université PARIS-SACLAY

école normale supérieure paris–saclay

NNT : 2016SACLN058

Thèse de doctorat de l'Université Paris-Saclay préparée à l' Ecole Normale Supérieure Paris-Saclay

Ecole doctorale n°580
Sciences et Technologies de l'Information et de la Communication
Spécialité de doctorat : Informatique

par

Mme. Marie Van Den Bogaard

Motifs de Flot d'Information dans les Jeux à Information Imparfaite

Thèse présentée et soutenue à Cachan, le 29 novembre 2016.

Composition du Jury :

Mme	Véronique Bruyère	Professeur Université de Mons	Rapporteur
M.	Sven Schewe	Professeur University of Liverpool	Rapporteur
Mme	Patricia Bouyer-Decitre	Directrice de Recherche CNRS	Examinatrice
M.	Arnaud Durand	Professeur Université Paris 7	Président
M.	Dietmar Berwanger	Chargé de Recherche CNRS	Directeur de thèse
M.	Laurent Doyen	Chargé de Recherche CNRS	Directeur de thèse

Laboratoire Spécification et Vérification
École Normale Supérieure Paris-Saclay, UMR 8643 du CNRS
61 avenue du Président Wilson, 94235 Cachan Cedex, France

Résumé

De plus en plus de tâches informatiques sont effectuées par des systèmes interactifs qui impliquent plusieurs agents distribués, qui n'ont qu'une connaissance locale de l'état global du système, et leurs propres ressources de calcul. Planifier les actions de ces agents partiellement informés de telle sorte qu'il coopèrent pour accomplir la tâche, et ce quelque soit le comportement de l'environnement, soulève un important défi de conception, connu comme le problème de la *synthèse distribuée*. Ceci motive le désir de mieux comprendre les mécanismes de l'interaction. Au niveau théorique, deux questions principales émergent: Qu'est-ce qui rend l'interaction entre agents si difficile à gérer? Et, comment trouver des solutions correctes au problème de la synthèse distribuée de façon automatique?

Notre travail explore ces questions par le prisme des *jeux synchrones sur graphes finis et à information imparfaite*. Les jeux sur les graphes sont une manière puissante de modéliser les systèmes distribués, car ils capturent des caractéristiques clés des situations d'interaction. La structure de graphe permet de modéliser des interactions qui impliquent des choix non-déterministes de l'environnement, et qui se déroulent sur une durée illimitée. Les actions des agents (*joueurs*), ainsi que les choix de l'environnement, déterminent l'*état* actuel du jeu, qui porte une observation privée pour chaque joueur. L'objectif commun des joueurs est exprimé comme une *condition de gain* définissant les séquences infinies d'états (*parties*) qui correspondent aux exécutions correctes du système.

Planifier les actions des joueurs pour remplir une condition de gain commune, en dépit de l'information imparfaite, revient à construire une stratégie gagnante jointe, c'est-à-dire une fonction qui prescrit une action concrète à choisir pour chaque état possible du jeu. Le problème de la synthèse distribuée peut être reformulé pour le cadre des jeux en deux parties: La *résolubilité* est la question de savoir si, pour un graphe fixé, il existe une stratégie gagnante commune qui satisfait une condition de gain donnée. L'*implémentabilité* est la tâche de construire, en se basant sur des automates finis, une stratégie gagnante, s'il en existe une. La contribution que l'on présente ici est articulée en trois parties.

Premièrement, on s'intéresse à la question de la difficulté intrinsèque des phénomènes interactifs, en étudiant une variante de notre modèle général de jeux, les *consensus game acceptors*. En considérant une des formes les plus simples de jeu, où les joueurs ont à s'accorder sur une unique décision simultanée après avoir passivement observé une séquence finie de symboles produits par l'environnement, en fonction d'un graphe de corrélation, on montre que, sous hypothèse d'information imparfaite, la condition de consensus est suffisante pour causer l'indécidabilité. En examinant la structure de ce graphe de corrélation, on obtient un aperçu de la *frontière de décidabilité* des jeux à information imparfaite. En utilisant les outils de la théorie des langages formels, on peut classifier

la complexité d'*exécuter* des stratégies gagnantes: On identifie des cas où des instances de consensus game acceptors requièrent la puissance de calcul des automates à piles non-déterministes ou des automates linéairement bornés.

Ensuite, on se concentre sur des cas décidables et sur la façon dont l'information est rendue accessible aux joueurs. On identifie un motif de flot d'information parmi les joueurs qui assure la décidabilité: l'information *hiérarchique*. Un jeu est considéré à information hiérarchique quand l'information à propos de l'historique actuel est accessible aux joueurs de manière ordonnée, de telle sorte qu'il n'y a pas deux joueurs ayant de l'information incomparable. On présente trois variantes de l'information hiérarchique: L'information hiérarchique *statique*, où il existe une hiérarchie d'information fixe entre les joueurs pour le jeu entier, l'information hiérarchique *dynamique*, où la hiérarchie d'information peut changer au cours d'une partie, et enfin l'information hiérarchique *récurrente* avec condition de gain observable, où il peut y avoir des phases à information incomparable, mais avec la garantie qu'une hiérarchie sera rétablie à nouveau dans le futur. Avec ces variantes, on élargit le paysage des architectures décidables.

Enfin, on considère une source particulière d'information imparfaite dans les jeux, qui peut apparaître en pratique: les *délais*. On regarde la situation dans laquelle la délivrance des signaux aux joueurs n'est pas instantanée, les empêchant de réagir comme prévu initialement par une stratégie basée sur un système fonctionnant parfaitement. On montre que sous certaines restrictions du modèle de jeux et pour une classe réaliste de conditions de gain, un scénario où les signaux sont délivrés avec des délais bornés n'est pas fatal pour résoudre la synthèse distribuée. Développant la technique de *delayed-response* de la littérature de théorie des jeux économique, on élabore une procédure qui prend en charge la structure de système de transition à états multiples de notre modèle de jeux, et, en recombinant prudemment des stratégies gagnantes dans les instances de jeu avec monitorat instantané, construit des solutions au problème de la synthèse distribuée différée qui préservent les équilibres.

Summary

More and more computing tasks are performed by interactive systems that involve several distributed agents, which have local knowledge about the global state of the system and their own computing resources. Planning the actions of such partially informed agents so that they cooperate to complete the task, regardless of the behaviour of the environment, raises a significant design challenge, known as the *distributed synthesis problem*. This motivates the desire to better understand interaction mechanisms. On a theoretical level, two main questions arise: What makes interaction between agents so difficult to handle? And, how to find correct solutions of the distributed synthesis problem in an automated way?

Our work investigates these questions through the prism of *synchronous games on finite graphs with imperfect information*. Games on graphs are a powerful way to model distributed systems, as they capture key features of interactive situations. The graph structure allows to model interactions that involve non-deterministic choices from the environment, and that unfold over an unlimited time period. Actions of the agents (*players*), along with the choices of the environment, determine the current *state* of the game, which carries private observations for each player. The common objective of the players is expressed as a *winning condition* defining the infinite sequences of states (*plays*) that correspond to correct executions of the system.

Planning the actions of the players to fulfil a common winning condition, despite the imperfect information, amounts to constructing a joint winning strategy, that is, a function that prescribes a concrete action to take in each possible state of the game. The distributed synthesis problem can be reformulated for the game setting in two flavours: *Solvability* is the question whether, on a fixed graph, there exists a joint winning strategy that satisfies a given winning condition. *Implementability* is the task of constructing, relying on finite-state automata, a joint winning strategy, should one exist. The contribution we present is three-fold.

First, we address the question of why interaction is difficult to handle, by studying the *consensus game acceptor* variant of the general game model. Considering one of its simplest forms, where players have to agree on only one simultaneous decision after passively observing a finite sequence of symbols produced by the environment according to a certain correlation graph, we show that, under imperfect information, the consensus condition is enough to cause undecidability. Investigating the structure of this correlation graph, we gain insight about the *frontier of decidability* of games with imperfect information. By using tools of formal language theory, we can also classify the complexity of *executing* joint winning strategies: We identify cases in which consensus acceptor games instances

require the computing power of non-deterministic pushdown automata or linear-bounded automata.

Second, we focus on decidable cases and the way information is made accessible to the players. We identify a pattern of information flow among players that leads to decidability: *hierarchical* information. A game has hierarchical information when the information about the current history is accessible to the players in an orderly fashion, so that no two players have incomparable information. We present three variants of hierarchical information: *Static* hierarchical information, where there is a fixed information hierarchy of players for the whole game, *dynamic* hierarchical information, where the information hierarchy can change along a play, and finally *recurring* hierarchical information and observable winning conditions, where there can be phases with incomparable information, but with the guarantee that there will be a hierarchy again in the future. With these variants, we enlarge the landscape of decidable architectures.

Finally, we consider a particular source of imperfect information in games, that can arise in practice: *delays*. We look at the situation where the delivery of signals to players is not instantaneous, preventing them to react as intended initially by a strategy based on a perfectly running system. We show that, under certain restrictions on the game model and for a realistic class of winning conditions, a signalling scenario with bounded delays is not fatal to solving the distributed synthesis problem. Extending the *delayed-response* technique from the economical game theory literature, we design a procedure that handles the transition state-structure of our game model, and constructs, by carefully recombining winning strategies from instant monitoring game instances, solutions to the delayed distributed synthesis problem that preserve the equilibrium outcomes.

Acknowledgements

First, I would like to sincerely thank Véronique Bruyère and Sven Schewe, who have had the kindness to accept the role of reviewers, even though the deadline was short, and their schedule already full. Merci beaucoup and vielen Dank for your reviews and your feedback.

Je voudrais aussi bien sûr remercier Patricia Bouyer et Arnaud Durand pour avoir accepté de faire partie du jury, encore une fois avec un temps limité avant la soutenance, et des emplois du temps bien chargés. Merci beaucoup Arnaud pour m'avoir toujours encouragée, depuis ce jour où j'ai débarqué de Caen dans une réunion d'information sur le LMFI.

Un très grand merci à Laurent Doyen pour avoir accepté d'être mon directeur et à Dietmar Berwanger pour m'avoir encadrée avec patience et bienveillance. La somme de tout ce que j'ai appris sous votre houlette dépasse largement le contenu de ce manuscrit.

Ma gratitude vient ensuite aux collègues du LSV, qui m'ont offert un entourage professionnel qui m'a permis d'ouvrir mes horizons scientifiques et de profiter d'un quotidien agréable au laboratoire. Mention spéciale au 4ème et à ses habitants, en particulier à ceux qui ont partagé mon bureau au fil du temps: Wojtek, Nadime, Emilien, Anup, David, Karima, Samy et Pierre. Merci à tous les doctorants pour les goûters des moniteurs et pour les bons moments: Mahsa (I miss our girly time!), Julien, Vincent, Guillaume, Lucca, Jérémy, Simon, Simon, Nathann, Daniel. Un deuxième très grand merci à Nadime (parce que trois, c'est trop) et à Guillaume pour m'avoir si souvent écoutée et guidée. Je voudrais également remercier les personnels administratifs et techniques du LSV, Virginie, Catherine, Thida, Imane, Francis et Hugues pour leur efficacité et leur grande gentillesse. Merci à l'équipe pédagogique qui m'a permis d'enseigner à Cachan dans de très bonnes conditions et en confiance, Paul, Serge H., Claudine et Hubert, l'expérience acquise à vos côtés est précieuse. J'ai bénéficié pendant ces quatre ans de conseils avisés et d'encouragements dans les moments de doute, qui furent nombreux, et pour cela je voudrais remercier notamment Alain (et ses super formations), Serge A. (et ses pointeurs vers les concours d'enseignement), Stéphane D. (et sa bienveillance), Thomas et Étienne pour les discussions quasi quotidiennes aux pauses café ou chocolat, mais pas que, et enfin un merci tout particulier à Stéphanie pour son suivi, ses encouragements et son aide au long cours.

Je n'aurais sûrement pas fini ma thèse si je n'avais pas eu l'opportunité de faire une quatrième année tout en enseignant à Versailles. Merci infiniment à Florent et Yann pour m'avoir parlé du poste, à Sandrine et Franck pour m'avoir accueillie au sein du département et m'avoir fait découvrir le monde merveilleux de l'IN100, ainsi qu'à Thierry pour avoir été le parfait responsable de cours, ce fut une très belle expérience de travailler avec vous.

Merci aux copains qui sont là, envers et contre tout, mais surtout avec moi, votre amitié me porte, n'en doutez pas. D'abord celles qui sont là depuis la période glorieuse de l'adolescence, Laura, Asma, Clémence, Juliane, Cécile. Ceux que la fac de maths de Caen m'aura apportés, Lise, Seb, Anais. Les heureuses conséquences de mon année à Dresde: Nico, Aurélien, Benoît, Lucien, Carole, Pedro, Julia. Les parisiens, Maÿlis pour m'avoir recueuillie rue Esquirol, Schweizer et son élégance de la pensée, Hadrien mon puits de culture, Maëlle et son dynamisme. À mon déménageur de choc Jocelyn, merci pour tout, (et surtout pour être aussi ronchon que moi!).

Merci à mon vieux et à ma Maman, notamment pour avoir toujours remis les choses en perspective. Enfin Rémy, merci d'avoir eu confiance en moi, bien souvent à ma place, et tout simplement d'être à mes côtés.

Contents

Résumé 1
Summary 3
Acknowledgements 5

1 Introduction **9**
1.1 Motivation 9
1.2 Imperfect Information 10
1.3 Information-Flow Patterns 10
1.4 Games on Graphs 12
1.5 Related Research 13
1.5.1 Asynchronous Setting 13
1.5.2 Quantitative Games 13
1.5.3 Stochastic Games 14
1.6 Outline 14
1.6.1 Consensus condition 14
1.6.2 Hierarchical patterns 16
1.6.3 Delayed monitoring 17
1.7 Organisation of the Thesis 17

2 Preliminaries **19**
2.1 Context 19
2.1.1 Church's problem 20
2.1.2 Distributed systems 22
2.2 Games with Imperfect Information 26
2.2.1 Game graph 26
2.2.2 Plays and histories 28
2.2.3 Indistinguishability relation 29
2.2.4 Strategies 29
2.2.5 Winning conditions 30
2.3 Background 32
2.3.1 The case of perfect information 32
2.3.2 The case of one player against Nature 35
2.3.3 Distributed games 38

3 Consensus Condition **41**
3.1 Consensus Game Acceptors . . . 42
3.2 Describing Languages by Games . . . 44
3.2.1 Characterising regular languages . . . 46
3.2.2 Domino frontier languages . . . 50
3.2.3 Uniform encoding of domino problems in games . . . 51
3.3 Characterising Context-Sensitive Languages . . . 53
3.4 Consensus and Iterated Transductions . . . 54
3.5 Consensus Games for Context-Free Languages . . . 58
3.6 Discussion . . . 63

4 Information Hierarchies **65**
4.1 Finite-State Strategies and Automata . . . 66
4.2 Static Hierarchies . . . 67
4.2.1 Hierarchical observation . . . 67
4.2.2 Incorporating perfect recall . . . 69
4.2.3 Signals and game transformations . . . 71
4.3 Dynamic Hierarchies . . . 72
4.3.1 Information rank signals . . . 74
4.3.2 Smooth overtaking . . . 75
4.3.3 Shadow players . . . 78
4.4 Transient Perturbations . . . 79
4.5 Monitored Architectures . . . 87
4.6 Discussion . . . 88

5 Delayed Signals **91**
5.1 Framework . . . 92
5.1.1 Repeated games . . . 93
5.1.2 From repeated games to games with multiple states . . . 97
5.2 Games with Delayed Signals . . . 99
5.2.1 General model . . . 99
5.2.2 Instant and bounded-delay monitoring . . . 101
5.2.3 Shift-invariant, submixing utilities . . . 102
5.2.4 The transfer theorem . . . 103
5.3 Proof . . . 103
5.3.1 Unravelling small cycles . . . 103
5.3.2 The Frankenstein procedure . . . 104
5.3.3 Correctness . . . 105
5.3.4 Equilibrium condition . . . 106
5.3.5 Finite-state strategies . . . 107
5.4 Discussion . . . 107

Bibliography **109**

A Résumé (long) en français **117**

Chapter 1

Introduction

Interaction between computational agents is an ubiquitous phenomenon in practice. For instance, on an online booking platform, several independent agents have to cooperate with one another: a customer, an airline and a hotel. Indeed, the airline and the hotel have to propose suitable options to the customer, and then proceed to the booking. However, every agent has a partial view on the situation: the customer has to take decisions despite the risk of the hotel or airline overbooking, while the hotel and airline do not know their respective offers, and whether the customer will actually use their service or has alternative plans that he may prefer in the end. Such systems, featuring several independent agents, are called *distributed systems*. One can see that, in these systems, information about the other agents is crucial, but may be limited due to the infrastructure of the system and each agent's partial view of the global system. A natural goal is to be able to coordinate in spite of the missing information. This implies designing programs to prescribe the behaviour of the agents, and corresponds to the *synthesis* problem for distributed systems. In this work, we investigate distributed synthesis on a theoretical level, on the model of *games with imperfect information* and via the analysis of *information-flow patterns* in different interactive scenarios.

1.1 Motivation

Situations involving cooperation between several independent agents, which can be modelled as distributed systems, arise in many different contexts. Building systems automatically so that they are correct by design has been a persistent ambition of the computing science. This raises the desire to better understand the role of imperfect information in distributed synthesis. Indeed, achieving cooperation under uncertainty is known to be hard, even undecidable in general [70], and uncovering the intrinsic causes of this hardness and finding classes of interactive systems for which specifications are enforceable would let us gain insight on a wide range of design issues and suggest approaches to overcome them. In this thesis, we address the two following main questions: (1) What makes interaction between agents so difficult to handle? (2) And, how to find correct solutions of the distributed synthesis problems in an automated way?

Viewing computation in a broad sense as an interactive process is an approach with strong roots in the theoretical computer science tradition: The fundamental concept of alternation, introduced by Chandra, Stockmeyer and Kozen [20] in the early eighties, where computation steps are attributed to conflicting players seeking to reach or avoid certain outcome states, relied on determined games with perfect information and gave important results, notably in automata theory. Concurrently, Peterson and Reif [69] initiated a study on computation via games with imperfect information, involving teams of players with a setting highly expressive, but also difficult to comprehend. However, it reveals games as a central analytic tool for modelling interactive scenarios, and as a framework rich enough to be interesting in its own right. Throughout this work, we address the questions (1) and (2) taking the perspective of information-flow patterns.

1.2 Imperfect Information

In distributed systems, information may be accessible only to a limited extent to the agents, depending on the way agents are connected, the unpredictability of the environment, and possible implementation failures. Games with *imperfect information* model the partial view of the agents on the whole system. There are different reasons that lead to imperfect information in games, more precisely, we distinguish two main *sources of uncertainty* :

First, uncertainty can arise from the structure of the distributed system itself. In our model, it is represented by the graph structure underlying the game. Global states of the system correspond to positions on the graph, and carry with them private information for the players in the form of *observations*. Transitions from one global state to another are represented by directed edges between positions in the graph, labelled with action profiles. Outgoing edges labelled with the same action profile but ending at different states account for the non-determinism induced by Environment/Nature's influence on the system. These structural properties allow for uncertainty caused by Nature's uncontrollable behaviour, and the limited field of vision of the players.

Second, failures due to the reality of the system execution may arise and cause uncertainty. In practice, several parameters can indeed alter the planned behaviour of a distributed system. Communication failures can be, to a certain extent, modelled in the game framework. Incorporating uncertainty about the delivery of observations is a way to represent mishaps in the communication process. However, the fact that such failures occur during the execution phase makes it difficult to embed this possibility directly in the game graph structure. Thus, we model them at the *monitoring* level, that is, the abstract layer that takes care of the time parameter and on how players actually receive observations throughout a play.

1.3 Information-Flow Patterns

Imperfect information in games allows to model distributed systems in scenarios where agents are not omniscient. In order to achieve their objective, the agents of such a system have to cooperate, and that most usually involves sharing or aggregating information, to overcome the handicap of individual partial knowledge. The way information flows in the

system, i.e. how agents receive information from the environment or communicate with other agents, is critical. Analysing the shape of the information flow in a game lets us gain insight on the complexity of the strategy synthesis, find decidable cases for distributed synthesis and alleviate difficulties to synthesise winning strategies under realistic perturbations. Therefore, focussing on identifiable patterns of information flow, we are able to exhibit classes of games that are representative of different aspects of interaction and its inherent difficulties. In the development of this thesis, we look at the information flow in games from three different angles:

Two-directional information flow In the most general case, information can be gained by any of the players and shared to support the cooperation towards the common objective. Therefore, the information gathered by individual players impacts the unfolding of a play. Actually, the extent of this impact is tremendous: It turns out that uncertainty about the knowledge of partners propagates iteratively, in the sense that a player, in addition to his own observations, has to take into account the possible observations of a partner, the speculations of the partner on his knowledge and so on and so forth. In our model, private observations for the players are attached to the states of the game, or positions of the graph. Therefore, the graph structure of a game can be seen as a *correlation graph* for observations of players. Iterating the relation described by the correlation graph gives insights about the complexity of solving games and executing winning strategies.

One-directional information flow Another perspective is to consider the way the information is distributed among the players. If, in general, distributed synthesis is undecidable, it becomes decidable when information flows in only one direction, that is, when there exists an order among players from the most informed to the least informed. Indeed, the fact that a player's information determines the information of players lower in the hierarchy eliminates the need to reason about other's knowledge, and thus the consequent propagation of uncertainty. In this work, we investigate three patterns of *hierarchical* information flow that yield decidability: *static* hierarchical information, where the hierarchy is fixed for the whole play; *dynamic*, where the hierarchy is allowed to change during the play; and *recurring*, where ephemeral phases without hierarchy can occur.

Information flow perturbations In practice, even with the most careful system design, hardware or software failures can happen. The task of designing strategies that would cope with such imperfect information is challenging, due to the many types of perturbations that can arise at the implementation level. In this work, we look at cases where the information flow is perturbed by a fault in the mechanism that handles the delivery of observations to players. Relaxing the assumption that the system is equipped with a perfect communication infrastructure and that every observation is accessible instantaneously to players is a way to describe real-life induced imperfect information at the modelling level. More precisely, we investigate the scenario where players do not receive their observations instantaneously but with finite delays. It turns out that distributed synthesis for some relevant classes of games can recover from such perturbations.

1.4 Games on Graphs

Our investigation is carried out through the prism of *synchronous games on finite graphs with imperfect information*. Games on graphs are a powerful way to model distributed systems, as they capture key features of interactive situations. We are interested in finite-state systems, thus we focus on finite graphs. Agents of a distributed system are modelled by *players* in a game. Each player has its own finite set of *actions* and finite set of *observations*. The environment behaviour in a distributed system is modelled partly as a distinguished player, called *Nature*, and by the way observations are distributed over the graph structure. The underlying graph represents the transition structure of a distributed system: nodes correspond to the different states of the system, while edges correspond to transition between states. Every edge is labelled with an action profile, that is, one action for each player, that determines, along with the choice of Nature, the next position on the graph, that is the next state of the game. In our model, we assume that there is no *dead-end*: at every state, each action profile leads to a successor state. This finite graph structure allows to model interactions that involve non-deterministic choices from the environment, and that unfold over an unlimited time period. Executions of a distributed system are modelled as a (possibly) infinite alternating sequence of states and action profiles, or a *play*. Specifications on the system behaviour are expressed as *winning conditions* in the form of *ω-regular* sets of plays. Planning the actions of the players is done by defining *strategies*, that is, functions that prescribe an action to take for each possible play prefix or *history*. Doing this in a way which ensures that the unfolding play satisfies the winning condition, regardless of Nature's choices amounts to defining *winning strategies*.

We outlined earlier what imperfect information meant in the context of interactive systems. In our model, it translates to the notion of *observations*. Each player has his own finite observation alphabet, and each state of the game graph is labelled with an observation profile. This labelling gives rise to observation functions, one for each player. Upon reaching a state, each player is assumed to receive his part of the observation profile. Observation functions may not be injective. Some states can look the same to a player, inducing an equivalence relation on the states set of the graph that we call the *indistinguishability* relation. As we also assume the players to have *perfect recall*, the indistinguishability relation naturally extends to histories and plays, and this has an effect on strategies design: Indeed, there may be several histories that look the same for a player, and, as strategies are functions implemented by deterministic machines, they have to prescribe the same action to all histories that are indistinguishable to be valid.

Another significant aspect of our model is that it is *synchronous*: It means we assume that the system is equipped with a global clock that every component of the system can refer to and that times the whole execution of the system. In terms of games, it corresponds to the assumption that every move on the graph happens after each player simultaneously chooses *one* action, so that transitions respects the global tick of the system. In addition to this global clock, we choose to work with a game model where actions of the players are *simultaneous*, in the sense that each transition from a state to another is assumed to be triggered by a complete action profile, with one action per player. This is a convenient and compact way to represent features of interaction situations under imperfect information.

The central problem of distributed synthesis for games with imperfect information is composed of the following two facets: *Solvability* is the question whether, on a fixed graph, there exists a joint winning strategy that satisfies a given winning condition. *Implementation* is the task of constructing, relying on finite-state automata, a joint winning strategy, should one exist.

1.5 Related Research

As distributed synthesis is undecidable in the general case ([69], [70], [82], [47], [77]), great efforts have been directed towards identifying computationally manageable classes. To do so, several approaches that involve restricting the way the information flows between players have been explored ([53], [36], [51], [72], [37]). Furthermore, the wide variety of scenarios where coordination between independent agents has to be achieved lead to different settings being investigated.

1.5.1 Asynchronous Setting

Contrary to our model, where players perform their action on a schedule regulated by a global clock, in the asynchronous case, agents are equipped with their own local clock, and have to actively synchronise with other agents to perform certain actions. This approach captures a certain aspect of the reality of designing distributed systems: Indeed, it is often the case in practice that components do not have access to a shared and reliable clock. Assuming there is no constraint on the number of actions any agent can make during an abstract global time unit models this feature well. A consequence of this absence of an implicit agreement on time is that, in order to accomplish their common goal and cooperate correctly, agents have to synchronise on shared actions to gain information on the global state of the system. In this setting, distributed synthesis is decidable only for systems where at most one process is a black-box [78]. Another approach considers agents to have *causal memory*: the schedule of communication can be specified, but the information transmitted during synchronisation events cannot be limited, as agents gather and forward as much information as possible about the past. Therefore, a new type of automata is needed to model the computing processes: the most widely used is the model of *asynchronous automata*, or *Zielonka automata*. These automata have indeed been the common ground for research on asynchronous systems since their introduction by Zielonka in [89]. This leads to more decidable classes, such as series-parallel systems [40], well-connected architectures [41], and, more recently, acyclic architectures [64].

1.5.2 Quantitative Games

In this thesis, we focus on ω-*regular* winning conditions in Chapters 2, 3 and 4, which are *qualitative* objectives, in the sense that players either win or lose. However, several problems involving interaction have a *quantitative* aspect, for instance when players try to optimise the value of a certain parameter during an execution of the system, and which we consider in Chapter 5. *Weighted* games, where transitions are labelled with *weights* or *payoffs* to evaluate the performance of an action (profile), allow to model this kind of

situations [12]. Depending on the problem, the interpretation of the payoffs may vary, for instance they can represent power consumption levels that players wish to maintain around a certain value, as in the setting of *Energy Games*, see [19] or [50], or individual rewards for players that want to maximise their average payoff in the long run, as in [91] or [33]. Games with quantitative objectives and imperfect information are hard already for two players: while in this setting, games with ω-regular winning condition are decidable, mean-payoff games are undecidable in general, and restrictions are necessary to obtain decidability [31]. As the situations are diverse, the form of the desired solutions can differ: from the classical Nash Equilibrium [66], to the more refined *Subgame Perfect* Equilibrium and *Secure* Equilibrium concepts, as in [26], [45], [16] or [17]. More recently, one can cite the approach of [24] that studies the effect of additional time bounds constraints on quantitative objectives.

1.5.3 Stochastic Games

We stated earlier that, in our model, Nature resolves non-determinism. In fact, we incorporate non-determinism as an abstraction of the uncontrollable behaviour of the environment (Nature), but we do not make any assumption on a potential probability distribution over the space of moves. The *stochastic* game framework allows to refine the expectation on Nature's behaviour by considering the effect of different probability distributions on the synthesis of winning strategies. The counterpart is that perfect information of the state is generally assumed. Players are also equipped with stochastic choice functions to implement *mixed* strategies: while in our setting, a strategy maps a state to an action, a mixed strategy maps a state to a probability distribution over the actions space. An important consequence of considering stochastic behaviour is that it allows for variants of the "winning" objective, introducing for instance the notions of *almost-surely* winning strategies, that is, strategies that ensure a win with a high probability threshold. For a survey on concurrent stochastic games with ω-regular objectives, see [25], and [21], [46] or [23] for more recent developments. Stochastic games with quantitative objectives have also been investigated, see for instance [15]. As we focus on the relation between information-flow patterns and distributed synthesis under imperfect information, in order to identify significant correlation between different patterns and the complexity of finding winning strategies, we choose to keep the model free of stochastic considerations.

1.6 Outline

In this section, we sum up the main contributions of this thesis.

1.6.1 Consensus condition

In a first approach to study interaction mechanisms, we investigate the dependencies created in the information flow by the observation of players. To do so, we restrict our general game model to a setting where two players cooperate against Nature and have to agree on a single yes/no decision after receiving a finite sequence of observations, or *inputs*, selected by Nature along the game graph. The winning condition we consider

is a *consensus condition*. The single decision is bound to be taken at designated *final* states, each of them associated with a set of *admissible* decisions, and carrying the same observation, so that players may not always be able to distinguish between them. The decision one player has to take at these final states is then not only dependent on what he has observed as an input, but also on what his partner has seen as his own private input. As the phase of receiving their input is passive and entirely driven by Nature, no communication is possible between players during a play. Thus, they have to "deduce" on their own which information the other player has to guarantee a *safe* decision, that is a decision that is both admissible and will be chosen by his partner as well. In our model, private observations of the players are attached to the states of the game, or positions of the graph. Therefore, the graph structure of a game can be seen as a correlation graph for observations. This helps to isolate the role of imperfect information has in the difficulty of solving distributed synthesis: the uncertainty one player has on the state of the game propagates on to the other player's information, which in turn has to be considered by the first player, and so on. In order to construct winning strategies for each player, the transitive closure of the *indistinguishability* relations of both players have to be taken into account. A second important aspect of these games is that the yes/no decision at the end of an observation phase actually amounts to accept or reject a finite input word: We call them *consensus game acceptors* to reflect that. Once we view consensus game acceptors as devices to recognise sets of words over finite alphabets, the connection with formal languages is natural and gives us tools to, first, prove undecidability of distributed synthesis in the general case, and second, provide a classification of consensus acceptors games in terms of complexity of executing a winning strategy, by showing correspondence with classes of formal languages.

The first contribution here is to revisit the classical proof of undecidability of the distributed synthesis problem by showing a reduction from the emptiness problem for context-sensitive languages. It gives us insight on the frontier of decidability of games with imperfect information by identifying cooperation, in the form of consensus, as a crucial criterion that is sufficient to cause undecidability, as the other factors, communication between players and multiple or infinite decision making during the play, have been eliminated from the model. The second contribution is the classification in terms of formal languages. By using the tools and results of formal languages theory, we show how the shape of the correlation graph of a consensus game acceptors determines the complexity of executing a winning strategy for the players. With the help of domino tiling results, we obtain the correspondence between consensus game acceptors and context-sensitive languages, and thus with linear-bounded automata. We then refine our classification by exploiting logical characterisations of formal languages and results on transducers, obtaining correspondence with context-free languages and smaller subclasses as deterministic context-free languages and Dyck languages. This approach via formal languages represents a departure from the usual focus on finite-state winning strategies, as we obtain strategies that can rely on pushdown automata or linear bounded automata to be executed.

These contributions are based on the results published with Dietmar Berwanger in the proceedings of the 19th International Conference on Developments in Language Theory (DLT 2015) [8], and the corresponding technical report [10].

1.6.2 Hierarchical patterns

After looking closely at the consequences of imperfect information on the information flow in games, we focus on *hierarchical information-flow patterns*. *Hierarchical* systems, that is, systems where the information is distributed among agents in an orderly fashion, forming a *hierarchy* from the most informed to the least informed agents, represent a fundamental case in which the distributed synthesis problem becomes decidable. The fact that each agent has access to the observations received by agents below in the hierarchy makes the analysis of the system simpler: intuitively, it is easier to cooperate with agents that can be simulated with one's own information. Already in 1979, Peterson and Reif [69] showed that, for games in this setting, it is decidable — although, with non-elementary complexity — whether distributed winning strategies exist and that, if so, finite-state winning strategies can be effectively synthesised. Later, the result was extended by Pnueli and Rosner [70] to the framework of distributed systems over fixed linear architectures where information can flow only in one direction. Kupferman and Vardi, in [53], developed a fundamental automata-theoretic approach that furthermore extends the decidability result from linear-time to branching-time specifications. Lastly, Finkbeiner and Schewe show in [36] the existence of an effective criterion on communication architectures that guarantees the decidability of the distributed synthesis problem: the absence of *information forks*, which implies a hierarchical order in which processes have access to the observations provided by the environment.

In this work, we go beyond the hierarchical observation pattern. We introduce relaxations of the hierarchy principle that lead to decidable cases and that are recognisable with relatively low complexity. More precisely, we first go from hierarchical *observation* to hierarchical *information* by assuming players have perfect recall. In this case, the decidability result is a direct consequence of the one from [70], [53] and [36]. To prove this, we introduce the tool of *finite-state signals* and explain how *deducing* is as helpful as *observing* in distributed synthesis problems. Secondly, we relax the (static) hierarchical information pattern by considering *dynamic* hierarchical information, that is when the information order among players changes along a play. We show how to reduce this case to the static one by using finite-state signals to construct an equivalent *shadow* game that has static hierarchical information. Intuitively, in the shadow game, the i-th shadow always plays the role of the current i-th most informed player in the original game, so that the shadow players are organised in a static hierarchy. Finally, we introduce *recurring* hierarchical information that corresponds to cases where there can be transient phases of perturbations of the hierarchical information. One can show that, for observable winning conditions, the distributed synthesis problem is decidable for games with recurring hierarchical information, by using the information tracking construction from [5]. These new classes broaden the landscape of decidable games with imperfect information. Furthermore, it highlights the fact that the game framework is flexible enough to model communication and information schemes that cannot directly captured by communication architectures, as we discuss in Section 4.5.

These contributions are based on the results published with Dietmar Berwanger and Anup B. Mathew in the proceedings of the 13th International Symposium on Automated Technology for Verification and Analysis (ATVA 2015) [6], and the corresponding technical

report [7].

1.6.3 Delayed monitoring

Finally, we shift our focus from looking at uncertainty caused by the structure of a distributed system itself to investigating the consequences of a source of uncertainty arising in practice: *delays*. We consider the case where the monitoring of a game, that is, the usually abstracted implementation layer that takes care of the players actually receiving information during an execution, may deliver signals with a finite and bounded delay. We study the impact of the delayed-monitoring of a game on the distributed synthesis problem: More precisely, we ask ourselves if delays are fatal to the existence of winning strategies for games that are solvable in the instant-monitoring setting. The delayed-monitoring case has been successfully addressed in economical game theory, and we are directly inspired by the work of Fudenberg, Ishii and Kominers in [38] that presents a transfer result that allows to construct equilibria-preserving strategies in the delayed-monitoring version of a game from strategies in the instant-monitoring version. This transfer result relies on the *delayed-response* technique, whose key idea is for the players to wait for the maximal delay before reacting to a signal. This way, it is guaranteed that all the players have received the information concerning a certain stage of the game, and constructing the delayed-response strategies amounts to recombine orderly *threads* of the instant-monitoring game. However, they work within the framework of *repeated games*, that correspond to the particular case of games with one state in our terminology.

In this work, we adapt their technique to games with several states. To do so, we introduce a new model of synchronous games with imperfect information, where the observation signals carrying information on the actions of the players are no longer attached to states, and where delays are modelled. We then present the *Frankenstein* procedure that effectively constructs strategies for the delayed monitoring versions of games solvable in the instant-monitoring case, relying on a reduction of a delayed-monitoring game to a collection of instances of an instant-monitoring game. We also identify the class of games, for which this transfer is applicable: games with *shift-invariant* and *submixing* utilities.

These results were published with Dietmar Berwanger in the proceedings of the 35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science Foundations (FSTTCS 2015) [9].

1.7 Organisation of the Thesis

In Chapter 2, we introduce the basic concepts and results that are necessary to proceed to our core work. To start, we briefly recall how synthesis, and later distributed synthesis, became prominent problems in theoretical computer science. Then, we detail the model of synchronous games on finite graphs with imperfect information and its links with other classical game models of the literature. We explicit our model by presenting the different notions on a simple example. Finally, we state the fundamental results on games on graphs that are preconditions to obtain the results in the remaining of the manuscript.

In Chapter 3, we restrict our general model so as to simplify the framework as much as possible. We focus on what we call *consensus game acceptors*: games with imperfect

information on finite graphs for two players against Nature, where players have a unique decision to take per play, in order to win or lose. They are bound to choose between accepting or rejecting finite sequences of observations. To win a play, they have to choose according to the graph structure (some states allow both decisions, while some allow only one) and in *consensus*, that is, agree on a common decision. In this chapter, we investigate the consequences of the information flow from and back to a player, by uncovering how the indistinguishability relations of each player are intertwined. We present an alternative proof of the undecidability of distributed synthesis and a classification of consensus game acceptors in terms of formal languages.

In Chapter 4, after looking closely at the flow of information from and back to one player, we consider a setting with more classical properties, that is, a game graph were actions are not limited to some specific final states. The perspective here is a more global one, as we focus on the way the information is distributed to the players. We investigate the synthesis of distributed strategies in cases where the information flow is ordered, in the sense that there exists a hierarchy among players, from the least informed to the most informed, with no incomparable information sets. We show that these cases lead to decidability of the distributed synthesis problem, as well as finite-state solutions and show effective procedures to detect one of the variants of hierarchical information in a game.

In Chapter 5, we consider a perturbation of the information flow that can arise in practice, as we look at the scenario of signals being delivered to the player with a finite delay. To do so, we introduce a slight modification of our game model: we shift the observations from the states of the graph to the edges, and we discuss this change in the beginning of the chapter. We show that, for a realistic class of winning conditions, this delayed monitoring is not fatal to the synthesis of winning strategies. Furthermore, we show that the outcome in the instant monitoring instantiation of a game can be guaranteed in the delayed monitoring one, thanks to a solution procedure that carefully recombines and reorganises strategies designed to be winning in the instant monitoring case.

Chapter 2

Preliminaries

In this chapter, we present basic notions that will be of use in the following chapters. We start by a brief historical account of the main problem we study, i.e. the distributed synthesis problem, before presenting our game model formally, and introducing the basic results that are the foundation of our work.

2.1 Context

As stated in the introduction of this thesis, a motivation to study games is that they are a great framework to study *interaction*. From the original challenge of constructing a machine able to interact with a human user to the more recent and sophisticated one of designing complex systems with several components, understanding the underlying mechanisms and limitations of interaction has been a source to prolific and ground-breaking research. Generally speaking, a *reactive* system describes a system that receives information or data released by an external (therefore, possibly uncontrollable) environment, and has to perform a certain task according to this information, or *input*. We view the environment as an external agent of the system and assume the components of the system to be *controllable*, which means we can prescribe them a certain way to behave. This view of a computing system naturally raises several questions. For instance, we may ask which kind of behaviour we can enforce on the system, both individually, and globally. Then, the question of how hard it is in terms of computation resources to enforce said behaviour arises. Furthermore, given a certain task, is it possible to design a system that will succeed in completing this task regardless of the behaviour of the uncontrollable environment? *Games* offer a framework to study these questions. *Players* model the components of the system, *Nature* models the uncontrollable environment, *strategies* model the prescription of behaviours to the controllable players, and *winning conditions* specify the desired global behaviour of the system. We present now a brief historical review of the fundamental (distributed) *synthesis* problem.

2.1.1 Church's problem

The fundamental, yet simply stated, *Church's problem* ([29], [30]) can be described in this way: Consider a process, that receives an infinite bitstream α as input from its environment, and has to output, bit by bit, an infinite stream β that satisfies certain constraints depending of α. The task is to design a finite-state procedure that produces, for any input sequence α, an output sequence β that respects the constraints. The key feature of this kind of systems is that it is *reactive*. Indeed, the process receives information from the environment, in the form of an infinite stream of bits that we call *input*, and returns an infinite stream of bits that we call *output*. This communication between the process and the environment occurs on *communication channels*. In the case of one process, we only have two channels: an input channel and an output channel. The desired behaviour of the system is expressed as a *specification*, in the form of a logical formula, or a language on pairs of inputs and outputs. More precisely, in the original setting of Church's problem, specification were given as formulae of $S1S$, the monadic second-order logic over the successor structure $(\mathbb{N}, +1)$. These formulae can also be understood as relations[1] over $\{0,1\}^\omega \times \{0,1\}^\omega$. Church's problem can then be stated as follows: Given a formula φ in $S1S$ that specifies a relation $R \subseteq \{0,1\}^\omega \times \{0,1\}^\omega$, does there exist a function $\sigma : \{0,1\}^\omega \to \{0,1\}^\omega$ which is computable by a finite-state automaton (with output), such that $(w, \sigma(w)) \in R$ for all $w \in \{0,1\}^\omega$?

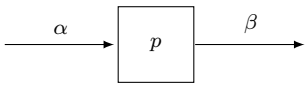


Figure 2.1: Church's problem

Example 2.1 (Church's problem)**.** A reactive system with one process p is represented in Figure 2.1. The environment sends a infinite bitstream α to the process p, which outputs an infinite bitstream β.

