjunction 3
1.2 HCS example model. NACs perform scheduling of the different messages exchanged by the server and the devices shown as the many boxes on the diagram. 6

2.1 Examples of Markov Chains and Interval Markov Chains. 9
2.2 Illustration of satisfaction relations using direct and indirect redistribution of the probability mass. 10
2.3 Illustration of strong and weak refinement relations. 12
2.4 IMCs I_4 and I_5 such that I_4 thoroughly but not weakly refines I_5 14
2.5 An example of the translation from Modal Transition Systems to IMCs 16
2.6 An IMC I whose semantics cannot be captured by a deterministic IMC 23
2.7 IMCs I_6, I_7, and I_8 26
2.8 An IMC and its pruned version 28

3.1 Two specifications (CMCs) and two implementations (MCs) of an optic relay . 33
3.2 Examples of refinement and conjunction for CMCs 35
3.3 Examples of refinement and satisfaction for CMCs 35
3.4 Illustration of normalization. 37
3.5 Illustration of the pruning algorithm. 39
3.6 CMCs S_a and S_b. 44
3.7 Correspondence matrices for $S_a \preceq S_b$. 44
3.8 CMCs S_c and S_d. 45
3.9 Parallel composition and synchronization of CMCs 49
3.11 Common structure of conjunction and parallel composition 54
3.12 CMCs S and S^x 56
3.13 A CMC T whose set of implementations cannot be represented with a deterministic CMC 57

3.14 Reducing a PA to CMC. There $\widehat{\pi}$ denotes a distribution constraint, which has a unique solution π. 66
3.15 An attempt to visualize the second encoding. π^a_* denotes a constraint expressing a probability vector that is a linear combination of all probability distributions labeled by a. Below this is formalized as $\varphi(2k + i')(x)$. 69

4.1 Illustration of mean-availability. 77
4.2 Illustration of a scheduler defining a probability measure on a set of executions 86
4.3 Reliability : Example 98
4.4 A symbolic transition system and its split. 100
4.5 The product of a split symbolic transition system with a Markov Chain. 100

5.1 An atomic component: Router. 111
5.2 Composite Component: Server. 111
5.3 A device component. 112
5.4 HCS Example Model. 113
5.5 Heterogeneous Communication System (HCS). 114
5.6 Component: Classifier 114
5.7 Abstract stochastic PTP between the server and a device. 118
5.8 One round of the PTP protocol. 119
5.9 Delay distribution for Device(0,3) and Device(3,3) using fixed priorities for 4000 measurements. 122
5.10 Delay distribution for Device(0,3) and Device(3,3) using weighted fair queuing mixed with priorities for 2000 measurements. 122
5.11 Probability of satisfying the bounded accuracy property for a bound $\Delta = 50\mu s$ and the asymmetric version of PTP. 123
5.12 Probability of satisfying the bounded accuracy property for a bound $\Delta = 50\mu s$ and weighted fair queuing mixed with priorities. 125
5.13 Probability of satisfying the bounded accuracy property as a function of the bound Δ for the asymmetric version of PTP. 126
5.14 Probability of satisfying bounded accuracy as a function of the bound for weighted fair queuing mixed with priorities. 127
5.15 Evolution of the probability of satisfying the bounded accuracy property with the length of the simulations. 128
5.16 Average proportion of failures for a bound $\Delta = 50\mu s$ and the asymmetric version of PTP. 129
5.17 Average proportion of failures for a bound $\Delta = 50\mu s$ and weighted fair queuing mixed with priorities. 130
5.18 Average proportion of failures as a function of the bound Δ for the asymmetric version of PTP. 131
5.19 Average proportion of failures for a bound $\Delta = 50\mu s$ and weighted fair queuing mixed with priorities. 132
5.20 Evolution of the average proportion of failures with the length of the simulations.133

Preamble

The results presented in this thesis are based on the following publications.

- A. Basu, S. Bensalem, M. Bozga, B. Caillaud, B. Delahaye, and A. Legay. Statistical abstraction and model-checking of large heterogeneous systems. In *FMOODS/FORTE, 5th IFIP International Conference on Formal Techniques for Distributed Systems, Amsterdam, The Netherlands,* volume 6117 of *Lecture Notes in Computer Science,* pages 32–46. Springer, 2010.
- B. Delahaye, B. Caillaud, A. Legay Probabilistic Contracts : A Compositional Reasoning Methodology for the Design of Stochastic Systems. In *ACSD, 10th International Conference on Application of Concurrency to System Design*, Braga, Portugal, 2010.
- B. Caillaud, B. Delahaye, K.G. Larsen, A. Legay, M.L. Pedersen, A. Wąsowski. Compositional Design Methodology with Constraint Markov Chains. In *QEST, 7th International Conference on Quantitative Evaluation of SysTems*, Williamsburg, Virginia, USA, 2010.
- A. Basu, S. Bensalem, M. Bozga, B. Delahaye, A. Legay, and E. Siffakis. Verification of an afdx infrastructure using simulations and probabilities. In *RV, 1st conference on Runtime Verification*, Malta, 2010.
- S. Bensalem, B. Delahaye, A. Legay. Statistical Model Checking: Present and Future. In *RV, 1st conference on Runtime Verification*, Malta, 2010.
- B. Delahaye, B. Caillaud, A. Legay Probabilistic Contracts : A Compositional Reasoning Methodology for the Design of Systems with Stochastic and/or non-Deterministic Aspects. In *Formal Methods in System Design*, 2010.

Other works by the same author:

- B. Delahaye, A. Legay. Statistical Model Checking: an overview. In *CoRR*, abs/1005.1327, 2010.
- B. Delahaye, J-P. Katoen, K.G. Larsen, A. Legay, M.L. Pedersen, F. Sher, A. Wąsowski. Abstract Probabilistic Automata. In *VMCAI, conference on Verification, Model Checking and Abstract Interpretation*, Austin, Texas, 2011.
- B. Caillaud, B. Delahaye, K.G. Larsen, A. Legay, M.L. Pedersen, A. Wąsowski. Decision Problems for Interval Markov Chains. *Submitted for publication*, 2010.
- B. Caillaud, B. Delahaye, K.G. Larsen, A. Legay, M.L. Pedersen, A. Wąsowski. Constraint Markov Chains. *Submitted for publication*, 2010.

Résumé de la thèse : Spécification Modulaire et Analyse Compositionnelle de Systèmes Stochastiques

0.1 Introduction

0.1.1 Contexte

De nombreux secteurs industriels dont le secteur des systèmes embarqués ont récemment connu de profonds changements dans leur organisation. Les secteurs de l'automobile et de l'aérospatiale en sont les principaux exemples. Dans le passé, ils étaient organisés autour de compagnies intégrées verticalement supportant toute la chaîne de développement, du design à l'implémentation. Aujourd'hui, les systèmes sont si gros et complexes qu'il est quasiment impossible qu'une même équipe ait le contrôle de toute la chaîne de design. Dans la pratique, les systèmes complexes résultent de l'assemblage de multiples composants, généralement développés par des équipes qui travaillent indépendamment les unes des autres, tout en étant d'accord sur les interfaces de chaque composant. Ces interfaces spécifient à la fois les comportements des composants et l'environnement dans lequel ils peuvent être utilisés. L'avantage principal de cette méthodologie est de n'imposer aucune contrainte sur la façon dont les composants sont implémentés.

> *Différents composants peuvent être implémentés par différentes équipes à la condition que chaque équipe respecte l'interface sur laquelle elles se sont toutes mises d'accord.*

Dans la pratique, les interfaces sont généralement décrites soit par des documents textuels, générés avec Word/Excel, soit dans des langages de modélisation tels qu'UML/XML. À l'inverse, afin de limiter au maximum les ambiguïtés, nous proposons de baser notre raisonnement sur des formalismes mathématiques. Le développement de formalismes mathématiques permettant de raisonner au niveau abstrait des interfaces dans le but de déduire des propriétés des systèmes globaux et de décrire ou (ré)utiliser des composants est un domaine de recherche particulièrement actif, appelé *raisonnement compositionnel* [77]. Selon le point de vue du "software engineering", nécessairement orienté vers l'implémentation, les propriétés nécessaires à une bonne théorie d'interface sont les suivantes.

Remarque 0.1. *Dans le reste du document, en fonction du contexte, plusieurs termes pourront être utilisés pour désigner les mêmes notions : spécification = interface; implémentation = composant.*

1. Il est nécessaire de pouvoir décider si une interface admet une implémentation (aussi appelée modèle). En particulier, il doit être possible de décider si les propriétés spécifiées par l'interface sont implémentables, et de générer une telle implémentation le cas échéant. Dans le cadre de notre théorie, une implémentation ne doit pas être vue comme un langage de programmation mais plutôt comme un objet mathématique représentant un ensemble de langages de programmation partageant un ensemble de propriétés. Le fait de pouvoir décider si un composant implémente une interface est d'une importance particulière et doit pouvoir être décidé à l'aide d'algorithmes efficaces.

   > *Si l'on considère qu'une spécification est une représentation mathématique d'une propriété devant être satisfaite, alors la satisfaction doit coïncider avec le principe de vérification par implémentation.*

2. Il est important de pouvoir remplacer un composant par un autre sans pour autant modifier le comportement du système global. Au niveau des interfaces, cela correspond au concept de *raffinement*. Le principe du raffinement est de permettre de remplacer, dans n'importe quel contexte, une interface par une interface plus détaillée. Le raffinement doit être garant de la substituabilité d'interface, i.e. que toute implémentation qui satisfait le raffinement satisfait aussi le raffiné. Dans le but de limiter la complexité du procédé de développement, il est important de pouvoir décider de l'existence d'une interface raffinant deux autres interfaces. C'est le principe du *raffinement partagé*. Dans de nombreux cas, on recherche la *plus grande borne inférieure*, i.e. le raffinement partagé qui serait raffiné par tous les autres raffinements partagés.

3. Les systèmes de grande taille sont développés de manière concurrente pour leurs différents *aspects* ou *points de vue* par différentes équipes utilisant différents outils et différentes méthodes. Par exemple, ces aspects incluent les aspects fonctionnels ou les aspects de sécurité. Chacun d'eux nécessite des méthodes et outils différents pour l'analyse et la conception. Pour autant, ils ne sont pas entièrement indépendants et il leur arrive d'interagir. Le problème de traiter des aspects multiples ou encore des points de vue multiples semble donc essentiel. Notamment, cela implique que plusieurs interfaces peuvent être associées à un même composant, en l'occurrence au moins une par point de vue. Ces interfaces doivent alors être traitées de manière conjonctive, et cette conjonction doit satisfaire la propriété suivante :

   > *Étant donnés deux points de vue, représentés par deux interfaces, toute implémentation satisfaisant la conjonction de ces interfaces doit satisfaire les deux points de vue.*

4. Une bonne théorie d'interface doit comprendre, en particulier, une opération de combinaison reflétant la notion standard d'interaction/composition entre systèmes. D'un point de vue pratique, l'existence d'un environnement dans lequel deux composants peuvent

interagir, d'un point de vue compositionnel, doit être décidable. Un autre objectif, certainement plus ambitieux, est de pouvoir synthétiser un tel environnement. Pour finir, la composition doit satisfaire la propriété suivante :

> *Étant donnés deux composants satisfaisant deux interfaces, la théorie doit assurer que la composition des deux composants satisfait la composition des interfaces correspondantes.*

5. Il doit être possible de vérifier si un système composé de plusieurs composants satisfait une propriété en raisonnant uniquement sur ses composants et en appliquant le raisonnement compositionnel.

Le développement de théories d'interface a été le sujet de nombreuses études. Aujourd'hui, les recherches dans ce domaine se concentrent sur deux modèles : (1) les *automates d'interface* [54], et (2) les *spécifications modales* [100]. Les automates d'interface sont basés sur des automates à entrées/sorties avec une sémantique de jeu. Ils traitent les systèmes ouverts, leur raffinement et composition, tout en mettant l'emphase sur la compatibilité d'interfaces. Les spécifications modales sont aussi expressives que le mu-calcul [64]. En ce sens, elles admettent une algèbre compositionnelle plus riche, comportant des opérateurs de conjonction, composition et même résiduation. Ces deux modèles sont aujourd'hui bien établis, et implémentés dans des outils [31, 102, 4, 57].

Dès lors que les systèmes contiennent des algorithmes aléatoires, des protocoles probabilistes ou encore dès lors qu'ils interagissent avec un environnement physique, des modèles stochastiques sont nécessaires pour les représenter et les étudier. Ce besoin est d'autant plus présent que des demandes de tolérance aux fautes, d'analyse quantitative pour connaître le nombre de fautes que les systèmes peuvent supporter, ou encore de mesure des délais pouvant apparaître se font de plus en plus importants. Comme le disent Henzinger et Sifakis [77], l'introduction des probabilités dans les théories de conception de systèmes permet de mesurer la dépendance de ces systèmes informatiques comme c'est effectué couramment dans d'autres sciences de l'ingénieur. Cette thèse sera donc consacrée à adapter les théories d'interface aux systèmes stochastiques.

0.1.2 Contributions et plan de la thèse

Cette thèse présente des contributions originales pour la conception et la vérification de systèmes mixant des aspects non-déterministes et stochastiques. Nos résultats peuvent être divisés en trois axes qui sont décrits ici.

Le premier axe d'étude concerne la généralisation des théories d'interface aux systèmes stochastiques. Dans la même lignée que les systèmes de transition modaux [100], les Chaînes de Markov à Intervalles (IMCs) généralisent les notions de modalités aux systèmes stochastiques. Les IMCs ont été introduites par Larsen et Jonsson [86] comme un formalisme de *spécification*, et donc une base pour une méthode de raffinement par étapes successives avec laquelle les spécifications initiales sont très abstraites et sous spécifiées, puis sont rendues de plus en plus précises jusqu'à être concrètes. Outre le fait qu'elles ont été introduites dans le but de servir à la spécification, les IMCs ont plutôt été utilisées dans un but d'*abstraction* dans le cadre du model-checking, où l'on abstrait les modèles concrets par des modèles moins précis, sur lesquels les propriétés sont plus faciles à prouver [43, 42, 61, 89].

De manière informelle, les IMCs étendent les Chaînes de Markov en étiquetant les transitions par des intervalles de probabilités autorisées à la place de valeurs concrètes. Les implémentations des IMCs sont des chaînes de Markov dont les distributions de probabilité correspondent aux contraintes introduites par les intervalles. Cette définition de satisfaction/ implémentation est similaire à la notion de simulation pour les automates. Les IMCs représentent un modèle efficace sur lequel le raffinement et la composition peuvent être effectués grâce à des algorithmes efficaces relevant de l'algèbre linéaire. Malheureusement, comme nous allons le voir, l'expressivité des IMCs n'est pas suffisante pour permettre de traiter à la fois la composition logique et la composition structurelle.

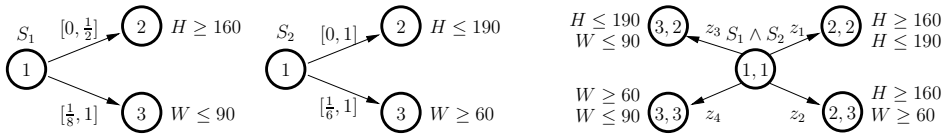


Figure 1: IMCs S_1 et S_2 illustrant la non-clôture par conjonction.

Soient les deux IMCs S_1 et S_2 données dans la figure 1.1. Elles spécifient différentes contraintes de probabilité relatives à la taille H et au poids W d'une personne lambda. Lorsque l'on tente d'exprimer la conjonction $S_1 \wedge S_2$ en tant qu'une IMC en effectuant simplement l'intersection des bornes des intervalles, nous obtenons le résultat suivant : $z_1 \leq \frac{1}{2}$, $\frac{1}{6} \leq z_2 \leq \frac{1}{2}$, $\frac{1}{8} \leq z_3$ et $\frac{1}{6} \leq z_4$. Pour autant, cette construction naïve n'est pas assez précise : la distribution $(z_1, z_2, z_3, z_4) = (\frac{1}{2}, \frac{1}{6}, \frac{1}{8}, \frac{5}{24})$ satisfait les contraintes, mais la probabilité résultante d'atteindre un état pour lequel $H \geq 160$, i.e. $z_1 + z_2 = \frac{2}{3}$, est en dehors des bornes spécifiées par S_1. Il nous faudrait donc pouvoir exprimer des dépendances entre les probabilités z_1, z_2, z_3 et z_4 en dehors du fait d'être une distribution de probabilité correcte ($z_1 + z_2 + z_3 + z_4 = 1$). La bonne combinaison conjonctive s'exprime avec les trois contraintes suivantes, qui ne pourraient être exprimées en utilisant des IMCs : $z_1 + z_2 \leq \frac{1}{2}$, $\frac{1}{8} \leq z_3 + z_4$, $\frac{1}{6} \leq z_2 + z_4$. Un exemple similaire montre que les IMCs ne sont pas non plus closes pour la composition parallèle. Pourtant, les IMCs sont largement acceptées par la communauté scientifique en tant que théorie de spécification pour les systèmes stochastiques [61, 89]. Il est donc intéressant d'étudier leurs propriétés ainsi que leurs limites.

Le chapitre 2, résumé en section 0.2, présente nos résultats concernant les IMCs et leur possible utilisation en tant que méthodologie de conception compositionnelle. En particulier, nous proposons une procédure polynomiale pour vérifier qu'une IMC est consistante (C), i.e. qu'elle admet au moins une implémentation. Nous proposons aussi une procédure exponentielle permettant de vérifier si k IMCs sont consistantes entre elles, i.e. si elles admettent une chaîne de Markov qui les satisfait toutes — une *implémentation commune* (CI). Nous prouvons de plus que ce problème (CI) est EXPTIME-complet. Lorsque k est constant, le problème devient polynomial. En particulier les deux problèmes consistant à vérifier si deux spécifications peuvent être satisfaites par une même implémentation et à synthétiser cette implémentation peuvent être résolus en temps polynomial. Dans [86], une relation de raffinement exhaustif (Thorough Refinement - TR) est définie comme l'inclusion des ensembles d'implémentations. Il est aussi défini une procédure pour vérifier TR. Nous prouvons dans ce chapitre que cette procédure peut être implémentée en temps simplement exponentiel, et montrons que TR est EXPTIME-complet. Pour finir, nous définissons des notions de déterminisme pour les IMCs et montrons que, pour les IMCs déterministes, le raffinement exhaustif coincide avec deux no-

tions de raffinement syntactique : le raffinement faible et le raffinement fort. Il existe, pour ces notions syntactiques, des algorithmes co-inductifs terminant en un nombre polynomial d'itérations. La théorie des MTS supportant le raffinement, la conjonction et la composition parallèle, les questions étudiées ici l'avaient déjà été dans le cadre des MTSs. En l'occurrence, il avait été prouvé que les deux problèmes de CI et TR correspondants étaient eux aussi EXPTIME-complets [10, 17]. Il est aussi prouvé, dans [86], que le formalisme des IMCs contient celui des MTSs, éclairant nos résultats sous une lumière surprenante : d'un point de vue complexité théorique, et considérant les problèmes de TR et CI, il semblerait que la généralisation de MTSs aux IMCs soit gratuite. Malheureusement, comme il a déjà été dit, les IMCs ne sont pas suffisamment expressives pour permettre de parler de conjonction, composition parallèle ou encore de disjonction. Il est donc nécessaire d'enrichir ce modèle pour obtenir une théorie de spécification qui soit fermée à la fois pour la conjonction et la composition parallèle.

Dans le chapitre 3, résumé en section 0.3, nous présentons un nouveau formalisme pour la spécification, basé sur une extension des IMCs : les Chaînes de Markov à Contraintes (CMC). Les CMCs autorisent des contraintes complexes sur les probabilités de transition, contraintes potentiellement plus expressives que les intervalles des IMCs. En ce sens, les CMCs généralisent le formalisme moins expressif des IMCs. Dans ce chapitre, nous prouvons que de simples contraintes linéaires suffisent à obtenir une clôture par conjonction, et que des contraintes polynomiales suffisent à obtenir la clôture par composition parallèle. Le formalisme des CMCs est la première théorie de spécification pour chaînes de Markov offrant de telles notions de clôture. Tout comme pour les IMCs, nous définissons des notions de déterminisme, et prouvons que, sur l'ensemble des CMCs déterministes, le raffinement exhaustif peut être approché de manière sûre par le raffinement faible et le raffinement fort. Pour finir, nous proposons des réductions de l'ensemble des automates probabilistes à l'ensemble des CMCs. Ces réductions prouvent que notre nouveau formalisme, les CMCs, est général. Pour autant, nous prouvons que toutes les opérations et toutes les relations sur les CMCs sont calculables.

Bien que les CMCs relèvent d'un formalisme général, la notion de satisfaction associée est basée sur le principe de vérification par implémentation [74, 105]. Il est pourtant parfois nécessaire de décrire des propriétés en utilisant des formules logiques. Pour cela, il faut considérer une nouvelle notion de satisfaction permettant de prendre en compte ces formules. Par exemple, considérons la notion de disponibilité. Cette notion permet de représenter une mesure du temps durant lequel un système satisfait une propriété donnée, une mesure importante lorsque l'on conçoit des systèmes critiques. Dans le but de permettre de spécifier de telles notions, nous avons développé une autre théorie de spécification offrant une relation de satisfaction plus expressive.

Dans le chapitre 4, résumé dans la section 0.4, nous présentons notre troisième contribution : une théorie de spécification qui étend les contrats hypothèse/garantie, introduits dans [21]. Le paradigme d'hypothèse/garantie a été introduit pour la première fois par Abadi et Lamport dans [3] comme un formalisme de spécification. L'avantage des contrats hypothèse/garantie tels qu'ils sont présentés dans [21] est qu'ils sont plus généraux que la notion classique d'automates d'interface. Dans ce chapitre, nous développons une théorie compositionnelle à base de contrats pour, d'un côté, des systèmes non-stochastiques, et, d'un autre côté, des systèmes stochastiques. Nous associons à ce formalisme deux notions de satisfaction *quantitatives* : la fiabilité et la disponibilité. De plus, nous proposons des définitions mathématiques pour les notions de composition, conjonction et raffinement.

Nous établissons par la suite une théorie de vérification compositionnelle pour les opéra-

tions définies ci-dessus et les deux notions de satisfaction considérées. Une telle théorie permet de raisonner sur le système complet en considérant uniquement les composants individuellement. Selon le type de contrats considérés, la théorie diffère : par exemple, nous prouvons que si un système S_1 satisfait un contrat probabiliste à un niveau α, et si un système S_2 satisfait un contrat probabiliste C_2 à un niveau β, alors la composition de S_1 et S_2 satisfait la composition de C_1 et C_2 à un niveau d'au moins $\alpha + \beta - 1$. Notre théorie est très générale : les systèmes et les contrats sont représentés par des ensembles d'exécutions. Pour finir, nous proposons des représentations symboliques et effectives pour les contrats et les systèmes, basées sur des automates pour représenter les ensembles, potentiellement infinis, d'exécutions. Si l'on suppose que les hypothèses et les garanties sont représentées par des automates de Büchi, permettant de les spécifier en utilisant des logiques comme LTL [108] ou PSL [60], nous observons que l'on peut vérifier si un système (éventuellement stochastique) satisfait une propriété de fiabilité en utilisant des techniques classiques, implémentées dans des outils comme SPIN [127] ou LIQUOR [35]. Nous prouvons que l'on peut vérifier la satisfaction de propriétés de disponibilité en utilisant une extension des résultats présentés dans [53]. Finalement, toutes les opérations présentées pour les contrats peuvent être facilement appliquées sur leurs représentations.

Les contributions présentées ci-dessus ont pour but la conception de systèmes, et, en partie, la vérification et la conception incrémentales. Cependant, il arrive que l'on doive vérifier des sous-systèmes implantés dans une architecture de grande taille. Dans ce cas, la difficulté est d'effectuer la vérification sans pour autant construire entièrement l'ensemble d'états correspondant à l'architecture totale.

Dans le chapitre 5, résumé en section 0.5, nous avons étudié ce problème au travers d'un exemple : un cas d'étude industriel appelé HCS (système de communication hétérogène), déployé pour assurer la communication en cabine dans un avion civil. La topologie de ce système est présentée dans la figure 2.

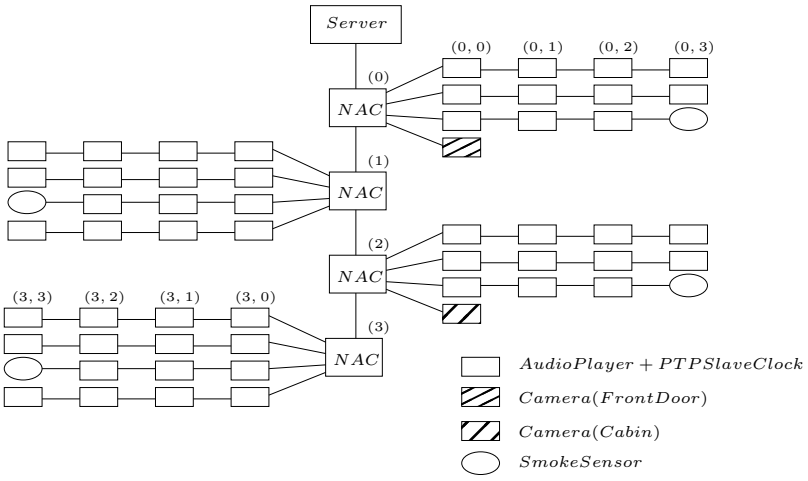


Figure 2: Modèle de l'exemple HCS. Les NACs effectuent l'ordonnancement des différents messages échangés entre le serveur et les différents dispositifs représentés par les rectangles.

Le HCS est un système hétérogène qui fournit aussi bien les services de divertissement (par exemple les services d'audio/vidéo à la demande des passagers) que les services critiques de sécurité (par exemple l'éclairage cabine, les annonces audio, les détecteurs de fumée), mis en

oeuvre dans des applications distribuées en parallèle, à travers différents dispositifs au sein de l'avion, et communiquant au travers d'un réseau partagé, basé sur le protocole Ethernet. Le système HCS doit satisfaire des exigences strictes telles que la fiabilité de la transmission des données, la tolérance aux fautes ainsi que des contraintes sur la synchronisation des différents périphériques. Nous avons étudié en détail une de ces propriétés : la précision de synchronisation des horloges internes entre périphériques.

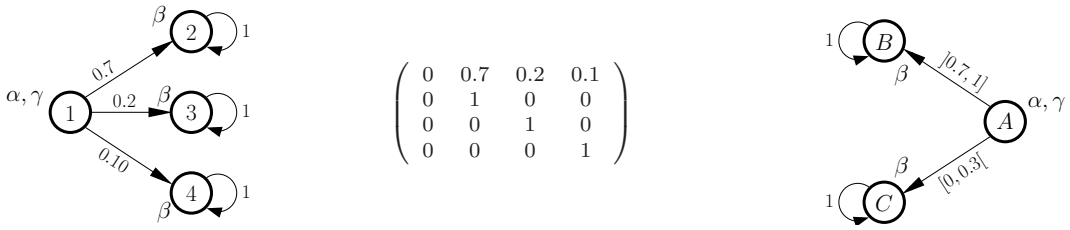
Une première solution aurait été d'explorer en détail l'espace d'états du système complet et de vérifier que la propriété est satisfaite pour toute paire de périphériques. Malheureusement, cette approche n'est pas réalisable à cause de la complexité trop importante du système global. La deuxième idée considérée a été d'appliquer les techniques de model-checking statistique. Encore une fois, la complexité trop importante du système global a fait que l'algorithme implémenté n'avait toujours pas terminé après trois jours d'exécution. Pour donner un ordre de grandeur de cette complexité, le système global est composé de plus de 280 périphériques travaillant en collaboration et échangeant de l'information. Une fois mise à plat, cette architecture génère un système comprenant plus de 2^{3000} états.

La solution que nous avons proposée est de construire une *abstraction stochastique* de l'environnement dans lequel les deux composants considérés sont implantés, i.e. une abstraction des comportements des autres composants du système et de leurs interactions avec le serveur et le périphérique choisi. Dans ce but, nous avons premièrement identifié les interactions entre deux périphériques quelconques et leur environnement, puis nous avons caractérisé ces interactions en introduisant des distributions de probabilité sur les comportements possibles de l'environnement. Ces distributions, qui remplacent l'environnement, et donc tous les autres composants du système, sont apprises en effectuant des simulations du système global. Ce que nous obtenons alors est un système stochastique d'une taille très inférieure à celle du système global, sur lequel les techniques comme le model-checking statistique sont applicables de manière efficace (sur ce système réduit, nos algorithmes terminent en moins d'une minute). En appliquant cette méthodologie, nous avons pu calculer des bornes sur la synchronisation et prouver que les exigences originellement demandées par EADS ne pouvaient physiquement pas être respectées. En plus d'améliorer l'efficacité, l'utilisation du model-checking statistique nous a aussi permis de vérifier des propriétés qui ne pouvaient être formalisées en utilisant les logiques temporelles classiques (par exemple, la gigue et la dérive d'horloges).

Dans [14], nous avons appliqué avec succès le concept d'abstraction stochastique dans le cas de la vérification de propriétés d'un système hétérogène basé sur l'*Avionics Full Duplex Switched Ethernet (AFDX)* [1], une technologie clé pour les systèmes embarqués dans les A380/A350, des avions de ligne. Les résultats que nous avons obtenus dans ce cadre sont plus précis que ceux de [32, 33, 118], obtenus par l'utilisation du model-checking temporel [5] ou encore du calcul réseau [47]. Nous présenterons un bref aperçu de ces travaux dans la thèse.

0.2 Chaînes de Markov à Intervalles

Comme présenté dans la section précédente, les IMCs ont été introduites pour la première fois par Larsen et Jonsson dans [86] en tant que formalisme de spécification pour les systèmes stochastiques, utilisant une sémantique de chaînes de Markov (voir figure 3 pour une illustration). Pourtant, les IMCs ont été principalement utilisées comme base pour des techniques de raffinement par étapes, par exemple pour le model-checking [43, 42, 61, 89].



(a) Une chaîne de Markov C

(b) La matrice de transition de C

(c) Une IMC I

Figure 3: Exemples de chaînes de Markov et de chaînes de Markov à intervalles.

En effet, les IMCs sont difficilement utilisables pour faire de la spécification compositionnelle car elles manquent d'opérations de composition générales. Dans [86], Jonsson et Larsen ont étudié en détail le raffinement pour ces modèles, mais ont laissé de côté les aspects compositionnels et les notions de complexité. Dans le chapitre 2, nos contributions principales sont les suivantes :

- Nous proposons dans un premier temps de répondre à un problème compositionnel — celui de *l'implémentation commune* — en utilisant les IMCs. Le problème de l'implémentation commune consiste en décider si un ensemble d'IMCs admet une chaîne de Markov les satisfaisant toutes. Dans un cadre compositionnel, ce problème est aisément résolu en construisant la conjonction de cet ensemble d'IMCs, puis en vérifiant si cette conjonction admet une implémentation. Malheureusement, les IMCs n'étant pas closes par conjonction, il est impossible d'appliquer cette méthode ici. Nous proposons donc une autre solution à ce problème, permettant de décider l'existence d'une implémentation commune, et, le cas échéant, d'en construire une. Nous prouvons que cette procédure est exponentielle et que le problème général (avec un ensemble d'IMCs de taille non bornée) est EXPTIME-complet.

- Dans le cas particulier d'un ensemble d'IMCs de taille bornée, nous montrons que le problème de l'implémentation commune est polynomial. Notamment, nous montrons que dans le cas de deux IMCs, il est possible de décider si elles admettent une implémentation commune, et, le cas échéant, d'en construire une, en un temps polynomial.

- Nous proposons aussi une procédure polynomiale permettant de décider si une IMC est consistante, i.e. si elle admet au moins une chaîne de Markov la satisfaisant.

- Par la suite, nous établissons que la procédure permettant de vérifier le raffinement exhaustif, présentée dans [86], peut être implémentée en temps simplement exponentiel, et prouvons de même que le problème général du raffinement exhaustif est EXPTIME-complet.

- Pour finir, nous définissons une notion de déterminisme pour les IMCs, et prouvons que, sur l'ensemble des IMCs déterministes, la notion sémantique de raffinement exhaustif coïncide avec les notions syntactiques de raffinement faible et raffinement fort, pour lesquelles il existe des algorithmes co-inductifs terminant en un nombre polynomial d'itérations.

Nos résultats sont d'un intérêt particulier, les IMCs et les notions de raffinement présentées ici étant couramment utilisées dans les travaux récents [86, 89, 61]. De plus, les bornes de complexité que nous proposons permettent de répondre à des problèmes restés 20 ans sans solution. Ces résultats sont robustes quant au formalisme utilisé pour représenter les IMCs. Par exemple, nous considérons que les états des IMCs sont étiquetés par un ensemble de propositions atomiques, mais nos résultats s'étendent aisément à des ensembles d'ensembles de propositions atomiques. De la même manière, si nos IMCs ont un unique état initial, les résultats sont aisément transférés au cas des IMCs avec une distribution de probabilité sur un ensemble d'états initiaux.

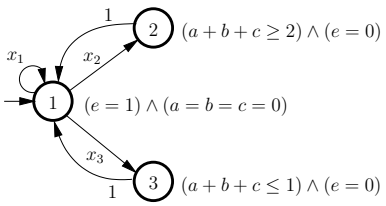
Finalement, bien que nous proposions une solution à un problème compositionnel, l'implémentation commune, il reste vrai que les IMCs ne permettent pas de répondre à de nombreux autres problèmes de la sorte. Nous proposons donc d'étendre les IMCs en un nouveau formalisme dont le but sera d'être pleinement compositionnel. Ce sera le sujet de la prochaine section.

0.3 Chaînes de Markov à Contraintes

Dans le chapitre 3, nous proposons une nouvelle approche pour le développement d'une théorie compositionnelle de spécification de systèmes stochastiques. Les *chaînes de Markov à contraintes* (CMCs) sont un tel formalisme, pouvant être utilisé pour faire de la conception à base de composants pour des systèmes stochastiques. Les CMCs sont une extension des IMCs permettant de spécifier des contraintes riches sur les probabilités de transitions plutôt que de simples intervalles. Nous montrons que des contraintes linéaires suffisent pour obtenir une clôture par conjonction, et que des contraintes polynomiales permettent d'obtenir la clôture par composition parallèle. Nous définissons des notions de raffinement, de consistance, de composition structurelle et de composition logique sur les CMCs, tous les ingrédients essentiels pour obtenir une théorie de conception compositionnelle. Des exemples de chaînes de Markov et de CMCs sont présentés dans la figure 4. Les notions de satisfaction, de raffinement faible et de raffinement fort sont des extensions conservatrices des notions similaires sur les IMCs. En plus de la définition de ce nouveau formalisme, nos contributions sont les suivantes :

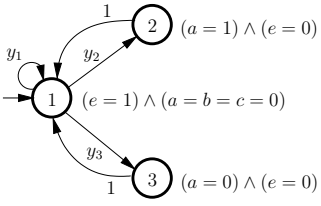
- Nous caractérisons les différentes relations de raffinements faible et fort en terme d'inclusion d'ensembles d'implémentations. En particulier, nous définissons une notion de déterminisme et prouvons que, sur l'ensemble des CMCs déterministes, ces relations coïncident avec l'inclusion des ensembles d'implémentations, aussi appelé le raffinement exhaustif ou sémantique. Enfin, nous proposons un algorithme permettant, à partir d'une CMC quelconque S, de générer une CMC déterministe contenant S.

- Nous proposons une notion de composition pour les CMCs, basée sur le principe de séparation des préoccupations. Selon ce principe, la composition parallèle des distributions de probabilité est effectuée séparément de la composition des ensembles de propositions atomiques. Cette séparation est présente dans tous les formalismes dont les automates probabilistes sont le modèle sémantique [121, 72, 87, 79]. Nous prouvons d'ailleurs que les automates probabilistes peuvent être représentés par des CMCs, et montrons comment la notion traditionnelle de composition parallèle sur ce modèle peut se traduire dans notre formalisme, en obtenant sans effort toutes les propriétés de précongruence.



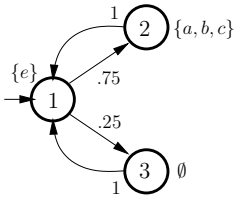
$$\varphi_1(1)(x) \equiv (x_1 = 0) \wedge (x_2 \geq 0.7) \wedge (x_2 + x_3 = 1)$$

(a) Un CMC S_1

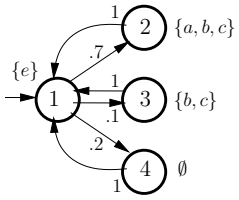


$$\varphi_2(1)(y) \equiv (y_1 = 0) \wedge (y_3 \geq 0.2) \wedge (y_2 + y_3 = 1)$$

(b) Un CMC S_2



(c) Une chaîne de Markov C_1 satisfaisant S_1 et S_2



(d) Une chaîne de Markov C_2 satisfaisant S_1 et S_2

Figure 4: Deux spécifications (CMCs) et deux implémentations (MC) d'un relais optique.

- Nous proposons une opération de conjonction et la comparons à l'opération de composition parallèle en terme d'expressivité. Nous prouvons que, lorsque l'on considère des ensembles de propositions atomiques indépendants, la composition parallèle est toujours un raffinement de la conjonction (le contraire étant faux). Cela permet de déduire un ensemble de CMCs à contraintes linéaires clos pour la conjonction ET la composition.
- Nous proposons une étude de la complexité des différentes relations et opérations pour l'ensemble des CMCs à contraintes polynomiales, une classe de CMCs close pour la conjonction et la composition parallèle en général.
- Finalement, nous montrons que les CMCs ne sont généralement pas closes par disjonction et étudions le problème de la décision de l'universalité d'une CMC.

Les CMCs représentent donc le premier formalisme compositionnel pour la spécification de systèmes stochastiques.

0.4 Contrats (Probabilistes)

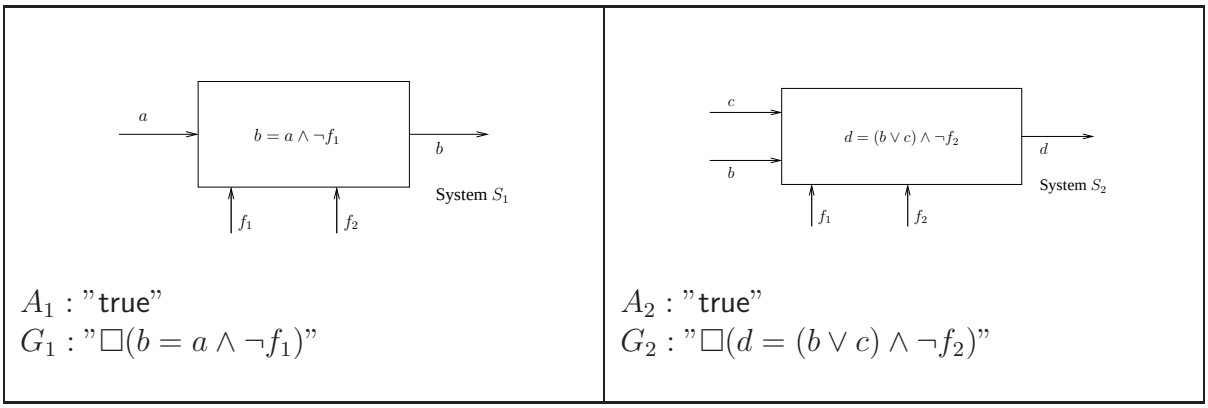
Dans [21], Benveniste et ses coauteurs ont proposé une théorie de conception basée sur les contrats hypothèse/garantie. Un tel contrat est une structure qui, contrairement aux automates d'interface [54, 52] et aux systèmes de transition modaux [100], autorise de séparer explicitement les hypothèse faites sur un composant (*les garanties*) des hypothèses faites sur son environnement (*les hypothèses*). Cette séparation explicite permet de définir une relation de satisfaction plus élaborée que celles définies dans le cadre des automates d'interface ou modaux. De plus, les auteurs de [21] utilisent pour représenter les hypothèses et les garanties une représentation permettant de s'abstraire de leur structure. De par ce fait, cette théorie permet de représenter des propriétés plus élaborées que les modèles graphiques classiques.

Dans le chapitre 4, nous développons une théorie de conception compositionnelle à base de contrats pour deux catégories de systèmes : nous présentons dans un premier temps une telle théorie pour des systèmes non-déterministes et non-stochastiques, puis nous étendons cette théorie à des systèmes non-déterministes et stochastiques. Comme dans le cadre de la vérification classique non-modulaire [37, 129], la relation de satisfaction que nous présentons est Booléenne pour les systèmes non-stochastiques et quantitative dans le cas contraire, ce qui nous pousse à développer deux notions de contrats. De plus, nous considérons deux notions de satisfaction différentes : la *fiabilité* et la *disponibilité*. La disponibilité représente une mesure du temps durant lequel un système satisfait une propriété donnée, pour toutes les exécutions possibles du système. La fiabilité, elle, exprime une mesure de l'ensemble des exécutions du système satisfaisant une propriété donnée. Ces deux quantités sont importantes pour la conception de systèmes critiques, par exemple. Les notions de satisfaction que nous introduisons sont dites dépendantes des hypothèses. En effet, nous considérons que les exécutions qui ne satisfont pas les hypothèses sont "correctes". Cette interprétation, suggérée par nos partenaires industriels, est nécessaire si l'on veut obtenir une théorie compositionnelle incluant, notamment, l'opérateur de conjonction. Dans ce chapitre, nos principales contributions sont les suivantes :

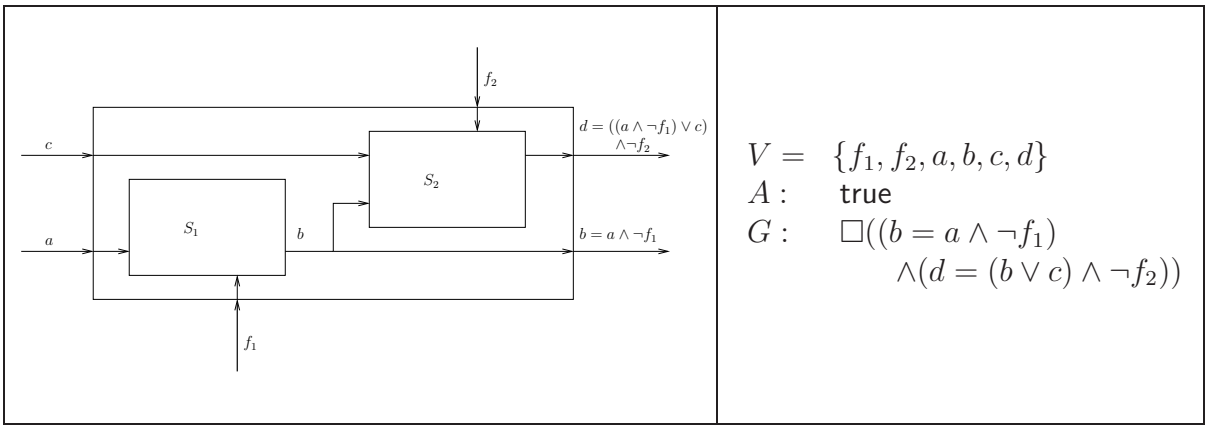
- Nous proposons des définitions mathématiques pour la composition parallèle, la conjonction et le raffinement, les trois opérations essentielles permettant de traduire la plupart des demandes des industriels (Notamment nos partenaires des projets Européens COMBEST [45] et SPEEDS [126]). La composition entre contrats, qui ressemble à la composition classique entre systèmes, consiste, informellement, à construire l'intersection des garanties et l'intersection des hypothèses. La conjonction, en revanche, construit un contrat dont les hypothèses sont l'union des hypothèses initiales, et les garanties les intersections des garanties initiales. Nous dirons qu'un contrat C_1 raffine un contrat C_2 si les garanties de C_1 contiennent celles de C_2 et si les hypothèses de C_2 contiennent celles de C_1. Cette définition booléenne n'est valable que pour les contrats non-probabilistes. Une définition quantitative sera proposée pour les contrats probabilistes.

- Nous établissons des propriétés de raisonnement compositionnel pour cette théorie, reliant les opérations présentées ci-dessus et les deux notions de satisfactions considérées. Ces propriétés permettent de raisonner sur un système composé en ne s'intéressant à ses composants qu'individuellement. Les résultats obtenus dépendent évidemment du type de contrat et du type de notion de satisfaction considérés. Par exemple, nous montrerons, dans le cadre des contrats non-probabilistes et pour la notion de fiabilité, que si un système S_1 satisfait un contrat C_1 et qu'un système S_2 satisfait un contrat C_2, alors la composition $S_1 \cap S_2$ des systèmes satisfait la composition $C_1 \parallel C_2$ des contrats. Dans le cadre des contrats probabilistes et toujours avec la notion de fiabilité, nous montrerons que si un système S_1 satisfait un contrat C_1 avec un niveau α, et qu'un système S_2 satisfait un contrat C_2 avec un niveau β, alors la composition des systèmes $S_1 \cap S_2$ satisfait la composition des contrats $C_1 \parallel C_2$ avec un niveau $\alpha + \beta - 1$. Un exemple est donné dans la figure 5.

- Nous proposons des représentations symboliques et effectives pour les contrats et les systèmes. Ces représentations sont basées sur des automates permettant de représenter des ensembles d'exécutions potentiellement infinis. Représenter les hypothèses et les



(a) Les systèmes S_1 et S_2 et les contrats probabilistes $\mathcal{C}_1$ et $\mathcal{C}_2$.



(b) Le système $S_1 \cap S_2$ et le contrat probabiliste $\mathcal{C}_1 \parallel \mathcal{C}_2$.

Figure 5: Un exemple de contrats et de propriété de fiabilité. Pour chaque contrat, la lettre V représente l'ensemble des variables, la lettre A représente les hypothèses et la lettre G représente les garanties.

garanties avec des automates de Büchi, permettant de spécifier des propriétés exprimables dans des logiques comme LTL [108] ou PSL [60], permettrait de vérifier si un système satisfait une propriété de fiabilité en utilisant des techniques classiques, déjà implémentées dans des outils tels que SPIN [127] ou encore LIQUOR [35]. Nous montrerons de plus que la satisfaction de propriétés de disponibilité peut s'effectuer en étendant les travaux présentés dans [53]. Pour finir, nous prouverons que toutes les opérations définies pour les contrats peuvent être aisément effectuées sur leur représentations symboliques.

0.5 Abstraction Stochastique et Model-Checking d'un Système Hétérogène de grande taille

Dans les chapitres précédents, nous nous sommes focalisés sur la conception de systèmes et la vérification incrémentale. Dans le chapitre 5, nous nous intéressons à la vérification d'applications évoluant à l'intérieur d'un système hétérogène. Les systèmes intégrant de multiples applications distribuées, communiquant au travers d'un réseau commun, sont rencontrés

fréquemment dans de nombreux domaines sensibles tels que l'avionique ou l'automobile. En général, la vérification d'applications particulières dans un tel cadre est une tâche ardue et souvent hors de portée des techniques de vérification exhaustive classiques. La principale difficulté de cette vérification provient des communications réseau qui permettent à toutes les applications d'interagir entre elles, et impliquent donc une exploration exhaustive de l'espace d'états du système complet.

Dans ce chapitre, nous proposons une solution à base de simulations, appelée model-checking statistique [78, 122, 136]. Contrairement aux méthodes classiques de vérification ou de tests exhaustifs, les méthodes à base de simulation ne donnent pas de résultat exact. Étant basées sur un nombre *fini* de simulations, elles permettent d'évaluer une notion quantitative de satisfaction d'une propriété donnée, tout en donnant des bornes sur la précision et sur la confiance que l'on peut avoir en le résultat. Malheureusement, la taille du système que nous considérons est telle que même ces méthodes basées sur un nombre fini de simulations ne sont pas applicables. Il est en effet impossible, en un temps raisonnable, de générer suffisamment de simulations pour donner des estimations avec une précision suffisante.

Nous proposons donc d'exploiter les connaissances de la structure du système complet pour augmenter l'efficacité de sa vérification. L'idée est simple : plutôt que d'effectuer la vérification sur le système complet, nous proposons d'analyser séparément chaque application dans un environnement, appelé *abstraction stochastique* du système, qui représente les interactions avec les autres parties du système. Cet environnement est généré en effectuant un nombre réduit de simulations du système complet, sur lesquelles nous mesurons les caractéristiques ayant un effet sur le comportement de l'application considérée dans le but de les remplacer par une distribution de probabilité.

Dans ce chapitre, nous appliquons cette méthode pour analyser le *système de communication hétérogène* (HCS) déployé pour assurer la communication réseau dans la cabine d'un avion de ligne. Une des propriétés critiques, que nous allons étudier dans ce chapitre, concerne la précision de la synchronisation des horloges des différents composants. Cette propriété, présente dans le cahier des charges du HCS, stipule que la différence entre les horloges locales de toute paire de périphériques doit être inférieure à une borne fixée. Il semble donc important de pouvoir générer la plus petite borne pour laquelle la propriété de synchronisation est satisfaite. Vu la complexité du système, il est clairement impossible d'obtenir manuellement une telle borne. Nous proposons donc de construire un modèle formel du HCS, puis de lui appliquer des algorithmes basés sur des simulations pour calculer cette borne. La méthode est la suivante : nous fixons une borne puis vérifions si la synchronisation est satisfaite. Selon le résultat, nous diminuons ou augmentons la borne jusqu'à trouver la plus petite borne pour laquelle la synchronisation est satisfaite. Pour que notre approche soit fonctionnelle, nous devons nous baser sur un outil qui soit capable de modéliser les systèmes hétérogènes ainsi que de simuler leurs exécutions et les interactions entre composants. Nous avons choisi d'utiliser BIP [15] (*Behaviour-Interaction-Priority*), un outil permettant de construire des systèmes à partir de composants atomiques communiquant au travers d'interactions. BIP offre aussi la possibilité de simuler les systèmes et permet, combiné avec des algorithmes de model-checking statistique, de vérifier des propriétés complexes. Les contributions du chapitre sont les suivantes :

- Le développement, à l'aide de BIP, d'un modèle complet du HCS sur lequel nous pourrons étudier les exigences d'EADS quant à ce système. Ce modèle est d'une taille importante : il comprend environ 300 composants atomiques et 245 horloges, ce qui correspond

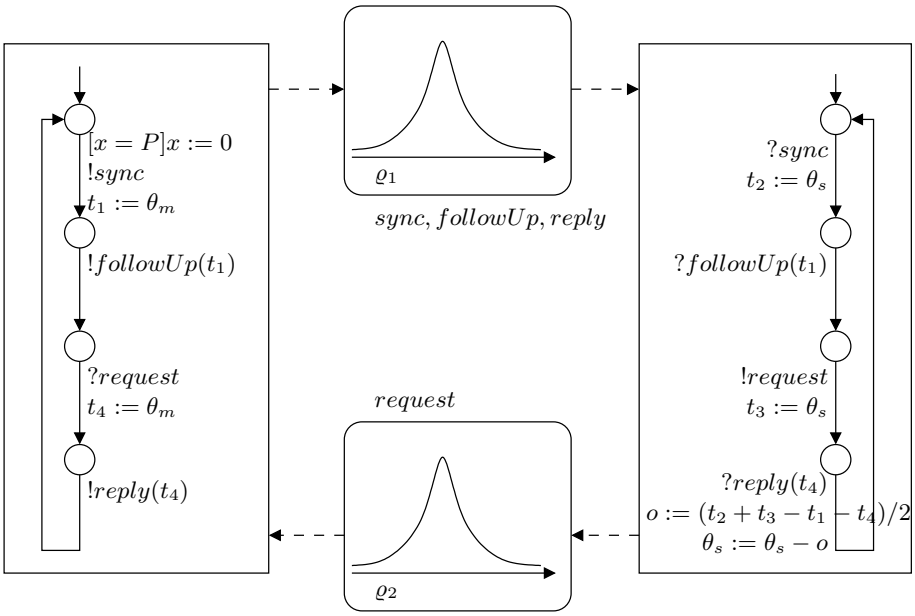


Figure 6: Abstraction stochastique du protocole PTP entre le serveur et un composant.

à 2468 lignes de code en BIP, soit 9018 lignes de code C générées automatiquement. Une fois linéarisé, ce système comporte environ 2^{3000} états.

- L'application de la méthode d'abstraction stochastique présentée ci-dessus au modèle BIP du HCS dans le but d'étudier avec précision la synchronisation des horloges des différents composants. Pour ce faire, nous avons simulé le modèle global et déduit, à partir de ces simulations, des distributions de probabilité représentant les délais associés à la transmission des messages du protocole de synchronisation, *Precision Time Protocol* (PTP) [2]. Ces distributions de probabilité permettent d'étudier en détail l'exécution de PTP entre le serveur central et les différents composants, un à un. Le principe est illustré dans la figure 6.

- La mesure, à partir du modèle réduit, de bornes précises quant à la synchronisation des horloges, pour chaque composant du système. Nous avons de plus mesuré, pour différentes bornes, la probabilité avec laquelle la synchronisation est assurée. Les exigences d'EADS quant à la synchronisation n'étant pas satisfaites, ces informations quantitatives ont été appréciées. Dans ce sens, nous avons proposé d'autres informations quantitatives : le nombre moyen, par exécution, d'erreurs de synchronisation. Finalement, nous avons étudié l'influence de la dérive d'horloges sur la synchronisation.

Chapter 1

Introduction

Context

Several industrial sectors involving complex embedded systems have recently experienced deep changes in their organization, aerospace and automotive being the most prominent examples. In the past, they were organized around vertically integrated companies, supporting in-house design activities from specification to implementation.

Nowadays, systems are tremendously big and complex, and it is almost impossible for one single team to have the complete control of the entire chain of design from the specification to the implementation. In fact, complex systems now result from the assembling of several components. These many components are in general designed by teams, working *independently* but with a common agreement on what the interface of each component should be. Such an interface precises the behaviors expected from the component as well as the environment in where it can be used. The main advantage is that it does not impose any constraint on the way the component is implemented:

> *Several components can be implemented by different teams of engineers providing that those teams respect the interfaces on which all of them agree.*

According to state of practice, interfaces are typically described using Word/Excel text documents or modeling languages such as UML/XML. We instead recommend relying most possibly on mathematically sound formalisms, thus best reducing ambiguities. Mathematical foundations that allow to reason at the abstract level of interfaces, in order to infer properties of the global implementation, and to design or to advisedly (re)use components is a very active research area, known as *compositional reasoning* [77]. Aiming at practical applications *in fine*, the software engineering point of view naturally leads to the following requirements for a good theory of interfaces.

Remark 1.1. *In the rest of the thesis, we will use the following equivalences (depending on the context): specification = interface; implementation = component.*

1. It should be decidable whether an interface admits an implementation (a model). This means that one should be able to decide whether the requirements stated by the interface can be implemented. One should also be capable of synthesizing an implementation for

such an interface. In our theory, an implementation shall not be viewed as a programming language but rather as a mathematical object that represents a set of programming languages sharing common properties. The ability to decide whether a given component implements a given interface is of clear importance, and this must be performed with efficient algorithms.

> *If one assumes that the specification is a mathematical representation of a property that should be satisfied, then satisfaction coincides with the so-called implementation verification principle.*

2. It is important to be able to replace a component by another one without modifying the behaviors of the whole design. At the level of interfaces, this corresponds to the concept of *Refinement*. Refinement allows replacing, in any context, an interface by a more detailed version of it. Refinement should entail substituability of interface implementations, meaning that every implementation satisfying a refinement also satisfies the larger interface. For the sake of controlling design complexity, it is desirable to be able to decide whether there exists an interface that refines two different interfaces. This is called *shared refinement*. In many situations, we are looking for the *greatest lower bound*, i.e., the shared refinement that could be refined by any other shared refinement.

3. Large systems are concurrently developed for their different *aspects* or *viewpoints* by different teams using different frameworks and tools. Examples of such aspects include the functional aspect and the safety aspect. Each of these aspects requires specific frameworks and tools for their analysis and design. Yet, they are not totally independent but rather interact. The issue of dealing with multiple aspects or multiple viewpoints is thus essential. This implies that several interfaces are associated with a given component, namely (at least) one per viewpoint. These interfaces are to be interpreted in a conjunctive way. This conjunction operation should satisfy the following property:

   > *Given two view-points represented by two interfaces, any implementation that satisfies the conjunction must satisfy the two view-points.*

4. The interface theory should also provide a combination operation, which reflects the standard interaction/composition between systems. In practice, one should be capable of deciding whether there exists at least one environment in where two components can work together, i.e., in where the composition makes sense. Another, but more difficult, objective is to synthesize such an environment. Finally, the composition operation should satisfy the following property:

   > *Given two components satisfying two interfaces, the theory must ensure that the composition of the two components satisfies the composition of their corresponding interfaces.*

5. *A verification procedure.* One should be capable of verifying whether a system composed of several components satisfies a property, simply by inspecting the various components and exploiting the compositional reasoning methodology.

Building good interface theories has been the subject of intensive studies. Nowadays, researchers concentrate on two models: (1) *interface automata* [54] and (2) *modal specifications* [100]. Interface automata is a game-based variation of input/output automata which deals with open systems, their refinement and composition, and puts the emphasis on interface compatibility. Modal specifications is a language theoretic account of a fragment of the modal mu-calculus logic [64] which admits a richer composition algebra with product, conjunction and even residuation operators. Both models are now well established and implemented in tools [31, 102, 4, 57].

As soon as systems include randomized algorithms, probabilistic protocols, or interact with physical environment, probabilistic models are required to reason about them. This is exacerbated by requirements for fault tolerance, when systems need to be analyzed quantitatively for the amount of failure they can tolerate, or for the delays that may appear. As Henzinger and Sifakis [77] point out, introducing probabilities into design theories allows assessing dependability of IT systems in the same manner as commonly practiced in other engineering disciplines. Lifting interface theory to stochastic systems will be the core subject of this thesis.

Contributions

This thesis presents new contributions in designing and verifying systems mixing both nondeterministic and stochastic aspects. Our results can be divided into three main contributions that are described hereafter.

We start our study by trying to generalize interface theories to the stochastic setting. Generalizing the notion of Modal Transition Systems [100] to the non-functional analysis of probabilistic systems, the formalism of Interval Markov Chains (IMCs) was introduced by Larsen and Jonsson [86] as a *specification* formalism, so a basis for a stepwise-refinement like modeling method, where initial designs are very abstract and underspecified, and then they are made continuously more precise, until they are concrete. Despite them being introduced with specification in mind, IMCs have not been used for this purpose intensively. Instead more commonly they served a dual purpose of *abstraction* in model checking, where a concrete system is being abstracted by a less precise system in order to prove the properties more easily [43, 42, 61, 89]. Informally, IMCs extend Markov Chains by labeling transitions with *intervals* of allowed probabilities rather than concrete probability values. Implementations of IMCs are Markov Chains (MCs) whose probabilistic distributions match the constraints induced by the intervals. This definition of satisfaction is similar to the notion of simulation for automata. IMCs is known to be an efficient model on which refinement and composition can be performed with efficient algorithms from linear algebra. Unfortunately, as we shall now see, the expressive power of IMCs is inadequate to support both logical and structural composition.

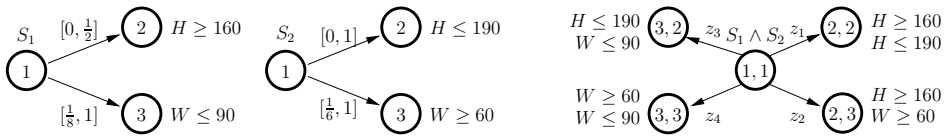


Figure 1.1: IMCs showing non-closure under conjunction

Consider two IMCs, S_1 and S_2, in Figure 1.1 specifying different probability constraints related to the height H and weight W of a given person. Attempting to express the logical

composition, also called conjunction, $S_1 \wedge S_2$ as an IMC by a simple intersection of bounds gives $z_1 \leq \frac{1}{2}$, $\frac{1}{6} \leq z_2 \leq \frac{1}{2}$, $\frac{1}{8} \leq z_3$ and $\frac{1}{6} \leq z_4$. However, this naive construction is too coarse: whereas $(z_1, z_2, z_3, z_4) = (\frac{1}{2}, \frac{1}{6}, \frac{1}{8}, \frac{5}{24})$ satisfies the constraints the resulting overall probability of reaching a state satisfying $H \geq 160$, i.e. $z_1 + z_2 = \frac{2}{3}$, violates the upper bound $\frac{1}{2}$ specified in S_1. What is needed is the ability to express dependencies between the probabilities z_1, z_2, z_3, z_4 besides that of being a probability distribution ($z_1 + z_2 + z_3 + z_4 = 1$). The correct conjunctive combination is expressed by three following constraints, exceeding the expressive power of IMCs: $z_1 + z_2 \leq \frac{1}{2}$, $\frac{1}{8} \leq z_3 + z_4$, $\frac{1}{6} \leq z_2 + z_4$. A similar example shows that IMCs are also not closed under parallel composition. Despite this fact, IMCs are widely accepted as a specification theory for stochastic systems [61, 89]. It is thus of interest to further study their properties and limits.

In Chapter 2, we aim at advancing our knowledge of IMCs and their use in a compositional design methodology. In particular, we propose a polynomial procedure for checking whether an IMC is *consistent* (C), i.e. it admits an implementation as a Markov Chain. We also contribute an exponential procedure for checking whether k IMCs are consistent in the sense that they share a Markov Chain satisfying all—a *common implementation* (CI). We show that this problem is EXPTIME-complete in general. As a special case we observe that CI is polynomial for any constant value of k. In particular checking whether two specifications can be simultaneously satisfied, and synthesizing their shared implementation can be done in polynomial time. In [86] a *thorough refinement* (TR) between IMCs is defined as an inclusion of implementation sets, and a procedure is given that establishes TR. Here we show that this procedure can be implemented in single exponential time, and having discussed the lower bound, show that TR is EXPTIME-complete. Finally, we define suitable notions of determinism for IMCs, and show that for deterministic IMCs thorough refinement coincides with two simulation-like preorders (the *weak refinement* and *strong refinement*). For these there exist natural co-inductive algorithms terminating in a polynomial number of iterations. The theory of MTS already supported refinement, conjunction and parallel composition. Within recent years, the same questions have been studied for MTSs, obtaining EXPTIME-completeness both for the corresponding notion of CI [10] and of TR [17]. In [86] it is shown that IMCs properly contain MTSs, which puts our new results in a somewhat surprising light: in the complexity theoretic sense, and as far as CI and TR are considered, the generalization from MTSs to IMCs does come for free. Unfortunately, as we already stated, IMCs are not expressive enough to capture many requirements of the compositional design methodology. This includes conjunction, parallel composition and disjunction. As a consequence, it is necessary to enrich the model of IMCs in order to obtain a specification theory that will be closed under both conjunction and parallel composition.

In chapter 3, we propose a new specification formalism, based on an extension of IMCs. Constraint Markov Chains permit rich constraints on probability distributions and thus generalize prior abstractions such as IMCs. We show that linear constraints suffice for closure under conjunction, while polynomial constraints suffice for closure under parallel composition. This is the first specification theory for MCs with such closure properties. Like for IMCs, we define suitable notions of determinism and show that, for deterministic CMCs, thorough refinement also coincides with the weak and strong refinements. Finally, we provide reductions from probabilistic automata to CMCs, showing that our formalism is fully general. Despite this generality, all operators and relations are computable.

Although CMCs are very general, their notion of satisfaction is based on the implementation

verification principle [74, 105]. However, it is sometimes necessary to describe properties by logical formulas. This requires a new definition of satisfaction relation. As an example consider availability, that represents a measure of the time during which a system satisfies a given property. This notion may play an important role when designing critical systems, but cannot be expressed using CMCs. This motivates the development of another specification theory with a richer satisfaction relation.

Our third contribution, presented in Chapter 4, extends the notion of assume-guarantee contracts introduced in [21]. The assume-guarantee paradigm was first proposed by Abadi and Lamport in [3] as a specification formalism. The advantage of assume-guarantee contracts, as presented in [21], is that they are more general than the classical notion of interface automata. In Chapter 4, we develop a contract-based compositional theory for both non-stochastic and stochastic systems. In this formalism, we propose two quantitative notions of satisfaction, namely reliability and availability. Moreover, we propose mathematical definitions for composition, conjunction and refinement. We then establish a *compositional reasoning verification* theory for those operations and the two notions of satisfiability we consider. This methodology allows to reason on the entire design by only looking at individual components. The theory differs with the type of contracts under consideration. As an example, we will show that if a system S_1 satisfies a probabilistic contract C_1 with probability α and a system S_2 satisfies a probabilistic contract C_2 with probability β, then their composition satisfies the composition of C_1 and C_2 with probability at least $\alpha + \beta - 1$. The theory is fully general as it assumes that both systems and contracts are represented by sets of runs. Finally, we propose effective and symbolic representations for contracts and systems. Those representations rely on an automata-based representation of possibly infinite sets of runs. Assuming that assumptions and guarantees are represented with Büchi automata (which allows to specify assumptions and guarantees with logics such as LTL [108] or PSL [60]), we observe that checking if a (stochastic) system satisfies a reliability property can be done with classical techniques implemented in tools such as SPIN [127] or LIQUOR [35]. We show that satisfaction of availability properties can be checked with an extension of the work presented in [53]. Finally, we also show that operations between and on contracts can easily be performed on the automata-based representations.

The above contributions focus on system design and, partially, on incremental design and verification. It is however sometimes required to verify subsystems within a huge architecture. The difficulty is to conduct this verification without considering the full state-space.

In Chapter 5, we begin the study of this problem through an experiment. We consider an industrial case study that is the *heterogeneous communication system* (HCS for short) deployed for cabin communication in a civil airplane. See Figure 1.2 for a topological view of the corresponding system architecture. HCS is an heterogeneous system providing entertainment services (ex: audio/video on passengers demand) as well as administrative safety critical services (ex: cabin illumination, control, audio announcements), which are implemented as distributed applications running in parallel, across various devices within the plane and communicating through a common Ethernet-based network, see Figure 1.2 for an illustration. The HCS system has to guarantee stringent requirements, such as reliable data transmission, fault tolerance, timing and synchronization constraints. An important requirement that we address is the *accuracy of clock synchronization* between any two devices.

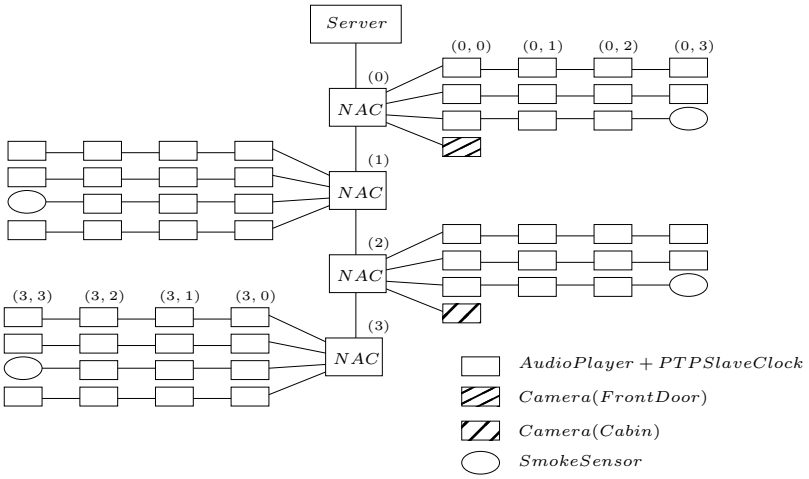


Figure 1.2: HCS example model. NACs perform scheduling of the different messages exchanged by the server and the devices shown as the many boxes on the diagram.

A first solution would have been to explore the entire state-space of the overall system and verify the property on each pair of devices (as an example, between Server and Device $(0, 3)$ given in Figure 1.2). Unfortunately, this approach is intractable due to the high complexity of the design. Our second idea was to apply statistical model checking. Unfortunately, due to the complexity of the design, computing simulation was too time consuming and the algorithm did not terminate in a reasonable time. To give the reader an intuitive idea about this complexity, we mention that the system is constituted of more than 280 components working in collaboration and exchanging information. This complex architecture gives rise to a flat system with more than 2^{3000} states.

