On the algorithmic complexity of two player zero-sum games of finite duration with imperfect information

Soumyajit Paul

université de BORDEAUX

LaBRI

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX

ECOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

SPÉCIALITÉ : INFORMATIQUE

Par **Soumyajit PAUL**

On the algorithmic complexity of two player zero-sum games of finite duration with imperfect information

Sous la direction de **Olivier LY**
et de **Hugo GIMBERT**
et de **B. Srivathsan**

Soutenue le 12 Décembre 2023 devant un jury composé de

M. Olivier LY	Full professor	LaBRI, Université de Bordeaux	Directeur
M. Hugo GIMBERT	Chargé de recherche	CNRS, LaBRI, Université de Bordeaux	Codirecteur
M. B. Srivathsan	Associate professor	CMI, India	Codirecteur
M. Mickael RANDOUR	Professor	Université de Mons	Rapporteur
M. Laurent DOYEN	Chargé de recherche	LMF, ENS Paris-Saclay	Rapporteur
M. Nathanaël FIJALKOW	Chargé de recherche	CNRS, LaBRI, Université de Bordeaux	Examinateur
Mme Nathalie BERTRAND	Directrice de recherche	INRIA, IRISA, Université de Rennes 1	Présidente et Examinatrice

Sur la complexité des jeux à somme nulle à deux joueurs de durée finie avec information imparfaite

Résumé : Dans cette thèse, nous étudions la complexité du calcul des stratégies optimales dans les jeux à somme nulle à deux joueurs avec une information imparfaite. Dans les jeux à information imparfaite, les joueurs n'ont qu'une connaissance partielle de leur position dans le jeu. Cela rend la tâche de calcul des stratégies optimales difficile, en particulier lorsque les joueurs oublient les informations précédemment acquises. Pour mieux justifier cette difficulté, nous considérons des jeux à somme nulle à deux joueurs avec des informations imparfaites modélisées sous la forme extensive et nous présentons plusieurs nouveaux résultats de complexité sur le calcul de la valeur maxmin pour différentes classes d'informations imparfaites. Pour les bornes inférieures, nous considérons des problèmes tels que le problème de la somme de la racine carrée ainsi que des classes de complexité qui impliquent le calcul sur les réels, plus précisément la théorie existentielle des réels (ETR) et d'autres fragments de la théorie du premier ordre des réels (FOT(R)). En revanche, nous identifions une nouvelle classe de jeux à mémoire imparfaite pour lesquels les stratégies optimales peuvent être calculées efficacement. Dans le but d'étudier théoriquement les enchères au bridge, nous proposons également un nouveau modèle d'étude des enchères au bridge. Nous faisons quelques observations initiales pour la classe des jeux de bridge à 2 joueurs avec une seule couleur sur ce modèle et nous traçons la voie pour des recherches futures.
Mots-clés : théorie des jeux, jeux à somme nulle, information imparfaite, complexité, Bridge

On the algorithmic complexity of two player zero-sum games of finite duration with imperfect information

Abstract: In this thesis we study the complexity of computing optimal strategies in two player zero-sum games with imperfect information. In games with Imperfect information players only have partial knowledge about their position in the game. This makes the task of computing optimal strategies hard especially when players forget previously gained information. To further substantiate this hardness we consider two player zero-sum games with imperfect information modeled in the extensive form and provide several new complexity results on computing maxmin value for various classes of imperfect information. For lower bound results we consider problems such as the Square-root sum problem and also complexity classes which involve computation over reals, more precisely Existential Theory of Reals (ETR) and other fragments of the First Order Theory of Reals (FOT(R)). On a positive note we identify a new class of imperfect recall games for which the optimal strategies can be computed efficiently. Towards the goal of studying Bridge Bidding theoretically we also provide a new model for studying Bridge Bidding. We make some initial observations for the class of 2 player Bridge games with single suit on this model and lay down the path for future investigations.
Keywords: game theory, zero-sum game, imperfect information, complexity, Bridge

---

Unité de recherche
Laboratoire Bordelais de Recherche en Informatique, CNRS UMR 5800
Universit´e de Bordeaux, 33405 Bordeaux, France

Acknowledgements

This thesis, in spite of bearing an individual's name as the author, would not have come into existence without the influence that several individuals had in different stages of my life.

Starting with my thesis advisors : thank you Hugo for providing me this rich opportunity pursue PhD at LaBRI and giving me all the time and support I needed during my PhD. The confidence that you bestowed in me during difficult times as well as while taking difficult decisions had a big positive impact on me. I have learnt a lot from your way of tackling problems and also your outlook on research in general. Next, my advisor from CMI, Srivathsan, without whom this PhD would have been almost impossible. Since the very first course taught by you (Timed automata) in BSc at CMI followed by MSc thesis guidance and then continuing to PhD guidance as well, you have been my guide in my academic journey in every sense of the word. No amount of words would be sufficient in this regard to express my gratitude for you playing this important role in my life. I still remember the time when we started exploring problems in game theory and I certainly hope to continue working with you on such interesting problems in the future. And finally thanks to Olivier for agreeing to be my thesis supervisor. You were always there to help me promptly whenever I needed any assistance.

I would like to thank Mickael and Laurent for accepting to be reviewers and undertaking the painful task of going though the manuscript in details. I would also like to thank Nathalie and Nathanaël for agreeing to be part of the jury. All of your comments on the manuscript as well engaging questions during the defense were quite helpful.

I would like to thank Kristoff for the few but insightful discussions we had in person as well as online. Thank you for helping in finding a fix for a error in one of the proofs.

I would like to thank Pascal for giving me advice from time to time which were extremely helpful.

I can't thank enough my PhD support team, my friends from CMI: Govind, Sougata and Sayan for entertaining my shenanigans from time to time and being there in the most difficult of times. Thank you for making this journey a bit less hard and providing moral and technical support whenever necessary.

I can't forget my constants from CMI : Aneek, Sayantan, Prantar, Debsuvra, Ash Da, Shreejit, Ritwik and Prantik. I'm fortunate to have you in my life.

Having spent more than 6 years at CMI, I have shared so many wonderful memories with so many wonderful people: Adwitee Di, Rajit, Ranadeep, Suman, Pranjal, Ritam, Rajarshi, Anirban, Abhisek, Debraj, Utsab, Sarjick Da, Sayantan Sen, Aneesh, Manu, Shanmugapriya, Debangshu Da just to name a few.

My stay in France during PhD can be divided into two chapters : Bordeaux and Paris.

I was lucky to have people in Bordeaux from LaBRI as well as outside LaBRI who made my stay at Bordeaux quite memorable. Kanka da(I'll greatly miss our deep discussions on any topic under the sun), Raj(I'll miss your energetic company) Somen Da, Bisu Da (I'll miss your house parties), Mallika Di, Debam, Tydiane, Varun, Kylian, Karim 1, Karim 2, Trang, Rupayan, Sourav Da, Shomreeta Di.

J'ai également eu la chance d'avoir des amis à Tauzin : Paul (c'est quand la prochaine soirée ?), Hélène, Nanor, Gaby, Emma. Bien que j'aie vécu à Bordeaux, Emma est la seule personne originaire de cette ville que je connaisse. Merci de m'avoir appris pas mal de chose sur la culture française et d'avoir été là dans les moments difficiles.

Merci également à Antonio, Baran, Klara, Marco, Margaux et autres camarades du cours de français pour votre merveilleuse compagnie.

Relocating to a different city in the middle of PhD can be challenging but my stay in Paris turned out to be quite pleasant only due to the friends I made there. Avinandan, Abhisek, Aymeric, Farzad, Essaie, Enrique, Daniel and others, thank you for making my short stay at IRIF memorable.

I would like to thank Olivier for often giving me helpful advice. I would like to thank Florian and Mahsa, I greatly enjoyed working with you. Thanks to Mahsa for being so supportive and providing me guidance whenever necessary.

J'ai la chance d'avoir rencontré Haïfa (nos sorties spontanées me manqueront), Rym (j'ai hâte de visiter la Tunisie avec vous), Greg et Lucrèce dans cette grande ville, qui sont devenus mes proches assez vite. You along with Kanka da, Ranadeep, Raj, Mrinal da, Arnab, Charles made my life at Paris so much enjoyable.

I would like to thank Bala and Ahad for kindly providing me shelter during my first month in France when I was still searching for a house. I would like to thank Pranjal for the spontaneous discussions we often have that habe helped me understand a topic much better.