This problem corresponds to the base case of the *synthesis* problem. In 1969, Büchi and Landweber [18] solved Church's Problem. Their solution is phrased in the automata theory framework and builds on the fundamental result of McNaughton [60] on the determinisation of Büchi-automata. Their proof methods are quite complicated and we refer to the tutorial [81], which presents a proof that relies on a game theoretic formulation. Indeed, it turns out that the relation between input and output given by an instance of Church's problem can actually be viewed as the winning condition of an infinite game for one player (the process) against Nature (the environment). Moreover, a specification given as an $S1S$-formula can be transformed into a *Muller* automaton that accepts pairs of input and output letter strings. Muller automata can be transformed into *parity* games, such that the emptiness of the automata corresponds to the existence of winning strategies in games. This problem is effectively solvable for parity games, as we will see later

[1] In our context, having ω-regular relations in mind is sufficient.

in Section 2.3.1. Furthermore, the Mealy automaton used to prescribe the behaviour of the process also corresponds to a strategy automaton in the game setting. Thus, we end up with modelling Church's problem in terms of games. A key feature of this problem is that the interaction between the process and the environment is of *infinite* duration. As a result, the input and output on which we specify a relation are words of infinite length, or *ω-words*. In order to satisfy the specification over these infinite input and output, we need devices that handle them. Muller automata and parity automata are such devices. They belong to the class of *ω-automata*. Their structure is similar to the one of automata on finite words, but the acceptance condition has a new form. Instead of looking only at a set of accepting states, we also allow for more involved conditions that take into account the infinite length of the words considered. For instance, one may wish to accept the infinite words that display an infinite number of occurrences of a certain letter. In this case, it is clear that a reachability condition is not sufficient to witness this property. Therefore, we extend the classical acceptance conditions of automata on finite words to ω-regular acceptance conditions for automata on infinite words, following [44].

Definition 2.1 (ω-automata)**.** An ω-automaton is a quintuple $\mathcal{A} := (Q, \Sigma, \delta, q_0, Acc)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, $q_0 \in Q$ is the initial state, and Acc is the *acceptance component*. When the transition function is of the form $\delta : Q \times \Sigma \to Q$, that is, for every state v and every letter a, there exists a unique successor state v', we say that $\mathcal{A}$ is deterministic. A *run* of $\mathcal{A}$ on an infinite word $w = w_1 w_2 \cdots \in \Sigma^\omega$ is an infinite sequence of states $\rho = \rho_0 \rho_1 \dots$ such that $\rho_0 = q_0$, and each pair of consecutive states in the sequence respects the transition function δ : for all $i > 0$, we have $\rho_i \in \delta(\rho_{i-1}, w_i)$ (and $\rho_i = \delta(\rho_{i-1}, w_i)$ if $\mathcal{A}$ is deterministic).

Several ω-regular acceptance conditions have been considered, we focus here on two prominent ones, that use a new ingredient : the *infinite occurrence record* of a run, that is the set of states of the automaton that are visited infinitely often in a run on an infinite word. For a run ρ of an ω-automaton, the infinite occurrence record is denoted $\mathrm{Inf}(\rho)$.

Definition 2.2 (Muller acceptance condition)**.** A *Muller* acceptance condition for an ω-automaton $\mathcal{A}$ is a subset $\mathcal{F}$ of the sets of states of $\mathcal{A}$ such that an ω-word w is accepted by $\mathcal{A}$ if, and only if, there exists a run ρ of $\mathcal{A}$ on w such that $\mathrm{Inf}(\rho) \in \mathcal{F}$.

Definition 2.3 (Parity acceptance condition)**.** A *parity* acceptance condition for an ω-automaton $\mathcal{A}$ is a colouring $c : Q \to \{1, \dots, k\}$, where $k \in \mathbb{N}$ such that an ω-word w is accepted by $\mathcal{A}$ if, and only if, there exists a run ρ of $\mathcal{A}$ on w such that $\min \{c(q) \mid q \in \mathrm{Inf}(\rho)\}$ is even.

We will see later that ω-automata are also a great tool for the game-theoretical approach to infinite duration interaction problems.

The model of finite-state machines (with output) we choose to implement the behaviour of the process is the model of *Mealy* automata, that we describe formally in Definition 2.4. It is a deterministic machine that, on each letter of the input, outputs a letter. It is equivalent to the model of *letter-to-letter transducers*, that we use in Chapter 3. They are also the model used to implement strategies for players in our games, hence we often refer to them as *strategy automata*.

Other approaches to solving Church's problem have been taken. A significant alternative proof from Rabin [71] relies on tree automata techniques. Instead of considering computations of the system as words and pursuing a linear approach like we informally presented above (and the original proof of Büchi and Landweber), Rabin models computations by trees. The tree representation allows to deal directly with the space of all sequence pairs of input and output sequences.

An important property of the setting of Church's problem is that the process has *perfect information*. It knows every bit of information produced by the environment. In addition to the restrictions on the specifications, it is a reason why synthesis works well in this case. When we extend the setting to several processes coordinating to satisfy a specification, *imperfect information* may arise. Indeed, it is likely that a process cannot communicate all the information it receives or produces to all the other processes. In applications it is often the case that processes do not have a complete view of the state of the system. Furthermore, when the processes actually have perfect information, the system can be modelled as a single global process, and reduces to the above presented setting. This motivates the study of distributed systems with several processes.

2.1.2 Distributed systems

When the reactive system is composed of more than one process, there is a need to model the communication between processes formally, namely to introduce *communication architectures*. We have already seen what a process, input and output communication channels are. We extend now these notions to the case of several processes.

A *distributed system* $\mathcal{D} = (\mathcal{A}, (\Sigma_c)_{c \in C})$ *with a set of* $n+1$ *processes* $P = \{p_0, p_1, \ldots, p_n\}$ is a pair composed of a *communication architecture* $\mathcal{A}$ and a family of finite *communication alphabets* $(\Sigma_c)_{c \in C}$, where

- A communication architecture $\mathcal{A}$ has the form (C, r, w) with:
  - C is a finite set of communication channels,
  - each channel $c \in C$ has its own finite *communication alphabet* Σ_c,
  - $r : C \to P$ is a *read* function that assigns a unique *reading* process to each communication channel,
  - $w : C \to P$ is a *write* function that assigns a unique *writing* process to each communication channel.

For a channel $c \in C$, we require that $r(c) \neq w(c)$, that is, there is no communication "self-loop". Furthermore, as each communication channel has its communication alphabet, we can also define the *input* alphabet of a process p as $\bigcup\{\Sigma_c \mid r(c) = p\}$ and its *output* alphabet as $\bigcup\{\Sigma_c \mid w(c) = p\}$. The set $P = \{p_0, p_1, \ldots, p_n\}$ can be partitioned into $P_{con} = \{p_1, \ldots, p_n\}$, which are the processes forming the grand coalition, and p_0, which represents the environment. The communication architecture $\mathcal{A}$ can also be represented as a finite directed graph, where the vertices correspond to processes, and edges are used to model the communication channels, and this presentation is the one we use here.

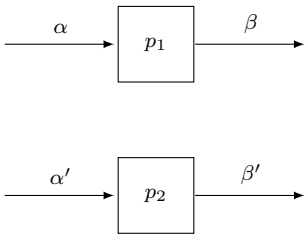


Figure 2.2: A communication architecture with two parallel processes

Example 2.2 (Two parallel processes)**.** *A distributed system with two* parallel *processes p_1 and p_2 is represented in Figure 2.2. By parallel it is meant that the two processes do not have any communication channel between them. The environment sends a infinite bit stream α to the process p_1, which outputs an infinite bit stream β. At the same time, the environment sends a infinite bit stream α' to the process p_2, which outputs an infinite bit stream β'. Here, the communication alphabets are $\{0, 1\}$.*

The synthesis problem

The *synthesis* problem for distributed systems can be stated as follows: Given a distributed system $\mathcal{D}$ and an ω-regular specification L, does there exist an implementation profile for the processes?

Pnueli, Rosner in their seminal paper *Distributed Systems are hard to synthesize* [70] showed that the synthesis problem for two processes and an environment is undecidable, already for *safety* specifications, that is, specifications requiring the system avoids certain states throughout its execution. They show how to reduce the Halting Problem to the distributed synthesis problem for a system with only two processes in parallel, as in Figure 2.2, and a safety specification. It follows that the synthesis problem is undecidable in the general case.

On a positive side, there are some decidability results. In [70] again, Pnueli and Rosner introduced the *pipeline* architectures, depicted in Figure 2.3, and showed that the synthesis problem for this restricted type of distributed systems is decidable, however non-elementarily, for LTL specifications. The key feature of these pipelines is that the information flow is *ordered*, namely that the first process to receive information has at least as much information as the second, and so on, so that no new information ignored by the first process arises in the computation line. In the end, the pipeline could be viewed as a single process where the different tasks are restricted to be performed with a certain amount of information.

In [53], Kupferman and Vardi extend this decidability result to branching time specifications, looking at executions in the form of trees and no longer as paths. They present three classes of communications architectures for which the synthesis problem is decidable for branching time specifications. First, two classes where the processes are linearly ordered, namely, *pipelines* in the sense of Pnueli and Rosner [70] or *one-way chains*, *two-way*

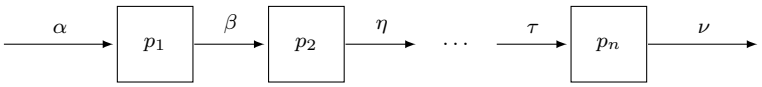


Figure 2.3: An architecture with linearly ordered processes, or pipeline

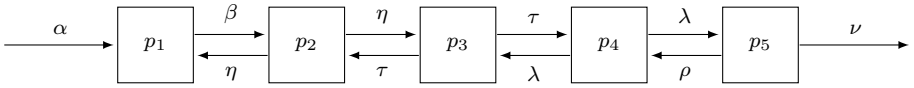


Figure 2.4: A two-way chain with five processes

chains, where the information is allowed to flow back and forth from the process that receives input from the environment, as in Figure 2.4. Second, one other class is presented, where processes are organised in a cyclic way: *one-way rings* architectures, that are similar to one-way chain but where the last process has a communication channel to the first process, as in Figure 2.5. To prove that the synthesis problem for these three classes of architectures is decidable, they rely on *alternating tree automata*. They successively apply *projection* and *re-shaping* transformation operations on alternating tree automata to first consider the possible behaviours of the system according to the most informed process to then prune the possibilities according to the processes below in the hierarchy. The key property of these three classes of architectures is that the environment provides input to only one process in the architecture.

In [62], Mohalik and Walukiewicz take a game-theoretical approach to the distributed synthesis problem by encoding directly instances of the problem in a so-called *distributed game*. Processes are modelled as players that have to cooperate against an hostile environment. While communication channels between processes can be explicitly represented in communication architectures, here players have no explicit means to communicate among themselves, and any such communication has to take place through the environment. Each player has only a *local* view of the global state of the system, hence it has access to its own local history to build its local strategy. In this setting, a *distributed* strategy is then

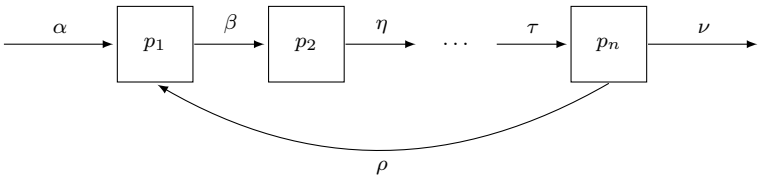


Figure 2.5: A one-way ring

a collection of local strategies, one for each player. On the other hand, the environment has access to the global history of the game. A play is an alternating sequence of moves of the players and of the environment. Solving the distributed synthesis problem for the original architecture and specification amounts to finding a distributed winning strategy for the game. Players need to communicate implicitly while making their moves, which is to compare with our framework (see Section 2.2), where communication between players is also implicit, in the sense that their choice of actions may trigger different observations for their fellow players, who can then deduce information on the global state of the game. However, the game they construct to model the distributed synthesis problem instance allows for *dead-end* in the graph game. Furthermore, it is a sort of *product* game assembled from local games for each player, where environment moves are more liberally designed in order to let implicit communication among processes happen, according to the original architecture communication channel. The authors show two simplification theorems to reduce the number of players and the amount of nondeterminism in a game to obtain an equivalent game, in the sense that there exists a distributed winning strategy in the original game if, and only if, there exists one in the simplified game. The original game is either simplified to the setting of one player against the environment, or to the setting where the environment has no choice of moves, both of which are decidable cases for the distributed synthesis problem. Their technique is sufficient to cover the decidable cases of distributed synthesis mentioned above.

Lastly, in [36], Finkbeiner and Schewe extend Kupferman and Vardi's ([53]) work by providing a criterion for characterising architectures with decidable synthesis problem : the absence of *information forks*. Intuitively, an information fork in an architecture is a situation where the information flow is split up between two processes: as in Figure 2.2 where the environment communicates with two distinct processes. Intuitively, there is a point in the architecture where the order in information levels is lost. Their proof method for the decidability of the synthesis problem for architectures that do not display any information fork uses similar alternating tree automata techniques to Kupferman and Vardi. The undecidability of the synthesis problem for architectures that display an information fork uses a new reduction of the Halting Problem to the synthesis problem. In fact, an information fork can be seen as a sub-architecture of the full communication architecture considered, where two independent processes receive information from the environment and cannot fully deduce the information the other process received.

As we have seen with the approach of [62], games with imperfect information have already been successfully used to model the intricate interactions that one can find in distributed systems. It actually subsumes the model of communication architectures, as we will suggest in Chapter 4. In a few words, the communication between players in games is *implicit* because it is embedded in the graph game structure, while it is *explicit* in architectures, which display communication channels. Furthermore, this implicit communication structure in games allows for more flexibility in design: while in communication architectures, the Environment is an uncontrollable agent, which can send any input to processes, which can read from it, in games, although Nature handles non-determinism, its range can be limited via the game graph structure, allowing for finer and more liberal designs of distributed systems. We choose this game-theoretical approach to investigate

further these interactive systems and their properties. We now turn to the core model used in this work: *synchronous games on finite graphs with imperfect information.*

2.2 Games with Imperfect Information

We consider a set $I = \{1, \dots, n\}$ of players and a distinguished agent called *Nature*. Each player i has her own finite set of *actions* A^i and her own finite set of *observations* B^i. A list of n elements $(x^i)_{i \in I}$, one for each player i, is called a *profile*. The set of all action profiles is $A = \prod_{i \in I} A^i$. The set of all observation profiles is $B = \prod_{i \in I} B^i$.

2.2.1 Game graph

A game graph is a structure $G = (V, E, (\beta^i)_{i \in I}, v_0)$, where:

- V is a finite set of *states*;
- $E \subseteq V \times A \times V$ is an edge relation, representing transitions from one state to another triggered by an action profile;
- $\beta^i : V \to B^i$ is the *observation function* for Player i, mapping every state $v \in V$ to an observation $b \in B^i$;
- $v_0 \in V$ is the *initial state*.

We assume that for every state v and every action profile a, there exists a *successor* $w \in V$ such that $(v, a, w) \in E$.

Example 2.3 (A game graph with two players against Nature)**.** Consider the graph G of Figure 2.6. It is a game graph for a game with two players against Nature, with action sets $A^1 = A^2 = \{L, R\}$. The states carry their name as a label. The initial state is v_0. Edges carry action profiles as labels. For instance, at state v_L, if the first player chooses action L and the second chooses action R, the successor state is v_C. For readability, we do not draw several edges whenever multiple action profiles link the same pair of states, but label one edge with the possible action profiles. For instance, from state v_L, if the players choices of actions yield the profile $R|L$, the successor state is also v_C. Furthermore, when every action choice of one player yield the same successor state, we use the symbol $*$ instead of listing every profile possible, as in the transition from v_C to $\oplus$, where the choice of the second player does not matter: both $L|L$ or $L|R$ yield $\oplus$ as the successor state. Similarly, at state $\oplus$, players enter a loop, regardless of the profile of actions played, hence the self-loop is labelled with $*|*$.

Nature plays a role: whenever there is more than one transition from a state with the same action profile, the successor state is selected by Nature. This is the case at the initial state v_0. In fact, any action profile can lead to v_L or v_R. Recall now the definition of a game graph: in addition to the states and edge relation, a game graph is given with a profile of observation functions, one for each player. The observation function $\beta^i : V \to B^i$

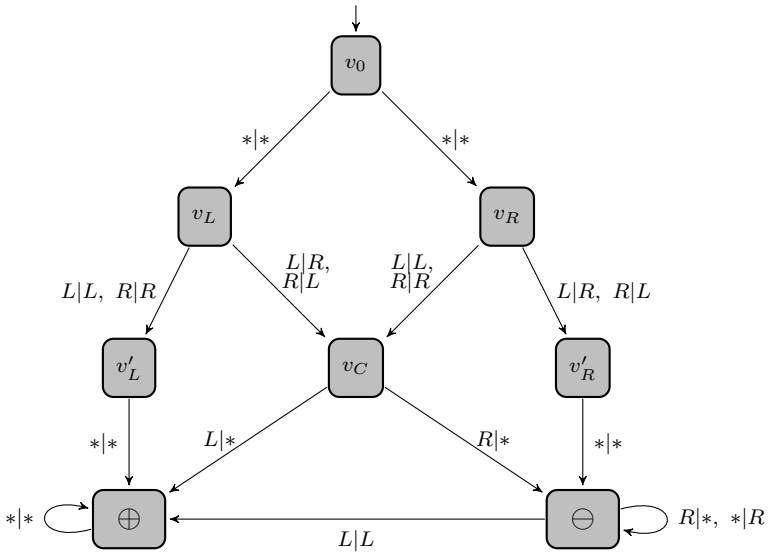


Figure 2.6: Example of a game graph with two players against Nature

describes the (maybe partial) view of Player i on every state $v \in V$. Therefore, even if the observations for the players are not explicitly defined as part of the state names, they are state-attributes, and we represent them attached to the states like in Figure 2.7. On the left side of the name state appears the observation of the first player on this state, and on the right side appears the observation of the second player. The state name component has a grey background as a reminder that it is not observed by the player, unless mentioned otherwise.

The game graph presented in Figure 2.6 corresponds to the case where the observation sets of the players are $B^1 = B^2 = \{\circ\}$, in other words, all states look the same to both players. Their observations are not represented, as they are irrelevant.

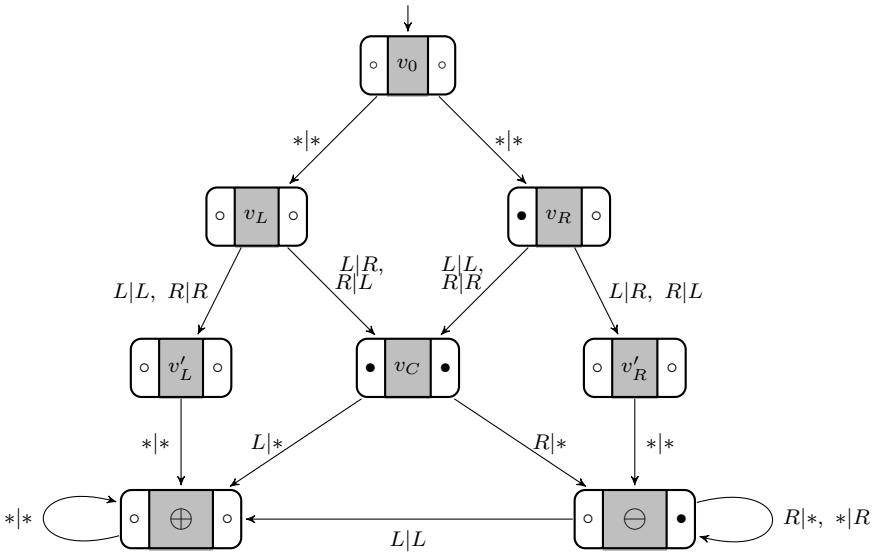


Figure 2.7: Example of a game graph with two players against Nature

2.2.2 Plays and histories

A *play* on a game graph G proceeds as follows: Starting from the initial state v_0, all n players choose simultaneously and independently an action a_1^i from their set A^i. The action profile $a_1 = (a_1^i)_{i \in I}$ yields a set $\{(v_0, a_1, v) \mid (v_0, a_1, v) \in E\}$ of possible edges according to the edge relation E. Among these, Nature picks an edge that determines the next state v. Each player receives a private observation $\beta^i(v)$ of the successor state. The play then continues from v onwards, following the same three steps: the players choose an action profile, Nature picks an edge, and the players receive their observation. We call an iteration of these three steps a *round* of the game, and an edge $(v, a, v') \in E$, composed of a starting state, an action profile and a destination state is called a *move*.

Formally, a play is an infinite[2] sequence $\pi = v_0 v_1 v_2 \cdots \in V^\omega$, such that, for all $\ell \geq 0$, there exists a move $(v_\ell, a, v_{\ell+1}) \in E$. A *history* is a finite prefix $v_0 v_1 v_2 \dots v_\ell \in V^*$ of a play. The number ℓ of rounds played in a history is referred to as its length. The set of all histories in the game graph G is denoted $\mathrm{Hist}(G)$. We extend the observation functions β^i from edges to plays by setting $\beta^i(\pi) = \beta^i(v_0)\beta^i(v_1)\beta^i(v_2)\dots$, and we call $\beta^i(\pi)$ the *observed play of Player i*. We define similarly the *observed histories of Player i* and write $\mathrm{Hist}^i(G) := \{\beta^i(\pi) \mid \pi \in \mathrm{Hist}(G)\}$ for the set of *observation histories* of Player i.

[2] *Finite* plays can also be considered, as in Chapter 3, without much change: it is sufficient to specify the winning condition as regular sets rather than ω-regular sets.

2.2.3 Indistinguishability relation

Two plays π and π' are *indistinguishable* for Player i if $\beta^i(\pi) = \beta^i(\pi')$. In this case, we write $\pi \sim^i \pi'$. Similarly, two histories are indistinguishable if the corresponding observed histories are the same.

Example 2.4 (Playing on G)**.** Consider the graph G of Figure 2.7. The infinite sequences of states $\pi = v_0 v_L v_C \ominus \oplus \dots$ and $\pi' = v_0 v_R v_C \ominus \oplus \dots$ are plays on G. The finite prefixes $\pi_2 = v_0 v_L v_C$ and $\pi'_2 = v_0 v_R v_C$ of π and π' are histories of length 2 . Observed histories are $\beta^1(\pi_2) = \circ \circ \bullet$, $\beta^1(\pi'_2) = \circ \bullet \bullet$ for the first player, $\beta^2(\pi_2) = \circ \circ \bullet = \beta^1(\pi'_2)$ for the second player. The first player can distinguish the two histories, while, for the second player, the observed histories of π_2 and π'_2 are equal: Hence, $\pi_2 \sim^2 \pi'_2$, the two histories are *indistinguishable* for the second player.

2.2.4 Strategies

A strategy for Player i prescribes an action in A^i at every history, such that any two indistinguishable histories yield the same action. Formally, a strategy for Player i is a mapping $s^i : V^* \to A^i$, that is *information-consistent*: for any two histories π, π' such that $\pi \sim^i \pi'$, we have $s^i(\pi) = s^i(\pi')$. Clearly, strategies extend to observed histories as well. The set of all strategies for Player i is called S^i, and S denotes the set of all strategy profiles. A play $v_0 v_1 v_2 \dots$ *follows* a strategy $s^i \in S^i$ if, for every $\ell \geq 0$, it holds that $a^i_{\ell+1} = s^i(v_0 v_1 v_2 \dots v_\ell)$. Similarly, a play π follows a strategy profile $s \in S$ if it follows the strategies s^i of all players. The plays that follow a strategy profile s are called the *outcomes* of s.

As suggested in the previous section, we use *Mealy* automata to describe finite-state strategies.

Definition 2.4 (Mealy automaton)**.** A *strategy automaton*, or Mealy automaton has the form $\mathcal{M} = (M, B, A, m_0, \mu, \nu)$, where:

- M is a finite set of memory states with a designated initial state m_0,
- B is a set of observations,
- A is a set of actions,
- $\mu : M \times B \to M$ is a memory update function,
- $\nu : M \times B \to A$ is an action choice function.

The strategy computed by $\mathcal{M}$ is the function $f_{\mathcal{M}}$ with

$$f_{\mathcal{M}}(q_0, \dots, q_k) = \nu(\mu(m_0, q_0 q_1 \dots q_{k-1}), q_k) \text{ for } k \geq 1.$$

Notice that a Mealy automaton is a *deterministic* automaton, as memory update and action choice are functions. Concretely, a Mealy automaton implements a strategy for a

player by prescribing an action based on the observation received by the player on the last transition in the game graph, and the current state of the memory. The computational power of such a strategy automaton is limited, as the memory set M is finite. This is the kind of strategies we are mostly interested in finding as solutions for distributed synthesis problems.

Definition 2.5 (Finite memory strategies)**.** For a game $\mathcal{G}$, a strategy s^i for a player i is said to be *finite memory*, or *forgetful*, if there exists a Mealy automaton that implements s^i. A strategy profile s is *finite memory* if all its elements are finite-memory.

We distinguish the particular case of memoryless strategies:

Definition 2.6 (Memoryless strategies)**.** A strategy s for a player i in a game $\mathcal{G}$ is *memoryless*, or *positional*, if there exists a function $f : V \to A^i$ that implements s: for any history $v_0, \ldots, v_\ell$ of the game, $s(v_0, \ldots, v_\ell) = f(v_\ell)$.

Informally, a strategy is memoryless if it relies only on the last observation received to prescribe the next action and thus, there exists a Mealy automaton that implements σ such that its memory states set M is a singleton.

2.2.5 Winning conditions

A *winning condition* on a game graph G is a set $W \subseteq V^\omega$ of plays on G. We first give three examples of winning conditions.

- A *safety* condition is specified by a set $F \subseteq V$ of *safe* states. A play is winning if all states occurring in it belong to F.

- A *reachability* condition is specified by a set $F \subseteq V$ of *target* states. A play is winning if there exists a state in F that appears in it.

- A *parity* condition is specified by a *priority* function $\Omega : V \to \mathbb{N}$. A play π is winning if $(\min \{\Omega(v) \mid v \in \mathrm{Inf}(\pi)\}) \equiv 0 \mod 2$, that is, if the minimal priority occurring infinitely often is even.

Finally, a *game* $\mathcal{G} = (G, W)$ consists of a game graph G and a winning condition W. A play π is winning if $\pi \in W$. A strategy profile S is winning if all plays following S yield plays that are winning. We say that a game is *solvable*, if there exists a winning strategy profile.

Example 2.5 (Winning a game)**.** We consider the game $\mathcal{G}_1 = (G, W_1)$ consisting of the game graph G of Figure 2.7 and the reachability winning condition W_1 with the target set $F = \{\oplus\}$. The plays $\pi = v_0 v_L v_C \ominus \oplus \dots$ and $\pi' = v_0 v_R v_C \ominus \oplus \dots$ are both winning plays. They are both outcomes of the strategy profile $s = (s^1, s^2)$ such that, for all $v \in V$,

$$s^1(v) = \begin{cases} L & \text{if } \beta^1(v) = \circ \\ R & \text{if } \beta^1(v) = \bullet \end{cases}$$

and,

$$s^2(v) = \begin{cases} L & \text{if } \beta^2(v) = \bullet \\ R & \text{if } \beta^2(v) = \circ \end{cases}$$

Notice that s is a positional, or memoryless, strategy profile: for each player, the action prescribed by their strategy depends only on the observation on the current state. The strategy profile s is moreover winning: π and π' are the only outcomes of s, and they are winning plays.
We consider now the game $\mathcal{G}_2 = (G, W_2)$ consisting of the game graph G of Figure 2.7 and the safety winning condition W_1 with the safe set $F = V \setminus \{v_C\}$. The plays $\pi = v_0 v_L v'_L \oplus \dots$ and $\pi' = v_0 v_R v'_R \ominus \oplus \ \dots$ are both winning plays. They are both outcomes of the memoryless strategy profile $s = (s^1, s^2)$ such that, for all $v \in V$,

$$s^1(v) = \begin{cases} L & \text{if } \beta^1(v) = \circ \\ R & \text{if } \beta^1(v) = \bullet \end{cases}$$

and,

$$s^2(v) = L$$

The strategy profile s is moreover winning: π and π' are the only outcomes of s, and they are winning plays.

2.3 Background

Throughout this work, we are interested in the two flavours of the distributed synthesis problem for games with imperfect information:

1. *Solvability:* For a given game, does there exist a joint winning strategy?

2. *Implementation:* Construct a winning strategy, if any exists.

In this section, we present the background notions and results that will be useful in the three following chapters.

2.3.1 The case of perfect information

The study of interaction mechanisms first considered a setting where information is *perfect*, meaning that players always know the current state of the game.[3] In fact, perfect information games form a special case of games with imperfect information that is easier to handle. They are well studied in the literature and we present here the key results, that involve notions that are also relevant to the imperfect information case. In the remaining of this section, a game $\mathcal{G}$ is a game between two players with perfect information.

The classical notion of *determinacy* is significant:

Definition 2.7 (Determinacy)**.** We say that a class $\mathcal{C}$ of two-player games is *determined* if, for each game $\mathcal{G} \in \mathcal{C}$, either the first or the second player has a winning strategy.

Notice that the definition is stated for a setting of two adversarial players, and so will the remaining results. As we consider in our model a *coalition* of players that cooperate against Nature, one can see this two-player with perfect information setting as the case of one player against Nature, where the player has perfect information about the play, namely, about the edge Nature chooses at each step. However, we do not consider Nature to be intrinsically hostile to the player, but rather to be a *passive* player, and the player needs to achieve his goal against all possible behaviours of this passive player. In this sense, it is clear that, even if we do not acknowledge the notion of strategy of Nature, nor try to construct any, determinacy is relevant as well in this context. Indeed, if a class of two active player games with perfect information is determined, then its one-player against Nature variant is also determined, in the sense that either the first player has a winning strategy against the second player, and thus this strategy is also winning against Nature, or the second player has a winning strategy, and thus the first player cannot prevent the case of Nature playing according to this strategy, therefore he has no winning strategy. For more considerations on the role of Nature as a passive player, see for instance the reflexion of Walliser in [87].

In the previous section, we presented the general model used in this thesis: synchronous games with imperfect information and simultaneous moves. However, simultaneous moves intrinsically generate imperfect information, in the sense that, after a move, a player

[3] Action profiles are not considered as part of the play: as the players are assumed to cooperate against Nature towards a common objective, there is no need to monitor the actions specifically. Things change when considering non purely cooperative players, see Chapter 5.

may not be able, upon receiving his observation on the new state, to distinguish the exact influence of every other player on this move, as different action profiles may trigger the same transition. Therefore, it seems that it only makes sense to speak of perfect information for games that are *turn-based*, that is, games where players choose their actions one after the other, so that the effect of an individual action is not hidden by other simultaneous actions. As we said earlier, the games we consider in this section correspond in fact to games for one player against Nature. In this case, there is no concern about the concurrency of the actions choices, as Nature is understood as the non-determinism resolver: it chooses a transition among the set of candidates that are labelled by a complete action profile. In this sense, a game in our model for one player against Nature can be seen as a turn-based game.

As a first result, we consider the reachability condition:

Theorem 2.1. *The class of reachability games is determined.*

A way to show the determinacy of reachability games of perfect information is to use the *Attractor construction*. The idea is to compute the winning regions of the players inductively: Starting from the target set of the first player as his attractor, one adds to it the states that allows to reach the target set in one round. Considering the set obtained, one continues to add states that allow to reach the current attractor in one round, and so on until no state can be added anymore. From any state in this final attractor set, the first player can force the successor state to stay in this set, and to eventually reach his target set. The set computed then corresponds to the winning region of the first player, and its complement to the winning region of the second player. For details on the construction, see [44]. Furthermore, from the attractor's construction, one can easily extract a memoryless winning strategy, as every iteration of the construction requires the attractor to be reachable in one round. Thus, reachability games are *memoryless determined*, that is, the winning player has a memoryless winning strategy.

Notice that the safety condition is *dual* to the reachability condition in the following sense: Consider a safety game $\mathcal{G}$ with the safe set for player 1 being $F \subseteq V$. Consider now the reachability game $\mathcal{G}'$ played on the same graph as $\mathcal{G}$ but with the target set for player 1 being $V \setminus F$, and where the roles of the player are switched. Player 1 has a winning strategy in $\mathcal{G}$ if, and only if, player 2 has a winning strategy in $\mathcal{G}'$: Clearly, if player 1 can maintain the play in F in $\mathcal{G}$, then he cannot force the play to go outside F, that is, to $V \setminus F$ in $\mathcal{G}'$, where he has the same actions available as player 2 in $\mathcal{G}$. Thus, the determinacy of safety games follows from the determinacy of parity games.

We now turn to the remaining ω-regular winning conditions. Many determinacy results for classes of games rely on proofs that reduce the considered winning condition to a parity condition, for which the determinacy is easy to obtain. We list here the results that lead to determinacy for games with ω-regular winning conditions.

Recall the definition of a parity winning condition: from a colouring function of finite range c on the states of the game graph, we restrict winning plays to the ones where the minimum colour number among the states appearing infinitely often in the play is even. Another important winning condition that uses a colouring of the states is the

Muller condition. It corresponds to the accepting condition of *Muller* automata presented earlier:

Definition 2.8 (Muller winning condition)**.** A *Muller* winning condition for a game $\mathcal{G}$ equipped with a colouring function $c : V \to C$, where $C \subseteq \mathbb{N}$, is specified by a subset $\mathcal{F} \subseteq \mathcal{P}(C)$. A play π in $\mathcal{G}$ is then winning if $\mathrm{Inf}(c(\pi)) \in \mathcal{F}$.

It turns out that Muller winning conditions subsume all ω-regular winning conditions, in the following sense:

Theorem 2.2 (Zielonka [90])**.** *For any game $\mathcal{G}$ with an ω-regular winning condition, there exists a Muller game $\mathcal{G}'$ on the same graph and with the same colouring function such that Player i has a winning strategy in $\mathcal{G}$ if, and only if, he has a winning strategy in $\mathcal{G}'$.*[4]

Furthermore, parity winning conditions subsume Muller winning conditions:

Proposition 2.1. *For every Muller game $\mathcal{G}$ with set of states V there exists a parity game $\mathcal{G}'$ with set of states V' and a function $r : V \to V'$ such that Player i has a winning strategy in $\mathcal{G}$ if, and only if, he has a winning strategy in $\mathcal{G}'$.*

A key tool to handle Muller games is the *Latest Appearance Record* technique, *LAR* for short, introduced by McNaughton in [59]: Intuitively, the idea is to keep track of the states visited in a play, as well as the order in which they were last visited, by maintaining an ordered list of states and an index to record the last permutation in the order, to account for the Muller winning condition that describe the possible *sets* of infinitely often occurring colours that specify winning plays.

Now that we know that any game with an ω-regular winning condition can be reduced to an equivalent game with a parity condition, it remains to show that the class of parity games is determined. This is a classic result that can be found in several places in the literature. For instance, Emerson and Jutla, in [34] presented a proof involving the techniques of μ-calculus. In [63], Mostowski also obtains the result of parity games determinacy. Finally, Zielonka shows, in [90], that, by effectively computing, in a similar manner than the Attractor construction used to show determinacy of reachability games, the winning regions of the players and showing they partition the state space, memoryless determinacy of parity games.

Theorem 2.3 ([90], [34], [63])**.** *The class of parity games is memoryless determined.*

With the Theorem 2.2 and Proposition 2.1, we obtain:

Corollary 2.1. *The class of ω-regular games is determined with finite memory.*

Remark. In this work, we focus on ω-regular winning conditions and we have seen that games with perfect information are determined in this case. In his paper [58] of 1975, Martin showed that games with perfect information and *Borel* winning conditions are

[4]Note that the winning strategy for $\mathcal{G}$ and the winning strategy for$\mathcal{G}'$ are most likely quite different. The theorem states that the existence of winning strategies for $\mathcal{G}$ and $\mathcal{G}'$ are equivalent, but does not specify how to get one from the other, if at all possible.

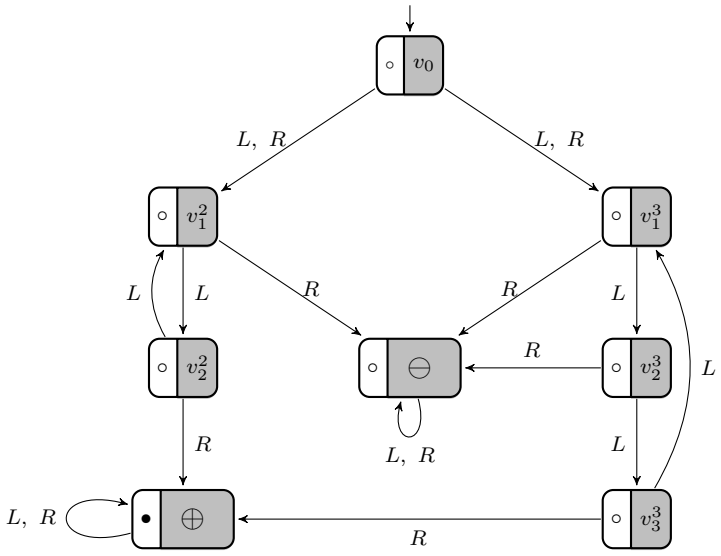


Figure 2.8: Example of a game graph with one player against Nature

determined. As all ω-regular winning conditions are Borel sets, the determinacy result for games with ω-regular winning conditions can also be seen as a consequence of Martin's result. Furthermore, by relying on the axiom of choice, thus in a non-constructive manner, to consider non-Borel sets as a winning condition, it is possible to design games with perfect information that are not determined, as shown by Gale and Stewart in [39].

2.3.2 The case of one player against Nature

Let us now turn to the setting with imperfect information. The case of one player against Nature has to be dissociated from the multiple player setting: indeed, even with imperfect information, the problem of deciding if the player has a winning strategy in a game is decidable, while the existence of a distributed winning strategy is undecidable, as we will see in the next section.

Example 2.6 (Counting modulo 6 game)**.** Consider the game graph of Figure 2.8 together with a reachability winning condition and target set $F = \{\oplus\}$. It is a game for one player against Nature with imperfect information: for instance, at the initial state v_0, the player, whether he chooses action L or R, lets Nature choose between v_1^2 and v_1^3, and receives the observation $\circ$ in both cases. The set of winning strategies clearly consists of the strategies that prescribe the player to play L for a number of rounds that is equal to a common multiple of 2 and 3 after the first move. Indeed, the game graph is designed so that after the first move, the player does not know if the current state is in the 2-cycle formed by the states $\{v_1^2, v_2^2\}$ or in the 3-cycle formed by the states $\{v_1^3, v_3^3, v_3^3\}$. Since the only way to reach $\oplus$ is to play R at v_2^2 or v_3^3, and that playing R elsewhere in the cycles leads to the sink $\ominus$, the player has to ensure the current state is v_2^2 or v_3^3 to play R and win the game, thus, wait, by playing L for any multiple of 6 after the

first round, before playing R. Let $\mathcal{M} = (M, B, A, m_0, \mu, \nu)$ be the Mealy automaton with memory set $M = \{m_0, m_1, m_2, m_3, m_4, m_5, m_6\}$, observation set $B = \{\circ, \bullet\}$, action set $A = \{L, R\}$ and such that: $\mu(m_0, \bullet) = m_0$ and $\mu(m_0, \circ) = m_1$, for $1 \leq i \leq 5$, $\mu(m_i, \circ) = \mu(m_i, \bullet) = m_{i+1}$, and $\mu(m_6, \circ) = \mu(m_6, \bullet) = m_0$; $\nu(m_0, \bullet) = \nu(m_0, \circ) = L$, for $1 \leq i \leq 5$, $\nu(m_i, \circ) = \nu(m_i, \bullet) = L$, and $\nu(m_6, \circ) = \mu(m_6, \bullet) = R$. $\mathcal{M}$ implements the finite memory winning strategy s that makes the player play L for seven rounds before playing R and reach $\oplus$, and then play L forever.

We have seen a game with imperfect information for one player against Nature for which it is easy to see that the synthesis problem has a positive answer. In fact, in 1984, in [73], Reif has studied and solved the synthesis problem for the case of one player:

Theorem 2.4 ([73])**.** *The problem of deciding whether the player in a game with imperfect information has a winning strategy is* ExpTime*-hard.*

The approach of Reif consists of eliminating imperfect information by constructing an equivalent game with perfect information, for which the synthesis problem is decidable, as games with perfect information are determined. Informally, the game with perfect information is constructed from the original game with imperfect information in a way that resembles the powerset construction used to determinise finite automata: The states of the game with perfect information consist of the subsets of states that the player cannot distinguish in the original game, that is, his *information sets*.

Example 2.7 (Knowledge-set construction)**.** We illustrate the knowledge-set construction idea on the Counting game example of Figure 2.8. The associated knowledge game of perfect information is depicted in Figure 2.9: The states of the graph game correspond to set of states of the original game, according to the indistinguishability relation defined by the observations distribution: for instance, after the first move in the Counting game, the player does not know if the state is v_1^2 or v_1^3, since the observation he receives is $\circ$ in either case. Therefore, in the knowledge game, the information set $\{v_1^1, v_1^3\}$ becomes the only successor state of the initial state v_0. The whole knowledge game is constructed in this manner: there exists a transition (C, a, C') in the knowledge game if, and only if, in the original game, for every state $q' \in C'$, there exists a state $q \in C$ such that (q, a, q') is a transition and all states of C' yield the same observation. The observation function in the resulting game is the identity function. (This is represented in the picture by the fact that the state names are written on a white background.) Thus, the resulting game has perfect information. Furthermore, the player has a winning strategy in the original game if, and only if, he has a winning strategy in the knowledge game. For instance, consider the strategy s^K for the knowledge game such that

$$s(q) = \begin{cases} R & \text{if } q = \{v_2^2, v_3^3\} \\ L & \text{otherwise} \end{cases}$$

Notice that s^K is a memoryless winning strategy, as it can be implemented by the Mealy automaton $\mathcal{M}^K = (\{m_0\}, B^K, A, m_0, \mu^K, \nu^K)$ such that its observation set is equal to the set of states of the knowledge game. The strategy s^K actually corresponds to the winning strategy s constructed in Example 2.6, and, as we have hinted in Example 2.6, winning

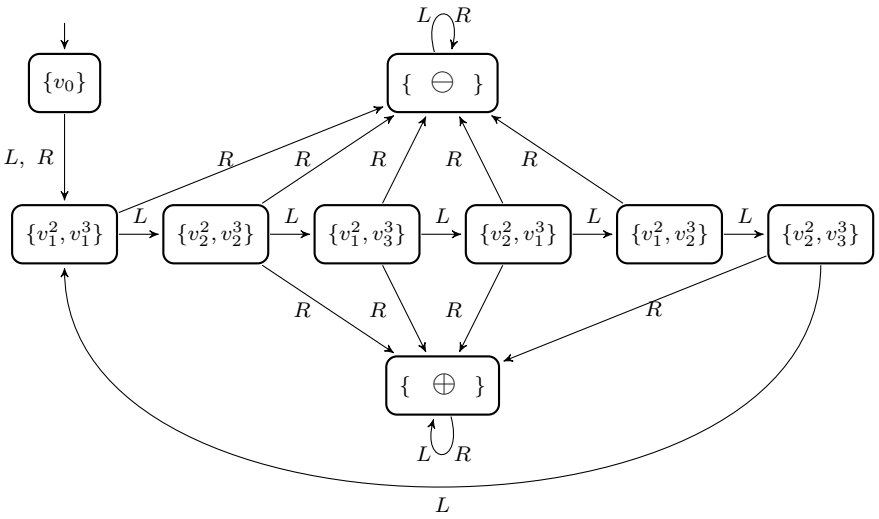


Figure 2.9: Perfect information game obtained by the knowledge-set construction

strategies in the original game require memory, to count the number of rounds after the initial one.