The solution we promoted was to build a *stochastic abstraction* of the environment where the two components are working, i.e., an abstraction of the behaviors of the other components and their interactions with the master and the chosen device. For doing so, we first identified the interactions between the two devices and their environment. We then characterized these interactions by introducing probability distributions on the behaviors of the components. Those distributions, which replace the environment (and hence the other components) were learnt by conducting simulations on the entire system. The result we obtain is a smaller stochastic system on which techniques such as statistical model checking can be applied in an efficient manner (there the algorithm terminates in less than a minute). By applying this methodology, we have been capable of deriving bounds on the synchronization and showing that the original requirements made by EADS were simply falsified by their design. In addition to improving the efficiency, the use of statistical model checking also made it possible to verify properties that could not be formalized with classical temporal logics (example: clock drift and jitter).

In [14], we have successfully applied the stochastic abstraction concept to verify properties of an *Avionics Full Duplex Switched Ethernet (AFDX)* [1] heterogeneous system, a key technology in the computer system of A380/A350 aircrafts. The results we obtained are more accurate than those of [32, 33, 118], which were obtained by using timed model checking [5] or network calculus [47]. We will give a brief overview of this work in the thesis.

Chapter 2

Interval Markov Chains

2.1 Introduction

Interval Markov Chains (IMCs for short) were first introduced by Larsen and Jonsson in [86] as a specification formalism for probabilistic systems. However, they have mainly been used as a basis for stepwise-refinement, for example in model-checking [43, 42, 61, 89].

Indeed, IMCs are difficult to use for compositional specification due to lack of basic modeling operators. In [86] Jonsson and Larsen have investigated refinement of such processes in detail, but have left out the compositional aspects unexplored as well as the complexity aspects.

Consider the issue of combining multiple specifications of the same system. As we already observed, it turns out that conjunction of IMCs cannot be expressed as an IMC itself. This is caused by relative lack of expressiveness of intervals. For instance, consider a simple specification of a user of coffee machine. Let the model prescribe that a typical user orders coffee with milk with probability $x \in [0, 0.5]$ and black coffee with probability $y \in [0.2, 0.7]$ (customers also buy tea with probability $t \in [0, 0.5]$). Now the vendor of the machine delivers another specification, which prescribes that the machine is serviceable only if coffee (white or black) is ordered with some probability $z \in [0.4, 0.8]$ from among other beverages, otherwise it will run out of coffee powder too frequently, or the powder becomes too old. A conjunction of these two models would describe users who have use patterns compatible with this particular machine. Such a conjunction effectively requires that all the interval constraints are satisfied and that $z = x + y$ holds. However the solution of this constraint is not described by an interval over x and y. This can be seen by pointing out an extrememal point, which is not a solution, while all its coordinates take part in some solution. Say $x = 0$ and $y = 0.2$ violates the interval for z, while for each of these two values it is possible to select another one in such a way that z's constraint is also held (for example $(x = 0, y = 0.4)$ and $(x = 0.2, y = 0.2)$). Thus the solution space is not an interval over x and y. It is worth mentioning that IMCs are also not closed under parallel composition, but this problem will be adressed in the next chapter.

This lack of closure properties for IMCs motivates us to address the problem of reasoning about conjunction, without constructing it — the so-called common implementation problem: Given a set of IMCs S, does there exist a Markov Chain satisfying all the IMCs in S ? In other words, is the conjunction of all the IMCs in S satisfiable ? In this chapter we aim at advancing our understanding of algorithms and complexities for consistency, common implementation, and refinement of IMCs, in order to enable compositional modeling. In particular, we contribute:

- A polynomial procedure for checking whether an IMC is *consistent* (C), i.e. it admits an implementation as a Markov Chain.

- An exponential procedure for checking whether k IMCs are consistent in the sense that they share a Markov Chain satisfying all—a *common implementation* (CI). We show that this problem is EXPTIME-complete.

- As a special case we observe that CI is polynomial for any constant value of k. In particular checking whether two specifications can be simultaneously satisfied, and synthesizing their shared implementation can be done in polynomial time.

- In [86] a *thorough refinement* (TR) between IMCs is defined as an inclusion of implementation sets, and a procedure is given that establishes TR. Here we show that this procedure can be implemented in single exponential time, and having discussed the lower bound, show that TR is EXPTIME-complete.

- We define suitable notions of determinism for IMCs, and show that for deterministic IMCs thorough refinement coincides with two simulation-like preorders (the *weak refinement* and *strong refinement*). For these there exist natural co-inductive algorithms terminating in a polynomial number of iterations.

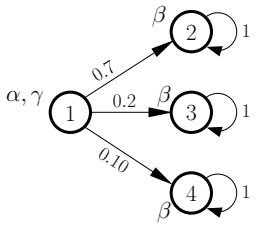
The theory of Modal Transition Systems (MTS), was introduced by Larsen in [100] as a specification formalism for discrete-time non-probabilistic systems. It supports refinement, conjunction and parallel composition. Within recent years, the same questions have been studied for MTSs, obtaining EXPTIME-completeness both for the corresponding notion of CI [10] and of TR [17]. In [86] it is shown that IMCs properly contain MTSs, which puts our new results in a somewhat surprising light: in the complexity theoretic sense, and as far as CI and TR are considered, the generalization from MTSs to IMCs does come for free.

The chapter proceeds as follows. In Section 2.2 we introduce the basic known definitions. In Section 2.3 we discuss deciding TR and other refinement procedures, expanding on the ramifications of determinism on refinements in Section 2.4. The problems of C and CI are addressed in Section 2.5. We close with discussing the results and a conclusion (Section 2.6).

2.2 Background

In this section, we introduce the basic definitions used throughout the chapter. In the following we will write $\mathsf{Intervals}_{[0,1]}$ for the set of all closed, half-open and open intervals included in $[0, 1]$.

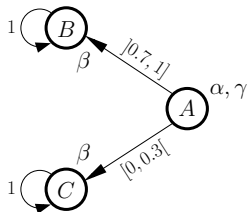
We begin with settling notation for Markov Chains. A Markov Chain (sometimes MC in short) is a tuple $C = \langle P, p_o, \pi, A, V_C \rangle$, where P is a set of states containing the initial state p_0, A is a set of atomic propositions, $V_C : P \to 2^A$ is a state valuation labeling states with propositions, and $\pi : P \to \mathsf{Distr}(P)$ is a probability distribution assignment such that $\sum_{p' \in P} \pi(p)(p') = 1$ for all $p \in P$. If the states of P are ordered, i.e. $P = \{p_1, \ldots, p_n\}$, then π can be seen as a matrix in $[0, 1]^{n \times n}$ such that the cell π_{kl} represents the probability of going from state p_k to state p_l. The probability distribution assignment is the only component that is relaxed in IMCs:



(a) A Markov Chain C

$$\begin{pmatrix} 0 & 0.7 & 0.2 & 0.1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(b) The transition matrix of the Markov Chain C



(c) An IMC I

Figure 2.1: Examples of Markov Chains and Interval Markov Chains.

Definition 2.1 (Interval Markov Chain)**.** *An* Interval Markov Chain *is a tuple* $I = \langle Q, q_o, \varphi, A, V_I \rangle$*, where* Q *is a finite set of states containing the initial state* q_o*,* A *is a set of atomic propositions,* $V_I : Q \to 2^A$ *is a state valuation, and* $\varphi : Q \to (Q \to \mathsf{Intervals}_{[0,1]})$*, which for each* $q \in Q$ *and* $q' \in Q$ *gives an interval of probabilities.*

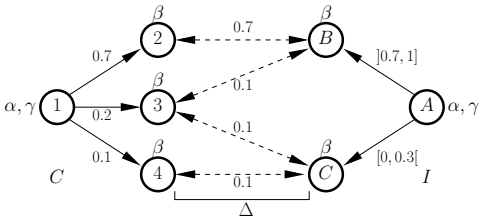
Instead of a distribution, as in MCs, in IMCs we have a function mapping elementary events (target states) to intervals of probabilities. We interpret this function as a constraint over distributions. This is expressed in our notation as follows. Given a state $q \in Q$ and a distribution $\sigma \in \mathsf{Distr}(Q)$, we say that $\sigma \in \varphi(q)$ iff $\sigma(q') \in \varphi(q)(q')$ for all $q' \in Q$. If the states of Q are ordered, i.e. $Q = \{q_1, \ldots, q_m\}$, a distribution $\sigma \in \mathsf{Distr}(Q)$ can be seen as a vector in $[0,1]^m$ such that the cell σ_k represents the probability of going to state q_k. We will say that the vector $\sigma \in [0,1]^m$ is in $\varphi(q)$ iff it defines a distribution, and this distribution is in $\varphi(q)$. Occasionally it is convenient to think about a Markov Chain as of an IMC, whose all probability intervals are closed point intervals.

We visualize IMCs as automata with intervals on transitions. As an example consider the IMC in Figure 2.1c. It has two outgoing transitions from the initial state A. No arc is drawn between two states if the probability is zero (or more precisely the interval is $[0,0]$). So in the example there is zero probability of going from state A to A, or from B to C, etc. Otherwise the probability distribution over successors of A is constrained to fall into $]0.7, 1]$ and $[0, 0.3]$ for B and C respectively. States B and C have valuation β, whereas state A has valuation α, γ. Also observe that Figure 2.1a presents a Markov Chain using the same convention, modulo the intervals. The corresponding transition matrix is given in Figure2.1b. Remark that our formalism does not allow "sink states", i.e. states with no outgoing transition. However, in order to avoid clutter in the figures, we sometimes represent states with no outgoing transitions. They must be interpreted as states with a self-loop of probability 1.

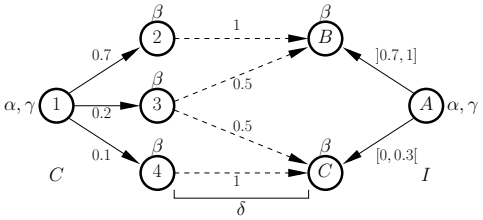
A *satisfaction relation* establishes compatibility of Markov Chains (implementations) and IMCs (specifications). The original definition, reported below, has been presented in [86, 87].

Definition 2.2 ((Direct) Satisfaction Relation)**.** *Let* $C = \langle P, p_o, \pi, A, V_C \rangle$ *be a MC and let* $I = \langle Q, q_o, \varphi, A, V_I \rangle$ *be an IMC. A relation* $\mathcal{R} \subseteq P \times Q$ *is called a satisfaction relation if whenever* $p \,\mathcal{R}\, q$ *then*

- *their valuation sets agree:*$V_C(p) = V_I(q)$ *and*
- *there exists a probability distribution* $\delta \in \mathsf{Distr}(P \times Q)$ *such that*
  1. $\sum_{q' \in Q} \delta(p', q') = \pi(p)(p')$ *for all* $p' \in P$*,*



(a) Illustration of a satisfaction relation between a MC C and an IMC I using a direct redistribution of the probability mass (Def 2.2).



$$\Delta = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}$$

(b) Illustration of a satisfaction relation between a MC C and an IMC I using a correspondence matrix (Def 2.3).

Figure 2.2: Illustration of satisfaction relations using direct and indirect redistribution of the probability mass.

2. $\sum_{p' \in P} \delta(p', q') \in \varphi(q)(q')$ *for all* $q' \in Q$*, and*
3. *if* $\delta(p', q') > 0$*, then* $p' \mathcal{R} q'$.

In our work, we use a slightly modified, but strictly equivalent definition using a concept of *correspondence matrix*. This notion of satisfaction is more intuitively linked to the notions of weak and strong refinement that will be presented later in this section. This definition requires that both the states of the MC and the states of the IMC are ordered. Figure 2.2 compares the two definitions using an example side by side.

Definition 2.3 (Satisfaction)**.** *Let* $C = \langle P, p_o, \pi, A, V_C \rangle$ *be a MC with* $P = \{p_1, \ldots, p_n\}$ *and let* $I = \langle Q, q_o, \varphi, A, V_I \rangle$ *be an IMC with* $Q = \{q_1, \ldots, q_m\}$*. A relation* $\mathcal{R} \subseteq P \times Q$ *is called a satisfaction relation if whenever* $p_i \mathcal{R} q_r$ *then*

- *valuations of* p_i *and* q_r *agree:* $V_C(p_i) = V_I(q_r)$ *and*
- *there exists a correspondence matrix* $\Delta \in [0, 1]^{n \times m}$ *such that*
  1. *for all* $p_j \in P$ *such that* $\pi_{i,j} > 0$*, the row* Δ_j *defines a distribution on* Q*,*
  2. *the vector* $\pi_i \times \Delta$ *is in* $\varphi(q_r)$*, and*
  3. *if* $\Delta_{j,s} > 0$*, then* $p_j \mathcal{R} q_s$.

In the above definition, $\pi_i \times \Delta$ represents the classical matrix product between the vector $\pi_i \in [0, 1]^n$ and the matrix $\Delta \in [0, 1]^{n \times m}$. Formally, $\sigma = \pi_i \times \Delta$ is a vector in $[0, 1]^m$ such that $\sigma_s = \sum_{j=0}^{n} \pi_{i,j} \cdot \Delta_{j,s}$ for all $1 \leq s \leq m$.

We write $C \models I$ iff there exists a satisfaction relation containing (p_o, q_o). In this case, C is an *implementation* of I. The set of implementations of I is written $[\![I]\!]$. Figure 2.2b presents an example of satisfaction between states 1 and A. The correspondence matrix is visualized using labels on the dashed arrows i.e. the probability mass going from state 1 to 3 is distributed to state B and C with half going to each.

We will say that a state q of an IMC is *consistent*, if its interval constraint $\varphi(q)$ is satisfiable, i.e. there exists a distribution $\sigma \in \mathsf{Distr}(Q)$ satisfying $\varphi(q)$, so $\sigma \in \varphi(q)$. Obviously, for a given IMC, it is sufficient that all its states are consistent in order to guarantee that the IMC is consistent itself—there exists a Markov Chain satisfying it. We discuss the problem of establishing consistency in a sound and complete manner in Section 2.5.

There are three known ways of defining refinement for IMCs from literature: the strong refinement (introduced as *simulation* in [86]), weak refinement (introduced under the name of *probabilistic simulation* in [61]), and thorough refinement (introduced as *refinement* in [86]). We will recall their formal definitions:

Definition 2.4 (Strong Refinement)**.** *Let $I_1 = \langle Q, q_o, \varphi_1, A, V_1\rangle$ and $I_2 = \langle S, s_o, \varphi_2, A, V_2\rangle$ be two IMCs such that $Q = \{q_1, \ldots, q_n\}$ and $S = \{s_1, \ldots, s_m\}$. A relation $\mathcal{R} \subseteq Q \times S$ is called a* strong refinement relation *if whenever $q_i \,\mathcal{R}\, s_r$, we have that*

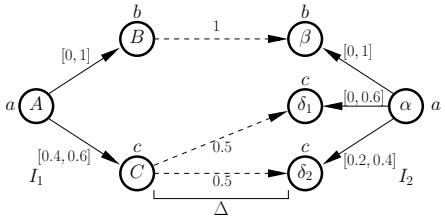
- *their valuation sets agree: $V_1(q_i) = V_2(s_r)$ and*
- *there exists a correspondence matrix $\Delta \in [0,1]^{n\times m}$ such that for any vector $\sigma \in [0,1]^n$, if $\sigma \in \varphi_1(q_i)$, then*
  1. *for each $q_j \in Q$ such that $\sigma_j > 0$, the row Δ_j defines a distribution on S,*
  2. *we have $\sigma \times \Delta \in \varphi_2(s_r)$, and*
  3. *for all $q_j \in Q$ and $s_t \in S$, if $\Delta_{j,t} > 0$, then $q_j \,\mathcal{R}\, s_t$.*

We say that I_1 strongly refines I_2, written $I_1 \leq_{\mathsf{S}} I_2$, iff there exists a strong refinement relation containing (q_o, s_o).

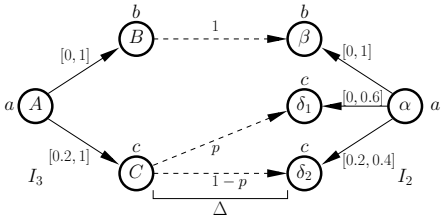
Intuitively the strong refinement between states of I_1 and states of I_2 requires the existence of a single correspondence, which witnesses satisfaction for any resolution of probability constraint over successors in I_1. Figure 2.3a illustrates such a correspondence between states A and α of two IMCs. The non-zero coefficients of the correspondence matrix are given by labels on the dashed lines. It is easy to see that regardless of how the probability constraints are resolved the correspondence matrix distributes the probability mass in a fashion satisfying I_2.

Contrasting with strong refinement, the weak refinement between states of I_1 and I_2 requires that for any resolution of probability constraint over successors in I_1 there exists a correspondence, which witnesses satisfaction of I_2. Thus the weak refinement achieves the weakening by swapping the order of quantifications. Figure 2.3b illustrates such a correspondence between states A and α of another two IMCs. Here x stands for a value in $[0.2, 1]$ (arbitrary choice of probability of going to state C). Notably, for each choice of x there exists $p \in [0,1]$ such that $p \cdot x \in [0, 0.6]$ and $(1-p)\cdot x \in [0.2, 0.4]$.

Definition 2.5 (Weak Refinement)**.** *Let $I_1 = \langle Q, q_o, \varphi_1, A, V_1\rangle$ and $I_2 = \langle S, s_o, \varphi_2, A, V_2\rangle$ be two IMCs such that $Q = \{q_1, \ldots, q_n\}$ and $S = \{s_1, \ldots, s_m\}$. A relation $\mathcal{R} \subseteq Q \times S$ is called a* weak refinement relation *if whenever $q_i \,\mathcal{R}\, s_r$, we have that*



(a) Illustration of a strong refinement relation between an IMC I_1 and an IMC I_2.



(b) Illustration of a weak refinement relation between an IMC I_3 and an IMC I_2; p is a parameter.

Figure 2.3: Illustration of strong and weak refinement relations.

- *their valuation sets agree:* $V_1(q_i) = V_2(s_r)$ *and*
- *for each* $\sigma \in [0,1]^n$ *such that* $\sigma \in \varphi_1(q_i)$*, there exists a correspondence matrix* $\Delta \in [0,1]^{n \times m}$ *such that*
  1. *for each* $q_j \in Q$ *such that* $\sigma_j > 0$*, the row* Δ_j *defines a distribution on* S*,*
  2. *we have* $\sigma \times \Delta \in \varphi_2(s_r)$*, and*
  3. *for all* $q_j \in Q$ *and* $s_t \in S$*, if* $\Delta_{j,t} > 0$*, then* $q_j \mathcal{R}\, s_t$*.*

We say that I_1 *weakly refines* I_2*, written* $I_1 \leq_{\mathsf{W}} I_2$*, iff there exists a weak refinement relation containing* (q_o, s_o)*.*

Finally, we introduce the thorough refinement as defined in [86]:

Definition 2.6 (Thorough Refinement)**.** *IMC* I_1 thoroughly refines *IMC* I_2*, written* $I_1 \leq_{\mathsf{T}} I_2$*, iff each implementation of* I_1 *implements* I_2*:* $[\![I_1]\!] \subseteq [\![I_2]\!]$

Thorough refinement is a semantic notion of refinement: an IMC I_1 thoroughly refines an IMC I_2 iff all the implementations of I_1 are implementations of I_2. In this way, it is the finest possible notion of refinement.

2.3 Refinement Relations

As said in the previous section, thorough refinement is the ultimate refinement relation for any specification formalism. In our formalism, both strong and weak refinements soundly approximate the thorough refinement. Indeed, since they are transitive and degrade to satisfaction if the left argument is a Markov Chain, whenever strong or weak refinement holds, thorough refinement also holds. However, the converse does not hold. Detailed proofs of these facts are given in a more general setting in the next chapter. We will now discuss procedures to compute weak and strong refinements, and then compare the granularity of these relations, which will lead us to procedures for computing thorough refinement.

2.3.1 Weak and Strong Refinement

Consider two IMCs $I_1 = \langle P, p_0, \varphi_1, A, V_1 \rangle$ and $I_2 = \langle Q, q_0, \varphi_2, A, V_2 \rangle$ with $P = \{p_0, \ldots, p_n\}$ and $Q = \{q_0, \ldots, q_m\}$. Informally, checking whether a given relation $\mathcal{R} \subseteq P \times Q$ is a weak refinement relation reduces to checking, for each pair $(p, q) \in \mathcal{R}$, whether the following formula is true: $\forall \pi \in \varphi_1(p),\ \exists \Delta \in [0,1]^{n \times m}$ such that $\pi \times \Delta$ satisfies a system of linear equations / inequations. Since the set of vector distributions satisfying $\varphi_1(p)$ is convex, checking such a system is exponential in the number of variables, here $|P| \cdot |Q|$. As a consequence, checking whether a relation on $P \times Q$ is a weak refinement relation is exponential in $|P| \cdot |Q|$. For strong refinement relations, the only difference appears in the formula that must be checked: $\exists \Delta \in [0,1]^{n \times m}$ such that $\forall \pi \in \varphi_1(p)$, we have that $\pi \times \Delta$ satisfies a system of linear equations / inequations. Therefore, checking whether a relation on $P \times Q$ is a strong refinement relation is also exponential in $|P| \cdot |Q|$.

Finally, deciding whether weak (strong) refinement holds between I_1 and I_2 can be done in the usual coinductive fashion by considering the total relation $P \times Q$ and successively removing all the pairs that do not satisfy the above formulae. The refinement holds iff the relation we reach contains the pair (p_0, q_0). The algorithm will terminate after at most $|P| \cdot |Q|$ iterations. This gives an upper bound on the complexity to check strong and weak refinements. To the best of our knowledge, the lower bound remains unknown.

2.3.2 Granularity

In [86] an informal statement is made, that the strong refinement is strictly stronger (finer) than the thorough refinement: $(\leq_{\mathsf{T}}) \supsetneq (\leq_{\mathsf{S}})$. In [61] the weak refinement is introduced, but without discussing its relations to neither strong nor thorough refinement. The following theorem resolves all open issues in relations between the three:

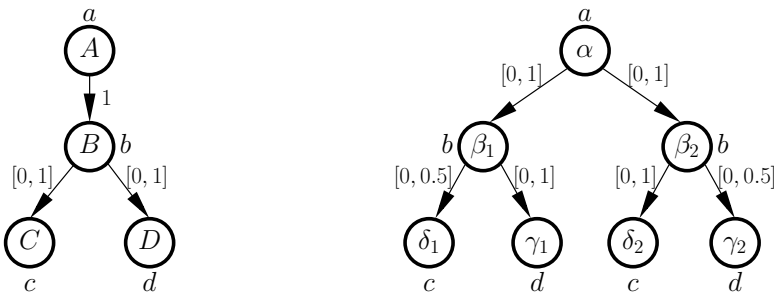
Theorem 2.1. *Thorough refinement is strictly weaker than weak refinement, which is strictly weaker than strong refinement :* $(\leq_{\mathsf{T}}) \supsetneq (\leq_{\mathsf{W}}) \supsetneq (\leq_{\mathsf{S}})$.

Proof. The first inequality is shown by exhibiting IMCs I_4 and I_5 such that I_4 thoroughly, but not weakly refines I_5: they are given in Figure 2.4.

Let M be an implementation of I_4 and $\mathcal{R}$ a corresponding satisfaction relation. Let P be the set of states of M implementing B. Each state $p \in P$ either satisfies β_1, β_2 or both. Call P_1 the set of states $p \in P$ such that p satisfies β_1 and P_2 the set of states $p \in P$ such that p satisfies β_2 and not β_1. We build a satisfaction relation $\mathcal{R}'$ such that, for all $q \in M$, if $q \,\mathcal{R}\, A$ then $q \,\mathcal{R}'\, \alpha$; if $q \in P_1$, then $q \,\mathcal{R}'\, \beta_1$; if $q \in P_2$, then $q \,\mathcal{R}'\, \beta_2$; if $q \,\mathcal{R}\, C$, then $q \,\mathcal{R}'\, \delta_1$ and $q \,\mathcal{R}'\, \delta_2$; and if $q \,\mathcal{R}\, D$ then $q \,\mathcal{R}'\, \gamma_1$ and $q \,\mathcal{R}'\, \gamma_2$. By construction, $\mathcal{R}'$ is a satisfaction relation, and M is an implementation of I_5. Thus, $[\![I_4]\!] \subseteq [\![I_5]\!]$.

However, it is impossible to define a weak refinement relation between I_4 and I_5 : obviously, B can neither refine β_1 nor β_2: Let σ be a vector distribution admitted in B giving probability 1 to state C. Because of the interval $[0, 0.5]$ on the transition from β_1 to δ_1, at least 0.5 must be assigned to γ_1, but C and γ_1 can not be related. A similar argument shows that B can not refine β_2.

The second inequality is shown by demonstrating two other IMCs, I_3 and I_2 such that I_3 weakly but not strongly refines I_2: they are given in Figure 2.3b.



(a) IMC I_4 (b) IMC I_5

Figure 2.4: IMCs I_4 and I_5 such that I_4 thoroughly but not weakly refines I_5

- State A weakly refines state α: Given a value x for the transition $A \to C$, we can split it in order to match both transitions $\alpha \xrightarrow{p \cdot x} \delta_1$ and $\alpha \xrightarrow{(1-p) \cdot x} \delta_2$. Define $\Delta \in [0,1]^{3 \times 4}$ as follows :

$$\Delta = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & p & (1-p) \end{pmatrix}$$

with

$$p = \begin{cases} 0 & \text{if } 0.2 \leq x \leq 0.4 \\ \frac{x-0.3}{x} & \text{if } 0.4 < x < 0.8 \\ 0.6 & \text{if } 0.8 \leq x \end{cases}$$

Δ is a correspondence matrix witnessing a weak refinement relation between A and α.

- However, one cannot find a coefficient p that would work for all x. It is thus impossible to build a strong refinement relation between I_3 and I_2.

□

2.3.3 Deciding Thorough Refinement

As weak and strong refinements are strictly stronger than thorough refinement, it is interesting to investigate complexity of deciding TR. In [86] a procedure computing TR is given, albeit without a complexity discussion. We close the problem of complexity class of TR as follows:

Theorem 2.2. *The decision problem TR of establishing whether there exists a thorough refinement between two given IMCs is EXPTIME-complete.*

The upper-bound is shown by observing that the algorithm presented in [86] runs in single exponential time. For the sake of completeness, and in order to clarify several typesetting inaccuracies of the original presentation, we report below the entire construction due to [86]. Then we analyze its complexity.

Definition 2.7 (Subset simulation)**.** *Let $I_1 = \langle P, p_0, \varphi_1, A, V_1 \rangle$ and $I_2 = \langle Q, q_0, \varphi_2, A, V_2 \rangle$ be IMCs with $P = \{p_0, \ldots, p_n\}$ and $Q = \{q_0, \ldots, q_m\}$. Let $2^Q = \{T_0, \ldots, T_{2^m}\}$. A total relation $\mathcal{R} \subseteq P \times 2^Q$ is a* subset-simulation *iff for each state $p \in P$:*

1. *$p \mathcal{R} T$ implies $V_1(p) = V_2(t)$ for all $t \in T$ and*

2. *for each vector distribution $\pi \in \varphi_1(p)$ and each correspondence matrix $\Delta^1 \in [0,1]^{n \times 2^m}$ such that* support$(\Delta^1) \subseteq \mathcal{R}$*, there exists a set T such that $p \mathcal{R} T$ and for each $t \in T$, there exists a vector distribution $\varrho \in \varphi_2(t)$ and a correspondence matrix $\Delta^2 \in [0,1]^{m \times 2^m}$ such that*

   (a) *if $\Delta^2_{i,j} > 0$ then $q_i \in T_j$, and*

   (b) *we have $\pi \times \Delta^1 = \varrho \times \Delta^2$.*

Intuitively, this relation associates to every state p of I_1 a sample of sets of states $(T_{i_1}, \ldots, T_{i_k})$ of I_2 that are "compatible" with p. Then, for each admissible redistribution Δ^1 of the successor states of p, it states that there exists one of the sets T_i such that for each of its states t', there is a redistribution Δ^2 of the successor states of t' that is compatible with Δ^1. In [86] it is shown that the existence of a subset-simulation between two IMCs I_1 and I_2 is equivalent to thorough refinement between them.

Example. *Consider the IMCs $I_4 = \langle \{A, B, C, D\}, A, \varphi_4, \{a, b, c, d\}, V_4 \rangle$ and $I_5 = \langle \{\alpha, \beta_1, \beta_2, \delta_1, \delta_2, \gamma_1, \gamma_2\}, \alpha, \varphi_5, \{a, b, c, d\}, V_5 \rangle$ given in Figure 2.4. They are such that I_4 thoroughly but not weakly refines I_5 (c.f. proof of Theorem 2.1). Since thorough refinement holds, we can exhibit a subset simulation $\mathcal{R} \subseteq P \times 2^Q$ between I_4 and I_5: Let $\mathcal{R} = \{(A, \{\alpha\}), (B, \{\beta_1\}), (B, \{\beta_2\}), (C, \{\delta_1, \delta_2\}), (D, \{\gamma_1, \gamma_2\})\}$. We illustrate the unfolding of $\mathcal{R}$ for states A and B of I_4. The rest is left to the reader.*

Consider state A of I_4.

1. *We have $A \mathcal{R} \{\alpha\}$, and $V_4(A) = a = V_5(\alpha)$.*

2. *The only distribution $\pi \in \varphi_4(A)$ is such that $\pi(B) = 1$. Let for example $\Delta^1 \in [0,1]^{4 \times 2^7}$ be the correspondance matrix such that $\Delta^1_{B,\{\beta_1\}} = 1/2$ and $\Delta^1_{B,\{\beta_2\}} = 1/2$. Let $\{\alpha\}$ be the set such that $A \mathcal{R} \{\alpha\}$. Let ϱ be the distribution on Q such that $\varrho(\beta_1) = \varrho(\beta_2) = 1/2$. ϱ is indeed in $\varphi_5(\alpha)$. Let $\Delta^2 \in [0,1]^{7 \times 2^7}$ be the correspondance matrix such that $\Delta^2_{\beta_1,\{\beta_1\}} = 1$ and $\Delta^2_{\beta_2,\{\beta_2\}} = 1$. It is then obvious that*

   (a) *for all t and T, if $\Delta^2_{t,T} > 0$, then $t \in T$, and*

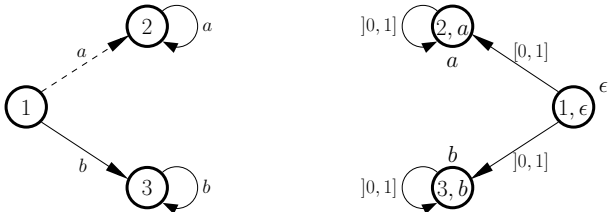
   (b) *$\pi \times \Delta^1 = \varrho \times \Delta^2$ holds.*

Consider state B of I_4.

1. *We have $B \mathcal{R} \{\beta_1\}$ and $B \mathcal{R} \{\beta_2\}$. It holds that $V_4(B) = b = V_5(\beta_1) = V_5(\beta_2)$.*

2. *Consider a distribution $\pi \in \varphi_4(B)$ (for example such that $\pi(C) < 1/2$). Let Δ^1 be an admissible correspondance matrix. We must have $\Delta^1_{C,\{\delta_1,\delta_2\}} = 1$ and $\Delta^1_{D,\{\gamma_1,\gamma_2\}} = 1$. Consider $\{\beta_1\}$ the set such that $B \mathcal{R} \{\beta_1\}$ (if $\pi(C) > 1/2$ then pick up $\{\beta_2\}$ instead). Let ϱ be the distribution such that $\varrho(\delta_1) = \pi(C)$ and $\varrho(\gamma_1) = \pi(D)$. Since $\pi(C) < 1/2$, we have $\varrho \in \varphi_5(\beta_1)$. Let Δ^2 be a correspondance matrix such that $\Delta^2_{\delta_1,\{\delta_1,\delta_2\}} = 1$ and $\Delta^2_{\gamma_1,\{\gamma_1,\gamma_2\}} = 1$. It is obvious that*

   (a) *for all t and T, if $\Delta^2_{t,T} > 0$, then $t \in T$, and*



(a) A MTS M (b) The IMC $\widehat{M}$

Figure 2.5: An example of the translation from Modal Transition Systems to IMCs

(b) $\pi \times \Delta^1 = \varrho \times \Delta^2$ *holds.*

The rest of the unfolding is obvious, and $\mathcal{R}$ *is thus a subset simulation.*

Given two IMCs I_1 and I_2, the existence of such a subset simulation between I_1 and I_2 is decidable, using a standard calculation. The algorithm works as follows: first consider the total relation and check whether it is a subset-simulation. Then refine it, by removing violating pairs of states, and check again until a fixpoint is reached (it becomes a subset-simulation or it is empty). Checking whether a given relation is a subset simulation has a single exponential complexity. Checking the second condition in the definition can be done in single exponential time by solving polynomial constraints with fixed quantifiers for each pair (p, T) in the relation. There are at most $|P| \cdot 2^{|Q|}$ such pairs, which gives a single exponential time bound for the cost of one iteration of the fixpoint loop. There are at most $|P| \cdot 2^{|Q|}$ elements in the total relation and at least one is removed in an iteration, which gives $O(|P| \cdot 2^{|Q|})$ as the bound on the number of iterations. Since a polynomial of two exponentials, is still a single exponential, we obtain a single exponential time for running time of this computation.

Interestingly this tells us that all three refinements are in EXPTIME. Still, weak refinement seems easier to check than thorough. For TR the number of iterations on the state-space of the relation is exponential while it is only quadratic for the weak refinement. Also, the formula to check at each step of the procedure involves a single quantifier alternation for the weak, and three alternations for the thorough refinement.

The lower bound of Theorem 2.2 is shown by a polynomial reduction of the thorough refinement problem for modal transition systems to TR of IMCs. That problem is known to be EXPTIME-complete [17].

First recall the following definitions of Modal Transition Systems and their implementations.

Definition 2.8 (Modal Transition System)**.** *A modal transition system (an MTS in short) [100] is a tuple* $M = (S, s_0, A, \rightarrow, \dashrightarrow)$*, where* S *is the set of states,* s_0 *is the initial state, and* $\rightarrow \subseteq S \times A \times S$ *are the transitions that* must *be taken and* $\dashrightarrow \subseteq S \times A \times S$ *are the transitions that* may *be taken. In addition, it is assumed that* $\rightarrow \subseteq \dashrightarrow$*.*

An implementation of an MTS is a labelled transition system, i.e., an MTS where $(\rightarrow) = (\dashrightarrow)$. We say that an MTS is deadlock-free if and only if each of its states has at least one outgoing must transition.

Consider now the following definition for refinement of MTS.

Definition 2.9 (Refinement for MTS)**.** *A modal transition system $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ refines another modal transition system $N = (T, t_0, A, \rightarrow, \dashrightarrow)$ iff there exists a refinement relation $R \subseteq S \times T$ containing (s_0, t_0) such that if $(s, t) \in R$ then*

1. *whenever $t \xrightarrow{a} t'$ then there exists $s' \in S$ such that $s \xrightarrow{a} s'$ and $(s', t') \in R$*
2. *whenever $s \overset{a}{\dashrightarrow} s'$ then there exists $t' \in T$ such that $t \overset{a}{\dashrightarrow} t'$ and $(s', t') \in R$*

A labelled transition system *implements* a MTS if it refines it in the above sense. Thorough refinement of MTSs is defined as inclusion of implementation sets, analogously to IMCs.

We describe here a translation of MTSs into IMCs which preserves implementations. By definition, Markov chains do not allow deadlock-states. Thus consistent IMCs must not allow deadlock-states either. As a consequence, in order to be coherent, we assume that we only work with modal transition systems that have no deadlock-states. This is a safe assumption: it is easy to transform two arbitrary MTSs into deadlock-free ones, without affecting the thorough refinement between them. We present a transformation that takes any two MTS and transforms them into MTSs without deadlocks preserving the notion of thorough refinement between them.

Let $M = \langle S, s_0, A, \rightarrow, \dashrightarrow \rangle$ be a MTS. Let $\perp \notin A$ be a new action variable, and $q \notin S$ be a new state variable. Define a new MTS $M_\perp = \langle S \cup \{q\}, s_0, A \cup \{\perp\}, \rightarrow_\perp, \dashrightarrow_\perp \rangle$ as follows: for all $s, s' \in S$ and $a \in A$, $s \xrightarrow{a}_\perp s' \iff s \xrightarrow{a} s'$ and $s \overset{a}{\dashrightarrow}_\perp s' \iff s \overset{a}{\dashrightarrow} s'$. In addition, consider the following transitions: for all $s \in S \cup \{q\}$, $s \xrightarrow{\perp}_\perp q$ and $s \overset{\perp}{\dashrightarrow}_\perp q$. In this way, every state of $M_\perp$ has at least one must outgoing transition. Moreover, it is trivial that this transformation preserves the notion of thorough refinement. This is stated in the following lemma:

Lemma 2.3. *Let M and M' be two MTS. If $\perp$ is in neither of their sets of actions, we have*

$$[\![M]\!] \subseteq [\![M']\!] \iff [\![M_\perp]\!] \subseteq [\![M'_\perp]\!]$$

We now describe the polynomial translation of MTSs without deadlock states into IMCs which preserves implementations. The IMC $\widehat{M}$ corresponding to a MTS $M = \langle S, s_0, A, \rightarrow, \dashrightarrow \rangle$ is defined by the tuple $\widehat{M} = \langle Q, q_0, A \cup \{\epsilon\}, \varphi, V \rangle$, with $\epsilon \notin A$, and where

- $Q = S \times (\{\epsilon\} \cup A)$,
- $q_0 = (s_0, \epsilon)$,
- for all $(s, x) \in Q$, $V((s, x)) = \{x\}$, and
- φ is defined as follows: for all $t, s \in S$ and $b, a \in (\{\epsilon\} \cup A)$,
  - $\varphi((t, b))((s, a)) =]0, 1]$ if $t \xrightarrow{a} s$,
  - $\varphi((t, b))((s, a)) = [0, 0]$ if $t \not\overset{a}{\dashrightarrow} s$, and
  - $\varphi((t, b))((s, a)) = [0, 1]$ otherwise.

Remark that since $\epsilon \notin A$, the only state that is associated to the valuation ϵ is the initial state.

The encoding is illustrated in Figure 2.5, where unreachable states are omitted.

We first state two lemmas that will be needed in order to prove the main theorem of the section: the encoding presented above reduces the problem of checking thorough refinement on modal transition systems to checking thorough refinement on IMCs.

Lemma 2.4. *Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. We have*

$$I \models M \Rightarrow [\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$$

Proof. We first recall the definition of a satisfaction relation for MTS: Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. $I \models M$ iff there exists a relation $\mathcal{R} \subseteq S_I \times S$ such that

1. $s_0^I \,\mathcal{R}\, s_0$
2. Whenever $s_I \,\mathcal{R}\, s$, we have
   (a) For all $a \in A$, $s_I' \in S_I$, $s_I \xrightarrow{a} s_I'$ in I implies that there exists $s' \in S$ such that $s \overset{a}{\dashrightarrow} s'$ in M and $s_I' \,\mathcal{R}\, s'$.
   (b) For all $a \in A$, $s' \in S$, $s \xrightarrow{a} s'$ in M implies that there exists $s_I' \in S_I$ such that $s_I \xrightarrow{a} s_I'$ in M and $s_I' \,\mathcal{R}\, s'$.

Such a relation is called a satisfaction relation for MTS.

Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. Let $\widehat{M} = \langle Q, q_0, A \cup \{\epsilon\}, \varphi, V\rangle$ and $\widehat{I} = \langle Q_I, q_0^I, A \cup \{\epsilon\}, \varphi_I, V_I\rangle$ be the IMCs defined using the above transformation. Let $Q = \{q_0, \dots q_n\}$ and $Q_I = \{q_0^I, \dots q_m^I\}$.

Suppose that $I \models M$. There exists a satisfaction relation for MTS $\mathcal{R} \subseteq S_I \times S$ such that $s_0^I \,\mathcal{R}\, s_0$. We prove that $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$.

Let $T = \langle Q_T, q_0^T, \pi^T, A, V_T\rangle$ be an MC such that $Q_T = \{q_0^T, \dots, q_k^T\}$ and $T \in [\![\widehat{I}]\!]$. As a consequence, there exists a satisfaction relation for IMCs $\mathcal{R}_1 \subseteq Q_T \times Q_I$ such that $q_0^T \,\mathcal{R}_1\, q_0^I = (s_0^I, \epsilon)$. Define the new relation $\mathcal{R}_2 \subseteq Q_T \times Q$ such that $q^T \,\mathcal{R}_2\, q = (s, x)$ iff there exists $s_I \in S_I$ such that $q^T \,\mathcal{R}_1\, q^I$ with $q^I = (s_I, x)$ and $s_I \,\mathcal{R}\, s$. We prove that $\mathcal{R}_2$ is a satisfaction relation for IMCs between T and $\widehat{M}$.

Let $q_i^T \in, Q^T, q = (s, x) \in Q$ and $q^I = (s_I, x) \in Q^I$ such that $q^T \,\mathcal{R}_1\, q^I$ and $s_I \,\mathcal{R}\, s$, i.e. $q^T \,\mathcal{R}_2\, q$. We have

1. Since $q^T \,\mathcal{R}_1\, q^I = (s_I, x)$, we have $V_T(q^T) = V_I((s_I, x)) = \{x\}$. Thus $V_T(q^T) = V((s, x)) = \{x\}$.
2. Let $\Delta^1[0,1]^{k\times m}$ be the correspondence matrix witnessing $q^T \,\mathcal{R}_1(s_I, x)$, and let $\Delta^2 \in [0,1]^{k\times n}$ such that for all $q_j^T \in Q_T$ and $q_l \in Q$ with $q_l = (s', y)$, if $\{s_I' \in S_I \mid s_I' \,\mathcal{R}\, s'\} \neq \emptyset$ and $s \overset{y}{\dashrightarrow} s'$, then define

$$\Delta^2_{j,l} = \sum_{\{q_r^I = (s_I', y) \in Q^I \;\mid\; s_I' \,\mathcal{R}\, s'\}} \frac{\Delta^1_{j,r}}{|\{s'' \in S \mid s_I' \,\mathcal{R}\, s'' \text{and } s \overset{y}{\dashrightarrow} s''\}|};$$

Else, $\Delta^2_{j,l} = 0$.

Recap that we suppose that all must transitions are also may transitions. The definition above potentially gives a non-zero value to $\Delta^2_{j,l}$, with $q_l = (s', y)$, if there exists a may (or must) transition from s to s' in S labelled with y and if there exists a state s'_I in I such that $s'_I \mathcal{R} s'$. Since there may be several states s'' such that $s \overset{y}{\dashrightarrow} s''$, the term $\Delta^1_{j,r}$ may appear in the definition of several coefficients $\Delta^2_{j,l}$. Thus the fraction in the definition of $\Delta^2_{j,l}$.

Let $q^T_j \in Q_T$. We prove that $\sum_{l=0}^{n} \Delta^2_{j,l} = 1$: By definition of Δ^1, we have $\sum_{r=0}^{m} \Delta^1_{j,r} = 1$. $\sum_{l=0}^{n} \Delta^2_{j,l} = \sum_{\{q_l=(s',y) \mid \exists s'_I,\ s'_I \mathcal{R} s' \text{and } s \overset{y}{\dashrightarrow} s'\}} \sum_{\{q^I_r=(s'_I,y) \mid s'_I \mathcal{R} s'\}} \frac{\Delta^1_{j,r}}{|\{s'' \in S \mid s'_I \mathcal{R} s'' \text{and } s \overset{y}{\dashrightarrow} s''\}|}$.

Clearly, for all $q^I_r = (s'_I, y)$ such that $\Delta^1_{j,r} > 0$, the term $\frac{\Delta^1_{j,r}}{|\{s'' \in S \mid s'_I \mathcal{R} s'' \text{and } s \overset{y}{\dashrightarrow} s''\}|}$ will appear exactly $|\{s'' \in S \mid s'_I \mathcal{R} s'' \text{ and } s \overset{y}{\dashrightarrow} s''\}|$ times in the expression above. As a consequence, $\sum_{l=0}^{n} \Delta^2_{j,l} = \sum_{r=0}^{m} \Delta^1_{j,r} = 1$. Finally, the row Δ^2_j defines a distribution on Q.

Moreover, we prove that the distribution vector $\pi^T_i \times \Delta^2$ is in $\varphi((s, x))$. Let $q_l = (s', y) \in Q$. By construction, $\varphi((s, x)(s', y))$ is either $\{0\}$, $[0, 1]$ or $]0, 1]$. We will thus prove that (a) if $\sum_{j=0}^{k} \pi^T_i(j)\Delta^2_{j,l} > 0$, then $\varphi((s, x)(s', y)) \neq \{0\}$; and (b) if $\varphi((s, x)(s', y)) =]0, 1]$, then $\sum_{j=0}^{k} \pi^T_i(j)\Delta^2_{j,l} > 0$.

(a) Suppose $\sum_{j=0}^{k} \pi^T_i(j)\Delta^2_{j,l} > 0$. By definition, there must exist j such that $\pi^T_i(j) > 0$ and $\Delta^2_{j,l} > 0$. As a consequence, by definition of Δ^2, there exists a transition $s \overset{y}{\dashrightarrow} s'$ in M and $\varphi((s, x), (s', y)) \neq \{0\}$.

(b) If $\varphi((s, x)(s', y)) =]0, 1]$, then there exists a transition $s \xrightarrow{y} s'$ in M. As a consequence, by $\mathcal{R}$, there exists $s'_I \in S_I$ such that $s_I \xrightarrow{y} s'_I$ in I and $s'_I \mathcal{R} s'$. Thus $\varphi_I((s_I, x), (s'_I, y)) =]0, 1]$. Let $q^I_r = (s'_I, y)$. By definition of Δ^1, we know that $\sum_{j=0}^{k} \pi^T_i(j)\Delta^1_{j,r} > 0$, thus there exists $q^T_x \in Q_T$ such that $\pi^T_i(x) > 0$ and $\Delta^1_{x,r} > 0$. Since $s'_I \mathcal{R} s'$ and $s \xrightarrow{y} s'$, we have $\Delta^2_{x,l} > 0$, thus $\sum_{j=0}^{k} \pi^T_i(j)\Delta^2_{j,l} > 0$.

Finally, if $\Delta^2_{j,l} > 0$ with $q_l = (s', y)$, there exists $s'_I \in S_I$ such that $s'_I \mathcal{R} s'$ and $\Delta^1_{j,r} > 0$ with $q^I_r = (s'_I, y)$. By definition of Δ^1, we have $q^T_j \mathcal{R}_1 q^I_r$. As a consequence, $q^T_j \mathcal{R}_2 q_l$.

$\mathcal{R}_2$ satisfies the axioms of a satisfaction relation for IMCs, thus $T \in [\![\widehat{M}]\!]$ and finally $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$.

□

Lemma 2.5. *Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s^I_0, A, \rightarrow)$ be a transition system. We have*

$$[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!] \Rightarrow I \models M$$

Proof.

Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$be an MTS and $I = (S_I, s^I_0, A, \rightarrow)$ be a transition system. Let $\widehat{M} = \langle Q, q_0, A \cup \{\epsilon\}, \varphi, V\rangle$ and $\widehat{I} = \langle Q_I, q^I_0, A \cup \{\epsilon\}, \varphi_I, V_I\rangle$ be the IMCs defined by the transformation form MTS to IMC. Let $Q = \{q_0, \ldots, q_n\}$ and $Q_I = \{q^I_0, \ldots, q^I_m\}$.

Suppose that $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$. We prove that $I \models M$.

Let $T = \langle Q_T, q_0^T, \pi^T, V_T, A\rangle$ be an MC with $Q_T = \{q_0^T, \ldots, q_k^T\}$ such that $T \in [\![\widehat{I}]\!]$. As a consequence, there exists two satisfaction relations for IMCs $\mathcal{R}_1 \subseteq Q_T \times Q_I$ and $\mathcal{R}_2 \subseteq Q_T \times Q$ such that $q_0^T \,\mathcal{R}_1\, q_0^I = (s_0^I, \epsilon)$ and $p_0 \,\mathcal{R}_2\, q_0 = (s_0, \epsilon)$. Define the new relation $\mathcal{R} \subseteq S_I \times S$ such that $s_I \,\mathcal{R}\, s$ iff there exists $q^T \in Q_T$ and $x \in (\{\epsilon\} \cup A)$ such that $q^T \,\mathcal{R}_1\, q = (s_I, x)$ and $q^T \,\mathcal{R}_2\, q^I = (s, x)$. We have

1. $q_0^T \,\mathcal{R}_1 (s_0^I, \epsilon)$ and $q_0^T \,\mathcal{R}_2 (s_0, \epsilon)$. As a consequence, $s_0^I \,\mathcal{R}\, s_0$.

2. Let $q_i^T \in Q_T$, $q = (s, x) \in Q$ and $q^I = (s_I, x) \in Q_I$ such that $q_i^T \,\mathcal{R}_1\, q^I$ and $q_i^T \,\mathcal{R}_2\, q$ and let $\Delta^1 \in [0,1]^{k \times m}$ and $\Delta^2 \in [0,1]^{k \times n}$ be the associated correspondance matrices.

   (a) Let $y \in A$ and $s_I' \in S_I$ such that $s_I \xrightarrow{y} s_I'$ in I. We prove that there exists $s' \in S$ such that $s \overset{y}{\dashrightarrow} s'$ and $s_I' \,\mathcal{R}\, s'$. Let $q_r^I = (s_I', y)$.

   By definition of $\widehat{I}$, we have $\varphi_I((s_I, x), (s_I', y)) =]0, 1]$. As a consequence, $[\pi_i^T \times \Delta^1]_r = \sum_{j=0}^{k} \pi_i^T(j) \Delta_{j,r}^1 > 0$. Thus there exists q_j^T in Q_T such that $\pi_i^T(j) > 0$ and $\Delta_{j,r}^1 > 0$. As a consequence, by definition of Δ^1, we have $q_j^T \,\mathcal{R}_1\, q_r^I = (s_I', y)$, thus $V_T(q_j^T) = V_I((s_I', y)) = \{y\}$.

   By definition of Δ^2, since $\pi_i^T(j) > 0$, we know that $\sum_{l=0}^{n} \Delta_{j,l}^2 = 1$. As a consequence, there exists $q_l = (s', z) \in Q$ such that $\Delta_{j,l}^2 > 0$. By definition of Δ^2, we have $q_j^T \,\mathcal{R}_2\, q_l = (s', z)$ and since $V_T(q_j^T) = \{y\}$, we must have $z = y$.

   Moreover, By definition of Δ^2, we know that $[\pi_i^T \times \Delta^2]_l \in \varphi((s, x), (s', y))$. Since $[\pi_i^T \times \Delta^2]_l = \sum_{t=0}^{k} \pi_i^T(t) \Delta_{t,l}^2 > 0$, we have $\varphi((s, x), (s', y)) \neq \{0\}$. Thus, by definition of $\widehat{M}$, there exists a transition $s \overset{y}{\dashrightarrow} s'$ in M. Finally, we have both $q_j^T \,\mathcal{R}_1 (s_I', y)$ and $q_j^T \,\mathcal{R}_2 (s', y)$, thus $s_I' \,\mathcal{R}\, s'$.

   (b) Let $y \in A$ and $s' \in S$ such that $s \xrightarrow{y} s'$ in M. We prove that there exists $s_I' \in S_I$ such that $s_I \xrightarrow{y} s_I'$ in I and $s_I' \,\mathcal{R}\, s'$. Let $q_l = (s', y)$.

   By definition of $\widehat{M}$, we have $\varphi((s, x), (s', y)) =]0, 1]$. As a consequence, $[\pi_i^T \times \Delta^2]_l = \sum_{j=0}^{k} \pi_i^T(j) \Delta_{j,l}^2 > 0$. Thus there exists q_j^T in Q_T such that $\pi_i^T(j) > 0$ and $\Delta_{j,l}^2 > 0$. By definition of Δ^2, we have $q_j^T \,\mathcal{R}_2\, q_l = (s', y)$, thus $V_T(q_j^T) = V((s', y)) = \{y\}$.

   By definition of Δ^1, since $\pi_i^T(j) > 0$, we have $\sum_{r=0}^{m} \Delta_{j,r}^1 = 1$. As a consequence, there exists $q_r^I = (s_I', z) \in Q_I$ such that $\Delta_{j,r}^1$. By definition of Δ^1, we have $q_j^T \,\mathcal{R}_1\, q_r^I = (s_I', z)$ and since $V_T(q_j^T) = \{y\}$, we must have $z = y$.

   Moreover, by definition of Δ^1 we know that $[\pi_i^T \times \Delta^1]_r \in \varphi_I((s_I, x), (s_I', y))$. Since $[\pi_i^T \times \Delta^1]_r = \sum_{t=0}^{k} \pi_i^T(t) \Delta_{t,r}^1 > 0$, we have $\varphi_I((s_I, x), (s_I', y)) \neq \{0\}$. Thus, by definition of $\widehat{I}$, there exists a transition $s_I \xrightarrow{y} s_I'$ in I (remember that I is a classical transition system). Finally, we have both $q_j^T \,\mathcal{R}_1 (s_I', y)$ and $q_j^T \,\mathcal{R}_2 (s', y)$, thus $s_I' \,\mathcal{R}\, s'$.

   Finally, $\mathcal{R}$ is a satisfaction relation for MTS, and $I \models M$

□

From the two lemmas stated above, we deduce the following theorem.

Theorem 2.6. *Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be an MTS and $I = (S_I, s_0^I, A, \rightarrow)$ be a transition system. We have*

$$I \models M \iff [\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$$

We now define a construction f that builds, for all implementations C of $\widehat{M}$, a corresponding implementation $f(C)$ of M:

Let $M = (S, s_0, A, \rightarrow, \dashrightarrow)$ be a MTS. Let $\widehat{M} = \langle S \times (\{\epsilon\} \cup A), (s_0, \epsilon), \{\epsilon\} \cup A, \varphi, V\rangle$ be the transformation of M defined as above. Let $C = \langle Q, q_0, A, \pi, V'\rangle$ be a MC such that $C \models \widehat{M}$ for some satisfaction relation on IMCs $\mathcal{R}$.

Define $f(C) = (Q, q_0, A, \rightarrow)$ the transition system such that $q \xrightarrow{a} q'$ whenever $\pi(q, q') > 0$ and $V'(q') = \{a\}$.

By construction, it is trivial that (1) $f(C) \models M$ for some satisfaction relation on MTS $\mathcal{R}'$ and (2) $C \models \widehat{f(C)}$ for some satisfaction relation on IMCs $\mathcal{R}''$. These satisfaction relations are defined as follows: $q\,\mathcal{R}'\,s$ whenever there exists $x \in \{\epsilon\} \cup A$ such that $q\,\mathcal{R}(s, x)$, and $q\,\mathcal{R}''(q', x)$ whenever $q = q'$.

From the above construction and Theorem 2.6, we obtain the main theorem of the section: the transformation $M \rightarrow \widehat{M}$ preserves thorough refinement.

Theorem 2.7. *Let M and M' be two Modal Transition Systems and $\widehat{M}$ and $\widehat{M'}$ be the corresponding IMCs defined. We have*

$$M \preceq_T M' \iff \widehat{M} \preceq_T \widehat{M'}$$

Proof. Let M and M' be two MTS, and $\widehat{M}$ and $\widehat{M'}$ the corresponding IMCs.

- $\Rightarrow$ Suppose that $M \preceq^{th} M'$, and let C be a MC such that $C \models \widehat{M}$. We have by construction $f(C) \models M$, thus $f(C) \models M'$. By Theorem 2.6, we have $[\![\widehat{f(C)}]\!] \subseteq [\![\widehat{M'}]\!]$, and we know that $C \models \widehat{f(C)}$. As a consequence, $C \models \widehat{M'}$.
- $\Leftarrow$ Suppose that $\widehat{M} \preceq^{th} \widehat{M'}$, and let I be a TS such that $I \models M$. By Theorem 2.6, we have $[\![\widehat{I}]\!] \subseteq [\![\widehat{M}]\!]$, thus by hypothesis $[\![\widehat{I}]\!] \subseteq [\![\widehat{M'}]\!]$. Finaly, by Theorem 2.6, we obtain that $I \models M'$.

□

Crucially the translation $M \rightarrow \widehat{M}$ is polynomial. Thus if we had a subexponential algorithm for TR of IMCs, we could use it to obtain a subexponential algorithm for TR of MTSs, which is impossible according to [17]. This proves that TR of IMCs is at least EXPTIME-hard.

2.4 Determinism

Although both are in EXPTIME, deciding weak refinement is easier than deciding thorough refinement. Nevertheless, since these two refinements do not coincide, in general, a procedure to check weak refinement cannot be used to decide thorough refinement. Observe that weak refinement has a syntactic definition very much like simulation for transition systems. On the

other hand thorough refinement is a semantic concept, just as trace inclusion for transition systems. It is well known that simulation and trace inclusion coincide for deterministic automata. Similarly for MTSs it is known that TR coincides with modal refinement for deterministic objects. It is thus natural to define deterministic IMCs and check whether thorough and weak refinements coincide on these objects.

In our context, an IMC is deterministic if, from a given state, one cannot reach two states that share common atomic propositions.

Definition 2.10 (Determinism)**.** *Let $I = \langle Q, q_0, \varphi, A, V\rangle$ be an IMC. I is* deterministic *iff for all states $q, r, s \in Q$, if there exists a probability distribution $\sigma \in \varphi(q)$ such that $\sigma(r) > 0$ and $\sigma(s) > 0$, then $V(r) \neq V(s)$.*

The above definition verifies that two states that are reachable *with the same admissible distribution* always have different valuations. In a semantic interpretation, this means that there exist no implementation of I in which two states with the same valuations can be successors of the same source state. One can also propose another, slightly more syntactic definition for determinism.

Definition 2.11 (Strong Determinism)**.** *Let $I = \langle Q, q_0, \varphi, A, V\rangle$ be an IMC. I is* strongly deterministic *iff for all states $q, r, s \in Q$, if there exist a probability distribution $\sigma \in \varphi(q)$ such that $\sigma(r) > 0$ and a probability distribution $\varrho \in \varphi(q)$ such that $\varrho(s) > 0$, then $V(r) \neq V(s)$.*

This definition differs from Definition 2.10 in that it requires that, from a given state q, one cannot possibly reach two states r and s with the same set of propositions, even using two different distributions (implementations).

Checking weak determinism requires solving a cubic number of linear constraints: for each state check the linear constraint of the definition—one per each pair of successors of a state. Checking strong determinism can be done by solving only a quadratic number of linear constraints—one per each successor of each state. Luckily, due to the convexity of the set of admissible distributions in a state, these two notions coincide for IMCs, so the more efficient, strong determinism can be used in algorithms. However, we will see in Chapter 3 that these notions differ when considering more expressive specifications.

Theorem 2.8. *An IMC I is deterministic iff it is strongly deterministic.*

Proof. It directly follows from the definitions that strong determinism implies weak determinism. We prove that if an IMC I is not strongly deterministic, then it is not weakly deterministic either.

Let $I = \langle Q, q_0, \varphi, A, V\rangle$ be an IMC. If I is not strongly deterministic, there exist two admissible distributions on next states for q: σ and $\varrho \in \varphi(q)$ such that $\sigma(r) > 0$, $\sigma(s) = 0$, $\varrho(r) = 0$, $\varrho(s) > 0$ and $V(r) = V(s)$. In order to prove that I is not weakly deterministic, we build a distribution γ that we prove correct w.r.t the interval specifications, i.e. $\gamma \in \varphi(q)$, and such that $\gamma(r) > 0$ and $\gamma(s) > 0$.

Since $\sigma(r) > 0$, there exists $a > 0$ such that $\varphi(q)(r) = [0, a]$ or $[0, a[$. Moreover, since $\varrho(s) > 0$, there exists $b > 0$ such that $\varphi(q)(s) = [0, b]$ or $[0, b[$. Let $c = \mathsf{Min}(a, b)$, and define $\gamma(q') = \sigma(q')$ for all $q' \notin \{r, s\}$, $\gamma(r) = \sigma(r) - c/2$, and $\gamma(s) = c/2$. By construction, $\gamma \in \varphi(q)$ and we have $\gamma(r) > 0$ and $\gamma(s) > 0$. As a consequence, I is not weakly deterministic.

Finally, an IMC I is strongly deterministic iff it is also weakly deterministic.

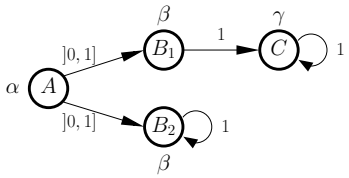


Figure 2.6: An IMC I whose semantics cannot be captured by a deterministic IMC

□

It is worth mentioning that deterministic IMCs is a strict subclass of IMCs. Figure 2.6 shows an IMC I whose set of implementations cannot be represented by a deterministic IMC.

We now state the main theorem of the section that shows that for deterministic IMCs, the weak refinement, and indeed also the strong refinement, correctly capture the thorough refinement. The proof of this theorem is postponed to Section 3.6 in Chapter 3, where it will be given in a more general setting.

Theorem 2.9. *Let I and I' be two deterministic IMCs with no inconsistent states. It is equivalent to say that (1) I thoroughly refines I', (2) I weakly refines I' and (3) I strongly refines I'.*

2.5 Common Implementation and Consistency

We now turn our attention to the problem of implementation of several IMC specifications by the same probabilistic system modeled as a Markov Chain. We start with defining the problem:

Definition 2.12 (Common Implementation (CI))**.** *Given $k > 1$ IMCs I_i, $i = 1 \dots k$, does there exist a Markov Chain C such that $C \models I_i$ for all i?*

Somewhat surprisingly we find out that, similar to the case of TR, the CI problem is not harder for IMCs than for modal transition systems. The following theorem summarizes our main result about CI:

Theorem 2.10. *Deciding the existence of a common implementation between k IMCs is EXPTIME-complete.*

We will establish lower and upper bound for common implementation. We will then use these results to solve the consistency problem.

To establish a lower bound for common implementation, we propose a reduction from the common implementation problem for modal transition systems (MTS). This latter problem has recently been shown to be EXPTIME-complete when the number of MTS is not known in advance and PTIME-complete otherwise [10]. For a set of modal transition systems M_i, $i = 1 \dots k$, translate each M_i, into an IMC $\widehat{M_i}$, using the same rules as in Section 2.3. It turns out that the set of created IMCs has a common implementation if and only if the original modal transition systems had. Thus the following theorem.

Theorem 2.11. *Let M_i be MTSs for $i = 1, \dots, k$. We have*

$$\exists I \forall i : I \models M_i \iff \exists C \forall i : C \models \widehat{M_i},$$

where I is a transition system, C is a Markov Chain and $\widehat{M_i}$ is the IMC obtained with the transformation defined in Section 2.3.3.

Proof. $\Rightarrow$: Let M_i be MTSs for $i = 1, \ldots, k$. Let I be a TS such that $\forall i, I \models M_i$. We prove that there exists a MC C such that $\forall i, C \models \widehat{M_i}$.

Let $1 \leq i \leq k$. Since $I \models M_i$, we have, by Theorem 2.6, that $[\![\widehat{I}]\!] \subseteq [\![\widehat{\widehat{M_i}}]\!]$.

Moreover, by definition, $[\![\widehat{I}]\!] \neq \emptyset$. Thus there exists a MC $C \in [\![\widehat{I}]\!]$, and $\forall i, C \models \widehat{M_i}$.

$\Leftarrow$: Let M_i be MTSs for $i = 1, \ldots, k$. Let M be a MC such that $\forall i, M \models \widehat{M_i}$. We prove that there exists a TS I such that $\forall i, I \models M_i$.

Let $1 \leq i \leq k$. Since $C \models \widehat{M_i}$, we have, by the transformation f defined in Section 2.3.3, that $f(C) \models M_i$. As a consequence, we have that $\forall i, f(C) \models M_i$. □

As for TR, since the translation is polynomial, the problem of CI for IMCs has to be at least EXPTIME-hard (otherwise it would give a sub-EXPTIME algorithm for CI of MTSs).

To address the upper bound we first propose a simple construction to check if there exists a CI for two IMCs. We start with the definition of *consistency relation* that witnesses a common implementation between two IMCs.

Definition 2.13. *Let $I_1 = \langle Q_1, q_0^1, \varphi_1, A, V_1 \rangle$ and $I_2 = \langle Q_2, q_0^2, \varphi_2, A, V_2 \rangle$ be IMCs. The relation $\mathcal{R} \subseteq Q_1 \times Q_2$ is a* consistency relation *on the states of I_1 and I_2 iff, whenever $(u, v) \in \mathcal{R}$ then*

- *$V_1(u) = V_2(v)$ and*
- *there exists a $\varrho \in \mathsf{Distr}(Q_1 \times Q_2)$ such that*
  1. *$\forall u' \in Q_1 : \sum_{v' \in Q_2} \varrho(u', v') \in \varphi_1(u)(u') \wedge \forall v' \in Q_2 : \sum_{u' \in Q_1} \varrho(u', v') \in \varphi_2(v)(v')$, and*
  2. *$\forall (u', v') \in Q_1 \times Q_2$ st. $\varrho(u', v') > 0$, then $(u', v') \in \mathcal{R}$.*

We now prove that the existence of a consistency relation is equivalent to the existence of a common implementation between two IMCs.

Theorem 2.12. *Let $I_1 = \langle Q_1, q_0^1, \varphi_1, A, V_1 \rangle$ and $I_2 = \langle Q_2, q_0^2, \varphi_2, A, V_2 \rangle$ be IMCs with $Q_1 = \{q_0^1, \ldots, q_n^1\}$ and $Q_2 = \{q_0^2, \ldots, q_m^2\}$. I_1 and I_2 have a common implementation iff there exists a consistency relation $\mathcal{R}$ such that $q_0^1 \, \mathcal{R} \, q_0^2$.*

Proof. $\Rightarrow$: Assume that there exists a MC $C = \langle P, p_0, \pi, A, V_C \rangle$, with $P = \{p_0, \ldots, p_k\}$, such that $C \models I_1$ and $C \models I_2$. This implies that there exists satisfaction relations $\mathcal{R}_1 \subseteq P \times Q_1$ and $\mathcal{R}_2 \subseteq P \times Q_2$ such that $p_0 \, \mathcal{R}_1 \, q_0^1$ and $p_0 \, \mathcal{R}_2 \, q_0^2$.

Let $\mathcal{R} \subseteq Q_1 \times Q_2$ the relation such that $q_1 \, \mathcal{R} \, q_2$ iff there exists $p \in P$ such that $p \, \mathcal{R}_1 \, q_1$ and $p \, \mathcal{R}_2 \; q_2$. We prove that $\mathcal{R}$ is a consistency relation relating q_0^1 and q_0^2. Indeed $(q_0^1, q_0^2) \in \mathcal{R}$ because by definition of C, we have $p_0 \, \mathcal{R}_1 \, q_0^1$ and $p_0 \, \mathcal{R}_2 \, q_0^2$. Let $(q^1, q^2) \in \mathcal{R}$ and $p_i \in P$ such that $p_i \, \mathcal{R}_1 \, q^1$ and $p_i \, \mathcal{R}_2 \, q^2$.

1. By $\mathcal{R}_1$ and $\mathcal{R}_2$, $V_1(q^1) = V_C(p_i) = V_2(q^2)$.

2. Let Δ^1 and Δ^2 be the correspondance matrices witnessing $p_i \mathcal{R}_1 q^1$ and $p_i \mathcal{R}_2 q^2$. Define $\varrho \in \mathsf{Distr}(Q_1 \times Q_2)$ such that for all $1 \leq l \leq n$ and $1 \leq r \leq m$,

$$\varrho(q_l^1, q_r^2) = \sum_{j=0}^{k} \pi_{i,j} \Delta_{j,l}^1 \cdot \Delta_{j,r}^2. \tag{2.1}$$

   By definition of Δ^1 and Δ^2, we have $\sum_{\substack{1 \leq l \leq n \\ 1 \leq r \leq m}} \varrho(q_l^1, q_r^2) = 1$, and ϱ is indeed a distribution on $Q_1 \times Q_2$.

