Factoring General Games using Propositional Automata

Evan Cox, Eric Schkufza, Ryan Madsen and **Michael R. Genesereth**
Stanford University, CA, USA

Abstract

In this paper we propose the design of a more robust General Game Player, able to successfully play the class of synchronous independent games using Propositional Automata, a framework for reasoning about discrete, dynamic, multiagent systems, developed by the Stanford Logic Group. We prove conditions under which a Propositional Automata represents multiple, synchronous independent subgames, and in doing so, prove when a Propositional Automata can be separated, or factored into multiple automata. A General Game Player able to recognize such independences can computationally save, by searching the smaller game trees represented by the independent factored automata for solutions. Additionally we explore the concept of independent substructures appearing within Propositional Automata given certain states, and prove conditions under which a Propositional Automata is contingently factorable into multiple Propositional Automata.

1 Introduction

General Game Playing (GGP) is the design of artificial intelligence agents that can successfully play games they has never seen before. In contrast to one dimensional game playing systems, such as Deep Blue [1] for chess and Chinook [4] for checkers, General Game Players must be robust enough to make intelligent decisions based on games given at runtime. The challenge of General Game Playing is to build an agent that performs intelligently over a wide variety of games.

While General Game Playing systems have proven successful on large classes of games [5], they have not fared well on games that consist of simultaneous independent subgames for several reasons. This paper is about building a more robust General Game Player, able to successfully play the class of games decomposable into multiple, simultaneous, independent sub-games.

To make intelligent decisions, General Game Players need to reason about the possible future state of a game resulting from their actions. General Game Players tend to rely on two classes of techniques for exploring game trees, variants of the MiniMax algorithm endowed with heuristic evaluation functions [2], and Monte Carlo simulation techniques [3]. Both classes of techniques rely on searching the game tree extensively, Minimax requires an exhaustive search of the game tree, while Monte Carlo takes a more probabilistic, machine learning approach for evaluating states. The problem in applying these techniques to multiple simultaneous independent games is that the size of the game tree increases exponentially with the size of each added game.

Take, for example, double tic tac toe, which is simply two independent games of tic tac toe played simultaneously that only ends if both independent games are in terminal states. The game tree for a single gane of tic tac toe has $n = 255,168$ fringe nodes. Consider the game tree for double tic tac toe. For each terminal state in a single game of tic tac toe, there are n terminal states in double tic tac toe, because there are n possible terminal states the other tic tac toe game could be in. As a consequence the game tree for double tic tac toe is intractably large, containing $n^2 \approx 65$ billion fringe nodes. Given the size of the fringe, any variant of Minimax or Monte Carlo to will have a difficult time performing an effective evaluation of a state with a limited time clock.

If the independence of the sub-games of tic tac toe are identified, and the sub-games are considered separately from each other, then the game tree can be reduced to one of two trees containing $2n = 510,336$ fringe nodes, four orders of magnitude reduction from the two games considered together. This is intuitive because it is only necessary to search the state space for each independent game, in order to find a solution for their pairing. A General Game Player only needs to search the game tree for each independent game of tic tac toe to find the best action for that game. There is no need to consider the other independent game of tic tac toe whose state has no bearing on the decision making process. A General Game Player able to solve single tic tac toe with t amount of play clock per turn, would be able to solve m independent games of tic tac toe if given play clock mt, if it were able to recognize the independence of the games[6].

In general, for m independent games, each with total fringe nodes $f_1 \dots f_m$, the game tree resulting from their pairing will contain $\prod_{i=1}^{m} f_i$ fringe nodes (if that game of their pairing is only terminal when all games are terminal). If the independence of each game is identified, the state space can be reduced to $\sum_{i=1}^{m} f_i$. The game tree for the pairing of two independent games G_1, G_2 with branching factors b_1, b_2 has at depth d, $(b_1 b_2)^d$ nodes. If the independence of G_1 and G_2 are recognized, such a game tree can be reduced to one of two trees, each containing $b_1^d + b_2^d$ fringe nodes.

In this paper we propose the construction of a General Game Player that can successfully play the class of simultaneous independent games, like double tic tac toe, by recognizing independent sub-games using Propositional Automata. Propositional Automata are a framework for discrete, dynamic, multi-agent systems, such as games, invented by the Stanford Logic Group. The class of simultaneous independent games are defined as those games in which each agent takes an action in each independent game on each timestep. The goal for an agent is to win every independent game. Alternatively, we will refer to games in this class as **conjoined**.

First we formally define Propositional Automata. Then we define what it means for a Propositional Automata to be factorable into multiple independent sub-automata, and prove conditions under which such a Propositional Automata is factorable. We show that under appropriate conditions a General Game Player able to detect such conditions will be able to play the class of simultaneous independent games successfully, if it is able to play each game successfully. Finally we consider conditions under which a game evolves into independent simultaneous sub-games over time, and propose an augmentation to a General Game Player that can recognize such games.