I cannot possibly forget my school and childhood friends with whom I spend wonderful time every time I return home: Baidya, Ritu, Kushal, Partha, Utpal, Abanti, Suki, Utsab, Pranay, Buru, Ayan, Arpan, Souro and Tejaswini.

I would also like to thank my close relatives, my pisis, mamas and mashis who had taken care of me whenever needed.

I also owe gratitude to my tutors from school time: Mishra aunty, Pankaj sir and Pushpendu sir who helped in keeping alive my interest in mathematics.

Finally, I would like to thank my Maa, Baba and Bhai for always supporting me and understanding my choices however difficult it be for them. It's only because of my parents, that I was fortunate enough to have the privilege of pursuing my own path in this world filled with underprivileged people.

I sincerely apologize if I've missed someone.

Résumé de la thèse

Les jeux à information imparfaite

Résoudre des jeux à durée finie est un problème central dans le domaine de l'intelligence artificielle. L'objectif principal de la recherche dans ce domaine est de construire une IA capable de surpasser les experts humains dans un jeu particulier. Les jeux à information parfaite, dans lesquels les joueurs ont une connaissance complète de l'état du jeu à chaque étape, ont naturellement été la première étape de ce défi. Échecs et go sont des exemples populaires dans cette catégorie, qui ont été au centre de la recherche sur l'IA pendant longtemps. Bien que les jeux à information parfaite puissent être résolus efficacement dans la taille de l'arbre de jeu en utilisant des idées de Zermelo [Zer13], en pratique, la principale difficulté dans le développement de bots pour ces jeux est due à l'énorme taille de l'espace d'états [Sha50]. Le tout premier succès aux échecs a eu lieu en 1996 lorsqu'une IA nommée Deep Blue, développée par IBM, a battu le champion du monde d'échecs Gary Kasparov [IBM02]. Des réussites similaires ont été réalisées récemment avec AlphaGo [SHMG16] de DeepMinds qui a battu le champion de Go, Lee Sedol en 2016 et ensuite AlphaZero [SHSA18] qui est une amélioration d'AlphaGo capable de jouer aux échecs et au go. Avec des réussites importantes dans plusieurs jeux à information parfaite, le prochain défi était les jeux à information imparfaite tels que le Poker et le Bridge. L'information imparfaite dans ces jeux provient du fait qu'un joueur n'a pas connaissance de la distribution des cartes aux autres joueurs. Résoudre les grands jeux à information imparfaite s'est avéré beaucoup plus difficile que les jeux à information parfaite. En ce qui concerne les jeux spécifiques, ce n'est que très récemment que les bots de poker, Libratus [BS17], DeepStack [MSBL17] et Pluribus [BS19] ont été en mesure de battre les champions humains, dans certaines configurations de jeu. Le développement de bots pour différents jeux avec des informations imparfaites et la démonstration de garanties théoriques est toujours un domaine de recherche actif en théorie des jeux.

La motivation principale de cette thèse est d'étudier la complexité des jeux à information imparfaite. Le bridge est une classe spécifique de jeux multi-joueurs à information imparfaite appelés jeux d'équipe, où deux équipes de joueurs ont des intérêts opposés, où les joueurs d'une même équipe ont les mêmes gains, mais où les joueurs ne peuvent pas communiquer librement, même au sein d'une même équipe. Dans les jeux à information imparfaite, les joueurs ont une mémoire parfaite si, à tout moment du jeu, ils peuvent se souvenir de toutes leurs actions précédentes.

La classe des jeux à somme nulle à deux joueurs où les joueurs ont une mémoire parfaite se résout efficacement à l'aide de la programmation linéaire [KM92]. En revanche, lorsque les joueurs ont une mémoire imparfaite, cela devient plus difficile [KM92][CBHL18]. Bien que dans Bridge chaque joueur individuel ait une mémoire parfaite, le jeu n'est pas un jeu à deux joueurs mais un jeu à quatre joueurs. Il est intéressant de noter que la suppression de l'hypothèse de mémoire parfaite dans les jeux à somme nulle à deux joueurs suffit à englober les jeux d'équipe comme le bridge : le manque de communication entre les joueurs pour l'échange de leurs informations privées peut être modélisé par une mémoire imparfaite. Les jeux à mémoire imparfaite peuvent également être utilisés pour abstraire les grands jeux à mémoire parfaite et obtenir des améliorations de calcul significatives de manière empirique [SL01][GS07][KS14][BS15][ČLB20]. Dans cette thèse, nous nous intéressons à la complexité de résoudre ces genres de jeux, en particulier les jeux avec rapperl imparfait.

La complexité du problème maxmin

Dans le cadre de la théorie des jeux, la valeur maxmin est un concept fondamental pour analyser les jeux. Le gain à la fin du jeu dépend de la façon dont les joueurs ont choisi leurs stratégies. Dans un jeu à deux joueurs avec les joueurs Max et Min, lorsque Max joue la stratégie σ et Min joue la stratégie τ, le gain gagné par Max est noté $\mathbb{E}[\sigma, \tau]$, et le gain maxmin est :

$$\max_{\sigma} \min_{\tau} \mathbb{E}(\sigma, \tau)$$

où σ et τ sont les stratégies de Max et Min respectivement. Le gain maxmin est le meilleur gain possible pour Max si Min choisit sa stratégie après avoir observé la stratégie choisie par Max. Plusieurs algorithmes ont été proposés pour calculer le gain maxmin de manière exacte ou approximative pour plusieurs classes de jeux de forme extensive [BML93] [KM92] [Ste96] [ZJBP07] [CBL17]. La question de la complexité se pose naturellement pour comprendre la faisabilité de ces algorithmes en général.

Le problème de décision correspondant est le suivant : étant donné un jeu à deux joueurs à somme nulle et à information imparfaite G sous forme extensive, la valeur maxmin de G sur les stratégies randomisées est-elle non-négative ? Étant un problème de décision fondamental, sa complexité est bien étudiée [KM92] [Ste96][HMS07][BP17] . Koller et Meggido ont montré dans leur travail classique que le problème maxmin peut être résolu en temps polynomial lorsque les joueurs ont une mémoire parfaite [KM92]. Ils ont montré qu'en supprimant l'hypothèse de mémoire parfaite, pour les jeux généraux de forme extensive, le problème

de décision maxmin devient NP-difficile. Suite à ces travaux, la complexité du problème de décision maxmin a été étudiée pour les jeux à mémoire imparfaite [HMS07][BP17].

Contribution de cette thèse

Nouvelles bornes de complexité

Dans cette thèse, nous présentons une image plus précise de la complexité du problème de décision maxmin pour les jeux avec des joueurs ayant une mémoire imparfaite. Aucune borne inférieure pour ce problème n'était connue pour la classe spécifique de mémoire imparfaite appelée jeux à des joueurs distraits (absentminded). Nous donnons des nouvelles bornes de complexité pour les jeux de cette catégorie Pour des bornes inférieures concernant les joueurs distraits, nous utilisons des classes de complexité issues de fragments de *Théorie des réels du premier ordre* ($\mathrm{FOT}(\mathbb{R})$), tels que $\exists\mathbb{R}$, $\forall\mathbb{R}$ et $\exists\forall\mathbb{R}$.

Toutes ces classes de complexité $\exists\mathbb{R}$, $\forall\mathbb{R}$ et $\exists\forall\mathbb{R}$ se retrouvent dans PSPACE [Can88][BPR06]. La complexité des jeux relatifs à la classe $\exists\mathbb{R}$ a déjà été étudiée, particulièrement pour les problèmes de décision liés au calcul de l'*équilibre* [GMVY15] [SS17] [BM16][BM21][BH22a][BH22b]. Pour les jeux à joueurs non distraits, nous montrons que le problème est co-NP-difficile même dans le cas où le joueur Max est un joueur trivial et que Min est pas distrait. Nous montrons également des bornes inférieures relatives au problème du Square-Root-Sum . Le problème de décision "Square-Root-Sum " est le suivant : Étant donné k entiers positifs $a_1, \ldots, a_k$ et un autre entier positif n, est-ce que $\sum_i \sqrt{a_i} \geq n$?

Ce problème apparaît souvent lors de l'étude de la complexité des problèmes numériques, comme indiqué dans [EY05]. Des études sur la complexité des jeux utilisent le problème Square-Root-Sum comme borne inférieure [EY10] [HMS10]. Dans notre cas, ce problème trouve sa place parce qu'il a été démontré que pour les stratégies optimales maxmin, les probabilités irrationnelles sont indispensables. Pour les jeux où Max a une mémoire *A-loss* et Min une mémoire parfaite, nous montrons que le problème du maxmin est Square-Root-Sum-difficile. Cela résout la question ouverte posée dans [CBHL18]. Comme on estime que le problème Square-Root-Sum n'est pas dans NP, la borne inférieure Square-Root-Sum et la borne inférieure co-NP renforcent l'idée que le problème maxmin n'est pas dans NP.

