Efficient Verification and New Reasoning Techniques for Concurrent Constraint Programming

Luis Fernando Pino Duque

ÉCOLE POLYTECHNIQUE

Thèse de Doctorat

Spécialité Informatique

EFFICIENT VERIFICATION AND NEW REASONING TECHNIQUES FOR CONCURRENT CONSTRAINT PROGRAMMING

Présentée et soutenue publiquement par

LUIS FERNANDO PINO DUQUE

le 25 Septembre 2014

devant le jury composé de

Rapporteurs:	Ugo MONTANARI
	Stefano BISTARELLI
Directeurs de thèse:	Catuscia PALAMIDESSI
	Frank D. VALENCIA
Examinateurs:	Frank DE BOER
	Filippo BONCHI
	Stéphane GRAHAM-LENGRAND

Contents

List of Figures

List of Tables

List of Definitions

To facilitate the reading of the present document we provide a list of the definitions (and symbols) used throughout the dissertation. Below you will find the list twice, first in order of appearance and then in alphabetical order.

In order of appearance

$\mathcal{M}(\langle P, c\rangle)$	Labeled Input-Output Set	5.2.11	89
$\mathcal{M}^C(\langle P, c\rangle)$	Compact Input-Output Set	5.2.13	90
	Improved partition refinement for CCP	5.3.1	94
$\approx_f$	Weak Full Bisimilarity	6.2.1	105
$\dot{\cong}$	Behavioral Congruence	6.2.8	108

In alphabetical order:

Symbol	**Name**	**Definition**	**Page**
	Algebraic dcpo	2.1.8	19
$\sim_{wb}$	Barb equivalence	5.2.1	87
$\downarrow_c, \Downarrow_c$	Barbs	2.4.7	28
$\dot{\cong}$	Behavioral Congruence	6.2.8	108
$\approx$	CCS-Weak Bisimilarity	4.2.12	69
	Closure under the addition of constraints	3.2.5	47
$\mathcal{M}^C(\langle P, c\rangle)$	Compact Input-Output Set	5.2.13	90
	Compact element	2.1.6	19
	Complete lattice	2.1.9	19
	Completeness	4.1.1	61
$\mathcal{C}, \mathcal{L}$	Configurations and Labels of a reachable pair	4.2.8	68
	Constraint Systems	2.4.1	22
$\mathsf{Deriv}(\gamma)$	Derivatives	5.1.4	82
	Directed set	2.1.4	18
	Directed-complete partial order	2.1.5	18
	Finitely Branching	4.2.7	67
	Improved partition refinement for CCP	5.3.1	94
$[\![\langle P, c\rangle]\!]$	Input-Output set	5.2.4	88
$\dot{\sim}_I$	Irredundant Bisimilarity	3.2.1	46
$\mathbf{IR}_{\rightsquigarrow}(\mathcal{P})$	Irredundant Refinement Function	3.1.9	44
$LTS_{\rightsquigarrow}$	LTSs and Graphs	2.2.3	20
$\mathcal{M}(\langle P, c\rangle)$	Labeled Input-Output Set	5.2.11	89
$(S, L, \rightsquigarrow)$	Labeled Transition System	2.2.1	20
$\sqcup, \bigsqcup$	Least upper bound	2.1.3	18

Acknowledgements

It is an extremely challenging task to acknowledge in just a few pages the support and advice I have received through all these years from so many amazing people. That said, in the next paragraphs, I will attempt to express my gratitude to those who were by my side and, in one way or another, helped me succeed in this journey.

I would like to start by thanking my supervisors Frank Valencia and Catuscia Palamidessi for giving me the unique opportunity to pursue my postgraduate studies in France. Their aid and encouragement in these past five years cannot be quantified with a single adjective; I am immensely grateful to have had them as my advisors and ultimately as my friends. Their cleverness and tenacity taught me the true meaning of never giving up and always striving for the best. Finally, it was truly an honor to be part of their team and I will certainly miss our endless discussions about science, technology, politics and sports.

I owe much to Juan Francisco Díaz for being my mentor since my undergraduate studies at Univalle. He introduced me to the world of computer science and his advice was a cornerstone in my decision to do a PhD. Most importantly, his continuous support was essential to my success in this endeavor.

I am also indebted to Filippo Bonchi, he has been the best collaborator one could imagine; his guidance was fundamental to the development of my thesis. I am deeply grateful to him for his advice before and during my PhD. My collaboration with him was crucial for undertaking this enterprise.

I want to thank Ugo Montanari and Stefano Bistarelli for reviewing my dissertation and providing such insightful comments. Moreover, I would like to thank Frank de Boer and Stéphane Graham-Lengrand for being part of my jury.

Thanks to INRIA and DGA for funding my PhD and to the Computer Science

Laboratory at École Polytechnique for hosting me during my doctoral studies.

I want to express my gratitude to Sara Södergren and Felipe Valencia for letting me be part of their family and for all the joyful moments we had during my period in France. I also want to thank Luzdary Restrepo for hosting and aiding me at the very beginning of my stay.

Of course none of this could have been achieved without the support of my wonderful friends. I am immensely grateful to Nicolás Bordenabe, Raphael Lopes and Miguel Andrés for being almost like my brothers during this period of my life. I am also profoundly thankful with Germán Aranzazu, Alejandro Arbelaez and Sophia Knight for their encouragement and the countless inspiring discussions throughout these years. I would also like to thank Andrés Aristizabal for the numerous times he helped me during my stance in France. My gratitude also goes to Andrés Barco, Guido Gamba, Carlos García, Felipe Gil, Alexandra Gómez, Jhon Gómez, Diego Lozada, Andrea Mairone, Federico Pousa, Carlos Potes, Catarina Silva, Marco Stronati, Elizabeth Vargas, Camilo Velez and Julián Zuluaga. Their support and advice was paramount for overcoming all sort of obstacles during my PhD,

Furthermore, I would like to thank the Comète research group and all its members for the wonderful times we spend these years. Special thanks to Mário Alvim, Kostas Chatzikokolakis, Jérémy Dubreil, Ehab ElSalamouny, Thomas Given-Wilson, Sardaouna Hamadou, Yusuke Kawamoto, Fernán Martinelli, Matteo Mio, Michela Paolini, Salim Perchy and Lili Xu.

I would also like to thank the AVISPA research group for letting me be part of their team. Thanks to Jesús Aranda, Julian Gutierrez, Carlos Olarte, Jorge Pérez and Camilo Rueda for their advice and guidance.

I am also grateful to my teachers at Univalle for inspiring me during my undergrad studies. Special thanks to Oscar Bedoya, Ivan Cabezas, Ángel Garcia, Raúl Gutierrez and Patricia Trujillo.

I want to thank the administrative staff of the École Polytechnique and INRIA for their help. Thanks to Alexandra Belus, Valérie Berthou, Audrey Lemarechal and Christine Ferret. Special thanks to Christelle Lievin and Marie-Jeanne Gaffard for their excellent job.

My gratitude also goes to all my friends at Cité Universitaire with whom I

spent some of the most amazing moments of my life. Thanks to Martín Andrade, Dario Azzimonti, Erasmo Batta, Luis Carlos Garcia Del Molino, Teresa Falcao, Andreia Furtado, Thomas Laich, Alex Lang, Nelson Luis, Erick Martins, Katerina Mityurova, Disha Pandit, Gabriel Papaiz, Marco Robalo, Juan Sebastián Rojas, Rosario, Maria Yeshchenko and Arturo Zamorategui.

I would like to thank to all my friends and colleagues at Google for making my summer in California such a fantastic experience. Special thanks to Federico Pousa, John Lenz, Ivan Arias, Diego Salas, Ezequiel Cura, Rodrigo Castaño, Matthew Loring, Vlatko Kolovski, Arjun Narayanan, Samuel de Sousa, the Flying Monkeys, the Social Frameworks and the JSCompiler Team.

I want to thank all the guys from the Panchester International FC for those winter nights of football and beers among friends. I also want to thank the Pumas and Khan Li Kung Fu schools for teaching me so many valuable lessons that turned out to come handy in many situations during my PhD studies. Special thanks to my masters Jhon Gonzalez and Marco Cobo.

I also want to express my gratitude to my friends around the world for helping me out in so many occasions. Thanks to Giulia Agostinelli, Jefferson Baracaldo, Catalina Bernal, Carlos Mario Borrero, Julian Camargo, John Carrillo, Héctor Cataño, Cinzia Di Giusto, Eduardo Gómez, Andrés Guerrero, Héctor Espitia, Carlos Guzman, Robinson Gonzalez, Anne-Lise Laurain, Luis Miguel Luna, Klaus Martinez, Susana Médina, Angelica Mera, Johan Ospina, Ramón Ospina, Arley Ossa, Carlos Ramos, Victor Romero, Daniel Sanchez, Alexandra Silva, Brian Vasquez, Johnathan Velazquez, Ivonne Villamil and Diego Villate.

I am profoundly grateful to life for having met my girlfriend Claudia during the course of my PhD. Her love and invaluable support have been central for keeping this project on track and overcoming any barriers in the path. Especially in those days where nothing made sense, she always had the right words to give me hope and help me succeed. There are no sentences to rightfully express all that you have done for me, thanks for everything my dear.

Finally, I want to thank my family, their love and support all these years is the reason why I am here. Thanks to my grandparents for giving me the strength to fight against anything in life. To my cousin Juan Manuel for always having that thorough advice and to all my uncles, aunts, cousins and my godfather Fernando

for their irreplaceable guidance. To my sister Jimena for teaching me the meaning of perseverance, to my dad Luis Fernando for always pushing me to the top, to my mom Olga Lucía and my aunt Mona for their unconditional love. To my brother Santiago for being my true companion. And to all of them for lending me their energy to accomplish this project.

To each one of you, thank you.

Luis Fernando Pino Duque
Cali, Colombia
February 2, 2015

Abstract

Concurrent constraint programming (CCP) is a mature linguistic formalism from the family of process calculi and hence it treats processes much like the λ-calculus treats computable functions. CCP is based on shared-memory communication where processes interact by adding and querying partial information represented as constraints (e.g., $x > 42$) in a global store. This dissertation is focused on the development of novel reasoning techniques for program equivalence in CCP and their efficient verification.

The first part of the thesis describes an algorithm for deciding strong bisimilarity for finite CCP processes. This is accomplished by first showing that the standard partition refinement approach does not work for CCP. Then it is shown how to adapt the standard approach for the case of CCP. Furthermore, it is proven that this procedure suffers from the state explosion, common in the verification of concurrent systems, due mostly to the presence of non-deterministic choices.

The second part is devoted to the development of a weak semantics for CCP. As pointed out in the literature, one can use the procedure for strong bisimilarity to decide weak bisimilarity. The idea is to define a new transition relation based on the operational semantics. This method is known as saturation. The standard saturation is defined by omitting the silent transitions in the calculus. This works for CCS and other calculi, however in the case of CCP, because of its involved labeled transitions, it is shown that the standard technique is not complete. Then a new saturation is defined called weak semantics for CCP. It consists of the reflexive and transitive closure under any constraint in CCP instead of just closing with respect to the silent transitions. Most importantly, it is proven that the proposed weak semantics is sound and complete for CCP. As a consequence, the new saturation can be used for checking weak bisimilarity.

In the third part the focus is shifted toward efficiency on the verification of weak bisimilarity. To achieve this, a representative sub-language of CCP is considered: the choice-free fragment (CCP\+). First, it is shown that the verification algorithms described above have an exponential-time complexity even for programs from CCP\+. Then by exploiting confluence, a distinctive feature from this fragment, two alternative polynomial-time decision procedures for CCP\+ weak bisimilarity are proposed. Each of these two procedures has an advantage over the other. One has a better time complexity, while the other can be easily adapted for the full language of CCP to produce significant state space reductions. The relevance of both procedures derives from the importance of CCP\+. This fragment, which has been the subject of much theoretical study, has strong ties to first-order logic and an elegant denotational semantics, and it can be used to model real-world situations. Most importantly, previously it was proven that weak bisimilarity in CCP\+ coincides with the standard observational equivalence for CCP\+. Hence from the results presented in this part, two efficient algorithms for checking program equivalence in CCP\+ are obtained.

The last part addresses the congruence issue for the full language of CCP. It is shown that weak saturated barbed bisimilarity is not a congruence for CCP, as is the case for weak bisimilarity in CCS. This problem is tackled by introducing a new notion that characterizes the weakest equivalence included in the congruence induced by weak bisimilarity for CCP. We call this new notion weak full bisimilarity. More importantly, it is proven that weak full bisimilarity is a congruence for the full language of CCP. It is also shown that weak full bisimilarity coincides with the standard notion of observational equivalence in CCP\+. To the best of our knowledge, this is the first notion of weak bisimilarity that is a congruence for the full language of CCP.

This dissertation contributes to the study of program equivalence in CCP. It provides an exponential-time decision procedure for strong bisimilarity in finite CCP as well as its adaptation to checking weak bisimilarity. Furthermore, this dissertation proposes two alternative polynomial-time algorithms for the verification of observational equivalence in the absence of nondeterministic choice. It concludes by proving that the existing notions of bisimilarity are not adequate for CCP with choice. Finally, it defines a novel reasoning technique which is proven

to be the right notion of equivalence for the full language of CCP.

Résumé

La Programmation Concurrente par Contraintes (CCP) est un formalisme linguistique mature de la famille des algèbres de processus, il traite les processus de la même façon que le λ-calcul traite les fonctions calculables. CCP est basé sur la communication à mémoire partagée où les processus interagissent en ajoutant et en interrogeant des informations partielles représentées comme des contraintes (par exemple, $x > 42$) dans une mémoire globale. Cette thèse se concentre sur le développement de nouvelles techniques de raisonnement pour léquivalence des processus dans CCP et leur vérification efficace.

La première partie de cette thèse décrit un algorithme pour calculer la bisimilarité forte pour des processus finis du CCP. Ceci est accompli en montrant d'abord que l'approche de raffinement des partitions standard ne fonctionne pas pour CCP. Ensuite, il est montré comment adapter l'approche standard pour le cas de CCP. En outre, il est prouvé que cette procédure souffre de l'explosion combinatoire, commune à la vérification de systèmes concurrents, principalement en raison de la présence de choix non-déterministe.

La deuxième partie est consacrée à l'élaboration d'une sémantique faible pour CCP. Comme il est souligné dans la littérature, on peut utiliser la procédure de bisimilarité forte pour décider la faible. L'idée est de définir une nouvelle relation de transition basée sur la sémantique opérationnelle, cette méthode est connue comme la saturation. La saturation standard est définie par l'omission des transitions silencieuses dans le calcul. Cela fonctionne pour le CCS et d'autres calculs, mais dans le cas de la CCP, en raison de ses transitions plus complexes, il est démontré que la technique standard n'est pas complète pour CCP. Ensuite, une nouvelle saturation est définie, nous l'appelons sémantique faible pour CCP. Elle consiste à la fermeture réflexive et transitive dans CCP au lieu de simplement la

clôture sur les transitions silencieuses. Surtout, il est prouvé que la sémantique faible proposée est correcte pour CCP. En conséquence, la nouvelle saturation peut łtre utilisée pour le calcul de la bisimilarité faible.

La troisième partie est dédiée à l'efficacité de la vérification de la bisimilarité faible. Pour ce faire, un sous-langage représentant du CCP est considérée: le fragment sans choix non-déterministe (CCP\+). Tout d'abord, il est montré que les algorithmes de vérification précédents ont une complexité exponentielle même pour les processus de CCP\+. Ensuite, en exploitant la confluence, une caractéristique distinctive de ce fragment, deux algorithmes en temps polynomiales alternatifs pour la bisimilarité faible CCP\+ sont proposés. Chacune de ces deux procédures a un avantage sur l'autre. La première présente une meilleure complexité en temps, alors que la seconde peut łtre facilement adaptée pour produire des importants améliorations dans lalgorithme pour tout le langage de CCP.

La dernière partie aborde la question de la congruence dans tout CCP. Il est montré que la bisimilarité faible n'est pas une congruence pour CCP, comme c'est le cas pour CCS. Ce problème est abordé par l'introduction d'une nouvelle notion qui caractérise l'équivalence la plus faible incluse dans la congruence induite par la bisimilarité faible pour CCP. Nous appelons cette nouvelle notion bisimilarité faible pleine. Plus important encore, il est prouvé que la bisimilarité faible pleine est une congruence pour CCP avec de choix non-déterministe. Il est également montré que la bisimilarité faible pleine concide avec la notion classique de l'équivalence observationnelle de CCP\+.

Chapter 1

Introduction

This dissertation proposes co-inductive reasoning techniques and algorithms for program equivalence in *concurrent constraint programming (CCP)*. The *thesis* of this dissertation is that these techniques represent a novel and significant contribution for the automatic verification of CCP programs.

We first give the general motivation and main goal of the dissertation and then continue to give more specific introduction to each one the results here presented.

1.1 Motivation

Concurrent and distributed systems have changed substantially with the advent of phenomena such as *social networks*. In past research on concurrent distributed systems the emphasis has mostly been on consistency, fault tolerance, resource management and related topics; these aspects were all characterized by interaction between processes. The new era of concurrent systems is marked by the importance of managing access to information to a much greater degree than before.

Social networks can be roughly described as group of agents that interact with each other by *posting* information in a *shared-medium*. The posted information can be partial (e.g., "I am somewhere in Europe") and it can be factual (e.g.,"today is Friday") and even epistemic and doxastic (e.g., "I believe/know Brazil won't be the world cup champion"). We believe that any model of this new era of systems

should single out these inherent aspects rather than encode them indirectly.

In this setting concurrent constraint programming language (CCP) [63], a process calculus for concurrency, has much to offer. CCP is a mature linguistic formalism from the family of *process calculi* and as such it treats processes like the λ-calculus treats computable functions. It provides a language in which the structure of terms represents the structure of processes together with an operational semantics to represent computational steps. Most representative process calculi for concurrency such as CCS [41] and the π-calculus [43] are based on point-to-point communication, instead CCP is based on *shared-medium communication* much like in social networks: Processes (or agents) interact by posting partial information represented as constraints in a shared *store*. Furthermore, CCP is parametric in a *constraint system* that can be specialized to express the domain-specific language (and its entailment) for the family of systems under consideration. In the case of social networks, a constraint system can be used to specify epistemic and doxastic modalities that are suitable for reasoning about social behavior: E.g., modalities to express familiar concepts in social networks such as beliefs, opinions, knowledge, lies, and hoaxes.

Despite being a suitable language for the emergent shared-memory systems of today's digital world, the automatic or machine-assisted verification of system properties in CCP has hitherto been far too little considered. This is unfortunate because these systems are intrinsically big and thus "pen-and-pencil" analysis are not sufficient. As argued by [30], the research agenda for theoretical concurrency should address the design of efficient algorithms for translating and verifying formal specifications of concurrent systems.

The automatic verification of concurrent systems, however, poses a fundamental challenge due to the state explosion problem. The number of states a system has is exponential in the number of concurrent processes. In this dissertation we rise to this challenge by identifying fragments of CCP amenable to automatic verification and developing novel techniques and tools to machine assist the verification of CCP program equivalence.

We have motivated this dissertation by arguing that CCP is a suitable language for the emergent shared-memory systems of today's digital world and yet, in comparison with more traditional calculi such as [41, 43], there is no much work on

verification techniques for CCP. The *main goal* of this dissertation is therefore to advance significantly the theory and practice of CCP by endowing it with reasoning techniques that allows for automated verification of program equivalence. We now wish to conclude this section with a quote from [30] that captures the goal of the present dissertation:

"*The times have gone, where formal methods were primarily a pen-and-pencil activity for mathematicians. Today, only languages properly equipped with software tools will have a chance to be adopted by industry. It is therefore essential for the next generation of languages based on process calculi to be supported by compilers, simulators, verification tools, etc.*".

In what follows we shall give some context and then explain the *specific goals* and *approach* of this dissertation.

1.2 Context

In this section we shall present the general context in which this dissertation is developed.

1.2.1 Concurrent Constraint Programming (CCP)

Concurrency theory studies the description and the analysis of systems consisting of interacting *processes*. Processes are typically viewed as infinite objects, in the sense that they can produce arbitrary and possibly endless interactions with their environment. *Process calculi* treat these processes much like the λ-calculus treats computable functions. They provide a formal language in which processes are represented by terms, and a set of rewriting rules to represent process evolution (or transitions). For example, the term $P \parallel Q$ represents the process that results from the parallel composition of the processes P and Q. A (labeled) transition $P \xrightarrow{\alpha} P'$ represents the evolution of P into P' given an interaction α with the environment.

Concurrent Constraint Programming (CCP) [62, 63] is a mature formalism from concurrency theory that combines the traditional algebraic and operational view of process calculi with a declarative one based on first-order logic (see a

survey in [47]). CCP processes can then be seen as computing agents as well as first-order logic formulae. In CCP, processes interact asynchronously by *posting* (or *telling*) and querying (or *asking*) information, traditionally referred to as *constraints*, in a shared-medium referred to as *the store*. Furthermore, CCP is parametric in a *constraint system* indicating interdependencies (entailment) between constraints and providing for the specification of data types and other rich structures. The above features have recently attracted renewed attention as witnessed by the works [52, 19, 11, 10] on calculi exhibiting data-types, logic assertions as well as tell and ask operations. More recently in [36] the authors proposed the post and ask interaction model of CCP as an abstraction of *social networks*.

The features of CCP (and variants) have been used in a variety of applications in areas such as security, biology and multimedia interaction. For instance, CCP has been exploited to analyze security breaches in cryptographic protocols [48], for the prediction of organic malfunctions [31] and rhythmic coherence in music improvisation [59]. Furthermore, CCP foundations and principles e.g., semantics, proof systems, axiomatizations, have been thoroughly studied for over the last two decades. In contrast, as we previously argued the development of algorithms and automatic verification procedures for CCP have hitherto been far too little considered. In the next section we shall discuss the existing notions of equivalence for CCP.

1.2.2 Equivalences for CCP

In any computational model of processes, a central notion is that of *behavioral equivalences* [22]. In the case of CCP, the standard notion of *observational equivalence* from [63] ($\sim_o$), roughly speaking, decrees that two CCP programs are observationally equivalent if each one can be replaced with the other in any CCP context and produce the same final result.

By *process context*, we mean a term C with a single hole $\bullet$ such that if we replace $\bullet$ with a process P, we obtain a process term $C[P]$. For example, for the parallel context $C = \bullet \parallel R$ we obtain $C[P] = P \parallel R$.

Reasoning on processes and their equalities therefore means dealing with, and comparing, infinite structures. For this, a widely used mathematical tool is *coin-*

duction (see e.g. [4]). Coinduction is the dual of induction; while induction is a pervasive tool for reasoning about finite and stratified structures, coinduction offers similar strengths on structures that are circular or infinite. The most widely applied coinductive concept is *bisimulation*: *bisimilarity* is used to study behavioral equivalences, and the bisimulation proof method is used to prove such equivalences. In fact, most process calculi are equipped with a notion of bisimilarity.

Intuitively, for two processes, say P and Q, to be bisimilar one typically requires that whenever a process P performs an action α and evolves to P', typically written $P \xrightarrow{\alpha} P'$, then Q must pick a successor Q' such that $Q \xrightarrow{\alpha} Q'$ where P' and Q' are now bisimilar. The idea is then that two processes are strongly equivalent if they can mimic each other's actions while arriving at equivalent successors. The intuition above corresponds, roughly speaking, to the standard notion of *strong bisimilarity*.

An alternative way of looking at bisimilarity is by using the so-called *bisimulation game* [53, 42] explained as follows. Given two players, called attacker and defender, the game starts with the attacker picking a transition $P \xrightarrow{\alpha} P'$ or $Q \xrightarrow{\alpha} Q'$. Next the defender must reply by choosing a transition in $Q \xrightarrow{\alpha} Q'$ or $P \xrightarrow{\alpha} P'$, respectively. Notice that they must perform the exact same action. Now if this game can be played forever then the defender wins, thus P and Q are considered strongly bisimilar. Otherwise, if the defender cannot reply successfully to the attacker, either because at the current process no transition is available, or no transition is tagged with the same action α, then P and Q are not deemed strongly bisimilar.

There have been few attempts to define the notion of bisimilarity equivalence for CCP processes and they have not been completely satisfactory. The first one was proposed in the seminal work on the semantic foundations of CCP by Saraswat et al. [62]. Their bisimilarity is based on the standard bisimilarity we described above. Unfortunately, as shown in [6], the *matching of actions* from the game above was proven to be too fine grained; i.e. it may tell apart processes whose logical interpretation is identical.

Furthermore, similar notions were used in [68] to relate the fusion calculus with the ρ-calculus, another concurrent constraint formalism. However, such notions are not defined for the ρ-calculus itself. More recently, in [32] the author

defines studies several notions of equivalence for linear CCP.

The latest attempt comes from [6] where the authors try to solve the aforementioned problems by introducing a notion of bisimilarity for CCP called *saturated barbed bisimilarity*. The results obtained in [6] were part of the author's master thesis, and for this reason they will be presented in the background material of the dissertation.

This approach is inspired by the notion of *saturated bisimilarity* from [14, 12] (pioneered by [46]). Intuitively, in order for two processes to be saturated bisimilar, (i) they should expose the same *barbs*, which are basic observations on the processes, (ii) whenever one of them moves then the other should reply and arrive at an equivalent process (i.e. follow a bisimulation game similar to the one described above), (iii) they should be equivalent under all the possible contexts of the language.

Nevertheless, to the best of our knowledge, none of these notions are provided with a decision procedure for their verification. Hence, one of the goals of this dissertation is to study whether the existing equivalences are adequate for CCP and to find efficient mechanisms to compute them, with particular emphasis on weak bisimilarity since it characterizes the standard notion of observational equivalence for CCP.

1.3 This Dissertation

We shall give some specific motivation and goals as well as a description of the approach we followed to obtain the results of this dissertation. Each of the sections below corresponds to a chapter in the present document.

1.3.1 Checking Strong Bisimilarity in CCP

Several efficient decision procedures and techniques for verifying bisimilarity have been developed [67, 25, 27]. But perhaps the best known is the *partition refinement algorithm* [34, 50]. This algorithm can be briefly explained as follows: First it generates the state space of a labeled transition system (LTS), i.e., the set of states reachable through the transitions. Then it creates a partition equating all

states and finally, iteratively, it refines these partitions by splitting non equivalent states. The criteria for splitting partitions usually depends on the notion of bisimilarity, for instance, for the case of standard bisimilarity, the criteria would be the matching of actions (as described in the previous section). The resulting partition equates all and only bisimilar states.

In Chapter 3.1.2 we propose an adaptation of the partition refinement algorithm to decide *strong saturated barbed bisimilarity* ($\dot{\sim}_{sb}$) in CCP. Inspired by [3], the adaptation is based on the observation that some of the transitions are *redundant*, in the sense that they are logical consequences of other transitions. Unfortunately, such a notion of redundancy is not syntactic, but semantic, more precisely, it is based on $\dot{\sim}_{sb}$ itself.

A possible solution would be then to consider the transition system having only non-redundant transitions, thus the ordinary notion of bisimilarity coincides with $\dot{\sim}_{sb}$. Hence, in principle, we could remove all the redundant transitions and then check bisimilarity with a standard algorithm. But how can we decide which transitions are redundant, if redundancy itself depends on $\dot{\sim}_{sb}$?

As we shall explain later on, the main idea is to compute $\dot{\sim}_{sb}$ and redundancy *at the same time*. In the first step, the algorithm deems all the states to be equivalent and all the (potentially redundant) transitions to be redundant. In any iteration, states are distinguished according to (the current estimation of) non-redundant transitions and then non-redundant transitions are updated according to the new computed partition.

One peculiarity of the algorithm in Chapter 3.1.2 is that in the initial partition, we insert not only the reachable states, but also extra ones which are needed to check for redundancy. We shall see that, unfortunately, the number of these states might be exponentially bigger than the size of the original set of reachable states and therefore the worst-case complexity is *exponential*. To the best of our knowledge, this is the first algorithm for checking strong bisimilarity in CCP.

1.3.2 Reducing Weak to Strong Bisimilarity

In strong equivalences, and specifically strong bisimilarity, all the transitions performed by a system are deemed observable. In other words, we have seen how two

processes are strongly bisimilar if and only if they can mimic each other's actions, one by one. Instead in *weak equivalences* internal transitions are unobservable, namely transitions $P \longrightarrow P'$ which requires no interaction with the environment are invisible for this notion of equivalence. Internal transitions (also called reductions) are usually denoted by $P \xrightarrow{\tau} P'$ where τ is an invisible action. On the one hand, weak equivalences are more abstract (and thus closer to the intuitive notion of behavior); on the other hand, strong equivalences are usually much easier to check (for instance, in [37] a strong equivalence is introduced which is computable for a Turing complete formalism).

The weak version of $\dot{\sim}_{sb}$ is called *weak saturated barbed bisimilarity* ($\dot{\approx}_{sb}$) and it is defined as $\dot{\sim}_{sb}$ but in this case both the observations on processes (barbs) and the game may use zero or more steps (reductions) instead of exactly one.

Following [2], the problem of checking *weak bisimilarity* can be reduced to the strong one. The standard reduction goes as follows. Given a (strong) LTS $\longrightarrow$ labeled with actions $a, b, \tau, \ldots$ one can build a (weak) LTS $\Longrightarrow$ using the following inference rules:

$$\frac{P \xrightarrow{a} Q}{P \overset{a}{\Longrightarrow} Q} \qquad \frac{}{P \overset{\tau}{\Longrightarrow} P} \qquad \frac{P \overset{\tau}{\Longrightarrow} P_1 \overset{a}{\Longrightarrow} Q_1 \overset{\tau}{\Longrightarrow} Q}{P \overset{a}{\Longrightarrow} Q}$$

One can prove that weak bisimilarity on $\xrightarrow{a}$ coincides with strong bisimilarity on $\overset{a}{\Longrightarrow}$. Hence weak bisimilarity can be checked with the algorithms for strong bisimilarity on the new LTS $\overset{a}{\Longrightarrow}$.

Unfortunately, in the case of CCP, the above assertion does not hold. In Chapter 4, we shall show that the standard method for reducing weak to strong bisimilarity does not work for CCP. The core of the problem lies on the actions performed by a CCP agent. Since the labels in the labeled transition system of a CCP agent are constraints, in fact, they are "the minimal constraints" that the store should satisfy in order to make the agent progress. These constraints form a lattice where the least upper bound (denoted by $\sqcup$) intuitively corresponds to conjunction and the bottom element is the constraint $true$. (As expected, transitions labeled by $true$ are internal transitions, corresponding to the τ moves in standard process calculi).

As we shall explain in Chapter 4, in CCP, rather than closing the transitions

just with respect to $true$, we shall need to close them with respect to all the constraints. Formally we build the new LTS with the following rules.

$$\frac{P \xrightarrow{a} Q}{P \stackrel{a}{\Longrightarrow} Q} \qquad \frac{}{P \stackrel{true}{\Longrightarrow} P} \qquad \frac{P \stackrel{a}{\Longrightarrow} Q \stackrel{b}{\Longrightarrow} R}{P \stackrel{a \sqcup b}{\Longrightarrow} R}$$

Notice that the above construction can also be done for CCS [41] by taking sequences of actions $a; b$ rather than $a \sqcup b$. Nevertheless, the resulting transition system may be infinite-branching and hence not amenable to automatic verification using standard algorithms such as partition refinement.

Since in CCP we have that $\sqcup$ is idempotent then if the original LTS $\xrightarrow{a}$ has finitely many transitions, then also $\stackrel{a}{\Longrightarrow}$ is finite. This allows us to use the algorithm in Chapter 4 to check $\dot{\approx}_{sb}$ on (the finite fragment) of CCP. To the best of our knowledge, this is the first algorithm for checking weak bisimilarity in CCP.

1.3.3 Computing Bisimilarity in choice-free CCP

Despite being able to successfully compute bisimilarity in CCP, the methods in Chapters 3 and 4 briefly discussed previously are inefficient since they have an *exponential time* (and space) complexity.

The main goal of Chapter 5 is to produce efficient decision procedures for program equivalence for CCP. To achieve this, our approach is to restrict CCP to a meaningful fragment. Namely, the *choice-free fragment of CCP*, henceforth CCP$\backslash$+. The CCP$\backslash$+ formalism is perhaps the most representative sublanguage of CCP. It has been the subject of much theoretical study because of its computational expressivity, strong ties to first-order logic, and elegant denotational semantics based on closure operators [63]. Its most distinctive property is that of *confluence* in the sense that the final resulting store is the same regardless of the execution order of the parallel processes. We shall use this property extensively in proving the correctness of our decision procedures. Furthermore, from [6], we know that in this fragment $\dot{\approx}_{sb}$ *coincides* with the standard CCP observational equivalence for CCP$\backslash$+ programs from [63].

When considering the weak equivalence $\dot{\approx}_{sb}$ and the approach from Chapter 4, confluence makes it possible to characterize redundant transitions syntactically,

i.e., without any information about $\dot{\approx}_{sb}$. Therefore for checking $\dot{\approx}_{sb}$ in CCP\+, we can first prune redundant transitions and then check the standard bisimilarity with one of the usual algorithms [34, 26, 16, 23]. Since redundancy can be determined statically, the additional states needed by the algorithm in Chapter 4 will not be necessary any more: in this way, the worst case complexity from exponential becomes *polynomial*.

Unfortunately, this approach still suffers from the explosion of transitions caused by the "closure" of the transition relation. In order to avoid this problem, we exploit a completely different approach (based on the semantic notion of *compact input-output sets*) that works directly on the original LTS. Intuitively, this approach consists of reducing the problem of checking $\dot{\approx}_{sb}$ in CCP\+ to the problem of whether the inputs have the same minimal finite representation regarding the outputs they produce in every possible context. We shall also show that the results obtained for CCP\+ can be exploited to optimize the partition refinement for the full language of CCP.

1.3.4 A Behavioral Congruence for CCP

The equivalences mentioned above determine what processes are deemed indistinguishable and they are expected to be *congruences*. The congruence issue is of great importance for algebraic and *compositional* reasoning: If two processes are equivalent, one should be able to replace one with the other in any context and preserve the equivalence (see e.g, [28]). For example, if $\bowtie$ is a behavioral congruence, then $P \bowtie Q$ should imply $P \parallel R \bowtie Q \parallel R$.

We shall build on a result of [6] showing that $\dot{\approx}_{sb}$ can be characterized by a novel bisimulation game (called, for simplicity, weak bisimulation) which relies at the same time on both *barbs* and *labeled transitions*. Recall that barbs are basically predicates on the states, processes or configuration stating the observation we can make of them. This is rather peculiar with respect to the existing notions of bisimulation introduced for other process calculi where one usually exploits labeled transitions to avoid thinking about barbs and contexts. Indeed, labeled transitions usually capture barbs, in the sense that a state exposes a certain barb if and only if it performs a transition with a certain label. This is not the case

in CCP, where barbs are observations on the store, while labeled transitions are determined by the processes. A more abstract understanding of this peculiarity of CCP can be given within the framework of [13] which is an extension of [38] featuring barbs and weak semantics.

As is customary for weak barbed equivalences, in our weak bisimulation game whenever a player exposes a barb $\downarrow_e$, the opponent should expose the weak barb $\Downarrow_e$, i.e. it should be able to reach a state satisfying $\downarrow_e$, but then the game restarts from the original state ignoring the arriving state. One of our contributions is to show that for CCP the arriving state cannot be ignored.

In Chapter 6 we shall show that the weak saturated barbed bisimilarity ($\dot{\approx}_{sb}$) [6] is not a congruence for CCP with nondeterministic choice. This problem is not particular to CCP, since many notions of weak bisimilarity are known not to be congruences in the presence of nondeterminism, for instance in CCS [42]. The problem is then to find a "good" variation of weak bisimilarity that is not too restrictive. For CCP, we introduce a new notion called *weak full bisimilarity* ($\approx_f$) and we prove that it is adequate for CCP since (i) it is a congruence for the full CCP and (ii) it corresponds to the weakest equivalence included in the congruence induced by $\dot{\approx}_{sb}$. In fact, it is also adequate since in the choice-free fragment $\approx_f$ corresponds to the standard notion of observational equivalence [63]. This is the first notion of weak bisimilarity that is a congruence for the full language of CCP.

1.4 Contributions and Organization

In this section we discuss the main contributions of this dissertation. Additionally, each chapter starts with an overview of its contents and ends with a detailed recap including a discussion of the related work.

Chapter 3. In this chapter we propose a decision procedure for verifying *strong saturated barbed bisimilarity* ($\dot{\sim}_{sb}$) in the finite fragment of CCP. To solve this problem we adapt the well-known *partition refinement* algorithm to the case of CCP, inspired by the results from [15]. We also explain in detail how to tackle the issues arising from the adaptation to CCP as well as the correctness and complexity of the algorithm proposed in this chapter. To the best of our knowledge, this is

the first algorithm for the automatic verification of strong bisimilarity in CCP.

Chapter 4. In this chapter we introduce a *weak semantics* for CCP which can be used, together with the algorithm from Chapter 3, to verify *weak saturated barbed bisimilarity* ($\dot{\approx}_{sb}$) in the finite fragment of CCP. Intuitively, one can start from the labeled transition system generated using the operational semantics and then add some extra transitions, called *weak transitions*. Thus checking strong bisimilarity in the transformed input (with the new transitions) is equivalent to deciding weak bisimilarity in the original input. We prove that the standard way of defining the weak transitions, namely omitting the silent transitions, does not work for CCP. Similarly, we define a new weak semantics and we prove it adequate for CCP. Finally, using this new method, and based on Chapter 3, we propose a decision procedure to check $\dot{\approx}_{sb}$. We conclude by proving the correctness and complexity of the algorithm defined in this chapter. To the best of our knowledge, this is the first algorithm for the automatic verification of weak bisimilarity in CCP.

Chapter 5. In this chapter we propose novel efficient methods for checking $\dot{\approx}_{sb}$ in a fragment of CCP. We first show that the algorithms from the previous chapters have an exponential time complexity even in the restricted case of finite CCP programs without nondeterministic choice. The main contribution of this chapter is the introduction of two novel decision procedures that can be used to decide $\dot{\approx}_{sb}$ for the finite choice-free fragment of CCP in *polynomial time*. This is achieved by exploiting certain distinctive features of this fragment such as *confluence*. Each of these two new procedures has an advantage over the other. One has a better time complexity. The other can be easily adapted for the full language of CCP to produce significant state space reductions. Recall that from [6] we know that, in the absence of nondeterminism, $\dot{\approx}_{sb}$ coincides with the standard notion of *observational equivalence* ($\sim_o$) for CCP from [63], hence from this chapter we obtain two polynomial decision procedures to decide program equivalence in choice-free CCP.

