Admissible Strategies for Safety and Reachability Objectives in Graph Games

Prasita Mukherjee
(174101042)

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Supervisor

Dr. Purandar Bhaduri

In partial fulfillment of the requirements for the degree of

Master of Technology, Computer Science and Engineering

May 16, 2019

Acknowledgements

I express my sincere gratitude towards my guide **Dr. Purandar Bhaduri** for his constant support, encouragement and inspiration throughout the project work. I specially acknowledge him for his advice, supervision, and the vital contribution as and when required. It is because of his constant and general interest and assistance that this project has been successful.

I want to acknowledge my parents for motivating me and providing moral support during the course time of M.Tech. I also want to acknowledge the Department Of Computer Science and inter department teaching members for helping me by giving the basic idea.

Certificate

This is to certify that the project work entitled **Admissible Strategies for Safety and Reachability Objectives in Graph Games** being submitted to Department of Computer Science and Engineering, Indian Institute of Technology Guwahati by **Prasita Mukherjee**, in partial fulfillment for the award of the degree of Master of Technology, Computer Science and Engineering in, is a bonafide work carried out by her under my supervision. To the best of my knowledge it has not been submitted elsewhere for award of degree.

.....

Dr. Purandar Bhaduri

Professor

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Abstract

Finding winning strategies for *Player 1* (the system) against *Player 2* (the environment) in graph games is the basis for synthesizing controllers satisfying a specification. In many situations where winning strategies do not exist, it is important to compute the best-effort (or admissible) strategies. The thesis aims to compute admissible strategies for safety and reachability objectives in graph games.

Contents

List of Figures

Chapter 1

Introduction

Games played on finite graphs, typically involving two players, have been a subject of widespread investigation in the field of computer science, with applications primarily in the field of controller synthesis [1]. Typically, a winning strategy for *Player 1* (the system) against *Player 2* (the environment) gives rise to a controller that satisfies a given specification.

The games considered have a set of winning and losing nodes for *Player 1*. In this work, the primary focus is on losing nodes and attempt to achieve *best-effort strategies*, also known as *admissible strategies* [2]. In many applications finding such a best-effort strategy when a winning strategy does not exist from a given node is very useful. Instances of such applications are problems which do not involve a strong sense of competition, *i.e.*, *Player 2* can be cooperative rather than adversarial or more realistically when *Player 2* is rational, *i.e.*, focused more on achieving its own objective rather than thwarting *Player 1* from meeting its goal. This happens, for instance, in component-based systems where the components are autonomous agents such as robots which have their own objectives to meet and are not necessarily adversarial. In such cases, *Player 1* can end up winning if this does not prevent *Player 2* from meeting its goal, even if there is no strategy

that will win against all *Player 2* strategies.

Safety and *reachability* are two very important goals in system design. Safety requires plays to remain forever within a safe set of nodes, whereas reachability requires a set of nodes to be visited after finitely many steps. The work computes admissible strategies for these two types of objectives.

Admissible strategies are also known as *non-dominated strategies i.e.*, the game can be lost by such strategies if and only if no other strategy can win the game starting from the same node against any strategy of *Player 2*. In this work, the perspective of *Player 1* is taken into consideration. The semantics are defined analogously when played from *Player 2's* perspective and the algorithms hold good. The notion of *admissibility* is slightly different from Faella [3] and is consistent with the definition of Brenguier *et al.* [2].

The rest of the thesis is organized as follows. First the related work done in this area is discussed in Chapter 2, followed by the preliminaries and definitions in 3. Then the two algorithms developed for safety and reachability objectives are discussed in Chapters 4 and 5 respectively, along with their time complexity.

Chapter 2

Related Work

Determining the winning set of nodes and the winning strategies in a graph game had been a primary concern in the past, and significant work has been done for safety, reachability and more general ω-regular objectives in this area [4]. Work on cooperation and rational behavior of players is now an emergent area of research [3; 5; 6; 7; 8].

Faella [3] discussed the potential benefits of real world applications in playing a game from a *losing* node in a rational way to achieve the best possible results. His primary focus was on *best-effort strategies* by relying on the cooperation of the other player. He outlined a procedure to compute admissible strategies for *positional* and *prefix-independent* goals, where the latter are goals closed under adding or deleting finite prefixes to/from its elements. But he did not provide a method for goals which are not prefix-independent, such as safety and reachability. He claimed that if all *Player 2* nodes are treated as *Player 1* nodes, then the winning strategies obtained, called *cooperatively winning*, are also admissible for safety and reachability objectives. It is shown via examples that this claim is incorrect and algorithms are provided for finding admissible strategies for safety and reachability objectives, arguably the most important among all goals which

are non-prefix-independent.

Bloem *et al.* [5] discussed various approaches for dealing with assumptions on environment behavior while solving games. They proposed four goals which should be met by all system designs, out of which what is relevant here is the goal of "not giving up" when the system guarantee cannot be enforced under worst-case assumptions. As mentioned above, in many situations, the environment may not be perfectly adversarial and provide the worst-case input. In such cases, it makes sense for the system to try to meet its goal as much as possible, assuming some level of cooperation from the environment. The work addresses precisely this situation, finding admissible strategies for the important cases of safety and reachability objectives. Note that, by definition, admissible strategies make the minimal assumptions about cooperation from the other player.

Damm and Finkbeiner [6; 7] proposed algorithms for synthesizing and verifying dominant strategies which is based on construction of tree automata for objectives specified in LTL. The notion of dominant strategy considered in the work is identical to what is called an admissible strategy in this work. The work is a special case of [6; 7], which considers safety and reachability objectives. The algorithms developed are also much simpler, based primarily on graph-theoretic notions and of lower complexity than the double exponential algorithm in [6; 7].

In [8], Fisman *et al.* introduce the idea of rational synthesis, *i.e.*, synthesis in the context of multiple autonomous agents where each agent has a goal of its own and the goals are not necessarily adversarial. The approach provides a strategy not just for the agent in question (called the *system agent* in [8]) but for all the agents such that the specification of the system agent is satisfied and the strategy profile for all the agents meet some desired solution concept, such as the existence of a dominant strategy, Nash equilibrium, or subgame-perfect equilibrium. In such equilibria, no agent has the incentive to unilaterally deviate

from its strategy. While admissible strategies are clearly related to the problem of rational synthesis, exploring the exact relationship is part of future work.

Chapter 3

Preliminaries

A *game structure* G is defined as a pair (V, E) where

1. $V = V_1 \uplus V_2$ represents the finite set of vertices/nodes in the game graph, where V_1 and V_2 are the sets of *Player 1* and *Player 2* nodes respectively.

2. $E \subseteq V \times V$ represents the finite set of edges in the game graph. Every node in V has a successor, *i.e.*, $\forall v \in V, \exists v' \in V.(v, v') \in E$.

A *play* in G is defined as a sequence of nodes $v_0 v_1 v_2 \ldots$ such that $(v_i, v_{i+1}) \in E$. If the current node $v \in V_1$, then *Player 1* makes the next move by selecting an edge $(v, v') \in E$ that takes the game to v'. If the current node $v \in V_2$, a *Player 2* move is defined analogously. A *game* $\mathcal{G} = (G, \mathit{Win})$ is a pair of a game structure and a winning condition *Win*, given by a set of plays in G.

