Synthesising Full-Information Protocols

Dietmar Berwanger
Université Paris-Saclay, CNRS, ENS Paris-Saclay, France
Laboratoire Méthodes Formelles

Laurent Doyen
Université Paris-Saclay, CNRS, ENS Paris-Saclay, France
Laboratoire Méthodes Formelles

Thomas Soullard
Université Paris-Saclay, CNRS, ENS Paris-Saclay, France
Laboratoire Méthodes Formelles

Abstract

We study a communication model where processes reveal their entire local information whenever they interact. However, the system involves an indeterminate environment that may control when a communication event occurs and which participants are involved. As a result, the amount of information a process may receive at once is unbounded.

Such full-information protocols are common in the distributed-computing literature. Here, we consider synchronous systems, modeled as infinite games with imperfect information played on finite graphs. We present a decision procedure for the synthesis of a process with an ω-regular specification in a system where the other participating processes are fixed. The challenge lies in constructing a finite representation of information trees with unbounded branching. Our construction is non-elementary in the size of the problem instance, and we establish a matching non-elementary lower bound for the complexity of the synthesis problem.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Representations of games and their complexity; Computing methodologies → Reasoning about belief and knowledge; Computing methodologies → Planning under uncertainty

Keywords and phrases Infinite Games on Finite Graphs; Imperfect Information; Automated Synthesis

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

One core paradigm for the analysis of complex systems is that of reactive processes, introduced by Harel and Pnueli [16]. A reactive process interacts with its environment in a stepwise, ongoing manner: at each stage, it observes an input signal and chooses a control action with the purpose of ensuring that the system satisfies a specified objective.

Unlike terminating programs that compute a function on a given input, reactive processes are meant to operate indefinitely. Their interaction with the environment is naturally modelled as an infinite-duration game between a strategic player, representing the process, and a non-strategic opponent, Nature. The process aims to satisfy the objective regardless of Nature's choices. The synthesis problem—constructing such a process—thus translates to the construction of a winning strategy in an infinite game [6, 27].

Under perfect information, where input and output uniquely determine the system's run, the game is fully observable and, for ω-regular objectives, it can be solved using established algorithms for parity games [9, 24, 27, 7]. In imperfect-information settings, where processes have only partial views of the system state, the problem becomes significantly harder. Classical approaches [25, 8] reduce such games to perfect-information ones via a powerset construction, incurring exponential cost but preserving strategy equivalence.

In distributed systems, complexity increases further: multiple processes operate based on their local observations and must coordinate to achieve a common objective. This yields coordination games with imperfect information, which are undecidable in general [22, 18, 12]. Even two processes receiving distinct inputs face an undecidable synthesis problem [26]. Coordination may require reasoning about other players' knowledge—a task that is algorithmically intractable over infinite plays [5].

To study distributed synthesis beyond known undecidability barriers, we consider a model where communication is not restricted in content. We formalise a framework of full-information protocols (FIP), inspired by concepts from distributed computing [21, 10, 29], in which all information held by a sender is transmitted during a communication event. Communication availability is controlled externally, and processes cannot choose which information to reveal.

This model captures systems with maximal information exchange constrained only by communication opportunities. Whenever synthesis is possible under arbitrary-bandwidth assumptions, it is also possible in the FIP model.

We formalise distributed systems with FIP semantics as repeated games between players and Nature. In each stage, players choose actions; the resulting action profile determines a set of enabled moves, from which Nature selects one. Each move yields an observation profile, including a local input and a set of views shared via communication. A player's view accumulates her own observations and, recursively, those of others with whom she communicates, directly or indirectly.

Strategies map such views to actions. A profile of strategies determines a set of plays—infinite sequences of moves—and is winning, if all resulting plays satisfy the objective, expressed by an automaton over infinite words.

As full synthesis remains undecidable in general, we focus on the case of a single active player and an arbitrary number of passive observers. Observers do not act, but their views may be communicated, conveying unbounded information in a single stage. This creates information trees with unbounded branching, obstructing classical synthesis approaches based on finite tree automata [24, 15, 1].

Our main contribution is a quotient construction for games described by finite-state automata that generalises Reif's powerset method to the FIP setting. It yields a finite game bisimilar to the original one, with winning strategies transferring via a homomorphism. The construction tracks information records along nested coalitions of observers, leading to a state space of nonelementary size.

We show that this complexity is tight: the synthesis problem for FIP games with one active player and n observers is as hard as the acceptance problem for Turing machines using n-fold exponential space. Nonelementary bounds have appeared in similar contexts [2, 18, 12, 13, 4], but are surprising here given the presence of only a single decision maker.

2 Basic Notions

For a function $f : X \to Y$ and a domain subset $Z \subseteq X$, we denote by $f(Z) = \{f(z) \mid z \in Z\}$ the set of images of elements in Z. The *kernel* of f relates elements with the same image $\ker f = \{(x, x') \in X \times X \mid f(x) = f(x')\}$.

For a directed graph (V, E) with edge relation $E \subseteq V \times V$, we designate the set of successors and predecessors of a node $v \in V$ by $vE = \{v' \mid (v, v') \in E\}$ and $Ev = \{v' \mid (v, v') \in E\}$.

For a set of players I, we refer to any nonempty subset $J \subseteq I$ as a *coalition*. A *profile* is a tuple of objects $x = (x^i)_{i \in I}$, one for each player. Given a profile x,we write x^i to designate the element corresponding to Player i. Likewise, for a coalition $J \subseteq I$, we write x^J to designate the profile $(x^j)_{j \in J}$ of objects associated to its members. In general, we use superscripts for (objects associated to) players or coalitions. To avoid confusion, we denote the powerset of a set X by $\wp(X)$ rather than 2^X.

For an alphabet Γ, the set of finite words over Γ is denoted by Γ^*, the set of finite nonempty words is $\Gamma^+ = \Gamma^* \setminus \{\varepsilon\}$, and the set of infinite words is Γ^ω.

2.1 Games on Graphs

In the context of reactive systems, games are used to study worst-case scenarios where a system interacts with an adversarial environment. The goal is to synthesise a strategy for the system player that ensures the specification is satisfied regardless of the environment's behaviour.

Accordingly, we focus on actions and strategies of one player —or a coalition of players— representing the system, and attribute the environement choices to a nonstrategic player, which we call Nature. In particular, we deviate from traditional terminology by featuring two-player games as games between one player and Nature.

To compare games played on different structures, we represent objectives in terms of colours assigned to game positions.

Perfect Information

Let A be a finite set of actions and C a finite set of colours. A *graph game with perfect information* is described by a coloured graph $\mathcal{G} = (V, v_\varepsilon, (E_a)_{a \in A}, \lambda)$, where V is a set of positions, $v_\varepsilon \in V$ the initial position, each $E_a \subseteq V \times V$ is a transition relation for action a, and $\lambda : V \to C$ labels positions with colours. For all $v \in V$ and $a \in A$, we assume that the successor set vE_a is nonempty.

The game is played in stages starting from the initial position v_ε. In a stage at position v, the player chooses an action $a \in A$, then Nature selects an edge $(v, w) \in E_a$, and the play moves to position w which is announced to the player. The outcome is an infinite path in $\mathcal{G}$ starting from v_ε, called a *play*; finite prefixes are called *histories*.

A reachability objective is given by a binary labelling $\lambda: V \to \{0,1\}$, and a play is winning if it visits a position labelled 1. A parity objective is specified by a labelling $\lambda: V \to \mathbb{N}$ that assigns priorities to positions, and a play is winning if the least priority seen infinitely often is even.

Strategies in games with perfect information are maps $s: V^* \to A$ from histories to actions. The *outcome* of a strategy s is the set $\mathrm{Out}(s)$ of plays $v_0 v_1 v_2 \dots$ such that $v_{t+1} \in v_t E_a$ for $a = s(v_0 v_1 \dots v_t)$ at every stage $t \geq 0$. A strategy is winning, if all the plays in $\mathrm{Out}(s)$ are winning. The synthesis problem asks whether such a strategy exists and aims to construct one effectively.

For parity and reachability objectives, it is known that whenever a winning strategy exists, there exists one that depends only on the last position in the history. Such positional strategies can be represented by a labelling of positions with actions. As the winning status of a positional strategy can be verified efficiently, the synthesis problem for parity games is decidable in NP $\cap$ CoNP; reachability games are solvable in polynomial time.