   Let $q_l^1 \in Q_1$.

$$\begin{aligned}
\sum_{r=0}^{m} \varrho(q_l^1, q_r^2) &= \sum_{r=0}^{m} \sum_{j=0}^{k} \pi_{i,j} \cdot \Delta_{j,l}^1 \cdot \Delta_{j,r}^2 \\
&= \sum_{j=0}^{k} \pi_{i,j} \cdot \Delta_{j,l}^1 \cdot (\sum_{r=0}^{m} \Delta_{j,r}^2) \\
&= \sum_{1 \leq j \leq k \mid \pi_{i,j} > 0} \pi_{i,j} \cdot \Delta_{j,l}^1 \quad \text{by definition of } \Delta^2 \\
&\in \varphi_1(q^1)(q_l^1) \quad \text{by definition of } \Delta^1.
\end{aligned}$$

   Similarly, for all $q_r^2 \in Q_2$, $\sum_{l=0}^{n} \varrho(q_l^1, q_r^2) \in \varphi_2(q^2)(q_r^2)$.

3. Let $q_l^1 \in Q_1$ and $q_r^2 \in Q_2$ be states such that $\varrho(q_l^1, q_r^2) > 0$. Then at least one term in Eq. (2.1) is positive. Thus, there exists $1 \leq j \leq k$ such that $\Delta_{j,l}^1 \cdot \Delta_{j,r}^2 > 0$. This implies that both factors are positive, and by definition of Δ^1 and Δ^2, we have that $(p_j, q_l^1) \in \mathcal{R}_1$ and $(p_j, q_r^2) \in \mathcal{R}_2$ and therefore $q_l^1 \mathcal{R} q_r^2$.

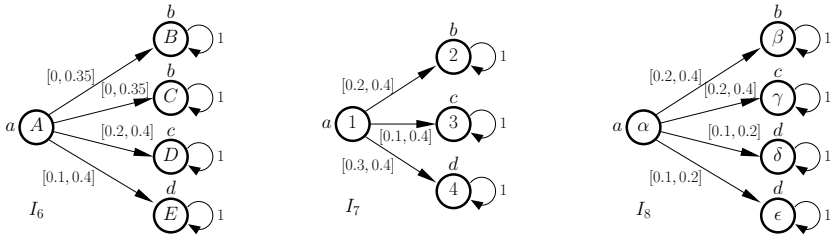
This proves that $\mathcal{R}$ is a consistency relation.

$\Leftarrow$: Assume that there exists a consistency relation $\mathcal{R}$ relating q_0^1 and q_0^2. We now construct a common implementation C, such that $C \models I_1$ and $C \models I_2$; we prove the former first. Let $C = \langle P, p_0, \pi, A, V_C \rangle$ such that

- $P = \{(q^1, q^2) \in Q_1 \times Q_2 \mid q^1 \mathcal{R} q^2\} = \{p_0, \ldots, p_k\}$;
- $p_0 = (q_0^1, q_0^2)$;
- $V_C((q^1, q^2)) = V_1(q^1) = V_2(q^2)$ by definition of $\mathcal{R}$;
- For each $p_i, p_j \in P$ with $p_i = (q^1, q^2)$ and $p_j = (q_l^1, q_r^2)$, let $\pi_{i,j} = \varrho(q_l^1, q_r^2)$, where ϱ is the distribution witnessing the membership of (q^1, q^2) in $\mathcal{R}$.

To show satisfaction between C and I_1, define $\mathcal{R}_s \subseteq P \times Q_1$ the relation such that $p = (q^1, q^2) \mathcal{R}_s q^{1\prime}$ iff $q^1 = q^{1\prime}$. We now prove that $\mathcal{R}_s$ is a satisfaction relation between C and I_1. Let $p_i = (q^1, q^2) \in P$ such that $(q^1, q^2) \mathcal{R}_s q^1$.

1. By definition of C, $V_C(p_i) = V_1(q^1)$.



(a) I_6 (b) I_7 (c) I_8

Figure 2.7: IMCs I_6, I_7, and I_8

2. Let $\Delta \in [0,1]^{k \times n}$ be the correspondance matrix derived as follows: $\Delta_{j,l} = 1$ if $p_j = (q_l^1, q_t^2)$ for some t and 0 else.

   (a) Let $p_j \in P$.

   $$\sum_{l=0}^{n} \Delta_{j,l} = 1 \quad \text{by definition.}$$

   (b) Let $q_l^1 \in Q_1$.

   $$\begin{aligned} \sum_{j=0}^{k} \pi_{i,j} \Delta_{j,l} &= \sum_{p_j \in P \mid p_j = (q_l^1, v')} \varrho(p_j) \\ &= \sum_{r=0}^{m} \varrho(q_l^1, q_r^2) \\ &\in \varphi_1(q^1)(q_l^1) \quad \text{by definition of } \mathcal{R}\,. \end{aligned}$$

   (c) Let $p_j = (q_l^1, q_r^2) \in P$ and $q_t^1 \in Q_1$ such that $\Delta_{j,t} > 0$. Then $q_l^1 = q_t^1$ and by definition, $(q_l^1, q_r^2)\, \mathcal{R}_s\, q_l^1$.

Thus $\mathcal{R}_s$ is a satisfaction relation, and $C \models I_1$. Analogously, it can be shown that $C \models I_2$. Finally C is a common implementation of I_1 and I_2. □

The consistency relation can be computed in polynomial time using a standard coinductive fixpoint iteration, where pairs violating Definition 2.13 are successively removed from $Q_1 \times Q_2$. Each iteration requires solving a polynomial number of linear equation systems, which can be done in polynomial time [93]. For the general problem of common implementation of k IMCs, we can extend the above definition of consistency relation to the k-ary relation in the obvious way, and the algorithm becomes exponential in the number of IMCs k, as the size of the state space $\prod_{i=0}^{k} |Q_i|$ is exponential in k.

As a side effect we observe that, exactly like for modal transition systems, CI becomes polynomial for any constant value of k, i.e. when the number of components to be checked is bounded by a constant.

Example. *Consider the three IMCs in Figure 2.7. We construct a consistency relation $\mathcal{R}$ for $k = 3$. The triple $(A, 1, \alpha)$ is in the relation $\mathcal{R}$ witnessed by the distribution ϱ that assigns $\frac{1}{6}$ to $(B, 2, \beta)$, $\frac{1}{6}$ to $(C, 2, \beta)$, $\frac{1}{3}$ to $(D, 3, \gamma)$, $\frac{1}{6}$ to $(E, 4, \delta)$, and $\frac{1}{6}$ to $(E, 4, \epsilon)$. The triples that are given positive probability by ϱ are also in the relation each by the distribution assigning*

probability 1 to itself. A common implementation $C = \langle P, p_0, \pi, A, V_C \rangle$ can be constructed as follows: $P = \{q | q \in \mathcal{R}\}$, $p_0 = (A, 1, \alpha)$, $V_C(p)$ is inherited from I_6, I_7, and I_8, and $\pi(p)(p') = \varrho(p')$, where ϱ is the distribution witnessing that $p \in \mathcal{R}$.

Consistency. A related problem is the one of checking consistency of a single IMC I, i.e. whether there exists a MC M such that $M \models I$.

Definition 2.14 (Consistency (C))**.** *Given an IMC I, does it hold that $[\![I]\!] \neq \emptyset$?*

It turns out that, in the complexity theoretic sense, this problem is easy:

Theorem 2.13. *The problem C, of deciding whether a single IMC is consistent can be solved in polynomial time.*

Given an IMC $I = \langle Q, q_0, \varphi, A, V \rangle$, this problem can be solved by constructing a consistency relation over $Q \times Q$ (as if searching for a common implementation of Q with itself). Now there exists an implementation of I iff there exists a consistency relation containing (q_0, q_0). Obviously, this can be checked in polynomial time.

The fact that C can be decided in polynomial time casts an interesting light on the ability of IMCs to express inconsistency. On one hand, one can clearly specify inconsistent states in IMCs (simply by giving intervals for successor probabilities that cannot be satisfied by any distribution). On the other hand, this inconsistency appears to be local. It does not induce any global constraints on implementations; it does not affect consistency of other states. In this sense IMCs resemble modal transition systems (which at all disallow expressing inconsistency), and are weaker than *mixed transition systems* [48, 9]. Mixed transition systems relax the requirement of modal transition systems, not requiring that $(\rightarrow) \subseteq (\dashrightarrow)$. It is known that C is trivial for modal transition systems, but EXPTIME-complete for mixed transition systems [11]. Clearly, with a polynomial time C, IMCs cannot possibly express sophisticated global behaviour inconsistencies in the style of mixed transition systems, where the problem is much harder.

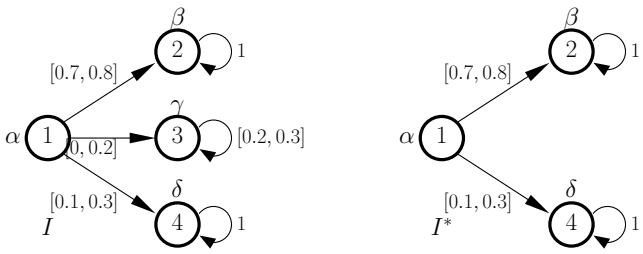
We conclude the section by observing that,given the IMC $I = \langle Q, q_0, \varphi, A, V \rangle$, with $Q = \{q_0, \ldots, q_n\}$, and a consistency relation $\mathcal{R} \subseteq Q \times Q$, it is possible to derive a *pruned* IMC $I^* = \langle Q^*, q_0^*, \varphi^*, A, V^* \rangle$ that contains no inconsistent states and accepts the same set of implementations as I. The construction of I^* is as follows: $Q^* = \{q \in Q | (q, q) \in \mathcal{R}\}$, $q_0^* = q_0$, $V^*(q^*) = V(q^*)$ for all $q^* \in Q^*$, and for all $q_1^*, q_2^* \in Q^*$, $\varphi^*(q_1^*)(q_2^*) = \varphi(q_1^*)(q_2^*)$.

Theorem 2.14. *Consider an IMC I and its pruned IMC I^*. We have $[\![I]\!] = [\![I^*]\!]$.*

Proof. By construction, the IMC I^* is a restriction of I. As a consequence, it is obvious that every implementation of I^* is also an implementation of I. Thus, $[\![I^*]\!] \subseteq [\![I]\!]$.

Moreover, if there exists an implementation $C = \langle P, p_0, \pi, A, V_P \rangle$ of $I = \langle Q, q_0, \varphi, A, V \rangle$ such that a state $p \in P$ statisfies a state $q \in Q$, it is obvious that there also exists a consistency relation $\mathcal{R}$ between I and I that includes q. Indeed there will exist a $\varrho \subseteq \mathsf{Distr}(Q \times Q)$ satisfying the constraints of q: $\pi(p)$. As a consequence, all the states belonging to a satisfaction relation for I will be states of Q^*. Thus all the satisfaction relations for I will also be satisfaction relations for I^*. Finally, we have $[\![I]\!] \subseteq [\![I^*]\!]$.

□



(a) IMC I (b) Pruned IMC I^*

Figure 2.8: An IMC and its pruned version

Example. *Consider the IMC I in Figure 2.8a. Building a consistency relation, we see that $(1,1)$ is in the relation witnessed by the distribution assigning probability 0.8 to $(2,2)$ and 0.2 to $(4,4)$. This probability distribution "avoids" the inconsistent state $(3,3)$; this state does not admit a probability distribution. Likewise, $(2,2)$ and $(3,3)$ are in the relation, witnessed by the distributions that gives probability 1 to $(2,2)$ and $(3,3)$, respectively.*
I^* is shown in Figure 2.8b.

In the next chapter, we present a more general algorithm for pruning, that does not involve constructing a consistency relation.

2.6 Conclusion and Related Work

This chapter provides new results for IMCs [86, 124, 34, 69] that is a specification formalism for probabilistic systems. We have studied the expressiveness and complexity of three refinement preorders for IMCs. The results are of interest as existing works on IMCs often use one of these preorders to compare specifications (for abstractions) [86, 89, 61]. We have established complexity bounds and decision procedures for these relations, closing a 20 years old left open problem in the seminal work on IMCs [86]. Finally, we have studied the common implementation problem that is to decide whether there exists an implementation that can match the requirements made by two or more specifications. Our solution is constructive in the sense that it can build such a common implementation.

Our results are robust with respect to simple variations of IMCs. For example sets of sets of propositions can be used to label states, instead of sets of propositions. This extends the power of the modeling formalism, which now can not only express abstractions over probability distributions, but also over possible state valuations. All our results easily translate to this setting without any changes to the complexity classes. Similarly an initial distribution, or even an interval constraint on the initial distribution, could be used instead of the initial state in IMCs (and MCs) without affecting the results. Finally, the setting we propose here only considers MCs and IMCs sharing the same sets of atomic propositions. This setting could easily extend to MCs / IMCs with distinct sets of atomic propositions, as will be done in the next chapter.

There exists many other specification formalisms for describing and analyzing stochastic systems; the list includes process algebras [79, 8, 103] or logical frameworks [73, 129]. We believe that IMCs is a good unification model for such formalisms. A logical representation is suited for conjunction, but nor for refinement and vice-versa for process algebra. As an

example, it is not clear how one can synthesize a MC (an implementation) that satisfies two Probabilistic Computation Tree Logic formulas.

IMCs and their extensions have been used as specification formalisms for stochastic systems. Unfortunately, as we already stated, IMCs are not expressive enough to capture many requirements of the compositional design methodology. This includes conjunction, parallel composition and disjunction. Conjunction allows solving several problems, notably common implementation. The solution promoted in this chapter provides a methodology in order to solve the common implementation problem for a set of IMCs without explicitly computing their conjunction. However, there are also problems usually solved using conjunction that the methodology presented in this chapter cannot solve. As an example, the problem of deciding whether the common implementations of two given specifications are also implementations of a third specification cannot be addressed using IMCs. In the same way, the methodology presented in this chapter does not allow us to reason on parallel composition and thus on incremental design. Disjunction, which allows to select between the requirements of many specifications remains an open problem. This operation is of importance for any procedure that would use IMCs as a symbolic representation for possibly infinite sets of MCs. Using such symbolic representation in a fixed point computation, one would have to decide whether the union of two IMCs is refined by another IMC. It is thus necessary to enrich the model of IMCs in order to obtain a specification theory that will be closed under both conjunction and parallel composition. This will be the subject of the next chapter.

Chapter 3

Constraint Markov Chains

3.1 Introduction

In the previous chapter, we have introduced IMCs that is a specification theory for stochastic systems. One of the main drawbacks of this model is that it is not closed under conjunction and composition, two requirements for a good interface theory. One way to approach this problem could be to work with two types of specifications: IMCs for refinement and structural composition, and a probabilistic logic such as PCTL [73] on which a logical conjunction is naturally defined. Such a solution is clearly non satisfactory. Indeed, it is not clear how one can synthesize a MC (an implementation) that satisfies two PCTL formulas. It is also not possible to structurally compose two PCTL formulas.

In this chapter, we promote a new approach to the problem: we develop *Constraint Markov Chains* (CMCs for short) as a new specification formalism that can be used as a foundation for component-based design of probabilistic systems. CMCs are a further extension of IMCs allowing rich constraints on the next-state probabilities from any state. Whereas linear constraints suffice for closure under conjunction, polynomial constraints are necessary for closure under parallel composition. We provide constructs for refinement, consistency checking, logical *and* structural composition of CMC specifications – all indispensable ingredients of a compositional design methodology.

The notions of satisfaction and strong/weak refinements for CMCs conservatively extend similar notions for IMCs [61, 86], presented in Chapter 2. We characterize these relations in terms of implementation set inclusion. In particular, in the main theorem, we prove that for deterministic CMCs weak and strong refinements are complete with respect to implementation set inclusion. In addition, we provide a construction, which for any CMC S returns a deterministic CMC $\varrho(S)$ containing the models of S. Refinement relations are not complete for non-deterministic CMCs, but one can show that the weak refinement is more likely to coincide with implementation set inclusion in such a context. We show that refinement between CMCs with polynomial constraints can be decided in essentially single exponential time.

In CMCs, each state is also labelled with a set of subsets of atomic propositions. Those propositions represent properties that should be satisfied by the implementation. The idea being that the satisfaction relation ensures that an implementation matches at least one of the subsets. This allows the specification to make additional assumptions on the behaviors of the implementation. Hence, at the level of specification, our model presents choices on subsets of actions. However these choices are independent from the probabilistic ones in the sense that

any CMC whose states are labelled with a set of subsets of atomic propositions can be turned to an equivalent (in terms of set of implementations) CMC whose states are labeled with a single subset of atomic propositions. There, choices between the subsets of actions disappear. It is thus not surprising that our notion of parallel composition is following the widely accepted *principle of separation of concerns*. The idea is to separate parallel composition of probability distributions from synchronization on sets of actions. This separation can be found in probabilistic specification theories that have probabilistic automata as an underlying semantic model [121, 72, 87, 79]. In fact, we show how probabilistic automata can be represented as CMCs, and how the traditional notions of parallel composition on such model can be derived in our framework with precongruence properties obtained for free. This latter result shows that CMCs capture computational structure of known models and operators, laying down a basis for studying shared properties of many probabilistic automata based languages. As already mentioned, we exemplify this by showing how precongruence properties for composition of probabilistic automata and known refinements can be obtained by reductions to CMCs.

We also compare the expressivity of the operation of parallel composition and the one of conjunction. It turns out that for independent sets of valuations, composition refines conjunction, but the opposite is not true. This result allows to isolate a class of CMCs and CMCs operations that is closed under linear constraints. Finally, we also show that CMCs are not closed under disjunction and we discuss the problem of deciding whether a CMC is universal.

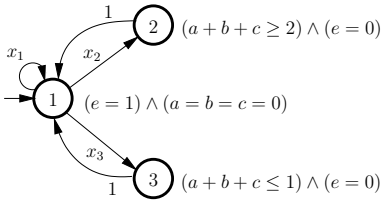
Structure of the chapter. In Section 3.2, we introduce the concept of CMCs and a satisfaction relation with respect to Markov Chains. Consistency, refinement and conjunction are discussed in Section 3.3. Structural composition is introduced in Section 3.4 where we also compare the operation to conjunction. Disjunction and universality are discussed in Section 3.5. In Section 3.6, we introduce deterministic CMCs and show that, for this class of CMCs, strong and weak refinements coincide with inclusion of implementation sets. Section 3.7 discusses the class of polynomial CMCs, which is the smallest class of CMCs closed under all the compositional design operations. Section 3.9 concludes the chapter with related and future work.

3.2 Constraint Markov Chains

Since we aim at building a compositional specification theory, it may be of interest to consider structures with possibly different sets of atomic propositions. In order to take care of this possibility, we propose the following operations on sets of atomic propositions. Let A, B be sets of propositions with $A \subseteq B$. The *restriction of* $W \subseteq B$ *to* A is given by $W{\downarrow_A} \equiv W \cap A$. If $T \subseteq 2^B$, then $T{\downarrow_A} \equiv \{W{\downarrow_A} \mid W \in T\}$. For $W \subseteq A$ define the *extension of* W *to* B as $W{\uparrow^B} \equiv \{V \subseteq B \mid V{\downarrow_A} = W\}$, so the set of sets whose restriction to A is W. Lift it to sets of sets as follows: if $T \subseteq 2^A$ then $T{\uparrow^B} \equiv \{W \subseteq B \mid W{\downarrow_A} \in T\}$. Let $M, \Delta \in [0,1]^{n \times k}$ be two matrices and $x \in [0,1]^{1 \times k}$ be a vector. We write M_{ij} for the cell in ith row and jth column of M, M_p for the pth row of M, and x_i for the ith element of x. Finally, Δ is a *correspondence matrix* iff $0 \leq \sum_{j=1}^{k} \Delta_{ij} \leq 1$ for all $1 \leq i \leq n$.

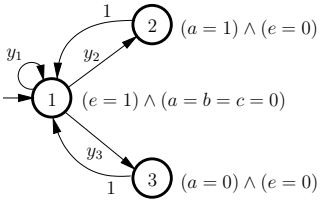
We recall the definition for Markov Chains, allready introduced in Chapter 2. Markov Chains act as models in our specification formalism.

Definition 3.1 (Markov Chain)**.** $C = \langle Q, o, M, A, V \rangle$ *is a* Markov Chain *if* Q *is a finite set of states containing the initial state* o*,* A *is a set of atomic propositions,* $V : Q \to 2^A$ *is a state*



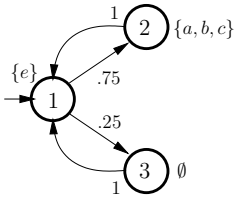
$$\varphi_1(1)(x) \equiv (x_1 = 0) \wedge (x_2 \geq 0.7) \wedge (x_2 + x_3 = 1)$$

(a) CMC S_1, the customer specification of the optical relay

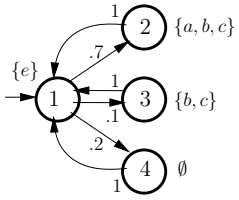


$$\varphi_2(1)(y) \equiv (y_1 = 0) \wedge (y_3 \geq 0.2) \wedge (y_2 + y_3 = 1)$$

(b) CMC S_2, The manufacturer specification of the relay



(c) Markov Chain C_1 satisfying S_1 and S_2



(d) Markov Chain C_2 satisfying S_1 and S_2

Figure 3.1: Two specifications (CMCs) and two implementations (MCs) of an optic relay

valuation. Assuming that the states in Q are ordered, i.e. $Q = \{q_1, \ldots, q_n\}$, $M \in [0,1]^{n \times n}$ is a probability transition matrix: $\sum_{j=1}^{n} M_{ij} = 1$ for $i = 1, \ldots, n$. The cell M_{ij} defines the probability of the transition from state q_i to state q_j.

Like in Chapter 2, our formalism does not allow "sink states", i.e. states with no outgoing transition. However, in order to avoid clutter in the figures, we sometimes represent states with no outgoing transitions. They must be interpreted as states with a self-loop of probability 1. We now introduce *Constraint Markov Chains* (CMCs for short), a finite representation for a possibly infinite set of MCs. Roughly speaking, CMCs generalize MCs in that, instead of specifying a concrete transition matrix, they only constrain probability values in the matrix. Constraints are modelled using a *characteristic function*, which for a given source state and a distribution of probabilities of leaving the state evaluates to 1 iff the distribution is permitted by the specification. Similarly, instead of a concrete valuation function for each state, a *constraint on valuations* is used. Here, a valuation is permitted iff it is contained in the set of admissible valuations of the specification.

Definition 3.2 (Constraint Markov Chain)**.** *A* Constraint Markov Chain *is a tuple $S = \langle Q, o, \varphi, A, V \rangle$, where Q is a finite set of states containing the initial state o, A is a set of atomic propositions, $V : Q \to 2^{2^A}$ is a set of admissible state valuations. Assuming that the states in Q are ordered, i.e. $Q = \{q_1, \ldots, q_k\}$, $\varphi : Q \to [0,1]^k \to \{0,1\}$ is a* constraint function *such that if $\varphi(j)(x) = 1$ then the x vector is a probability distribution: $x \in [0,1]^k$ and $\sum_{i=1}^{k} x_i = 1$.*

As introduced in Chapter 2, *Interval Markov Chains* (IMCs for short) [86] are CMCs whose constraint functions are represented by intervals, so for all $1 \leq i \leq k$ there exist constants α_i, β_i such that $\varphi(j)(x) = 1$ iff $\forall 1 \leq i \leq k,\ x_i \in [\alpha_i, \beta_i]$.

Example. *Two parties, a customer and a vendor, are discussing a design of a relay for an optical telecommunication network. The relay is designed to amplify an optic signal transmitted over a long distance over an optic fiber. The relay should have several modes of operation, modelled by four dynamically changing properties and specified by atomic propositions* a*,* b*,* c*, and* e*:*

Atomic propositions in the optic relay specifications		
a	ber $\leq 10^{-9}$	bit error rate lower than 1 per billion bits transmitted
b	br > 10Gbits/s	The bit rate is higher than 10 Gbits/s.
c	$P < 10$W	Power consumption is less than 10 W.
e	Standby	The relay is not transmitting.

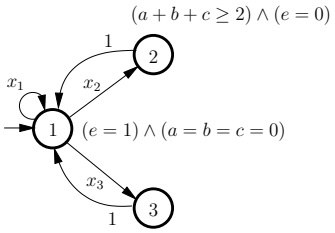
The customer presents CMC S_1 *(Figure 3.1a) specifying the admissible behaviour of the relay from their point of view. States are labelled with formulas characterizing sets of valuations. For instance, "*$(a + b + c \geq 2) \wedge (e = 0)$*" at state* 2 *of* S_1 *represents* $V_1(2) = \{\{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$*, where* a*,* b*,* c*, and* e *range over Booleans. State 1 specifies a standby mode, where no signal is emitted and only marginal power is consumed. State 2 is the high power mode, offering a high signal/noise ratio, and hence a high bitrate and low error rate, at the expense of a high power consumption. State 3 is the low power mode, with a low power consumption, low bitrate and high error rate. The customer prescribes that the probability of the high power mode (state 2) is higher than* 0.7*. The vendor replies with CMC* S_2 *(Figure 3.1b), which represents possible relays that they can build. Because of thermal limitations, the low power mode has a probability higher than* 0.2*.*

A state u of S is (directly) *reachable* from a state i if there exists a probability distribution $x \in [0, 1]^k$ with a nonzero probability x_u, which satisfies $\varphi(i)(x)$.

We relate CMC specifications to MCs implementing them, by extending the definition of satisfaction presented in Chapter 2 to observe the valuation constraints and the full-fledged constraint functions. Crucially, like in Chapter 2 and in [86], we abstract from syntactic structure of transitions—a single transition in the implementation MC can contribute to the satisfaction of more than one transition in the specification, by distributing its probability mass against several transitions. Similarly many MC transitions can contribute to the satisfaction of just one specification transition. The concept is strictly the same as defined for IMCs in Chapter 2. Again, this definition is slightly different but strictly equivalent to the definition used in [86]. Unlike in [86], our definition is a particular case of the refinement relations that will be presented later in this section.

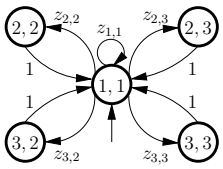
Definition 3.3 (Satisfaction Relation)**.** *Let* $C = \langle \{1, \ldots, n\}, o_C, M, A_C, V_C \rangle$ *be a MC and* $S = \langle \{1, \ldots, k\}, o_S, \varphi, A_S, V_S \rangle$ *be a CMC with* $A_S \subseteq A_C$*. Then* $\mathcal{R} \subseteq \{1, \ldots, n\} \times \{1, \ldots, k\}$ *is a* satisfaction relation *between states of* C *and* S *iff whenever* $p \,\mathcal{R}\, u$ *then*

1. *their valuations are compatible:* $V_C(p)\downarrow_{A_S} \in V_S(u)$*, and*
2. *there exists a correspondence matrix* $\Delta \in [0, 1]^{n \times k}$ *such that*
   - *for all* $1 \leq p' \leq n$ *with* $M_{pp'} \neq 0$*,* $\sum_{j=1}^{k} \Delta_{p'j} = 1$*,*
   - $\varphi(u)(M_p \times \Delta)$ *holds, and*
   - *if* $\Delta_{p'u'} \neq 0$ *then* $p' \,\mathcal{R}\, u'$*.*



$$\varphi_4(1)(x) \equiv (x_1 = 0) \wedge (x_2 \geq 0.7)$$
$$\wedge (x_3 \geq 0.2) \wedge (x_2 + x_3 = 1)$$

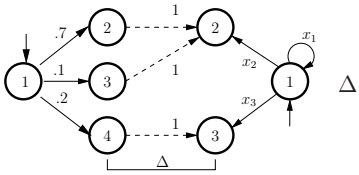
(a) CMC S_4 generalizing S_3, so $S_3 \preceq S_4$



$$\varphi_3(1,1)(Z) \equiv (\forall j.\ z_{1,j} = 0) \wedge (z_{2,2} + z_{2,3} \geq 0.7)$$
$$\wedge\ (z_{2,2}+z_{2,3}+z_{3,2}+z_{3,3} = 1) \wedge\ (\forall i.\ z_{i,1} = 0)$$
$$\wedge\ (z_{2,3} + z_{3,3} \geq 0.2)$$

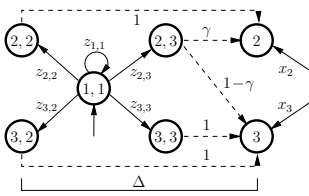
(b) $S_3 = S_1 \wedge S_2$. Constraints on propositions, pairwise conjunctions of constraints of S_1 and S_2, are left out to avoid clutter

Figure 3.2: Examples of refinement and conjunction for CMCs



$$\Delta = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a) Correspondence for initial states of C_2 and S_1



$$\Delta = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \gamma & 1-\gamma \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) Weak refinement for initial states of S_3 and S_4

Figure 3.3: Examples of refinement and satisfaction for CMCs

We write $C \models S$ iff there exists a satisfaction relation relating o_C and o_S, and call C an *implementation* of S. The set of all implementations of S is given by $[\![S]\!] \equiv \{C \mid C \models S\}$. Rows of Δ that correspond to reachable states of C always sum up to 1. This is to guarantee that the entire probability mass of implementation transitions is allocated. For unreachable states, we leave the corresponding rows in Δ unconstrained. C may have a richer alphabet than S, in order to facilitate abstract modelling: this way an implementation can maintain local information using internal variables. Algorithms to decide satisfaction are particular cases of algorithms to decide *refinement* between CMCs. See the next section.

Example. *We illustrate the concept of correspondence matrix between Specification S_1 (given in Figure 3.1a) and Implementation C_2 (given in Figure 3.1d). The CMC S_1 has three outgoing transitions from state 1 but, due to constraint function in 1, the transition labelled with x_1 cannot be taken (the constraint implies $x_1 = 0$). The probability mass going from state 1 to states 2 and 3 in C_2 corresponds to the probability allowed by S_1 from its state 1 to its state 2; The redistribution is done with the help of the matrix Δ given in Figure 3.3a. The ith column in Δ describes how big fraction of each transition probability (for transitions leaving 1) is associated with probability x_i in S_1. Observe that the constraint function $\varphi_1(1)(0, 0.8, 0.2) = \varphi_1(1)((0, 0.7, 0.1, 0.2) \times \Delta)$ is satisfied.*

CMC semantics follows the Markov Decision Process (MDP) tradition [124, 34]. The MDP semantics is typically opposed to the Uncertain Markov Chain semantics, where the probability distribution from each state is fixed a priori.

States of CMCs are labeled with a set of subsets of atomic propositions. A single set of propositions represents properties that should be satisfied by the implementation. A set of sets models a choice of properties, with the idea being that the satisfaction relation ensures that an implementation matches at least one of the subsets. This allows the specification to make additional assumptions on the behaviors of the implementation. For an implementation, in each state the discrete choice of proposition set and the probabilistic choice of successor are independent.

It turns out that any CMC whose states are labelled with a set of subsets of atomic propositions can be turned into an equivalent (in terms of sets of implementations) CMC whose states are labeled with sets that contains a single subset of atomic propositions. Hence working with sets of subsets of valutations is a kind of modeling sugar that can be removed with a transformation to the *single valuation normal form*.

Definition 3.4. *We say that a CMC $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ is in* Single Valuation Normal Form *if all its admissible valuation sets are singletons ($|V(i)| = 1$ for each $1 \leq i \leq k$).*

More precisely every consistent CMC with at most one admissible valuation in the initial state can be transformed into the normal form preserving its implementation set by using the following polynomial algorithm.

The normalization algorithm basically separates each state u with m possible valuations into m states $u_1, \ldots, u_m$, each with a single admissible valuation. Then the constraint function is adjusted, by substituting sums of probabilities going to the new states in place of the old probabilities targeting u. The transformation is local and syntax based. It can be performed in polynomial time and it only increases the size of the CMC polynomially. We will write $\mathcal{N}(S)$ for a result of normalization of S.

Definition 3.5 (Normalization)**.** *Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a CMC. If there exists a function $\mathcal{N} : \{1, \ldots, k\} \to 2^{\{1,\ldots,m\}}$ such that*

(a) $\{1, \ldots, m\} = \cup_{i \in \{1,\ldots,k\}} \mathcal{N}(i)$*;*

(b) For all $1 \leq i \neq j \leq k$*,* $\mathcal{N}(i) \cap \mathcal{N}(j) = \emptyset$*;*

(c) $\forall 1 \leq i \leq k$*,* $|\mathcal{N}(i)| = |V(i)|$*;*

If, moreover, $|V(o)| = 1$*, the normalization of S is the CMC $\mathcal{N}(S) = \langle \{1, \ldots, m\}, o', \varphi', A, V' \rangle$ such that $\mathcal{N}(o) = \{o'\}$ and*

1. $\forall 1 \leq j \leq m$*,* $|V'(j)| = 1$*;*
2. $\forall 1 \leq i \leq k$*,* $V(i) = \cup_{u \in \mathcal{N}(i)} V'(u)$*;*
3. $\forall 1 \leq i \leq k, \forall u, v \in \mathcal{N}(i)$*,* $u \neq v \iff V'(u) \neq V'(v)$*;*
4. $\forall 1 \leq j \leq m$*,*

$$\varphi'(j)(x_1, \ldots x_m) = \varphi(\mathcal{N}^{-1}(j))(\sum_{u \in \mathcal{N}(1)} x_u, \ldots, \sum_{u \in \mathcal{N}(k)} x_u).$$



(a) CMC S before normalisation

(b) CMC $\mathcal{N}(S)$

Figure 3.4: Illustration of normalization.

By construction, $\mathcal{N}(S)$ is in single valuation normal form. Moreover, if S is consistent, then a function $\mathcal{N}$ satisfying the conditions above exists.

The following example illustrates the normalization algorithm.

Example. *Consider the CMC $S = \langle\{1,2,3,4\}, 1, \varphi, \{a,b,c,d,e\}, V\rangle$ given in Figure 3.4a. Since states 2 and 3 have two subsets of atomic propositions, S is not in single valuation normal form. Define the following normalisation function*

$$\mathcal{N} : \begin{cases} 1 \to \{1\} \\ 2 \to \{2, 2'\} \\ 3 \to \{3, 3'\} \\ 4 \to 4 \end{cases}$$

The result of applying the normalisation algorithm to S is the CMC $\mathcal{N}(S) = \langle\{1,2,2',3,3',4\}, 1, \varphi', \{a,b,c,d,e\}, V'\rangle$ given in Figure 3.4b. Following the algorithm, States 2 and 3 of S have been each separated into two states with a single subset of atomic propositions. The constraint function of state 1 uses $y_2 + y_{2'}$ and $y_3 + y_{3'}$ instead of x_2 and x_3 respectively.

As expected, the above algorithm builds a CMC in single valuation normal form that has the exact same set of implementations as the initial CMC.

Theorem 3.1. *Let $S = \langle\{1,\dots k\}, o, \varphi, A, V\rangle$ be a consistent CMC. If $|V(o)| = 1$, then for all MC C, we have $C \models S \iff C \models \mathcal{N}(S)$.*

Proof. Let $S = \langle\{1,\dots,k\}, o, \varphi, A, V\rangle$ be a consistent CMC such that $|V(o)| = 1$. Let $S' = \mathcal{N}(S) = \langle\{1,\dots,m\}, o', \varphi', A, V'\rangle$ and $\mathcal{N} : \{1,\dots,k\} \to 2^{\{1,\dots,m\}}$ the associated function.

$\Rightarrow$ Let $C = \langle\{1,\dots,n\}, o_C, M, A_C, V_C\rangle$ be a MC such that $C \models S$. Let $\mathcal{R}$ be the associated satisfaction relation. Let $\mathcal{R}' \subseteq \{1,\dots,n\} \times \{1,\dots,m\}$ such that $p\,\mathcal{R}'\,u \iff V_C(p) \in V'(u)$ and $p\,\mathcal{R}\,\mathcal{N}^{-1}(u)$. We will show that $\mathcal{R}'$ is a satisfaction relation. Let p, u such that $p\,\mathcal{R}'\,u$.

1. By definition, we have $V_C(p) \in V'(u)$.

2. We have $p \mathcal{R} \mathcal{N}^{-1}(u)$. Let $\Delta \in [0,1]^{n \times k}$ be the associated correspondence matrix. Define $\Delta' \in [0,1]^{n \times m}$ such that $\Delta'_{q,v} = \Delta_{q,\mathcal{N}^{-1}(v)}$ if $V_p(q) \in V'(v)$ and 0 else. As every coefficient of Δ appears once and only once in the same row of Δ', it is clear that Δ' is a correspondence matrix. Moreover,
   (a) If q is such that $M_{pq} \neq 0$, then $\sum_{j=1}^{m} \Delta'_{q,j} = \sum_{i=1}^{k} \Delta_{q,i} = 1$;
   (b) For all $1 \leq i \leq k$, $\sum_{j \in \mathcal{N}(i)} ([M_p \times \Delta']_j) = [M_p \times \Delta]_i$. As a consequence, $\varphi'(u)(M_p \times \Delta') = \varphi(\mathcal{N}^{-1}(u))(M_p \times \Delta)$ holds.
   (c) If q, v are such that $\Delta'_{q,v} \neq 0$, then $\Delta_{q,\mathcal{N}^{-1}(v)} \neq 0$ and $V_C(q) \in V'(v)$, thus $q \mathcal{R}' v$.

Finally, $\mathcal{R}'$ is a satisfaction relation. It is easy to see that $o_p \mathcal{R}' o'$. As a consequence, we have $C \models \mathcal{N}(S)$.

$\Leftarrow$ Let $C = \langle \{1, \ldots, n\}, o_C, M, A_C, V_C \rangle$ be a MC such that $C \models \mathcal{N}(S)$. Let $\mathcal{R}$ be the associated satisfaction relation. Let $\mathcal{R}' \subseteq \{1, \ldots, n\} \times \{1, \ldots, k\}$ such that $p \mathcal{R}' u \iff \exists j \in \mathcal{N}(u)$ s.t. $p \mathcal{R} j$. We will show that $\mathcal{R}'$ is a satisfaction relation. Let p, u such that $p \mathcal{R}' u$.

1. We have $V_C(p) \in V(u) = \cup_{j \in \mathcal{N}(u)} V'(j)$.
2. Let $j \in \mathcal{N}(u)$ such that $p \mathcal{R} j$, and let $\Delta \in [0,1]^{n \times m}$ be the associated correspondence matrix. Define $\Delta' \in [0,1]^{n \times k}$ such that $\Delta'_{q,v} = \sum_{i \in \mathcal{N}(v)} \Delta_{q,i}$. It is clear that for all q, $\sum_{v=1}^{k} \Delta'_{q,v} = \sum_{r=1}^{m} \Delta_{qr}$. Thus Δ' is a correspondence matrix. Moreover,
   (a) If q is such that $M_{pq} \neq 0$, then $\sum_{i=1}^{k} \Delta'_{q,i} = \sum_{r=1}^{m} \Delta_{q,r} = 1$;
   (b) For all $1 \leq i \leq k$, $[M_p \times \Delta']_i = \sum_{r \in \mathcal{N}(i)} ([M_p \times \Delta]_r)$. As a consequence, $\varphi(u)(M_p \times \Delta) = \varphi'(j)(M_p \times \Delta')$ holds.
   (c) If q, v are such that $\Delta'_{q,v} \neq 0$, then there exists $r \in \mathcal{N}(v)$ such that $\Delta_{q,r} \neq 0$, thus $q \mathcal{R}' v$.

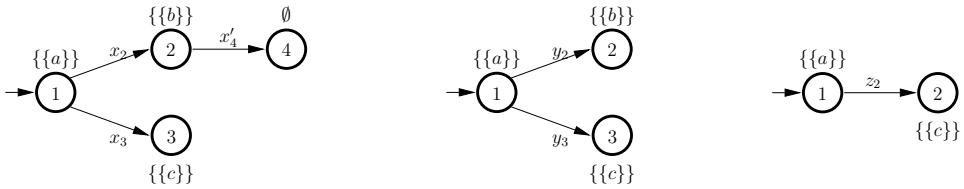
Finally, $\mathcal{R}'$ is a satisfaction relation. It is easy to see that $o_C \mathcal{R}' o$. As a consequence, we have $C \models S$.

□

Crucially, note that this algorithm cannot be applied to IMCs. Indeed, normalization introduces a linear complexity in the constraint functions, as can be seen in Item 4 of Definition 3.5. Finally, normalization obviously preserves determinism.

3.3 Consistency, Refinement and Conjunction

In this section, we study the consistency problem that is to decide whether a CMC admits at least an implementation. Then we propose algorithms in order to check refinement and implementation set inclusion. Finally, we propose a methodology to compute the conjunction of two CMCs.



(a) CMC S before pruning (b) CMC $\beta(S)$ (c) CMC $\beta^*(S)$

Figure 3.5: Illustration of the pruning algorithm.

3.3.1 Consistency

A CMC S is *consistent* if it admits at least one implementation. We now discuss how to decide consistency. A state u of S is *valuation consistent* iff $V(u) \neq \emptyset$; it is *constraint consistent* iff there exists a probability distribution vector $x \in [0,1]^{1\times k}$ such that $\varphi(u)(x) = 1$. It is easy to see that if *each state* of S is both valuation and constraint consistent then S is also consistent. However, inconsistency of a state does not imply inconsistency of the specification. Indeed, an inconsistent state could be made unreachable by forcing the probabilities to reach it to zero. The operations presented later in this chapter may introduce inconsistent states, leaving a question if a resulting CMC is consistent. In order to decide whether S is inconsistent, state inconsistencies are propagated throughout the entire state-space using a *pruning operator* β that removes inconsistent states from S. The result $\beta(S)$ is a new CMC, which may still contain some inconsistent states. We define β formally.

Definition 3.6 (Pruning operator (β))**.** *Let $S = \langle\{1,\ldots,k\}, o, \varphi, A, V\rangle$. The pruning operator β is defined as follows:*

- *If o is locally inconsistent then let $\beta(S) = \emptyset$.*
- *If S does not contain locally inconsistent states then $\beta(S) = S$.*
- *Else proceed in two steps. First for $k' < k$ define a function $\nu : \{1,\ldots,k\} \to \{\bot, 1, \ldots, k'\}$, which will remove inconsistent states. All inconsistent states are mapped to $\bot$. For all $1 \leq i \leq k$ take $\nu(i) = \bot$ iff $[(V(i) = \emptyset) \vee (\forall x \in [0,1]^k,\ \varphi(i)(x) = 0)]$. All remaining states are mapped injectively into $\{1,\ldots,k'\}$: $\nu(i) \neq \bot \implies \forall j \neq i,\ \nu(j) \neq \nu(i)$. Then let $\beta(S) = \langle\{1,\ldots,k'\}, \nu(o), \varphi', A, V'\}$, where $V'(i) = V(\nu^{-1}(i))$ and for all $1 \leq j \leq k'$ the constraint $\varphi'(j)(y_1,\ldots,y_{k'})$ is: $\exists x_1,\ldots,x_k$ such that*

$$\big[\nu(q) = \bot \Rightarrow x_q = 0\big] \wedge \big[\forall 1 \leq l \leq k' :\ y_l = x_{\nu^{-1}(l)}\big] \wedge \big[\varphi(\nu^{-1}(j))(x_1,\ldots,x_k)\big]$$

The constraint makes the inconsistent states unreachable, and then $\bot$ is dropped as a state.

The operator is applied iteratively, until a fixpoint is reached. S is consistent iff the resulting CMC $\beta^*(S)$ contains at least one state. The following example illustrates the pruning algorithm.

Example. *Consider the CMC $S = \langle\{1,2,3,4\}, 1, \varphi, \{a,b,c\}, V\rangle$ given in Figure 3.5a. Define φ as following : $\varphi(1)(x) = (x_2 \leq 0.3) \wedge (x_2 + x_3 = 1)$, $\varphi(2)(x') = (x'_4 = 1)$. The constraint of States 3 and 4 are not relevant for this example.*

State 4 *is obviously not valuations consistent. States* $1, 2$ *and* 3 *are all valuations and constraint consistent. As a consequence, the first step of the pruning algorithm will only mark state* 4 *as inconsistent. For this, define the following function*

$$\nu : \begin{cases} 1 \to 1 \\ 2 \to 2 \\ 3 \to 3 \\ 4 \to \bot \end{cases}$$

Then define $\beta(S) = \langle\{1,2,3\}, 1, \varphi', \{a,b,c\}, V'\rangle$ *such that, after reduction,* $\varphi'(1)(y) = (y_2 \leq 0.3) \wedge (y_2 + y_3 = 1)$*, and* $\varphi'(2)(y') = \exists x'_4,\ (x'_4 = 0) \wedge (x'_4 = 1)$*.* $\beta(S)$ *is given in Figure 3.5b.*

Obviously, State 2 *of* $\beta(S)$ *is now constraint inconsistent:* $\varphi'(2)(y')$ *is not satisfiable. We thus apply another time the pruning operator* β *in order to remove State* 2*. This time we obtain a consistent CMC* $\beta^*(S)$*, given in Figure 3.5c.*

The fixpoint of β, and thus the entire consistency check, can be computed using a quadratic number of state consistency checks. The complexity of each check depends on the constraint language chosen. The following proposition shows that pruning preserves the set of implementations.

Proposition 3.2. *Let* $S = \langle\{1,\ldots,k\}, o, \varphi, A, V\rangle\}$ *be a CMC and* $\beta^*(S) = \lim_{n\to\infty} \beta^n(S)$ *be the fixpoint of* β*. For any MC* C*, we have (1)* $C \models S \iff C \models \beta(S)$ *and (2)* $[\![S]\!] = [\![\beta^*(S)]\!]$*.*

Proof. Let $S = \langle\{1,\ldots,k\}, o, \varphi, A, V\rangle$ be a CMC (with at least an inconsistent state) and $C = \langle\{1,\ldots,n\}, o_C, M, A_C, V_C\rangle$ be a MC. Let $S' = \langle\{1,\ldots,k'\}, o', \varphi', A, V'\rangle = \beta(S)$. If $\beta(S)$ is empty, then both S and $\beta(S)$ are inconsistent.

Consider a function ν for removing inconsistent states (one exists because there are inconsistent states), such that $k' < k$ and for all $1 \leq i \leq k$, $\nu(i) = \bot \iff [(V(i) = \emptyset) \vee (\forall x \in [0,1]^k,\ \neg\varphi(i)(x))]$ and $\nu(i) \neq \bot \Rightarrow \forall j \neq i,\ \nu(j) \neq \nu(i)$. We first prove that $C \models S \iff C \models \beta(S)$.

$\Rightarrow$ Suppose that $C \models S$. Then there exists a satisfaction relation $\mathcal{R}$ such that $o_C\, \mathcal{R}\, o$. Define the relation $\mathcal{R}' \subseteq \{1,\ldots,n\} \times \{1,\ldots,k'\}$ such that $p\, \mathcal{R}'\, v$ iff there exists $u \in \{1,\ldots,k\}$ such that $p\, \mathcal{R}\, u$ and $\nu(u) = v$. It is clear that $o_C\, \mathcal{R}'\, o'$. We prove that $\mathcal{R}'$ is a satisfaction relation. Let p, u, v such that $p\, \mathcal{R}\, u$ and $\nu(u) = v$.

- As $\nu(u) \neq \bot$, we have by definition that $V'(v) = V(u)$, thus $V_C(p)\downarrow_A \in V'(v)$.
- Let $\Delta \in [0,1]^{n\times k}$ be the correspondence matrix witnessing $p\, \mathcal{R}\, u$. Let $\Delta' \in [0,1]^{n\times k'}$ such that $\Delta'_{qw} = \Delta_{q\nu^{-1}(w)}$. It is clear that Δ' is a correspondence matrix. We first show that

$$\forall u' \in \{1,\ldots,k\}, (\nu(u') = \bot) \Rightarrow (\forall q \in \{1,\ldots,n\},\ \Delta_{qu'} = 0). \tag{3.1}$$

  Let $u' \in \{1,\ldots,k\}$ such that $\nu(u') = \bot$, and suppose that there exists $q \in \{1,\ldots,n\}$, $\Delta_{qu'} \neq 0$. As Δ is a correspondence matrix, we have $q\, \mathcal{R}\, u'$. Thus

$V_C(q) \downarrow_A \in V(u')$, which means that $V(u') \neq \emptyset$, and there exists Δ'' such that $\varphi(u')(M_q \times \Delta'')$. Thus, there exists $x \in [0,1]^{1\times k}$ such that $\varphi(u')(x)$. As a consequence, we cannot have $\nu(u') = \bot$, which is a contradiction, thus (3.1).

We now prove that $\mathcal{R}'$ satisfies the axioms of a satisfaction relation.

1. Let $p' \in \{1,\dots,n\}$ such that $M_{pp'} \neq 0$. This implies, by definition, that $\sum_{j=1}^{k} \Delta_{p'j} = 1$. We have $\sum_{j=1}^{k'} \Delta'_{p'j} = \sum_{r\in\{1,\dots,k\} \mid \nu(r)\neq\bot} \Delta_{p'r}$.
   By (3.1), $\sum_{r\in\{1,\dots,k\} \mid \nu(r)\neq\bot} \Delta_{p'r} = \sum_{r=1}^{k} \Delta_{p'r} = 1$.
2. Let $y = M_p \times \Delta' \in [0,1]^{1\times k'}$ and $x = M_p \times \Delta \in [0,1]^{1\times k}$. We know that $\varphi(u)(x)$ holds. Moreover, by (3.1), if $\nu(q) = \bot$, then $x_q = 0$, and for all $l \in \{1,\dots,k'\}$, $y_l = x_{\nu^{-1}(l)}$. Clearly, this implies that $\varphi'(v)(M_p \times \Delta')$ holds.
3. Let $p', v' \in \{1,\dots,n\} \times \{1,\dots,k'\}$ such that $\Delta'_{p'v'} \neq 0$. We have $\Delta'_{p'v'} = \Delta_{p'\nu^{-1}(v')} \neq 0$, thus there exists $u' \in \{1,\dots,k\}$ such that $p' \,\mathcal{R}\, u'$ and $\nu(u') = v'$. Finally $p' \,\mathcal{R}'\, v'$.

Finally, $\mathcal{R}'$ is a satisfaction relation such that $o_C \,\mathcal{R}'\, o'$, thus $C \models \beta(S)$.

$\Leftarrow$ Conversely, the reasoning is the same, except that we now build Δ from Δ' saying that $\Delta_{qv} = 0$ if $\nu(v) = \bot$ and $\Delta_{qv} = \Delta'_{q\nu(v)}$ otherwise.

We have proved that β is implementations-conservative, thus the fixpoint of β verifies the same property.

□

3.3.2 Refinement

Comparing specifications is central to stepwise design methodologies. Systematic comparison enables simplification of specifications (abstraction) and adding details to specifications (elaboration). Usually specifications are compared using a *refinement* relation. Roughly, if S_1 refines S_2, then any model of S_1 is also a model of S_2.

We will now introduce two notions of refinement for CMCs that extend two well known refinements for IMCs [86, 61], that we introduced in Chapter 2. We not only generalize these refinements, but, unlike [86, 61], we also characterize them in terms of implementation set inclusion – also called *thorough refinement* – and computational complexity. We will prove that the ordering we obtain between the three refinement relations is the same as the one we obtained in Chapter 2. We then propose algorithms to compute these refinements for CMCs.

The strong refinement between IMCs, by Jonsson and Larsen [86], extends to CMCs in the following way:

Definition 3.7 (Strong Refinement)**.** *Let $S_1 = \langle \{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle \{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be CMCs with $A_2 \subseteq A_1$. A relation $\mathcal{R} \subseteq \{1,\dots,k_1\} \times \{1,\dots,k_2\}$ is a* strong refinement relation *between states of S_1 and S_2 iff whenever $v \,\mathcal{R}\, u$ then*

1. *their valuations are compatible: $V_1(v)\downarrow_{A_2} \subseteq V_2(u)$, and*
2. *there exists a correspondence matrix $\Delta \in [0,1]^{k_1\times k_2}$ such that, for all probability distribution vectors $x \in [0,1]^{1\times k_1}$, if $\varphi_1(v)(x)$ holds then*

- $x_i \neq 0 \Rightarrow \sum_{j=1}^{k_2} \Delta_{ij} = 1$,
- $\varphi_2(u)(x \times \Delta)$ *holds, and*
- *if* $\Delta_{v'u'} \neq 0$ *then* $v' \mathcal{R} u'$.

We say that CMC S_1 *strongly refines CMC* S_2 *, written* $S_1 \preceq_S S_2$*, iff* $o_1 \mathcal{R} o_2$ *for some strong refinement relation* $\mathcal{R}$*.*

Strong refinement imposes a "fixed-in-advance" correspondence matrix regardless of the probability distribution satisfying the constraint function. In contrast, the *weak refinement*, which generalizes the one proposed in [61] for IMCs, allows choosing a different correspondence matrix for each probability distribution satisfying the constraint:

Definition 3.8 (Weak Refinement)**.** *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs with* $A_2 \subseteq A_1$*. The relation* $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ *is a* weak refinement relation *iff whenever* $v \mathcal{R} u$ *then*

1. *their constraints are compatible:* $V_1(v)\downarrow_{A_2} \subseteq V_2(u)$*, and*
2. *for any distribution* $x \in [0,1]^{1 \times k_1}$ *satisfying* $\varphi_1(v)(x)$*, there exists a matrix* $\Delta \in [0,1]^{k_1 \times k_2}$ *such that*
   - *for all* S_1 *states* $1 \leq i \leq k_1$*,* $x_i \neq 0 \implies \sum_{j=1}^{k_2} \Delta_{ij} = 1$,
   - $\varphi_2(u)(x \times \Delta)$ *holds, and*
   - *If* $\Delta_{v'u'} \neq 0$*, then* $v' \mathcal{R} u'$.

We say that CMC S_1 *(weakly) refines CMC* S_2*, written* $S_1 \preceq S_2$*, iff* $o_1 \mathcal{R} o_2$ *for some weak refinement relation* $\mathcal{R}$*.*

Example. *Figure 3.3b illustrates a family of correspondence matrices parametrized by* γ*, witnessing the weak refinement between initial states of* S_3 *and* S_4 *(defined in Figures 3.2a–3.2b). The actual matrix used in proving the weak refinement depends on the probability distribution vector* z *that satisfies the constraint function* φ_3 *of state* $(1,1)$*. Take* $\gamma = \frac{0.7 - z_{22}}{z_{23}}$ *if* $z_{22} \leq 0.7$ *and* $\gamma = \frac{0.8 - z_{22}}{z_{23}}$ *otherwise. It is easy to see that* $\varphi_3((1,1))(z)$ *implies* $\varphi_4(1)(z \times \Delta)$*.*

The following theorem shows that weak refinement implies implementation set inclusion.

Theorem 3.3 (Soundness of weak refinement)**.** *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be two CMCs. If* $S_1 \preceq S_2$*, then we have* $[\![S_1]\!] \subseteq [\![S_2]\!]$*.*

Proof.

Since $S_1 \preceq S_2$, there exists a weak refinement relation $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ such that $o_1 \mathcal{R} o_2$. Consider $C = \langle \{1, \ldots n\}, o_C, M, A_C, V_C \rangle$ such that $C \models S_1$. By definition, we have $o_C \models o_1$ and there exists a satisfaction relation $\mathcal{R}' \subseteq \{1, \ldots, n\} \times \{1, \ldots, k_1\}$ such that $o_C \mathcal{R}' o_1$.

Let $\mathcal{R}'' \subseteq \{1, \ldots, n\} \times \{1, \ldots, k_2\}$ such that $p \mathcal{R}'' u \iff \exists v \in \{1, \ldots, k_1\}$ with $p \mathcal{R}' v$ and $v \mathcal{R} u$. Let's show that $\mathcal{R}''$ is a satisfaction relation. First, it is clear that $A_2 \subseteq A_1 \subseteq A_C$.

Now, consider p, u such that $p \mathcal{R}'' u$. By definition, there exists v such that $p \mathcal{R}' v$ and $v \mathcal{R} u$. Since $V_C(p)\downarrow_{A_1} \in V_1(v)$ and $V_1(v)\downarrow_{A_2} \in V_2(u)$, we have $V_C(p)\downarrow_{A_2} \in V_2(u)$.

We now build a correspondence matrix Δ'' that satisfies the axioms of Definition 3.3. Let $x = M_p \in [0,1]^{1\times n}$ and $\Delta' \in [0,1]^{n\times k_1}$ be a correspondence matrix witnessing $p \models v$. Let $y = x \times \Delta' \in [0,1]^{1\times k_1}$. By definition of Δ', we have $\varphi_1(v)(y)$. Let $\Delta \in [0,1]^{k_1\times k_2}$ be the correspondence matrix witnessing $v \preceq u$ and define $\Delta'' = \Delta' \times \Delta \in [0,1]^{n\times k_2}$. By Lemma 3.5, Δ'' is also a correspondence matrix. We prove that Δ'' satisfies the axioms of Definition 3.3.

1. Let $1 \leq p' \leq n$ such that $M_{pp'} \neq 0$. As a consequence, $\sum_{j=1}^{k_1} \Delta'_{p'j} = 1$. We want to prove that $\sum_{j=1}^{k_2} \Delta''_{p'j} = 1$.

$$\begin{aligned}\sum_{j=1}^{k_2} \Delta''_{p'j} &= \sum_{j=1}^{k_2}(\sum_{q=1}^{k_1} \Delta'_{p'q} \cdot \Delta_{qj}) \\ &= \sum_{q=1}^{k_1} \Delta'_{p'q} \cdot (\sum_{j=1}^{k_2} \Delta_{qj})\end{aligned}$$

   Let q such that $\Delta'_{p'q} \neq 0$. It is then clear that $y_q \geq M_{pp'} \cdot \Delta'_{p'q} > 0$. As Δ is a witness of $v \preceq u$, we have $\sum_{j=1}^{k_2} \Delta_{qj} = 1$. Finally, this implies that $\sum_{j=1}^{k_2} \Delta''_{p'j} = 1$.

2. By construction, $\varphi_2(u)(M_p \times \Delta'')$ holds.

3. Let p', u' such that $\Delta''_{p'u'} \neq 0$. By construction, it is clear that there exists v' such that $\Delta'_{p'v'} \neq 0$ and $\Delta_{v'u'} \neq 0$. By definition of Δ' and Δ, this implies that $p' \mathcal{R}' v'$ and $v' \mathcal{R} u'$, thus $p' \mathcal{R}'' u'$.

From 1-3, we can conclude that $\mathcal{R}''$ is a satisfaction relation. Since $o_C \mathcal{R}'' o_2$, we have $C \in [\![S_2]\!]$ and $[\![S_1]\!] \subseteq [\![S_2]\!]$.

□

Since strong refinement implies weak refinement by construction, it also holds that strong refinement imply implementation set inclusion. In Section 3.6, we shall see that the converse holds for a particular class of CMCs. However, this is not the case in general: strong refinement is strictly stronger than weak refinement, which is strictly stronger than implementation set inclusion. Formally, we have the following proposition.

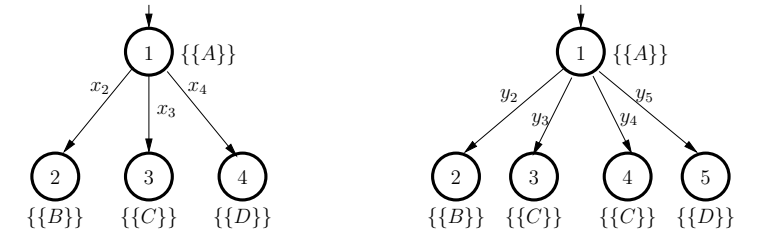
Proposition 3.4. *There exist CMCs S_a, S_b, S_c and S_d such that*

- *S_a weakly refines S_b, and S_a does not strongly refine S_b;*
- *$[\![S_c]\!] \subseteq [\![S_d]\!]$, and S_c does not weakly refine S_d.*

Proof.

We provide separate constructions for the two items of this theorem:

1. Consider the CMCs S_a and S_b given in Figures 3.6a and 3.6b respectively. Call X_a (resp. X_b) the state X in S_a (resp. S_b). We first show that there exists a weak refinement relation $\mathcal{R}$ such that $S_a \preceq S_b$, with $1_a \mathcal{R} 1_b$. We then show that there exists no strong refinement relation between S_a and S_b.



(a) CMC S_a (b) CMC S_b

Figure 3.6: CMCs S_a and S_b.

$$\Delta_x = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \gamma & (1-\gamma) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \Delta = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & a & (1-a) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.7: Correspondence matrices for $S_a \preceq S_b$.

(a) Let $\mathcal{R} = \{(1_a, 1_b), (2_a, 2_b), (3_a, 3_b), (3_a, 4_b), (4_a, 5_b)\}$. We show that $\mathcal{R}$ is a weak refinement relation. We first focus on building the correspondence matrix for the couple $(1_a, 1_b)$. Let x be a "valid" valuation of the outgoing transitions of 1_a. Let $\gamma = \frac{0.7-x_2}{x_3}$ if $x_2 \leq 0.7$ and $\frac{0.8-x_2}{x_3}$ otherwise. As x satisfies $\varphi_a(1_a)$, we have $0 \leq \gamma \leq 1$. Consider the correspondence matrix Δ_x given in Figure 3.7.

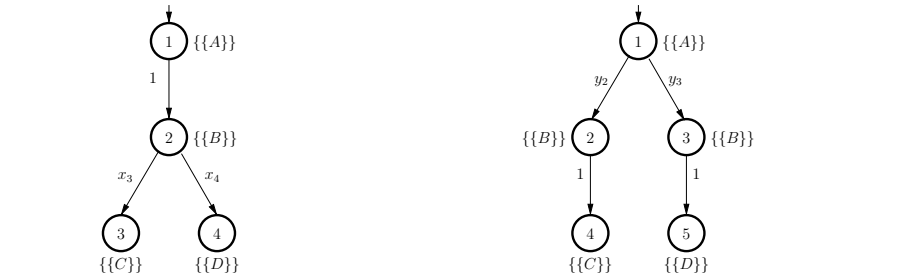
It is easy to see that for all valuation x satisfying $\varphi_a(1_a)$, $\varphi_b(1_b)(x \times \Delta_x)$ also holds. The correspondence matrices for the other pairs in $\mathcal{R}$ are trivial since there are no outgoing transitions from those states. Thus $\mathcal{R}$ is a weak refinement relation between S_a and S_b.

(b) Suppose that there exists a strong refinement relation $\mathcal{R}'$ such that $1_a \, \mathcal{R}' \, 1_b$. Let Δ be the correspondence matrix associated to $1_a \, \mathcal{R}' \, 1_b$. Since $2_a, 3_a$ and 4_a can all be reached from 1_a with an admissible transition, the sum of the elements in the corresponding rows in Δ must be one. From the valuations of the states, we obtain that Δ is of the type given in Figure 3.7, with $a \geq 0$.

Moreover, if $\mathcal{R}'$ is a strong refinement relation, then we have that for all valuation x satisfying $\varphi_a(1_a)$, $\varphi_b(1_b)(x \times \Delta)$ also holds.

Let $x^1 = (0, 0.6, 0.1, 0.3)$ and $x^2 = (0, 0.8, 0.1, 0.1)$. Both x^1 and x^2 satisfy $\varphi_a(1)$. If there exists a strong refinement, this implies that $\varphi_b(1)(x^1 \times \Delta)$ and $\varphi_b(1)(x^2 \times \Delta)$ also hold. However, $\varphi_b(1)(x^1 \times \Delta) = 1$ implies that $a \geq 1$ and $\varphi_b(1)(x^2 \times \Delta)$ implies that $a \leq 0$.

It is thus impossible to find a unique correspondence matrix working for all the "valid" valuations of the outgoing transitions of 1_a. As a consequence, there cannot exist a strong refinement relation $\mathcal{R}'$ such that $1_a \, \mathcal{R}' \, 1_b$.



(a) CMC S_c (b) CMC S_d

Figure 3.8: CMCs S_c and S_d.

2. Consider the CMCs S_c and S_d given in Figures 3.8a and 3.8b. It is easy to see that S_c and S_d share the same set of implementations. However, due to the constraints, State 2 of S_c cannot refine any state of S_d. As a consequence, S_c cannot refine S_d.

□

So our refinement relations for CMCs can be ordered from finest to coarsest: the strong refinement, the weak refinement, and the implementation set inclusion. As the implementation set inclusion is the *ultimate* refinement, checking finer refinements is used as a pragmatic syntax-driven, but sound, way of deciding it.

As we shall see in the next paragraphs, the algorithms for checking weak and strong refinements are polynomial in the number of states, but the treatment of each state depends on the complexity of the constraints. For the case of implementation set inclusion, the algorithm is exponential in terms of number of states. Checking implementation set inclusion seems thus harder than checking weak or strong refinement. In Section 3.6, we will propose a class of CMCs for which strong and weak refinements coincide with implementation set inclusion.

We now briefly discuss algorithms for checking implementation set inclusion and refinements. We start with algorithms for checking weak and strong refinements between two CMCs $S_1 = \langle\{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle\{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2\rangle$. Let $n = \mathsf{max}(k_1, k_2)$. Checking whether a relation $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ is a strong (resp. weak) refinement relation reduces to checking, for all $(i, j) \in \mathcal{R}$, the validity of the following *refinement formulas*: $\exists\Delta, \forall x, \varphi_1(i)(x) \Rightarrow \varphi_2(j)(x \times \Delta) \wedge \bigwedge_{i'}(\sum_{j'} \Delta_{i'j'} = 1) \wedge \bigwedge_{i',j'}(i'\mathcal{R}j' \vee \Delta_{i'j'} = 0)$ for the strong refinement, and $\forall x, \varphi_1(i)(x) \Rightarrow \exists\Delta, \varphi_2(j)(x \times \Delta) \wedge \bigwedge_{i'}(\sum_{j'} \Delta_{i'j'} = 1) \wedge \bigwedge_{i',j'}(i'\mathcal{R}j' \vee \Delta_{i'j'} = 0)$ for the weak refinement. Strong and weak refinements can be decided by iterated strengthening of $\mathcal{R}$ with refinement formulas, starting from $\mathcal{R}_0 = \{(i, j) | V_1(i) \downarrow_{A_2} \subseteq V_2(j)\}$, until either $(o_1, o_2) \notin \mathcal{R}$, in which case S_1 does not strongly (resp. weakly) refine S_2, or $\mathcal{R}$ is found to be a strong (resp. weak) refinement.

The exact complexity of the algorithm depends on the type of constraints that are used in the specifications. As an example, consider that all the constraints in S_1 and S_2 are polynomial of degree d with less than k bound variables – we shall see that polynomial constraints is the least class under which CMCs are closed. There, deciding refinement formulas can be done by quantifier elimination. When the number of quantifier alternations is constant, the *cylindrical algebraic decomposition algorithm* [27, 28], implemented in Maple [135], performs

this quantifier elimination in time double exponential in the number of variables. Consequently, refinement can be checked in $O(n^2 2^{2^{n^2}})$ time.

However, considering constraints φ contain only existential quantifiers, quantifier alternation is either one or two for strong refinement and exactly one for weak refinement. There are quantifier elimination algorithms that have a worst case complexity single exponential only in the number of variables, although they are double exponential in the number of quantifier alternations [16]. Thanks to these algorithms, deciding whether $\mathcal{R}$ is a strong (resp. weak) refinement relation can be done in time single exponential in the number of states n and in the number of bound variables appearing in the constraints k: $O(n^2 s^{P(n,k)} d^{P(n,k)})$ where P is a polynomial.

We now turn to the case of implementation set inclusion. In [86], Larsen and Jonsson proposed an algorithm for solving such problem for the case of IMCs considering that we generalize the constraints. This algorithm has been analyzed in more detail in Chapter 2. It directly extends to CMCs. The main difference with the algorithms for solving weak and strong refinements is that the algorithm for implementation set inclusion is exponential in the number of states of the two CMCs.

Finally, let us mention that lower-bounds for the strong and weak refinement checking remain open problems. In Chapter 2, we have shown that implementation set inclusion is EXPTIME-hard for IMCs, hence providing a lower bound also for CMCs.

3.3.3 Conjunction

Conjunction, also called *logical composition*, combines requirements of several specifications. One of the most important uses of conjunction is the so-called common implementation problem, that we introduced in Chapter 2. This problem consists in deciding wether there exists an implementation that will satisfy all the elements of a set of CMCs, and eventually computing one such implementation. As we will see later, it can be done by computing the conjunction of all the considered CMCs and checking wether it is consistent.