A game can have various winning conditions (like *Büchi*, *Muller*, *Parity*, etc.) that specify the set *Win* of plays that are won by *Player 1*. In the work, only safety and reachability objectives are considered. For a game structure $G = (V, E)$, a *safety objective* is given by a set $S \subseteq V$ of nodes. A play $v_0 v_1 v_2 \ldots$ in V^ω is safe, *i.e.*, winning for *Player 1*, if it never leaves the set S, *i.e.*, $v_i \in S$ for all i$\in \mathbb{N}$. *Player 1* loses the game if the play enters V\S at any point. A *reachability*

objective is specified by a set $R \subseteq V$ of nodes. The goal for *Player 1* is to force the game to reach a node in R, *i.e.*, a play $v_0v_1v_2\ldots$ is winning for *Player 1* if $v_i \in R$ for some i$\in \mathbb{N}$.

A *strategy* for *Player 1* is defined by a function $\sigma : V^*V_1 \to V$ that maps the history of observed nodes to the next node. A *Player 2* strategy is defined analogously. Given strategies σ and ρ for *Player 1* and *Player 2* respectively, the work defines $Out(\sigma, \rho, v)$ as the unique word $v_0v_1v_2..$ in V^ω such that $v_{i+1} = \sigma(v_i)$ if $v_i \in V_1$ and $v_{i+1} = \rho(v_i)$ otherwise, for all $i \in \mathbb{N}$ and $v_0 = v$. The work defines $Out(\sigma, v) = \{Out(\sigma, \rho, v) \mid \rho$ is a *Player 2* strategy$\}$ as the set of outcomes from node v when *Player 1* plays according to strategy σ. A strategy σ for *Player 1* from node v is *winning* if $Out(\sigma, v) \subseteq Win$, where *Win* is a set of winning plays, *i.e.*, σ is a winning strategy for *Player 1* if for all *Player 2* strategies ρ, $Out(\sigma, \rho, v) \in Win$. The work defines $val(\sigma, \rho, v)$ as 1 if $Out(\sigma, \rho, v) \in Win$ and 0 otherwise. For a game $\mathcal{G}$, a strategy for σ for *Player 1* is said to be *positional* (or *memoryless*) if σ depends only on the last node of the current history of the game *i.e.*, there is a function $f : V_1 \to V$ such that $\sigma(sv_1) = f(v_1)$ for all $s \in V^*$. For the winning conditions of safety and reachability, positional strategies suffice and hence attention is restricted to such strategies in the rest of the thesis.

The following definitions appear in [2]. For any two *Player 1* strategies σ_1 and σ_2, σ_1 *very weakly dominates* σ_2 at node v, if $val(\sigma_1, \rho, v) \geq val(\sigma_2, \rho, v)$. A *Player 1* strategy σ_1 is *dominated* by another *Player 1* strategy σ_2 at node v, if $val(\sigma_1, \rho, v) \leq val(\sigma_2, \rho, v)$, for all *Player 2* strategies ρ and $val(\sigma_1, \rho, v) < val(\sigma_2, \rho, v)$ for at least one *Player 2* strategy ρ. It is also said σ_2 *dominates* σ_1 at v in such a case. An *admissible strategy* for *Player 1* is a strategy σ_1 that is not dominated by any other *Player 1* strategy σ_2 at any vertex v.

Figures 3.1 and 3.2 depict examples of admissible strategies for safety and reachability, when there does not exist a winning strategy for *Player 1* from node

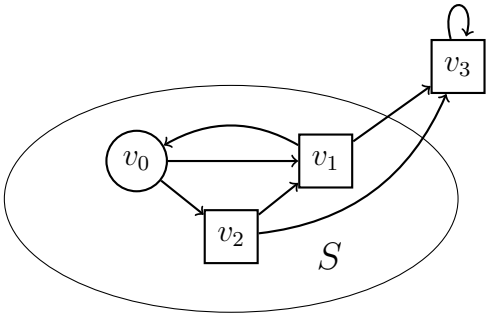

Figure 3.1: Admissible strategy for safety

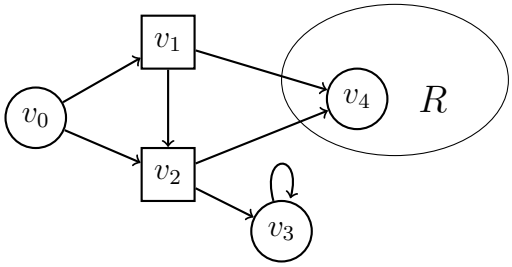

Figure 3.2: Admissible strategy for reachability

v_0. Throughout the thesis *Player 1* nodes are depicted by circles and *Player 2* nodes by squares. Figure 3.1 is an example of a game with a safety objective, where $S = \{v_0, v_1, v_2\}$. Here *Player 1* does not have a winning strategy from v_0 as *Player 2* can always take the game to v_3. But the *Player 1* strategy σ_1, which chooses the edge (v_0,v_1) is better than the strategy σ_2, which chooses the edge (v_0,v_2), because the former performs strictly better than the latter against the *Player 2* strategy $\{v_1 \mapsto v_0, v_2 \mapsto v_3, v_3 \mapsto v_3\}$. Similarly for the reachability game depicted in Figure 3.2 with $R = \{v_4\}$, there does not exist a winning strategy for *Player 1* from v_0. But the *Player 1* strategy σ_1 which chooses the edge (v_0, v_1) is better than the strategy σ_2 which chooses the edge (v_0, v_2), because the former performs strictly better than the latter against the *Player 2* strategy $\{v_1 \mapsto v_4, v_2 \mapsto v_3\}$. These two examples also show that treating all *Player 2* nodes as *Player 1* nodes and finding winning strategies for *Player 1* does not

identify the admissible strategies for safety and reachability games as claimed by Faella [3].

Chapter 4

Admissible Strategies for Safety Games

In this chapter, the algorithm is described for finding admissible strategies for safety games and proof for its correctness is provided. Here the controllable predecessor operator $CPre(T)$ for $T \subseteq V$ determines the set of nodes from which *Player 1* can force the game to move to a node in T in one step: $CPre(T) = \{u | u \in V_1 \, and \, \exists v \in T, (u,v) \in E\} \cup \{u | u \in V_2 \, and \, \forall v, (u,v) \in E \Rightarrow v \in T\}$.

Algorithm 1 presented below computes the set of winning nodes for safety games specified by a safe set S, while playing from the perspective of *Player 1*. It computes the winning nodes by computing the greatest fixed point of a monotone operator on $(2^V, \subseteq)$ [9].

Before presenting the algorithm for computing admissible strategies for safety games (Algorithm 2), a few definitions are needed. These definitions apply to the subgraph $G' := (S, E')$ of G induced by the safe set S of nodes, since for nodes in $V \backslash S$ all strategies are losing for *Player 1*.