Partial Observation

One way to model uncertainty is by associating to each position an observation from a finite alphabet B, via a function $\beta: V \to B$. At each stage, only $\beta(w)$ is revealed to the player.

A *graph game with partial observation* is then given by a coloured graph $(V, v_\varepsilon, (E_a)_{a \in A}, \beta, \lambda)$. Every history $\tau = v_0 v_1 \dots v_t$ yields an observation history $\hat{\beta}(\tau) := \beta(v_0)\beta(v_1)\dots\beta(v_t)$. Strategies are functions $s: V^* \to A$ from histories to actions that do not distinguish between histories with the same observation. Under the assumption that objectives are *visible*, in the sense that indistinguishable histories end in positions with the same colour given by λ, the synthesis problem for partial-observation games on graphs with parity or reachability objectives can be reduced to a perfect-information game via a powerset construction, yielding an EXPTIME-complete problem [25, 8].

To model distributed systems, graph games with partial observation are extended to multiple players $i \in I$, each with an action set A^i, an observation set B^i, and a local observation function $\beta^i: V \to B^i$. The edges are then indexed by action profiles $a = (a^i)_{i \in I}$. At the stage game from position v, each player i selects an action $a^i \in A^i$, Nature chooses an edge $(v, w) \in E_a$, and each player observes $\beta^i(w)$. Strategies s^i of Player i are functions that do not distinguish between her observation histories. The outcome of a strategy profile is formed by the plays that are in the outcome of all component strategies.

The synthesis problem for a given, common objective asks whether there exists a strategy profile $s = (s^i)_{i \in I}$, such that all outcoming plays satisfy the objective. This problem is undecidable in general for distributed games with partial observation, even with only two players against Nature (see, e.g.,[26]).

2.2 Automata

We use deterministic finite automata as language acceptors for infinite words and as transducers on finite words.

A *Mealy automaton* is given by a tuple $(Q, \Gamma, \Sigma, q_\varepsilon, \delta, \lambda)$ consisting of a finite set Q of states, a finite input alphabet Γ, a finite output alphabet Σ, a designated initial state $q_\varepsilon \in Q$, a transition function $\delta : Q \times \Gamma \to Q$, and an output function $\lambda : Q \times \Gamma \to \Sigma$. To describe the internal behaviour, we extend the transition function from input letters to words. The extended transition function $\delta: Q \times \Gamma^* \to Q$ is defined, for every state $q \in Q$, by setting $\delta(q, \varepsilon) := q$, for the empty word ε, and $\delta(q, \tau c) := \delta(\delta(q, \tau), c)$, for any word obtained by concatenation of a word $\tau \in \Gamma^*$ and a letter $c \in \Gamma$. Likewise, to describe the external behaviour, we extend the output function to $\lambda : \Gamma^+ \to \Sigma$ by setting $\lambda(\tau c) = \lambda(\delta(q_\varepsilon, \tau), c)$ for all $\tau \in \Gamma^*$ and $c \in \Gamma$. We say that a function on Γ^* is *regular* if there exists a Mealy automaton that defines it. Further, we define the *cumulative output*, for an input word $\tau = c_1 c_2 \dots c_n \in \Gamma^*$, as the sequence $\hat{\lambda}(\tau) = \lambda(c_1)\lambda(c_1c_2)\dots\lambda(c_1c_2\dots c_n)$ consisting of the outputs of all prefixes of τ. Finally, we extend $\hat{\lambda}$ to infinite words $\pi = c_1c_2 \dots \in \Gamma^\omega$, by setting $\hat{\lambda}(\pi) = \lambda(c_1)\lambda(c_1c_2)\dots$

3 The Model

3.1 Repeated Games with Imperfect Information

Our purpose is to model dynamical systems driven by occurrences of discrete state changes that we call *moves*. System runs correspond to infinite sequences of moves drawn from a finite set Γ as the outcome of a multistage game played between a fixed set $I = \{0, 1, \dots, n\}$ of players and Nature — or Environment, in control-theoretic terminology. Each player $i \in I$ has a set A^i of actions, and any action profile $(a^i)_{i \in I}$ enables a set of moves, according to a move-action map $\mathrm{act}\colon \Gamma \to A$, which is *surjective*. In every stage, a one-shot *base game* is played as follows: each player $i \in I$ chooses an *action* a^i from her given action set A^i; the chosen profile $a = (a^i)_{i \in I}$ constrains the set of possible outcomes to the subset of moves $\{c \in \Gamma \mid \mathrm{act}(c) = a\}$ supported by a, from which Nature chooses one. The outcoming move is recorded in the play history, and the game proceeds to the next stage. The outcome of the multistage game, called a *play*, is thus an infinite sequence $\pi = c_1c_2 \dots \in \Gamma^\omega$ of moves. A *history* (of length ℓ) is a finite prefix $\tau = c_1c_2 \dots c_\ell \in \Gamma^*$ of a play; the empty history ε has length zero. The objective of a player is described by a subset of plays declared to be winning.

To pursue their objectives, players choose actions based on the information available to them. The information of a player $i \in I$ is modeled by a partition $\mathcal{U}^i$ of the set Γ^* of histories; the parts of $\mathcal{U}^i$ are called *information sets* (of the player). The intended meaning is that if the actual history belongs to an information set $U \in \mathcal{U}^i$, then Player i considers every history in U possible. The particular case where all information sets in the partition are singletons characterises the setting of *perfect information*.

Our model is *synchronous*, which means, intuitively, that players always know how many stages have been played. Formally, this amounts to asserting that all histories in an information set have the same length; in particular the empty history forms a singleton information set. Further, we assume that every player has *perfect recall* — he never forgets what he knew previously, and which actions he took. Formally, if an information set of Player i contains nontrivial histories τc and $\tau' c'$, then the predecessor history τ is in the same information set as τ' and the moves c and c' are supported by the same action of Player i.

A *decision function* for a player $i \in I$ is a map $f^i\colon \Gamma^* \to A^i$ from histories to actions. We say that a play, or a history, $c_1c_2\dots$ *follows* f^i if $\mathrm{act}^i(c_t) = f^i(c_1 \dots c_{t-1})$, for every period $t > 0$. Further, we say that an information set $U \in \mathcal{U}^i$ is *reachable* if there exists a history in U that follows f^i. A decision function f^i is *information consistent* if it is constant on all reachable information sets. A *strategy* for Player i is a decision function that is information consistent (with respect to her information partition). The outcome of a strategy f^i is the set $\mathrm{Out}(f^i) \subseteq \Gamma^\omega$ of plays that follow it.

To capture coordination problems in distributed systems, we assume that the players have a common objective specified by a set $W \subseteq \Gamma^\omega$. A strategy profile $f = (f^i)_{i \in I}$ is *winning* if $\mathrm{Out}(f) \subseteq W$.

3.1.1 Finite-state representation

As we are interested in algorithmic questions for repeated games, we will consider instances described by finite objects. We assume that the move set Γ and the set A of action profiles are finite. Hence, the action map provides a finite description of the move structure, that is, Γ^* equipped with the labelling of action profiles.

To represent objectives —sets of infinite words from Γ^ω— we use colouring functions defined by Mealy automata over Γ. We mainly work with *parity* automata, which provide a canonical form for ω-regular specifications relevant as system-design objectives [28]. These are automata that input moves —drawn during the play— and output natural numbers called *priorities*: a play is winning if the least priority output infinitely often is even. We sometimes refer to reachability objectives, given by an automaton that outputs flags 0 or 1: a play is winning if some prefix history leads to output 1. Overall, objectives are represented by Mealy automata that define a function from histories to colours from a finite alphabet —priorities in the case of parity objectives, and the accepting-state flag for reachability objectives.

To describe information partitions, it will be convenient to refer to their representation as *indistinguishability relations*, which equate histories that the player cannot distinguish. An information partition $\mathcal{U}$ thus corresponds to the equivalence relation $\sim \in \Gamma^* \times \Gamma^*$ with $\tau \sim \tau'$ whenever $\tau, \tau' \in U$ for some $U \in \mathcal{U}$. Generally, an indistinguishability relation $\sim \in \Gamma^* \times \Gamma^*$ is an equivalence relation that satisfies the following conditions, for all $\tau, \tau' \in \Gamma^*$ and $c, c' \in \Gamma$:

- if $\tau \sim \tau'$, then $|\tau| = |\tau'|$ (indistinguishable histories have the same length),
- if $\tau c \sim \tau' c'$, then $\tau \sim \tau'$ (the relation is prefix-closed),
- if $\tau c \sim \tau' c'$, then $\mathrm{act}(c) = \mathrm{act}(c')$ (the action is visible).