Chapter 6. In this chapter, we tackle the *congruence* issues related to $\dot{\approx}_{sb}$. First we prove that $\dot{\approx}_{sb}$ is a congruence for CCP without nondeterministic choice but

not for the full language of CCP. We then propose a new notion of bisimilarity, called *weak full bisimilarity* ($\approx_f$). We show that $\approx_f$ is a congruence for the full language of CCP. We also show the adequacy of the new notion by establishing that it is the largest congruence included in $\dot{\approx}_{sb}$. In other words $\approx_f$ coincides with the congruence induced by closing $\dot{\approx}_{sb}$ under all contexts. Beyond being a congruence, the advantage of $\approx_f$ is that it does not require quantifying over infinitely many contexts. This is also important as it may simplify decision procedures for the equivalence. To the best of our knowledge, this is the first behavioral equivalence, which does not appeal to quantification over arbitrary process contexts in its definition, that is a congruence for CCP with nondeterministic choice.

Remark 1.4.1. *Part of the material in Chapters 3 and 4 is also included in the thesis of Aristizabal [5]. However, those chapters are based on the journal paper [54] instead of [8, 7] as in [5].*

Chapters 3 and 4 are a generalization of the results in [8, 7]. This version includes more detailed explanations and essential corrections to the original material. It also contains new contributions such as the specification of the algorithm for deciding observational equivalence in CCP.

1.5 Publications from this Dissertation

The material in the present thesis has been published in the following papers:

Journals

- [54] **L. Pino**, A. Aristizabal, F. Bonchi, F. Valencia, *Weak CCP bisimilarity with strong procedures*. Science of Computer Programming, 100(0):84-104, 2015.
  DOI: `http://dx.doi.org/10.1016/j.scico.2014.09.007`.

  The contributions of this paper are presented in Chapter 4.

- [55] **L. Pino**, F. Bonchi, F. Valencia. *Efficient Algorithms for Program Equivalence for Confluent Concurrent Constraint Programming*. To appear

in the Journal of Science of Computer Programming, 2015.
DOI: http://dx.doi.org/10.1016/j.scico.2014.12.003.

The contributions of this paper are presented in Chapter 5 and 6.

Proceedings of International Conferences

- [8] A. Aristizabal, F. Bonchi, **L. Pino**, F. Valencia, *Partition Refinement for Bisimilarity in CCP*, in: S. Ossowski, P. Lecca (Eds.), Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012), Constraint Solving and Programming Track, ACM, 2012, pp. 88-93.
DOI: http://dx.doi.org/10.1145/2245276.2245296.

  The contributions of this paper are presented in Chapter 3.

- [56] **L. Pino**, F. Bonchi, F. Valencia, *Efficient Computation of Program Equivalence for Confluent Concurrent Constraint Programming*, in: R. Peña, T. Schrijvers (Eds.), Proceedings of the 15th International Symposium on Principles and Practice of Declarative Programming (PPDP 2013), ACM, 2013, pp. 263-274.
DOI: http://dx.doi.org/10.1145/2505879.2505902.

  The contributions of this paper are presented in Chapter 5.

- [58] **L. Pino**, F. Bonchi, F. Valencia. *A Behavioral Congruence for Concurrent Constraint Programming with Non-deterministic Choice*, in: In G. Ciobanu and D. Méry (Eds.), Proceedings of the 11th International Colloquium on Theoretical Aspects of Computing (ICTAC 2014), volume 8687 of Lecture Notes in Computer Science, pages 351-368. Springer, 2014.
DOI: http://dx.doi.org/10.1007/978-3-319-10882-7_21

  The contributions of this paper are presented in Chapter 6.

Proceedings of International Workshops

- [7] A. Aristizabal, F. Bonchi, **L. Pino**, F. Valencia, *Reducing Weak to Strong Bisimilarity in CCP*, in: M. Carbone, I. Lanese, A. Silva, A. Sokolova

(Eds.), Proceedings of the 5th Interaction and Concurrency Experience (ICE 2012), volume 104 of Electronic Proceedings in Theoretical Computer Science, 2012, pp. 2-16.
DOI: `http://dx.doi.org/10.4204/EPTCS.104.2`.

The contributions of this paper are presented in Chapter 4.

Extended abstract with informal proceedings

- [57] **L. Pino**, F. Bonchi, F. Valencia. *Efficient Computation of Program Equivalence for Confluent Concurrent Constraint Programming (Extended Abstract)*. (Informal) Proceedings of the 5th Young Researchers Workshop on Concurrency Theory (YR-CONCUR 2013).

  The contributions of this paper are presented in Chapter 5.

Chapter 2

Preliminaries

In this chapter we introduce the background we shall use throughout the dissertation. In Section 2.1 we recall some basic notions about domain theory. In Section 2.2 we define the notions of labeled transition system (LTS), partition and the graph induced by an LTS. In Section 2.3 we proceed to present the standard partition refinement algorithm used to check bisimilarity between two LTSs. Finally, in Section 2.4, we present in detail the concurrent constraint programming (CCP) formalism. We introduce its syntax together with its reduction and labeled semantics, as well as several equivalence relations including strong, weak bisimilarity and the standard observational equivalence for CCP. The reader familiar with these concepts may skip directly to Chapter 3.

2.1 Domain theory

Here we present the basic elements from domain theory that we shall use throughout the dissertation. Most definitions are presented as in [35]. For a more detailed explanation, see for instance [1].

We start with the notion of *partially ordered set*. Intuitively, a set is partially ordered if it is equipped with a binary relation that is reflexive, transitive and antisymmetric. Its name comes from the fact that the relation indicates the partial order of the elements in the set.

Definition 2.1.1 (Partially ordered set)**.** *For a set S with a binary relation $\sqsubseteq$,*

we say that $(S, \sqsubseteq)$ *is a* partially ordered set*, or a* poset*, if the following three properties hold for all* $x, y, z \in S$*:*

1. *Reflexivity:* $x \sqsubseteq x$.
2. *Transitivity: if* $x \sqsubseteq y$ *and* $y \sqsubseteq z$ *then* $x \sqsubseteq z$.
3. *Antisymmetry: if* $x \sqsubseteq y$ *and* $y \sqsubseteq x$ *then* $x = y$.

If S *with relation* $\sqsubseteq$ *is a partially ordered set then* $\sqsubseteq$ *is called a* partial order.

Now we define the concept of *upper bound*. Roughly speaking, given a poset $(S, \sqsubseteq)$, an upper bound for a subset A of S is an element u that belongs to S and is greater or equal, w.r.t. the order $\sqsubseteq$, than any element in A.

Definition 2.1.2 (Upper bound)**.** *If* $(S, \sqsubseteq)$ *is a poset and* $A \subseteq S$*, then* $u \in S$ *is an* upper bound *for* A *if for all* $a \in A$*,* $a \sqsubseteq u$.

The next definition is that of *least upper bound*. Intuitively, for a poset $(S, \sqsubseteq)$ and a subset A of S, the least upper bound of A is the smallest, w.r.t. $\sqsubseteq$, of all possible upper bounds for A.

Definition 2.1.3 (Least upper bound)**.** *If* $(S, \sqsubseteq)$ *is a poset and* $A \subseteq S$*, then* $u \in S$ *is the* supremum *or* least upper bound *of* A *if* u *is an upper bound for* A *and for all* $u' \in S$*, if* u' *is an upper bound for* A *then* $u \sqsubseteq u'$*. We denote the least upper bound of a set* A *as* $\bigsqcup A$*, and we denote the least upper bound of the set* $\{a_1, a_2\}$ *as* $a_1 \sqcup a_2$*. We also refer to* $a_1 \sqcup a_2$ *as the* lub *of* a_1 *and* a_2.

Now we shall define the *directed set*: a set is directed if for any pair of elements of the set one can find another element in the set that is above them.

Definition 2.1.4 (Directed set)**.** *Suppose* $(S, \sqsubseteq)$ *is a poset, and* $D \subseteq S$*. We say that* D *is* directed *if for all* $a, b \in D$ *there exists* $c \in D$ *such that* $a \sqsubseteq c$ *and* $b \sqsubseteq c$.

Using this notion we proceed to define the concept of *directed-complete partial order*.

Definition 2.1.5 (Directed-complete partial order)**.** *Suppose* $(S, \sqsubseteq)$ *is a poset.* $(S, \sqsubseteq)$ *is a* directed-complete partial order*, or* dcpo*, if every directed subset of* S *has a least upper bound.*

Next we define the concept of *compact element*. Intuitively, an element is compact if whenever it is smaller than the least upper bound of a directed set D then it is smaller than some element in D. In other words, an element is compact if it can be finitely approximated by one of the elements in D.

Definition 2.1.6 (Compact element)**.** *Suppose $(S, \sqsubseteq)$ is a dcpo and $d \in S$. Then d is a* compact element *of S if whenever D is a directed subset of S and $d \sqsubseteq \bigsqcup D$ then there exists $d^* \in D$ such that $d \sqsubseteq d^*$.*

The proposition below states that the least upper bound of a finite set of compact elements is also compact.

Proposition 2.1.7. *If $(S, \sqsubseteq)$ is a dcpo and C is a finite set of compact elements of S, then if $\bigsqcup C$ exists, it is a compact element of S as well.*

The following notion says that a dcpo is *algebraic* if each element c is the least upper bound of the compact elements below c.

Definition 2.1.8 (Algebraic dcpo)**.** *Suppose $(S, \sqsubseteq)$ is a dcpo. Let $K(S)$ denote the compact elements of S. $(S, \sqsubseteq)$ is* algebraic *if for every $a \in S$,*

$$a = \bigsqcup \{d \mid d \sqsubseteq a \text{ and } d \in K(S)\}.$$

Finally, the notion of *complete lattice*.

Definition 2.1.9 (Complete lattice)**.** *$(S, \sqsubseteq)$ is a* complete lattice *if $(S, \sqsubseteq)$ is a poset and for every subset A of S, A has a least upper bound.*

Note that a complete lattice is always a dcpo. We finish this section with the following proposition.

Proposition 2.1.10. *Every complete lattice has a unique greatest element and a unique least element.*

2.2 Labeled Transition System, Partition and Graphs

In this section we present the basic notions of labeled transition system, partition and graphs. Intuitively, a transition system consists of a set of *states* and arrows

between those states representing a *transition* from one state to the other. Additionally, its labeled version uses a set of *labels* to tag the transitions typically to denote an interaction with the environment using this piece of information.

Definition 2.2.1 (Labeled Transition System)**.** *A labeled transition system (LTS) is a triple* $(S, L, \rightsquigarrow)$ *where* S *is a set of states,* L *a set of labels and* $\rightsquigarrow \subseteq S \times L \times S$ *a transition relation. We shall use* $s \overset{a}{\rightsquigarrow} r$ *to denote the transition* $(s, a, r) \in \rightsquigarrow$*. Given a transition* $t = (s, a, r)$ *we define the source, the target and the label as follows:* $src(t) = s$*,* $tar(t) = r$ *and* $lab(t) = a$*. For a transition* $s \overset{a}{\rightsquigarrow} r$*, we shall often say that* s *performs the action* a *and then becomes* t*.*

We now proceed to define the notion of *partition of a set*. Roughly speaking, a partition of a set S is a division of the set into blocks such that no element is in two blocks at the same time and where the union of all blocks is S.

Definition 2.2.2 (Partition)**.** *Given a set* S*, a* partition $\mathcal{P}$ *of* S *is a set of non-empty* blocks*, i.e., subsets of* S*, that are all disjoint and whose union is* S*. We write* $\{B_1\} \ldots \{B_n\}$ *to denote a partition consisting of (non-empty) blocks* $B_1, \ldots, B_n$*. A partition represents an equivalence relation where equivalent elements belong to the same block. We write* $s \mathcal{P} r$ *to mean that* s *and* r *are equivalent in the partition* $\mathcal{P}$*.*

Finally, we adopt some notation relating LTSs and regular graphs.

Definition 2.2.3 (LTSs and Graphs)**.** *Given a LTS* $(S, L, \rightsquigarrow)$*, we write* $LTS_{\rightsquigarrow}$ *for the directed graph whose vertices are the states in* S *and edges are the transitions in* $\rightsquigarrow$*. Given a set of initial states* $IS \subseteq S$*, we write* $LTS_{\rightsquigarrow}(IS)$ *for the subgraph of* $LTS_{\rightsquigarrow}$ *reachable from* IS*. Given a graph* G *we write* $\mathtt{V}(G)$ *and* $\mathtt{E}(G)$ *for the set of vertices and edges of* G*, respectively.*

2.3 Partition Refinement

In this section we recall the partition refinement algorithm [34, 50] for checking bisimilarity over the states of an LTS $(S, L, \rightsquigarrow)$.

First let us introduce the *standard* notion of *strong bisimilarity*. Intuitively, two states s and t are considered strongly bisimilar if whenever s performs an

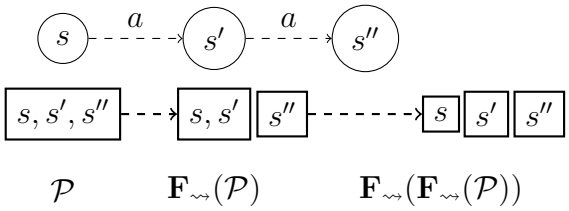

Figure 2.3.1: An example of the use of $\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$ from Definition 2.3.2.

action a and becomes s', i.e. $s \overset{a}{\rightsquigarrow} s'$, then t is able to perform a and reach a t', i.e. $t \overset{a}{\rightsquigarrow} t'$, such that t' is equivalent to s'. In short, s and t are bisimilar if they are able to mimic each other's moves.

Definition 2.3.1 (Standard Strong Bisimilarity)**.** *A symmetric relation $\mathcal{R}$ is a standard strong bisimulation if for every $(s,t) \in \mathcal{R}$:*

- *If $s \overset{a}{\rightsquigarrow} s'$ then there exists t' s.t. $t \overset{a}{\rightsquigarrow} t'$ and $(s',t') \in \mathcal{R}$.*

We say that s and t are standard strong bisimilar, written $s \sim t$, iff there is a standard strong bisimulation containing (s,t).

Given a set of initial states $IS \subseteq S$, the partition refinement algorithm (see Algorithm 2.3.1) checks bisimilarity on IS as follows. First, it computes $IS^{\star}_{\rightsquigarrow}$, that is the set of all states that are reachable from IS using $\rightsquigarrow$. Then it creates the partition $\mathcal{P}^0$ where all the elements of $IS^{\star}_{\rightsquigarrow}$ belong to the same block (i.e., they are all equivalent). After the initialization, it iteratively refines the partitions by employing the function on partitions $\mathbf{F}_{\rightsquigarrow}(-)$, defined as follows:

Definition 2.3.2 (Refinement Function)**.** *Given a partition $\mathcal{P}$ we define $\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$ as follows: $s\mathbf{F}_{\rightsquigarrow}(\mathcal{P})r$ iff*

$$\textit{if } s \overset{a}{\rightsquigarrow} s' \textit{ then exists } r' \textit{ s.t. } r \overset{a}{\rightsquigarrow} r' \textit{ and } s'\mathcal{P}r'.$$

The Figure 2.3.1 shows an example of $\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$. Algorithm 2.3.1 terminates whenever two consecutive partitions are equivalent. In such a partition two states (reachable from IS) belong to the same block iff they are bisimilar (using the standard notion of bisimilarity [41]).

Algorithm 2.3.1 $\mathtt{pr}(IS, \rightsquigarrow)$

Initialization

1. $IS^{\star}_{\rightsquigarrow}$ is the set of all states reachable from IS using $\rightsquigarrow$,
2. $\mathcal{P}^0 := IS^{\star}_{\rightsquigarrow}$,

Iteration $\mathcal{P}^{n+1} := \mathbf{F}_{\rightsquigarrow}(\mathcal{P}^n)$ as in Definition 2.3.2
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

Proposition 2.3.3. *If $IS^{\star}_{\rightsquigarrow}$ is finite then Algorithm 2.3.1 terminates and the resulting partition equates all the standard strongly bisimilar states (Definition 2.3.1).*

2.4 Concurrent Constraint Programming (CCP)

We dedicate this section entirely to presenting the Concurrent Constraint Programming (CCP) formalism. We begin this section by recalling the notion of constraint system. We then introduce the syntax of CCP, its operational and labeled semantics as well as several notions of equivalence for CCP.

2.4.1 Constraint Systems

The CCP model is parametric in a *constraint system (cs)* specifying the structure and interdependencies of the information that processes can ask or add to a *central shared store*. This information is represented as assertions traditionally referred to as *constraints*.

Following [21, 40] we regard a cs as a complete algebraic lattice (Definitions 2.1.8 and 2.1.9) in which the ordering $\sqsubseteq$ is the reverse of an entailment relation: $c \sqsubseteq d$ means d *entails* c, i.e., d contains "more information" than c. The top element $false$ represents inconsistency, the bottom element $true$ is the empty constraint, and the *least upper bound* (lub) $\sqcup$ is the join of information.

Definition 2.4.1 (Constraint Systems)**.** *A constraint system (cs)* **C** *is a complete algebraic lattice* $(Con, Con_0, \sqsubseteq, \sqcup, true, false)$ *where* Con*, the set of constraints, is a partially ordered set w.r.t.* $\sqsubseteq$*,* Con_0 *is the subset of* compact *elements of* Con*,*

$\sqcup$ is the lub operation defined on all subsets, and $true$, $false$ are the least and greatest elements of Con, respectively.

Recall that **C** is a *complete lattice* if every subset of Con has a least upper bound in Con. An element $c \in Con$ is *compact* if for any directed subset D of Con, $c \sqsubseteq \bigsqcup D$ implies $c \sqsubseteq d$ for some $d \in D$. **C** is *algebraic* if each element $c \in Con$ is the least upper bound of the compact elements below c.

In order to model *hiding* of local variables and *parameter passing*, in [63] the notion of constraint system is enriched with *cylindrification operators* and *diagonal elements*, concepts borrowed from the theory of cylindric algebras [45].

Let us consider a (denumerable) set of variables Var with typical elements $x, y, z, \ldots$ and let us define $\exists_{Var}$ as the family of operators and D_{Var} as the set as follows:

$$\exists_{Var} = \{\exists_x \mid x \in Var\} \text{ (\textit{cylindric operators})}$$

$$D_{Var} = \{d_{xy} \mid x, y \in Var\} \text{ (\textit{diagonal elements})}$$

A *cylindric constraint system* over a set of variables Var is a constraint system whose underlying support set $Con \supseteq D_{Var}$ is closed under the cylindric operators $\exists_{Var}$ and quotiented by Axioms $\mathsf{C1} - \mathsf{C4}$, and whose ordering $\sqsubseteq$ satisfies Axioms $\mathsf{C5} - \mathsf{C7}$:

C1. $\exists_x \exists_y c = \exists_y \exists_x c$	C2. $d_{xx} = true$
C3. if $z \neq x, y$ then $d_{xy} = \exists_z (d_{xz} \sqcup d_{zy})$	C4. $\exists_x (c \sqcup \exists_x d) = \exists_x c \sqcup \exists_x d$
C5. $\exists_x c \sqsubseteq c$	C6. *if* $c \sqsubseteq d$ *then* $\exists_x c \sqsubseteq \exists_x d$
C7. if $x \neq y$ then $c \sqsubseteq d_{xy} \sqcup \exists_x (c \sqcup d_{xy})$	

where c and d indicate compact constraints, and $\exists_x c \sqcup d$ stands for $(\exists_x c) \sqcup d$. For our purposes, it is enough to think of the operator $\exists_x$ as *existential quantifier* and the constraint d_{xy} as the equality $x = y$.

Cylindrification and diagonal elements allow us to model the variable renaming of a formula ϕ; in fact, by the aforementioned axioms, we have that the formula $\exists_x (d_{xy} \sqcup \phi)$ can be depicted as the formula $\phi[y/x]$, i.e., the formula obtained from ϕ by replacing all free occurrences of x by y.

We assume notions of *free variable* and of *substitution* that satisfy the follow-

ing conditions, where $c[y/x]$ is the constraint obtained by substituting x by y in c and $fv(c)$ is the set of free variables of c:

(1) if $y \notin fv(c)$ then $(c[y/x])[x/y] = c$;

(2) $(c \sqcup d)[y/x] = c[y/x] \sqcup d[y/x]$;

(3) $x \notin fv(c[y/x])$;

(4) $fv(c \sqcup d) = fv(c) \cup fv(d)$.

We now illustrate a constraint system for linear-order arithmetic.

Example 2.4.2 (A Constraint System of Linear Order Arithmetic)**.** *Consider the following syntax:*

$$\phi, \psi \ldots := \ t = t' \mid t > t' \mid \phi \ \vee \ \psi \mid \neg\phi$$

where the terms t, t' can be elements of a set of variables Var, or constant symbols $0, 1, \ldots$. Assume an underlying first-order structure of linear-order arithmetic with the obvious interpretation in the natural numbers ω of $=, >$ and the constant symbols.

A variable assignment is a function $\mu : Var \longrightarrow \omega$. We use $\mathcal{A}$ to denote the set of all assignments; $\mathcal{P}(X)$ to denote the powerset of a set X, $\emptyset$ the empty set and $\cap$ the intersection of sets. We use $\mathcal{M}(\phi)$ to denote the set of all assignments that satisfy *the formula ϕ, where the definition of* satisfaction *is as expected.*

We can now introduce our constraint system *as follows: the set of constraints is $\mathcal{P}(\mathcal{A})$, and define $c \sqsubseteq d$ iff $c \supseteq d$. The constraint $false$ is $\emptyset$, while $true$ is $\mathcal{A}$. Given two constraints c and d, $c \sqcup d$ is the intersection $c \cap d$. By abusing the notation, we will often use a formula ϕ to denote the corresponding constraint, i.e., the set of all assignments satisfying ϕ. E.g. we use $x > 1 \sqsubseteq x > 5$ to mean $\mathcal{M}(x > 1) \sqsubseteq \mathcal{M}(x > 5)$. For this constraint system one can show that e is a compact constraint (i.e., e is in Con_0) iff e is a co-finite set in $\mathcal{A}$ (i.e., iff the complement of e in $\mathcal{A}$ is a finite set). For example, $x > 10 \wedge y > 42$ is a compact constraint for $Var = \{x, y\}$.*

From this structure, let us now define the cylindric constraint system $\mathcal{S}$ *as follows. We say that an assignment μ' is* an x-variant *of μ if $\forall y \neq x$, $\mu(y) = \mu'(y)$.*

Given $x \in Var$ and $c \in \mathcal{P}(\mathcal{A})$, the constraint $\exists_x c$ is the set of assignments μ such that exists $\mu' \in c$ that is an x-variant of μ. The diagonal element d_{xy} is $x = y$.

Remark 2.4.3. *We shall assume that the constraint system is well-founded. Namely, there is no infinite strictly descending chain such that $c_1 \sqsupset c_2 \sqsupset \ldots$. Well-foundedness is needed for technical reasons in the definition of the labeled transition semantics in Section 2.4.6. Moreover, for practical reasons, we assume that the ordering of the constraint system $\sqsubseteq$ is decidable.*

2.4.2 Syntax of CCP

Let $\mathbf{C} = (Con, Con_0, \sqsubseteq, \sqcup, true, false)$ be a constraint system. The CCP processes are given by the following syntax:

$$P, Q, \ldots ::= \mathbf{tell}(c) \mid \sum_{i \in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i \mid P \parallel Q \mid \exists_x P \mid p(\vec{z})$$

where I is a finite set of indexes and $c, c_i \in Con_0$. We use $Proc$ to denote the set of all processes.

Finite processes. Intuitively, the tell process $\mathbf{tell}(c)$ adds c to the global store. The addition is performed regardless of the generation of inconsistent information. The process $P \parallel Q$ stands for the *parallel execution* of P and Q.

The guarded-choice $\sum_{i \in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i$ where I is a finite set of indexes, represents a process that can *nondeterministically* choose one of the P_j (with $j \in I$) whose corresponding guard constraint c_j is entailed by the store. The chosen alternative, if any, precludes the others. We shall often write

$$\mathbf{ask}\ (c_{i_1})\ \rightarrow\ P_{i_1} + \ldots + \mathbf{ask}\ (c_{i_n})\ \rightarrow\ P_{i_n}$$

if $I = \{i_1, \ldots, i_n\}$. If no ambiguity arises, we shall omit the "$\mathbf{ask}(c) \rightarrow$" when $c = \mathit{true}$. The *blind-choice* process $\sum_{i \in I} \mathbf{ask}\ (\mathit{true})\ \rightarrow\ P_i$, for example, can be written $\sum_{i \in I} P_i$. We shall omit the "$\sum_{i \in I}$" when I is a singleton. We use **stop** as an abbreviation of the empty summation $\sum_{i \in \emptyset} P_i$.

$\exists_x$ is a *hiding operator*, namely it indicates that in $\exists_x P$ the variable x is *local*

to P. The occurrences of x in $\exists_x P$ are said to be bound. The bound variables of P, $bv(P)$, are those with a bound occurrence in P, and its free variables, $fv(P)$, are those with an unbound occurrence[1].

Infinite processes. To specify infinite behavior, CCP provides parametric process definitions. A process $p(\vec{z})$ is said to be a *procedure call* with identifier p and actual parameters $\vec{z}$. We presuppose that for each procedure call $p(z_1 \ldots z_m)$ there exists a unique *procedure definition* possibly *recursive*, of the form $p(x_1 \ldots x_m) \stackrel{\texttt{def}}{=} P$ where $fv(P) \subseteq \{x_1, \ldots, x_m\}$. Furthermore we require recursion to be *guarded*: I.e., each procedure call within P must occur within an ask process. The behavior of $p(z_1 \ldots z_m)$ is that of $P[z_1 \ldots z_m / x_1 \ldots x_m]$, i.e., P with each x_i replaced with z_i (applying α-conversion to avoid clashes). We shall use $\mathcal{D}$ to denote the set of all process definitions.

Remark 2.4.4 (Choice-free fragment of CCP)**.** *Henceforth, we use CCP\+ to refer to the fragment of CCP without nondeterministic choice. More precisely CCP\+ processes are those in which every occurrence of* $\sum_{i \in I} \mathbf{ask}\ (c_i) \ \rightarrow \ P_i$ *has its index set* I *of cardinality 0 or 1.*

2.4.3 Reduction Semantics

A configuration is a pair $\langle P, d \rangle$ representing a *state* of a system; d is a constraint representing the global store, and P is a process, i.e., a term of the syntax given in the previous section. We use $Conf$ with typical elements $\gamma, \gamma', \ldots$ to denote the set of all configurations. We will use $Conf_{\texttt{CCP}\backslash+}$ for the configurations whose processes are in the CCP\+ fragment.

The operational semantics of CCP is given by an *unlabeled* transition relation between configurations: a transition $\gamma \longrightarrow \gamma'$ intuitively means that the configuration γ can reduce to γ'. We call these kind of unlabeled transitions *reductions* and we use $\longrightarrow^*$ to denote the reflexive and transitive closure of $\longrightarrow$.

Formally, the reduction semantics of CCP is given by the relation $\longrightarrow$ defined in Table 2.4.1. Rules R1 and R2 are easily seen to realize the intuitions described

[1]Notice that we also defined $fv(.)$ on constraints in the previous section.

$$\textsf{R1}\ \langle \mathbf{tell}(c), d\rangle \longrightarrow \langle \mathbf{stop}, d \sqcup c\rangle \qquad \textsf{R2}\ \frac{\langle P, d\rangle \longrightarrow \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \longrightarrow \langle P' \parallel Q, d'\rangle}$$

$$\textsf{R3}\ \frac{j \in I \text{ and } c_j \sqsubseteq d}{\langle \sum_{i\in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i, d\rangle \longrightarrow \langle P_j, d\rangle} \qquad \textsf{R4}\ \frac{\langle P, e \sqcup \exists_x d\rangle \longrightarrow \langle P', e' \sqcup \exists_x d\rangle}{\langle \exists_x^e P, d\rangle \longrightarrow \langle \exists_x^{e'} P', d \sqcup \exists_x e'\rangle}$$

$$\textsf{R5}\ \frac{\langle P[\vec{z}/\vec{x}], d\rangle \longrightarrow \gamma'}{\langle p(\vec{z}), d\rangle \longrightarrow \gamma'} \text{ where } p(\vec{x}) \stackrel{\texttt{def}}{=} P \text{ is a process definition in } \mathcal{D}$$

Table 2.4.1: Reduction semantics for CCP (symmetric rule for R2 is omitted). $\mathcal{D}$ is the set of process definitions.

in Section 2.4.2. Rule R3 states that $\sum_{i\in I}\ \mathbf{ask}\ (c_i)\ \rightarrow\ P_i$ can evolve to P_j whenever the global store d entails c_j and $j \in I$.

Rule R4 is somewhat more involved. First we extend the syntax by introducing a process $\exists_x^e P$ representing the evolution of a process of the form $\exists_x P$, where e is the local information (*local store*) produced during this evolution. The process $\exists_x P$ can be seen as a particular case of $\exists_x^e P$: it represents the situation in which the local store is empty. Namely, $\exists_x P = \exists_x^{true} P$.

Intuitively, $\exists_x^e P$ behaves like P, except that the variable x possibly present in P must be considered local, and that the information present in e has to be taken into account. It is convenient to distinguish between the *external* and the *internal* points of view. From the internal point of view, the variable x, possibly occurring in the global store d, is hidden. This corresponds to the usual scoping rules: the x in d is *global*, hence "covered" by the local x. Therefore, P has no access to the information on x in d, and this is achieved by filtering d with $\exists_x$. Furthermore, P can use the information (which may also concern the local x) that has been produced locally and accumulated in e. In conclusion, if the visible store at the external level is d, then the store that is visible internally by P is $e \sqcup \exists_x d$. Now, if P is able to make a step, thus reducing to P' and transforming the local store into e', what we see from the external point of view is that the process is transformed into $\exists_x^{e'} P'$, and that the information $\exists_x e$ present in the global store is transformed

into $\exists_x e'$. To show how this works we show an instructive example.

Example 2.4.5. *Consider the constraint system from Example 2.4.2 and let* $Var = \{x\}$. *Let* $P = \exists_x^e(\mathbf{ask}\ (x > 10)\ \rightarrow\ Q)$ *with local store* $e = x > 42$, *and global store* $d = x > 2$.

$$\textsf{R3}\quad \frac{(x > 10) \sqsubseteq e \sqcup \exists_x d = (x > 42 \sqcup \exists_x (x > 2))}{\textsf{R4}\quad \dfrac{\langle \mathbf{ask}\ (x > 10)\ \rightarrow\ Q, e \sqcup \exists_x d\rangle \longrightarrow \langle Q, e \sqcup \exists_x d\rangle}{\langle \exists_x^{x>42}(\mathbf{ask}\ (x > 10)\ \rightarrow\ Q), x > 2\rangle \longrightarrow \langle \exists_x^e Q, d \sqcup \exists_x e\rangle}}$$

Note that the x *in* d *is hidden, by using existential quantification in the reduction obtained by Rule* R3. *This expresses that the* x *in* d *is different from the one bound by the local process. Otherwise the ask process* $\mathbf{ask}\ (x > 10)\ \rightarrow\ Q$ *would not be executed since the guard is not entailed by the global store* d. *Rule* R3 *applies since* $(x > 10) \sqsubseteq e \sqcup \exists_x d$. *Note that the free* x *in* $e \sqcup \exists_x d$ *is hidden in the global store, i.e.* $d \sqcup \exists_x e$, *to indicate that is different from the global* x.

Finally, notice that in CCP\+ configurations are *confluent* in the following sense.

Proposition 2.4.6 (Confluence [63])**.** *Let* $\gamma \in Conf_{\texttt{CCP}\backslash+}$. *If* $\gamma \longrightarrow^* \gamma_1$ *and* $\gamma \longrightarrow^* \gamma_2$ *then there exists* γ' *such that* $\gamma_1 \longrightarrow^* \gamma'$ *and* $\gamma_2 \longrightarrow^* \gamma'$.

The proposition above will be a cornerstone for the results we shall obtain in CCP\+.

2.4.4 Barbed Semantics and Barbed Bisimilarity

In [6], the authors introduced a *barbed semantics* for CCP. Barbed equivalences have been introduced in [44] for CCS, and have become a classical way to define the semantics of formalisms equipped with unlabeled reduction semantics. Intuitively, *barbs* are basic observations (predicates) on the states of a system. In the case of CCP, barbs are taken from the underlying set Con_0 of the constraint system.

Definition 2.4.7 (Barbs)**.** *A configuration* $\gamma = \langle P, d\rangle$ *is said to satisfy the barb* c, *written* $\gamma \downarrow_c$, *iff* $c \in Con_0$ *and* $c \sqsubseteq d$. *Similarly,* γ *satisfies a weak barb* c, *written* $\gamma \Downarrow_c$, *iff there exists* γ' *s.t.* $\gamma \longrightarrow^* \gamma' \downarrow_c$.

Example 2.4.8. *Consider the constraint system from Example 2.4.2 and let* $Vars = \{x\}$. *Let* $\gamma = \langle \mathbf{ask}\ (x > 10) \rightarrow \mathbf{tell}(x > 42), x > 10\rangle$. *We have* $\gamma \downarrow_{x>5}$ *since* $(x > 5) \sqsubseteq (x > 10)$ *and* $\gamma \Downarrow_{x>42}$ *since* $\gamma \longrightarrow \langle \mathbf{tell}(x > 42), x > 10\rangle \longrightarrow \langle \mathbf{stop}, (x > 42)\rangle \downarrow_{x>42}$.

In this context, the equivalence proposed is the *saturated bisimilarity* [14, 12]. Intuitively, in order for two states to be saturated bisimilar, then (i) they should expose the same barbs, (ii) whenever one of them moves then the other should reply and arrive at an equivalent state (i.e. follow the bisimulation game), (iii) they should be equivalent under all the possible contexts of the language.

Using this idea, in [6], the authors propose a saturated bisimilarity for CCP where condition (iii) requires the bisimulations to be *upward closed* instead of closing under any process context. Recall that a process context C is a term with a single hole $\bullet$ such that if we replace $\bullet$ with a process P, we obtain a process term $C[P]$. For example, for the parallel context $C = \bullet \parallel R$ we obtain $C[P] = P \parallel R$.

Definition 2.4.9 (Saturated Barbed Bisimilarity)**.** *A saturated barbed bisimulation is a symmetric relation* $\mathcal{R}$ *on configurations s.t. whenever* $(\gamma_1, \gamma_2) \in \mathcal{R}$ *with* $\gamma_1 = \langle P, c\rangle$ *and* $\gamma_2 = \langle Q, d\rangle$ *implies that:*

(i) if $\gamma_1 \downarrow_e$ *then* $\gamma_2 \downarrow_e$,

(ii) if $\gamma_1 \longrightarrow \gamma_1'$ *then there exists* γ_2' *s.t.* $\gamma_2 \longrightarrow \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$,

(iii) for every $a \in Con_0$, $(\langle P, c \sqcup a\rangle, \langle Q, d \sqcup a\rangle) \in \mathcal{R}$.

We say that γ_1 *and* γ_2 *are saturated barbed bisimilar (*$\gamma_1 \dot{\sim}_{sb} \gamma_2$*) if there is a saturated barbed bisimulation* $\mathcal{R}$ *s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$. *We shall write* $P \dot{\sim}_{sb} Q$ *iff* $\langle P, true\rangle \dot{\sim}_{sb} \langle Q, true\rangle$.

Weak saturated barbed bisimilarity ($\dot{\approx}_{sb}$) is obtained from Definition 2.4.9 by replacing the strong barbs in condition *(i)* for its weak version ($\Downarrow$) and the transitions in condition *(ii)* for the reflexive and transitive closure of the transition relation ($\longrightarrow^*$).

Definition 2.4.10 (Weak Saturated Barbed Bisimilarity)**.** *A weak saturated barbed bisimulation is a symmetric relation* $\mathcal{R}$ *on configurations s.t. whenever* $(\gamma_1, \gamma_2) \in \mathcal{R}$ *with* $\gamma_1 = \langle P, c\rangle$ *and* $\gamma_2 = \langle Q, d\rangle$ *implies that:*

(i) if $\gamma_1 \Downarrow_e$ *then* $\gamma_2 \Downarrow_e$,

(ii) if $\gamma_1 \longrightarrow^* \gamma_1'$ *then there exists* γ_2' *s.t.* $\gamma_2 \longrightarrow^* \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$,

(iii) for every $a \in Con_0$, $(\langle P, c \sqcup a\rangle, \langle Q, d \sqcup a\rangle) \in \mathcal{R}$.

We say that γ_1 *and* γ_2 *are weak saturated barbed bisimilar (*$\gamma_1 \dot{\approx}_{sb} \gamma_2$*) if there exists a weak saturated barbed bisimulation* $\mathcal{R}$ *s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$. *We shall write* $P \dot{\approx}_{sb} Q$ *iff* $\langle P, true\rangle \dot{\approx}_{sb} \langle Q, true\rangle$.

We now illustrate $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$ with the following two examples.

Example 2.4.11. *Consider the constraint system from Example 2.4.2. Let* $Vars = \{x\}$. *Take* $P = \mathbf{ask}\ (x > 5) \rightarrow \mathbf{stop}$ *and* $Q = \mathbf{ask}\ (x > 7) \rightarrow \mathbf{stop}$. *One can check that* $P \not\sim_{sb} Q$ *since* $\langle P, x > 5\rangle \longrightarrow$, *while* $\langle Q, x > 5\rangle \not\longrightarrow$. *Then consider* $\langle P + Q, true\rangle$ *and observe that* $\langle P + Q, true\rangle \dot{\sim}_{sb} \langle P, true\rangle$. *Indeed, for all constraints* e, *s.t.* $x > 5 \sqsubseteq e$, *both the configurations evolve into* $\langle \mathbf{stop}, e\rangle$, *while for all* e *s.t.* $x > 5 \not\sqsubseteq e$, *both configurations cannot proceed. Since* $x > 5 \sqsubseteq x > 7$, *the behavior of* Q *is somehow absorbed by the behavior of* P.

Example 2.4.12. *Consider* P *and* Q *as in Example 2.4.11. We shall prove that* $P \dot{\approx}_{sb} Q$. *First notice that* $\langle P, true\rangle \not\longrightarrow$ *and also* $\langle Q, true\rangle \not\longrightarrow$. *Moreover, since none of the processes has the ability of adding information to the store, then for all* $a \in Con_0$, $\langle P, a\rangle \Downarrow_e$ *iff* $e \sqsubseteq a$ *iff* $\langle Q, a\rangle \Downarrow_e$. *Recall from Example 2.4.11 that* $P \dot{\not\sim}_{sb} Q$ *since an observer can plug* P *into* $x > 5$ *and observe a reduction, which cannot be observed with* Q. *Instead* $P \dot{\approx}_{sb} Q$ *since, intuitively, the discriminating power of* $\dot{\approx}_{sb}$ *cannot observe any reductions.*

2.4.5 Observational Equivalence

In this section we shall define the standard notion of *observational equivalence* ($\sim_o$) for CCP first introduced in [63]. Intuitively, two CCP programs are observationally equivalent if each one can be replaced with the other in any CCP context and produce the same final result. In order to define $\sim_o$ we need to first talk about CCP computations and fairness.

The notion of *fairness* is central to the definition of observational equivalence for CCP. We introduce this notion following [24]. First notice that any derivation

of a transition involves an application of R1 or R3. Now we say that P is *active* in a transition $t = \gamma \longrightarrow \gamma'$ if there exists a derivation of t where rule R1 or R3 is used to produce a transition of the form $\langle P, d\rangle \longrightarrow \gamma''$. Moreover, we say that P is *enabled* in γ if there exists γ' such that P is active in $\gamma \longrightarrow \gamma'$.