A *potentially winning node* is a *Player 2* node not in W (*i.e.*, it is a losing node for *Player 1*) which lies on a cycle contained in the safe set S. A *safety-admissible*

Algorithm 1 Winning Nodes for Safety Games

Input Game structure $G := (V, E)$ and $S \subseteq V$, the set of safe nodes
Output Set of winning nodes W

1: $T :=$ S
2: $Y_0 := T$
3: i := 1
4: $Y_i := T \cap CPre(Y_{i-1})$
5: if $Y_i \neq Y_{i-1}$ then i := i + 1. Goto step 4
6: $W := Y_i$
7: **return** W

cycle (s-admissible cycle in brief) is a cycle that lies within S and contains at least one potentially winning node and does not contain a node in W. An s-admissible cycle C with a set of potentially winning nodes H is *minimal* if there is no other s-admissible cycle C' with the set of potentially winning nodes H' such that H' is strictly contained in H. *Paths*(u,v) is the set of paths from u to v in the subgraph G' of G induced by S. Minimal s-admissible cycles play a crucial role in identifying all admissible strategies for safety games. At any *Player 1* node v in a minimal s-admissible cycle C, if *Player 1* chooses its successor in C as the next node, then this gives rise to an admissible strategy. Also, for a *Player 1* node v in $S \backslash W$, which is not in an s-admissible cycle, the work defines a *minimal safety-admissible path* (minimal s-admissible path in brief) from v as a path P from v in S leading either to a minimal s-admissible cycle or to a node in W satisfying the following minimality condition: for no other path Q from v leading either to a minimal s-admissible cycle or to a node in W is the set of potentially winning nodes on Q a proper subset of the corresponding set on P. At any *Player 1* node v on a minimal s-admissible path P, if *Player 1* chooses its successor in P as the next node, then this also gives rise to an admissible strategy. The work claims that identifying all minimal s-admissible cycles and minimal s-admissible paths is necessary and sufficient for identifying all admissible strategies. This claim is

proved in Theorem 1 below.

Algorithm 2 is presented, which computes the set of admissible strategies for safety games. In step 1, the winning set of nodes W for *Player 1* is computed using Algorithm 1. In step 2, the subgraph G' of G induced by S is constructed. For every node v that is in $(V_1 \cap S) \backslash W$, Algorithm 3 is executed to determine the set of admissible strategies. In steps 5,6 and 7 the work outputs the set of all admissible strategies for *Player 1* nodes. If a node is winning, *Player 1* can play according to any winning strategy. If a node is not in S, *Player 1* can play arbitrarily. For the remaining nodes, *Player 1* plays according to the strategies obtained from Algorithm 3.

Algorithm 2 Admissible Strategies for Safety Games

Input Game structure $G := (V, E)$ and $S \subseteq V$, the set of safe nodes
Output Set of admissible strategies for *Player 1* nodes

1: Compute the set of winning nodes W for *Player 1* using Algorithm 1
2: Construct subgraph $G' := (S, E')$ of G induced by S
3: Mark every $v \in (V_1 \cap S) \backslash W$ as not visited
4: For every node in step 3 which is not visited, execute Algorithm 3 on v
5: At any *Player 1* node $v \in W$, play according to any winning strategy
6: At any *Player 1* node $v \notin S$, choose an arbitrary move
7: At any remaining *Player 1* node $v \in S$, play according to strategies returned for v in Algorithm 3

Algorithm 3 presented below computes the values of all admissible strategies for a *Player 1* node v, considering the induced subgraph $G' := (S, E')$. For each such node v set $PW(v)$ is maintained, each element of which is a pair consisting of one of the following:

1. a set of potentially winning *Player 2* nodes on an s-admissible path from v to a set in W and v' the successor of v on such a path, or

2. a set of potentially winning *Player 2* nodes on an s-admissible cycle containing v, and v' the successor of v on such a cycle, or

3. a set of potentially winning *Player 2* nodes on a path from v which is an s-admissible path followed by an s-admissible cycle (also called a *safety-admissible lasso* (s-admissible lasso in brief)) and v' the successor of v on such a lasso.

The algorithm explores every path starting from v. Whenever one of the following happens, update $PW(v)$:

1. a node $u \in W$ is encountered, or
2. an s-admissible cycle containing v is found or,
3. an s-admissible path leading to an s-admissible cycle is found.

Admissible strategies are determined by finding the minimal sets $MinPW(v)$ in $PW(v)$, *i.e.*, ones that are not strictly contained in any other set, for each successor v' of v. Then an admissible strategy will always choose an edge (v, v') which lies on a path containing the set of potentially winning nodes v such that $(v, v') \in MinPW(v)$.

Theorem 1. *The following statements are equivalent:*

1. *σ is an admissible Player 1 strategy*
2. *$\sigma(v) = v'$, where v is a Player 1 node satisfying one of the following conditions:*
   (i) $v \in S \backslash W$ and v lies on a minimal s-admissible cycle, or v lies on a minimal s-admissible lasso or there is a minimal s-admissible path from v to a node in W, and v' is the successor of v on such a cycle or lasso or path, or
   (ii) $v \in W$ and $\sigma'(v) = v'$ for some winning strategy σ', or
   (iii) v is any other node, i.e., $v \notin S$, and $\sigma(v)$ is any successor of v in G

Algorithm 3 Values of admissible strategies at $v \in (V_1 \cap S) \backslash W$

Input $v \in V_1$

Output Values of all admissible strategies for v

1: Mark v as visited

2: Initialize $PW(v) := \emptyset$

3: Explore every path from v and update $PW(v)$ when one of the following happens:

Case 1. *A node $u \in W$, the winning set of nodes is encountered*
$PW(v) := PW(v) \cup (\{X | X$ *is the set of all Player 2 nodes on the path from* v *to* $u \in W\}, v')$, *where* v' *is the successor of* v *on the path*

Case 2. *An s-admissible cycle containing v is found*
$PW(v) := PW(v) \cup (\{X | X$ *is the set of all Player 2 nodes on the s-admissible cycle containing* $v\}, v')$, *where* v' *is the successor of* v *on the s-admissible cycle*

Case 3. *An s-admissible lasso is found*
$PW(v) := PW(v) \cup (\{X | X$ *is the set of all Player 2 nodes on the s-admissible lasso*$\}, v')$, *where* v' *is the successor of* v *on the s-admissible lasso*

4: Determine all minimal s-admissible paths and cycles by comparing the sets in $PW(v)$ for each v' with respect to containment and store them in $MinPW(v)$

5: Store the (v, v') obtained in step 4 in the set $Next(v)$

6: If $Next(v) \neq \emptyset$ then, **return** each element of $Next(v)$ as values of admissible strategies at v

7: else **return** all pairs $(v, v') \in E$ as values of admissible strategies at v

Proof. 1 ⇒ 2

Suppose σ is an admissible *Player 1* strategy such that for some *Player 1* node $v \in S \backslash W$ $\sigma(v) = v'$ is not the successor of v in a minimal s-admissible cycle, a minimal s-admissible path or a minimal s-admissible lasso. Clearly v' must either be on an s-admissible path or on an s-admissible cycle or lasso, since otherwise even *Player 2's* cooperation cannot ensure that the resulting play is in S. Suppose v' is on an s-admissible cycle C', which has the set of potentially winning nodes H' which is not minimal, and hence by assumption there is a successor v'' of v on an s-admissible path leading to a node in W, cycle or lasso with the potentially winning node set being H", such that H" is strictly contained inside H'. Let $u \in H' \backslash H$". Then $val(\sigma, \rho, v) < val(\sigma', \rho, v)$, where σ is identical to σ' at all nodes except at node v where $\sigma'(v) = v''$ and the *Player 2* strategy ρ plays according to the winning strategy for *Player 2* at u *i.e.*, it takes the game outside S (which exists by definition of a potentially winning node) and cooperates at every other node, *i.e.*, chooses a successor node in the s-admissible cycle. This is a contradiction. The case for v' being on an s-admissible path or lasso is similar.

