- -

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

Department of Computer Science and Engineering

- -

S Y N O P S I S

of the Ph.D. thesis entitled

Games on Asynchronous Transition Systems

Proposed to be submitted in

Partial fulfilment of the requirements for the Degree of

DOCTOR OF PHILOSOPHY

of the

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

by

Nehul Jain
Roll 134050006

Supervisor : Prof. Bharat Adsul

- -

Introduction

Full information two-player graph games [6] are played by two competing players. The study of these games has been central to *reactive synthesis*. The two players we call *system* and *environment* alternate to make their moves. This generates a sequence of interactions. It is expected that this sequence of interactions satisfies certain properties for the system to win. These desirable properties are expressed as specifications using some formalism. Two-player full information games are well studied.

There are various approaches to automatic formal synthesis. *Church's synthesis problem* asks how a system should respond to the environment's moves when specification is regular. System and Environment interact in an interleaved fashion.

The problem was solved by Büchi and Landweber in 1969. A modern view of their solution,demonstrated that finite-state procedures suffice to meet the requirements. It turns out that Church's Problem can be studied as an infinite two-player game, where a player's goal is to satisfy a given *state-based winning condition* while producing an output stream in response to the input stream. In this game-theoretic setting, the synthesis problem reduces to finding a winning strategy for the system. This perspective connects naturally to well-studied classes of infinite games—such as safety, reachability, and parity games. Parity games generalize both safety and reachability by allowing the winning condition to be defined through an assignment of priorities (non-negative integers) to game positions. The system wins if the maximum priority that occurs infinitely often along the play is even. Parity conditions are particularly important because they can express a wide range of liveness and fairness properties. While there are polynomial-time algorithms for solving safety and reachability games, no polynomial-time algorithm is known for solving parity games. Notably, for all three objectives—safety, reachability, and parity—there always exist *zero-memory* (memoryless) strategies: the system can choose its next move based only on the current state, without needing to remember the history of the play.

In the presence of concurrent components, there has been significant work aimed at finding logics and algorithms that make the synthesis of reactive systems both efficient and scalable.

In the foundational work on distributed synthesis [10] the authors consider finite-state distributed reactive systems. A set of processes communicates synchronously using a set of single-reader and single-writer variables. Here it is proved that realizing even propositional specification is undecidable in general, and non elementarily decidable for a very restricted class of pipeline architecture.

There have been various attempts to incorporate features such as independence between distributed components, concurrency, and communication into a formal computational model. Significant work has focused on models where components share a *causal past*—that is, the history of events and decisions that causally precede and influence a given point in the system's execution. By

explicitly tracking this causal history, these models can more accurately capture the partial ordering of events that arises in concurrent and distributed systems.

Works such as [3, 7] have introduced richer models in which processes have access to their entire *causal past*, thereby broadening the range of problems that can be addressed. These models typically rely on asynchronous automata introduced by Zielonka, which operate over Mazurkiewicz traces: algebraic structures that capture the partial order of concurrent events. In this setting, each process evolves independently, and synchronizations allow the processes to share their causal past, enabling coordinated decision-making. In particular, [3] introduced the notion of causal past and addressed the distributed controller synthesis problem in series-parallel systems and established that controlled reachability is decidable. The work [7] introduced systems with connectedly communicating processes and proved that the MSO theory in this setting is decidable, which implies that the associated distributed synthesis problems are decidable.

Related Work

Two closely related lines of work are *control games* and *Petri games*.

Control games [4] are played on deterministic asynchronous automata over a distributed alphabet, where processes make decisions based on their causal past. In such games, processes move independently on disjoint actions and synchronize on overlapping actions, at which point they share the information they might have stored in their local states, effectively gaining access to their entire causal past. The controller synthesis problem is formulated, where the specification is given over a deterministic asynchronous automaton on traces. The set of actions is partitioned into controllable and uncontrollable ones. At any state, a process may block some of the controllable actions, based on its causal past, while uncontrollable actions can never be blocked. A action can be executed only if it is allowed by all participating processes.