2.3.3 Distributed games

As we already pointed out, involving more than one player in a game with imperfect information increases greatly the complexity of the decision problem. Actually, the problem turns out to be undecidable, and even when it is decidable, the construction problem is far more complex.

In [69], Peterson and Reif[5] establish the fundamental results about games with imperfect information and several players. They present a classification of these games in regards of their specific rules and a correspondence with computing models. They generalise the *alternation* machines of Chandra, Kozen and Stockmeyer [20], to *multiple person alternation* machines to model interaction among several agents. In our setting, alternation machines correspond to games of one player against Nature, while multiple person alternation machines correspond to games with more than two players against Nature. They show that for a two player (against Nature) game with reachability winning condition, the question whether there exists a joint winning strategy for the players is undecidable. They also identified a class of multiplayer games for which this question is decidable, although, with non-elementary complexity: *hierarchical games*. Informally, hierarchical games are a special case of multiplayer games with imperfect information where players are ordered so that a player higher in the order has access to all the information available to the lower-ranked players. Furthermore, they present an effective procedure to construct a joint finite-state winning strategy, whenever one exists. The decidability result was later extended by Pnueli and Rosner [70] for pipelines in the distributed systems framework, as we have seen earlier.

In general, the question whether there exists a winning strategy for the players in a safety game with imperfect information is undecidable:

Theorem 2.5 ([69], [70])**.** *The solvability problem for a game of two players with imperfect information against Nature and a safety winning condition is undecidable.*

The proof relies on a reduction of the Halting Problem so that, for a given Turing Machine, one constructs a game for two players against Nature, along with an ω-regular winning condition, that has a joint winning strategy if, and only if, the machine halts on the empty input.

We do not go into further details here as we give an alternative proof of this undecidability result in Chapter 3, using the tools of formal languages.

Even when the game admits a joint winning strategy for the players, it may be that this winning strategy cannot be implemented with finite memory:

Proposition 2.2 ([5])**.** *There are finite safety games for two players against Nature that require infinite memory.*

Remark. In fact, Janin, in [47], even showed that there exist two-player safety games that admit a winning strategy, but none that can be implemented by a Turing machine.

[5] for a more recent account, see also [1]

Finally, in [5], a semi-decision procedure for the solvability of distributed games is proposed. It relies on the *information tracking* construction that extends the knowledge-set construction mentioned earlier to the distributed case by transforming a game with several players coordinating against Nature to a two-player game with perfect information, while preserving winning strategy profiles. This tracking construction is based on the unravelling of a game with imperfect-information as a tree with epistemic models as nodes, that represent the knowledge of the players at every state of the game. As with the knowledge-set construction, this yields a game of perfect information, however on an infinite tree in this case. They show how to obtain a finite-graph representation via an abstraction technique relying on homomorphic equivalence of epistemic models, that is sound for games with observable ω-regular winning conditions.

Chapter 3

Consensus Condition

In this chapter, we propose a game model of a language acceptor based on coordination games between two players with imperfect information. Compared to the model of Peterson and Reif [69], similar to the model of games presented in Chapter 2, our setting is utterly simple: the games are played on a finite graph, plays are of finite duration, they involve only one yes-or-no decision, and the players have no means to communicate. Moreover, they have to take their decisions in consensus. Given an input word that may yield different observations to each of the players, they have to settle simultaneously and independently on a common decision, otherwise they lose.

Without the restrictions to consensus and to a single decision per play, the distributed synthesis problem, as stated in Section 2.3, for coordination games with safety winning conditions is known to be undecidable [69, 70]. Furthermore, Janin [47] points out that there exist two-player safety games that admit a winning strategy, but none that can be implemented by a Turing machine.

Here, we analyse how the information gathered by a single player in a two-player setting impacts the unfolding of a play. In our model, private observations for the players are attached to the states of the game, or positions of the graph. Therefore, the graph structure of a game can be seen as a correlation graph for observations and observation sequences of players. Furthermore, in this particular case, we put aside some other general aspects of interaction by considering games that require only one action (or decision) of the players before the play is either won or lost. By letting Nature be responsible for the entire path in the graph, or the succession of global states, we can better identify the influence of the information a player receives on the common decision process.

Our first main result establishes a correspondence between context-sensitive languages and consensus games: We prove that, for every context-sensitive language L, there exists a solvable consensus game in which every winning strategy extends the characteristic function of L, and conversely, that every solvable consensus game admits a winning strategy characterised by a context-sensitive language. As a second result, we characterise winning strategies for consensus games in terms of iterated transductions of the (synchronous rational) relation between the observations of players. This allows us to identify a subclass of games that corresponds to context-free languages. Although it is still undecidable whether a game of the class admits a winning strategy, we can effectively construct optimal strategies implemented by push-down automata. The results provide insight on the

inherent complexity of coordination in games with imperfect information. With regard to the basic problem of agreement on a simultaneous action, they substantiate the assertion that 'optimality requires computing common knowledge" put forward by Dwork and Moses in their analysis of Byzantine agreement in distributed systems [32]. Indeed, the constraints induced by our acceptor model can be reproduced in virtually any kind of games with imperfect information and plays of unbounded length, with the consequence that implementing optimal strategies amounts to deciding the transitive closure of the transduction induced by the game graph.

3.1 Consensus Game Acceptors

Consensus acceptors are games between two cooperating players, 1 and 2, and a passive agent called Nature. Given a finite *observation alphabet* Γ common to both players, a *consensus game acceptor* $G = (V, E, (\beta^1, \beta^2), v_0, \Omega)$ is described by a finite set V of *states*, a *transition* relation $E \subseteq V \times V$, and a pair of *observation* functions $\beta^i : V \to \Gamma$ that label every state with an observation for each player $i = 1, 2$. There is a distinguished *initial* state $v_0 \in V$ with no incoming transition. States with no outgoing transitions are called *final* states; the admissibility condition $\Omega : V \to \mathcal{P}(\{0, 1\})$ maps every final state $v \in V$ to a nonempty subset of admissible decisions $\Omega(v) \subseteq \{0, 1\}$. The observations at the initial and the final states do not matter, we assume that they correspond to a special symbol $\# \in \Gamma$ for both players.

The game is played as follows: Nature chooses a finite path $\pi = v_0\, v_1 \ldots v_{n+1}$ in G from the initial state v_0, following transitions $(v_\ell, v_{\ell+1}) \in E$, for all $\ell \leq n$, to a final state v_{n+1}. Then, each player i receives a private sequence of observations $\beta^i(\pi) := \beta^i(v_1)\, \beta^i(v_2) \ldots \beta^i(v_n)$ and is asked to take a *decision* $a^i \in \{0, 1\}$, independently and simultaneously. The players win if they agree on an admissible decision, that is, $a^1 = a^2 \in \Omega(v_{n+1})$; otherwise they lose. Without risk of confusion we sometimes write $\Omega(\pi)$ for $\Omega(v_{n+1})$.

We say that two plays π, π' are *indistinguishable* to Player i, and write $\pi \sim^i \pi'$, if $\beta^i(\pi) = \beta^i(\pi')$. This is an equivalence relation, and its classes, called the *information sets* of Player i, correspond to observation sequences $\beta^i(\pi)$. A *strategy* for Player i is a mapping $s^i : V^* \to \{0, 1\}$ from plays π to decisions $s^i(\pi) \in \{0, 1\}$ such that $s^i(\pi) = s^i(\pi')$, for any pair $\pi \sim^i \pi'$ of indistinguishable plays. A joint strategy is a pair $s = (s^1, s^2)$; it is *winning*, if $s^1(\pi) = s^2(\pi) \in \Omega(\pi)$ for all plays π. In this case, the components s^1 and s^2 are equal, and we use the term *winning strategy* to refer to the joint strategy or either of its components. Finally, a game is *solvable*, if there exists a (joint) winning strategy.

Compared to the game model we introduced in the preliminaries, consensus game acceptors seem quite different but are actually only a restriction of our initial model. Indeed, for clarity purposes, we say here that the players have only one decision to take, at final states, and then the play is over, while before that, they passively observe the input Nature gives them. To make it correspond to our game model, which requires actions of the players at each step and an infinite duration of a play, it is sufficient to proceed to slight changes in the description of consensus game acceptors. In order to comply with the first requirement, one can label every transition between two states with all possible action profiles, therefore considering players to have acted, while in fact keeping the passive

character of this phase, since Nature is still the only responsible of any choice of direction in the game graph. For the infinite duration, it is simply enough to consider the winning and losing states to have self-loops, labelled, once again, with all possible action profiles.

Whenever we refer to games in the following, we mean consensus game acceptors.

Strategies and knowledge. With the consensus condition in mind, one can see that winning strategies cannot be constructed for each player relying solely on their own indistinguishability relation. Indeed, if player i can distinguish between two plays π and π', but player j cannot, it would be unwise for a strategy to prescribe to player i different decisions after π and π', since, as we stated above, it is necessary for a (joint) strategy to be winning that the two individual strategies coincide on all plays π. Therefore, we need to take into account cases where the uncertainty on plays propagates back and forth between the players. We say that two plays π and π' are *connected*, and write $\pi \sim^* \pi'$, if there exists a sequence of plays $\pi_1, \ldots, \pi_k$ such that

$$\pi \sim^1 \pi_1 \sim^2 \cdots \sim^1 \pi_k \sim^2 \pi'.$$

Then, a mapping $f : V^* \to \{0,1\}$ from plays to decisions is a strategy that satisfies the consensus condition if, and only if, $f(\pi) = f(\pi')$, for all $\pi \sim^* \pi'$. In terms of distributed knowledge, this means that the decisions have to be based on events that are common knowledge among the players at every play. (For an introduction to knowledge in distributed systems, see Chapters 4 – 6 in the book of Fagin, Halpern, Moses, and Vardi [35].) Such a consensus strategy — or, more precisely, the pair (f, f)— may still fail, due to prescribing inadmissible decisions. We say that a decision $a \in \{0,1\}$ is *safe* at a play π if $a \in \Omega(\pi')$ for all $\pi' \sim^* \pi$. Then, a consensus strategy f is winning if, and only if, it prescribes a safe decision $f(\pi)$ for every play π.

It is sometimes convenient to represent a strategy for Player i as a function $f^i : \Gamma^* \to \{0,1\}$. Every such function describes a valid strategy, because observation sequences identify information sets; we refer to an *observation-based* strategy in contrast to the *state-based* representation $s^i : V^* \to \{0,1\}$. Note that the components of a joint winning strategy need no longer be equal in the observation-based representation. However, once the strategy for one player is fixed, the strategy of the other player is determined by the consensus condition, so there is no risk of confusion in speaking of a winning strategy rather than a joint strategy pair.

As an example, consider the game depicted in Figure 3.1, with observation alphabet $\Gamma = \{a, b, \triangleleft, \triangleright, \square\}$. States v at which the two players receive different observations are split, with $\beta^1(v)$ written in the upper part and $\beta^2(v)$ in the lower part; states at which the players receive the same observation carry only one symbol. The admissible decisions at final states are indicated on the outgoing arrow. Notice that upon receiving the observation sequence a^2b^2, for instance, the first player is constrained to choose decision 1, due to the following sequence of indistinguishable plays that leads to a play where deciding 0 is not admissible.

$$\begin{pmatrix} a,a \\ a,\triangleleft \\ b,\triangleright \\ b,b \end{pmatrix} \sim^2 \begin{pmatrix} a,a \\ \triangleleft,\triangleleft \\ \triangleright,\triangleright \\ b,b \end{pmatrix} \sim^1 \begin{pmatrix} a,\triangleleft \\ \triangleleft,\triangleright \\ \triangleright,\triangleleft \\ b,\triangleright \end{pmatrix} \sim^2 \begin{pmatrix} \triangleleft,\triangleleft \\ \triangleright,\triangleright \\ \triangleleft,\triangleleft \\ \triangleright,\triangleright \end{pmatrix} \sim^1 \begin{pmatrix} \triangleleft,\square \\ \triangleright,\square \\ \triangleleft,\square \\ \triangleright,\square \end{pmatrix} \sim^2 \begin{pmatrix} \square,\square \\ \square,\square \\ \square,\square \\ \square,\square \end{pmatrix}$$

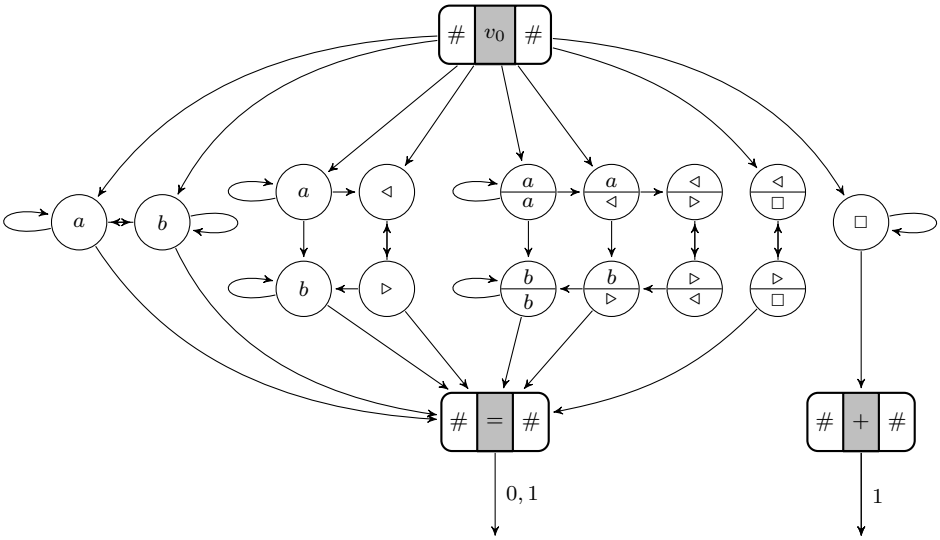


Figure 3.1: A consensus game acceptor

In contrast, decision 0 may be safe when Player 1 receives input a^3b^2, for instance. Actually, the strategy $s^1(w)$ that prescribes 1 if, and only if, $w \in \{a^n b^n \mid n \in \mathbb{N}\}$ determines a joint winning strategy. Next, we shall make the relation between games and languages more precise.

Remark. The representation scheme of Figure 3.1 slightly differs from the one presented in Chapter 2: While the initial and final states keep the same format, we omitted the state names on the remaining states for readability and because the state names here are irrelevant. Furthermore, the observations on these states are organised vertically rather than horizontally, with the observation of the first player on the top part of the state, and the observation of the second player on the bottom part: this is to visually highlight the correspondence with domino tilings rows, as we will see in Section 3.2.2.

3.2 Describing Languages by Games

We use the characterisation of context-sensitive languages in terms of nondeterministic linear-bounded automata given by Kuroda [54], and the following well-known results from the same article: (1) For a fixed context-sensitive language L over an alphabet Σ, the problem whether a given word $w \in \Sigma^*$ belongs to L is PSPACE-hard. (2) The problem of determining whether a given context-sensitive language represented by a linear-bounded automaton contains any non-empty word is undecidable. For background on formal languages, see for instance [76].

We consider languages L over a terminal alphabet Σ. The empty word ε is excluded from the language, and also from its complement $\bar{L} := (\Sigma^* \setminus \{\varepsilon\}) \setminus L$. As acceptors for such languages, we consider games over an observation alphabet $\Gamma \supseteq \Sigma$, and we assume that no observation sequence in Σ^+ is omitted: for every word $w \in \Sigma^+$, and each player i, there exists a play π that yields the observation sequence $\beta^i(\pi) = w$.

Given a consensus game acceptor G, we associate to every observation-based strategy $s \in S^1$ of the first player, the language $L(s) := \{w \in \Sigma^* \mid s(w) = 1\}$. We say that the game G *covers* a language $L \subseteq \Sigma^*$, if G is solvable and

- $L = L(s)$, for *some* winning strategy $s \in S^1$, and
- $L \subseteq L(s)$, for *every* winning strategy $s \in S^1$.

If, moreover, $L = L(s)$ for *every* winning strategy in G, we say that G *characterises* L. In this case, all winning strategies map L to 1 and $\bar{L}$ to 0.

Notice that every solvable game covers a unique language L over the full observation alphabet Γ. With respect to a given terminal alphabet $\Sigma \subseteq \Gamma$, the covered language is hence $L \cap \Sigma^*$. For instance, the consensus game acceptor represented in Figure 3.1 covers the language $\{a^n b^n \mid n \in \mathbb{N}\}$ over $\{a, b\}$. To characterise a language rather than covering it, we need to add constraints that require to reject inputs.

Given two games G, G', we define the *union* $G \cup G'$ as the consensus game obtained by taking the disjoint union of G and G' and identifying the initial states. Then, winning strategies of the component games can be turned into winning strategies of the composite game, if they agree on the observation sequences over the common alphabet.

Lemma 3.1. *Let G, G' be two consensus games over observation alphabets Γ, Γ'. Then, an observation-based strategy r is winning in $G \cup G'$ if, and only if, there exist observation-based winning strategies s, s' in G, G' that agree with r on Γ^* and on Γ'^*, respectively.*

Proof. $\Longrightarrow$: Let r be a winning observation-based strategy in $G \cup G'$. Consider the function $s : \Gamma^* \to \{0, 1\}$ such that $s(w) = r(w)$ for all $w \in \Gamma^*$. As Γ^* is included in the domain of r, and r is observation-based, s is an observation-based strategy in G. Furthermore, r is winning in $G \cup G'$, so $r(w)$ is safe in $G \cup G'$ for, in particular, every word $w \in \Gamma^*$. Assume that there exists $w \in \Gamma^*$ such that the decision $s(w) = r(w)$ is not safe in G. It then means that there exists $w' \in \Gamma^*$ such that $\Omega(w') = 1 - s(w)$ and $w \sim^* w'$. That implies that $s(w)$ is not safe in $G \cup G'$ either, since $G \cup G'$ is the union of G and G'. Since $s(w) = r(w)$, that contradicts the fact that r is a winning strategy in $G \cup G'$, therefore $s(w)$ is safe in G and s is an observation-based winning strategy in G. Similarly, the strategy $s' : \Gamma'^* \to \{0, 1\}$ such that $s(w) = r(w)$ for all $w \in \Gamma'^*$ is an observation-based winning strategy in G'.

$\Longleftarrow$: Let r be an observation-based strategy in $G \cup G'$ and let s and s' be observation-based winning strategies in G and G' that agree with r on Γ^* and on Γ'^*, respectively. Let $w \in \Gamma^* \cup \Gamma'^*$. If $w \in \Gamma^* \cap \Gamma'^*$, then $r(w) = s(w) = s'(w)$, since s and s' agree with r on Γ^* and on Γ'^*, respectively. Furthermore, as s ans' are winning strategies in G and G', $r(w)$ is a safe decision in G and G'. As $G \cup G'$ is the union of G and G', the decision $r(w)$ is also safe in $G \cup G'$. Now if $w \in \Gamma^* \setminus \Gamma'^*$, then $r(w) = s(w)$, which is a safe decision in G. If there exists $w' \in \Gamma^* \cap \Gamma'^*$ such that $w \sim^* w'$, then $s(w') = s(w)$ and is also a safe decision in G, since s is a winning strategy in G. As s' is a winning strategy in G' and agrees with r on Γ'^*, the decision $s'(w')$ is equal to $r(w')$ and is safe in G'. The decision $r(w')$ is then safe in $G \cup G'$, and therefore $r(w)$ is safe in $G \cup G'$. Similarly, if $w \in \Gamma'^* \setminus \Gamma^*$, the decision $r(w)$ is safe in $G \cup G'$. Thus, the strategy r is an observation-based winning strategy in $G \cup G'$.

□

Whenever a language and its complement are covered by two consensus games, we can construct a new game that characterises the language. The construction involves *inverting* the decisions in a game, that is, replacing the admissible decisions for every final state $v \in V$ with $\Omega(v) = \{0\}$ by $\Omega(v) := \{1\}$ and vice versa; final states v with $\Omega(v) = \{0, 1\}$ remain unchanged.

Lemma 3.2. *Suppose two consensus games G, G' cover a language $L \subseteq \Sigma^*$ and its complement $\bar{L}$, respectively. Let G'' be the game obtained from G' by inverting the admissible decisions. Then, the game $G \cup G''$ characterises L.*

Proof. First, let us notice that no sequence of observations in Σ^* is omitted in G'', since G covers the language $L \subseteq \Sigma^*$. Consider now a winning strategy s in G such that $L(s) = L$. We argue that s is also a winning strategy in $G \cup G''$. Let w be a word in Σ^*. By definition, the decision $s(w)$ is safe in G. It is also the case in G'': If $w \in L$, $s(w) = 1$ since G covers L. As G' covers $\bar{L}$, there exists a winning strategy s' such that $L(s') = \bar{L}$, therefore $s'(w) = 0$ and is a safe decision. G'' has inverted admissible decisions from G', so $s(w) = 1$ is safe in G''. thus, $s(w)$ is safe in $G \cup G''$. Symmetrically, if $w \in \bar{L}$, the decision $s(w) = 0$ is safe in $G \cup G''$. The strategy s prescribes safe decisions on every word $w \in \Sigma^*$, hence it is a winning strategy and $G \cup G''$ is solvable. We finally show that for every winning strategy s in $G \cup G''$, we have $L = L(s)$. Let r be a winning strategy in $G \cup G''$. From Lemma 3.1, we know that there exist winning strategies s, s'' in G, G'' that agree with r on their respective observation alphabets, which both include Σ^*. Since G covers L, we know that $L \subseteq L(s)$ in G, and as r agrees with s on Σ^*, it is also true that $L \subseteq L(r)$. Furthermore, r agrees with s'' on Σ^* as well, and in G'', we have $L(s'') \subseteq L$: G'' is constructed by inverting the admissible decisions of G', which covers $\bar{L}$, so every winning strategy s' in G' is such that $\bar{L} \subseteq L(s')$. Hence, $L(r) = L$ and $G \cup G''$ characterises L. □

3.2.1 Characterising regular languages

As a first exercise, we show how to describe regular languages with consensus game acceptors. The result relies on the fact that a consensus game acceptor with common observations for both players is essentially a finite automaton. We detail the proof to fix the notations before doing more complex manipulations.

We say that a consensus game acceptor $G = (V, E, (\beta^1, \beta^2), v_0, \Omega)$ has *common observation* if $\beta^1 = \beta^2$. In this case, we will simply denote by β the common observation function.

Theorem 3.1. *Let Σ be a finite alphabet. A language $L \subseteq \Sigma^* \setminus \{\varepsilon\}$ is regular if, and only if, there exists a consensus game acceptor with common observation that characterises L.*

One direction of the proof of Theorem 3.1 relies on the following lemma.

Lemma 3.3. *For a language $L \subseteq \Sigma^* \setminus \{\varepsilon\}$, let $G = (V, E, \beta, v_0, \Omega)$ be a consensus game acceptor with common observation and observation alphabet $\Gamma \supseteq \Sigma \cup \{\#\}$ that characterises L. Then, there exists a consensus game acceptor with common observation $G' = (V', E', \beta', v'_0, \Omega')$ that also characterises L, such that for every final state t of G', the set $\Omega'(t)$ is a singleton.*

Proof. Let T be the set of all final states of G and let $T' \subseteq T$ be the set of final states that admits a unique decision. Consider $G' = (V', E', \beta', v'_0, \Omega')$ where:

- $V' = \{v \mid v \in V$, there exists $t \in T'$ reachable from v in $G\}$;
- $E' = E \cap (V' \times V')$;
- $\beta' = \beta_{\restriction V'}$;
- $v'_0 = v_0$;
- $\Omega' = \Omega_{\restriction T'}$.

By construction, G' is a consensus game acceptor with common observation such that $\Omega'(t)$ is a singleton for every final state t. We can remark two facts about G'.

Claim 3.1. *A play in G' corresponds to a play in G that yields the same observation sequence and that admits the same unique decision.*

Proof. Let $\pi = v_0, v_1, \ldots, t$ be a play in G'. Let v and v' be successive states in π. By construction of G', both v and v' are in V and $\beta(v) = \beta'(v)$. Moreover $(v, v') \in E'$, so $(v, v') \in E$. Finally, t is reachable from every state of π, and $t \in T' \subseteq T$. Since $\Omega' = \Omega_{\restriction T'}$, we also have $\Omega(t) = \Omega'(t)$, which is a singleton. Hence the sequence $v_0, v_1, \ldots, t$ is indeed a play in G that yields the same observation sequence and that admits the same unique decision. □

Claim 3.2. *For every winning strategy s in G, there exists a winning strategy s' in G', that agrees with s on all plays that end in T'.*

Proof. Let s be a winning strategy in G. By Claim 3.1, every play in G' is also a play in G that yields the same observation sequence. Consider the strategy s' that agrees with s on all plays that are also in G'. Since s is winning in G and $\Omega' = \Omega_{\restriction T'}$, it is clear that s' also prescribes a safe decision in G'. Hence, s' is a winning strategy in G'. □

We are now ready to show that G' also characterises L: Towards this, we need to show three facts: first, that no observation sequence on Σ^* is omitted in G', second that G' is indeed solvable, and finally that, for every winning strategy s' in G', we have $L(s') = L$.

We know that G characterises L, which means that: (1) for every non-empty word $w \in \Sigma^*$, there exists a play that yields the observation sequence w, (2) G is solvable, and (3) for every winning strategy s, it is true that $s(w) = 1$ if, and only if, $w \in L$.

- *No observation sequence on Σ^* is omitted in G':* Let w be a word in Σ^*. By property (1), there exists a play π with $\beta(\pi) = w$. Assume now that all plays that yield the observation sequence w admit both decisions 0 and 1. By property (2), there exists a winning strategy s such that $s(w) = 1$ if and only if $w \in L$. The strategy s' that agrees with s on $\Sigma^* \setminus \{w\}$, and such that $s'(w) = 0$ if and only if $w \in L$ is a winning strategy in G, since all plays that yield the observation sequence w also admit the decision 0. This contradicts property (3) of G. Thus, for every non-empty word $w \in \Sigma^*$, there exists a play $\pi' = v_0, v_1, ..., t$ in G where $\beta(\pi') = w$ and $t \in T'$.

Since π' is a play in G, the final state t is reachable from every state of π and $(v, v') \in E$ for every two successive states of π'. Since $t \in T'$, all states in π' are in V' and $(v, v') \in E'$ for every two successive states of π'. Therefore π' is a play in G', and since $\beta' = \beta_{\restriction V'}$, it also yields the same observation sequence w. Hence, no observation sequence in Σ^* is omitted in G'.

- *G' is solvable:* Since G characterises L, it means in particular that it is solvable. Let s be a winning strategy of G. By Claim 3.2, we already know that the strategy that agrees with s on all plays that are also in G' is a winning strategy in G'. Therefore, G' is solvable.

- *$L(s') = L$ for every winning strategy s' of G':* Let s' be a winning strategy of G'. By construction, all final states of G' admit only one decision. By Claim 3.1, every play in G' corresponds to a play in G that yields the same observation sequence and ends at the same final node. Since G characterises L, we know that for every non-empty word $w \in \Sigma^*$, and every final state $t \in T'$ such that there exists a play with observation sequence w ending at t, $\Omega(t) = \{1\}$ if, and only if, $w \in L$. As $\Omega' = \Omega_{\restriction T'}$, it follows that $s'(w) = 1$ if, and only if, $w \in L$.

Hence, G' characterises L. □

□

We are now ready to prove Theorem 3.1:

Proof. $\Longleftarrow$: Let L be a language on a finite alphabet Σ, and G a consensus game acceptor that characterises L, such that $\beta^1 = \beta^2$. Thanks to Lemma 3.3, we can assume, without loss of generality, that all final states in G admit only one decision. To prove that L is regular, we extract from G a finite automaton that accepts L. We keep the notations of Lemma 3.3: $G = (V, E, \beta, v_0, \Omega)$ with observation alphabet $\Gamma \supseteq \Sigma \cup \{\#\}$, the set of all final states of G is T.

Consider the finite automaton $\mathcal{A} = (Q, q_0, \Delta, F, \Sigma)$, where:

(1) $Q = \{v \mid v \in V \text{ and } \beta(v) \in \Sigma\} \cup \{v_0\}$ and $q_0 = v_0$;

(2) $F = \{v \in Q \mid \text{ there exists } t \in T,\ (v, t) \in E \text{ and } \Omega(t) = \{1\}\}$;

(3) $\Delta = \{(q, a, q') \in Q \times \Sigma \times Q \mid \beta(q') = a, (q, q') \in E\}$.

Intuitively, $\mathcal{A}$ has the same transition structure as the game graph of G, but the observations are lifted from the states to their incoming edges. We claim that $\mathcal{A}$ indeed accepts L:

- $L \subseteq L(\mathcal{A})$*:* Let $w = a_1 a_2 \dots a_n$ be a word in Σ^*. If $w \in L$, then there exists a play $\pi = v_0, v_1, \dots, v_n$ in G such that $\beta(\pi) = w$ and $\Omega(v_n) = \{1\}$. By construction, $\rho = v_0, v_1, \dots, v_{n-1}$ is a run of $\mathcal{A}$ that accepts w: Indeed, $v_i, v_{i+1} \in Q$ for every two consecutive states v_i, v_{i+1} in ρ, and $(v_i, a_{i+1}, v_{i+1}) \in \Delta$. Furthermore, $\Omega(v_n) = \{1\}$, so $v_{n-1} \in F$.

- $L(\mathcal{A}) \subseteq L$*:* Let w be a word in Σ^*. If w is accepted by $\mathcal{A}$, then, by (1), $w \in \Sigma^*$. Furthermore, there exists a run $\rho = v_0, v_1, \dots, v_n$ on input w with $v_n \in F$. By (2), if $v_n \in F$, then there exists $t \in T$ such that $(v_n, t) \in E$ and $\Omega(t) = \{1\}$. By (3), the play $\pi = v_0, v_1, \dots, v_n, t$ in G is such that $\beta(\pi) = w$. Since G characterises L, it follows that $w \in L$.

$\mathcal{A}$ is a finite automaton that accepts L. Hence, L is regular.

$\Longrightarrow$: Let L be a regular language on a finite alphabet Σ. Let $\mathcal{A}$ be a deterministic finite automaton that accepts L, and $\mathcal{A}'$ be a DFA that accepts its complement $\overline{L} := (\Sigma^* \backslash \{\varepsilon\}) \backslash L$. We assume that $Q \cap Q' = \emptyset$, up to renaming.

Consider the game $G = (V, E, \beta, v_0, \Omega)$ with observation alphabet $\Gamma = \Sigma \cup \{\#\}$, initial state v_0, and set of final states T. The set of states V is composed of an initial state v_0, two final states t and t' with $\Omega(t) = 1$ and $\Omega(t') = 0$, as well as a state (v, a) for each transition in $\mathcal{A}$ or $\mathcal{A}'$ that reaches state v upon reading a. From the initial state v_0, there are edges to states (v, a) such that $\delta(q_0, a) = v$ or $\delta'(q'_0, a) = v$. There is an edge from a state (v, a) to a state (v', a') when either $\delta(v, a') = v'$ or $\delta'(v, a') = v'$. Finally, there is an edge from states (v, a) with $v \in F$ to the final state t, whereas there is an edge from states (v, a) such that $v \in F'$ to the final state t' and the observation function β is defined as follows: $\beta(v_0) = \beta(t) = \beta(t') = \#$, and $\beta((v, a)) = a$ for every state of the form (v, a).

Now, G is a consensus game acceptor with common observation such that $\Omega(t)$ is a singleton for every final state t of G. We first remark two facts about G.

Claim 3.3. *An accepting run labelled by a word w in $\mathcal{A}$ corresponds to a play in G with observation sequence w and ending in t. Similarly, an accepting run labelled by a word w in $\mathcal{A}'$ corresponds to a play in G with observation sequence w and ending in t'.*

Proof. Let $w = a_1 \dots a_n$ be a word accepted by $\mathcal{A}$. There exists a run $\rho = q_0, v_1, \dots, v_n$ in $\mathcal{A}$ labelled by w and such that $q_n \in F$. Thus, for each $i \in [0, n-1]$, we have $\delta(v_i, a_{i+1}) = v_{i+1}$. By (1), $(v_{i+1}, a_{i+1}) \in V$, and by (3), $\beta((v_{i+1}, a_{i+1})) = a_{i+1}$. By (2), for all $j \in [1, n-1]$, we have $((v_j, a_j), (v_{j+1}, a_{j+1})) \in E$, and $(v_0, (v_1, a_1)) \in E$. Moreover, since $v_n \in F$, we have $((v_n, a_n), t) \in E$.

□

Claim 3.4. *For every observable sequence α in G, their exists a unique play π such that $\beta(\pi) = \alpha$.*

Proof. Let $\alpha := w$ be an observable sequence in G. As the observation alphabet Γ is $\Sigma \cup \{\#\}$, and $\#$ is only observed at v_0, t and t', we deduce that $w \in \Sigma^* \setminus \{\varepsilon\}$. By definition of G, there is a play π with observation sequence w, and it has the form $v_0, (v_1, a_1), \ldots, (v_n, a_n), z$, where $w = a_1, \ldots, a_n$ and $z \in T$. Since $w \in \Sigma^* \setminus \{\varepsilon\}$, it is either accepted by $\mathcal{A}$ or by $\mathcal{A}'$, because $\mathcal{A}$ accepts L and $\mathcal{A}'$ accepts its complement. If w is accepted by $\mathcal{A}$, then $q_0, v_1, \ldots, v_n$ is an accepting run labelled by w in $\mathcal{A}$. As $\mathcal{A}$ is deterministic, there exists no other accepting run in $\mathcal{A}$ of the word w. Therefore π is actually unique and equal to $v_0, (v_1, a_1), \ldots, (v_n, a_n), t$, by Claim 3.3. The same argument applies if w is accepted by $\mathcal{A}'$, with $\pi = v_0, (v_1, a_1), \ldots, (v_n, a_n), t'$.

□

We show now that G characterises L:

- *No observation sequence on Σ^* is omitted in G:* Let $w = a_1 \ldots a_n$ be a non-empty word in Σ^*. Since $\mathcal{A}$ accepts L and $\mathcal{A}'$ accepts its complement, w is either accepted by $\mathcal{A}$ or by $\mathcal{A}'$. By Claim 3.3, there exists a play in G with observation sequence w. Hence, no observation sequence on Σ^* is omitted in G.

- *G is solvable:* Consider the function $s : T \to \{0, 1\}$ defined by: $s(t) := 1$ and $s(t') := 0$. Then, s describes a strategy, as, by Claim 3.4, any two plays of G are distinguishable with respect to β, in particular any two plays that end in different final states. Furthermore, as $\Omega(t) = \{1\}$ and $\Omega(t') = \{0\}$ in G, the strategy s is winning. Hence, G is solvable.

- *$L(s) = L$ for every winning strategy s:* We first notice that the strategy s described above is the only winning strategy in G. Let w be a word in $L(s) = \{w \in \Sigma^* \mid s(w) = 1\}$. Hence, we know that $s(w) = 1$. Since s is winning, it means that the play with observation sequence w ends in t. By construction of G, it implies that w is accepted by $\mathcal{A}$. Thus, $w \in L$, and $L(s) \subseteq L$.

  Let now w be a word in L. There exists a run in $\mathcal{A}$ accepting w. We showed in Claim 3.3 that this implies that there exists of a play in G with observation sequence w and ending in t. Therefore, $s(w) = 1$. Thus, $w \in L(s)$, and $L \subseteq L(s)$. In conclusion, $L(s) = L$.

Hence, if L is regular, there exists a consensus game acceptor G with common observation that characterises L. □ □

3.2.2 Domino frontier languages

To show equivalences with other classes of formal languages, we need more involved tools for encoding languages into consensus game acceptors. We use domino systems as an alternative to encoding machine models and formal grammars (See [86] for a survey.). A *domino system* $\mathcal{D} = (D, E_h, E_v)$ is described by a finite set of *dominoes* together with a horizontal and a vertical compatibility relation $E_h, E_v \subseteq D \times D$. The generic domino

tiling problem is to determine, for a given system $\mathcal{D}$, whether copies of the dominoes can be arranged to tile a given space in the discrete grid $\mathbb{Z} \times \mathbb{Z}$, such that any two vertically or horizontally adjacent dominoes are compatible. Here, we consider finite rectangular grids $Z(\ell, m) := \{0, \ldots, \ell + 1\} \times \{0, \ldots, m\}$, where the first and last column, and the bottom row are distinguished as border areas. Then, the question is whether there exists a *tiling* $\tau : Z(\ell, m) \to D$ that assigns to every point $(x, y) \in Z(\ell, m)$ a domino $\tau(x, y) \in D$ such that:

- if $\tau(x, y) = d$ and $\tau(x + 1, y) = d'$ then $(d, d') \in E_h$, and
- if $\tau(x, y) = d$ and $\tau(x, y + 1) = d'$ then $(d, d') \in E_v$.

The *Border-Constrained Corridor* tiling problem takes as input a domino system $\mathcal{D}$ with two distinguished border dominoes $\#$ and $\square$, together with a sequence $w = w_1\, w_2 \ldots w_\ell$ of dominoes $w_i \in D$, and asks whether there exists a height m such that the rectangle $Z(\ell, m)$ allows a tiling τ with w in the top row, $\#$ in the first and last column, and $\square$ in the bottom row:

- $\tau(i, 0) = w_i$, for all $i = 1, \ldots, \ell$;
- $\tau(0, y) = \tau(\ell + 1, y) = \#$, for all $y = 0, \ldots, m - 1$;
- $\tau(x, m) = \square$, for all $x = 1, \ldots, \ell$.

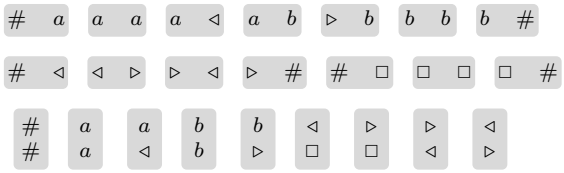
Domino systems can be used to recognise formal languages. For a domino system $\mathcal{D}$ with side and bottom border dominoes as above, the *frontier language* $L(\mathcal{D})$ is the set of words $w \in D^*$ that yield positive instances of the border-constrained corridor tiling problem. We use the following correspondence between context-sensitive languages and domino systems established by Latteux and Simplot.

Theorem 3.2 ([55, 56])**.** *For every context-sensitive language $L \subseteq \Sigma^*$, we can effectively construct a domino system $\mathcal{D}$ over a set of dominoes $D \supseteq \Sigma$ with frontier language $L(\mathcal{D}) = L$.*

Figure 3.2 describes a domino system for recognising the language $a^n b^n$ also covered by the game in Figure 3.1. In the following, we show that domino systems can generally be described in terms of consensus game acceptors.

3.2.3 Uniform encoding of domino problems in games

Game formulations of domino tiling problems are standard in complexity theory, going back to the early work of Chlebus [27]. However, these reductions are typically non-uniform: they construct for every input instance consisting of a domino system together with a border constraint a different game, which depends, in particular, on the size of the constraint. Here, we use imperfect information to define a *uniform* reduction that associates to a fixed domino system $\mathcal{D}$ a game $G(\mathcal{D})$, such that, for every border constraint w, the question whether $\mathcal{D}, w$ allows a correct tiling is reduced to the question of whether decision 1 is safe in a certain play associated to w in $G(\mathcal{D})$.



(a) domino system for $a^n b^n$

#	a	a	a	b	b	b	#
#	a	a	$\triangleleft$	$\triangleright$	b	b	#
#	a	$\triangleleft$	$\triangleright$	$\triangleleft$	$\triangleright$	b	#
#	$\triangleleft$	$\triangleright$	$\triangleleft$	$\triangleright$	$\triangleleft$	$\triangleright$	#
#	$\square$	$\square$	$\square$	$\square$	$\square$	$\square$	#

(b) tiling $a^3 b^3$

Figure 3.2: Characterising a language with dominoes

Proposition 3.1. *For every domino system D, we can construct, in polynomial time, a consensus game acceptor that covers the frontier language of D.*

Proof. Let us fix a domino system $\mathcal{D} = (D, E_h, E_v)$ with a left border domino # and a bottom domino $\square$. We construct a consensus game G for the alphabet $\Sigma := D \setminus \{\#, \square\}$ to cover the frontier language $L(\mathcal{D})$. There are domino states of two types: singleton states d for each $d \in D \setminus \{\#\}$ and pair states (d, b) for each $(d, b) \in E_v$. At singleton states d, the two players receive the same observation d. At states (d, b), the first player observes d and the second player b. The domino states are connected by transitions $d \to d'$ for every $(d, d') \in E_h$, and $(d, b) \to (d', b')$ whenever (d, d') and (b, b') are in E_h. There is an initial state v_0 and two final states $\widehat{z}$ and z, all associated to the observation # for the border domino. From v_0 there are transitions to all compatible domino states d with $(\#, d) \in E_h$, and all pair states (d, b) with $(\#, d)$ and $(\#, b) \in E_h$. Conversely, the final state z is reachable from all domino states d with $(d, \#) \in E_h$, and all pair states (d, b) with $(d, \#)$ and $(b, \#) \in E_h$; the final $\widehat{z}$ is reachable only from the singleton bottom domino state $\square$. Finally, admissible decisions are $\Omega(z) = \{0, 1\}$ and $\Omega(\hat{z}) = \{1\}$. Clearly, G is a consensus game, and the construction can be done in polynomial time.

Note that any sequence $x = d_1\, d_2\, \ldots\, d_\ell \in D^\ell$ that forms a horizontally consistent row in a tiling by $\mathcal{D}$ corresponds in the game to a play $\pi_x = v_0\, d_1\, d_2\, \ldots\, d_\ell, z$ or $\pi_x = v_0\, \square^\ell\, \widehat{z}$. Conversely, every play in G corresponds either to one possible row, in case Nature chooses a single domino in the first transition, or to two rows, in case it chooses a pair. Moreover, a row x can appear on top of a row $y = b_1\, b_2\, \ldots\, b_\ell \in D^\ell$ in a tiling if, and only if, there exists a play ρ in G such that $\pi_x \sim^1 \rho \sim^2 \pi_y$, namely $\rho = v_0\, (d_1, b_1)\, (d_2, b_2)\, \ldots\, (d_\ell, b_\ell)\, z$.