2 ⇒ 1

Suppose σ is a *Player 1* strategy satisfying the following: for $v \in S \backslash W$ lying on an s-admissible path or cycle $\sigma(v) = v'$, where v' is the successor of v in a minimal s-admissible cycle; for $v \in W$, σ plays according to any winning strategy, and for any other v plays arbitrarily. Suppose σ is not admissible, *i.e.*, it is dominated by a *Player 1* strategy σ'. Clearly for nodes in W and nodes in $S \backslash W$ from which there is no s-admissible paths, cycles or lassos, σ' cannot be better than σ. Suppose $val(\sigma, \rho, v) < val(\sigma', \rho, v)$ at some node v, where v lies on an s-admissible cycle, path or lasso. This implies that $Out(\sigma, \rho, v) \notin Win$ but $Out(\sigma', \rho, v) \in Win$. Then $Out(\sigma', \rho, v)$ must be an s-admissible cycle, s-admissible lasso or contain an s-admissible path leading to a winning node. By assumption $\sigma(v) = v'$ for

$v \in S \backslash W$ lying on a minimal s-admissible path or cycle where v' is the successor of v on such a path, cycle or lasso. This means that there is a potentially winning *Player 2* node u on an s-admissible path, lasso or cycle from v which occurs on $Out(\sigma', \rho, v)$ but not on $Out(\sigma, \rho, v)$. This implies there is a *Player 2* strategy ρ (in which *Player 2* cooperates at every potentially winning node on $Out(\sigma, \rho, v)$ but not at u) where $Out(\sigma, \rho, v) \in Win$ but $Out(\sigma', \rho, v) \notin Win$. This is a contradiction since σ is dominated by σ'. □

Corollary 1. *Algorithm 2 finds all admissible strategies for a safety game.*

Proof. A *Player 1* node v is losing if all its successor nodes are losing. From Theorem 1 the correctness of admissible strategies for minimal s-admissible paths, cycles and lassos is obtained. From the algorithm, there are three cases:
Case 1: v has a minimal s-admissible path to $u \in W$. If *Player 1* plays according to the strategy that leads to such a path, then it is admissible as *Player1* can play according to any winning strategy from u.
Case 2: v belongs to a minimal s-admissible cycle. A strategy $\sigma(v) = v'$ where (v, v') lies in a minimal s-admissible cycle is an admissible strategy, whose correctness is proved by Theorem 1.
Case 3: v belongs to a minimal s-admissible path that leads to an s-admissible cycle. A strategy $\sigma(v) = v'$ where (v, v') lies in a minimal s-admissible path that leads to an s-admissible cycle is an admissible strategy, as a consequence of Theorem 1. □

4.1 Complexity

The complexity of Algorithm 1 is $O(|V| + |E|)$. For Algorithm 2, determining all possible cycles and paths for a node v is $O((|V|+|E|)*(|C|+1))$ [10] where $|C|$ the number of cycles. Comparing the cycles and paths for minimality takes $O(|P|^2 +$

$|C|^2$) time, where $|P|$ is the number of paths. Hence the overall complexity can be expressed as: $O(|V|+|E|+|V_1 \backslash W|(|P|^2+|C|^2+(|V|+|E|)*(|C|+1)))$, which is $O(|V|!)$, *i.e.*, $O(2^{|V|log|V|})$.

Chapter 5

Admissible Strategies for Reachability Games

The reachability goal requires the game to visit a node in a set R after finitely many steps. Algorithm 4 [9] is presented for computing the set of winning nodes in a reachability game. Algorithm 4 computes the winning set of nodes by computing a least fixed point of a monotone operator on $(2^V, \subseteq)$ [9].

Algorithm 4 Winning Nodes for Reachability Games

Input Game structure $G := (V, E)$ and $R \subseteq V$, the set of reachable nodes
Output Set of Winning States W

1: $T :=$ R
2: $Y_0 := T$
3: i := 1
4: $Y_i := T \cup CPre(Y_{i-1})$
5: if $Y_i \neq Y_{i-1}$ then i := i + 1. Goto step 4
6: $W := Y_i$
7: **return** W

Before presenting the algorithm for computing admissible strategies for reachability games (Algorithm 5), few definitions are required. These definitions apply to the nodes which are not winning for *Player 1* (*i.e.*, not in W) in the reachability

game.

A *frontier node* is a *Player 2* node which is in $V \backslash W$ and has at least one edge to a node in W. The set of all frontier nodes is denoted by *Front*. $FPaths(v, u)$ denotes the set of all paths from the *Player 1* node v in $V_1 \backslash W$ to the *Player 2* node u in *Front* that do not include any other *Player 2* node w in *Front*. $Front(v)$ denotes the set of all frontier nodes u for which $FPaths(v, u) \neq \emptyset$.

A *reachability-admissible path* (r-admissible path in brief) P from v in $V_1 \backslash W$ to a node u in *Front* is a path that satisfies the following properties:

1. $P \in FPaths(v, u)$, and

2. there is no other path P' from v to u such that $F(P') \subset F(P)$ where $F(Q)$ is the set of *Player 2* nodes on the path Q that are not in *Front*.

A *potentially hazardous* node is a node in $(V_2 \backslash Front) \backslash W$ which has an outgoing path satisfying one of the following conditions:

1. It can be extended to an infinite path that does not contain a node in *Front*, or

2. It can be extended to two or more paths that end in distinct nodes in *Front* and do not contain any other node in *Front*.

The set of all potentially hazardous nodes is denoted by PH. To determine the values of *Player 1* admissible strategies at a node v from which W is reachable in the graph G, a rank is assigned to each node in the components of the underlying undirected subgraph G' of G induced by $Front(v)$. The idea is an admissible strategy would choose a successor of v that lies on an r-admissible path to a highest ranked node in $Front(v)$. If all r-admissible paths to highest rank nodes have at least one potentially hazardous node satisfying condition 1 (in the definition of a potentially hazardous node), then an admissible strategy

would also choose a successor of v that lies on an r-admissible path to the next highest ranked node in $\mathit{Front}(v)$. To differentiate between the two conditions in the definition of a potentially hazardous node, ranks are assigned to the vertices in $V \backslash (W \cup \mathit{Front})$ as well. The rank for the frontier nodes is defined inductively starting with rank 1 for the nodes in G' which have out-degree 0. Any node with unassigned rank that has an edge to a rank i node is assigned rank $i + 1$. For example, in Figure 3.2 the rank of v_2 is 1 and v_1 is 2 and the *Player 1* strategy σ_1 that chooses the edge (v_0, v_1), *i.e.*, toward a higher ranked node, is better than the *Player 1* strategy σ_2 that chooses the edge (v_0, v_2), *i.e.*, toward a lower ranked node, against the *Player 2* strategy $\{v_1 \mapsto v_4, v_2 \mapsto v_3\}$ as explained in Chapter 3. The rank of the remaining nodes in $V \backslash (W \cup \mathit{Front})$ is computed as follows:

1. All *Player 1* nodes and *Player 2* nodes that are not potentially hazardous are assigned rank 0.

2. The potentially hazardous nodes satisfying condition 1 (in the definition of a potentially hazardous node) are assigned rank -1.