Definition 2.4.13 (Fair Computation)**.** *A computation $\gamma_0 \longrightarrow \gamma_1 \longrightarrow \gamma_2 \longrightarrow \ldots$ is said to be* fair *if for each process enabled in some γ_i there exists $j \geq i$ such that the process is active in γ_j.*

Example 2.4.14. *Consider the following process definition:*

$$\mathsf{inf}(x, y) \stackrel{\mathtt{def}}{=} \mathbf{tell}(x > 10) \parallel \mathbf{tell}(y > 20) \parallel \mathsf{inf}(x, y)$$

Now consider the following infinite computation:

$$\begin{aligned}
&\xi = \langle \mathsf{inf}(x, y), \mathit{true}\rangle \longrightarrow \\
&\quad \langle \mathbf{tell}(x > 10) \parallel \mathsf{inf}(x, y), y > 20\rangle \longrightarrow \\
&\quad \langle \mathbf{tell}(x > 10) \parallel \mathbf{tell}(x > 10) \parallel \mathsf{inf}(x, y), y > 20\rangle \longrightarrow \\
&\quad \langle \mathbf{tell}(x > 10) \parallel \mathbf{tell}(x > 10) \parallel \mathbf{tell}(x > 10) \parallel \mathsf{inf}(x, y), y > 20\rangle \longrightarrow \ldots
\end{aligned}$$

Note that ξ is not fair since $x > 10$ is never added to the store. For ξ to be fair the computation should be, for instance, of the form:

$$\begin{aligned}
&\xi = \langle \mathsf{inf}(x, y), \mathit{true}\rangle \longrightarrow \\
&\quad \langle \mathbf{tell}(x > 10) \parallel \mathsf{inf}(x, y), y > 20\rangle \longrightarrow \\
&\quad \langle \mathsf{inf}(x, y), x > 10 \sqcup y > 20\rangle \longrightarrow \\
&\quad \langle \mathbf{tell}(y > 20) \parallel \mathsf{inf}(x, y), x > 10 \sqcup y > 20\rangle \longrightarrow \\
&\quad \langle \mathsf{inf}(x, y), x > 10 \sqcup y > 20\rangle \longrightarrow \ldots
\end{aligned}$$

In other words, each of the tell operations has to be performed eventually.

Note that a finite fair computation is guaranteed to be *maximal*, namely no outgoing transitions are possible from its last configuration.

The standard notion of observables for CCP are the *results* computed by a process for a given initial store, where a result of a computation is defined as the

least upper bound of all the stores occurring in the computation, which, due to the monotonic properties of CCP, form an increasing chain. More formally:

Definition 2.4.15 (Result of a computation)**.** *Given a finite or infinite computation ξ of the form:*

$$\xi = \langle Q_0, d_0\rangle \longrightarrow \langle Q_1, d_1\rangle \longrightarrow \langle Q_2, d_2\rangle \longrightarrow \dots$$

The result of ξ, denoted by $\mathsf{Result}(\xi)$*, is the constraint* $\bigsqcup_i d_i$.

Note that for a finite computation the result coincides with the store of the last configuration. In the case of CCP\+, because of the confluence property, all the fair computations of a configuration have the same result as stated by the following theorem from [63].

Proposition 2.4.16 ([63])**.** *Let γ be CCP\+ configuration and let ξ_1 and ξ_2 be two computations of γ. If ξ_1 and ξ_2 are fair, then* $\mathsf{Result}(\xi_1) = \mathsf{Result}(\xi_2)$.

Before introducing the notion of observational equivalence we need to fix some notation. Below we define the set of possible computations of a given configuration.

Definition 2.4.17 (Set of Computations)**.** *The set of computations starting from γ, denoted* $\mathsf{Comp}(\gamma)$*, is defined as:*

$$\mathsf{Comp}(\gamma) = \{\xi \mid \xi = \gamma \longrightarrow \gamma' \longrightarrow \gamma'' \longrightarrow \dots\}$$

Now we introduce the notion of observables. Intuitively, the set of observables of γ is the set of results of the fair computations starting from γ.

Definition 2.4.18 (Observables)**.** *Let $\mathcal{O} : Proc \to Con_0 \to 2^{Con}$ be given by:*

$$\mathcal{O}(P)(d) = \{e \mid \xi \in \mathsf{Comp}(\langle P, d\rangle), \xi \textit{ is fair and } \mathsf{Result}(\xi) = e\}.$$

Using these elements we define the notion of observational equivalence. Two configurations are deemed equivalent if they have the same set of observables for any given store.

Definition 2.4.19 (Observational equivalence)**.** *We say that P and Q are observational equivalent, written $P \sim_o Q$, iff $\mathcal{O}(P) = \mathcal{O}(Q)$.*

Notice that in the case of CCP\+, as defined in [63], the set of observables for a given input is a singleton because of Proposition 2.4.16.

Remark 2.4.20. *Let $\langle P, d\rangle \in Conf_{\texttt{CCP}\backslash+}$. Note that $\mathcal{O} : Proc \to Con_0 \to Con$ because of Proposition 2.4.16 and it is defined as $\mathcal{O}(P)(d) = \mathsf{Result}(\xi)$ where ξ is any fair computation of $\langle P, d\rangle$.*

In [6] it was shown that, in CCP\+, weak saturated barbed bisimilarity and observation equivalence coincide. Recall that $P \dot{\approx}_{sb} Q$ stands for $\langle P, true\rangle \dot{\approx}_{sb} \langle Q, true\rangle$.

Proposition 2.4.21 ([6])**.** *Let P and Q be CCP\+ processes. Then $P \sim_o Q$ iff $P \dot{\approx}_{sb} Q$.*

2.4.6 Labeled Semantics

In the previous section we saw that, in CCP\+, $\dot{\approx}_{sb}$ is *fully abstract* with respect to the standard observational equivalence from [63] (See proposition 2.4.21). Unfortunately, the quantification over all constraints in condition (iii) of Definition 2.4.9 and Definition 2.4.10 makes it hard to check $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$, since one should check infinitely many constraints. In order to avoid this problem the authors in [6] introduced a labeled transition semantics where labels are constraints.

In a labeled transition of the form

$$\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle$$

the label $\alpha \in Con_0$ represents a *minimal* piece of information (from the environment) that needs to be added to the store d to reduce from $\langle P, d\rangle$ to $\langle P', d'\rangle$, i.e., $\langle P, d \sqcup \alpha\rangle \longrightarrow \langle P', d'\rangle$. As a consequence, the transitions labeled with the constraint $true$ are in one to one correspondence with the reductions defined in the previous section. For this reason, hereafter we will sometimes write $\longrightarrow$ to mean $\xrightarrow{true}$. Before formally introducing the labeled semantics, we fix some notation.

Notation 2.4.22. *We will use $\rightsquigarrow$ to denote a generic transition relation on the state space $Conf$ and labels Con_0. Also in this case $\rightsquigarrow$ mean $\overset{true}{\rightsquigarrow}$.*

$$\textsf{LR1}\ \langle \mathbf{tell}(c), d\rangle \xrightarrow{true} \langle \mathbf{stop}, d \sqcup c\rangle \qquad \textsf{LR2}\ \frac{\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \xrightarrow{\alpha} \langle P' \parallel Q, d'\rangle}$$

$$\textsf{LR3}\ \frac{j \in I \text{ and } \alpha \in \min\{a \in Con_0 \mid c_j \sqsubseteq d \sqcup a\}}{\langle \sum_{i\in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i, d\rangle \xrightarrow{\alpha} \langle P_j, d \sqcup \alpha\rangle}$$

$$\textsf{LR4}\ \frac{\langle P[z/x], e[z/x] \sqcup d\rangle \xrightarrow{\alpha} \langle P', e' \sqcup d \sqcup \alpha\rangle}{\langle \exists^{e}_{x} P, d\rangle \xrightarrow{\alpha} \langle \exists^{e'[x/z]}_{x} P'[x/z], \exists_x(e'[x/z]) \sqcup d \sqcup \alpha\rangle}$$

$$\text{with } x \notin fv(e'), z \notin fv(P) \cup fv(e \sqcup d \sqcup \alpha)$$

$$\textsf{LR5}\ \frac{\langle P[\vec{z}/\vec{x}], d\rangle \xrightarrow{\alpha} \gamma'}{\langle p(\vec{z}), d\rangle \xrightarrow{\alpha} \gamma'} \text{ where } p(\vec{x}) \stackrel{\texttt{def}}{=} P \text{ is a process definition in } \mathcal{D}$$

Table 2.4.2: Labeled semantics for CCP (symmetric rule for LR2 is omitted).

The LTS $(Conf, Con_0, \longrightarrow)$ is defined by the rules in Table 2.4.2. The rule LR3, for example, says that $\langle \sum_{i\in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i, d\rangle$ can evolve to $\langle P_j, d \sqcup \alpha\rangle$ if $j \in I$ and the environment provides a minimal constraint α that added to the store d entails the guard c_j, i.e., $\alpha \in \min\{a \in Con_0 \mid c_j \sqsubseteq d \sqcup a\}$. Notice that Remark 2.4.3 guarantees the existence of α. The rule LR4 follows the same approach as R4, however it uses variable substitution instead of hiding with the existential operator. Namely, in the antecedent derivation, the rule LR4 uses a fresh variable z that acts as a substitute for the free occurrences of (the local) x in P and its local store e. (Recall that $T[z/x]$ represents T with x replaced with z). This way we identify with z the free occurrences of x in P and e and avoid clashes with those in α and d.[2] The other rules are easily seen to realize the intuition given in Section 2.4.2.

Finally, we fix some notation for the set of reachable states of an initial set of configurations as follows.

[2] See [6] for a detailed explanation of the rule LR4.

Definition 2.4.23 (Reachable Configurations)**.** *Let IS be a set of initial configurations, then the set of reachable configurations starting from IS and using $\rightsquigarrow$ is defined as:*

$$\texttt{Config}_{\rightsquigarrow}(IS) = \{\gamma' \mid \exists \gamma \in IS \text{ s.t. } \gamma \overset{\alpha_1}{\rightsquigarrow} \ldots \overset{\alpha_n}{\rightsquigarrow} \gamma' \text{ for some } n \geq 0\}$$

2.4.7 Soundness and Completeness

In [6] it was proven that the labeled semantics is *sound* and *complete* w.r.t. the unlabeled one. Soundness states that $\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle$ corresponds to our intuition that if α is added to d, P can reach $\langle P', d'\rangle$.

Lemma 2.4.24 (Soundness of $\longrightarrow$, [6])**.** *If $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ then $\langle P, c \sqcup \alpha\rangle \longrightarrow \langle P', c'\rangle$.*

Completeness states that if we add a to (the store in) $\langle P, d\rangle$ and reduce to $\langle P', d'\rangle$, there exists a minimal piece of information α below a such that $\langle P, d\rangle \xrightarrow{\alpha} \langle P', d''\rangle$ with d'' below d'.

Lemma 2.4.25 (Completeness of $\longrightarrow$, [6])**.** *If $\langle P, c \sqcup a\rangle \longrightarrow \langle P', c'\rangle$ then there exist α and b such that $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c''\rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$.*

2.4.8 Strong and Weak Labeled Bisimilarity

Having defined the labeled transitions for CCP, we now proceed to define an equivalence from [6] that characterizes $\dot{\sim}_{sb}$ (and $\dot{\approx}_{sb}$) without the upward closure condition.

When defining bisimilarity over a labeled transition system, barbs are not usually needed because they can be somehow inferred by the labels of the transitions. For example in CCS, $P \downarrow_a$ iff $P \xrightarrow{a}$. The case of CCP is different: barbs cannot be removed from the definition of bisimilarity because they cannot be inferred by the transitions. In order to remove barbs from CCP, one could insert labels showing the store of processes (as in [62]) but this is contrary to the philosophy of "labels as minimal constraints". Then, one could be tempted to define the labeled bisimilarity in the standard way as follows.

Definition 2.4.26 (Syntactic bisimilarity)**.** *A syntactic bisimulation is a symmetric relation $\mathcal{R}$ on configurations such that whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$:*

(i) if $\gamma_1 \downarrow_e$ then $\gamma_2 \downarrow_e$,

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ then $\exists \gamma_2'$ s.t. $\gamma_2 \xrightarrow{\alpha} \gamma_2'$ and $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 and γ_2 are syntactically bisimilar, written $\gamma_1 \sim_S \gamma_2$, if there exists a syntactic bisimulation $\mathcal{R}$ such that $(\gamma_1, \gamma_2) \in \mathcal{R}$. We write $P \sim_S Q$ iff $\langle P, true\rangle \sim_S \langle Q, true\rangle$.

The bisimilarity above is called "syntactic", because it does not take into account the "real meaning" of the labels. This equivalence coincides with the one in [62] (apart from the fact that in the latter, barbs are implicitly observed by the transitions) and, from a more general point of view can be seen as an instance of bisimilarity in [38] (by identifying contexts with constraints). In [14], it is argued that the equivalence in [38] is often over-discriminating. This is also the case of CCP, given that syntactic bisimilarity would distinguish configurations which are in $\dot{\sim}_{sb}$, as illustrated by the next example.

Example 2.4.27. *Recall $P = \mathbf{ask}\ (x > 5) \rightarrow \mathbf{stop}$ and $Q = \mathbf{ask}\ (x > 7) \rightarrow \mathbf{stop}$ from Example 2.4.11. We saw that $\langle P + Q, true\rangle \dot{\sim}_{sb} \langle P, true\rangle$. However, $\langle P + Q, true\rangle \xrightarrow{x>7} \langle T, x > 7\rangle$, while $\langle P, true\rangle \not\xrightarrow{x>7}$.*

For this reason we need a notion of bisimilarity that captures the meaning of the labels. To give some intuition on how this will be achieved, let us recall that in $\langle P, c\rangle \xrightarrow{\alpha} \gamma'$ the label α represents *minimal* information from the environment that needs to be added to the store d to evolve from $\langle P, c\rangle$ into γ'. We do not require the transitions from $\langle Q, d\rangle$ to match α. Instead (ii) requires something weaker: If α is added to the store e, it should be possible to reduce into some γ'' that it is in bisimulation with γ'. This condition is weaker because α may not be a minimal piece of information allowing a transition from $\langle Q, d\rangle$ into a γ'' in the bisimulation, as shown in the previous example.

Definition 2.4.28 (Strong Bisimilarity)**.** *A strong bisimulation is a symmetric relation $\mathcal{R}$ on configurations such that whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, c\rangle$ and $\gamma_2 = \langle Q, d\rangle$:*

(i) if $\gamma_1 \downarrow_e$ *then* $\gamma_2 \downarrow_e$,

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ *then* $\exists \gamma_2'$ *s.t.* $\langle Q, d \sqcup \alpha \rangle \longrightarrow \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 *and* γ_2 *are strongly bisimilar, written* $\gamma_1 \dot{\sim} \gamma_2$*, if there exists a strong bisimulation* $\mathcal{R}$ *such that* $(\gamma_1, \gamma_2) \in \mathcal{R}$*. We write* $P \dot{\sim} Q$ *iff* $\langle P, true \rangle \dot{\sim} \langle Q, true \rangle$.

Weak bisimilarity ($\dot{\approx}$) is obtained from the definition above by replacing the strong barb requirement of γ_2 in condition *(i)* for its weak version ($\Downarrow$) and the transition of γ_2 in condition *(ii)* for the reflexive and transitive closure of the transition relation ($\longrightarrow^*$).

Definition 2.4.29 (Weak bisimilarity)**.** *A weak bisimulation is a symmetric relation* $\mathcal{R}$ *on configurations such that whenever* $(\gamma_1, \gamma_2) \in \mathcal{R}$ *with* $\gamma_1 = \langle P, c \rangle$ *and* $\gamma_2 = \langle Q, d \rangle$:

(i) if $\gamma_1 \downarrow_e$ *then* $\gamma_2 \Downarrow_e$,

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ *then* $\exists \gamma_2'$ *s.t.* $\langle Q, d \sqcup \alpha \rangle \longrightarrow^* \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 *and* γ_2 *are weakly bisimilar, written* $\gamma_1 \dot{\approx} \gamma_2$*, if there exists a weak bisimulation* $\mathcal{R}$ *such that* $(\gamma_1, \gamma_2) \in \mathcal{R}$*. We write* $P \dot{\approx} Q$ *iff* $\langle P, true \rangle \dot{\approx} \langle Q, true \rangle$.

To illustrate this definition consider the following example.

Example 2.4.30. *Let* $\gamma_1 = \langle \mathbf{tell}(true), true \rangle$ *and* $\gamma_2 = \langle \mathbf{ask}\,(c) \rightarrow \mathbf{tell}(d), true \rangle$*. We can show that* $\gamma_1 \dot{\approx} \gamma_2$ *when* $d \sqsubseteq c$*. Intuitively, this corresponds to the fact that the implication* $c \Rightarrow d$ *is equivalent to* $true$ *when* c *already entails* d*. The LTSs of* γ_1 *and* γ_2 *are the following:* $\gamma_1 \longrightarrow \langle \mathbf{stop}, true \rangle$ *and* $\gamma_2 \xrightarrow{c} \langle \mathbf{tell}(d), c \rangle \longrightarrow \langle \mathbf{stop}, c \rangle$*. It is now easy to see that the symmetric closure of the relation*

$$\mathcal{R} = \{(\gamma_2, \gamma_1),\ (\gamma_2, \langle \mathbf{stop}, true \rangle),\ (\langle \mathbf{tell}(d), c \rangle, \langle \mathbf{stop}, c \rangle), (\langle \mathbf{stop}, c \rangle, \langle \mathbf{stop}, c \rangle)\}$$

is a weak bisimulation as in Definition 2.4.29.

In [6] it is shown that strong and weak bisimilarity coincide with strong and weak saturated barbed bisimilarity respectively.

Proposition 2.4.31 ([6])**.** $\dot{\sim}_{sb} = \dot{\sim}$ *and* $\dot{\approx}_{sb} = \dot{\approx}$.

Notice that, by exploiting the labeled semantics, $\dot{\sim}$ and $\dot{\approx}$ avoid the upward closure from condition (iii) in $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$.

Chapter 3

Verifying Strong Bisimilarity in CCP

As we saw in Section 2.3 we can use the partition refinement algorithm (Algorithm 2.3.1) to verify whether two states are bisimilar. This approach works for the standard notion of bisimilarity where the defender replies by matching the exact action performed by the attacker, namely both the attacker and the defender produce a transition with the same label. However, in Section 2.4.8 we saw how this bisimulation game is too discriminating for CCP as evidenced by Example 2.4.27.

In this chapter we shall adapt Algorithm 2.3.1 to check strong saturated barbed bisimilarity ($\dot{\sim}_{sb}$) introduced in Definition 2.4.9. This chapter is divided in two sections. Section 3.1 concerns the definition of the partition refinement algorithm for CCP. The intuition of this algorithm, which is inspired on the abstract approach from [15], is to detect certain kind of transitions that carry enough information to reason about the behavior of the processes. As we shall see later on, these transitions will allow the matching of exact labels as in the standard partition refinement, however, to accomplish this task, some additional unreachable states may need to be added to the LTS of the input. Section 3.2 contains all the technical details regarding the correctness and complexity of the decision procedure defined in 3.1.

3.1 Partition Refinement for CCP

In this section we start by introducing the notion of derivation and domination between transitions. These concepts arise from the idea that certain transitions carry more information about the behavior of the program than others, i.e. they are *stronger* in a way that we shall explain later on. Next we shall use these notions to define in detail the partition refinement algorithm for CCP.

3.1.1 Derivation and Domination

In the case of CCP, syntactic bisimilarity is over-discriminating because not all the transitions are important regarding the behavior of a program. To better explain this, consider the following example:

Example 3.1.1. *Let* $P = \mathbf{ask}\ (x > 5)\ \rightarrow\ \mathbf{stop}$ *and* $Q = \mathbf{ask}\ (x > 7)\ \rightarrow\ \mathbf{stop}$ *as in Example 2.4.11 and consider the following transitions:*

$$(\mathbf{1})\langle P + Q, true\rangle \xrightarrow{x>5} \langle \mathbf{stop}, x > 5\rangle \quad (\mathbf{2})\langle P + Q, true\rangle \xrightarrow{x>7} \langle \mathbf{stop}, x > 7\rangle$$

Transition $(\mathbf{1})$ *means that for all constraints* e *s.t.* $x > 5 \sqsubseteq e$ *we have:*

$$(\mathbf{3})\langle P + Q, e\rangle \longrightarrow \langle \mathbf{stop}, e\rangle$$

while transition $(\mathbf{2})$ *means that the reduction* $(\mathbf{3})$ *is possible for all* e *s.t.* $x > 7 \sqsubseteq e$*. Since* $x > 5 \sqsubseteq x > 7$*, transition* $(\mathbf{2})$ *is "redundant", in the sense that its meaning is "logically derived" by transition* $(\mathbf{1})$*.*

The following two notions capture the intuition described above:

Definition 3.1.2 (Transition Derivation)**.** *Let* t *and* t' *be two transitions of the form* $t = (\gamma, \alpha, \langle P', c'\rangle)$ *and* $t' = (\gamma, \beta, \langle P', c''\rangle)$*. We say that* t *derives* t'*, written* $t \vdash_D t'$*, iff* $\alpha \sqsubseteq \beta$ *and* $c'' = c' \sqcup \beta$*.*

Now we introduce the concept of *domination*, which consists in strengthening the notion of derivation by requiring labels to be different.

Definition 3.1.3 (Transition Domination)**.** *Let t and t' be two transitions of the form $t = (\gamma, \alpha, \langle P', c' \rangle)$ and $t' = (\gamma, \beta, \langle P', c'' \rangle)$. We say that t dominates t', written $t \succ_D t'$, iff $t \vdash_D t'$ and $\alpha \neq \beta$.*

The intuition is that the transition t dominates t' iff t requires less information from the environment than t' does (hence $\alpha \sqsubset \beta$), and they end up in configurations which differ only by the additional information in β not present in α (hence $c'' = c' \sqcup \beta$).

One can verify in Example 3.1.1 that $(\mathbf{1}) \succ_D (\mathbf{2})$. Notice that in order to check if $\langle P + Q, true \rangle \dot{\sim}_{sb} \langle P, true \rangle$, we could first remove the redundant transition $(\mathbf{2})$ and then check syntactic bisimilarity $\sim_S$ (Definition 2.4.26).

More generally, a naive approach to compute $\dot{\sim}_{sb}$ would be to first remove all those transitions that can be dominated by others, and then apply the partition refinement algorithm (Algorithm 2.3.1).

Algorithm 3.1.1 `naive-pr-ccp`$(IS, \rightsquigarrow)$

Initialization

1. Compute $G = LTS_{\rightsquigarrow}(IS)$, i.e. all states reachable from IS using $\rightsquigarrow$,
2. $G' =$ `remDom`(G) where the graph `remDom`(G) results from removing from G the transitions that are dominated by another transition,
3. $\mathcal{P}^0 = \{B_1\} \ldots \{B_m\}$ is a partition of $IS^{\star}_{\rightsquigarrow}$ where γ and γ' are in B_i iff they satisfy the same barbs ($\downarrow_c$),

Iteration $\mathcal{P}^{n+1} := \mathbf{F}_{\rightsquigarrow}(\mathcal{P}^n)$ as in Definition 2.3.2
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

Notice that on top of the reachable states used in Algorithm 2.3.1, because of condition (i) in $\dot{\sim}_{sb}$ (Definition 2.4.9), the third step equates all the states that satisfy the same barbs.

Nevertheless, Algorithm 3.1.1 approach would fail as proven by the following proposition.

Counterexample 3.1.4. *There exist γ, γ' s.t. (i) $\gamma \dot{\sim}_{sb} \gamma'$, (ii) γ and γ' are not related in $\mathcal{P}$ where $\mathcal{P}$ is the output of* `naive-pr-ccp`$(\{\gamma, \gamma'\}, \longrightarrow)$ *in Algorithm 3.1.1.*

Proof. Let $P = \mathbf{ask}\ (x > 5)\ \rightarrow\ \mathbf{stop}$ and $Q = \mathbf{ask}\ (x > 7)\ \rightarrow\ \mathbf{stop}$ as in Example 2.4.11 and let $R = \mathbf{ask}\ (x > 1)\ \rightarrow\ (P + Q)$ and $S = \mathbf{ask}\ (x > 3)\ \rightarrow\ P$. Now take $\gamma = \langle R + S, true\rangle$ and $\gamma' = \langle R, true\rangle$. To prove (i), following the same reasoning as in Example 2.4.11 $\gamma \dot{\sim}_{sb} \gamma'$ since for all constraints e, s.t. $x > 1 \sqsubseteq e$, both the configurations evolve into $\langle \mathbf{stop}, e\rangle$, while for all e s.t. $x > 1 \not\sqsubseteq e$, both configurations cannot proceed. Since $x > 1 \sqsubseteq x > 3$, the behavior of S is somehow absorbed by the behavior of R.

As for (ii), consider the transitions of $\langle R + S, true\rangle$:

$$t_1 = \langle R + S, true\rangle \xrightarrow{x>1} \langle P, x > 1\rangle \quad t_2 = \langle R + S, true\rangle \xrightarrow{x>3} \langle P + Q, x > 3\rangle$$

Now notice that $t_1 \not\succ_D t_2$ since $tar(t_1) \neq tar(t_2)$, namely P is different from $P + Q$. Therefore both t_1 and t_2 would pass the test in step 2 of Algorithm 3.1.1. However, $\langle R, true\rangle \xrightarrow{x>1}$ but $\langle R, true\rangle \not\xrightarrow{x>3}$ thus in the final partition γ and γ' are not related. □

Note that in the theorem above t_1 does not dominate t_2 because P is syntactically different from $P + Q$. However, following Example 2.4.11, one can prove that $\langle P, x > 1\rangle \dot{\sim}_{sb} \langle P + Q, x > 3\rangle$. Thus t_2 is also "redundant", since its behavior "does not add anything" to the behavior of t_1.

In Definition 3.1.3 we have that t and t' end up in configurations whose processes are *syntactically identical* (i.e., P'). The following notion parameterizes the notion of derivation and dominance w.r.t. a relation on configurations $\mathcal{R}$ (rather than fixing it to the identity on configurations). Recall that for a given transition $t = (s, a, r)$ the source, the target and the label are $src(t) = s$, $tar(t) = r$ and $lab(t) = a$ respectively.

Definition 3.1.5 (Transition Derivation w.r.t. $\mathcal{R}$)**.** *We say that the transition t derives a transition t' w.r.t a relation on configurations $\mathcal{R}$, written $t \vdash_{\mathcal{R}} t'$, iff there exists t'' such that $t \vdash_D t''$, $lab(t'') = lab(t')$ and $tar(t'')\mathcal{R}\ tar(t')$.*

And similarly for the domination relation we have:

Definition 3.1.6 (Transition Domination w.r.t. $\mathcal{R}$ and Irredundant Transition w.r.t. $\mathcal{R}$)**.** *We say that the transition t dominates a transition t' w.r.t a relation on configurations $\mathcal{R}$, written $t \succ_{\mathcal{R}} t'$, iff there exists t'' such that $t \succ_D t''$, $lab(t'') = lab(t')$*

and $tar(t'')\mathcal{R}\ tar(t')$. *A transition is said to be* redundant *w.r.t. to* $\mathcal{R}$ *when it is dominated by another w.r.t.* $\mathcal{R}$*, otherwise it is said to be* irredundant *w.r.t. to* $\mathcal{R}$.

To better understand this definition consider the following examples.

Example 3.1.7. *Consider* P, Q, R *and* S *as in Counterexample 3.1.4. Take* $\mathcal{R} = \dot{\sim}_{sb}$ *and* t_1, t_2 *as follows:*

$$t_1 = \langle R + S, true\rangle \xrightarrow{x>1} \langle P, x > 1\rangle \quad t_2 = \langle R + S, true\rangle \xrightarrow{x>3} \langle P + Q, x > 3\rangle$$

Notice that $t_1 \succ_{\mathcal{R}} t_2$ *since* $x > 1 \sqsubset x > 3$ *and* $\langle P, x > 1\rangle \dot{\sim}_{sb} \langle P + Q, x > 3\rangle$.

Example 3.1.8. *Consider the following processes:*

$$Q_1 = (\mathbf{ask}\ (b)\ \rightarrow\ (\mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d))) \text{ and } Q_2 = (\mathbf{ask}\ (a)\ \rightarrow\ \mathbf{stop})$$

Now let $P = Q_1 + Q_2$ *where* $d \sqsubseteq c$ *and* $a \sqsubset b$. *Then take* $\gamma = \langle P, true\rangle$ *and consider the transitions* t *and* t' *as:*

$$t = \gamma \xrightarrow{a} \langle \mathbf{stop}, a\rangle \text{ and } t' = \gamma \xrightarrow{b} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), b\rangle$$

Finally, let $\mathcal{R} = \dot{\approx}_{sb}$ *and take* $t'' = (\gamma, b, \langle \mathbf{stop}, b\rangle)$. *One can check that* $t \succ_{\mathcal{R}} t'$ *as in Definition 3.1.6. Firstly,* $t \succ_D t''$ *follows from* $a \sqsubset b$. *Secondly, we know* $tar(t'')\mathcal{R}\ tar(t')$ *from Example 2.4.12, i.e.* $\langle \mathbf{stop}, b\rangle \dot{\approx}_{sb} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), b\rangle$ *since* $\langle \mathbf{stop}, true\rangle \dot{\approx}_{sb} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), true\rangle$.

3.1.2 The Algorithm

From the previous section we can see that we could compute $\dot{\sim}_{sb}$, by removing all those transitions that are redundant w.r.t. $\dot{\sim}_{sb}$. Unfortunately redundancy itself depends on $\dot{\sim}_{sb}$, therefore we must compute bisimilarity and redundancy *at the same time*.

To be able to check for redundant transitions we need not only the states reachable using $\longrightarrow$ but some additional *unreachable* states used specifically for detecting redundancy. More concretely, to compute $IS^{\star}_{\rightsquigarrow}$ we use the rules in Table

$$(\mathsf{IS}^{IS}_{\rightsquigarrow})\ \frac{\gamma \in IS}{\gamma \in IS^{\star}_{\rightsquigarrow}} \qquad (\mathsf{RS}^{IS}_{\rightsquigarrow})\ \frac{\gamma \in IS^{\star}_{\rightsquigarrow} \quad \gamma \overset{\alpha}{\rightsquigarrow} \gamma'}{\gamma' \in IS^{\star}_{\rightsquigarrow}}$$

$$(\mathsf{RD}^{IS}_{\rightsquigarrow})\ \frac{\gamma \in IS^{\star}_{\rightsquigarrow} \quad t_1 = \gamma \overset{\alpha}{\rightsquigarrow} \langle P_1, c_1\rangle \quad t_2 = \gamma \overset{\beta}{\rightsquigarrow} \langle P_2, c_2\rangle \quad \alpha \sqsubset \beta \quad c_2 = c_1 \sqcup \beta}{\langle P_1, c_2\rangle \in IS^{\star}_{\rightsquigarrow}}$$

Table 3.1.1: Rules for generating the states used in the partition refinement for CCP

3.1.1. Rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$ and $(\mathsf{RS}^{IS}_{\rightsquigarrow})$ say that all the reachable states using $\rightsquigarrow$ should be included, i.e., $\mathtt{Config}_{\rightsquigarrow}(IS) \subseteq IS^{\star}_{\rightsquigarrow}$.

The rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ adds the additional states needed to check redundancy. Consider the transitions $t_1 = \gamma \overset{\alpha}{\rightsquigarrow} \langle P_1, c_1\rangle$ and $t_2 = \gamma \overset{\beta}{\rightsquigarrow} \langle P_2, c_2\rangle$ with $\alpha \sqsubset \beta$ and $c_2 = c_1 \sqcup \beta$ in Rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$. Suppose that at some iteration of the partition refinement algorithm the current partition is $\mathcal{P}$ and that $\langle P_2, c_2\rangle \mathcal{P} \langle P_1, c_2\rangle$. Then, according to Definition 3.1.6 the transitions t_1 would dominate t_2 w.r.t $\mathcal{P}$. This makes t_2 redundant w.r.t $\mathcal{P}$. Since $\langle P_1, c_2\rangle$ may allow us to witness a potential redundancy of t_2, we include it in $IS^{\star}_{\rightsquigarrow}$ (and thus, from the definition of the initial partition $\mathcal{P}^0$, also in the block of $\mathcal{P}^0$ where $\langle P_2, c_2\rangle$ is).

Finally, in the case of CCP the refinement must be done only w.r.t. the *irredundant transitions*. Hence instead of using the function $\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$ of Algorithm 2.3.1, the partitions are refined by employing the function $\mathbf{IR}_{\rightsquigarrow}(\mathcal{P})$ defined as:

Definition 3.1.9 (Irredundant Refinement Function)**.** *Given a partition $\mathcal{P}$ we define $\mathbf{IR}_{\rightsquigarrow}(\mathcal{P})$ as follows: $\gamma_1\ \mathbf{IR}_{\rightsquigarrow}(\mathcal{P})\ \gamma_2$ iff*

if $\gamma_1 \overset{\alpha}{\rightsquigarrow} \gamma_1'$ is irredundant w.r.t. $\mathcal{P}$ then there exists γ_2' s.t. $\gamma_2 \overset{\alpha}{\rightsquigarrow} \gamma_2'$ and $\gamma_1' \mathcal{P} \gamma_2'$

See Figure 3.1.1 for an example of the use of $\mathbf{IR}_{\rightsquigarrow}(-)$. Using all these elements we can now define the partition refinement algorithm for CCP as follows:

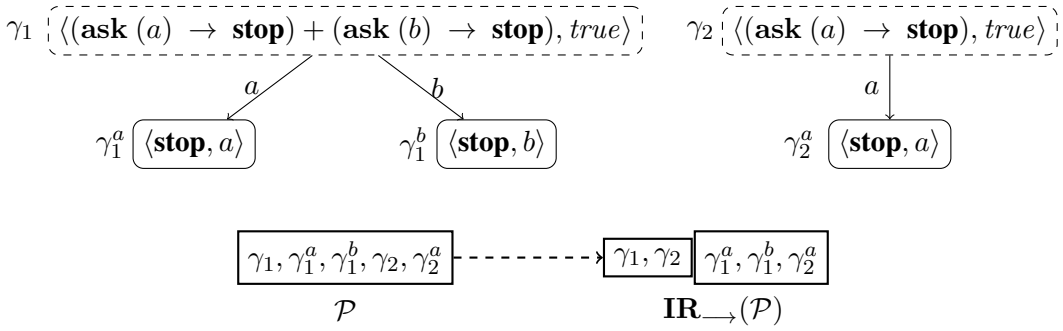

Figure 3.1.1: An example of the use of $\mathbf{IR}_{\longrightarrow}(\mathcal{P})$ as in Definition 3.1.9. Let $a \sqsubset b$, notice that γ_1 and γ_2 end up in the same block after the refinement since $\gamma_1 \xrightarrow{b} \gamma_1^b$ is a redundant transition w.r.t $\mathcal{P}$ hence it is not required that γ_2 matches it.

Algorithm 3.1.2 `pr-ccp`$(IS, \rightsquigarrow)$
Initialization

1. Compute $IS^{\star}_{\rightsquigarrow}$ with the rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ defined in Table 3.1.1,
2. $\mathcal{P}^0 = \{B_1\} \ldots \{B_m\}$ is a partition of $IS^{\star}_{\rightsquigarrow}$ where γ and γ' are in B_i iff they satisfy the same barbs ($\downarrow_c$),

Iteration $\mathcal{P}^{n+1} := \mathbf{IR}_{\rightsquigarrow}(\mathcal{P}^n)$ as in Definition 3.1.9
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

3.2 Correctness and Complexity

In this section we shall prove the correctness and efficiency of Algorithm 3.1.2. We do this by first introducing the notion of irredundant and symbolic bisimilarity to prove that Algorithm 3.1.2 computes $\dot{\sim}_{sb}$. Finally we discuss the complexity of Algorithm 3.1.2.

3.2.1 Irredundant and Symbolic Bisimilarity

Now we shall see that the irredundant transitions (Definition 3.1.6) carry enough information to check $\dot{\sim}_{sb}$. *Irredundant bisimilarity* follows the standard bisimulation game where labels need to be matched, however only the irredundant transitions must be considered.

Definition 3.2.1 (Irredundant Bisimilarity)**.** *An irredundant bisimulation is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ implies that:*

(i) if $\gamma_1 \downarrow_e$ then $\gamma_2 \downarrow_e$,

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ and it is irredundant in $\mathcal{R}$ then there exists γ_2' s.t. $\gamma_2 \xrightarrow{\alpha} \gamma_2'$ and $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 and γ_2 are irredundant bisimilar ($\gamma_1 \dot{\sim}_I \gamma_2$) if there exists an irredundant bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Notice that this bisimulation game is exactly what the $\mathbf{IR}_{\longrightarrow}(-)$ function in Definition 3.1.9 captures. To better understand the definition above consider the following example.

Example 3.2.2. *Let $P = \mathbf{ask}\ (x > 5) \rightarrow \mathbf{stop}$ and $Q = \mathbf{ask}\ (x > 7) \rightarrow \mathbf{stop}$ as in Example 2.4.11 and let $\mathcal{R} = \{(\langle P + Q, true\rangle, \langle P, true\rangle)\} \cup id$ where id is the identity relation. We can verify that $\mathcal{R}$ is an irredundant bisimulation to show that $\langle P + Q, true\rangle \dot{\sim}_I \langle P, true\rangle$. We take the pair $(\langle P + Q, true\rangle, \langle P, true\rangle)$. The first item in Definition 3.2.1 is obvious, and now notice that:*

$$\langle P + Q, true\rangle \xrightarrow{x>5} \langle \mathbf{stop}, x > 5\rangle \succ_{\mathcal{R}} \langle P + Q, true\rangle \xrightarrow{x>7} \langle \mathbf{stop}, x > 7\rangle$$

hence $\langle P + Q, true\rangle \xrightarrow{x>7} \langle \mathbf{stop}, x > 7\rangle$ is redundant since $(x > 5) \sqsubset (x > 7)$, therefore it does not need to be matched by $\langle P, true\rangle$. On the other hand, $\langle P + Q, true\rangle \xrightarrow{x>5} \langle \mathbf{stop}, x > 5\rangle$ is irredundant (Definition 3.1.6), thus this transition must be matched by $\langle P, true\rangle$. Hence we can take $\langle P, true\rangle \xrightarrow{x>5} \langle \mathbf{stop}, x > 5\rangle$ and then $\langle \mathbf{stop}, x > 5\rangle \mathcal{R} \langle \mathbf{stop}, x > 5\rangle$. The other pairs are trivially verified using the identity relation.

We shall prove that $\dot{\sim}_I$ and $\dot{\sim}_{sb}$ coincide, but in order to do so, we need to introduce another notion of bisimilarity called *symbolic* bisimilarity.