Every such indistinguishability relation defines an information partition with information sets given by equivalence classes $[\tau]_\sim = \{\tau' \in \Gamma^* \mid \tau' \sim \tau\}$, for each history $\tau \in \Gamma^*$. The information structures arising from the full-information protocols introduced in this paper are a particular case of indistinguishability relations recognisable by two-tape deterministic finite automata, as studied in [3].

We shall distinguish between the finite-state representation of a repeated game, for instance as a tuple $(\mathrm{act}, (\mathcal{R}^i)_{i\in I}, \mathcal{A})$, on the one hand, that includes automata $\mathcal{R}^i$ and $\mathcal{A}$ describing the indistinguishability relations and the objective, and its presentation as a logical structure $\mathcal{G} = (\Gamma^*, \mathrm{act}, (\sim^i)_{i\in I}, \lambda)$, on the other hand, with $\sim^i = L(\mathcal{R}^i)$ and $\lambda = \lambda^{\mathcal{A}}$.

3.2 Full-Information Protocols

We formalise a communication model that gives rise to a particular class of finitely-representable information structures for repeated games. Our formalisation and the application scenario are inspired from [20].

In full-information protocols, players receive local observations, as they do in partial-observation games, but they additionally communicate with other players. The information transfer in that event is idealistically efficient: the entire information available to the sending party is revealed to the receiving player. However, the opportunity of communication may not be in the control of the players: whether a communication event occurs in a particular stage, and which player it includes, is determined by the current move.

Communication opportunities for a player $i \in I$ are specified by a function $\mathrm{Com}^i \colon B^i \to \wp(I)$ that associates with each of her local observation $b \in B^i$, a set of players $\mathrm{Com}^i(b^i) \subseteq I$ to which communication links are enabled. Intuitively, when Player i receives the local observation $b^i \in B^i$, she also receives the information of every player $j \in \mathrm{Com}^i(b^i)$, which includes the observation history $b^j_1, b^j_2, \ldots, b^j_\ell$, but also (recursively) the observation history of all players in $\mathrm{Com}^j(b^j_t)$ at previous stages $t = 1, 2, \dots, \ell$. Thus, a link $j \in \mathrm{Com}^i(b^i)$ specifies a one-way communication event from sender j to receiver i: upon observing b, Player i can peek at player j. We refer to such links as *direct* links. Our semantics of communication links is transitive. If at some history, there is a direct link from Player i to Player j, and also a direct link from Player j to Player k, then an *indirect* communication link is established from Player i to Player k. Even if the protocol specifies no direct link from i to k, the information of k is revealed to i.

Formally, we represent the information available to the players at a history $\tau = c_1 c_2 \dots c_\ell$ by a *view graph* $\mathrm{View}(\tau) = (V, E)$, defined as follows:

- $V = I \times \{0, 1, \dots, \ell\}$ is the set of nodes, and a node $(i, t) \in V$ represents the viewpoint of Player i in stage t;
- $E \subseteq V \times V$ is the set of edges, where an edge $((i, t), (j, u))$ intuitively means that in stage t, Player i has access to the view of Player j in stage u; the set E contains the edges $((i, t), (i, t-1))$ for all stages $1 < t \le \ell$ and every player i —which correspond to looking into the past—, and the edges $((i, t), (j, t))$ are included, for all stages $1 \le t \le \ell$ and all players $i \in I$ and $j \in \mathrm{Com}^i(b^i)$ where $b = \beta^i(c_1 c_2 \dots c_t)$ —which correspond to revealing the current view of Player j to Player i (via a direct link).

Two histories $\tau, \tau' \in \Gamma^*$ are indistinguishable for Player i, denoted $\tau \sim^i \tau'$, if $|\tau| = |\tau'|$ and $\beta^j(\tau(t)) = \beta^j(\tau'(t))$ for all nodes (j, t) reachable from $(i, |\tau|)$ in the view graph $\mathrm{View}(\tau)$. Note that the definition implies that, if $\tau \sim^i \tau'$, then the reachable nodes from $(i, |\tau|)$ in $\mathrm{View}(\tau)$ and in $\mathrm{View}(\tau')$ coincide. We say that the histories τ, τ' are indistinguishable for a coalition $J \subseteq I$, denoted $\tau \sim^J \tau'$, if they are indistinguishable for all players of the coalition, that is, $\tau \sim^i \tau'$ for all $i \in J$.

A *full-information protocol* (FIP) for a set of players I, a move alphabet Γ, and an alphabet B of observation profiles, is described by a profile $F = (\mathcal{M}^i, \mathrm{Com}^i)_{i \in I}$ specifying for each player $i \in I$, a Mealy automaton $\mathcal{M}^i$ that defines the local observation function $\beta^i : \Gamma^* \to B^i$ and the communication map $\mathrm{Com}^i : B^i \to \wp(I)$ that specifies, for every local observation symbol, the set of players to which a communication link from i is enabled. The protocol F defines an indistinguishability relation, for each player $i \in I$, which we denote by $\sim^i_F$.

To turn a FIP instance $F = (\mathcal{M}^i, \mathrm{Com}^i)_{i \in I}$ into a game, consider now a profile A of action sets A^i for the players $i \in I$ together with a suitable action map $\mathrm{act} : \Gamma \to A$. It is necessary that the local observation functions render his own action visible to each player, in the sense that two moves with different actions yield different observations: if $\mathrm{act}^i(c) \ne \mathrm{act}^i(c')$, then $\beta^i(\tau c) \ne \beta^i(\tau' c')$, for all histories $\tau, \tau' \in \Gamma^*$ and moves $c, c' \in \Gamma$. If the condition is satisfied, then F induces a profile of indistinguishability relations for the repeated game form described by act. Together with the action map act and a parity automaton $\mathcal{A}$ for defining the objective, we thus obtain the repeated game $\mathcal{G}(F) \coloneqq (\Gamma^*, \mathrm{act}, (\sim^i_F)_{i \in A}, \lambda^{\mathcal{A}})$, which we call the *FIP game* associated to F.

3.3 Example: Leader election with failures and recoveries

To illustrate the model, consider a classical leader election scenario. A set $V = \{0, \dots, n\}$ of processes is connected in a network graph (V, E), where each edge $(i, j) \in E$ represents a unidirectional channel through which process i can send messages to process j. The system runs indefinitely in synchronous rounds.

In each round $t \in \mathbb{N}$, every process i declares a leader candidate $a^i_t \in V$ and proposes a message m^i_t for broadcasting to its neighbours $j \in iE$. The environment may crash a subset

$F_t \subseteq E$ of channels, so that process i receives messages only from $j \in (E \setminus F_t)i$. In the next round, some channels may recover while others may fail. A process i is said to fail at time t if any of its outgoing channels are crashed. Failures are bounded by parameters (N, L): at most N channels may fail in any window of L rounds. A process is valid at time t if it does not fail.

The objective of leader election is to ensure that, unless all processes fail repeatedly, there is a time t_0 after which all valid processes i agree on a leader $a_t^i = k$ who is also valid.

Note that we did not specify the message space. This is intentional: in practice, it is often difficult to settle on the structure of internal messages for distributed algorithms that are yet to be designed. To maintain focus on external behaviour, we instead assume that processes broadcast all available information whenever communication is possible—in other words, they follow a full-information protocol. Once a solution is found for this idealised setting, it may later be implemented more efficiently. It might turn out that the problem is unsolvable even under full information, hence no effective protocol exists either.

Process implementation must answer two questions: (1) How to choose a leader based on the observed failure history; (2) What to transmit. The full-information assumption focuses on the first question, assuming the second is solved or deferred.

We model this as a FIP game with players in $I = V \cup E$. Each process player $i \in V$ has actions $A^i = I$, representing candidates, whereas channel players are passive, with $|A^e| = 1$, for all $e \in E$. Moves are tuples (a, F) of declarations and crash sets, with $\Gamma = \prod_{i \in V} V \times \wp(E)$ and $\mathrm{act}(a, F) = a$.