3. The potentially hazardous nodes satisfying condition 2 (in the same definition) are assigned the rank of the lowest ranked node in $\mathit{Front}(v)$ to which they have an r-admissible path.

Figure 5.1 explains the need for assigning ranks to nodes in $V \backslash (W \cup \mathit{Front})$. *Player 2* can prevent *Player 1* from entering W forever by choosing the edge (x, z) at the potentially hazardous node x while the potentially hazardous node y will always help *Player 1* in reaching a frontier node u or w depending on the outgoing edge *Player 2* chooses at y. These nodes are differentiated while determining the admissible strategy for *Player 1* at v, hence x is assigned the rank -1 and y the rank 0.

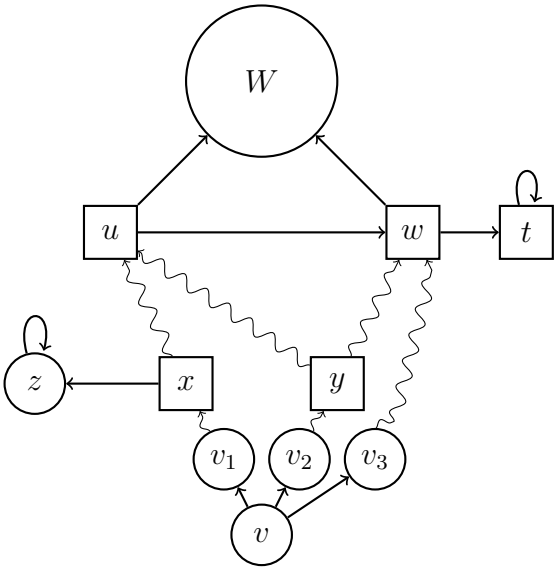

Figure 5.1: Example demonstrating conditions

Algorithm 5 computes the set of admissible strategies for reachability games. In step 1, the winning set of nodes are computed W for *Player 1* using Algorithm 4. Since any winning strategy of a *Player 1* winning node is *admissible*, the rest of the algorithm computes values of admissible strategies for the losing nodes. In step 3, the set of frontier nodes are computed and stored in *Front*. In steps 5-14, the ranks of the remaining nodes are computed in $V \backslash W$. The loop terminates when all nodes in $V \backslash (W \cup Front)$ have been assigned a rank. In step 15, the set of all nodes are computed in $Front(v)$ for $v \in V_1 \backslash W$. In steps 17-30, the induced subgraph G' is constructed for the nodes in $Front(v)$, if $Front(v) \neq \emptyset$. In step 31, Algorithm 6 is executed to compute the rank of every vertex in the induced subgraph G' and the remaining vertices in $V \backslash W$. In step 34, Algorithm 7 is executed to compute the values of admissible strategies for every component C of the underlying undirected graph of G'. In steps 40-42, the strategies for *Player 1* are returned.

Algorithm 6 computes the rank for the nodes in G'. In steps 5 and 6, the nodes with out-degree 0 to 1 are initialized. In steps 9-12, ranks to the remaining nodes in G' are assigned by checking if they have an outgoing edge to a node with the previously assigned rank. The value of rank at every iteration is incremented. The algorithm terminates when every node of G' has been assigned a rank.

Algorithm 7 computes the values of admissible strategies at v for component C of the underlying undirected graph of G'. The following predicates are defined to describe the algorithm:

1. $\text{Lies}(v, v', P) \triangleq$ the edge (v, v') lies on the path P from v;
2. $\text{rAdmissible}(P, v, S, r) \triangleq P$ is an r-admissible path from v to a node in S where all nodes in S have rank r.

The following sets are defined to describe the algorithm:

1. $PH(P) \triangleq$ the set of potentially hazardous nodes on the r-admissible path P;
2. $\text{Nodes}(P) \triangleq$ the set of nodes on an r-admissible path P except the first and last one.

Let X be the set of all nodes u in C, the present component of G' whose rank is r, where r is the current rank in the iteration. Let h be the highest rank assigned to any node in C. The algorithm includes v' in the set A of values of admissible strategies at v when one of the following conditions is satisfied, where the conditions are checked from top to bottom:

1. $cond1 \triangleq \exists Q, r[r = h \wedge \text{rAdmissible}(Q, v, X, r) \wedge \text{Lies}(v, v', Q) \wedge PH(Q) = \emptyset]$
2. $cond2 \triangleq \exists Q, r[r \neq h \wedge \text{rAdmissible}(Q, v, X, r) \wedge \text{Lies}(v, v', Q) \wedge PH(Q) \neq \emptyset \wedge \forall w \in \text{Nodes}(Q)[rank(w) = 0 \ \vee \ r \leq rank(w) \leq h]]$

3. $cond3 \triangleq \exists Q, r[r \neq h \wedge \text{rAdmissible}(Q, v, X, r) \wedge \text{Lies}(v, v', Q) \wedge PH(Q) = \emptyset]$

4. $cond4 \triangleq \exists Q, r[\text{rAdmissible}(Q, v, X, r) \ \wedge \ \text{Lies}(v, v', Q) \ \wedge \ PH(Q) \neq \emptyset]$

Figure 5.1 explains the need for conditions *cond*1 to *cond*4. The r-admissible path $(v - -x - -u)$ illustrates the use for *cond*4 due to the edge (x, z) which results in x having rank -1. Further, the r-admissible paths $(v - -y - -u)$ and $(v - -y - -w)$ show the use for *cond*2 due to the nature of y whereas the path $(v - -w)$ shows the use for *cond*3 as the path does not have any potentially hazardous node. Since *cond*1 is not satisfied (which is the same as *cond*3 except the path should be from v to u instead of w), the *Player 1* strategy σ_1 that chooses (v, v_1) that lies on the r-admissible path $(v - -x - -u)$ is no better than the *Player 1* strategy σ_2 that chooses (v, v_2) that lies on any of the two r-admissible paths $(v - -y - -u)$ or $(v - -y - -w)$ against any *Player 2* strategy. On the other hand, the *Player 1* strategy σ_2 that chooses (v, v_2) performs better than the *Player 1* strategy σ_3 that chooses (v, v_3) that lies on the r-admissible path $(v - -w)$ against the *Player 2* strategy $\{u \mapsto k \in W, w \mapsto t, t \mapsto t\}$. Hence σ_1 and σ_2 are admissible strategies and not σ_3. The algorithm iterates from the highest rank to the lowest rank in C. For each rank r, it checks from conditions *cond*1 to *cond*4 in order. Whenever a condition from *cond*1 to *cond*3 is satisfied, it returns the set A. If *cond*4 is satisfied, the algorithm proceeds to the next lower rank after storing the set A of possible values of admissible strategies at node $v \in V_1 \backslash W$.