These games are shown to be decidable in the interesting setting of acyclic architectures [4] whose underlying process-communication graph is acyclic. More general formulations like decomposable games have also been identified and proven to be decidable. However, the remarkable undecidability results for asynchronous games with simple objectives such as local reachability or termination or deadlock-freeness, even in systems with six processes, from [5] highlights the complexity of distributed synthesis problems.

Petri games [2] are distributed games played on Petri nets. The players are tokens on a Petri net. The places in the underlying Petri net are divided into system and environment places. Players places make decisions based on their causal history which they come to know on synchronizations at joint actions. Several interesting classes of Petri games have been shown to be decidable. In particular, Petri games with a bounded number of system players against a single environment player as well as Petri games with a single system player against a bounded number of environment players are shown to be decidable. Later work shows that Petri games with global winning conditions are undecidable, even

with only two system players and one environment player. The global winning condition is chosen to enforce certain linearizations of the parallel runs using it to encode Post Correspondence Problem. Further work also establishes formal connections with control games.

Clearly it is desirable to identify different settings of the distributed synthesis problems which are interesting, practically well motivated and render them decidable.

Our contributions

In this work we propose one such model called *asynchronous transition system games (ATS games)*. These games are played on a non-deterministic asynchronous transition system involving an environment and a distributed system composed of cooperating processes. A play is an interleaved sequence of *environment* and *system* moves. The environment acts as a scheduler of actions without revealing any information about previous scheduling events. The system responds by choosing an enabled transition.

To model concurrent behavior formally, we rely on Mazurkiewicz traces [1, 8]. These traces capture causality and concurrency among events distributed across multiple processes

A *trace* is a labeled partial order, where events are occurrences of actions from a distributed alphabet within a finite set of processes. *Configurations* of a trace are finite subsets of events that are closed under the causal past relation.

An *asynchronous transition system (ATS)* $A = (\{S_i\}, \{\xrightarrow{a}\})$ over $\widetilde{\Sigma}$ is defined by finite non-empty set of local i-states S_i and a *non-deterministic* transition relation $\xrightarrow{a} \subseteq S_a \times S_a$ on the states of participating processes $S_a = \Pi_{i \in \mathrm{loc}(a)} S_i$ for each action $a \in \Sigma$. Each process has its own local states, and transitions only affect the states of the processes participating in the action, ensuring locality. A run of A on a trace t starting from an initial state, maps every configuration of t to a global state, respecting these local transitions.

Importantly, our approach can be viewed as a natural extension of the classical Büchi–Landweber synthesis method. In the sequential setting, synthesis is reduced to solving a game on an automaton derived from the specification. We generalize this idea to the distributed, asynchronous setting by defining games on non-deterministic asynchronous transition systems. This lifts the automaton-to-game translation central to classical synthesis into a richer, concurrent framework.

ATS games

Definition 1. *The ATS game* $\mathcal{G} = (A, s_0, \mathrm{Win})$ *is defined by an asynchronous transition system* $A = (\{S_i\}, \{\xrightarrow{a}\})$ *over a distributed alphabet* $\widetilde{\Sigma}$ *on processes* $\{i_1, i_2 ...\}$*, an initial global state* s_0*, and a winning condition* Win.

A play in an ATS game is the ongoing interaction between the environment and the distributed system: the environment schedules actions, and the pro-