2 Propositional Networks and Automata

In this section we introduce and formally define Propositional Networks and Propositional Automata. A Propositional Network (PN) is a directed bipartite graph consisting of nodes representing propositions connected to either boolean gates, or transitions. PNs serve as a natural graph representation of Game Description Language (GDL), which expresses the dynamics of a world in terms of the logical relationships of between different propositions. PNs are useful because they allow us to apply our intuitions about graphs to game representations.

A proposition's truth value is either a function of incoming transition or set by an agent. The truth value of boolean gates are a function of their inputs. Transitions are identity gates, used to control the flow of information from one state to the next. Propositions can be partitioned into three classes: base propositions, whose truth values are a function of incoming transitions; input propositions, whose truth values are set by agents; and view propositions, whose truth values are a function of incoming boolean gates. The state of a PN is the truth assignment of its base propositions. State update is performed by setting base propositions to true if and only if their incoming transition is true in the current state. Because transitions serve as inputs to base propositions, they define the dynamics of a PN [6].

Definition 2.1. *A **Propositional Network** (PN) is a 4 tuple $\langle P, G, T, w\rangle$, where P is a a finite set of propositions $dom(p) = \{0,1\}$, G is a finite set of boolean gates $g : \{0,1\} \times \{0,1\} \mapsto \{0,1\}$ T is a set of transitions, which are identity gates, $t\,\{0,1\} \mapsto \{0,1\}$, and a connectivity function $w : P \mapsto G \cup T \cup \{\emptyset\}; G \mapsto P \times P; T \mapsto P$. Alternatively w can be thought of as a binary relation on the domain $P \cup T \cup G \cup \{P \times P\}$, where pWq iff $w(p) \mapsto q$.*

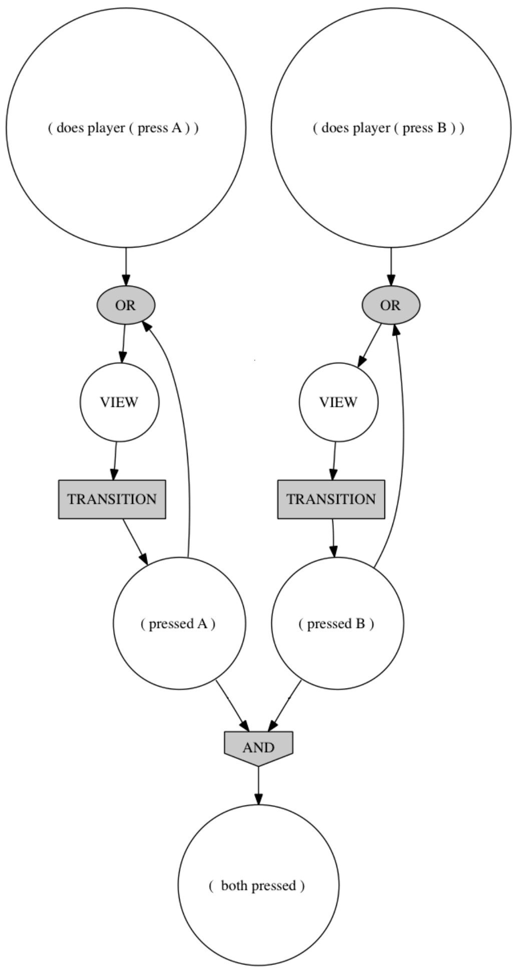

Figure 1: A simple Propositional Network representing the physics of a game in which a player can press two buttons A and B, which once pressed, remain in a pressed state. The state is defined by the base propositions **pressed A** and, **pressed B**. Both pressed propositions are involved in positive feedback loops, ensuring that their truth assignment in the next state will be true if their truth assignment is true in the current one. The input propositions consist of the **press A** and **press B** propositions.

A Propositional Automaton (PA) is a representation for discrete dynamic multi-agent systems that consists of a physics, represented as a PN, an initial state in the form of an truth assignment of the base propositions, and a legality function that restricts what actions an agent in the system can perform given the current state.

Definition 2.2. *A **marking** m is a truth assignment of a set of propositions $Q \subseteq P$. A **Propositional Automata** is a triple $\langle N, M_B^0, l\rangle$ where N is a propositional network, M_B^0 is an intial truth assignment of the base propositions, and l is a* *legality function mapping base truth assignments to finite sets of input truth assignments $l : m_B \mapsto 2^{m_I}$*

For the purposes of GGP, we propose two augmentations to a PA. First we augment our PA representation to include a set of distinguished view propositions, $Goal \subseteq N_V$, that when true, represent the satisfaction of a goal for an agent. Second, in GDL legality is expressed as a series of logical statements. Consequently we reify the legality function into a PN by adding propositions, boolean gates, and a distinguished proposition, $Legal$ such that $m(Legal) \mapsto 1$ if and only if the truth assignment for the input propositions is legal given the current state. In this paper we relax the assumption that agents can only take one action per turn.