The following results are needed to prove the correctness of Algorithm 5 (Corollary 2). Lemma 1 states that any admissible strategy would choose an edge along an r-admissible path from any node $v \in V_1 \backslash W$ from which W is reachable. Lemma 2 states that if any of the conditions *cond*1 or *cond*2 or *cond*3 is satisfied for an r-admissible path which terminates at a node in *Front* with rank r, then no strategy that chooses an edge along an r-admissible path which

Algorithm 5 Admissible Strategies for Reachability Games

Input Game structure $G := (V, E)$ and $R \subseteq V$
Output Set of admissible strategies for *Player 1*

1: Compute W for *Player 1* using Algorithm 4
2: $G' := null$
3: $Front := \{v | v \in V_2 \backslash W \; and \; \exists u.(v, u) \in W\}$
4: **for** every $v \in V_1 \backslash W$ **do**
5: **for** every node $u \in V_1 \backslash W \cup V_2 \backslash (W \cup Front \cup PH)$ **do**
6: $rank(u) := 0$
7: **end for**
8: **for** every node $v \in PH$ **do**
9: **if** v is the source of an infinite path that does not contain a node in *Front* **then**
10: $rank(v) := -1$
11: **else**
12: $rank(v) := r$, where r is the lowest ranked node in *Front* to which it has an r-admissible path
13: **end if**
14: **end for**
15: Let $Front(v)$ be the set containing all $u \in Front$ such that $FPaths(v, u) \neq \emptyset$.
16: **if** $Front(v) \neq \emptyset$ **then**
17: $G' = (V', E') := (\emptyset, \emptyset)$
18: **for** every vertex $p \in Front(v)$ **do**
19: $V' := V' \cup \{p\}$
20: **end for**
21: **for** every pair of nodes $p_1, p_2 \in Front(v)$ **do**
22: **if** all paths from p_1 lead to p_2 in $G \backslash W$ such that no node in *Front* repeat itself and do not contain a node u with $rank(u) = -1$ **then**
23: **if** $(p_2, p_1) \in E'$ **then**
24: $E' := E' \backslash \{(p_2, p_1)\}$
25: **else**
26: $E' := E' \cup \{(p_1, p_2)\}$
27: **end if**
28: Goto step 11.
29: **end if**
30: **end for**
31: Call Algorithm 6 to compute the rank of nodes in G'

32: $S(v) := \emptyset$ // S is the set of all v' s.t. $\sigma(v) = v'$ for some admissible strategy σ
33: **for** every component C in the underlying undirected graph of G' **do**
34: Execute Algorithm 7 for C
35: $S(v) := S(v) \cup A$
36: **end for**
37: Re-initialize $G' = (V', E')$ with $V' := \emptyset$, $E' := \emptyset$
38: **end if**
39: **end for**
40: At any *Player 1* node $v \in W$, play according to any winning strategy
41: At any *Player 1* node $v \notin W$ such that $Front(v) = \emptyset$, choose an arbitrary move
42: At any other *Player 1* node v, choose any node in $S(v)$

Algorithm 6 Compute rank for nodes in G'

1: **for** every component C of the underlying undirected graph of G' **do**
2: $r := 1$
3: **while** C has vertices left to be assigned a rank **do**
4: **if** $r = 1$ **then**
5: **for** every vertex v in G' with out-degree 0 **do**
6: $rank(v) := r$
7: **end for**
8: **else**
9: **for** every vertex v in G' that does not have a rank and has an edge to v' where $rank(v') = r - 1$ **do**
10: $rank(v) := r$
11: **end for**
12: **end if**
13: $r := r + 1$
14: **end while**
15: **end for**

Algorithm 7 Values of admissible strategies at v for component C

Output $A = \{v' \mid \sigma(v) = v' \text{ for an admissible strategy at } v\}$

1: $A := \emptyset$
2: $h :=$ highest rank in C
3: $l :=$ lowest rank in C
4: **for** $r := h$ to l **do**
5: $flag := 0$
6: $flag1 := 0$
7: $X :=$ set of all nodes in C with rank r
8: Determine the set A' of all v' s.t. (v, v') lies on an r-admissible path Q from v to a node $u \in X$ where $rank(w) = 0$ for every $w \in \text{Nodes}(Q)$
9: $A := A \cup A'$ // Condition $cond1$ holds. Q does not contain any node in PH
10: **if** $A' \neq \emptyset$ **then**
11: $flag := 1$
12: **end if**
13: **if** $r = h$ and $flag = 1$ **then**
14: return A // $A' \neq \emptyset$
15: **end if**
16: Determine the set A' of all v' s.t. (v, v') lies on an r-admissible path Q from v to a node in X // Q contains at least one node in PH
17: **if** $r = h$ **then**
18: $A := A \cup A'$ // Condition $cond2$ holds
19: **else**
20: **if** $A' = \emptyset$ **then**
21: return A // $r \neq h$
22: **else**
23: **for** every $v' \in A'$ **do**
24: $\mathbb{Q} :=$ the set of all r-admissible paths that have the edge (v, v')
25: **for** every $Q \in \mathbb{Q}$ **do**
26: **if** $w \in \text{Nodes}(Q)$ s.t. $rank(w) = -1$ or $1 \leq rank(w) < r$ **then**
27: $flag1 := 1$
28: break
29: **end if**
30: **end for**
31: **if** $flag1 = 0$ **then**
32: $A := A \backslash \{v | v \in A' \text{ in step 8}\}$ // Condition $cond3$ holds
33: return A
34: **end if**
35: **end for**

```
36:          if flag = 1 then
37:            return A
38:          else
39:            A := A ∪ A' from step 16
40:          end if
41:        end if
42:      end if
43:    end for
```

terminates at a node in *Front* with rank $k < r$ is admissible. Lemma 3 states that if there is an r-admissible path P which terminates at a node in *Front* with rank k and there is an admissible strategy that chooses an edge along an r-admissible path which terminates at a node in *Front* with rank $r < k$, then P must satisfy only condition *cond*4 and not any of *cond*1 or *cond*2 or *cond*3. Theorem 2 states that a strategy is admissible iff at least one of the conditions $cond1, \ldots, cond4$ holds when they are checked in that order.

Lemma 1. *If σ is an admissible Player 1 strategy, $v \in V_1 \backslash W$ and the set W is reachable from v in G, then $\sigma(v) = v'$ implies the pair (v, v') satisfies* $\exists Q, r[\text{rAdmissible}(Q, v, X, r) \wedge \text{Lies}(v, v', Q)]$.

Proof. If $v \in V_1 \backslash W$ and W is reachable from v then there is always an r-admissible path from v by considering only the paths that end in the first node in *Front* and choosing those among them that contain a minimal number of *Player 2* nodes. An admissible strategy σ would always choose a successor node v' at v where v' lies on an r-admissible path because such a path contains a minimal set of *Player 2* nodes that can take the game away from W. The proof is along similar lines as Theorem 1 for safety. □

Lemma 2. *If σ is an admissible strategy with $\sigma(v) = v'$ where (v, v') lies on an r-admissible path Q satisfying the following properties:*

1. *Q terminates at a node in Front(v) with rank r and,*

2. no node in Q has rank -1

then, for any other strategy σ' with $\sigma'(v) = v''$ where (v, v'') lies on an r-admissible path Q' that terminates at a node in $Front(v)$ with rank $k > r$, the relation $val(\sigma, \rho, v) > val(\sigma', \rho, v)$ holds for all Player 2 strategies ρ.