Now, we claim that at an observation sequence $\pi = w$ for $w \in \Sigma^\ell$ the decision 0 is safe if, and only if, there exists no correct corridor tiling by $\mathcal{D}$ with w in the top row. According to our remark, there exists a correct tiling of the corridor with top row w, if and only if, there exists a sequence of rows corresponding to plays $\pi_1, \ldots, \pi_m$, and a sequence of witnessing plays $\rho_1, \ldots, \rho_{m-1}$ such that $w = \pi_1 \sim^1 \rho_1 \sim^2 \pi_2 \cdots \sim^1 \rho_{m-1} \sim^2 \pi_m = \square^\ell$. However, the decision 0 is unsafe in the play $\square^\ell$ and therefore at w as well. Hence, every winning strategy s for G must prescribe $s(w) = 1$, for every word w in the frontier language of $\mathcal{D}$, meaning that $L(s) \subseteq L(\mathcal{D})$.

Finally, consider the mapping $s : D^* \to A$ that prescribes $s(w) = 1$ if, and only if, $w \in L(\mathcal{D})$. The observation-based strategy s in the consensus game G is winning since $s(\square^*) = 1$, and it witnesses the condition $L(s) = L(\mathcal{D})$. This concludes the proof that the constructed consensus game G covers the frontier language of $\mathcal{D}$. $\square$

3.3 Characterising Context-Sensitive Languages

Our first main result establishes a correspondence between context-sensitive languages and consensus games.

Theorem 3.3. *For every context-sensitive language $L \subseteq \Sigma^*$, we can effectively construct a consensus game acceptor that characterises L.*

Proof. Let $L \subseteq \Sigma^*$ be an arbitrary context-sensitive language, represented, e.g., by a linear-bounded automaton. By Theorem 3.2, we can effectively construct a domino system $\mathcal{D}$ with frontier language L. Further, by Proposition 3.1, we can construct a consensus game G that covers $L(\mathcal{D}) = L$. Due to the Immerman-Szelepcsényi Theorem, context-sensitive languages are effectively closed under complement, so we can construct a consensus game G' that covers $\Sigma^* \setminus L$ following the same procedure. Finally, we combine the games G and G' as described in Lemma 3.2 to obtain a consensus game that characterises L. □

One interpretation of the characterisation is that, for every context-sensitive language, there exists a consensus game that is as hard to play as it is to decide membership in the language. On the one hand, this implies that winning strategies for consensus games are in general PSPACE-hard. Indeed, there are instances of consensus games that admit winning strategies, however, any machine that computes the decision to take in a play requires space polynomial in the length of the play.

Remark. In Section 3.2, given a consensus game acceptor G, and to every observation-based strategy $s \in S^1$ of the first player, we associated the language $L(s) := \{w \in \Sigma^* \mid s(w) = 1\}$. Therefore, we say that a winning strategy s is PSPACE-hard if the membership problem for $L(s)$ is PSPACE-hard.

Theorem 3.4. *There exists a solvable consensus game for which every winning strategy is* PSPACE*-hard.*

Proof. There exist context-sensitive languages with a PSPACE-hard word problem [54]. Let us fix such a language $L \subseteq \Sigma^*$ together with a consensus game G that characterises it, according to Theorem 3.3. This is a solvable game, and every winning strategy can be represented as an observation-based strategy s for the first player. Then, the membership problem for L reduces (in linear time) to the problem of deciding the value of s in a play in G: For every input word $w \in \Sigma^*$, we have $w \in L$ if, and only if, $s(w) = 1$. In conclusion, for every winning strategy s in G, it is PSPACE-hard to decide whether $s(w) = 1$. □

On the other hand, it follows that determining whether a consensus game admits a winning strategy is no easier than solving the emptiness problem of context-sensitive languages, which is well known to be undecidable.

Theorem 3.5. *The question whether a consensus game admits a winning strategy is undecidable.*

Proof. We reduce the emptiness problem for a context-sensitive grammar to the solvability problem for a consensus game. For an arbitrary context-sensitive language $L \in \Sigma^*$ given

as a linear bounded automaton, we construct a consensus game G that characterises L, in polynomial time, according to Theorem 3.3. Additionally, we construct a consensus game G' that characterises the empty language over Σ^*: this can be done, for instance, by connecting a clique over letters in Σ observable for both players to a final state at which only the decision 0 is admissible. Now, for any word $w \in \Sigma^*$, the game G' requires decision 0 at every observation sequences $w \in \Sigma^*$, whereas G requires decision 1 whenever $w \in L$. Accordingly, the consensus game $G \cup G'$ is solvable if, and only if, L is empty. As the emptiness problem for context-sensitive languages is undecidable [54], it follows that the solvability problem is undecidable for consensus game acceptors. □

We have seen that every context-sensitive language corresponds to a consensus game acceptor such that language membership tests reduce to winning strategy decisions in a play. Conversely, every solvable game admits a winning strategy that is the characteristic function of some context-sensitive language. Intuitively, a strategy should prescribe 0 at a play π whenever there exists a connected play π' at which 0 is the only admissible decision. Whether this is the case can be verified by a nondeterministic machine using space linear in the length of the play π.

Theorem 3.6. *Every solvable consensus game admits a winning strategy that is implementable by a nondeterministic linear bounded automaton.*

Proof. Let $G = (V, E, (\beta^1, \beta^2), v_0, \Omega)$ be a solvable consensus game acceptor over an observation alphabet Γ. The fact that G is solvable means, in particular, that every play admits at least one *safe* decision: for every play π, there exist no two plays π' and π'' with admissible decision sets $\Omega(\pi') = \{0\}$ and $\Omega(\pi'') = \{1\}$ such that $\pi' \sim^* \pi \sim^* \pi''$. Hence, the function s that prescribes 0 at a play π whenever there exists a connected play π' at which 0 is the only admissible decision and 1 otherwise is a winning strategy. Furthermore, this winning strategy s is implementable by a nondeterministic linear bounded automaton: on input π, the automaton the automaton successively guesses pairs of connected plays, and checks if the consecutive plays are indeed indistinguishable for one of the players, and if the admissible decision set of the second play is $\{0\}$, in which case the run is accepting. If the second play admits both decisions 0 and 1, it is written over the previous play in the sequence, and the automaton proceeds to guess the next connected play in the sequence, in order to ensure the used space stays linear in the size of the original play π. □

3.4 Consensus and Iterated Transductions

Our aim in the following is to investigate how the structure of a consensus game relates to the complexity of the described language which, in turn, determines the complexity of winning strategies. Towards this, we view games as finite-state automata representing the relation between the observation sequences received by the players and the admissible decisions.

A synchronous *transducer* is a two-tape automaton $(Q, \Gamma, \Delta, q_0, F)$ over an alphabet Γ, with state set Q, a transition relation $\Delta \subseteq Q \times \Gamma \times \Gamma \times Q$ labelled by pairs of letters, an initial state $q_0 \in Q$ and a non-empty set $F \subseteq Q$ of final states; in contrast to games,

final states of transducers may have outgoing transitions. We write $p \xrightarrow{a|b} q$ to denote a transition $(p, a, b, q) \in \Delta$. An *accepting run* of the transducer is a path

$$\rho \;=\; q_0 \xrightarrow{a_1|b_1} q_1 \xrightarrow{a_2|b_2} \dots \xrightarrow{a_n|b_n} q_n$$

that follows transitions in Δ starting from the initial state q_0 and ending at a final state $q_n \in F$. The *label* of the run is the pair of words $(a_1 \dots a_n, b_1, \dots b_n)$. A pair of words $(w, w') \in \Gamma^* \times \Gamma^*$ is *accepted* by the transducer if it is the label of some accepting run. The relation *recognised* by the transducer is the set $R \subseteq \Gamma^* \times \Gamma^*$ of accepted pairs of words. A relation is *synchronous* if it is recognised by a synchronous transducer. In general, we do not distinguish notationally between transducers and the relation they recognise. For background on synchronous, or letter-to-letter, transducers, we refer to the survey [3] of Berstel and to Chapter IV in the book [75] of Sakarovitch.

Given a consensus game acceptor $G = (V, E, \beta^1, \beta^2, v_0, \Omega)$ over an observation alphabet Γ, we define the *seed* of G to be the triple $(R, L_{\text{acc}}, L_{\text{rej}})$ consisting of the relation $R := \{\, (\beta^1(\pi), \beta^2(\pi)) \in (\Gamma \times \Gamma)^* \mid \pi \text{ play in } G \,\}$ together with the languages $L_{\text{acc}} \subseteq \Gamma^*$ and $L_{\text{rej}} \subseteq \Gamma^*$ of observation sequences $\beta^1(\pi)$ on plays π in G with $\Omega(\pi) = \{1\}$ and $\Omega(\pi) = \{0\}$, respectively. The seed of any finite game can be represented by finite-state automata.

Lemma 3.4. *For every consensus game, the seed languages $L_{\text{acc}}, L_{\text{rej}}$ are regular and the seed relation R is recognised by a synchronous transducer.*

Proof. We construct automata from the game graph by moving observations from each state to the incoming transitions. Let $G = (V, E, \beta, v_0, \Omega)$ be a consensus game over an observation alphabet Γ, and let $(R, L_{\text{acc}}, L_{\text{rej}})$ be its seed. We define three automata over the alphabet Γ, on the subset of V consisting of non-final game states and with initial state $q_0 = v_0$.

The automata for the seed languages L_{acc} and L_{rej} allow transitions $u \xrightarrow{a} v$ if $(u, v) \in E$ and $\beta^1(v) = a$. The set of final states consists of all game states v with an outgoing transition $(v, v') \in E$ to some final state with $\Omega(v') = \{1\}$ for L_{acc}, and with $\Omega(v') = \{0\}$ for L_{rej}. Then, for all words $a_1 \dots a_n \in \Gamma^*$, accepting runs $v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} v_n$ of the automata L_{acc} and L_{rej} correspond to plays $\pi := v_0 v_1 \dots v_n v'$ with observations $\beta^1(\pi) = a_1 \dots a_n$ such that $\Omega(\pi) = \{1\}$ and $\Omega(\pi) = 0$, respectively. Hence, the automata recognise L_{acc} and L_{rej}.

Similarly, the transducer for the seed relation has transitions $u \xrightarrow{a|b} v$ whenever $(u, v) \in E$ with $\beta^1(v) = a$, $\beta^2(v) = b$, and its final states are states $v \in V$ with an outgoing transition $(v, v') \in E$ to some final state v' in G. Then, for any pair of words $(a_1 \dots a_n, b_1 \dots b_n) \in \Gamma^* \times \Gamma^*$, there exists an accepting run $v_0 \xrightarrow{a_1|b_1} v_1 \xrightarrow{a_2|b_2} \dots \xrightarrow{a_n|b_n} v_n$ of the transducer if, and only if, there exists a play $\pi = v_0, v_1, \dots, v_n, v'$ with observations $\beta^1(\pi) = a_1 \dots a_n$, and $\beta^2(\pi') = b_1 \dots b_n$, for some final game state v'. So, the transducer recognises the seed relation R, as intended. □

Conversely, we can turn synchronous transducers and automata over matching alphabets into games.

Lemma 3.5. *Given a synchronous relation $R \subseteq \Gamma^* \times \Gamma^*$ and two disjoint regular languages L_{acc}, $L_{\text{rej}} \subseteq \Gamma^*$, we can construct a consensus game with seed $(R, L_{\text{acc}}, L_{\text{rej}})$.*

Proof. Let us consider a synchronous transducer $\mathcal{R}$ and two word automata $\mathcal{A}$, $\mathcal{B}$ that recognise R, L_{acc} and L_{rej}, respectively. We assume that the word automata are deterministic, with transition functions $\delta^{\mathcal{A}} : Q^{\mathcal{A}} \times \Gamma \to Q^{\mathcal{A}}$ and $\delta^{\mathcal{B}} : Q^{\mathcal{B}} \times \Gamma \to Q^{\mathcal{B}}$. To avoid confusion, we label the components of each automaton with its name and write $p \xrightarrow[\mathcal{R}]{a} q$ for the transitions in $\mathcal{R}$.

We construct a game G over the observation alphabet $\Gamma \cup \{\#\}$ with states formed of three components: a transducer transition $p \xrightarrow[\mathcal{R}]{a|b} q$, a state of $\mathcal{A}$, and one of $\mathcal{B}$. The game transitions follow the adjacency graph of the transducer in the first component and update the state of $\mathcal{A}$ and $\mathcal{B}$ in the second and third component according to the observation a of Player 1 in the first component. More precisely, the set of game states is $V := \Delta \times Q^{\mathcal{A}} \times Q^{\mathcal{B}} \cup \{v_0, v_{\text{acc}}, v_{\text{rej}}, v_{=}\}$, where v_0 is a fresh initial state whereas v_{acc}, v_{rej} and $v_{=}$ are final states. Game transitions lead from the initial state v_0 to all states $(q_0^{\mathcal{R}} \xrightarrow[\mathcal{R}]{a|b} q, q_0^{\mathcal{A}}, q_0^{\mathcal{B}})$; for each pair of incident transducer transitions $e := p \xrightarrow[\mathcal{R}]{a|b} q$ and $e' := q \xrightarrow[\mathcal{R}]{a'|b'} q'$, and for all automata states $q^{\mathcal{A}} \in Q^{\mathcal{A}}$, $q^{\mathcal{B}} \in Q^{\mathcal{B}}$, there is a game transition from $(e, q^{\mathcal{A}}, q^{\mathcal{B}})$ to $(e', \delta^{\mathcal{A}}(q^{\mathcal{A}}, a), \delta^{\mathcal{A}}(q^{\mathcal{B}}, a))$; finally, from any state $(p \xrightarrow[\mathcal{R}]{a|b} q, q^{\mathcal{A}}, q^{\mathcal{B}})$ with $q \in F^{\mathcal{R}}$ there is a transition to v_{acc} if $q^{\mathcal{A}} \in F^{\mathcal{A}}$, to v_{rej} if $q^{\mathcal{B}} \in F^{\mathcal{B}}$, and otherwise to $v_{=}$. The observation at state $(p \xrightarrow[\mathcal{R}]{a|b} q, q^{\mathcal{A}}, q^{\mathcal{B}})$ is (a, b); at the initial and the final states both players observe $\#$. Admissible decisions are $\Omega(v_{\text{acc}}) = \{1\}$, $\Omega(v_{\text{rej}}) = \{0\}$, and $\Omega(v_{=}) = \{0, 1\}$.

Now, for any play π, the first component corresponds to an accepting run of $\mathcal{R}$ on the pair of observation sequences $(\beta^1(\pi), \beta^2(\pi))$, whereas the second and third components correspond to runs of $\mathcal{A}$ and $\mathcal{B}$ on $\beta^1(\pi)$ which are accepting if $\Omega(\pi) = \{1\}$ and $\Omega(\pi) = \{0\}$, respectively. Accordingly, the game G has seed $(R, L_{\text{acc}}, L_{\text{rej}})$. □

Thanks to the translation between games and automata, we can reason about games in terms of elementary operations on their seed. Our notation is close to the one of Terlutte and Simplot [80]. Given a relation $R \subseteq \Gamma^* \times \Gamma^*$, the inverse relation is

$$R^{-1} := \{(x, y) \in \Gamma^* \times \Gamma^* \mid (y, x) \in R\}.$$

The composition of R with a relation $R' \in \Gamma^*$ is

$$RR' := \{(x, y) \in (\Gamma \times \Gamma)^* \mid (x, z) \in R \text{ and } (z, y) \in R' \text{ for some } z \in \Gamma^*\}.$$

For a language $L \subseteq \Gamma^*$, we write $RL := \{x \in \Gamma^* \mid (x, y) \in R \text{ and } y \in L\}$. For a subalphabet $\Sigma \subseteq \Gamma$, we denote the identity relation by $(\cap\Sigma^*) := \{(x, x) \in \Sigma^* \times \Sigma^*\}$. The power R^k of R is defined by $R^0 := (\cap\Gamma)$, and $R^{k+1} := R^k R$ for $k > 0$. Finally, the iteration of R is $R^* := \cup_{0 \leq k < \omega} R^k$.

One significant relation obtained from the seed transducer R of a game G is the *reflection* relation $\tau(R) := RR^{-1}$. That is, a word $w \in \Gamma^*$ over the observation alphabet is a reflection of $u \in \Gamma^*$ if, whenever Player 1 observes u, Player 2 considers it possible that 1 actually observes w. Obviously, this relation is reflexive and symmetric. Its transitive closure relates observations received on connected plays.

Lemma 3.6. *Let G be a consensus game with seed relation $R \subseteq \Gamma^* \times \Gamma^*$, and let $\tau := RR^{-1}$ be its reflection. Then,*

(i) $\tau = \{ (\beta^1(\pi), \beta^1(\pi')) \mid \textit{plays } \pi \sim^2 \pi' \textit{ in } G \}$, *and*

(ii) $\tau^* = (\cap\Gamma^*) \cup \{ (\beta^1(\pi), \beta^1(\pi')) \mid \textit{plays } \pi \sim^* \pi' \textit{ in } G \}$.

Proof. (*i*) By definition of the seed relation, for any pair of words $(w, w') \in RR^{-1}$, there exist plays π, π' such that $\beta^1(\pi) = w$, $\beta^1(\pi') = w'$, and $\beta^2(\pi) = \beta^2(\pi')$, that is, $\pi \sim^2 \pi'$. Conversely, for any pair of plays $\pi \sim^2 \pi'$, the observation sequences are related by $(\beta^1(\pi), \beta^2(\pi)) \in R$ and $(\beta^2(\pi), \beta^1(\pi')) = (\beta^2(\pi'), \beta^1(\pi')) \in R^{-1}$. Hence $(\beta^1(\pi), \beta^1(\pi')) \in RR^{-1}$.

(*ii* "$\supseteq$"): Clearly, $(\cap\Gamma^*) \subseteq \tau^*$. Further, by definition of connectedness, if $\pi \sim^* \pi'$, then there exists a sequence of plays $(\pi_\ell)_{\ell \leq 2k}$ with $\pi_0 = \pi$, $\pi_{2k} = \pi'$, and $\pi_\ell \sim^1 \pi_{\ell+1} \sim^2 \pi_{\ell+2}$ for all even $\ell < 2k$. Therefore, the observation sequences $x := \beta^1(\pi_\ell) = \beta^1(\pi_{\ell+1})$, $y := \beta^1(\pi_{\ell+2})$, and $z := \beta^2(\pi_{\ell+1}) = \beta^2(\pi_{\ell+2})$ are related by $(x, z) \in R$ and $(z, y) \in R^{-1}$, which means that $(x, y) = (\beta^1(\pi_\ell), \beta^1(\pi_{\ell+2})) \in RR^{-1}$, for all even $\ell < 2k$. Accordingly, we obtain $(\beta^1(\pi), \beta^1(\pi')) \in (RR^{-1})^*$.

(*ii* "$\subseteq$") To show that every pair of distinct words in τ^* can be observed by Player 1 on connected plays, we verify by induction on the power $k \geq 1$ that for every pair $(w, w') \in (RR^{-1})^k$, there exists a sequence of plays $\pi_0 \sim^1 \pi_1 \sim^2 \cdots \sim^2 \pi_{2k}$ such that $\beta^1(\pi_0) = w$ and $\beta^1(\pi_{2k}) = w'$.

The base case for $k = 1$ follows from point (*i*) of the present lemma: if $(w, w') \in (RR^{-1})$, then there exist $\pi \sim^2 \pi'$ with $\beta^1(\pi) = w$, $\beta^1(\pi') = w'$, and we set $\pi_0 = \pi_1 = \pi$ and $\pi_2 = \pi'$. For the induction step, assume that the hypothesis holds for a power $k \geq 1$ and consider $(w, w') \in (RR^{-1})^{k+1}$. That is, there exists a word $z \in \Gamma^*$ such that $(w, z) \in (RR^{-1})^k$ and $(z, w') \in (RR^{-1})$. The former implies, by induction hypothesis, that we have a chain $\pi_0 \sim^1 \pi_1 \sim^2 \cdots \sim^1 \pi_{2k-1} \sim^2 \pi_{2k}$ with $\beta^1(\pi_0) = w$ and $\beta^1(\pi_{2k}) = z$; from the latter it follows, by definition of R, that there exist plays π and π' with $\beta^1(\pi) = z = \beta^1(\pi_{2k})$, $\beta^2(\pi) = \beta^2(\pi')$, and $\beta^1(\pi') = w'$. Therefore, we can prolong the witnessing chain by setting $\pi_{2k+1} := \pi$, $\pi_{2k+2} := \pi'$, which concludes the induction argument. □

The consensus condition requires decisions to be invariant under the reflection relation. This yields the following characterisation of winning strategies.

Lemma 3.7. *Let G be a consensus game with seed $(R, L_{\mathrm{acc}}, L_{\mathrm{rej}})$ over an alphabet Γ, and let $\tau := RR^{-1}$ be its reflection relation. Then, a strategy $s : \Gamma^* \to \{0, 1\}$ of Player 1 is winning if, and only if, it assigns $s(w) = 1$ to every observation sequence $w \in \tau^* L_{\mathrm{acc}}$ and $s(w) = 0$ to every observation sequence $w \in \tau^* L_{\mathrm{rej}}$.*

Proof. ("$\Longrightarrow$") According to Lemma 3.6(*ii*), every word $w \in \tau^* L_{\mathrm{acc}}$ corresponds to the observation sequence $\beta^1(\pi) = w$ of a play π in G, and there exists a connected play $\pi' \sim^* \pi$ with $\pi' \in L_{\mathrm{acc}}$. Therefore, any winning strategy $s : \Gamma^* \to \{0, 1\}$ for Player 1 must assign $s(w) = 1$ to all $w \in L_{\mathrm{acc}}$ and further, to all $w \in \tau^* L_{\mathrm{acc}}$, by the conditions of consensus and indistinguishability. Likewise, it follows that $s(w) = 0$ for all $w \in \tau^* L_{\mathrm{rej}}$.

(“$\Longleftarrow$”) Consider the mapping $s : V^* \to \{0,1\}$ with $s(w) = 1$ if, and only if, $\beta^1(\pi) \in \tau^* L_{\text{acc}}$. Then s is a valid strategy, as for all $\pi \sim^2 \pi'$ we have $(\beta^1(\pi), \beta^1(\pi')) \in \tau$ and $(\beta^1(\pi'), \beta^1(\pi)) \in \tau$, hence $\beta^1(\pi) \in \tau^* L_{\text{acc}}$ if, and only if, $\beta^1(\pi') \in \tau^* L_{\text{acc}}$. If it is the case that $s(\pi) = 0$ for all $\pi \in \tau^* L_{\text{rej}}$, that is, $\tau^* L_{\text{acc}} \cap \tau^* L_{\text{rej}} = \emptyset$, then s is a winning strategy. □

As a direct consequence, we can characterise the language defined by a game in terms of iterated transductions.

Theorem 3.7. *Let G be a consensus game with seed $(R, L_{\text{acc}}, L_{\text{rej}})$, and let Σ be a subset of its alphabet. Then, for the reflection $\tau := RR^{-1}$, we have:*

(i) *G is solvable if, and only if, $\tau^* L_{\text{acc}} \cap \tau^* L_{\text{rej}} = \emptyset$.*

(ii) *If G is solvable, then it covers the language $(\cap\Sigma^*)\tau^* L_{\text{acc}}$.*

(iii) *If G is solvable and $(\cap\Sigma^*)(\tau^* L_{\text{acc}} \cup \tau^* L_{\text{rej}}) = \Sigma^*$, then G characterises the language $(\cap\Sigma^*)\tau^* L_{\text{acc}}$.*

3.5 Consensus Games for Context-Free Languages

Properties of iterated letter-to-letter transductions, or equivalently, length-preserving transductions, have been investigated by Latteux, Simplot, and Terlutte in [57, 80], where it is also shown that iterated synchronous transductions capture context-sensitive languages. Our setting is, however, more restrictive in that games correspond to symmetric transductions. In the following, we investigate a family of consensus game acceptors that captures context-free languages. Since the class is not closed under complement, we will work with the weaker notion of covering a language rather than characterising it. For the language-theoretic discussion, we generally assume that the rejecting seed language L_{rej} is empty and specify the seed $(R, L_{\text{acc}}, \emptyset)$ as (R, L_{acc}).

Firstly, we remark that regular languages correspond to games where the two players have the same observation function. Clearly, such games admit regular winning strategies whenever they are solvable.

Remark. We have already shown this result in Section 3.2.1 as a way to familiarise with the specificities of consensus game acceptors and suggest their similarities with automata. Here, we see that the transducer point of view provides a much more direct proof.

Proposition 3.2. *A language $L \subseteq \Sigma^*$ is regular if, and only if, it is characterised by a consensus game acceptor where the seed relation is the identity.*

Proof. Every regular language $L \subseteq \Sigma^*$ is characterised by the game with seed $(\,(\cap\Sigma^*), L, \Sigma^* \setminus L\,)$. Conversely, suppose a language $L \subseteq \Sigma^*$ is characterised by a game G over an alphabet $\Gamma \supseteq \Sigma$ with seed $(\,(\cap\Gamma^*), L_{\text{acc}}, L_{\text{rej}}\,)$. Then, G also covers L and, by Theorem 3.7(*ii*), it follows that $L = (\cap\Sigma^*)(\cap\Gamma^*)^* L_{\text{acc}} = \Sigma^* \cap L_{\text{acc}}$. Hence, L is regular. □

Dyck languages. As a next exercise, let us construct games for covering Dyck languages, that is, languages of well-balanced words of brackets; we also allow neutral symbols, which may appear at any position without affecting the bracket balance. Our terminal alphabet A consists of an alphabet $B_n = \{\ [_k,]_k \mid 1 \leq k \leq n\ \}$ of $n \geq 1$ matching brackets and a set C of neutral symbols. For a word $w \in A^*$ and an index k, we denote by $\mathrm{excess}^k(w)$ the difference between the number of opening and of closing brackets $[_k$ and $]_k$. Then, the Dyck language D_A over A consists of the words $w \in A^*$ such that, for each kind of brackets $k \in \{1, \ldots, k\}$, $\mathrm{excess}^k(w) = 0$, whereas for all prefixes w' of w, $\mathrm{excess}^k(w') \geq 0$.

Given a terminal alphabet $A = B_n \cup C$, we define the transducer $R_{n,C}$ over the observation alphabet $\Gamma := A \cup \{\square\}$ with a set of states $\{q_0, q_1, \ldots, q_n\}$, among which q_0 is the initial and the only final state, and with the following two kinds of transitions: *copying* transitions $q_0 \xrightarrow{a|a} q_0$ for all $a \in \Gamma$, as well as $q_k \xrightarrow{\square|\square} q_k$ for every $k \in \{1, \ldots n\}$, and *erasing* transitions $q_0 \xrightarrow{[_k|\square} q_k$ and $q_k \xrightarrow{]_k|\square} q_0$ for the brackets of each kind k, and $q_0 \xrightarrow{c|\square} q_0$ for each neutral symbol $c \in C$. Essentially, $R_{n,C}$ erases neutral symbols and any innermost pair of brackets.

Lemma 3.8. *The Dyck language over an alphabet n of matching brackets and a set C of neutral symbols is covered by the game with seed $(R_{n,C}, \square^*)$.*

Proof. For a terminal alphabet A with partitions B_n, C as in the statement, we denote the corresponding Dyck language by D_A and the previously defined transduction by $R := R_{n,C}$. The observation alphabet $\Gamma = B_n \cup C \cup \{\square\}$ extends A with an additional neutral symbol $\square$; let $D_\Gamma \supseteq D_A$ be the Dyck language over this extended alphabet. Consider now the game G over Γ with seed $(R, \square^*)$. We using the reflection relation $\tau := RR^{-1}$ to argue that $D_A = \tau^*\square^*$.

To see that the Dyck language D_A is contained in the language $\tau^*\square^*$ covered by G, observe that for every pair of words $u, u' \in \Gamma^*$ where $u \in D_\Gamma$ and u' is obtained from u by replacing one innermost pair of matching brackets with $\square$, we have $(u, u') \in R$. Since the relation R contains the identity on Γ, it follows that $(u, u') \subseteq RR^{-1}$, so $(u, u') \in \tau$. If we set out with an arbitrary word $w \in A^*$, first erase all neutral symbols by applying R once, and then repeat applying R to erase an innermost pair of brackets, we end up with $\square^*$, hence $w \in \tau^*\square^*$.

Conversely, to verify that every word in $\tau^*\square^*$ has well-balanced brackets, we show that D_Γ is invariant under the transductions R and R^{-1} in the sense that for any pair $(u, w) \in R \cup R^{-1}$, we have $u \in D_\Gamma$ if, and only if, $w \in D_\Gamma$. Towards this, let us fix an accepting run of R on $u|w$ and compare the values $\mathrm{excess}^k(u')$ and $\mathrm{excess}^k(w')$ of its prefixes $u'|w'$, for any $k \geq n$: the values are equal until a transition $q_0 \xrightarrow{[_k|\square} q_k$ is taken at some prefix $u'[_k|w'\square$. Since q_k is not final, the run will take the transition $q_k \xrightarrow{]_k|\square} q_0$ at some later position; let $u''{]_k}|w''\square$ be the shortest continuation of $u'|w'$ at which q_0 is reached again. Hence, we set out with $\mathrm{excess}^k(w') = \mathrm{excess}^k(u')$, and also have $\mathrm{excess}^k(u') = \mathrm{excess}^k(u'')$, since none of $[_k$ or $]_k$ is transduced while looping in q_k; after returning to q_0, again $\mathrm{excess}^k(w'') = \mathrm{excess}^k(u'')$, because the opening bracket was matched. So, it is the case that $\mathrm{excess}^k(u') \geq 0$ for all prefixes u' of u if, and only if, $\mathrm{excess}^k(w') \geq 0$ for all prefixes w' of w, which means that $RD_\Gamma \subseteq D_\Gamma$ and $R^{-1}D_\Gamma \subseteq D_\Gamma$;

the converse inclusions hold because R contains the identity on Γ. Accordingly, for every sequence $w_0, \ldots, w_\ell$ of words with $w_0 = w$ such that $(w_i, w_{i+1}) \in \tau$ for each $i < \ell$, we have $w_i \in D_\Gamma$ if, and only if, $w_{i+1} \in D_\Gamma$. Since $\Box^* \in D_\Gamma$, it follows that $w \in D_\Gamma$, for any word $w \in \tau^*\Box^*$. In conclusion, $(\cap A^*)\tau^*\Box^* \subseteq D_A$. □

Context-free languages. To extend the consensus-game description of Dyck languages to arbitrary context-free languages, we use the Chomsky-Schützenberger representation theorem [28] in the non-erasing variant proved by Okhotin [67]. A letter-to-letter homomorphism $h : A^* \to \Sigma^*$ is a functional synchronous transduction that preserves concatenation, that is, $h(uw) = h(u)h(w)$ for all words $u, w \in A^*$. Such a homomorphism is identified by its restriction $f : A \to \Sigma$ to single letters.

Theorem 3.8 ([67])**.** *A language $L \subseteq \Sigma^*$ is context-free if, and only if, there exists a Dyck language D_A over an alphabet A of brackets and neutral symbols, a regular language $M \subseteq A^*$, and a letter-to-letter homomorphism $h : A^* \to \Sigma^*$, such that $L = h(\, D_A \cap M\,)$.*

We will show how a game acceptor that covers an arbitrary language $L \subseteq A^*$ can be extended to cover a homomorphic image of the intersection of L with a regular language. Let $h : A \to \Sigma$ be a letter-to-letter homomorphism, and let R be a synchronous transducer over an alphabet $\Gamma \supseteq A$. We construct from R a new transducer R_h over the enlarged alphabet $\Sigma \cup \Gamma \times A$ by adding a *coding* cycle. This is done by including a fresh final state q_h, as well as transitions $q_0 \xrightarrow{h(a)|(a,a)} q_h$ and $q_h \xrightarrow{h(a)|(a,a)} q_h$ for all $a \in A$, and then relabelling each transition $p \xrightarrow{a|b} q$ of R to $p \xrightarrow{(a,x)|(b,x)} q$, for all $x \in A$. Intuitively, the new transducer duplicates the automaton tapes into two tracks which are both initialised with a homomorphic pre-image $u \in A^*$ of a terminal word $w \in \Sigma^*$, in a transduction via the coding cycle. The first track is intended to simulate R on the pre-image u, whereas the second track stores u: the contents is looped through every other transduction of R_h or of its inverse. Notice that every run of R_h proceeds either through the new coding cycle, or through the original transducer R, in the sense that, for any pair $(w, w') \in R_h$ we have $\{w, w'\} \subseteq \Sigma^* \cup (\Gamma \times A)^*$.

Lemma 3.9. *Suppose that a game acceptor with seed (R, L_{acc}) covers a language $L \subseteq A^*$. Let $M \subseteq A^*$ be a regular language and let $h : A^* \to \Sigma^*$ be a letter-to-letter homomorphism. Then, the game acceptor with seed $(R_h, L_{\mathrm{acc}} \times M)$ covers the language $h(L \cap M)$ over the terminal alphabet Σ.*

Proof. Let G be a game over an alphabet $\Gamma \supseteq A$ with seed (R, L_{acc}). Without loss of generality, we assume that R contains the identity on A, otherwise we take the reflexive transduction RR^{-1} to obtain the seed of a game that covers the same language. Further, let $M \subseteq A^*$ be a regular language and let $h : A \to \Sigma$ represent a letter-to-letter homomorphism as in the statement. We argue for the case where the alphabets Σ and Γ are disjoint; the general case follows by composition with a relabelling homomorphism. Now, consider the game G' with seed transducer R_h and accepting language $L'_{\mathrm{acc}} := L_{\mathrm{acc}} \times M$. We denote the reflection relations associated R and R_h by $\tau := RR^{-1}$ and $\tau' := R_h R_h^{-1}$.

To see that $h(L \cap M)$ is included in the language covered by G', consider a word $w = h(u)$ for some $u \in \tau^* L_{\mathrm{acc}} \cap M$. By Theorem 3.7, there exists a witnessing sequence

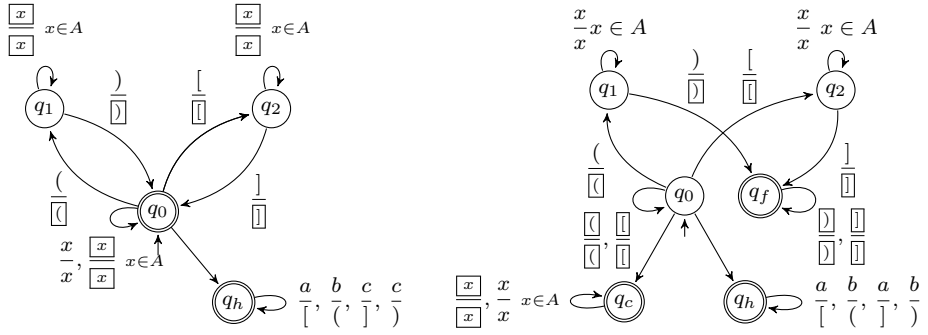
$(u_i)_{i\leq\ell}$ with $u_0 = u$, $u_\ell \in L_{\mathrm{acc}}$, and $(u_i, u_{i+1}) \in \tau$ for all $i < \ell$. By construction of R_h, we have $(w, (u, u)) \in R_h$ which implies $(w, (u, u)) \in \tau'$, thanks to our assumption that $(\cap A^*) \subseteq R$. Since $(u_\ell, u) \in L_{\mathrm{acc}} \times M$, the sequence starting with w and followed by $((u_i, u))_{i<\ell}$ is witnessing that $w \in \tau'^* L'_{\mathrm{acc}}$.

Conversely, consider a word $w \in (\cap\Sigma^*)\tau'^* L'_{\mathrm{acc}}$ and let $(w_i)_{i\leq\ell}$ be a witnessing sequence with $w_0 = w$, $w_\ell \in L'_{\mathrm{acc}}$, and $(w_i, w_{i+1}) \in \tau'$ for all $i \leq \ell$. By construction of R_h, the initial word w is preserved at each term w_i of the sequence, in the sense that either $w_i = w$, or $w_i = (u_i, x_i) \in \Gamma^* \times A^*$ for some $x_i \in A^*$ such that $w = h(x_i)$. By our assumption that Σ and Γ are disjoint, we have $w \notin L_{\mathrm{acc}}$, so there exists a last position $k < \ell$ with $w_k = w$. As $w \in \Sigma^*$ can only be transduced via the coding cycle, it follows that $w_{i+1} = (u, u)$ for some $u \in A^*$ with $h(u) = w$. For each following position $i > k$, the terms of the sequence are of the form $w_i = (u_i, u)$ for a certain word $u_i \in \Gamma^*$. Hence the coding cycle cannot be applied and the sequence $(u_i)_{k<i\leq\ell}$ satisfies, $(u_i, u_{i+1}) \in \tau$ for all $i < \ell$. Moreover, $w_\ell = (u_\ell, u) \in L'_{\mathrm{acc}} = L_{\mathrm{acc}} \times M$. Thus, the sequence witnesses that $u = u_{k+1} \in \tau^* L_{\mathrm{acc}} \cap M$, and since $h(u) = w$, it follows that $w \in h(L \cap M)$. □

Now, we can construct a game acceptor for covering an arbitrary context-free language $L \subseteq \Sigma^*$ represented as $L = h(D_A \cap M)$ according to Theorem 3.8, by applying Lemma 3.9 to the particular case of Dyck languages: we set out with the seed transducer $R_{n,C}$ for the Dyck language D_A over the alphabet $A = B_n \cup C$ and add a coding cycle for the homomorphism h. This yields a transducer R_h over the alphabet $\Sigma \cup (A \cup \{\Box\}) \times A$ such that the game with seed $(R_h, \Box^* \times M)$ covers L.

The generic construction of R_h can be simplified in the case where $R = R_{n,C}$ is the seed transducer of a Dyck language. Notice that, if we start from a word $w \in \Sigma^*$, then every distinct word $w' \neq w$ reached in the iteration $(w, w') \in (R_h R_h^{-1})^*$ consists only of letters of the form (x, x) or $(\Box, x)$ with $x \in B_n \cup C$. Hence, the *reduced* transducer $\hat{R}_h$ obtained from R_h, by restricting to the (subset of transitions labelled with letters in the) subalphabet $\Sigma \cup \{(x, x) \mid x \in A\} \cup \{(\Box, x) \mid x \in A\}$ and identifying each pair $(x, x) \in A \times A$ with x, is equivalent to R_h in the sense that, for every regular language $M \subseteq A^*$, the game with seed $(\hat{R}_h, \Box^* \times M)$ covers the same language over Σ as the one with seed $(R_h, \Box^* \times M)$. In contrast to R_h, however, the reduced transducer $\hat{R}_h$ has fewer transition and a smaller alphabet, which extends the one of the underlying Dyck language D_A only with a *neutralised* copy of each letter in A.

We argue that the shape of the seed constructed above is prototypical for games that cover context-free languages. Therefore, we focus on games with a seed isomorphic to the seed $(\hat{R}_h, \Box^* \times M)$ obtained for the homomorphic image of a Dyck-language over n bracket pairs intersected with a regular language. An *n-flower* transducer is a transducer $R = (Q, \Gamma, \Delta, q_0, F)$ on a set of states $Q = \{q_0, q_1, \ldots, q_n, q_h\}$ with initial state q_0 and final state set $F = \{q_0, q_h\}$, over an alphabet that can be partitioned into $\Gamma = \Sigma \cup B_n \cup C \cup A'$, where B_n is a set of n matching brackets $[_k,]_k$ and A' is a disjoint copy of $A := B_n \cup C$, associating a neutralised variant $\boxed{a}$ to each letter $a \in A$, such that Δ contains copying transitions $q_0 \xrightarrow{a|a} q_0$ for all $a \in A \cup A'$, and $q_k \xrightarrow{\boxed{a}|\boxed{a}} q_k$ for all $a \in A$ and each $k \in \{1, \ldots n\}$, as well as erasing transitions $q_0 \xrightarrow{c|\boxed{c}} q_0$ for each $c \in C$, and $q_0 \xrightarrow{[_k|\boxed{[_k}} q_k$, $q_k \xrightarrow{]_k|\boxed{]_k}} q_0$ for each k. Furthermore, we require that there is a homomorphism $h : A \to \Sigma$, such that



(a) 2-flower

(b) loose flower for palindromes

Figure 3.3: Flower transducers

the remaining transitions of Δ are coding transitions $q_0 \xrightarrow{h(a)|a} q_h$ and $q_h \xrightarrow{h(a)|a} q_h$ for all $a \in A$. Finally, a seed (R, L_{acc}) is an n-flower if R is an n-flower transducer and L_{acc} is a regular language over the alphabet A' of its neutralised symbols. An example of a 2-flower transducer is depicted in Figure 3.3(a).

Theorem 3.9. *A language is context-free if, and only if, it is covered by a consensus game acceptor with an n-flower seed.*

Proof. Let $L \subseteq \Sigma^*$ be an arbitrary context-free language. According to the Representation Theorem 3.8, there exists an alphabet A partitioned into a bracket alphabet B_n and a set of neutral symbols C, a letter-to-letter homomorphism given by $h : A \to \Sigma$, and a regular language $M \subseteq A^*$, such that $L = h(D_A \cap M)$. With these in hand, we construct the seed transducer $R_{n,C}$ for the Dyck language D_A as in Lemma 3.8 and then add a coding cycle q_h for the homomorphism h. According to Lemma 3.9, the transducer R constructed in this way together with the seed language $L_{\text{acc}} := \Box^* \times M$ describes a game that covers L. By reducing R to the alphabet $\Sigma \cup A \cup \{\Box\} \times A$, we finally obtain the n-flower seed seed $(\hat{R}, \Box^* \times M)$ that also covers L over Σ.