Definition 2.3. *A **General Game Playing Propositional Automaton** (GGPA) is a 4-tuple. $\langle N, M_B^0, Legal, Goal\rangle$. N is a PN augmented by the reification of the legality function and goal propositions, M_B^0 is an initial truth assignment of the base propositions, Legal is a distinguished proposition, that when true, represents a legal configuration of actions from the agents for a given state, and Goal is a set of distinguished view propositions that when true represent the satisfaction of a goal for an agent.*

3 Equivalence and Conjunctive Factorability

We first formally define what it means for a General Game Playing Propositional Automaton (GGPA) to be conjunctively factorable, and then prove conditions under which a PA is conjunctively factorable. We begin by constructing an appropriate vocabulary for formalizing the equivalence and conjunction of PA. Using this vocabulary, we give a formal definition of conjunctive factorability and prove conditions under which a PA is conjunctively factorable. A General Game Player that can successfully detect these conditions can therefore detect independent conjoined sub-games, and analyze them independently using a preexisitng evaluation technique.

The conjoining of games is an intuitive concept. The conjoining of $n \geq 2$ independent games is the game where the goal of each player is to win all n subgames. A game is **conjunctively factorable** or decomposable when it is the conjunction of $n \geq 2$ independent sub-games. Truth-functional relationships of propositions in a PN are represented by the connectivity relation. A component p's truth value in a PN, except for input propositions, is determined by the component(s) q from which an edge exists from q to p. Here we can apply a graph intuition to a PN to yield an interesting result. Since edges represent truth functional dependency, if a directed path exists from a proposition q to a proposition p then the truth assignment of p depends on the truth assignment of q.

Let $\mathfrak{W} := W^*$, the transitive closure of the connectivity function w represented as a binary relation W, such that pWq iff $w(p) \mapsto q$

Definition 3.1. *(Directed Path) A **directed path** exists from, $p \in P \cup G \cup T$ to $q \in P \cup G \cup T$ iff $p\mathfrak{W}q$*

Lemma 3.2. *The marking (truth assignment) of a proposition p is a function of a transition or boolean gate q, such that pWq. By induction, the truth assignment of p is a function of all propositions from which a directed path exists to p. Since $\mathfrak{W}$ is the transitive closure of W, $p\mathfrak{W}r$ iff there exists a directed path from p to r. Therefore, the marking of a proposition p is a truth function of only the propositions from which a directed path exists to p.*

Using the notion of a directed path we can formalize what it means for propositions to be truth-functionally independent of each other. If there is not a directed path from a proposition p to a proposition q then the truth assignment of q in any state is irrelevant to the truth assignment of p.

Corollary 3.3. *If there is no directed path from a proposition p to a proposition q then then the truth value of q is not a function of p.*

Extending this notion to sets, we can formalize what it means for a set of propositions to be truth-functionally independent of another set of propositions. If two sets of propositions are independent of each other, then they can be reasoned about independently of one another.

Definition 3.4. *$A \subseteq (P \cup G \cup T), B \subseteq (P \cup G \cup T)$ are **disconnected** iff there is no directed path from any proposition in A to any proposition in B. $C \subseteq (P \cup G \cup T)$, is **independent** iff $\neg(\exists a \exists b(a \in C \wedge b \notin C \wedge b\mathfrak{W}a)$ An independent set , D is **action independent** if $D \cap N_I = \emptyset$.*

The truth values of sets that are disconnected from one another can be reasoned about independently of one another in the current state of a PA. However, this might not be true for reasoning continuously about the disconnected sets, as other propositions outside of the two sets may be affect their truth values. Independent sets (ISs) are connected sub-graphs in a PN, where no outside components serve as inputs to any members of the IS. The truth values of action independent sets (AIS) are functionally independent of the truth assignments of input propositions, whose analogue are actions of agents. Their truth assignments are directly computable given a number of base marking updates and the initial base marking.

Since no edges exist from the rest of a PN to an IS, the markings of ISs of a PN can be fully simulated independently of the remaining network. The lack of connection to a PN means that the truth values of the propositions in an IS can be determined regardless of the current or past states of the rest of a PN. Because an IS represents an independent physics within the larger network, an IS represents its own independent PN.

Theorem 3.5. *An IS of a PN, C, along with connectivity function $w_C = w$ with domain restricted to C, of a PN, N can be represented as an isomorphic PN.*

Proof. Construct a new PN, N' in the following way. For every p in $N_P \cap C$ insert a proposition p' into N'_P For every g in $N_G \cap C$ insert a proposition g' into N'_G. For every proposition $t \in N_T \cap C$ insert a proposition t' into N'_T. For every mapping in w_C insert a mapping into w'. Since $\neg(\exists a \exists b(a \in C' \wedge b \in C \wedge w(b) = a)$, there will be no undefined mappings in w'. So for every p in $N_P \cap C$ there is a p' such that if $w(p) \mapsto q$, $w(p') \mapsto q'$. Hence, it follows that N'_P is isomorphic to $N_P \cap C$. Similarly $N_G \cap C$ is isomorphic to N'_G, and $N_T \cap C$ is isomorphic to N'_T, and w_C is isomorphic to w' by construction. Since there is an isomorphism between every part of C and N', it follows that there is an isomorphism between C and N'. □