Local observations, for each process player i, specify the subset of channels from which it can read, i.e., $B^i \subseteq \{(j, i) \in E\}$, and the communication function Com^i is the identity. Including the channels as observers is an artifice to prevent players from seeing the entire network, since we choose a semantics where communication is transitive—which is unwanted here. Accordingly, each channel (i, j) has two observations: $\{\mathrm{read}, \mathrm{lock}\}$ with $\mathrm{Com}^{(i,j)}(\mathrm{read}) = \{i\}$—when the source process is revealed—and $\mathrm{Com}^{(i,j)}(\mathrm{lock}) = \varnothing$–when it may reveal to the target. The observation automata for channels alternate between read at odd times and lock at even times. For processes, the observation automaton of i sends the $\varnothing$ observation at odd times, whereas it reveals the enabled incoming channels from $E \setminus F$ at even times. The Mealy automaton is further adjusted to keep track of the number of crashes within the time window. To ensure that the bounds (N, L) are respected, the automata send $\varnothing$ when the budget does not allow to crash the subset F recorded in the current move.

The objective is described by a parity automaton with three priorities: 2 when all valid processes agree on a valid leader; 1 when there is disagreement or the leader is invalid; 0 after a phase where all processes fail at some point. It rejects plays that visit priority 1 infinitely often.

Now we can verify that the model faithfully captures our scenario: any distributed protocol—regardless of which messages the processes pass—corresponds to a strategy profile. It solves the leader election problem under the given bounds if, and only if, the strategy is winning in the FIP game.

4 The Synthesis Problem

following synthesis problem: Given a finite FIP-game instance $\mathcal{F} = (\mathrm{act}, (\mathcal{M}^i, \mathrm{Com}^i)_{i \in I}, \mathcal{A})$, decide whether there exist a winning strategy in the repeated game $\mathcal{G}(\mathcal{F})$, and if so, effectively construct one.

Graph games are special cases of FIP games. For instance, a perfect-information game on a graph $(V, v_\varepsilon, (E_a)_{a\in A}, \lambda)$ corresponds to the repeated game $(\mathrm{act}, \mathrm{id}, \mathcal{M})$ with $\Gamma = A \times V$, where $\mathrm{act}(a, v) = a$ and $\mathcal{M}$ maps a history $\tau = (a_1, v_1)\dots(a_t, v_t)$ to $\lambda(v_t)$ if τ follows a path from v_ε, and to a fresh colour $\perp$ otherwise. Given an objective $L \subseteq C^\omega$, a play is winning in the repeated game, if $\mathcal{M}$ maps it to a sequence in L or to one reaching $\perp$ — essentially requiring the environment to follow valid transitions.

Partial-observation games translate similarly by expressing observation functions as Mealy automata that output observation labels on transitions. Standard partial-observation games [25] correspond precisely to FIP games with one player. In such repeated games, the indistinguishability relation $\sim$ is defined by a regular observation function $\beta : \Gamma^* \to B$, such that $\tau \sim \tau'$ if $\hat{\beta}(\tau) = \hat{\beta}(\tau')$. Assuming perfect recall, $\tau c \sim \tau' c'$ iff $\tau \sim \tau'$ and $\beta(\tau c) = \beta(\tau' c')$. Accordingly, every information set U has at most $|B|$ successors, so the information tree has bounded branching.

For two or more players, FIP games yield strictly richer information structures. Consider a FIP game with players $I = \{0, 1\}$, binary observations, and a communication map where Player 0 can communicate with Player 1 only on observation 1. Suppose Player 0 observes 0 for seven stages, while Player 1 observes an arbitrary bit sequence. In stage 8, Player 0 observes 1, revealing all 8 bits seen by Player 1. Thus, her current information set, with 2^7 histories, has 2^8 successors (that are singletons). If the scenario continues in the same way, until the next communication event occurs, say in stage 88, there will be a branching of degree 2^{80}, and so on. Hence, FIP information trees may have infinite branching.

Restriction to one active player, visible objectives

Since synthesis is undecidable already for two players in partial-observation games, we restrict our analysis to one active player. Accordingly, we consider games with a set of players $I = \{0, \dots, n\}$, where Player 0 is active and the others are passive observers ($|A^i| = 1$ for $i \neq 0$). References to actions or indistinguishability relations refer to Player 0 unless specified.

We further restrict to *visible* objectives, where the colouring function $\lambda^{\mathcal{A}}$ defined by the objective automaton $\mathcal{A}$ is information-consistent: $\lambda^{\mathcal{A}}(\tau) = \lambda^{\mathcal{A}}(\tau')$ whenever $\tau \sim^0 \tau'$.

Applications

Despite these restrictions, the model captures significant system-design problems. One motivation is monitoring and diagnosis: constructing a decentralised supervisor that runs alongside a fixed distributed system —using the given communication architecture— to log data and detect faults. The active player outputs new data according to the specified objective, leaving the system unchanged.

However, an active player added to fixed distributed protocol can also influence the system meaningfully. For example, in leader election, we may let one process repair channels in addition to its original role, with the objective to ensure stability, that is, that is, that leaders are revoked only if they fail. Formally this is done by extending Player 0's action set to $A^0 \times E$, and adapting the action map to exclude the channel given in the action from the failure set in the stage outcome. The synthesis objective is the stability condition in conjunction with the condition that passive players follow the actions prescribed by the original protocol. Under standard connectivity assumptions (already necessary to solve basic leader election), this can be specified by a visible automaton.

In the rest of the paper, we show that the synthesis problem for FIP games with one active player is computationally solvable, albeit with daunting theoretical complexity.

4.1 Method

The key tool for synthesis in infinite games is the automata-theoretic approach, founded by Büchi and Landweber [6] and Rabin [23, 24]. Strategies are viewed as trees with bounded branching. However, FIP games may yield unbounded information trees. Indeed, [3] shows that a regular indistinguishability relation defines a finitely-branching tree if, and only if, it corresponds to an observation function. The counterexample there is a FIP protocol with one active player and one passive observer. Thus, we cannot rely on tree automata to recognise the set of winning strategies of a FIP game.

To overcome this obstacle, we reduce a finite FIP game (with parity objectives) to a perfect-information graph game, preserving winning strategies: (1) if a winning strategy exists in the original game, one exists in the reduct; (2) from a regular strategy in the reduct, we can construct one for the original.

The reduction builds a homomorphism from histories Γ^* to a finite set of abstract states, preserving information sets. By collapsing the abstract states of an information set, we obtain a finite perfect-information game equivalent (bisimilar) to the original. This allows pulling back winning strategies.

Information quotient

A repeated game with imperfect information and one active player can be represented as a game with perfect information played on the information tree — the infinite graph with information sets as positions.

Let us fix a move alphabet Γ and a set A of actions. The *information quotient* of a game $\mathcal{G} = (\Gamma^*, \mathrm{act}, \sim, \lambda)$ is the graph $\mathcal{U}(\mathcal{G})$ on $U = \{[\tau]_\sim \mid \tau \in \Gamma^*\}$, with initial node $u_\varepsilon = [\varepsilon]_\sim$, edges:

$$E_a^{\mathcal{U}} := \{([\tau]_\sim, [\tau c]_\sim) \mid \mathrm{act}(c) = a\}, \quad \text{for each } a \in A,$$

and colouring $\lambda^{\mathcal{U}}([\tau]_\sim) = \lambda(\tau)$. As $\sim$ has perfect recall, $\mathcal{U}(\mathcal{G})$ is a tree. Therefore, each node $u \in U$ identifies a unique path from u_ε, and we view strategies as functions $s\colon U \to A$.

Although structurally different, a game with imperfect information and the perfect-information game played on its information tree are the same (from the perspective of the active player), in the following sense.

▶ **Lemma 1.** *For every game $\mathcal{G}$ with indistinguishability relation $\sim$, there is a one-to-one correspondence mapping each strategy s in $\mathcal{U}(\mathcal{G})$ to a strategy $\tilde{s}$ in $\mathcal{G}$, such that $\tilde{s}(\tau) = s([\tau]_\sim)$, and their outcomes agree on colours: $\hat{\lambda}(\mathrm{Out}(\tilde{s})) = \hat{\lambda}^{\mathcal{U}}(\mathrm{Out}(s))$.*

Bisimulation

Further, we use bisimulation to relate strategies across games.