For the converse statement, let G be a consensus game with an n-flower seed (R, L_{acc}) over an alphabet Γ. We wish to prove that the language covered by G over Γ is context-free. First, we partition Γ into an alphabet Σ, a set B_n of n matching brackets, a set C of (prime) neutral symbols, and a neutralised copy A' of $A := B_n \cup C$. Let D_A be the Dyck language over A, and let $h : A \to \Sigma$ be the letter-to-letter homomorphism determined by the coding cycle in R. Then, consider the *neutralising* letter-to-letter homomorphism $\nu : A \cup A' \to A'$ which maps both a and $\boxed{a}$ to the neutralised copy $\boxed{a}$, and set $M := \nu^{-1} L_{\text{acc}}$; as an inverse homomorphic image of a regular language, M is regular. By construction of the flower transducer, we know that, over the alphabet Σ, the game G covers the context-free language $h(D_A \cap M)$.

To describe the language covered over the full alphabet Γ, consider the Dyck language D'_A over the alphabet $A \cup A'$ with the same set B_n of brackets as D_A, but with an extended set $C \cup A'$ of neutral symbols. We observe that D'_A is closed under R and R^{-1} in the sense that, for any pair of non-terminal words $w, w' \in \Gamma^* \setminus \Sigma^*$ with $(w, w') \in R$, we have

$w \in D'_A$ if, and only if, $w' \in D'_A$. Moreover, $\nu(w) = \nu(w')$ (letters are neutralised, but never forgotten). This implies that, over $\Gamma \setminus \Sigma$, the game G covers the language $D'_A \cap M$. Since every observation sequence of G is either contained in Σ^* or in $(\Gamma \setminus \Sigma)^*$, it follows that, over the full alphabet Γ, the consensus game G covers the context-free language $h(D_A \cap M) \cup (D'_A \cap M)$. □

Notice that, without restricting the alphabet of the accepting seed language to neutralised symbols, the structure of the transducer alone would not guarantee that the language covered by a game with an n-flower seed is context-free. For instance, the one-flower transducer over a bracket pair $[,]$ one neutral symbol $\#$, and their neutralised copies, together with the seed language $L_{\text{acc}} := [^+ \# \boxed{[}^+]^+$ give rise to a game where the covered language L is not context-free, since the intersection $L \cap \boxed{[}^+ \# [^+]^+ = \boxed{[}^n \# [^n]^n$ is not context-free.

Returning to games, the argument from Lemma 3.7 shows that, given a game with seed $(R, L_{\text{acc}}, L_{\text{rej}})$, at every play π with observation $\beta^1(\pi)$ in the language L_1 covered by (R, L_{acc}) over the full observation alphabet, the only safe decision is 1, whereas at each play with observations in the language L_0 covered by (R, L_{rej}) the only safe decision is 0. An observation-based strategy prescribing $s^1(\pi) := 1$ precisely if $\beta^1(\pi) \in L_1$ induces a joint strategy that is *optimal* in the sense that it prescribes a safe decision whenever one exists. Likewise, a strategy that prescribes 0 precisely at sequences in L_0 is optimal. Optimal strategies are *undominated*, that is, no other strategy wins strictly more plays. Clearly, if a game is solvable, then every optimal strategy is winning.

One consequence of Theorem 3.9 is that, for games where one of (R, L_{acc}) or (R, L_{rej}) is an n-flower, the set L_1 or L_0 is context-free, and therefore recognisable by a nondeterministic push-down automaton, which we can construct effectively from the game description. The obtained automaton hence implements an optimal strategy that is indeed a winning strategy, in case the game is solvable. According to Theorem 3.7, already for games where both L_0 and L_1 are context-free, the question of whether a winning strategy exists amounts to solving the disjointness problem for context-free languages and is hence undecidable. Under these circumstances, it is remarkable that we can effectively construct strategies that are optimal and, moreover, winning whenever the game is solvable.

Corollary 3.1. *For any consensus game G with seed $(R, L_{\text{acc}}, L_{\text{rej}})$ where either (R, L_{acc}) or (R, L_{rej}) is an n-flower, we can effectively construct a nondeterministic push-down automaton $\mathcal{S}$ that implements an optimal strategy.*

3.6 Discussion

We presented a simple kind of games with imperfect information where constructing optimal strategies requires iterating the (synchronous rational) relation that correlates the observation of players. This establishes a correspondence between winning strategies in games on the one hand, and main classes of formal languages on the other hand. The correspondence leads to several insights on games with imperfect information.

Firstly, we obtain simple examples that illustrate the computational complexity of coordination under imperfect information. The classical constructions for proving that the

problem is undecidable in the general case typically involve an unbounded number of non-trivial decisions by which the players describe configurations of a Turing machine [70, 1, 77]. In contrast, our undecidability argument in Theorem 3.5 relies on a single simultaneous decision.

Secondly, we identify families of games where optimal strategies exist and can be constructed effectively, but the complexity of the strategic decision necessarily grows with the length of the play. This opens a new perspective for distributed strategy synthesis that departs from the traditional focus on finite-state winning strategies and from game classes, on which the existence of such is decidable. In consensus games, the implementation of winning strategies requires arbitrary linear-bounded automata in the general case. However, we have also described a structural condition on game graphs that ensures that winning strategies can be implemented by push-down automata.

One challenging objective is to classify games with imperfect information according to the complexity of strategies required for solving them. The insights developed for consensus games allow a few more steps in this direction. For instance, games with one-flower seeds cover one-counter languages and therefore admit optimal strategies implemented by one-counter automata. Likewise, we can build up a variant of n-flower seeds from Dyck languages restricted to palindromes, as illustrated in Figure 3.3(b). Games with such loose n-flower seeds cover a subclass of linear languages and hence admit optimal strategies implemented by one-turn push-down automata.

Chapter 4

Information Hierarchies

One fundamental case in which the distributed synthesis problem becomes decidable is that of hierarchical systems: these correspond to games where there is a total order among the players such that, informally speaking, each player has access to the information received by the players that come later in the order. Peterson and Reif [69] showed that, for games in this setting, it is decidable — although, with non-elementary complexity — whether distributed winning strategies exist and if so, finite-state winning strategies can be effectively synthesised. The result was extended by Pnueli and Rosner [70] to the framework of distributed systems over fixed linear architectures where information can flow only in one direction. Later, Kupferman and Vardi developed a fundamental automata-theoretic approach [53] that allows to extend the decidability result from linear-time to branching-time specifications, and also removes some of the syntactic restrictions imposed by the fixed-architecture setting of Pnueli and Rosner. Finally, Finkbeiner and Schewe [36] give an effective characterisation of communication architectures on which distributed synthesis is decidable. The criterion requires absence of information forks, which implies a hierarchical order in which processes, or players, have access to the observations provided by the environment.

In this chapter, we study a relaxation of the hierarchical information pattern underlying the basic decidability results on games with imperfect information. Firstly, we extend the assumption of hierarchical observation, that is *positional* information, by incorporating perfect recall. Rather than requiring that a player *observes* the signal received by a less-informed player, we will require that he can *deduce* it from his observation of the play history. It can easily be seen that this gives rise to a decidable class, and it is likely that previous authors had a perfect-recall interpretation in mind when describing hierarchical systems, even if the formal definitions in the relevant literature generally refer to observations.

Secondly, we investigate the case when the hierarchical information order is not fixed, but may change dynamically along the play. This lets us model situations, where the schedule of the interaction allows a less-informed player to become more informed than others, or where the players may coordinate on designating one to receive certain signals, and thus become more informed than others. We show that this condition of dynamic hierarchical observation also leads to a decidable class of the distributed synthesis problem.

As a third extension, we consider the case where the condition of hierarchical informa-

tion (based on perfect recall) is intermittent. That is, along every play, it occurs infinitely often that the information sets of players are totally ordered; nevertheless, there may be histories, at which incomparable information sets arise, as it is otherwise typical of information forks. We show that, at least for the case of winning conditions over attributes observable by all players, this condition of recurring hierarchical observation is already sufficient for the decidability of the synthesis problem, and that finite-state winning strategies exist for all solvable instances.

For all three conditions of hierarchical information, it is decidable with relatively low complexity whether they hold for a given game. However, the complexity of solving a game is non-elementary in all cases, as they are more general than the condition of hierarchical observation, known to admit no elementary lower bound [70].

4.1 Finite-State Strategies and Automata

In this chapter, we work on the game model presented in Chapter 2, however we make some precisions on the kind of strategies we are interested here: Our focus is on finitely-represented games, where the game graphs are finite and the winning conditions described by finite-state automata. Specifically, winning conditions are given by a colouring function $\gamma : V \to C$ and an ω-regular set $W \subseteq C^\omega$ describing the set of plays $v_0, v_1, \ldots$ with $\gamma(v_0), \gamma(v_1), \cdots \in W$. In certain cases, we assume that the colouring is *observable* to each player i, that is, $\beta^i(v) \neq \beta^i(v')$ whenever $\gamma(v) \neq \gamma(v')$. For general background on automata for games, we refer to the survey [44].

Strategies shall also be represented as finite-state machines. A *Moore* machine over an input alphabet Σ and an output alphabet Γ is described by a tuple (M, m_0, μ, ν) consisting of a finite set M of *memory states* with an initial state m_0, a memory *update* function $\mu : M \times \Sigma \to M$ and an *output* function $\nu : M \to \Gamma$ defined on memory states. Intuitively, the machine starts in the initial memory state m_0, and proceeds as follows: in state m, upon reading an input symbol $x \in \Sigma$, updates its memory state to $m' := \mu(m, x)$ and then outputs the letter $\nu(m)$. Formally, the update function μ is extended to input words in Σ^* by setting, $\mu(\varepsilon) := m_0$, for the empty word, and by setting,

$$\mu(x_0 \ldots x_{\ell-1} x_\ell) := \mu(\, \mu(x_0 \ldots x_{\ell-1}),\, x_\ell),$$

for all nontrivial words $x_0 \ldots x_{\ell-1} x_\ell$. This gives rise to the function $M : \Sigma^* \to \Gamma^*$ *implemented* by M, defined by $M(x_0, \ldots, x_\ell) := \nu(\, \mu(\, x_0 \ldots x_\ell)\,)$. A *strategy automaton* for Player i on a game G, is a Moore machine M with input alphabet B^i and output alphabet A^i. The strategy implemented by M is defined as $s^i(v_0, \ldots, v_{\ell-1}) := M(\beta^i(v_0 \ldots v_{\ell-1})\,)$. A *finite-state strategy* is one implemented by a strategy automaton.

Sometimes it is convenient to refer to Mealy machines rather than Moore machines. These are finite-state machines of similar format, with the only difference that the output function $\nu : M \times \Sigma \to \Gamma$ refers to transitions rather than their target state (see Definition 2.4).

In the following we will refer to several classes $\mathcal{C}$ of finite games, always assuming that winning conditions are given as ω-regular languages. The *finite-state synthesis problem* for a class $\mathcal{C}$ is the following: Given a game $\mathcal{G} \in \mathcal{C}$,

(i) decide whether $\mathcal{G}$ admits a finite-state distributed winning strategy, and

(ii) if yes, construct a profile of finite-state machines that implements a distributed winning strategy for $\mathcal{G}$.

We refer to the set of distributed (finite-state) winning strategies for a given game $\mathcal{G}$ as the (finite-state) *solutions* of $\mathcal{G}$. We say that the synthesis problem is *finite-state solvable* for a class $\mathcal{C}$ if every game $\mathcal{G} \in \mathcal{C}$ that admits a solution also admits a finite-state solution, and if the above two synthesis tasks can be accomplished for all instances in $\mathcal{C}$.

4.2 Static Hierarchies

One fundamental case in which the distributed synthesis problem becomes decidable is that of hierarchical systems: these correspond to games where there is a total order among the players such that, informally speaking, each player has access to the information received by the players that come later in the order.

Peterson and Reif [69] showed that for games in this setting, it is decidable — although, with non-elementary complexity — whether distributed winning strategies exist and if so, finite-state winning strategies can be effectively synthesised. The result was extended by Pnueli and Rosner [70] to the framework of distributed systems over fixed linear architectures where information can flow only in one direction. Later, Kupferman and Vardi developed a fundamental automata-theoretic approach [53] that allows to extend the decidability result from linear-time to branching-time specifications, and also removes some of the syntactic restrictions imposed by the fixed-architecture setting of Pnueli and Rosner. Finally, Finkbeiner and Schewe [36] give an effective characterisation of communication architectures on which distributed synthesis is decidable. The criterion requires absence of information forks, which implies a hierarchical order in which processes, or players, have access to the observations provided by the environment.

The setting of games is more liberal than that of architectures with fixed communication channels. For instance, Muscholl and Walukiewicz [64] present a decidable class of synthesis problems under different assumptions on the communication between processes that are not subsumed by information-fork free architectures. A rather general, though non-effective condition for games to admit finite-state distributed winning strategies is given in [5], based on epistemic models representing the knowledge acquired by players in a game with perfect recall. This condition suggests that, beyond the fork-free architecture classification there may be further natural classes of games for which the distributed synthesis problem is decidable.

4.2.1 Hierarchical observation

We set out from the basic pattern of hierarchical information underlying the decidability results cited in the introduction [53, 69, 70]. These results rely on a positional interpretation of information, i.e. on observations.

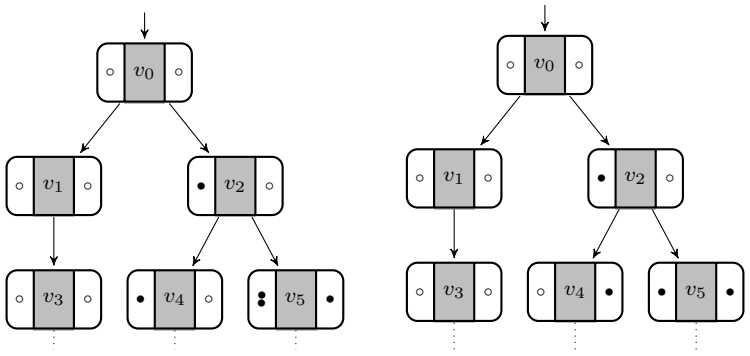
Definition 4.1. *A game graph yields* hierarchical observation *if there exists a total order $\preceq$ among the players such that, whenever $i \preceq j$, then for all pairs v, v' of positions, $\beta^i(v) = \beta^i(v')$ implies $\beta^j(v) = \beta^j(v')$.*

In other words, if $i \preceq j$, then the observation of Player i determines the observation of Player j.

Peterson and Reif [69] study a game with players organised in a hierarchy, such that each player i sees the data observed by Player $i-1$. The setting is actually generic for games with reachability winning conditions and the authors show that winning strategies can be synthesised in n-fold exponential time and this complexity is unavoidable. Later, [70] consider a similar model in the context of distributed systems with linear-time specifications given by finite automata on infinite words. Here, the hierarchical organisation is represented by a pipeline architecture, which allows each process to send signals only to the following one. The authors show that the distributed synthesis problem for such a system, is solvable via an automata-theoretic technique. This technique is further extended by [53] to more general, branching-time specifications. The key operation of the construction is that of *widening* – an interpretation of strategies for a less-informed player j within the strategies of a more-informed player $i \preceq j$. Intuitively, this allows to first solve a game as if all the moves were performed by the most-informed player, which comes first in the order $\preceq$, and successively discard solutions that cannot be implemented by the less-informed players, i.e. those which involve strategies that are not in the image of the widening interpretation.

The automata-theoretic method for solving the synthesis problem on pipeline architectures, due to [70] and [53], can be adapted directly to solve the synthesis problem for games with hierarchical observation.

Theorem 4.1 ([70, 53])**.** *For games with hierarchical observation, the synthesis problem is finite-state solvable.*



(a) hierarchical observation

(b) static hierarchical information

Figure 4.1: Basic patterns of hierarchical information: game positions show the observation of Player 1 (left) and Player 2 (right); the name of the position (middle) is unobservable

4.2.2 Incorporating perfect recall

In a first step, we extend the notion of hierarchical observation to incorporate the power of perfect recall that players have. To do so, we need to take into account not only the observation received by a player at a certain step in the play, but the entire history of observations the player received since the start of the play. We have already seen how the observation functions β^i extend from states to histories, as well as the indistinguishability relations $\sim^i$ on observation histories. The notion of *information sets* of a player designates the equivalence classes of its indistinguishability relation. More precisely, the *information set* of player i at history π is $P^i(\pi) := \{\pi' \in Hist(G) \mid \pi' \sim^i \pi\}$. While maintaining the requirement of a fixed order, we now ask that the *information set* of a player determines the information sets of those who follow in the order.

Definition 4.2. *A game graph yields* (static) hierarchical information *if there exists a total order $\preceq$ among the players such that, for all histories π, if $i \preceq j$, then $P^i(\pi) \subseteq P^j(\pi)$.*

Example 4.1 (Hierarchical observation vs. hierarchical information)**.** In Figure 4.1(a), the game graph yields hierarchical *observation*. Indeed, the observations of the first player determine the observations of the second player. In fact, at every round, the first player observes a different symbol for each reachable state, while the second player does not distinguish between states v_1 and v_2 after the first round, and between v_3 and v_4 after the next round. Clearly, the game graph also yields hierarchical *information*: For instance, after two rounds, the information sets of the first player are singletons, as he can distinguish every history from any other one, while the second player has the following information sets: $P^2(v_0v_1v_3) = P^2(v_0v_2v_4) = \{v_0v_1v_3,\ v_0v_2v_4\}$ and $P^2(v_0v_2v_5) = \{v_0v_2v_5\}$, which correspond to the observation histories $\circ\circ\circ$ and $\circ\circ\bullet$, respectively. On the other hand, the game graph in Figure 4.1(b) yields hierarchical information but not hierarchical observation. Indeed, without the assumption of perfect recall, the first player cannot distinguish between states v_3 and v_4, while the second cannot distinguish between states v_4 and v_5. However, considering observation histories, one can see that the first player is better informed than the second: For instance, after two rounds, his information sets are $P^1(v_0v_1v_3) = \{v_0v_1v_3\}$ corresponding to the observation history $\circ\circ\circ$, $P^1(v_0v_2v_4) = \{v_0v_2v_4\}$ corresponding to the observation history $\circ\bullet\circ$, and $P^1(v_0v_2v_5) = \{v_0v_2v_5\}$ corresponding to the observation history $\circ\bullet\bullet$. The second player information sets are $P^2(v_0v_1v_3) = \{v_0v_1v_3\}$ and $P^2(v_0v_2v_4) = P^2(v_0v_2v_5) = \{v_0v_2v_4,\ v_0v_2v_5\}$, which correspond to the observation histories $\circ\circ\circ$ and $\circ\circ\bullet$, respectively.

The following lemma provides an operational characterisation of the hierarchical information condition. We detail the proof, as its elements will be used later.

Lemma 4.1. *A game graph G yields static hierarchical information if, and only if, for every pair $i \preceq j$ of players, there exists a Moore machine that outputs $\beta^j(\pi)$ on input $\beta^i(\pi)$, for every history π in G.*

Proof. For an arbitrary game graph G, let us denote the relation between the observations of two players i and j along the histories of G by

$$T^{ij} := \{(\beta^i(\pi), \beta^j(\pi)) \in (B^i \times B^j)^* \mid \pi \in \mathrm{Hist}(G)\}.$$

This is a regular relation, recognised by the game graph G when viewed as a finite-word automaton A_G^{ij} over the alphabet of observation pairs $B^i \times B^j$.

Concretely, $A_G^{ij} := (V, B^i \times B^j, v_0, \Delta, V)$ is a nondeterministic automaton on states corresponding to positions of G, with transitions $(v, (b^i, b^j), v') \in \Delta$ if there exists a move $(v, a, v') \in E$ such that $\beta^i(v') = b^i$ and $\beta^j(v') = b^j$; all states are accepting.

($\Leftarrow$) If there exists a Moore machine that recognises T^{ij}, then T^{ij} is actually a function. Thus, $\pi \sim^i \pi'$ implies $\beta^j(\pi) = T^{ij}(\beta^i(\pi)) = T^{ij}(\beta^i(\pi')) = \beta^j(\pi')$, and therefore $\pi \sim^j \pi'$.

($\Rightarrow$) Assuming that G yields static hierarchical information, consider the automaton M^{ij} obtained by determinising A_G^{ij} and trimming the result, that is, removing all states that do not lead to an accepting state. As G yields hierarchical information, the relation T^{ij} recognised by M^{ij} is functional, and hence M^{ij} is deterministic in the input component i: for any state v there exists precisely one outgoing transition along each observation $b^i \in B^i$. In other words, M^{ij} is a Mealy machine, which we can transform into an equivalent Moore machine, as desired. □

Theorem 4.2. *For games with static hierarchical information, the synthesis problem is finite-state solvable.*

Proof. Intuitively, we transform an arbitrary game graph $G = (V, E, \beta)$ with static hierarchical *information* into one with hierarchical *observation*, by taking the synchronised product of G with automata that signal to each player i the observations of all players $j \succeq i$. We shall see that this preserves the solutions to the distributed synthesis problem, for any winning condition on G.

To make the construction precise, let us fix a pair $i \preceq j$ of players, and consider the Moore machine $M^{ij} = (M, m_0, \mu, \nu)$ from the proof of Lemma 4.1, which translates the observations $\beta^i(\pi)$ into $\beta^j(\pi)$, for every history π in G. We define the product $G \times M^{ij}$ as a new game graph with the same sets of actions as G, and the same observation alphabets $(B^k)_{k \neq i}$, except for Player i, for which we expand the alphabet to $B^i \times B^j$ to also include observations of Player j. The new game is over positions in $V \times M$ with moves $((v, m), a, (v', m'))$ if $(v, a, v') \in E$ and $\mu(m, \beta^i(v)) = m'$. The observations for Player i are given by $\beta^i(v, m) = (\beta^i(v), \nu(m))$, whereas they remain unchanged for all other players $\beta^k(v, m) = \beta^k(v)$, for all $k \neq i$.

The obtained product graph is equivalent to the original game graph G, in the sense that they have the same tree unravelling, and the additional components in the observations of Player i (representing observations of Player j, given by the Moore machine M^{ij}) are already determined by his own observation history, so Player i cannot distinguish any pair of histories in the new game that he could not distinguish in the original game. Accordingly, the strategies on the expanded game graph $G \times M^{ij}$ correspond to strategies on G, such that the outcomes of any distributed strategy are preserved. In particular, for any winning condition over G, a distributed strategy is winning in the original game if, and only if, it is winning in the expanded game $G \times M^{ij}$. On the other hand, the (positional) observations of Player i in the expanded game determine the observations of Player j.

By applying the transformation for each pair $i \preceq j$ of players successively, we obtain a game graph that is equivalent to G under every winning condition, and which additionally

yields hierarchical observation. Due to Theorem 4.1, we can thus conclude that, under ω-regular winning condition, the synthesis problem is finite-state solvable for games with static hierarchical information. □

To decide whether a given game graph yields static hierarchical information, the collection of Moore machines according to Lemma 4.1, for all players i, j, may be used as a witness. However, this yields an inefficient procedure, as the determinisation of a functional transducer involves an exponential blowup; precise bounds for such translations are given by Weber and Klemm in [88]. More directly, one could verify that each of the transductions A_G^{ij} relating observation histories of Players i, j, as defined in the proof of Lemma 4.1, is functional. This can be done in polynomial time using, e.g. the procedure described by Béal et al. in [2].

We can give a precise bound in terms of nondeterministic complexity.

Lemma 4.2. *The problem of deciding whether a game yields static hierarchical information is* NLogSpace*-complete.*

Proof. The complement problem — of deciding whether for a given game there exists a pair of players i, j that cannot be ordered in either way — is solved by the following nondeterministic procedure: Guess a pair i, j of players, then check that $i \not\preceq j$, by following nondeterministically a pair of histories $\pi \sim^i \pi'$, such that $\pi \not\sim^j \pi'$; symmetrically, check that $j \not\preceq i$. The procedure requires only logarithmic space for maintaining pointers to four positions while tracking the histories. Accordingly, the complement problem is in NLogSpace, and since the complexity class is closed under complement, our decision problem of whether a game yields static hierarchical information also belongs to NLogSpace.

For hardness, we reduce the emptiness problem for nondeterministic finite automata, known to be NLogSpace-hard [49], to the problem of deciding whether the following game for two players playing on the graph of the automaton yields hierarchical information: Nature chooses a run in the automaton, the players can only observe the input letters, unless an accepting state is reached; if this happens, Nature sends to each player privately one bit, which violates the condition of hierarchical information. Thus, the game has hierarchical information if, and only if, no input word is accepted. □

4.2.3 Signals and game transformations

Functions that return information about the current history, such as those constructed in the proof of Lemma 4.1 will be a useful tool in our exposition, especially when the information can be made observable to some players.

Given a game graph G, a *signal* is a function defined on the set of histories in G, or on the set of observation histories of some player i. We say that a signal $f : \mathrm{Hist}(G) \to C$ is *information-consistent* for Player i if any two histories that are indistinguishable to Player i have the same image under f. A finite-state signal is one implemented by a Moore machine. Any finite-state signal $f : \mathrm{Hist}(G) \to C$ can also be implemented by a Moore machine M^i over the observation alphabet B^i, such that that $M(\pi) = M^i(\beta^i(\pi))$ for every history π. The *synchronisation* of G with a finite-state signal f is the expanded game graph (G, f) obtained by taking the synchronised product $G \times M$, as described in

the proof of Lemma 4.1. In case f is information-consistent for Player i, it can be made *positionally observable* to this player, without changing the game essentially. Towards this, we consider the game graph (G, f^i) that expands (G, f) with an additional observation component $f^i(v)$ for player i at every position v, such that $f(\pi) = f^i(v)$ for each history π that ends at v. The game graph (G, f^i) is *finite-state equivalent* to G, in the sense that every strategy for G maps via finite-state transformations to a strategy for (G, f^i) with the same outcome and vice versa. Indeed, any strategy for G is readily a strategy with the same outcome for (G, f^i) and, conversely, every strategy profile s in (G, f^i) can be synchronised with the Moore machines implementing the signals f^i for each player i, to yield a finite-state strategy profile s' for G with the same outcome as s. In particular, the transformation preserves solutions to the finite-state synthesis problem.

4.3 Dynamic Hierarchies

In this section, we maintain the requirement on the information sets of players to be totally ordered at every history. However, in contrast to the case of static hierarchical information, we allow the order to depend on the history and to change dynamically along a play.

Definition 4.3. *We say that a history π yields* hierarchical information *if the information sets $\{P^i(\pi) \mid i \in N\}$ are totally ordered by inclusion. A game graph G yields* dynamic *hierarchical information if every history yields hierarchical information.*

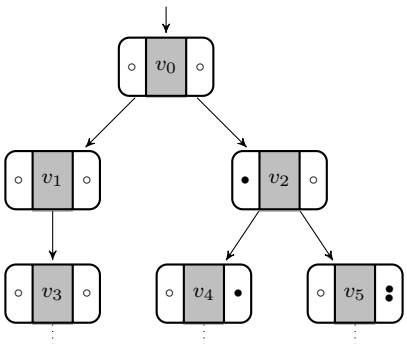


Figure 4.2: Dynamic hierarchical information

Example 4.2. Figure 4.2 shows an example of such a situation: for instance, at the history reaching v_2, player 1 is more informed than player 2, however, the order switches when the play proceeds to position v_4: Indeed, after the history v_0v_2, we have $P^1(v_0v_2) = \{v_0v_2\}$, corresponding to observation history $\circ\bullet$ and $P^2(v_0v_2) = \{v_0v_1,\ v_0v_2\}$ corresponding to observation history $\circ\circ$, hence $P^1(v_0v_2) \subseteq P^2(v_0v_2)$. But after history $v_0v_2v_4$, we have $P^1(v_0v_2v_4) = \{v_0v_2v_4,\ v_0v_2v_5\}$ corresponding to observation history $\circ\bullet\circ$ and $P^2(v_0v_2v_4) = \{v_0v_2v_4\}$ corresponding to observation history $\circ\circ\bullet$ (while $P^2(v_0v_2v_5) = \{v_0v_2v_5\}$ corresponding to observation history $\circ\circ$:), hence $P^2(v_0v_2v_4) \subseteq P^1(v_0v_2v_4)$. Similarly, after

histories $v_0v_1v_3$ and $v_0v_2v_5$, the information sets are totally ordered by inclusion, as we have $P^1(v_0v_1v_3) = P^2(v_0v_1v_3) = \{v_0v_1v_3\}$ and $P^2(v_0v_2v_5) \subseteq P^1(v_0v_2v_5)$, respectively. Therefore, the game graph yields *dynamic* hierarchical information.

In other words, a game has dynamic hierarchical information if there is no history at which the information sets of two players are incomparable. To decide whether this is the case, we can use a nondeterministic procedure similar to the one in Lemma 4.2, to guess two players i, j and three histories $\pi \sim^i \pi'$ and $\pi'' \sim^j \pi$, such that $\pi' \not\sim^i \pi''$ and $\pi' \not\sim^j \pi''$. Since, for a history π, witnesses π', π'' can be guessed and verified by a nondeterministic automaton, it also follows that, for every finite game, the set of histories that yield hierarchical information is regular.

Lemma 4.3. *For every finite game graph G, we can construct a nondeterministic finite automaton such that a history in G is accepted if and only if it does not yield hierarchical information. If G has n players and $|V|$ positions, the number of automaton states is at most $2n^2|V|^2$.*

Proof. Let us fix a game graph G. A history π in G fails to yield hierarchical information if there are two players with incomparable information sets at π. To verify this, we construct an automaton that chooses nondeterministically a pair i, j of players, then, while reading the input π, it guesses a pair π', π'' of histories such that $\pi' \sim^i \pi$ and $\pi'' \sim^j \pi$ and updates two flags indicating whether $\pi' \not\sim^i \pi''$ or $\pi' \not\sim^j \pi''$; the input is accepted if both flags are set. Hence, a word $\pi \in V^*$ that corresponds to a history in G is accepted if, and only if, the corresponding history does not yield hierarchical information. [1]

In its states, the constructed automaton stores the indices of the two players i, j, a pair of game positions to keep track of the witnessing histories π' and π'', and a two-bit flag to record whether the current input prefix is distinguishable from π' for Player j or from π'' for Player i. Clearly, it is sufficient to consider each pair of players only once, hence, the automaton needs at most $4\frac{n(n-1)}{2}|V|^2$ states, that is, less than $2n^2|V|^2$. □

To decide whether a given game graph G yields dynamic hierarchical information, we may check whether the automaton described in Lemma 4.3 accepts all histories in G. However, more efficient than constructing this automaton, we can use a nondeterministic procedure similar to the one of Lemma 4.2 to verify on-the-fly if there exists a history at which the information sets of two players are incomparable: guess two players i, j and three histories $\pi \sim^i \pi'$ and $\pi'' \sim^j \pi$, such that $\pi' \not\sim^i \pi''$ and $\pi' \not\sim^j \pi''$. Obviously, the lower bound from Lemma 4.3 is preserved.

Lemma 4.4. *The problem of deciding whether a game graph yields dynamic hierarchical information is* NLogSpace*-complete.*

In the remainder of the section, we show that, under this more liberal condition, distributed games are still decidable.

[1] Notice that the automaton may also accept words that do not correspond to game histories; to avoid this, we can take the synchronised product with the game graph G and obtain an automaton that recognises precisely the set of histories that do not yield hierarchical information.

Theorem 4.3. *For games with dynamic hierarchical information, the synthesis problem is finite-state solvable.*

For the proof, we transform an arbitrary game $\mathcal{G}$ with dynamic hierarchical information into one with static hierarchical information, among a different set of n *shadow* players $1', \ldots, n'$, where each shadow player i' plays the role of the i-most informed player in the original game, in a sense that we will make precise soon. The information sets of the shadow players follow their nominal order, that is, if $i < j$ then $P^{i'}(\pi) \subseteq P^{j'}(\pi)$. The resulting shadow game inherits the graph structure of the original game, and we will ensure that, for every history π,

(i) each shadow player i' has the same information (set) as the i-most informed actual player, and

(ii) each shadow player i' has the same choice of actions as the i-most informed actual player.

This shall guarantee that the shadow game preserves the winning status of the original game.

The construction proceeds in two phases. Firstly, we expand the game graph G so that the correspondence between actual and shadow players does not depend on the history, but only on the current position. This is done by synchronising G with a finite-state machine that signals to each player his rank in the information hierarchy at the current history. Secondly, we modify the game graph, where the shadow-player correspondence is recorded as a positional attribute, such that the observation of each player is received by his shadow player, at every position; similarly, the actions of each player are transferred to his shadow player. Finally, we show how finite-state winning strategies for the shadow game can be re-distributed to the actual players to yield a winning profile of finite-state winning strategies for the original game.

4.3.1 Information rank signals

For the following, let us fix a game $\mathcal{G}$ with dynamic hierarchical information with the usual notation. For a history π, we write $\preceq_\pi$ for the total order among players induced by the inclusions between their information sets at π. To formalise the notion of an i-most informed player, we use the shortcut $i \approx_\pi j$ for $i \preceq_\pi j$ and $j \preceq_\pi i$; likewise, we write $i \prec_\pi j$ for $i \preceq_\pi j$ and not $j \preceq_\pi i$.

Then, the *information rank* of Player i over the game graph G is a signal $\mathrm{rank}^i : \mathrm{Hist}(G) \to N$ defined by

$$\mathrm{rank}^i(\pi) := |\{j \in N \mid j \prec_\pi i \text{ or } (j < i \text{ and } j \approx_\pi i)\}|.$$

Likewise, we define the *order* of Player i *relative* to Player j as a signal $\preceq^i_j$: $\mathrm{Hist}(G) \to \{0, 1\}$ with $\preceq^i_j(\pi) = 1$ if, and only if, $i \preceq_\pi j$.

Lemma 4.5. *The information rank of each player i and his order relative to any player j are finite-state signals that are information-consistent to Player i.*

Proof. We detail the argument for the rank, the case of relative order is similar and simpler.

Given a game $\mathcal{G}$ as in the statement, let us verify that the signal rank^i is information-consistent, for each player i. Towards this, consider two histories $\pi \sim^i \pi'$ in G, and suppose that some player j does not count for the rank of i at π, in the sense that either $i \prec_\pi j$ or ($i \approx_\pi j$ and $i < j$) — in both cases, it follows that $\pi \sim^j \pi'$, hence $P^j(\pi) = P^j(\pi')$, which implies that j does not count for the rank of i at π' either. Hence, the set of players that count for the rank of Player i is the same at π and at π', which means that $\mathrm{rank}^i(\pi) = \mathrm{rank}^i(\pi')$.

To see that the signal rank^i can be implemented by a finite-state machine, we first build, for every pair i, j of players, a nondeterministic automaton A_i^j that accepts the histories π where $j \prec_\pi i$, by guessing a history $\pi' \sim^i \pi$ and verifying that $\pi' \not\sim^j \pi$. To accept the histories that satisfy $i \approx_\pi j$, we take the product of the automata A_i^j and A_j^i for $i \preceq_\pi j$ and $j \preceq_\pi i$ and accept if both accept. Combining the two constructions allows us to describe, for every player j, an automaton A_j to recognise the set of histories at which j counts for $\mathrm{rank}^i(\pi)$.

Next, we determinise each of the automata A_j and take appropriate Boolean combinations to obtain a Moore machine M^i with input alphabet V and output alphabet $\mathcal{P}(N)$, which upon reading a history π in G, outputs the set of players that count for $\mathrm{rank}^i(\pi)$. Finally we replace each set in the output of M^i by its size to obtain a Moore machine that returns on input $\pi \in V^*$, the rank of Player i at the actual history π in G.

As we showed that rank^i is an information-consistent signal, we can conclude that there exists a Moore machine that inputs observation histories $\beta^i(\pi)$ of Player i and outputs $\mathrm{rank}^i(\pi)$. □

One consequence of this construction is that we can view the signals rank^i and $\preceq_j^i$ as attributes of positions rather than properties of histories. Accordingly, we can assume without loss of generality that the observations of each player i have an extra rank component taking values in N and that the symbol j is observed at history π in this component if, and only if, $\mathrm{rank}^i(\pi) = j$. When referring to the positional attribute $\preceq_j^i$ at v, it is more convenient to write $i \preceq_v j$ rather than $\preceq_j^i$. Figure 4.3 illustrates the view of the relative order $\preceq_j^i$ as a state attribute: at each state v, the observations of player i feature a component set to 1 if $i \preceq_v j$ and 0 otherwise.

4.3.2 Smooth overtaking

As we suggested in the proof outline, each player i and his shadow player, identified by the observable signal rank^i, should be equally informed. To achieve this, we will let the observation of Player i be received by his shadow, in every round of a play. However, since the rank of players, and hence the identity of the shadow, changes with the history, an information loss can occur when the information order between two players, say $1 \prec 2$ along a move is swapped to become $2 \prec 1$ in the next round. Intuitively, the observation received by Player 2 after this move contains one piece of information that allows him to catch up with Player 1, and another piece of information to overtake Player 1. Due to their rank change along the move, the players would now also change shadows. Consequently, the shadow of 1 at the target position, who was previously as (little) informed as Player 2,

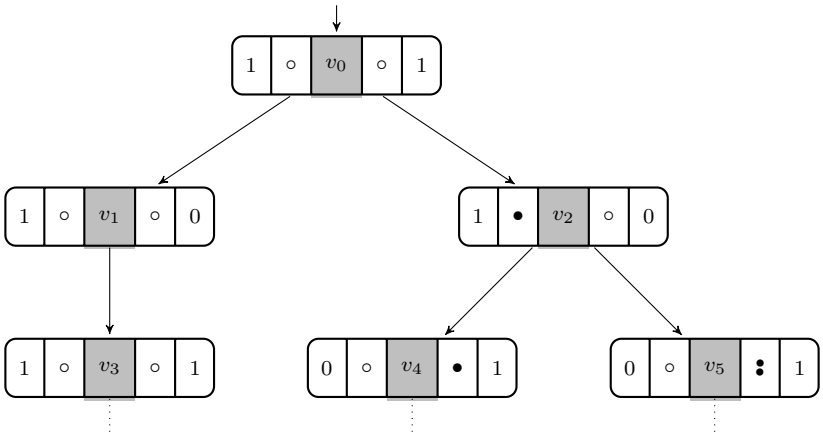
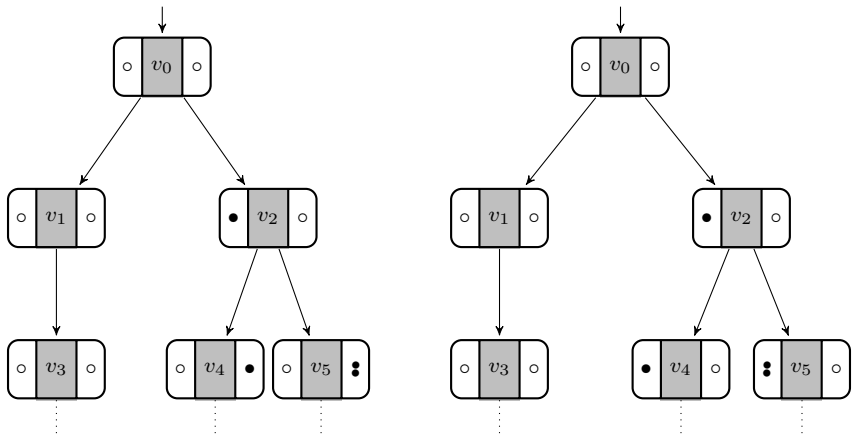


Figure 4.3: Relative order as a state attribute

just receives the new observation of Player 1, but he may miss the piece of information that allowed Player 2 to catch up (and which Player 1 had).



(a) crossing: $1 \prec 2$ at v_2 to $2 \prec 1$ at v_4 (b) "naive shadowing" fails

Figure 4.4: The crossing issue

Figure 4.4(a) pictures such a situation. Indeed, the first player is strictly more informed than the second player after the first round: The information sets of the first player are $P^1(v_0v_1) = \{v_0v_1\}$ and $P^1(v_0v_2) = \{v_0v_2\}$, while the information sets of the second player are $P^2(v_0v_1) = P^2(v_0v_2) = \{v_0v_1, v_0v_2\}$. But after the second round, the second player is *strictly* more informed than the first: The information sets of the first player are $P^1(v_0v_1v_3) = \{v_0v_1v_3\}$ and $P^1(v_0v_2v_4) = P^1(v_0v_2v_5) = \{v_0v_2v_4,\ v_0v_2v_5\}$, while the information sets of the second player are $P^2(v_0v_1v_3) = \{v_0v_1v_3\}$, $P^2(v_0v_2v_4) = \{v_0v_2v_4\}$ and $P^2(v_0v_2v_5) = \{v_0v_2v_5\}$. One desired property of our shadow game is that, at every

history, every shadow player sh_i has the same information set as the original player with information rank i. If we distribute the observations of a player to the corresponding shadow according to his rank at every round of the game, we end up with the situation depicted in Figure 4.4(b). Notice that the second shadow sh_2 gets only $\circ$-observations during the game, since in the first round, the second player had rank 2 and got a $\circ$-observation, and in the second round, the first player has rank 2 and gets $\circ$-observations. Therefore, shadow player sh_2 information set at every history π after the second round consists of all possible histories, namely $P^{sh_2}(\pi) = \{v_0v_1v_3,\ v_0v_2v_4,\ v_0v_2v_5\}$. However, in the original game, even if the first player is the least informed after the second round, his information set is strictly smaller: remember that, in the first round, he could distinguish between the two possible histories, and by perfect recall assumption, this knowledge carries over to the next round, so he can distinguish between paths on the left and right sides of the game graph. The original game and the "naive" shadow construction from Figure 4.4(b) yield games with possibly different information sets, and may have different solutions. The missing piece of information that the second shadow does not get with this direct approach is therefore crucial.

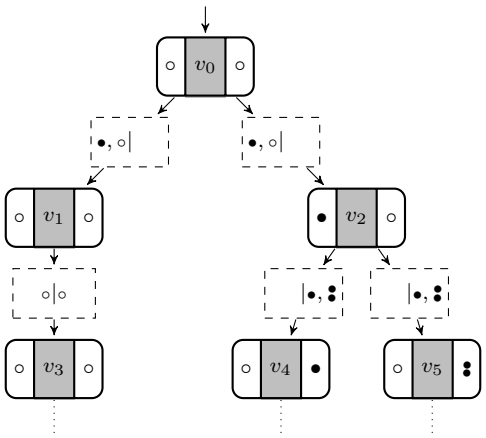


Figure 4.5: Eliminating crossings by introducing half-step lookaheads

To fix this problem, we add a step to our construction, and construct an intermediary finite-state equivalent game to the original, where all changes of hierarchy (or *overtaking* of one player by another) are *smooth*, in the sense that there is a round where these two players share the same information set before there is a strict inclusion again. We call a such a game *cross-free*. The idea is that when a game features only smooth changes in the hierarchy, the shadow players will always receive enough information to catch up with the former better informed players that become less informed. Intuitively, at each round, the smooth-overtaking construction adds a *half-step lookahead* for less informed players that will have overtaken another player in the information order at the end of the round. The knowledge gained by the shadow player is equivalent to this crucial piece of information mentioned above.