Detecting classes of games that represent independent simultaneous sub-games hinges upon this idea. A necessary condition for conjunctive factorability is the representation of multiple independent physics within the network itself. In order to represent independent sub-games, there need to be separate physics for each of those games. However, it is not a sufficient condition. The Legal proposition itself may cause the separate physics to become joined in determining what actions are legal for a given physics represented as a PN. Consider the conjoining of chess and checkers where a player is only allowed to move their queen after they have captured two pieces. While the physics of the two games are separable, the state of checkers affects the legality of chess, and cannot be considered separately.

We now define equivalence between PNs and PAs. Following that we provide a definition for equivalence between a single PA and multiple conjoined PA. The notion of equivalence between PNs is straightforward. If there exists a mapping between states of two distinct games that respects the next state operation, them the two games are equivalent. Accordingly there must be a mapping between the set of facts that define a state or N_B, that respects next state or base marking update operation in order for two PNs to be considered equivalent. We extend the notion of homomorphism to PNs in order to capture the intended requirements for equivalence.

Definition 3.6. *A PN homomorphism is a function mapping between two PNs N, M that respects base marking update operation resulting from a base and input marking. $h : N \rightarrow M$, such that for all $i \in 2^{m_I}$, $b \in 2^{m_B}$, and an operator, $\otimes$, denotes the updated base marking computed from a current base marking and input marking , $h(i \otimes b) = h(i) \otimes h(b)$.*

Definition 3.7. *PNs N_1, N_2 are **equivalent** if there is exists a homomorphism between N_1 and N_2. A PN N is equivalent to n independent PNs, $Net_1, Net_2 \ldots Net_n$ if $\bigcup_{i=1}^{n} Net_i$ is equivalent to N. We denote N_1, N_2 as being equivalent by $N_1 \equiv N_2$.*

For two games to be considered equivalent they must not only share the same physics, but must also have the equivalent rules for when actions can be performed, and equivalent goal states. The equivalence between PA captures the important aspects of equivalence between games; games must have the functionally same goals, physics, and restrictive legality in order to be considered equivalent.

Definition 3.8. *A homomorphism between PA is one that respects the base marking update operation $\oplus$, the Legal marking operation $\ominus$, and goal marking operation $\otimes$. For all $i \in 2^{m_I}$, $b \in 2^{m_B}$,*
$h(i \oplus b) = h(i) \oplus h(b)$
$h(i \ominus b) = h(i) \ominus h(b)$
$h(i \otimes b) = h(i) \otimes h(b)$

Definition 3.9. *PA A_1, A_2 are **equivalent** if there is a homomorphism between A_1 and A_2, and $h(M^0_{B_1}) = h(M^0_{B_2})$.*

We now define the conjoining of games, represented as PA. For brevity we only consider the case in which there is exactly one goal proposition, and one agent. The ideas are extendable to multiple goals and multiple agents. Using the formal definitions of conjoined games and equivalence, we proceed to show when a game, represented as a PA is conjunctively factorable.

Definition 3.10. *The conjunction of $n \geq 2$ PA, $A_1 \ldots A_n$ is defined as $\langle \bigcup_{i=1}^{n} N_i, \bigcup_{j=1}^{n} M_{B^0_j}, \bigwedge_{k=1}^{n} Legal_k, \bigwedge_{l=1}^{n} Goal_l \rangle$.*

The conjoining of PA is straightforward, the PN of the conjoined games is the union of the physics of these sub-games, the $Legal$ proposition is the conjunction of all the $Legal$ propositions of each sub-game, $Goal$ is satisfied iff the goal for each sub-game is satisfied, and the initial state is the union of each sub-game's initial state.

Theorem 3.11. *A PN N that consists of $n \geq 2$ independent sets $N_1, N_2 \ldots N_n$ is equivalent to the union of n independent PNs.*

Proof. For each independent set N_i, construct a new PN N'_i. By construction each N_i is isomorphic to N'_i (By Previous Lemma). Therefore since $\cup_{j=1}^{n} N'_j \equiv \cup_{k=1}^{n} N_i$, and $\cup_{k=1}^{n} N_i \equiv N$, it follows that $\cup_{j=1}^{n} N'_j \equiv N$. □

Lemma 3.12. ***IID** is the largest action independent, set of a PN N. The PN equivalent to IID is equivalent to the union of $n >= 1$ PNs each isomorphic to IID.*

Proof. Let N be a PN such that $N = \cup_{i=1}^{n} IID_i$ where each $IID_i \equiv IID$. For each proposition $p' \in IID_i$ there is an isomorphic proposition $p \in IID$. So for every proposition p, there is a homomorphism between the singelton set $\{p\}$, and $\{p' | p' \equiv p \wedge p' \in N\}$ Hence, there is a homomorphism between $m(p)$ and $m(P')$, where P' is the set of all p's.