Intuitively, two configurations γ_1 and γ_2 are symbolic bisimilar iff (i) they have the same barbs and (ii) whenever there is a transition from $\gamma_1 \xrightarrow{\alpha} \gamma_1'$, then we require that γ_2 must reply with a similar transition $\gamma_2 \xrightarrow{\alpha} \gamma_2'$ (where γ_1' and γ_2' are now equivalent) or some other transition that derives it. In other words,

the move from the defender does not need to use exactly the same label, but a transition that is "stronger or equal" (in terms of derivation $\vdash_D$) could also do the job. Formally we have:

Definition 3.2.3 (Symbolic Bisimilarity)**.** *A symbolic bisimulation is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, c\rangle$ and $\gamma_2 = \langle Q, d\rangle$ implies that:*

(i) if $\gamma_1 \downarrow_e$ then $\gamma_2 \downarrow_e$,

(ii) if $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ then there exists a transition $t = \langle Q, d\rangle \xrightarrow{\beta} \langle Q', d''\rangle$ and a store d' s.t. $t \vdash_D \langle Q, d\rangle \overset{\alpha}{\rightsquigarrow} \langle Q', d'\rangle$ and $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$.

We say that γ_1 and γ_2 are symbolic bisimilar ($\gamma_1 \dot{\sim}_{sym} \gamma_2$) if there exists a symbolic bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$.

To better explain this definition consider the following example.

Example 3.2.4. *Let P, Q and $\mathcal{R}$ as in Example 3.2.2. We shall prove that $\langle P + Q, true\rangle$ and $\langle P, true\rangle$ are symbolic bisimilar. To do this we prove that $\mathcal{R}$ is a symbolic bisimulation. Take the pair $(\langle P + Q, true\rangle, \langle P, true\rangle)$. The first condition in Definition 3.2.3 is trivial. As for the second one, the only interesting transition is $\langle P + Q, true\rangle \xrightarrow{x>7} \langle \mathbf{stop}, x > 7\rangle$. We have to check that $\langle P, true\rangle$ is able to reply with a stronger transition. Thus consider the transitions t and t' as follows:*

$$t = \langle P, true\rangle \xrightarrow{x>5} \langle \mathbf{stop}, x > 5\rangle \text{ and } t' = \langle P, true\rangle \overset{x>7}{\rightsquigarrow} \langle \mathbf{stop}, x > 7\rangle$$

Then we can observe that $t \vdash_D t'$ and $\langle \mathbf{stop}, x > 7\rangle \mathcal{R} \langle \mathbf{stop}, x > 7\rangle$. The remaining pairs are trivially verified using the identity relation.

The rest of this section is devoted to prove that $\dot{\sim}_{sb}$, $\dot{\sim}_{sym}$ and $\dot{\sim}_I$ coincide. To skip the details go directly to Section 3.2.2. First we need to fix some notation.

Definition 3.2.5 (Closure under the addition of constraints)**.** *We say that a relation $\mathcal{R} \subseteq Conf \times Conf$ is closed under the addition of constraints iff whenever $\langle P, c\rangle \mathcal{R} \langle Q, d\rangle$ then for all $e \in Con_0$ we have that $\langle P, c \sqcup e\rangle \mathcal{R} \langle Q, d \sqcup e\rangle$*

We can now prove that $\dot{\sim}_{sb}$ and $\dot{\sim}_{sym}$ coincide as shown in the following two lemmata.

Lemma 3.2.6. *If* $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ *then* $\langle P, c\rangle \dot{\sim}_{sb} \langle Q, d\rangle$

Proof. We will prove that $\mathcal{R} = \{(\langle P, c \sqcup a\rangle, \langle Q, d \sqcup a\rangle) \mid \langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle\}$ is a saturated barbed bisimulation.

(i) Since $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ then from condition (i) and by monotonicity we have that for all e s.t. $\langle P, c \sqcup a\rangle \downarrow_e$ then $\langle Q, d \sqcup a\rangle \downarrow_e$.

(ii) We need to prove that if $\langle P, c \sqcup a\rangle \longrightarrow \langle P_1, c_1\rangle$ then there exists $\langle Q_1, d_1\rangle$ s.t. $\langle Q, d \sqcup a\rangle \longrightarrow \langle Q_1, d_1\rangle$ and $\langle P_1, c_1\rangle \mathcal{R} \langle Q_1, d_1\rangle$. Now let us assume that $\langle P, c \sqcup a\rangle \longrightarrow \langle P', c'\rangle$ (take $P_1 = P'$ and $c_1 = c'$) then by completeness (Lemma 2.4.25) we know that there exists α and b s.t. $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c''\rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$, thus if such transition exists then by $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ we get that there exists a transition $t = \langle Q, d\rangle \xrightarrow{\beta} \langle Q', d'\rangle$ s.t. $t \vdash_D \langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d''\rangle$ and $\langle P', c''\rangle \dot{\sim}_{sym} \langle Q', d''\rangle$. Expanding the definition of $\vdash_D$ there is a b' s.t. $\beta \sqcup b' = \alpha$ and $d' \sqcup b' = d''$. We are now able to apply soundness (Lemma 2.4.24) on t hence $\langle Q, d \sqcup \beta\rangle \longrightarrow \langle Q', d'\rangle$, and by monotonicity we are able to add $b \sqcup b'$ to the store to obtain $\langle Q, d \sqcup \beta \sqcup b \sqcup b'\rangle \longrightarrow \langle Q', d' \sqcup b \sqcup b'\rangle$ which is equivalent to say (by using the equations above) $\langle Q, d \sqcup a\rangle \longrightarrow \langle Q', d'' \sqcup b\rangle$. To conclude, take $Q_1 = Q'$ and $d_1 = d'' \sqcup b$, therefore $\langle Q, d \sqcup a\rangle \longrightarrow \langle Q_1, d_1\rangle$ and since $\langle P', c''\rangle \dot{\sim}_{sym} \langle Q', d''\rangle$ then $\langle P_1, c_1\rangle = \langle P', c'\rangle = \langle P', c'' \sqcup b\rangle \mathcal{R} \langle Q', d'' \sqcup b\rangle = \langle Q_1, d_1\rangle$.

(iii) By definition of $\mathcal{R}$, it is already closed under the addition of constraints.

□

Lemma 3.2.7. *If* $\langle P, c\rangle \dot{\sim}_{sb} \langle Q, d\rangle$ *then* $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$

Proof. We will prove that $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \langle P, c\rangle \dot{\sim}_{sb} \langle Q, d\rangle\}$ is a symbolic bisimulation.

(i) Since $\langle P, c\rangle \dot{\sim}_{sb} \langle Q, d\rangle$ then from condition (i) we have that if $\langle P, c\rangle \downarrow_e$ then $\langle Q, d\rangle \downarrow_e$

(ii) Let us start by assuming that $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ then we need to prove that there exists a transition $t = \langle Q, d\rangle \xrightarrow{\beta} \langle Q', d''\rangle$ s.t. $t \vdash_D \langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d'\rangle$

and $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$. By soundness we know $\langle P, c \sqcup \alpha\rangle \longrightarrow \langle P', c'\rangle$, now since $\langle P, c\rangle \dot{\sim}_{sb} \langle Q, d\rangle$ hence by condition (ii) we obtain $\langle Q, d \sqcup \alpha\rangle \longrightarrow \langle Q', d'\rangle$ where $\langle P', c'\rangle \dot{\sim}_{sb} \langle Q', d'\rangle$. From this transition and completeness we can deduce that there exist β and b s.t. $\langle Q, d\rangle \xrightarrow{\beta} \langle Q', d''\rangle$ (let us call this transition t) where $\beta \sqcup b = \alpha$ and $d'' \sqcup b = d'$. Thus by definition of $\vdash_D$ we can conclude that $t \vdash_D \langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d'\rangle$ and since $\langle P', c'\rangle \dot{\sim}_{sb} \langle Q', d'\rangle$ then $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$.

□

Using the two lemmata above we can state the following theorem.

Theorem 3.2.8. $\langle P, c\rangle \dot{\sim}_{sb} \langle Q, d\rangle$ *iff* $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$

Moreover, the following corollary is derived from the definition of $\dot{\sim}_{sb}$.

Corollary 3.2.9. *If* $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ *then* $\forall a \in Con_0$, $\langle P, c \sqcup a\rangle \dot{\sim}_{sym} \langle Q, d \sqcup a\rangle$

Proof. Given that $\dot{\sim}_{sb}$ is closed under the addition of constraints, then it follows directly from Theorem 3.2.8. □

We proceed to prove that $\dot{\sim}_{sym}$ coincide with $\dot{\sim}_I$. To achieve this goal we need the following proposition stating that the derivation and domination relations are well-founded.

Proposition 3.2.10. $\vdash_D, \succ_D, \vdash_{\mathcal{R}}$ *and* $\succ_{\mathcal{R}}$ *are well-founded.*

Proof. Follows from the well-foundedness of $\sqsubseteq$ (Remark 2.4.3). □

We also need to prove that $\dot{\sim}_I$ is closed under the addition of constraints.

Lemma 3.2.11. *If* $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$ *then for all* $a \in Con_0$, $\langle P, c \sqcup a\rangle \dot{\sim}_I \langle Q, d \sqcup a\rangle$

Proof. We need to prove that $\mathcal{R} = \{(\langle P, c \sqcup e\rangle, \langle Q, d \sqcup e\rangle) \mid \langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle\}$ is an irredundant bisimulation.

(i) Since $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$ then from condition (i) and by monotonicity we have that for all e s.t. $\langle P, c \sqcup a\rangle \downarrow_e$ then $\langle Q, d \sqcup a\rangle \downarrow_e$.

(ii) Let us start by assuming that

$$\langle P, c \sqcup a\rangle \xrightarrow{\alpha} \langle P', c'\rangle \text{ which is irredundant in } \mathcal{R} \tag{3.1}$$

then we need to prove that there exists $\langle Q', d'\rangle$ s.t. $\langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d'\rangle$ and $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$. Now by soundness on the transition from $\langle P, c \sqcup a\rangle$ we know that $\langle P, c \sqcup a \sqcup \alpha\rangle \longrightarrow \langle P', c'\rangle$ and from completeness we get that there exist b and β s.t.

$$t_1 = \langle P, c\rangle \xrightarrow{\beta} \langle P', c''\rangle \text{ where } \beta \sqcup b = a \sqcup \alpha \text{ and } c'' \sqcup b = c' \tag{3.2}$$

Since $\succ_{\mathcal{R}}$ is well founded then there exist an irredundant (not dominated by the rest) transition $t_2 = \langle P, c\rangle \xrightarrow{\lambda} \langle P_1, c_1\rangle$ that dominates t_1 in $\mathcal{R}$ ($t_2 \succ_{\mathcal{R}} t_1$), therefore

$$t_2 \succ_D \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c_2\rangle \text{ and } \langle P_1, c_2\rangle \mathcal{R} \langle P', c''\rangle \tag{3.3}$$

by definition of $\succ_D$ there exists b' s.t.

$$\lambda \sqcup b' = \beta, c_1 \sqcup b' = c_2 \text{ and } \lambda \neq \beta \tag{3.4}$$

Now since $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$ and the irredundant t_2, then there exists $\langle Q_1, d_1\rangle$ s.t.

$$\langle Q, d\rangle \xrightarrow{\lambda} \langle Q_1, d_1\rangle \text{ and } \langle P_1, c_1\rangle \dot{\sim}_I \langle Q_1, d_1\rangle \tag{3.5}$$

using soundness $\langle Q, d \sqcup \lambda\rangle \longrightarrow \langle Q_1, d_1\rangle$ and monotonicity we can obtain $\langle Q, d \sqcup \overbrace{\underbrace{\lambda \sqcup b'}_{a \sqcup \alpha} \sqcup b}^{\beta}\rangle \longrightarrow \langle Q_1, d_1 \sqcup b' \sqcup b\rangle$ and from the latter condition in (3.5) we can also deduce that $\langle Q_1, d_1 \sqcup b' \sqcup b\rangle \mathcal{R} \langle P_1, \underbrace{c_1 \sqcup b'}_{c_2} \sqcup b\rangle$ therefore by the second condition in (3.3) we know that $\langle P_1, c_2 \sqcup b\rangle \mathcal{R} \langle P', \underbrace{c'' \sqcup b}_{c'}\rangle$. Let $d_1' = d_1 \sqcup b' \sqcup b$, we can summarize this part by saying that

$$\langle Q, d \sqcup a \sqcup \alpha\rangle \longrightarrow \langle Q_1, d_1'\rangle \text{ and } \langle P', c'\rangle \mathcal{R} \langle Q_1, d_1'\rangle \tag{3.6}$$

Now let us reason on this transition, by completeness there exist α_1 and b_1 s.t.

$$t_3 = \langle Q, d \sqcup a\rangle \xrightarrow{\alpha_1} \langle Q_1, d_2\rangle \text{ where } \alpha_1 \sqcup b_1 = \alpha \text{ and } d_2 \sqcup b_1 = d'_1 \quad (3.7)$$

by means of contradiction let us assume that $\alpha_1 \neq \alpha$, then by soundness (on t_3) $\langle Q, d \sqcup a \sqcup \alpha_1\rangle \longrightarrow \langle Q_1, d_2\rangle$ and, by completeness on this transition, we know there exist α_2 and b_2 s.t.

$$t_4 = \langle Q, d\rangle \xrightarrow{\alpha_2} \langle Q_1, d'_2\rangle \text{ where } \alpha_2 \sqcup b_2 = a \sqcup \alpha_1 \text{ and } d'_2 \sqcup b_2 = d_2 \quad (3.8)$$

by the well-foundedness of $\succ_{\mathcal{R}}$, we know there exists an irredundant transition $t_5 = \langle Q, d\rangle \xrightarrow{\alpha'} \langle Q_3, d_3\rangle$ s.t. $t_5 \succ_{\mathcal{R}} t_4$, namely,

$$t_5 \succ_D \langle Q, d\rangle \xrightarrow{\alpha_2} \langle Q_3, d'_3\rangle \text{ and } \langle Q_3, d'_3\rangle \mathcal{R} \langle Q_1, d'_2\rangle \quad (3.9)$$

hence, there exists b_3 s.t. $\alpha' \sqcup b_3 = \alpha_2$, $d_3 \sqcup b_3 = d'_3$ and $\alpha' \neq \alpha_2$. Now since $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$ then from the irredundant t_5 we can deduce that

$$\langle P, c\rangle \xrightarrow{\alpha'} \langle P_3, c_3\rangle \text{ and } \langle P_3, c_3\rangle \mathcal{R} \langle Q_3, d_3\rangle \quad (3.10)$$

by soundness we get $\langle P, c \sqcup \alpha'\rangle \longrightarrow \langle P_3, c_3\rangle$ and by monotonicity

$$\langle P, c \sqcup \overbrace{\alpha' \sqcup b_2 \sqcup b_3}^{a \sqcup \alpha_1}\rangle \longrightarrow \langle P_3, \overbrace{c_3 \sqcup b_2 \sqcup b_3}^{c_4}\rangle, \quad (3.11)$$

from the latter condition in (3.10) and by definition of $\mathcal{R}$ we have $\langle P_3, c_3 \sqcup b_2 \sqcup b_3\rangle \mathcal{R} \langle Q_3, d_3 \sqcup b_2 \sqcup b_3\rangle$, and since $d_3 \sqcup b_3 = d'_3$ and from the latter condition in (3.9) then $\langle Q_3, d'_3 \sqcup b_2\rangle \mathcal{R} \langle Q_1, d'_2 \sqcup b_2\rangle$ and using $d'_2 \sqcup b_2 = d_2$ we can conclude that $\langle P_3, c_4\rangle \mathcal{R} \langle Q_1, d_2\rangle$. Going back to the transition in (3.11), we can rewrite it as $\langle P, c \sqcup a \sqcup \alpha_1\rangle \longrightarrow \langle P_3, c_4\rangle$, then by completeness we know there exist b_4 and α'_1 s.t.

$$t_6 = \langle P, c \sqcup a\rangle \xrightarrow{\alpha'_1} \langle P_3, c'_4\rangle \text{ where } \alpha'_1 \sqcup b_4 = \alpha_1 \text{ and } c'_4 \sqcup b_4 = c_4 \quad (3.12)$$

notice that if such transition exists then $t_6 \succ_{\mathcal{R}}$ (3.1), as we prove as follows

$$t_6 \succ_D \langle P, c \sqcup a\rangle \overset{\alpha_1' \sqcup b_4 \sqcup b_1}{\longrightarrow} \langle P_3, \overbrace{c_4' \sqcup b_4}^{c_4} \sqcup b_1\rangle \tag{3.13}$$

and now it is left to prove that $\langle P_3, c_4 \sqcup b_1\rangle \mathcal{R} \langle P', c'\rangle$, given that $\langle P_3, c_4\rangle \mathcal{R} \langle Q_1, d_2\rangle$ then by definition of $\mathcal{R}$ we have $\langle P_3, c_4 \sqcup b_1\rangle \mathcal{R} \langle Q_1, d_2 \sqcup b_1\rangle$. Since $\langle Q_1, d_2 \sqcup b_1\rangle = \langle Q_1, d_1'\rangle$ and we have already proven that $\langle P', c'\rangle \mathcal{R} \langle Q_1, d_1'\rangle$ (latter condition in (3.6)), finally we can conclude that $\langle P_3, c_4 \sqcup b_1\rangle \mathcal{R} \langle P', c'\rangle$ and therefore (3.1) is redundant, a contradiction. To conclude, α_1 must be equal to α, otherwise we would get an absurd as shown previously, thus we can conclude our main result by using (3.6), (3.7) and assuming $Q' = Q_1$, $d' = d_1'$,

$$\langle Q, d \sqcup a\rangle \xrightarrow{\alpha} \langle Q_1, d_1'\rangle \text{ and } \langle P', c'\rangle \mathcal{R} \langle Q_1, d_1'\rangle \tag{3.14}$$

indeed $\langle Q, d \sqcup a\rangle$ can defend from the attacker's move by using the same label and still remain in the relation.

□

Using the lemma above we prove the correspondence between symbolic and irredundant bisimilarity in the two following lemmata.

Lemma 3.2.12. *If* $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ *then* $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$

Proof. We will prove that $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle\}$ is an irredundant bisimulation.

(i) Since $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ it is direct result from condition (i)

(ii) Assume that $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ (1) which is irredundant in $\mathcal{R}$ then we need to prove that there exists $\langle Q', d'\rangle$ s.t. $\langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d'\rangle$ and $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$. Since $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ and from (1), we have that there exists $t = \langle Q, d\rangle \xrightarrow{\beta} \langle Q', d''\rangle$ s.t. $t \vdash_D \langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d'\rangle$ and $\langle P', c'\rangle \dot{\sim}_{sym} \langle Q', d'\rangle$, thus by definition of $\vdash_D$, there is b s.t. $\beta \sqcup b = \alpha$ and $d'' \sqcup b = d'$. Now let us assume by means of contradiction that $\beta \neq \alpha$, then we can use t to reason about what $\langle P, c\rangle$ can do, again from $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ and t we

can say that there exists a transition $t' = \langle P, c\rangle \xrightarrow{\lambda} \langle P_1, c_1'\rangle$ s.t. $t' \vdash_D \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c_1\rangle$ and $\langle P_1, c_1\rangle \mathcal{R} \langle Q', d''\rangle$ (2). By definition of $\vdash_D$ the last derivation means that there is a b' s.t. $\lambda \sqcup b' = \beta$ and $c_1' \sqcup b' = c_1$. Now we can use Corollary 3.2.9 on (2) to get $\langle P_1, c_1 \sqcup b\rangle \mathcal{R} \langle Q', d'' \sqcup b\rangle$ therefore, given that $d'' \sqcup b = d'$, then $\langle P_1, c_1 \sqcup b\rangle \mathcal{R} \langle Q', d'\rangle$, which by definition of $\mathcal{R}$ means $\langle P_1, c_1 \sqcup b\rangle \dot{\sim}_{sym} \langle Q', d'\rangle$ (3). We can also conclude that since $\langle P', c'\rangle \dot{\sim}_{sym} \langle Q', d'\rangle$ then by transitivity $\langle P_1, c_1 \sqcup b\rangle \dot{\sim}_{sym} \langle P', c'\rangle$ and hence $\langle P_1, c_1 \sqcup b\rangle \mathcal{R} \langle P', c'\rangle$ (4). On the other hand, notice that now t' is able to dominate our originally irredundant transition, namely $t' \succ_{\mathcal{R}} (1)$, as follows

$$t' \succ_D \langle P, c\rangle \xrightarrow{\overbrace{\overbrace{\lambda \sqcup b'}^{\beta} \sqcup b}^{\alpha}} \langle P', \overbrace{c_1' \sqcup b'}^{c_1} \sqcup b\rangle \text{ and } \langle P_1, c_1 \sqcup b\rangle \mathcal{R} \langle P', c'\rangle \text{ from (4)}$$

therefore if $\alpha \neq \beta$ then we would get an absurd since (1) would be redundant in $\mathcal{R}$, thus we can finally say that $\alpha = \beta$ which allow us to conclude that $\langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d'\rangle$ and $\langle P', c'\rangle \dot{\sim}_{sym} \langle Q', d'\rangle$ then $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$.

□

Lemma 3.2.13. *If* $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$ *then* $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$

Proof. We will prove that $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle\}$ is a symbolic bisimulation.

(i) Since $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$ it is direct result from condition (i)

(ii) Take $t = \langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ then since $\succ_{\mathcal{R}}$ is well founded then there exists an irredundant transition $t' = \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c_1\rangle$ s.t. $t' \succ_{\mathcal{R}} t$,

$$t' \succ_D \langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c_1'\rangle \text{ where } \langle P_1, c_1'\rangle \mathcal{R} \langle P', c'\rangle \tag{3.15}$$

by definition of $\succ_D$, there exists a b s.t. $\beta \sqcup b = \alpha$, $c_1 \sqcup b = c_1'$ and $\alpha \neq \beta$. Now since $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$ and t' then we know there is a transition $t'' = \langle Q, d\rangle \xrightarrow{\beta} \langle Q', d'\rangle$ and $\langle P_1, c_1\rangle \mathcal{R} \langle Q', d'\rangle$. Thus, from Lemma 3.2.11 $\langle P_1, c_1 \sqcup b\rangle \mathcal{R} \langle Q', d' \sqcup b\rangle$, equivalently $\langle P_1, c_1'\rangle \mathcal{R} \langle Q', d' \sqcup b\rangle$ and using

the latter condition in (3.15) then $\langle Q', d' \sqcup b\rangle \mathcal{R} \langle P', c'\rangle$. We can finally conclude that $t'' \succ_D \langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d' \sqcup b\rangle$ and $\langle Q', d' \sqcup b\rangle \mathcal{R} \langle P', c'\rangle$, therefore the condition for being a symbolic bisimulation is proven.

□

Using these lemmata we can conclude the following theorem.

Theorem 3.2.14. $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ *iff* $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$

Proof. It follows directly from Lemma 3.2.12 and Lemma 3.2.13. □

Finally, we obtain the correspondence between $\dot{\sim}_{sb}$ and $\dot{\sim}_I$.

Corollary 3.2.15. $\langle P, c\rangle \dot{\sim}_{sb} \langle Q, d\rangle$ *iff* $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$

Proof. Follows from Theorem 3.2.8 and 3.2.14. □

3.2.2 Proof of Correctness and Complexity

Using the results from the previous section we can establish the correctness of Algorithm 3.1.2. Recall that $\texttt{Config}_{\longrightarrow}(IS)$ represents the set of states that are reachable from the initial states IS using $\longrightarrow$.

Theorem 3.2.16. *Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of $\texttt{pr-ccp}(IS, \longrightarrow)$ in Algorithm 3.1.2. If $IS^{\star}_{\longrightarrow}$ is finite then the algorithm terminates and:*

- $\gamma \; \mathcal{P} \; \gamma'$ *iff* $\gamma \dot{\sim}_{sb} \gamma'$.
- $\texttt{pr-ccp}(IS, \longrightarrow)$ *may take exponential time in the size of* $\texttt{Config}_{\longrightarrow}(IS)$.

Proof. The first item is obtained by using Corollary 1 of [15] and Corollary 3.2.15 as well as the decidability of $\vdash_D$ (which follows from the decidability of $\sqsubseteq$, Remark 2.4.3). For the second item let $n > 0$. We define $P^n = P^n_0$ with P^n_i, for $i \in \{0, \ldots, n-1\}$, given by:

$$P^n_i = (\mathbf{ask}\ b_i \ \rightarrow\ \mathbf{stop}) + (\mathbf{ask}\ a_i \ \rightarrow\ P^n_{i+1})$$

and $P_n^n = \mathbf{tell}(b_n)$. Furthermore, we assume that for all $i \in \{0, \ldots, n-1\}$ we have $a_i \sqsubset b_i$ and for all $j \in \{0, \ldots, n-1\}$ if $i \neq j$ then $a_i \not\sqsubseteq a_j$ and $b_i \not\sqsubseteq b_j$. The reachable states from $\langle P^n, true\rangle$ are illustrated below.

$$\begin{array}{ccc}
\langle P^n, true\rangle & \xrightarrow{\quad b_0 \quad} & \langle \mathbf{stop}, b_0\rangle \\
\Big\downarrow a_0 & & \\
\langle P_1^n, a_0\rangle & \xrightarrow{\quad b_1 \quad} & \langle \mathbf{stop}, a_0 \sqcup b_1\rangle \\
\Big\downarrow a_1 & & \\
\cdots & & \\
\Big\downarrow a_{n-1} & & \\
\langle P_n^n, \bigsqcup_{i=0}^{n-1} a_i\rangle & \xrightarrow{\quad b_n \quad} & \langle \mathbf{stop}, b_n \sqcup \bigsqcup_{i=0}^{n-1} a_i\rangle
\end{array}$$

Notice that that size of the set of reachable states $|\mathtt{Config}_{\longrightarrow}(IS)|$ is $2n$. By using the rules $(\mathsf{IS}_{\rightsquigarrow}^{IS})$, $(\mathsf{RS}_{\rightsquigarrow}^{IS})$ and $(\mathsf{RD}_{\rightsquigarrow}^{IS})$ in Table 3.1.1 with $\rightsquigarrow = \longrightarrow$ and also $IS = \{\langle P^n, true\rangle\}$, we obtain an $IS^{\star}_{\longrightarrow}$ whose size is given by the following recurrence relation, for $n > 0$: $f(n) = 2f(n-1) + 2$ with $f(0) = 2$. Without loss of generality consider the first level of transitions of $\langle P^n, true\rangle$:

$$\begin{array}{ccc}
\langle P^n, true\rangle & \xrightarrow{\quad b_0 \quad} & \langle \mathbf{stop}, b_0\rangle \\
\Big\downarrow a_0 & & \\
\langle P_1^n, a_0\rangle & &
\end{array}$$

First, we count $\langle P^n, true\rangle$ and $\langle \mathbf{stop}, b_0\rangle$, hence the $+2$ in $f(n)$. Furthermore, the rest of the states are those generated by $\langle P_1^n, a_0\rangle$ and these are exactly $f(n-1)$. Moreover, since $a_0 \sqsubset b_0$ then by rule $(\mathsf{RD}_{\rightsquigarrow}^{IS})$ a new state $\langle P_1^n, b_0\rangle$ is added to $IS^{\star}_{\Longrightarrow}$. Note that $\langle P_1^n, b_0\rangle$ produces the same number of states as $\langle P_1^n, a_0\rangle$ since by construction a_0 and b_0 are irrelevant in P_1^n. Therefore we count the states generated by $\langle P_1^n, b_0\rangle$ as another $f(n-1)$. Also note that the same reasoning applies at every level of the process. By solving the recurrence we can conclude that $f(n) = \Omega(2^n)$ hence $|IS^{\star}_{\Longrightarrow}| = \Omega(2^n)$ as wanted. □

We now prove that if the set $\mathtt{Config}_{\longrightarrow}(IS)$ of all configurations reachable from IS is finite, then $IS^{\star}_{\longrightarrow}$ is finite. This condition can be easily guaranteed by

imposing some syntactic restrictions on CCP terms, like for instance, by excluding either the procedure call or the hiding operator.

Theorem 3.2.17. *Let IS be a set of configurations. If $\texttt{Config}_{\longrightarrow}(IS)$ is finite, then $IS^{\star}_{\longrightarrow}$ is finite.*

Proof. As a first step, we observe that a configuration $\gamma \in IS^{\star}_{\longrightarrow}$ only if $\gamma = \langle P, d \sqcup e\rangle$ and $\langle P, d\rangle \in \texttt{Config}_{\longrightarrow}(IS)$. Then we prove that there are only finitely many such constraints e. Let $\texttt{Label}(IS)$ be the set of all labels in $LTS_{\longrightarrow}(IS)$. This set is finite (given that $\texttt{Config}_{\longrightarrow}(IS)$ is finite), and its downward closure $\downarrow\texttt{Label}(IS) = \{a \mid \exists b \in \texttt{Label}(IS) \text{ with } a \sqsubseteq b\}$ is also finite (since $\sqsubseteq$ is well-founded). The set of all e s.t. $\langle P, d \sqcup e\rangle \in IS^{\star}_{\longrightarrow}$ (with $\langle P, d\rangle \in \texttt{Config}_{\longrightarrow}(IS)$) is a subset of $\downarrow\texttt{Label}(IS)$ and thus it is finite. Indeed, observe that if $\langle P, d \sqcup e\rangle \xrightarrow{\alpha}$, then $\langle P, d\rangle \xrightarrow{\beta}$ with $\alpha \sqsubseteq \beta$. Therefore $\texttt{Label}(IS^{\star}_{\longrightarrow}) \subseteq \downarrow\texttt{Label}(IS)$. Moreover, if $\langle P, d \sqcup e\rangle$ is added to $IS^{\star}_{\longrightarrow}$ by the rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ then $e \sqsubseteq c$ for c being a label in $\texttt{Label}(IS^{\star}_{\longrightarrow})$ (i.e., in $\downarrow\texttt{Label}(IS)$). □

Finally we conclude this section by proving an upper-bound on the runtime of Algorithm 3.1.2. Let $f_{\mathbf{C}}$ be the function that represents the time complexity of deciding (whether two given constraints are in) $\sqsubseteq$. Recall that $\texttt{V}(G)$ and $\texttt{E}(G)$ denote, respectively, the set of vertices and edges of a graph.

Theorem 3.2.18. *Let $n = |\texttt{V}(LTS_{\longrightarrow}(IS))|$ and let $m = |\texttt{E}(LTS_{\longrightarrow}(IS))|$. Then $n \times 2^{O(m)} \times f_{\mathbf{C}}$ is an upper bound for the running time of Algorithm 3.1.2.*

Proof. Let G be the graph obtained after constructing $IS^{\star}_{\longrightarrow}$ using rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RS}^{IS}_{\rightsquigarrow})$ and $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ in Table 3.1.1. Let $N = |\texttt{V}(G)|$ and let $M = |\texttt{E}(G)|$. One can verify that each state s in G corresponds to a state of s' in $LTS_{\longrightarrow}(IS)$ so that s and s' have the same process and the store of s results from some least upper bound of the stores in the transitions between the states of $LTS_{\longrightarrow}(IS)$. Hence, N is bounded by $O(n \times 2^m)$. Similarly, we can conclude that M is bounded by $O(m \times 2^m)$. Notice that we need to check for $\succ_{\mathcal{R}}$ in each transition (between the states) in $IS^{\star}_{\longrightarrow}$. Hence, we conclude that constructing $IS^{\star}_{\longrightarrow}$ takes $O(f_{\mathbf{C}} \times m \times 2^m)$ time. Using the partition refinement from Paige and Tarjan [50] and taking into account the checks for irredundant transitions we can obtain a polynomial time

bound on N and M for the overall executions of the iterations of Algorithm 3.1.2. From the above upper-bounds for N and M it follows that $n \times 2^{O(m)} \times f_{\mathbf{C}}$ is an upper bound for the execution time of Algorithm 3.1.2. □

3.3 Summary and Related Work

In this chapter we introduced a novel decision procedure (Algorithm 3.1.2) for computing strong saturated barbed bisimilarity for CCP ($\dot{\sim}_{sb}$, Definition 2.4.9). Algorithm 3.1.2 is an adaptation of the standard partition refinement algorithm from [34, 51] typically used to check strong bisimilarity between two LTSs. The adaptation is based on the idea that only certain special kind of transitions are necessary (and sufficient) to analyze the behavior of the program. These transitions, called irredundant, are the strongest transitions w.r.t. the order given by the domination relation ($\succ_{\mathcal{R}}$, Definition 3.1.6). In this chapter, we proposed a notion of bisimilarity that considers only the irredundant transitions, we call it irredundant bisimilarity ($\dot{\sim}_I$, Definition 3.2.1), and we prove that it coincides with $\dot{\sim}_{sb}$.

Moreover, we demonstrate that using the irredundant transitions in the partition refinement algorithm we can compute $\dot{\sim}_I$, thus also $\dot{\sim}_{sb}$. However, we show that, from the LTS of the input, we may not have enough information to determine which of the transitions are irredundant. Hence, the procedure must consider some additional unreachable states. Furthermore, from [15] we know that, even with the new states, redundancy must be computed together with bisimilarity. Finally, we prove that Algorithm 3.1.2 runs in exponential time in the size of the LTS of the input, mainly because of the additional states needed for detecting redundancy.

The results of this chapter are based on the abstract approach of [15]. The authors define a symbolic semantics for labeled transition systems in general and they show how to use this semantics to compute saturated barbed bisimilarity. We borrow their ideas of irredundant transitions and irredundant bisimilarity and we adapt this approach to the case of CCP. In [15] the authors discuss that checking for dominance may be costly in general. Hence, the novelty of our procedure is that, for CCP, we characterize dominance between transitions ($\succ_{\mathcal{R}}$, Definition 3.1.6) in a simple manner. This allows us to exploit the partition refinement approach from [15] to obtain the main goal of this chapter: computing $\dot{\sim}_{sb}$.

Publications from this Chapter

The material of this chapter has been published in the following paper:

- [8] A. Aristizabal, F. Bonchi, **L. Pino**, F. Valencia, *Partition Refinement for Bisimilarity in CCP*, in: S. Ossowski, P. Lecca (Eds.), Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012), Constraint Solving and Programming Track, ACM, 2012, pp. 88-93. DOI: `http://dx.doi.org/10.1145/2245276.2245296`.

Chapter 4

A Weak Semantics for CCP

In the previous chapter we defined an algorithm to verify strong saturated barbed bisimilarity in CCP ($\dot{\sim}_{sb}$, Definition 2.4.9). To achieve this goal we used the irredundant transitions, namely those that are not dominated by any other transition (w.r.t. $\succ_{\mathcal{R}}$, Definition 3.1.6). Using them we proposed a notion of bisimilarity, called irredundant bisimilarity $\dot{\sim}_I$ (Definition 3.2.1), Which consists in playing the standard label-matching bisimulation game but where the attacker can only use irredundant transitions. We proved that $\dot{\sim}_I$ coincides with $\dot{\sim}_{sb}$ and that partition refinement, together with the check for redundancy, can be used to compute $\dot{\sim}_I$, thus also $\dot{\sim}_{sb}$.

The goal of this chapter is to exploit the algorithm for $\dot{\sim}_{sb}$ in order to verify its weak version $\dot{\approx}_{sb}$ (Definition 2.4.10). Recall that, from the Proposition 2.4.21 we know that, in the choice-free fragment (CCP\+), $\dot{\approx}_{sb}$ coincides with the observational equivalence $\sim_o$ (Definition 2.4.19). In Section 4.1, we present the standard strategy to reduce weak to strong bisimilarity. The idea is to add some extra transitions to the LTS of the input, this idea is usually called *saturation*. Then the problem of checking weak bisimilarity is reduced to check the strong version in the saturated LTS. The standard saturation method closes the LTS w.r.t. to the internal transitions. However, as we shall explain later on, such approach does not work for CCP.

In Section 4.2 we shall prove that, because of the involved transitions in CCP, the saturation for CCP must be the reflexive and transitive closure of the input

LTS. Moreover, we demonstrate that the our method is adequate for computing $\dot{\approx}_{sb}$ using the algorithm for $\dot{\sim}_{sb}$ presented in Section 3. Finally, in Section 4.3 we specify the decision procedure for checking $\dot{\approx}_{sb}$ based on Algorithm 3.1.2 and we also prove its correctness and complexity.

4.1 The standard reduction from weak to strong

Following [2] the reduction of the problem of deciding weak bisimilarity to the problem of deciding the strong one is obtained by adding some additional transitions, so called weak transitions, to the LTS. The idea is to start from the LTS generated using the operational semantics ($\longrightarrow$) and then *saturate* it using the rules described in Table 4.1.1. Now the problem whether two states s and s' are weakly bisimilar can be reduced to checking whether they are strongly bisimilar w.r.t. $\Longrightarrow$. Formally, we can call $\mathtt{pr}(\{s, s'\}, \Longrightarrow)$, i.e. the partition refinement algorithm, to check whether s and s' are weakly bisimilar. Henceforth, we shall refer to this as *Milner's saturation method.*

As we will show later on, this approach does not work in a formalism like CCP. We shall see that the problem is related to the fact that the transitions in CCP are labeled with constraints, thus they can be arbitrary combined (using $\sqcup$) to form a new label. Notice that this is not the case for the transitions in CCS-like process calculi. Therefore we will need to define a different saturation method to be able to use the standard approach for verifying weak bisimilarity in CCP. Namely, to achieve this goal, we shall modify the rules in Table 4.1.1.

MR1 $\dfrac{P \stackrel{a}{\longrightarrow} P'}{P \stackrel{a}{\Longrightarrow} P'}$ MR2 $\dfrac{}{P \stackrel{\tau}{\Longrightarrow} P}$

MR3 $\dfrac{P \stackrel{\tau}{\Longrightarrow} P_1 \stackrel{a}{\Longrightarrow} P_2 \stackrel{\tau}{\Longrightarrow} P'}{P \stackrel{a}{\Longrightarrow} P'}$

Table 4.1.1: Milner's Saturation Method

4.1.1 Incompleteness of Milner's saturation method in CCP

The problem is that the transition relation proposed by Milner is not complete for CCP, hence the relation among the saturated, symbolic and irredundant equivalences (presented in Section 3.2.1) is broken.

In the next section we will provide a stronger saturation, which is complete, and allows us to use the CCP partition refinement to compute $\dot{\approx}_{sb}$. But first, let us show why Milner's approach does not work. For this we need to introduce formally the concept of *completeness* for a given transition relation.

Definition 4.1.1 (Completeness)**.** *We say that $\rightsquigarrow$ is complete if and only if whenever $\langle P, c \sqcup a\rangle \rightsquigarrow \langle P', c'\rangle$ then there exist $\alpha, b \in Con_0$ s.t. $\langle P, c\rangle \overset{\alpha}{\rightsquigarrow} \langle P', c''\rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$.*

Notice that $\longrightarrow$ (i.e the reduction semantics, see Table 2.4.1) is complete as proven in Lemma 2.4.25. Now Milner's method defines a new transition relation $\Longrightarrow$ using the rules in Table 4.1.2. However, the resulting $\Longrightarrow$ is not complete (in the sense of Definition 4.1.1) as proven below.