Formally, for a play π in a game, we say that Player i and j cross at stage ℓ if

$P^i(\pi_\ell) \subsetneq P^j(\pi_\ell)$ and $P^j(\pi_{\ell+1}) \subsetneq P^i(\pi_{\ell+1})$. We say that a game with dynamic hierarchical information is *cross-free* if there are no crossing players in any play.

Lemma 4.6. *Every game with dynamic hierarchical information is finite-state equivalent to a game that is cross-free.*

Proof. Let G be a game graph with dynamic hierarchical information. We define a signal for each pair of players i, j that represents the knowledge that player j has about the current observation of Player i. If this signal is made observable to Player i only at histories π at which $i \preceq_\pi j$, the game remains essentially unchanged, as players only receive information from less-informed players, which they could hence deduce from their observation. Concretely, we define the signal $\lambda^i_j : V^* \to \mathcal{P}(B^i)$ by

$$\lambda^i_j(\pi) := \{\beta^i(v') \ : \ v' \text{ is the last state of some history } \pi' \in P^j(\pi)\}.$$

Clearly, this is a finite-state signal.

Now we look at the synchronised product of G with the signals $(\lambda^i_j)_{i,j \in N}$ and the relative-order signal $\preceq^i_j$ constructed in the proof of Lemma 4.5. In the resulting game graph, the signal value $\lambda^i_j(\pi)$ at a history π that ends at a position w is represented by the position attribute $\lambda^i_j(w)$. We add to every move (v, a, w) an intermediary position u, at which we assign, for every player i the observation $\{\lambda^i_j(w) \ : \ i \preceq_w j\}$. Intuitively, this can be viewed as a half-step lookahead signal that Player i receives from Player j who may have been more informed at the source position v – thus the signal is not necessarily information-consistent for Player i. Nevertheless, the game remains essentially unchanged after adding the signal, as the players cannot react to the received observation before reaching the target w, at which point the information is readily revealed. On the other hand, along moves at which the information order between players switches, the intermediary position ensures that the players attain equal information. The construction is illustrated in Figure 4.5.

For any game $\mathcal{G}$ on G, we adjust the winning condition to obtain one for the new game graph, by ignoring the added intermediary positions. As the added positions have only one successor, the players have no relevant choice, so any distributed winning strategy for the new game corresponds to one for $\mathcal{G}$ and vice versa. In particular, for ω-regular winning conditions, the construction yields a game with no crossings that is finite-state equivalent to the original game. □

4.3.3 Shadow players

We are now ready to describe the construction of the shadow game associated to a game $\mathcal{G} = (V, E, \beta, W)$ with dynamic hierarchical information. Without loss of generality, we can assume that every position in G is marked with the attributes $\text{rank}^i(v)$ and $\sim^i_j$, for all players i, j according to Lemma 4.5 and that the game graph is cross-free, according to Lemma 4.6.

The shadow game $\mathcal{G}' = (V \cup \{\ominus\}, E', \beta', W)$ is also played by n players and has the same winning condition as $\mathcal{G}$. The action and the observation alphabet of each shadow player consists of the union of the action and observation alphabets of all actual players.

The game graph G' has the same positions as G, plus one sink $\ominus$ that absorbs all moves along unused action profiles. The moves of G' are obtained from G by assigning the actions of each player i to his shadow player $j = \mathrm{rank}^i(v)$ as follows: for every move $(v, a, v') \in E$, there is a move $(v, x, v') \in E'$ labelled with the action profile x obtained by a permutation of a corresponding to the rank order, that is, $a^i = x^j$ for $j = \mathrm{rank}^i(v)$, for all players i. Finally, at every position $v \in V$, the observation of any player i in the original game G is assigned to his shadow player, that is $\beta'^j(v) := \beta^i(v)$, for $j = \mathrm{rank}^i(v)$.

By construction, the shadow game yields static hierarchical information, according to the nominal order of the players. We can verify, by induction on the length of histories, that for every history π, the information set of Player i at π in G is the same as the one of his shadow player $\mathrm{rank}^i(\pi)$ in G'.

Finally, we show that the distributed synthesis problem for G reduces to the one on G', and vice versa. To see that $\mathcal{G}'$ admits a winning strategy if $\mathcal{G}$ does, let us fix a distributed strategy s for the actual players in $\mathcal{G}$. We define a signal $\sigma^j : \mathrm{Hist}(G') \to A$ for each player in $\mathcal{G}'$, by setting $\sigma^j(\pi) := s^i(\pi)$ if $j = \mathrm{rank}^i(\pi)$, for each history π. This signal is information-consistent for Player j, since, at any history π, his information set is the same as for the actual player i with $\mathrm{rank}^i(\pi) = j$, and because the strategy of the actual player i is information-consistent for himself. Hence, σ^j is a strategy for Player j in G'. Furthermore, at every history, the action taken by the shadow player $j = \mathrm{rank}^i(\pi)$ has the same outcome as if it was taken by the actual player i in G. Hence, the set of play outcomes of the profiles s and σ are the same and we can conclude that, if there exists a distributed winning strategy for G, then there also exists one for G'. Notice that this implication holds under any winning condition, without assuming ω-regularity.

For the converse implication, let us suppose that the shadow game $\mathcal{G}'$ admits a winning profile σ of finite-state strategies. We consider, for each actual player i of G, the signal $s^i : \mathrm{Hist}(G) \to A^i$ that maps every history π to the action $s^i(\pi) := \sigma^j(\pi)$ of the shadow player $j = \mathrm{rank}^i(\pi)$. This is a finite-state signal, as we can implement it by synchronising G with rank^i, the observations on the shadow players, and the winning strategies σ^j, for all shadow players j. Moreover, s^i is information-consistent to the actual player i, because all histories $\pi \in P^i(\pi)$, have the same value $\mathrm{rank}^i(\pi) =: j$, and, since s^j is information-consistent for Player j, the actions prescribed by $s^j(\pi)$ must be the same, for all $\pi \in P^j(\pi) = P^i(\pi)$. In conclusion, the signal s^i represents a finite-state strategy for Player i. The profile s has the same set of play outcomes outcome as σ, so s is indeed a distributed finite-state strategy, as desired.

In summary, we have shown that any game G with dynamic hierarchical information admits a winning strategy if, and only if, the associated shadow game with static hierarchical observation admits a finite-state winning strategy. The latter question is decidable according to Theorem 4.2. We showed that for every positive instance G', we can construct a finite-state distributed strategy for G. This concludes the proof of Theorem 4.3.

4.4 Transient Perturbations

As a third pattern of hierarchical information, we consider the case where incomparable information sets may occur at some histories along a play, but it is guaranteed that a total

order will be re-established in a finite number of rounds.

Definition 4.4. *A play yields* recurring hierarchical information *if there are infinitely many histories that yield hierarchical information. A game yields* recurring hierarchical information *if all its plays do so.*

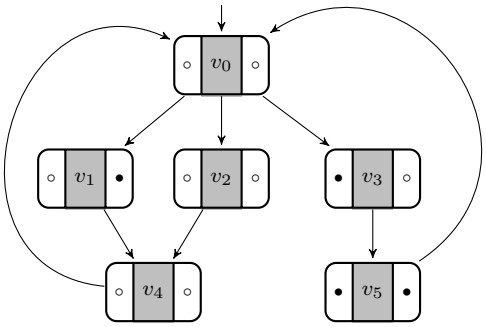


Figure 4.6: Recurring hierarchical information

Example 4.3. Figure 4.6 shows an example of such a situation: for instance, at the history v_0v_2, the information sets of the two players are incomparable, as $P^1(v_0v_2) = \{v_0v_1,\ v_0v_2\}$ and $P^2(v_0v_2) = \{v_0v_2,\ v_0v_3\}$. However, after the next round, we have $P^1(v_0v_2v_4) = P^1(v_0v_1v_4) = \{v_0v_1v_4,\ v_0v_1v_5\}$ corresponding to observation history $\circ\circ\circ$ and $P^2(v_0v_1v_4) = \{v_0v_1v_4\}$ corresponding to observation history $\circ\bullet\circ$, but $P^2(v_0v_2v_4) = \{v_0v_2v_4\}$ corresponding to observation history $\circ\circ\circ$, hence $P^2(v_0v_2v_4) \subseteq P^1(v_0v_2v_4)$. Similarly, $P^2(v_0v_1v_4) \subseteq P^1(v_0v_1v_4)$. Furthermore, after history $v_0v_3v_5$, we have $P^1(v_0v_3v_5) = P^1(v_0v_3v_5) = \{v_0v_3v_5\}$ with observation history $\circ\bullet\bullet$ for the first player and $\circ\circ\bullet$ for the second player. Thus, after the second round, every history yields hierarchical information, even if the pattern may have been perturbed at the previous round. The play then proceeds back to state v_0, leaving the hierarchy unchanged, and after that can again yield uncomparable information sets, that will be comparable at the next round, and so on.

Since the set of histories that yield hierarchical information is regular in any finite game, according to Lemma 4.3, it follows that the set of plays that yield recurring hierarchical information is ω-regular as well.

Lemma 4.7. *For every finite game, we can construct a deterministic Büchi automaton a play in G is accepted if, and only if it yields recurring hierarchical information. If G has n players and $|V|$ positions, the number of automaton states is bounded by $2^{O(n^2|V|^2)}$.*

Proof. Given a game graph G, we can construct a nondeterministic finite automaton A such that a history in G is accepted if and only if it does not yield hierarchical information,

as in Lemma 4.3. By determinising the automaton A via the standard powerset construction, and then complementing the set of accepting states, we obtain a deterministic automaton $A^{\complement}$ with at most $2^{2n^2|V|^2}$ states that accepts a word $\pi \in V^*$ if either $\pi \notin \mathrm{Hist}(G)$, or π represents a history in G that yields hierarchical information. Finally, we take the synchronised product B of $A^{\complement}$ with the graph G. If we now view the resulting automaton as a Büchi automaton, which accepts an infinite words if infinitely many prefixes are accepted by B, we obtain a deterministic automaton that recognises the set of plays in G that yield hierarchical information. The number of states in B is at most $|V|\,2^{2n^2|V|^2}$, hence bounded by $2^{O(n^2|V|^2)}$. □

The automaton construction provides an important insight about the number of consecutive rounds in which players may have incomparable information. Given a play π on a game graph G, we call a *gap* any interval $[t, t+\ell]$ of rounds such that the histories of π in any round of $[t, t+\ell]$ do not yield hierarchical information; the length of the gap is $\ell+1$. The game graph has *gap size* k if the length of all gaps in its plays is uniformly bounded by k.

Clearly, every game graph with finite gap size yields recurring hierarchical information. Conversely, the automaton construction of Lemma 4.7 implies, via a standard pumping argument, that the gap size of any game graph with recurring hierarchical information is at most the number of states in the constructed Büchi automaton. In conclusion, a game G yields recurring hierarchical information if and only if, the size of a gap in any play of G is bounded by $2^{O(n^2|V|^2)}$.

Corollary 4.1. *If a game yields recurring hierarchical information, then its gap size is bounded by $2^{O(n^2|V|^2)}$, where n is the number of players and $|V|$ is the number of positions.*

A family of game graphs where the gap size grows exponentially with the number of positions is illustrated in Figure 4.7. The example is adapted from [4]: There are two players with no relevant action choices and they can observe one bit, or a special symbol that identifies a unique sink position $v_\bullet$. The family is formed of graphs $(G_m)_{m\geq 1}$, each constructed of m disjoint cycles $(C^r)_{1\leq r\leq m}$ of lengths $p_1, p_2, \ldots, p_m$ corresponding to the first m prime numbers, respectively. We number the positions on the cycle corresponding to the r-th prime number as $C^r := \{c^r_0, \ldots, c^r_{p_r-1}\}$. On each cycle, both players receive the same observation 0. Additionally, there are two special positions v_{01} and v_{10}, that yield different observations to the players: $\beta^1(v_{01}) = \beta^2(v_{10}) = 0$ and $\beta^1(v_{01}) = \beta^2(v_{10}) = 1$. From the initial position v_0 of a game graph G_m, Nature can choose a cycle C^r with $r \leq m$. From each position $c^r_\ell \in C^r$, except for the last one with $\ell = p_r - 1$, there are is a move to the subsequent cycle position $c^r_{\ell+1}$ and, additionally, to v_{01} and v_{10}. In contrast, the last cycle position $c^r_{p_r-1}$ has only the first position c^r_0 of the same cycle as a successor. From the off-cycle positions v_{01} and v_{10} the play proceeds to the unique sink state $v_\bullet$ that emits the special observation $\bullet$ to both players.

For instance, consider the game G_2 and a play π cycling infinitely through C^2. After the corresponding history of length 4, the actual state is c^2_2 and both players observe $\circ$. However, the first player cannot distinguish c^2_2 from c^1_0, and v_{01}. Similarly, the second player cannot distinguish c^2_2 from c^1_0, and v_{10}. Their information sets are $P^1(v_0c^2_0c^2_1c^2_2) = \{v_0c^2_0c^2_1c^2_2,\ v_0c^1_0c^1_1c^1_0,\ v_0c^2_0c^2_1v_{01},\ v_0c^1_0c^1_1v_{01}\}$ and

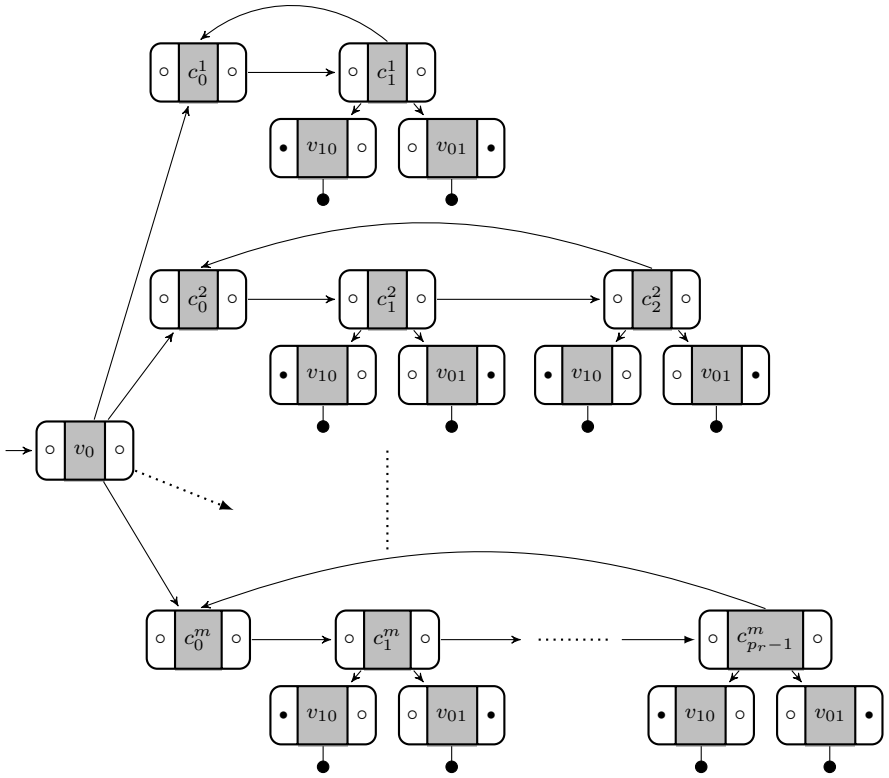


Figure 4.7: Game with an exponential gap of non-hierarchical information (for better readability, the positions v_{01}, v_{10}, and $v_\bullet$ are multiply represented)

$P^2(v_0c_0^2c_1^2c_2^2) = \{v_0c_0^2c_1^2c_2^2,\ v_0c_0^1c_1^1c_0^1,\ v_0c_0^2c_1^2v_{10},\ v_0c_0^1c_1^1v_{10}\}$, and players have incomparable information. But after the corresponding history of length 8, $P^1(v_0c_0^2c_1^2c_2^2c_0^2c_1^2c_2^2c_1^2) = P^2(v_0c_0^2c_1^2c_2^2c_0^2c_1^2c_2^2c_1^2) = \{v_0c_0^2c_1^2c_2^2c_0^2c_1^2c_2^2c_1^2\}$, as there is no transition to v_{01} or v_{10} from c_0^1 nor from c_0^2.

Now, we can verify, for each game graph G_m, that in every play π that proceeds only through cycle positions, the information sets of the two players are comparable at a prefix history of length $t > 2$ in π, if and only if, all the first m primes divide $t-2$; and that any play that leaves a cycle reaches the sink $v_\bullet$, where the information sets of both players coincide. Accordingly, G_m yields recurring hierarchical information. On the other hand, since the product of the first m primes is exponential in their sum, (for a more precise analysis, see [4]), we can conclude that the gap size of the game graphs G_m grows exponentially with the number of positions.

As a further consequence of the automaton construction in Lemma 4.7, it follows that we can decide whether a game graph G yields recurring hierarchical information, by constructing the corresponding Büchi automaton and checking whether it accepts all plays in G. However, due to the exponential blow-up in the determinisation of the automaton, this straightforward approach would require exponential time (and space) in the size of the game graph and the number of (pairs of) players. Here, we describe an on-the fly

procedure that yields better complexity bounds.

Theorem 4.4. *The problem of deciding whether a game graph yields recurring hierarchical information is* PSPACE*-complete.*

Proof. We describe a nondeterministic procedure for verifying that an input game G does *not* yield recurring hierarchical information, that is, there exists a play in G such that from some round t onwards, no prefix of length $\ell \geq t$ yields hierarchical information. By looking at the deterministic Büchi automaton B constructed in Lemma 4.7, we can tell that this is the case if, and only if, there exists a finally periodic play $\pi = \tau\rho^\omega \in V^\omega$ such that the run of B on π visits only non-accepting states after reading the prefix τ and, moreover, it returns to a previously visited state when, but not earlier than, reaching the prefix $\tau\rho$. In other words, the run on $\tau\rho$ induces a *lasso* in B, and hence, the length of $\tau\rho$ is bounded by the number of states $|V|\, 2^{2n^2|V|^2}$ in the automaton.

The idea of the procedure, pictured in Algorithm 1, is to guess such a history $\tau\rho$, and to keep track of the states visited in the corresponding run of the automaton B, or, more precisely, in the powerset construction of the automaton A from Lemma 4.3.

Let us first fix a pair of players i, j. Then, every state reachable by A upon reading the prefix π of an input word from V^ω is described by a tuple (u, c, w, d) consisting of a pair of game positions u, w and two binary flags c, d such that there exist histories π', π'' in G that end at u and w, satisfying $\pi \sim^i \pi'$, $\pi \sim^j \pi''$, and the flags c or d are set if, and only if, $\pi'' \not\sim^i \pi$ or $\pi \not\sim^j \pi'$, respectively. Essentially, this means that π', π'' are candidates for witnessing that players i, j have incomparable information at some continuation of π. Every state in the automaton B on the powerset of A records a set $Z_{ij} \in (V \times \{0,1\})^2$ of such tuples, each collecting the terminal positions of all witness candidates for i, j flagged correspondingly – we call such a set a *cell*. At the beginning, the procedure guesses one cell $\hat{Z}_{ij}$ for every pair of players i, j (Line 1); we call such a collection of cells a *configuration*. The guessed configuration stands for a state of B that shall be reached after reading τ and to which the run returns at $\tau\rho$. Then, starting from the initial configuration, where all players just see the initial position v_0 (Line 3), the procedure generates successively game positions of τ while updating the current configuration to simulate the run of B on τ. The current configuration Z summarises the possible runs of A on the prefix of τ generated so far, and it is updated for every new game position v according to Procedure Update. Once the configuration $Z = \hat{Z}$ is reached, the procedure enters a new loop. Here, it verifies in every iteration that the current configuration indeed contains a witness of incomparability for the input history, that is, there exists, in the cell of some pair of players, a state of A in which both distinguishability flags are set (Line 8). Provided the test succeeds, the procedure successively guesses the positions of ρ while updating the current configuration, until $\hat{Z}$ is reached again, in which case the procedure accepts. Apart of the case when the test in Line 8 fails, the procedure also rejects by looping.

Correctness and soundness The procedure mainly requires space to store the configurations $Z, \hat{Z}$ that collect a set of tuples from $(V \times \{0,1\})^2$ for each pair of players, hence it runs in polynomial space. To show correctness, we consider the sequence π of positions v generated in Lines 5 and 9, and argue that it forms a history in G and that, at every iteration of the loops in Lines 4 and 10, the configuration Z contains, in each cell Z_{ij}, precisely

the set of pairs u, w of terminal positions reachable by candidate witnesses $\pi' \sim^i \pi$ such that $\pi'' \sim^j \pi$, with associated flags c, d indicating correctly whether $\pi' \sim^j \pi''$ and $\pi'' \sim^i \pi$, due to the way they are maintained in Lines 4 and 5 of Procedure Update. Accordingly, if the procedure accepts, then there exists a finally periodic play $\pi := \tau\rho^\omega$ in G such that the information sets of at least two players are incomparable, in every round from τ onwards. Soundness follows from the construction of the Büchi automaton in Lemma 4.7 and the observation that every run of the procedure corresponds to a run of the automaton with the same acceptance status.

Hardness Finally, we argue that the problem of deciding whether a game graph yields recurring hierarchical information is PSPACE-hard by reduction from the universality problem for nondeterministic finite automata, shown by [61] to be PSPACE-hard.

Given a nondeterministic automaton $A = (Q, \Sigma, \Delta, q_0, F)$ over an alphabet Σ, we construct a game graph G for two players with no action choices and with observations in $B^1 = B^2 = \Sigma \times \{0, 1\}$ corresponding to input letters for A tagged with one bit. The set of positions in G contains q_0 and all letter-state pairs $(a, q) \in \Sigma \times Q$. From every state (a, q) and from $q = q_0$, there is a move to position (a', q') if $(q, a', q') \in \Delta$. Each position (a, q) yields the same observation $(a, 0)$ to both players (the observation at q_0 is irrelevant). Thus, every run of A on a word $\alpha \in \Sigma^*$ corresponds to a unique history in G where both players observe the letters of α tagged with 0. In addition, G has two fresh positions v_a and v'_a, for every letter $a \in A$, with observations $\beta^1(v_a) = \beta^2(v'_a) = (a, 1)$ whereas $\beta^1(v'_a) = \beta^2(v_a) = (a, 0)$. Whenever there is a move in G from a position v to some position (a, q) with $q \in F$, we also allow a move from v to v_a. These fresh positions have one common successor, identified by a distinct observations to both players (essentially, indicating that the game is over). Clearly, G can be constructed from A in polynomial time.

Now, consider a nontrivial word $\alpha \in \Sigma^*$ and suppose that it admits an accepting run in A. At the corresponding history π in G, which yields the letters of α tagged with 1 as an observation to both players, the information sets are incomparable, because each player considers it possible that the other received the last letter with a 0-tag. In contrast, if α is rejected, the information sets at the corresponding history in G coincide, hence we have hierarchical information. In conclusion, the language of the automaton A is universal if, and only if, the constructed game graph yields hierarchical information. □

We can show that the synthesis problem for the class of games with recurring hierarchical information is finite state-solvable, at least in the case when the winning conditions are observable. We conjecture that the result extends to the general case.

Theorem 4.5. *For games with recurring hierarchical information and observable ω-regular winning conditions, the synthesis problem is finite-state solvable.*

Proof. The argument relies on the tracking construction decribed in [5], which reduces the problem of solving distributed games with imperfect information for n players against Nature to that of solving a zero-sum game for two players with perfect information. The construction proceeds via an unravelling process that generates epistemic models of the player's information along the rounds of a play, and thus encapsulates their uncertainty.

Algorithm 1: Deciding recurring hierarchical information

Data: game graph $G = (V, E, \beta)$ for n players
Result: accept if G does not yield recurring hierarchical information

type *cell*: subset of $(V \times \{0,1\})^2$
type *configuration*: matrix of *cells* $(Z_{i,j})_{1 \le i < j \le n}$
var $Z, \hat{Z}$: configurations
var $v, \hat{v}$: positions in V
var i, j: players in $\{1, \dots, n\}$

1 **guess** $(\hat{v}, \hat{Z})$ // fix target on cycle
2 $v \leftarrow v_0$
3 **foreach** i, j *with* $i < j$ **do** $Z_{i,j} \leftarrow \{(v_0, 0, v_0, 0)\}$ // initial configuration
4 **while** $(v, Z) \neq (\hat{v}, \hat{Z})$ **do**
5 | **guess** $v \in vEA$ // guess next history state
6 | $Z \leftarrow \mathrm{Update}(v, Z)$ // follow powerset construction
end
7 **repeat**
8 | **if** $\bigwedge_{i,j} Z_{i,j} \cap (V \times \{1\})^2 = \emptyset$ **then reject** // hierarchical information
9 | **guess** $v \in vEA$
10 | $Z \leftarrow \mathrm{Update}(v, Z)$
until $(v, Z) = (\hat{v}, \hat{Z})$ // cycle found
11 **accept**

This process described as "epistemic unfolding" in the paper [5, Section 3] is outlined as follows. An *epistemic model* for a game graph G with the usual notation, is a Kripke structure $\mathcal{K} = (K, (Q_v)_{v \in V}, (\sim^i)_{1 \le i \le n})$ over a set K of histories of the same length in in G, equipped with predicates Q_v designating the histories that end in position $v \in V$ and with the players' indistinguishability relations $\sim^i$. The construction keeps track of how the knowledge of players about the actual history is updated during a round, by generating for each epistemic model $\mathcal{K}$ a set of new models, one for each assignment of an action profile a_k to each history $k \in K$ such that the action assigned to any player i is compatible with his information, i.e. for all $k, k' \in K$ with $k \sim^i k'$, we have $a^i_k = a^i_{k'}$. The update of a model $\mathcal{K}$ with such an action assignment $(a_k)_{k \in K}$ leads to a new, possibly disconnected epistemic model $\mathcal{K}'$ over the universe

$$K' = \{k a_k w \mid k \in K \cap Q_v \text{ and } (v, a_k, w) \in E\},$$

with predicates Q_w designating the histories $k a_k w \in K'$, and with $k a_k w \sim^i k' a_k w'$ whenever $k \sim^i k'$ in $\mathcal{K}$ and $w \sim^i w'$ in G. By taking the connected components of this updated model under the coarsening $\sim := \bigcup_{i=1}^n \sim^i$, we obtain the set of epistemic successor models of $\mathcal{K}$ in the unfolding. The tracking construction starts from the trivial model that consists only of the initial position of the game G. By successively applying the update, it unfolds a tree labelled with epistemic models, which corresponds to a game graph G'

Procedure Update(v, Z)

Data: new position v, current configuration Z
Result: successor configuration after observing $\beta(v)$

foreach i, j **do**
  foreach $(u, c, w, d) \in Z_{i,j}$ **do**
    foreach $u' \in uEA, w' \in wEA$ *with* $\beta^i(u') = \beta^i(v)$ *and* $\beta^j(w') = \beta^j(v)$ **do**
      $c' \leftarrow c \vee \beta^j(u') \neq \beta^j(w')$ `// flag witness for` $j \preceq i$
      $d' \leftarrow d \vee \beta^i(u') \neq \beta^i(w')$ `// flag witness for` $i \preceq j$
      $Z'_{i,j} \leftarrow Z'_{i,j} \cup \{(u', c', w', d')\}$
    end
  end
end
return Z'

for two players with perfect information where the strategies of one player translate into distributed strategies in G and vice versa. According to [5, Theorem 5], for any winning condition, a strategy in $\mathcal{G}'$ is winning if and only if the corresponding joint strategy in $\mathcal{G}$ is so.

The construction can be exploited algorithmically if the perfect-information tracking of a game can be folded back into a finite game. A homomorphism from an epistemic model $\mathcal{K}$ to $\mathcal{K}'$ is a function $f : K \to K'$ that preserves the state predicates and the indistinguishability relations, that is, $Q_v(k) \Rightarrow Q_v(f(k))$ and $k \sim^i k' \Rightarrow f(k) \sim^i f(k')$. The main result of [5] shows that, whenever two nodes of the unfolded tree carry homomorphically equivalent labels, they can be identified without changing the (winning or losing) status of the game [5, Theorem 9]. This holds for all imperfect-information games with ω-regular winning conditions that are observable. Consequently, the strategy synthesis problem is decidable for a class of such games, whenever the unravelling process of any game in the class is guaranteed to generate only finitely many epistemic models, up to homomorphic equivalence.

Game graphs with recurring hierarchical information satisfy this condition. Firstly, for a fixed game, there exist only finitely many epistemic models, up to homomorphic equivalence, where the $\sim^i$-relations are totally ordered by inclusion [5, Section 5]. In other words, epistemic models of bounded size are sufficient to describe all histories with hierarchical information. Secondly, by Corollary 4.1, from any history with hierarchical information, the (finitely branching) tree of continuation histories with incomparable information is of bounded depth, hence only finitely many epistemic models can occur in the unravelling. Overall, this implies that every game with recurring hierarchical information and observable winning condition has a finite quotient under homomorphic equivalence. According to [5, Theorems 9 and 11], we can conclude that the distributed strategy problem for the class is finite-state solvable. □

We point out that the number of hierarchic epistemic models in games, and thus the complexity of our synthesis procedure, grows non-elementarily with the number of

players. This should not come as a surprise, as the solution applies in particular to games with hierarchical observation under safety or reachability conditions (here, the distinction between observable and non-observable conditions is insubstantial), and it is known that already in this case no elementary solution exists (see [69, 1]).

4.5 Monitored Architectures

Finally, we discuss the links between games on graphs and the classical model of communication architectures of [70]. We presented new decidable classes for distributed synthesis on games on graphs with imperfect information. Notice that in the framework of communication architectures, only the information-flow pattern that corresponds to what we call hierarchical *observation* yields decidable distributed synthesis instances. The communication infrastructure of a game is more flexible than in communication architectures: This can be partly explained by the fact that observations, or communication signals between players, or from Nature to players, are embedded in the game graph. Therefore, the range of observations that a player can receive can vary throughout a play depending on the current state of the game. In the classical communication architectures model, a system is represented as a directed graph, where nodes represent processes and edges represent communication channels (see Section 2.1.2 for a formal definition of communication architectures), and the global state of the system is absent from the representation.

In addition to this change of perspective, the base assumption that the communication channels and the range of signals they can carry at each step of the execution are fixed establishes a crucial discrepancy between the two models. For instance, consider a distributed system where the environment sends exclusive information to an agent, then switches to a different agent as the exclusive recipient of information, and where agents have the ability to communicate so that they maintain a hierarchy of information among themselves. It is easy to model this situation with a game that will feature the dynamic hierarchical information pattern. However, any attempt to model this situation via communication architectures is bound to fail: As the environment sends information to two agents implies two communication channels directed to two different processes, the architecture involves an information fork, which is a criterion for undecidability of the distributed synthesis problem. Furthermore, the hierarchical information patterns we presented in this chapter are properties of the game graph, thus structural properties. As suggested with the previous example, such relaxed hierarchical patterns for communication architectures are not enforceable at the architecture level.

With the implicit communication structure of games on graphs, the focus is set on the interaction of players, but their individuality is left aside. Indeed, there may be instances of games modelling distributed systems, where two players interact locally while ignoring a third one, or teams of players, each responsible for independent tasks. However, these organisational specificities remain hardly extractable from the game graph itself. On the other hand, communication architectures have the advantage of displaying the connections among the players explicitly.

In order to model distributed systems in a way that combines the game graph flexibility and the architectures explicitness of the communication infrastructure, one approach is to consider the variant of *monitored* architectures. The idea is to add a *monitoring* layer to

the architecture, in a form of a specification of the communication infrastructure, called a *view monitor*, introduced in [7], that will take care of the refined distribution of signals that is provided by the graph structure in the game model. Intuitively, this monitor, modelled as a Mealy machine with the set of global actions of the processes as input alphabet and the product of observations alphabets of the processes as output alphabet, decides on passing along or withholding any information that transits through it, either from the environment or from processes, in order to guarantee a hierarchy of information at all times, avoiding information forks in the architecture. Thus, at each execution step, the view monitor transforms a global action into a profile of observations to be distributed to the processes. Furthermore, processes are modelled as finite-state automata that are equipped with partial transition functions that allow to enable or disable actions depending on the current local state and observation: this possibility to disable actions allows to restrict at the design level the set of observations that can be generated after certain histories, accounting for the influence of the position in the game graph on the available successors and their corresponding observation profiles in the game setting.

It turns out that monitored architectures can be seen as a syntactic variant of games, as it is possible to translate back and forth instances of the distributed synthesis problem between the game and monitored architectures models, while preserving the set of finite-state solutions. This correspondence allows to recover the result of Theorem 4.1 as a direct consequence of the correspondence between games and monitored architectures. This section is meant an informal presentation of monitored architectures, thus we leave out the formal arguments and technical details that justify this correspondence between the two models, that have been developed in [7].

4.6 Discussion

The bottom-line message of our investigation is that the principle of ordered information flow can afford some flexibility. Still, this might not open floodgates for natural applications to automated synthesis under imperfect information. Rather than expecting information in a real-world system to respect a total order, we see applications in high-level synthesis towards systems on which hierarchical information patterns are enforced to allow for further refinement.

One possible scenario is inspired from multi-level synthesis as proposed in [48] for program repair. Here, the objective is to synthesise a system in several steps: firstly, construct a high-level strategy for a system prototype, in which only a subset of actions is controllable or/and not all observations are reliable, and subsequently refine this strategy to fulfil further specifications, by controlling more actions or relying on more observations.

For our concrete setting, the first-level synthesis problem can be formulated as follows: given an arbitrary distributed game, determine whether it admits a distributed finite-state winning strategy such that the synchronised product with the original game yields a residual game graph with hierarchical information; if possible, construct one. For the next level, the residual game graph can then be equipped with another winning condition, and the actions or observations may be refined. In either case, the condition of hierarchical information enforced by the first-level procedure is in place and guarantees decidability of the synthesis problem, for each subsequent level.

It can be easily seen that, for any arbitrary graph game, the set of strategies that maintain dynamic hierarchical information is regular. In this case, the multi-level synthesis approach can hence be combined with existing automata-theoretic methods. Unfortunately, this would not work out when the objective is to synthesise a graph with recurring hierarchical information; already the problem of eventually attaining dynamic hierarchical information is undecidable.

Finally, a promising approach towards handling coordination problems under imperfect information is proposed in recent work of Genest, Katz, Peled and Schewe [42, 52], in which strategies are viewed by separating the control and communication layers. The shadow game in our reduction of dynamic to static hierarchical information can be understood as an instance of this idea, with the scheduling of shadow players corresponding to a communication layer, and the actual execution of their strategy (as in the static hierarchical game), to the control layer.

Chapter 5

Delayed Signals

Appropriate behaviour of an interactive system component often depends on events generated by other components. The ideal situation, in which perfect information is available across components, occurs rarely in practice – typically a component only receives signals more or less correlated with the actual events. Apart from imperfect signals generated by the system components, there are multiple other sources of uncertainty due to actions of the system environment or to unreliable behaviour of the infrastructure connecting the components: For instance, communication channels may delay or lose signals, or deliver them in a different order than they were emitted. Coordinating components with such imperfect information to guarantee optimal system runs is a significant, but computationally challenging, problem, in particular when the interaction is of infinite duration. It appears worthwhile to study the different sources of uncertainty in separation rather than as a global phenomenon, to understand their computational impact on the synthesis of multi-component systems.

In this chapter, we consider interactive systems modelled by concurrent games among multiple players with imperfect information over finite state-transition systems, or labelled graphs. Each state is associated to a stage game in which the players choose simultaneously and independently a joint action, which triggers a transition to a successor state and generates a local payoff and possibly further private signals to each player. Plays correspond to infinite paths through the graph and yield, to each player, a global payoff according to a given aggregation function, such as mean payoff, limit superior payoff, or parity. As solutions to such games, we are interested in synthesising Nash equilibria in pure strategies, i.e. profiles of deterministic strategies that are self-enforcing when prescribed to all players by a central coordinator.

We study the effect of imperfect, delayed monitoring on equilibria in concurrent games. Towards this, we first introduce a refined game model in which observations about actions are separated from observations about states, and we incorporate a representation for nondeterministic delays for observing action signals. To avoid the general undecidability results from the basic setting, we restrict to the case where the players have perfect information about the current state.

Our main result is that, under the assumption that the delays are uniformly bounded, every equilibrium payoff in the variant of a game where signals are delivered instantly is preserved as an equilibrium payoff in the variant where they are delayed. To prove this,

we construct strategies for the delayed-monitoring game by combining responses for the instant-monitoring variant in such a way that any play with delayed signals corresponds to a shuffle of several plays with instant signals, which we call threads. Intuitively, delayed-monitoring strategies are constructed, in a Frankenstein manner, from a collection of instant-monitoring equilibrium strategies. Under the additional assumption that the payoff structure is insensitive to shuffling plays, this procedure allows to transfer equilibrium payoffs from the instant to the delayed-monitoring game.

Firstly, the transfer result can be regarded as an equilibrium existence theorem for games with delayed monitoring based on classes of games with instant monitoring that admit equilibria in pure strategies. Defining existence conditions is a fundamental prerequisite to using Nash equilibria as a solution concept. If an application model leads to games that may not admit equilibria, this is a strong reason to look for another solution concept. As mixed strategies are conceptually challenging in the context of infinite games, guarantees for pure equilibrium existence are particularly desirable.

Secondly, our result establishes an outcome equivalence between games with instant and delayed monitoring, within the given restrictions: As the preservation of equilibrium values from delayed-monitoring games to the instant-monitoring variant holds trivially (the players may just buffer the received signals until an admissible delay period passed, and then respond), we obtain that the set of pure equilibrium payoffs is the same, whether signals are delayed or not — although, of course, the underlying equilibrium strategies differ between the two variants. In terms of possible equilibrium payoffs, these games are hence robust under changing signalling delivery guarantees, as long as the maximal delays are commonly known. In particular, payoff-related results obtained for the instant-signalling variant apply directly to the delayed variant.

Thirdly, the transfer procedure has some algorithmic content. When we set out with finite-state equilibrium strategies for the instant-monitoring game, the procedure will also yield a profile of finite-state strategies for the delayed-monitoring game. Hence, the construction is effective, and can be readily applied to cases where synthesis procedures for finite-state equilibria in games with instant monitoring exist.

5.1 Framework

In this chapter, we depart from the game model we introduced in Chapter 2 and used in the following chapters 3 and 4: the observations are not attached to the states of the game graph anymore, but to the transitions, and we assume perfect information on the states. Furthermore, the kind of winning conditions we consider is not restricted to the *qualitative* ω-regular conditions we studied in the previous chapters. Indeed, we also consider *quantitative* objectives, where each move in a play yields a reward for each player, or a *payoff* in the form of a relative number and the objectives are stated according to a fixed way of aggregating payoffs in the long-run. These shifts partly stem from the fact that our study in this chapter is largely inspired by the work of Fudenberg, Ishii and Kominers in [38]. Indeed, they work within the framework of *repeated games*, which is a widely used model in economical game theory. In this section, we present briefly the basic notions around repeated games before showing how to adapt this framework to fit our vision of games as state-transition systems. When the number of states is reduced to

only *one*, our model actually coincides with the one of *repeated games*.

5.1.1 Repeated games

On a historical perspective, repeated games are an extension of strategic games, that consider "one-shot" interactive situations, and for which the fundamental concepts of *preferences* and *payoff* functions as well as the equilibrium solution were originally defined. We concentrate on repeated games that unfold over an extended (finite or infinite) period of time (by presenting to the players the same strategic game over and over again), as it coincides with the infinite length game setting in this thesis. Notice as well that in the following, we take some liberties with the traditional terminology and omit certain notions in order to keep our exposition as close as possible to our general framework. For a thorough introduction to economical game theory, we refer the reader to [68] by Osborne and Rubinstein or [11] by Binmore.

Strategic and repeated games

For our simplified presentation of repeated games, we assume that players have *perfect information*. In this context, it means that players observe the actions of all the other players. We discuss in Section 5.1.2 the implications of imperfect information about the actions in the setting of games with multiple states and non-purely cooperative objectives.

A *stage* game describes an interactive situation that require the players to simultaneously and independently choose an action. Depending on the action profile, each player receives his own *payoff*.

Formally, a stage game G for n players is a triplet $G = (N, A, (p^i)_{i \in N})$ where:

- $N = \{1, \dots, n\}$;
- $A = \prod_{i=1}^{n} A^i$, where A^i is the finite set of *actions* of Player i;
- each player i has its own *payoff* function $p^i : A \to \mathbb{Z}$ that attributes to each action profile a payoff for Player i.

A *repeated* game consists of the infinite[1] iteration of the same *stage* game. At each stage, each player chooses his own action, and the resulting action profile determines the payoff profile of the stage game, according to the payoff functions p^i. However, since the interaction is infinitely repeated, there is a need to *aggregate* the payoffs gathered during a play to be able to reason about the outcome of the game as a whole. Formally, a repeated game $\mathcal{G}$ is a pair consisting of a stage game G and an aggregation function common to all players $u : \mathbb{Z}^\omega \to \mathbb{R}$. A wide variety of aggregation functions have been considered, depending on the kind of situations the game models. We describe the most common ones here: For an infinite sequence $p = (p_0^i, p_1^i, \dots)$ of stage payoffs for player i, a function $u : \mathbb{Z}^\omega \to \mathbb{R}$ is said to be:

- A *lim-sup* aggregation function if p evaluates to $\limsup_{t \geq 1} p_t^i$

[1] finite horizon can also be considered for repeated games, however we focus on infinite horizon

- A *lim-inf* aggregation function if p evaluates to $\liminf_{t\geq 1} p_t^i$
- A *mean-payoff* aggregation function if p evaluates to $\limsup_{t\geq 1} \sum_{r=1}^{t} p_r^i/t$
- A *discounted* aggregation function if there exists a real number $\delta \in]0,1[$, called the *discount factor*[2] and p evaluates to $\sum_{t=1}^{\infty} \delta^t p_t^i$

While in a stage game considered independently, a strategy for a player amounts to a single action, in the repeated setting, as in games on graphs, it is worth taking into account the past events before choosing the next action. As, at every round, the same stage game is played, the substance of a repeated game is the sequence of action profiles that unfolds during the game. For repeated games, a *history* of length ℓ is a sequence $(a_1, \ldots, a_\ell)$ of ℓ action profiles. A play π is an infinite sequence of action profiles.