Similarly the singelton set containing a boolean gate $g \in IID$ is homomorphic to the set of isomorphic boolean gates in N, and the singelton set containing a transition in $t \in IID$ is homomorphic to the set of isomorphic transitions N. Therefore there is an homomorphism between IID and N, namely the one that maps each element in IID to the set of isomorphically similar elements in each IID_i. Since there $N = \cup_{i=1}^{n} IID_i$, and there is a homomorphism between $\cup_{i=1}^{n} IID_i$ and IID, $N \equiv IID$ □

Action independent propositions are unique in that their marking is unaffected by agents' actions. Consequently, an action independent base proposition will be isomorphic to any other action independent base proposition whose truth value is a function of a homomorphic structure, contingent upon the same initial truth assignment. IID simply represents the union of all action independent components of a PN.

Definition 3.13. *A Legal proposition of a PA A with $n > 1$ ISs $N_1 \ldots N_n$ is **legally partitionable**, if $m(Legal) \mapsto \bigwedge_{i=1}^{n} w(w(Legal))_i$ and for each*

$w(w(Legal))_1 \ldots w(w(Legal))_n$, there only exist directed paths from $w(w(Legal))_i$ to at most one $k < n$ N_k and IID.

If the legality of actions from some independent subnetwork of a PN are dependent only on the state of that network, or action-independent propositions then $Legal$ is legally partitonable. This is one of the conditions for conjunctive factorability. If the legality of actions for some sub-game is dependent in some way on the state of another sub-game, then they cannot be considered independently of one another. Synchronous indepedent games are legally partitionable, because the legality of moves for each independent game depends only on the state of that game, with possibly some shared, action independent state, such as a control counter.

Definition 3.14. *For a Propositional Automaton A, Goal = A_{Goal}, with $N = (\bigcup_{i=1}^{n} N_i) \cup IID$, where each N_i is an IS, is* ***goal partitionable*** *iff for Goal $= \bigwedge_{j=0}^{n} w(w(Goal))_j \in N_j$.*

The property of Goal partitionability is analogous to Legal Partitionability. The goal of an individual agent is partitionable if it can be represented as the conjunction of subgoals, and the truth value of each subgoals is determined by an independent, distinct physics. A conjoined game, where an agent's goal is to satisfy the goal of every subgame, will be goal partitionable.

Using this vocabulary we can now formalize what it means for a PA to be conjunctively factorable.

Definition 3.15. *A Propositional Automata $A\langle N, M_B^0, Legal, Goal\rangle$ is* ***conjunctively factorable*** *if there exists $n > 1$ PA $A_1\langle N_1, M_{B_1}^0, Legal_1, Goal_1\rangle$, $A_2\langle N_2, M_{B_2}^0, Legal_2, Goal_2\rangle \ldots$ $A_n\langle N_n, M_{B_n}^0, Legal_n, Goal_n\rangle$ such that $(\bigwedge_{i=1}^{n} A_i) \equiv A$.*

Theorem 3.16. *A Propositional Automaton A that contains $n > 1$ ISs, that is legally partitionable and goal partitionable is conjunctively factorable into n Propositional Automata.*

Proof. Construct n PA, $A_1, A_2, \ldots A_N$ in the following way. Let IID be the union of all AIS. Enumerate each IS of the PN $A_N = N$, as $N_1 \ldots N_n$, $N = (\bigcup_{i=1}^{n} N_i) \cup IID$. For each N_i construct a PN $N'_1 = N_1 \cup IID, N'_2 = N_2 \cup IID \ldots N'_n = N_n \cup IID$. Let N_i be the PN of A_i. Let $M_{B_i}^0$, the function M_B^0 with domain restricted to B'_i (the set of base propositions for N'_i) be the base marking for A_i. Since $Legal$ is conjunctively partitionable, it follows that for each N'_i, there is a $w(w(Legal))_i$ to which directed paths only exist to $N'_i \cup IID$. Let each $w(w(Legal))_i$ be the $Legal$ proposition for A_i and any additional components from which a directed path exists from $w(w(Legal))_i$ be in N'_i. Since $Goal$ is conjunctively partitionable, let $w(w(Goal))_i$ be $Goal$ for A_i.

We will show that there is a homomorphism between $\bigwedge_{j=1}^{n} A_j$ and A. We now show that the union of the PNs of $A_1, A_2, \ldots A_n$ respect the base marking update operation. $\bigcup_{i=1}^{n} N'_i = \bigcup_{j=1}^{n} N_j \cup \bigcup_{k=1}^{n} IID_k$. There is a homomorphism between $\bigcup_{k=1}^{n} IID_k$ and IID. So there is a homomorphism between $\bigcup_{i=1}^{n} N'_i$ and $(\bigcup_{j=1}^{n} N_j) \cup IID$. Since $(\bigcup_{j=1}^{n} N_j) \cup IID) = N$ it follows that there is a homomorphism between $\bigcup_{i=1}^{n} N'_i$, which is the PN of $\bigwedge_{l=1}^{n} A_l$.