Proposition 4.1.2. *The relation $\Longrightarrow$ defined in Table 4.1.2 is not complete.*

Proof. We will show a counter-example where the completeness for $\Longrightarrow$ does not hold. Let $P = \mathbf{ask}\ \alpha \ \rightarrow\ (\mathbf{ask}\ \beta \ \rightarrow\ \mathbf{stop})$ and $d = \alpha \sqcup \beta$. Now consider the transition $\langle P, d\rangle \Longrightarrow \langle \mathbf{stop}, d\rangle$ and let us apply the completeness lemma. We can take $c = true$ and $a = \alpha \sqcup \beta$, therefore by completeness there must exist b and λ s.t. $\langle P, true\rangle \overset{\lambda}{\Longrightarrow} \langle \mathbf{stop}, c''\rangle$ where $\lambda \sqcup b = \alpha \sqcup \beta$ and $c'' \sqcup b = d$. However, notice that the only transition possible is $\langle P, true\rangle \overset{\alpha}{\Longrightarrow} \langle(\mathbf{ask}\ \beta \ \rightarrow\ \mathbf{stop}), \alpha\rangle$, hence completeness does not hold since there is no transition from $\langle P, true\rangle$ to $\langle \mathbf{stop}, c''\rangle$ for some c''. Figure 4.1.1 illustrates the problem. □

MR1 $\dfrac{\gamma \xrightarrow{\alpha} \gamma'}{\gamma \overset{\alpha}{\Longrightarrow} \gamma'}$ MR2 $\dfrac{}{\gamma \Longrightarrow \gamma}$

MR3 $\dfrac{\gamma \Longrightarrow \gamma_1 \overset{\alpha}{\Longrightarrow} \gamma_2 \Longrightarrow \gamma'}{\gamma \overset{\alpha}{\Longrightarrow} \gamma'}$

Table 4.1.2: Milner's Saturation Method for CCP

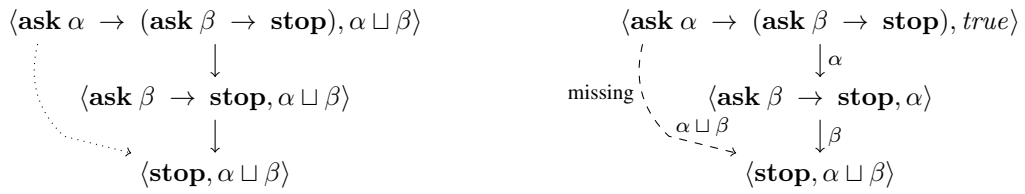

Figure 4.1.1: Counterexample for completeness using Milner's saturation method (cycles from MR2 omitted). Both graphs are obtained using Table 4.1.2. Also notice that the dotted transition is *not* generated by the rules in Table 4.1.2.

We can now use this fact to see why the method does not work for computing $\dot{\approx}_{sb}$ using CCP partition refinement. First, let us redefine some concepts using the new transition relation $\Longrightarrow$. Because of condition (i) in $\dot{\approx}_{sb}$, we need a new definition of barbs, namely *weak barbs w.r.t.* $\Longrightarrow$.

Definition 4.1.3 (Weak Barb w.r.t. $\Longrightarrow$)**.** *We say γ satisfies a weak barb e w.r.t. $\Longrightarrow$, written $\gamma \Downarrow_e$, if and only if $\gamma \Longrightarrow^* \gamma' \downarrow_e$.*

We shall see later on that $\Downarrow$ coincides with $\Downarrow$. Using this notion, we introduce weak symbolic and weak irredundant bisimilarity denoted by $\dot{\sim}^{\Longrightarrow}_{sym}$ and $\dot{\sim}^{\Longrightarrow}_{I}$ respectively. They are defined as in Definition 3.2.1 and 3.2.3, where in condition (i) weak barbs ($\Downarrow$) are replaced with $\Downarrow$ and in condition (ii) the transition relation is now $\Longrightarrow$. More precisely:

Definition 4.1.4 (Weak Symbolic Bisimilarity)**.** *A weak symbolic bisimulation is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, c\rangle$ and $\gamma_2 = \langle Q, d\rangle$:*

(i) if $\gamma_1 \Downarrow_e$ then $\gamma_2 \Downarrow_e$,

(ii) if $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$ then there exists a transition $t = \langle Q, d\rangle \overset{\beta}{\Longrightarrow} \langle Q', d''\rangle$ s.t. $t \vdash_D \langle Q, d\rangle \overset{\alpha}{\rightsquigarrow} \langle Q', d'\rangle$ and $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$

We say that γ_1 and γ_2 are weakly symbolic bisimilar, written $\gamma_1 \dot{\sim}^{\Longrightarrow}_{sym} \gamma_2$, if there exists a weak symbolic bisimulation s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Definition 4.1.5 (Weak Irredundant Bisimilarity)**.** *A weak irredundant bisimulation is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$:*

(i) if $\gamma_1 \not\downarrow_e$ *then* $\gamma_2 \not\downarrow_e$,

(ii) if $\gamma_1 \stackrel{\alpha}{\Longrightarrow} \gamma_1'$ *and it is irredundant in* $\mathcal{R}$ *then there exists* γ_2' *s.t.* $\gamma_2 \stackrel{\alpha}{\Longrightarrow} \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 *and* γ_2 *are weakly irredundant bisimilar, written* $\gamma_1 \dot{\sim}_I^{\Longrightarrow} \gamma_2$, *if there exists a weak irredundant bisimulation s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$.

One would expect that since $\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_I$ then it is the case that $\dot{\approx}_{sb} = \dot{\sim}_{sym}^{\Longrightarrow} = \dot{\sim}_I^{\Longrightarrow}$, given that these new notions are supposed to be the weak versions of the former ones when using the saturation method. However, completeness is necessary for proving $\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_I$, and from Proposition 4.1.2 we know that $\Longrightarrow$ is not complete hence we might expect that $\dot{\approx}_{sb}$ does not imply $\dot{\sim}_{sym}^{\Longrightarrow}$ nor $\dot{\sim}_I^{\Longrightarrow}$. In fact, the following counter-example proves this.

Example 4.1.6. *Let* P, P' *and* Q *as in Figure 4.1.2 and 4.1.3. The former shows* $LTS_{\longrightarrow}(IS)$ *where* $IS = \{\langle P, true\rangle, \langle Q, true\rangle\}$. *The latter presents* $LTS_{\Longrightarrow}(IS)$ *where* $\Longrightarrow$ *is defined in Table 4.1.2 (Milner's saturation method).*

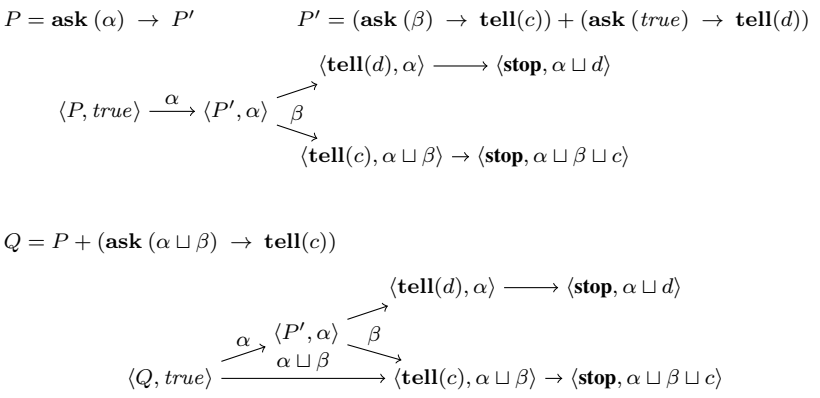

Figure 4.1.2: $LTS_{\longrightarrow}(\{\langle P, true\rangle, \langle Q, true\rangle\})$

First, notice that $\langle P, true\rangle \dot{\approx}_{sb} \langle Q, true\rangle$, *since there exists a saturated weak barbed bisimulation:*

$$\mathcal{R} = \{(\langle P, true\rangle, \langle Q, true\rangle)\} \cup id$$

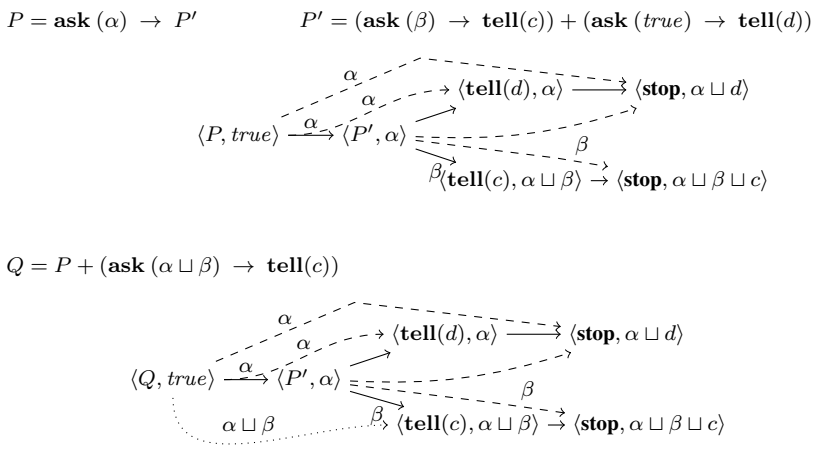

Figure 4.1.3: $LTS_{\Longrightarrow}(\{\langle P, true\rangle, \langle Q, true\rangle\})$ where $\Longrightarrow$ is defined in Table 4.1.2 (Milner's saturation method). The cycles from rule MR2 are omitted. The dashed transitions are those added by the rules in Table 4.1.2. The dotted transition is the (irredundant) one that $\langle Q, true\rangle$ can take but $\langle P, true\rangle$ cannot match, therefore showing that $\langle P, true\rangle \dot{\not\sim}_I^{\Longrightarrow} \langle Q, true\rangle$

However, $\langle P, true\rangle \dot{\not\sim}_I^{\Longrightarrow} \langle Q, true\rangle$. To prove that, we need to pick an irredundant transition from $\langle P, true\rangle$ or $\langle Q, true\rangle$ (after saturation) s.t. the other cannot match. Thus, take $\langle Q, true\rangle \xrightarrow{\alpha\sqcup\beta} \langle \mathbf{tell}(c), \alpha \sqcup \beta\rangle$ which is irredundant and given that $\langle P, true\rangle$ does not have a transition labeled with $\alpha \sqcup \beta$ then we know that we cannot find an irredundant bisimulation containing $(\langle P, true\rangle, \langle Q, true\rangle)$ therefore $\langle P, true\rangle \dot{\not\sim}_I^{\Longrightarrow} \langle Q, true\rangle$. Using the same reasoning we can also show that $\dot{\approx}_{sb}$ does not imply $\dot{\sim}_{sym}^{\Longrightarrow}$.

4.2 Reducing Weak Bisimilarity to Strong in CCP

In this section we shall provide a method for deciding weak bisimilarity in CCP. We shall proceed by redefining $\Longrightarrow$ in such a way that it is sound and complete for CCP. Then we prove that weak symbolic and irredundant bisimilarity coincide with $\dot{\approx}_{sb}$, i.e. $\dot{\approx}_{sb} = \dot{\sim}_{sym}^{\Longrightarrow} = \dot{\sim}_I^{\Longrightarrow}$. We therefore conclude that the partition refinement algorithm in Section 3.1 can be used to verify $\dot{\approx}_{sb}$.

4.2.1 Defining a new saturation method for CCP

If we analyze the counter-example to completeness (see Figure 4.1.1), one can see that the problem arises because of the nature of the labels in CCP, namely using this method $\langle \mathbf{ask}\ \alpha \ \rightarrow\ (\mathbf{ask}\ \beta \ \rightarrow\ \mathbf{stop}), \mathit{true}\rangle$ does not have a transition with $\alpha \sqcup \beta$ to $\langle \mathbf{stop}, \alpha \sqcup \beta\rangle$, hence that fact can be exploited to break the relation among the weak equivalences. Following this reasoning, instead of only forgetting about the silent actions we also take into account that labels in CCP can be added together. Thus we have a new rule that creates a new transition for each two consecutive ones, whose label is the lub of the labels in them. This method can also be thought as the reflexive and transitive closure of the labeled transition relation $\xrightarrow{\alpha}$. Such transition relation turns out to be sound and complete and it can be used to decide $\dot{\approx}_{sb}$.

The remaining part of this section is structured as follows. First we define a new saturation method and we proceed to prove that the weak barbs resulting from such method are consistent with that of CCP (as in Definition 2.4.7). Then under the assumption that $\longrightarrow$ is finitely branching we prove that $\Longrightarrow$ is also finitely branching. Moreover, we follow by showing how this method would be inaccurate in CCS-like formalisms since it could turn a finitely branching LTS into an infinitely branching one. With these elements we can prove soundness and completeness, which can be finally used to prove the correspondence among $\dot{\approx}_{sb}$, $\dot{\sim}^{\Longrightarrow}_{sym}$ and $\dot{\sim}^{\Longrightarrow}_{I}$.

Formally, our new transition relation $\Longrightarrow$ is defined by the rules in Table 4.2.1.

Remark 4.2.1. *For simplicity, we shall use the same notation ($\Longrightarrow$) we used for Milner's method (Table 4.1.2) to denote the new saturation method (Table 4.2.1).Consequently the definitions of weak barbs, symbolic and irredundant bisimilarity are now interpreted w.r.t. the new $\Longrightarrow$ as in Table 4.2.1 ($\Downarrow$, $\dot{\sim}^{\Longrightarrow}_{sym}$ and $\dot{\sim}^{\Longrightarrow}_{I}$ respectively).*

First we show that $\Downarrow$ coincides with $\Downarrow$ since a transition in $\Longrightarrow$ corresponds to a sequence of reductions.[1]

Lemma 4.2.2. $\gamma \longrightarrow^{*} \gamma'$ *iff* $\gamma \Longrightarrow \gamma'$.

[1]Notice that Lemma 4.2.2 also holds for the Milner's saturation method (Table 4.1.2)

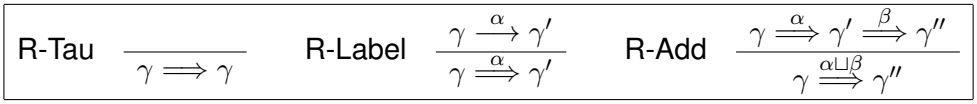

Table 4.2.1: New saturation method.

Proof. ($\Rightarrow$) We can decompose $\gamma \longrightarrow^* \gamma'$ as follows $\gamma \longrightarrow \gamma_1 \longrightarrow \ldots \longrightarrow \gamma_i \longrightarrow \gamma'$, now we proceed by induction on i. The base case is $i = 0$, then $\gamma \longrightarrow \gamma'$ and by rule R-Label we have $\gamma \Longrightarrow \gamma'$. For the inductive step, first we have by induction hypothesis that $\gamma \longrightarrow^i \gamma_i$ implies $\gamma \Longrightarrow \gamma_i$ (1), on the other hand, by rule R-Label on $\gamma_i \longrightarrow \gamma'$ we can deduce $\gamma_i \Longrightarrow \gamma'$ (2). Finally by R-Add on (1) and (2) $\gamma \Longrightarrow \gamma'$.

($\Leftarrow$) We proceed by induction on the depth of the inference of $\gamma \Longrightarrow \gamma'$. First, using R-Tau, we can directly conclude $\gamma \longrightarrow^* \gamma$. Then, using R-Label, $\gamma \Longrightarrow \gamma'$ implies that $\gamma \longrightarrow \gamma'$. Finally, using R-Add and since $\alpha \sqcup \beta = \mathit{true}$ implies $\alpha = \beta = \mathit{true}$, we get $\gamma \Longrightarrow \gamma'' \Longrightarrow \gamma'$ and by induction hypothesis this means that $\gamma \longrightarrow^* \gamma'' \longrightarrow^* \gamma'$ therefore $\gamma \longrightarrow^* \gamma'$.

□

Using this lemma, it is straightforward to see that the notions of weak barbs coincide.[2]

Lemma 4.2.3. *$\gamma \Downarrow_e$ iff $\gamma \not\downarrow_e$.*

Proof. First, let us assume that $\gamma \Downarrow_e$ then by definition $\gamma \longrightarrow^* \gamma' \downarrow_e$, and from Lemma 4.2.2 we know that $\gamma \Longrightarrow \gamma' \downarrow_e$, hence $\gamma \not\downarrow_e$. On the other hand, if $\gamma \not\downarrow_e$ then by definition $\gamma \Longrightarrow^* \gamma' \downarrow_e$, if we decompose these transitions then $\gamma \Longrightarrow \ldots \Longrightarrow \gamma'$, and from Lemma 4.2.2 $\gamma \longrightarrow^* \ldots \longrightarrow^* \gamma'$, therefore $\gamma \longrightarrow^* \gamma' \downarrow_e$, finally $\gamma \Downarrow_e$. □

As we shall see later on, the lemma above will be used to prove the correspondence between $\dot{\approx}_{sb}$ and $\dot{\sim}_I^{\Longrightarrow}$.

[2]Notice that Lemma 4.2.3 also holds for the Milner's saturation method (Table 4.1.2) because of Lemma 4.2.2

4.2.2 The new saturation method is finitely branching

An important property that the labeled transition system defined by the new relation $\Longrightarrow$ must fulfill is that it must be finitely branching, given that $\longrightarrow$ is also finitely branching. We prove this next but first let us introduce some useful notation.

Remark 4.2.4. *Notice that, because of Rule* R-Add*, one of the requirements for* $\Longrightarrow$ *to be finitely branching is that the number of reachable states of any configuration is finite. For this reason, in this section, we shall restrict to the finite fragment of CCP. However, even if we restrict the syntax of CCP to finite processes, for some constraint systems* $\longrightarrow$ *may be infinitely-branching due to Rule* LR3 *in Table 2.4.2; i.e., there may be infinitely many minimal labels allowing the transition. For this reason we shall sometimes explicitly assume that* $\longrightarrow$ *is defined in constraint systems that do not cause* $\longrightarrow$ *to be infinitely-branching.*

The set $\mathsf{Reach}(\gamma, \rightsquigarrow)$, defined below, represents the set of configurations which results after performing one step starting from a given configuration γ and using a relation $\rightsquigarrow$. Such set contains pairs of the form $[\gamma', \alpha]$ in which the first item (γ') is the configuration reached and the second one (α) is the label used for that purpose. Formally we have:

Definition 4.2.5 (Single-step Reachable Pairs)**.** *The set of Single-step reachable pairs is defined as* $\mathsf{Reach}(\gamma, \rightsquigarrow) = \{[\gamma', \alpha] \mid \gamma \overset{\alpha}{\rightsquigarrow} \gamma'\}$.

We can extend this definition to consider more than one step at a time. We will call this new set $\mathsf{Reach}^*(\gamma, \rightsquigarrow)$ and it is defined below.

Definition 4.2.6 (Reachable Pairs)**.** *The set of reachable pairs is defined as follows:* $\mathsf{Reach}^*(\gamma, \rightsquigarrow) = \{[\gamma', \alpha] \mid \exists \alpha_1, \ldots, \alpha_n.\ \gamma \overset{\alpha_1}{\rightsquigarrow} \ldots \overset{\alpha_n}{\rightsquigarrow} \gamma' \wedge \alpha = \alpha_n\}$.

Using the notation defined above, we shall define formally what we mean by finitely branching as follows.

Definition 4.2.7 (Finitely Branching)**.** *We say that a transition relation* $\rightsquigarrow$ *is finitely branching if for all* γ *we have* $|\mathsf{Reach}(\gamma, \rightsquigarrow)| < \infty$.

For convenience, in order to project the first or the second item of the reachable pairs we will define the functions $\mathcal{C}$ and $\mathcal{L}$ which, respectively, extract the configuration and the label respectively (hence the name).

Definition 4.2.8 (Configurations and Labels of a reachable pair)**.** *The functions $\mathcal{C}$ and $\mathcal{L}$ are defined as follows, $\mathcal{C}([\gamma, \alpha]) = \{\gamma\}$ and $\mathcal{L}([\gamma, \alpha]) = \{\alpha\}$. They extend to set of pairs as expected, namely given a set of pairs $S = \{p_1, \dots\}$ then $\mathcal{L}(S) = \bigcup_{p_i \in S} \mathcal{L}(p_i)$ and similarly for $\mathcal{C}$.*

Now, under the assumption that $\longrightarrow$ is finitely branching, we can observe that if the number of configurations that can be reached (in one or more steps) is finite, then the number of labels should also be finite.

Proposition 4.2.9. *Suppose $\longrightarrow$ is finitely branching. If $|\mathcal{C}(\mathsf{Reach}^*(\gamma, \longrightarrow))| < \infty$ for any γ, then $|\mathcal{L}(\mathsf{Reach}^*(\gamma, \longrightarrow))| < \infty$.*

Proof. From the hypothesis we know that there are finite γ' that can be reached, and using the assumption of being finitely branching we can see that from each of those γ' there are only finitely possible α, hence the conclusion. □

Finally, the nature of the labels in CCP is one of the reasons why our new transition system works. The following lemma illustrates the fact that when generating new labels, with the rule R-Add (Table 4.2.1) we will not add an infinite number of those. In the following, $C^{*\sqcup}$ will represent the Kleene closure over $\sqcup$ of the set of constraints C.

Lemma 4.2.10. *Given a set of constraints C, if $|C| < \infty$ then $|C^{*\sqcup}| < \infty$.*

Proof. Follows from the commutativity and idempotence of $\sqcup$ ($c \sqcup c = c$). □

Using these elements, the finitely branching property of $\Longrightarrow$ follows directly under the assumption that $\longrightarrow$ is finitely branching and our restriction to finite processes (see Remark 4.2.4).

Lemma 4.2.11. *If $\longrightarrow$ is finitely branching then $\Longrightarrow$ is finitely branching.*

Proof. (1) $|\mathcal{C}(\mathsf{Reach}^*(\gamma, \longrightarrow))| < \infty$ for every γ due to our restriction to finite processes and our assumption that $\longrightarrow$ is finitely branching. One can verify that $\mathcal{C}(\mathsf{Reach}^*(\gamma, \Longrightarrow)) = \mathcal{C}(\mathsf{Reach}^*(\gamma, \longrightarrow))$ hence $|\mathcal{C}(\mathsf{Reach}^*(\gamma, \Longrightarrow))| < \infty$.

(2) From the hypothesis and from Proposition 4.2.9 $|\mathcal{L}(\mathsf{Reach}^*(\gamma, \longrightarrow))| < \infty$.

(3) One can check that $\mathcal{L}(\mathsf{Reach}(\gamma, \Longrightarrow)) \subseteq \mathcal{L}(\mathsf{Reach}^*(\gamma, \longrightarrow))^{*\sqcup}$ therefore from Lemma 4.2.10 and (2) $|\mathcal{L}(\mathsf{Reach}(\gamma, \Longrightarrow))| < \infty$.
From (1) and (3) we can conclude that for any γ, $\Longrightarrow$ is finitely branching. □

4.2.3 A Remark about our Saturation in CCS

In this section the transitions, processes and relations are understood in the context of CCS [41]. It is worth noticing that we could use the saturation method mentioned in the previous section for other formalisms like CCS, but unlike in CCP it would not work as intended: now, the actions that a process can perform need to be sequences and the rules in Table 4.2.1 must be replaced by those in Table 4.2.2 (essentially, the lub operation $\sqcup$ of CCP is replaced by the concatenation of sequence).

$$\mathsf{RCCS1}\ \frac{}{P \stackrel{\tau}{\Longrightarrow}_{CCS} P} \qquad \mathsf{RCCS2}\ \frac{P \stackrel{s}{\longrightarrow}_{CCS} P'}{P \stackrel{s}{\Longrightarrow}_{CCS} P'}$$

$$\mathsf{RCCS3}\ \frac{P \stackrel{s}{\Longrightarrow}_{CCS} P' \stackrel{s'}{\Longrightarrow}_{CCS} P''}{P \stackrel{s.s'}{\Longrightarrow}_{CCS} P''}$$

Table 4.2.2: New labeled transition system for CCS. Let $s = a_1 \ldots a_n$ be a sequence of observable actions. For the Rule RCCS3 we assume that $s.\tau = \tau.s = s$.

Using these rules we can now define weak bisimilarity in terms of the new relation $\Longrightarrow_{CCS}$ as follows.

Definition 4.2.12 (CCS-Weak Bisimilarity)**.** *A symmetric relation $\mathcal{R}$ is a CCS-weak bisimulation if for every $(P, Q) \in \mathcal{R}$:*

- *If $P \stackrel{s}{\Longrightarrow}_{CCS} P'$ then there exists Q' s.t. $Q \stackrel{s}{\Longrightarrow}_{CCS} Q'$ and $(P', Q') \in \mathcal{R}$.*

(a) Original LTS (finitely branching). (b) Saturated LTS (infinitely branching).

Figure 4.2.1: CCS Process $P = a.P$ of Example 4.2.13

We say that P and Q are CCS-weakly bisimilar ($P \approx Q$) iff there is a CSS-weak bisimulation containing (P, Q).

This definition resembles the standard definition of strong bisimilarity, hence we could use the procedure to verify the strong version under $\Longrightarrow$ in order to verify the weak one. However, by applying the rules in Table 4.2.2, even for a finite LTS, we could end up adding an infinite number of transitions. The following example illustrates the problem.

Example 4.2.13. *Let us take a typical one-state finitely-branching CCS process with a single transition labeled with a into itself (See Figure 4.2.1-(a)). However, if we apply the rules in Table 4.2.2 for the example above, we end up adding an infinite number of transitions. Hence, becoming infinitely branching (see Figure 4.2.1-(b)).*

4.2.4 Soundness and Completeness

As mentioned before, soundness and completeness of the relation are the core properties when proving $\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_{I}$. We now proceed to show that our method enjoys these properties and they will allow us to prove the correspondence among the equivalences for the weak case.

Recall that in Definition 4.1.1 we introduced the formal definition of completeness, now we shall introduce the notion of soundness.

Definition 4.2.14 (Soundness)**.** *We say that* $\rightsquigarrow$ *is sound iff whenever* $\langle P, c\rangle \overset{\alpha}{\rightsquigarrow} \langle P', c'\rangle$ *then* $\langle P, c \sqcup \alpha\rangle \rightsquigarrow \langle P', c'\rangle$.

Below we shall prove that $\Longrightarrow$ from Table 4.2.1 is sound and complete.

Lemma 4.2.15 (Soundness of $\Longrightarrow$)**.** *If* $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$ *then* $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', c'\rangle$.

Proof. We proceed by induction on the depth of the inference of $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$.

- Using R-Tau we have $\langle P, c\rangle \Longrightarrow \langle P, c\rangle$ and the result follows directly given that $\alpha = true$.
- Using R-Label we have $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$ then $\langle P, c\rangle \overset{\alpha}{\longrightarrow} \langle P', c'\rangle$. By Lemma 2.4.24 (soundness of $\longrightarrow$) we get $\langle P, c \sqcup \alpha\rangle \longrightarrow \langle P', c'\rangle$ and finally by rule R-Label $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', c'\rangle$.
- Using R-Add then we have $\langle P, c\rangle \overset{\beta\sqcup\lambda}{\Longrightarrow} \langle P', c'\rangle$ then $\langle P, c\rangle \overset{\beta}{\Longrightarrow} \langle P'', c''\rangle \overset{\lambda}{\Longrightarrow} \langle P', c'\rangle$ where $\beta \sqcup \lambda = \alpha$. By induction hypothesis, $\langle P, c \sqcup \beta\rangle \Longrightarrow \langle P'', c''\rangle$ (1) and $\langle P'', c'' \sqcup \lambda\rangle \Longrightarrow \langle P', c'\rangle$ (2). By monotonicity on (1), $\langle P, c \sqcup \beta \sqcup \lambda\rangle \Longrightarrow \langle P'', c'' \sqcup \lambda\rangle$ and by rule R-Add on this transition and (2) then, given that $\beta \sqcup \lambda = \alpha$, we obtain $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', c'\rangle$.

□

Lemma 4.2.16 (Completeness of $\Longrightarrow$)**.** *If* $\langle P, c \sqcup a\rangle \Longrightarrow \langle P', c'\rangle$ *then there exist* α *and* b *s.t.* $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c''\rangle$ *where* $\alpha \sqcup b = a$ *and* $c'' \sqcup b = c'$.

Proof. Assuming that $\langle P, c \sqcup a\rangle \Longrightarrow \langle P', c'\rangle$ then, from Lemma 4.2.2, we can say that $\langle P, c \sqcup a\rangle \longrightarrow^* \langle P', c'\rangle$ which can be written as $\langle P, c \sqcup a\rangle \longrightarrow \ldots \longrightarrow \langle P_i, c_i\rangle \longrightarrow \langle P', c'\rangle$, we will proceed by induction on i.

- (Base Case) Assuming $i = 0$ then $\langle P, c \sqcup a\rangle \longrightarrow \langle P', c'\rangle$ and the result follows directly from Lemma 2.4.25 (Completeness of $\longrightarrow$) and R-Label.
- (Induction) Let us assume that $\langle P, c \sqcup a\rangle \longrightarrow^i \langle P_i, c_i\rangle \longrightarrow \langle P', c'\rangle$ then by induction hypothesis there exist β and b' s.t. $\langle P, c\rangle \overset{\beta}{\Longrightarrow} \langle P_i, c_i'\rangle$ (1)

where $\beta \sqcup b' = a$ and $c'_i \sqcup b' = c_i$. Now by completeness on the last transition $\langle P_i, \overbrace{c_i}^{c'_i \sqcup b'} \rangle \longrightarrow \langle P', c' \rangle$, there exists λ and b'' s.t. $\langle P_i, c'_i \rangle \xrightarrow{\lambda} \langle P', c'' \rangle$ where $\lambda \sqcup b'' = b'$ and $c'' \sqcup b'' = c'$, thus by rule R-Label we have $\langle P_i, c'_i \rangle \stackrel{\lambda}{\Longrightarrow} \langle P', c'' \rangle$ (2). We can now proceed to apply rule R-Add on (1) and (2) to obtain the transition $\langle P, c \rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c'' \rangle$ where $\alpha = \beta \sqcup \lambda$ and finally take $b = b''$, therefore the conditions hold $\alpha \sqcup b = \beta \sqcup \lambda \sqcup b'' = a$ and $c'' \sqcup b = c'' \sqcup b'' = c'$.

□

4.3 Deciding Weak Bisimilarity

In this section we shall show the main result of this chapter, a method for deciding $\dot{\approx}_{sb}$. Recall that $\dot{\approx}_{sb}$ is the standard weak bisimilarity for CCP [6], and it is defined in terms of $\longrightarrow$, therefore it does not depend on $\Longrightarrow$. We start from the fact that using CCP partition refinement (Section 3.1) one can check whether two configurations are irredundant bisimilar ($\dot{\sim}_I$). Following the same approach, we prove that $\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$ hence we give a reduction from $\dot{\approx}_{sb}$ to $\dot{\sim}^{\Longrightarrow}_{I}$. Finally, $\dot{\sim}^{\Longrightarrow}_{I}$ can be verified using a modified version of the CCP partition refinement as we shall see at the end of next section.

4.3.1 Weak Irredundant and Weak Saturated Bisimilarity coincide

First, given that $\Longrightarrow$ is sound and complete (Lemma 4.2.15 and Lemma 4.2.16), the correspondence between the symbolic and irredundant bisimilarity follows from the results in Section 3.2.1.

Corollary 4.3.1. *$\gamma \dot{\sim}^{\Longrightarrow}_{sym} \gamma'$ iff $\gamma \dot{\sim}^{\Longrightarrow}_{I} \gamma'$*

Finally, in the next two lemmata, we prove that $\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{sym}$.

Lemma 4.3.2. *If $\gamma \dot{\approx}_{sb} \gamma'$ then $\gamma \dot{\sim}^{\Longrightarrow}_{sym} \gamma'$*

Proof. We need to prove that $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle\}$ is a symbolic bisimulation over $\Longrightarrow$. The first condition (i) of the bisimulation follows directly from Lemma 4.2.3. As for (ii), let us assume that $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$ then by soundness of $\Longrightarrow$ we have $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', c'\rangle$, now by Lemma 4.2.2 we obtain $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', c'\rangle$. Given that $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$ then from the latter transition we can conclude that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\langle P', c'\rangle \dot{\approx}_{sb} \langle Q', d'\rangle$, hence we can use Lemma 4.2.2 again to deduce that $\langle Q, d \sqcup \alpha\rangle \Longrightarrow \langle Q', d'\rangle$. Finally, by completeness of $\Longrightarrow$, there exist β and b s.t. $t = \langle Q, d\rangle \overset{\beta}{\Longrightarrow} \langle Q', d''\rangle$ where $\beta \sqcup b = \alpha$ and $d'' \sqcup b = d'$, therefore $t \vdash_D \langle Q, d\rangle \overset{\alpha}{\rightsquigarrow} \langle Q', d'\rangle$ and $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$. □

Lemma 4.3.3. *If* $\gamma \dot{\sim}^{\Longrightarrow}_{sym} \gamma'$ *then* $\gamma \dot{\approx}_{sb} \gamma'$

Proof. We need to prove that $\mathcal{R} = \{(\langle P, c \sqcup a\rangle, \langle Q, d \sqcup a\rangle) \mid \langle P, c\rangle \dot{\sim}^{\Longrightarrow}_{sym} \langle Q, d\rangle\}$ is a weak saturated bisimulation. First, condition (i) follows form Lemma 4.2.3 and (iii) by definition of $\mathcal{R}$. Let us prove condition (ii), assume $\langle P, c \sqcup a\rangle \longrightarrow^* \langle P', c'\rangle$ then by Lemma 4.2.2 $\langle P, c \sqcup a\rangle \Longrightarrow \langle P', c'\rangle$. Now by completeness of $\Longrightarrow$ there exist α and b s.t. $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c''\rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$. Since $\langle P, c\rangle \dot{\sim}^{\Longrightarrow}_{sym} \langle Q, d\rangle$ then we know there exists a transition $t = \langle Q, d\rangle \overset{\beta}{\Longrightarrow} \langle Q', d'\rangle$ s.t. $t \vdash_D \langle Q, d\rangle \overset{\alpha}{\rightsquigarrow} \langle Q', d''\rangle$ and $(\mathbf{a})\langle P', c''\rangle \dot{\sim}^{\Longrightarrow}_{sym} \langle Q', d''\rangle$, by definition of $\vdash_D$ there exists b' s.t. $\beta \sqcup b' = \alpha$ and $d' \sqcup b' = d''$. Using soundness of $\Longrightarrow$ on t we get $\langle Q, d \sqcup \beta\rangle \Longrightarrow \langle Q', d'\rangle$, thus by Lemma 4.2.2 $\langle Q, d \sqcup \beta\rangle \longrightarrow^* \langle Q', d'\rangle$ and finally by monotonicity

$$\langle Q, d \sqcup \underbrace{\overbrace{\beta \sqcup b'}^{a} \sqcup b}_{\alpha}\rangle \longrightarrow^* \langle Q', \overbrace{d' \sqcup b'}^{d''} \sqcup b\rangle \tag{4.1}$$

Then, the transition $\langle P, c \sqcup a\rangle \longrightarrow^* \langle P', c'\rangle$ can be rewritten as $\langle P, c \sqcup a\rangle \longrightarrow^* \langle P', c'' \sqcup b\rangle$, and using (4.1), $\langle Q, d \sqcup a\rangle \longrightarrow^* \langle Q', d'' \sqcup b\rangle$. It is left to prove that $\langle P', c'' \sqcup b\rangle \mathcal{R} \langle Q', d'' \sqcup b\rangle$ which follows from $(\mathbf{a})$ and Corollary 3.2.15. □

Using Lemma 4.3.2 and Lemma 4.3.3 we obtain the following theorem.

Theorem 4.3.4. $\langle P, c\rangle \dot{\sim}^{\Longrightarrow}_{sym} \langle Q, d\rangle$ *iff* $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$

From the above results, we conclude that $\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{I}$.

4.3.2 Algorithm for weak bisimilarity in CCP

In this section we describe the decision procedure for verifying weak version of saturated bisimilarity ($\dot{\approx}_{sb}$). Given two configurations γ and γ', the first step is to build $G = LTS_{\longrightarrow}(IS)$ where $IS = \{\gamma, \gamma'\}$. We proceed to compute $G' = LTS_{\Longrightarrow}(IS)$, and then run an adaptation of Algorithm 3.1.2 on G'. The adaptation consists in using weak barbs ($\Downarrow$) instead of barbs ($\downarrow$) for the initial partition $\mathcal{P}^0$, and in using $\Longrightarrow$ as a parameter of Algorithm 4.3.1.

Algorithm 4.3.1 `weak-pr-ccp`$(IS, \rightsquigarrow)$

Initialization

1. Compute $IS^{\star}_{\rightsquigarrow}$ with the rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ defined in Table 3.1.1,
2. $\mathcal{P}^0 = \{B_1\} \ldots \{B_m\}$ is a partition of $IS^{\star}_{\rightsquigarrow}$ where γ and γ' are in B_i iff they satisfy the same *weak barbs ($\Downarrow_c$)*,

Iteration $\mathcal{P}^{n+1} := \mathbf{IR}_{\rightsquigarrow}(\mathcal{P}^n)$ as in Definition 3.1.9
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

Using this algorithm we can decide $\dot{\approx}_{sb}$ also with exponential time as we shall see below. First, let us determine the complexity of computing $LTS_{\Longrightarrow}(IS)$.

Proposition 4.3.5. *Assume that $\longrightarrow$ is finitely branching (Definition 4.2.7). Let $G = LTS_{\longrightarrow}(IS)$ and $G' = LTS_{\Longrightarrow}(IS)$. Let $N = |\mathtt{V}(G)| = |\mathtt{E}(G)|$, $N' = |\mathtt{V}(G')|$ and $M' = |\mathtt{E}(G)|$. We have that $N' = N$ and $M' = O(N^2)$.*

Proof. By construction of γ' the rules in Table 4.2.1 never add a new vertex, thus it follows directly that $N' = N$. As for the edges, the rules in Table 4.2.1 will add, at most, a transition from each element in $\mathtt{V}(G)$ to every other configuration in $\mathtt{V}(G)$. Since $\mathtt{V}(G) = N$ then the resulting transitions are $M' = O(N^2)$. □

We now show that the size of $IS^{\star}_{\Longrightarrow}$ may be exponential w.r.t. $|\mathtt{Config}_{\longrightarrow}(IS)|$ exploiting the same example as in Theorem 3.2.16.

Lemma 4.3.6. *There exists IS such that $|IS^{\star}_{\Longrightarrow}| = \Omega(2^{|\mathtt{Config}_{\longrightarrow}(IS)|})$.*

Proof. Using the same approach as in Theorem 3.2.16 □

We can now state correctness and complexity of `weak-pr-ccp`$(IS, \Longrightarrow)$.

Theorem 4.3.7. *Assume that $\longrightarrow$ is finitely branching (Definition 4.2.7). Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of* `weak-pr-ccp`$(IS, \Longrightarrow)$ *in Algorithm 4.3.1. If $IS^{\star}_{\Longrightarrow}$ is finite then the algorithm terminates and:*

- $\gamma \; \mathcal{P} \; \gamma'$ *iff* $\gamma \; \dot{\approx}_{sb} \; \gamma'$.
- `weak-pr-ccp`$(IS, \Longrightarrow)$ *may take exponential time in* $|\texttt{Config}_{\longrightarrow}(IS)|$.