Let $\mathcal{G}$, $\mathcal{H}$ be perfect-information graphs with the same vocabulary $(v_\varepsilon, (E_a)_{a \in A}, \lambda)$. A *bisimulation* between $\mathcal{G}$ and $\mathcal{H}$ is a relation $Z \subseteq V^{\mathcal{G}} \times V^{\mathcal{H}}$ such that:

(Zig) for each $a \in A$ and $(u, u') \in E_a^{\mathcal{G}}$, there is $(v, v') \in E_a^{\mathcal{H}}$ with $(u', v') \in Z$;

(Zag) for each $a \in A$ and $(v, v') \in E_a^{\mathcal{H}}$, there is $(u, u') \in E_a^{\mathcal{G}}$ with $(u', v') \in Z$.

The graphs $\mathcal{G}$, $\mathcal{H}$ are *bisimilar* if their initial states are related: $(v_\varepsilon^{\mathcal{G}}, v_\varepsilon^{\mathcal{H}}) \in Z$.

A graph *homomorphism* from $\mathcal{G}$ to $\mathcal{H}$ is a function $h\colon V^{\mathcal{G}} \to V^{\mathcal{H}}$ that preserves the initial state: $h(v_\varepsilon^{\mathcal{G}}) = v_\varepsilon^{\mathcal{H}}$, the edges: $h(E_a^{\mathcal{G}}) \subseteq E_a^{\mathcal{H}}$ for all $a \in A$, and the colouring: $\lambda^{\mathcal{G}} = \lambda^{\mathcal{H}} \circ h$. A *p-morphism* is a homomorphism $h\colon \mathcal{G} \to \mathcal{H}$ such that the relation $\{(x, h(x)) \mid x \in V^{\mathcal{G}}\}$ forms a bisimulation. Equivalently, h is a homomorphism and, for every edge $(v, w) \in E_a^{\mathcal{H}}$ and node $x \in V^{\mathcal{G}}$ with $h(x) = v$, there exists an edge $(x, y) \in E_a^{\mathcal{G}}$ such that $h(y) = w$ [14].

▶ **Lemma 2** ([11]). *Let $\mathcal{G}$ and $\mathcal{H}$ be bisimilar game graphs. Then, $\mathcal{G}$ has a winning strategy iff $\mathcal{H}$ has one. Moreover, if there exists a p-morphism $h{:}\mathcal{G} \to \mathcal{H}$, then it takes every strategy s in $\mathcal{H}$, to a strategy $s \circ h$ in $\mathcal{G}$ such that the outcomes agree on colours: $\hat{\lambda}^{\mathcal{G}}(\mathrm{Out}(s \circ h)) = \hat{\lambda}^{\mathcal{H}}(\mathrm{Out}(s))$.*

5 The Reduction

For this section, let us fix a set $I = \{0, \dots, n\}$ of players with 0 being the active player, with an action set A, and the others being passive observers.

5.1 Simplifications

To simplify the presentation, we assume that all players in I share a common observation alphabet B. Moreover, we identify moves with observation profiles: $\Gamma = \prod_{i \in I} B$. Each player's observation function is defined trivially by $\beta^i(\tau c) = c^i$ for all $\tau \in \Gamma^*$ and $c \in \Gamma$.

These assumptions can be made without loss of generality. Any FIP instance can be brought to this form by adding a dummy observer and modifying the objective to check that the observations in the view of the active players match the original Mealy-automaton outputs, or else go to a sink state labelled with an even priority. This modification incurs an exponential blowup in size, but preserves visibility of the objective.

In the simplified model, we can characterise indistinguishability relations without reference to view graphs. Given a move $c \in \Gamma$, let $\mathrm{Link}(c) := \bigcup_{i \in I} \{i\} \times \mathrm{Com}^i(c^i)$ denote the set of all communication links enabled under c. For any player $i \in I$, let now $\mathrm{sync}^i(c)$ be is the set of players reachable from i via the reflexive-transitive closure of $\mathrm{Link}(c)$. Hence, $\mathrm{sync}^i(c)$ describes the set of players that communicate (directly or indirectly) with Player i under move c. For a coalition $J \subseteq I$, let $\mathrm{sync}^J(c) := \bigcup_{i \in J} \mathrm{sync}^i(c)$.

We note the following properties for later use: $\mathrm{sync}^J(c)$ always includes J, every coalition $L = \mathrm{sync}^J(c)$ that occurs as a synchronisation target is *autonomous*, meaning $L = \mathrm{sync}^L(c)$, intermediary coalitions $J \subseteq K \subseteq \mathrm{sync}^J(c)$ share the same target: $\mathrm{sync}^K(c) = \mathrm{sync}^J(c)$, and $\mathrm{sync}^{J \cup K}(c) = \mathrm{sync}^J(c) \cup \mathrm{sync}^K(c)$.

To describe how the information sets of a coalition J evolve with each move, we state the following lemma.

▶ **Lemma 3.** *Let $J \subseteq I$ be a coalition. Then, for all histories $\tau \in \Gamma^*$ and all moves $c \in \Gamma$:*

(i) *For the target coalition $L = \mathrm{sync}^J(c)$, we have $[\tau c]_{\sim^L} = \{\tau' d \mid \tau' \sim^L \tau \text{ and } d^L = c^L\}$.*
(ii) *For every intermediary coalition K with $J \subseteq K \subseteq \mathrm{sync}^J(c)$, we have $[\tau c]_{\sim^K} = [\tau c]_{\sim^J}$.*

Lemma 3 provides an inductive definition of indistinguishability in simplified FIP games: $[\tau c]_{\sim^J} = [\tau c]_{\sim^L} = \{\tau' d \mid \tau' \in [\tau]_{\sim^L} \text{ and } d^L = c^L\}$, for $L = \mathrm{sync}^J(c)$. Recall that, for the grand coalition, $\sim^I$ is the identity.

5.2 Abstraction morphism

As a first step towards our reduction, we map each history $\tau \in \Gamma^*$ to an abstract state from a finite domain. This state encodes the knowledge of every coalition containing the active player 0. For a coalition $J \subseteq I$, its own knowledge is information-consistent, hence visible; whereas knowledge of strictly larger coalitions $K \supsetneq J$ might not be information-consistent for coalition J — these values form a hidden part of the abstract state, called the configuration.

Knowledge is represented by a set of configurations, and configurations are profiles of knowledge sets indexed by coalitions. For the grand coalition, the configuration is just the state of the objective automaton.

For any $J \subseteq I$, let $J^{\uparrow} := \{K \subseteq I \mid J \subseteq K\}$ be the set of coalitions containing J, and $J^{\Uparrow} := J^{\uparrow} \setminus \{J\}$. We call any $J \in \{0\}^{\uparrow}$ an *active coalition*.

Let $\mathcal{F} = (\mathrm{act}, (\mathrm{sync}^J)_{J \in \{0\}^{\uparrow}}, \mathcal{A})$ be a FIP game in simplified form, with observation alphabet B, moves Γ, action map $\mathrm{act}\colon \Gamma \to A$, a profile of synchronisation maps $(\mathrm{sync}^J\colon \Gamma \to \wp(I))$, and a Mealy automaton $\mathcal{A}$ defining the objective. We refer to the corresponding one-player game as $\mathcal{G} := (\Gamma^*, \mathrm{act}, \sim, \lambda^{\mathcal{A}})$.

For each $J \in \{0\}^{\uparrow}$ and history $\tau \in \Gamma^*$, we define the *configuration* $\varphi_J(\tau)$ and the *knowledge set* $\Phi^J(\tau)$ by mutual induction:

- $\varphi_I(\tau) := \delta^{\mathcal{A}}(q_\varepsilon, \tau)$,
- $\varphi_J(\tau) := (\Phi^K(\tau))_{K \in J^{\Uparrow}}$ for $J \neq I$,
- $\Phi^J(\tau) := \{\varphi_J(\tau') \mid \tau' \sim^J \tau\}$.

The *abstract state* for coalition J is $h_J(\tau) := (\Phi^J(\tau), \varphi_J(\tau))$. We write Conf_J and KSet^J for the respective ranges of φ_J and Φ^J. In particular, $\mathrm{Conf}_I = Q$ and $\mathrm{Conf}_J \subseteq \prod_{K \in J^{\Uparrow}} \mathrm{KSet}^K$, while $\mathrm{KSet}^J \subseteq \wp(\mathrm{Conf}_J)$. The set of abstract states is $Q^{\mathcal{H}}_J \subseteq \mathrm{KSet}^J \times \mathrm{Conf}_J$. We omit indices for $J = \{0\}$ and write $h = h_{\{0\}}$.

Note that in any $(\Psi, \psi) \in Q^{\mathcal{H}}_J$, we have $\psi \in \Psi$. Also, the knowledge-set component of h_J is information-consistent by construction.