Proof. It is observed that if no node u along the r-admissible path Q has $rank(u) = -1$, then the path Q will inevitably reach a node in $Front(v)$ with rank $k \geq 1$ as only the nodes ranked as -1 have the potential to prevent the game from reaching $Front(v)$ forever. Further, it is clear from the computation of rank that for any two nodes $u, w \in Front(v)$ and in the same component of the underlying undirected graph of G' where $rank(u) > rank(w)$, there does not exist any path from w to u and all paths from u lead to w in $G \backslash W$ and do not include any potentially hazardous node u' with $rank(u') = -1$.

From the observations the following cases are to be considered:

Case 1: Q' has at least one node u where $rank(u) = -1$

This implies that a node in $Front(v)$ with rank k may not be reached at all, whereas by following strategy σ, one can definitely reach $Front(v)$ and enter W through one of the nodes $w \in Front(v)$, with $rank(w) \in [r, h]$. Hence $val(\sigma, \rho, v) > val(\sigma', \rho, v)$ in this case.

Case 2: Q' has no node u where $rank(u) = -1$

This implies that the path Q will inevitably reach a node in $Front(v)$ with rank $k \geq 1$. So, by following strategy σ, one can definitely reach $Front(v)$ and enter W through one of the nodes $w \in Front(v)$, with $rank(w) \in [r, h]$ and by following strategy σ', W can be entered through one of the nodes $w \in Front(v)$, with $rank(w) \in [k, h]$ as shown in Figure 5.2. Since $k < r$, if at one of the nodes $u' \in Front(v)$ with $k < rank(u') <= r$ *Player 2* chooses the edge that leads to W and at none of the nodes with rank in $[k, h]$ *Player 2* choose the edge that leads to W, then $Out(\sigma, \rho, v) \in Win$ but $Out(\sigma', \rho, v) \notin Win$. Hence $val(\sigma, \rho, v) >$

$val(\sigma', \rho, v)$ in this case as well. □

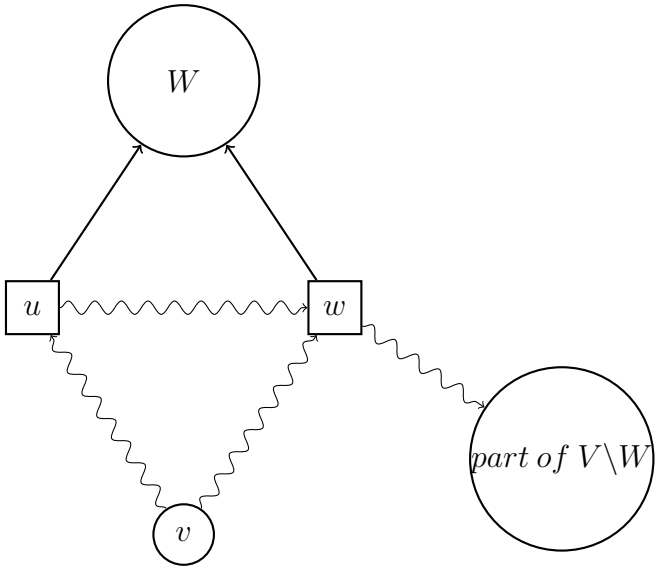

Figure 5.2: Player 1 chooses edge leading to u over edge leading to v

Lemma 3. *1. Suppose σ is an admissible strategy with $\sigma(v) = v'$ and (v, v') lies on an r-admissible path leading to a node in Front(v) with rank $r \neq h$. Then for all admissible strategies σ' with $\sigma'(v) = v''$ where $v' \neq v''$ if the pair (v, v'') lies on an r-admissible path leading to a node in Front(v) with rank k, where $r+1 \leq k \leq h$ then the pair (v, v'') satisfies only cond4 among the four conditions on page 22-23.*

2. Suppose σ' is an admissible strategy with $\sigma'(v) = v''$ and the pair (v, v'') lies on an r-admissible path leading to a node in Front(v) with rank k, where $r + 1 \leq k \leq h$, for some r and further satisfies only cond4 among the four conditions on page 22-23. Further suppose there is a v' with $v' \neq v''$ and (v, v') lies on an r-admissible path leading to a node in Front(v) with rank r. Then there exists an admissible strategy σ satisfying $\sigma(v) = v'$.

Proof. (1): For the sake of contradiction assume that σ is an admissible strategy and for an admissible strategies σ' with $\sigma'(v) = v''$ where $v' \neq v''$ the pair (v, v'')

lies on an r-admissible path Q leading to a node in $Front(v)$ with rank k, where $r+1 \leq k \leq h$ and satisfies $cond1$, $cond2$ or $cond3$. If $cond1$ or $cond3$ is satisfied, then there are no potentially hazardous nodes along the r-admissible path Q. So a node in $Front(v)$ with rank $k > r$ will definitely be reached along all paths starting with the pair (v, v''). From Lemma 2, $val(\sigma', \rho, v) > val(\sigma, \rho, v)$ for any *Player 2* strategy ρ. This implies that $Out(\sigma', \rho, v) \in Win$ but $Out(\sigma, \rho, v) \notin Win$, a contradiction. If $cond2$ is satisfied, then all *Player 2* nodes along the r-admissible path Q have rank $r' \in [r+1, h]$. So a node in $Front(v)$ with rank $k > r$ will definitely be reached along all paths starting with the pair (v, v''). From Lemma 2, $val(\sigma', \rho, v) > val(\sigma, \rho, v)$ for any *Player 2* strategy ρ. This implies that $Out(\sigma', \rho, v) \in Win$ but $Out(\sigma, \rho, v) \notin Win$, again a contradiction.

(2): For the sake of contradiction assume that for all admissible strategies σ' with $\sigma'(v) = v''$ the pair (v, v'') lies on an r-admissible path leading to a node in $Front(v)$ with rank k, where $r+1 \leq k \leq h$, for some r and satisfies only $cond4$. Further, assume there is a v' with $v' \neq v''$ and (v, v') lies on an r-admissible path leading to a node in $Front(v)$ with rank r, and there does not exist an admissible strategy σ satisfying $\sigma(v) = v'$. If only $cond4$ is satisfied for the pair (v, v''), then there is at least one potentially hazardous node u with $rank(u) = -1$ in the r-admissible path Q. From Lemma 2, there is no guarantee that a node in $Front(v)$ with rank $k > r$ will be reached. Hence it cannot be said that $val(\sigma', \rho, v) > val(\sigma, \rho, v)$ for any *Player 2* strategy ρ, thus arriving at a contradiction. Hence both σ and σ' are admissible. □

Theorem 2. *The following statements are equivalent:*

1. *$\sigma(v) = v'$ for an admissible Player 1 strategy σ where $v \in V_1 \backslash W$ and W is reachable from v in G*

2. *the pair (v, v') satisfies one of the following statements where $1 \leq r \leq h$:*

(a) $cond1$

(b) $\sim cond1 \Rightarrow cond2$

(c) $\sim (cond1 \vee cond2) \Rightarrow cond3$

(d) $\sim (cond1 \vee cond2 \vee cond3) \Rightarrow cond4$.

Proof. 1 ⇒ 2

By Lemma 1 (v, v') lies on an r-admissible path Q from v to $u \in \mathit{Front}(v)$. Let u be of rank r.