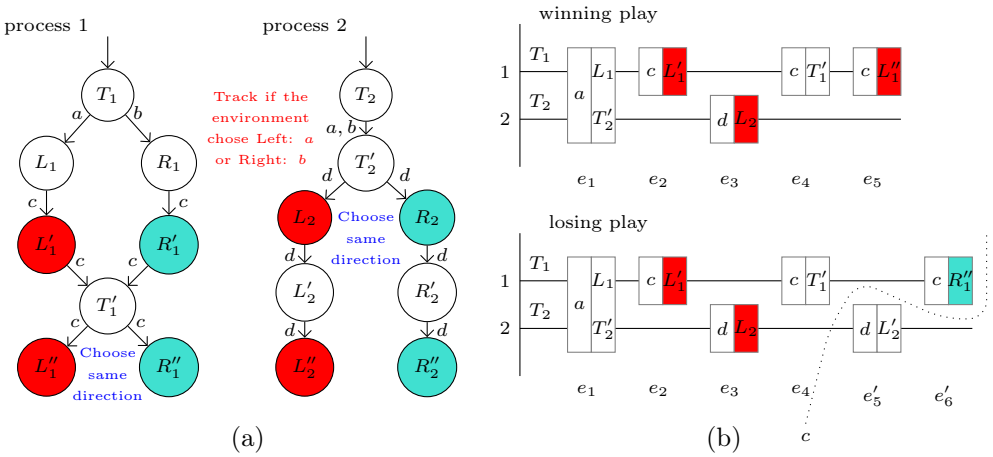

Figure 4: An ATS game: unsafe state = $\{L_1', L_1''\} \times \{R_2, R_2''\} \cup \{R_1', R_1''\} \times \{L_2, L_2''\}$

cesses respond by choosing available transitions, thereby advancing the global state. For example, under a global safety objective, the system wins if no unsafe global state is ever reached during the play.

A distributed *play* is modeled as a tuple consisting of a trace and a run of the transition system starting from the initial state. The trace captures the environment/scheduler's choices, while the run reflects the system/processes' responses. The processes participating in an action know the entire causal past before they decide the next state to visit in a play. This is *not the case in controller synthesis problem* described earlier. There the individual process enable or disable a set controllable actions based on their individual causal past which are then scheduled allowing them to share their causal past. We formalize the notion of a distributed strategy for the distributed system.

Definition 2. *A distributed strategy is a partial function mapping each partial play to a global state of the ATS that advises the processes on how to respond to the environment's actions, based on their collective causal history.*

We establish that a distributed strategy is determined by its effect on the prime traces in its domain. A strategy is winning if all maximal plays conforming to it are winning.

Our work also introduces and formalizes the notion of a finite-state distributed strategy, represented by a memory automaton. This automaton deterministically specifies the processes' responses to each environment move and updates its memory accordingly. A memory automaton thus induces a distributed strategy, and any play conforming to the automaton also conforms to the induced strategy. Our investigation into two process and CDM games reveals that they admit such a finite-state implementation whenever a winning strategy exists.

An example illustrates a game where processes must remember past states to avoid unsafe conditions, demonstrating the need for memory in winning strategies. The example includes a strategy idea that guides the processes' actions to ensure a winning play.

Example 3. *We consider an example illustrated in Fig. 3. In Fig. 3(a) system consists of the processes* 1 *and* 2*. The only synchronizations are* $(T_1, T_2) \xrightarrow{a} (L_1, T_2')$ *and* $(T_1, T_2) \xrightarrow{b} (T_2', T_2')$ *on joint actions* a *and* b*. The purely local transitions are clear from the figure. We now intuitively explain a winning distributed strategy illustrated in Fig. 3(a) whilst using the example of some partial plays in Fig. 3(b).*

The top play begins with the environment choosing the joint action a *at* (T_1, T_2) *and the distributed team advancing the global state to* (L_1, T_2') *by choosing the matching transition. This is depicted by event* e_1*, with the environment's choice* a *shown as the left label and the response of participating processes as the right label. Events* e_2 *and* e_3 *occur concurrently, with each participating process updating its local state independently, without affecting others. However,* $e_1 < e_2$ *and* $e_1 < e_3$*, as each of these events shares at least one common participating process with* e_1*, establishing a causal dependency. Note that, the local choices shown in blue text in states* T_1' *and* T_2' *to choose the same direction as environment are really made after the first synchronization based on the choice of the environment shown in red text at joint action* a *or* b*.*

In the bottom play the choice made in state T_1' *by process* 1 *is to choose differently than the environment resulting in the system loosing by visiting an unsafe state in the partial play given by events* $\{e_1, e_2, e_3, e_4, e_6'\}$ *as shown by dotted line labeled* c_1*.*

Two processes ATS games

ATS games for a single process correspond directly to classical two-player graph games. In this setting, there is no concurrency, and the sequential nature of the game aligns perfectly with standard graph games. Consequently, memoryless (zero-memory) fixpoint solutions suffice.