▶ **Lemma 4** (Preservation of indistinguishability). *If $\tau \sim^J \tau'$, then $\Phi^J(\tau) = \Phi^J(\tau')$, for all $\tau, \tau' \in \Gamma^*$ and $J \in \{0\}^{\uparrow}$.*

In particular, this holds for the active player singleton $J = \{0\}$, hence $\sim \subseteq \ker \Phi^{\{0\}}$. The abstraction map also respects the colour assigned by the evaluation automaton.

▶ **Lemma 5** (Preservation of evaluation colour). *If $h_J(\tau) = h_J(\tau')$, then $\lambda(\tau) = \lambda(\tau')$, for all τ, τ' and $J \in \{0\}^{\uparrow}$.*

Proof. The statement is immediate for $J = I$, since $\varphi_I(\tau) = \varphi_I(\tau')$ is a state q, so $h_I(\tau) = h_I(\tau') = (\{q\}, q)$ and $\lambda(\tau) = \lambda(\tau') = \lambda(q)$. Otherwise, if $J^{\Uparrow}$ is nonempty and $h_J(\tau) = h_J(\tau')$, then $\Phi^K(\tau) = \Phi^K(\tau')$ for all $K \in J^{\Uparrow}$, in particular $\Phi^I(\tau) = \Phi^I(\tau') = \{q\}$, and again $\lambda(\tau) = \lambda(\tau') = \lambda(q)$. ◀

Accordingly, we can map abstract states to colours via $\lambda^{\mathcal{H}}(\Phi, \varphi) := \lambda(\tau)$ for any history $\tau \in \Gamma^*$ such that $h(\tau) = (\Phi, \varphi)$.

An elementary, but important insight is that the abstract state of a history determines the set of abstract states of its information set.

▶ **Lemma 6** (Information commuting). *If $h_J(\tau) = h_J(\tau')$ then $h_J([\tau]_{\sim^J}) = h_J([\tau']_{\sim^J})$ for all τ, τ' and each active coalition J.*

Proof. Any such τ, τ' share $\Phi^J(\tau) = \Phi^J(\tau') =: \Psi$, so their information sets map to:

$$\begin{aligned} h[\tau]_{\sim^J} &= \{(\Phi^J(\pi), \varphi) \mid \varphi \in \Phi^J(\tau)\} = \{(\Psi, \varphi) \mid \varphi \in \Psi\} \\ &= \{(\Phi^J(\tau'), \varphi) \mid \varphi \in \Phi^J(\tau')\} = h[\tau']_{\sim^J}. \end{aligned}$$

◀

Regularity

To show that the abstraction map h is indeed a homomorphism on the repeated game $\mathcal{G}$, we need to prove that it preserves the move operations: for every pair of histories $\tau, \tau' \in \Gamma^*$, and every $c \in \Gamma$, if $h(\tau) = h(\tau')$, then $h(\tau c) = h(\tau' c)$. Towards this, we define operations on abstract states that mimic the information updates triggered either by local observations or by communication.

Intuitively, when a coalition J communicates with coalition L, it is as if J communicates with $J \cup L$. Therefore, we focus on communication between coalitions that are comparable with respect to inclusion. We define, for every increasing pair of active coalitions $J \subseteq L$, a lifting operator $\lceil \cdot \rceil_L^J \colon \mathrm{KSet}^L \to \mathrm{KSet}^J$ that maps every knowledge set Φ for L to a knowledge set $\lceil \Phi \rceil_L^J$ for J, by setting $\lceil \Phi \rceil_J^J := \Phi$ and

$$\lceil \Phi \rceil_L^J := \{ \lceil \Phi, \varphi \rceil_L^J \mid \varphi \in \Phi \}, \quad \text{for all } L \in J^{\dagger}.$$

The construction uses an auxiliary operator that maps every abstract state $(\Phi^L, \varphi_L) \in \mathrm{KSet}^L \times \mathrm{Conf}_L$ for the larger coalition L to a configuration $\lceil \Phi, \varphi \rceil_L^J \in \mathrm{Conf}^J$ of the smaller one J:

$$\lceil \Phi, \varphi \rceil_L^J := (\Psi^K)_{K \in J^{\dagger}} \quad \text{with } \Psi^K := \begin{cases} \lceil \Phi \rceil_L^K & \text{if } K \subseteq L; \\ \lceil \varphi^{K \cup L} \rceil_{K \cup L}^K & \text{otherwise.} \end{cases}$$

By unfolding the definition, we can see that the operators compose naturally.

▶ **Lemma 7** (Composition of lifting). *For all increasing coalitions $J \subseteq K \subseteq L$,*

$$\left\lceil \lceil \Phi \rceil_L^K \right\rceil_K^J = \lceil \Phi \rceil_L^J, \quad \textit{for every knowledge set } \Psi \in \mathrm{KSet}^J.$$

The following lemma states that the operator indeed captures that the information of a coalition L is revealed to smaller (hence, possibly less informed) coalition J.

▶ **Lemma 8** (Communication). *For coalition J and a move $c \in \Gamma$, consider the target coalition $L = \mathrm{sync}^J(c)$. Then, for every history $\tau \in \Gamma^*$,*

$$\lceil \Phi^L(\tau c) \rceil_L^J = \Phi^J(\tau c).$$

Proof. We proceed by induction starting with the grand coalition. As $\mathrm{sync}^I(d) = I$, the base case is trivial. For the induction step, suppose that the statement holds for all coalitions strictly larger than J. By unfolding the definitions, we obtain:

$$\begin{aligned} \lceil \Phi^L(\tau c) \rceil_L^J &= \left\{ \left(\lceil \Phi^L(\tau c), \varphi \rceil_L^K \right)_{K \in J^{\dagger}} \mid \varphi \in \Phi^L(\tau c) \right\} && \text{(Set lifting, configuration)} \\ &= \left\{ \left(\lceil \Phi^L(\tau' d), \varphi_L(\tau' d) \rceil_L^K \right)_{K \in J^{\dagger}} \mid \tau' d \sim^L \tau c \right\} && \text{(Lemma 3(i), Lemma 4)} \\ &= \left\{ \left(\lceil \Phi^{L \cup K}(\tau' d) \rceil_{L \cup K}^K \right)_{K \in J^{\dagger}} \mid \tau' d \sim^L \tau c \right\}. && \text{(Abstract-state lifting)} \end{aligned}$$

Observe that for every $K \in J^{\dagger}$, we have $\mathrm{sync}^K(d) = \mathrm{sync}^{J \cup K}(d) = \mathrm{sync}^J(d) \cup \mathrm{sync}^K(d) = L \cup \mathrm{sync}^K(d)$. On the other hand $\mathrm{sync}^{L \cup K}(d) = \mathrm{sync}^L(d) \cup \mathrm{sync}^K(d) = L \cup \mathrm{sync}^K(d)$. Hence, the coalitions $L \cup K$ and K share the same target $T := \mathrm{sync}^{L \cup K}(d) = \mathrm{sync}^K(d)$, strictly larger than J. Therefore, we can develop the knowledge sets in the last expression as follows:

$$\begin{aligned} \lceil \Phi^{L \cup K}(\tau' d) \rceil_{L \cup K}^K &= \lceil \lceil \Phi^T(\tau' d) \rceil_T^{L \cup K} \rceil_{L \cup K}^K && \text{(Induction hypothesis for } \mathrm{sync}^{K \cup L}(d) = T) \\ &= \lceil \Phi^T(\tau' d) \rceil_T^K && \text{(Lemma 7)} \\ &= \Phi^K(\tau' d) && \text{(Induction hypothesis for } \mathrm{sync}^K(d) = T). \end{aligned}$$

In conclusion, $\lceil \Phi^L(\tau c) \rceil_L^J = \{ \varphi_J(\tau' d) \mid \tau' d \sim^L \tau c \} = \{ \varphi_J(\tau' d) \mid \tau' d \sim^J \tau c \} = \Phi^J(\tau c)$, according to Lemma 3(ii). ◀

With the lifting operators in place, we define update operations on configurations, knowledge sets, and ultimately on abstract states, for each active coalition.