Definition 3.9 (Conjunction)**.** *Let* $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs. The conjunction of* S_1 *and* S_2*, written* $S_1 \wedge S_2$*, is the CMC* $S = \langle \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}, (o_1, o_2), \varphi, A, V \rangle$ *with* $A = A_1 \cup A_2$*,* $V((u,v)) = V_1(u)\uparrow^A \cap V_2(v)\uparrow^A$*, and*

$$\begin{aligned}\varphi((u,v))(x_{1,1}, x_{1,2}, \ldots, x_{2,1}, \ldots, x_{k_1,k_2}) \equiv \\ \varphi_1(u)(\textstyle\sum_{j=1}^{k_2} x_{1,j}, \ldots, \sum_{j=1}^{k_2} x_{k_1,j}) \wedge \\ \varphi_2(v)(\textstyle\sum_{i=1}^{k_1} x_{i,1}, \ldots, \sum_{i=1}^{k_1} x_{i,k_2}).\end{aligned}$$

Conjunction may introduce inconsistent states and thus its use should normally be followed by applying the pruning operator β^*. As already stated in the introduction, the result of conjoining two IMCs is not an IMC in general, but a CMC whose constraint functions are systems of linear inequalities. Figure 3.2b depicts a CMC S_3 expressing the conjunction of IMCs S_1 and S_2 (see Figures 3.1a–3.1b). The constraint $z_{2,3} + z_{3,3} \geq 0.2$ in state $(1,1)$ cannot be expressed as an interval.

In order to build correspondence matrices for a conjunction, we need to define the following operation $\otimes$ on matrices: if $\Delta \in [0,1]^{k \times q}$ and $\Delta' \in [0,1]^{k \times r}$ are two correspondence matrices,

we define $\Delta'' = \Delta \otimes \Delta'$ by $\Delta'' \in [0,1]^{k \times (q \cdot r)}$ and $\Delta''_{i(j,n)} = \Delta_{ij} \cdot \Delta'_{in}$. As stated in the following lemma, this operation preserves the structure of correspondence matrices.

Lemma 3.5. *Let $\Delta \in [0,1]^{k \times q}$ and $\Delta' \in [0,1]^{k \times r}$ be two correspondence matrices. The matrix $\Delta'' = \Delta \otimes \Delta'$ is a correspondence matrix.*

Proof. Let $1 \leq i \leq k$ and $(j,n) \in \{1, \dots q\} \times \{1, \dots r\}$. We have $\Delta''_{i(j,n)} = \Delta_{ij} \cdot \Delta'_{in}$. Thus,

$$\begin{aligned}\sum_{(j,n) \in \{1,\dots q\} \times \{1,\dots r\}} \Delta''_{i(j,n)} &= \sum_{j=1}^{q} \sum_{n=1}^{r} \Delta''_{i(j,n)} \\ &= \sum_{j=1}^{q} \sum_{n=1}^{r} \Delta_{ij} \cdot \Delta'_{in} \\ &= \sum_{j=1}^{q} \Delta_{ij} \sum_{n=1}^{r} \Delta'_{in} \leq 1.\end{aligned}$$

□

As expected, conjunction of two specifications coincides with their greatest lower bound with respect to the weak refinement (also called *shared refinement*).

Theorem 3.6. *Let S_1, S_2 and S_3 be three CMCs. We have (a) $((S_1 \wedge S_2) \preceq S_1)and((S_1 \wedge S_2) \preceq S_2)$ and (b) if $(S_3 \preceq S_1)$ and $(S_3 \preceq S_2)$, then $S_3 \preceq (S_1 \wedge S_2)$.*

Proof.

Let $S_1 = \langle \{1, \dots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$, $S_2 = \langle \{1, \dots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ and $S_3 = \langle \{1, \dots, k_3\}, o_3, \varphi_3, A_3, V_3 \rangle$ be three CMCs.

(a) Let $S_1 \wedge S_2 = S = \langle \{1, \dots, k_1\} \times \{1, \dots, k_2\}, o, \varphi, A, V \rangle$.

Let $\mathcal{R} \subseteq (\{1, \dots, k_1\} \times \{1, \dots, k_2\}) \times \{1, \dots, k_1\}$ such that $(u,v)\,\mathcal{R}\,w \iff u = w$. We will prove that $\mathcal{R}$ is a **strong** refinement relation. Let $u \in \{1, \dots, k_1\}$ and $v \in \{1, \dots, k_2\}$. We have $(u,v)\,\mathcal{R}\,u$. By definition of S, we also have $V((u,v))\downarrow_{A_1} = (V_1(u)\uparrow^A \cap V_2(v)\uparrow^A)\downarrow_{A_1} \subseteq V_1(u)$.

Let $\Delta \in [0,1]^{k_1 \cdot k_2 \times k_1}$ such that $\Delta_{(i,j),i} = 1$ and $\Delta_{(i,j),k} = 0$ if $k \neq i$. By definition, we have $\forall (i,j)$, $\sum_{k=1}^{k_1} \Delta_{(i,j),k} = 1$. As a consequence, Δ is correspondence matrix. We now prove that it satisfies the axioms of a satisfaction relation for $(u,v)\,\mathcal{R}\,u$.

(a) If $x \in [0,1]^{1 \times k_1 \cdot k_2}$ is such that $\varphi((u,v))(x)$, it implies by definition that $\varphi_1(u)(\sum_{j=1}^{k_2} x_{1,j}, \dots, \sum_{j=1}^{k_2} x_{k_1,j}) = \varphi_1(u)(x \times \Delta)$ holds.

(b) If $\Delta_{(u',v'),w'} \neq 0$, we have by definition $u' = w'$ and $(u',v')\,\mathcal{R}\,u'$.

From (a) and (b), we conclude that $\mathcal{R}$ is a **strong** refinement relation.
Since $(o_1, o_2)\,\mathcal{R}\,o_1$, we have $S_1 \wedge S_2 \preceq S_1$. By symmetry, we also have $S_1 \wedge S_2 \preceq S_2$.

(b) Suppose that $S_3 \preceq S_1$ and $S_3 \preceq S_2$. By definition, there exist two refinement relations $\mathcal{R}_1 \subseteq \{1, \dots, k_3\} \times \{1, \dots, k_1\}$ and $\mathcal{R}_2 \subseteq \{1, \dots, k_3\} \times \{1, \dots, k_2\}$ such that $o_3\,\mathcal{R}_1\,o_1$ and $o_3\,\mathcal{R}_2\,o_2$. Let $S_1 \wedge S_2 = S = \langle \{1, \dots, k_1\} \times \{1, \dots, k_2\}, o, \varphi, A, V \rangle$.

Let $\mathcal{R} \subseteq \{1,\ldots,k_3\} \times (\{1,\ldots,k_1\} \times \{1,\ldots,k_2\})$ such that $u\,\mathcal{R}(v,w) \iff u\,\mathcal{R}_1\,v$ and $u\,\mathcal{R}_2\,w$. We now prove that $\mathcal{R}$ is a weak refinement relation.

Consider u, v, w such that $u\,\mathcal{R}(v,w)$.

(a) By definition, we have $V_3(u)\downarrow_{A_1} \subseteq V_1(v)$ and $V_3(u)\downarrow_{A_2} \subseteq V_2(w)$. As a consequence, $V_3(u)\downarrow_A \subseteq V((v,w))$.

(b) Let $x \in [0,1]^{1\times k_3}$ such that $\varphi_3(u)(x)$. Consider the correspondence matrices $\Delta \in [0,1]^{k_3\times k_1}$ and $\Delta' \in [0,1]^{k_3\times k_2}$ given by $u\,\mathcal{R}_1\,v$ and $u\,\mathcal{R}_2\,w$ for the transition vector x. Let $\Delta'' \in [0,1]^{k_3\times k_1\cdot k_2} = \Delta \otimes \Delta'$. By Lemma 3.5, Δ'' is a correspondence matrix. We now prove that it satisfies the axioms of a refinement relation for $u\,\mathcal{R}(v,w)$.

i. Let $1 \leq i \leq k_3$ such that $x_i \neq 0$. By definition of Δ and Δ', we have $\sum_{j=1}^{k_1} \Delta_{ij} = 1$ and $\sum_{q=1}^{k_2} \Delta'_{iq} = 1$. By construction, $\sum_{(j,q)\in\{1,\ldots,k_1\}\times\{1,\ldots,k_2\}} \Delta''_{i(j,q)} = (\sum_{j=1}^{k_1} \Delta_{ij}) \cdot (\sum_{q=1}^{k_2} \Delta'_{iq}) = 1$.

ii. By definition of Δ and Δ', both $\varphi_1(v)(x \times \Delta)$ and $\varphi_2(w)(x \times \Delta')$ hold. Let $x' = x \times \Delta''$. It is clear that $x \times \Delta = (\sum_{j=1}^{k_2} x'_{1,j}, \ldots, \sum_{j=1}^{k_2} x'_{k_1,j})$ and $x \times \Delta' = (\sum_{i=1}^{k_1} x'_{i,1}, \ldots, \sum_{i=1}^{k_1} x'_{i,k_2})$. As a consequence, $\varphi((v,w))(x \times \Delta'')$ holds.

iii. Let u', v', w' such that $\Delta''_{u'(v',w')} \neq 0$. By construction, this implies $\Delta_{u'v'} \neq 0$ and $\Delta'_{u'w'} \neq 0$. As a consequence, $u'\,\mathcal{R}_1\,v'$ and $u'\,\mathcal{R}_2\,w'$, thus $u'\,\mathcal{R}(v',w')$.

From (i) - (iii), we conclude that $\mathcal{R}$ is a weak refinement relation. Since $o_3\,\mathcal{R}(o_1,o_2)$, we have $S_3 \preceq (S_1 \wedge S_2)$.

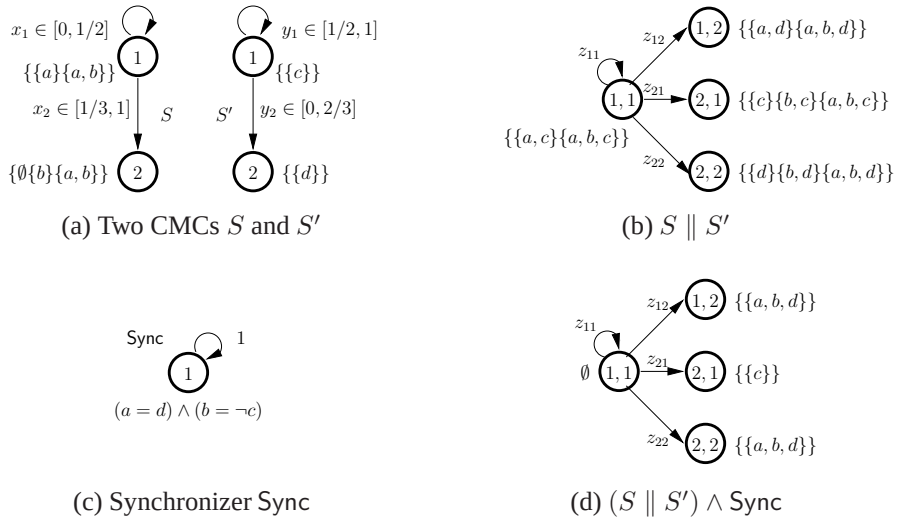
□

In fact, as follows from the later results of Section 3.6, the set of implementations of a conjunction of two *deterministic* specifications S_1 and S_2 coincides with the intersection of implementation sets of S_1 and S_2 (the greatest lower bound in the lattice of implementation sets).

3.4 Compositional Reasoning

Let us now turn to *structural* composition. In our theory, as we already said in the introduction and after presenting CMCs, choices regarding the set of valuations and stochastic choices are independent from each others. This property of the model naturally leads to a definition of the parallel composition operator based on the principle of *separation of concerns*. The idea is that probabilistic behaviours are composed separately from the synchronization of the sets of state valuations. This allows realizing probabilistic composition as a simple product of independent distributions.

Remark 3.1. *The principle of separation of concerns is intensively used in the definition of parallel composition for many systems that mix stochastic and non-deterministic choices. Among them, one can cite many theories for probabilistic process algebra [121, 87]. Similar principles are also applied for continuous time stochastic models, in a slightly different setting based on CTMCs [79]. In Section 3.8, we shall see that our structural composition covers the one of probabilistic automata.*



(a) Two CMCs S and S'

(b) $S \parallel S'$

(c) Synchronizer Sync

(d) $(S \parallel S') \wedge$ Sync

Figure 3.9: Parallel composition and synchronization of CMCs

Following the separation of concerns principle, components are composed first into a product (or effectively just a vector of independent entities), and then synchronized by constraining their behaviour. This design is both simple and expressive: it allows applying diverse synchronization mechanisms, beyond just matching inputs to outputs. Moreover it elegantly exploits the prior knowledge on logical composition, as the synchronization operator turns out to be realizable using conjunction.

We start by discussing how systems and specifications can be composed in a non-synchronizing way, then we introduce a notion of synchronization.

3.4.1 Independent parallel composition

The non-synchronizing *independent* composition is largely just a product of two MCs (or CMCs).

Definition 3.10 (Parallel Composition of MCs)**.** *Let* $C_1 = \langle \{1, \dots, n_1\}, o_1, M', A_1, V_1 \rangle$ *and* $C_2 = \langle \{1, \dots, n_2\}, o_2, M'', A_2, V_2 \rangle$ *be two MCs with* $A_1 \cap A_2 = \emptyset$*. The parallel composition of* C_1 *and* C_2 *is the MC* $C_1 \parallel C_2 = \langle \{1, \dots, n_1\} \times \{1, \dots, n_2\}, (o_1, o_2), M, A_1 \cup A_2, V \rangle$ *where:* $M \in [0, 1]^{(n_1 \times n_2) \times (n_1 \times n_2)}$ *is such that* $M_{(p,q)(r,s)} = M'_{pr} \cdot M''_{qs}$*, and* $V((p, q)) = V_1(p) \cup V_2(q)$*.*

And in general for CMCs:

Definition 3.11 (Parallel Composition of CMCs)**.** *Let* $S_1 = \langle \{1, \dots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ *and* $S_2 = \langle \{1, \dots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ *be CMCs with* $A_1 \cap A_2 = \emptyset$*. The parallel composition of* S_1 *and* S_2 *is the CMC* $S_1 \parallel S_2 = \langle \{1, \dots, k_1\} \times \{1, \dots, k_2\}, (o_1, o_2), \varphi, A_1 \cup A_2, V \rangle$*, where* $\varphi((u, v))(z_{1,1}, z_{1,2}, \dots z_{2,1}, \dots, z_{k_1,k_2}) = \exists x_1, \dots, x_{k_1}, y_1, \dots, y_{k_2} \in [0, 1]$ *such that* $\forall (i, j) \in \{1, \dots, k_1\} \times \{1, \dots, k_2\}$ *we have* $z_{i,j} = x_i \cdot y_j$ *and* $\varphi_1(u)(x_1, \dots, x_{k_1}) = \varphi_2(v)(y_1, \dots, y_{k_2}) = 1$*, and* $V((u, v)) = \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\}$*.*

It is worth mentioning that IMCs are not closed under composition. Consider IMCs S and S' given in Figure 3.9a and their composition $S \parallel S'$ given in Figure 3.9b. Assume first that $S \parallel S'$ is an IMC. As a variable z_{ij} is the product of two variables x_i and y_j, if $S \parallel S'$ is an IMC,

then one can show that the interval for z_{ij} is obtained by computing the products of the bounds of the intervals over which x_i and y_j range. Hence, we can show that $z_{11} \in [0, 1/2], z_{12} \in [0, 1/3], z_{21} \in [1/6, 1], z_{22} \in [0, 2/3]$. Let $[a, b]$ be the interval for the constraint z_{ij}, it is easy to see that there exists implementations I_1 of S_1 and I_2 of S_2 such that $I_1 \parallel I_2$ satisfies the constraint $z_{ij} = a$ (resp. $z_{ij} = b$). However, while each bound of each interval can be satisfied independently, some points in the polytope defined by the intervals and the constraint $\sum z_{ij} = 1$ cannot be reached. As an example, consider $z_{11} = 0, z_{12} = 1/3, z_{21} = 1/3, z_{22} = 1/3$. It is clearly inside the polytope, but one cannot find an implementation I of $S \parallel S'$ satisfying the constraints given by the parallel composition. Indeed, having $z_{11} = 0$ implies that $x_1 = 0$ and thus that $z_{12} = 0$.

In order to build correspondence matrices for a composition, we need to define the following operation $\odot$ on matrices: if $\Delta \in [0, 1]^{k \times q}$ and $\Delta' \in [0, 1]^{r \times s}$ are two correspondence matrices, we define $\Delta'' = \Delta \odot \Delta'$ by $\Delta'' \in [0, 1]^{(k \cdot r) \times (q \cdot s)}$ and $\Delta''_{(i,j)(n,p)} = \Delta_{in} \cdot \Delta'_{jp}$. As stated in the following Lemma, this operation preserves the structure of correspondence matrices.

Lemma 3.7. *Let $\Delta \in [0, 1]^{k \times q}$ and $\Delta' \in [0, 1]^{r \times s}$ be two correspondence matrices. The matrix $\Delta'' = \Delta \odot \Delta'$ is a correspondence matrix.*

Proof.

Let $(i, j) \in \{1, \dots k\} \times \{1, \dots r\}$ and $(n, p) \in \{1, \dots q\} \times \{1, \dots s\}$. We have $\Delta''_{(i,j)(n,p)} = \Delta_{in} \cdot \Delta'_{jp}$. Thus,

$$\begin{aligned}\sum_{(n,p) \in \{1,\dots q\} \times \{1,\dots s\}} \Delta''_{(i,j)(n,p)} &= \sum_{n=1}^{q} \sum_{p=1}^{s} \Delta_{in} \cdot \Delta'_{jp} \\ &= (\sum_{n=1}^{q} \Delta_{in}) \cdot (\sum_{p=1}^{s} \Delta'_{jp}) \\ &\leq 1.\end{aligned}$$

□

The following theorem shows that the weak refinement is a precongruence with respect to parallel composition. Remark that the same is also true for strong refinement.

Theorem 3.8. *If S'_1, S'_2, S_1, S_2 are CMCs then $S'_1 \preceq S_1$ and $S'_2 \preceq S_2$ implies $S'_1 \parallel S'_2 \preceq S_1 \parallel S_2$, so the weak refinement is a precongruence with respect to parallel composition. Consequently, for any MCs C_1 and C_2 we have that $C_1 \models S_1 \wedge C_2 \models S_2$ implies $C_1 \parallel C_2 \models S_1 \parallel S_2$.*

Proof.

Let $S'_1 = \langle \{1, \dots, k'_1\}, o'_1, \varphi'_1, A'_1, V'_1 \rangle$, $S'_2 = \langle \{1, \dots, k'_2\}, o'_2, \varphi'_2, A'_2, V'_2 \rangle$, $S_1 = \langle \{1, \dots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$, $S_2 = \langle \{1, \dots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be four CMCs. Suppose $S'_1 \preceq S_1$ and $S'_2 \preceq S_2$.

Let $S = \langle \{1, \dots, k_1\} \times \{1, \dots, k_2\}, (o_1, o_2), \varphi, A, V \rangle = S_1 \parallel S_2$ and $S' = \langle \{1, \dots, k'_1\} \times \{1, \dots, k'_2\}, (o'_1, o'_2), \varphi', A', V' \rangle = S'_1 \parallel S'_2$.

By definition, there exist two weak refinement relations $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $o'_1 \mathcal{R}_1 o_1$ and $o'_2 \mathcal{R}_2 o_2$. Define $\mathcal{R}$ such that $(u', v') \mathcal{R} (u, v) \iff u' \mathcal{R}_1 u$ and $v' \mathcal{R}_2 v$. Consider now such (u', v') and (u, v). We prove that $\mathcal{R}$ satisfies the axioms of a refinement relation between (u', v') and (u, v).

1. We have $(V'((u',v')))\downarrow_A = \{Q \subseteq 2^{A'} \mid \exists Q_1 \in V'_1(u'),\ Q_2 \in V'_2(v'),\ Q = Q_1 \cup Q_2\}\downarrow_A = \{Q \subseteq 2^A \mid \exists Q_1 \in V'_1(u'),\ Q_2 \in V'_2(v'),\ Q = Q_1\downarrow_{A_1} \cup Q_2\downarrow_{A_2}\}$. Thus $(V'((u',v')))\downarrow_A \subseteq V((u,v))$.

2. Let $z' \in [0,1]^{1\times k'_1\cdot k'_2}$ such that $\varphi'(u',v')(z')$. We now build the correspondence matrix Δ witnessing $(u',v')\,\mathcal{R}(u,v)$. Consider the correspondence matrices Δ_1 and Δ_2 given by $u'\,\mathcal{R}_1\,u$ and $v'\,\mathcal{R}_2\,v$ for the transition vector z'. Define $\Delta = \Delta_1 \odot \Delta_2 \in [0,1]^{k'_1\cdot k'_2 \times k_1\cdot k_2}$. By Lemma 3.7, Δ is a correspondence matrix. Moreover, since $\varphi'(u',v')(z')$ holds, there exists $x' \in [0,1]^{1\times k'_1}$ and $y' \in [0,1]^{1\times k'_2}$ such that $\forall i,j,\ z'_{(i,j)} = x'_i \cdot y'_j$ and $\varphi'_1(u')(x')$ and $\varphi'_2(v')(y')$.

   (a) Let $(u'',v'') \in \{1,\dots,k'_1\} \times \{1,\dots,k'_2\}$ such that $z_{(u'',v'')} \neq 0$. By definition of x' and y', this implies that $x'_{u''} \neq 0$ and $y'_{v''} \neq 0$. Thus $\sum_{j=1}^{k_1} \Delta_{1u''j} = 1$ and $\sum_{j=1}^{k_2} \Delta_{2v''j} = 1$.

$$\begin{aligned}
&\sum_{(r,s)\in\{1,\dots,k_1\}\times\{1,\dots,k_2\}} \Delta_{(u'',v'')(r,s)} = \\
&\sum_{(r,s)\in\{1,\dots,k_1\}\times\{1,\dots,k_2\}} \Delta_{1u''r}\cdot\Delta_{2v''s} \\
&= \sum_{r=1}^{k_1}\sum_{s=1}^{k_2} \Delta_{1u''r}\cdot\Delta_{2v''s} \\
&= (\sum_{r=1}^{k_1} \Delta_{1u''r}) \cdot (\sum_{s=1}^{k_2} \Delta_{2v''s}) = 1.
\end{aligned}$$

   (b) Let $z = z' \times \Delta \in [0,1]^{1\times k_1\cdot k_2}$. Remark that $z = (x' \times \Delta_1) \otimes (y' \times \Delta_2)$.
   Let $x = x' \times \Delta_1$ and $y = y' \times \Delta_2$. Since $u'\,\mathcal{R}_1\,u$ and $v'\,\mathcal{R}_2\,v$, we have $\varphi_1(u)(x)$ and $\varphi_2(v)(y)$. Thus $\varphi(u,v)(z' \times \Delta)$.

   (c) Let $u'', v'', u'''v'''$ such that $\Delta_{(u'',v'')(u''',v''')} \neq 0$. By definition, it implies that $\Delta_{1u''u'''} \neq 0$ and $\Delta_{2v''v'''} \neq 0$, and as a consequence $(u'',v'')\,\mathcal{R}(u''',v''')$.

From (a),(b),(c), we conclude that $\mathcal{R}$ is a weak refinement relation. Since $(o'_1,o'_2)\,\mathcal{R}(o_1,o_2)$, we have $S' \preceq S$.

The proof of the second part of the theorem is similar, and left to the reader. Remark that this proof easily adapts to the case of strong refinement.

□

3.4.2 Synchronization

As alphabets of composed CMCs have to be disjoint, the composition does not synchronize the components on state valuations like it is typically done for other (non-probabilistic) models. However, synchronization can be introduced by conjoining the composition with a *synchronizer*—a single-state CMC whose valuation function relates the atomic propositions of the composed CMCs.

Example. *CMC $S \parallel S'$ of Figure 3.9b is synchronized with the synchronizer* Sync *given in Figure 3.9c.* Sync *removes from $S \parallel S'$ all the valuations that do not satisfy $(a = d) \wedge (b = \neg c)$. The result is given in Figure 3.9d. Observe that an inconsistency appears in State $(1, 1)$. Indeed, there is no implementation of the two CMCs that can synchronize in the prescribed way. In general inconsistencies like this one can be uncovered by applying the pruning operator, which would return an empty specification. So synchronizers enable discovery of incompatibilities between component specifications in the same way as it is known for non-probabilistic specification models.*

Synchronization is associative with respect to composition, which means that the order of synchronization and composition is inessential for final functionality of the system.

Theorem 3.9. *Let S_1, S_2 and S_3 be three CMCs with pairwise disjoint sets of propositions A_1, A_2 and A_3. Let Sync_{123} be a synchronizer over $A_1 \cup A_2 \cup A_3$ and let Sync_{12} be the same synchronizer with its set of propositions restricted to $A_1 \cup A_2$. The following holds $[\![[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}]\!] = [\![(S_1 \parallel S_2 \parallel S_3) \wedge \mathsf{Sync}_{123}]\!]$.*

Proof.

Let S_1, S_2 and S_3 be three CMCs with disjoint sets of atomic propositions A_1, A_2 and A_3. Let $\mathsf{Sync}_{123} = \langle \{1\}, 1, "\lambda x.x = 1", A_1 \cup A_2 \cup A_3, V_{\mathsf{Sync}} \rangle$ be a synchronizer between A_1, A_2 and A_3. Consider $\mathsf{Sync}_{12} = \langle \{1\}, 1, "\lambda x.x = 1", A_1 \cup A_2, V_{\mathsf{Sync}} \downarrow_{A_1 \cup A_2} \rangle$. We want to prove that $[\![[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}]\!] = [\![[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123}]\!]$.

We first prove the following statement. Let S_1 and S_2 be two CMCs with disjoint sets of atomic propositions A_1 and A_2. Let Sync_1 be a synchronizing vector on A_1. We have $(S_1 \parallel S_2) \wedge \mathsf{Sync}_1 = (S_1 \wedge \mathsf{Sync}_1) \parallel S_2$.

First, remember that synchronizers are single state CMCs, with a single transition taken with probability 1. As a consequence, computing the conjunction with a synchronizer preserves the structure of any CMC. The only change lies in the sets of valuations.

Let p be a state of S_1 and q be a state of S_2. We have $(V_1(p) \cup V_2(q)) \cap V_{\mathsf{Sync}_1} \uparrow^{A_1 \cup A_2} = (V_1(p) \cap V_{\mathsf{Sync}_1}) \cup V_2(q)$. As a consequence, the valuations of $(S_1 \wedge \mathsf{Sync}_1) \parallel S_2$ are the same as the valuations of $(S_1 \parallel S_2) \wedge \mathsf{Sync}_1$.

By monotony of conjunction, we have $(S_1 \parallel S_2) \wedge \mathsf{Sync}_{12} \preceq (S_1 \parallel S_2)$. By Theorem 3.8, it implies that $[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123} \preceq [S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123}$, and finally $[\![[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}]\!] \subseteq [\![[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{S}_{123}]\!]$.

We now prove that $[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123} \preceq [((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}$. By monotony of conjunction, we have $[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123} \preceq [S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{12} \wedge \mathsf{Sync}_{123}$. Moreover, by the statement proved above, we have $[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{12} \preceq ((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3$. As a consequence, we have $[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123} \preceq [((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}$, and thus $[\![[S_1 \parallel S_2 \parallel S_3] \wedge \mathsf{Sync}_{123}]\!] \subseteq [\![[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}]\!]$. □

Finally, synchronized composition also supports component-based refinement in the style of Theorem 3.8:

Theorem 3.10. *If S'_1, S'_2, S_1, S_2 are CMCs,* Sync *is a synchronizer and $S'_1 \preceq S_1 \wedge S'_2 \preceq S_2$ then $(S'_1 \parallel S'_2) \wedge \mathsf{Sync} \preceq (S_1 \parallel S_2) \wedge \mathsf{Sync}$.*

Consequently, a modeller can continue independent refinement of specifications under synchronization, knowing that the original synchronized specification will not be violated.

3.4.3 Comparison of conjunction and parallel composition

We now compare conjunction and composition with respect to implementation set inclusions. We shall see that if the two operations are defined on CMCs with independent sets of valuations, then composition refines conjunction; the opposite does not hold. We first show that composition refines conjunction.

Theorem 3.11. *Let S_1 and S_2 be consistent CMCs with $A_1 \cap A_2 = \emptyset$. It holds that $S_1 \parallel S_2 \preceq S_1 \wedge S_2$.*

Proof. Let $S_1 \parallel S_2 = \langle \{1,\dots,k_1\} \times \{1,\dots,k_2\}, (o_1,o_2), \varphi^{\parallel}, A, V^{\parallel} \rangle$ and $S_1 \wedge S_2 = \langle \{1,\dots,k_1\} \times \{1,\dots,k_2\}, (o_1,o_2), \varphi^{\wedge}, A, V^{\wedge} \rangle$, where $A = A_1 \cup A_2$. We build a refinement relation $\mathcal{R}$ on $(\{1,\dots,k_1\} \times \{1,\dots,k_2\}) \times (\{1,\dots,k_1\} \times \{1,\dots,k_2\})$ as $(u,v)\,\mathcal{R}(u',v')$ if and only if $u = u'$ and $v = v'$.

Let $(u,v) \in \{1,\dots,k_1\} \times \{1,\dots,k_2\}$ such that $(u,v)\,\mathcal{R}(u,v)$. We now show that $\mathcal{R}$ is a refinement relation:

1. By construction, we have that $V^{\parallel}((u,v)) = \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\}$. Moreover, since $A_1 \cap A_2 = \emptyset$, we have that $V^{\wedge}((u,v)) = V_1(u)\uparrow^A \cap V_2(v)\uparrow^A = \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\}$. Thus $V^{\parallel}((u,v)) = V^{\wedge}((u,v))$.

2. Let $z = (z_{1,1}, z_{1,2}, \dots, z_{k_1,k_2}) \in [0,1]^{k_1 \cdot k_2}$ such that $\varphi^{\parallel}((u,v))(z)$ holds. Define the correspondence matrix $\Delta \in [0,1]^{(k_1 \cdot k_2) \times (k_1 \cdot k_2)}$ as the matrix with $\Delta_{(u,v),(u,v)} = 1$ if $z_{u,v} \neq 0$ and 0 otherwise. Observe that $z \times \Delta = z$.

   - Trivially, by construction, for all $(i,j) \in \{1,\dots,k_1\} \times \{1,\dots,k_2\}$ such that $z_{i,j} \neq 0$, we have that $\sum_{i',j'} \Delta_{(i,j),(i',j')} = 1$.
   - We prove that $\varphi^{\wedge}((u,v))(z)$ holds: By hypothesis, $\varphi^{\parallel}((u,v))(z)$ holds. Thus, by definition, there exist $x \in [0,1]^{k_1}$ and $y \in [0,1]^{k_2}$ such that $\varphi_1(u)(x)$ holds, $\varphi_2(v)(y)$ holds and for all $i \in \{1,\dots,k_1\}$ and $j \in \{1,\dots,k_2\}$, we have $z_{i,j} = x_i \cdot y_j$. As a consequence, we have $\sum_{i=1}^{k_1} z_{i,j} = y_j$ for all $j \in \{1,\dots,k_2\}$ and $\sum_{j=1}^{k_2} z_{i,j} = x_i$ for all $i \in \{1,\dots,k_1\}$. Since both $\varphi_1(u)(x)$ and $\varphi_2(v)(y)$ hold, we have that $\varphi^{\wedge}((u,v))(z \times \Delta)$ holds.
   - By construction of Δ, $\Delta_{(u,v),(u',v')} \neq 0$ implies that $u = u'$ and $v = v'$, and therefore implies $(u,v)\,\mathcal{R}(u',v')$.

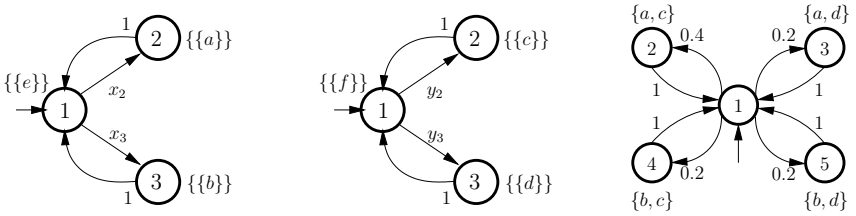
We conclude that $\mathcal{R}$ is a refinement relation, and $(o_1,o_2)\,\mathcal{R}(o_1,o_2)$. Thus, $S_1 \parallel S_2 \preceq S_1 \wedge S_2$. □

A direct consequence of the above theorem is that any model of the composition is a model for the conjunction, i.e.,$[\![S_1 \parallel S_2]\!] \subseteq [\![S_1 \wedge S_2]\!]$. We now show that the opposite inclusion does not hold.

Theorem 3.12. *Let S_1 and S_2 be consistent CMCs with $A_1 \cap A_2 = \emptyset$. It holds that $[\![S_1 \wedge S_2]\!] \not\subseteq [\![S_1 \parallel S_2]\!]$.*

Proof. We establish the proof by providing in Figure 3.10 CMCs S_1 and S_2 and a MC I, such that $I \models S_1 \wedge S_2$ and $I \not\models S_1 \parallel S_2$.

The common structure of conjunction and parallel composition is shown in Figure 3.11. However the constraint function is not equal: we have $\varphi^{\wedge}(1,1)(z) \equiv z_{2,2} + z_{2,3} = z_{2,2} + z_{3,2} =$



(a) S_1, $\varphi_1(1)(x) \equiv x_2 = .6 \wedge x_3 = .4$ (b) S_2, $\varphi_2(1)(y) \equiv y_2 = .6 \wedge y_3 = .4$ (c) I

Figure 3.10

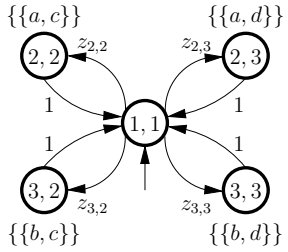


Figure 3.11: Common structure of conjunction and parallel composition

$0.6 \wedge z_{3,2} + z_{3,3} = z_{2,3} + z_{3,3} = 0.4$ and $\varphi^{\|}(1,1)(z) \equiv z_{2,2} = 0.36 \wedge z_{2,3} = z_{3,2} = 0.24 \wedge z_{3,3} = 0.16$. I will satisfy the conjunction, but not the parallel composition, since the probability mass 0.4 of going to state 2 in I, can not be distributed to $(2,2)$ of $S_1 \parallel S_2$.

□

Remark 3.2. *Crucially, a conjunction of two MCs is not a MC, but a proper CMC, while parallel composition of MCs results in a new MC.*

3.5 Disjunction and Universality

In this section we show that CMCs are not closed under disjunction. We then solve the *universality problem*, that is the problem of deciding whether a CMCs admits any implementation.

3.5.1 On the Existence of a Disjunction of CMCs

In this section we discuss the problem of computing a CMC S whose models are the union of the models accepted by two other CMCs $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$. In general, such a CMC may not exist. Indeed, assume that S_1 and S_2 have independent initial state valuations, and that the constraint functions of o_1 and o_2 do not share the same set of satisfying probability vectors. The initial state o of any specification representing the union could take valuations admissible according to o_1 and a distribution according to o_2 (but not o_1). That is, we can not express that, depending on the valuation of the initial state of the implementation, a certain constraint should be chosen.

However, if S_1 and S_2 have the same initial state valuation, i.e., $V_1(o_1) = V_2(o_2)$, then we can construct the disjunction explicitly. Let $S_1 \vee S_2 = \langle Q, 0, \varphi, A, V \rangle$ where

- $Q = \{1, \ldots, k_1\} \cup (\{1, \ldots, k_2\} \cup \{0\}$,
- $A = A_1 \cup A_2$,
- $V(0) = V_1(o_1) = V_2(o_2)$, and $V(u) = V_1(u)$ if $u \in \{1, \ldots, k_1\}$, $V(v) = V_2(v)$ if $v \in \{1, \ldots, k_2\}$,
- The constraint function $\varphi : Q \to [0,1]^{k_1+k_2+1} \to \{0,1\}$ is given by

$$\varphi(0)(x_0, x_1, \ldots, x_{k_1}, x_{1'}, \ldots, x_{k'_2}) \equiv \left(\sum_{i=1}^{k_1} x_i = 1 \wedge \varphi_1(o_1)(x_1, \ldots, x_{k_1})\right) \vee \left(\sum_{i=1'}^{k'_2} x_i = 1 \wedge \varphi_2(o_2)(x'_1, \ldots, x_{k'_2})\right)$$

$$\varphi(i)(x_0, x_1, \ldots, x_{k_1}, x_{1'}, \ldots, x_{k'_2}) \equiv \sum_{i=1}^{k_1} x_i = 1 \wedge \varphi_1(o_1)(x_1, \ldots, x_{k_1}), \;\; i \in \{1, \ldots, k_1\}$$

$$\varphi(j)(x_0, x_1, \ldots, x_{k_1}, x_{1'}, \ldots, x_{k'_2}) \equiv \sum_{i=1'}^{k'_2} x_i = 1 \wedge \varphi_2(o_2)(x'_1, \ldots, x_{k'_2}), \;\; j \in \{1, \ldots, k_2\}$$

By construction, the so defined disjunction of S_1 and S_2, $S_1 \vee S_2$, is such that its set of implementations is the union of the sets of implementations of S_1 and S_2.

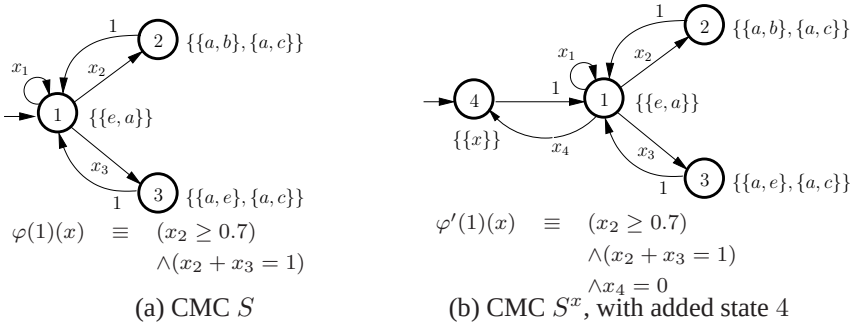
Theorem 3.13. *Let $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be two CMCs such that $V_1(o_1) = V_2(o_2)$. It holds that*

$$[\![S_1]\!] \cup [\![S_2]\!] = [\![S_1 \vee S_2]\!].$$

3.5.2 The Universality Problem for CMCs

We study the universality problem for CMCs, i.e., the problem of deciding whether a CMC S admits any model defined on a given set of atomic propositions A. For doing so we simply check whether the universal CMC Univ^A representing all these models thoroughly refines S. The CMC Univ^A is formally defined as $\mathsf{Univ}^A = \langle \{1\}, 1, \varphi, A, V \rangle$, where $\varphi(1)(x) \equiv 1$ and $V(1) = 2^{2^A}$.

Theorem 3.14. *Let $\mathsf{Univ}^A = \langle \{1\}, 1, \varphi, A, V \rangle$ be the universal CMC on the set of atomic propositions A and let $I = \langle \{1, \ldots, n\}, o, M, A_I, V_I \rangle$ be any implementation such that $A \subseteq A_I$. We have that $I \models \mathsf{Univ}^A$.*



(a) CMC S (b) CMC S^x, with added state 4

Figure 3.12: CMCs S and S^x

Proof. Construct the relation $\mathcal{R} = \{1, \ldots, n\} \times \{1\}$. We show that $\mathcal{R}$ is a satisfaction relation: Let $i \in \{1, \ldots, n\}$ such that $i \,\mathcal{R}\, 1$.

1. It is clear that $V_I(i)\downarrow_A \in V(1) = 2^{2^A}$.

2. Consider M_i. We build a correspondence matrix $\Delta \in [0, 1]^{n \times 1}$ such that $\Delta_{j1} = 1$ if $M_{ij} > 0$, and 0 else.

   - By construction, $\Delta_{j1} = 1$ for all j such that $M_{ij} > 0$.
   - Since $M_i \times \Delta = 1$, $\varphi(1)(M_i \times \Delta)$ holds.
   - Let i' such that $\Delta_{i',1} > 0$. By construction of $\mathcal{R}$, $i' \,\mathcal{R}\, 1$.

We conclude that $\mathcal{R}$ is a satisfaction relation, since $o \,\mathcal{R}\, 1$, and thus, $I \models \mathsf{Univ}^A$. □

We now switch to the problem of deciding whether the union of two CMCs S_1 and S_2 is universal. This is a more intriguing problem as we have seen that CMCs are not closed under union. As a solution to this problem, we propose a *state-extended* notion that consists in creating a new initial state with a new special valuation $x \notin A$ and then redistribute the entire probability mass to the original initial state. Formally, we propose the following definition.

Definition 3.12. *For a CMC $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ and a valuation $x \notin A$, we define the state-extended CMC $S^x = \langle \{1, \ldots, k, o'\}, o', \varphi', A', V' \rangle$ where*

- $A' = A \cup \{\{x\}\}$,
- $V'(o') = \{x\}$ *and* $V'(i) = V(i)$ *for all* $i \in \{1, \ldots, k\}$, *and*
- $\varphi'(o')(x) \equiv x_o = 1$ *and* $\varphi'(i)(x) \equiv \varphi(i)(x_1, \ldots, x_k) \wedge x_{o'} = 0$ *for all* $i \in \{1, \ldots, k\}$.

An example is given in Figure 3.12. The union of the state-extended versions of S_1 and S_2 can now be computed and compared to the state-extended version of Univ^A. It is obvious that all the implementations of the state-extended version of a given CMC C are state-extended versions of implementations of C.

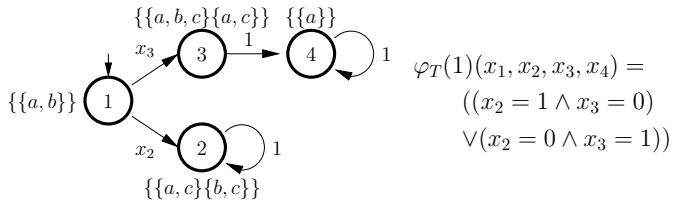


Figure 3.13: A CMC T whose set of implementations cannot be represented with a deterministic CMC

3.6 Deterministic CMCs

Clearly, if all implementations of a specification S_1 also implement a specification S_2, then the former is a proper strengthening of the latter. Indeed, S_1 specifies implementations that break no assumptions that can be made about implementations of S_2. Thus, as it was previously said in Chapter 2, implementation set inclusion – also called *thorough refinement – is* a desirable refinement for specifications. Unfortunately, this problem is still open for CMCs, and, as we have said, the weak and the strong refinement soundly approximate it. Had that approximation been complete, we would have had an effective decision procedure for implementation set inclusion. In this section, we argue that, as proven for IMCs in Chapter 2, this indeed is the case for an important subclass of specifications: *deterministic CMCs*. The definition for determinism is the same as the notion of strong determinism for IMCs introduced in Chapter 2. A CMC S is *deterministic* iff for every state i, states reachable from i have pairwise disjoint admissible valuations:

Definition 3.13. *Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a CMC. S is* deterministic *iff for all states $i, u, v \in \{1, \ldots, k\}$, if there exists $x \in [0,1]^k$ such that $(\varphi(i)(x) \wedge (x_u \neq 0))$ and $y \in [0,1]^k$ such that $(\varphi(i)(y) \wedge (y_v \neq 0))$, then we have that $V(u) \cap V(v) = \emptyset$.*

In Figures 3.1a and 3.1b, both S_1 and S_2 are deterministic specifications. In particular states 2 and 3, reachable from 1 in both CMCs, have disjoint constraints on valuations. On the other hand, the CMC T given in Figure 3.13 is non-deterministic. Indeed, for States 2 and 3, which can both be reached from State 1, we have that $V_T(2) \cap V_T(3) = \{\{a, c\}\} \neq \emptyset$.

Deterministic CMCs are less expressive than non-deterministic ones, in the sense that the same implementation sets cannot sometimes be expressed. Consider again the CMC T given in Figure 3.13. It is such that its set of implementations cannot be represented by a deterministic CMC. Indeed, any merging of States 2 and 3 in T would result in a CMC that accepts models where one can loop on valuation $\{a, c\}$ and then accept valuation $\{a\}$ with probability 1. Such a model cannot be accepted by T.

Proposition 3.15. *Conjunction and composition preserve determinism.*

Determinism of a CMC with polynomial constraints can be decided in single exponential time in the number of states. The problem becomes polynomial when restricting constraints to linear inequalities. Consider a CMC $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ with linear constraints of the form $\varphi(i)(x) = x \times C_i \leq b_i$. Since S is deterministic, for each states i, j such that $i < j$, we must have that $V(i) \cap V(j) \neq \emptyset$ implies for all k, $\{x \mid x \times C_k \leq b_k \wedge x_i = 0\} = \emptyset$ or $\{y \mid y \times C_k \leq b_k \wedge y_j = 0\} = \emptyset$. This can be decided in polynomial time using Fourier-Motzkin elimination [119].

We now present a determinization algorithm that can be applied to any CMC S whose initial state is a single valuation set. This algorithm relies on normalizing the specification first, and otherwise applies an algorithm which resembles determinization of automata (a subset construction). The result of the algorithm is a new CMC refined by S. Consequently the implementation set of the result includes the one of S (see Theorem 3.16 below). This weakening character of determinization resembles the known determinization algorithms for modal transition systems [18].

Definition 3.14. *Let $S = \langle\{1,\dots,k\}, o, \varphi, A, V\rangle$ be a consistent CMC in the single valuation normal form. Let $m < k$ and $h : \{1,\dots,k\} \to \{1,\dots,m\}$ be a surjection such that (1) $\{1,\dots,k\} = \cup_{v\in\{1,\dots,m\}} h^{-1}(v)$ and (2) for all $1 \leq i \neq j \leq k$, if there exists $1 \leq u \leq k$ and $x, y \in [0,1]^k$ such that $(\varphi(u)(x) \wedge x_i \neq 0)$ and $(\varphi(u)(y) \wedge y_j \neq 0)$, then $(h(i) = h(j) \iff V(i) = V(j))$; otherwise $h(i) \neq h(j)$. A deterministic CMC for S is the CMC $\varrho(S) = \langle\{1,\dots,m\}, o', \varphi', A, V'\rangle$ where $o' = h(o)$, $\forall 1 \leq i \leq k$, $V'(h(i)) = V(i)$, and for each $1 \leq i \leq m$,*

$$\varphi'(i)(y_1,\dots,y_m) = \exists x_1,\dots,x_k,$$
$$\bigvee_{u\in h^{-1}(i)} [(\forall 1 \leq j \leq m,\ y_j = \sum_{v\in h^{-1}(j)} x_v) \wedge \varphi(u)(x_1,\dots,x_k)].$$

Theorem 3.16. *Let S be a CMC in* single valuation normal form, *we have $S \preceq \varrho(S)$.*

Proof. Let $S = \langle\{1,\dots,k\}, o, \varphi, A, V\rangle$ be a CMC in single valuation normal form. Let $\varrho(S) = \langle\{1,\dots,m\}, o', \varphi', A, V'\rangle$ be a determinization of S and $h : \{1,\dots,k\} \to \{1,\dots,m\}$ the associated projection.

Define $\mathcal{R} \subseteq \{1,\dots,k\} \times \{1,\dots,m\}$ such that $u\,\mathcal{R}\,v \iff h(u) = v$. We will show that $\mathcal{R}$ is a strong refinement relation. Let u, v such that $u\,\mathcal{R}\,v$.

1. By definition, we have $h(u) = v$, thus $V'(v) = V(u)$.
2. Let $\Delta \in [0,1]^{k\times m}$ such that $\Delta_{i,j} = 1$ if $h(i) = j$ and 0 else. Δ is clearly a correspondence matrix.
   (a) Let $x \in [0,1]^k$ such that $\varphi(u)(x)$. For all $1 \leq j \leq m$, we have $y_j = \sum_{i\in h^{-1}(j)} x_i$ and $\varphi(u)(x)$, thus $\varphi'(v)(x \times \Delta)$. Moreover, for all $1 \leq i \leq k$, $\sum_{j=1}^{m} \Delta_{i,j} = 1$ by construction.
   (b) If $\Delta_{u',v'} \neq 0$, then $h(u') = v'$ and thus $u'\,\mathcal{R}\,v'$.

Finally, $\mathcal{R}$ is a strong refinement relation and $o\,\mathcal{R}\,o'$, thus S strongly refines $\varrho(S)$. As strong refinement implies weak refinement, we also have $S \preceq \varrho(S)$.

□

We now state the main theorem of the section, and one of the central results of the chapter: the weak refinement is complete with respect to implementation set inclusion for deterministic CMCs in single valuation normal form (recall that it is sound for all CMCs):

Theorem 3.17. *Let $S_1 = \langle\{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle\{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be two consistent and deterministic CMCs in single valuation normal form with $A_2 \subseteq A_1$. We have $[\![S_1]\!] \subseteq [\![S_2]\!] \Rightarrow S_1 \preceq S_2$.*

We suppose that the CMCs we consider in this proof are pruned. Moreover we only consider CMCs in single valuation normal form. Given two CMCs S_1 and S_2 such that $[\![S_1]\!] \subseteq [\![S_2]\!]$, we prove that $S_1 \preceq S_2$. The proof is structured as following.

1.
   - We define the relation $\mathcal{R}$ between S_1 and S_2.

     $$R = \{(v, u) \mid \forall I, \forall p \in I,\ p \models v \Rightarrow p \models u\}$$

     We consider u and v such that $v \,\mathcal{R}\, u$ and prove that $\mathcal{R}$ satisfies Axiom (1) of the refinement relations.
   - Axiom (2) of the weak refinement relations : Given a distribution X on the outgoing transitions of v, we must find a correspondence matrix Δ satisfying Axioms $2(a)$, $2(b)$ and $2(c)$ of the refinement relation :
     - We consider a distribution X on the outgoing transitions from v and we build a MC I satisfying S_1 such that the outgoing probabilities of the state v_I are exactly X.
     - This leads to $v_I \models u$ and gives a correspondence matrix Δ_2, which we will take as our correspondence matrix Δ.
     - By definition, Δ satisfies the axioms $2(a)$ and $2(b)$ of the weak refinement relations.

2. As Δ comes from a satisfaction relation, the axiom $2(c)$ of the refinement relation is not so immediate. It tells us that if a coefficient $\Delta_{v'u'}$ is not 0, then there exists an implementation I and a state v'_I such that $v'_I \models v'$ and $v'_I \models u'$. What we need is that for all implementations I' and state p' such that $p' \models v'$, we have $p' \models u'$. **The rest of the proof is dedicated to proving that this statement being false leads to a contradiction**.

   Assuming there exists I' and p' such that $p' \models v'$ and $p' \not\models u'$, we build an implementation $\widehat{I}$ from I and I' such that the state v' of $\widehat{I}$ is syntactically equivalent to the state p'. We then prove that this state v' of $\widehat{I}$ still satisfies the state u' of S_2 because it is a successor of v and S_2 is deterministic. As the state v' of $\widehat{I}$ is syntactically equivalent to the state p' of I', this means that $p' \models u'$, which is a contradiction.

We now go through the mathematical foundations of this proof.

Proof.

Let $S_1 = \langle \{1, \dots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \dots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be two consistent and deterministic CMCs in single valuation normal form such that $A_2 \subseteq A_1$ and $[\![S_1]\!] \subseteq [\![S_2]\!]$.

First, remark that $S_1 \preceq S_2 \iff S'_1 = \langle \{1, \dots, k_1\}, o_1, \varphi_1, A_2, V_1{\downarrow_{A_2}} \rangle \preceq S_2$. It is thus safe to suppose that $A_1 = A_2$. Similarly, if $I = \langle \dots, , A_I, V_I \rangle$ is a MC, we have $I \models S_1 \iff I' = \langle \dots, , A_1, V_I{\downarrow_{A_1}} \rangle \models S_1$. As a consequence, it is also safe to suppose that implementations have the same set of atomic propositions as S_1 and S_2.

1. Let $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ such that $v \mathcal{R} u$ iff for all MC I and state p of I, $p \models v \Rightarrow p \models u$. As we consider pruned CMCs, there exist implementations for all states.

   Consider v and u such that $v \mathcal{R} u$.

   (a) By definition of $\mathcal{R}$, there exists a MC I and a state p of I such that $p \models v$ and $p \models u$. Thus $V_I(p) \in V_1(v)$ and $V_I(p) \in V_2(u)$. As S_1 and S_2 are in single valuation normal form, $V_1(v)$ and $V_2(u)$ are singletons, so $V_1(v) = V_2(u)$.

   (b) Consider $x \in [0,1]^{1 \times k_1}$ such that $\varphi_1(v)(x)$ and build the MC $I = \langle \{1, \ldots, k_1\}, o_1, M, A_1, V_1' \rangle$ such that for all $1 \leq w \leq k_1$,

   - $V_1'(w)$ is the only valuation T such that $V_1(w) = \{T\}$;
   - If $w \neq v$, the line M_w is any solution of $\varphi_1(w)$. One exists because S_1 is pruned;
   - $M_v = x$.

   When necessary, we will address state w of I as w_I to differentiate it from state w of S_1. We will now build the correspondence matrix Δ.

   I clearly satisfies S_1 with a satisfaction relation $\mathcal{R}_1 = \mathsf{Identity}$, and $v_I \models v$. By hypothesis, we thus have $v_I \models u$. Consider $\mathcal{R}_2$ the satisfaction relation such that $v_I \mathcal{R}_2 u$ and Δ_2 the corresponding correspondence matrix. Let $\Delta = \Delta_2$.

   (c) As a consequence,

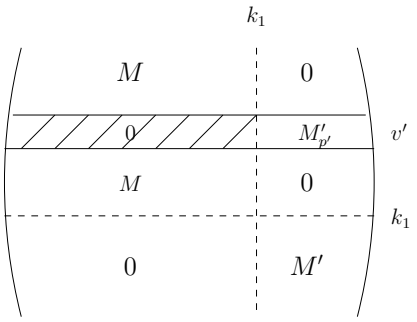
   i. $\forall 1 \leq i \leq k_1, x_i \neq 0 \Rightarrow \sum_{j=1}^{k_2} \Delta_{ij} = 1$;
   ii. $\varphi_2(u)(x \times \Delta)$ holds;

2. Let v' be a state of S_1 such that If $x_{v'} \neq 0$ and $\Delta_{v'u'} \neq 0$. By definition of I and Δ, we have $v_I' \models v'$ and $v_I' \models u'$. We want to prove that for all implementations I' and state p' in I', $p' \models v'$ implies $p' \models u'$.

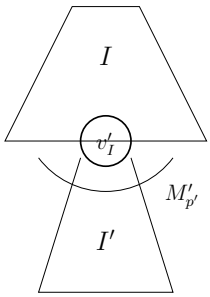
   Suppose this is not the case. There exists an implementation $I' = \langle \{1, \ldots, n\}, o', M', A_1, V' \rangle$ and a state p' of I' such that $p' \models v'$ and $p' \not\models u'$. Let $\mathcal{R}'$ be the correspondence matrix witnessing $p' \models v'$.

   Consider the MC $\widehat{I} = \langle \{1, \ldots, k_1, k_1+1, \ldots, k_1+n\}, o_I, \widehat{M}, A_1, \widehat{V} \rangle$. Intuitively, the first k_1 states correspond to I and the next n states to I'. The state v_I' will be the link between the two and its outgoing transitions will be the ones of p'. Define

   - $\widehat{M}_{ij} = M_{i,j}$ if $1 \leq i, j \leq k_1$ and $i \neq v'$
   - $\widehat{M}_{v'j} = 0$ if $1 \leq j \leq k_1$
   - $\widehat{M}_{ij} = 0$ if $1 \leq i \leq k_1$ and $i \neq v'$ and $j > k_1$
   - $\widehat{M}_{v'j} = m'_{p',j-k_1}$ if $j > k_1$
   - $\widehat{M}_{ij} = 0$ if $i > k_1$ and $1 \leq j \leq k_1$
   - $\widehat{M}_{ij} = m'_{i-k_1,j-k_1}$ if $i > k_1$ and $j > k_1$.



(a) The transition matrix $\widehat{M}$



(b) The MC $\widehat{I}$

- $\widehat{V}(i) = V'_1(i)$ if $i \leq k_1$
- $\widehat{V}(i) = V'(i - k_1)$ if $i > k_1$

We want to prove that $v'_{\widehat{I}}$ satisfies u'. This should imply that $p'_{I'}$ also satisfies u', which is absurd.

Consider the relation $\widehat{\mathcal{R}}$ between the states of $\widehat{I}$ and the states of S_1 defined as follows :

$$\begin{aligned}\widehat{\mathcal{R}} =& \{(q, w) \in R_1 \mid q \neq v'\} \cup \\ & \{(q, w) \mid (q - k_1)\, \mathcal{R}'\, w\} \cup \\ & \{(v', w) \mid p'\, \mathcal{R}'\, w\}\end{aligned}$$

Intuitively, $\widehat{\mathcal{R}}$ is equal to $\mathcal{R}_1$ for the states $q \leq k_1$, except v', and equal to $\mathcal{R}'$ for the states $q > k_1$. The states related to $v'_{\widehat{I}}$ are the ones that were related to p' with $\mathcal{R}'$.

We will show that $\widehat{\mathcal{R}}$ is a satisfaction relation between $\widehat{I}$ and S_1.

Let q, w such that $q\widehat{\mathcal{R}}w$. For all the pairs where $q \neq v'_{\widehat{I}}$, the conditions of the satisfaction relation obviously still hold because they held for $\mathcal{R}_1$ if $q \leq k_1$ and for $\mathcal{R}'$ otherwise. It remains to check the conditions for the pairs where $q = v'_{\widehat{I}}$.

Consider w such that $v'_{\widehat{I}}\widehat{\mathcal{R}}w$.

(a) Because (v'_I) and $(p'_{I'})$ are both implementations of v', it is clear that $\widehat{V}(v'_{\widehat{I}}) = \widehat{V}(p')$. As $p'\, \mathcal{R}'\, w$, we know that $V'(p') \in V_1(w)$. Thus, $\widehat{V}(v'_{\widehat{I}}) \in V_1(w)$.

(b) Consider the correspondence matrix Δ' given by $p'\, \mathcal{R}'\, w$. Let $\widehat{\Delta} \in [0, 1]^{(k_1+n)\times k_1}$ such that $\widehat{\Delta}_{ij} = 0$ if $i \leq k_1$, and $\widehat{\Delta}_{ij} = \Delta'_{(i-k_1)j}$ otherwise.

i. We want to show that if $\widehat{M}_{(v'_{\widehat{I}})(w')} \neq 0$, then $\sum_{j=1}^{k_1} \widehat{\Delta}_{w'j} = 1$. We know that $\widehat{M}_{(v'_{\widehat{I}})(w')} = 0$ if $w' \leq k_1$. Take $w' > k_1$ such that $\widehat{M}_{(v'_{\widehat{I}})(w')} \neq 0$. Then we know that $\widehat{M}_{(v'_{\widehat{I}})(w')} = M'_{p'(w'-k_1)}$. Because $\mathcal{R}'$ is a satisfaction relation, it implies that $\sum_{j=1}^{k_1} \Delta'_{(w'-k_1)j} = 1$. Thus, $\sum_{j=1}^{k_1} \widehat{\Delta}_{w'j} = \sum_{j=1}^{k_1} \Delta'_{(w'-k_1)j} = 1$.

ii. We want to show now that $\varphi_1(w)(\widehat{M}_{v'_{\widehat{I}}} \times \widehat{\Delta})$ holds. Let $1 \leq j \leq k_1$. We have

$$\begin{aligned} M\widehat{M}_{v'_{\widehat{I}}} \times \widehat{\Delta} t_j &= \sum_{l=1}^{k_1+n} \widehat{M}_{(v'_{\widehat{I}})l} \cdot \widehat{\Delta}_{lj} \\ &= 0 + \sum_{l=k_1+1}^{k_1+n} \widehat{M}_{(v'_{\widehat{I}})l} \cdot \widehat{\Delta}_{lj} \\ &= \sum_{l=1}^{n} M'_{p'l} \cdot \Delta'_{lj} = [M'_{p'} \times \Delta']_j \end{aligned}$$

As a consequence, $\widehat{M}_{v'_{\widehat{I}}} \times \widehat{\Delta} = M'_{p'} \times \Delta'$. Since Δ' is a witness of $p' \,\mathcal{R}'\, w$, $\varphi_1(w)(M'_{p'} \times \Delta')$ holds. So does $\varphi_1(w)(\widehat{M}_{v'_{\widehat{I}}} \times \widehat{\Delta})$.

iii. We want to show that if $\widehat{M}_{(v'_{\widehat{I}})q} \neq 0$ and $\widehat{\Delta}_{qw'} \neq 0$, then $q\widehat{\mathcal{R}}w'$. We only need to consider $q > k_1$ (since otherwise $\widehat{M}_{(v'_{\widehat{I}})q} = 0$) and w' such that $\widehat{\Delta}_{qw'} \neq 0$. In this case, $\widehat{M}_{(v'_{\widehat{I}})q} = M'_{p'(q-k_1)} \neq 0$ and $\Delta'_{(q-k_1)w'} \neq 0$. As Δ' is a witness of $p' \,\mathcal{R}'\, w$, it has to be that $(q-k_1) \,\mathcal{R}'\, w'$, which implies, by definition of $\widehat{\mathcal{R}}$, that $q\widehat{\mathcal{R}}w'$.

Finally $\widehat{I}$ satisfies S_1, and in particular, $v_{\widehat{I}} \models v$. As $v \,\mathcal{R}\, u$, it implies that $v_{\widehat{I}} \models u$. As a consequence, there exists $\Delta'' \in [0,1]^{(k_1+n)\times k_2}$ such that $\varphi_2(u)(\widehat{M}_{v_{\widehat{I}}} \times \Delta'')$.

(A) Consider $u'' \neq u'$ such that $V_2(u'') = V_2(u')$. Due to determinism of S_2, and to the fact that u' is accessible from u, we have $[\widehat{M}_{v_{\widehat{I}}} \times \Delta'']_{u''} = 0$. Since $\widehat{M}_{(v_{\widehat{I}})(v'_{\widehat{I}})} \neq 0$ and $\widehat{M}_{(v_{\widehat{I}})(v'_{\widehat{I}})} \cdot \Delta''_{(v'_{\widehat{I}})u''}$ is part of $[\widehat{M}_{v_{\widehat{I}}} \times \Delta'']_{u''}$, we must have $\Delta''_{(v'_{\widehat{I}})u''} = 0$.

(B) Consider u''' such that $V(u''') \neq V(u')$. It is clear that $\Delta''_{(v'_{\widehat{I}})u'''} = 0$ since Δ'' is witnessing satisfaction between $\widehat{I}$ and S_2.

(C) Moreover, we know that $\widehat{M}_{(v_{\widehat{I}})(v'_{\widehat{I}})} \neq 0$. Thus, $\sum_{j=1}^{k_2} \Delta''_{v'_{\widehat{I}}j} = 1$.

According to (A) and (B), the only non-zero value in the sum in (C) must be $\Delta''_{(v'_{\widehat{I}})u'}$. Since Δ'' is witnessing $\widehat{I} \models S_2$, this means that $v'_{\widehat{I}} \models u'$.

By construction, $v'_{\widehat{I}}$ and p' only differ by state names. This contradicts the assumption that $p' \not\models u'$. Thus $v' \,\mathcal{R}\, u'$, and $\mathcal{R}$ is a weak refinement relation.

Finally, we have by hypothesis that $[\![S_1]\!] \subseteq [\![S_2]\!]$, which implies that $o_1 \,\mathcal{R}\, o_2$.

□

Since any consistent CMC with a single valuation in initial state can be normalized, Theorem 3.17 holds even if S_1 and S_2 are not in single valuation normal form, but only have a single valuation in the initial state. Thus, weak refinement and the implementation set inclusion coincide on the class of deterministic CMCs with at most single valuation in the initial state. Finally, Theorem 3.17 also holds for strong refinement. Indeed, the following theorem states that weak and strong refinements coincide on the class of deterministic CMCs.

Theorem 3.18. *Let $S_1 = \langle\{1,\dots,k_1\}, o_1, \varphi_1, A, V_1\rangle$ and $S_2 = \langle\{1,\dots,k_2\}, o_2, \varphi_2, A, V_2\rangle$ be two deterministic CMCs in normal form. If there exists a weak refinement relation $\mathcal{R}$ such that $S_1 \,\mathcal{R}\, S_2$, then $\mathcal{R}$ is also a strong refinement relation.*

We start with the following lemma, which is a direct consequence of the notion of determinism. It states that correspondence matrices associated to a satisfaction relation for a deterministic CMC have at most one non-zero value per row.

Lemma 3.19. *Let $S = \langle\{1,\dots,k\}, o_S, \varphi, A, V_S\rangle$ be a deterministic CMC in single valuation normal form. Let $C = \langle\{1,\dots,n\}, o_C, M, A, V_C\rangle \in [\![S]\!]$ and a satisfaction relation $\mathcal{R}$ such that $o_C \,\mathcal{R}\, o_S$. Let $p \in \{1,\dots,n\}$ and $u \in \{1,\dots,k\}$ such that $p \,\mathcal{R}\, u$, and let Δ be the associated correspondence matrix. We have*

$$\forall p' \in \{1,\dots,n\},\ M_{pp'} \neq 0 \Rightarrow |\{u' \in \{1,\dots,k\} \mid \Delta_{p'u'} \neq 0\}| = 1.$$

Let $S_1 = \langle\{1,\dots,k_1\}, o_1, \varphi_1, A, V_1\rangle$ and $S_2 = \langle\{1,\dots,k_2\}, o_2, \varphi_2, A, V_2\rangle$ be two deterministic CMCs in normal form such that $S_1 \preceq S_2$ with a weak refinement relation $\mathcal{R}$. We prove that $\mathcal{R}$ is in fact a strong refinement relation.

Proof.

Let $v \in \{1,\dots,k_1\}$ and $u \in \{1,\dots,k_2\}$ such that $v \,\mathcal{R}\, u$.

1. By hypothesis, $V_1(v) \subseteq V_2(u)$;
2. We know that for all $x \in [0,1]^{k_1}$ satisfying $\varphi_1(v)$, there exists a correspondence matrix Δ^x satisfying the axioms of a (weak) refinement relation. We will build a correspondence matrix Δ^0 that will work for all x. Let $p \in \{1,\dots,k_1\}$.
   - If for all $x \in [0,1]^{k_1}$, $\varphi_1(v)(x) \Rightarrow x_p = 0$, then let $\Delta^0_p = (0,\dots,0)$.
   - Else, consider $x \in [0,1]^{k_1}$ such that $\varphi_1(v)(x)$ and $x_p \neq 0$. By hypothesis, there exists a correspondence matrix Δ^x associated to $v \,\mathcal{R}\, u$. Let $\Delta^0_p = \Delta^x_p$. By Lemma 3.19, there is a single $u' \in \{1,\dots,k_2\}$ such that $\Delta^x_{pu'} \neq 0$. Moreover, by definition of Δ^x, we know that $\sum_{r=1}^{k_2} \Delta^x_{pr} = 1$, thus $\Delta^x_{pu'} = 1$.

     Suppose there exists $y \neq x \in [0,1]^{k_1}$ such that $\varphi_1(v)(y)$ and $y_p \neq 0$. Let Δ^y be the associated correspondence matrix. As for x, there exists a unique $u'' \in \{1,\dots,k_2\}$ such that $\Delta^y_{pu''} \neq 0$. Moreover $\Delta^y_{pu''} = 1$. Let $x' = x \times \Delta^x$ and $y' = y \times \Delta^y$. By definition, both $\varphi_2(v)(x')$ and $\varphi_2(y')$ hold, $x'_{u'} \neq 0$ and $y'_{u''} \neq 0$. As $\Delta^x_{pu'} = \Delta^y_{pu''} = 1$, we have $V_2(u') \cap V_2(u'') \neq \emptyset$. By hypothesis, S_2 is deterministic, thus $u' = u''$.