Nouvelle classe de mémoire imparfaite résoluble efficacement

En revanche, nous identifions une classe de jeux à joueurs non distraits appelés *A-loss recall shuffle* pour lesquels il est possible d'utiliser le programme linéaire pour calculer le gain optimal efficacement. Un joueur a une mémoire A-loss recall shuffle lorsque l'historique de ses actions peut être réorganisé de manière à obtenir une mémoire de A-loss. Nous démontrons que les jeux avec une mémoire A-loss recall shuffle *équivalents* à un jeu avec une mémoire A-loss de taille linéaire. Cette classe élargit la classe des jeux précédemment connus pour être résolus en temps polynomial. Grâce à cela, nous pouvons calculer la valeur maxmin en temps polynomials dans le cas où Max a une mémoire parfaite et Min a une mémoire A-loss recall shuffle.

Nous fournissons également une généralisation de cette classe appelée *A-loss recall span*. Nous prouvons que tout jeu à joueurs non distraits est équivalent à un jeu avec une mémoire A-loss recall, sous réserve que la taille du jeu de mémoire A-loss recall puisse augmenter de manière exponentielle dans le pire des cas. Nous obtenons ainsi un algorithme permettant de calculer la valeur optimale d'un jeu à un seul joueur en un temps polynomial dans la taille de son plus petit jeu équivalent de mémoire A-loss . De la même manière que pour le jeu A-loss recall shuffle, nous étendons cette technique au cas des jeux à deux joueurs où Max a une mémoire parfaite.

Étude des enchères au bridge

Dans le cadre de la longue tradition de développement de bots pour les jeux dans la recherche sur l'IA et la théorie des jeux, le bridge en fait partie [Gin99][Gin01](voir [Bet21] pour une étude récente). Le bridge est un jeu d'équipe à 4 joueurs (2 dans chaque équipe) avec une information imparfaite qui consiste en deux phases : la phase d'enchères et la phase de jeu. Le bridge est l'un des jeux pour lesquels il reste peu de succès contre les experts humains par rapport à d'autres jeux populaires tels que les échecs, le Go, etc. Ce n'est que très récemment, en 2022, que le bot de bridge de Nukkai a battu 8 champions de bridge dans une version limitée du jeu, sans enchères. Les programmes actuels utilisent l'*évaluations double mort* (où l'on suppose que l'information est parfaite) pour évaluer la phase de jeu. Cependant, la recherche actuelle sur le bridge est privée d'une analyse théorique de la phase d'enchères ainsi que de la difficulté de calculer de bonnes stratégies d'enchères.

Dans cette thèse, nous proposons un modèle pour étudier les enchères au bridge et nous présentons quelques résultats préliminaires sur le calcul de la valeur maxmin dans ce modèle. Bien que le bridge soit un jeu d'équipe à 4 joueurs, nous nous concentrons tout d'abord sur une version plus simple de ce modèle.

Nous considérons la version à somme nulle du jeu pour deux joueurs où les cartes ne peuvent avoir qu'une seule couleur (au lieu de 4). Nous imposons plusieurs contraintes aux stratégies telles que (i) les stratégies non-randomisées et (ii) les stratégies de *surenchère* limitée : les stratégies où les joueurs ne peuvent pas enchérir au-delà d'un certain seuil. Nous étudions quels types de stratégies sont suffisants pour assurer l'optimalité de maxmin.

Nous construisons des exemples où ces classes restreintes de stratégies ne sont pas assez puissantes stratégiquement pour garantir le maxmin. Nous montrons également que la complexité du calcul de la valeur maxmin dépend du nombre de surenchères nécessaires à un joueur pour obtenir l'optimalité maxmin. Nous fournissons une borne sur le nombre de surenchères nécessaires pour l'optimalité maxmin en fonction des *croyances* des joueurs.

Contents

Chapter 1

Introduction

In 2018, in a professional football match in the FA Women's super league, referee David McNamara just moments before the pre-game toss, finds out that he forgot to carry a coin [BBC18]. In order to not waste time running and fetching a coin, he proceeds with a quick and classic solution that we have all known since childhood: a game of rock, paper, scissors between the two captains. Unfortunately this act gets him banned from refereeing activities for a few matches. The FA might consider tossing a coin to be fairer than a game of rock, paper, scissors but it is arguably one of the most widespread ways to settle ties playfully. But why is rock, paper, scissors so popularly considered as a fair way to settle ties? Besides being a super simple game it is considered fair because both the players *play* simultaneously. In classical *game theory* this can be modeled in the *strategic* or *normal form* as in Fig. 1.1. The Captains choose a row and a column simultaneously and the matrix entry corresponding to their choices decides the winner: 1 signifies Captain 1 wins, -1 signifies Captain 2 wins and 0 signifies a draw.

At first instance playing simultaneously might seem to be the crux of this game. But in fact this game can be played in *turns* as well all while keeping intact the spirit of the game. Captain 1 writes down her *choice* on a piece of paper, folds

		Captain 2		
		R	P	S
Captain 1	R	0	-1	1
	P	1	0	-1
	S	-1	1	0

Figure 1.1: Rock, Paper and Scissor in the normal form

it and gives it to the referee. Captain 2 doesn't see what Captain 1 wrote. Next, Captain 2 writes her choice in another piece of paper and gives it to the referee[1]. The referee compares the two choices and declares the winner or a tie. The order in which Captain 1 or 2 submitted their choice is of no importance here. The only thing that matters is that the players do not *observe* each other's *move* before making their own move. This *sequential* version can be accurately captured in the *extensive-form game model* as in Fig. 1.2 which illustrates a different game. In fact any game in strategic form can be modeled in this manner. The arena in an extensive-form game is a tree consisting of three kinds of nodes: triangles, circles and squares corresponding to players Chance, Maximizer (Max) and Minimizer (Min), respectively. The Chance player models probabilistic moves. At nodes corresponding to Maximizer and Minimizer, the respective player makes a choice of the next action. The lack of knowledge about the other player's actions is encoded in the dotted lines at the two-players' nodes. It signifies that these states are indistinguishable to her. Extensive-form games are a classical way to model turn-based sequential games [Osb09]. In this thesis we are concerned with *finite duration* games with *imperfect information* modeled in the extensive-form. In the grand scheme of things our study belongs to the larger field called *game theory*.

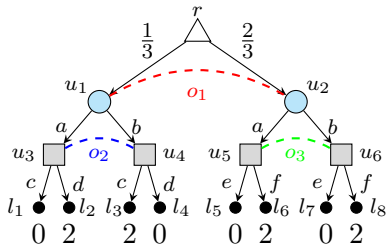


Figure 1.2: Two player game in extensive-form

Game theory is the study of the interaction between several *agents* in an *environment* where agents have their own *objectives*. Modern-day applications of this field are ubiquitous starting from economics, market research, warfare, politics to building robots, AI and verification of automated systems just to name a few.

One of the earliest notable works in game theory was by Zermelo, who proved the *determinacy* of chess [Zer13]. But the advent of game theory as a field is often attributed to Von Neumann for his famous *minimax* theorem in [Neu28] as well as for his subsequent work with Morgenstern which resulted into one of the classic

[1]Captain 2 does not need to fold her paper, since the game ends after her move

books in game theory [NM47]. Many also believe that the field in its modern form would not have existed without Nash's famous theorem on the existence of *Nash Equilibrium* [Nas50]. Following Nash's seminal work, game theory has been a continuously growing field with numerous new models being studied and the emergence of new sub-fields. Our interest lies mostly in the computational aspects of game theory and their *computational complexity*.

Although from a pure computational viewpoint, there is no dearth of inspiring problems in game theory, our study has more specific applications to the advancement of *Artificial Intelligence*. Computation in finite-duration games is a central problem in AI. The primary goal of research in this field is to build AI that can outperform human experts in a particular game.