For configurations, define $\mathrm{succ}_J\colon \mathrm{Conf}_J \times \Gamma \to \mathrm{Conf}_J$ as follows. For the grand coalition, this coincides with the automaton update: $\mathrm{succ}_I(\varphi, c) \coloneqq \delta^{\mathcal{A}}(\varphi, c)$. For other coalitions J, the update is componentwise, based on synchronisation targets:

$$\mathrm{succ}_J(\varphi, c) \coloneqq \big(\mathrm{Succ}^K(\varphi^{\mathrm{sync}^K(c)})\big)_{K \in J^\dagger}.$$

Knowledge sets are updated by a partial function $\mathrm{Succ}^J\colon \mathrm{KSet} \times \Gamma \to \mathrm{KSet}^J$, defined on pairs (Φ, c) with $\Phi \in \mathrm{KSet}^L$ and $L = \mathrm{sync}^J(c)$:

- If J is autonomous ($J = L$), set
$$\mathrm{Succ}^J(\Phi, c) \coloneqq \{\mathrm{succ}_J(\psi, d) \mid \psi \in \Phi,\ d^L = c^L\},$$
- Otherwise, for $L \neq J$, define
$$\mathrm{Succ}^J(\Phi, c) \coloneqq \lceil \mathrm{Succ}^L(\Phi, c)\rceil^J_L.$$

The state update $\delta^{\mathcal{H}}_J\colon Q^{\mathcal{H}}_J \to Q^{\mathcal{H}}_J$ is defined as:

- If J is autonomous, then
$$\delta^{\mathcal{H}}_J((\Phi, \varphi), c) = (\mathrm{Succ}^J(\Phi, c),\ \mathrm{succ}_J(\varphi, c)),$$
- Otherwise, letting $L = \mathrm{sync}^J(c)$, set
$$\delta^{\mathcal{H}}_J((\Phi, \varphi), c) \coloneqq (\mathrm{Succ}^J(\varphi^L, c),\ \mathrm{succ}_J(\varphi, c)).$$

▶ **Lemma 9** (Preservation of moves)**.** *Let J be an active coalition. Then,*

$$h_J(\tau c) = \delta^{\mathcal{H}}_J(h(\tau), c), \quad \textit{for all } \tau \in \Gamma^*,\ c \in \Gamma.$$

Proof. We detail the inductive argument for knowledge sets to show that for $L = \mathrm{sync}^J(c)$,

$$\mathrm{Succ}^J(\Phi^L(\tau), c) = \Phi^J(\tau c).$$

The base case, with $I = \mathrm{sync}^I(c)$, concerns moves in the objective automata:

$$\begin{aligned}\mathrm{Succ}^I(\Phi^I(\tau), c) &= \{\mathrm{succ}_I(\varphi, d) \mid \varphi \in \Phi^I(\tau), c^I = d^I\}\\ &= \{\delta(\delta(q_\varepsilon, \tau), c)\} = \{\delta(q_\varepsilon, \tau c)\} = \mathrm{Succ}^I(\tau c).\end{aligned}$$

Inductive step: if J is autonomous,

$$\begin{aligned}\mathrm{Succ}^J(\Phi^J(\tau), c) &= \{\mathrm{succ}_J(\varphi, d) \mid \varphi \in \Phi^J(\tau), d^J = c^J\} && \text{(Definition of } \mathrm{Succ}^J)\\ &= \{\mathrm{succ}_J(\varphi_J(\tau'), d) \mid \tau' \sim^J \tau, d^J = c^J\} && \text{(Knowledge set } \Phi^J)\\ &= \{\big(\mathrm{Succ}^K(\Phi^{\mathrm{sync}^K(d)}(\tau'), d)\big)_{K \in J^\dagger} \mid \tau' d \sim^J \tau c\} && \text{(Lemma 3(i))}\\ &= \{(\Phi^K(\tau' d))_{K \in J^\dagger} \mid \tau' d \sim^J \tau c\} = \Phi^J(\tau c) && \text{(Induction hypothesis).}\end{aligned}$$

Otherwise, if $J \neq L \coloneqq \mathrm{sync}^J(c)$:

$$\begin{aligned}\mathrm{Succ}^J(\Phi^L(\tau), c) &= \lceil \mathrm{Succ}^L(\Phi^L(\tau), c)\rceil^J_L && \text{(Definition of } \mathrm{Succ}^J)\\ &= \lceil \Phi^L(\tau c)\rceil^J_L = \Phi^J(\tau c). && \text{(Induction hypothesis, Lemma 8)}\end{aligned}$$

The statement extends to configurations and abstract states by structural induction. ◀

We define the automaton $\mathcal{H} = (Q^{\mathcal{H}}, \Gamma, Q^{\mathcal{H}}, \delta^{\mathcal{H}}, q^{\mathcal{H}}_\varepsilon, \delta^{\mathcal{H}})$ on the set $Q^{\mathcal{H}}$ of abstract states, with input alphabet Γ, transition function $\delta^{\mathcal{H}}_0$, initial state $q^{\mathcal{H}}_\varepsilon \coloneqq h(\varepsilon)$ and $\lambda^{\mathcal{H}}$ as an output function. With the transition function used for the output, this automaton defines h. By projection on the first component, we obtain, for every intput history τ, the knowledge set $\Phi^{\{0\}}(\tau)$ as output. In conclusion, the functions h and $\Phi^{\{0\}}$ are regular.

5.3 Knowledge Quotient

Now, we construct a new game from the automaton $\mathcal{H}$ by collapsing abstract states that share the same knowledge set. Concretely, we define the perfect-information game graph $\mathcal{K} = (Q^{\mathcal{K}}, (E_a^{\mathcal{K}})_{a \in A}, q_\varepsilon^{\mathcal{K}}, \lambda^{\mathcal{K}})$ with the following ingredients. The set of positions consists of knowledge sets: $Q^{\mathcal{K}} := \mathrm{KSet}^0 = \Phi^{\{0\}}(\Gamma^*)$. For every action $a \in A$, the set E_a contains the edge from Φ to Φ' if there exist abstract states (Φ, φ) and (Φ', φ') in $Q^{\mathcal{H}}$ and a move $c \in \Gamma$ such that $\delta^{\mathcal{H}}((\Phi, \varphi), c) = (\Phi', \varphi')$. The initial position is obtained from the initial abstract state $q_\varepsilon^{\mathcal{H}} = (\Phi_\varepsilon, \varphi_\varepsilon)$ by taking the first component $q_\varepsilon^{\mathcal{K}} := \Phi_\varepsilon$. As the objective coloring $\lambda^{\mathcal{A}}$ and the knowledge-set map $\Phi^{\{0\}}$ are both information consistent, we can lift the coloring from the configuration component of the states in $Q^{\mathcal{H}}$ defined in Lemma 5 to the knowledge set in their knowledge-set component: for any $\Phi \in Q^{\mathcal{K}}$, we assign $\lambda^{\mathcal{K}}(\Phi) := \lambda^{\mathcal{H}}(\Phi, \varphi)$ for some/any abstract state $(\Phi, \varphi) \in Q^{\mathcal{H}}$ with Φ in the first component.

The same game graph can be obtained directly from the original repeated game by taking the quotient $\Gamma/_{\ker \Phi}$. Indeed, $Q^{\mathcal{K}} = \Phi^0(\Gamma^*)$, $E_a = \{(\Phi^0(\tau), \Phi^0(\tau c)) \mid \tau \in \Gamma^*, c \in \Gamma\}$ for all $a \in A$, $q^\varepsilon = \Phi^0(\varepsilon)$, and $\lambda^{\mathcal{K}}(\Phi^0(\tau)) = \lambda(\tau)$, for all histories $\tau \in \Gamma^*$.

Let $k\colon \Gamma^* \to \mathrm{KSet}^0$ be the quotienting map that associates to every history its knowledge set: $k(\tau) := \Phi^{\{0\}}(\tau)$. Since k is information consistent with respect to $\sim$, it induces a map $\bar{k}\colon \Gamma^*/_\sim \to \mathrm{KSet}^0$ with $\bar{k}([\tau]) = k(\tau) = \Phi^0(\tau)$ The map $\bar{k}$ is actually a graph homomorphism from the information quotient $\mathcal{U}(\mathcal{G}) = \mathcal{G}^*/\sim$ of the original game to $\mathcal{K} = k(\mathcal{G})$, as it preserves edges, actions and the coloring λ.

As an immediate consequence of Lemma 6, the factor map describes a functional bisimulation.