     As a consequence, we have $\Delta^x_p = \Delta^y_p$, so $\forall z \in [0,1]^{k_1},\ (\varphi_1(v)(z) \wedge (z_p \neq 0)) \Rightarrow \Delta^z_p = \Delta^0_p$.

     Finally, consider Δ^0 defined as above. Let $x \in [0,1]^{k_1}$ such that $\varphi_1(v)(x)$. We have

(a) $x_i \neq 0 \Rightarrow \Delta_i^0 = \Delta_i^x \Rightarrow \sum_{j=1}^{k_2} \Delta_{ij}^0 = 1$;

(b) $x \times \Delta^0 = x \times \Delta^x$, thus $\varphi_2(v)(x \times \Delta^0)$ holds;

(c) If $\Delta_{v'u'}^0 \neq 0$, then there exists $y \in [0,1]^{k_1}$ such that $\varphi_1(v)(y)$ and $\Delta_{v'u'}^0 = \Delta_{v'u'}^y$, thus $v' \mathcal{R} u'$.

Finally, $\mathcal{R}$ is a strong refinement relation.

□

Finally, we remark that the above results on completeness for deterministic specifications carry over to IMCs, proving Theorem 2.9 of Chapter 2. These results also translate to refinements of [86] and [61], which are special cases of our refinements. Completeness properties for these refinements were open problems until now.

Discussion: A weaker Definition of Determinism. The notion of determinism presented here may look too strong. Indeed, it assumes that, from a given state i, one cannot reach two states u and v that share common sets of valuations. The assumption is made independently of the distributions used to reach the two states, i.e., it may be the case that there exists no distribution in where both u and v can be reached simultaneously. As presented in Chapter 2, there is another natural way to solve the problem: consider a weaker version of determinism, that would be equivalent to the notion of determinism introduced in Chapter 2. More precisely, we say that a CMC $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \}$ is weakly deterministic if whenever there exists $x \in [0,1]^k$ and states i, u, v such that $\varphi(i)(x)$ and $x_u > 0$ and $x_v > 0$, we have $V(u) \cap V(v) = \emptyset$. This version of determinism is weaker than the one given in Definition 3.13. Indeed, only states that can be reached by the same distribution should have disjoint sets of valuations.

In Chapter 2, it is proven that the two notions coincide for the particular case of IMCs. However, this is not the case for CMCs because the constraint do not necessarily allow only convex solutions.

Moreover, though this notion seems reasonable, the CMCs S_c and S_d given in Figures 3.8a and 3.8b are both weakly deterministic, and S_c thoroughly but not weakly refines S_d. Hence working with this weaker, but natural, version of determinism does not close the gap between weak and thorough refinements.

3.7 Polynomial CMCs

It is not surprising that CMCs are closed under both logical and structural compositions. Indeed, CMCs do not make any assumptions on constraint functions. There are however many classes of constraints that are practically intractable. While this chapter is mainly concerned with the development of the theoretical foundations for CMCs, we now briefly study classes of CMCs for which operations on constraints required by our algorithms can be managed quite efficiently.

A first candidate could be linear constraints, which is the obvious generalization of interval constraints. Unfortunately, linear constraint CMCs are not closed under structural composition. Indeed, as we have seen in Section 3.4 the composition of two linear constraints leads to a polynomial constraint. However, what is more interesting is that polynomial constraints *are* closed under both logical and structural composition and that these operations do not increase the quantifier alternations since they only introduce existential quantifiers. Hence, one can

claim that CMCs with polynomial constraints and only existential quantifiers are certainly the smallest extension of IMCs closed under all operations.

From the algorithmic point of view, working with polynomial constraints should not be seen as an obstacle. First, we observe that algorithms for logical and structural composition do not require any complex operations on polynomials. The refinement algorithms (presented in Section 3.3) are polynomial in the number of states, and each iteration requires a quantifier elimination. This procedure is known to be double exponential in general, but there exist efficient single exponential algorithms [27, 28] when quantifier alternations are fixed. Those algorithms are implemented in Maple [135]. The pruning operation is polynomial in the number of states, but each iteration also requires an exponential treatment as one has to decide whether the constraints have at least a solution. Again, such problem can be solved with efficient algorithms. Finally, determinizing a CMC can be performed with a procedure that is similar to the determinization procedure for finite-state automata. Such a procedure is naturally exponential in the number of states.

Remark 3.3. *In Section 3.4, it was shown that, assuming independent sets of valuations, parallel composition is refined by conjunction. We have also observed that the conjunction or disjunction of two linear constraints remains linear, but that composition may introduce polynomial constraints. From an implementation point of view it may thus be more efficient to work with linear constraints only. For doing so, one can simply approximate composition with conjunction.*

3.8 On the relation with Probabilistic Automata

CMCs are a newcomer in a long series of probabilistic modeling languages and abstractions for them. Throughout the chapter we have indicated that many of our results directly translate to simpler abstractions, like IMCs. We shall now further discuss this foundational aspect of CMCs, showing how they subsume a few established notions of refinement and composition for probabilistic automata (and for process algebra based on them).

Below we write $\mathrm{Dist}(S)$ for the set of all probability distributions over a finite set S. Given two sets S and T and a probability distribution $\alpha \in \mathrm{Dist}(S \times T)$, we denote the marginal distribution over S as $\alpha_{s,T} = \sum_{t \in T} \alpha_{s,t}$, and similarly for T. We say that φ is a *non-deterministic distribution constraint* over set I if all solutions x of φ are point distributions, i.e. $\exists i.\, x_i = 1$. Write $[{}^{i}_{S}]$ to denote a particular point distribution for which $[{}^{i}_{S}]_i = 1$. Notice that non-deterministic distribution constraints model a non-deterministic choice of an element from S. They will be used to encode non-determinism in CMCs.

A probabilistic automaton (PA for short) [121] is a tuple $\mathbb{S} = (S, \mathit{Act}, \rightarrow, s_1)$, where S is a finite set of states, $\rightarrow \subseteq S \times \mathit{Act} \times \mathrm{Dist}(S)$ is a finite transition relation and $s_1 \in S$ is the initial state. The *derived combined transition relation* of $\mathbb{S}$ is given by $\rightarrow_c \in S \times \mathrm{Act} \times \mathrm{Dist}(S)$. If $\pi \in \mathrm{Dist}(S)$ and $\varrho \in \mathrm{Dist}(T)$ then $\pi \otimes \varrho$ denotes the unique independent product distribution such that $(\pi \otimes \varrho)_{s,t} = \pi_s \cdot \varrho_t$.

We say that $t \xrightarrow{a}_c \varrho$ iff ϱ is a convex linear combination of vectors from $\boldsymbol{\varrho} = \{\varrho_i \mid t \xrightarrow{a} \varrho_i\}$, so $\varrho = \boldsymbol{\varrho} \times \lambda$, where λ is a distribution vector $\lambda \in [0,1]^{|\boldsymbol{\varrho}|}$. We interpret $\boldsymbol{\varrho}$ as a matrix, where ith column is a distribution ϱ_i.

Consider two PA $\mathbb{S} = (S, \mathit{Act}, \rightarrow^S, s_0)$ and $\mathbb{T} = (T, \mathit{Act}, \rightarrow^T, t_0)$. For a binary relation $R \subseteq S \times T$ we define a derived relation $R^* \subseteq \mathrm{Dist}(S) \times \mathrm{Dist}(T)$ such that $\pi R^* \varrho$ iff there exists

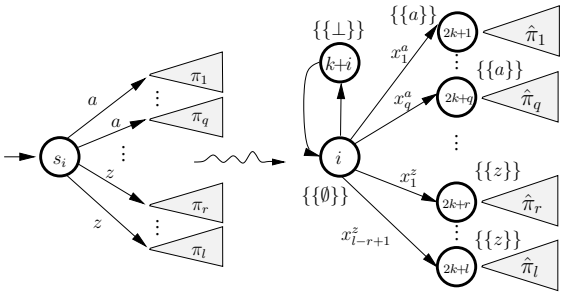


Figure 3.14: Reducing a PA to CMC. There $\widehat{\pi}$ denotes a distribution constraint, which has a unique solution π.

a distribution $\alpha \in \text{Dist}(S \times T)$ and (1) $\alpha_{q,T} = \pi_q$ for all $q \in S$, (2) $\alpha_{S,r} = \varrho_r$ for all $r \in T$ and (3) $\alpha_{s,t} \neq 0$ implies sRt.

Definition 3.15 (Simulation [121])**.** *A relation $R \subseteq S \times T$ is a* simulation *iff $(s,t) \in R$ implies that whenever $s \xrightarrow{a} \pi$ for a distribution π, then $t \xrightarrow{a} \varrho$ for distribution ϱ such that $\pi R^* \varrho$.*

R is a probabilistic *simulation iff $(s,t) \in R$ implies that if $s \xrightarrow{a} \pi$ then $t \xrightarrow{a}_c \varrho$ for some distribution ϱ, and $\pi R^* \varrho$.*

Let $A \subseteq \mathit{Act}$ be the subset of actions on which $\mathbb{S}$ and $\mathbb{T}$ should synchronize. The *parallel composition* of $\mathbb{S}$ and $\mathbb{T}$ is a PA $\mathbb{S} \parallel \mathbb{T} = (S \times T, \mathit{Act}, \rightarrow, (s_0, t_0))$, where $\rightarrow$ is the largest transition relation such that $(s,t) \xrightarrow{a} \pi \otimes \varrho$ if: $a \in A$ and $s \xrightarrow{a} {}^S \pi$ and $t \xrightarrow{a} {}^T \varrho$, or

$a \notin A$ and $s \xrightarrow{a} {}^S \pi$ and $\varrho = [{}^t_T]$, or
$a \notin A$ and $\pi = [{}^s_S]$ and $t \xrightarrow{a} {}^T \varrho$.

3.8.1 Reduction from Simulation

We now propose a linear encoding of PAs into CMCs, which reduces simulation and composition of PAs to refinement and composition of CMCs (see Fig. 3.14). Let $\mathbb{S} = (\{s_1, \ldots, s_k\}, \mathit{Act}, \rightarrow, s_0)$ be a PA. And let l be the number of reachable action-distribution pairs, so $\Omega_{\mathbb{S}} = \{(a_1, \pi_1), \ldots, (a_l, \pi_l)\} = \{(a, \pi) \mid \exists s \in S.\, s \xrightarrow{a} \pi\}$. The corresponding CMC is $\widehat{\mathbb{S}} = (\{1, \ldots, 2k{+}l\}, 1, \widehat{\varphi}, \text{Act} \cup \bot, \widehat{V})$, where $\bot \notin \mathit{Act}$. $\widehat{\mathbb{S}}$ has three kinds of states. Type-1 states, $1 \ldots k$, correspond directly to states of $\mathbb{S}$. Distributions leaving these states model a non-deterministic choice. Type-2 states, $k+1, \ldots, 2k$, model a possibility that a component remains idle in a state. Type-3 states, $2k{+}1, \ldots, 2k{+}l$ model the actual distributions of $\mathbb{S}$.

$\widehat{V}$ assigns value $\{\emptyset\}$ to type-1 states and value $\{\{\bot\}\}$ to type-2 states. For type-3: $\widehat{V}(2k + i') = \{\{a_{i'}\}\}$ for $1 \leq i' \leq l$. The distribution constraints are as follows:

$\widehat{\varphi}(i)(x)$ if i is type-1 and $x = [{}^{k+i}_{1..2k+l}]$ or $s_i \xrightarrow{a_{i'}} \pi_{i'} \wedge x = [{}^{2k+i'}_{1..2k+l}]$ for $1 \leq i' \leq l$.
$\widehat{\varphi}(k+i)(x)$ if $k+i$ is type-2 and $x = [{}^{i}_{1..2k+l}]$.
$\widehat{\varphi}(2k+i')(x)$ if $2k+i'$ is type-3 and $x = \pi_{i'}$.

We can now relate simulation of PA to refinement of CMCs:

Theorem 3.20. *$\mathbb{T}$ simulates $\mathbb{S}$ iff $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$.*

We begin by demonstrating a lemma about non-deterministic distribution constraints.

We say that a constraint is a single-point constraint, if it is only satisfied by a unique distribution. Observe that all constraints in the encoding presented in Section 3.8 are non-deterministic distribution constraints or single-point constraints.

Lemma 3.21. *Let φ and ψ be single-point constraints. If for each $x \in [0,1]^{1\times k_1}$ such that $\varphi(x)$ holds, there exists a correspondence matrix $\Delta_x \in [0,1]^{k_1\times k_2}$ such that $\psi(x \times \Delta_x)$ holds then there exists a correspondence matrix $\Delta \in [0,1]^{k_1\times k_2}$ such that for all $x \in [0,1]^{1\times k_1}$ we have that $\varphi(x) \implies \psi(x \times \Delta)$.*

The lemma holds trivially because there is only one distribution satisfying φ.

Lemma 3.22. *Let φ (respectively ψ) is a non-deterministic distribution constraint over $\{1,\ldots,k_1\}$ (respectively $\{1,\ldots,k_2\}$). Then if for each distribution vector x satisfying φ there exists a correspondence matrix $\Delta_x \in [0,1]^{k_1\times k_2}$ such that $\psi(x \times \Delta_x)$ holds then there exists a correspondence matrix $\Delta \in [0,1]^{k_1\times k_2}$ such that for all $x \in [0,1]^{1\times k_1}$ we have that $\varphi(x) \implies \psi(x \times \Delta)$.*

Proof. Let x be such that $\varphi(x)$ holds (then there exists $1 \le i \le k_1$ such that $x_i = 1$). There is a finite number of such vectors. Let x^i denote the one that has 1 on the ith position. Take Δ such that $\Delta_i = (\Delta_{x^i})_i$ (the witness from the lemma assumption) if x^i satisfies φ and $\Delta_i = 0^{1\times k_2}$ otherwise.

Now for each x^i satisfying φ we have that $x^i \times \Delta = x^i \times \Delta_{x^i}$ and then $\varphi(x^i) \implies \psi(x^i \times \Delta_{x^i}) \iff \psi(x^i \times \Delta)$. □

Corollary 3.23. *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ we have that $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$ iff $\widehat{\mathbb{S}}$ weakly refines $\widehat{\mathbb{T}}$.*

Lemma 3.24. *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ such that $\mathbb{T}$ simulates $\mathbb{S}$ we have that $\widehat{\mathbb{S}}$ weakly refines $\widehat{\mathbb{T}}$.*

Proof. (sketch) Let $R \subset S \times T$ be the relation witnessing the simulation of $\mathbb{S}$ by $\mathbb{T}$. Consider a relation Q as follows:

$$
\begin{aligned}
Q_1 &= \{(i,j) \mid i \in \{1,\ldots,k_1\}, j \in \{1,\ldots,k_2\}, (s_i,t_j) \in R\}\\
Q_2 &= \{(k_1+i, k_2+j) \mid i \in \{1,\ldots,k_1\}, j \in \{1,\ldots,k_2\}, (s_{i-k_1}, t_{j-k_2}) \in R\}\\
Q_3 &= \{(2k_1+i', 2k_2+j') \mid i' \in \{1,\ldots,l_1\}, j' \in \{1,\ldots,l_2\}, (a_i,\pi_i) \in \Omega_S, (a_j,\varrho_j) \in \Omega_T,\\
&\qquad a_i = a_j, (\pi_i,\varrho_i) \in R^*\}\\
Q &= Q_1 \cup Q_2 \cup Q_3
\end{aligned}
$$

It is easy to show that Q is a weak refinement. First observe that valuations always match for pairs in Q. The valuation is empty for both S and T in Q_1, it is $\{\bot\}$ in Q_2, and $\{a_i\}$ in Q_3.

For a pair in $(i,j) \in Q_1$ a distribution vector x satisfying the constraint of S is always a point distribution. If $x_{k_1+i} = 1$, take $\Delta_{k_1+i,k_2+j} = 1$ and zero otherwise. If $x_{2k_1+i'} = 1$ take $\Delta_{2k_1+i',2k_2+j'} = 1$ and zero otherwise, where j' is such that $t_{j'} \xrightarrow{a_{i'}} \varrho_{j'}$ and $\pi_{i'} R^* \varrho_{j'}$.

For a pair $(k_1+i, k_2+j) \in Q_2$ take $\Delta_{i,j} = 1$, and zero otherwise.

For a pair $(2k_1+i', 2k_2+j') \in Q_3$ take Δ such that for $(i,j) \in \{1,\ldots,k_1\} \times \{1,\ldots,k_2\}$ we have $\Delta_{ij} = \alpha_{ij}/x_i$, or zero if $x_i = 0$, where α is the distribution witnessing $\pi_{i'} R^* \varrho_{j'}$. □

Lemma 3.25. *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ such that $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$ we have that $\mathbb{T}$ simulates $\mathbb{S}$.*

Proof. (sketch) Assume that $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$ is witnessed by a relation $R \subseteq \{1,\ldots,2k_1+l_1\} \times \{1,\ldots,2k_2+l_2\}$. Show that a relation $Q = \{(s_i,t_j) \in S \times T \mid (i,j) \in R, i \in \{1,\ldots,k_1\}, j \in \{1,\ldots,k_2\}\}$ is a simulation relation.

In the crucial point of the proof consider $\alpha_{s_i,t_j} = \Delta_{i,j} \cdot \pi_{i'}(s_i)$, where $\pi_{i'}$ is a distribution being the only solution of a point constraint for state $i' \in \{2k_1,\ldots,2k_2+l_1\}$. □

Theorem 3.20 follows as a corollary from the above two lemma and the Corollary 3.23.

Another, very similar, but slightly more complicated, encoding exists, for which weak refinement coincides with *probabilistic* simulation. It will be presented at the end of this section.

The same encoding is used to characterize parallel composition of PAs using parallel composition of CMCs.

Theorem 3.26. *For two PAs $\mathbb{S}$ and $\mathbb{T}$ over the same set of synchronizing actions Act and a set $A \subseteq Act$ we have that $\widehat{\mathbb{S} \parallel \mathbb{T}}$ is isomorphic to*

$$((\widehat{\mathbb{S}} \parallel \widehat{\mathbb{T}}[{}^{a'}\!/_{a}]_{a\in Act}) \wedge \mathsf{S}_A)\,[{}^{a}\!/_{(a,a')};\ {}^{a}\!/_{(a,\perp')};\ {}^{a}\!/_{(\perp,a')}]_{a\in Act}$$

where S_A is a synchronizer over $Act_\perp \times Act'_{\perp'}$ defined by

$$(\forall a \in A.\ a \Longleftrightarrow a') \wedge (\forall a \notin A.\ (a \Longrightarrow \perp') \wedge (a' \Longrightarrow \perp)).$$

Expression $S[{}^{a'_1}\!/_{a_1};\ldots;{}^{a'_n}\!/_{a_n}]_{a_1,\ldots,a_n\in Act}$ denotes a substitution, substituting a primed version of name a_i for each occurrence in a_i, for all actions in *Act*.

Interestingly, the precongruence property for the parallel composition of PAs is obtained for free as a corollary of the above two reduction theorems and Thm. 3.8. Similarly, we obtain precongruence with probabilistic simulation using a suitable encoding—a good example how CMCs can be used to study properties of simpler languages in a generic way.

3.8.2 Encoding Probabilistic Simulation

We now present another encoding of PAs into CMCs, which aims at capturing probabilistic simulation (as opposed to simulation).

Consider a PA $\mathbb{S} = (S, Act, \rightarrow, s_1)$, where $S = \{s_1,\ldots,s_k\}$. Let $\{(s^1,a_1),\ldots,(s^l,a_l)\} = \{(s,a) \mid s \in S \wedge a \in Act\}$. The corresponding CMC is

$$\check{\mathbb{S}} = (\{1,\ldots,2k+l\}, 1, \check{\varphi}, Act \cup \perp, \check{V}\}),$$

where $\perp$ is a fresh symbol not in *Act* . We have three types of states (see Figure 3.15). Type-1 states, $\{1,\ldots,k\}$, correspond directly to states $\{s_1,\ldots,s_k\}$—their distribution constraints encode the non-deterministic choice of action. Type-2 states, $\{k+1,\ldots,2k\}$, represent ability of a state to be idle. We will use them in parallel composition. Type-3 states, $\{2k+1,\ldots,2k+l\}$, encode choice of a probability distribution as a linear combination of distributions allowed by the automaton.

The valuation functions are given by:

$$\begin{aligned}
&\check{V}(i) = \{\emptyset\} && \text{for } 1 \le i \le k,\\
&\check{V}(k+i) = \{\{\perp\}\} && \text{for } 1 \le i \le k,\\
&\check{V}(2k+i') = \{\{a_{i'}\}\} && \text{for } 1 \le i' \le l.
\end{aligned}$$

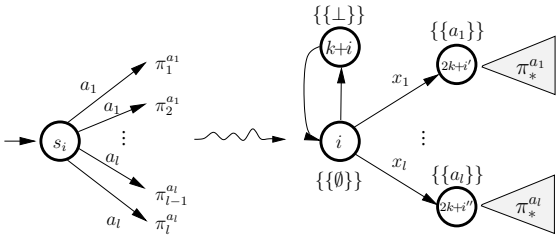


Figure 3.15: An attempt to visualize the second encoding. π_*^a denotes a constraint expressing a probability vector that is a linear combination of all probability distributions labeled by a. Below this is formalized as $\varphi(2k+i')(x)$.

and

$$\begin{aligned}
&\check{\varphi}(i)(x) \text{ is } x_{k+i} = 1 \text{ or } \exists 1 \leq i' \leq l.\ x_{2k+i'} = 1 \ \wedge\ s^{i'} = s_i \\
&\qquad \text{for } 1 \leq i \leq k \quad \text{(type-1 states)}, \\
&\check{\varphi}(k+i)(x) \text{ is } x_i = 1 \\
&\qquad \text{for } 1 \leq i \leq k \quad \text{(type-2 states)}, \\
&\check{\varphi}(2k+i')(x) \text{ is } \exists \lambda \in \mathrm{Dist}(1, \ldots, |\boldsymbol{\pi}|).\ x = \boldsymbol{\pi}\lambda \\
&\qquad \text{for } 1 \leq i' \leq l \quad \text{(type-3 states)},
\end{aligned}$$

where $\boldsymbol{\pi} = \{\pi \mid s^j \xrightarrow{a_j} \pi\}$. Technically speaking $\boldsymbol{\pi}$ is a matrix, whose columns are distributions π. We write $|\boldsymbol{\pi}|$ for the number of columns in $\boldsymbol{\pi}$. Additionally x is implicitly required to be a probability distribution over $\{1, \ldots, 2k+l\}$.

Observe that $\check{\mathbb{S}}$ is only polynomially larger than $\mathbb{S}$.

Lemma 3.27 (Soundness)**.** *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ such that $\check{\mathbb{S}}$ weakly refines $\check{\mathbb{T}}$, we have that $\check{\mathbb{T}}$ probabilistically simulates $\check{\mathbb{S}}$.*

Proof. Let $\mathbb{S} = (S, \mathit{Act}, \rightarrow^S, s_1)$ and $\mathbb{T} = (T, \mathit{Act}, \rightarrow^T, t_1)$, with $S = \{s_1, \ldots, s_{k_1}\}$ and $T = \{t_1, \ldots, t_{k_2}\}$. In the proof we write $\check{\varphi}$ to refer to the constraint function of $\check{\mathbb{S}}$, and $\check{\varrho}$ to refer to the constraint function of $\check{\mathbb{T}}$. Also l_1 and l_2 are used to refer to the number of combinations of state-action of respectively $\check{\mathbb{S}}$ and $\check{\mathbb{T}}$. Finally q_i and r_j are used to range over states in S (respectively in T), when s_i and t_j are bound to some concrete value.

Let $R \in \{1, \ldots, 2k_1 + l_1\} \times \{1, \ldots, 2k_2 + l_2\}$ be a weak refinement relation between $\check{\mathbb{S}}$ and $\check{\mathbb{T}}$, witnessing the assumption of the lemma. The proof proceeds by showing that

$$Q = \{(s_i, t_j) \mid (i,j) \in R \wedge 1 \leq i \leq k_1 \wedge 1 \leq j \leq k_2\}$$

is a probabilistic simulation relation between $\mathbb{S}$ and $\mathbb{T}$.

We apply the usual coinductive proof technique. Take $(s_i, t_j) \in Q$. Let $\pi \in \mathrm{Dist}(S)$ be such that $s_i \xrightarrow{a} \pi$, and $(s^{i'}, a_{i'}) = (s_i, a)$.[1]

By construction of the encoding we know that any probability distribution x satisfying $\varphi(i)(x)$ is a point distribution, and x such that $x_{2k+i'} = 1$ is possible. So consider such a

[1]The equality binds i' to be the index of (s_i, a) on the list of state-action pairs in the encoding of $\mathbb{S}$.

distribution x. Since $(i,j) \in R$ we know that there exists a correspondence matrix $\Delta \in [0,1]^{2k_1+l_1 \times 2k_2+l_2}$ such that $\psi(j)(x \times \Delta)$ holds. Moreover $x \times \Delta$ must be a point distribution by construction of the encoding. So $(x \times \Delta)_{2k_2+j'} = 1$ for some $1 \leq j' \leq l_2$. And, by refinement again, we get that valuation functions for both $2k_1 + i'$ and for $2k_2 + j'$ both return $\{\{a\}\}$ and that $(2k_1 + i', 2k_2 + j') \in R$.

But $\check{\mathbb{T}}$ is also constructed using the encoding, so it necessarily is that $t_j \xrightarrow{a} \varrho$ for some $\varrho \in \text{Dist}(T)$.

Observe that $\varphi(2k_1 + i')(\pi)$ holds, because π is always a convex linear combination of a set of vectors containing it. Since $(2k_1 + i', 2k_2 + j') \in R$, there exists a correspondence matrix $\Delta' \in [0,1]^{2k_1+l_1 \times 2k_2+l_2}$ such that $\psi(2k_2 + j')(\pi \times \Delta')$ holds. The latter implies that $\pi \times \Delta'$ is a linear combinations of vectors in $\boldsymbol{\varrho} = \{\varrho \mid t_j \xrightarrow{a} \varrho\}$.

It remains to show that $\pi R^* (\pi \times \Delta')$. Take $\alpha_{q_i,q_j} = \pi_i \cdot \Delta'_{ij}$. We first argue that $\alpha \in \text{Dist}(S \times T)$. Clearly $\pi_i \Delta'_{ij} \in [0,1]$ for all i,j. Also $\sum_{i=1}^{k_1} \sum_{j=1}^{k^2} \pi_i \Delta'_{ij} = \sum_{i=1}^{k_1} \pi_i = 1$ (the former because each row of a correspondence matrix sums up to 1).

Consider $\alpha_{q_i,T} = \sum_{j=1}^{k_2} \alpha_{q_i,t_j} = \sum_{j=1}^{k_2} \pi_i \cdot \Delta'_{ij} = \pi_i \sum_{j=1}^{k_2} \Delta'_{ij} = \pi_i$ as required by $\pi R^* (\pi \times \Delta')$.

Now consider $\alpha_{S,r_j} = \sum_{i=1}^{k_1} \alpha_{s_i,r_j} = \sum_{i=1}^{k_1} \pi_i \cdot \Delta'_{ij} = (\pi \times \Delta')_j$ as required by $\pi R^* (\pi \times \Delta')$.

Now if $\alpha_{q_i,r_j} \neq 0$ then $\Delta'_{ij} \neq 0$, which in turn with refinement of $2k_2 + j'$ by $2k_1 + i'$ implies that $(i,j) \in R$, and furthermore $(s_i, s_j) \in Q$ by construction, as required by $\pi R^* (\pi \times \Delta')$. This finishes the proof.

□

Lemma 3.28 (Completeness)**.** *For any two probabilistic automata $\mathbb{S}$ and $\mathbb{T}$ such that $\mathbb{T}$ probabilistically simulates $\mathbb{S}$, we have that $\check{\mathbb{S}}$ weakly refines $\check{\mathbb{T}}$.*

Proof. Let $\mathbb{S} = (S, \mathit{Act}, \rightarrow^S, s_1)$ and $\mathbb{T} = (T, \mathit{Act}, \rightarrow^T, t_1)$, with $S = \{s_1, \ldots, s_{k_1}\}$ and $T = \{t_1, \ldots, t_{k_2}\}$. Let $Q \subseteq S \times T$ be the probabilistic simulation relation between $\mathbb{S}$ and $\mathbb{T}$, witnessing the assumption of the lemma.

The proof proceeds by showing that a relation $R \subseteq \{1, \ldots, 2k_1 + l_1\} \times \{1, \ldots, 2k_2 + l_2\}$ is a weak refinement relation between $\check{\mathbb{S}}$ and $\check{\mathbb{T}}$.

Take the following candidate for R:

$$
\begin{aligned}
R_1 &= \{(i,j) \mid (s_i, t_j) \in Q\} \\
R_2 &= \{(k_1 + i, k_2 + j) \mid (s_i, t_j) \in Q\} \\
R_3 &= \{(2k_1 + i', 2k_2 + j') \mid (s_i, t_j) \in R \wedge s_i = s^{i'} \wedge t_j = t^{j'}\} \\
R &= R_1 \cup R_2 \cup R_3
\end{aligned}
$$

We apply the usual coinductive proof technique.

Case 1. Take $(i,j) \in R_1$ and x satisfying $\varphi(i)(x)$. We know that x can only be a point-distribution. If $x_{k_1+i} = 1$ then we take Δ such that $\Delta_{k_1+i,k_2+j} = 1$ (and Δ is zero for all other cells). Clearly Δ is a correspondence matrix. Moreover $x \times \Delta$ is a point distribution with 1 on $(k_2 + j)$th position, so $\psi(j)(x \times \Delta)$ holds by construction of the encoding (see first case in encoding of constraints). Also $(k_1 + i, k_2 + j) \in R_2$ since $(s_i, t_j) \in Q$.

If $x_{2k_1+i'} = 1$ then it means that $s_i \xrightarrow{\check{V}(i)} \pi$ for some π and action $\check{V}(i)$. But then, since $(s_i, t_j) \in Q$, it is possible that $t_j \xrightarrow{\check{V}(i)}_c \varrho$,for some distribution ϱ. Let j' be such that $t_j = t^{j'}$

and $a_{j'} = \check{V}(i)$. Take a correspondence matrix Δ such that $\Delta_{2k_1+i',2k_2+j'} = 1$ (and Δ is zero for all other cells). We have that $x \times \Delta$ is a point distribution with 1 on $2k_2 + j'$th position, so $\psi(j)(x \times \Delta)$ holds by construction of encoding resulting in j (see first case in encoding of constraints). Also $(2k_1 + i', 2k_2 + j') \in R_3 \subseteq R$ by definition of R_3.

Case 2. Take $(k_1 + i, k_2 + j) \in R_2$. The argument is almost identical to the first subcase in Case 1. We omit it here.

Case 3. Take $(2k_1 + i', 2k_2 + j') \in R_3$ and x satisfying $\varphi(2k_1 + i')(x)$. Let $s_i = s^{i'}$ and $t_j = t^{j'}$. By R_3 we know that $(s_i, t_j) \in Q$. By construction of the encoding $s_i \xrightarrow{\check{V}(2k_1+i')} x$ and furthermore $t_j \xrightarrow{\check{V}(2k_1+i')}_c \varrho$, where $\varrho = \boldsymbol{\varrho} \times \lambda$ for some probability distribution $\lambda \in \mathrm{Dist}(1, \ldots, |\boldsymbol{\varrho}|)$. Clearly $\psi(2k_2 + j')(\varrho) = 1$. It remains to check that π can be correspondence to ϱ.

To this end consider a correspondence matrix Δ such that

$$\Delta_{ij} = \begin{cases} \alpha_{s_i,t_j}/x_i & \text{if } x_i \neq 0 \text{ and } i \leq k_1, j \leq k_2 \\ 0 & \text{otherwise} \end{cases}$$

Now $(x \times \Delta)_j = \sum_{i=1}^{2k_1+l_1} x_i \Delta_{ij} = \sum_{i=1}^{k_1} x_i \cdot \alpha_{s_i,t_j}/x_i = \sum_{i=1}^{k_1} \alpha_{s_i,t_j} = \alpha_{S,t_j} = \varrho_j$ by $xR^*\varrho$ (this discussion only holds for $j \leq k_2$, but the remaining cells are zero, which is easy to argue for. Also somewhat sloppily we ignored the possibilty of division by zero – indeed it cannot happen since for $x_i = 0$ we said that Δ_{ij} is simply zero). Effectively $x \times \Delta = \varrho$, so it satisfies $\psi(2k_2 + j')$. Valuations obviously match.

Moreover if $\Delta_{ij} \neq 0$ then $\alpha_{s_i,t_j} \neq 0$. then $(s_i, t_j) \in Q$ and then $(i, j) \in R_1 \subseteq R$, which finishes the proof. □

Theorem 3.29 is a corollary from Lemmas 3.27 and 3.28.

Theorem 3.29. *$\mathbb{T}$ probabilistically simulates $\mathbb{S}$ iff $\check{\mathbb{S}}$ weakly refines $\check{\mathbb{T}}$.*

Similarly, we obtain precongruence with probabilistic simulation using a suitable encoding.

3.9 Related Work and Concluding Remarks

In this chapter, we have presented CMCs—a new model for representing a possibly infinite family of MCs. Unlike the previous attempts [86, 61], our model is closed under many design operations, including composition and conjunction. We have studied these operations as well as several classical compositional reasoning properties, showing that, among others, the CMC specification theory is equipped with a complete refinement relation (for deterministic specifications), which naturally interacts with parallel composition, synchronization and conjunction. We have also demonstrated how our framework can be used to obtain properties for less expressive languages, by using reductions.

Two recent contributions [61, 90] are related to our work. Fecher et al. [61] propose a model checking procedure for PCTL [36] and Interval Markov Chains (other procedures recently appear in [34, 69]), which is based on weak refinement. However, our objective is not to use CMCs within a model checking procedure for probabilistic systems, but rather as a specification theory.

Very recently Katoen and coauthors [90] have extended Fecher's work to *Interactive* Markov Chains, a model for performance evaluation [80, 82]. Their abstraction uses the continuous

time version of IMCs [89] augmented with may and must transitions, very much in the spirit of [100]. Parallel composition is defined and studied for this abstraction, however conjunction has been studied neither in [61] nor in [90].

Over the years process algebraic frameworks have been proposed for describing and analyzing probabilistic systems based on Markov Chains (MCs) and Markov Decision Processes [79, 8, 103]. Also a variety of probabilistic logics have been developed for expressing properties of such systems, e.g., PCTL [73]. Both traditions support refinement between specifications using various notions of probabilistic simulation [61, 86] and, respectively, logical entailment [81]. Whereas the process algebraic approach favors structural composition (parallel composition), the logical approach favors logical composition (conjunction). Neither of the two supports *both* structural and logical composition.

There are many directions in which we can still contribute both for IMCs and CMCs. First, it would be interesting to see whether the results presented in Chapter 2 extend to the continuous-time model of [89, 90]. Another interesting future work would be to extend these results to other specification formalisms for systems that mix both stochastic and non-deterministic aspects. Among them, one finds probabilistic automata [112] where weak/strong refinement would be replaced by (probabilistic) simulation [121, 87].

It would also be of interest to design, implement and evaluate efficient algorithms for procedures outlined both in this chapter and in Chapter 2. Defining a quotient relation for CMCs, presumably building on results presented in [101], seems an important next step. The quotienting operation is of particular importance for component reuse [116, 113, 114, 25, 115]. One could also investigate applicability of our approach in model checking procedures, in the same style as Fecher and coauthors have used IMCs for model checking PCTL [61].

Another interesting direction consists in using CMCs or IMCs as abstraction models for solving stochastic games, following the approach initiated by Larsen et al in [49]. We could also propose to use our IMCs or CMCs in an abstraction-based probabilistic model checking procedure [42, 96, 71]. For this purpose, it would be important to study the logical fragment that can be expressed using CMCs.

We should also mix our results with those recently obtained for timed specifications [50, 23, 22], hence leading to the first theory for specification of timed probabilistic systems [98].

In the spirit of [56], it would be interesting to extend our composition operation by considering products of dependent probability distributions. Finally, one should propose a more quantitative version of the refinement operation like this will be done for contracts in Chapter 4.

Chapter 4

Probabilistic contracts: a compositional reasoning methodology for the design of stochastic systems

4.1 Introduction

In [21], Benveniste et al. have proposed a component-based design theory called *contracts*. An *assume-guarantee* contract is a structure that, contrary to interface automata [54, 52] and modal transition systems [100], allows to distinguish hypotheses on a component (*guarantees*) from hypotheses made on its environment (*assumptions*). This explicit separation allows defining a more elaborate satisfaction relation than the ones defined for interface or modal theories. Moreover, the authors of [21] use a language theoretic abstraction of systems behavior to represent both assumptions and guarantees, hence allowing to represent more general properties than the classical graphical-based models.

In this chapter we will focus on developing a contract-based compositional theory for two classes of systems, that are (1) non-stochastic and possibly non-deterministic, and (2) stochastic and possibly non-deterministic. As in classical non-modular verification [37, 129], the satisfaction relation will be Boolean for non-stochastic systems and quantitative otherwise, hence leading to two notions of contracts. In addition, we will consider two notions of satisfaction, namely *reliability* and *availability*. Availability is a measure of the average time during which a system satisfies a given property, for all possible runs of the system. In contrast, reliability is a measure of the set of runs of a system that satisfy a given property. Both quantities play an important role when designing, for instance, mission-critical systems. Our notion of satisfaction is assumption-dependent in the sense that runs that do not satisfy the assumptions are considered to be "correct". This interpretation, which has been suggested by many industrial partners, is needed to propose compositional design operations such as conjunction.

Aside from the satisfaction relation, any good contract-based theory should also support the following requirements.

1. *Refinement and shared refinement. Refinement* of contracts expresses inclusion of sets of models, and therefore allows to compare contracts.

2. *Structural composition.* The contract theory should also provide a combination operator on contracts, reflecting the standard composition of models by, *e.g.* parallel product.

3. *Logical composition/conjunction.* Different aspects of systems are often specified by different teams. The issue of dealing with multiple aspects or multiple viewpoints is thus essential. It should be possible to represent several contracts (viewpoints) for the same system, and to combine them in a logical/conjunctive fashion.

The theory should also support incremental design (contracts can be composed/conjunct in any order) and independent implementability (composable contracts can always be refined separately) [55].

We propose mathematical definitions for composition, conjunction and refinement. It is in fact known that most of industrial requirements[1] for component-based design translate to those operations. Composition between contracts, which mimics classical composition for systems, consists in taking the intersection between the assumptions and the intersection between the guarantees. Conjunction produces a contract whose assumptions are the union of the original ones and guarantees are the intersection of the original ones. We say that a contract refines another contract if it guarantees more and assumes less. The definition is Boolean for non-probabilistic systems and quantitative otherwise.

We also establish a *compositional reasoning verification* theory for those operations and the two notions of satisfiability we consider. This methodology allows to reason on the entire design by only looking at individual components. The theory differs with the type of contracts under consideration. As an example, we will show that if a non-stochastic system S_1 reliably satisfies[2] a contract C_1 and a non-stochastic system S_2 reliably satisfies a contract C_2, then the composition of the two systems reliably satisfies the composition of the two contracts. When moving to stochastic systems, we will show that if S_1 satisfies C_1 with probability α and S_2 satisfies C_2 with probability β, then their composition satisfies the composition of C_1 and C_2 with probability at least $\alpha + \beta - 1$. The theory is fully general as it assumes that both systems and contracts are represented by sets of runs.

Our last contribution is to propose effective and symbolic representations for contracts and systems. Those representations rely on an automata-based representation of possibly infinite sets of runs. Assuming that assumptions and guarantees are represented with Büchi automata (which allows to specify assumptions and guarantees with logics such as LTL [108] or PSL [60]), we observe that checking if a (stochastic) system satisfies a reliability property can be done with classical techniques implemented in tools such as SPIN [127] or LIQUOR [35]. We show that satisfaction of availability properties can be checked with an extension of the work presented in [53]. Finally, we also show that operations between and on contracts can easily be performed on the automata-based representations.

4.2 Preliminaries

In this section, we recap some definitions and concepts related to automata theory. We then introduce some notations and concepts that will be used in the rest of the chapter.

Let Σ be an alphabet. A finite word over Σ is a mapping $w : \{0, \ldots, n-1\} \to \Sigma$. An *infinite word* (or ω-*word*) w over Σ is a mapping $w : \mathbb{N} \to \Sigma$. An automaton is a tuple $A = (\Sigma, Q, Q_0, \delta, F)$, where Σ is a finite alphabet, Q is a set of *states*, $Q_0 \in Q$ is the

[1] Example: those of the European projects COMBEST [45] and SPEEDS [126].

[2] "Reliably satisfy" means that all the runs that satisfy the assumption must satisfy the guarantee.

set of *initial states*, $\delta : Q \times \Sigma \rightarrow 2^Q$ is a *transition function* ($\delta : Q \times \Sigma \rightarrow Q$ if the automaton is deterministic), and $F \subseteq Q$ is a set of *accepting* states. A *finite run* of A on a finite word $w : \{0, \ldots, n-1\} \rightarrow \Sigma$ is a labeling $\varrho : \{0, \ldots, n\} \rightarrow Q$ such that $\varrho(0) \in Q_0$, and $(\forall 0 \leq i \leq n-1)(\varrho(i+1) \in \delta(\varrho(i), w(i)))$. A finite run ϱ is *accepting* for w if $\varrho(n) \in F$. An *infinite run* of A on an infinite word $w : \mathbb{N} \rightarrow \Sigma$ is a labeling $\varrho : \mathbb{N} \rightarrow Q$ such that $\varrho(0) \in Q_0$, and $(\forall 0 \leq i)(\varrho(i+1) \in \delta(\varrho(i), w(i))$. An infinite run ϱ is *accepting* for w with the Büchi condition if $\mathit{inf}(\varrho) \cap F \neq \emptyset$, where $\mathit{inf}(\varrho)$ is the set of states that are visited infinitely often by ϱ. We distinguish between finite-word automata that are finite automata accepting finite words, and Büchi automata [29] that are finite automata accepting infinite words. A finite-word automaton accepts a finite word w if there exists an accepting finite run for w in this automaton. A Büchi automaton accepts an infinite word w if there exists an accepting infinite run for w in this automaton. The set of words accepted by A is called the *language accepted by* A, and is denoted by $L(A)$. Finite-word and Büchi automata are closed under intersection and union. Inclusion and emptiness are also decidable. Both finite-word and Büchi automata are closed under complementation and, in both cases, the construction is known to be exponential. However, the complementation operation for Büchi automata requires intricate algorithms that not only are worst-case exponential, but are also hard to implement and optimize (see [130] for a survey).

Let $\mathbb{N}_\infty = \mathbb{N} \cup \{\omega\}$ be the closure of the set of natural integers and $\mathbb{N}_n = [0 \ldots n-1]$ the interval ranging from 0 to $n-1$. Let V be a finite set of *variables* that takes values in a *domain* D. A *step* $\sigma : V \rightarrow D$ is a valuation of variables of V. A *run* on V is a sequence of valuations of variables of V. More precisely, a finite or infinite run is a mapping $w : \mathbb{N}_n \rightarrow V \rightarrow D$, where $n \in \mathbb{N}_\infty$ is the length of w, also denoted $|w|$. Let ε be the run of length 0. Given a variable $v \in V$ and a time $i \geq 0$, the value of v at time i is given by $w(i)(v)$. Given w a finite run on V and σ a step on the same variables, $w.\sigma$ is the run of length $|w|+1$ such that $\forall i < |w|$, $(w.\sigma)(i) = w(i)$ and $(w.\sigma)(|w|) = \sigma$. The set of all finite (respectively infinite) runs on V is denoted by $[V]^*$ (respectively $[V]^\omega$). The set of finite and infinite runs on V is denoted $[V]^\infty = [V]^* \cup [V]^\omega$. Denote $[V]^n$ (respectively $[V]^{\leq n}$) the set of all runs on V of length exactly n (respectively not greater than n). The *complement* of $\Omega \subseteq [V]^\infty$ is given by $\neg\Omega = [V]^\infty \setminus \Omega$. The *projection* of w on $V' \subseteq V$ is the run $w \downarrow_{V'}$ such that $|w \downarrow_{V'}| = |w|$ and $\forall v \in V', \forall n \geq 0$, $w \downarrow_{V'} (n)(v) = w(n)(v)$. Given a run w' on V', the *inverse-projection* of w' on V is the set of runs defined by $w' \uparrow^V = \{w \in [V]^\infty \mid w \downarrow_{V'} = w'\}$. A *system* over V is a pair (V, Ω), where Ω is a set of (finite and/or infinite) runs on V. Let $S = (V, \Omega)$ and $S' = (V', \Omega')$ be two systems. The *composition* of S and S', denoted $(V, \Omega) \cap (V', \Omega')$, is given by $(V \cup V', \Omega'')$ with $\Omega'' = \Omega \uparrow^{V \cup V'} \cap \Omega' \uparrow^{V \cup V'}$. The *complement* of S, denoted $\neg S$, is given by $\neg S = (V, \neg\Omega)$. The restriction of system $S = (V, \Omega)$ to runs of length not greater than $n \in \mathbb{N}_\infty$ (respectively exactly n) is the system $S|^{\leq n} = (V, \Omega \cap [V]^{\leq n})$ (respectively $S|^n = (V, \Omega \cap [V]^n)$). In Section 4.4, it will be assumed that systems can respond to every possible input on a set of probabilistic variables. Such systems are said to be *receptive* to those variables. Given $U \subseteq V$, a set of distinguished variables, system $S = (V, \Omega)$ is *U-receptive* if and only if for all finite run $w \in \Omega \cap [V]^*$ and for all input $\varrho : U \rightarrow D$, there exists a step $\sigma : V \rightarrow D$ such that $\sigma \downarrow_U = \varrho$ and $w.\sigma \in \Omega$. Given $U \subseteq V \cap V'$, two U-receptive systems $S = (V, \Omega)$ and $S' = (V', \Omega')$ are U-compatible if and only if $S \cap S'$ is U-receptive.

A *symbolic transition system* over V is a tuple $Symb = (V, Q_s, T, Q_{s0})$, where V is a set of variables defined over a *finite* domain D, Q_s is a set of states (a state is a mapping from V to D), $T \subseteq Q_s \times Q_s$ is the transition relation, and $Q_{s0} \subseteq Q_s$ is the set of initial states. A run of $Symb$ is a possibly infinite sequence of states $q_{s0} q_{s1} \ldots$ such that for each $i \geq 0$ $(q_{si}, q_{s(i+1)}) \in T$

and $q_{s0} \in Q_{s0}$. A symbolic transition system for a system (V, Ω) is a symbolic transition system over V whose set of runs is Ω. Operations of (inverse) projection and intersection easily extend from systems to their symbolic representations (such representation may not exist). Let $\mathcal{B}_A = (\Sigma, Q, Q_0, \delta, F \subseteq Q)$ be an automaton such that Σ is a mapping $V \rightarrow D$. The *synchronous product* between $\mathcal{B}_A$ and $Symb$ is the automaton $\mathcal{B}_{\mathcal{B}_A \times Symb} = (\emptyset, Q', Q'_0, \delta', F')$, where $Q' = Q_s \times Q$, $Q'_0 = Q_{s0} \times Q_0$, $(a', b') \in \delta'((a, b), \emptyset)$ iff $(a, a') \in T$ and $b' \in \delta(b, a)$, $F' = \{(a, b) \in Q' | b \in F\}$. Each state in the product is a pair of states: one for $Symb$ and one for $\mathcal{B}_A$. If we do not take the information from $\mathcal{B}_A$ into account, a run of the product corresponds to a run of $Symb$.

4.3 Non-Probabilistic Contracts

In this section, we introduce the concept of contract for non-stochastic systems. We also study compositional reasoning for such contracts. We will present the theory in the most general case by assuming that contracts and systems are given by (pair of) possibly infinite sets of runs [21]. In practice, a finite representation of such sets is required and there are many ways to instantiate our theory depending on this representation. At the end of the section, we will give an example of such a representation. More precisely, we will follow a successful trend in Model Checking and use automata as a finite representation for systems and contracts. We will also derive effective algorithms based on this symbolic representation.

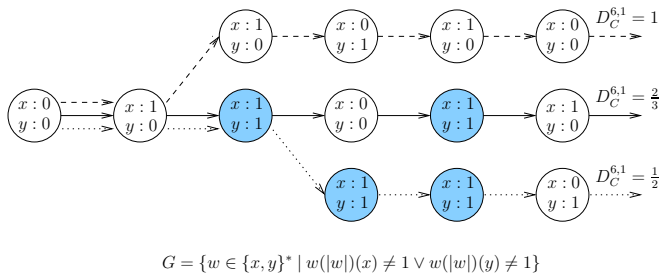
4.3.1 Contracts

We first recap the concept of *contract* [20], a mathematical representation that allows to distinguish between assumptions made on the environment and properties of the system.

Definition 4.1 (Contract)**.** *A contract over V is a tuple $C = (V, A, G)$, where V is the set of variables of C, system $A = (V, \Omega_A)$ is the* assumption *and system $G = (V, \Omega_G)$ is the* guarantee*.*

The contract C is said to be in *canonical form* if and only if $\neg A \subseteq G$. As we shall see in Section 4.3.2, the canonical form is needed to have uniform notions of composition and conjunction between contracts.

We now turn to the problem of deciding whether a system satisfies a contract. A system that satisfies a contract is an *implementation* of the contract. There are two types of implementation relations, depending on the property captured by a contract. A first possible interpretation is when the contract represents properties that are defined on runs of the system. This includes safety properties. In this context, a system satisfies a contract if and only if all system runs that satisfy the assumption are included in the guarantee. This applies to reliability properties, and a system implementing a contract in this way is said to *R-satisfy* the contract. Another possible interpretation is when the contract represents properties that are defined on finite prefixes of the runs of the system and when one wants to evaluate how often the system satisfies the contract. We will say that a system *A-satisfies* a contract with level m ($0 \leq m \leq 1$) if and only if for each of its runs, the proportion of prefixes of system runs that are either in the guarantee or in the complement of the assumption is greater or equal to m. This concept can be used to check *average safeness* or *reliability*, i.e., to decide for each run whether the average number of positions of the run that do satisfy a local condition is greater or equal to a given threshold.



Mean-availability until position 6 is computed for the runs of the system w.r.t a contract with assumption $\{x, y\}^*$ and guarantee the set of finite runs over $\{x, y\}$ such that in the final state $x \neq 1$ or $y \neq 1$. Positions where the contract is satisfied are white.

Figure 4.1: Illustration of mean-availability.

Definition 4.2 (R-Satisfaction)**.** *System $S = (U, \Omega)$ R-satisfies contract $C = (V, A, G)$ up to time $t \in \mathbb{N}_\infty$, denoted $S \models^{R(t)} C$, if and only if $S|^{\leq t} \cap A \subseteq G$.*

Discussion. In this chapter, we assume that runs that do not satisfy the assumptions are "good" runs, i.e., they do not need to satisfy the guarantee. In our theory, assumptions are thus used to distinguish runs that must satisfy the property from those that are not forced to satisfy the property. There are other interpretations of the paradigm of assume/guarantee in which the runs that do not satisfy the assumptions are considered to be bad. We (and our industrial partners) believe that our definition is a more natural interpretation as there is no reason to eliminate runs on which no assumption is made. Another advantage of this approach, which will be made more explicit in Section 4.4, is that this interpretation allows to define a conjunction operation in the stochastic case.

The definition of A-satisfiability is more involved and requires additional notations. The objective is to compute an invariant measure of the amount of time during which the system satisfies a contract. This relation can be combined with *discounting*[3], which allows to give more weight to faults that arise in the early future. Let $w \in [V]^\infty$ be a (finite or infinite) run and $C = (V, A, G)$ be a contract. We define the function $\varphi_w^C : \mathbb{N}_{|w|} \to \{0, 1\}$ such that $\varphi_w^C(n) = 1 \iff w_{[0,n]} \in G \cup \neg A$. If we fix an horizon in time $t \in \mathbb{N}_\infty$ and a *discount factor* $d \leq 1$, define $D_C^{t,d}(w) = \frac{1}{t} \sum_{i=0}^{t} \varphi_w^C(i)$ if $d = 1$ and $D_C^{t,d}(w) = \frac{1-d}{1-d^{t+1}} \sum_{i=0}^{t} d^i \varphi_w^C(i)$ if $d < 1$. $D_C^{t,d}(w)$ is the mean-availability until position t along the execution corresponding to w with discount factor d. The concept is illustrated in Figure 4.1. A-Satisfaction can now be defined.

Definition 4.3 (A-Satisfaction)**.** *A system $S = (U, \Omega)$ A-satisfies at level m contract $C = (V, A, G)$ until position k with discount factor d, denoted $S \models_{d,m}^{A(k)} C$, iff:*

$$\min_{w \in (S\uparrow^{U\cup V})|^k} D_{C\uparrow^{U\cup V}}^{k,d}(w) \geq m \qquad \text{if } k < \omega$$

$$\inf_{w \in (S\uparrow^{U\cup V})|^k} \liminf_{t \to k} D_{C\uparrow^{U\cup V}}^{t,d}(w) \geq m \qquad \text{if } k = \omega.$$

[3]Discounting is a concept largely used in many areas such as economy.

It is easy to see that the limit in Definition 4.3 converges, since $D_C^{t,d} \geq 0$. In Section 4.3.4 we will propose techniques to check satisfiability for contracts that are represented with symbolic structures.

In the rest of the section, we propose definitions for composition, conjunction, and refinement. We also study compositional verification with respect to these definitions and the satisfaction relations we considered above.

4.3.2 Compositional reasoning

In this section, we first define operations between and on contracts and then propose a compositional reasoning framework for contracts. We start with the definition for *composition* and *conjunction*. Composition between contracts mimics classical composition between systems at the abstraction level. Informally, it consists in taking the intersection between the assumptions and the intersection between the guarantees. Conjunction is a more intriguing operation that has no translation at the level of systems; it consists in producing a contract whose assumptions are the union of the original ones and guarantees are the intersection of the original ones. Roughly speaking, the conjunction of two contracts represents their common requirements.

Definition 4.4. *Let $C_i = (V_i, A_i, G_i)$ with $i = 1, 2$ be two contracts in canonical form. We define*

- *The* parallel composition *between C_1 and C_2, denoted $C_1 \parallel C_2$, to be the contract* $(V_1 \cup V_2, A_1 \cap A_2 \cup \neg(G_1 \cap G_2), G_1 \cap G_2)$.
- *The* conjunction *between C_1 and C_2, denoted $C_1 \wedge C_2$, to be the contract* $(V_1 \cup V_2, A_1 \cup A_2, G_1 \cap G_2)$.

It is easy to see that both conjunction and composition preserve canonicity.

Discussion. As pointed out in [20], the canonical form is needed to have uniform notions of composition and conjunction between contracts. Indeed, consider two contracts $C_1 = (V, \emptyset, [V]^\infty)$ and $C_2 = (V, \emptyset, \emptyset)$. Observe that C_1 is in canonical form and C_2 is not. Assume also that any system can satisfy both C_1 and C_2. The parallel composition between C_1 and C_2 is the contract $(V, [V]^\infty, \emptyset)$. This contract can only be satisfied by the empty system. Consider now the contract $C_2' = (V, \emptyset, [V]^\infty)$, which is the canonical form for C_2. It is easy to see that the composition between C_1 and C_2' is satisfied by any system. Non-canonical contracts can also be composed. Indeed, the composition of two non-canonical contracts $C_1 = (V_1, A_1, G_1)$ and $C_2 = (V_2, A_2, G_2)$ is given by the following formula $C_1 \parallel_{nc} C_2 = (V_1 \cup V_2, (A_1 \cup \neg G_1) \cap (A_2 \cup \neg G_2), G_1 \cap G_2)$. Observe that this composition requires one more complementation operation, which may be computationally intensive depending on the data-structure used to represented A and G (see Section 4.3.4).

We now turn to the definition of *refinement*, which leads to a preorder relation on contracts.

Definition 4.5. *We say that C_1* refines *C_2 up to time $t \in \mathbb{N}_\infty$, denoted $C_1 \preceq^{(\leq t)} C_2$, if it guarantees more and assumes less, for all runs of length not greater than t: $A_1 \uparrow^{V_1 \cup V_2} \supseteq (A_2 \uparrow^{V_1 \cup V_2})|^{\leq t}$ and $(G_1 \uparrow^{V_1 \cup V_2})|^{\leq t} \subseteq G_2 \uparrow^{V_1 \cup V_2}$.*

Property 4.1. *By a simple inspection of Definitions 4.4 and 4.5, one observes that both conjunction and composition are associative, i.e.,* $C_1 \parallel (C_2 \parallel C_3) = (C_1 \parallel C_2) \parallel C_3$ *and* $C_1 \wedge (C_2 \wedge C_3) = (C_1 \wedge C_2) \wedge C_3$ *(incremental design). Consider* $C_2 \parallel C_3$ *(respectively,* $C_2 \wedge C_3$*). We also observe that if* $C_1 \preceq^{(\leq t)} C_2$*, then* $(C_1 \parallel C_3) \preceq^{(\leq t)} (C_2 \parallel C_3)$ *(respectively,* $(C_1 \wedge C_3) \preceq^{(\leq t)} (C_2 \wedge C_3)$*) (independent implementability).*

It is interesting to see that the conjunction of two contracts coincide with their *greatest lower bound* with respect to refinement preorder. Thus the following theorem.

Theorem 4.2. *Consider two contracts* C_1 *and* C_2*, we have that*

- $C_1 \wedge C_2 \preceq^{(\leq t)} C_1$ *and* $C_1 \wedge C_2 \preceq^{(\leq t)} C_2$*, and*
- *for each* C *such that* $C \preceq^{\leq t} C_1$ *and* $C \preceq^{\leq t} C_2$*, we have* $C \preceq^{\leq t} (C_1 \wedge C_2)$*.*

4.3.3 Compositional Verification

In this chapter, *compositional verification* refers to a series of results that allow to deduce correctness of a global system by observing its atomic components only. We start with the following theorem for reliability.

Theorem 4.3 ([20])**.** *Consider* S_1*,* S_2 *two systems and* C_1*,* C_2 *two contracts in canonical form. The following propositions hold for all* $t \in \mathbb{N}_\infty$*:*

- $S_1 \models^{R(t)} C_1$ *and* $S_2 \models^{R(t)} C_2$ *implies that* $(S_1 \cap S_2) \models^{R(t)} (C_1 \parallel C_2)$*;*
- $S_1 \models^{R(t)} C_1$ *and* $S_1 \models^{R(t)} C_2$ *iff* $S_1 \models^{R(t)} (C_1 \wedge C_2)$*;*
- $S_1 \models^{R(t)} C_1$ *and* $C_1 \preceq^{(\leq t)} C_2$ *implies that* $S_1 \models^{R(t)} C_2$*.*

The above theorem can thus be used to deduce satisfaction w.r.t. to conjunction or composition without computing the result of these operations explicitly. The double implication in the second item of the theorem is valid as conjunction is not defined at the level of systems. The theorem can also be used to decide satisfaction on a refined contract without performing any computation. By combining the definitions of composition, conjunction, refinement, and Theorem 4.3, we get the following corollary.

Corollary 4.4. *Let* S *be a system and* C_1, C_2, C_3 *be three contracts in canonical form. We have the following results.*

- $S \models^{R(t)} C_1 \parallel (C_2 \parallel C_3)$ *iff* $S \models^{R(t)} (C_1 \parallel C_2) \parallel C_3$*;*
- $S \models^{R(t)} C_1 \wedge (C_2 \wedge C_3)$ *iff* $S \models^{R(t)} (C_1 \wedge C_2) \wedge C_3$*;*
- *If* $C_1 \preceq^{(\leq t)} C_2$ *and* $S \models^{R(t)} (C_1 \parallel C_3)$ *(respectively,* $S \models^{R(t)} (C_1 \wedge C_3)$*), then* $S \models^{R(t)} (C_2 \parallel C_3)$ *(respectively,* $S \models^{R(t)} (C_2 \wedge C_3)$*).*

We now switch to the case of availability. We propose the following theorem that, for example, gives a lower bound on availability for conjunction and disjunction without computing them explicitly.

Theorem 4.5. *Consider S_1 and S_2 two systems and C_1, C_2 two contracts in canonical form. Let $d \leq 1$ be a discount factor. The following propositions hold for all $t \in \mathbb{N}_\infty$:*

- $S_1 \models^{A(t)}_{d,m_1} C_1$ *and* $S_2 \models^{A(t)}_{d,m_2} C_2$ *implies that* $(S_1 \cap S_2) \models^{A(t)}_{d,m_1+m_2-1} (C_1 \parallel C_2)$*;*
- $S_1 \models^{A(t)}_{d,m_1} C_1$ *and* $S_1 \models^{A(t)}_{d,m_2} C_2$ *implies that* $S_1 \models^{A(t)}_{d,m_1+m_2-1} (C_1 \wedge C_2)$*;*
- $S_1 \models^{A(t)}_{d,m} C_1$ *and* $C_1 \preceq^{(\leq t)} C_2$ *implies that* $S_1 \models^{A(t)}_{d,m} C_2$*.*

The above theorem is an extension of Theorem 4.3 to the case of availability. It is interesting that the double implication in item two of Theorem 4.3 does not remain valid in this extension. This is because of the definition of availability. Observe that the last item of Theorems 4.3 and 4.5 also stands if C_1 and C_2 are not in canonical form. Observe also that Theorem 4.5 calls for a direct extension of Corollary 4.4 to the case of availability. Before we give the proof for Theorem 4.5 and discuss the extension, we first recap the following classical algebraic properties.

Property 4.6. *Consider $V \subseteq V' \subseteq V''$ three sets of variables and E and E'' two sets of runs over V and V'' respectively. We have:*

$$(E \uparrow^{V'}) \uparrow^{V''} = E \uparrow^{V''}; \tag{4.6:1}$$
$$(E \uparrow^{V''}) \downarrow_{V'} = E \uparrow^{V'}; \tag{4.6:2}$$
$$(E'' \downarrow_{V'}) \downarrow_{V} = E \downarrow_{V}; \tag{4.6:3}$$
$$w \in E'' \Rightarrow w \downarrow_V \in E'' \downarrow_V; \tag{4.6:4}$$
$$w \in E \Rightarrow w \uparrow^{V'} \subseteq E \uparrow^{V'}. \tag{4.6:5}$$

We now give the proof of Theorem 4.5.

Proof of Theorem 4.5.

For the sake of simplicity, we will consider that $k = \omega$. The proofs for $k < \omega$ are simpler versions of those presented here. We consider the three items of the theorem.

1. Let $S = (U, \Omega) = S_1 \cap S_2$ and $C = (V, A, G) = C_1 \parallel C_2$. Since C_1 and C_2 are contracts in canonical form, we have $G_1 = G_1 \cup \neg A_1$ and $G_2 = G_2 \cup \neg A_2$. Similarly, since composition preserves canonicity, we have $G = G \cup \neg A$.

   Consider $w \in ((S_1 \uparrow^{U_1 \cup U_2} \cap S_2 \uparrow^{U_1 \cup U_2}) \uparrow^{U \cup V})|^k$. Let $w_1 = w \downarrow_{U_1 \cup V_1}$ and $w_2 = w \downarrow_{U_2 \cup V_2}$. By (4.6:4), we have
   $w_1 \in (((S_1 \uparrow^{U_1 \cup U_2}) \uparrow^{U \cup V}))|^k \downarrow_{U_1 \cup V_1}$. By (4.6:1) and (4.6:2), this implies that $w_1 \in (S_1 \uparrow^{U_1 \cup V_1})|^k$. Similarly, we also have $w_2 \in (S_2 \uparrow^{U_2 \cup V_2})|^k$.

   Consider $t \leq k$ and $i \leq t$. By definition, if $\varphi_w^{C \uparrow^{U \cup V}}(i) = 0$, then $w_{[0,i]} \notin G \uparrow^{U \cup V}$. By (4.6:5), we deduce $[(w_{1[0,i]} \notin G_1 \uparrow^{U_1 \cup V_1}) \vee (w_{2[0,i]} \notin G_2 \uparrow^{U_2 \cup V_2})]$. As a consequence,

$$\varphi_w^{C \uparrow^{U \cup V}}(i) \geq \varphi_{w_1}^{C_1 \uparrow^{U_1 \cup V_1}}(i) + \varphi_{w_2}^{C_2 \uparrow^{U_2 \cup V_2}}(i) - 1$$

$$\Rightarrow \forall t \leq k,\ D^{(t,d)}_{C\uparrow^{U\cup V}}(w) \geq D^{(t,d)}_{C_1\uparrow^{U_1\cup V_1}}(w_1) + D^{(t,d)}_{C_2\uparrow^{U_2\cup V_2}}(w_2) - 1$$

$$\Rightarrow \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U\cup V}}(w) \geq \liminf_{t\to k} D^{(t,d)}_{C_1\uparrow^{U_1\cup V_1}}(w_1) + \liminf_{t\to k} D^{(t,d)}_{C_2\uparrow^{U_2\cup V_2}}(w_2) - 1.$$

By hypothesis, we have

$$\begin{cases} \liminf_{t\to k} D^{(t,d)}_{C_1\uparrow^{U_1\cup V_1}}(w_1) \geq m_1 \\ \liminf_{t\to k} D^{(t,d)}_{C_2\uparrow^{U_2\cup V_2}}(w_2) \geq m_2. \end{cases}$$

As a consequence,

$$\liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U\cup V}}(w) \geq m_1 + m_2 - 1.$$

Finally,

$$\forall w \in (S\uparrow^{U\cup V})|^k,\ \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U\cup V}}(w) \geq m_1 + m_2 - 1$$

$$\Rightarrow \inf_{w\in(S\uparrow^{U\cup V})|^k} \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U\cup V}}(w) \geq m_1 + m_2 - 1.$$

□

2. Let $C = (V, A, G) = C_1 \wedge C_2$. Since C_1 and C_2 are contracts in canonical form, we have $G_1 = G_1 \cup \neg A_1$ and $G_2 = G_2 \cup \neg A_2$. Similarly, since conjunction preserves canonicity, we have $G = G \cup \neg A$.

   Consider $w \in (S_1\uparrow^{U_1\cup V})|^k$. Let $w_1 = w\downarrow_{U_1\cup V_1}$ and $w_2 = w\downarrow_{U_1\cup V_2}$. By (4.6:4), we have $w_1 \in ((S_1\uparrow^{U_1\cup V}))|^k\downarrow_{U_1\cup V_1}$. By (4.6:2), this implies that $w_1 \in (S_1\uparrow^{U_1\cup V_1})|^k$. Similarly, we also have $w_2 \in (S_1\uparrow^{U_1\cup V_2})|^k$.

   Consider $t \leq k$ and $i \leq t$. By definition, if $\varphi_w^{C\uparrow^{U_1\cup V}}(i) = 0$, then $w_{[0,i]} \notin G\uparrow^{U_1\cup V}$. By (4.6:5), we deduce $[(w_{1[0,i]} \notin G_1\uparrow^{U_1\cup V_1}) \vee (w_{2[0,i]} \notin G_2\uparrow^{U_1\cup V_2})]$. As a consequence,

$$\varphi_w^{C\uparrow^{U_1\cup V}}(i) \geq \varphi_{w_1}^{C_1\uparrow^{U_1\cup V_1}}(i) + \varphi_{w_2}^{C_2\uparrow^{U_1\cup V_2}}(i) - 1$$

$$\Rightarrow \forall t \leq k,\ D^{(t,d)}_{C\uparrow^{U_1\cup V}}(w) \geq D^{(t,d)}_{C_1\uparrow^{U_1\cup V_1}}(w_1) + D^{(t,d)}_{C_2\uparrow^{U_1\cup V_2}}(w_2) - 1$$

$$\Rightarrow \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U_1\cup V}}(w) \geq \liminf_{t\to k} D^{(t,d)}_{C_1\uparrow^{U_1\cup V_1}}(w_1) + \liminf_{t\to k} D^{(t,d)}_{C_2\uparrow^{U_1\cup V_2}}(w_2) - 1.$$

By hypothesis, we have

$$\begin{cases} \liminf_{t\to k} D^{(t,d)}_{C_1\uparrow^{U_1\cup V_1}}(w_1) \geq m_1 \\ \liminf_{t\to k} D^{(t,d)}_{C_2\uparrow^{U_1\cup V_2}}(w_2) \geq m_2. \end{cases}$$

As a consequence,

$$\liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U_1\cup V}}(w) \geq m_1 + m_2 - 1.$$

Finally,

$$\begin{aligned} \forall w \in (S_1 \uparrow^{U_1\cup V})|^k,\ \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U_1\cup V}}(w) \geq & m_1 + m_2 \\ & - 1 \\ \Rightarrow \inf_{w\in(S_1\uparrow^{U_1\cup V})|^k} \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U_1\cup V}}(w) \geq & m_1 + m_2 \\ & - 1. \end{aligned}$$

□

3. Consider $w \in (S_1 \uparrow^{U_1\cup V_2})|^k$. Let $w' \in w \uparrow^{U_1\cup V_1\cup V_2}$ and $w_1 = w' \downarrow_{U_1\cup V_1}$. By (4.6:1) and (4.6:2), we have $w_1 \in (S_1 \uparrow^{U_1\cup V_1})|^k$.