A (pure) strategy for a player i is a function $s^i : A^* \to A^i$ that prescribes an action $a^i \in A^i$ after any history of the game $\mathcal{G}$. A *mixed* strategy allows for stochastic prescriptions according to a probability distribution over the set of actions. The notion of mixed strategies is fundamental in game theory, as we suggest in Example 5.3. However, in this work, as in the rest of this thesis, we restrict ourselves to deterministic strategies, that is, pure strategies in the current terminology.

Since we consider repeated games with perfect information and pure strategies, a strategy profile corresponds to exactly one play: as every strategy of the profile is fixed and deterministic, the resulting play is unique. This allows to speak directly of the outcome profile of a strategy profile in terms of aggregated payoffs: For a strategy profile $s = (s^1, \ldots, s^n)$, its unique resulting play $\pi = a_1, a_2, \ldots$ and an aggregation function u, the outcome of s is the aggregated payoff profile $(u(s^1), \ldots, u(s^n))$, where, for each player i, the aggregated payoff $u(s^i)$ has value $u(p^i(a_1), p^i(a_2), \ldots)$.

As players may have diverging objectives, the notion of *best response* arises to describe strategies that guarantee to a player the maximal payoff against a fixed strategy profile of his opponents.

A strategy s^i for player i is a *best response* to a joint strategy s^{-i} of the other players if for every every strategy s'^i for player i, we have $u(s^{-i}, s^i) \geq u(s^{-i}, s'^i)$.

Equilibrium concept

While the concept of a *solution* for games with qualitative objectives, as we have considered so far, is naturally captured by *winning strategies*, in the sense that the winning condition is either fulfilled or not during a play, defining what an acceptable solution is for a game with quantitative objectives is not straightforward. The most common hypothesis is that players of such games are *rational*, that is, their own interest comes first, which translates into the assumption that they try to maximise their own (aggregated) payoff. Thus, a player is considered to prefer a strategy s to a strategy s' whenever he can expect that s will bring him a better payoff than s' against all possible strategies of the other players. With this in mind, and since players have their own payoff distributions, so that

[2] that represents the notion of *patience* for a player

there may not be one play that yield every player his maximal payoff, the search for a solution of a repeated game can be understood as the quest for a strategy profile that would achieve some form of optimisation: a situation where the behaviours of the players would enforce a *steady state* of the game, so that no player would increase his payoff by unilaterally changing his behaviour, or *deviating*. This solution concept is captured to a certain extentby the notion of *Nash equilibrium*, introduced by Nash in [66] and [65] :

A strategy profile $s = (s^1, \ldots, s^n)$ is a Nash equilibrium, or *NE* for short, if no player has an incentive to deviate. More formally, s is a NE if for every player i, if, for every strategy s'^i for player i, it is true that $u^i((s^{-i}, s'^i)) \leq u^i((s^{-i}, s^i))$, where s^{-i} denotes the profile s without the strategy of Player i. In other terms, for each player i, the strategy s^i is a best response to s^{-i}. A Nash equilibrium is *pure* if it consists of pure strategies, and *mixed* if it consists of mixed strategies.

As we will see in Example 5.3, the existence of a pure Nash equilibrium is not guaranteed in a strategic game, however, it is always possible to find a mixed Nash equilibrium in a strategic game.

We present now three simple examples of repeated games:

Example 5.1 (Prisoner's Dilemma)**.** We recall briefly the setting of the most famous example in game theory: Consider two acquainted bank robbers. Police take them in simultaneously but separately, making sure they cannot communicate while in their respective interrogation rooms. There is a lack of evidence to convict them, so the inspectors need them to confess or at least to betray their partner. Otherwise, if they both choose to remain silent, they will get a lesser jail time. Trying to convict both robbers, and not counting on willing confessions of their own guilt, each inspector presents the following outcomes to the robber he interrogates: "If both of you remain silent, you will both get a sentence of one year in prison. If you tell us he did it, either he remains silent and you go free, or he told my colleague you did it, and you both end up with five years in jail. But if you remain silent and he tells on you, he will go free and you will spend ten years in prison. It is your choice."

In the point of view of the robbers, both of them remaining silent is their shot at the most desirable outcome. At least the best compromise. It is quite risky, however, since remaining silent exposes to the other betraying and getting the less desirable outcome of a ten years sentence.

	Cooperate	Defect
Cooperate	(1,1)	(0,10)
Defect	(10,0)	(5,5)

If this situation is a one-time occurrence, in other terms, if this is a stage game, it is well-known that the only equilibrium is (D, D), meaning both robbers should betray their partner, if they are rational. Now, if this happens repeatedly over time, things change. Indeed, with repetition comes the possibility of punishing or *retaliation*. For this particular example, a player being aware that if he betrays his partner once, he may go free on this

instance, but it will trigger a never ending commitment from his partner to betray him in the future is enough to annihilate the incentive he could have had to betray his partner in the first place. More concretely, what he would gain by switching from cooperate to defect would be one year at this round, and then at each round and forever, serve four years more than if he would have continued to cooperate. Hence, the strategy profile $s = (s^1, s^2)$ such that for any history $\pi \in A^*$ ending at an action profile $a = (a^1, a^2)$,

$$s^1(\pi) = \begin{cases} C & \text{if } a^1 = C \text{ and } a^2 = C \\ D & \text{if } a^1 = D \text{ or } a^2 = D \end{cases}$$

and s^2 is defined symmetrically, is an equilibrium in the repeated setting. [3] These strategies are known as *grim-trigger* strategies. Other pure strategy profile yield equilibria, such as *tit-for-tat* strategies and their variants, where a betrayed player punishes his partner, but only for a fixed number of subsequent turns before returning to cooperating.

Example 5.2 (Bach or Stravinsky)**.** An other classic repeated game example is called *Bach or Stravinsky*[4]. The setting is the following: there are two friends, let us name them *Alice* and *Bob*, that want to have a nice themed-evening together every week. They share a passion for classical music, so they settle to make their weekly meeting an occasion to attend a concert playing their favourite pieces. However, Alice's favourite composer is Johann Sebastian Bach, while Bob is a fan of Igor Stravinsky's compositions. Their town is culturally big enough to offer opportunities to listen to both composer's music every week. One way to see this is that their choice of concert depends entirely on their own preferences. Of course, if they both go to a concert playing Bach, Alice will be thrilled, while Bob will enjoy the great music but would rather have gone with his friend to the Stravinsky concert across town, and symmetrically. At least, they are friends, so they both prefer to go to the concert playing the other's favourite composer over splitting and both going to different concerts. These preferences can be summed up in the following payoff matrix:

	Bach	Stravinsky
Bach	(3,2)	(0,0)
Stravinsky	(0,0)	(2,3)

In this case, the stage game has two pure Nash equilibria: (B, B) and (S, S) yielding equilibrium payoffs $(3, 2)$ and $(2, 3)$ respectively. In the repeated setting, along with the mean-payoff aggregation function, players can both achieve the equilibrium payoff profile of $(\frac{5}{2}, \frac{5}{2})$ by alternating Bach and Stravinsky (or the other way around) every other week.

Example 5.3 (Matching Pennies)**.** The previous two games were examples of a *non-zero sum* games, that is, games where the players do not have strictly adversarial objectives. We present here an example of *zero-sum* game, where the payoff of one player corresponding

[3] Notice that a finite horizon would not admit s as an equilibrium strategy.

[4] Also known as *Battle of the Sexes*, involving Alice being fond of ballet and Bob of football...

to a certain action profile is the opposite number of the payoff of his opponent. The following game is also an example of a game that does not admit any pure equilibrium. *Matching Pennies* is a game for two players where, at each stage game, both players have to independently and simultaneously choose *Heads* or *Tail*. When the chosen actions match, the first player receives a payoff of 1, while the second player receives the negative payoff of -1. Symmetrically, when the actions mismatch, the second player receives a payoff of 1, while the first player receives the negative payoff of -1. A way to see it is to think that the losing player of the stage game pays 1 unit of their shared currency to the winning player.

	H	T
H	(1,-1)	(-1,1)
T	(-1,1)	(1,-1)

In the stafe game, there is no pure strategy that yields an equilibrium: for every strategy profile among the four pure existing, one of the player would profit by deviating. As suggested by the name of the game, the optimal strategy is *mixed* and corresponds to play the game as if they would both toss a coin and let chance determine the outcome of the stage game. That way, the equilibrium payoff profile is $(\frac{1}{2}, \frac{1}{2})$. Similarly, in the repeated setting, no pure equilibrium can be found: each pure strategy profile gives one or the other player an incentive to deviate.

5.1.2 From repeated games to games with multiple states

The setting from the previous chapters is standard for the automated verification and synthesis of reactive modules that maintain ongoing interaction with their environment, seeking to satisfy a common global specification. Imperfect information about the play is modelled as uncertainty about the current state in the underlying transition system, whereas uncertainty about the actions of other players is not represented explicitly. This is because the main question concerns *distributed* winning strategies, i.e., Nash equilibria in the special case where the players have a common utility function and should each receive maximal payoff. If every player wins when all follow the prescribed strategy, unilateral deviations cannot be profitable, hence there is no need to monitor actions of other players. Accordingly, distributed winning strategies can be defined on possible histories of visited states, independently of the history of played actions. Nevertheless, these games are computationally intractable in general, already with respect to the question of whether distributed winning strategies exist as seen in Chapter 3 and [70, 69, 1].

However, if no equilibria exist that yield maximal payoffs to all players in a game, and we consider arbitrary Nash equilibria rather than distributed winning strategies, it becomes crucial for a player to monitor the actions of other players. To illustrate, one elementary scheme for constructing equilibria in games of infinite duration relies on *grim-trigger* strategies: cooperate on the prescribed equilibrium path until one player deviates, and at that event, enter a coalition with the remaining players and switch to a joint punishment strategy against the deviator. Most procedures for constructing Nash equilibria in games for verification and synthesis are based on this scheme, which relies essentially on the ability of players to detect *jointly* the deviation [83, 85, 16, 14].

The grim-trigger scheme works well under perfect, instant monitoring, where all players have common knowledge about the most recent action performed by any other player. In contrast, the situation becomes more complicated when players receive only imperfect signals about the actions of other players, and worse, if the signals are not delivered instantly, but with uncertain delays that may be different for each player. Imagine a scenario with three players, where Player 1 deviates from the equilibrium path and this is signalled to Player 2 immediately, but only with a delay to Player 3. If Player 2 triggers a punishment strategy against Player 1 as soon as she detects the deviation, Player 3 may monitor the action of Player 2 as a deviation from the equilibrium and trigger, in his turn, a punishment strategy against her, overthrowing the equilibrium outcome to the profit of Player 1.

One motivation for studying infinite games with delays comes from the work of Shmaya [79] considering sequential games on finitely branching trees (or equivalently, on finite graphs) where the actions of players are monitored perfectly, but with arbitrary finite delays. In the setting of two-player zero-sum games with Borel winning conditions, Shmaya shows that these delayed-monitoring games are determined in mixed strategies. Apart of revealing that infinite games on finite graphs are robust under monitoring delays, the paper is enlightening for its proof technique, which relies on a reduction of the delayed-monitoring game to a game with a different structure that features instant monitoring but, in exchange, involves stochastic moves.

Our analysis is inspired directly from recent work of Fudenberg, Ishii, and Kominers [38] on infinitely repeated games with bounded-delay monitoring with stochastically distributed observation lags. The authors prove a transfer result that is much stronger than ours, which also covers the relevant case of discounted payoffs (modulo a controlled adjustment of the discount factor). The key idea for constructing strategies in the delayed-response game is to modify strategies from the instant-response game by letting them respond with a delay equal to the maximal monitoring delay so that all players received their signals. This amounts to combining different threads of the instant-monitoring game, one for every time unit in the delay period. Thus, the proof again involves a reduction between games of different structure, with the difference that here one game is reduced to several instances of another one.

Infinitely repeated games correspond to the particular case of concurrent games with only one state. This allows applying classical methods from strategic games, which are no longer accessible in games with several states [74]. Additionally, the state-transition structure of our setting induces a combinatorial effort to adapt the delayed-response strategies from [38]: As the play may reach a different state until the monitoring delay expires, the instant-monitoring threads must be scheduled more carefully to make sure that they combine to a valid play of the delayed-monitoring variant. In particular, the time for returning to a particular game state may be unbounded, which makes it hard to deliver guarantees under discounted payoff functions. As a weaker notion of patience, suited for games with state transitions, we consider payoff aggregation functions that are *shift-invariant and submixing*, as introduced by Gimbert and Kelmendi in their work on memoryless strategies in stochastic games [43].

Our model generalises concurrent games of infinite duration over finite graphs. Equilibria in such models have been investigated for the perfect-information case, and it was

shown that it is decidable with relatively low complexity whether equilibria exist, and if this is the case, finite-state equilibrium profiles can be synthesised for several cases of interest in automated verification. Ummels [83] considers turned-based games with parity conditions and shows that deciding whether there exists a pure Nash equilibrium payoff in a given range is an NP-complete problem. For the case of concurrent games with mean-payoff conditions, Ummels and Wojtczak [84], show that the problem for pure strategies is still NP-complete, whereas it becomes undecidable for mixed strategies. For the case of concurrent games with Büchi conditions, that is, parity conditions with priorities 1 and 2, Bouyer et al. [13] show that the complexity of the problem drops to PTime. These results are in the setting of perfect information about the actual game state and perfect monitoring. However, as pointed out in the conclusion of [13], the generic complexity increases when actions are not monitored by any player.

The basic method for constructing equilibria in the settings of perfect monitoring relies on grim-trigger strategies that react to deviations from the equilibrium path by turning to a zero-sum coalition strategy opposing the deviating player. Such an approach can hardly work under imperfect monitoring where deviating actions cannot be observed directly. Alternative approaches for constructing equilibria without relying on perfect monitoring comprise, on the one hand distributed winning strategies for games that allow all players of a coalition to attain the most efficient outcome [53, 36, 5], and at the other extreme, Doomsday equilibria, proposed by Chatterjee et al. in [22], for games where any deviation leads to the most inefficient outcome, for all players.

5.2 Games with Delayed Signals

There are n players $1, \ldots, n$ and a distinguished agent called Nature. We refer to a list $x = (x^i)_{1 \leq i \leq n}$ that associates one element x^i to every player i as a *profile*. For any such profile, we write x^{-i} to denote the list $(x^j)_{1 \leq j \leq n, j \neq i}$ where the element of Player i is omitted. Given an element x^i and a list x^{-i}, we denote by (x^i, x^{-i}) the full profile $(x^i)_{1 \leq i \leq n}$. For clarity, we always use superscripts to specify to which player an element belongs. If not quantified explicitly, we refer to Player i to mean any arbitrary player.

5.2.1 General model

For every player i, we fix a set A^i of *actions*, and a set Y^i of *signals*; these sets are finite. The *action space* A consists of all action profiles, and the *signal space* Y of all signal profiles.

Transition structure

The transition structure of a game is described by a *game graph* $G = (V, E)$ over a finite set V of *states* with an edge relation $E \subseteq V \times A \times Y \times V$ that represents *transitions* labelled by action and signal profiles. We assume that for each state v and every action profile a, there exists at least one transition $(v, a, y, v') \in E$.

The game is played in stages over infinitely many periods starting from a designated initial state $v_0 \in V$ known to all players. In each period $t \geq 1$, starting in a state v_{t-1},

every player i chooses an action a_t^i, and Nature chooses a transition $(v_{t-1}, a_t, y_t, v_t) \in E$, which determines a profile y_t of emitted signals and a successor state v_t. Then, each player i observes a set of signals depending on the monitoring structure of the game, and the play proceeds to period $t+1$ with v_t as the new state.

Accordingly, a *play* is an infinite sequence $v_0,\ a_1, y_1, v_1,\ a_2, y_2, v_2\ \cdots \in V(AYV)^\omega$ such that $(v_{t-1}, a_t, y_t, v_t) \in E$, for all $t \geq 1$. A *history* is a finite prefix

$$v_0,\ a_1, y_1, v_1,\ \ldots,\ a_t, y_t, v_t \in V(AYV)^*$$

of a play. We refer to the number of stages played up to period t as the *length* of the history.

Monitoring structure

We assume that each player i always knows the current state v and the action a^i she is playing. However, she is not informed about the actions or signals of the other players. Furthermore, she may observe the signal y_t^i emitted in a period t only in some later period or, possibly, never at all.

The signals observed by Player i are described by an *observation* function

$$\gamma^i : V(AYV)^+ \to 2^{Y^i},$$

which assigns to every nontrivial history $\pi = v_0,\ a_1, y_1, v_1,\ \ldots,\ a_t, y_t, v_t$ with $t \geq 1$, a set of signals that were actually emitted along π for the player:

$$\gamma^i(\pi) \subseteq \{y_r^i \in Y^i \mid 1 \leq r \leq t\}.$$

For an actual history $\pi \in V(AYV)^*$, the *observed history* of Player i is the sequence

$$\beta^i(\pi) := v_0,\ a_1^i, z_1^i, v_1,\ \ldots,\ a_t^i, z_t^i, v_t$$

with $z_r^i = \gamma^i(v_0,\ a_1, y_1, v_1,\ \ldots,\ a_r, y_r, v_r)$, for all $1 \leq r \leq t$. Analogously, we define the *observed play* of Player i.

A *strategy* for Player i is a mapping $s^i : V(A^i 2^{Y^i} V)^* \to A^i$ that associates to every observation history $\pi \in V(A^i 2^{Y^i} V)^*$ an action $s^i(\pi)$. The *strategy space* S is the set of all strategy profiles. We say that a history or a play π *follows* a strategy s^i, if $a_{t+1}^i = s^i(\beta^i(\pi_t))$, for all histories π_t of length $t \geq 0$ in π. Likewise, a history or play follows a profile $s \in S$, if it follows the strategy s^i of each player i. The *outcome* $\mathrm{out}(s)$ of a strategy profile s is the set of all plays that follow it. Note that the outcome of a strategy profile generally consist of multiple plays, due to the different choices of Nature.

Strategies may be partial functions. However, we require that, for any history π that follows a strategy s^i, the observed history $\beta^i(\pi)$ is also included in the domain of s^i.

With the above definition of a strategy, we implicitly assume that players have perfect recall, that is, they may record all the information acquired along a play. Nevertheless, in certain cases, we can restrict our attention to strategy functions computable by automata with finite memory. In this case, we speak of *finite-state strategies*.

Payoff structure

Every transition taken in a play generates an integer payoff to each player i, described by a *payoff* function $p^i : E \to \mathbb{Z}$. These stage payoffs are combined by a *payoff aggregation* function $u : \mathbb{Z}^\omega \to \mathbb{R}$ to determine the *utility* received by Player i in a play π as $u^i(\pi) := u(\, p^i(v_0, a_1, y_1, v_1), p^i(v_1, a_2, y_2, v_2), \dots)$. Thus, the profile of *utility*, or global payoff, functions $u^i : V(AYV)^\omega \to \mathbb{R}$ is represented by a profile of payoff functions p^i and an aggregation function u, which is common to all players.

We generally consider utilities that depend only on the observed play, in the sense that, $u^i(\pi) = u^i(\pi')$, for any plays π, π' that are indistinguishable to Player i, that is, $\beta^i(\pi) = \beta^i(\pi')$. To extend payoff functions from plays to strategy profiles, we set

$$u^i(s) := \inf\{u^i(\pi) \mid \pi \in \mathrm{out}(s)\}, \text{ for each strategy profile } s \in S.$$

Overall, a game $\mathcal{G} = (G, \gamma, u)$ is described by a game graph with a profile of observation functions and one of payoff functions. We are interested in *Nash equilibria*, that is, strategy profiles $s \in S$ such that $u^i(s) \geq u^i(r^i, s^{-i})$, for every player i and every strategy $r^i \in S^i$. The payoff $w = u^i(s)$ generated by an equilibrium $s \in S$ is called an equilibrium payoff. An equilibrium payoff w is *ergodic* if it does not depend on the initial state of the game, that is, there exists a strategy profile s with $u(s) = w$ for every choice of an initial state.

5.2.2 Instant and bounded-delay monitoring

We focus on two particular monitoring structures, one where the players observe their component of the signal profile instantly, and one where each player i observes his private signal emitted in period t in some period $t + d_t^i$, with a bounded delay $d_t^i \in \mathbb{N}$ chosen by Nature.

Formally, a game with *instant monitoring* is one where the observation functions γ^i return, for every history $\pi = v_0,\ a_1, y_1, v_1,\ \dots,\ a_t, y_t, v_t$ of length $t \geq 1$, the private signal emitted for Player i in the current stage, that is, $\gamma^i(\pi) = \{y_t^i\}$, for all $t \geq 1$. As the value is always a singleton, we may leave out the enclosing set brackets and write $\gamma^i(\pi) = y_t^i$.

To model bounded delays, we consider signals with an additional component that represents a timestamp. Concretely, we fix a set B^i of *basic signals* and a finite set $D^i \subseteq \mathbb{N}$ of *possible delays*, for each player i, and consider the product $Y^i := B^i \times D^i$ as a new set of signals. Then, a game with *delayed monitoring* is a game over the signal space Y with observation functions γ^i that return, for every history $\pi = v_0,\ a_1, (b_1, d_1), v_1,\ \dots,\ a_t, (b_t, d_t), v_t$ of length $t \geq 1$, the value

$$\gamma^i(\pi) = \{(b_r^i, d_r^i) \in B^i \times D^i \mid r \geq 1,\ r + d_r^i = t\}.$$

In our model, the role of Nature is limited to choosing the delays for observing the emitted signals. Concretely, we postulate that the basic signals and the stage payoffs associated to transitions are determined by the current state and the action profile chosen by the players, that is, for every global state v and action profile a, there exists a unique profile b of basic signals and a unique state v' such that $(v, a, (b, d), v') \in E$, for some $d \in D$; moreover, for any other delay profile $d' \in D$, we require $(v, a, (b, d'), v') \in E$, and also that $p^i(v, a, (b, d), v') = p^i(v, a, (b, d'), v')$. Here again, D denotes the *delay space*

composed of the sets D^i. Notice that under this assumption, the plays in the outcome of a strategy profile s differ only by the value of the delays. In particular, all plays in $\mathrm{out}(s)$ yield the same payoff.

To investigate the effect of observation delays, we will relate the delayed and instant-monitoring variants of a game. Given a game $\mathcal{G}$ with delayed monitoring, the corresponding instant-monitoring game $\mathcal{G}'$ is obtained by projecting every signal $y^i = (b^i, d^i)$ onto its first component b^i and then taking the transition and payoff structure induced by this projection. As we assume that transitions and payoffs are independent of delays, the operation is well defined.

Conversely, given a game $\mathcal{G}$ with instant monitoring and a delay space D, the corresponding game $\mathcal{G}'$ with delayed monitoring is obtained by extending the set B^i of basic signals in $\mathcal{G}$ to $B^i \times D^i$, for each player i, and by lifting the transition and payoff structure accordingly. Thus, the game $\mathcal{G}'$ has the same states as $\mathcal{G}$ with transitions $E' := \{(v, a, (b, d), w) \mid (v, a, b, w) \in E, d \in D\}$, whereas the payoff functions are given by $p'^i(v, a, (b, d), w) := p^i(v, a, b, w)$, for all $d \in D$.

As the monitoring structure of games with instant or delayed monitoring is fixed, it is sufficient to describe the game graph together with the profile of payoff functions, and to indicate the payoff aggregation function. It will be convenient to include the payoff associated with a transition as an additional edge label and thus represent the game simply as a pair $\mathcal{G} = (G, u)$ consisting of a finite labelled game graph and an aggregation function $u : \mathbb{Z}^\omega \to \mathbb{R}$.

5.2.3 Shift-invariant, submixing utilities

Our result applies to a class of games where the payoff-aggregation functions are invariant under removal of prefix histories and shuffling of plays. Gimbert and Kelmendi [43] identify these properties as a guarantee for the existence of simple strategies in stochastic zero-sum games.

A function $f : \mathbb{Z}^\omega \to \mathbb{R}$ is *shift-invariant*, if its value does not change when adding an arbitrary finite prefix to the argument, that is, for every sequence $\alpha \in \mathbb{Z}^\omega$ and each element $a \in \mathbb{Z}$, we have $f(a\alpha) = f(\alpha)$.

An infinite sequence $\alpha \in \mathbb{Z}^\omega$ is a *shuffle* of two sequences $\varphi, \eta \in \mathbb{Z}^\omega$, if $\mathbb{N}$ can be partitioned into two infinite sets $I = \{i_0, i_1, \dots\}$ and $J = \{j_0, j_1, \dots\}$ such that $\alpha_{i_k} = \varphi_k$ and $\alpha_{j_k} = \eta_k$, for all $k \in \mathbb{N}$. A function $f : \mathbb{Z}^\omega \to \mathbb{R}$ is called *submixing* if, for every shuffle α of two sequences $\varphi, \eta \in \mathbb{Z}^\omega$, we have

$$\min\{f(\varphi), f(\eta)\} \leq f(\alpha) \leq \max\{f(\varphi), f(\eta)\}.$$

In other words, the image of a shuffle product always lies between the images of its factors.

The proof of our theorem relies on payoff aggregation functions $u : \mathbb{Z}^\omega \to \mathbb{R}$ that are shift-invariant and submixing. Many relevant game models used in economics, game theory, and computer science satisfy this restriction. Prominent examples are mean payoff

or limsup payoff, which aggregate sequences of stage payoffs $p_1, p_2, \dots \in \mathbb{Z}^\omega$ by setting:

$$\text{mean-payoff}(p_1, p_2, \dots) := \limsup_{t \geq 1} \frac{1}{t} \sum_{r=1}^{t} p_r, \quad \text{and}$$
$$\text{limsup}(p_1, p_2, \dots) := \limsup_{t \geq 1} p_t.$$

Finally, parity conditions which map non-negative integer payoffs $p_1, p_2, \dots$ called priorities to $\text{parity}(p_1, p_2, \dots) = 1$ if the least priority that occurs infinitely often is even, and 0 otherwise, also satisfy the conditions.

5.2.4 The transfer theorem

We are now ready to formulate our result stating that, under certain restrictions, equilibrium profiles from games with instant monitoring can be transferred to games with delayed monitoring.

Theorem 5.1. *Let $\mathcal{G}$ be a game with instant monitoring and shift-invariant submixing payoffs, and let D be a finite delay space. Then, for every ergodic equilibrium payoff w in $\mathcal{G}$, there exists an equilibrium of the D-delayed monitoring game $\mathcal{G}'$ with the same payoff w.*

The proof relies on constructing a strategy for the delayed-monitoring game while maintaining a collection of virtual plays of the instant-monitoring game, on which the given strategy is queried. The responses are then combined according to a specific schedule to ensure that the actual play arises as a shuffle of the virtual plays.

5.3 Proof

Consider a game $\mathcal{G} = (G, u)$ with instant monitoring where the payoff aggregation function u is shift-invariant and submixing, and suppose that $\mathcal{G}$ admits an equilibrium profile s. For an arbitrary finite delay space D, let $\mathcal{G}'$ be the delayed-monitoring variant of $\mathcal{G}$. In the following steps, we will construct a strategy profile s' for $\mathcal{G}'$, that is in equilibrium and yields the same payoff $u(s')$ as s in $\mathcal{G}$.

5.3.1 Unravelling small cycles

To minimise the combinatorial overhead for scheduling delayed responses, it is convenient to ensure that, whenever the play returns to a state v, the signals emitted at the previous visit at v have been received by all players. If every cycle in the given game graph G is at least as long as any possible delay, this is clearly satisfied. Otherwise, the graph can be expanded to avoid small cycles, e.g. by taking the product with a cyclic group of order equal to the maximal delay.

Concretely, let m be the greatest delay among $\max D^i$, for all players i. We define a new game graph $\hat{G}$ as the product of G with the additive group $\mathbb{Z}_m$ of integers modulo m, over the state set $\{v_j \mid v \in V, j \in \mathbb{Z}_m\}$ by allowing transitions (v_j, a, b, v'_{j+1}), for every $(v, a, b, v') \in E$ and all $j \in \mathbb{Z}_m$, and by assigning, for all transitions $(v, a, b, v') \in E$, stage payoffs $\hat{p}^i(v_j, a, b, v'_{j+1}) := p^i(v, a, b, v')$. Obviously, every cycle in this game has length at

least m. Moreover, the games $(\hat{G}, u)$ and (G, u) are equivalent: Since the index component $j \in \mathbb{Z}_m$ is not observable to the players, the two games have the same sets of strategies, and profiles of corresponding strategies yield the same observable play outcome, and hence the same payoffs.

In conclusion, we can assume without loss of generality that each cycle in the game graph G is longer than the maximal delay $\max D^i$, for all players i.

5.3.2 The Frankenstein procedure

We describe a strategy f^i for Player i in the delayed monitoring game G' by a reactive procedure that receives observations of states and signals as input and produces actions as output.

The Frankenstein[5] procedure maintains a collection of virtual plays of the instant-monitoring game. More precisely, these are observation histories for Player i, following the strategy s^i in G, which we call *threads*. The observations collected in a thread

$$\pi = v_0,\ a^i_1, (b^i_1, d^i_1), v_1,\ \ldots,\ a^i_r, (b^i_r, d^i_r), v_r$$

are drawn from the play of the main delayed-monitoring game G'. Due to delays, it may occur that the signal (b^i_r, d^i_r) emitted in the last period of a thread has not yet been received. In this case, the signal entry is replaced by a special symbol #, and we say that the thread is *pending*. As soon as the player receives the signal, the placeholder # is overwritten with the actual value, and the thread becomes *active*. Active threads π are used to query the strategy s^i; the prescribed action $a^i = s^i(\pi)$ is played in the main delayed-monitoring game and it is also used to continue the thread of the virtual instant-monitoring game.

To be continued, a thread must be active and its current state needs to match the actual state of the play in the delayed-monitoring game. Intuitively, threads advance more slowly than the actual play, so we need multiple threads to keep pace with it. Here, we use a collection of $|V| + 1$ threads, indexed by an ordered set $K = V \cup \{\varepsilon\}$. The main task of the procedure is to schedule the continuation of threads. To do so, it maintains a data structure (τ, h) that consists of the threads $\tau = (\tau_k)_{k \in K}$ and a scheduling sequence $h = h[0], \ldots, h[t]$ of indices from K, at every period $t \geq 0$ of the actual play. For each previous $r < t$, the entry $h[r]$ points to the thread according to which the action of period $r + 1$ in the actual play has been prescribed; the last entry $h[t]$ points to an active thread that is currently scheduled for prescribing the action to be played next.

The version of Procedure Frankenstein^i for Player i, given below, is parametrised by the game graph G with the designated initial state, the delay space D^i, and the given equilibrium strategy s^i in the instant-monitoring game. In the initialisation phase, the initial state v_0 is stored in the initial thread τ_ε to which the current scheduling entry $h[0]$ points. The remaining threads are initialised, each with a different position from V. Then, the procedure enters a non-terminating loop along the periods of the actual play. In every

[5]The intuition behind this unusual name for a computing procedure is that the operational mode of the Frankenstein procedure is not without resemblance with the one of the famous literary doctor, who assembles several limbs from different sources to give life to a functioning creature, mimicking well the behaviour of the former owners of his parts, at least for a certain class of tasks!

period t, it outputs the action prescribed by strategy s^i for the current thread scheduled by $h[t]$ (Line 5). Upon receiving the new state, this current thread is updated by recording the played action and the successor state; as the signal emitted in the instant-monitoring play is not available in the delayed-monitoring variant, it is temporarily replaced by #, which marks the current thread as pending (Line 7). Next, an active thread that matches the new state is scheduled (Line 9), and the received signals are recorded with the pending threads to which they belong (Line 11 – 14). As a consequence, these threads become active.

Procedure: Frankenstein$^i(G, v_0, D^i, s^i)$

`// initialisation`

1 $\tau_\varepsilon := v_0$; $h[0] = \varepsilon$

2 **foreach** $v \in V$ **do** $\tau_v := v$

`// play loop`

for $t = 0$ **to** ω **do**

3 $\quad k := h[t]$

4 $\quad$ **assert** (τ_k is an active thread)

5 $\quad$ **play** action $a^i := s^i(\tau_k)$ `//` a^i_{t+1}

6 $\quad$ **receive** new state v `//` v_{t+1}

7 $\quad$ update $\tau_k := \tau_k\, a^i \# v$

8 $\quad$ **assert** (there exists an index $k' \neq k$ such that $\tau_{k'}$ is active and ends at state v)

9 $\quad$ set $h[t+1]$ to the least such index k'

10 $\quad$ **receive** observation $z^i \subseteq B^i \times D^i$ `//` z^i_{t+1}

11 $\quad$ **foreach** $(b^i, d^i) \in z^i$ **do**

12 $\quad\quad k := h[t - d^i]$

13 $\quad\quad$ **assert** ($\tau_k = \rho \# v'$, for some prefix ρ, state v')

14 $\quad\quad$ update $\tau_k := \rho(b^i, d^i)v'$

$\quad$ **end**

end

5.3.3 Correctness

In the following, we argue that the procedure Frankenstein^i never violates the assertions in Line 4, 8, and 13 while interacting with Nature in the delayed-monitoring game $\mathcal{G}'$, and thus implements a valid strategy for Player i.

Specifically, we show that for every history

$$\pi = v_0,\ a_1, (b_1, d_1), v_1,\ \ldots,\ a_t, (b_t, d_t), v_t$$

in the delayed-monitoring game that follows the prescriptions of the procedure up to period $t > 0$, (1) the scheduling function $h[t] = k$ points to an active thread τ_k that ends at state v_t, and (2) for the state v_{t+1} reached by playing $a_{t+1} := s^i(\tau_k)$ at π, there exists an

active thread $\tau_{k'}$ that ends at v_{t+1}. We proceed by induction over the period t. In the base case, both properties hold, due to the way in which the data structure is initialised: the (trivial) thread τ_ε is active, and for any successor state v_1 reached by $a_1 := s^i(\tau_\varepsilon)$, there is a fresh thread τ_{v_1} that is active. For the induction step in period $t+1$, property (1) follows from property (2) of period t. To verify that property (2) holds, we distinguish two cases. If v_{t+1} did not occur previously in π, the initial thread $\tau_{v_{t+1}}$ still consists of the trivial history v_{t+1}, and it is thus active. Else, let $r < t$ be the period in which v_{t+1} occurred last. Then, for $k' = h[r]$, the thread $\tau_{k'}$ ends at v_{t+1}. Moreover, by our assumption that the cycles in G are longer than any possible delay, it follows that the signals emitted in period $r < t - m$ have been received along π and were recorded (Line 12–14). Hence, $\tau_{k'}$ is an active thread ending at v_{t+1}, as required.

To see that the assertion of Line 13 is never violated, we note that every observation history $\beta^i(\pi)$ of the actual play π in $\mathcal{G}'$ up to period t corresponds to a finitary shuffle of the threads τ in the t-th iteration of the play loop, described by the scheduling function h: The observations $(a_r^i, (b_r, d_r)^i, v_r)$ associated to any period $r \leq t$ appear at the end of $\tau_{h[r]}$, if the signal $(b_r, d_r)^i$ was delivered until period t, and with the placeholder #, otherwise.

In summary, it follows that the reactive procedure Frankenstein^i never halts, and it returns an action for every observed history $\beta^i(\pi)$ associated to an actual history π that follows it. Thus, the procedure defines a strategy $f^i : V(A^i 2^{Y^i} V)^* \to A^i$ for Player i.

5.3.4 Equilibrium condition

Finally, we show that the interplay of the strategies f^i described by the reactive procedure Frankenstein^i, for each player i, constitutes an equilibrium profile for the delayed-monitoring game $\mathcal{G}'$ yielding the same payoff as s in $\mathcal{G}$.

According to our remark in the previous subsection, every transition taken in a play π that follows the strategy f^i in $\mathcal{G}'$ is also observed in some thread history, which in turn follows s^i. Along the non-terminating execution of the reactive Frankenstein^i procedure, some threads must be scheduled infinitely often, and thus correspond to observations of plays in the perfect-monitoring game G. We argue that the observation by Player i of a play that follows the strategy f^i corresponds to a shuffle of such infinite threads (after discarding finite prefixes).

To make this more precise, let us fix a play π that follows f^i in $\mathcal{G}'$, and consider the infinite scheduling sequence $h[0], h[1], \ldots$ generated by the procedure. Since there are finitely many thread indices, some must appear infinitely often in this sequence; we denote by $L^i \subseteq K$ the subset of these indices, and look at the least period ℓ^i, after which only threads in L^i are scheduled.[6] Then, the suffix of the observation $\beta^i(\pi)$ from period ℓ^i onwards can be written as a $|L^i|$-partite shuffle of suffixes of the threads τ_k for $k \in L^i$.

By our assumption that the payoff aggregation function u is shift-invariant and submixing, it follows that the payoff $u^i(\pi)$ lies between the values of $\min\{u^i(\tau_k) \mid k \in L^i\}$ and $\max\{u^i(\tau_k) \mid k \in L^i\}$. Now, we apply this reasoning to all players to show that f^i is an equilibrium profile with payoff $u(s)$.

[6] Discarding the threads that go off after a finite number of rounds allows to obtain a well-defined shuffle as a relevant play. As the payoff aggregation functions we consider are shift-invariant, it does not alter the value of the play payoff to do so.

To see that the profile f in the delayed-monitoring game $\mathcal{G}'$ yields the same payoff as s in the instant-monitoring game $\mathcal{G}$, consider the unique play π that follows f, and construct L^i, for all players i, as above. Then, all threads of all players i follow s^i, which by ergodicity implies, for each infinite thread τ_k with $k \in L^i$ that $u^i(\tau_k) = u^i(s)$. Hence $\min\{u^i(\tau_k) \mid k \in L^i\} = \max\{u^i(\tau_k) \mid k \in L^i\} = u^i(\pi)$, for each player i, and therefore $u(f) = u(s)$.

To verify that f is indeed an equilibrium profile, consider a strategy g^i for the delayed-monitoring game and look at the unique play π that follows (f^{-i}, g^i) in $\mathcal{G}'$. Towards a contradiction, assume that $u^i(\pi) > u^i(f)$. Since $u^i(\pi) \leq \max\{u^i(\tau_k) \mid k \in L^i\}$, there must exist an infinite thread τ_k with index $k \in L^i$ such that $u^i(\tau_k) > u^i(f) = u^i(s)$. But τ_k corresponds to the observation $\beta^i(\rho)$ of a play ρ that follows s^{-i} in $\mathcal{G}$, and since s is an equilibrium strategy we obtain $u^i(s) \geq u^i(\rho) = u^i(\tau_k)$, a contradiction. This concludes the proof of our theorem.

5.3.5 Finite-state strategies

The transfer theorem makes no assumption on the complexity of equilibrium strategies in the instant-monitoring game at the outset; informally, we may think of these strategies as oracles that the Frankenstein procedure can query. Moreover, the procedure itself runs for infinite time along the periods of the play, and the data structure it maintains grows unboundedly.

However, if we set out with an equilibrium profile of finite-state strategies, it is straightforward to rewrite the Frankenstein procedure as a finite-state automaton: instead of storing the full histories of threads, it is sufficient to maintain the current state reached by the strategy automaton for the relevant player after reading this history, over a period that is sufficiently long to cover all possible delays.

Corollary 5.1. *Let $\mathcal{G}$ be a game with instant monitoring and shift-invariant submixing payoffs, and let D be a finite delay space. Then, for every ergodic payoff w in $\mathcal{G}$ generated by a profile of finite-state strategies, there exists an equilibrium of the D-delayed monitoring game $\mathcal{G}'$ with the same payoff w that is also generated by a profile of finite-state strategies.*

5.4 Discussion

We presented a transfer result that implies effective solvability of concurrent games with a particular kind of imperfect information, due to imperfect monitoring of actions, and delayed delivery of signals. This is a setting where we cannot rely on grim-trigger strategies, typically used for constructing Nash equilibria in games of infinite duration for automated verification. Our method overcomes this obstacle by adapting the idea of delayed-response strategies of [38] from infinitely repeated games with one state, to arbitrary finite state-transition structures.

Our transfer result imposes stronger restrictions than the one in [38], in particular, it does not cover discounted payoff functions. Nevertheless, the class of submixing payoff functions is general enough to cover most applications relevant in automated verification and synthesis.

The restriction to ergodic payoffs was made for technical convenience. We believe it is not critical for using the main result: The state space of every game can be partitioned into ergodic regions, where all initial states lead to the same equilibrium value. As the outcome of every equilibrium profile will stay within an ergodic region, we may analyse each ergodic region in separation, and apply standard zero-sum techniques to combine the results. A challenging open question is whether the assumption of perfect information about the current state can be relaxed.

Bibliography

[1] Salman Azhar, Gary Peterson, and John Reif. Lower bounds for multiplayer non-cooperative games of incomplete information. *Journal of Computers and Mathematics with Applications*, 41:957–992, 2001.

[2] Marie-Pierre Béal, Olivier Carton, Christophe Prieur, and Jacques Sakarovitch. Squaring transducers: An efficient procedure for deciding functionality and sequentiality of transducers. In *Proc. of Latin American Symposium on Theoretical Informatics (LATIN 2000)*, volume 1776 of *Lecture Notes in Computer Science*, pages 397–406. Springer, 2000.

[3] Jean Berstel. *Transductions and Context-Free Languages*. Teubner, 1979.

[4] Dietmar Berwanger and Laurent Doyen. On the power of imperfect information. In *Proceedings of the 28th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'08)*, volume 2 of *Leibniz International Proceedings in Informatics*. Leibniz-Zentrum für Informatik, December 2008.

[5] Dietmar Berwanger, Łukasz Kaiser, and Bernd Puchala. Perfect-information construction for coordination in games. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'11)*, volume 13 of *LIPICS*, pages 387–398, Mumbai, India, December 2011. Leibniz-Zentrum für Informatik.

[6] Dietmar Berwanger, Anup Basil Mathew, and Marie van den Bogaard. Hierarchical information patterns and distributed strategy synthesis. In *Automated Technology for Verification and Analysis - 13th International Symposium, (ATVA 2015), Shanghai, China, October 12-15, 2015, Proceedings*, volume 9364 of *Lecture Notes in Computer Science*, pages 378–393. Springer, 2015.

[7] Dietmar Berwanger, Anup Basil Mathew, and Marie van den Bogaard. Hierarchical information and the synthesis of distributed strategies. *CoRR*, abs/1506.03883v2, 2016.