▶ **Lemma 10** (p-morphism). *For every history $\tau c \in \Gamma^*$, and every $\tau' \in \Gamma^*$ with $k(\tau') = k(\tau)$, there exists a history $\tau'' \sim \tau'$ such that $k(\tau'' c) = k(\tau c)$.*

Accordingly, $\{([\tau], \Phi^0(\tau)) \in (\Gamma^*/\sim) \times \mathrm{KSet} \mid \tau \in \Gamma^*\}$ is a bisimulation between $\mathcal{U}(\mathcal{G})/\sim$ and $k(\mathcal{G})$ and, by Lemma 2, we can conclude that the games are equivalent.

▶ **Corollary 11** (Bisimulation). *The finite parity game $\mathcal{K}$ on $k(\Gamma^*)$ is bisimilar to the information quotient $\mathcal{U}(\mathcal{G})$ of the original FIP game on $\Gamma^*/\sim$. Moreover, for every winning strategy s in the parity game $\mathcal{K}$, the strategy $s \circ k = s \circ \Phi^{\{0\}}$ is winning in the FIP game $\mathcal{G}$.*

By positional determinacy of parity games, if there exists a winning strategy in $\mathcal{K}$, then there exists one represented by a labelling $s\colon Q^{\mathcal{K}} \to A$ of positions with actions. Recall that the positions of $\mathcal{K}$ correspond to knowledge sets $\Phi^0(\tau)$ output by the automaton $\mathcal{H}$. By composing the output function of the automaton with the action labelling s, we thus obtain a winning strategy for the game $\mathcal{G}$ at the outset.

▶ **Theorem 12.** *The synthesis problem for FIP games with parity objectives is decidable. Whenever a winning strategy exists, we can effectively construct a Mealy automaton that defines one.*

Given a FIP game with n observers, the pre-processing step from Subsection 5.1 adds one observer, and thus the size $|\mathcal{K}|$ of the perfect-information game constructed in the reduction is $(n+1)$-fold exponential in the size of the FIP game. Note that the number of priorities for the parity objective does not change. Since perfect-information parity games can be solved in time at most exponential in the number of priorities (actually, in quasi-polynomial time in the number of nodes, see e.g. [17, 19]), we derive an $(n+1)$-EXPTIME upper bound for the synthesis problem.

Theorem 12 extends to all visible objectives for which perfect-information games are decidable, such as mean-payoff or discounted-sum conditions as objectives.

6 Complexity Lower Bound

We show a matching lower bound for reachability objectives, by a reduction of the membership problem for alternating n-EXPSPACE Turing machines, which is $(n+1)$-EXPTIME-complete, to the synthesis problem for FIP games with n observers.

Intuitively, the FIP game simulates an execution of an alternating Turing machine M on an input word w of length ℓ, where the player chooses transitions in existential states, and Nature chooses transitions in universal states. The winning condition requires the player to announce successive configurations of M—consisting of the tape content (of n-fold exponential size), the head position, and the control state—until an accepting configuration is reached.

The main difficulty is to verify the consistency of these configurations using only polynomial-size Mealy automata. To address this, we use the structure of FIP games with imperfect information and n observers. Instead of checking transitions directly, the game allows Nature to challenge the equality of successive configurations. Nature may mark differing positions in the two configurations by sending observations to designated observers. These marks are hidden to the player until a communication occurs.

If the marked bits differ, the player may claim the marks refer to different positions, which reduces to checking inequality between two $(n-1)$-fold exponential-size numbers. The player does so by pointing out a differing bit in their binary encodings and announcing its address (a $(n-2)$-fold exponential-size number)—again recursively allowing Nature to challenge this claim. Each level of this recursive protocol uses one of the n observers. Eventually, the values are small enough to be verified by the objective automaton.

The encoding uses nested counters: a level-n counter is a sequence of bits, each followed by its address as a level-$(n-1)$ counter. This encoding is directly inspired by a similar definition in previous work [13, Section 4.2]. We define actions for the player to emit bits, counters, Turing machine transitions, and specific claims like r^{01} or r^{10} asserting bit differences. Nature's moves include sending observations (marks), initiating communication, and branching to verification components in the Mealy automaton.

Correct encoding is enforced by separate Mealy components that check syntactic properties: correct counter format, input encoding, control state updates, and that configurations follow transitions. Nature can branch to any component. The marking mechanism allows checking bit-equality claims, and address comparison proceeds recursively with up to n observers.

We argue that a small (polynomial-size) Mealy automaton can store the values of the marked bits in order to verify the player's claims (r^{01} or r^{10}). Storing two bits at each level (hence $2n$ bits) would suffice if Nature never challenges the format of the counters announced by the player. However, if a challenge occurs, the game restarts at a lower level to check the counter format, which requires storing only $2(n-1)$ bits at that level—assuming no further challenges. Repeating this argument recursively, the total number of bits needed for storage is $O(n^2)$. Since n is fixed independently of the input word w and the Turing machine M, the resulting Mealy automata remain polynomial in size. This ensures that the overall reduction is polynomial-time.

We conclude that the player has a winning strategy in the FIP game if and only if M accepts w.

▶ **Theorem 13.** *The synthesis problem for FIP games with n observers is $(n+1)$-EXPTIME-complete, both for parity and reachability objectives.*

References

1. André Arnold, Aymeric Vincent, and Igor Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical computer science*, 303(1):7–34, 2003.
2. 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.
3. D. Berwanger and L. Doyen. Observation and distinction. representing information in infinite games. *Theory of Computing Systems*, 67(1):4–27, 2023.
4. D. Berwanger, A. B. Mathew, and M. van den Bogaard. Hierarchical information and the synthesis of distributed strategies. *Acta Informatica*, 55(8):669–701, 2018.
5. Dietmar Berwanger, Lukasz Kaiser, and Bernd Puchala. A perfect-information construction for coordination in games. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*, volume 13 of *LIPIcs*, pages 387–398. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.
6. J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.
7. Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasi-polynomial time. *SIAM J. Comput.*, 51(2):17–152, 2022.
8. K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games of incomplete information. *Logical Methods in Computer Science*, 3(3:4), 2007.
9. A. Church. Logic, arithmetics, and automata. *Proc. Int. Congr. Math.*, pages 23–35, 1962.
10. Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment: Crash failures. *Inf. Comput.*, 88(2):156–186, 1990.
11. Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. *SIAM J. Comput.*, 34(5):1159–1175, 2005.
12. Bernd Finkbeiner and Sven Schewe. Uniform distributed synthesis. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 321–330. IEEE Computer Society, 2005.
13. B. Genest, H. Gimbert, A. Muscholl, and I. Walukiewicz. Asynchronous games over tree architectures. In *Proc. of ICALP: Automata, Languages, and Programming*, LNCS 7966, pages 275–286. Springer, 2013.
14. Valentin Goranko and Martin Otto. Model theory of modal logic. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 249–329. North-Holland, 2007.
15. Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Proceedings of the fourteenth annual ACM symposium on theory of computing*, pages 60–65, 1982.
16. D. Harel and A. Pnueli. On the development of reactive systems. In Krzysztof R. Apt, editor, *Logics and Models of Concurrent Systems*, pages 477–498, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.
17. M. Jurdzinski, R. Morvan, and K. S. Thejaswini. Universal algorithms for parity games and nested fixpoints. In *Principles of Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, LNCS 13660, pages 252–271. Springer, 2022.
18. Orna Kupferman and Moshe Y. Vardi. Synthesizing distributed systems. In *Proc. of LICS '01*, pages 389–398. IEEE Computer Society Press, June 2001.
19. Karoliina Lehtinen, Pawel Parys, Sven Schewe, and Dominik Wojtczak. A recursive approach to solving parity games in quasipolynomial time. *Log. Methods Comput. Sci.*, 18(1), 2022.
20. Tal Mizrahi and Yoram Moses. Continuous consensus with ambiguous failures. *Theor. Comput. Sci.*, 411(34-36):3031–3041, 2010.
21. M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

22 A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proc. of FOCS: Foundations of Computer Science*, pages 746–757. IEEE, 1990.

23 M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–35, 1969.

24 M. O. Rabin. *Automata on Infinite Objects and Church's Problem.* American Mathematical Society, Boston, MA, USA, 1972.

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

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

27 W. Thomas. On the synthesis of strategies in infinite games. In *Proc. of STACS: Symposium on Theoretical Aspects of Computer Science*, LNCS 900. Springer, 1995.

28 W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.

29 Thomas YC Woo and Simon S Lam. A lesson on authentication protocol design. *ACM SIGOPS Operating Systems Review*, 28(3):24–37, 1994.