   Consider now $t \leq k$ and $i \leq t$. By definition, $\varphi_{w_1}^{C_1\uparrow^{U_1\cup V_1}}(i) = 1 \iff w_{1[0,i]} \in (G_1 \cup \neg A_1) \uparrow^{U_1\cup V_1}$. By hypothesis, $((G_1 \cup \neg A_1) \uparrow^{V_1\cup V_2})|^{\leq k} \subseteq ((G_2 \cup \neg A_2) \uparrow^{V_1\cup V_2})|^{\leq k}$. Thus, by (4.6:5), $((G_1 \cup \neg A_1) \uparrow^{U_1\cup V_1\cup V_2})|^{\leq k} \subseteq ((G_2 \cup \neg A_2) \uparrow^{U_1\cup V_1\cup V_2})|^{\leq k}$.
   If $\varphi_{w_1}^{C_1\uparrow^{U_1\cup V_1}}(i) = 1$, then

$$\begin{aligned} & w_{1[0,i]} \in ((G_1 \cup \neg A_1) \uparrow^{U_1\cup V_1})|^{\leq k} \\ \Rightarrow\ & w_1[0,i] \uparrow^{U_1\cup V_1\cup V_2} \subseteq ((G_1 \cup \neg A_1) \uparrow^{U_1\cup V_1\cup V_2})|^{\leq k} \\ \Rightarrow\ & w_1[0,i] \uparrow^{U_1\cup V_1\cup V_2} \subseteq ((G_2 \cup \neg A_2) \uparrow^{U_1\cup V_1\cup V_2})|^{\leq k} \\ \Rightarrow\ & w'_{[0,i]} \in (G_2 \cup \neg A_2) \uparrow^{U_1\cup V_1\cup V_2} \\ \Rightarrow\ & w'_{[0,i]} \downarrow_{U_1\cup V_2} \in (G_2 \cup \neg A_2) \uparrow^{U_1\cup V_1\cup V_2} \downarrow_{U_1\cup V_2} \text{ by (4.6:4)} \\ \Rightarrow\ & w_{[0,i]} \in (G_2 \cup \neg A_2) \uparrow^{U_1\cup V_2} \text{ by (4.6:2)} \\ \Rightarrow\ & \varphi_w^{C_2\uparrow^{U_1\cup V_2}}(i) = 1. \end{aligned}$$

   Thus,

$$\begin{aligned} & \forall t \leq k,\ \forall i \leq t,\ \varphi_w^{C_2\uparrow^{U_1\cup V_2}}(i) \geq \varphi_{w_1}^{C_1\uparrow^{U_1\cup V_1}}(i) \\ \Rightarrow\ & \forall t \leq k,\ D^{t,d}_{C_2\uparrow^{U_1\cup V_2}}(w) \geq D^{t,d}_{C_1\uparrow^{U_1\cup V_1}}(w_1) \\ \Rightarrow\ & \liminf_{t\to k} D^{t,d}_{C_2\uparrow^{U_1\cup V_2}}(w) \geq \liminf_{t\to k} D^{t,d}_{C_1\uparrow^{U_1\cup V_1}}(w_1). \end{aligned}$$

By hypothesis,

$$\liminf_{t\to k} D^{t,d}_{C_1\uparrow^{U_1\cup V_1}}(w_1) \geq m.$$

As a consequence,

$$\begin{aligned}&\forall w \in (S_1 \uparrow^{U_1\cup V_2})|^k,\ \liminf_{t\to k} D^{t,d}_{C_2\uparrow^{U_1\cup V_2}}(w) \geq m\\ \Rightarrow &\inf_{w\in(S_1\uparrow^{U_1\cup V_2})|^k} \liminf_{t\to k} D^{t,d}_{C_2\uparrow^{U_1\cup V_2}}(w) \geq m.\end{aligned}$$

□

Theorem 4.4 also extends to the case of availability. Hence, we have the following corollary

Corollary 4.7. *Let S be a system and C_1, C_2, C_3 be three contracts in canonical form. We have the following results.*

- $S \models^{A(k)}_{d,m} C_1 \parallel (C_2 \parallel C_3)$ *iff* $S \models^{A(k)}_{d,m} (C_1 \parallel C_2) \parallel C_3$*;*
- $S \models^{A(k)}_{d,m} C_1 \wedge (C_2 \wedge C_3)$ *iff* $S \models^{A(k)}_{d,m} (C_1 \wedge C_2) \wedge C_3$*;*
- *If* $C_1 \preceq^{(\leq t)} C_2$ *and* $S \models^{A(k)}_{d,m} C_1 \parallel C_3$ *(respectively,* $S \models^{A(k)}_{d,m} C_1 \wedge C_3$*), then* $S \models^{A(k)}_{d,m} (C_2 \parallel C_3)$ *(respectively,* $S \models^{A(k)}_{d,m} (C_2 \wedge C_3)$*).*

4.3.4 Effective algorithms/representations

We propose *symbolic* and *effective* automata-based representations for contracts and systems. Those representations are needed to handle possibly infinite sets of runs with a finite memory. We will be working with variables defined over a *finite* domain D. According to our theory, a symbolic representation is effective for an assumption (resp. a guarantee) if inclusion is decidable and the representation is closed under complementation (needed for refinement), union, and intersection. A representation is effective for a system (that is not an assumption or a guarantee) if it is closed under intersection and (inverse) projection, and reliability/availability are decidable.

We assume that systems that are not assumptions or guarantees are represented with *symbolic transition systems* (see Section 4.2 for properties) and that assumptions and guarantees are represented with either finite-word or Büchi automata. Let $C = (V, A, G)$ be a contract, a *symbolic contract* for C is thus a tuple $(V, \mathcal{B}_A, \mathcal{B}_G)$, where $\mathcal{B}_A$ and $\mathcal{B}_G$ are automata with $L(\mathcal{B}_A) = A$ and $L(\mathcal{B}_G) = G$. Observe that there are systems and contracts for which there exists no symbolic representation.

Since both finite-word and Büchi automata are closed under complementation, union and intersection, it is easy to see that the composition and the conjunction of two symbolic contracts is still a symbolic contract. Moreover, since inclusion is decidable for those automata, we can always check whether refinement holds. We now focus on the satisfaction relations. We distinguish between R-Satisfiability and A-Satisfiability. We consider a symbolic contract $C = (V, \mathcal{B}_A, \mathcal{B}_G)$ and a symbolic transition system $Symb = (V, Q_s, T, Q_{s0})$.

- **Reliability**. When considering R-satisfaction, we will assume that $\mathcal{B}_A$ and $\mathcal{B}_G$ are Büchi automata. It is conceptually easy to decide whether $Symb$ R-satisfies C. Indeed, following results obtained for temporal logics [131, 132], implemented in the *SPIN* toolset [127], this amounts to check whether the Büchi automaton obtained by taking the synchronous product between $Symb$ and $\neg(\mathcal{B}_G \cup \neg\mathcal{B}_A)$ is empty. Observe that assumptions and guarantees can also be represented by logical formalisms that have a translation to Büchi automata – this includes *LTL* [108] and *ETL* [134]. The theory generalizes to other classes of infinite word automata closed under negation and union and other logical formalisms such as *CTL* [40] or *PSL* [60].

- **Availability with level m and discount factor d**. In [53], de Alfaro et al. proposed *DCTL*, a quantitative version of the CTL logic [40]. DCTL has the same syntax as CTL, but its semantics differs : in DCTL, formulas and atomic propositions take values between 0 and 1 rather than in $\{0, 1\}$. Let φ_1 and φ_2 be two DCTL formulas, the value of $\varphi_1 \wedge \varphi_2$ (resp. $\varphi_1 \vee \varphi_2$) is the minimum (resp. maximum) between the values of φ_1 and φ_2. The value of $\forall\varphi_1$ (resp. $\exists\varphi_1$) is the minimum (resp. maximum) valuation of φ_1 over all the runs. In addition to its quantitative aspect, DCTL also allows to discount on the value of the formula as well as to compute its average ($\triangle_d$ operator, where d is the discount : see the semantics with $d = 1$ and $d < 1$ page 6 of [53]) on a possibly infinite run. We assume that $\mathcal{B}_A$ and $\mathcal{B}_G$ are *complete* finite-word automata and show how to reduce A-satisfaction to the evaluation of a DCTL property. Our first step is to compute $Symb'$, the synchronous product between $Symb$ and $\mathcal{B}_G \cup \neg\mathcal{B}_A$. The resulting automaton can also be viewed as a symbolic transition system whose states are labelled with a proposition p which is true if the state is accepting and false otherwise. In fact, finite sequences of states of $Symb'$ whose last state is accepting are prefixes of runs of $Symb$ that satisfy $\mathcal{B}_G \cup \neg\mathcal{B}_A$. Hence, checking whether $Symb$ A-satisfies C boils down to compute the minimal average to see $p = 1$ in $Symb'$. Our problem thus reduces to the one of checking for each initial state of $Symb'$ whether the value of the DCTL property $\forall\triangle_d\, p$ is greater or equal to m.

4.4 Probabilistic Contracts

We now extend the results of the previous section to systems that mix stochastic and non-deterministic aspects. As for the previous section, all our results will be developed assuming that contracts and systems are represented by sets of runs and then an automata-based representation will be proposed.

Consider a system whose set of variables is U. Our way to mix stochastic and non-deterministic information consists in assuming that, at any moment of time, the value of a set of variables P are chosen with respect to a given probability distribution. The value of the variables in $U \setminus P$ are chosen in a non-deterministic manner. From the point of view of compositional reasoning, it matters whether variables in P are local to a given system or global and shared by all the systems. Indeed, without going to the details, dealing with local probabilistic variables would require to handle conditional probabilities in composition and conjunction operations. To simplify the problem, we assume that variables in P are global and shared by all the systems involved in the design. Remark that one can already model a lot with global

variables. Classically, the idea is to view some of the variables as "don't care" in the systems in where they do not matter. Without loss of generality, we also assume that for a given system, the value of the non-deterministic variables remain the same for the initial position of all the runs. This allows to select the initial value of the variables of the run by using the probability distribution only.

We will assume that systems are receptive on P. Due to this property, one can see that runs of a system on a set of variables U with $P \subseteq U$ are runs on P in where each position is augmented with an assignment for the variables in $U \setminus P$. In addition, we suppose that, in a given position, the probability to select the next values of the variables in P is independent from the non-deterministic choice. This is done by assuming the existence of a unique probability distribution $\mathbb{P}$ over $[P]^\omega$ and extending it to $[P]^*$ as follows: $\forall w \in [P]^*,\ \mathbb{P}(w) = \int_{\{w' \in P^\omega \mid w < w'\}} \mathbb{P}(w')dw'$, where $<$ is the prefix order on runs.

Remark 4.1. *Our model of computation is clearly not as powerful as Markov Decision Processes (MDPs). Indeed, in an MDP, at any given moment of time, the choice of the values of variables in $U \setminus P$ may influence the distribution on the next values of variables in P. As we assume a unique global distribution on the set of runs, the choice of the values of the variables in $U \setminus P$ does not influence the probability distribution that is fixed in advance and only depend on the probabilistic choices.*

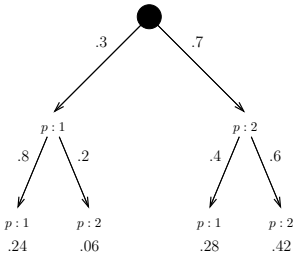
Before defining relations between systems and contracts, it is first necessary to define a probability measure on the set of runs of the system. By hypothesis, this measure has been defined on the set of runs over P and we have to lift it to runs on U. As the system is receptive on P, one could think that the measure directly extends to the runs of the system. This is actually not true. Indeed, one can associate several different values of the non-deterministic variables to a given run of the stochastic variables. This problem can be solved with the help of a scheduler that, in a given moment of time, associates a unique value to each non-deterministic variable with a given value of the probabilistic variables. In practice, systems are not defined as sets of runs but rather as symbolic objects, e.g., Markov Decision Processes, that generate runs from a set of initial states. In such context, the resolution of the non-determinism is incremental. The process starts from an initial value of the probabilistic variables to which is associated a unique value of the non-deterministic variables. Then, at any moment of time and for any run, the scheduler associates a unique non-deterministic choice to a given value of the probabilistic variables. As the system is receptive on P, a scheduler basically associates to any position of any run on P a value for the non-deterministic variables in order to retrieve a run of the system. This is sufficient to define a probability measure on subsets of runs of the system. The assignments can either depend (1) on the last position of the run, in which case the scheduler is said to be memoryless, or (2) on a prefix of the run, in which case the scheduler is said to be history-dependent.

We now propose a general definition of the "effect of a scheduler",i.e., computing a subset of runs of S receptive on P and on which a probability measure can be defined. Characterizing the effect of the scheduler is enough to reason on compositional design. This is different from the application of the scheduler itself, i.e, the choice made at a given position. Consider a system $S = (U, \Omega)$. From a definition point of view, since the system is receptive on P, the effect of a scheduler f can be characterized by a mapping from every finite (or infinite) run w on probabilistic variables P to a run $f(w)$ of S which coincides with w for every probabilistic variable. This can be formalized with the following definition.

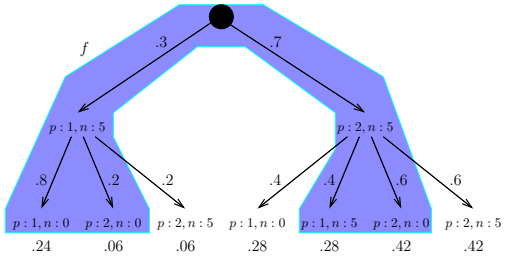
Definition 4.6 (Scheduler)**.** *A scheduler f of system $S = (U, \Omega)$, with $P \subseteq U$, is a monotonous mapping $[P]^* \to \Omega$ such that for all $w \in [P]^*$, $f(w) \downarrow_P = w$. The set of schedulers corresponding to a system S is denoted by* $\mathsf{Sched}(S)$.

For simplicity of the presentation, we use the term scheduler to refer either to the resolution of the non-determinism in a given position (which will be needed in Section 4.4.3) of the run or to the effect of applying the scheduler to generate a subset of runs of the system whose probability measure is defined. Let f be a scheduler defined on a finite set of runs of length k. To be coherent with classical definitions of schedulers that resolve non-determinism starting from the initial set of states, we have to suppose that f is causal. More precisely, given a run of length $k+1$, this means that f cannot change the non-deterministic assignments to the prefix of length k of the run. Formally, $\forall w, w' \in [P]^*, w < w' \Rightarrow f(w) < f(w')$. In practice, this is a natural assumption that is only emphasized as it will be used in the proofs.

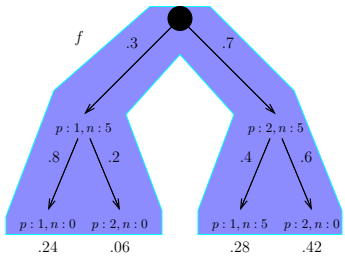
The above theory is illustrated in Figure 4.2. Figure 4.2a presents the set of runs of a probabilistic variable p that can take two values: 1 and 2. Figure 4.2b presents the set of runs of a system whose unique probabilistic variable is p. The runs colored in dark are those selected by the schedulers. One can see that the probability measure of these runs is $1 = 0.24 + 0.06 + 0.28 + 0.42$, while the measure on all runs is 1.76. The reason is that probability values are duplicated due to non-determinism. As an example, from the state $p:1, n:5$, the probability that $p = 2$ in the next step is 0.2. However, this probability is duplicated because $p:2$ can either be associated to $n:0$ or to $n:5$. The scheduler will choose between those two values. For doing so, it may use the history of the run.



(a) Set of runs for a probability variable p and its probability distribution.



(b) Set of runs with a probabilistic variable p and a non-deterministic variable n, and a scheduler f



(c) Measure on the sets of runs after applying the scheduler f

Figure 4.2: Illustration of a scheduler defining a probability measure on a set of executions

4.4.1 Probabilistic contracts

We will say that a contract $C = (V, A, G)$ is a *probabilistic contract* iff $P \subseteq V$, i.e. iff its set of variables contains all the probabilistic variables. We now turn to the problem of deciding whether a system $S = (U, \Omega)$ satisfies a probabilistic contract $\mathcal{C} = (V, A, G)$. As it was already the case for non-probabilistic contracts, we will distinguish R-Satisfaction and A-Satisfaction.

In Section 4.3, R-Satisfaction was defined with respect to a Boolean interpretation: either the system R-satisfies a contract or it does not. When moving to the probabilistic setting, we can give a *quantitative* definition for R-Satisfaction that is: *for any scheduler, is the probability to satisfy the contract greater or equal to a certain threshold?*

Definition 4.7 (P-R-Satisfaction)**.** *A system $S = (U, \Omega)$ R-satisfies a probabilistic contract $\mathcal{C} = (V, A, G)$ for runs of length k ($k \in \mathbb{N}^\infty$) with level α, denoted $S \models_\alpha^{R(k)} \mathcal{C}$, iff*

$$\inf_{f \in \mathsf{Sched}(S \uparrow^{U \cup V})} (\mathbb{P}([f([P]^k) \cap (G \cup \neg A) \uparrow^{U \cup V}] \downarrow_P) \geq \alpha.$$

Observe that, as for the non-probabilistic case, we consider that runs that do not satisfy the assumption are good runs. In addition to the motivation given in Section 4.3.1, we will see that using such an interpretation is needed when considering the conjunction operation (see the observation after Theorem 4.8).

Though A-Satisfaction was already qualitative, we now have to take into account the probabilistic point of view: instead of considering the minimal value of the mean-availability for all runs of the system, we now consider the *minimal expected value* of the mean-availability for all schedulers.

Definition 4.8 (P-A-Satisfaction)**.** *A system $S = (U, \Omega)$ A-satisfies a probabilistic contract $\mathcal{C} = (V, A, G)$ for runs of length k ($k \in \mathbb{N}^\infty$) with level α and discount factor d, denoted $S \models_{d,\alpha}^{A(k)} \mathcal{C}$, iff*

$$\inf_{f \in \mathsf{Sched}(S \uparrow^{U \cup V})} \int_{w \in [P]^k} \mathbb{P}(w) \cdot F(w) dw \geq \alpha$$

with

$$F(w) = \begin{cases} D^{k,d}_{\mathcal{C} \uparrow^{U \cup V}}(f(w)) & \text{if } k < \omega \\ \liminf_{t \to k} D^{t,d}_{\mathcal{C} \uparrow^{U \cup V}}(f(w)) & \text{if } k = \omega. \end{cases}$$

4.4.2 Operations on probabilistic contracts and Compositional reasoning

We now leverage the compositional reasoning results of Section 4.3.2 to probabilistic contracts. We consider composition/conjunction and refinement separately.

Composition and Conjunction

Composition and conjunction of probabilistic contracts is defined as for non-probabilistic contracts (see Definition 4.4). We thus propose an extension of Theorems 4.3 and 4.5 which takes the probabilistic aspects into account.

Theorem 4.8 (P-R-Satisfaction)**.** *Consider three systems $S = (U, \Omega)$, $S_1 = (U_1, \Omega_1)$ and $S_2 = (U_2, \Omega_2)$ and two probabilistic contracts $\mathcal{C}_1 = (V_1, A_1, G_1)$ and $\mathcal{C}_2 = (V_2, A_2, G_2)$ that are in canonical form. We have the following results:*

1. *Composition. Assume that S_1 and S_2 are P-compatible. If $S_1 \models_\alpha^{R(k)} \mathcal{C}_1$ and $S_2 \models_\beta^{R(k)} \mathcal{C}_2$, then $S_1 \cap S_2 \models_\gamma^{R(k)} \mathcal{C}_1 \parallel \mathcal{C}_2$ with $\gamma \geq \alpha + \beta - 1$ if $\alpha + \beta \geq 1$ and 0 otherwise.*
2. *Conjunction. Assume that S is P-receptive. If $S \models_\alpha^{R(k)} \mathcal{C}_1$ and $S \models_\beta^{R(k)} \mathcal{C}_2$, then $S \models_\gamma^{R(k)} \mathcal{C}_1 \wedge \mathcal{C}_2$ with $\gamma \geq \alpha + \beta - 1$ if $\alpha + \beta \geq 1$ and 0 otherwise.*

Remark that the choice of γ in Theorem 4.8 is tight: this bound is matched in many cases. We first state a classical algebraic property, which in fact justifies the choice for γ in the theorem, and two lemmas that will be needed in the proof of Theorem 4.8. We then present the proof.

Property 4.9. *Let E_1 and E_2 be two sets of runs over P. We have:*

$$\begin{aligned} &\mathbb{P}(\neg(E_1 \cap E_2)) \leq \mathbb{P}(\neg E_1) + \mathbb{P}(\neg E_2) \\ \Rightarrow\ &1 - \mathbb{P}(E_1 \cap E_2) \leq (1 - \mathbb{P}(E_1)) + (1 - \mathbb{P}(E_2)) \\ \Rightarrow\ &\mathbb{P}(E_1 \cap E_2) \geq \mathbb{P}(E_1) + \mathbb{P}(E_2) - 1. \end{aligned} \tag{4.9:1}$$

We now propose the two lemmas.

Lemma 4.10. *Consider $S = (U, \Omega)$ a P-receptive system, $f \in \mathsf{Sched}(S)$ a scheduler of S and U' a set of variables. If $P \subseteq U' \subseteq U$, then we have:*

$$f \downarrow_{U'} : \left\{ \begin{array}{ccc} [P]^\infty & \rightarrow & S \downarrow_{U'} \\ w & \mapsto & f(w) \downarrow_{U'} \end{array} \right\} \in \mathsf{Sched}(S \downarrow_{U'}).$$

Proof. Let $f' = f \downarrow_{U'}$. By definition, $f' : [P]^* \rightarrow S \downarrow_{U'}$. Consider now $w \in [P]^*$ and $w' < w$. Since $w' < w$, we have $f(w') < f(w)$. As a consequence, $f'(w') < f'(w)$. Moreover, $f(w) \downarrow_P = w$ and $P \subseteq U'$, thus by (4.6:3), $(f(w) \downarrow_{U'}) \downarrow_P = w$.

□

Lemma 4.11. *Consider $S = (U, \Omega)$ a P-receptive system, $f \in \mathsf{Sched}(S)$ a scheduler of S and U' and U'' two sets of variables. If $P \subseteq U' \subseteq U$, $P \subseteq U'' \subseteq U$ and $U' \cup U'' = U$, then*

$$\forall w \in (P)^\infty,\ f \downarrow_{U'} (w) \cap f \downarrow_{U''} (w) = \{f(w)\}.$$

Proof.

Let $w' = f \downarrow_{V'} (w)$ and $w'' = f \downarrow_{V''} (w)$. w, w' and w'' are such that $\forall i \in \mathbb{N}, \forall v \in V',\ f(w)(i)(v) = w'(i)(v)$ and $\forall i \in \mathbb{N}, \forall v \in V'',\ f(w)(i)(v) = w''(i)(v)$. Moreover, because w' and w'' are both projections of $f(w)$, $\forall i \in \mathbb{N}, \forall v \in V' \cap V'',\ f(w)(i)(v) = w'(i)(v) = w''(i)(v)$.

Now, consider $w_0 \in f \downarrow_{V'} (w) \cap f \downarrow_{V''} (w)$. Since $w_0 \in (f \downarrow_{V'} (w)) \uparrow^V$, we have $w_0 \downarrow_{V'} = w'$. Thus $\forall i \in \mathbb{N}, \forall v \in V',\ w_0(i)(v) = w'(i)(v) = f(w)(i)(v)$.

Similarly, since $w_0 \in (f \downarrow_{V''} (w)) \uparrow^V$, we have $\forall i \in \mathbb{N}, \forall v \in V',\ w_0(i)(v) = w''(i)(v) = f(w)(i)(v)$.
Finally, $\forall i \in \mathbb{N}, \forall v \in V = V' \cup V'',\ w''(i)(v) = f(w)(i)(v)$, thus $w'' = f(w)$.

□

We now give the proof of Theorem 4.8

Proof of Theorem 4.8.

We separately prove the two items of the theorem.

1. Let $S = (U, \Omega) = S_1 \cap S_2$ and $\mathcal{C} = (V, A, G) = \mathcal{C}_1 \parallel \mathcal{C}_2$. Since $\mathcal{C}_1$ and $\mathcal{C}_2$ are in canonical form and since composition preserves canonicity, we will consider that $G_1 = G_1 \cup \neg A_1$, $G_2 = G_2 \cup \neg A_2$ and $G = G \cup \neg A$.

   Consider $f \in \mathsf{Sched}(S \uparrow^{U \cup V})$. Since S_1 and S_2 are P-compatible, f is defined over all runs in $[P]^k$. Moreover, since $S = (S_1 \uparrow^{U_1 \cup U_2}) \cap (S_2 \uparrow^{U_1 \cup U_2})$, we have $(f \in \mathsf{Sched}((S_1 \uparrow^{U_1 \cup U_2}) \uparrow^{U \cup V})) \wedge (f \in \mathsf{Sched}((S_2 \uparrow^{U_1 \cup U_2}) \uparrow^{U \cup V}))$. By (4.6:1), we obtain

   $$(f \in \mathsf{Sched}(S_1 \uparrow^{U \cup V})) \wedge (f \in \mathsf{Sched}(S_2 \uparrow^{U \cup V})).$$

   Let $f_1 = f \downarrow_{U_1 \cup V_1}$ and $f_2 = f \downarrow_{U_2 \cup V_2}$. By Lemma 4.10, we have

   $$\begin{cases} \wedge & (f_1 \in \mathsf{Sched}((S_1 \uparrow^{U \cup V}) \downarrow_{U_1 \cup V_1})) \\ & (f_2 \in \mathsf{Sched}((S_2 \uparrow^{U \cup V}) \downarrow_{U_2 \cup V_2})) \end{cases}$$

   Thus, by (4.6:2),

   $$(f_1 \in \mathsf{Sched}(S_1 \uparrow^{U_1 \cup V_1}) \wedge (f_2 \in \mathsf{Sched}(S_2 \uparrow^{U_2 \cup V_2})).$$

   Consider now $w \in [P]^k$. If $f_1(w) \in G_1 \uparrow^{U_1 \cup V_1}$, then by (4.6:5) and (4.6:1), $f_1(w) \uparrow^{U \cup V} \subseteq G_1 \uparrow^{U \cup V}$. Similarly, if $f_2(w) \in G_2 \uparrow^{U_2 \cup V_2}$, then $f_2(w) \uparrow^{U \cup V} \subseteq G_2 \uparrow^{U \cup V}$. As a consequence, $f_1(w) \uparrow^{U \cup V} \cap f_2(w) \uparrow^{U \cup V} \subseteq (G_1 \cap G_2) \uparrow^{U \cup V}$, and, by Lemma 4.11, $f(w) \in (G_1 \cap G_2) \uparrow^{U \cup V}$. As a consequence,

   $$\begin{gathered} \overbrace{[f_1([P]^k) \cap G_1 \uparrow^{U_1 \cup V_1}] \downarrow_P}^{E_1} \cap \overbrace{[f_2([P]^k) \cap G_2 \uparrow^{U_2 \cup V_2}] \downarrow_P}^{E_2} \\ \subseteq \underbrace{[f([P]^k) \cap G \uparrow^{U \cup V}] \downarrow_P}_{E}. \end{gathered}$$

   This implies, by (4.9:1), that $\mathbb{P}(E) \geq \mathbb{P}(E_1) + \mathbb{P}(E_2) - 1$. Moreover, by hypothesis,

   $$\begin{cases} \mathbb{P}(E_1) \geq \alpha \\ \mathbb{P}(E_2) \geq \beta. \end{cases}$$

   Thus, $\mathbb{P}(E) \geq \alpha + \beta - 1$ and

   $$\begin{gathered} \forall f \in \mathsf{Sched}(S \uparrow^{U \cup V}), \\ \mathbb{P}([f([P]^k) \cap G \uparrow^{U \cup V}] \downarrow_P) \geq \alpha + \beta - 1. \end{gathered}$$

$$\Rightarrow \inf_{f \in \mathsf{Sched}(S\uparrow^{U\cup V})} \mathbb{P}([f([P]^k) \cap G \uparrow^{U\cup V}] \downarrow_P) \geq \alpha + \beta - 1.$$

2. We will use $\mathcal{C} = (V, A, G) = \mathcal{C}_1 \wedge \mathcal{C}_2$. Since $\mathcal{C}_1$ and $\mathcal{C}_2$ are in canonical form and since conjunction preserves canonicity, we will consider that $G_1 = G_1 \cup \neg A_1$, $G_2 = G_2 \cup \neg A_2$ and $G = G \cup \neg A$.

   Consider $f \in \mathsf{Sched}(S \uparrow^{U\cup V})$. Since S is P-receptive, f is defined over all runs in $[P]^k$.

   Let $f_1 = f \downarrow_{U\cup V_1}$ and $f_2 = f \downarrow_{U\cup V_2}$. By Lemma 4.10, we have

$$\begin{cases} \wedge & (f_1 \in \mathsf{Sched}((S \uparrow^{U\cup V}) \downarrow_{U\cup V_1})) \\ & (f_2 \in \mathsf{Sched}((S \uparrow^{U\cup V}) \downarrow_{U\cup V_2})) \end{cases}$$

   Thus, by (4.6:2),

$$(f_1 \in \mathsf{Sched}(S \uparrow^{U\cup V_1}) \wedge (f_2 \in \mathsf{Sched}(S \uparrow^{U_2\cup V_2})).$$

   Consider now $w \in [P]^k$. If $f_1(w) \in G_1 \uparrow^{U\cup V_1}$, then by (4.6:5) and (4.6:1), $f_1(w) \uparrow^{U\cup V} \subseteq G_1 \uparrow^{U\cup V}$. Similarly, if $f_2(w) \in G_2 \uparrow^{U\cup V_2}$, then $f_2(w) \uparrow^{U\cup V} \subseteq G_2 \uparrow^{U\cup V}$. As a consequence, $f_1(w) \uparrow^{U\cup V} \cap f_2(w) \uparrow^{U\cup V} \subseteq (G_1 \cap G_2) \uparrow^{U\cup V}$, and, by Lemma 4.11, $f(w) \in (G_1 \cap G_2) \uparrow^{U\cup V}$. As a consequence,

$$\overbrace{[f_1([P]^k) \cap G_1 \uparrow^{U\cup V_1}] \downarrow_P}^{E_1} \cap \overbrace{[f_2([P]^k) \cap G_2 \uparrow^{U\cup V_2}] \downarrow_P}^{E_2} \subseteq \underbrace{[f([P]^k) \cap G \uparrow^{U\cup V}] \downarrow_P}_{E}.$$

   This implies, by (4.9:1), that $\mathbb{P}(E) \geq \mathbb{P}(E_1) + \mathbb{P}(E_2) - 1$. Moreover, by hypothesis,

$$\begin{cases} \mathbb{P}(E_1) \geq \alpha \\ \mathbb{P}(E_2) \geq \beta. \end{cases}$$

   Thus, $\mathbb{P}(E) \geq \alpha + \beta - 1$ and

$$\forall f \in \mathsf{Sched}(S \uparrow^{U\cup V}), \quad \mathbb{P}([f([P]^k) \cap G \uparrow^{U\cup V}] \downarrow_P) \geq \alpha + \beta - 1$$

$$\Rightarrow \inf_{f \in \mathsf{Sched}(S\uparrow^{U\cup V})} \mathbb{P}([f([P]^k) \cap G \uparrow^{U\cup V}] \downarrow_P) \geq \alpha + \beta - 1.$$

□

Remark 4.2. *Consider two contracts (A_1, G_1) and (A_2, G_2) such that $A_1 \subset G_1$, $A_2 \subset G_2$ and $(A_1 \cup A_2) \cap (G_1 \cap G_2) = \emptyset$. It is easy to see that any system will reliably satisfy both contracts with probability 1. According to an interpretation where one considers that runs that do not satisfy assumptions are bad runs, the probability that a system satisfies the conjunction is always 0. With our interpretation, there are situations where this probability is strictly higher than 0: those where there are runs that do not belong to A_1 or A_2.*

Let us now consider to the case of P-A-Satisfaction. we propose the following theorem.

Theorem 4.12 (P-A-Satisfaction)**.** *Consider three systems $S = (U, \Omega)$, $S_1 = (U_1, \Omega_1)$ and $S_2 = (U_2, \Omega_2)$ and two probabilistic contracts $\mathcal{C}_1 = (V_1, A_1, G_1)$ and $\mathcal{C}_2 = (V_2, A_2, G_2)$ that are in canonical form. We have the following results:*

1. *Composition. Assume that S_1 and S_2 are P-compatible. If $S_1 \models^{A(k)}_{d,\alpha} \mathcal{C}_1$ and $S_2 \models^{A(k)}_{d,\beta} \mathcal{C}_2$, then $S_1 \cap S_2 \models^{A(k)}_{d,\gamma} \mathcal{C}_1 \parallel \mathcal{C}_2$ with $\gamma \geq \alpha + \beta - 1$ if $\alpha + \beta \geq 1$ and 0 otherwise.*

2. *Conjunction. Assume that S is P-receptive. If $S \models^{A(k)}_{d,\alpha} \mathcal{C}_1$ and $S \models^{A(k)}_{d,\beta} \mathcal{C}_2$, then $S \models^{A(k)}_{d,\gamma} \mathcal{C}_1 \wedge \mathcal{C}_2$ with $\gamma \geq \alpha + \beta - 1$ if $\alpha + \beta \geq 1$ and 0 otherwise.*

Proof.

For the sake of simplicity, we will consider that $k = \omega$. The proofs for $k < \omega$ are simpler versions of the ones presented here. We consider the two items of the theorem.

1. Let $S = (U, \Omega) = S_1 \cap S_2$ and $\mathcal{C} = (V, A, G) = \mathcal{C}_1 \parallel \mathcal{C}_2$.Since $\mathcal{C}_1$ and $\mathcal{C}_2$ are in canonical form and since composition preserves canonicity, we will consider that $G_1 = G_1 \cup \neg A_1$, $G_2 = G_2 \cup \neg A_2$ and $G = G \cup \neg A$.

   Consider $f \in \mathsf{Sched}(S \uparrow^{U \cup V})$. Since S_1 and S_2 are P-compatible, f is defined over all runs in $[P]^k$. Moreover, since $S = (S_1 \uparrow^{U_1 \cup U_2}) \cap (S_2 \uparrow^{U_1 \cup U_2})$, it is clear that $(f \in \mathsf{Sched}((S_1 \uparrow^{U_1 \cup U_2}) \uparrow^{U \cup V})) \wedge (f \in \mathsf{Sched}((S_2 \uparrow^{U_1 \cup U_2}) \uparrow^{U \cup V}))$. Thus, by (4.6:1),

$$\Rightarrow (f \in \mathsf{Sched}(S_1 \uparrow^{U \cup V})) \wedge (f \in \mathsf{Sched}(S_2 \uparrow^{U \cup V})).$$

   Let $f_1 = f \downarrow_{U_1 \cup V_1}$ and $f_2 = f \downarrow_{U_2 \cup V_2}$. By Lemma 4.10, we have

$$\Rightarrow \begin{cases} \wedge & (f_1 \in \mathsf{Sched}((S_1 \uparrow^{U \cup V}) \downarrow_{U_1 \cup V_1})) \\ & (f_2 \in \mathsf{Sched}((S_2 \uparrow^{U \cup V}) \downarrow_{U_2 \cup V_2})) \end{cases}$$

   Thus, by (4.6:2),

$$(f_1 \in \mathsf{Sched}(S_1 \uparrow^{U_1 \cup V_1}) \wedge (f_2 \in \mathsf{Sched}(S_2 \uparrow^{U_2 \cup V_2})).$$

   Consider $w \in [P]^k$, $t \leq k$ and $i \leq t$. If $\varphi^{\mathcal{C} \uparrow^{U \cup V}}_{f(w)}(i) = 0$, then $f(w)_{[0,i]} \notin G \uparrow^{U \cup V}$. By (4.6:5) and (4.6:2), we deduce that $[(f_1(w)_{[0,i]} \notin G_1 \uparrow^{U_1 \cup V_1}) \vee (f_2(w)_{[0,i]} \notin G_2 \uparrow^{U_2 \cup V_2})]$. As a consequence,

$$\varphi_{f(w)}^{C\uparrow^{U\cup V}}(i) \geq \varphi_{f_1(w)}^{\mathcal{C}_1\uparrow^{U_1\cup V_1}}(i) + \varphi_{f_2(w)}^{\mathcal{C}_2\uparrow^{U_2\cup V_2}}(i) - 1$$

$$\begin{aligned}\Rightarrow \forall t \leq k,\ D_{C\uparrow^{U\cup V}}^{(t,d)}(f(w)) \geq & D_{\mathcal{C}_1\uparrow^{U_1\cup V_1}}^{(t,d)}(f_1(w)) \\ & + D_{\mathcal{C}_2\uparrow^{U_2\cup V_2}}^{(t,d)}(f_2(w)) \\ & - 1\end{aligned}$$

$$\begin{aligned}\Rightarrow \liminf_{t\to k} D_{C\uparrow^{U\cup V}}^{(t,d)}(f(w)) \geq & \liminf_{t\to k} D_{\mathcal{C}_1\uparrow^{U_1\cup V_1}}^{(t,d)}(f_1(w)) \\ & + \liminf_{t\to k} D_{\mathcal{C}_2\uparrow^{U_2\cup V_2}}^{(t,d)}(f_2(w)) \\ & - 1.\end{aligned}$$

As a consequence, $\forall w \in [P]^k$,

$$\begin{aligned}\liminf_{t\to k} D_{C\uparrow^{U\cup V}}^{(t,d)}(f(w)) \geq & \liminf_{t\to k} D_{\mathcal{C}_1\uparrow^{U_1\cup V_1}}^{(t,d)}(f_1(w)) \\ & + \liminf_{t\to k} D_{\mathcal{C}_2\uparrow^{U_2\cup V_2}}^{(t,d)}(f_2(w)) \\ & - 1\end{aligned}$$

$$\begin{aligned}\Rightarrow \int_{w\in[P]^k} & \mathbb{P}(w)\cdot \liminf_{t\to k} D_{C\uparrow^{U\cup V}}^{(t,d)}(f(w))dw \geq \\ & \int_{w\in[P]^k} \mathbb{P}(w)\cdot \liminf_{t\to k} D_{\mathcal{C}_1\uparrow^{U_1\cup V_1}}^{(t,d)}(f_1(w))dw \\ & + \int_{w\in[P]^k} \mathbb{P}(w)\cdot \liminf_{t\to k} D_{\mathcal{C}_2\uparrow^{U_2\cup V_2}}^{(t,d)}(f_2(w))dw \\ & - 1.\end{aligned}$$

By hypothesis, we have

$$\begin{cases} \displaystyle\int_{w\in[P]^k} \mathbb{P}(w)\cdot \liminf_{t\to k} D_{\mathcal{C}_1\uparrow^{U_1\cup V_1}}^{(t,d)}(f_1(w))dw \geq \alpha \\ \displaystyle\int_{w\in[P]^k} \mathbb{P}(w)\cdot \liminf_{t\to k} D_{\mathcal{C}_2\uparrow^{U_2\cup V_2}}^{(t,d)}(f_2(w))dw \geq \beta. \end{cases}$$

Thus, $\forall f \in \mathsf{Sched}(S\uparrow^{U\cup V})$,

$$\int_{w\in[P]^k} \mathbb{P}(w)\cdot \liminf_{t\to k} D_{C\uparrow^{U\cup V}}^{(t,d)}(f(w))dw \geq \alpha + \beta - 1$$

2. Let $\mathcal{C} = (V, A, G) = \mathcal{C}_1 \wedge \mathcal{C}_2$.Since $\mathcal{C}_1$ and $\mathcal{C}_2$ are in canonical form and since conjunction preserves canonicity, we will consider that $G_1 = G_1 \cup \neg A_1$, $G_2 = G_2 \cup \neg A_2$ and $G = G \cup \neg A$.

Consider $f \in \mathsf{Sched}(S \uparrow^{U\cup V})$. Since S is P-receptive, f is defined over all runs in $[P]^k$. Let $f_1 = f \downarrow_{U\cup V_1}$ and $f_2 = f \downarrow_{U\cup V_2}$. By Lemma 4.10, we have

$$\Rightarrow \begin{cases} \wedge & (f_1 \in \mathsf{Sched}((S \uparrow^{U\cup V}) \downarrow_{U\cup V_1})) \\ & (f_2 \in \mathsf{Sched}((S \uparrow^{U\cup V}) \downarrow_{U\cup V_2})) \end{cases}$$

Thus, by (4.6:2)

$$(f_1 \in \mathsf{Sched}(S \uparrow^{U\cup V_1}) \wedge (f_2 \in \mathsf{Sched}(S \uparrow^{U\cup V_2})).$$

Consider $w \in [P]^k$, $t \leq k$ and $i \leq t$. If $\varphi^{\mathcal{C}\uparrow^{U\cup V}}_{f(w)}(i) = 0$, then $f(w)_{[0,i]} \notin G \uparrow^{U\cup V}$. By (4.6:5) and (4.6:2), we deduce that $[(f_1(w)_{[0,i]} \notin G_1 \uparrow^{U\cup V_1}) \vee (f_2(w)_{[0,i]} \notin G_2 \uparrow^{U\cup V_2})]$. As a consequence,

$$\varphi^{C\uparrow^{U\cup V}}_{f(w)}(i) \geq \varphi^{\mathcal{C}_1\uparrow^{U\cup V_1}}_{f_1(w)}(i) + \varphi^{\mathcal{C}_2\uparrow^{U\cup V_2}}_{f_2(w)}(i) - 1$$

$$\begin{aligned} \Rightarrow \forall t \leq k,\ D^{(t,d)}_{C\uparrow^{U\cup V}}(f(w)) \geq & D^{(t,d)}_{\mathcal{C}_1\uparrow^{U\cup V_1}}(f_1(w)) \\ & + D^{(t,d)}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f_2(w)) \\ & - 1 \end{aligned}$$

$$\begin{aligned} \Rightarrow \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U\cup V}}(f(w)) \geq & \liminf_{t\to k} D^{(t,d)}_{\mathcal{C}_1\uparrow^{U\cup V_1}}(f_1(w)) \\ & + \liminf_{t\to k} D^{(t,d)}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f_2(w)) \\ & - 1. \end{aligned}$$

As a consequence, $\forall w \in [P]^k$,

$$\begin{aligned} \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U\cup V}}(f(w)) \geq & \liminf_{t\to k} D^{(t,d)}_{\mathcal{C}_1\uparrow^{U\cup V_1}}(f_1(w)) \\ & + \liminf_{t\to k} D^{(t,d)}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f_2(w)) \\ & - 1 \end{aligned}$$

$$\begin{aligned} \Rightarrow \int_{w\in[P]^k} & \mathbb{P}(w) \cdot \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U\cup V}}(f(w))dw \geq \\ & \int_{w\in[P]^k} \mathbb{P}(w) \cdot \liminf_{t\to k} D^{(t,d)}_{\mathcal{C}_1\uparrow^{U\cup V_1}}(f_1(w))dw \\ & + \int_{w\in[P]^k} \mathbb{P}(w) \cdot \liminf_{t\to k} D^{(t,d)}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f_2(w))dw \\ & - 1. \end{aligned}$$

By hypothesis, we have

$$\begin{cases} \displaystyle\int_{w\in[P]^k} \mathbb{P}(w)\cdot \liminf_{t\to k} D^{(t,d)}_{\mathcal{C}_1\uparrow^{U\cup V_1}}(f_1(w))dw \geq \alpha \\ \displaystyle\int_{w\in[P]^k} \mathbb{P}(w)\cdot \liminf_{t\to k} D^{(t,d)}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f_2(w))dw \geq \beta. \end{cases}$$

Thus, $\forall f \in \mathsf{Sched}(S\uparrow^{U\cup V})$,

$$\int_{w\in[P]^k} \mathbb{P}(w)\cdot \liminf_{t\to k} D^{(t,d)}_{C\uparrow^{U\cup V}}(f(w))dw \geq \alpha+\beta-1$$

□

We now discuss the incremental design property. In fact, as Property 4.1 is independent from the systems and because of Theorems 4.8 and 4.12, we directly obtain extensions to the availability case for the two first items of Theorems 4.4 and 4.7. More precisely, we have the following results.

Theorem 4.13. *Consider three probabilistic contracts $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ and a system S. Assume that $S \models^{R(k)}_{\alpha_1} \mathcal{C}_1$, $S \models^{R(k)}_{\alpha_2} \mathcal{C}_2$, $S \models^{R(k)}_{\alpha_3} \mathcal{C}_3$. Let $\gamma = \alpha_1+\alpha_2+\alpha_3-2$ if $\alpha_1+\alpha_2+\alpha_3 > 2$ and 0 otherwise. We have*

- $S \models^{R(k)}_{\gamma} \mathcal{C}_1 \parallel (\mathcal{C}_2 \parallel \mathcal{C}_3)$ *iff* $S \models^{R(k)}_{\gamma} (\mathcal{C}_1 \parallel \mathcal{C}_2) \parallel \mathcal{C}_3$.
- $S \models^{R(k)}_{\gamma} \mathcal{C}_1 \wedge (\mathcal{C}_2 \wedge \mathcal{C}_3)$ *iff* $S \models^{R(k)}_{\gamma} (\mathcal{C}_1 \wedge \mathcal{C}_2) \wedge \mathcal{C}_3$.

Theorem 4.14. *Consider three probabilistic contracts $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ and a system S. Assume that $S \models^{A(k)}_{d,\alpha_1} \mathcal{C}_1$, $S \models^{A(k)}_{d,\alpha_2} \mathcal{C}_2$, $S \models^{A(k)}_{d,\alpha_3} \mathcal{C}_3$. Let $\gamma = \alpha_1+\alpha_2+\alpha_3-2$ if $\alpha_1+\alpha_2+\alpha_3 > 2$ and 0 otherwise. We have*

- $S \models^{A(k)}_{d,\gamma} \mathcal{C}_1 \parallel (\mathcal{C}_2 \parallel \mathcal{C}_3)$ *iff* $S \models^{A(k)}_{d,\gamma} (\mathcal{C}_1 \parallel \mathcal{C}_2) \parallel \mathcal{C}_3$.
- $S \models^{A(k)}_{d,\gamma} \mathcal{C}_1 \wedge (\mathcal{C}_2 \wedge \mathcal{C}_3)$ *iff* $S \models^{A(k)}_{d,\gamma} (\mathcal{C}_1 \wedge \mathcal{C}_2) \wedge \mathcal{C}_3$.

Refinement

We consider refinement for probabilistic contracts. Contrary to the case of non-probabilistic contracts, we will distinguish between R-Satisfaction and A-Satisfaction.

Following our move from R-Satisfaction to P-R-Satisfaction, we propose the notion of *P-Refinement* that is the quantitative version of the refinement we proposed in Section 4.3. We have the following definition.

Definition 4.9 (P-Refinement)**.** *A probabilistic contract $\mathcal{C}_1 = (V_1, A_1, G_1)$ P-Refines a probabilistic contract $\mathcal{C}_2 = (V_2, A_2, G_2)$ for runs of length k ($k \in \mathbb{N}^\infty$) with level α, denoted $\mathcal{C}_1 \preceq^{R(k)}_{\alpha} \mathcal{C}_2$, iff*

$$\forall f \in \mathsf{Sched}((G_1 \cup \neg A_1)\uparrow^{V_1\cup V_2}),$$
$$\mathbb{P}([f([P]^k) \cap (G_2 \cup \neg A_2)\uparrow^{V_1\cup V_2}]\downarrow_P) \geq \alpha.$$

Consider $C_2 \parallel C_3$ (respectively, $C_2 \wedge C_3$). If $C_1 \preceq_\alpha^{R(k)} C_2$, then $(C_1 \parallel C_3) \preceq_\alpha^{R(k)} (C_2 \parallel C_3)$ (respectively, $(C_1 \wedge C_3) \preceq_\alpha^{R(k)} (C_2 \wedge C_3)$). Observe that P-Refinement is not a preorder relation. As a consequence, conjunction is not a greatest lower bound with respect to P-Refinement. Quantitative refinement is compatible with the definition of P-R-Satisfaction, which brings the following result.

Theorem 4.15. *Consider a P-receptive system $S = (U, \Omega)$ and two probabilistic contracts $\mathcal{C}_i = (V_i, A_i, G_i)$ for $i = 1, 2$. If $(G_1 \cup \neg A_1)$ is P-receptive and prefix-closed, then*

$$S \parallel\!\models_\alpha^{R(k)} \mathcal{C}_1 \wedge \mathcal{C}_1 \preceq_\beta^{R(k)} \mathcal{C}_2 \Rightarrow S \parallel\!\models_{\alpha+\beta-1}^{R(k)} \mathcal{C}_2.$$

Before giving the proof of the theorem, we propose the following Lemma, which proves the existence of corresponding schedulers in two P-receptive systems.

Lemma 4.16. *Consider $S = (U, \Omega)$ and $S' = (U, \Omega')$ two systems over the same set of variables U. If S and S' are P-receptive and if S' is prefix-closed, then for all $f \in \mathsf{Sched}(S)$, there exists $f' \in \mathsf{Sched}(S')$ such that*

$$\forall w \in [P]^*,\ f(w) \in S' \Rightarrow f'(w) = f(w).$$

Proof.

Consider $f \in \mathsf{Sched}(S)$ and let $f' : [P]^* \to S'$ such that :

$$\begin{cases} f'(\varepsilon) = \varepsilon \\ f'(w.\sigma) = f(w.\sigma) \text{ if } f(w.\sigma) \in S' \\ f'(w.\sigma) = f'(w).\sigma' \text{ s.t. } f'(w).\sigma' \in S' \text{ and } \sigma' \downarrow_P = \sigma. \end{cases}$$

First of all, since S' is prefix-closed, if $f(w) \in S'$, then for all $w' < w$, $f(w') \in S'$, and as a consequence $f'(w') = f(w')$. Moreover, since S' is P-receptive, if $f'(w) \in S'$, then for all $\sigma \in P \to D$, there exists $\sigma' \in U \to D$ such that $\sigma' \downarrow_P = \sigma$ and $f'(w).\sigma' \in S'$. This ensures that the definition of f' is coherent.
We will now prove by induction that $f' \in \mathsf{Sched}(S')$.

- $f'(\varepsilon) = \varepsilon$ satisfies the prefix property.
- Let $w \in [P]^k$ and $w' < w$. Suppose that $f'(w') < f'(w)$. Let $\sigma \in P \to D$.
  - If $f(w.\sigma) \in S'$, then $f'(w.\sigma) = f(w.\sigma)$ and $\forall w'' < w,\ f'(w'') = f(w'')$. Since f is a scheduler, we have $f(w') < f(w.\sigma)$.
  - Else, $f'(w.\sigma) = f'(w).\sigma'$ and as a consequence, $f'(w') < f'(w) < f'(w).\sigma'$.

□

We now give the proof for Theorem 4.15

Proof of Theorem 4.15.

Consider $f \in \mathsf{Sched}(S \uparrow^{U \cup V_2})$. By Lemma 4.10, there exists $f' \in \mathsf{Sched}(S \uparrow^{U \cup V_1 \cup V_2})$ such that $f' \downarrow_{U \cup V_2} = f$. Let $f_1 = f' \downarrow_{U \cup V_1}$. By Lemma 4.10, we have $f_1 \in \mathsf{Sched}(S \uparrow^{U \cup V_1})$. Lemma 4.16 states that there exists $f_2' \in \mathsf{Sched}((G_1 \cup \neg A_1) \uparrow^{U \cup V_1 \cup V_2})$ such that $\forall w \in [P]^*,\ f'(w) \in (G_1 \cup \neg A_1) \uparrow^{U \cup V_1 \cup V_2} \Rightarrow f_2'(w) = f'(w)$. Let $f_2 = f_2' \downarrow_{V_1 \cup V_2}$. By Lemma 4.10, we have $f_2 \in \mathsf{Sched}((G_1 \cup \neg A_1) \uparrow^{V_1 \cup V_2}$.

Consider $w \in [P]^k$. If $f_1(w) \in (G_1 \cup \neg A_1) \uparrow^{U \cup V_1}$, then by (4.6:5), $f'(w) \in (G_1 \cup \neg A_1) \uparrow^{U \cup V_1 \cup V_2}$ $\Rightarrow f'_2(w) = f'(w)$. Moreover, if $f_2(w) \in (G_2 \cup \neg A_2) \uparrow V_1 \cup V_2$, then by (4.6:5), $f'_2(w) \in (G_2 \cup \neg A_2) \uparrow^{U \cup V_1 \cup V_2}$. Thus,

$$\begin{aligned} & f'(w) \in (G_2 \cup \neg A_2) \uparrow^{U \cup V_1 \cup V_2} \\ & \Rightarrow f(w) \in (G_2 \cup \neg A_2) \uparrow^{U \cup V_2} \quad \text{by (4.6:4)}. \end{aligned}$$

As a consequence, let

$$\begin{aligned} E_1 =& [f_1([P]^k) \cap (G_1 \cup \neg A_1) \uparrow^{U \cup V_1}] \downarrow_P \\ E_2 =& [f_2([P]^k) \cap (G_2 \cup \neg A_2) \uparrow^{V_1 \cup V_2}] \downarrow_P \\ E =& [f([P]^k) \cap (G_2 \cup \neg A_2) \uparrow^{U \cup V_2}] \downarrow_P \end{aligned}$$

We have $E_1 \cap E_2 \subseteq E$.
This implies, by (4.9:1), that $\mathbb{P}(E) \geq \mathbb{P}(E_1) + \mathbb{P}(E_2) - 1$. Moreover, by hypothesis,

$$\begin{cases} \mathbb{P}(E_1) \geq \alpha \\ \mathbb{P}(E_2) \geq \beta. \end{cases}$$

Thus, $\mathbb{P}(E) \geq \alpha + \beta - 1$ and $\forall f \in \mathsf{Sched}(S \uparrow^{U \cup V_2})$,

$$\mathbb{P}([f([P]^k) \cap (G_2 \cup \neg A_2) \uparrow^{U \cup V_2}] \downarrow_P) \geq \alpha + \beta - 1$$

□

P-A-satisfaction and quantitative refinement are orthogonal measures. Indeed, P-A-satisfaction measures the infimal expected availability of a system for all schedulers, while quantitative refinement measures the infimal set of traces of a probabilistic contract that corresponds to another probabilistic contract. In such context, the minimal schedulers for the two notions may differ. We propose the following result, which links P-A-Satisfaction with the definition of refinement proposed for non-probabilistic contracts.

Theorem 4.17. *Consider a P-receptive system $S = (U, \Omega)$ and two probabilistic contracts $\mathcal{C}_i = (V_i, A_i, G_i)$ for $i = 1, 2$. If $S \models^{A(k)}_{d,\alpha} \mathcal{C}_1$ and $\mathcal{C}_1 \preceq^{(\leq k)} \mathcal{C}_2$, then $S \models^{A(k)}_{d,\alpha} \mathcal{C}_2$.*

Proof.

For the sake of simplicity, we will consider that $k = \omega$. The proof for $k < \omega$ is a simpler version of the one presented here.

Consider $f \in \mathsf{Sched}(S \uparrow^{U \cup V_2})$. By Lemma 4.10, there exists $f' \in \mathsf{Sched}(S \uparrow^{U \cup V_1 \cup V_2})$ such that $f' \downarrow_{U \cup V_2} = f$. Let $f_1 = f' \downarrow_{U \cup V_1}$. By Lemma 4.10, we also have $f_1 \in \mathsf{Sched}(S \uparrow^{U \cup V_1})$. Consider now $w \in [P]^k$, $t \leq k$ and $i \leq t$. By definition, $\varphi^{\mathcal{C}_1 \uparrow^{U \cup V_1}}_{f_1(w)}(i) = 1 \iff f_1(w)_{[0,i]} \in (G_1 \cup \neg A_1) \uparrow^{U \cup V_1}$. By hypothesis,

$$((G_1 \cup \neg A_1) \uparrow^{V_1 \cup V_2})|^{\leq k} \subseteq ((G_2 \cup \neg A_2) \uparrow^{V_1 \cup V_2})|^{\leq k}.$$

Thus, by (4.6:5),

$$((G_1 \cup \neg A_1) \uparrow^{U \cup V_1 \cup V_2})|^{\leq k} \subseteq ((G_2 \cup \neg A_2) \uparrow^{U \cup V_1 \cup V_2})|^{\leq k}.$$

If $\varphi_{f_1(w)}^{\mathcal{C}_1\uparrow^{U\cup V_1}}(i) = 1$, then

$$\begin{aligned}
&f_1(w)_{[0,i]} \in ((G_1 \cup \neg A_1) \uparrow^{U\cup V_1})|^{\leq k} \\
&\quad \Rightarrow f_1(w)w[0,i] \uparrow^{U\cup V_1\cup V_2} \subseteq ((G_1 \cup \neg A_1) \uparrow^{U\cup V_1\cup V_2})|^{\leq k} \\
&\Rightarrow f_1(w)w[0,i] \uparrow^{U\cup V_1\cup V_2} \subseteq ((G_2 \cup \neg A_2) \uparrow^{U\cup V_1\cup V_2})|^{\leq k} \\
&\Rightarrow f'(w)_{[0,i]} \in (G_2 \cup \neg A_2) \uparrow^{U\cup V_1\cup V_2} \\
&\Rightarrow f'(w)_{[0,i]} \downarrow_{U\cup V_2} \in (G_2 \cup \neg A_2) \uparrow^{U\cup V_1\cup V_2} \downarrow_{U\cup V_2} \text{ by (4.6:4)} \\
&\Rightarrow f(w)_{[0,i]} \in (G_2 \cup \neg A_2) \uparrow^{U\cup V_2} \qquad \text{by (4.6:2)} \\
&\Rightarrow \varphi_{f(w)}^{\mathcal{C}_2\uparrow^{U\cup V_2}}(i) = 1.
\end{aligned}$$

Thus,

$$\begin{aligned}
&\forall t \leq k,\ \forall i \leq t,\ \varphi_{f(w)}^{\mathcal{C}_2\uparrow^{U\cup V_2}}(i) \geq \varphi_{f_1(w)}^{\mathcal{C}_1\uparrow^{U\cup V_1}}(i) \\
&\Rightarrow \forall t \leq k,\ D^{t,d}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f(w)) \geq D^{t,d}_{\mathcal{C}_1\uparrow^{U\cup V_1}}(f_1(w)) \\
&\Rightarrow \liminf_{t\to k} D^{t,d}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f(w)) \geq \liminf_{t\to k} D^{t,d}_{\mathcal{C}_1\uparrow^{U\cup V_1}}(f_1(w)).
\end{aligned}$$

By hypothesis,

$$\liminf_{t\to k} D^{t,d}_{\mathcal{C}_1\uparrow^{U\cup V_1}}(f_1(w)) \geq \alpha.$$

As a consequence,

$$\begin{aligned}
&\forall w \in [P]^k,\ \liminf_{t\to k} D^{t,d}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f(w)) \geq m \\
&\Rightarrow \int_{w\in[P]^k} \mathbb{P}(w) \cdot \liminf_{t\to k} D^{t,d}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f(w))dw \geq m.
\end{aligned}$$

Finally, $\forall f \in \mathsf{Sched}(S \uparrow^{U\cup V_2})$,

$$\int_{w\in[P]^k} \mathbb{P}(w) \cdot \liminf_{t\to k} D^{t,d}_{\mathcal{C}_2\uparrow^{U\cup V_2}}(f(w))dw \geq m$$

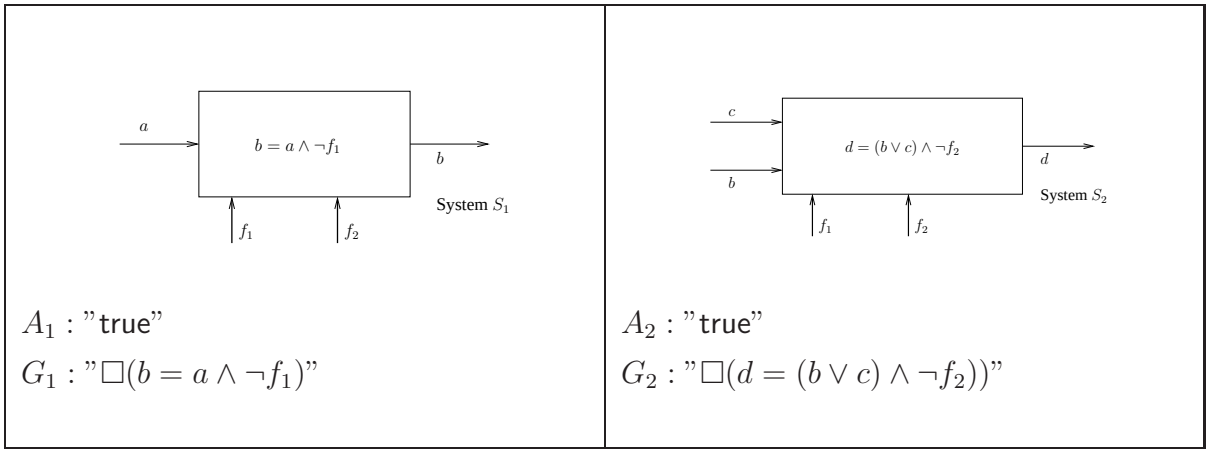
□

We now briefly discuss independent implementability in the probabilistic case. For P-R-Satisfaction, the property is defined with respect to P-Refinement. For P-A-satisfaction we use the notion of refinement introduced for non-probabilistic contracts. We have the following theorem, whose proof is a direct consequence of Theorems 4.8, 4.12, 4.15 and 4.17.

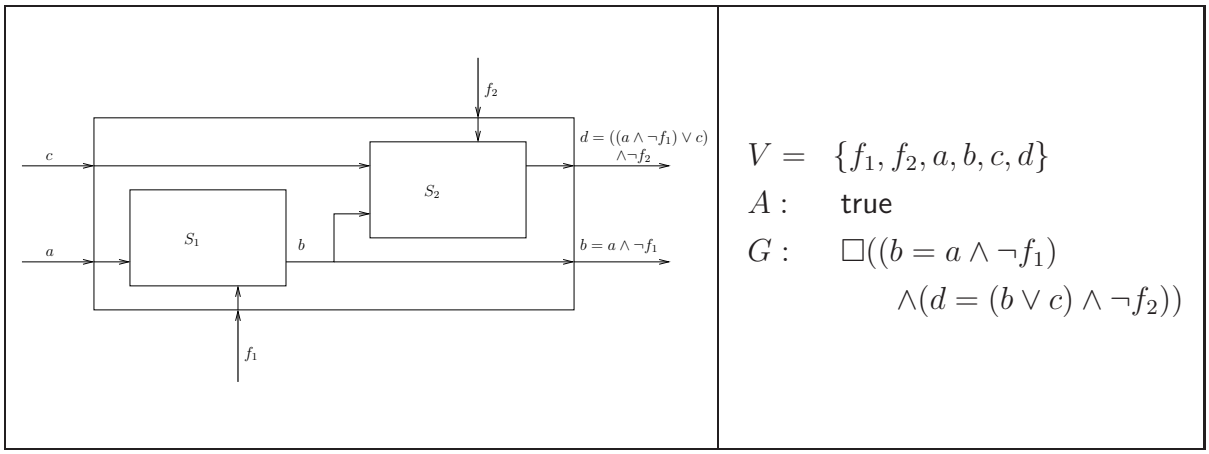
Theorem 4.18. *Let S be a P-receptive system and $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ be three probabilistic contracts such that $\mathcal{C}_1$ and $\mathcal{C}_3$ are P-compatible, and $\mathcal{C}_2$ and $\mathcal{C}_3$ are also P-compatible. We have the following results.*

- *Assume that $(G_1 \cup \neg A_1)$ is prefix-closed and P-receptive. If $\mathcal{C}_1 \preceq_\alpha^{R(k)} \mathcal{C}_2$ and $S \parallel\!\models_\beta^{R(k)} (\mathcal{C}_1 \parallel \mathcal{C}_3)$ (respectively, $S \parallel\!\models_\beta^{R(k)} (\mathcal{C}_1 \wedge \mathcal{C}_3)$), then $S \parallel\!\models_\gamma^{R(k)} (\mathcal{C}_2 \parallel \mathcal{C}_3)$ (respectively, $S \parallel\!\models_\gamma^{R(k)} (\mathcal{C}_2 \wedge \mathcal{C}_3)$), with $\gamma \geq \alpha + \beta - 1$ if $\alpha + \beta \geq 1$ and 0 else.*

- *If $\mathcal{C}_1 \preceq^{(\leq k)} \mathcal{C}_2$ and $S \parallel\!\models_{d,\alpha}^{A(k)} (\mathcal{C}_1 \parallel \mathcal{C}_3)$ (respectively, $S \parallel\!\models_{d,\alpha}^{A(k)} (\mathcal{C}_1 \wedge \mathcal{C}_3)$), then $S \parallel\!\models_{d,\alpha}^{A(k)} (\mathcal{C}_2 \parallel \mathcal{C}_3)$ (respectively, $S \parallel\!\models_{d,\alpha}^{A(k)} (\mathcal{C}_2 \wedge \mathcal{C}_3)$).*



(a) Systems S_1 and S_2 and probabilistic contracts $\mathcal{C}_1$ and $\mathcal{C}_2$.



(b) Systems $S_1 \cap S_2$ and probabilistic contract $\mathcal{C}_1 \parallel \mathcal{C}_2$.

Figure 4.3: Reliability : Example

An illustration

The objective of this chapter is to introduce the theoretical foundations for contracts and their stochastic extensions. Deliverable 5.1.1 of the SPEEDS project (available at [126]) shows the interest of industrials for our methodology and discusses other examples for the case of non-stochastic contracts. Also, the work in [66], which can be subsumed by our contribution, has been applied to an interesting case study. We now present a simple example that illustrates the approach.

Consider the systems and contracts given in Figure 4.3. Assume that $\forall i \in \mathbb{N}, \mathbb{P}(f_1(i) = 1) = 10^{-3}$ and $\mathbb{P}(f_2(i) = 1) = 2.10^{-3}$. It is easy to show that $S_1 \models^{R(50)}_{(1-10^{-3})^{50}} \mathcal{C}_1$ and $S_2 \models^{R(50)}_{(1-2.10^{-3})^{50}} \mathcal{C}_2$. It is however more difficult to deduce the probability for which $S_1 \cap S_2$ satisfies the contract $\mathcal{C}_1 \parallel \mathcal{C}_2$. Thanks to Theorem 4.8, we know that this probability is at least $(0.999)^{50} + (0.998)^{50} - 1 = 0.86$. Considering $\mathcal{C}_3 = (\{f_1, f_2, a, c, d\}, \text{"true"}, \text{"}\Box(d = ((a \wedge \neg f_1) \vee c) \wedge \neg f_2)\text{"})$, it is clear that $\mathcal{C}_1 \parallel \mathcal{C}_2 \preceq^{R(50)}_1 \mathcal{C}_3$, which implies that $S_1 \cap S_2 \models^{R(50)}_{0.86} \mathcal{C}_3$.