Proof. The first point follows from Corollary 4.3.1 ($\dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$) and Theorem 4.3.4 ($\dot{\sim}^{\Longrightarrow}_{sym} = \dot{\approx}_{sb}$). For the second point, as for the strong case, the exponential time is due to construction of the set $IS^{\star}_{\Longrightarrow}$ in step 1 of `weak-pr-ccp`$(IS, \Longrightarrow)$, whose size is exponential in $|\texttt{Config}_{\longrightarrow}(IS)|$ as shown in Lemma 4.3.6). □

As we did for the algorithm for strong bisimilarity (Algorithm 3.1.2 we need to prove that if the set $\texttt{Config}_{\longrightarrow}(IS)$ of all configurations reachable from IS is finite, then $IS^{\star}_{\longrightarrow}$ is finite. As we said before this condition can be easily guaranteed by restricting to a finite fragment of CCP.

Theorem 4.3.8. *Let IS be a set of configurations. If $\texttt{Config}_{\longrightarrow}(IS)$ is finite, then $IS^{\star}_{\Longrightarrow}$ is finite.*

Proof. Follows from Theorem 3.2.17 since $\Longrightarrow$ does not add any new state. □

4.4 Summary and Related Work

We showed that the weak transition relation using Milner's saturation method is not complete for CCP (in the sense of Definition 4.1.1). This implied that $\dot{\approx}_{sb}$ (Definition 2.4.10) cannot be computed immediately by using the CCP partition refinement algorithm for (strong) bisimilarity CCP on the saturated transition relation. We then introduced a new transition relation using a different saturation mechanism and showed that it is complete for CCP and also that it is finitely branching. As a consequence, we also showed that the CCP partition refinement can be used to compute $\dot{\approx}_{sb}$ using this new relation. Likewise, we have shown that although this new saturation method elaborated for CCP could be used for any other formalisms such as CCS, it would not work as desired because it could

transform a finitely branching LTS into an infinitely branching one. To the best of our knowledge, this is the first approach to verifying weak bisimilarity for CCP.

CCP is not the only formalism where weak bisimilarity cannot be naively reduced to the strong one. Probably the first case in literature can be found in [67] that introduces an algorithm for checking weak open bisimilarity of π-calculus. This algorithm is rather different from ours, since it is on-the-fly [25] and thus it checks the equivalence of only two given states (while our algorithm, and more generally all algorithms based on partition refinement, check the equivalence of all the states of a given LTS). These algorithms have a polynomial upper bound time complexity. The present algorithm has an exponential time complexity. However, they deal with very different formalisms. Also [9] defines weak labeled transitions following the above-mentioned standard method which does not work in the CCP case.

Analogous problems to the one discussed in this paper arise in Petri nets [65, 18], in tile transition systems [29, 17] and, more generally, in the theory of reactive systems [64, 39, 33] (the interested reader is referred to [66] for an overview). In all these cases, labels form a monoid where the neutral element is the label of internal transitions. In the case of CCS, the fact that a system may perform a transition with a composed label $a; b$ means that it may perform first a transition with a and then a transition with b. This property, which in tile systems [29, 17] is known as vertical decomposition, does not hold for CCP and for the other formalisms mentioned above. As a consequence of this fact, when reducing from weak to strong bisimilarity, one needs to close the transitions with respect to the composition of the monoid (and not only with respect to the neutral element). However, usually, labels composition is not idempotent (as it is for CCP) and thus a finite LTS might be transformed into an infinite one. For this reason, this procedure applied to the afore mentioned cases is not effective for automatic verification.

Publications from this Chapter

The material of this chapter has been published in the following papers:

- [54] **L. Pino**, A. Aristizabal, F. Bonchi, F. Valencia, *Weak CCP bisimilarity with strong procedures*. Science of Computer Programming, 100(0):84-104, 2015.
  DOI: `http://dx.doi.org/10.1016/j.scico.2014.09.007`.

- [7] A. Aristizabal, F. Bonchi, **L. Pino**, F. Valencia, *Reducing Weak to Strong Bisimilarity in CCP*, in: M. Carbone, I. Lanese, A. Silva, A. Sokolova (Eds.), Proceedings of the 5th Interaction and Concurrency Experience (ICE 2012), volume 104 of Electronic Proceedings in Theoretical Computer Science, 2012, pp. 2-16.
  DOI: `http://dx.doi.org/10.4204/EPTCS.104.2`.

Chapter 5

Computing bisimilarity in Choice-Free CCP

In the previous chapters we presented two methods for computing strong saturated barbed bisimilarity $\dot{\sim}_{sb}$ (Definition 2.4.9) and its weak version $\dot{\approx}_{sb}$ (Definition 2.4.10). This is achieved by adapting the well-known partition refinement algorithm from Section 2.3 together with the concept of irredundant bisimilarity $\dot{\sim}_I$ (Definition 3.2.1). This chapter is devoted to the development of more efficient methods for computing $\dot{\approx}_{sb}$. Recall that, from the Proposition 2.4.21 we know that, in the choice-free fragment (CCP$\backslash$+), $\dot{\approx}_{sb}$ coincides with the observational equivalence $\sim_o$ (Definition 2.4.19).

In Section 5.1 we begin by proving that the partition refinement for CCP from Section 3.1 is inefficient even for the choice-free fragment (CCP$\backslash$+). Then we explain some properties which are particular to CCP$\backslash$+. Such properties are used to develop a polynomial procedure, based on the partition refinement algorithm, for checking observational equivalence in CCP$\backslash$+. The intuition is that, in CCP$\backslash$+, irredundant transitions can be precomputed, thus we can avoid the exponential explosion by deleting them before running the partition refinement. In Section 5.2 we introduce our second, more efficient, method for deciding observational equivalence by using the compact input-output sets. Roughly speaking, the idea is to find a canonical representation of the set of outputs obtained after running the initial configurations with every possible input. This set is then compared to

check whether or not the initial configurations have the same behavior. Finally, in Section 5.3, we show how the procedure from Section 5.1 can be used as a sort of heuristic for checking $\dot{\approx}_{sb}$ in the full CCP.

5.1 Using Partition Refinement in choice-free CCP

In the previous chapter, we presented a procedure to verify $\dot{\approx}_{sb}$ for CCP and we saw how this method takes exponential time (in the size of the LTS) to check whether two configurations are weakly bisimilar.

Remark 5.1.1. *While some of our results in previous chapters regarding the different bisimilarity notions are true even for infinite CCP processes, in this chapter we shall focus on a finite CCP since we shall only deal with new algorithms. Notice that this is the same restriction we imposed for the correctness (Theorem 3.2.16 and 4.3.7) of the partition refinement algorithms defined previously (Algorithm 3.1.2 and 4.3.1).*

In this section, we will explore what happens with such procedure when we restrict ourselves to CCP\+. We shall see that $\texttt{pr-ccp}(IS, \longrightarrow)$ may also be exponential time for inputs from the CCP\+ fragment.

Let us consider the following CCP\+ construction.

Example 5.1.2. *Let $n > 0$. We define $P^n = P^n_0$ with P^n_i, for $i \in \{0, \ldots, n-1\}$, given by:*

$$P^n_i = (\mathbf{ask}\ (a_i)\ \rightarrow\ (\mathbf{ask}\ (b_i)\ \rightarrow\ P^n_{i+1})) \parallel (\mathbf{ask}\ (b_i)\ \rightarrow\ \mathbf{stop})$$

and $P^n_n = \mathbf{tell}(b_n)$. Furthermore, we assume that for all $i \in \{0, \ldots, n-1\}$ we have $a_i \sqsubseteq b_i$ and for all $j \in \{0, \ldots, n-1\}$ if $i \neq j$ then $a_i \not\sqsubseteq a_j$ and $b_i \not\sqsubseteq b_j$. The LTS for $\langle P^n, true\rangle$ is illustrated in Figure 5.1.1.

One can verify that by taking $IS = \{\langle P^n, true\rangle\}$ as in the example above, then the size of $IS^{\star}_{\longrightarrow}$ in Algorithm 3.1.2 grows exponentially with n, essentially because of the rule $(\mathsf{RD}^{IS}_{\longrightarrow})$.

$$P^n = (\mathbf{ask}\ (a_0)\ \rightarrow\ (\mathbf{ask}\ (b_0)\ \rightarrow\ P_1^n))\ \|\ (\mathbf{ask}\ (b_0)\ \rightarrow\ \mathbf{stop})$$
$$LP_0^n = (\mathbf{ask}\ (b_0)\ \rightarrow\ P_1^n)\ \|\ (\mathbf{ask}\ (b_0)\ \rightarrow\ \mathbf{stop})$$
$$RP_0^n = (\mathbf{ask}\ (a_i)\ \rightarrow\ (\mathbf{ask}\ (b_i)\ \rightarrow\ P_{i+1}^n))\ \|\ \mathbf{stop})$$
$$LLP_0^n = P_1^n\ \|\ (\mathbf{ask}\ (b_0)\ \rightarrow\ \mathbf{stop})$$
$$LRP_0^n = (\mathbf{ask}\ (b_0)\ \rightarrow\ P_1^n)\ \|\ \mathbf{stop}$$

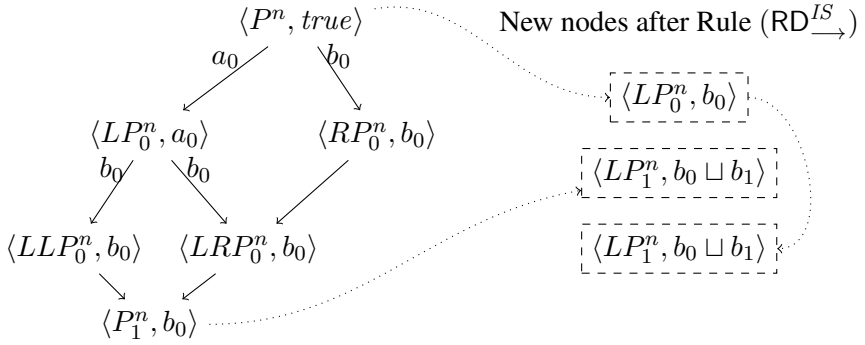

Figure 5.1.1: $LTS_{\longrightarrow}(IS)$ where $IS = \{\langle P^n, true\rangle\}$ as in Example 5.1.2. The configurations in the right part are generated by $(\mathsf{RD}^{IS}_{\longrightarrow})$ applied to the source nodes of the dotted arrows. Some transitions and **stop** processes were omitted for clarity.

Proposition 5.1.3. *Let $\gamma = \langle P^n, true\rangle$ and take $IS = \{\gamma\}$. Let $\mathcal{P}$ be the output of $\texttt{pr-ccp}(IS, \longrightarrow)$ using Algorithm 3.1.2, then $\texttt{pr-ccp}(IS, \longrightarrow)$ takes at least exponential time in n.*

Proof. Following the same approach of the proof of Theorem 3.2.16 notice that that size of the set of reachable states $|\texttt{Config}_{\longrightarrow}(IS)|$ is $6n$. By using the rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RS}^{IS}_{\rightsquigarrow})$ and $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ in Table 3.1.1 with $\rightsquigarrow = \longrightarrow$ and also $IS = \{\langle P^n, true\rangle\}$, we obtain an $IS^{\star}_{\longrightarrow}$ whose size is given by the following recurrence relation, for $n > 0$: $f(n) = 2f(n-1) + 6$ with $f(0) = 2$. In this case, there are six states per level as well as two copies of $\langle LP_1^n, b_0 \sqcup b_1\rangle$ which generate $f(n-1)$ states (one level less). By solving the recurrence we can conclude that $f(n) = \Omega(2^n)$ thus constructing $IS^{\star}_{\Longrightarrow}$ takes at least exponential time in n. □

The main problem is that the procedure does not distinguish between choice-free processes and the normal CCP processes. Therefore, it is unable to exploit

the underlying properties of CCP\+ and the algorithm will perform (in the worst-case) inherently the same as for the full CCP, as evidenced in the example above.

5.1.1 Properties of CCP without choice

In this section we will state some features that (unlike the full CCP) this fragment possess. The first property is that CCP\+ is *confluent* as shown in Proposition 2.4.6. Intuitively, in CCP\+, if from a given configuration we have two possible reductions ($\longrightarrow$), then we are guaranteed that they will coincide at some point of the computation. Recall that $Conf_{\texttt{CCP}\backslash+}$ is the set of all CCP\+ configurations, i.e. configurations whose process is choice-free.

Before discussing the second property, we need to introduce some notation. We shall call *derivatives* (of γ) the successors reached via (zero or more) reductions ($\longrightarrow^*$) starting from a given configuration γ.

Definition 5.1.4 (Derivatives)**.** *The derivatives of a configuration γ, written* $\mathsf{Deriv}(\gamma)$*, are defined as* $\mathsf{Deriv}(\gamma) = \{\gamma' \mid \gamma \longrightarrow^* \gamma'\}$.

Using this notation, we can now state another property of CCP\+. A CCP\+ configuration is weakly bisimilar to all its derivatives.

Lemma 5.1.5. *Let $\gamma \in Conf_{\texttt{CCP}\backslash+}$. For all $\gamma' \in \mathsf{Deriv}(\gamma)$ we have $\gamma \dot{\approx}_{sb} \gamma'$.*

Proof. (i) If $\gamma_1 \Downarrow_e$ then by definition $\gamma_1 \longrightarrow^* \gamma_1' \downarrow_e$. By confluence (Proposition 2.4.6) $\gamma_1' \longrightarrow^* \gamma_3$ and thus $\gamma_3 \downarrow_e$ (since constraints can only be added). Since $\gamma_2 \longrightarrow^* \gamma_3 \downarrow_e$ we conclude that $\gamma_2 \Downarrow_e$.

(ii) If $\gamma_1 \longrightarrow^* \gamma_1'$, then by confluence $\gamma_1' \longrightarrow^* \gamma_3$ and therefore $(\gamma_1', \gamma_2) \in \mathcal{R}$.

(iii) Finally, let $\gamma_1 = \langle P_1, c_1\rangle$ and $\gamma_2 = \langle P_2, c_2\rangle$. If $\langle P_1, c_1\rangle \longrightarrow^* \langle P_3, c_3\rangle$ and $\langle P_2, c_2\rangle \longrightarrow^* \langle P_3, c_3\rangle$, then $\langle P_1, c_1 \sqcup e\rangle \longrightarrow^* \langle P_3, c_3 \sqcup e\rangle$ and $\langle P_2, c_2 \sqcup e\rangle \longrightarrow^* \langle P_3, c_3 \sqcup e\rangle$ and thus $(\langle P_1, c_1 \sqcup e\rangle, \langle P_2, c_2 \sqcup e\rangle) \in \mathcal{R}$.

□

The proof above relies on the intrinsic confluent nature of CCP\+ (Proposition 2.4.6) and this lemma will be central for the results we will present next. In the next section we shall take advantage of these properties to check $\dot{\approx}_{sb}$ for CCP\+ configurations.

5.1.2 Optimizing partition refinement for choice-free CCP

We presented how the partition refinement for CCP performs for CCP\+ as well as some properties of the configurations of this fragment. In this section, using such features, we shall show that the complexity of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$ can be improved, thus we can check $\dot{\approx}_{sb}$ in a more efficient manner.

Due to the nature of CCP\+, determining which are the redundant transitions w.r.t. $\dot{\approx}_{sb}$ (Definition 3.1.6) becomes an easier task. As we explained in Section 3.1, the purpose of rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ from Table 3.1.1 is to add some configurations to $IS^{\star}_{\rightsquigarrow}$ that will be used to check redundancy at each iteration of Algorithm 3.1.2. In CCP\+ these additional configurations are not necessary. But before we arrive to this let us introduce some definitions first.

Definition 5.1.6 (Maximal Weak Transition)**.** *We say that γ goes with α to γ' with a* maximal weak transition*, written $\gamma \stackrel{\alpha}{\Longrightarrow}_{\max} \gamma'$, iff $\gamma \stackrel{\alpha}{\Longrightarrow} \gamma' \not\longrightarrow$.*

The definition above reflects the fact that when $\gamma \stackrel{\alpha}{\Longrightarrow}_{\max} \gamma'$ then γ' has no more information to deduce without the aid of the environment, namely no further reduction ($\longrightarrow$) is possible. As $\Longrightarrow$, the maximal weak transition relation $\Longrightarrow_{\max}$ is sound and complete.

Lemma 5.1.7. *(Soundness) If $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow}_{\max} \langle P', c'\rangle$ then $\langle P, c \sqcup \alpha\rangle \Longrightarrow_{\max} \langle P', c'\rangle$. (Completeness) If $\langle P, c \sqcup a\rangle \Longrightarrow_{\max} \langle P', c'\rangle$ then there exists α and b such that $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow}_{\max} \langle P', c''\rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$.*

Proof. Follows from the soundness and completeness of $\Longrightarrow$ (Lemma 4.2.15 and 4.2.16) and from the fact that $LTS_{\longrightarrow}(\{\langle P, c\rangle\})$ is finite. □

As one would expect, $\Longrightarrow_{\max}$ can also be used to compute $\dot{\approx}_{sb}$ and the complexity of the procedure is similar to the case of $\Longrightarrow$ (Theorem 4.3.7).

Theorem 5.1.8. *Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$, let $\mathcal{P}$ be the output $\texttt{weak-pr-ccp}(IS, \Longrightarrow_{\max})$ using Algorithm 4.3.1. If $IS^{\star}_{\Longrightarrow_{\max}}$ is finite then the algorithm terminates and:*

- *$\gamma \mathcal{P} \gamma'$ iff $\gamma \dot{\approx}_{sb} \gamma'$.*
- *$\texttt{weak-pr-ccp}(IS, \Longrightarrow_{\max})$ may take exponential time in $|\texttt{Config}_{\longrightarrow}(IS)|$.*

Proof. Follows from the correctness of $\Longrightarrow_{\max}$ (Lemma 5.1.7), the results in Chapter 4 and Theorem 4.3.7. □

Nevertheless, in CCP\+, the maximal weak transitions $\Longrightarrow_{\max}$ satisfy a particular property that allow us to erase the redundant transitions w.r.t. $\dot{\approx}_{sb}$ before computing $\dot{\approx}_{sb}$ itself.

Proposition 5.1.9. *Let* $\gamma = \langle P, c\rangle \in Conf_{\texttt{CCP}\backslash+}$. *Let* $t_1 = \gamma \stackrel{\alpha}{\Longrightarrow}_{\max} \langle P_1, c_1\rangle$ *and* $t_2 = \gamma \stackrel{\beta}{\Longrightarrow}_{\max} \langle P_2, c_2\rangle$. *We have that* $\alpha \sqsubset \beta$ *and* $\langle P_1, c_1 \sqcup \beta\rangle \longrightarrow^* \langle P', c_2\rangle \not\longrightarrow$ *iff* $t_1 \succ_{\dot{\approx}_{sb}} t_2$.

Proof. (⇒) By soundness on t_1 we have $\langle P, c \sqcup \alpha\rangle \Longrightarrow_{\max} \langle P_1, c_1\rangle$ then by definition $\langle P, c\sqcup\alpha\rangle \Longrightarrow \langle P_1, c_1\rangle$ now by monotonicity $\langle P, c\sqcup\beta\rangle \Longrightarrow \langle P_1, c_1\sqcup\beta\rangle$ and then $\langle P, c\sqcup\beta\rangle \longrightarrow^* \langle P_1, c_1\sqcup\beta\rangle$ then by Lemma 5.1.5 $\langle P, c\sqcup\beta\rangle \dot{\approx}_{sb} \langle P_1, c_1\sqcup\beta\rangle$. Using a similar reasoning on t_2 we can conclude that $\langle P, c\sqcup\beta\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$ and by transitivity $\langle P_1, c_1 \sqcup \beta\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$. Finally take $t' = (\gamma, \beta, \langle P_1, c_1 \sqcup \beta\rangle)$, hence we can conclude that $t_1 \succ_{\dot{\approx}_{sb}} t_2$ since $t_1 \succ_D t'$ and $\langle P_1, c_1 \sqcup \beta\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$.

(⇐) Assume that $t_1 \succ_{\dot{\approx}_{sb}} t_2$ then there exists $t' = (\gamma, \beta, \langle P_1, c'\rangle)$ such that $t_1 \succ_D t'$ and $\langle P_1, c'\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$. By $t_1 \succ_D t'$ we know that $\alpha \sqsubset \beta$ and $c' = c_1 \sqcup \beta$. Now since $\langle P_2, c_2\rangle \not\longrightarrow$ by definition of $\Longrightarrow_{\max}$, therefore by condition (i) of $\dot{\approx}_{sb}$ we have $c' \sqsubseteq c_2$. Moreover, $\langle P_1, c'\rangle \longrightarrow^* \langle P', c_3\rangle$ where $c_2 \sqsubseteq c_3$. By contradiction let $c_2 \neq c_3$ then $c_2 \sqsubset c_3$, thus there is e s.t. $\langle P_1, c'\rangle \Downarrow_e$ but since $\langle P_2, c_2\rangle \not\longrightarrow$ then $\langle P_2, c_2\rangle \not\Downarrow_e$ and so $\langle P_1, c'\rangle \not\dot{\approx}_{sb} \langle P_2, c_2\rangle$, an absurd. Thus $c_3 = c_2$ hence $\langle P_1, c'\rangle \longrightarrow^* \langle P', c_2\rangle \not\longrightarrow$. □

Using this property we can define a new procedure for deciding $\dot{\approx}_{sb}$ that does not use Rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ since redundancy can be checked and erased using Proposition 5.1.9 (Algorithm 5.1.1, Step 2).

Algorithm 5.1.1 `weak-pr-dccp`(IS)

Initialization

1. Compute $G = LTS_{\Longrightarrow_{\max}}(IS)$ using the rules $(\mathsf{IS}^{IS}_{\Longrightarrow_{\max}})$ and $(\mathsf{RS}^{IS}_{\Longrightarrow_{\max}})$,
2. $G' = \texttt{remRed}(G)$ where the graph $\texttt{remRed}(G)$ results from removing from G the redundant transitions w.r.t. $\dot{\approx}_{sb}$,
3. $\mathcal{P}^0 = \{B_1\} \ldots \{B_m\}$ is a partition of $\texttt{V}(G')$ where γ and γ' are in B_i iff they satisfy the same weak barbs ($\Downarrow_e$),

Iteration $\mathcal{P}^{n+1} := \mathbf{F}_{\Longrightarrow_{\max}}(\mathcal{P}^n)$ as defined in Definition 2.3.2
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

The key idea is that in order to compute $\dot{\approx}_{sb}$, with the redundancy removed, it suffices to refine the partitions using $\mathbf{F}_{\Longrightarrow_{\max}}(\mathcal{P})$ (defined by Definition 2.3.2) instead of $\mathbf{IR}_{\Longrightarrow_{\max}}(\mathcal{P})$. Algorithm 5.1.1 can be used to decide $\dot{\approx}_{sb}$ for configurations in $Conf_{\texttt{CCP}\setminus+}$ with polynomial time.

Theorem 5.1.10. *Let γ and γ' be two CCP\+ configurations. Let $IS = \{\gamma, \gamma'\}$, let $\mathcal{P}$ be the output of* `weak-pr-dccp`(IS) *in Algorithm 5.1.1 and let $N = |\texttt{Config}_{\longrightarrow}(IS)|$. If $IS^{\star}_{\Longrightarrow_{\max}}$ is finite then the algorithm terminates and:*

- *$\gamma \; \mathcal{P} \; \gamma'$ iff $\gamma \dot{\approx}_{sb} \gamma'$.*
- `weak-pr-dccp`(IS) *takes $O(N^3)$ time and uses $O(N^2)$ space.*

Proof. The first item follows from the Theorem 4.3.7 and Proposition 5.1.9. As for the second item:
(Step 1) $G = LTS_{\Longrightarrow_{\max}}(IS)$ takes $O(N^2)$ time and space since $\Longrightarrow_{\max}$ will add, at most, a transition from each element in $\texttt{V}(G)$ to every other configuration in $\texttt{V}(G)$ and $|\texttt{V}(G)| = |\texttt{Config}_{\longrightarrow}(IS)| = N$.
(Step 2) Each node in $\texttt{V}(G)$ has at most $N - 1$ outgoing transitions, then $G' = \texttt{remRed}(G)$ takes $O((N-1) * (N-1)) = O(N^2)$ per node, thus this step takes $O(N^2 * N) = O(N^3)$ time.
(Step 3) $\mathcal{P}^0$ can be created in $O(N^2)$ by definition of $\Longrightarrow_{\max}$.
(Iteration) Using the procedure from Tarjan et al. [50], this step takes $O(|E| \log |V|)$ time and uses $O(|E|)$ space. Therefore, since $|\texttt{V}(G)| = N$ and $|\texttt{E}(G)| = N^2$,

hence we have $O(N^2 \log N)$ and $O(N^2)$ space.
We can conclude that `weak-pr-dccp`(IS) takes $O(N^3)$ time and uses $O(N^2)$ space. □

Thanks to Proposition 5.1.9, by removing redundant transitions, we can solve the problem of checking bisimilarity for CCP\+ with the standard solutions for checking bisimilarity. In Algorithm 5.1.1, we have used the "classical" partition refinement, but different, more effective solutions, are possible. For instance, executing the algorithm in [23] (after having removed all the redundant transitions) would require at most $O(|E| + |V|)$ steps. Note however that, due to the closure needed for weak transitions (Table 4.2.1), $|E|$ is usually quadratic w.r.t. the number of states $|V|$. In the following section, we introduce a novel procedure which avoids such expensive closure.

5.2 The compact input-output sets approach

In the previous section we improved the CCP exponential-time decision procedure for $\dot{\approx}_{sb}$ to obtain a polynomial-time procedure for the special case of the choice-free fragment CCP\+. Recall that in CCP\+, the relation $\dot{\approx}_{sb}$ coincides with the standard notion of observational equivalence.

In this section, we will present an alternative approach for verifying observational equivalence for CCP\+ that improves on the time and space complexity of Algorithm 5.1.1.

Roughly speaking our approach consists in reducing the problem of whether two given CCP\+ configurations γ, γ' are in $\dot{\approx}_{sb}$ to the problem of whether γ and γ' have the same minimal finite representation of the set of weak barbs they satisfy in every possible context.

5.2.1 Weak bisimilarity and barb equivalence

First we will show that, in CCP\+, we can give characterization of $\dot{\approx}_{sb}$ in terms of the simpler notion of weak-barb equivalence defined below. Intuitively, two configurations are saturated weakly bisimilar if and only if for *every* possible aug-

mentation of their stores, the resulting configurations satisfy the same weak barbs. More precisely,

Definition 5.2.1 (Barb equivalence)**.** *$\langle P, c\rangle$ and $\langle Q, d\rangle$ are (weak) barbed equivalent, written $\langle P, c\rangle \sim_{wb} \langle Q, d\rangle$, iff*

$$\forall e, \alpha \in Con_0.\ \langle P, c \sqcup e\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d \sqcup e\rangle \Downarrow_\alpha$$

Let us give an example.

Example 5.2.2. *Let $P = \mathbf{tell}(true)$ and $Q = \mathbf{ask}\ (c) \rightarrow \mathbf{tell}(d)$. We can show that $\langle P, true\rangle \sim_{wb} \langle Q, true\rangle$ when $d \sqsubseteq c$ (as in Example 2.4.12). One can check that for all c' we have $\langle P, c'\rangle \Downarrow_{c'}$ and $\langle Q, c'\rangle \Downarrow_{c'}$. Notice that whenever c is entailed then $\mathbf{tell}(d)$ does not add any more information since $d \sqsubseteq c$.*

The full characterization of $\dot{\approx}_{sb}$ in terms of weak-barbed equivalence is given next. Notice that the following theorem uses Lemma 5.1.5 which itself depends on the confluent nature of CCP\+.

Theorem 5.2.3. *Let $\langle P, c\rangle$ and $\langle Q, d\rangle$ be CCP\+ configurations. $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$ iff $\langle P, c\rangle \sim_{wb} \langle Q, d\rangle$*

Proof. **($\Rightarrow$)** Assume that $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$ then by condition (i) of $\dot{\approx}_{sb}$ (Definition 2.4.9) we have $\forall \alpha \in Con_0. \langle P, c\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d\rangle \Downarrow_\alpha$, hence in combination with condition (iii) we can conclude $\langle P, c \sqcup e\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d \sqcup e\rangle \Downarrow_\alpha$.

($\Leftarrow$) Let $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \forall e, \alpha \in Con_0.\ \langle P, c \sqcup e\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d \sqcup e\rangle \Downarrow_\alpha\}$, we prove that $\mathcal{R}$ is a weak saturated barbed bisimulation:
(i) Take $e = true$ then $\forall \alpha \in Con_0. \langle P, c\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d\rangle \Downarrow_\alpha$.
(ii) Assume that $\langle P, c\rangle \longrightarrow^* \langle P', c'\rangle$, by Lemma 5.1.5 $\langle P, c\rangle \dot{\approx}_{sb} \langle P', c'\rangle$ hence by ($\Rightarrow$) we can conclude that $\langle P', c'\rangle \mathcal{R} \langle Q, d\rangle$.
(iii) Assume $\langle P, c\rangle \mathcal{R} \langle Q, d\rangle$ then for all e' we have $\langle P, c \sqcup e'\rangle \mathcal{R} \langle Q, d \sqcup e'\rangle$ just by taking $e = e'$. □

We shall show a compact representation of the set of weak barbs of a configuration under any possible context. First we introduce some convenient notation for this purpose. The set $[\![\langle P, c\rangle]\!]$ will contain pairs of the form (α, e).

Definition 5.2.4 (Input-Output set)**.** *The* input-output *set of a given configuration* $\langle P, c\rangle$ *is defined as follows:*

$$[\![\langle P, c\rangle]\!] \stackrel{\text{def}}{=} \{(\alpha, e) \mid \langle P, c \sqcup \alpha\rangle \Downarrow_e\}$$

Let us give an example.

Example 5.2.5. *Let* $\gamma = \langle \mathbf{ask}\ a \ \rightarrow\ \mathbf{tell}(b), \mathit{true}\rangle$ *where* $b \not\sqsupseteq a$*.*

$$[\![\gamma]\!] = \{(\alpha, \alpha) | \alpha \sqsubset a \text{ or } a \not\sqsupseteq \alpha\} \cup \{(\beta, \beta \sqcup b) | a \sqsubseteq \beta\}$$

Intuitively, each pair $(\alpha, e) \in [\![\langle P, c\rangle]\!]$ denotes a stimulus-response, or input-output, interaction of $\gamma = \langle P, c\rangle$: If the environment adds α to the store of γ, the resulting configuration $\langle P, c \sqcup \alpha\rangle$ may evolve, without any further interaction with the environment, into a configuration whose store entails e. In other words $\langle P, c \sqcup \alpha\rangle \ \Downarrow_e$. We can think of e as piece of information that $\langle P, c \sqcup \alpha\rangle$ may produce.

The following corollary is an immediate consequence of the definitions.

Corollary 5.2.6. $[\![\langle P, c\rangle]\!] = [\![\langle Q, d\rangle]\!]$ *iff* $\langle P, c\rangle \sim_{wb} \langle Q, d\rangle$

Proof. ($\Rightarrow$) Assume $(\beta, a) \in [\![\langle P, c\rangle]\!]$ then $\langle P, c \sqcup \beta\rangle \ \Downarrow_a$. Hence, we can take $e = \beta$ and by hypothesis $\langle Q, d \sqcup \beta\rangle \ \Downarrow_a$ therefore $(\beta, a) \in [\![\langle Q, d\rangle]\!]$.

($\Leftarrow$) Assume that $\langle P, c \sqcup \beta\rangle \ \Downarrow_a$ for some β and a, then $(\beta, a) \in [\![\langle P, c\rangle]\!]$ and by hypothesis $(\beta, a) \in [\![\langle Q, d\rangle]\!]$ therefore by definition $\langle Q, d \sqcup \beta\rangle \ \Downarrow_a$.

□

We now introduce the notion of relevant input-output pair.

Definition 5.2.7 (Relevant Pair)**.** *Let* (α, e) *and* (β, e') *be elements from* $Con_0 \times Con_0$*. We say that* (α, e) *is more* relevant *than* (β, e')*, written* $(\alpha, e) \succeq (\beta, e')$*, iff* $\alpha \sqsubseteq \beta$ *and* $e' \sqsubseteq (e \sqcup \beta)$*. Similarly, given* $p = (\beta, e')$ *s.t.* $p \in \mathcal{S}$*, we say that the pair* p *is* irrelevant *in* $\mathcal{S}$ *if there is a pair* $(\alpha, e) \in \mathcal{S}$ *more relevant than* p*, else* p *is said to be* relevant *in* $\mathcal{S}$*.*

Let us illustrate this with an example.

Example 5.2.8. *Let* $\mathcal{S} = \{(true, true), (\alpha, \alpha), (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)\}$ *where* $\alpha, \beta, c \in Con_0$ *are not related to each other. Notice that* $(true, true) \succeq (\alpha, \alpha)$ *however* $(true, true) \not\succeq (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)$. *This means that* $(true, true)$ *and* $(\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)$ *are relevant in* $\mathcal{S}$ *and* (α, α) *is irrelevant in* $\mathcal{S}$.

Recall the stimulus-response intuition given above. In other words, the pair (β, e') is *irrelevant* in a given input-output set if there exists another pair (α, e) in the set that represents the need for less stimulus from the environment, hence the condition $\alpha \sqsubseteq \beta$, to produce at least as much information, with the possible exception of information that β may entail but α does not. Hence $e' \sqsubseteq e \sqcup \beta$.

We now list two important properties of $\succeq$ that will be useful later on. The set $[\![\langle P, c\rangle]\!]$ is closed w.r.t. $\succeq$.

Proposition 5.2.9. *Let* $(\alpha, e) \in [\![\langle P, c\rangle]\!]$. *If* $(\alpha, e) \succeq (\beta, e')$ *then* $(\beta, e') \in [\![\langle P, c\rangle]\!]$.

Proof. By definition $\langle P, c \sqcup \alpha\rangle \Downarrow_e$ then by monotonicity $\langle P, c \sqcup \beta\rangle \Downarrow_{e \sqcup \beta}$ so $\langle P, c \sqcup \beta\rangle \Downarrow_{e'}$ since $e' \sqsubseteq (e \sqcup \beta)$, therefore $(\beta, e') \in [\![\langle P, c\rangle]\!]$. □

Moreover, the relation $\succeq$ is well-founded. More precisely,

Proposition 5.2.10. *There is no infinite* strictly *descending chain* $p_1 \prec p_2 \prec \ldots$.

Proof. Follows from the well-foundedness of $\sqsubseteq$ (Remark 2.4.3) □

5.2.2 A canonical representation of choice-free configurations

Clearly $[\![\langle P, c\rangle]\!]$ may be infinite due potential existence of infinitely many arbitrary stimuli (inputs). By using the labeled transition semantics (Table 2.4.2) we shall show that we do not need consider arbitrary inputs but only the minimal ones. Recall that in $\gamma \xrightarrow{\alpha} \gamma'$ the label α represents the minimal information needed to evolve from γ to γ'.

Definition 5.2.11 (Labeled Input-Output Set)**.** *The* Labeled Input-Output Set *of a configuration* $\langle P, c\rangle$, *denoted as* $\mathcal{M}(\langle P, c\rangle)$, *is inductively defined as follows:*

$$\{(true, c)\} \cup \bigcup_{\langle P,c\rangle \xrightarrow{\alpha} \langle P',c'\rangle} (\{(\alpha, c')\} \cup (\alpha \otimes \mathcal{M}(\langle P', c'\rangle)))$$

where $\otimes : Con_0 \times 2^{Con_0 \times Con_0} \longrightarrow 2^{Con_0 \times Con_0}$ *is defined as:*

$$\alpha \otimes \mathcal{S} \stackrel{\text{def}}{=} \{(\alpha \sqcup \beta, e) \mid (\beta, e) \in \mathcal{S})\}$$

Let us illustrate this definition with an example.

Example 5.2.12. *Let* $\gamma = \langle \mathbf{ask}\ (\alpha) \rightarrow (\mathbf{ask}\ (\beta) \rightarrow \mathbf{tell}(c)), true\rangle$ *and* $\gamma' = \langle \mathbf{ask}\ (\alpha \sqcup \beta) \rightarrow \mathbf{tell}(c), true\rangle$ *where* $\alpha, \beta, c \in Con_0$ *are not related to each other. Let us assume that* α *and* β *are the minimal elements that allow* γ *and* γ' *to proceed. Their corresponding labeled input-output sets are as follows:*

$$\mathcal{M}(\gamma) = \{(true, true), (\alpha, \alpha), (\alpha \sqcup \beta, \alpha \sqcup \beta), (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)\}$$

$$\mathcal{M}(\gamma') = \{(true, true), (\alpha \sqcup \beta, \alpha \sqcup \beta), (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)\}$$

Nevertheless, labeled-based input-output sets do not give us a fully-abstract representation of the input-output sets because of the existence of irrelevant pairs. By excluding these pairs we obtain a compact and fully-abstract representation of input-output sets.

Definition 5.2.13 (Compact Input-Output Set)**.** *The* Compact *Input-Output Set of a configuration* $\langle P, c\rangle$ *is defined as follows:*

$$\mathcal{M}^C(\langle P, c\rangle) \stackrel{\text{def}}{=} \{(\alpha, e) \mid (\alpha, e) \in \mathcal{M}(\langle P, c\rangle) \text{ and } (\alpha, e) \text{ is relevant in } \mathcal{M}(\langle P, c\rangle)\}$$

Let us give an example.

Example 5.2.14. *Let* γ *and* γ' *as in Example 5.2.12. Using the same reasoning as in Example 5.2.8 one can check that:*

$$\mathcal{M}^C(\gamma) = \mathcal{M}^C(\gamma') = \{(true, true), (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)\}$$

We shall now show the full-abstraction of the compact input-output sets. We need the following lemmata. First, compact sets are closed under weak transitions ($\Longrightarrow$). More precisely:

Proposition 5.2.15. *If* $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c'\rangle$ *then* $(\alpha, c') \in \mathcal{M}(\langle P, c\rangle)$.

Proof. By induction on the depth of the inference of $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c'\rangle$.

- Using Rule R-Tau we have $\langle P, c\rangle \Longrightarrow \langle P, c\rangle$ and $(true, c) \in \mathcal{M}(\langle P, c\rangle)$ by definition.
- Using Rule R-Label we have $\langle P, c\rangle \stackrel{\alpha}{\longrightarrow} \langle P', c'\rangle$ and $(\alpha, c') \in \mathcal{M}(\langle P, c\rangle)$ by definition.
- Using Rule R-Add we have $\langle P, c\rangle \stackrel{\alpha''}{\Longrightarrow} \langle P'', c''\rangle \stackrel{\alpha'}{\Longrightarrow} \langle P', c'\rangle$ where $\alpha' \sqcup \alpha'' = \alpha$. Then by induction hypothesis $(\alpha'', c'') \in \mathcal{M}(\langle P, c\rangle)$ and $(\alpha', c') \in \mathcal{M}(\langle P'', c''\rangle)$, hence by definition of $\mathcal{M}(\langle P, c\rangle)$ we have $(\alpha' \sqcup \alpha'', c') \in \mathcal{M}(\langle P, c\rangle)$ so $(\alpha, c') \in \mathcal{M}(\langle P, c\rangle)$.