We showed that the union of the base markings of $A_1, A_2, \ldots A_n$ are equivalent to the base marking of A. Since $\cup_{i=0}^{n} N'_i \equiv N$, it follows that $M_B^0 \equiv \bigcup_{j=0}^{n} \equiv M_{B'_j}^0$ by construction.

We now show that $m(Legal) = \bigwedge_{i=1}^{n} Legal_i$. The marking of $Legal_i$ of A_i is isomorphic to the marking of $w(w(Legal))_i$ by construction. Since $m(\bigwedge_{i=1}^{n} Legal_i) = m(\bigwedge_{j=1}^{n} w(w(Legal))_j$, and $m(Legal) \mapsto \bigwedge_{k=1}^{n} w(w(Legal))_j$, it follows that there is a homomorphism between $m(Legal)$ and $\bigwedge_{j=1}^{n} Legal_j$.

We now show that $m(Goal) = \bigwedge_{i=1}^{n} Goal_i$. The marking of $Goal_i$ of A_i is homomorphic to the marking of $w(w(Goal))_i$ by construction. Since $m(\bigwedge_{i=1}^{n} Legal_i) = m(\bigwedge_{j=1}^{n} w(w(Goal))_j)$, and $m(Goal) \mapsto \bigwedge_{k=1}^{n} w(w(Goal))_j$, it follows that $m(Goal) = \bigwedge_{j=1}^{n} Goal_j$. Hence by construction, A is equivalent to $\bigwedge_{i=1}^{n} A_i$ and is conjunctively factorable. ☐

The result makes sense. Consider a game where a goal is defined as a conjunction of propositions, each of which is exclusively dependent on distinct independent physics. Furthermore suppose that the legality of a game is defined as the conjunction of propositions that are dependent on these distinct physics. Since both the goal and legal propositions are decomposable in terms of the independent physics, the game itself decomposable in terms of the independent physics. Consequently, goal partitionability, representing goal decomposability and legal partitionability, representing legal decomposability, along with $n > 1$ IS, representing independent physics are requirements a General Game Player can test in order to determine decomposability a game.

When independent PA are identified, a General Game Player can construct game trees for the independent subgames using the independent PAs, rather than the game tree represented by the original PA. The sum of the state space of the independent game trees is nontrivially smaller than the state space represented by the original game tree, allowing whatever evaluation function the General Game Player implements to be more successful, by providing it with an equivalent, but significantly smaller state space to evaluate.

4 Contingent Conjunctive Factorability

We now consider the class of games, that over time may become conjunctively factorable. For example, a game might not initially be separable into independent games, but it may, after entering a certain state, become representative of multiple independent games. Consider the following game, called joined tic tac toe. Two games of tic tac toe connected by a single square that connects the two. The goal of the game for a player is to get two lines, a row, column or diagonal of that players mark, with at least two of the marks residing in a specific tic tac toe domain. Diagonals through the middle square do not count. Each turn the player in control can place two marks, either one in each distinct tic tac toe domain, or one mark in a tic tac toe domain, and one in the center square. Suppose that the state of the game is as follows

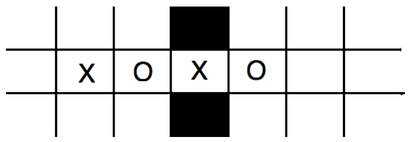

Figure 2: Joined Tic Tac Toe

Once it is not possible for either player to achieve a row utilizing the center square, the only possible solutions lie in the domains of the tic tac toe games that are joined by the center square. The states of the two tic tac toe games can be considered independently to find the remaining optimal moves for the duration of the game. Only the game trees for each tic tac toe game, modulo the center square, need to be searched to determine the remaining optimal moves. Given the current state, the game can be factored into two independent sub-games. However, from the initial state, this game cannot be factored into independent games, because the shared middle square intertwines the two domains as it is relevant to the satisfaction of goals in both sub-games.

In general while this game is not conjunctively factorable, it is conjunctively factorable, contingent upon entering a state in which no row through the middle is possible for either player. We define the game as being **contingently conjunctively factorable**, since it reduces to independent simultaneous sub-games, given that it enters a certain state. The raw computational savings acquired from recognizing contingent conjunctive factorability are less than those of recognizing conjunctive factorability, because it requires that the game enter into a specific state. However, the relative computational savings are still the same, because the number of accessible fringe nodes reduces to the sum of the remaining accessible fringe nodes of the individual games, rather than the product.