4.4.3 Effective algorithms/representations

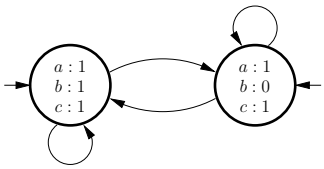
The constructions are similar to those given in Section 4.3.4. We assume the reader to be familiar with the concepts of (discrete) Markov Chains and turn-based Markov Decision Processes (else, see [24, 117, 24, 46] for an introduction and references). Roughly speaking, a Markov Chain is a symbolic transition system whose states are labeled with valuations for variables in P and whose transitions are labeled by probabilities. The labelling by probabilities follows a probability distribution, i.e., for a given state, the sum of the probability values for all outgoing transitions must be less or equal to one. In a given state, one picks up the next valuation for the probability variables, i.e., the next state. The probability to pick up a valuation is the value given on the transition that links the current state to the next chosen one. There is a special state called "$init$" from where one has to chose the first value. The concept of representing P with a Markov Chain is illustrated in Figure 4.5a, where $P = \{b\}$ and $D = \{0, 1\}$. In this example, the probability that a run starts with $b = 0$ is $1/2$. The probability that a run starts with the prefix $(b = 0)(b = 1)(b = 0)$ is given by $(1/2) \times (1/4) \times (1/3) = 1/24$.

Let $C = (V, \mathcal{B}_A, \mathcal{B}_G)$ be a symbolic contract and $Symb = (V, Q_s, T, Q_{s0})$ be a symbolic transition system. We consider a set $P \subseteq V$ of probabilistic variables. We assume that the distribution over P is symbolically represented with a Markov Chain. At each state, we have a probability distribution over the possible set of valuations for the variables. The Markov chain is finitely-branching as D is finite. Observe that each state of $Symb$ can be split into two states, one for the valuations of the non-probabilistic variables followed by one for the valuations of the probabilistic variables. The result is a new symbolic system $Symb''$ where one first evaluates $V \setminus P$ and then P.

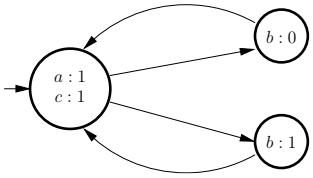
Example. *The split is illustrated in Figure 4.4. Consider the state $X = \{a = 1, b = 0, c = 1\}$ in the system given in Figure 4.4a. This state can be split into two states, $A = \{a = 1, c = 1\}$ and $E = \{b = 0\}$. The state $Y = \{a = 1, b = 1, c = 1\}$ can be split into $B = \{a = 1, c = 1\}$ and $F = \{b = 1\}$. In the split, there will be transitions from A to E and from B to F. Any transition from X (resp. Y) to Y (resp. X) will now be from E (resp. F) to B (resp. A). Since A and B have the same label and successors, they can be merged, which gives the split in Figure 4.4b.*

It is easy to see that we can use the Markov Chain that represents the probability distribution in order to "transform" the transitions from a non-deterministic variable state of $Symb''$ into a probability distribution over the probabilistic variable states simply by synchronizing the two systems. By doing so, $Symb''$ becomes a *turn-based Markov Decision Process* (MDP). Recall that a turn-based MDP mixes both non-determinism and probabilities. In our setting, non-determinism thus comes from the choice of the values for the non-probabilistic variables, while probabilities arise when evaluating variables in P. The transitions from states that are labeled with probabilistic variables are thus non-deterministic (since one has to pick up the next values for the non-probabilistic variables). Transitions from states that are labeled with non-probabilistic variables form a probability distribution on the possible values of the probabilistic variables. In this context, a run for the MDP is simply an alternance of valuations of the non-probabilistic and the probabilistic variables.

Example. *The concept of turn-based Markov Decision Process resulting from the product of a split and a Markov Chain for P is illustrated in Figure 4.5. Observe that the state $\{a = 1, c =$*

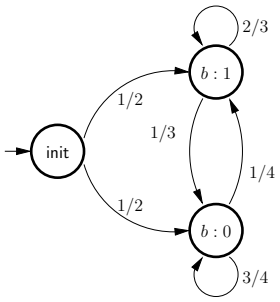


(a) A symbolic transition system $Symb$ for $V = \{a, b, c\}$, where the domain of a, c is $\{1\}$ and the domain of b is $\{0, 1\}$, b is the probabilistic input and the set of runs is given by $((a = 1, b = 1, c = 1) \cup (a = 1, b = 0, c = 1))^{\omega}$.

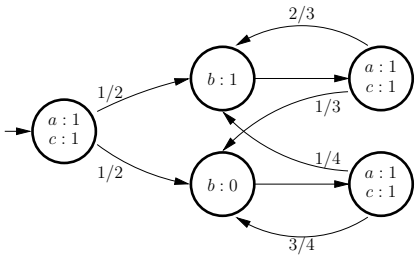


(b) The split $Symb''$ for $Symb$.

Figure 4.4: A symbolic transition system and its split.



(a) A Markov Chain for the distribution over variables in P.



(b) A MDP for the product between the Markov chain in Figure 4.5a and the transition system in Figure 4.4b.

Figure 4.5: The product of a split symbolic transition system with a Markov Chain.

$1\}$ *has been duplicated. Indeed, according to the Markov Chain in Figure 5.(a), the probability to select* $\{b = 0\}$ *in the first step is not the same as the one to select it after the first step.*

Assuming that the combination of the system with the distribution can be represented with a MDP, we now briefly discuss P-R-Satisfaction and P-A-Satisfaction. A *scheduler* for a Markov Decision Process [36] is a mechanism that, in a non-deterministic state, selects the successor state without taking predecessors into account. This definition matches the one we proposed in Definition 4.6. In this context, we have the following methodology.

- **P-R-Satisfaction**. Assuming that $\mathcal{B}_A$ and $\mathcal{B}_G$ are Büchi automata, P-R-Satisfaction can be checked with the technique introduced in [129, 51, 30] (which requires a determinization step from Büchi to deterministic Rabin [111]) and implemented in the *LIQUOR* toolset [35]. Indeed, this technique allows to compute the minimal probability for a Markov decision process to satisfy a property which is representable with a Büchi automaton. We can thus consider assumptions and guarantees represented with logical

formalisms that have a translation to Büchi automata, e.g., ETL [134].

- **P-A-Satisfaction with level m and discount factor d**. The DCTL logic can also be interpreted over MDPs. The definition of synchronous product easily extends to MDPs. The product between a MDP and an automaton can be interpreted as a MDP. We can thus use the labelling technique with propositions that was proposed for the non-probabilistic case (assuming that the states of the automaton have also been split (see the split for transition system)). For a given scheduler (which transforms the MDP into a Markov chain), we can compute the *expected value* for the formula $\triangle_d\ p$. We then compute the minimum between the expected values for all schedulers and check whether it is greater than m. More details about model checking DCTL over MDPs can be found in Section 2.2 of [53]. The overall formula we model check is $\forall E[\triangle_d\ p]$, where E states for "expected value".

4.5 Some Related Work

In this section, we compare our work with related work on contracts, process algebra, modal automata, and interface automata.

In [20], Benveniste et al. have presented a contract theory where availability, effective representations, and stochastic aspects are not considered. Other definitions of contracts have been proposed in [109, 67] and in [66], where the mathematical theory of [20] is recast in a reactive synchronous language setting. In [107], Pace and Schneider study the satisfaction of contracts that combines deontic and temporal concepts. Composition for such contracts is studied in [63, 62].

Works on behavioral types in process algebras bear commonalities with contract theories. In a similar way, the probabilistic contract theory must be compared with stochastic process algebras [103, 8]. In both cases, the main difference is that compositional reasoning is possible only in contract theories thanks to the fact that contracts are implications where an assumption implies a guarantee. A second major difference with process algebras, is that contract theories are general and can be instantiated in many different effective automata-based settings. This covers many logical frameworks (CTL [40], LTL [108], PCTL [73], PSL [60], ...) for specifying properties of components.

In [100], Larsen proposed *modal specifications* that correspond to *deterministic modal automata*, i.e., automata whose transitions are typed with *may* and *must* modalities. A modal specification thus represents a set of models; informally, a must transition is available in every component that implements the modal specification, while a may transition needs not be. The components that implement modal specifications are prefix-closed languages, or equivalently deterministic automata. As contracts, modal specifications support both refinement, conjunction, and composition operations. Moreover, modal specifications support a quotient operation which is the adjunct of parallel composition [114]. The theory has recently been extended to the timed setting [23, 22]. However, contrary to contracts, modal specifications do not allow an explicit treatment of assumptions and guarantees. It is also known that modal specifications are not more expressive than nu-calculus [64], while the theory of contracts is general and could potentially embed any type of property. Finally, aside from some attempts that we present in the next paragraph, there is no stochastic extension for modal specifications.

In Chapters 2 and 3, we have presented two specification formalisms for stochastic systems: Interval Markov Chains and Constraint Markov Chains. Both IMCs and CMCs are meant to provide a modeling language that allows designing, evolving and reusing components. However, contrary to the theory we present in this chapter, IMCs and CMCs do not allow an explicit treatment of assumption and guarantees. Moreover, they are graphical-based models, which is easy to use in a design setting. Unfortunately, they do not embed any notion of complex memory such as unbounded stack. Hence they are not capable of modeling calls and returns in software. The formalism we present in this chapter is more general, as it provides, for example, quantitative notions of satisfaction and refinement. Unfortunately, this generality comes with a cost: the operations involved in the assume-guarantee probabilistic contracts formalism are more complex than the equivalent operations for CMCs or IMC, often involving union and complementation.

In interface automata [54, 52], an interface is represented by an input/output automaton [104], *i.e.*, an automaton whose transitions are labeled with *input* or *output* actions. The semantics of such an automaton is given by a two-player game: an *Input* player represents the environment, and an *Output* player represents the component itself. Interface automata do not encompass any notion of model, because one cannot distinguish between interfaces and implementations. Alternatively, properties of interfaces are described in game-based logics, *e.g.*, ATL [6], with a high-cost complexity. The game-based interpretation offers a more elaborated version of the composition operation than our contracts approach. More precisely, the game-based interpretation offers an *optimistic* treatment of composition: two interfaces can be composed if there exists at least one environment (i.e., one strategy for the Input player) in which they can interact together in a safe way (i.e., whatever the strategy of the Output player is). This is referred as compatibility of interfaces. However, contrary to contracts, interface automata do not allow an explicit treatment of assumptions and guarantees and there is no stochastic extension.

Another assume-guarantee approach for the verification of systems consists in decomposing the system into sub-systems and choosing an adequate assumption for a particular decomposition (see [44] for a survey). As we already said, those works clearly differ from ours. First, they have to find a decomposition of the system in sub-systems, and second, they do not support compositional design operators (conjunction, refinement). Our work is much related to the work by Basu et al. [19] on the BIP toolset [26]. In their work, they do consider a much more elaborated composition operation. However, they do not consider conjunction, availability (they mostly restrict themselves to safety properties), and stochastic aspects. Finally, [97] presents assume-guarantee verification in which both assumption and guarantees are represented with finite probabilistic automata. Like IMCs and CMCs, probabilistic automata are graphical-based models, hence less general than the model we provide in this chapter. Though quantitative notions of satisfaction are proposed for safety properties, they do not consider availability.

4.6 Achievements and Future Work

In this chapter we have proposed a new theory for (probabilistic) contracts, which extends the one we developed for the European project *SPEEDS* [126]. Our contributions are : (1) a theory for reliability and availability, (2) a treatment of the stochastic aspects and (3) a discussion on effective symbolic representations.

In addition to implementation, there are various other directions for future research. A first direction is to develop a notion of quantitative refinement that is compatible with P-A-satisfaction. We also plan to consider other symbolic representations such as visibly pushdown systems [65]. Considering such representations will require new DCTL model checking algorithms. We also plan to extend our results to the continuous-time setting, which would also require a new DCTL algorithm based on the results in [12, 70, 106]. Considering the case of dependent probability distributions like in [56] is also a challenging issue. Finally, it would be interesting to define another satisfaction for contracts. Indeed, in this chapter, effective algorithms to check satisfaction of probabilistic contracts rely on formal exhaustive techniques [129]. Unfortunately, improvements in developments of formal methods do not seem to follows the increasing complexity in system design. In the next chapter, we will propose statistical model checking [137, 136, 123, 39] that is a scalable solution to this problem. The idea of the approach is to simulate the system and deduce whether it satisfies the property with some degree of confidence. Up to now statistical model checking algorithms have only been used to verify properties of stochastic system; it would be of interest to adapt the technique to the satisfaction of a probabilistic contracts. Unfortunately our contract formalism would be out of scope of existing statistical model checking algorithms. Indeed, these algorithms assume that all the samples are generated from the same distribution while our contracts allow non-deterministic aspects and hence build on several distributions.

Chapter 5

Statistical Abstraction and Model-Checking of Large Heterogeneous Systems

5.1 Introduction

In the previous chapters, we have mainly focused on system design and incremental verification. In this chapter, we are interested in verifying applications working within an heterogeneous system. Systems integrating multiple heterogeneous distributed applications communicating over a shared network are typical in various sensitive domains such as aeronautic or automotive embedded systems. Verifying the correctness of a particular application inside such a system is known to be a challenging task, which is often beyond the scope of existing exhaustive validation techniques. The main difficulty comes from network communication which makes all applications interfering and therefore forces exploration of the full state-space of the system.

A solution to this problem would be to use a test-based approach. After the computer system is constructed, it is tested using a number of *test cases* with predicted outcomes. Testing techniques have shown effectiveness in bug hunting in many industrial problems. Unfortunately, testing is not a panacea. Indeed, since there is, in general, no way for a finite set of test cases to cover all possible scenarios, errors may remain undetected.

This lack of accuracy has motivated the development of new algorithms that combine testing techniques with algorithms coming from the statistical area. Those techniques, also called *Statistical Model Checking techniques* (SMC) [78, 122, 136], can be seen as a trade-off between testing and formal verification. The core idea of the approach is to conduct some simulations of the system and verify if they do satisfy the property. The results are then used together with algorithms from the statistic area in order to decide whether the system satisfies the property with some probability. Statistical model checking techniques can also be used to estimate the probability that a system satisfies a given property [78, 68]. Of course, in contrast with an exhaustive approach, a simulation-based solution does not guarantee a result with 100% confidence. However, it is possible to bound the probability of making an error. Simulation-based methods are known to be far less memory and time intensive than exhaustive ones, and are sometimes the only option [138, 84]. Statistical model checking gets widely accepted in various research areas such as software engineering, in particular for industrial applications, or

even for solving problems originating from systems biology [41, 85]. There are several reasons for this success. First, it is very simple to implement, understand and use. Second, it does not require extra modeling or specification effort, but simply an operational model of the system, that can be simulated and checked against state-based properties. Third, it allows to verify properties [38, 39, 13] that cannot be expressed in classical temporal logics.

Unfortunately, SMC is also not a panacea and many important classes of systems are still out of its scope. Among them, one finds systems integrating multiple heterogeneous distributed applications communicating over a shared network. Those applications, also called *heterogeneous systems* are typical in various sensitive domains such as aeronautic or automotive embedded systems. Verifying the correctness of a particular application (also called subsystem, or combination of components) within such a system is known to be a challenging task, which is often beyond the scope of any validation technique. The main difficulty comes from network communication which makes all applications interfering and therefore forces to explore the full state-space of the system. One could hope that statistical model checking provides an alternative solution to this problem. Unfortunately, there are many cases where the design is so complex that it is even impossible to generate enough simulations for the algorithm to terminate in a decent time while providing estimates with sufficient accuracy.

We propose to exploit the structure of the system in order to increase the efficiency of the verification process. The idea is conceptually simple: instead of performing an analysis of the entire system, we propose to analyze each application separately, but under some particular context/execution environment. This context is a *stochastic abstraction* that represents the interactions with other applications running within the system and sharing the computation and communication resources. We propose to build such a context automatically by simulating the entire system and learning the probability distributions of key characteristics impacting the functionality of the given application.

The overall contribution of this chapter is an application of the above method on an industrial case study, the *heterogeneous communication system* (HCS for short) deployed for cabin communication in a civil airplane. HCS is a heterogeneous system providing entertainment services (e.g., audio/video on passengers demand) as well as administrative services (e.g., cabin illumination, control, audio announcements), which are implemented as distributed applications running in parallel, across various devices within the plane and communicating through a common Ethernet-based network. The HCS system has to guarantee stringent requirements, such as reliable data transmission, fault tolerance, timing and synchronization constraints. An important requirement, which will be studied in this chapter, is the *accuracy of clock synchronization* between different devices. This latter property states that the difference between the clocks of any two devices should be bounded by a small constant, which is provided by the user and depends on his needs. Hence, one must be capable of computing the smallest bound for which synchronization occurs and compare it with the bound expected by the user. Unfortunately, due to the large number of heterogeneous components that constitute the system, deriving such a bound manually from the textual specification is an unfeasible task. In this chapter, we propose a formal approach that consists in building a formal model of the HCS, then applying simulation-based algorithms to this model in order to deduce the smallest value of the bound for which synchronization occurs. We start with a fixed value of the bound and check whether synchronization occurs. If yes, then we make sure that this is the best one. If no, we restart the experiment with a new value.

At the top of our approach, there should be a tool that is capable of modeling heteroge-

neous systems as well as simulating their executions and the interactions between components. In this chapter, we propose to use the BIP toolset [15] for doing so. BIP (*Behaviour-Interaction-Priority*) supports a methodology for building systems from atomic components encapsulating behavior, that communicate through interactions, and coordinate through priorities. BIP also offers a powerful engine to simulate the system and can thus be combined with a statistical model checking algorithm in order to verify properties. Our first contribution is to study all the requirements for the HCS to work properly and then derive a model in BIP. Our second contribution is to study the accuracy of clock synchronization between several devices of the HCS. In HCS the clock synchronization is ensured by the *Precision Time Protocol* (PTP for short) [2], and the challenge is to guarantee that PTP maintains the difference between a master clock (running on a designated server within the system) and all the slave clocks (running on other devices) under some bound. Since this bound cannot be pre-computed, we have to verify the system for various values of the bound until we find a suitable one. Unfortunately, the full system is too big to be analyzed with classical exhaustive verification techniques. A solution could be to remove all the information that is not related to the devices under consideration. This is in fact not correct as the behavior of the PTP protocol is influenced by the other applications running in parallel within the heterogeneous system. Our solution to this state-space explosion problem is in two steps (1) we will build a stochastic abstraction for a part of the PTP application between the server and a given device; the stochastic part will be used to model the general context in which PTP is used, (2) we will apply statistical model checking on the resulting model.

Thanks to this approach, we have been able to derive precise bounds that guarantee proper synchronization for all the devices of the system. We also computed the probability of satisfying the property for smaller values of the bound, i.e., bounds that do not satisfy the synchronization property with probability 1. Being able to provide such information is of clear importance, especially when the best bound is too high with respect to the user's requirements. We have observed that the values we obtained strongly depend on the position of the device in the network. We also estimated the average proportion of failures per simulation for bounds that are smaller than the one that guarantees synchronization. Checking this latter property has been made easy because BIP allows us to reason on one execution at a time. Finally, we have also considered the influence of clock drift on the synchronisation results. The experiments highlight the generality of our technique, which could be applied to other versions of the HCS as well as to other heterogeneous applications.

The chapter is structured as follows. Section 5.2 briefly introduces the theory of statistical model checking. Section 5.3.1 describes the methods and tool we use in order to model and abstract the HCS. The case study and its modelization are then described in Section 5.4. In Section 5.5, we give details of the experiments we perform on the HCS, while the results of these experiments are presented in Section 5.6. Section 5.7 briefly presents another application of the methodology presented in the chapter. Finally, Section 5.8 concludes the chapter and discusses future work.

5.2 An Overview of Statistical Model Checking

Consider a stochastic system $\mathcal{S}$ and a property φ. *Statistical model checking* refers to a series of simulation-based techniques that can be used to answer two questions: (1) **Qualitative:** Is

the probability that $\mathcal{S}$ satisfies φ greater or equal to a certain threshold? and (2) **Quantitative:** What is the probability that $\mathcal{S}$ satisfies φ? Contrary to numerical approaches, the answer is given up to some correctness precision. In the rest of the section, we survey several statistical model checking techniques. Let B_i be a discrete random variable with a Bernoulli distribution of parameter p. Such a variable can only take 2 values (0 and 1) with $Pr[B_i = 1] = p$ and $Pr[B_i = 0] = 1 - p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies φ and 0 otherwise.

5.2.1 Qualitative Answer using Statistical Model Checking

The main approaches [136, 122, 92, 91, 139] proposed to answer the qualitative question are based on *hypothesis testing*. Let $p = Pr(\varphi)$, to determine whether $p \geq \theta$, we can test $H : p \geq \theta$ against $K : p < \theta$. A test-based solution does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* (α, β) of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds, called a Type-I error (respectively, a Type-II error) is less or equal to α (respectively, β). A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [136] for details). A solution is to use an *indifference region* $[p_1, p_0]$ (with θ in $[p_1, p_0]$) and to test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$. We now sketch two hypothesis testing algorithms.

Single Sampling Plan.

To test H_0 against H_1, we specify a constant c. If $\sum_{i=1}^{n} b_i$ is larger than c, then H_0 is accepted, else H_1 is accepted. The difficult part in this approach is to find values for the pair (n, c), called a *single sampling plan (SSP in short)*, such that the two error bounds α and β are respected. In practice, one tries to work with the smallest value of n possible so as to minimize the number of simulations performed. Clearly, this number has to be greater if α and β are smaller but also if the size of the indifference region is smaller. This results in an optimization problem, which generally does not have a closed-form solution except for a few special cases [136]. In his thesis [136], Younes proposes a binary search based algorithm that, given p_0, p_1, α, β, computes an approximation of the minimal value for c and n.

Sequential probability ratio test. The sample size for a single sampling plan is fixed in advance and independent of the observations that are made. However, taking those observations into account can increase the performance of the test. As an example, if we use a single plan (n, c) and the $m > c$ first simulations satisfy the property, then we could (depending on the error bounds) accept H_0 without observing the $n - m$ other simulations. To overcome this problem, one can use the *sequential probability ratio test (SPRT in short)* proposed by Wald [133]. The approach is briefly described below.

In SPRT, one has to choose two values A and B ($A > B$) that ensure that the strength of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr(B_i = b_i \mid p = p_1)}{Pr(B_i = b_i \mid p = p_0)} = \frac{p_1^{d_m}(1 - p_1)^{m-d_m}}{p_0^{d_m}(1 - p_0)^{m-d_m}}, \tag{5.1}$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea behind the test is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. The SPRT algorithm computes $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied; the algorithm terminates with probability 1[133]. This has the advantage of minimizing the number of simulations. In his thesis [136], Younes proposed a logarithmic based algorithm SPRT that given p_0, p_1, α and β implements the sequential ratio testing procedure.

Computing ideal values A_{id} and B_{id} for A and B in order to make sure that we are working with a test of strength (α, β) is a laborious procedure. In his seminal paper [133], Wald showed that if we define $A_{id} \geq A = \frac{(1-\beta)}{\alpha}$ and $B_{id} \leq B = \frac{\beta}{(1-\alpha)}$, then we obtain a new test whose strength is (α', β') such that $\alpha' + \beta' \leq \alpha + \beta$. This means that either $\alpha' \leq \alpha$ or $\beta' \leq \beta$. In practice, we often find that both inequalities hold.

5.2.2 Quantitative Answer using Statistical Model Checking

In [78, 99] Peyronnet et al. propose an estimation procedure to compute the probability p for $\mathcal{S}$ to satisfy φ. Given a *precision* δ, Peyronnet's procedure, which we call PESTIMATION, computes a value for p' such that $|p' - p| \leq \delta$ with *confidence* $1 - \alpha$. The procedure is based on the *Chernoff-Hoeffding bound* [83]. Let $B_1 \ldots B_m$ be m discrete random variables with a Bernoulli distribution of parameter p associated with m simulations of the system. Recall that the outcome for each of the B_i, denoted b_i, is 1 if the simulation satisfies φ and 0 otherwise. Let $p' = (\sum_{i=1}^{m} b_i)/m$, then Chernoff-Hoeffding bound [83] gives $Pr(|p' - p| > \delta) < 2e^{-\frac{m\delta^2}{4}}$. As a consequence, if we take $m \geq \frac{4}{\delta^2} \log(\frac{2}{\alpha})$, then $Pr(|p' - p| \leq \delta) \geq 1 - \alpha$. Observe that if the value p' returned by PESTIMATION is such that $p' \geq \theta - \delta$, then $\mathcal{S} \models Pr_{\geq\theta}$ with confidence $1 - \alpha$.

5.2.3 Playing with Statistical Model Checking Algorithms

The efficiency of the above algorithms is characterized by the number of simulations needed to obtain an answer. This number may change from executions to executions and can only be estimated (see [136] for an explanation). However, some generalities are known. For the qualitative case, it is known that, except for some situations, SPRT is always faster than SSP. When $\theta = 1$ (resp. $\theta = 0$) SPRT degenerates to SSP; this is not problematic since SSP is known to be optimal for such values. PESTIMATION can also be used to solve the qualitative problem, but it is always slower than SSP [136]. If θ is unknown, then a good strategy is to estimate it using PESTIMATION with a low confidence and then validate the result with SPRT and a strong confidence.

5.3 Validation Method and the BIP Toolset

We first present the method we use in order to abstract our case study. Then we describe the tool used for the modelization: BIP.

5.3.1 Validation Method: Stochastic Abstraction

Consider a system consisting of a set of distributed applications running on several computers and exchanging messages on a shared network infrastructure. Assume also that network

communication is subject to given bandwidth restrictions as well as to routing and scheduling policies applied on network elements. Our method attempts to reduce the complexity of validation of a particular application of such system by decoupling the timing analysis of the network and functional analysis of each application.

We start by constructing a model of the whole system. This model must be executable, i.e., it should be possible to obtain execution traces, annotated with timing information. For a chosen application, we then learn the probability distribution laws of its message delays by simulating the entire system. The method then constructs a reduced stochastic model by combining the application model where the delays are defined according to the laws identified at the previous step. Finally, the method applies statistical model-checking on the resulting stochastic model.

Our models are specified within the BIP framework [15]. BIP is a component-based framework for construction, implementation and analysis of systems composed of heterogeneous components. In particular, BIP fulfills all the requirements of the method suggested above, that are, models constructed in BIP are operational and can be thoroughly simulated. BIP models can easily integrate timing constraints, which are represented with discrete clocks. Probabilistic behaviour can also be added by using external C functions.

The BIP framework is implemented within the BIP toolset [26], which includes a rich set of tools for modeling, execution, analysis (both static and on-the-fly) and static transformations of BIP models. It provides a dedicated programming language for describing BIP models. The front-end tools allow editing and parsing of BIP programs, and generating an intermediate model, followed by code generation (in C) for execution and analysis on a dedicated middleware platform. The platform also offers connections to external analysis tools. A more complete description of BIP is given in the next section.

5.3.2 An Overview of BIP

The BIP framework, presented in [15], supports a methodology for building systems from *atomic components*. It uses *connectors*, to specify possible interaction patterns between components, and *priorities*, to select amongst possible interactions. In BIP, data and their transformations can be written directly in C.

Atomic components are finite-state automata extended with variables and ports. Ports are action names, and may be associated with variables. They are used for synchronization with other components. Control states denote locations at which the components await for synchronization. Variables are used to store local data.

We provide in Figure 5.1 an example of an atomic component, named *Router*, that models the behavior of a network router. It receives network packets through an input port and delivers them to the respective output port(s), based on the destination address of the packets. The port *srvRecv* acts as an input port, while $s0$, $s1$, $s2$, $s3$, and *subNetSend* act as output ports. The port *tick* is used for modeling time progress and specific timing constraints. The control locations are *RECV*, *SEND*, *SEND0*, *SEND1*, *SEND2*, *SEND3*, *SENDING* and *GAP*, with *RECV* being the initial location. It also has the variables *t, p, to_0, to_1, to_2, to_3, to_sub, to_all,frame,* and parameter *frameGap*.

A transition is a step from a control location to another, guarded by a Boolean condition on the set of its variables, labeled by a port. An example transition is from the initial location *RECV* to *SEND*, which is executed when an interaction including port *srvRecv* takes place, the

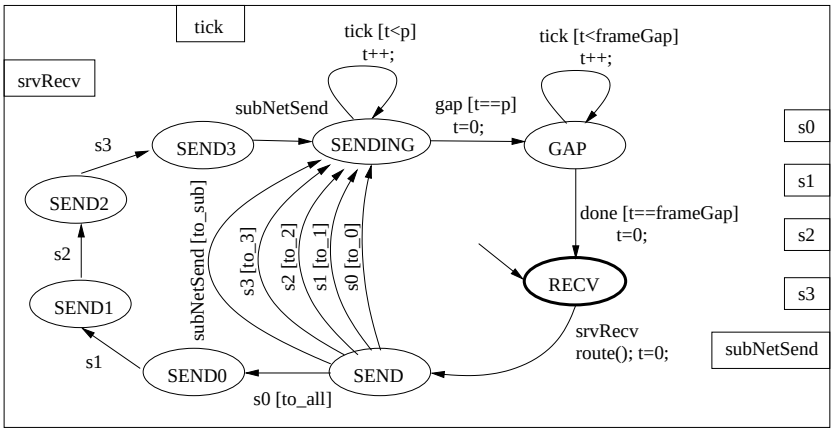


Figure 5.1: An atomic component: Router.

default guard being *true*. On execution, the internal computation step is the execution of the C routine *route()*, followed by the reset of the variable *t*.

Composite components allow defining new components from sub-components (atomic or composite). Components are connected through flat or hierarchical *connectors*, which relate ports from different sub-components. Connectors represent sets of interactions, that are, non-empty sets of ports that have to be jointly executed. They also specify guards and transfer functions for each interaction, that is, the enabling condition and the exchange of data across the ports of the interacting components.

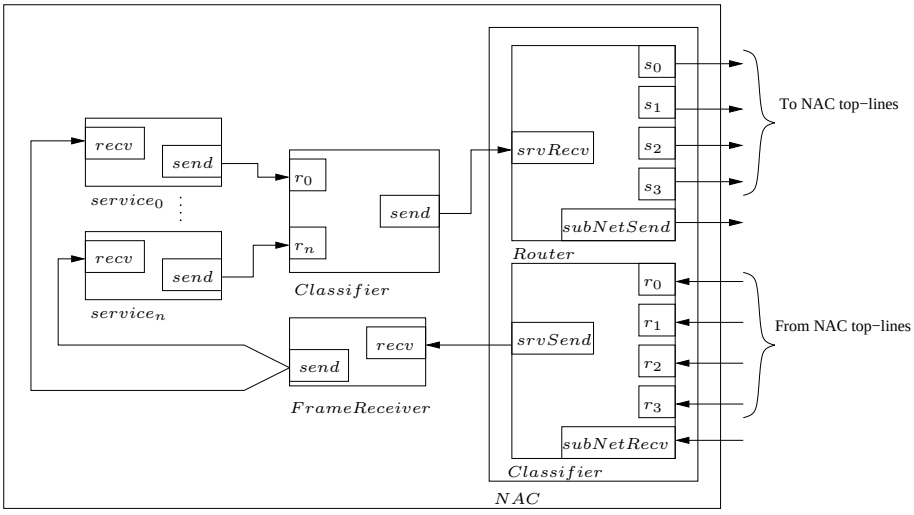


Figure 5.2: Composite Component: Server.

Figure 5.2 shows a composite component *Server*. It contains atomic components *service*$_0$ $\cdots$ *service*$_n$, *FrameReceiver*, and composite components *Classifier* and *NAC*. The *NAC* contains a *Router* and a*Classifier*. The connectors are shown by lines joining the ports of the components.

Priorities are used to select amongst simultaneously enabled interactions. They are a set of

rules, each consisting of an ordered pair of interactions associated with a condition. When the condition holds and both interactions of the corresponding pair are enabled, only the one with higher-priority can be executed.

The architecture of a generic device is shown in Figure 5.3.

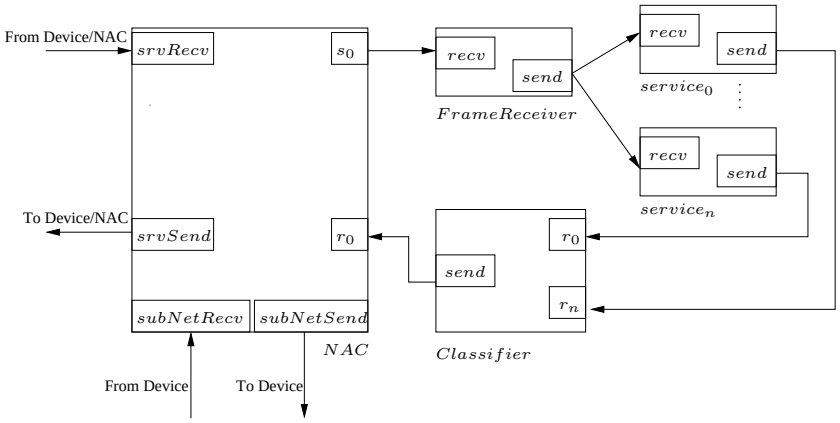


Figure 5.3: A device component.

5.4 Case Study: Heterogeneous Communication System

The case study concerns a distributed heterogeneous communication system (HCS) providing an all electronic communication infrastructure to be deployed, typically for cabin communication in airplanes or for building automation. The HCS system contains various devices such as sensors (video camera, smoke detector, temperature, pressure, etc.) and actuators (loudspeakers, light switches, temperature control, signs, etc.) connected through a wired communication network to a common server. The server runs a set of services to monitor the sensors and to control the actuators. The devices are connected to the server using network access controllers (NAC) as shown for an example architecture in Figure 5.4. An extended star-like HCS architecture with several daisy chains of NACs and devices is presented in Figure 5.5.

The system-wide HCS architecture is highly heterogeneous. It includes hardware components and software applications ensuring functions with different characteristics and degree of criticality e.g, audio streaming, device clock synchronisation, sensor monitoring, video surveillance. It also integrates different communication and management protocols between components. The HCS system has to guarantee stringent requirements, such as reliable data transmission, fault tolerance, timings and synchronization constraints. For example, the latency for delivering alarm signals from sensors, or for playing audio announcements should be smaller than certain predefined thresholds. Or, the accuracy of clock synchronization between different devices, should be guaranteed under the given physical implementation of the system.

The HCS case study poses challenges that require component-based design techniques, since it involves heterogeneous components and communication mechanisms, e.g. streaming based on the data-flow paradigm as well as event driven computation and interaction. Its modeling needs combination of executable and analytic models especially for performance evaluation and analysis of non-functional properties.

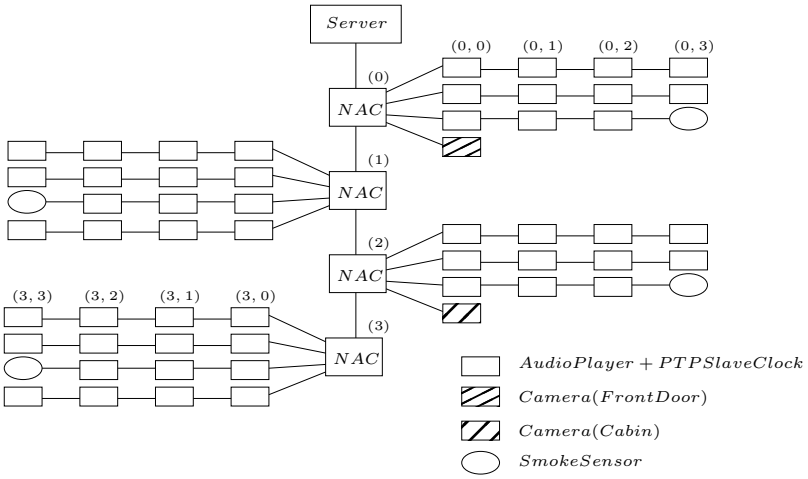


Figure 5.4: HCS Example Model.

We have modeled an instance of the HCS system in BIP. As shown in Figure 5.4, the system consists of one *Server* connected to a daisy chain of four NACs, addressed $0 \cdots 3$, and several devices. Devices are connected in daisy chains with the NACs, the length of each chain being limited to four in our example. For simplicity, devices are addressed (i, j), where i is the address of the NAC and j is the address of the device. The model contains three types of devices, namely *Audio Player*, *Video Camera* and *Smoke Sensor*. The devices connected to NAC(0) and NAC(2) have similar topology. The first two daisy-chains consist of only *Audio Player* devices. The third daisy-chain ends with a *Smoke Sensor*, and the fourth daisy-chain consists of just one *Video Camera*. The devices connected to NAC(1) and NAC(3) have exactly the same topology, consisting of several *Audio Player* and one *Smoke Sensor* devices.

The system depicted in Figure 5.4 contains 58 devices in total. The BIP model contains 297 atomic components, 245 clocks, and its state-space is of order 2^{3000}. The size of the BIP code for describing the system is 2468 lines, which is translated to 9018 lines in C. A description of the key components of the HCS is given hereafter.

5.4.1 Server

The server runs various applications including: 1) PTP *MasterClock*, that runs the PTP master-clock protocol between the server and the devices in order to keep the device clocks synchronized with the master-clock. The protocol exchanges PTP packets of size 512 bits between the server and the devices, and runs once every 2 minutes. 2) *AudioGenerator*, that generates audio streams to be playbacked by the *Audio Player* devices. It generates audio streams at 32kHz with 12 bit resolution (audio chunks). We have assumed that 100 audio chunks are sent in a single frame over the network, (that gives the size of an audio frame to be 1344 bits) at the rate of 33 frames per second. 3) *SmokeDetector* service that keeps track of the event packets (size 736 bits) sent from the *Smoke Sensor*, and 4) *VideoSurveillance* service for monitoring the *Video Cameras*. In addition, the server needs to handle the scheduling and routing of the generated Ethernet packets over the communication backbone. The scheduling and routing of the packets is handled by the NAC component.

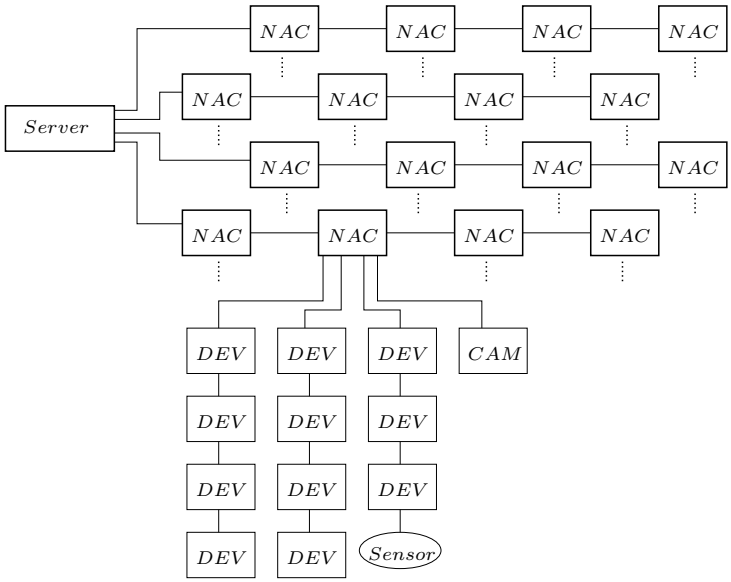


Figure 5.5: Heterogeneous Communication System (HCS).

5.4.2 Network Access Controller (NAC)

The NACs perform the data routing from the server to the subnet devices and vice versa. A NAC essentially consists of a *router* (as shown in Figure 5.1 in Section 5.3.2), that transmits the packets from the server to the devices, and a *classifier* (see Figure 5.6), that sends the packets from the devices to the server. The classifier enforces a scheduling on the packets to be sent, based on their types. As a result, packets may be queued up in the NACs adding to their delay en route to the server. Hence, the scheduling policy in the classifier plays an important role in the transmission delay of the packets.

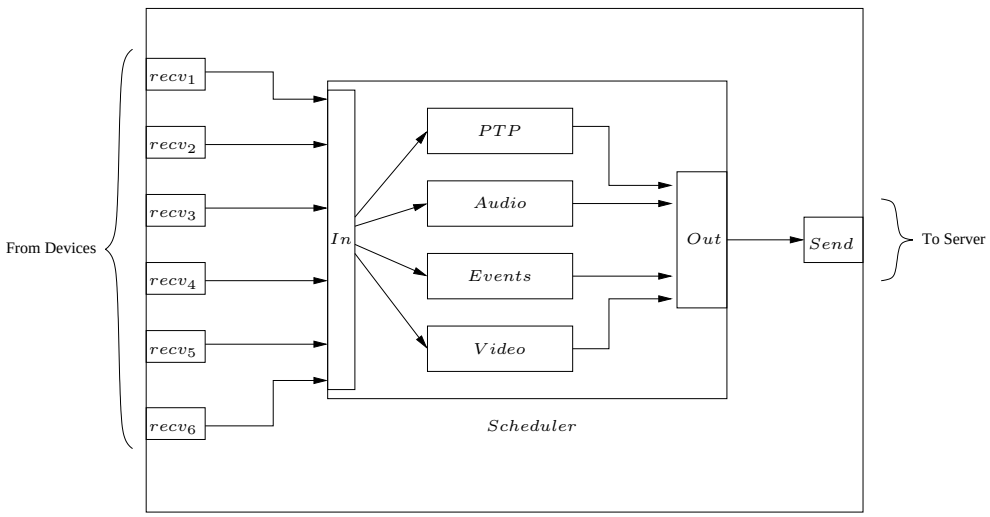


Figure 5.6: Component: Classifier

We have implemented and tested two scheduling schemes. The first scheduling policy is

based on static priorities of the packets. The second policy, called weighted fair queuing mixed with priorities, is introduced in order to give a fair share of the bandwidth of the network to each type of packets.

Fixed priorities. With this algorithm, the packets transfered on the network are classified in four categories that are (1) PTP, (2) Audio, (3) Events and (4) video. The PTP packets are exchanged in the process of the PTP synchronization. They will be further detailed in Section 5.5.1. Audio packets are sent from the server to the audio devices. A priori, since these packets are going from the server to the devices, they will not have to be scheduled – the scheduling is done by the server before sending the packets. Events packets are sent by smoke detectors to the server. Finally, Video packets correspond to traffic between video camera devices and the server.
It is possible to classify these packets by order of importance. The highest priority goes to PTP packets. Indeed, they need to be transmitted as fast as possible because they are critical for the synchronization of the system. Audio and Events packets may be critical in case of a punctual problem during the flight: if a fire is detected, then the information has to be transmitted as soon as possible to the server. On the other hand, if a critical problem is detected, the passengers have to be informed without delay. Finally, the Video packets are less critical.
One can use this classification to define scheduling in the NACs by following the order of importance it defines. This is the principle of fixed priorities: use as many FIFO buffers to store the incoming packets as there are levels of priorities. When several buffers are ready to send, empty first the one with the highest priority, then the next, etc...

Unfortunately, if the network is flooded by high-priority packets, then the low-priority packets are never sent. This problem may be solved by using another scheduling algorithm that we now present.

Weighted fair queuing (WFQ). Weighted fair queuing (WFQ) is a scheduling algorithm that allocates to each data flow a share of the total data rate of the physical links. In WFQ, as in fixed priorities, the packets are classified in categories. Each category has its own FIFO buffer in the switches. The difference with fixed priorities is that for WFQ, each category is given a weight that will not act has a priority but has a regulation of the data flow: Consider a physical link of data rate R. If there are N active categories with weights $w_1, w_2, \ldots w_N$, meaning that their buffers are non-empty, the WFQ scheduler will ensure that the category number i achieves an average data rate of

$$\frac{R \cdot w_i}{w_1 + w_2 \ldots + w_N}.$$

In practice, a scheduler will keep a dynamic information on the data rate of each category of packets, and will only transmit packets corresponding to a category for which the current data rate is under the specified rate. What we have implemented is a modified version of WFQ called WFQ mixed with priorities.

WFQ mixed with priorities. In this version, packets are both assigned a weight and an order of priority. Moreover, we fix a size for the window over which each category must satisfy its allocated rate. This means that packets of a size much smaller than the window may sometimes be transfered with a higher rate than allocated, provided that the average rate on

the total window is still respected. The priorities are added in order to partially resolve non-determinism in the scheduling: when several packets can be transmitted without violating the rates, the one with the highest priority is sent first.

The main drawback of WFQ mixed with priorities is that the delays of packets within one category can have a high variance. Indeed, the packets of a category with a high priority may be sent at the beginning of a window, introducing low delays for them. Once the "quota" is reached, however, all the remaining packets have to wait for the next window before being sent, introducing a high delay.

5.4.3 Device

Each device run one or more services which either generate packets for the server, or consumes packets generated from the server. As devices are connected in daisy chains, they also perform routing of packets, hence each device provides a NAC functionality. Services modeled in our example are *Audio Player*, PTP *SlaveClock*, *Smoke Sensor* and *Video Camera*. Video frames are generated at a rate of 25 frames per second, the size of the video frames being given as a distribution. Separate distributions are provided for high-resolution camera (with mean frame size of 120 kb) and for the low-resolution camera (with mean frame size of 30kb). The architecture of a generic device is shown in Figure 5.3 in Section 5.3.2.

5.4.4 Complexity of the modeling

Table 5.1 gives an overview about the number and the complexity of model components defined in BIP. The columns are as follows: S is the number of control locations; V_d is the number of discrete variables (can be Boolean or arbitrary type like a frame or an array of frames); V_t is number of clocks; C is the clock range; $Size$ is the approximated size of the state-space; and $Number$ is the number of occurrences of the module in the example.

5.5 Experiments on the HCS

One of the core applications of the HCS case study is the PTP protocol, which allows the synchronization of the clocks of the various devices with the one of the server. It is important that this synchronization occurs properly, i.e., that the difference between the clock of the server and the one of any device is bounded by a small constant. Studying this problem is the subject of this section. Since the BIP model for the HCS is extremely large (number of components, size of the state space, ...), there is no hope to analyse it with an exhaustive verification technique. Here, we propose to apply our stochastic abstraction. Given a specific device, we will proceed in two steps. First, we will conduct simulations on the entire system in order to learn the probability distribution on the communication delays between this device and the server. Second, we will use this information to build a stochastic abstraction of the application on which we will apply statistical model checking. We start with the stochastic abstraction for the PTP.

5.5.1 The Precision Time Protocol IEEE 1588

The Precision Time Protocol [2] has been defined to synchronize clocks of several computers interconnected over a network. The protocol relies on multicast communication to distribute a

Component type	Name	S	V_d	V_t	C	Size	Number
Atomic	Router	8	7	1	5-120	2^{11}	63
	Forwarder	4	1	1	5-120	2^{8}	-
	FrameReceiver	2	1	1	5-120	2^{7}	-
	MasterClock	3	1	1	0-2000	2^{12}	1
	AudioGenerator	2	1	1	0-3125	2^{13}	1
	SmokeDetector	3	1	1	0-300	2^{10}	4
	VideoGenerator	3	1	1	0-40000	2^{16}	2
Compound	NAC	-	-	-	-	2^{34}	63
	Server	-	-	-	-	2^{86}	1
	Audio Player	-	-	-	-	2^{44}	52
	Camera	-	-	-	-	2^{50}	2
	SmokeSensor	-	-	-	-	2^{51}	4
	HCS System	-	-	-	-	2^{3122}	1

Table 5.1: State-space estimation.

reference time from an accurate clock (*the master*) to all other clocks in the network (*the slaves*) combined with individual offset correction, for each slave, according to its specific round-trip communication delay to the master. The accuracy of synchronization is negatively impacted by the jitter (i.e., the variation) and the asymmetry of the communication delay between the master and the slaves. Obviously, these delay characteristics are highly dependent on the network architecture as well as on the ongoing network traffic.

We present below the abstract stochastic model of the PTP protocol between a device and the server in the HCS case study. The model consists of two (deterministic) application components respectively, the master and the slave clocks, and two probabilistic components, the media, which are abstraction of the communication network between the master and the slave. The former represent the behaviour of the protocol and are described by extended timed i/o-automata. The latter represent a random transport delay and are simply described by probabilistic distributions. Recap that randomization is used to represent the context, i.e., behaviors of other devices and influence of these behaviors on those of the master and the device under consideration.

The time of the master process is represented by the clock variable θ_m. This is considered the reference time and is used to synchronize the time of the slave clock, represented by the clock variable θ_s. The synchronization works as follows. Periodically, the master broadcast a *sync* message and immediately after a *followUp* message containing the time t_1 at which the *sync* message has been sent. Time t_1 is observed on the master clock θ_m. The slave records in t_2 the reception time of the *sync* message. Then, after the reception of the *followUp*, it sends a delay *request* message to the master and records its emission time t_3. Both t_2 and t_3 are observed on the slave clock θ_s. The master records on t_4 the reception time of the *request* message and sends it back to the slave on the *reply* message. Again, t_4 is observed on the

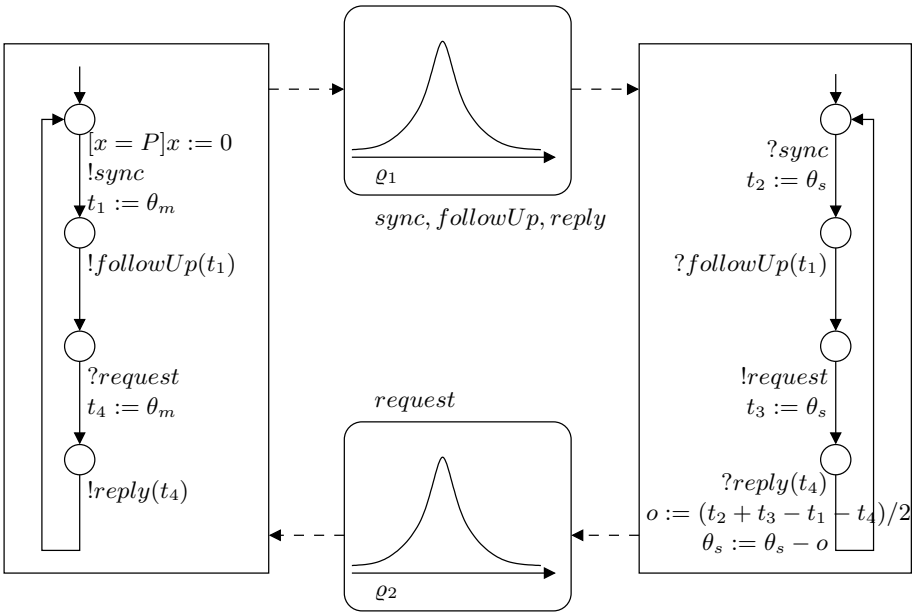


Figure 5.7: Abstract stochastic PTP between the server and a device.

master clock θ_m. Finally, upon reception of *reply*, the slave computes the offset between its time and the master time based on $(t_i)_{i=1,4}$ and updates its clock accordingly. In our model, the offset is computed differently in two different situations. In the first situation, which is depicted in Figure 5.7, the average delays from master to slave and back are supposed to be equal i.e., $\mu(\varrho_1) = \mu(\varrho_2)$. In the second situation, delays are supposed to be asymmetric, i.e., $\mu(\varrho_1) \neq \mu(\varrho_2)$. In this case, synchronization is improved by using an extra offset correction which compensate for the difference, more precisely, $o := (t_2 + t_3 - t_1 - t_4)/2 + (\mu(\varrho_2) - \mu(\varrho_1))/2$. This offset computation is an extension of the PTP specification and has been considered since it ensures better precision when delays are not symmetric (see Section 5.5).

Encoding the abstract model of timed i/o-automata given in Figure 5.7 in BIP is quite straightforward and can be done with the method presented in [15]. The distribution on the delay is implemented as a new C function in the BIP model. It is worth mentioning that, since the two i/o automata are deterministic, the full system depicted in Figure 5.7 is purely stochastic.

The accuracy of the synchronization is defined by the absolute value of the difference between the master and slave clocks $|\theta_m - \theta_s|$, during the time. Our aim is to check the (safety) property of bounded accuracy φ_Δ, that is, *always* $|\theta_m - \theta_s| \leq \Delta$ for arbitrary fixed non-negative real Δ.

Finally, a simpler version of this protocol is considered and analyzed in Section 5.5.2. In that study, delay components have been modeled using non-deterministic timed i/o automata as well and represent arbitrary delays bounded in some intervals $[L, U]$. It is shown that, if the clock drift is negligible, the best accuracy $\Delta^\star$ that can be obtained using PTP is respectively $\frac{U-L}{2}$ in the symmetric case, and $\frac{U_1+U_2-L_1-L_2}{4}$ in the asymmetric case. That is, the property of bounded accuracy holds trivially iff $\Delta \geq \Delta^*$.

5.5.2 Parametric Precision Estimation for PTP

We introduce hereafter an analytic method to estimate the precision achieved within one round of the PTP protocol, depending on several (abstract) parameters such as the initial difference and the bounds (lower, upper) on the allowed drift of the two clocks, the bounds (lower, upper) of the communication delay between the master and the slave, etc.

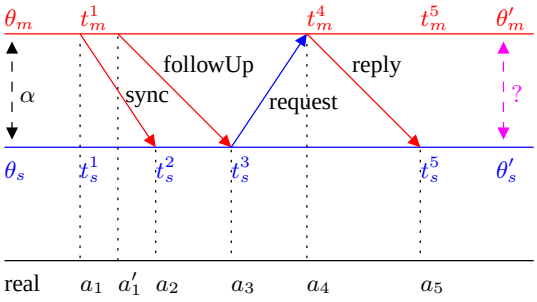


Figure 5.8: One round of the PTP protocol.

The difference between the master and the slave clocks after one PTP round can be determined from a system of arithmetic non-linear constraints extracted from the model of the protocol and communication media. Let us consider one complete round of the protocol as depicted in Figure 5.8. The first two axes correspond to the (inaccurate) clocks of the master and slave respectively. The third axis correspond to a perfect reference clock. Using the notation defined on the figure we can establish several constraints relating initial and final values of the master and slave clocks ($\theta_m, \theta_s, \theta'_m, \theta'_s$), timestamps ($t_1, t_2, t_3, t_4$), offset ($o$), communication delays (L_1, U_1, L_2, U_2), reference dates (a_1, a'_1, a_2, a_2, a_4) as follows:

- initial constraints and initial clock difference α
  $\theta_m - \theta_s = \alpha$, $\theta_m = t_m^1$, $\theta_s = t_s^1$
- evolution of the master clock is constrained by some maximal drift ϵ_m
  $(1 - \epsilon_m)(a_4 - a_1) \leq t_m^4 - t_m^1 \leq (1 + \epsilon_m)(a_4 - a_1)$
  $(1 - \epsilon_m)(a_5 - a_4) \leq t_m^5 - t_m^4 \leq (1 + \epsilon_m)(a_5 - a_4)$
- evolution of the slave clock is constrained by some maximal drift ϵ_s
  $(1 - \epsilon_s)(a_2 - a_1) \leq t_s^2 - t_s^1 \leq (1 + \epsilon_s)(a_2 - a_1)$
  $(1 - \epsilon_s)(a_3 - a_2) \leq t_s^3 - t_s^2 \leq (1 + \epsilon_s)(a_3 - a_2)$
  $(1 - \epsilon_s)(a_5 - a_3) \leq t_s^5 - t_s^3 \leq (1 + \epsilon_s)(a_5 - a_3)$
- communication delays, forward (L_1, U_1) and backward (L_2, U_2)
  $L_1 \leq a_2 - a_1 \leq U_1$
  $L_1 \leq a_3 - a'_1 \leq U_1$
  $L_2 \leq a_4 - a_3 \leq U_2$
  $L_1 \leq a_5 - a_4 \leq U_1$
- internal master delay (l, u) for sending the *followUp* after *sync*
  $l \leq a'_1 - a_1 \leq u$

- offset computation and final clocks values
  $o = (t_s^2 + t_s^3 - t_m^1 - t_m^4)/2$, $\theta'_m = t_m^5$, $\theta'_s = t_s^5 - o$

This system of constraints encodes precisely the evolution of the two clocks within one round of the protocol. The synchronization achieved correspond to the difference $\theta'_m - \theta'_s$. We analyze different configurations and we obtain the following results:

1. symmetric delays $L_1 = L_2 = L$, $U_1 = U_2 = U$, no drift $\epsilon_m = \epsilon_s = 0$

$$-\frac{U-L}{2} \leq \theta'_m - \theta'_s \leq \frac{U-L}{2}$$

2. symmetric delays $L_1 = L_2 = L$, $U_1 = U_2 = U$, no master drift $\epsilon_m = 0$

$$-\frac{U-L}{2} - \frac{\epsilon_s(5U - L + u)}{2} \leq \theta'_m - \theta'_s \leq \frac{U-L}{2} + \frac{\epsilon_s(2U + 2L + u)}{2}$$

3. asymmetric delays, no drift $\epsilon_m = \epsilon_s = 0$

$$-\frac{U_2 - L_1}{2} \leq \theta'_m - \theta'_s \leq \frac{U_1 - L_2}{2}$$

4. asymmetric delays, no master drift $\epsilon_m = 0$

$$-\frac{U_2 - L_1}{2} - \frac{\epsilon_s(3U_1 + 2U_2 - L_1 + u)}{2} \leq \theta'_m - \theta'_s \leq \frac{U_1 - L_2}{2} + \frac{\epsilon_s(2U_1 + 2L_2 + u)}{2}$$

We remark that, in general, the precision achieved does not depend on the initial difference between the two clocks. Nevertheless, it is strongly impacted by the communication jitter, which is, the difference $U - L$ in the symmetric case and differences $U_2 - L_1, U_1 - L_2$ in the asymmetric case.

Moreover, we remark that in the asymmetric case, the lower and upper bounds are not *symmetric* i.e., the precision interval obtained is not centered around 0. The bounds of the interval suggest us an additional offset correction:

$$\delta_o = \frac{(U_2 - U_1) + (L_2 - L_1)}{4}$$

which will *shift* the interval towards 0. For example, using this additional correction we obtain in the case of asymmetric delays with no drift better precision:

$$-\frac{(U_2 + U_1) - (L_1 + L_2)}{4} \leq \theta'_m - \theta'_s \leq \frac{(U_1 + U_2) - (L_1 + L_2)}{4}$$

5.5.3 Model Simulations

In this section, we describe our approach to learn the probability distribution over the delays. Consider the server and a given device. In a first step, we run simulations on the system and measure the end-to-end delays of all PTP messages between the selected device and the server. For example, consider the case of delay *request* messages and assume that we made 33 measures. The result will be a series of delay values and, for each value, the number of times it has been observed. As an example, delay 5 has been observed 3 times, delay 19 has been observed 30 times. The probability distribution is represented with a table of 33 cells. In our case, 3 cells of the table will contains the value 5 and 30 will contain the value 19. The BIP engine will select a value in the table following a uniform probability distribution. This method is used both for the fixed priority scheduling and for the weighted fair queuing mixed with priorities.

According to our experiments, 2000 delay measurements are enough to obtain an accurate estimation of the probability distribution. Indeed, from a statistics point of view, a sample consisting of 2000 values is more than enough in order to learn accurately a probability distribution without having to apply kernel methods [125, 88, 120] or bootstraping [58, 59]. In the case of fixed priorities, we have observed that it is possible to conduct 4000 measurements without being too time-consuming. Indeed, each simulation for 4000 measurements takes approximately 40 minutes on a Pentium 4 running under a Linux distribution. In this case, we have thus conducted 4000 measurements. In the case of weighted fair queuing mixed with priorities, since the scheduling algorithm is more complex, only 2000 measurements have been performed for each delay. Indeed, in this case, each simulation for 2000 measurements takes approximately 3 hours on the same Pentium 4.

Regardless of the scheduling algorithm, we have observed that the value of the distribution clearly depends on the position of the device in the topology. This is shown in Figure 5.9 for fixed priorities and Figure 5.10 for weighted fair queuing mixed with priorities. In both figures, the solid plot shows the distribution of delays from Device(0,3) to the server and the dashed plot shows the delay from Device(3,3)to the server.

It is worth mentioning that running one single simulation allowing 4000/2000 measurements of the delay of PTP frames requires running the PTP protocol with an increased frequency i.e., the default PTP period (2 minutes) being far too big compared with the period for sending audio/video packets (tens of milliseconds). Therefore, we run simulations where PTP is executed once every 2 milliseconds and, we obtain 4000/2000 measurements by simulating approximately 8/4 seconds of the global system lifetime. Each simulation uses microsecond time granularity.

5.6 Experiments on Precision Estimation for PTP

Three sets of experiments are conducted. The first one is concerned with the bounded accuracy property (see Section 5.5.1). In the second one, we study average failure per execution for a given bound. Finally, we study the influence of drift on the results. We do these experiments for the two scheduling policies described in Section 5.4.2.

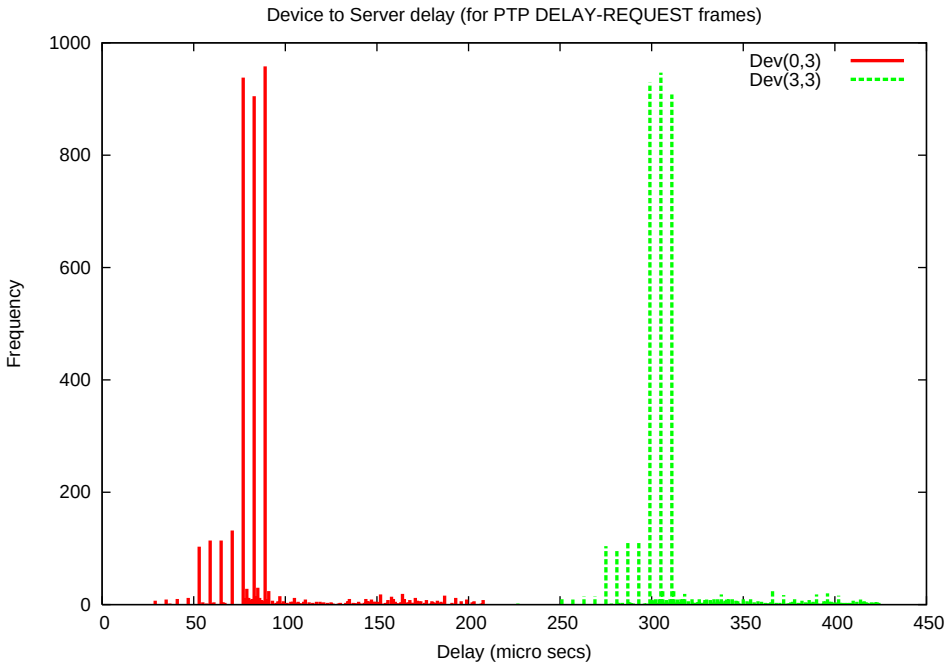


Figure 5.9: Delay distribution for Device(0,3) and Device(3,3) using fixed priorities for 4000 measurements.

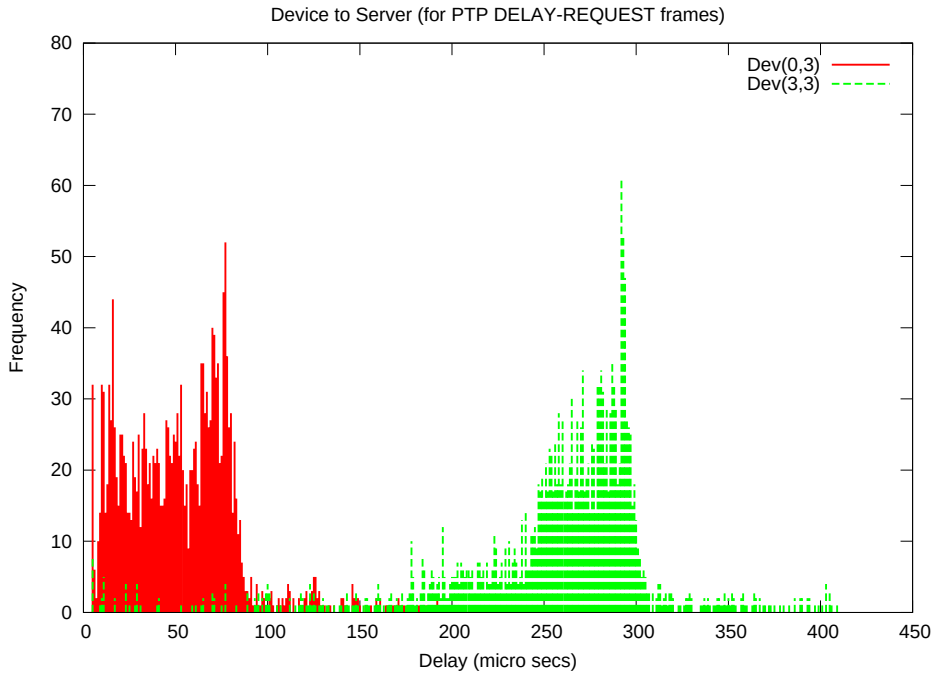


Figure 5.10: Delay distribution for Device(0,3) and Device(3,3) using weighted fair queuing mixed with priorities for 2000 measurements.

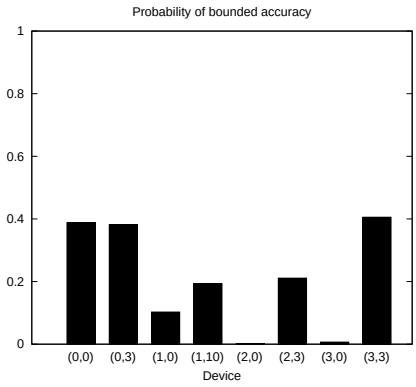


Figure 5.11: Probability of satisfying the bounded accuracy property for a bound $\Delta = 50\mu s$ and the asymmetric version of PTP.

5.6.1 Property 1: Synchronization

Our objective is to compute the smallest bound Δ under which synchronization occurs properly for any device.

Experiment 1. We start with an experiment that shows that the value of the bound depends on the place of the device in the topology. For doing so, we use $\Delta = 50\mu s$ as a bound and then compute the probability for synchronization to occur properly for all the devices. For the sake of presentation, we will only report on a sampled set of devices: $(0,0)$, $(0,3)$, $(1,0)$, $(1,10)$, $(2,0)$, $(2,3)$, $(3,0)$, $(3,3)$, but our global observations extend to any device. We use PESTIMATION with a confidence of 0.1. We first report the results we obtained using the fixed priority scheduling algorithm. Then we compare these results to the ones obtained for weighted fair queuing mixed with priorities, for several configurations of the weights and window size.
Fixed Priorities: The results, which are reported in Figure 5.11, show that the place in the topology plays a crucial role. Device $(3,3)$ has the best probability value and Device $(2,0)$ has the worst one. All the results in Figure 5.11 have been conducted on the model with asymmetric delays. For the symmetric case, the probability values are much smaller. As an example, for Device $(0,0)$, it decreases from 0.388 to 0.085. The above results have been obtained in less than 4 seconds.
For **weighted fair queuing mixed with priorities**, we have selected three different configurations of the weights in order to give a hint of the different behavior we obtain. Since we want to study bounded accuracy, it is legitimate to always give a higher priority to PTP packets. As a consequence, all the studied configurations give a higher weight to PTP packets. We have also selected two distinct window size. Indeed, the size of the window can have some importance when it is either close to the size of the packets or much bigger. All the experiments have thus been done both for a window of $1.5ms$ and for a window of $100ms$.

The first (resp. second, third) configuration gives a weight 3 (resp. 5, 8) to PTP packets, while giving weight 2 to audio and events packets, and weight 1 to video packets. It will be referred as 3:2:2:1 (resp. 5:2:2:1, 8:2:2:1). For this configurations, the results of bounded accuracy for a bound $\Delta = 50\mu s$ are given in Figure 5.12a for the window of $1.5ms$ and in Figure 5.12b for the window of $100ms$. The results for similar experiments are given in Figures 5.12c and 5.12d for configuration 5:2:2:1 and in Figures 5.12e and 5.12f for configuration

Precision	10^{-1}		10^{-2}		10^{-3}	
Confidence	10^{-5}	10^{-10}	10^{-5}	10^{-10}	10^{-5}	10^{-10}
PESTIMATION	4883 $17s$	9488 $34s$	488243 $29m$	948760 $56m$	48824291 $> 3h$	94875993 $> 3h$
SSP	1604 $10s$	3579 $22s$	161986 $13m$	368633 $36m$	16949867 $> 3h$	32792577 $> 3h$
SPRT	316 $2s$	1176 $7s$	12211 $53s$	22870 $1m38s$	148264 $11m$	311368 $31m$

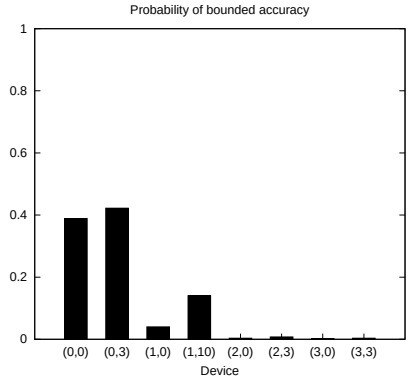
Table 5.2: Number of simulations / Amount of time required for PESTIMATION, SSP and SPRT.