This makes the case of two-process games particularly interesting, as the presence of concurrency fundamentally changes the landscape. For example, Peterson's mutual exclusion problem is already relevant at this level: mutual exclusion itself can be formulated as a global safety objective, while ensuring liveness for each process naturally leads to local parity objectives.

We discuss strategies and algorithms for solving ATS games with two processes under various state-based winning conditions, highlighting both the decidability of winning strategies and the memory structures required to implement them.

We first consider the global safety objective, where the system wins if no "bad" global state is ever reached during the play. Next, we study local and global reachability objectives. In local reachability, each individual process must eventually visit its own designated goal set at least once during the play. In contrast, under global reachability, the system wins if there exists a configuration of the play in which a global goal state is reached simultaneously.

Surprisingly, for none of these objectives do memoryless strategies suffice; winning strategies necessarily require some form of memory. Nevertheless,

whenever a winning strategy exists, it can always be implemented using a finite amount of memory, and we describe the precise memory structures needed.

For all these objectives—global safety, local reachability, and global reachability—we present fixpoint algorithms, inspired by the sequential (single-process) setting, that rely on the standard notions of traps and attractors from graph games.

The key results of this section are as follows-

Theorem 5. *Consider ATS games with two processes:*

1. *Deciding the existence of a winning strategy for* global safety *and* local reachability *objectives is NP-complete.*
2. *Deciding the existence of a winning strategy for* global reachability *objectives is solvable in NEXPTIME and is PSPACE-hard.*

Theorem 6. *Consider a two-process game. Whenever there exists a winning strategy:*

1. *For* global safety *and* local reachability *objectives, there exists a winning distributed memory automaton in which each process stores, in its local memory, the global state at the last synchronization.*
2. *For* global reachability *objectives, there exists a winning distributed memory automaton in which each process stores, in its local memory, both the global state at the last synchronization point and the set of local states visited since that synchronization.*

Moreover, these memory automata are optimal. There exist instances of global safety and local reachability games where every winning strategy necessarily requires a distributed memory automaton whose size is polynomial in the size of the game. And there also exist instances of global reachability games where every winning strategy requires a distributed memory automaton whose size is exponential in the size of the game.

ATS games with a central decision maker

The chapter introduces the concept of *Central Decision Maker (CDM) games* within the framework ATS games. In an ATS some actions can be determinisitc and some may not be determinisitc. An action a is said to be *not deterministic* if there exists a state that has at least two distinct a-successors. A set of processes is called the *decision makers* if, for every not deterministic action a, at least one of the processes participates in a.

A game is called a *central decision maker (CDM) game* if there is exactly one process that participates in every not deterministic action.

Practical applications of CDM games include version control systems in which a central decision maker (e.g., a project maintainer) evaluates all change requests. Systems like Git or SVN can be modeled in this way. These games are

non trivial because, although all system choices are made by a single process, a winning strategy still requires the active involvement of other processes. Other process gather and propagate relevant information from the causal past through synchronization, enabling the central decision maker to act correctly. Our work essentially addresses all games where all system choices are causally ordered.

Our work shows how to transform sequential strategies from associated sequential games into distributed strategies for CDM games, focusing on the existence of winning strategies under global safety and local parity winning conditions.

We describe the construction of special linearizations of finite traces, which are central to proving our results for CDM games. In the special linearization, all events of the CDM process appear as early as possible. As a result, the response of a sequential strategy along this linearization is based on the least amount of information available to the central decision maker. We then lift these sequential responses to construct a distributed strategy. Crucially, the linearization of the causal past of each CDM event extends that of the preceding CDM events. This property ensures that the distributed strategy can faithfully follow the sequential one on CDM events. In fact, on any trace, the distributed strategy maps the trace to the same global state that the sequential strategy would reach when run on the special linearization of that trace.