□

The following proposition states that whenever a pair (α, e) is in $\mathcal{M}(\langle P, c\rangle)$, it means that e can be reached from $\langle P, c \sqcup \alpha\rangle$ without aid of the environment.

Proposition 5.2.16. *If* $(\alpha, e) \in \mathcal{M}(\langle P, c\rangle)$ *then* $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', e\rangle$

Proof. By definition of $\mathcal{M}(\langle P, c\rangle)$, since $(\alpha, e) \in \mathcal{M}(\langle P, c\rangle)$ then there exist $\alpha_1, \ldots, \alpha_n$ such that $\alpha = \bigsqcup_{i=1}^{n} \alpha_n$ and $\langle P, c\rangle \stackrel{\alpha_1}{\longrightarrow} \ldots \stackrel{\alpha_n}{\longrightarrow} \langle P', e\rangle$. Hence by soundness on each transition $\langle P, c \sqcup \bigsqcup_{i=1}^{n} \alpha_n\rangle = \langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', e\rangle$. □

We can now prove our main result, given two configurations $\langle P, c\rangle$ and $\langle Q, d\rangle$, they are observationally equivalent if and only if their compact input-output sets are identical. We split the proof in the following two lemmata.

Lemma 5.2.17. *If* $\mathcal{M}^C(\langle P, c\rangle) = \mathcal{M}^C(\langle Q, d\rangle)$ *then* $[\![\langle P, c\rangle]\!] = [\![\langle Q, d\rangle]\!]$

Proof. Let us assume that $(\alpha, \beta) \in [\![\langle P, c\rangle]\!]$ then by definition $\langle P, c \sqcup \alpha\rangle \Downarrow_\beta$ hence there exists P' and β' such that $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', \beta'\rangle$ and $\beta \sqsubseteq \beta'$. By Lemma 4.2.2 we have $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', \beta'\rangle$, then by completeness of $\Longrightarrow$ (Lemma 4.2.16) there exist α', b s.t. $\langle P, c\rangle \stackrel{\alpha'}{\Longrightarrow} \langle P', c'\rangle$ where $\alpha' \sqcup b = \alpha$ and $c' \sqcup b = \beta'$ **(1)**. Now by Proposition 5.2.15 we know $(\alpha', c') \in \mathcal{M}(\langle P, c\rangle)$, then since $\succeq$ is well-founded (Proposition 5.2.10) there is (α'', c'') that is relevant in $\mathcal{M}(\langle P, c\rangle)$ (then it belongs to $\mathcal{M}^C(\langle P, c\rangle)$) such that $(\alpha'', c'') \succeq (\alpha', c')$, namely $\alpha'' \sqsubseteq \alpha'$ (or equivalently $\exists x.(\alpha'' \sqcup x) = \alpha'$ **(2)**) and $c' \sqsubseteq (c'' \sqcup \alpha')$. Given that

$(\alpha'', c'') \in \mathcal{M}^C(\langle P, c\rangle)$ then by hypothesis $(\alpha'', c'') \in \mathcal{M}^C(\langle Q, d\rangle)$, this means also that $(\alpha'', c'') \in \mathcal{M}(\langle Q, d\rangle)$ and by Proposition 5.2.16 we know that $\langle Q, d \sqcup \alpha''\rangle \longrightarrow^* \langle Q', c''\rangle$. By monotonicity we have the following transition $\langle Q, d \sqcup \alpha'' \sqcup x \sqcup b\rangle \longrightarrow^* \langle Q', c'' \sqcup x \sqcup b\rangle$, now notice that from **(1)** and **(2)** we have $(d \sqcup \alpha'' \sqcup x \sqcup b) = (d \sqcup \alpha' \sqcup b) = (d \sqcup \alpha)$ then $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', c'' \sqcup x \sqcup b\rangle$. Finally, we have to prove that $\beta \sqsubseteq (c'' \sqcup x \sqcup b)$ to conclude that $(\alpha, \beta) \in [\![\langle Q, d\rangle]\!]$, for that purpose, recall that $\beta \sqsubseteq \beta' = (c' \sqcup b) \sqsubseteq (c'' \sqcup \alpha' \sqcup b)$ and since $(c'' \sqcup \alpha'' \sqcup x \sqcup b) = (c'' \sqcup x \sqcup b)$ then $\beta \sqsubseteq (c'' \sqcup x \sqcup b)$ and so $(\alpha, \beta) \in [\![\langle Q, d\rangle]\!]$. □

Lemma 5.2.18. *If* $[\![\langle P, c\rangle]\!] = [\![\langle Q, d\rangle]\!]$ *then* $\mathcal{M}^C(\langle P, c\rangle) = \mathcal{M}^C(\langle Q, d\rangle)$

Proof. Assume that $(\alpha, \beta) \in \mathcal{M}^C(\langle P, c\rangle)$ our goal is to prove that $(\alpha, \beta) \in \mathcal{M}^C(\langle Q, d\rangle)$. By definition (α, β) is relevant in $\mathcal{M}(\langle P, c\rangle)$, moreover, by Proposition 5.2.16 we have $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', \beta\rangle$ then by definition $(\alpha, \beta) \in [\![\langle P, c\rangle]\!]$ and by hypothesis $(\alpha, \beta) \in [\![\langle Q, d\rangle]\!]$. This means that $\langle Q, d \sqcup \alpha\rangle \Downarrow_\beta$ then there exists Q', d' s.t. $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\beta \sqsubseteq d'$. By Lemma 4.2.2 we have $\langle Q, d \sqcup \alpha\rangle \Longrightarrow \langle Q', d'\rangle$, now by completeness of $\Longrightarrow$ (Lemma 4.2.16) there exist α', b s.t. $\langle Q, d\rangle \overset{\alpha'}{\Longrightarrow} \langle Q', d''\rangle$ where $(\alpha' \sqcup b) = \alpha$ and $(d'' \sqcup b) = d'$. Now let us assume by means of contradiction that $\alpha' \neq \alpha$. By soundness of $\Longrightarrow$ (Lemma 4.2.15) we have $\langle Q, d \sqcup \alpha'\rangle \Longrightarrow \langle Q', d''\rangle$ then by Lemma 4.2.2 we get $\langle Q, d \sqcup \alpha'\rangle \longrightarrow^* \langle Q', d''\rangle$ hence $(\alpha', d'') \in [\![\langle Q, d\rangle]\!]$. By hypothesis then $(\alpha', d'') \in [\![\langle P, c\rangle]\!]$, now this means that $\langle P, c \sqcup \alpha'\rangle \longrightarrow^* \langle P'', e\rangle$ where $d'' \sqsubseteq e$ (equivalently $\exists z.(d'' \sqcup z) = e$). By Lemma 4.2.2 we get that $\langle P, c \sqcup \alpha'\rangle \Longrightarrow \langle P'', e\rangle$ and by completeness there exist x, b' s.t. $\langle P, c\rangle \overset{x}{\Longrightarrow} \langle P'', c'\rangle$ where $(x \sqcup b') = \alpha'$ and $c' \sqcup b' = e$. Using Lemma 5.2.15 we have that $(x, c') \in \mathcal{M}(\langle P, c\rangle)$, now we will prove that $(x, c') \succeq (\alpha, \beta)$, namely $x \sqsubseteq \alpha$ and $\beta \sqsubseteq (\alpha \sqcup c')$. Recall that $x \sqsubseteq \alpha' \sqsubseteq \alpha$, now for the latter condition $(\alpha \sqcup c') = (\alpha' \sqcup b \sqcup c') = (x \sqcup b' \sqcup b \sqcup c') = (x \sqcup b \sqcup e)$ then since $d'' \sqsubseteq e$ we can check that $\beta \sqsubseteq d' \sqsubseteq (d' \sqcup x) = (d'' \sqcup b \sqcup x) \sqsubseteq (e \sqcup b \sqcup x) = (\alpha \sqcup c')$. Thus, this would mean that (α, β) is irrelevant in $\mathcal{M}(\langle P, c\rangle)$, a contradiction, therefore $\alpha' = \alpha$ and by consequence $d'' = d'$. Therefore, we know that $\langle Q, d\rangle \overset{\alpha}{\Longrightarrow} \langle Q', d'\rangle$, now let us assume by contradiction that $d' \neq \beta$ (i.e. $\beta \sqsubset d'$). By soundness and Lemma 4.2.2 we have that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$, this means that $(\alpha, d') \in [\![\langle Q, d\rangle]\!]$. By hypothesis then $(\alpha, d') \in [\![\langle P, c\rangle]\!]$ so there exist P_1, c_1 s.t.

$\langle P, c \sqcup \alpha \rangle \longrightarrow^* \langle P_1, c_1 \rangle$ and $d' \sqsubseteq c_1$. By Lemma 4.2.2 then $\langle P, c \sqcup \alpha \rangle \Longrightarrow \langle P_1, c_1 \rangle$, now by completeness, there exist y, b'' s.t. $\langle P, c \rangle \stackrel{y}{\Longrightarrow} \langle P_1, c_1' \rangle$ where $y \sqcup b'' = \alpha$ and $c_1' \sqcup b'' = c_1$. Using Lemma 5.2.15 we get that $(y, c_1') \in \mathcal{M}(\langle P, c \rangle)$. Now let us prove that $(y, c_1') \succeq (\alpha, \beta)$, namely $y \sqsubseteq \alpha$ and $\beta \sqsubseteq (\alpha \sqcup c_1')$. The first condition follows from $y \sqcup b'' = \alpha$ and for the latter condition we proceed as follows $\beta \sqsubset d' \sqsubseteq c_1 \sqsubseteq (c_1 \sqcup y) = (c_1' \sqcup b'' \sqcup y) = (c_1' \sqcup \alpha)$. Again, this would mean that (α, β) is irrelevant in $\mathcal{M}(\langle P, c \rangle)$, a contradiction, therefore $d' = \beta$. Hence, we know that $\langle Q, d \rangle \stackrel{\alpha}{\Longrightarrow} \langle Q', \beta \rangle$ then by Proposition 5.2.15 $(\alpha, \beta) \in \mathcal{M}(\langle Q, d \rangle)$. Finally, let us assume by contradiction that (α, β) is irrelevant in $\mathcal{M}(\langle Q, d \rangle)$. Then there exists $(\alpha_1, \beta_1) \in \mathcal{M}(\langle Q, d \rangle)$ such that $(\alpha_1, \beta_1) \succeq (\alpha, \beta)$, namely $\alpha_1 \sqsubseteq \alpha$ (equivalently $\exists z'. \alpha_1 \sqcup z' = \alpha$) and $\beta \sqsubseteq \alpha \sqcup \beta_1$. By Proposition 5.2.16 we have that $\langle Q, d \sqcup \alpha_1 \rangle \longrightarrow^* \langle Q_1, \beta_1 \rangle$, then $(\alpha_1, \beta_1) \in [\![\langle Q, d \rangle]\!]$ and by hypothesis $(\alpha_1, \beta_1) \in [\![\langle P, c \rangle]\!]$. This means that $\langle P, c \sqcup \alpha_1 \rangle \longrightarrow^* \langle P_2, c_2 \rangle$ where $\beta_1 \sqsubseteq c_2$, now by Lemma 4.2.2 we get $\langle P, c \sqcup \alpha_1 \rangle \Longrightarrow \langle P_2, c_2 \rangle$. By completeness of $\Longrightarrow$ there exist a, b_1 s.t. $\langle P, c \rangle \stackrel{a}{\Longrightarrow} \langle P_2, c_2' \rangle$ where $(a \sqcup b_1) = \alpha_1$ and $(c_2' \sqcup b_1) = c_2$. Hence, by Proposition 5.2.15 we know that $(a, c_2') \in \mathcal{M}(\langle P, c \rangle)$. Now let us prove that $(a, c_2') \succeq (\alpha, \beta)$ namely $a \sqsubseteq \alpha$ and $\beta \sqsubseteq (\alpha \sqcup c_2')$. First $a \sqsubseteq \alpha_1 \sqsubseteq \alpha$, for the latter condition we proceed as follows $(\alpha \sqcup c_2') = (\alpha_1 \sqcup z' \sqcup c_2') = (a \sqcup b_1 \sqcup z' \sqcup c_2') = (c_2 \sqcup z')$ and since $\beta_1 \sqsubseteq c_2$ and $\alpha \sqsubseteq c_2$ then $\beta \sqsubseteq (\alpha \sqcup \beta_1) \sqsubseteq (c_2 \sqcup z') = (\alpha \sqcup c_2')$. Once again, this would mean that (α, β) is irrelevant in $\mathcal{M}(\langle P, c \rangle)$, a contradiction. Finally, we can conclude that (α, β) is relevant in $\mathcal{M}(\langle Q, d \rangle)$ therefore $(\alpha, \beta) \in \mathcal{M}^C(\langle Q, d \rangle)$. □

Using the these lemmata above we conclude the following theorem.

Theorem 5.2.19. $[\![\langle P, c \rangle]\!] = [\![\langle Q, d \rangle]\!]$ *iff* $\mathcal{M}^C(\langle P, c \rangle) = \mathcal{M}^C(\langle Q, d \rangle)$

Proof. Using Lemma 5.2.17 and Lemma 5.2.18. □

By combining Theorem 5.2.3 and Theorem 5.2.19 we get a simple decision procedure for $\dot{\approx}_{sb}$ by reducing weak saturated equivalence between two given configuration to the set equivalence of the corresponding compact input-output representations. The complexity of this procedure is clearly determined by the complexity of constructions of the compact input-output sets.

Theorem 5.2.20. *Let γ and γ' be two CCP\+ configurations. Let $IS = \{\gamma, \gamma'\}$ and let $N = |\texttt{Config}_{\longrightarrow}(IS)|$. Then*

- $\mathcal{M}^C(\gamma) = \mathcal{M}^C(\gamma')$ *iff* $\gamma \dot{\approx}_{sb} \gamma'$.
- *Checking whether $\mathcal{M}^C(\gamma) = \mathcal{M}^C(\gamma')$ takes $O(N^2)$ time and uses $O(N)$ space.*

Proof. The first item follows from Follows from Theorem 5.2.3 and Theorem 5.2.19 and the second item is derived from the construction of $\mathcal{M}^C(\gamma)$ and $\mathcal{M}^C(\gamma')$. □

5.3 Improving the general partition refinement for CCP

In this section we show that in the general case of CCP systems, the strategy from Section 5.1.2 can be used for their CCP\+ components, thus producing a $IS^{\star}_{\rightsquigarrow}$ which may be significant smaller (although the worst case remains exponential).

Given a configuration γ the idea is to detect when an evolution of γ, i.e. a γ' s.t. $\gamma \stackrel{\alpha_1}{\Longrightarrow} \ldots \stackrel{\alpha_k}{\Longrightarrow} \gamma'$, is a CCP\+ configuration. This way we can avoid adding new configurations with Rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ whenever $\gamma' \in \mathit{Conf}_{\texttt{CCP}\backslash+}$, and redundancy can be then checked using Proposition 5.1.9.

$$(\mathsf{IS'} \Longrightarrow) \ \frac{\gamma \in IS}{\gamma \in IS^{\star}_{\rightsquigarrow}} \qquad (\mathsf{RS'} \Longrightarrow) \ \frac{\gamma \in IS^{\star}_{\rightsquigarrow} \quad \gamma \stackrel{\alpha}{\rightsquigarrow} \gamma'}{\gamma' \in IS^{\star}_{\rightsquigarrow}}$$

$$(\mathsf{opt\text{-}RD} \Longrightarrow) \ \frac{\begin{array}{c}\gamma \in IS^{\star}_{\rightsquigarrow} \quad \gamma \notin \mathit{Conf}_{\texttt{CCP}\backslash+} \quad t_1 = \gamma \stackrel{\alpha}{\rightsquigarrow} \langle P_1, c_1\rangle \\ t_2 = \gamma \stackrel{\beta}{\rightsquigarrow} \langle P_2, c_2\rangle \quad \alpha \sqsubset \beta \quad c_2 = c_1 \sqcup \beta\end{array}}{\langle P_1, c_2\rangle \in IS^{\star}_{\rightsquigarrow}}$$

Table 5.3.1: Rules for improved version of the partition refinement for CCP.

Definition 5.3.1 (Improved partition refinement for CCP)**.** *We define the procedure* `imp-weak-pr-ccp`$(IS, \rightsquigarrow)$ *by using the rules in Table 5.3.1 in Step 1 of* `weak-pr-ccp`$(IS, \rightsquigarrow)$ *from Algorithm 4.3.1.*

Using this algorithm we can decide $\dot{\approx}_{sb}$ in a more efficient manner, although, in the worst-case scenario, still with exponential time. This follows from Proposition 5.1.9 and Theorem 3.2.16.

Theorem 5.3.2. *Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of $\texttt{imp-weak-pr-ccp}(IS, \Longrightarrow)$ in Definition 5.3.1. If $IS^{\star}_{\Longrightarrow}$ is finite then the algorithm terminates and:*

- *$\gamma \; \mathcal{P} \; \gamma'$ iff $\gamma \dot{\approx}_{sb} \gamma'$.*

- *$\texttt{imp-weak-pr-ccp}(IS, \Longrightarrow)$ may take exponential time in the size of $\texttt{Config}_{\longrightarrow}(IS)$.*

It is clear that it is better to use $\texttt{imp-weak-pr-ccp}(IS, \Longrightarrow)$ instead of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$ since the new procedure avoids adding new states whenever they are not necessary to check redundancy w.r.t. $\dot{\approx}_{sb}$. Unfortunately, this improvement does not escape from the worst-case of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$. Nevertheless, this approach shows the applicability of the strategy developed in Section 5.1.2. Notice that this scenario arises with the use of the nondeterministic choice hence one may expect a practical impact on this improvement since most configurations are composed by several CCP\+ subconfigurations. We want to conclude this section by referencing Figure 5.3.1 which shows the main advantage of using this method.

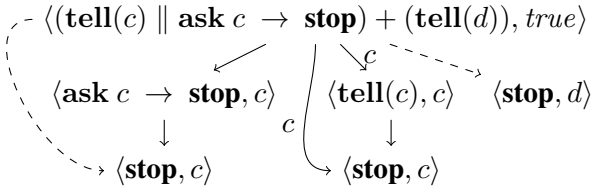

Figure 5.3.1: This example above reflects the advantages of using the method discussed in Section 5.3 since only the dashed transitions need to be considered for checking $\dot{\approx}_{sb}$. Transitions are computed according to $\Longrightarrow$ (Table 4.2.1) and loops are omitted.

5.4 Summary and Related Work

In this chapter we explored the use of the partition refinement algorithm for CCP from Chapters 3 and 4 for checking observational equivalence in the CCP\+ fragment. In Chapter 3 we gave a decision procedure for $\dot{\sim}_{sb}$ and in Chapter 4 we proved how the algorithm for $\dot{\sim}_{sb}$ can be used to compute $\dot{\approx}_{sb}$. In this chapter, we proved that this procedure takes exponential time and space (in the size of the set of reachable configurations) even for the restricted case of CCP\+. We then proposed two alternative methods for checking observational equivalence in CCP\+ by exploiting some of the intrinsic properties of this fragment, in particular confluence. We proved that both procedures take polynomial time (in the size of the set of reachable configurations), thus significantly improving the exponential-time approach from Chapters 3 and 4, which is, to the best of our knowledge the only algorithm for checking observational equivalence in CCP. Each of the two methods has its advantages over the other. On the one hand, the algorithm from Section 5.1 uses significantly more time and space than the one from Section 5.2, however it can be easily adapted for verifying observational equivalence for the full CCP as shown in Section 5.3. On the other hand, the procedure from Section 5.2 takes less time and uses only linear space nevertheless there is no "trivial" adaptation for the full language since it does not use the partition refinement approach. Finally, most of the related work was already discussed in the introduction, in Chapters 3 and 4.

Publications from this Chapter

The material of this chapter has been published in the following papers:

- [56] **L. Pino**, F. Bonchi, F. Valencia, *Efficient Computation of Program Equivalence for Confluent Concurrent Constraint Programming*, in : R. Peña, T. Schrijvers (Eds.), Proceedings of the 15th International Symposium on Principles and Practice of Declarative Programming (PPDP 2013), ACM, 2013, pp. 263-274.
  DOI: `http://dx.doi.org/10.1145/2505879.2505902`.

- [57] **L. Pino**, F. Bonchi, F. Valencia. *Efficient Computation of Program Equivalence for Confluent Concurrent Constraint Programming (Extended Abstract)*. (Informal) Proceedings of the 5th Young Researchers Workshop on Concurrency Theory (YR-CONCUR 2013).

- [55] **L. Pino**, F. Bonchi, F. Valencia. *Efficient Algorithms for Program Equivalence for Confluent Concurrent Constraint Programming*. To appear in the Journal of Science of Computer Programming, 2015.
DOI: `http://dx.doi.org/10.1016/j.scico.2014.12.003`.

Chapter 6

A Behavioral Congruence for CCP

In the previous chapters we developed efficient methods for computing $\dot{\sim}_{sb}$ (Definition 2.4.9) and $\dot{\approx}_{sb}$ (Definition 2.4.10). This was accomplished by defining several decision procedures which exploited various characteristics of CCP as well as the well-known partition refinement algorithm. Moreover, by utilizing certain properties of the choice-free fragment (CCP$\backslash$+), we were able to define polynomial-time algorithms for checking $\dot{\approx}_{sb}$ in finite CCP$\backslash$+. In this chapter the focus is shifted towards the congruence issues related to $\dot{\approx}_{sb}$

The goal of this chapter is then to tackle these issues by proposing a novel notion of bisimilarity which is a congruence for the full CCP. In Section 6.1 we show the relation between observational equivalence ($\sim_o$, Definition 2.4.19) and weak saturated barbed bisimilarity ($\dot{\approx}_{sb}$, Definition 2.4.10) for CCP with nondeterministic choice. We also prove that $\dot{\approx}_{sb}$ is not a congruence for the full CCP. Finally, in Section 6.2 we introduce our new notion $\approx_f$, and we prove that (i) $\approx_f$ coincides with $\dot{\approx}_{sb}$ in the choice-free fragment of CCP; (ii) $\approx_f$ is a congruence for CCP with summation; and (iii) $\approx_f$ coincides with the equivalence obtained after closing $\dot{\approx}_{sb}$ under any context.

6.1 Congruence issues

A typical question in the realm of process calculi, and concurrency in general, is whether a given process equivalence is a *congruence*. In other words, whether

the fact that P and Q are equivalent implies that they are still equivalent in any context. More precisely, an equivalence $\cong$ is a congruence if $P \bowtie Q$ implies $C[P] \bowtie C[Q]$ for every process context C[1].

The congruence issue is fundamental for algebraic as well as practical reasons; one may not be content with having $P \bowtie Q$ equivalent but $R \parallel P \not\bowtie R \parallel Q$. Nevertheless, some of the representative equivalences in concurrency are not congruences. For example, in CCS [41], trace equivalence and strong bisimilarity are congruences but weak bisimilarity is not because it is not preserved by summation contexts. So given a notion of equivalence one may wonder in what contexts the equivalence is preserved. For instance, the problem with weak bisimilarity can be avoided by using guarded-summation (see [43]).

We shall see that $\dot{\approx}_{sb}$ is a congruence for CCP\+. However, this is not the case for the full language of CCP. Moreover, unlike CCS, the problem arises even in the presence of guarded summation/choice. In fact, our counterexample reveals that the problem is intrinsic to CCP.

In section 2.4.5 we introduced the standard notion of observational equivalence ($\sim_o$) [63] for CCP and we saw that, in CCP\+, it coincides with $\dot{\approx}_{sb}$.

Nevertheless, this is not true for the full language of CCP. We can show this by using a counter-example reminiscent from the standard one for CCS. Namely, in CCS one can prove that, in general, $a + \tau.P$ is not weakly bisimilar to $a + P$. We can use a similar approach for CCP as follows.

Claim 6.1.1. *In finite CCP there are P, Q s.t. $P \sim_o Q$ but $P \not\dot{\approx}_{sb} Q$.*

Proof. Let $P = (\mathbf{ask}\ (b)\ \rightarrow\ \mathbf{tell}(c)) + (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{ask}\ (d)\ \rightarrow\ \mathbf{tell}(e))$ and $Q = (\mathbf{ask}\ (b)\ \rightarrow\ \mathbf{tell}(c)) + (\mathbf{ask}\ (d)\ \rightarrow\ \mathbf{tell}(e))$ for unrelated elements $b, c, d, e \in Con_0$. It is straightforward to see that $P \sim_o Q$ since the only relevant inputs are $true, b$ and d for which both processes produce $true, c$ and e respectively. Now let us show that $P \not\dot{\approx}_{sb} Q$. First notice that if we take the transition $\langle P, true\rangle \longrightarrow \langle \mathbf{ask}\ (d)\ \rightarrow\ \mathbf{tell}(e), true\rangle$ then $\langle Q, true\rangle$ can only stay still. Now if we take the transition $\langle Q, true\rangle \xrightarrow{b} \langle \mathbf{tell}(c), b\rangle$ then note that $\langle \mathbf{tell}(c), b\rangle \Downarrow_c$ but $\langle \mathbf{ask}\ (d)\ \rightarrow\ \mathbf{tell}(e), b\rangle \not\Downarrow_c$. Hence $P \not\dot{\approx}_{sb} Q$. □

[1] Recall that the expression $C[P]$ denotes the process that results from replacing in C, the hole $\bullet$ with P. For example $C = R \parallel \bullet$ then $C[P] = R \parallel P$.

However, the the ($\Leftarrow$) direction of the theorem does hold. We show this next.

Theorem 6.1.2. *Let P and Q be finite CCP processes. If $P \dot{\approx}_{sb} Q$ then $P \sim_o Q$.*

Proof. Let us assume by means of contradiction that $P \dot{\approx}_{sb} Q$ and there exists d s.t. $\mathcal{O}(P)(d) \neq \mathcal{O}(Q)(d)$. By definition, this means that there is an α s.t. $\alpha \in \mathcal{O}(P)(d)$ but $\alpha \notin \mathcal{O}(Q)(d)$. Hence $\langle P, d\rangle \longrightarrow^* \langle P', \alpha\rangle \not\longrightarrow$ however there is no Q' s.t. $\langle Q, d\rangle \longrightarrow^* \langle Q', \alpha\rangle \not\longrightarrow$. Therefore, for each computation $\langle Q, d\rangle \longrightarrow^* \langle Q', \beta\rangle \not\longrightarrow$ we have that either (i) $\beta \sqsubset \alpha$, (ii) $\alpha \sqsubset \beta$ or (iii) $\beta \not\sqsubseteq \alpha$. We shall prove that there is no saturated barbed bisimulation $\mathcal{R}$ containing the pair $\langle P', \alpha\rangle \mathcal{R} \langle Q', \beta\rangle$ for any of the three cases of β. First, consider the computations of type (i). Notice that since $\langle Q', \beta\rangle \not\longrightarrow$ and $\beta \sqsubset \alpha$ then $\langle Q', \beta\rangle \not\Downarrow_\alpha$ while $\langle P', \alpha\rangle \Downarrow_\alpha$. Now take the type (ii) and this time $\langle Q', \beta\rangle \Downarrow_\beta$ but $\langle P', \alpha\rangle \not\Downarrow_\beta$ since $\alpha \sqsubset \beta$. Similarly, in the type (iii) computations we have that $\langle Q', \beta\rangle \not\Downarrow_\alpha$ however $\langle P', \alpha\rangle \Downarrow_\alpha$. From these three cases we can conclude that $\langle P', \alpha\rangle$ cannot be related with $\langle Q', \beta\rangle$ in $\mathcal{R}$. Finally, in order to match $\langle P', \alpha\rangle \Downarrow_\alpha$ then the only case left would correspond to $\langle Q, d\rangle \longrightarrow^* \langle Q', \alpha\rangle \longrightarrow^* \langle Q'', \alpha'\rangle$ with $\alpha \sqsubset \alpha'$, but again $\langle Q', \alpha\rangle \Downarrow_{\alpha'}$ while $\langle P', \alpha\rangle \not\Downarrow_{\alpha'}$. Since $\langle Q, d\rangle$ is not able to match $\langle P, d\rangle \longrightarrow^* \langle P', \alpha\rangle$ then there cannot be weak saturated barbed bisimulation relating $\langle P, d\rangle$ and $\langle Q, d\rangle$, a contradiction to the hypothesis $P \dot{\approx}_{sb} Q$. □

On the other hand, weak bisimilarity is a congruence in a restricted sense: It is preserved by all the contexts from the choice-free fragment.

Theorem 6.1.3. *Let P and Q be CCP\+ processes and assume that $P \dot{\approx}_{sb} Q$. Then for every process context $C[\bullet]$ in CCP\+ we have $C[P] \dot{\approx}_{sb} C[Q]$.*

Proof. We will focus on the parallel context since the other cases are easily verified. We shall prove that $\mathcal{R} = \{(\langle P \parallel R, c\rangle, \langle Q \parallel R, d\rangle) \mid \langle P, c\rangle \dot{\approx} \langle Q, d\rangle\}$ is a weak bisimulation (Definition 2.4.29). The result then follows from Proposition 2.4.31.

(i) Assume that $\langle P \parallel R, c\rangle \downarrow_\alpha$ then, since $\langle P, c\rangle \dot{\approx} \langle Q, d\rangle$, by condition (i) we have that $\langle Q \parallel R, d\rangle \Downarrow_\alpha$.

(ii) Now suppose that $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P_1, c_1\rangle$ for some $\langle P_1, c_1\rangle$. By induction on (the depth) of the inference of $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P_1, c_1\rangle$ we have the following two cases:

- Using Rule LR2 (left) we have that $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P' \parallel R, c'\rangle$, hence $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ by a shorter inference. Using this transition and the hypothesis in $\mathcal{R}$ we can conclude that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ and $\langle P', c'\rangle \dot{\approx} \langle Q', d'\rangle$ **(1)**. Now using Rule LR2 we get that $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R, d'\rangle$ and, because of **(1)**, then $\langle P' \parallel R, c'\rangle \mathcal{R} \langle Q' \parallel R, d'\rangle$.
- Using Rule LR2 (right) we have that $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P \parallel R', c'\rangle$, hence $\langle R, c\rangle \xrightarrow{\alpha} \langle R', c'\rangle$ by a shorter inference, where $c' = c \sqcup \alpha \sqcup \beta$. Given that $\langle P, c\rangle \downarrow_c$ then by the hypothesis we know that $\langle Q, d\rangle \Downarrow_c$. This means that $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle \downarrow_c$, moreover by Lemma 5.1.5, Proposition 2.4.31 and the hypothesis we can conclude that $\langle Q, d\rangle \dot{\approx} \langle Q', d'\rangle \dot{\approx} \langle P, c\rangle$. Therefore $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R, d' \sqcup \alpha\rangle$ since $c \sqsubseteq d'$. Now from this transition notice that $\langle Q' \parallel R, d' \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R', d' \sqcup \alpha \sqcup \beta\rangle$ and it is the case that $\langle Q' \parallel R', d' \sqcup \alpha \sqcup \beta\rangle \mathcal{R} \langle P \parallel R', c \sqcup \alpha \sqcup \beta\rangle$ since $\langle Q', d'\rangle \dot{\approx} \langle P, c\rangle$.

□

Notice that this result implies that observational equivalence ($\sim_o$) is a congruence. Unfortunately, due to the nondeterministic choice, the theorem above does not hold for the full language of CCP, as shown next.

Theorem 6.1.4. *There exists P', Q, R in CCP such that:*

(a) $P' \dot{\approx}_{sb} Q$ *but*

(b) $P' \parallel R \not\dot{\approx}_{sb} Q \parallel R$.

Proof. To prove this claim let:

$$P = (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{tell}(c)) + (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{tell}(d))$$

$$P' = P \parallel \mathbf{tell}(e)$$

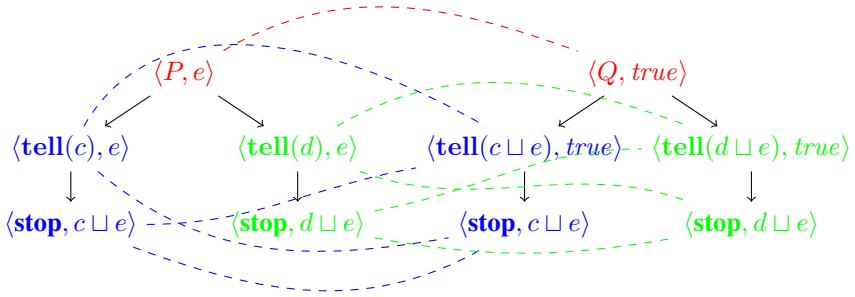

Figure 6.1.1: Let $P = (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{tell}(c)) + (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{tell}(d))$ and $Q = (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{tell}(c \sqcup e)) + (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{tell}(d \sqcup e))$. The linked configurations are weakly bisimilar.

$$Q = (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{tell}(c \sqcup e)) + (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{tell}(d \sqcup e))$$

with $c \not\sqsupseteq d, c \not\sqsupseteq e, d \not\sqsupseteq c, d \not\sqsupseteq e, e \not\sqsupseteq c, e \not\sqsupseteq d$.

For (a) we can show that $\langle P', true\rangle \dot{\approx}_{sb} \langle P, e\rangle \dot{\approx}_{sb} \langle Q, true\rangle$. The first equation is trivial. For the second we define a relation on configurations $\mathcal{R}$. The set of pairs in $\mathcal{R}$ are those linked in Figure 6.1.1. It can easily be verified that (the symmetric closure of) $\mathcal{R}$ is a weak bisimulation (see Definition 2.4.29). The point (a) then follows from Proposition 2.4.31.

For proving the part (b) of the above claim, we let $R = (\mathbf{ask}\ (e)\ \rightarrow\ \mathbf{tell}(\alpha)) + (\mathbf{ask}\ (e)\ \rightarrow\ \mathbf{tell}(\beta))$. We shall prove that no weak bisimulation can contain the pair $(\langle P \parallel R, e\rangle, \langle Q \parallel R, true\rangle)$. The results then follows from Proposition 2.4.31 and the fact that $\langle P' \parallel R, true\rangle \dot{\approx}_{sb} \langle P \parallel R, e\rangle$ which can be easily verified.

Consequently, let us assume that $\langle P \parallel R, e\rangle \longrightarrow \langle P \parallel \mathbf{tell}(\alpha), e\rangle$ by executing the left summand of R. By condition (ii) of weak bisimulation $\langle Q \parallel R, true\rangle$ must match the move. We have two cases:

- $\langle Q \parallel R, true\rangle$ does not make a transition. And now let us suppose that $\langle Q \parallel R, true\rangle \stackrel{e}{\longrightarrow} \langle Q \parallel \mathbf{tell}(\beta), true\rangle$. This means that $\langle P \parallel \mathbf{tell}(\alpha), e\rangle$ now has to match this transition. However $\langle Q \parallel \mathbf{tell}(\beta), true\rangle \longrightarrow \langle Q, \beta\rangle \Downarrow_\beta$ while $\langle P \parallel \mathbf{tell}(\alpha), e\rangle \not\Downarrow_\beta$. Thus we cannot satisfy condition (i) of weak bisimulation.

- $\langle Q \parallel R, true\rangle$ makes a transition. To match the move it should also execute the left summand of R. However, since e is not the store of $\langle Q \parallel R, true\rangle$, Q

must be executed first. and this means executing of one of summands in Q to be able to add e to the store. If the left summand of Q is executed, we get $\langle Q \parallel R, \mathit{true}\rangle \longrightarrow^* \langle \mathbf{tell}(\alpha), c \sqcup e\rangle$. In this case we could then take the move $\langle P \parallel \mathbf{tell}(\alpha), e\rangle \longrightarrow \langle \mathbf{tell}(d) \parallel \mathbf{tell}(\alpha), e\rangle$. But then $\langle \mathbf{tell}(\alpha), c \sqcup e\rangle \Downarrow_c$ and notice that $\langle \mathbf{tell}(d) \parallel \mathbf{tell}(\alpha), e\rangle \not\Downarrow_c$, thus we cannot satisfy condition (i) of weak bisimulation. The case where the right summand of Q is executed is symmetric.

□

6.2 Weak full bisimilarity

In the previous section we showed that $\dot{\approx}$ (and $\dot{\approx}_{sb}$) for the full CCP is not entirely satisfactory since it is not a congruence. By building on $\dot{\approx}$, in this section we propose a new equivalence which we call *(weak) full bisimilarity*, written $\approx_f$. This new equivalence does not quantify over infinitely many process contexts in its definition yet we will show that is a congruence. Furthermore, we will also prove that adequacy of $\approx_f$ by showing that it is the largest congruence included in $\dot{\approx}_{sb}$.

6.2.1 More than weak barbs

The key to figure out the element missing in the definition of $\dot{\approx}_{sb}$ (Definition 2.4.10) lies in Figure 6.1.1. If we look at the configurations in the figure we can see that while $\langle P, e\rangle$ is able to *produce* a barb e without choosing between c and d, $\langle Q, \mathit{true}\rangle$ is not. The definition of $\dot{\approx}_{sb}$ tries to capture this in the condition (i), namely by checking that $\langle P, e\rangle \Downarrow_e$ then requiring that $\langle Q, \mathit{true}\rangle \Downarrow_e$. However, this condition does not capture the fact that in order to produce e, $\langle Q, \mathit{true}\rangle$ may have to evolve into a configuration which can no longer produce some of the weak barbs $\langle Q, \mathit{true}\rangle$ can produce. [2]

Using this insight, we shall define a new notion of weak bisimilarity that changes condition (i) in $\dot{\approx}$ (Definition 2.4.29) in order to deal with the problem

[2] In the case of CCP\+ this is not a concern given that in this fragment weak barbs are always preserved during evolution.

present in Figure 6.1.1. More concretely, condition (i) requires that whenever $\langle P, c\rangle \downarrow_\alpha$ then $\langle Q, d\rangle \Downarrow_\alpha$, $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle \downarrow_\alpha$ without imposing any condition between $\langle P, c\rangle$ and $\langle Q', d'\rangle$. This makes it possible that $\langle P, c\rangle \downarrow_\beta$ and $\langle Q', d'\rangle$ does *not*: indeed, it might be the case that that $\langle Q, d\rangle \longrightarrow^* \langle Q'', d''\rangle \downarrow_\beta$ for some other branch $\langle Q'', d''\rangle$. Hence $\langle P, c\rangle$ and $\langle Q, d\rangle$ would pass condition (i) as in Figure 6.1.1.

Weak full bisimilarity deals with this problem by adding a condition between $\langle P, c\rangle$ and $\langle Q', d'\rangle$, namely $\langle Q, d\rangle \Downarrow_c$ has to hold by reaching a bisimilar configuration: $\langle P, c\rangle$ has to be weakly bisimilar $\langle Q', d'\rangle$.