8:2:2:1.

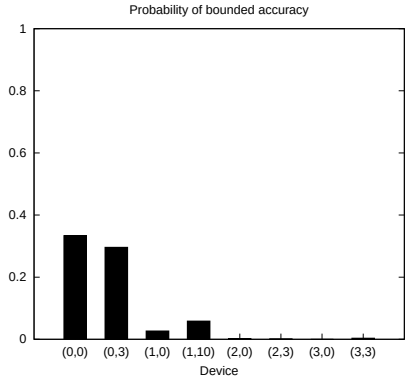
From these results, we observe that the different weights given to PTP packets do not seem to influence a lot the bounded accuracy. The reason for this is that weighted fair queuing allocates to each types of packet a bandwidth that is function of their weight. Since PTP packets are very small, they do not need a very high bandwidth to be almost always transmitted without delay. As a consequence, increasing the size of the bandwidth allocated to them has almost no impact on bounded accuracy. However, we also observe that the size of the window has some impact on bounded accuracy. Indeed, the priorities that are mixed with weighted fair queuing ensure that, while respecting the bandwidth allocation, PTP packets are always transmitted first. However, once the "quota" of PTP packets has been reached, the other types of packets are all transmitted – creating a gap in the transmission of PTP packets. The largest the window, the largest this gap will be. This gap will greatly influence synchronization and thus bounded accuracy.

Experiment 2. As a second experiment, we have used SPRT and SSP to validate the probability value found by PESTIMATION with a higher degree of confidence. Table 5.2 compares the computation times of SPRT, SSP and Estimation. The results presented are computed for Device $(0, 0)$ with fixed priorities, but they are representative of the results for all the experiments presented here, both for **fixed priorities** and **WFQ mixed with priorities**. We observe that SPRT is faster than SSP and PESTIMATION.

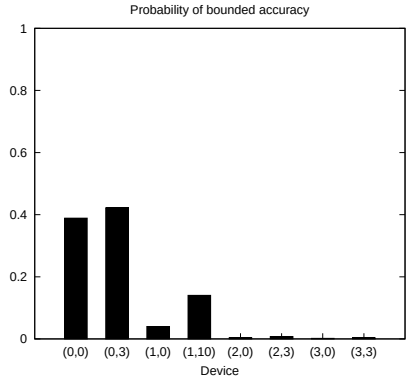
Experiment 3. The next step was to estimate the best bound. For doing so, for each device we have repeated the previous experiments for values of Δ between $10\mu s$ and $120\mu s$ for fixed priorities and between $10\mu s$ and $140\mu s$ for weighted fair queuing mixed with priorities.
For **fixed priorities**, Figure 5.13 gives the results of the probability of satisfying the bounded accuracy property as a function of the bound Δ for the asymmetric version of PTP. The figure shows that the smallest bound which ensure synchronization for any device is $105\mu s$ (for Device $(3, 0)$). However, devices $(0, 3)$ and $(3, 3)$ already satisfy the property with probability 1 for $\Delta = 60\mu s$.
For **WFQ mixed with priorities**, the results are presented in Figure 5.14. In this case, we observe that the smallest bound ensuring synchronization for any device is $125\mu s$ regardless of the configuration and window size. It is the exact bound for Device $(3, 3)$. Still, some devices



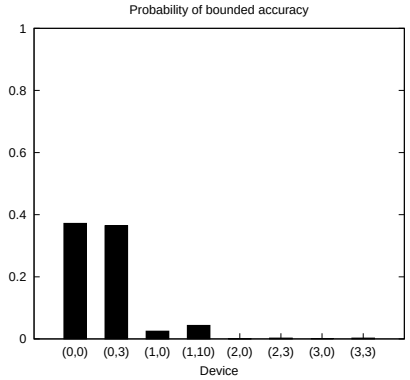
(a) Probability of satisfying bounded accuracy for configuration 3:2:2:1 and a window of $1.5ms$.



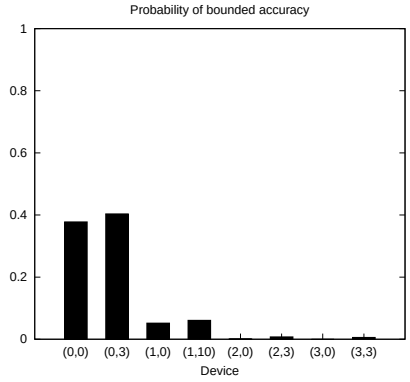
(b) Probability of satisfying bounded accuracy for configuration 3:2:2:1 and a window of $100ms$.



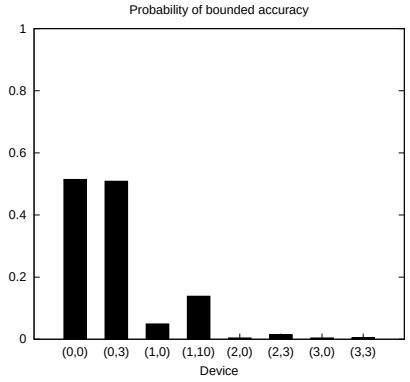
(c) Probability of satisfying bounded accuracy for configuration 5:2:2:1 and a window of $1.5ms$.



(d) Probability of satisfying bounded accuracy for configuration 5:2:2:1 and a window of $100ms$.



(e) Probability of satisfying bounded accuracy for configuration 8:2:2:1 and a window of $1.5ms$.



(f) Probability of satisfying bounded accuracy for configuration 8:2:2:1 and a window of $100ms$.

Figure 5.12: Probability of satisfying the bounded accuracy property for a bound $\Delta = 50\mu s$ and weighted fair queuing mixed with priorities.

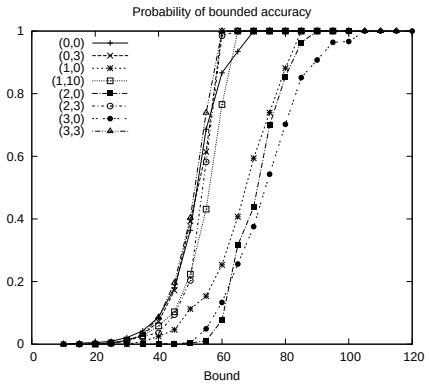


Figure 5.13: Probability of satisfying the bounded accuracy property as a function of the bound Δ for the asymmetric version of PTP.

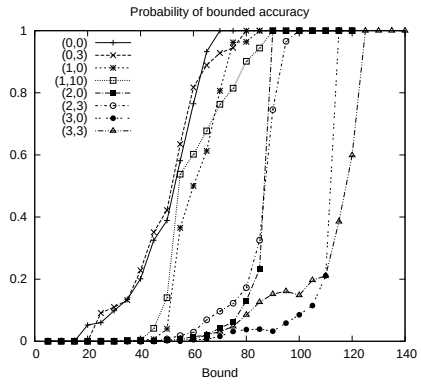
Precision	10^{-1}		10^{-2}		10^{-3}	
Confidence	10^{-5}	10^{-10}	10^{-5}	10^{-10}	10^{-5}	10^{-10}
SSP / SPRT	110 $1s$	219 $1s$	1146 $6s$	2292 $13s$	11508 $51s$	23015 $1m44s$

Table 5.3: Number of simulations / Amount of time required for PESTIMATION and SSP.

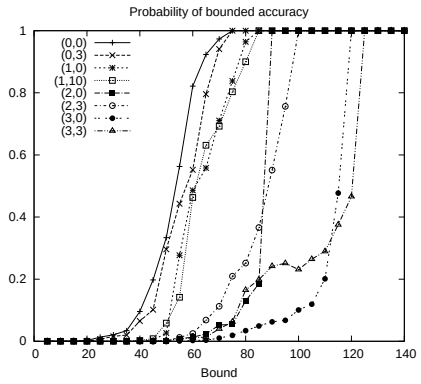
(Device $(0,0)$ and Device $(0,3)$ for instance) already satisfy the property with probability 1 for $\Delta = 65\mu s$.

Experiment 4. Table 5.3 shows, for Device (0,0) and fixed priorities, a comparison of the time and number of simulations required for PESTIMATION and SSP with the same degree of confidence. Once again, these results are representative of the results obtained for all devices and scheduling policies.

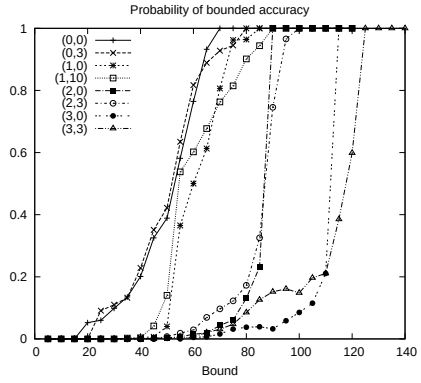
Experiment 5. The above experiments have been conducted assuming simulations of 1000 BIP interactions and 66 rounds of the PTP protocol. Since each round of the PTP takes two minutes, this also corresponds to 132 minutes of the system's life time. We now check whether the results remain the sames if we lengthen the simulations and hence system's life time. Figures 5.15a and 5.15b show, for Devices $(0,0)$ and $(3,0)$ respectively, the probability of synchronization for various values of Δ and various length of simulations (1000, 4000, 8000 and 10000 (660 minutes of system's life time) steps) for **fixed priorities**. For **WFQ mixed with priorities**, the same results are presented in Figures 5.15c and 5.15d in the case of configuration 8:2:2:1. These results are representative of all the other configurations. We used PESTIMATION with a precision and a confidence of 0.1. The best bounds do not change. However, the longest the simulations are, the more the probability tends to be either 0 or 1 depending on the bound.



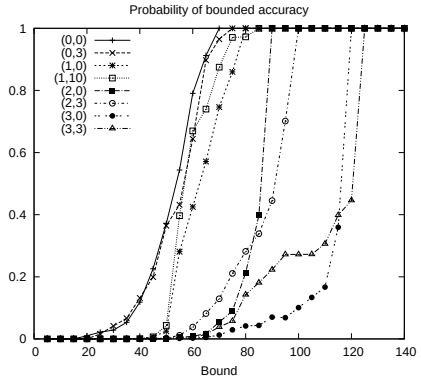
(a) Probability of satisfying bounded accuracy as a function of the bound for configuration 3:2:2:1 and a window of $1.5ms$.



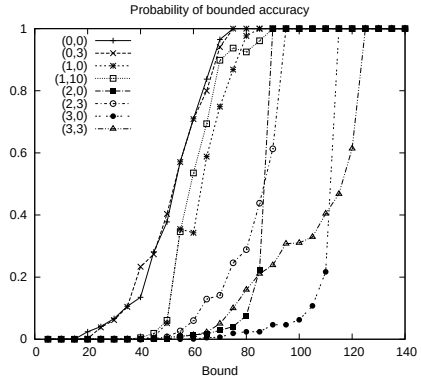
(b) Probability of satisfying bounded accuracy as a function of the bound for configuration 3:2:2:1 and a window of $100ms$.



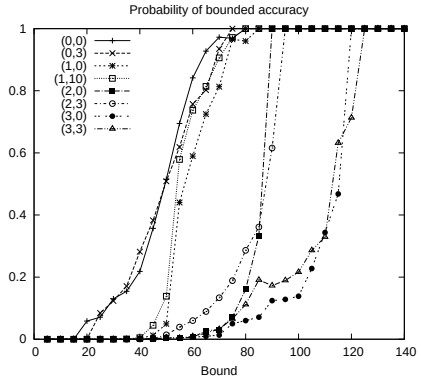
(c) Probability of satisfying bounded accuracy as a function of the bound for configuration 5:2:2:1 and a window of $1.5ms$.



(d) Probability of satisfying bounded accuracy as a function of the bound for configuration 5:2:2:1 and a window of $100ms$.

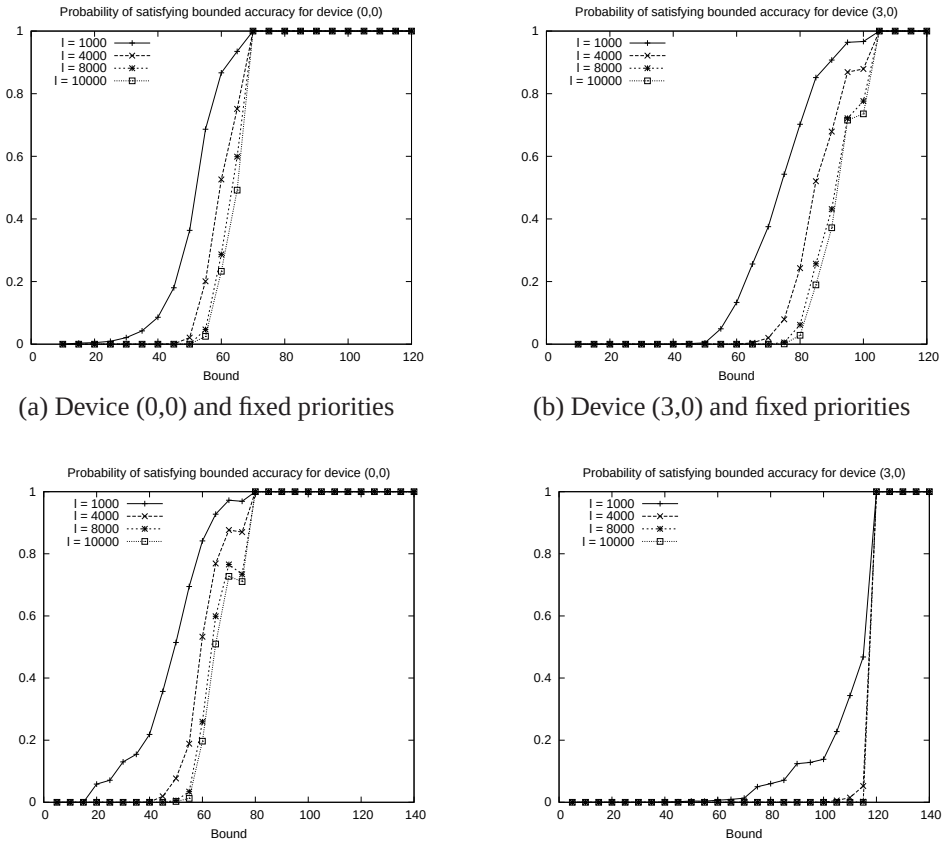


(e) Probability of satisfying bounded accuracy as a function of the bound for configuration 8:2:2:1 and a window of $1.5ms$.



(f) Probability of satisfying bounded accuracy as a function of the bound for configuration 8:2:2:1 and a window of $100ms$.

Figure 5.14: Probability of satisfying bounded accuracy as a function of the bound for weighted fair queuing mixed with priorities.



(a) Device (0,0) and fixed priorities

(b) Device (3,0) and fixed priorities

(c) Device (0,0) and WFQ 8:2:2:1

(d) Device (3,0) and WFQ 8:2:2:1

Figure 5.15: Evolution of the probability of satisfying the bounded accuracy property with the length of the simulations.

5.6.2 Property 2: Average failure

In the previous experiments, we have computed the best bound to guarantee the bounded accuracy property. It might be the case that the bound is too high regarding the user's requirements. In such case, using the above results, we can already report on the probability for synchronization to occur properly for smaller values of the bound. We now give a finer answer by quantifying the average number of failures in synchronization that occur *per simulation* when working with smaller bounds. For a given simulation, the *proportion of failures* is obtained by dividing the number of failures by the number of rounds of PTP. We will now estimate, for a simulation of 1000 steps (66 rounds of the PTP), the average value for this proportion.

Experiment 1. As a first experiment, we have measured (for each device) this proportion on 1199 simulations with a synchronization bound of $\Delta = 50\mu s$. Each of these measures takes about 6 seconds.
For **fixed priorities**, as an example, we obtain average proportions of 0.036 and 0.014 for Device $(0, 0)$ using the symmetric and asymmetric versions of PTP respectively. As a comparison, we obtain average proportions of 0.964 and 0.075 for Device $(3, 0)$. The average proportion of failures with the bound $\Delta = 50\mu s$ and the asymmetric version of PTP is given in Figure 5.16.
For **WFQ mixed with priorities**, the results are presented in Figure 5.17 for all the studied

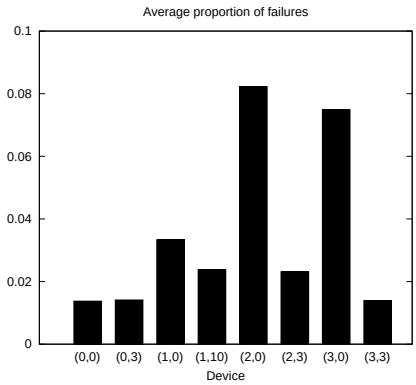


Figure 5.16: Average proportion of failures for a bound $\Delta = 50\mu s$ and the asymmetric version of PTP.

configurations. Once again, we observe that the configuration of the weights does not have a strong influence on the average proportion of failures. However, the size of the window does not influence the average proportion of failures either.

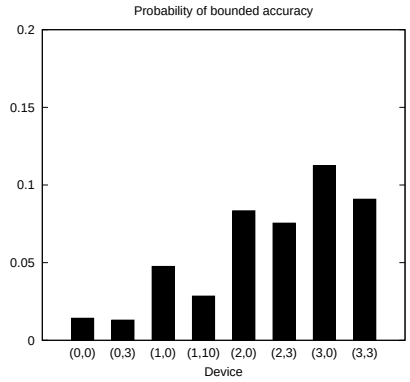
Experiment 2. The first experiment was then generalized to other values of the bound. Figure 5.18 gives the average proportion of failures as a function of the bound for **fixed priorities**. Figure 5.19 presents the results for the three configurations and two window sizes for **WFQ mixed with priorities**.

The above experiment gives, for several value of Δ and each device, the average proportion of failures with respect to 1199 simulations. We have also used PESTIMATION with confidence of 0.1 and precision of 0.1 to verify that this value remains the same whatever the number of simulations is. The result was then validated using SSP with precision of 10^{-3} and confidence of 10^{-10}. Each experiment took approximately two minutes.

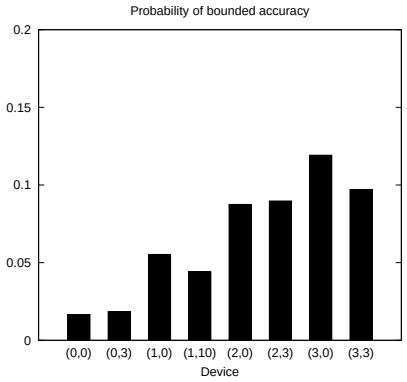
Experiment 3. Finally, we have conducted experiments to check whether the results still stand for longer simulations. Figures 5.20a and 5.20b present the results for **fixed priorities**, for Device (0,0) and Device (3,0) respectively. Figures 5.20c and 5.20d present the results for Device (0,0) and Device (3,0) using **WFQ mixed with priorities**, with the configuration 8:2:2:1 and a window of $100\mu s$. Observe that the average proportion of failures never changes with the length of the simulation, which confirms that using simulations of length 1000 is fully representative of the system.

5.6.3 Clock Drift

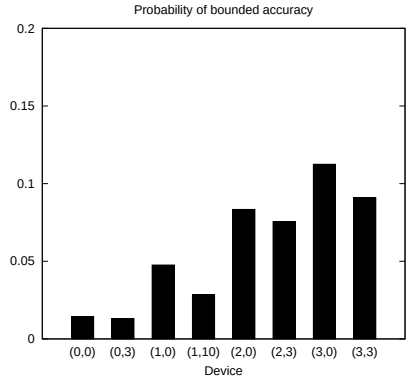
We have considered a modified version of the stochastic PTP model with drifting clocks. Drift is used to model the fact that, due to the influence of the hardware, clocks of the master and the device may not progress as the same rate. In our model, drift is incorporated as follows: each time the clock of the server is increased by 1 time unit, the clock of the device is increased by $1 + \varepsilon$ time units, with $\varepsilon \in [-10^{-3}, 10^{-3}]$. Using this modified model, we have re-done the experiments of the previous sections and observed that the result remains almost the same. This is not surprising as the value of the drift significantly smaller than the communication jitter, and therefore it has less influence of the synchronization. A drift of 1 time unit has a much higher



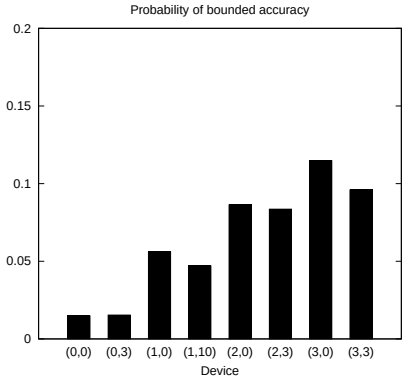
(a) Average proportion of failures for configuration 3:2:2:1 and a window of $1.5ms$.



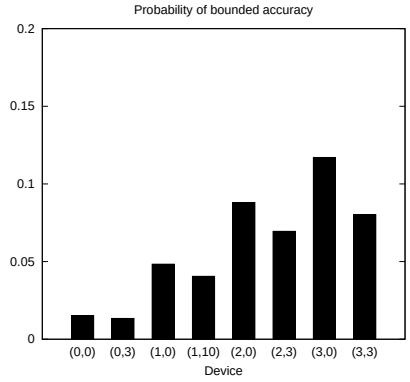
(b) Average proportion of failures for configuration 3:2:2:1 and a window of $100ms$.



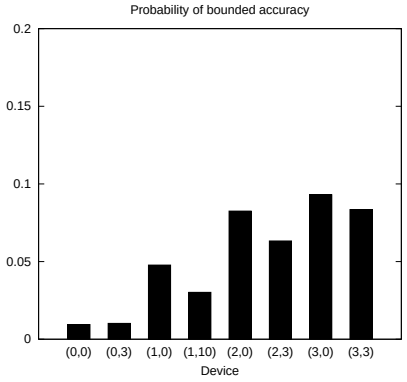
(c) Average proportion of failures for configuration 5:2:2:1 and a window of $1.5ms$.



(d) Average proportion of failures for configuration 5:2:2:1 and a window of $100ms$.



(e) Average proportion of failures for configuration 8:2:2:1 and a window of $1.5ms$.



(f) Average proportion of failures for configuration 8:2:2:1 and a window of $100ms$.

Figure 5.17: Average proportion of failures for a bound $\Delta = 50\mu s$ and weighted fair queuing mixed with priorities.

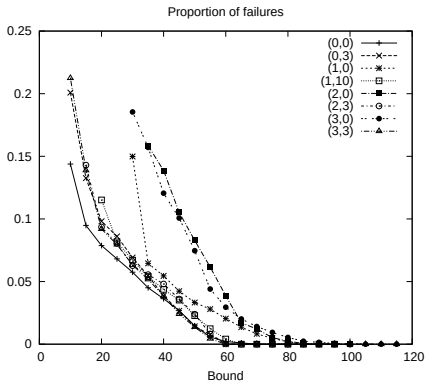


Figure 5.18: Average proportion of failures as a function of the bound Δ for the asymmetric version of PTP.

impact on the probability. As an example, for Device $(0, 0)$, it goes from a probability of $0,387$ to a probability of $0,007$ in the case of fixed priorities. It is worth mentioning that exhaustive verification of a model with drifting clocks is not an easy task as it requires to deal with complex differential equations. When reasoning on one execution at a time, this problem is avoided.

5.7 Another case study: the AFDX Network

In this section, we briefly introduce another application of the methodology presented in this chapter. A full description is available in [14].

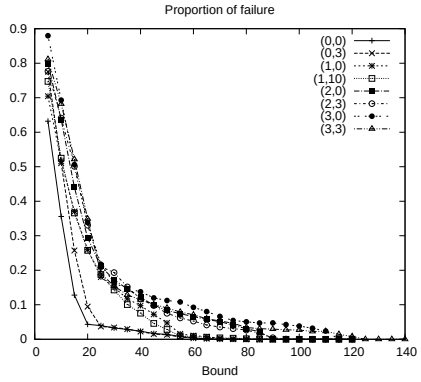
As we have already seen, the evolution of avionics embedded systems and the number of integrated functions in civilian aircrafts implied a huge increase in the quantity of data exchanged and thus in the number of connections between functions. The Aircraft Data Networks used until now had either point to point connections, which incurred a high cost in aircraft production as well as increase of weight, or mono transmitter buses with very low performances (100Kbits/s).

The HCS architecture presented in the previous sections is a solution to this problem. However, HCS also has drawbacks. As an example, there is a strong need for synchronization between the devices. As we have seen, studying synchronization in the HCS architecture is not easy. A different solution to this problem would be to use the *Avionics Full Duplex Switched Ethernet (AFDX)* [1]. Because of the property they guarantee – reliability and determinism, AFDX networks offer synchronization for free.

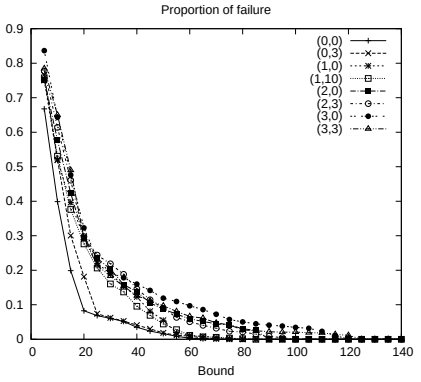
In AFDX reliability is achieved with redundancy while determinism with the definition of Virtual Links (VL), which put constraints on the allowed traffic. A network is deterministic if we can guarantee an upper bound for the time a message needs to be delivered to its destination. For AFDX such upper bounds can be provided with analytical methods [47]. The bounds obtained are over approximations of the worst case and the analysis can only be performed on very abstract models [33]. There is thus the need for new methods that will guarantee more realistic upper bounds on more realistic models.

In [14], we have proposed such a method. More precisely, our contributions are the following.

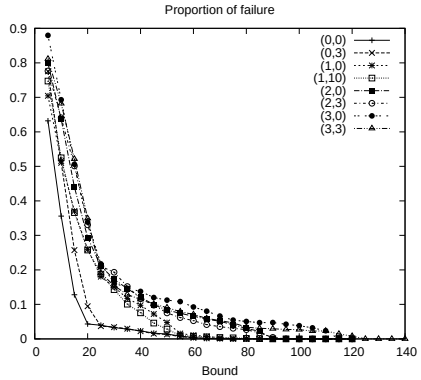
1. **Model of the network.** We propose a BIP model for AFDX architecture. To the best



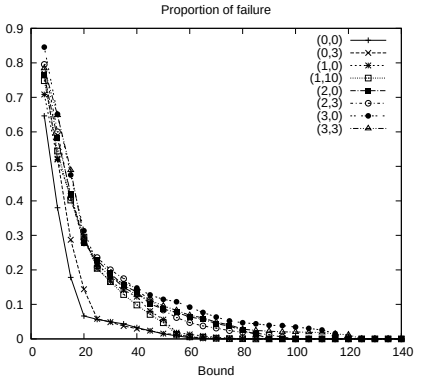
(a) Average proportion of failures as a function of the bound for configuration 3:2:2:1 and a window of $1.5ms$.



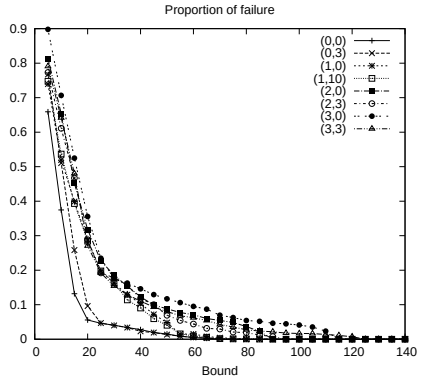
(b) Average proportion of failures as a function of the bound for configuration 3:2:2:1 and a window of $100ms$.



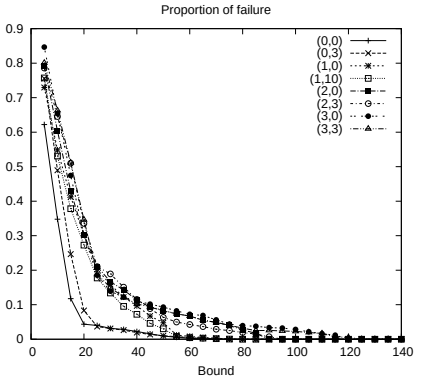
(c) Average proportion of failures as a function of the bound for configuration 5:2:2:1 and a window of $1.5ms$.



(d) Average proportion of failures as a function of the bound for configuration 5:2:2:1 and a window of $100ms$.

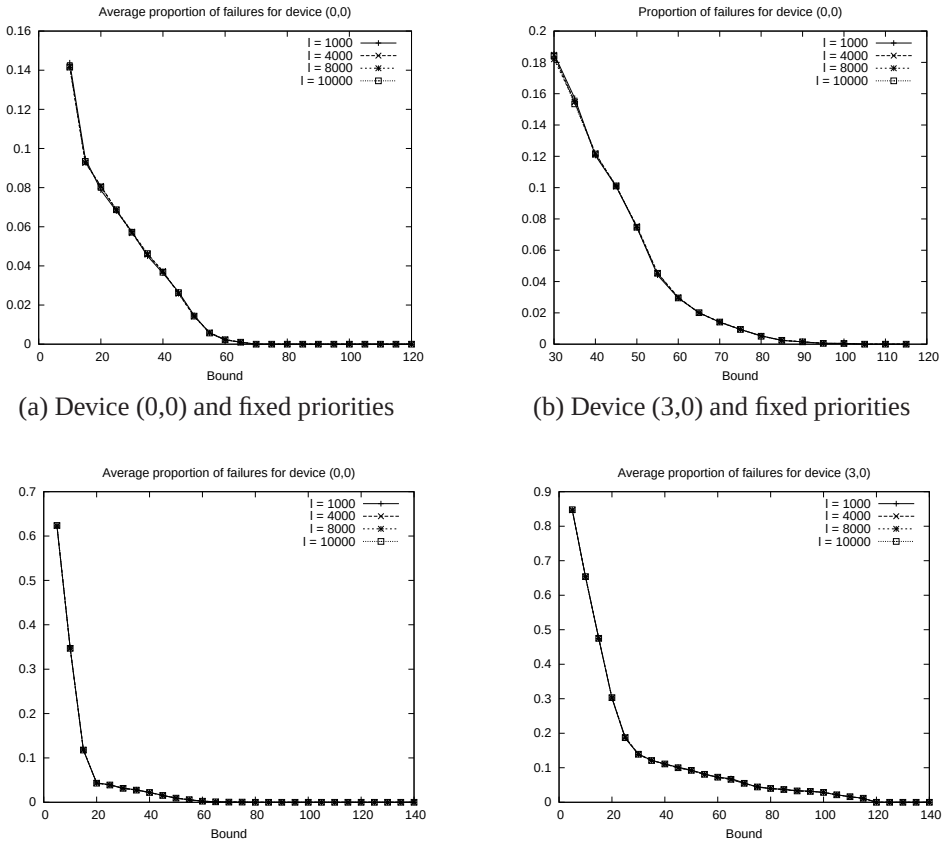


(e) Average proportion of failures as a function of the bound for configuration 8:2:2:1 and a window of $1.5ms$.



(f) Average proportion of failures as a function of the bound for configuration 8:2:2:1 and a window of $100ms$.

Figure 5.19: Average proportion of failures for a bound $\Delta = 50\mu s$ and weighted fair queuing mixed with priorities.



(a) Device (0,0) and fixed priorities

(b) Device (3,0) and fixed priorities

(c) Device (0,0) and WFQ 8:2:2:1

(d) Device (3,0) and WFQ 8:2:2:1

Figure 5.20: Evolution of the average proportion of failures with the length of the simulations.

of our knowledge, this is the first complete, fully operational and timing accurate, model of AFDX developed using a formal framework. One shall observe that our construction can be adapted to more complex network topologies.

2. **Verification.** We then examine the *latency requirements* property in AFDX, i.e., we check that the total delivery time for packets on virtual links is smaller than some predefined values. The difficulty is that our model of AFDX is constituted of many BIP components – this is needed to obtain an accurate model of the network. Combining these components leads to a system that is too big (in terms of states) to be analyzed by classical verification techniques such as model checking. In order to overcome the problem, we suggest to abstract some of these components with probability distributions, hence producing another BIP model of the network that is a stochastic abstraction of the original one. We then apply statistical model checking to estimate a value of the bound for which the requirement is satisfied with probability 1. This is an important feature as correct upper bounds are mandatory for certification. We also show that one can use our approach to compute the probability that the latency requirement is satisfied for a given value of the bound. This latest feature is of interest to adapt/reconfigure the network for better average performances.

We are not the first to propose the use of formal methods to analyze AFDX networks. Other

models are either performance models built within network simulators or timed automata models, restricted to few functionalities or describing very simple network configuration. The work of [7] focused on redundancy management and identified several issues occuring in the presence of particular network faults. Alternatively, [32, 33, 118] deal with computing bounds for end-to-end delays in AFDX networks. The papers [32, 33] report experiments using three analysis methods: network calculus, stochastic simulation using QNAP2 and timed model-checking using Uppaal. The results confirm the well-established knowledge about these methods. Network calculus [47] provides pessimistic, unreachable bounds. Network stochastic simulation provide reachable bounds, however, these bounds hardly depend on the simulation scenario considered and can be too optimistic. Timed model-checking [5] provides exact bounds, however, it suffers from state explosion due to model complexity, and hence, cannot scale to realistic networks. Finally, the work in [118] provides a method for compositional analysis of end-to-end delays. It is shown that, to measure delays for a given virtual link, it is enough to consider only the traffic generated by the virtual links influencing, i.e., which share paths within the network. This observation allows to *slice* the network and therefore to reduce the complexity of any forthcoming analysis. However (1) our model is more detailed and easier to extend/modify due to the use of the component-based design approach and (2) we are capable to retrieve stochastic information on the network.

This second experiments suggests that our stochastic abstraction method can be automatized and hence further developed.

5.8 Achievements and Future work

The contributions of this chapter are twofolds: (1) the modeling of the HCS using the BIP framework, and (2) a verification method and experimental results obtained on this case study.

We have proposed a complete method for modeling and abstracting an industrial case study using the BIP toolset [15]. Due to the size of the HCS, we propose a stochastic abstraction of the global model in order to verify properties using statistical model checking. Thanks to this approach, we have been able to reduce the size of the model and to derive precise bounds that guarantee proper synchronization for all the devices of the system. This technique is fully general and can be applied to other case studies.As an example, we have showed in Section 5.6.3, that the bounds we obtain for synchronization still hold on a modified model with drifting clocks – this property could not easily be verified by classical verification techniques. In Section 5.7, we have also applied the method to an AFDX network [14].

We now illustrate a key feature of our approach. We consider the HCS example introduced in Section 5.4. Assume that we are interested in verifying properties of the subsystem composed of (1) the server, and (2) one arbitrary device. Our approach consists in abstracting away the rest of the system. Our technique proceeds by first simulating the entire heterogeneous system in order to compute the stochastic abstraction. This is done by an on-the-fly generation of executions/simulations of the system resulting from the composition of the many components that participate to the design. When performing this computation, one has to resolve the non-determinism that arises from the composition of the components. This is generally done by random choices using uniform distributions among enabled choices. The key observation relevant to statistics is that, the mixing of those many random effects result in smooth distributions characterizing the random behaviors of the subsystems ((1),(2)) of interest. Furthermore,

the particular form for the random choices performed during the simulation does not really influence the resulting stochastic behavior of the stochastic abstraction — this relies on arguments of convergence toward so-called *stable distributions* [140]. Our approach is thus clearly different from those who would have artificially characterized the stochastic behavior of the subsystems.

Several directions can be considered as future work. First, we computed many simulations in order to learn the probability distributions of the delays for PTP packets. This is necessary if we want to produce a stochastic abstraction whose distributions are accurate estimations of the real distributions. In the case of systems of a higher order complexity, we cannot always assume that it will be possible to generate as many simulations as needed. However, there are techniques from the statistics area (for example kernel methods [125, 88, 120], wavelets [88] or bootstraping [58, 59]) that could enable us to reason on a smaller number of simulations, and still produce a reasonable approximation of the stochastic abstraction. Second, our experiments highly depend on the ESTIMATION algorithm, which is potentially computationally expensive. We could adapt Bayesian statistical model checking [85] in order to improve the efficiency of ESTIMATION. Preliminary results are given in [141]. Third, it would be of interest to integrate statistical model checking and BIP in a tool that would be used to design and analyse probabilistic systems in a compositional way. This is not an easy task because of the requirements of statistical model checking that the systems must be fully probabilistic, which may be a big obstacle to compositionality. Moreover, it would be of interest to extend statistical model checking in order to verify more complex properties like Availability, presented in Chapter 4 or unbounded properties [92, 91, 139, 110]. Finally, such a tool could be used, for example, to verify satisfaction of probabilistic contracts.

Chapter 6

Conclusion

In this thesis, we have presented new results for the design and verification of systems that mix both non-deterministic and stochastic aspects.

Our first main contribution was to lift the interface theories to the stochastic setting. We started with new results on the expressiveness and complexity of three refinement preorders for Interval Markov Chains. Those results are of clear importance as existing works on IMCs often use one of these preorders to compare specifications [86, 89, 61]. We also proposed a constructive solution to the common implementation problem, i.e. the problem of deciding whether there exists an implementation satisfying all the specifications in a given set. It is worth mentioning that these results are robust and still hold on simple variations of IMCs. As an example, one can use sets of sets of propositions to label the states instead of sets of propositions. We can also use an initial distribution instead of an initial state. However, even though IMCs are an attractive formalism, they are not powerful enough to capture all the good requirement for an interface theory (composition, conjunction, disjunction). This motivates the development of Constraint Markov Chains, the first complete compositional specification theory for stochastic systems. CMCs are an extension of IMCs, which allows complex constraints on the transition probabilities instead of simple intervals. We have provided definitions for satisfaction and refinement, which extend those proposed for IMCs. In addition, we have designed algorithms for consistency checking and structural and logical composition. Moreover, we have provided a comparison between the structural and logical operators. More precisely, we have shown that conjunction acts as an abstraction for composition. We have also observed that the conjunction or disjunction of two linear constraints remains linear, but that composition may introduce polynomial constraints. From an implementation point of view it may thus be more efficient to work with linear constraints only. For doing so, one can simply approximate composition with conjunction. Finally, we provide reductions from probabilistic automata to CMCs, showing that our formalism is fully general. Despite this generality, all operators and relations are computable.

There are various research directions to continue this work. Some of them have already been presented in Section 3.9. The most promising directions are twofolds: We think it is important implement and evaluate the algorithms proposed in these two chapters. Extending CMCs to the continuous-time setting seems a natural next step. Indeed, Continuous-time Markov Chains (CTMCs) are one of the most important semantical models for real-time probabilistic systems. CTMCs have been widely used in performance and dependability analysis, their applications

ranging from Markovian queuing networks to calculi for systems biology [75, 94, 76, 95].

Our second main contribution is a new theory for (probabilistic) contracts, which extends the work of [21]. First, we have proposed new notions for satisfaction of both probabilistic and non-probabilistic contracts: reliability and availability. Reliability is a classical notion as it gives a measure of the sets of runs of a system that satisfies a given property. In contrast, availability is a new notion, measuring the amount of time during which a system satisfies a given property. Both notions play an important role in the design of mission-critical systems. Second, the theory has been adapted in order to treat stochastic aspects. In this way, the probabilistic assume guarantee contracts theory allows considering systems evolving in a stochastic environment. Finally, we have proposed effective symbolic representations of contracts and systems. These representations are based on automata representing possibly infinite sets of runs. We have showed that if assumptions and guarantees are represented with Büchi automata, checking reliability satisfaction and refinement can be done with classical techniques.

In addition to what has already been discussed in Section 4.6, we believe that the main direction for future work is on the implementation. The non-probabilistic setting could be implemented in the SPIN toolset [127], while the LIQUOR toolset [35] seems more appropriate for the probabilistic approach.

Finally, our most promising contribution may be our study of the EADS HCS. We propose a new simulation-based technique for verifying applications running within the HCS, which is the cabin communication system of an airplane. Our technique starts by performing simulations of the system in order to learn the context in where the application is used. Then, it creates a stochastic abstraction for the application, which takes the context information into account. This smaller model can be verified using efficient techniques such as statistical model checking. This technique has been applied to verify the clock synchronization protocol i.e., the application used to synchronize the clocks of all computing devices within the system.

The important lessons we learnt from this experiment are that (1) probabilities can be used as a concise representation of the context in where a given subsystem is running, and (2) simulations combined with statistics make verification and validation faster and more general. We thus believe that the concept of stochastic abstraction should be further formalized, automatized, and developed. We are also convinced that statistical model checking algorithms can be made more efficient by taking the methodology used to design the system into account. Studying stochastic abstraction and improving statistical model checking algorithms are the two main directions for future research.

The *stochastic abstraction* shall be obtained by simulating the entire design (system level model) in order to learn the environment in where the subsystem under consideration is running. Generating simulations of a complex design may take time. We thus suggest to use techniques from the statistical area to better exploit the simulations in generating an accurate estimate of the distribution. Stochastic abstraction may also be combined with classical abstraction techniques, especially when memory has to be considered in the design.

We also suggest to improve the efficiency of statistical model checking algorithms in two ways. First, as the system is assumed to be "well-designed", one can postulate that the property under verification should rarely be falsified. This means that we are trying to compute probabilities of violation that should be very close to 0. Statistical model checking algorithms should address this issue in an efficient manner. This is actually not the case. Also, due to

her engineering knowledge about the system, the designer may guess some prior knowledge regarding the probability for the system to violate the property. This information could be used to improve the efficiency of statistical model checking.

One of the recurrent difficulties with formal verification techniques is the development of specific tools and the acceptance of the underlying technology, by engineers, as part of their design process. To ensure that our approach is accepted by industrials, we will collaborate with an industrial partner who develops tools for designing heterogeneous systems. At the very beginning of our study, the experiments will be conducted with academic tools such as the BIP toolset [15, 26] used for EADS and AFDX in Chapter 5. According to EADS designers, the language of BIP is expressive enough to "mimic" the concrete implementation of the HCS. However, the experiment could not have been conducted without the help of EADS designers who validated our mathematical model of the system. In order to cope with other case studies, we will have to integrate our technology in the tool chain of industrials. Such an integration creates new difficulties. As an example, it requires to be able to jointly simulate models of different parts of the system, possibly expressed using different formalisms. Fortunately, corresponding so-called "co-simulation" (also called "hosted simulation") technologies (see [128] for an illustration) have been recently developed by tool vendors (such our industrial partner) to cope with this problem. We plan to integrate this technology and extend it to a more general context. Another major difficulty will be to provide feedback to the user in case her requirements are not satisfied.

Bibliography

[1] *ARINC 664, Aircraft Data Network, Part 7: Avionics Full Duplex Switched Ethernet (AFDX) Network.* , 2005.

[2] I. I. 61588. *Precision clock synchronization protocol for networked measurement and control systems*, 2004.

[3] M. Abadi and L. Lamport. Composing specifications. *ACM Trans. Program. Lang. Syst.*, 15(1):73–132, 1993.

[4] B. T. Adler, L. de Alfaro, L. D. da Silva, M. Faella, A. Legay, V. Raman, and P. Roy. Ticc: A tool for interface compatibility and composition. In *Proc. 18th International Conference on Computer Aided Verification (CAV), Seattle, WA, USA,*, volume 4144 of *Lecture Notes in Computer Science*, pages 59–62. Springer, 2006.

[5] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Comput. Sci.*, 126(2):183–235, 1994.

[6] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[7] M. Anand, S. Dajani-Brown, S. Vestal, and I. Lee. Formal modeling and analysis of afdx frame management design. In *Proc. 9th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC), Gyeongju, Korea*, pages 393–399. IEEE, 2006.

[8] S. Andova. Process algebra with probabilistic choice. In *Proc. 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems (ARTS), Bamberg, Germany*, volume 1601 of *Lecture Notes in Computer Science*, pages 111–129. Springer, 1999.

[9] A. Antonik, M. Huth, K. G. Larsen, U. Nyman, and A. Wąsowski. 20 years of modal and mixed specifications. *beatcs*, 95, June 2008. Available at http://processalgebra.blogspot.com/2008/05/concurrency-column-for-beatcs-june-2008.html.

[10] A. Antonik, M. Huth, K. G. Larsen, U. Nyman, and A. Wąsowski. Modal and mixed specifications: key decision problems and their complexities. *Mathematical Structures in Computer Science*, 20(01):75–103, 2010.

[11] A. Antonik, M. Huth, K. G. Larsen, U. Nyman, and A. Wąsowski. Complexity of decision problems for mixed and modal specifications. In *Proc. 11th International Conference on Foundations of Software Science and Computational Structures (FOSSACS), Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary*, volume 4962 of *Lecture Notes in Computer Science*. Springer, 2008.

[12] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

[13] A. Basu, S. Bensalem, M. Bozga, B. Caillaud, B. Delahaye, and A. Legay. Statistical abstraction and model-checking of large heterogeneous systems. In *Proc. 5th IFIP International Conference on Formal Techniques for Distributed Systems (FORTE 2010), Amsterdam, The Netherlands*, volume 6117 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2010.

[14] A. Basu, S. Bensalem, M. Bozga, B. Delahaye, A. Legay, and E. Siffakis. Verification of an afdx infrastructure using simulations and probabilities, 2010.

[15] A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-time Systems in BIP. In *Proc. 4th IEEE International Conference on Software Engineering and Formal Methods (SEFM), Pune, India*, pages 3–12, 2006.

[16] S. Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of the ACM*, 46(4):537–555, July 1999.

[17] N. Benes, J. Kretínský, K. G. Larsen, and J. Srba. Checking thorough refinement on modal transition systems is exptime-complete. In *Proc. 6th International Colloquium on Theoretical Aspects of Computing (ICTAC), Kuala Lumpur, Malaysia*, volume 5684 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2009.

[18] N. Benes, J. Kretínský, K. G. Larsen, and J. Srba. On determinism in modal transition systems. *Theor. Comput. Sci.*, 410(41):4026–4043, 2009.

[19] S. Bensalem, M. Bozga, T. Nguyen, and J. Sifakis. D-finder: A tool for compositional deadlock detection and verification. In *Proc. 21st International Conference on Computer Aided Verification (CAV), Grenoble, France*, volume 5643 of *Lecture Notes in Computer Science*, pages 614–619. Springer, 2009.

[20] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis. Multiple viewpoint contract-based specification and design. In *Proc. 6th International Symposium on Formal Methods for Components and Objects (FMCO), Amsterdam, The Netherlands*, volume 5382 of *Lecture Notes in Computer Science*, pages 200–225. Springer, October 2007.

[21] A. Benveniste, B. Caillaud, and R. Passerone. A generic model of contracts for embedded systems. *CoRR*, abs/0706.1456, 2007.

[22] N. Bertrand, A. Legay, S. Pinchinat, and J.-B. Raclet. A compositional approach on modal specifications for timed systems. In *Proc. 11th International Conference on Formal Engineering Methods (ICFEM), Rio de Janeiro, Brazil*, volume 5885 of *Lecture Notes in Computer Science*, pages 679–697. Springer, 2009.

[23] N. Bertrand, S. Pinchinat, and J.-B. Raclet. Refinement and consistency of timed modal specifications. In *Proc. 3rd International Conference on Language and Automata Theory and Applications (LATA), Tarragona, Spain*, volume 5457 of *Lecture Notes in Computer Science*, pages 152–163, Tarragona, Spain, 2009. Springer.

[24] D. P. Bertsekas and J. N. Tsitsiklis. *Introduction to Probability*. MIT press, 2008.

[25] P. Bhaduri. Synthesis of interface automata. In *Proc. 3rd International Symposium on Automated Technology for Cerification and Analysis (ATVA),Taipei, Taiwan*, volume 3707 of *Lecture Notes in Computer Science*, pages 338–353, 2005.

[26] The BIP Toolset. http://www-verimag.imag.fr/∼async/bip.php.

[27] C. W. Brown. Simple CAD construction and its applications. *Journal of Symbolic Computation*, 31(5), 2001.

[28] C. W. Brown and J. H. Davenport. The complexity of quantifier elimination and cylindrical algeraic decomposition. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC), Waterloo, Ontario, Canada*, pages 54–60, Waterloo, ON, Canada, 2007. ACM.

[29] J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift Math. Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[30] D. Bustan, S. Rubin, and M. Vardi. Verifying omega-regular properties of Markov chains. In *Proc. 16th International Conference on Computer Aided Verification (CAV), Boston, MA, USA*, volume 3114 of *Lecture Notes in Computer Science*, pages 189–201. Springer, 2004.

[31] K. Cerans, J. C. Godskesen, and K. G. Larsen. Timed modal specification - theory and tools. In *Proc. 5th International Conference on Computer Aided Verification (CAV), Elounda, Greece*, volume 697 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 1993.

[32] H. Charara and C. Fraboul. Modelling and simulation of an avionics full duplex switched ethernet. In *Proc. Advanced Industrial Conference on Telecommunications/ Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunication Workshop*. IEEE, 2005.

[33] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Methods for bounding end-to-end delays on AFDX network. In *Proc. 18th Euromicro Conference on Real-Time Systems (ECRTS), Dresden, Germany*. IEEE Computer Society, 2006.

[34] K. Chatterjee, K. Sen, and T. A. Henzinger. Model-checking omega-regular properties of interval Markov chains. In *Proc. 11th International Conference on Foundations of Software Science and Computational Structures (FOSSACS), Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary*, volume 4962 of *Lecture Notes in Computer Science*, pages 302–317. Springer, 2008.

[35] F. Ciesinski and C. Baier. Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *Proc. 3rd International Conference on the Quantitative Evaluaiton of Systems (QEST), Riverside, California, USA*, pages 131–132. IEEE Computer Society, 2006.

[36] F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems - A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 147–188. Springer, 2004.

[37] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[38] E. M. Clarke, A. Donzé, and A. Legay. Statistical model checking of mixed-analog circuits with an application to a third order delta-sigma modulator. In *Proc. 4th International Haifa Verification Conference on Hardware and Software: Verification and Testing (HVC), Haifa, Israel*, volume 5394 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2008.

[39] E. M. Clarke, A. Donzé, and A. Legay. On simulation-based probabilistic model checking of mixed-analog circuits. *Formal Methods in System Design*, 2009. to appear.

[40] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proc. Workshop on Logics of Programs, Yorktown Heights, New York, USA*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[41] E. M. Clarke, J. R. Faeder, C. J. Langmead, L. A. Harris, S. K. Jha, and A. Legay. Statistical model checking in biolab: Applications to the automated analysis of t-cell receptor signaling pathway. In *Proc. 6th International Conference on Computational Methods in Systems Biology (CMSB), Rostock, Germany*, volume 5307 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2008.

[42] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[43] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[44] J. M. Cobleigh, G. S. Avrunin, and L. A. Clarke. Breaking up is hard to do: An evaluation of automated assume-guarantee reasoning. *ACM Trans. Softw. Eng. Methodol.*, 17(2), 2008.

[45] Combest. http://www.combest.eu.com.

[46] D. R. Cox and H. D. Miller. The theory of stochastic processes. 1965.

[47] R. Cruz. A calculus for network delay. *IEEE Transactions on Information Theory*, 37(1):114–141, January 1991.

[48] D. Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. PhD thesis, Eindhoven University of Technology, July 1996.

[49] P. R. D'Argenio, B. Jeannet, H. E. Jensen, and K. G. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proc. Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM-PROBMIV), Aachen, Germany*, volume 2165 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2001.

[50] A. David, K. G. Larsen, A. Legay, U. Nyman, and A. Wąsowski. Timed i/o automata : A complete specification theory for real-time systems. In *Proc. 13th ACM International Conference on Hybrid Systems: Computation and Control (HSCC), Stockholm, Sweden*, pages 91–100. ACM, 2010.

[51] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[52] L. de Alfaro, L. D. da Silva, M. Faella, A. Legay, P. Roy, and M. Sorea. Sociable interfaces. In *Proc. 5th International Workshop on Frontiers of Combining Systems (FroCoS), Vienna, Austria*, volume 3717 of *Lecture Notes in Computer Science*, pages 81–105. Springer, 2005.

[53] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain*, volume 2988 of *Lecture Notes in Computer Science*, pages 77–92. Springer, 2004.

[54] L. de Alfaro and T. A. Henzinger. Interface automata. In *Proc. 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC / SIGSOFT FSE), Vienna, Austria*, pages 109–120. ACM Press, 2001.

[55] L. de Alfaro and T. A. Henzinger. Interface-based design. In *Engineering Theories of Software-intensive Systems*, volume 195 of *NATO Science Series: Mathematics, Physics, and Chemistry*, pages 83–104. Springer, 2005.

[56] L. de Alfaro, T. A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In *Proc. 12th International Conference on Concurrency Theory (CONCUR), Aalborg, Denmark*, volume 2154 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2001.

[57] Ecdar. http://www.cs.aau.dk/ adavid/ecdar/.

[58] B. Efron. The jacknife, the bootstrap, and other resampling plans. *Society of Industrial and Applied Mathematics*, 1982.

[59] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Hall/CRC Press Monographs on Statistics and Applied Probability, 1994.

[60] C. Eisner and D. Fisman. *A Practical Introduction to PSL*. Springer, 2006.

[61] H. Fecher, M. Leucker, and V. Wolf. *Don't Know* in probabilistic systems. In *Proc. 13th International SPIN Workshop on Model Checking Software (SPIN), Vienna, Austria*, volume 3925 of *Lecture Notes in Computer Science*, pages 71–88. Springer, 2006.

[62] S. Fenech, G. J. Pace, and G. Schneider. Automatic conflict detection on contracts. In *Proc. 6th International Colloquium on Theoretical Aspects of Computing (ICTAC), Kuala Lumpur, Malaysia*, volume 5684 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2009.

[63] S. Fenech, G. J. Pace, and G. Schneider. Clan: A tool for contract analysis and conflict discovery. In *Proc. 7th International Symposium on Automated Technology for Verification and Analysis (ATVA), Macao, China*, volume 5799 of *Lecture Notes in Computer Science*, pages 90–96. Springer, 2009.

[64] G. Feuillade and S. Pinchinat. Modal specifications for the control theory of discrete event systems. *Discrete Event Dynamic Systems*, 17(2):211–232, 2007.

[65] A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. *Electr. Notes Theor. Comput. Sci.*, 9, 1997.

[66] Y. Glouche, P. L. Guernic, J.-P. Talpin, and T. Gautier. A boolean algebra of contracts for logical assume-guarantee reasoning. *CoRR*, inria-00292870, 2009.

[67] G. Goessler and J.-B. Raclet. Modal contracts for component-based design. In *Proc. 7th IEEE International Conference on Software Engineering and Formal Methods (SEFM), Hanoi, Vietnam*, pages 295–303. IEEE Computer Society, 2009.

[68] R. Grosu and S. A. Smolka. Monte carlo model checking. In *Proc. 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK*, volume 3440 of *lncs*, pages 271–286. Springer, 2005.

[69] S. Haddad and N. Pekergin. Using stochastic comparison for efficient model checking of uncertain Markov chains. In *Proc. Sixth International Conference on the Quantitative Evaluation of Systems (QEST), Budapest, Hungary*, pages 177–186. IEEE Computer Society Press, 2009.

[70] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. Time-bounded model checking of infinite-state continuous-time Markov chains. *Fundam. Inform.*, 95(1):129–155, 2009.

[71] T. Han, J.-P. Katoen, and B. Damman. Counterexample generation in probabilistic model checking. *IEEE Trans. Software Eng.*, 35(2):241–257, 2009.

[72] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabitilies. In *IEEE Real-Time Systems Symposium*, pages 278–287, 1990.

[73] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.

[74] D. Harel, O. Kupferman, and M. Y. Vardi. On the complexity of verifying concurrent transition systems. *Inf. Comput.*, 173(2):143–161, 2002.

[75] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In C. Priami, editor, *Proc. Conference on Computational Methods in Systems Biology (CMSB)*, volume 4210 of *Lecture Notes in Bioinformatics*, pages 32–47. Springer Verlag, 2006.

[76] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 319(3):239–257, 2008.

[77] T. A. Henzinger and J. Sifakis. The embedded systems design challenge. In *Proc. 14th International Symposium on Formal Methods (FM), Hamilton, Canada*, volume 4085 of *lncs*, pages 1–15. Springer, 2006.

[78] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *Proc. 5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI), Venice, Italy*, volume 2937 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2004.

[79] H. Hermanns. *Interactive Markov Chains*. Springer, 2002.

[80] H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theor. Comput. Sci.*, 274(1-2):43–87, 2002.

[81] H. Hermanns, B. Wachter, and L. Zhang. Probabilistic CEGAR. In *Proc. 20th International Conference on Computer Aided Verification (CAV), Princeton, NJ, USA*, volume 5123 of *Lecture Notes in Computer Science*. Springer, 2008.

[82] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[83] W. Hoeffding. Probability inequalities. *Journal of the American Statistical Association*, 58:13–30, 1963.

[84] D. N. Jansen, J.-P. Katoen, M.Oldenkamp, M. Stoelinga, and I. S. Zapreev. How fast and fat is your probabilistic model checker? an experimental performance comparison. In *Proc. 3rd International Haifa Verification Conference on Hardware and Software: Verification and Testing (HVC), Haifa, Israel*, volume 4899 of *Lecture Notes in Computer Science*. Springer, 2007.

[85] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A bayesian approach to model checking biological systems. In *Proc. 7th International Conference on Computational Methods in Systems Biology (CMSB), Bologna, Italy*, volume 5688 of *Lecture Notes in Computer Science*, pages 218–234. Springer, 2009.

[86] B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *Proc. Sixth Annual IEEE Symposium on Logic in Computer Science (LICS), Amsterdam, The Netherlands*. IEEE Computer, 1991.

[87] B. Jonsson, K. G. Larsen, and W. Yi. Probabilistic extensions of process algebras. In *Handbook of Process Algebra*, pages 681–710. Elsevier, 2001.

[88] A. Juditsky, H. Hjalmärsson, A. Benveniste, B. Delyon, L. Ljung, J. Sjöberg, and Q. Zhang. Nonlinear black-box modelling in system identification: mathematical foundations. *Automatica*, 31, 1995.

[89] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for continuous-time Markov chains. In *Proc. 19th International Conference on Computer Aided Verification (CAV), Berlin, Germany*, volume 4590 of *Lecture Notes in Computer Science*, pages 311–324. Springer, 2007.

[90] J.-P. Katoen, D. Klink, and M. R. Neuhäußer. Compositional abstraction for stochastic systems. In *Proc. 7th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS), Budapest, Hungary*, volume 5813 of *Lecture Notes in Computer Science*, pages 195–211. Springer, 2009.

[91] J.-P. Katoen and I. S. Zapreev. Simulation-based CTMC model checking: An empirical evaluation. In *Proc. 6th International Conference on the Quantitative Evaluation of Systems (QEST), Budapest, Hungary*, pages 31–40. IEEE Computer Society, 2009.

[92] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker MRMC. In *Proc. 6th International Conference on the Quantitative Evaluation of Systems (QEST), Budapest, Hungary*, pages 167–176. IEEE Computer Society, 2009.

[93] L. G. Khachiyan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244(5):1093–1096, 1979.

[94] M. Kwiatkowska, G. Norman, and D. Parker. Using probabilistic model checking in systems biology. *ACM SIGMETRICS Performance Evaluation Review*, 35(4):14–21, 2008.

[95] M. Kwiatkowska, G. Norman, and D. Parker. *Symbolic Systems Biology*, chapter Probabilistic Model Checking for Systems Biology. Jones and Bartlett, 2010.

[96] M. Z. Kwiatkowska, G. Norman, and D. Parker. Game-based abstraction for Markov decision processes. In *Proc. 3rd International Conference on the Quantitative Evaluation of Systems (QEST), Riverside, California, USA*, pages 157–166. IEEE Computer Society, 2006.

[97] M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Assume-guarantee verification for probabilistic systems. In *Proc. 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS),Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus*, volume 6015 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 2010.

[98] M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Inf. Comput.*, 205(7):1027–1077, 2007.

[99] S. Laplante, R. Lassaigne, F. Magniez, S. Peyronnet, and M. de Rougemont. Probabilistic abstraction for model checking: An approach based on property testing. *ACM Trans. Comput. Log.*, 8(4), 2007.

[100] K. G. Larsen. Modal specifications. In *Proc. International Workshop on Automatic Verification Methods for Finite State Systems (AVMS), Grenoble, France*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 1989.

[101] K. G. Larsen and A. Skou. Compositional verification of probabilistic processes. In *Proc. 3rd International Conference on Concurrency Theory (CONCUR), Stony Brook, NY, USA*, volume 630 of *Lecture Notes in Computer Science*, pages 456–471. Springer, 1992.

[102] E. A. Lee and Y. Xiong. A behavioral type system and its application in Ptolemy II. *Formal Asp. Comput.*, 16(3):210–237, 2004.

[103] N. López and M. Núñez. An overview of probabilistic process algebras and their equivalences. In *Validation of Stochastic Systems - A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 89–123. Springer, 2004.

[104] N. Lynch and M. R. Tuttle. An introduction to Input/Output automata. *CWI-quarterly*, 2(3), 1989.

[105] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[106] M. R. Neuhäußer, M. Stoelinga, and J.-P. Katoen. Delayed nondeterminism in continuous-time Markov decision processes. In *Proc. 12th International Conference on Foundations of Software Science and Computational Structures, (FOSSACS), Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK*, volume 5504 of *Lecture Notes in Computer Science*, pages 364–379. Springer, 2009.

[107] G. Pace and G.Schneider. Challenges in the specification of full contracts. In *Proc. 7th International Conference on Integrated Formal Methods (IFM), Düsseldorf, Germany*, volume 5423 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 2009.

[108] A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA*, pages 46–57. IEEE, 1977.

[109] S. Quinton and S. Graf. Contract-based verification of hierarchical systems of components. In *Proc. 6th IEEE International Conference on Software Engineering and Formal Methods (SEFM),Cape Town, South Africa*, pages 377–381. IEEE Computer Society, 2008.

[110] D. E. Rabih and N. Pekergin. Statistical model checking using perfect simulation. In *Proc. 7th International Symposium on Automated Technology for Verification and Analysis (ATVA), Macao, China*, volume 5799 of *Lecture Notes in Computer Science*, pages 120–134. Springer, 2009.

[111] M. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, pages 115–125, 1959.

[112] M. O. Rabin. Probabilistic automata. *Inf. and Cont.*, 6(3):230–245, 1963.

[113] J.-B. Raclet. *Quotient de spécifications pour la réutilisation de composants*. PhD thesis, Université de Rennes I, december 2007. (In French).

[114] J.-B. Raclet. Residual for component specifications. In *Proc. 4th International Workshop on Formal Aspects of Component Software (FACS), Sophia-Antipolis, France*, 2007.

[115] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone. Modal interfaces: Unifying interface automata and modal specifications. In *Proc. 9th ACM & IEEE International conference on Embedded software (EMSOFT), Grenoble, France*, pages 87–96. ACM, 2009.

[116] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, and R. Passerone. Why are modalities good for interface theories? In *Proc. 9th International Conference on Application of Concurrency to System Design (ACSD), Augsburg, Germany*, pages 119–127. IEEE Computer Society Press, 2009.

[117] J. J. M. M. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, volume 23. American Mathematical Society, 2004.

[118] J.-L. Scharbarg and C. Fraboul. Simulation for end-to-end delays distribution on a switched ethernet. In *Proc. 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Patras, Greece*. IEEE, 2007.

[119] A. Schrijver. *Theory of linear and integer programming*. Wiley, April 1998.

[120] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley Series in Probability and Mathematical Statistics, 1992.

[121] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. In *Proc. 5th International Conference on Concurrency Theory (CONCUR), Uppsala, Sweden*, volume 836 of *Lecture Notes in Computer Science*, pages 481–496. springer, 1994.

[122] K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *Proc. 16th International Conference on Computer Aided Verification (CAV), Boston, MA, USA*, volume 3114 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2004.

[123] K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *Proc. 17th International Conference on Computer Aided Verification (CAV), Edinburgh, Scotland*, volume 3576 of *Lecture Notes in Computer Science*, pages 266–280, 2005.

[124] K. Sen, M. Viswanathan, and G. Agha. Model-checking Markov chains in the presence of uncertainties. In *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria*, volume 3920 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.

[125] J. Sjoberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmärsson, and A. Juditsky. Nonlinear black-box modelling in system identification: a unified overview. *Automatica*, 31, 1995.

[126] Speeds. http://www.speeds.eu.com.

[127] The spin tool (spin). Available at http://spinroot.com/spin/whatispin.html.

[128] S. Steinkellner, H. Andersson, I. Lind, and P. Krus. Hosted simulation for heterogeneous aircraft system development. In *Proc. 26th congress of the International Council of the Aeronautical Sciences (ICAS 2008), Anchorage, Alaska*. AIAA, 2008.

[129] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th Annual Symposium on Foundations of Computer Science (FOCS), Portland, Oregon, USA*, pages 327–338. IEEE, 1985.

[130] M. Y. Vardi. From Church and Prior to PSL, 2007. Available at http://www.cs.rice.edu/˜vardi/papers/index.html.

[131] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proc. 1st Symposium on Logic in Computer Science (LICS), Cambridge, Massachusetts, USA*, pages 332–344. IEEE Computer Society, 1986.

[132] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

[133] A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[134] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.

[135] H. Yanami and H. Anai. SyNRAC: a Maple toolbox for solving real algebraic constraints. *ACM Communications in Computer Algebra*, 41(3):112–113, September 2007.

[136] H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.

[137] H. L. S. Younes. Error control for probabilistic model checking. In *Proc. 7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI), Charleston, SC, USA*, volume 3855 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2006.

[138] H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.

[139] I. S. Zapreev. *Model Checking Markov Chains: Techniques and Tools*. PhD thesis, University of Twente, 2008.

[140] V. M. Zolotarev. One-dimensional stable distribution. *American Mathematical Society*, 1986.

[141] P. Zuliani, A. Platzer, and E. M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *Proc. 3th ACM International Conference on Hybrid Systems: Computation and Control (HSCC), Stockholm, Sweden*, pages 243–252. ACM ACM, 2010.

Abstracts

Abstract. This thesis presents new contributions in the design and verification of systems mixing both non-deterministic and stochastic aspects. Our results can be divided into three main contributions. First, we generalize interface theories to the stochastic setting. We build upon the known formalism of Interval Markov Chains to develop Constraint Markov Chains, the first fully compositional specification theory for stochastic systems. Second, we extend the notion of assume-guarantee contracts and develop a contract-based theory for stochastic systems, proposing quantitative notions for satisfaction and refinement. Finally, we propose a methodology for the verification of complex systems. This methodology is based on a stochastic abstraction of the environment where two components are working, allowing to verify the components individually. Combined with statistical model checking, this methodology is successfully applied to the verification of an industrial case study.

Résumé. Cette thèse présente des contributions originales pour la conception et la vérification de systèmes non-déterministes et stochastiques. Nos résultats sont divisés selon trois lignes directrices. Premièrement, nous généralisons la théorie des interfaces au cas stochastique, en s'appuyant sur le formalisme classique des chaînes de Markov à intervalles pour construire la première théorie de spécification compositionnelle pour systèmes stochastiques : les chaînes de Markov à contraintes. Deuxièmement, nous étendons la notion de contrats hypothèse-garantie et développons une théorie compositionnelle à base de contrats pour systèmes stochastiques, pour laquelle nous proposons des notions quantitatives de raffinement et de satisfaction. Finalement, nous proposons une méthodologie pour la vérification de systèmes complexes, basée sur une abstraction stochastique. Cette méthodologie, combinée avec le model-checking statistique, est appliquée avec succès à un cas d'étude industriel.