We consider the global safety objective, where as expected the system wins if no "bad" global state is ever reached during the play. The other winning condition we explore is local.-parity winning condition A local-parity winning condition is given by a coloring of CDM local state. The system wins if the maximum color that occurs infinitely often along the play is even.

The key results of this section are as follows:

Theorem 7. *Global safety and local parity CDM games are* EXPTIME*-complete.*

Theorem 8. *Consider a CDM game with a global safety or local parity winning condition. Whenever there exists a winning strategy, there also exists a winning distributed memory automaton in which each process maintains, in its local memory, its best knowledge of the global state.*

Upon synchronization, the participating processes update their knowledge of the latest global state. They do this by updating, in their local memory, the stored local state of each process based on which of them most recently synchronized with that process. This information is provided by a *gossip automaton* [9], whose number of states depends only on the number of processes. Consequently, the size of the memory required by each process is constrained to be exponential.

Moreover, there exist instances of global safety CDM games where any memory automaton realizing a winning distributed strategy must have at least an exponential number of local memory states for the central decision maker. Hence, these memory automata are optimal.

ATS games with two decision makers

It is natural to ask about the decidability status of ATS games in the presence of multiple decision making processes. Towards this, we establish the following theorem.

Theorem 9. *ATS games with* two *decision makers are undecidable.*

Our proof of the above theorem uses the key ideas from [5] which showed that six-process asynchronous control games are undecidable. We revisit the ideas in the proof of the undecidability of six-process asynchronous control games and adapt them to the setting of fourteen-process safety ATS games with only two decision making processes.

As soon as the game allows the system to make choices *concurrently* in presence of other deterministic distributed phenomenon, the problem of deciding whether there exists a winning strategy becomes undecidable—even for safety objectives. Importantly, enabling such concurrent choices necessarily requires the game to have two decision-making processes (DMs). Thus, the presence of two DMs is a legitimate source of undecidability: any game that permits two genuinely concurrent choices must inherently involve two DMs.

Equivalence with asynchronous control games

We establish an initial connection between control games and asynchronous transition system (ATS) games by showing that they are equivalent in expressive power. Control games—well-studied and known to be undecidable—can be systematically transformed into equivalent ATS games, and conversely, ATS games can be encoded as control games. This correspondence highlights a close relationship between the two models and will eventually allow insights from one to inform the other.

In control games actions are partitioned into controllable and uncontrollable sets. Each individual process can enable or disable a subset of controllable actions based on its own causal past; these choices are then scheduled, allowing processes to share and update their causal pasts. To construct an equivalent ATS game, we introduce nondeterministic local actions that let each process select a subset of controllable actions. Subsequently, deterministic transitions allow the environment to act as a scheduler, determining the next global state, after which the participating processes exchange their causal pasts.

We next move to the construction of an equivalent control game from given a ATS game. In this translation, environment moves in the ATS game—where the environment plays actions—are modeled as uncontrollable actions in the control game. Controllable actions correspond to joint states for each action in the ATS game, enabling the system to choose its next move. In an ATS game, when the environment plays an action, all processes participating in that action know the entire causal past before deciding on the next state. To capture this in a control game, each environment move (modeled as an uncontrollable action)

deterministically stores this move in the participating processes, effectively synchronizing their causal pasts. Each process can then enable or disable the next combined state of the participating processes, which is subsequently scheduled as a controllable action.

These constructions establish a correspondence between plays in the two game models, which in turn ensures that strategies and winning conditions are preserved.

The key result is summarized in the following theorem:

Theorem 10. *For any state-based winning condition:*

1. *Given an ATS game, we can construct a control game with the same number of processes and a comparable number of actions.*
2. *Given a control game, we can construct an ATS game with the same number of processes and a comparable number of actions.*

such that there exists a winning strategy in the ATS game if and only if there exists a winning strategy in the control game.