Games with perfect information where players have complete knowledge of the state of the game at any stage were naturally the first step for this challenge. Chess and Go are popular examples in this class which have been the focus of AI research for a long time. Although games with perfect information can be *solved* efficiently in the size of the game tree using ideas from Zermelo's work [Zer13], in practice, the major difficulty in developing bots for these games is due to the huge size of the state space [Sha50]. The very first success in Chess came in 1996 when an AI named Deep Blue, developed by IBM, beat world chess champion Gary Kasparov [IBM02]. Similar feats have been achieved recently with DeepMinds's AlphaGo [SHMG16] that beat Go champion, Lee Sedol in 2016 and later AlphaZero [SHSA18] which is an improvement of AlphaGo that can play both Chess and Go. With a considerable amount of success in several perfect information games, the next challenge was games with imperfect information such as Poker and Bridge. The imperfect information in these games arises from the fact that a player doesn't have knowledge about the hand of cards dealt to other players. Solving large imperfect information games has turned out to be much more difficult than perfect information games. In regards to specific games, only very recently Poker bots, Libratus [BS17], DeepStack [MSBL17] and Pluribus [BS19] has been able to beat Poker champions in some categories of Poker. Developing bots for various games with imperfect information and proving theoretical guarantees is still an active area of research in game theory.

The primary motivation of this thesis is to look into the complexity of games with imperfect information. Bridge is a specific class of multi-player imperfect information games called team games, where two teams of players have opposite interests, players of the same team have the same payoffs, but players cannot freely communicate, even inside the same team. In games with imperfect information players have *perfect recall* if at any stage of the game they can recall all their previous actions in the play so far. The class of two-player zero-sum game where players have perfect recall can be efficiently solved using linear program-

ming [KM92]. On the other hand, when players have imperfect recall this becomes harder [KM92][CBHL18]. Although in Bridge each individual player has perfect recall, instead of two-player game it is a four player game. Interestingly, dropping the perfect recall assumption in zero-sum two-player games is enough to encompass team games like Bridge: the lack of communication between the players about their private information can be modeled with imperfect recall. Games with imperfect recall can also be used to abstract large perfect recall games and obtain significant computational improvements empirically [SL01][GS07][KS14][BS15][ČLB20]. In this thesis, we address the complexity of *solving* games particularly for this class of imperfect information games.

Maxmin Computation

In game theory it is generally assumed that in a game, players are rational agents. In other words, players can reason about the outcomes of their own decisions and act accordingly. Several suitable concepts for games have been studied that capture the notion of rationality of players [Osb09][Aum74]. One such fundamental solution concept in two-player games is the *maxmin payoff*. Maxmin payoff comes into play in games where the incentive of each player is to maximize the *payoff* she receives at the end. The payoff at the end of the game depends on how players chose their actions. The choice of actions is based on a rule, also called a *strategy*. In a two-player game with players Max and Min, when Max plays strategy σ and Min plays strategy τ, the payoff earned by Max is denoted as $\mathbb{E}[\sigma, \tau]$, and is given by the quantity:

$$\max_{\sigma} \min_{\tau} \mathbb{E}(\sigma, \tau)$$

where σ and τ are strategies of Max and Min respectively. The maxmin payoff is the best possible payoff to Max given Min chooses her strategy after observing the strategy chosen by Max. This is of course with the assumption that Min plays antagonistically against Max i.e. Min wants to reduce Max's gains as much as possible. A strategy of Max that attains the maxmin value is called a *maxmin strategy*. By choosing moves according to a maxmin strategy, Max can always assure a payoff which is equal to the maxmin value.

In general in a game it is possible that two players do not always play antagonistically. However this is the case in *zero-sum* games which are fully competitive games. One can interpret this games as Min having to pay the payoff amount to Max. When Min plays strategy τ and Max plays strategy σ, Min pays the final payoff $\mathbb{E}[\sigma, \tau]$ to Max at the end. Hence the objective of Max is to maximize the payoff, whereas that of Min is to minimize the payoff. Zero-sum games are the first point of study for any model of multiplayer games and have a rich theory [Osb09].

In this thesis we will restrict ourselves to the domain of *two-player zero-sum games in extensive-form with imperfect information.*

As illustrated in Fig. 1.2 extensive-form representation encapsulates turn-based games with the aid of trees where each tree node encodes a state of the game. The game tree unfolds from the start state or *root* node to the terminal states or *leaves* depending on the players' decision at their nodes. Some nodes in the game belong to neither players and represent nature or *chance* events. Since players have imperfect information, at each node the players receive a signal or *observation* which encodes their knowledge of the game-play up to that point. For two nodes with the same observations, intrinsically the player cannot distinguish one from another when they reach either of them. Extensive-form model presents the opportunity to the players to play strategies where decisions can be made locally after each observation as the game progresses.

A strategy where players randomize from the available actions after each observation is called a *behavioral strategy*. The special kind of behavioral strategy where players chose every action deterministically is called a *pure* strategy. Behavioral strategies are quite practical, adaptable and capture very well the sequential decision making process true to the nature of extensive-form model as opposed to *mixed strategies*.

Mixed strategies are strategies where players instead of choosing locally, randomize globally over the set of pure strategies before the game starts. Since the number of pure strategies can be exponential in the size of the game, a mixed strategy can possibly have exponential support. On the other hand, the size of the support of a behavioral strategy is always small since it is bounded by the number of actions in the game. Besides, as pointed out in [PR97], mixed strategies can potentially provide more information to the player than intended in the rules of the game. This issue never comes up in a behavioral strategy. So in this thesis we are particularly interested in the maxmin computation over behavioral strategies.

Several algorithms have been proposed than can compute the maxmin value either exactly or approximately for various classes of extensive-form games [BML93] [KM92] [Ste96] [ZJBP07] [CBL17]. The question of complexity comes up naturally to understand the feasibility of these algorithms in general. We are interested in the complexity of the decision problem concerning computation of the maxmin value.

Complexity

Towards the goal of studying the computational complexity of computing the maxmin value of a game we consider the following decision problem.

Decision Problem 1.1 (Two Player)**.** Given a two-player zero-sum imperfect

information game G in extensive-form, is the maxmin value of G over behavioral strategies non-negative?

Being a fundamental decision problem, the complexity of this problem is well studied [KM92][Ste96][HMS07][BP17] . Koller and Meggido showed in their classic work that the maxmin decision problem can be solved in PTIME when players have a kind of memory called *perfect recall* [KM92].

A player has perfect recall when at all instances in the game she never forgets the sequence of actions that she has played previously. In this case they formulated the problem as a linear program by successfully encoding the space of behavioral strategies of a perfect recall player into a linear system called *realization plan*. This was reformulated by Von Stengel using sequence form representation [Ste96]. In [KM92], the authors also initiated the complexity theoretic study of the general maxmin problem. They showed that by lifting the perfect recall assumption, for general extensive-form games the maxmin decision problem becomes NP-hard. In addition to that they dismiss the possibility of using a maxmin strategy as a short certificate for membership in NP by constructing games where irrational probabilities are necessary in a maxmin strategy.

An important component of maxmin computation is the *best response* computation which is the optimal strategy of Min (Max) against a fixed strategy of Max (Min). This problem can be rephrased as finding the optimal value and the optimal strategy in a one-player game. Hence we also consider the corresponding decision problem for one-player.

Decision Problem 1.2 (One player)**.** Given a one-player imperfect information game G in extensive-form, is the maximum value in G over all behavioral strategies non-negative?

When the player is not *absentminded*, the one-player decision problem is known to be NP-complete [BML93]. Absentmindedness is a class of imperfect recall most notably discussed in [PR97] with the absentminded driver example. A player is absentminded if at any moment on observing a signal, she cannot remember if she has observed that exact same signal before. In other words it is possible that she observes the same signal at two nodes lying on the same path from root to a leaf.

When the player is not absentminded, the inclusion of the one-player problem in NP owes to the fact that pure strategies are sufficient for achieving optimality in such games. This is not the case when players are absentminded. In the maxmin computation using linear program for two-player case with perfect recall, implicitly there is an efficient sub-routine of best response computation. It follows that the problem is in P for a perfect recall player. It turns out the best response computation

sub-routine can be extended efficiently to a sub-class of non-absentminded games that subsumes perfect recall called *A-loss recall* [KK95] [Kli02].

A player has A-loss recall (action-loss recall) when any loss of her information about her own actions can be traced back to a point in the past where she forgets what action she played after observing some signal. In other words, she either (i) never forgets her actions (i.e. has perfect recall) or (ii) she remembers her action up to some point after which she could have potentially played one among several possible actions. In Poker if a player suddenly forgets the initial cards dealt to her, she doesn't have A-loss recall because she forgot a 'piece of information' that she had gained even before she started playing. It can be shown that when a player with A-loss recall plays a fixed strategy, she can deduce her exact past history of actions. As a consequence of the fact that best response sub-routine for A-loss recall is PTIME, the maxmin computation for perfect recall vs A-loss recall is also in PTIME [BP17]. However it was also shown in [CBHL18] that as soon as Max has A-loss recall this problem is NP-hard even when Min has perfect recall. As for games with absentmindedness, for the one-player case it was shown that strategies with irrational probabilities might be necessary for achieving optimality [HMS07]. Essentially the hardness of the maxmin decision problem can arise from the kind of memory recall of either of the player Max or Min.