Definition 4.1. *Two PA, A_1, A_2 are **contingently equivalent**, if there is a homomorphism between A_1 and A_2 given any base marking M_B^n or subsequent updated base marking.*

Given a specific state, two PA are contingently equivalent if for any subsequent state of either PA the base marking, legal marking, and goal marking operations are all respected. Consider two PA representing different state of tic tac toe, where the next action will result in a draw. These PA will be contingently equivalent. Contingent equivalency depends on what facts currently hold in a game state. Using these intuition, we now show how the graphical nature of a PA can be leveraged to discover contingently equivalent PA.

Lemma 4.2. *A proposition p is a **latch** if $m(p) = 1 \rightarrow m'(p) = 1$. If a proposition is a latch then $m(p)^k \mapsto 1 \rightarrow m(p)^{k+1} \mapsto 1$, where $m(p)^k$ represents the marking of p after k base marking updates with legal input markings.*

Proof. Consider a propositional latch p, $m(p)^k \mapsto 1$, where $m(p)^k$ is the marking of p after k base marking updates. We prove by induction that for $n \geq 1$, $m(p)^{k+n} \mapsto 1$. Basis: $m(p)^{k+1} \mapsto 1$. By assumption $m(p)^k \mapsto 1$. Since p is a positive latch, by Lemma X, $m(p)^{k+1} \mapsto 1$. Induction: IH, $m(p)^{k+n-1} \mapsto 1$. Show that $m(p)^{k+n} \mapsto 1$. Since, by assumption p is a positive latch, and $m(p)^{k+n-1} \mapsto 1$ by the Induction Hypothesis, by Lemma X, $m(p)^{k+n} \mapsto 1$.
By induction we have proved that $m(p)^k \mapsto 1 \rightarrow m(p)^{k+n} \mapsto 1$. □

Lemma 4.3. *A base proposition p, with transition t, such that $w(p) = t$ is a latch if $w(t) = p$.*

Proof. Suppose that $m(p)^k \mapsto 1$. Since transitions are identity gates $m(t)^k \mapsto 1$. Since the truth value of t in the current state represents the truth value of p in the next state $m(p)^{k+1} \mapsto 1$. By the previous lemma, it follows that p is a latch. □

Simply put, a latch is a proposition, that once it becomes true, is always true. Latches are useful discoveries in PA, because they allow the direct encoding of information. For example, if a boolean gate g, $m(g) = m(p) \vee m(q)$, and p is a latch and p is true, for any following state, $m(g) \mapsto 1$. The connection between p, q, and the g can be removed and the g can be replaced as mapping to 1. Since this sort of modification only consists of taking advantage of a logical fact, it follows that the base marking update for a propositional automaton modified in this way, will be equivalent to the unmodified version of the automaton. We present the following formal modification of a PA, and prove that it is contingently equivalent to its unmodified form.

Definition 4.4. *For a PN N, with marking M_B^k, let $p \in B$ be a latch. Modify N in the following way. Let X be the set of boolean gates such that for $g \in X$, $w(g)_1 = p$ or $w(g)_2 = p$. For every $a \in P$ insert a' into P'. For every $t \in T$, insert t' into T'. For every $g \in X^c$, insert g' into G'. For every $g \in X$, construct g' in the following way. Let $q = w(g)_2$.*
$m(g) \mapsto m(p) \vee m(q)$ $m(g') \mapsto 1$. Remove the edge from q' and p' to g'.
$m(g) \mapsto m(p) \wedge m(q)$ $g' \mapsto m(q')$. Remove the edge from p' and g'.
$m(g) \mapsto \neg m(p)$ $m(g') \mapsto 0$.
If for $v \in V$, $w(v) = g$, $w(v') = g'$.

Latch modfication represents the encoding of fixed information into a PN. For example, once an input to an OR gate will remain true, the marking of the OR gate will always be true as well, because its truth value is a function of the disjunction of its inputs. There are more latch modifications than the one we provide here. However, we will only consider positive latch modification. We prove that this form of latch modification produces contingently equivalent PN and take it to be representative of the class of latch modifications.

Lemma 4.5. *A is contingently equivalent to A' on M_B^k, if N' is latch modified with respect to p, and $m(p)^k \mapsto 1$.*

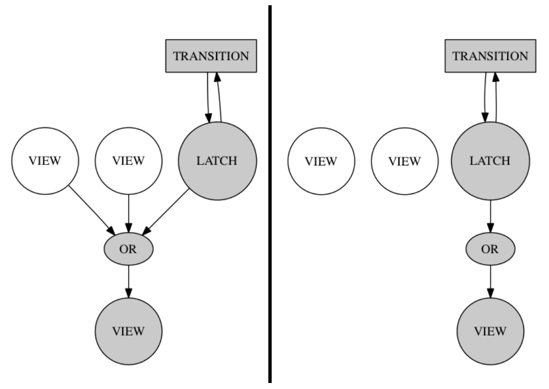

Figure 3: A fragment of a Propositional Network modified by Latch Modification. Components of the PN that are true are represented by as grey. Since transitions are identity gates, the Proposition labeled Latch is a latch. The right hand side of the figure shows the fragment before latch modification, and the contingently equivalent fragment after latch modification. Note the broken connection between the other view propositions and the OR gate

Proof. Consider a base marking M_B^k such that $m(p)^k \mapsto 1$. It follows that for $n \geq 1$, $m(p)^{k+n} \mapsto 1$. We will prove that A and A' are contingently equivalent upon $M_B^0 = M_B^k$. Since, N and N' only differ on modified boolean gates in G ,G' we will show that given that $m(p)^k \mapsto 1$, that there is an isomorphism between the marking of every $g \in G$ and the modified $g' \in G'$, and consequently show that A and A' are contigently equivalent on M_B^k. Proof by cases.