The equivalence between control games and ATS games has significant implications. It extends the known undecidability results from control games to ATS games, revealing the inherent complexity of analyzing distributed systems under these models.

At the same time, this equivalence enables results established for control games to carry over to ATS games, and vice versa, thereby unifying the two frameworks. While carrying out such translations may need further investigation it allows insights from control games to directly enrich our understanding of ATS games, and vice versa, leading to a more cohesive view of distributed synthesis.

Conclusion and future directions

In this work, we investigated distributed games in the framework of *asynchronous transition system (ATS) games*, focusing in particular on two special cases that naturally arise in this model: two-process ATS games and games with a *central decision maker (CDM)*. These cases are not only easily formulated within the ATS framework, but also capture practically relevant scenarios in distributed synthesis. Thanks to the equivalence we established between ATS games and control games, results obtained in one model can be systematically lifted to the other, thereby unifying the two frameworks.

For two-process ATS games, we studied objectives such as global safety, local reachability, and global reachability. Our results address two aspects: first, the decidability of whether there exists a winning strategy; and second, the memory structure of such strategies, where we establish both upper and lower bounds. Extending these results to more general objectives—such as parity conditions—offers a natural and interesting direction for future work.

For CDM games—where all non-deterministic choices are controlled by a single process—we established two main results: first, we proved that deciding the existence of winning strategies for global safety and local parity objectives is EXPTIME-complete; and second, we described optimal memory structures for these strategies whenever they exist. Generalizing these results to ω-regular winning conditions is a natural next step. We expect that analogues of the Büchi–Landweber theorem should hold—namely, that whenever a winning strategy exists, there is also a finite one. Establishing this formally will be the subject of future work.

Overall, these contributions answer several fundamental questions and highlight the need for a richer model that can naturally capture important special cases like two-process and CDM games. At the same time, they lay the groundwork for unifying results across different models of distributed synthesis and motivate further investigation into more complex winning conditions and broader classes of distributed games.

List of Publications

Published

- Bharat Adsul and Nehul Jain. *Asynchronous Transition System Games for Two Processes and Their Analysis*. In: Logic and Its Applications — 11th Indian Conference, ICLA 2025, Kolkata, India, February 3–5, 2025, Proceedings. Lecture Notes in Computer Science, vol. 15402, pp. 69–83. Springer, 2025. doi:10.1007/978-3-031-89610-1_5

Under Submission

- Bharat Adsul and Nehul Jain. "Distributed Games with a Central Decision Maker," under submission.

References

[1] Volker Diekert and Grzegorz Rozenberg. *The Book of Traces*. World Scientific Publishing Co., Inc., USA, 1995.

[2] Bernd Finkbeiner and Ernst-Rüdiger Olderog. Petri games: Synthesis of distributed systems with causal memory. *Information and Computation*, 253:181–203, 2017.

[3] Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed games with causal memory are decidable for series-parallel systems. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, 24th International Conference, Chennai, India, December 16-18, 2004, Proceedings*, volume 3328 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 2004.

[4] Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz. Asynchronous games over tree architectures. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 2013.

[5] Hugo Gimbert. Distributed asynchronous games with causal memory are undecidable. *Logical Methods in Computer Science*, Volume 18, Issue 3, Sep 2022.

[6] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[7] P. Madhusudan, P. S. Thiagarajan, and Shaofa Yang. The MSO theory of connectedly communicating processes. In Ramaswamy Ramanujam and Sandeep Sen, editors, *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, volume 3821 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 2005.

[8] Madhavan Mukund. Automata on distributed alphabets. In *Modern Applications of Automata Theory*, IISc Research Monographs Series 2, pages 257–288. World Scientific, 2012.

[9] Madhavan Mukund and Milind A. Sohoni. Keeping track of the latest gossip in a distributed system. *Distributed Computing*, 10(3):137–148, 1997.

[10] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 746–757. IEEE Computer Society, 1990.