Complexity theoretic study of decision problems arsing from games has been carried out on several occasions before. Majority of them concerns computation of *Nash equilibrium* [CD06][DGP09][DGP09][HMS10] [EY10] [GMVY15] [SS17] [BM16][BM21][BH22a][BH22b]. Another line of research in the complexity of games is related to finding winning strategies in various *combinatorial games* (see [Dem01] for a survey).

1.1 Our contribution

1.1.1 Complexity bounds

In this thesis we present a clearer picture of the complexity of the maxmin decision problem for games with players having imperfect recall. No lower bounds for this problem were known to exist for the specific case when players are absentminded.

We give complexity bounds both for absentminded and non-absentminded games.

For our lower-bound results involving absentminded players we use complexity classes arising from fragments of *First Order Theory of Reals* ($\mathrm{FOT}(\mathbb{R})$) namely $\exists\mathbb{R}$, $\forall\mathbb{R}$ and $\exists\forall\mathbb{R}$.

$\exists\mathbb{R}$ and $\forall\mathbb{R}$

A formula in $\mathrm{FOT}(\mathbb{R})$ is a logical statement containing Boolean connectives $\vee$, $\wedge$, $\neg$ and quantifiers $\exists, \forall$ over the signature $(0, 1, +, *, \leq, <, =)$. We can consider it to be a first order logic formula in which each atomic term is a polynomial equation or inequation, for instance $\exists x_1, x_2 \forall y (0 \leq y \leq 1) \rightarrow (4x_1y + 5x_2^2y + 3x_1^3x_2 > 4)$ (where integers have been used freely since they can be eliminated without a significant blow-up in the size of the formula [SS17], and the implication operator $\rightarrow$ with the usual meaning). The complexity class $\exists\mathbb{R}$ is the set of problems which have a polynomial-time many-one reduction to a sentence of the form $\exists X \Phi(X)$ where X is a tuple of real variables, $\Phi(X)$ is a quantifier free formula in the theory of reals. Similarly, the complexity classes $\forall\mathbb{R}$ and $\exists\forall\mathbb{R}$ stand for the problems that reduce to formulae of the form $\forall X \Phi(X)$ and $\exists X \forall Y \Phi(X, Y)$ where X, Y are tuples of variables.

All these complexity classes $\exists\mathbb{R}$, $\forall\mathbb{R}$ and $\exists\forall\mathbb{R}$ are known to be contained in PSPACE [Can88]([BPR06] see Remark 13.11). Complexity of games with respect to the class $\exists\mathbb{R}$ has been studied before particularly for decision problems related to *equilibrium* computation [GMVY15] [SS17] [BM16][BM21][BH22a][BH22b].

For non-absentminded games we show co-NP hardness even for the case when Max player is a trivial or dummy player and Min is non-absentminded. We also show lower bound with respect to the Square-Root-Sum problem.

Square-Root-Sum

The Square-Root-Sum decision problem is the following: Given k positive integers $a_1, \ldots, a_k$ and another positive integer n, is $\sum_i \sqrt{a_i} \geq n$?

This problem often arises while studying complexity of numerical problems and was first put forward in [EY05]. There have been studies in game complexity that use the Square-Root-Sum problem as a lower bound [EY10] [HMS10]. In our case this problem finds its place because it has been shown that in maxmin optimal strategies, irrational probabilities are necessary[KM92]. For games where Max has A-loss recall and Min has perfect recall we show that the maxmin problem is Square-Root-Sum-hard. This settles the open question asked in [CBHL18]. Since the Square-Root-Sum problem is not believed to be in NP, the Square-Root-Sum lower bound and the co-NP lower bound puts more weight on the belief that the maxmin problem is not in NP.

1.1.2 New tractable class of imperfect recall

On a positive note we identify a class of one-player non-absentminded games called *A-loss recall shuffle* for which it is possible to use the linear program to compute optimal payoff in PTIME. A player has A-loss recall shuffle, when the

action histories of this player can be rearranged in order to achieve A-loss recall. We demonstrate that games with A-loss recall shuffle are *equivalent* to an A-loss recall game of linear size. This class extends the class of games previously known to be solvable in PTIME. Using this we can compute the maxmin value in PTIME for the case when Max has perfect recall and Min has A-loss recall shuffle.

We also provide a generalization of this class called *A-loss recall span*. An A-loss recall span of a non-absentminded game is a game with A-loss recall over the same set of actions that is *equivalent* to the actual game. We prove that any non absentminded game is equivalent to a game with A-loss recall with the caveat that the size of the A-loss recall game may blow up exponentially in the worst case. As a consequence, we obtain an algorithm to compute the optimal value in a one-player games in time polynomially bounded in size of its smallest equivalent A-loss recall game. Similar to A-loss recall shuffle, we extend this technique to the two-player case where Max has perfect recall.

Study of equivalence of games in extensive-form has been done before ([Osb09] see Section 11.2), but that equivalence was based on similar strategic situations in the games. Our notion of equivalence comes purely from a computational perspective since the final playoff plays a major role in it.

1.1.3 Study of Bridge Bidding

In the long tradition of building bots for games in AI and game theory research, Bridge is no exception [Gin99][Gin01](see [Bet21] for a recent survey). Bridge is a 4 player team game (2 in each team) with imperfect information which consists of two phases: bidding phase and playing phase. Bridge is one of the games without considerable success in beating human experts as compared to other popular games such as Chess, Go, etc. Only very recently in 2022 the Bridge bot of Nukkai beat 8 bridge champions in a restricted version of bridge with no bidding involved [Gua22]. Current programs use the *Double Dummy Analysis* (where perfect information is assumed) for evaluating the playing phase. However, current research in Bridge is deprived of a theoretical analysis of the bidding phase as well as the hardness of computing good bidding strategies.

In this thesis we propose a model to study Bridge bidding and report some preliminary results on the computation of the maxmin value in this model. Even though Bridge is a team game with 4 players, as a first step we focus on a simpler version of this model. We consider the two-player zero-sum version of the game where cards can only have a single *suit* (as opposed to 4). We place various restrictions on the strategies such as (i) pure strategies and (ii) bounded *overbidding* strategies : strategies where players cannot bid more than a fixed number of times. We investigate what kind of strategies are sufficient for maxmin optimality.

We construct examples where these restricted classes of strategies are not strategically powerful enough to guarantee the maxmin. We also show that the complexity of computing maxmin value depends on the number of *overbids* necessary for a player for maxmin optimality. We provide a bound on the number of bids necessary for maxmin optimality with respect to the *beliefs* of the players.

1.2 Organization of Thesis

The rest of the thesis is organized into five chapters.

In Chapter 2 we discuss preliminaries on extensive-form games where we formally define the game model and related terminologies.

In Chapter 3 we present the state-of-the-art on the complexity of the maxmin decision problem and then dive into our complexity results.

In Chapter 4 we present our results on transformation of games into equivalent A-loss recall games and its implications.

In Chapter 5 we present our Bridge Bidding model and discuss some preliminary results on the maxmin problem in this model.

In Chapter 6 we summarize our contributions and state some future directions of research.

We begin each chapter with a short overview of the chapter content and in each of the Chapters 3,4 and 5 we end the chapter with a conclusion where we provide a quick recap, raise relevant open questions and touch upon future research directions.

Chapter 2

Preliminaries: Extensive Form Games

In this chapter, we present extensive-form games and various relevant notions. First, we define two-player zero-sum extensive-form games of finite duration with imperfect information. Then we talk about different kinds of strategies and the notion of payoffs under these strategies. In games with imperfect information, a player might forget information during the course of the game which induces different kinds of memory recalls on players. We define these recalls and discuss equivalences between different kinds of strategies. We define the primary concept of this thesis, the maxmin value of a game and finally end by recalling the linear program to compute the maxmin value for a class of games.

Contents

2.1 Extensive-form games with imperfect information

An extensive-form game has a directed tree structure which is defined as follows:

Definition 1 (Directed Tree)**.** A *directed tree* is a finite directed rooted graph given by a tuple $\mathcal{T} = (V, L, E, r)$ where

- V is the set of nodes
- $E \subseteq V \times V$ is the set of directed *edges* such that the underlying undirected graph (V, E) is a tree.
- $r \in V$ is the unique node with no predecessor, called the *root* of $\mathcal{T}$ i.e. $\forall (u, v) \in E, v \neq r$ and $\forall v \in V \setminus \{r\}\ \exists u,\ (u, v) \in E$.
- L is the set of nodes with no successor, called the *leaves* of $\mathcal{T}$.