Case 1: $r = h$

For the sake of contradiction assume that none of the above four statements is true. For this to hold, all the conditions $cond1$ to $cond4$ are required to be false. Since $r = h$, $cond2$ and $cond3$ are false. Since $cond1$ is false and σ is an admissible strategy, it can be concluded that $PH(Q) \neq \emptyset$. Since statement (d) and therefore $cond4$ is false, by Lemma 1 $PH(Q) = \emptyset$ for all r-admissible paths Q, which is a contradiction.

Case 2: $r \neq h$

Again for the sake of contradiction assume that none of the above four statements is true. Since $r \neq h$, $cond1$ is false. Since, Lemma 3 holds for all admissible strategies σ' with $\sigma'(v) = v''$ and $v'' \neq v'$, the pair (v, v'') lies on an r-admissible path leading to a node in $\mathit{Front}(v)$ with rank k, where $r + 1 \leq k \leq h$, for some r and satisfies only $cond4$ among the four conditions on page 22-23. Since by assumption statement (b) is false, $cond2$ is false. Since σ is an admissible strategy, it can be concluded that either of the following two properties must hold for Q:

1. $PH(Q) = \emptyset$
2. $\exists w \in \text{Nodes}(Q)[1 \leq rank(w) < r]$.

Since by assumption, *cond*3 is also false, and σ is an admissible strategy, it can be concluded that $PH(Q) \neq \emptyset$. This implies property 2 must hold and property 1 must not for *cond*2 and *cond*3 to be false at the same time. Since statement (d) and therefore *cond*4 is false, by Lemma 1 $PH(Q) = \emptyset$ for all r-admissible paths Q, which is a contradiction.

2 ⇒ 1

For the sake of contradiction assume that one of the four statements is true and $\sigma(v) \neq v'$ for all admissible strategies σ. If statement (a) is true and therefore *cond*1 is true, from the first observation in the proof of Lemma 2 any r-admissible path starting with the pair (v, v') will inevitably reach a node in $Front(v)$. Hence σ is admissible which is a contradiction. If statement (b) is true and statement (a) is false, then *cond*1 is false and *cond*2 is true, from the first observation in the proof of Lemma 2 any r-admissible path starting with the pair (v, v') will inevitably reach a node in $Front(v)$. Hence σ is admissible, a contradiction. If statement (c) is true and statements (a) and (b) are false, then *cond*1 and *cond*2 are false and *cond*3 is true. Since *cond*3 is true, from the first observation in the proof of Lemma 2 any r-admissible path starting with the pair (v, v') will inevitably reach a node in $Front(v)$. Hence σ is admissible, a contradiction. If statement (d) is true and statements (a), (b) and (c) are false, then only *cond*4 is true. If $r = h$ and conditions *cond*1 to *cond*3 are false, σ is admissible. Otherwise, from Lemma 3 it is known that for all admissible strategies σ' with $\sigma'(v) = v''$, the pair (v, v'') lies on an r-admissible path leading to a node in $Front(v)$ with rank k, where $r + 1 \leq k \leq h$, for some r and satisfies only *cond*4. Hence σ is admissible, which is again a contradiction. □

The following result about the correctness of Algorithm 5 is a consequence of Theorem 2.

Corollary 2. *Algorithm 5 returns all the admissible strategies for a reachability game.*

5.1 Complexity

The time complexity of Algorithm 4 is $O(|V| + |E|)$. Step 3 of Algorithm 5 takes $O(|V| + |E|)$ time. In step 5, computing the set of all paths takes $O((|V| + |E|) * (|P| + 1))$ time [10], where $|P|$ is the number of paths. In steps 16 through 30, to determine an edge from p_1 to p_2 for $p_1, p_2 \in V'$, all paths need to be checked from p_1 to p_2 which takes $O(|P|^2)$ time. Algorithm 6 requires $O(|Front(v)|)$ time to compute the rank of a vertex. Algorithm 7 runs for every component C in G'. The computation of Algorithm 7 takes $O(|P|^2)$ time. Hence the overall complexity of Algorithm 5 is $O(|V| + |E| + |V_1 \backslash W| * ((|V| + |E|) * (|P| + 1) + |P|^2 + |Front(v)| + |C| * |P|^2))$, where $|C|$ denotes the number of components in G' for $v \in V_1 \backslash W$. This is $O(|V|!)$, *i.e.*, $O(2^{|V| log |V|})$.

Chapter 6

Conclusions

The work aims to solve admissible strategies for safety and reachability objectives which are arguably the most important among all non-prefix-independent goals. The work uses graph-theoretic notions to provide a solution for the problem at hand.

The work is a special case of [6; 7] where only two properties are taken into consideration and much simpler algorithms are proposed for them which are also lower in complexity (precisely $O(2^{|V|log|V|})$ time) than the double exponential algorithm in [6; 7].

The algorithms have been proven to be correct and exhaustive. Previous claims made by Faella [3] have also shown to be incorrect. Work on cooperation and rational behavior of players is now an emergent area of research [3; 5; 6; 7; 8] and the work done claims to be the first approach in solving zero sum games with winning conditions as safety and reachability in exponential time with graph-theoretic notions. The work can be extended to solve objectives like *Büchi*, *Muller*, *Parity*, etc. in future. The work is also useful in compositional synthesis where each component is to be composed and admissible strategies are to be determined and shown to hold good.

References

[1] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, pp. 179–190, ACM, 1989. 1

[2] R. Brenguier, J.-F. Raskin, and M. Sassolas, "The complexity of admissibility in omega-regular games," in *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, pp. 23:1–23:10, ACM, 2014. 1, 2, 7

[3] M. Faella, "Admissible strategies in infinite games over graphs," in *Mathematical Foundations of Computer Science (MFCS'09)*, vol. 5734 of *LNCS*, pp. 307–318, Springer, 2009. 2, 3, 9, 34

[4] E. Grädel, W. Thomas, and T. Wilke, eds., *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, vol. 2500 of *LNCS*, Springer, 2002. 3

[5] R. Bloem, R. Ehlers, S. Jacobs, and R. Könighofer, "How to handle assumptions in synthesis," in *Proceedings 3rd Workshop on Synthesis, SYNT 2014, Vienna, Austria, July 23-24, 2014.*, pp. 34–50, 2014. 3, 4, 34

[6] W. Damm and B. Finkbeiner, "Does it pay to extend the perimeter of a

world model?," in *17th International Symposium on Formal Methods (FM 2011)*, vol. 6664 of *LNCS*, Springer, 2011. 3, 4, 34

[7] W. Damm and B. Finkbeiner, "Automatic compositional synthesis of distributed systems," in *FM 2014: Formal Methods* (C. Jones, P. Pihlajasaari, and J. Sun, eds.), vol. 8442 of *LNCS*, pp. 179–193, Springer International Publishing, 2014. 3, 4, 34

[8] D. Fisman, O. Kupferman, and Y. Lustig, "Rational synthesis," in *TACAS 2010*, vol. 6015 of *LNCS*, pp. 190–204, Springer, 2010. 3, 4, 34

[9] W. Thomas, "On the synthesis of strategies in infinite games," in *STACS*, pp. 1–13, 1995. 10, 18

[10] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM J. Comput.*, vol. 4, pp. 77–84, 03 1975. 16, 33