[8] Dietmar Berwanger and Marie van den Bogaard. Consensus game acceptors. In *Developments in Language Theory - 19th International Conference, (DLT 2015), Liverpool, UK, July 27-30, 2015, Proceedings.*, volume 9168 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2015.

[9] Dietmar Berwanger and Marie van den Bogaard. Games with delays - A frankenstein approach. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, (FSTTCS 2015), December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 307–319. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[10] Dietmar Berwanger and Marie van den Bogaard. Consensus game acceptors and iterated transductions. *CoRR*, abs/1501.07131v3, 2016.

[11] Ken Binmore. *Fun and games, a text on game theory*. DC Heath and Company, 1992.

[12] Roderick Bloem, Krishnendu Chatterjee, Thomas A Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In *International Conference on Computer Aided Verification*, pages 140–156. Springer, 2009.

[13] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Nash equilibria in concurrent games with Büchi objectives. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011), Proc.*, volume 13 of *LIPIcs*, pages 375–386. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

[14] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Pure Nash equilibria in concurrent games. *Logical Methods in Computer Science*, 11(2:9), June 2015.

[15] Patricia Bouyer, Nicolas Markey, and Daniel Stan. Mixed Nash equilibria in concurrent games. In Venkatesh Raman and S. P. Suresh, editors, *Proceedings of the 34th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'14)*, volume 29 of *Leibniz International Proceedings in Informatics*, pages 351–363, New Dehli, India, December 2014. Leibniz-Zentrum für Informatik.

[16] Thomas Brihaye, Véronique Bruyère, and Julie De Pril. On equilibria in quantitative games with reachability/safety objectives. *Theory of Computing Systems*, 54(2):150–189, 2014.

[17] Véronique Bruyère, Noémie Meunier, and Jean-François Raskin. Secure equilibria in weighted games. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 26:1–26:26. ACM, 2014.

[18] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[19] Arindam Chakrabarti, Luca De Alfaro, Thomas A Henzinger, and Mariëlle Stoelinga. Resource interfaces. In *International Workshop on Embedded Software*, pages 117–133. Springer, 2003.

[20] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[21] Krishnendu Chatterjee, Luca de Alfaro, and Thomas A Henzinger. Strategy improvement for concurrent reachability and turn-based stochastic safety games. *Journal of computer and system sciences*, 79(5):640–657, 2013.

[22] Krishnendu Chatterjee, Laurent Doyen, Emmanuel Filiot, and Jean-François Raskin. Doomsday equilibria for omega-regular games. In *Verification, Model Checking, and Abstract Interpretation*, volume 8318 of *Lecture Notes in Computer Science*, pages 78–97. Springer, 2014.

[23] Krishnendu Chatterjee, Laurent Doyen, Sumit Nain, and Moshe Y Vardi. The complexity of partial-observation stochastic parity games with finite-memory strategies. In *International Conference on Foundations of Software Science and Computation Structures*, pages 242–257. Springer, 2014.

[24] Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. *Information and Computation*, 242:25–52, 2015.

[25] Krishnendu Chatterjee and Thomas A Henzinger. A survey of stochastic ω-regular games. *Journal of Computer and System Sciences*, 78(2):394–413, 2012.

[26] Krishnendu Chatterjee, Thomas A Henzinger, and Marcin Jurdziński. Games with secure equilibria. *Theoretical Computer Science*, 365(1):67–82, 2006.

[27] Bogdan S. Chlebus. Domino-tiling games. *Journal of Computer and System Sciences*, 32(3):374 – 392, 1986.

[28] N. Chomsky and M.P. Schützenberger. The algebraic theory of context-free languages. In *Computer Programming and Formal Systems*, volume 35 of *Studies in Logic and the Foundations of Mathematics*, pages 118 – 161. Elsevier, 1963.

[29] Alonzo Church. Application of recursive arithmetic to the problem of circuit synthesis. In *Summaries of the Summer Institute for Symbolic Logic*, volume 1, pages 3–50. Cornell University, Ithaca, 1957.

[30] Alonzo Church. Logic, arithmetic and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962.

[31] Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Toruńczyk. Energy and mean-payoff games with imperfect information. In *International Workshop on Computer Science Logic*, pages 260–274. Springer, 2010.

[32] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, 1990.

[33] Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8(2):109–113, 1979.

[34] E Allen Emerson and Charanjit S Jutla. Tree automata, mu-calculus and determinacy. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 368–377. IEEE, 1991.

[35] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[36] B. Finkbeiner and S. Schewe. Uniform distributed synthesis. In *Proc. of LICS '05*, pages 321–330. IEEE, 2005.

[37] Bernd Finkbeiner and Sven Schewe. Coordination logic. In *International Workshop on Computer Science Logic*, pages 305–319. Springer, 2010.

[38] Drew Fudenberg, Yuhta Ishii, and Scott Duke Kominers. Delayed-response strategies in repeated games with observation lags. *J. Economic Theory*, 150:487–514, 2014.

[39] David Gale and Frank M. Stewart. Infinite games with perfect information. In *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematical Studies*, pages 245–266. Princeton University Press, 1953.

[40] Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed games with causal memory are decidable for series-parallel systems. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 275–286. Springer, 2004.

[41] Paul Gastin, Nathalie Sznajder, and Marc Zeitoun. Distributed synthesis for well-connected architectures. *Formal Methods in System Design*, 34(3):215–237, 2009.

[42] Blaise Genest, Doron Peled, and Sven Schewe. Knowledge = observation + memory + computation. In *Proc. of Foundations of Software Science and Computation Structures (FOSSACS 2015)*, volume 9034 of *Lecture Notes in Computer Science*, pages 215–229. Springer, 2015.

[43] Hugo Gimbert and Edon Kelmendi. Two-player perfect-information shift-invariant submixing stochastic games are half-positional. *CoRR*, abs/1401.6575, 2014.

[44] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*. Number 2500 in Lecture Notes in Computer Science. Springer, 2002.

[45] Erich Grädel and Michael Ummels. Solution concepts and algorithms for infinite multiplayer games. *New Perspectives on Games and Interaction*, 4:151–178, 2008.

[46] Ernst Moritz Hahn, Sven Schewe, Andrea Turrini, and Lijun Zhang. A simple algorithm for solving qualitative probabilistic parity games. In *International Conference on Computer Aided Verification*, pages 291–311. Springer, 2016.

[47] David Janin. On the (high) undecidability of distributed synthesis problems. In *Proc. of Theory and Practice of Computer Science (SOFSEM 2007)*, volume 4362 of *Lecture Notes in Computer Science*, pages 320–329. Springer, 2007.

[48] Barbara Jobstmann, Andreas Griesmayer, and Roderick Bloem. Program repair as a game. In *Proc. of Computer Aided Verification (CAV'05)*, volume 3576 of *Lecture Notes in Computer Science*, pages 226–238. Springer, 2005.

[49] Neil D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68 – 85, 1975.

[50] Line Juhl, Kim Guldstrand Larsen, and Jean-François Raskin. Optimal bounds for multiweighted and parametrised energy games. In *Theories of Programming and Formal Methods*, pages 244–255. Springer, 2013.

[51] Łukasz Kaiser. Game quantification on automatic structures and hierarchical model checking games. In *Proc. of CSL '06*, volume 4207 of *LNCS*, pages 411–425. Springer, 2006.

[52] Gal Katz, Doron Peled, and Sven Schewe. Synthesis of distributed control through knowledge accumulation. In *Proc. of Computer Aided Verification (CAV 2011)*, volume 6806 of *Lecture Notes in Computer Science*, pages 510–525. Springer, 2011.

[53] Orna Kupferman and Moshe Y. Vardi. Synthesizing distributed systems. In *Proc. of LICS '01*, pages 389–398. IEEE Computer Society Press, June 2001.

[54] Sige-Yuki Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, June 1964.

[55] M. Latteux and D. Simplot. Context-sensitive string languages and recognizable picture languages. *Information and Computation*, 138(2):160 – 169, 1997.

[56] Michel Latteux and David Simplot. Recognizable picture languages and domino tiling. *Theoretical Computer Science*, 178(1–2):275 – 283, 1997.

[57] Michel Latteux, David Simplot, and Alain Terlutte. Iterated length-preserving rational transductions. In *Mathematical Foundations of Computer Science 1998*, pages 286–295. Springer, 1998.

[58] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[59] Robert McNaughton. Finite-state infinite games. *Project MAC Rep*, 1965.

[60] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521 – 530, 1966.

[61] Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. Annual Symposium on Switching and Automata Theory (SWAT'72)*, pages 125–129. IEEE Computer Society, 1972.

[62] Swarup Mohalik and Igor Walukiewicz. Distributed games. In *FSTTCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, pages 338–351, 2003.

[63] Andrzej Włodzimierz Mostowski. *Games with forbidden positions*. 1991.

[64] Anca Muscholl and Igor Walukiewicz. Distributed synthesis for acyclic architectures. In *Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, Proc.*, volume 29 of *LIPIcs*, pages 639–651. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

[65] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[66] John F Nash et al. Equilibrium points in n-person games. *Proc. Nat. Acad. Sci. USA*, 36(1):48–49, 1950.

[67] Alexander Okhotin. Non-erasing variants of the chomsky–schützenberger theorem. In *Developments in Language Theory*, volume 7410 of *Lecture Notes in Computer Science*, pages 121–129. Springer, 2012.

[68] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[69] Gary L. Peterson and John H. Reif. Multiple-Person Alternation. In *Proc 20th Annual Symposium on Foundations of Computer Science, (FOCS 1979)*, pages 348–363. IEEE, 1979.

[70] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science, FoCS '90*, pages 746–757. IEEE Computer Society Press, 1990.

[71] Michael Oser Rabin. *Automata on Infinite Objects and Church's Problem*. American Mathematical Society, Boston, MA, USA, 1972.

[72] Ramaswamy Ramanujam and Sunil Simon. A communication based model for games of imperfect information. In *International Conference on Concurrency Theory*, pages 509–523. Springer, 2010.

[73] John H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and Systems Sciences*, 29(2):274–301, 1984.

[74] Dinah Rosenberg, Eilon Solan, and Nicolas Vieille. Stochastic games with imperfect monitoring. In *Advances in Dynamic Games*, volume 8 of *Annals of the International Society of Dynamic Games*, pages 3–22. Birkhäuser Boston, 2006.

[75] Jacques Sakarovitch. *Elements of automata theory*. Cambridge University Press, 2009.

[76] Arto Salomaa. *Formal Languages*. Academic Press, New York, NY , USA, 1973.

[77] Sven Schewe. Distributed synthesis is simply undecidable. *Inf. Process. Lett.*, 114(4):203–207, April 2014.

[78] Sven Schewe and Bernd Finkbeiner. Synthesis of asynchronous systems. In *International Symposium on Logic-Based Program Synthesis and Transformation*, pages 127–142. Springer, 2006.

[79] Eran Shmaya. The determinacy of infinite games with eventual perfect monitoring. *Proc. Am. Math. Soc.*, 139(10):3665–3678, 2011.

[80] Alain Terlutte and David Simplot. Iteration of rational transductions. *Informatique théorique et applications*, 34(2):99–129, 2000.

[81] Wolfgang Thomas. Church's problem and a tour through automata theory. In *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, pages 635–655, 2008.

[82] Stavros Tripakis. Undecidable problems of decentralized observation and control on regular languages. *Inf. Process. Lett.*, 90(1):21–28, 2004.

[83] Michael Ummels. The complexity of Nash equilibria in infinite multiplayer games. In *Foundations of Software Science and Computational Structures*, volume 4962 of *Lecture Notes in Computer Science*, pages 20–34. Springer, 2008.

[84] Michael Ummels and Dominik Wojtczak. The complexity of Nash equilibria in limit-average games. In *CONCUR 2011 – Concurrency Theory*, volume 6901 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2011.

[85] Michael Ummels and Dominik Wojtczak. The complexity of Nash equilibria in stochastic multiplayer games. *Logical Methods in Computer Science*, 7(3), 2011.

[86] Peter van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, volume 18 of *Lecture Notes in Pure and Applied Mathematics*, pages 331–363. Marcel Dekker Inc, 1997.

[87] Bernard Walliser. Information and beliefs in game theory. In *J. van Benthem, P. Adriaans (eds): Handbook on the Philosophy of Information*. Elsevier, 2008.

[88] Andreas Weber and Reinhard Klemm. Economy of description for single-valued transducers. *Inf. Comput.*, 118(2):327–340, May 1995.

[89] Wiesław Zielonka. Notes on finite asynchronous automata. *Informatique Théorique et Applications*, 21(2):99–135, 1987.

[90] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.

[91] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1):343–359, 1996.

Appendix A

Résumé (long) en français

L'interaction entre agents calculatoires est un phénomène omniprésent en pratique. Par exemple, si l'on considère une plateforme de réservations de vacances en ligne, plusieurs agents indépendants doivent coopérer les uns avec les autres : le client, une compagnie aériennne et un hôtel. En effet, la compagnie aérienne et l'hôtel doivent proposer des options correspondant aux exigences du client, puis effectuer les réservations sur la base de ses choix. Par ailleurs, chaque agent impliqué dans ce scénario a seulement une vue partielle de la situation : le client doit prendre des décisions en dépit du risque de "surbooking" de la part des prestataires, tandis que la compagnie aérienne et l'hôtel ne connaissent pas leurs propositions respectives, ni si le client utilisera bien les services qu'il a réservé ou s'il préférera finalement changer d'avis pour choisir d'autres prestataires aux offres plus attractives. De tels systèmes, impliquant plusieurs agents indépendants, sont appelés des *systèmes distribués*. Dans ces systèmes, l'information à propos des autres agents est cruciale, mais peut être limitée en raison de l'infrastructure même du système et la vue partielle de chaque agent sur la globalité du système. Un objectif naturel est pour les agents de pouvoir se coordonner malgré l'information éventuellement manquante. Cela implique de concevoir des programmes prescrivant le comportement des agents, et correspond au problème de *synthèse* pour les systèmes distribués. Dans cette thèse, nous étudions la synthèse distribuée au niveau théorique, sur le modèle de *jeux à information imparfaite*, et par le prisme de l'analyse des *motifs de flot d'information* dans différents scénarios interactifs.

A.1 Motivation

Les situations nécessitant de la coopération entre plusieurs agents indépendants, qui peuvent être modélisées comme des systèmes distribués, apparraissent dans une grande variété de contextes. Construire des systèmes de manière automatique de sorte à ce qu'ils soient corrects dès l'étape de conception a toujours été une ambition de la science informatique. Cette ambition suscite la volonté de mieux comprendre le rôle de l'information imparfaite dans les problèmes de synthèse distribuée. En effet, réussir la coopération en dépit de l'incertitude est un objectif connu pour être difficile à réaliser, et est même indécidable en général [70]. Découvrir les causes intrinsèques de cette difficulté et trouver des classes

de systèmes interactifs pour lesquels les spécifications sont réalisables de manière certaine nous donneraient des clés de compréhension sur de nombreux problèmes de conception, et nous suggéreraient des pistes pour surmonter ces problèmes.

Considérer le calcul au sens large comme un processus interactif est une approche fortement ancrée dans la tradition de l'informatique théorique: le concept fondamental d'alternance, introduit par Chandra, Stockmeyer et Kozen [20] au début des années quatre-vingts, où les étpaes de calcul sont attribuées à des joueurs adverses cherchant à atteindre ou éviter certains états finals, reposait sur les jeux déterminés à information parfaite et produit nombre de résultats importants, notamment en théorie des automates. En parallèle, Peterson and Reif [69] ont initié une étude du calcul via les jeux à information imparfaite, impliquant des équipes de joueurs dans un cadre hautement expressif, mais par ailleurs difficile à appréhender. Pour autant, cette approche révèle les jeux comme un outil analytique central pour modéliser les scénarios interactifs, et comme un domaine suffisament riche pour être intéressant en soi. Dans ce document, nous traitons les questions (1) et (2) sous l'angle des motifs de flot d'information.

A.2 Information Imparfaite

Dans les systèmes distribués, l'information peut n'être accessible que de façon limitée pour les agents, suivant la manière dont les agents sont connectés, le caractère imprévisible de l'environnement, et de possibles pannes au niveau de l'implémentation. Les jeux à *information imparfaite* modèlisent la vision partielle des agents sur le système entier. Plusieurs raisons peuvent causer le statut d'information imparfaite dans un jeu, et nous distinguons plus précisément deux principales *sources d'incertitude*:

Tout d'abord, l'incertitude peut émerger de la structure elle-même du système distribué. Dans notre modèle, cela est représenté par la structure de graphe sous-jacente au jeu. Les états globaux du système correspondent aux positions sur le graphe, et comportent des informations privées pour chaque joueur sous la forme d'*observations*. Les transitions d'un état global à un autre sont elles représentées par des arcs dirigés entre les positions du graphe, et sont étiquetées par des profils d'actions. Des arcs sortant d'un même état, étiquetés par le même profil d'action mais arrivant dans différents états constituent la façon de modéliser le non-déterminisme induit par l'influence de l'environnement (la Nature) sur le système. Ces propriétés structurelles permettent la prise en compte de l'incertitude causée par le comportement imprévisible de la Nature et le champ de vision restreint des joueurs.

Deuxièmement, des pannes dues à la réalité de l'exécution du système peuvent aussi apparaître et générer de l'incertitude. En pratique, plusieurs paramètres peuvent en effet altérer le comportement planifié d'un système distribué. Les problèmes de communication peuvent être, jusqu'à un certain point, modélisées dans le cadre formel des jeux. Incorporer dans le modèle une possible incertitude sur la réception des observations par les joueurs est une façon de représenter les erreurs dans le processus de communication. En revanche, le fait que ces erreurs arrivent lors de la phase d'exécution du système rendent difficiles d'inclure cette possibilité directement dans la structure de graphe des jeux. Pour cette raison, nous les modélisons au niveau du *monitorat* du jeu, c'est-à-dire la couche abstraite qui prend en charge le paramètre temporel et de la façon dont les joueurs reçoivent

effectivement leurs observations au long d'une partie.

A.3 Motifs de Flot d'Information

L'information imparfaite dans les jeux permet de modéliser des systèmes distribués dans des scénarios où les agents ne sont pas omniscients. Pour atteindre leur objectif commun, les agents de tels systèmes doivent coopérer, et cela requiert le plus souvent partager ou agréger de l'information, pour surmonter ce handicap de connaissance individuelle partielle. La façon dont l'information voyage dans le système, c'est-à-dire la façon dont les agents reçoivent de l'information de la part de l'environnement, ou la partage entre eux, est critique. Analyser la forme du flot d'information nous permet d'en apprendre plus sur la complexité de la synthèse de stratégies, de trouver des cas décidables pour la synthèse distribuée, et de diminuer la difficulté de synthétiser des stratégies gagnantes sous l'hypothèse de perturbations réalistes pendant l'exécution du système. Par conséquent, en se concentrant sur des motifs identifiables de flot d'information, nous sommes capables d'exhiber des classes de jeux représentatives de différents aspects des phénomènes d'interaction et des difficultés associées. Dans le développement de cette thèse, nous considérons le flot d'information dans les jeux sous trois angles différents:

Flot d'information bi-directionnel dans le cas le plus général, l'information peut être acquise par n'importe quel joueur et partagée pour soutenir la coopération vers l'objectif commun. De ce fait, l'information collectée individuellement par les joueurs impacte le déroulement d'une partie. En réalité, la portée de cet impact est particulièrement important: Il s'avère que l'incertitude au sujet de la connaissance des partenaires se propage itérativement, dans le sens où, un joueur, en plus de ses propres observations, doit prendre en compte les observations possibles de ses partenaires, ainsi que les spéculations de ses partenaire sur sa connaissance, et ainsi de suite. Dans notre modèle, les observations privées des joueurs sont rattachées aux états du jeux, ou positions du graphe. Par conséquent, la structure de graphe du jeu peut être vue comme un *graphe de corrélation* pour les observations des joueurs. Itérer la relation décrite par ce graphe de corrélation permet de mieux saisir la complexité de résoudre des jeux et d'exécuter des stratégies gagnantes.

Flot d'information uni-directionnel Une autre approche est de considérer la façon dont l'information est distribuée parmi les joueurs. Si, en général, la synthèse distribuée est indécidable, elle devient décidable quand l'information voyage seulement dans une direction, c'est-à-dire quand il existe un ordre parmi les joueurs, du plus informé au moins informé. En effet, le fait que l'information d'un joueur détermine l'information des joueurs plus bas dans la hiérarchie élimine le besoin de raisonner à propos de la connaissance de ces joueurs, et par conséquent la propagation de l'incertitude. Dans cette thèse, nous étudions trois motifs de flot d'information *hiérarchique*, qui assure la décidabilité: l'information hiérarchique *statique*, pour laquelle la hiérarchie est fixée pour la partie entière, l'information hiérarchique *dynamique*, pour laquelle la hiérarchie peut changer au long d'une partie, et enfin l'information hiérarchique *récurrente*, pour laquelle des phases de perturbations éphémères peuvent survenir.

Perturbations du flot d'information En pratique, même avec la conception de système la plus minutieuse et prévoyante, des pannes de matériel ou de logiciel peuvent avoir lieu. La tâche de concevoir des stratégies qui gèreraient de telles sortes d'information imparfaite est ardue, en raison des nombreux types de perturbations qui peuvent survenir au niveau de l'implémentation. Dans cette thèse, nous considérons des cas où le flot d'information est perturbé par la défaillance du mécanisme de distribution des observations aux joueurs. Assouplir l'hypothèse que le système est équippé avec une infrastructure de communication parfaite et que chaque observation est accessible instantanément pour les joueurs est une façon de décrire l'information imparfaite apparaissant en réalité au niveau de la modélisation. Plus précisément, nous étudions le scénario où les joueurs ne reçoivent pas leur observations instantanément mais avec un délai fini. Il s'avère que la synthèse distribuée peut surmonter de telles perturbations, pour certaines classes de jeux pertinentes.

A.4 Jeux sur Graphes

Notre étude s'effectue par le prisme des *jeux à information imparfaite synchrones sur graphes finis*. Les jeux sur graphes sont une façon puissante de modéliser les systèmes distribués, puisqu'ils capturent des propriétés clés des situations interactives. Nous nous intéressons aux systèmes à états finis, aussi nous concentrons-nous sur les graphes finis. Les agents d'un système distribué sont modélisés par les *joueurs* d'un jeu. Chaque joueur a son ensemble fini d'*actions* et son ensemble fini d'*observations*. Le comportement de l'environnement dans un système distribué est modélisé en partie comme un joueur particulier appelé *Nature* et ensuite par la façons dont les observations sont distribuées dans la structure de graphe. Le graphe sous-jacent représente la structure de transition du système distribué: les positions du graphe correspondent aux différents états du système, tandis que les arcs correspondent aux transitions entre les états. Chaque arc est étiqueté par un profil d'actions, c'est-à-dire, une action par joueur, qui détermine (avec le choix de la Nature) la prochaine position sur le graphe, c'est-à-dire le prochain état du jeu. Dans notre modèle, nous supposons qu'il n'existe pas de *cul-de-sac*: à chaque état, chaque profil d'action mène à un état successeur. La structure de graphe fini permet de modéliser des interactions qui impliquent des choix non-déterministes de l'environnement et qui se déroule sur une période de temps illimitée. Les exécutions d'un système distribuée sont modélisées comme des séquences (éventuellement) infinies et alternantes d'états et de profils d'actions, ou *parties*. Les spécifications sur le comportement du système sont exprimées comme des *conditions de gain* sous forme d'ensemble ω-réguliers de parties. Planifier les actions des joueurs est effectué en définissant des *stratégies*, c'est-à-dire des fonctions prescrivant une action à prendre pour chaque préfixe fini de partie, ou *historique*. Les stratégies telles que chaque partie qui les suis satisfait la condition de gain, peu importe les choix de la Nature, sont appelées des *stratégies gagnantes*.

Nous avons souligné plus tôt ce que l'information imparfaite signifiait dans le contexte des systèmes interactifs. Dans notre modèle, cela se traduit par la notion d'*observations*. Chaque joueur a son propre alphabet fini d'observations, et chaque état du graphe de jeu est étiqueté par un profil d'observations. Cet étiquetage donne naissance aux fonctions d'observation, une pour chaque joueur. À l'arrivée sur un état, chaque joueur est supposé recevoir sa composante du profil d'observations. Les fonctions d'observation peuvent ne

pas être injectives. Certains états peuvent en effet sembler similaires à un joueur, impliquant une relation d'équivalence sur l'ensemble d'états du graphe, que nous appelons la relation d'*indistinguabilité*. Comme nous supposons par ailleurs que les joueurs ont la propriété de *mémoire parfaite*, la relation d'indistinguabilité s'étend naturellement aux historiques et aux parties, et cela affecte la conception des stratégies : En effet, il se peut que plusieurs historiques soient observés pour un joueur de manière identiques, et puisque les stratégies sont des fonctions implémentées par des machines déterministes, elle doivent prescrire la même action après tous les historiques indistinguables pour être valides.

Un autre aspect significatif pour notre modèle est qu'il est *synchrone* : Cela signifie que nous supposons que le système est équipé d'une horloge globale que chaque composant du système peut consulter et à laquelle se fier pendant toute l'exécution du système. En termes de jeux, cela correspond à l'hypothèse que chaque mouvement sur le graphe a lieu après que les joueurs ont chacun et simultanément choisi *une* action, de sorte que les transitions respectent l'unité de temps du système global. En plus de cette horloge globale, nous choisissons de travailler avec un modèle de jeu où les actions des joueurs sont *simultanées*, dans le sens où chaque transition d'un état à l'autre est supposé être permis par un profil d'actions complet, avec une action par joueur. Ces hypothèses forment une manière compacte et opportune de représenter les caractéristiques des situations d'interaction sous information imparfaite.

Le problème central de la synthèse distribuée pour les jeux à information imparfaire est composé des deux facettes suivantes: La *résolubilité* est la question de savoir si, sur un graphe fixé, il existe une stratégie gagnante jointe qui satisfasse une condition de gain donnée. L'implémentation est la tâche de construire, en s'appuyant sur des automates finis, une stratégie gagnante jointe, s'il en existe une.

A.5 Plan de la Thèse

Dans cette section, nous résumons les contributions principales de cette thèse.

A.5.1 Condition de consensus

Dans une première approche pour étudier les mécanismes d'interaction, nous explorons les dépendences créées dans le flot d'information par les observations des joueurs. Pour cela, nous restreignons notre modèle général de jeu à un cadre où deux joueurs doivent coopérer contre la Nature, et ont à se mettre d'accord sur une unique décision oui/non après avoir reçu une séquence finie d'observations, ou *entrées*, sélectionnées par la Nature dans le graphe du jeu. La condition de gain considérée est une condition de *consensus*. L'unique décision doit être prise par les joueurs à certains états finals désignés, chacun d'entre eux étant associé avec un ensemble de décisions *admissibles*, et étant étiquetés avec la même observation, de sorte que les joueurs ne sont pas toujours capables de les distinguer les uns des autres. La décision qu'un joueur a à prendre à ces états finals est donc non seulement dépendant de ce qu'il a observé comme entrée, mais aussi de ce que son partenaire a observé comme entrée. Puisque la phase de réception des entrées est passive et entièrement contrôlée par la Nature, aucune communication n'est possible entre les joueurs durant la partie. De ce fait, les joueurs doivent "déduire" de leur propre

chef quelle information l'autre joueur a pour garantir une décision *sécure*, c'est-à-dire une décision qui est à la fois admissible et sera aussi choisie par son partenaire. Dans notre modèle, les observations privées des joueurs sont rattéachées aux états du jeu, ou positions du graphe. Par conséquent, la structure de graphe du jeu peut être vue comme un *graphe de corrélation* pour les observations des joueurs. Cette vision aide à isoler le rôle de l'information imparfaite dans la difficulté à résoudre la synthèse distribuée : l'incertitude d'un joueur à propos d'un état du jeu se propage sur l'information de l'autre joueur, qui à son tour doit être prise en compte par le premier, etc. Pour construire des stratégies gagnantes pour chaque joueur, la clôture transitive de l'union des deux relations d'indistinguabilité des joueurs doit être considérée. Un deuxième aspect important de ces jeux est que la décision oui/non à prendre au terme de la phase d'observation se résume en réalité à accepter ou rejeter un mot fini : Nous appelons ces jeux *consensus game acceptors* pour refléter ce fait. Une fois que nous voyons les consensus game acceptors comme des objets pour reconnaître des ensembles de mots sur des alphabets finis, la connexion avec les langages formels est naturelle et nous donnent les outils pour, tout d'abord, prouver l'indécidabilité de la synthèse distribuée dans le cas général, et ensuite, fournir une classification des consensus games acceptors en termes de complexité d'exécutions des stratégies gagnantes, en montrant des correspondances avec différentes classes de langages formels.

La première contribution ici est de revisiter la preuve classique d'indécidabilité du problème de synthèse distribuée en montrant une réduction du problème du vide pour les langages contextuels. Cela éclaire la frontière de décidabilité des jeux à information imparfaite en identifiant la coopération, dans sa forme la plus épurée de consensus, comme un critère crucial suffisant à causer l'indécidabilité, puisque les autres facteurs, comme la communication entre les joueurs et la multiplicité (voire l'infinité) de décisions à prendre au cours d'une partie, ont été éliminés du modèle. La seconde contribution est la classification en termes de langages formels. En utilisant les outils et résultats de la théorie des langages formels, nous montrons comment la forme du graphe de corrélation d'un consensus game acceptor détermine la complexité d'exécuter une stratégie gagnante pour les joueurs. avec l'aide de résultats de pavages de dominos, nous établissons la correspondance entre les consensus game acceptors et les langages contextuels, et par conséquent avec les automates linéairement bornés. Nous raffinons ensuite la classification en exploitant des caractérisations logiques des langages formels et des résultats sur les transducteurs, et obtenons des correspondances avec les langages algébriques et certaines sous-classes, comme les langages algébriques déterministes ou les langages de Dyck. Cette approche via les langages formels représente un écart de l'habituelle concentration sur les stratégies gagnantes nécessitant uniquement une mémoire finie, puisque nous obtenons des stratégies qui peuvent requérir la puissance d'automates à piles ou linéairement bornés pour être exécutées.

Ces contributions sont basées sur les résultats publiés avec Dietmar Berwanger à la conférence DLT 2015 [8], et le rapport technique correspondant [10].

A.5.2 Motifs hiérarchiques

Après avoir considéré attentivement les conséquences de l'information imparfaite sur le flot d'information dans les jeux, nous nous concentrons ensuite sur les *motifs hiérarchiques de flot d'information*. Les systèmes *hiérarchiques*, c'est-à-dire les systèmes où l'information est distribuée parmi les agents de manière ordonnée, établissant ainsi une hiérarchie de l'agent le plus informé à l'agent le moins informé, représentent un cas fondamental pour lequel le problème de synthèse distribuée devient décidable. Le fait que chaque agent ait accès aux observations reçues par les agents plus bas dans la hiérarchie rend l'analyse du système plus simple : intuitivement, il est plus facile pour un agent de coopérer avec des agents qui peuvent être simulés par sa propre information. Déjà en 1979, Peterson et Reif [69] montraient que pour des jeux dans ce cas, il est décicable — certes, avec une complexité non-élémentaire — si des stratégies gagnantes distribuées existent et si oui, que des stratégies gagnantes à mémoire finie peuvent être effectivement synthétisées. Plus tard, ce résultat sera étendu par Pnueli et Rosner [70] au modèle des systèmes distribués sur des archietectures linéaires fixes où l'information peut voyager dans une seule direction. Kupferman et Vardi, dans [53], développent une approche fondamentale de théorie des automates qui étend encore le résultat de décidabilité de spécifications en temps linéaire aux spécifications en temps arborescent. Enfin, Finkbeiner et Schewe montrent dans [36] l'existence d'un critère effectif sur les architectures de communication qui garantit la décidabilité du problème de synthèse distribuée : l'absence de *fourche d'information*, qui implique un ordre hiérarchique dans lequel les processus ont accèse aux observations fournies par l'environnement.

Dans cette thèse, nous allons plus loin que le motif d'observation hiérarchique. Nous introduisons des relaxations de ce strict principe de hiérarchie qui assurent tout de même la décidabilité et qui sont reconnaissables avec une complexité relativement basse. Plus précisément, nous passons d'abord de l'*observation* hiérarchique à l'*information* hiérarchique en supposant que les jouers ont la propriété de mémoire parfaite. Dans ce cas, le résultat de décidabilité est une conséquence directe du résultat de [70], [53] et [36]. Pour prouver ceci, nous introduisons l'outil de *signal à états finis* et expliquons comment *déduire* est aussi utile qu'*observer* dans les problèmes de synthèse distribuée. Deuxièmement, on assouplit le motif d'information hiérarchique (statique) en considérant l'information hiérarchique *dynamique*, c'est-à-dire quand l'ordre d'information entre les joueurs change au cours d'une partie. Nous montrons comment réduire ce cas au cas statiuqe en utilisant des signaux à états finis pour construire un jeu d'*ombres* équivalent qui bénéficie d'information hiérarchique statique. Enfin, nous introduisons l'information hiérarchique *récurrente*, qui correspond aux cas où des phases éphémères de perturbation du motif hiérarchique peuvent survenir. Nous pouvons montrer que pour les conditions de gain observables, le problème de synthèse distribuée est décidable pour les jeux à information hiérarchique récurrente, en nous appuyant sur la construction de suivi d'information de [5]. De plus, cela souligne le fait que le modèle de jeu est suffisamment flexible pour modéliser des schémas d'information et de communication qui ne peuvent pas être directement capturés par les architectures de communication, comme nous le suggérons dans la section 4.5.

Ces contributions sont basées sur les résultats publiés avec Dietmar Berwanger et Anup

B. Mathew à la conférence ATVA 2015 [6], et sur le rapport technique correspondant [7].

A.5.3 Monitorat avec délais

Enfin, nous déplaçons notre regard de l'incertitude causée par la structure elle-même des systèmes distribués pour explorer les conséquences d'une source d'incertitude apparaissant en pratique : les *délais*. Nous considérons le cas où le monitorat d'un jeu, c'est-à-dire, la couche habituelllement abstraite d'implémentation qui s'occupe de faire effectivement recevoir aux joueurs l'information au cours de l'exécution, peut délivrer les signaux avec un délai fini et borné. Nous étudions l'impact de ce monitorat avec délais d'un jeu sur le problème de synthèse distribuée : Plus précisément, nous nous demandons si les délais sont fatals pour l'existence de stratégies gagnantes pour des jeux qui sont résolubles dans le cas d'un monitorat instantané (sans délai). Le cas du monitorat avec délais a été brillamment traité dans le cadre de la théorie des jeux économique, et nous nous sommes directement inspirés du travail de Fudenberg, Ishii et Kominers dans [38], qui présente un résultat de transfert qui permet de construire des stratégies préservant les équilibres pré-existant dans la version d'un jeu avec délais à partir de stratégies gagnantes du jeu sans délai. Ce résultat de transfert repose sur la technique de *réponse retardée*, dont l'idée principale est pour les joueurs d'attendre le délai maximal avant de réagir à un signal. Ce faisant, il est garanti que tous les joueurs auront reçu l'information concernant une certaine étape du jeu, et construire les stratégies à réponse retardée revient à recombiner dans un ordre précis des différents *fils d'exécution* du jeu à monitorat instantané. Leur travail s'inscrit en revanche dans le modèle des *jeux répétés*, qui correspond au cas particulier de jeux avec un seul état dans notre terminologie.

Dans cette thèse, nous adaptons leur technique aux jeux à états multiples. Pour ce faire, nous introduisons un nouveau modèle de jeux synchrones à information imparfaite, où les signaux d'observations prtant l'information sur les actions des autres joueurs ne sont plus rattachés aux états, et où les délais sont modélisés. Nous présentons ensuite la procédure *Frankenstein*; qui construit de manière effective des stratégies pour les versions à monitorat avec délais de jeux résolubles dans le cas du monitorat instantané, en se reposant sur une réduction du jeu à monitorat avec délais à une collection d'instances du jeu à monitorat instantané. Nous identifions par ailleurs la classe de jeux pour laquelle ce tranfert est applicable : les jeux à conditions de gain *shift-invariant* et *submixing*.

Ces résultats ont été publiés avec Dietmar Berwanger à la conférence FSTTCS 2015 [9].

A.6 Organisation de la Thèse

Dans le Chapitre 2, nous introduisons les concepts et résultats de base nécessaires pour procéder au coeur de notre travail. Pour commencer, nous rappelons brièvement comment la synthèse, puis plus tard la synthèse distribuée, sont devenues des problèmes proéminent en informatique théorique. Ensuite, nous détaillons le modèle de jeux synchrones à information imparfaite sur graphes finis et ses liens avec d'autres modèles de jeux classiques dans la littérature. Nous explicitons notre modèles en présentant les différentes notions sur un exemple simple. Enfin, nous énonçons les résultats sur les jeux sur graphes nécessaires à l'obtention des résultats présentés dans le reste du manuscrit.

Dans le Chapitre 3, nous restreignons notre modèle général pour le simplifier le plus possible. Nous nous concentrons sur ce que nous appelons des *consensus game acceptors* : des jeux à information imparfaite sur graphe finis pour deux joueurs contre la Nature, où les joueurs ont une seule décision à prendre par partie pour la gagner ou la perdre. Ils doivent choisir entre accepter ou rejeter des séquences d'observations finies. Pour gagner une partie, ils doivent choisir en fonction de la structure du graphe (certains états permettent les deux décisions, alors que d'autres en permettent seulement une des deux) et en *consensus*, c'est-à-dire en se mettant d'accord sur la même décision. Dans ce chapitre, nous explorons les conséquences du flot d'information depuis et vers un joueur, en mettant à jour la façon dont les relations d'indistinguabilité des joueurs sont interconnectées. Nous présentons une preuve alternative d'indécidabilité de la synthèse distribuée et une classification des consensus game acceptors en termes de langages formels.

Dans le Chapitre 4, après avoir considéré le flot d'information du point de vue d'un joueur, nous nous intéressons à un contexte plus classique, c'est-à-dire un modèle où les actions ne sont pas limitées à un certain nombre d'états finals. La perspective ici est plus globale, puisque nous nous concentrons sur la façon dont l'information est distribuée parmi les joueurs. Nous explorons la synthèse de stratégies distribuées dans les cas où l'information est ordonnée, dans le sens où il existe une hiérarchie parmi les joueurs, du plus informé au moins informé, sans ensembles d'information incomparables. Nous montrons que ces cas implique la décidabilité du problème de synthèse distribuée, ainsi que l'existence de solutions implémentables avec mémoire finie, et montrons enfin des procédures éffectives pour détecter les variantes du motif d'information hiérarchique dans un jeu.

Dans le Chapitre 5, nous considérons une perturbation du flot d'information pouvant survenir en pratique, puisque nous nous intéressons au scenario où les signaux sont délivrés aux joueurs avec un délai fini. Pour ce faire, nous introduisons une modification de notre modèle de jeu : nous détachons les observations des états du jeu pour les attacher aux transitions, et nous justifions ce changement en début de chapitre. De plus, nous montrons que les isssues d'un jeu dans son instantiation avec monitorat instantané peuvent être préservées et garantie dans sa variante à monitorat avec délais, grâce à une procédure construisant des solutions en recombinant et réorganisant prudemment les stratégies conçues pour être gagnantes dans le cas du monitorat instantané.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Motifs de Flot d'Information dans les Jeux à Information Imparfaite

Mots clefs : Jeux, information imparfaite, synthèse distribuée, flot d'information

Résumé : De plus en plus de tâches informatiques sont effectuées par des systèmes interactifs qui impliquent plusieurs agents distribués, qui n'ont qu'une connaissance locale de l'état global du système, et leurs propres ressources de calcul. Notre travail explore des scénarios interactifs par le prisme des *jeux synchrones sur graphes finis et à information imparfaite*.
Premièrement, on s'intéresse à la question de la difficulté intrinsèque des phénomènes interactifs, en étudiant une variante de notre modèle général de jeux, les *consensus game acceptors*. On montre que, sous hypothèse d'information imparfaite, la condition de consensus est suffisante pour causer l'indécidabilité. En utilisant les outils de la théorie des langages formels, on donne une classification de la complexité d'*exécuter* des stratégies gagnantes.
Ensuite, on se concentre sur des cas décidables et sur la façon dont l'information est rendue accessible aux joueurs. On identifie un motif de flot d'information parmi les joueurs qui assure la décidabilité: l'information *hiérarchique*. Un jeu est considéré à information hiérarchique quand l'information à propos de l'historique actuel est accessible aux joueurs de manière ordonnée, de telle sorte qu'il n'y a pas deux joueurs ayant de l'information incomparable. On présente trois variantes de l'information hiérarchique: L'information hiérarchique *statique*, *dynamique*, et enfin *récurrente* qui élargissent le paysage des architectures décidables.
Enfin, on considère une source particulière d'information imparfaite dans les jeux, qui peut apparaître en pratique: les *délais*. Développant la technique de *delayed-response* de la littérature de théorie des jeux économique, on élabore une procédure qui prend en charge la structure de système de transition à états multiples de notre modèle de jeux, et, en recombinant prudemment des stratégies gagnantes dans les instances de jeu avec monitorat instantané, construit des solutions au problème de la synthèse distribuée différée qui préservent les équilibres.

Title : Information-Flow Patterns in Games with Imperfect Information

Keywords : Games, imperfect information, distributed synthesis, information flow

Abstract : More and more computing tasks are performed by interactive systems that involve several distributed agents which have local knowledge about the global state of the system and their own computing resources. Our work investigates interactive scenarios through the prism of *synchronous games on finite graphs with imperfect information*.
First, we address the question why interaction is difficult to handle, by studying the *consensus game acceptor* variant of the general game model. We show that, under imperfect information, the consensus condition is enough to cause undecidability. By using tools of formal language theory, we also give a classification of the complexity of *executing* joint winning strategies.
Second, we focus on decidable cases and the way information is made accessible to the players. We identify a pattern of information flow among players that leads to decidability: *hierarchical* information. A game has hierarchical information when the information about the current history is accessible to the players in an orderly fashion, so that no two players have incomparable information. We present three variants of hierarchical information: *static*, *dynamic*, and finally *recurring* hierarchical information that enlarge the landscape of decidable architectures.
Finally, we consider a particular source of imperfect information in games, that can arise in practice: *delays*. Extending the *delayed-response* technique from the economical game theory literature, we design a procedure that handles the transition state-structure of our game model, and constructs, by carefully recombining winning strategies from instant monitoring game instances, solutions to the delayed distributed synthesis problem that preserve the equilibrium outcomes.

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