  $(u, v) \in E \implies u \notin L$

Since the underlying graph is a tree, $\forall v \in V \setminus \{r\}$ there is a unique path from the node r to v. We denote the sequence of vertices on this path as $\mathrm{PathTo}(v)$. For a node v and another node u on $\mathrm{PathTo}(v)$, $\mathrm{Path}(u, v)$ is the sequence of vertices on the path from u to v. $\mathrm{Path}(u, u)$ is the trivial empty path with only node u. We write $u \to v$ if $(u, v) \in E$.

Now we define two-player zero-sum extensive-form games with imperfect information played by players Max and Min.

Definition 2 (Two-player zero-sum extensive-form games)**.** A two-player zero-sum extensive-form game with imperfect information with a set of players $\mathcal{N} = \{\mathsf{Max}, \mathsf{Min}\}$ and with a special agent Chance is given by a tuple $(\mathcal{T}, A, \{V_i\}_{i \in \mathcal{N} \cup \{\mathsf{Chance}\}}, \{A_i\}_{i \in \mathcal{N} \cup \{\mathsf{Chance}\}}, \delta_C, \delta_A, \{\mathcal{O}_i, \mathrm{Obs}_i\}_{i \in \mathcal{N}}, \mathcal{U})$ where

- $\mathcal{T} = (V, L, E, r)$ is a directed tree and $\{V_i\}_{i \in \mathcal{N} \cup \{\mathsf{Chance}\}}$ is a partition of V. Nodes in V_{Chance} are called *chance nodes* while other nodes are called *control nodes*.
- $\mathcal{O}_i$ is the set of observations of player i and $\mathrm{Obs}_i : V_i \to \mathcal{O}_i$ maps every node of player i to an observation.
- A is the set of *actions* partitioned as $(A_i)_{i \in \mathcal{N} \cup \{\mathsf{Chance}\}}$.
- $\delta_A : E \to A$ is the transition label function which labels each edge $u \to v$ with an action $a \in A$, in which case we write $u \xrightarrow{a} v$. Actions on edges out of $u \in V_i$ belong to A_i. No two outgoing edges from the same control node are labeled with the same action. For every node u, we denote the set

of actions in u by $\mathrm{Act}(u) = \{a \in A \mid u \xrightarrow{a} v \text{ for some } v \in V\}$. Also, for every observation $o \in \mathcal{O}_i$, all nodes with observation o have the same set of actions which we denote by $\mathrm{Act}(o)$.

- $\delta_C : E \cap (V_{\mathsf{Chance}} \times V) \to [0, 1]$ associates a transition probability to every edge (u, v) such that u is a chance node. The transition probabilities are rational and they sum up to 1 from a given chance node.
- The payoff function $\mathcal{U} : L \mapsto \mathbb{R}$ associates to each leaf of $\mathcal{T}$ a rational number called the *terminal payoff*.

Example 2.1. Fig. 2.1 is a representation of a game in extensive-form. Max, Min, and Chance nodes are denoted by circle, square, and triangle nodes respectively. Black circular nodes represent the leaf nodes and the value next to them denotes the terminal payoffs. In this example, $V_{\mathsf{Max}} = \{u_1, u_2\}, V_{\mathsf{Min}} = \{u_3, u_4, u_5, u_6\}$, $V_{\mathsf{Chance}} = \{r\}$ and $L = \{l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8\}$. $\mathcal{U}(l_i) = 2$ for $i \in \{2, 3, 6, 8\}$ and 0 otherwise.

Any two nodes of the same player connected by a dotted line signify that the player has the same observation at these two nodes. Edges out of control nodes are labeled by the actions of the controlling player. In the example $\mathcal{O}_{\mathsf{Max}} = \{o_1\}$ and $\mathcal{O}_{\mathsf{Min}} = \{o_2, o_3\}$. $\mathrm{Obs}_{\mathsf{Max}}(u_1) = \mathrm{Obs}_{\mathsf{Max}}(u_2) = o_1$, $\mathrm{Obs}_{\mathsf{Min}}(u_3) = \mathrm{Obs}_{\mathsf{Min}}(u_4) = o_2$ and $\mathrm{Obs}_{\mathsf{Min}}(u_5) = \mathrm{Obs}_{\mathsf{Min}}(u_6) = o_3$. Notice that nodes having the same observation have the same set of labels on outgoing edges. Edges out of chance nodes are labeled by corresponding probabilities assigned by the function δ_C. $\delta_C(r \to u_1) = \frac{1}{3}$ and $\delta_C(r \to u_2) = \frac{2}{3}$.

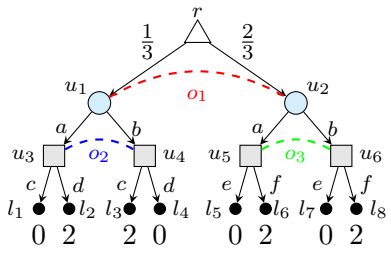


Figure 2.1: Two player game in extensive-form

A finite extensive-form game is the representation of a turn-based game with a finite number of states. Each node of the tree represents a state of the game. To start the game a pebble is placed on the root node r representing the start state. When the pebble is in a particular node it moves to a successor node depending on

the type of its current node. At a chance node, the successor is chosen according to the probability distribution δ_C. At a control node, the successor is chosen by the player controlling the node according to a *strategy*.

When the pebble is in some control node v, the controlling player i's knowledge about the position of the pebble (hence the situation in the game) is encoded by the observation $\mathrm{Obs}_i(v)$. So when player i observes $o \in \mathcal{O}_i$, she knows that the pebble is in one of the nodes in $\mathrm{Obs}_i^{-1}(o)$, the set of nodes with observation o, but doesn't know in which one exactly. The sets $\mathrm{Obs}_i^{-1}(o)$ are called *information sets* of player i. An extensive-form game with *perfect information* is a special situation where every player observes the node itself. This essentially means for every i and $o \in \mathcal{O}_i$, $|\mathrm{Obs}_i^{-1}(o)| = 1$. For some observation o when $\mathrm{Obs}_i^{-1}(o) = \{u\}$ we often abuse the notation and use u as the observation instead of o.

When the pebble reaches a leaf node $t \in L$ player Min has to pay to player Max the payoff $\mathcal{U}(t)$. The two-players have opposite goals: player Max wants to maximize this payoff whereas player Min wants to minimize it.

Next, we will see how players play according to a strategy.

2.2 Three kinds of strategies

A *strategy* is a policy or a rule utilized by a player to decide the successor whenever the game is in a control state. Strategies can be classified based on whether they are deterministic or randomized.

Definition 3 (Behavioral strategy)**.** A *behavioral strategy* σ for player $i \in \{\mathsf{Max}, \mathsf{Min}\}$ is a function $\sigma : \mathcal{O}_i \mapsto \Delta(A_i)$ which maps every observation o to a probability distribution over $\mathrm{Act}(o)$. When player i has observation o, she plays an action $a \in \mathrm{Act}(o)$ with probability $\sigma(o)(a)$, also denoted $\sigma(o, a)$. For every node u controlled by i, we write $\sigma(u, a)$ for the probability $\sigma(\mathrm{Obs}_i(u), a)$.

Definition 4 (Pure Strategy)**.** A *pure strategy* σ is a behavioral strategy that plays every action with probability either 0 or 1. We identify every action a with the Dirac distribution on a, and $\sigma(o)$ denotes the action played in observation o.

Let Σ_i be the set of all pure strategies of player i.

Definition 5 (Mixed Strategy)**.** A *mixed strategy* α is a distribution over Σ. For $\rho \in \Sigma$, $\alpha(\rho)$ is the probability of playing ρ.

A *strategy profile* in a game is a pair of strategies (σ, τ) where σ is a strategy of player Max and τ is a strategy of player Min.

A pure strategy can be seen as a special case of both behavioral or mixed strategies. On the other hand in general there is no such relation between mixed

and behavioral strategies. However, we will see later that these two kinds of strategies are in fact *equivalent* for a player with a kind of memory called perfect recall [Kuh53]. Moreover, for a larger class of memory recall when a player is not absentminded it turns out that every behavioral strategy is equivalent to a mixed strategy.

Next, we will see what is the payoff obtained when two-players play certain strategies.