(OR case) If g is of the form $m(g)^k \mapsto m(p)^k \vee m(q)^k$, then $m(g)^k \mapsto 1$, since $m(p)^k \mapsto 1$. Since $m(p)^{k+n} \mapsto 1$, it follows that $m(g)^{k+n} \mapsto 1$. Since, by construction, $m(g') \mapsto m(p')^k$, it follows that $m(g')^k \mapsto 1$, and $m(g')^{k+n} \mapsto 1$. Thus the markings of g and g' are isomorphic, given M_B^k.

(AND case) If g is of the form $g^k \mapsto m(p)^k \wedge m(q)$, then $g^k \mapsto m(q)^k$, since $m(p)^k \mapsto 1$. Since $m(p)^{k+n} \mapsto 1$, it follows that $m(g)^{k+n} \mapsto m(q)^{k+n}$. Since, by construction, $m(g') \mapsto m(q')$, it follows that $m(g')^k \mapsto m(q')^k$, and $m(g')^{k+n} \mapsto m(q')^{k+n}$. Thus the markings of g and g' are isomorphic, given M_B^k.

(NEGATION case) If $m(g)$ is of the form $m(g)^k \mapsto \neg m(p)^k$. By construction g' is not modified in this case.

Thus markings of g and g' are isomorphic, given M_B^k.

Therefore the marking of each g is isomorphic to the marking of g', given $m(p)^k \mapsto 1$. Since this is the only difference between the components, of N, and N', it follows that N, and N' are contingently equivalent given M_B^k. □

Definition 4.6. *A PA A is* ***contingently conjunctively factorable*** *if A' is contingently equivalent to A given M_B^k and there exists $n > 1$ PA $A'_1 \dots A'_n$ such that there is a homomorphism between $\bigwedge_{i=1}^n A'_i$ and A'.*

Theorem 4.7. *A Propositional Automaton A is contingently conjunctively factorable if A', latch modified with respect to latches $p_1 \dots p_m$ is legally partitionable, goal partitionable, and consists of $n > 1$ independent sets.*

Proof. Since A' is assumed to be legally partitionable, goal partitionable, and consists of $n > 1$ independent sets, it follows that there exists n PA $A'_1 \dots A'_n$ such that there is a homomorphism between $\bigwedge_{i=1}^n A'_i$ and A' 3.16. Since A' is contingently equivalent to A given M_B^k such that $m(p_1) \mapsto 1 \dots m(p_m) \mapsto 1$, and $\bigwedge_{i=1}^n A'_i$ is equivalent to A' it follows that A is contingently conjunctively factorable given M_B^k. □

Reformulation of a PA via latch modification can result in the discovery of contingently conjunctive factorablility. Thus, if a game enters into a state where it is contingently conjunctively factorable, the remaining state space search for a General Game Player is further reduced, allowing for more intelligent decision making, when constrained by a time limit.

5 Conclusion

The class of games consisting of multiple, simultaneous, independent sub-games represent a challenge General Game Players because of the multiplicative nature of the state space growth with each independent sub-game. Discovery of the independence of sub-games, however, reduces the game state space to the sum of each independent game, rather than their product. PA are useful because they make discovery of independent, simultaneous sub-games straightforward, and even make for convenient discovery of independent sub-games over time. We recommend further exploration of the properties of PA, as they reveal interesting, computationally advantageous structure of games in a straightforward manner.

References

[1] M. Campbell, A.J. Hoane, and F. Hsu. Deep blue. *Artificial Intelligence*, 134(1-2):57–84, 2002.

[2] J. Clune. Heuristic evaluation functions for general game playing. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1134. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[3] Hilmar Finnsson and Yngvi Björnsson. Simulation-based approach to general game playing. In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 259–264. AAAI Press, 2008.

[4] J. Schaeffer, R. Lake, P. Lu, and M. Bryant. Chinook: The world man-machine checkers champion. *AI Magazine*, 17(1):21–29, 1996.

[5] S. Schiffel and M. Thielscher. Fluxplayer: A successful general game player. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, page 1191. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[6] Eric Schkufza. Decomposition of games for efficient reasoning. In Ian Miguel and Wheeler Ruml, editors, *SARA*, volume 4612 of *Lecture Notes in Computer Science*, pages 409–410. Springer, 2007.