Definition 6.2.1 (Weak Full Bisimilarity)**.** *A weak full bisimulation is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, c\rangle$ and $\gamma_2 = \langle Q, d\rangle$ implies that:*

(i) there exists $\gamma_2' = \langle Q', d'\rangle$ such that $\langle Q, d\rangle \longrightarrow^* \gamma_2'$ where $c \sqsubseteq d'$ and $(\gamma_1, \gamma_2') \in \mathcal{R}$,

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ then there exists $\gamma_2' = \langle Q', d'\rangle$ s.t. $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \gamma_2'$ where $c' \sqsubseteq d'$ and $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 and γ_2 are weak fully bisimilar ($\gamma_1 \approx_f \gamma_2$) if there exists a weak full bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$. We write $P \approx_f Q$ iff $\langle P, true\rangle \approx_f \langle Q, true\rangle$.

In the definition above, the first condition states that $\langle Q, d\rangle$ has to produce c by reaching a (weakly) bisimilar configuration. The second condition is the bisimulation game from $\dot{\approx}$ (Definition 2.4.29) plus a condition requiring the store c' to be matched too.

To better explain this notion consider again the counterexample to $\dot{\approx}$ from Figure 6.1.1.

Example 6.2.2. *Let $\langle P, e\rangle, \langle Q, true\rangle$ as in Figure 6.1.1. Let us build a relation $\mathcal{R}$ that is a weak full bisimulation where $(\langle P, e\rangle, \langle Q, true\rangle) \in \mathcal{R}$. By condition (i) in Definition 6.2.1 we need a $\gamma_2' = \langle Q', d'\rangle$ s.t. $\langle Q, d\rangle \longrightarrow^* \gamma_2'$ and $e \sqsubseteq d'$ and $(\gamma_1, \gamma_2') \in \mathcal{R}$. We have two options $Q' =$ **stop** and $d' = c \sqcup e$ or $d' = d \sqcup e$.*[3] *However, if we take $(\langle P, e\rangle, \langle$**stop**$, c \sqcup e\rangle) \in \mathcal{R}$ we have that $\langle P, e\rangle \Downarrow_d$ while*

[3]The cases for $Q' = \mathbf{tell}(c \sqcup e)$ or $Q' = \mathbf{tell}(d \sqcup e)$ with $d' =$ *true* are equivalent.

$\langle \mathbf{stop}, c \sqcup e\rangle \not\Downarrow_d$. *A similar argument works for* $\langle \mathbf{stop}, d \sqcup e\rangle$. *Therefore, no weak full bisimulation may contain* $(\langle P, e\rangle, \langle Q, \mathit{true}\rangle)$. *Hence* $\langle P, e\rangle \not\approx_f \langle Q, \mathit{true}\rangle$.

6.2.2 Full Bisimilarity is a Congruence

We shall now prove that full bisimilarity is a congruence w.r.t all possible contexts in CCP. Namely, whenever γ and γ' are in $\approx_f$ then they can be replaced for one another in any context.

Theorem 6.2.3. *Let* P *and* Q *be CCP processes and assume that* $P \approx_f Q$. *Then for every process context* $C[\bullet]$ *we have that* $C[P] \approx_f C[Q]$.

Proof. Here we consider the parallel case; the other cases are trivial or easier to verify. We shall prove that $\mathcal{R} = \{(\langle P \parallel R, c\rangle, \langle Q \parallel R, d\rangle) \mid \langle P, c\rangle \approx_f \langle Q, d\rangle\}$ is a weak full bisimulation as in Definition 6.2.1.

To prove (i), since $\langle P, c\rangle \approx_f \langle Q, d\rangle$ we have that $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle$ where $c \sqsubseteq d'$ and $\langle Q', d'\rangle \dot{\approx} \langle P, c\rangle$ **(1)**. Therefore by R2 we get $\langle Q \parallel R, d\rangle \longrightarrow^* \langle Q' \parallel R, d'\rangle$ and by **(1)** we can conclude that $(\langle Q' \parallel R, d'\rangle, \langle P \parallel R, c\rangle) \in \mathcal{R}$.

To prove (ii) let us assume that $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P_1, c_1\rangle$. We proceed by induction (on the depth) of the inference of $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle$.

Using LR2 (left), then $P_1 = (P' \parallel R)$ with $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ by a shorter inference. Since $\langle P, c\rangle \approx_f \langle Q, d\rangle$ then $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\langle P', c'\rangle \approx_f \langle Q', d'\rangle$ and $c' \sqsubseteq d'$ **(3)**. By R2 we have $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R, d'\rangle$ and from **(3)** we can conclude that $(\langle P' \parallel R, c\rangle, \langle Q' \parallel R, d'\rangle) \in \mathcal{R}$.

Using LR2 (right), then $P_1 = (P \parallel R')$ and $c' = (c \sqcup \alpha \sqcup e)$ with $\langle R, c\rangle \xrightarrow{\alpha} \langle R', c'\rangle$ by a shorter inference. From **(1)** we know that $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle$ where $c \sqsubseteq d'$ and $\langle Q', d'\rangle \dot{\approx} \langle P, c\rangle$. Hence $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R, d' \sqcup \alpha\rangle$. Now since $c \sqsubseteq d'$ then by monotonicity $\langle R, d' \sqcup \alpha\rangle \longrightarrow \langle R, d''\rangle$ where $d'' = d' \sqcup \alpha \sqcup e$. Therefore by R2 we get $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R', d''\rangle$ and from **(1)** and monotonicity $\langle P, c'\rangle = \langle P, c \sqcup \alpha \sqcup e\rangle \dot{\approx} \langle Q', d' \sqcup \alpha \sqcup e\rangle = \langle Q', d''\rangle$. Using this we can conclude that $(\langle P \parallel R', c'\rangle, \langle Q' \parallel R', d''\rangle) \in \mathcal{R}$. □

It is clear that $\approx_f$ is more distinguishing than $\dot{\approx}$ and the result above shows that this level of granularity is needed if we want a weak bisimilarity that is a congruence for the full CCP.

6.2.3 Relation with observational equivalence

In section 2.4.5 we described the relation between weak (saturated) bisimilarity ($\dot{\approx}_{sb}$, Definition 2.4.10) and the standard observational equivalence ($\sim_o$, Definition 2.4.19) for CCP. Concretely, we know that, in CCP\+, $\dot{\approx}_{sb}$ coincides with $\sim_o$, while for the full CCP $\dot{\approx}_{sb}$ implies $\sim_o$ but the converse does not hold. In this section we shall see the relation between weak full bisimilarity ($\approx_f$, Definition 6.2.1) and $\sim_o$. We shall prove that $\approx_f$ coincides with $\sim_o$ in CCP\+ by proving that $\approx_f$ corresponds to $\dot{\approx}_{sb}$ in the choice-free fragment of CCP. Furthermore, for the full language of CCP, we shall prove that $\approx_f$ implies $\sim_o$ again by showing that $\approx_f$ implies $\dot{\approx}_{sb}$ in CCP.

Let us start by showing that $\approx_f$ and $\dot{\approx}$ coincide in CCP\+. This theorem strongly relies on the confluent nature of CCP\+ (Proposition 2.4.6).

Theorem 6.2.4. *Let* $\gamma, \gamma' \in Conf_{\texttt{CCP}\backslash+}$, $\gamma \approx_f \gamma'$ *iff* $\gamma \dot{\approx} \gamma'$.

Proof. For the ($\Rightarrow$) direction consider $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \langle P, c\rangle \approx_f \langle Q, d\rangle\}$. We shall prove that $\mathcal{R}$ is a weak bisimulation (Definition 2.4.29).

(i) Assume that $\langle P, c\rangle \downarrow_\alpha$ then $\alpha \sqsubseteq c$. Since $\langle P, c\rangle \approx_f \langle Q, d\rangle$ then $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle$ and $c \sqsubseteq d'$. Hence $\langle Q, d\rangle \Downarrow_{d'}$, given that $\alpha \sqsubseteq c \sqsubseteq d'$ then $\langle Q, d\rangle \Downarrow_\alpha$.

(ii) Assume that $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ then since $\langle P, c\rangle \approx_f \langle Q, d\rangle$ we have that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\langle Q', d'\rangle \approx_f \langle P', c'\rangle$ and $c' \sqsubseteq d'$. Thus we can conclude that $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$.

For the ($\Leftarrow$) direction consider $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \langle P, c\rangle \dot{\approx} \langle Q, d\rangle\}$. We shall prove that $\mathcal{R}$ is a weak full bisimulation (Definition 6.2.1).

(i) By definition $\langle P, c\rangle \downarrow_c$. Now since $\langle P, c\rangle \dot{\approx} \langle Q, d\rangle$ we have that $\langle Q, d\rangle \Downarrow_c$, namely $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle \downarrow_c$. Hence by Lemma 5.1.5 (and Proposition 2.4.21) we get $\langle Q', d'\rangle \dot{\approx} \langle Q, d\rangle \dot{\approx} \langle P, c\rangle$. Finally $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle \mathcal{R} \langle P, c\rangle$ and $c \sqsubseteq d'$.

(ii) Assume that $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ then since $\langle P, c\rangle \dot{\approx} \langle Q, d\rangle$ we have that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\langle Q', d'\rangle \dot{\approx} \langle P', c'\rangle$. This means that $\langle Q', d'\rangle \Downarrow_{c'}$,

namely $\langle Q', d' \rangle \longrightarrow^* \langle Q'', d'' \rangle \downarrow_{c'}$. Now by Lemma 5.1.5 (and Proposition 2.4.21) we obtain $\langle Q'', d'' \rangle \dot{\approx} \langle Q', d' \rangle \dot{\approx} \langle P', c' \rangle$. Finally $\langle Q, d \sqcup \alpha \rangle \longrightarrow^* \langle Q'', d'' \rangle \mathcal{R} \langle P', c' \rangle$ and $c' \sqsubseteq d''$.

□

The corollary below follows from Proposition 2.4.21 and 2.4.31, and Theorem 6.2.4.

Corollary 6.2.5. *Let P and Q be CCP\+ processes. Then $P \approx_f Q$ iff $P \sim_o Q$.*

We shall now prove that $\approx_f$ implies $\sim_o$ for the full CCP. In order to do this we first prove that $\approx_f$ implies $\dot{\approx}_{sb}$.

Theorem 6.2.6. *If $\gamma \approx_f \gamma'$ then $\gamma \dot{\approx} \gamma'$.*

Proof. Following the same approach from the $(\Rightarrow)$ direction in Theorem 6.2.4. □

The corollary below follows from Theorem 6.1.2 and 6.2.6, and Proposition 2.4.31.

Corollary 6.2.7. *If $P \approx_f Q$ then $P \sim_o Q$.*

The above statement allows us to use the co-inductive techniques of full bisimulation to prove observational equivalence.

6.2.4 Behavioral congruence

Finally, we prove that $\approx_f$ is the largest congruence included in $\dot{\approx}$ by showing that it coincides with the congruence $\dot{\cong}$ defined next.

Definition 6.2.8 (Behavioral Congruence)**.** *We say that P is behaviorally congruent to Q, denoted $P \dot{\cong} Q$, iff for every process context $C[\bullet]$ we have $C[P] \dot{\approx} C[Q]$. We use $\langle P, e \rangle \dot{\cong} \langle Q, d \rangle$ to denote $(P \parallel \mathbf{tell}(e)) \dot{\cong} (Q \parallel \mathbf{tell}(d))$.*

We now state that $\approx_f$ coincides with $\dot{\cong}$.

Theorem 6.2.9. *$\langle P, e \rangle \approx_f \langle Q, d \rangle$ iff $\langle P, e \rangle \dot{\cong} \langle Q, d \rangle$.*

Proof. For the ($\Rightarrow$) direction assume that $\langle P, e\rangle \approx_f \langle Q, d\rangle$ then since $\langle P, e\rangle \approx_f \langle P \parallel \mathbf{tell}(e), \mathit{true}\rangle$ and $\langle Q, d\rangle \approx_f \langle Q \parallel \mathbf{tell}(d), \mathit{true}\rangle$ therefore we obtain that $(P \parallel \mathbf{tell}(e)) \approx_f (Q \parallel \mathbf{tell}(d))$. Hence using Theorem 6.2.3 we know that for every process context $C[\bullet]$ it is the case $C[P \parallel \mathbf{tell}(e)] \approx_f C[Q \parallel \mathbf{tell}(d)]$, thus from Theorem 6.2.6 we have $C[P \parallel \mathbf{tell}(e)] \dot{\approx} C[Q \parallel \mathbf{tell}(d)]$ hence $\langle P, e\rangle \dot{\cong} \langle Q, d\rangle$.

For the ($\Leftarrow$), by contradiction $\langle P, e\rangle \dot{\cong} \langle Q, d\rangle$ and $\langle P, e\rangle \not\approx_f \langle Q, d\rangle$. Hence we have two cases from conditions (i) and (ii) of $\approx_f$ (Definition 6.2.1). For the case (i), since $\langle P, e\rangle \not\approx_f \langle Q, d\rangle$, then there is no $\langle Q', d'\rangle$ such that:

$$\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle \text{ where } \langle P, e\rangle \approx_f \langle Q', d'\rangle \text{ and } e \sqsubseteq d' \textbf{ (1)}$$

Now let $C[\bullet] = (R \parallel \bullet)$ where:

$$R = ((\mathbf{ask}\ (e) \ \rightarrow\ \mathbf{tell}(\alpha)) + (\mathbf{ask}\ (e) \ \rightarrow\ \mathbf{tell}(\beta))) \text{ and}$$

$$(\mathit{fv}(\alpha) \cup \mathit{fv}(\beta)) \cap (\mathit{fv}(P) \cup \mathit{fv}(Q) \cup \mathit{fv}(e) \cup \mathit{fv}(d)) = \emptyset \textbf{ (2)}$$

Now consider $C[P \parallel \mathbf{tell}(e)]$ and $C[Q \parallel \mathbf{tell}(d)]$ and let us prove that they are not in $\dot{\approx}$. For that purpose take:

$$\langle C[P \parallel \mathbf{tell}(e)], \mathit{true}\rangle \longrightarrow \langle C[P], e\rangle \longrightarrow \langle P \parallel \mathbf{tell}(\alpha), e\rangle \longrightarrow \langle P, e \sqcup \alpha\rangle$$

First notice that $\langle C[P \parallel \mathbf{tell}(e)], \mathit{true}\rangle \dot{\approx} \langle C[P], e\rangle$ and $\langle P \parallel \mathbf{tell}(\alpha), e\rangle \dot{\approx} \langle P, e \sqcup \alpha\rangle$. By **(1)** and Theorem 6.2.6 we know that there is no $\langle Q', d'\rangle$ s.t. $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\langle P, e\rangle \dot{\approx} \langle Q', d'\rangle$ and $e \sqsubseteq d'$. From this we can conclude that for all $\langle Q', d'\rangle$ we have $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\langle P, e\rangle \not\dot{\approx} \langle Q', d'\rangle$ and $e \sqsubseteq d'$ **(3)**.[4] Note that $\langle C[Q \parallel \mathbf{tell}(d)], \mathit{true}\rangle \dot{\approx} \langle C[Q], d\rangle = \langle Q \parallel R, d\rangle$, moreover $\langle Q \parallel R, d\rangle$ has to reach a $\langle Q'', d''\rangle$ such that $\langle Q'', d''\rangle \Downarrow_\alpha$ and $\langle Q'', d''\rangle \not\Downarrow_\beta$ since $\langle P, e \sqcup \alpha\rangle \downarrow_\alpha$ however $\langle P, e \sqcup \alpha\rangle \not\Downarrow_\beta$. Hence the only possible strategy is as follows:

$$\langle Q \parallel R, d\rangle \longrightarrow^* \langle Q' \parallel R, d'\rangle \longrightarrow^* \langle Q', d' \sqcup \alpha\rangle \text{ where } e \sqsubseteq d'$$

[4] Assuming $e \not\sqsubseteq d'$ leads trivially to conclude that $C[P \parallel \mathbf{tell}(e)]$ and $C[Q \parallel \mathbf{tell}(d)]$ are not in $\dot{\approx}$ since $\langle Q, d\rangle \not\Downarrow_e$, thus a contradiction to the hypothesis.

However, from **(2)** and **(3)** we know that $\langle P, e \sqcup \alpha\rangle \not\dot{\approx} \langle Q', d' \sqcup \alpha\rangle$ for all $\langle Q', d'\rangle$ where $e \sqsubseteq d'$. Therefore $C[P \parallel \mathbf{tell}(e)] \not\dot{\approx} C[Q \parallel \mathbf{tell}(d)]$, a contradiction to the hypothesis $\langle P, e\rangle \dot{\cong} \langle Q, d\rangle$. The case (ii) is similar, we assume that $\langle P, e\rangle \xrightarrow{a} \langle P', c'\rangle$ and the same reasoning applies. □

6.3 Summary and Related Work

In this chapter we showed that the weak saturated barbed bisimilarity ($\dot{\approx}_{sb}$) proposed in [6] is not a congruence for CCP. Nevertheless, we also showed that the upward closure, i.e. condition (iii), is enough to make $\dot{\approx}_{sb}$ a congruence in the choice-free fragment (CCP\+). We then proposed a new notion of bisimilarity, called weak full bisimilarity ($\approx_f$), and we proved that it is a congruence for the full CCP despite the fact that $\approx_f$ does not require any quantification over a (potentially) infinite number of contexts in its definition. Furthermore, we showed that $\approx_f$ implies the standard observational equivalence ($\sim_o$) for CCP from [63]. Finally we demonstrated that $\approx_f$ is not too restrictive by showing that it is the largest congruence included in $\dot{\approx}_{sb}$. See Table 6.3.1 for a summary of the contributions of this chapter. This is the first weak behavioral CCP congruence for CCP with nondeterministic choice that does not require implicit quantification over all contexts.

There has been other attempts for finding a good notion of bisimilarity for CCP such as [62] and [40]. In [62] the authors propose a CCP bisimilarity that requires processes to match the exact label in the bisimulation game, a condition which is standard in process calculi realm, however this notion is known to be too distinguishing for CCP as shown in [6]. As for [40], their notion of (strong) bisimilarity resembles to the saturated barbed bisimilarity from [6] and, although they do not give a notion of weak bisimilarity, the results in this chapter can be related directly.

Publications from this Chapter

The material of this chapter has been published in the following papers:

- [58] **L. Pino**, F. Bonchi, F. Valencia. *A Behavioral Congruence for Concurrent Constraint Programming with Non-deterministic Choice*, in: In G. Ciobanu and D. Méry (Eds.), Proceedings of the 11th International Colloquium on Theoretical Aspects of Computing (ICTAC 2014), volume 8687 of Lecture Notes in Computer Science, pages 351-368. Springer, 2014.
  DOI: `http://dx.doi.org/10.1007/978-3-319-10882-7_21`

- [55] **L. Pino**, F. Bonchi, F. Valencia. *Efficient Algorithms for Program Equivalence for Confluent Concurrent Constraint Programming*. To appear in the Journal of Science of Computer Programming, 2015.
  DOI: `http://dx.doi.org/10.1016/j.scico.2014.12.003`.

Language	Relation among equivalences	Congruence w.r.t.	
		$C[\bullet]$	$C[\bullet]\backslash+$
CCP\+	$\dot{\cong} = \approx_f = \dot{\approx} = \dot{\approx}_{sb} = \sim_o$	N/A	$\dot{\cong}, \approx_f, \dot{\approx}, \dot{\approx}_{sb}, \sim_o$
CCP	$\dot{\cong} = \approx_f \subseteq \dot{\approx} = \dot{\approx}_{sb} \subseteq \sim_o$	$\dot{\cong}, \approx_f$	$\dot{\cong}, \approx_f, \dot{\approx}, \dot{\approx}_{sb}$

Table 6.3.1: Summary of the contributions of Chapter 6. Recall that $\dot{\approx}_{sb}$ stands for the weak saturated barbed bisimilarity (Definition 2.4.10), $\sim_o$ is the standard observational equivalence (Definition 2.4.19), $\dot{\approx}$ represents weak bisimilarity (Definition 2.4.29), $\approx_f$ is the notion of weak full bisimilarity proposed in this paper (Definition 6.2.1) and $\dot{\cong}$ stands for the behavioral congruence (Definition 6.2.8). $C[\bullet]\backslash+$ stands for the contexts where the summation operator does not occur, while $C[\bullet]$ represents any possible context, hence the summation operator may occur in $C[\bullet]$. For this reason we put N/A (Not Applicable) in the row corresponding to CCP\+. Notice that the correspondence $\dot{\approx} = \dot{\approx}_{sb} = \sim_o$ comes from [6].

Chapter 7

Conclusions

We shall conclude this dissertation with an overall summary of its contents. The reader can find a more detailed recap, including related work, at the end of each chapter. We shall also describe possible future research directions.

7.1 Summary

In this dissertation we developed new reasoning techniques as well as efficient methods for the verification of CCP programs.

We have proposed in Chapter 3 an algorithm for computing strong saturated barbed bisimilarity ($\dot{\sim}_{sb}$, Definition 2.4.9) and we proved that it runs in exponential-time on the size of the LTS of the input. This is due to the combination of non-deterministic choice and the peculiarity of the saturated bisimilarity where labels are not matched exactly as in the standard definition of bisimilarity. We adapted the well-known partition refinement algorithm [34, 50] to oblige the attacker, in the bisimulation game, to use only irredundant transitions. These are transitions which require less interaction with the environment than the rest and are able to capture the behavior of the process. Unfortunately, to determine whether a transition is irredundant we need $\dot{\sim}_{sb}$ itself, therefore they have to be computed at the same time. For this task we may need to add some unreachable states, sometimes an exponential number of them. We could then exploit the procedure for $\dot{\sim}_{sb}$ in Chapter 3 to check weak saturated barbed bisimilarity ($\dot{\approx}_{sb}$, Definition 2.4.9). We

proved that in CCP the weak transition relation ($\Longrightarrow$, Table 4.2.1) must be defined as the reflexive and transitive closure of the labeled semantics ($\longrightarrow$, Table 2.4.2) as opposed to omitting just the silent transitions as is standard in the process calculi realm [2]. Using this procedure we could also check observational equivalence in CCP, since from [6] we know it coincides with $\dot{\approx}_{sb}$.

Nevertheless, both algorithms in Chapters 3 and 4 have exponential-time complexity. Hence, to improve the efficiency we restricted CCP to its fragment without nondeterministic choice (CCP\+). In Chapter 5 we proposed more efficient methods for computing program equivalence in CCP\+. We showed that the procedure in Chapter 4 for computing observational equivalence is exponential even for CCP\+. We then proposed two alternative polynomial-time (on the size of the LTS of the input) algorithms for verifying $\dot{\approx}_{sb}$ in CCP\+. The first algorithm is based on the idea that redundancy can be precomputed in this fragment, thus reducing significantly the complexity. In the second algorithm the intuition is that detecting if two programs have the same behavior is equivalent to check whether the two inputs have the same minimal finite representation of the set of weak barbs that they satisfy in every possible context. Both approaches are interesting since even if the second procedure has better time complexity, the first one can be used as a heuristic for the full CCP.

Finally, in Chapter 6, we addressed the congruence issues related to $\dot{\approx}_{sb}$. We showed that $\dot{\approx}_{sb}$ is a congruence for CCP\+ however this is not the case for the full CCP. We then proposed weak full bisimilarity ($\approx_f$, Definition 6.2.1) as an alternative way of characterizing the behavior of CCP programs. The idea is that $\approx_f$ is based on $\dot{\approx}_{sb}$ but requires a stronger condition on barbs where in order for a configuration to produce a weak barb it must do so by preserving its behavior. We proved that $\approx_f$ is a congruence for the full CCP and we also showed that it is the largest congruence included in $\dot{\approx}_{sb}$.

To the best of our knowledge, these are the first algorithms for verifying program equivalence in CCP. Furthermore, in the case of CCP\+, it is the first time observational equivalence can be checked efficiently. Finally, $\approx_f$ is also the first behavioral equivalence, which does not appeal to quantification over arbitrary process contexts in its definition, that is a congruence for the full CCP.

The author believes that the study of adequate equivalences for CCP and their

efficient computation contributes to a better understanding of CCP-like languages. Also that the study of efficient verification techniques is an essential for CCP and its applications. Finally, the author thinks that the novel notion of weak full bisimilarity contributes to the broad area of concurrency theory since it may provide a way of obtaining more elegant behavioral congruences for asynchronous process calculi.

7.2 Future Work

We now discuss some of the possible future research directions.

Concurrent Constraint Workbench (CCWB). Inspired in the seminal work on tools for verification in process calculi, such as the concurrency workbench [20] for CCS and the mobility workbench [67] for the π-calculus, as future work, we plan to implement the algorithms presented in the present thesis as the first step towards a workbench for the specification of concurrent constraint processes. It is worth noticing that the algorithms in Chapters 3 and 4 have been implemented in a early prototype as part of [5].

Aim for Efficiency. It remains as a future work to consider more efficient partition refinement algorithms [23] to see whether the algorithm from Section 5.1 can be further improved. The challenge would be to find a more efficient version of $\Longrightarrow$ that can still be used for deciding $\dot{\approx}_{sb}$ and so it can be adapted to the case of the full CCP.

Applications to CCP extensions We plan to investigate how the procedures here defined can be extended to different versions of CCP, specially to those where the choice operator is not present since program equivalence can be checked efficiently. Some examples include timed CCP (tcc) [61], universal temporal CCP (utcc) [49] and, more recently, epistemic and spatial CCP (eccp) [36].

Computing full bisimilarity. We also plan to adapt the algorithms from Chapters 4 and 5 to verify $\approx_f$. We conjecture that the decision procedures for $\dot{\approx}_{sb}$ can

be adapted to check $\approx_f$. The challenge would be to find an efficient way of determining whether the two configurations can produce the same weak barbs and remain in the bisimulation rather than just checking the barbs.

Full bisimilarity for asynchronous calculi. In this dissertation we obtained a notion of weak bisimilarity that is a congruence even if we do not consider a label for observing the tell actions. Since CCP is an asynchronous language, not observing the tell follows the philosophy of considering as labels the minimal information needed to proceed, namely a tell process does not need a stimulus from the environment to post its information in the store. Following the same reasoning, we plan to investigate whether it is possible to define a labeled semantics for the asynchronous π-calculus ($A\pi$) [43, 60] with a τ label for the output transitions, instead of a co-action, and we shall check if a notion of bisimilarity similar to ours would also be a congruence.

Bibliography

[1] S. Abramsky and A. Jung. Domain theory. In T. S. E. M. S. Abramsky, D. M. Gabbay, editor, *Handbook of Logic in Computer Science, vol. III*. Oxford University Press, 1994.

[2] L. Aceto, A. Ingolfsdottir, and J. Srba. The algorithmics of bisimilarity. In D. Sangiorgi and J. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*, pages 100–172. Cambridge University Press, 2011.

[3] R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous pi-calculus. In U. Montanari and V. Sassone, editors, *7th International Conference on Concurrency Theory (CONCUR 1996)*, volume 1119 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 1996.

[4] F. Arbab and J. J. M. M. Rutten. A coinductive calculus of component connectors. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *16th International Workshop on Recent Trends in Algebraic Development Techniques (WADT 2002)*, pages 34–55, 2002.

[5] A. Aristizábal. *Bisimulation Techniques and Algorithms for Concurrent Constraint Programming*. PhD thesis, École Polytechnique, 2012.

[6] A. Aristizábal, F. Bonchi, C. Palamidessi, L. Pino, and F. D. Valencia. Deriving labels and bisimilarity for concurrent constraint programming. In M. Hofmann, editor, *14th International Conference on Foundations of Software Science and Computational Structures (FOSSACS 2011)*, volume 6604 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 2011.

[7] A. Aristizábal, F. Bonchi, L. F. Pino, and F. Valencia. Reducing weak to strong bisimilarity in CCP. In M. Carbone, I. Lanese, A. Silva, and A. Sokolova, editors, *5th Interaction and Concurrency Experience (ICE 2012)*, volume 104 of *EPTCS*, pages 2–16, 2012.

[8] A. Aristizábal, F. Bonchi, F. D. Valencia, and L. F. Pino. Partition refinement for bisimilarity in CCP. In S. Ossowski and P. Lecca, editors, *27th ACM Symposium on Applied Computing (SAC 2012), Constraint Solving and Programming Track*, pages 88–93. ACM, 2012.

[9] P. Baldan, A. Bracciali, and R. Bruni. A semantic framework for open processes. *Theoretical Computer Science*, 389(3):446–483, 2007.

[10] M. Bartoletti and R. Zunino. A calculus of contracting processes. In *25th Annual IEEE Symposium on Logic in Computer Science (LICS 2010)*, pages 332–341. IEEE Computer Society, 2010.

[11] J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: Mobile processes, nominal data, and logic. In *24th Annual IEEE Symposium on Logic in Computer Science (LICS 2009)*, pages 39–48. IEEE Computer Society, 2009.

[12] F. Bonchi, F. Gadducci, and G. V. Monreale. Reactive systems, barbed semantics, and the mobile ambients. In L. de Alfaro, editor, *12th International Conference on Foundations of Software Science and Computational Structures (FOSSACS 2009)*, volume 5504 of *Lecture Notes in Computer Science*, pages 272–287. Springer, 2009.

[13] F. Bonchi, F. Gadducci, and G. V. Monreale. Towards a general theory of barbs, contexts and labels. In H. Yang, editor, *9th Asian Symposium on Programming Languages and Systems (APLAS 2011)*, volume 7078 of *Lecture Notes in Computer Science*, pages 289–304. Springer, 2011.

[14] F. Bonchi, B. König, and U. Montanari. Saturated semantics for reactive systems. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 69–80. IEEE Computer Society, 2006.

[15] F. Bonchi and U. Montanari. Minimization algorithm for symbolic bisimilarity. In G. Castagna, editor, *18th European Symposium on Programming (ESOP 2009)*, volume 5502 of *Lecture Notes in Computer Science*, pages 267–284. Springer, 2009.

[16] A. Bouali and R. de Simone. Symbolic bisimulation minimisation. In G. von Bochmann and D. K. Probst, editors, *4th International Workshop Computer Aided Verification (CAV 1992)*, volume 663 of *Lecture Notes in Computer Science*, pages 96–108. Springer, 1992.

[17] R. Bruni, F. Gadducci, U. Montanari, and P. Sobocinski. Deriving weak bisimulation congruences from reduction systems. In M. Abadi and L. de Alfaro, editors, *16th International Conference on Concurrency Theory (CONCUR 2005)*, volume 3653 of *Lecture Notes in Computer Science*, pages 293–307. Springer, 2005.

[18] R. Bruni, H. C. Melgratti, and U. Montanari. A connector algebra for p/t nets interactions. In J.-P. Katoen and B. König, editors, *22nd International Conference on Concurrency Theory (CONCUR 2011)*, volume 6901 of *Lecture Notes in Computer Science*, pages 312–326. Springer, 2011.

[19] M. G. Buscemi and U. Montanari. Open bisimulation for the concurrent constraint pi-calculus. In S. Drossopoulou, editor, *17th European Symposium on Programming Languages and Systems (ESOP 2008)*, volume 4960 of *Lecture Notes in Computer Science*, pages 254–268. Springer, 2008.

[20] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transactions Programming Languages Systems*, 15(1):36–72, 1993.

[21] F. S. de Boer, A. D. Pierro, and C. Palamidessi. Nondeterminism and infinite computations in constraint programming. *Theoretical Computer Science*, 151(1):37–78, 1995.

[22] R. De Nicola. Behavioral equivalences. In *Encyclopedia of Parallel Computing*, pages 120–127. Springer, 2011.

[23] A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *Theorerical Computer Science*, 311(1-3):221–256, 2004.

[24] M. Falaschi, M. Gabbrielli, K. Marriott, and C. Palamidessi. Confluence in concurrent constraint programming. *Theoretical Computer Science*, 183(2):281–315, 1997.

[25] J.-C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13(1):219–236, 1989.

[26] J.-C. Fernandez and L. Mounier. Verifying bisimulations "On the Fly". In J. Quemada, J. A. Mañas, and E. Vázquez, editors, *3rd International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE 1990)*, pages 95–110. North-Holland, 1990.

[27] G. Ferrari, S. Gnesi, U. Montanari, M. Pistore, and G. Ristori. Verifying mobile processes in the HAL environment. In A. J. Hu and M. Y. Vardi, editors, *10th International Conference on Computer Aided Verification (CAV 1998)*, volume 1427 of *Lecture Notes in Computer Science*, pages 511–515. Springer, 1998.

[28] W. Fokkink, J. Pang, and A. Wijs. Is timed branching bisimilarity a congruence indeed? *Fundamenta Informaticae*, 87(3-4):287–311, 2008.

[29] F. Gadducci and U. Montanari. The tile model. In G. D. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 133–166. The MIT Press, 2000.

[30] H. Garavel. Reflections on the future of concurrency theory in general and process calculi in particular. *Electronic Notes in Theoretical Computer Science*, 209:149–164, 2008.

[31] J. Gutierrez, J. A. Pérez, C. Rueda, and F. D. Valencia. Timed concurrent constraint programming for analysing biological systems. *Electronic Notes Theorerical Computer Science*, 171(2):117–137, 2007.

[32] R. Haemmerlé. Observational equivalences for linear logic concurrent constraint languages. *TPLP*, 11(4-5):469–485, 2011.

[33] O. H. Jensen. *Mobile Processes in Bigraphs*. PhD thesis, University of Cambridge, 2006.

[34] P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In R. L. Probert, N. A. Lynch, and N. Santoro, editors, *2nd Annual ACM Symposium on Principles of Distributed Computing (PODC 1983)*, pages 228–240. ACM, 1983.

[35] S. Knight. *The Epistemic View of Concurrency Theory*. PhD thesis, École Polytechnique, 2013.

[36] S. Knight, C. Palamidessi, P. Panangaden, and F. D. Valencia. Spatial and epistemic modalities in constraint-based process calculi. In M. Koutny and I. Ulidowski, editors, *23rd International Conference on Concurrency Theory (CONCUR 2012)*, volume 7454 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2012.

[37] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the expressiveness and decidability of higher-order process calculi. *Information and Computation*, 209(2):198–226, 2011.

[38] J. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In C. Palamidessi, editor, *11th International Conference on Concurrency Theory (CONCUR 2005)*, volume 1877 of *Lecture Notes in Computer Science*, pages 243–258. Springer, 2000.

[39] J. J. Leifer and R. Milner. Transition systems, link graphs and petri nets. *Mathematical Structures in Computer Science*, 16(6):989–1047, 2006.

[40] N. P. Mendler, P. Panangaden, P. J. Scott, and R. A. G. Seely. A logical view of concurrent constraint programming. *Nordic Journal of Computing*, 2(2):181–220, 1995.

[41] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[42] R. Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.

[43] R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.

[44] R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *19th International Colloquium on Automata, Languages and Programming (ICALP 1992)*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer, 1992.

[45] J. Monk, L. Henkin, and A. Tarski. *Cylindric Algebras (Part I)*. North-Holland, 1971.

[46] U. Montanari and V. Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae*, 16(1):171–199, 1992.

[47] C. Olarte, C. Rueda, and F. D. Valencia. Models and emerging trends of concurrent constraint programming. *Constraints*, 18(4):535–578, 2013.

[48] C. Olarte and F. D. Valencia. The expressivity of universal timed CCP: undecidability of monadic FLTL and closure operators for security. In S. Antoy and E. Albert, editors, *10th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP 2008)*, pages 8–19. ACM, 2008.

[49] C. Olarte and F. D. Valencia. Universal concurrent constraint programing: symbolic semantics and applications to security. In R. L. Wainwright and H. Haddad, editors, *23rd Annual ACM Symposium on Applied Computing (SAC 2008)*, pages 145–150. ACM, 2008.

[50] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[51] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[52] C. Palamidessi, V. A. Saraswat, F. D. Valencia, and B. Victor. On the expressiveness of linearity vs persistence in the asychronous pi-calculus. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 59–68. IEEE Computer Society, 2006.

[53] D. M. R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *5th GI-Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[54] L. F. Pino, A. Aristizabal, F. Bonchi, and F. Valencia. Weak CCP bisimilarity with strong procedures. *Science of Computer Programming*, 100(0):84–104, 2015.

[55] L. F. Pino, F. Bonchi, and F. Valencia. Efficient algorithms for program equivalence for confluent concurrent constraint programming. *Science of Computer Programming*, 2015. http://dx.doi.org/10.1016/j.scico.2014.12.003.

[56] L. F. Pino, F. Bonchi, and F. D. Valencia. Efficient computation of program equivalence for confluent concurrent constraint programming. In R. Peña and T. Schrijvers, editors, *15th International Symposium on Principles and Practice of Declarative Programming (PPDP 2013)*, pages 263–274. ACM, 2013.

[57] L. F. Pino, F. Bonchi, and F. D. Valencia. Efficient computation of program equivalence for confluent concurrent constraint programming (extended abstract). Informal proceedings of the 5th Young Researchers Workshop on Concurrency Theory (YR-CONCUR 2013), 2013.

[58] L. F. Pino, F. Bonchi, and F. D. Valencia. A behavioral congruence for concurrent constraint programming with non-deterministic choice. In G. Ciobanu and D. Méry, editors, *11th International Colloquium on Theoretical Aspects of Computing (ICTAC 2014)*, volume 8687 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2014.

[59] C. Rueda and F. D. Valencia. On validity in modelization of musical problems by CCP. *Soft Computing*, 8(9):641–648, 2004.

[60] D. Sangiorgi and D. Walker. *The π-Calculus - a theory of mobile processes*. Cambridge University Press, 2001.

[61] V. A. Saraswat, R. Jagadeesan, and V. Gupta. Foundations of timed concurrent constraint programming. In *9th Annual Symposium on Logic in Computer Science (LICS 1994)*, pages 71–80. IEEE Computer Society, 1994.

[62] V. A. Saraswat and M. C. Rinard. Concurrent constraint programming. In F. E. Allen, editor, *17th Annual ACM Symposium on Principles of Programming Languages (POPL 1990)*, pages 232–245. ACM Press, 1990.

[63] V. A. Saraswat, M. C. Rinard, and P. Panangaden. Semantic foundations of concurrent constraint programming. In D. S. Wise, editor, *18th Annual ACM Symposium on Principles of Programming Languages (POPL 1991)*, pages 333–352. ACM Press, 1991.

[64] P. Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*, 274(1-2):183–230, 2002.

[65] P. Sobocinski. Representations of petri net interactions. In P. Gastin and F. Laroussinie, editors, *21th International Conference on Concurrency Theory (CONCUR 2010)*, volume 6269 of *Lecture Notes in Computer Science*, pages 554–568. Springer, 2010.

[66] P. Sobocinski. Relational presheaves as labelled transition systems. In D. Pattinson and L. Schröder, editors, *11th International Workshop on Coalgebraic Methods in Computer Science (CMCS 2012)*, volume 7399 of *Lecture Notes in Computer Science*, pages 40–50. Springer, 2012.

[67] B. Victor and F. Moller. The mobility workbench - a tool for the π-calculus. In D. L. Dill, editor, *6th International Conference on Computer Aided Verification (CAV 1994)*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer, 1994.

[68] B. Victor and J. Parrow. Concurrent constraints in the fusion calculus. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *25th International Colloquium on Automata, Languages and Programming (ICALP 1998)*, volume 1443 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1998.

Index