2.3 Expected Payoff

According to the semantics when the game reaches a leaf node t, Min pays the terminal payoff $\mathcal{U}(t)$ to Max. The chance nodes and the randomization in strategies induce a probability distribution of reaching the leaves from the root, hence we deal with expected payoffs. However, this probability computation depends on the kind of strategy involved. The probability of reaching a leaf can be decomposed into the contributions of players' individual strategies as well as the chance nodes.

For any node $w \in V$, let

$$\mathcal{P}_{\mathsf{Chance}}(w) = \prod_{u \in V_{\mathsf{Chance}},\ u \to v \in \mathrm{PathTo}(w)} \delta_C(u \to v)$$

denote the product of probabilities along the edges controlled by Chance in $\mathrm{PathTo}(w)$.

For a behavioral strategy σ of player i, let $\mathcal{P}^i_\sigma(w)$ given by

$$\mathcal{P}^i_\sigma(w) = \prod_{u \in V_i,\ u \xrightarrow{a} v \in \mathrm{PathTo}(w)} \sigma(u, a)$$

denote the product of probabilities assigned by σ at nodes in V_i to edges on $\mathrm{PathTo}(w)$.

For a mixed strategy α of player i

$$\mathcal{P}^i_\alpha(w) = \sum_{\rho \in \Sigma_i \mid \mathrm{PathTo}(w) \text{ is realizable under } \rho} \alpha(\rho)$$

Finally the probability of reaching w under a strategy profile (σ, τ) is given by

$$\mathcal{P}_{\sigma,\tau}(w) = \mathcal{P}_{\mathsf{Chance}}(w)\mathcal{P}_\sigma(w)^{\mathsf{Max}}\mathcal{P}_\tau(w)^{\mathsf{Min}}$$

The *expected payoff* denoted by $\mathbb{E}(\sigma, \tau)$ under a strategy profile σ, τ is given by

$$\mathbb{E}(\sigma, \tau) = \sum_{t \in L} \mathcal{P}_{\sigma,\tau}(t)\mathcal{U}(t).$$

Whenever the player is clear from the context, for the sake of brevity, we write $\mathcal{P}_\sigma(w)$ instead of $\mathcal{P}^i_\sigma(w)$. Further, when σ is also clear from the context, we write $\mathcal{P}(w)$.

Example 2.2. In Fig. 2.1, consider the behavioral strategy of Max σ given by $\frac{1}{2}a + \frac{1}{2}b$ and the mixed strategy of Min τ given by $\frac{1}{2}\{c, f\} + \frac{1}{2}\{d, f\}$.

For terminal node l_8, $\mathcal{P}_{\mathsf{Chance}}(l_8) = \frac{2}{3}, \mathcal{P}^{\mathsf{Max}}_\sigma(l_8) = \frac{1}{2}, \mathcal{P}^{\mathsf{Min}}_\tau(l_8) = 1$, therefore $\mathcal{P}_{\sigma,\tau}(l_8) = \frac{1}{3}$. Again for terminal node l_3, $\mathcal{P}_{\mathsf{Chance}}(l_3) = \frac{1}{3}, \mathcal{P}^{\mathsf{Max}}_\sigma(l_3) = \frac{1}{2}, \mathcal{P}^{\mathsf{Min}}_\tau(l_3) = \frac{1}{2}$, therefore $\mathcal{P}_{\sigma,\tau}(l_3) = \frac{1}{12}$. With similar computation for other terminal nodes, it can be seen that $\mathbb{E}(\sigma, \tau) = \frac{5}{3}$.

The objective of Max is to maximize the expected payoff whereas that of Min is to minimize it. So each player has an incentive to play according to a strategy aligning with this objective. In the next section, we will see the notions of *best response* and *maxmin* value that capture the optimal behavior of a rational player.

2.4 Best Response and Maxmin Value

In this thesis, we study the optimal behavior of players over the space of behavioral strategies. In fact it can be argued that the choice of behavioral strategies is more natural than mixed strategies. This is because, as observed by Kuhn in [Kuh53] mixed strategies can sometimes contradict the amount of information revealed by some observation, i.e. they may provide more information to a player than dictated by an observation. For example, in Fig. 2.2b when Max plays the mixed strategy $\frac{1}{2}\{a, c\} + \frac{1}{2}\{b, d\}$, when Max observes o_1 she is supposed to forget which action she played previously. But Max can effectively infer her last action from her strategy since $\{a, c\}$ and $\{b, d\}$ are perfectly correlated in the mixed strategy. On the other hand, in behavioral strategy randomizations are done locally and independently at every observation.

Besides, mixed strategy representation can possibly need exponential size support whereas the size of the support of a behavioral strategy is the same as the number of distinct actions of the player.

Also, recall that pure strategies are a sub-class of behavioral strategies.

2.4.1 Best response to a strategy

In a game G, fix a strategy σ of Max. The best response value of Min against σ is the minimum payoff to Max over all strategies of Min when Max plays σ. Formally:

Definition 6 (Best response)**.** The best response value of Min to strategy σ of Max, denoted by $\mathrm{BR}_{\mathsf{Min}}(\sigma)$ is defined as:

$$\mathrm{BR}_{\mathsf{Min}}(\sigma) = \inf_{\tau} \mathbb{E}(\sigma, \tau)$$

Similarly the best response value of Max against a strategy τ of Min can be defined as follows.

$$\mathrm{BR}_{\mathsf{Max}}(\tau) = \sup_{\sigma} \mathbb{E}(\sigma, \tau)$$

Best response computation as a one-player game
In a two-player game G, fixing a strategy τ of Min produces a one-player game G_τ with only Max player. The control nodes of Min in G act as chance nodes in G_τ and the chance edge probabilities out of these nodes are given by τ. The value $\mathrm{BR}_{\mathsf{Max}}(\tau)$ can be interpreted as the optimal payoff obtained by Max in the game G_τ. As a result, very often the problem of finding the best response against a behavioral strategy is reformulated as computing the optimal payoff in one-player games.

A strategy that attains the best response payoff is called a *best response strategy*.

2.4.2 Maxmin value

The maxmin value of a game is the optimal payoff to Max when she reveals her strategy ex-ante to Min. Effectively Max maximizes the payoff assuming Min responds with her best-response strategy.

Definition 7. The maxmin value is defined as:

$$\mathrm{MaxMin}_{\mathrm{beh}}(G) = \sup_{\sigma} \mathrm{BR}_{\mathsf{Min}}(\sigma) = \sup_{\sigma} \inf_{\tau} \mathbb{E}(\sigma, \tau)$$

where σ and τ are behavioral strategies[1] of Max and Min respectively.

The maxmin value is the greatest value Max can always be assured of achieving irrespective of how Min plays. Since the space of strategies as well the set of all possible payoffs are compact sets, and limit points of a compact set lie inside the set, there exists at least one strategy of Max which provides the supremum value. Hence the sup and inf in the definition can very well be replaced by max and min giving us $\mathrm{MaxMin}_{\mathrm{beh}}(G) = \max_{\sigma} \min_{\tau} \mathbb{E}(\sigma, \tau)$.

[1] Maxmin value can also be defined similarly over pure and mixed strategies by modifying the space of strategies considered in the definition

A *maxmin strategy* of Max is a strategy that gives the maxmin payoff given by $\arg\max_{\sigma} \mathrm{BR}_{\mathsf{Min}}(\sigma)$.

The efficiency with which one can compute the best response and maxmin value in a game depends on the memory of players. In the next section, we will see the notion of history and different kinds of recalls of players.

2.5 Histories and Recalls

A node $v \in V$ is reached by the unique $\mathrm{PathTo}(v)$ from the root. The history at v of a player is the sequence of observations and the actions played by the player on this path. Chance nodes and chance edges are not taken into account in the history of a node.

Definition 8 (Histories)**.** For a vertex $w \in V$ the history at w denoted by $\mathit{hist}(w)$ is the sequence of actions a where $u \xrightarrow{a} v$ is an edge on $\mathrm{PathTo}(w)$ with $u \notin V_{\mathsf{Chance}}$. For a player i the history of i at w denoted by $\mathit{hist}_i(w)$ is the sequence of player i's actions on $\mathrm{PathTo}(w)$ which is the sub-sequence of $\mathit{hist}(w)$ restricted to actions from A_i.

For the root node r, $\mathit{hist}(r) = \epsilon$ is the empty history. Let $\mathcal{H}$ denote the set of all histories and $\mathcal{H}_i$ be the set of all histories of player i. For an observation $o \in \mathcal{O}_i$ let $\mathcal{H}(o) = \{\mathit{hist}(u) \mid \mathrm{Obs}_i(u) = o\}$ be the set of histories of all nodes with observation o. $\mathcal{H}_i(o)$ is defined similarly with respect to $\mathcal{H}_i$.

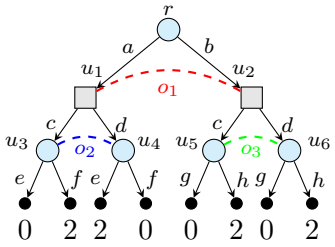
For any player i, there may be multiple nodes with different observations that have the same history $h \in \mathcal{H}_i$. For a history h, let $\mathcal{O}_i^h = \{o \in \mathcal{O}_i \mid h \in \mathcal{H}(o)\}$ be the set of all observations of player i in which there exists a node with history h.

When $\mathcal{H}_i(o)$ has multiple histories, at a node v with $\mathrm{Obs}_i(v) = o$ the player doesn't remember which history she traversed to reach v. Hence the player loses information. So at a control node $v \in V_i$, the observation $\mathrm{Obs}_i(v)$ encodes the information that player i possesses at the point the game reaches node v. For two nodes u and v in V_i with the same observation, comparing $\mathit{hist}_i(u)$ and $\mathit{hist}_i(v)$ reveals the loss or retention of previously withheld information at the respective nodes. To capture this we have different notions of *recall*.

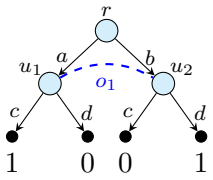
A player has *perfect recall* when she retains all her past observations and doesn't undergo any loss of information.

Definition 9 (Perfect recall)**.** Player i is said to have perfect recall if for every $u, v \in V_i$, if $\mathrm{Obs}_i(u) = \mathrm{Obs}_i(v)$ then $\mathit{hist}_i(u) = \mathit{hist}_i(v)$. Otherwise, the player is said to have imperfect recall.

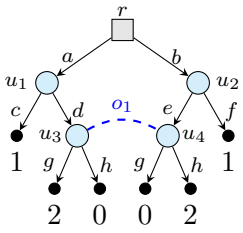
Example 2.3. In Fig. 2.2a, Max has perfect recall.



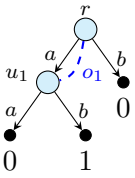
(a) Max with perfect recall



(b) Max without perfect recall but A-loss recall



(c) Max without A-loss recall



(d) Max with absentmindedness

Figure 2.2: Different kinds of recall possible for Max

For a player with perfect recall, every vertex in an information set has the same history with respect to that player. Effectively player i has perfect recall iff $\forall o \in \mathcal{O}_i \; |\mathcal{H}_i(o)| = 1$. It is to be noted that perfect recall still doesn't mean perfect information. Max doesn't have perfect information in Fig. 2.2a. Perfect recall talks about remembering past observations whereas a player with perfect information not only retains information but also observes the exact node at every turn.

Next, we will see a class of imperfect recall called absentmindedness [PR97].

Definition 10 (Absentminded)**.** Player i is said to have *absentmindedness* if there are two distinct nodes $u, v \in V_i$ with $\text{Obs}_i(u) = \text{Obs}_i(v)$ such that u lies in the unique path from root to v i.e. $\mathit{hist}_i(u)$ is a proper prefix of $\mathit{hist}_i(v)$ (player i at node u with $\text{Obs}_i(u) = o$ cannot recall if she has observed o before, i.e. passed via a node with observation o before.) .

Example 2.4. In Fig. 2.2d Max is absentminded since she observes o_1 both at r and u_1. Notice that Max can never reach the leaf with payoff 1 by playing a pure strategy since she is forced to choose the same action at both r and u_1.

Remark 2.1. Absentmindedness is even stronger than saying that a player forgets the length of her own action history. In fact, a non-absentminded player can still forget how many actions she has played. For example, from Fig. 2.2c imagine a game with the same underlying game tree, but with nodes u_1 and u_4 having the

same observation. Max would be non-absentminded in this new game. But Max forgets the length of her history. Because if Max could count the number of actions she played, she could have distinguished between u_1 and u_4.

Next, we see a special class of games with imperfect recall called A-loss recall. In this kind of recall any loss of information can be traced back to a point in the game from where the player chose distinct actions.

Definition 11 (A-loss recall)**.** Player i has *A-loss* recall if for every $u, v \in V_i$ with $\mathrm{Obs}_i(u) = \mathrm{Obs}_i(v)$ either $\mathit{hist}_i(u) = \mathit{hist}_i(v)$ or $\mathit{hist}_i(u)$ is of the form sas_1 and $\mathit{hist}_i(v)$ of the form sbs_2 where $a, b \in \mathrm{Act}(o)$ for some $o \in \mathcal{O}_i$ with $a \neq b$.

A player with A-loss recall does not necessarily have perfect recall, but when she fixes a pure strategy σ, she regains her perfect recall in the following sense: for every observation $o \in \mathcal{O}_i$, all plays consistent with σ and ending with observation o have the same history with respect to i. This is because σ either plays a or b at o and hence both the nodes u and v are not reachable by playing σ. In other words, in an A-loss recall game, a player using a certain pure strategy can infer from her current observation the history of the node she is in.

Example 2.5. In Fig. 2.2b Max has A-loss recall since $\mathit{hist}_{\mathsf{Max}}(u_1) = a$ and $\mathit{hist}_{\mathsf{Max}}(u_2) = b$. At o_1 Max can reach exactly one node between u_1 or u_2 via a pure strategy. On the other hand in *Fig. 2.2c* Max doesn't have A-loss recall since at o_1, $\mathit{hist}_{\mathsf{Max}}(u_3) = d$ and $\mathit{hist}_{\mathsf{Max}}(u_4) = e$. So if Max plays the pure strategy $\{d, e, g\}$ and observes o_1, she cannot exactly tell which action between d and e she played previously.

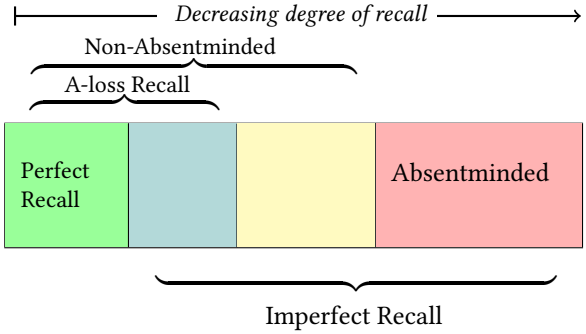


Figure 2.3: Relation between different recalls

Relation between recalls

The relation between different kinds of recalls is depicted in Fig. 2.3 as a spectrum. As we go from left to right the recall of the player weakens in some sense. The

diagram represents the set of all extensive-form games with imperfect information. Perfect Recall is a special class of A-loss recall. Both A-loss recall and perfect recall are sub-classes of non-absentminded games. Absentminded games are a special class of imperfect recall games.

Next, we see the hierarchy between strategies in the light of different recalls.

2.6 Recalls and equivalence of strategies

Definition 12. Two strategies σ_1 and σ_2 of Max are said to be equivalent if for all strategies τ of Min, $\mathbb{E}(\sigma_1, \tau) = \mathbb{E}(\sigma_2, \tau)$. Similarly for Min, two strategies τ_1 and τ_2 are equivalent if for all strategies σ of Max, $\mathbb{E}(\sigma, \tau_1) = \mathbb{E}(\sigma, \tau_2)$.

Essentially two strategies σ_1 and σ_2 of Max are equivalent if her payoff doesn't change by switching from one to the other against a fixed strategy of Min.

Equivalence of two strategies in extensive-form games can also be characterized using the probability distribution induced on the leaf nodes based on the following observation from ([KM96] Lemma 2.5).

Proposition 2.2. [KM96] Two strategies σ_1 and σ_2 of Max (Min) are *equivalent* iff $\forall t \in L, \mathcal{P}^{\text{Max}}_{\sigma_1}(t) = \mathcal{P}^{\text{Max}}_{\sigma_2}(t)$.

Equivalent strategies induce an equivalence relation over the space of all strategies. Hence for optimality Max (Min) can choose any strategy from an equivalent class of strategies.

Depending on the type of recall it is possible that for every strategy of one class, there is an equivalent strategy of another class. This can create a hierarchy between different kinds of strategies with respect to optimality. When a player is not absentminded mixed strategies are as expressive as behavioral strategies, i.e., for every behavioral strategy there is an equivalent mixed strategy.

Proposition 2.3. [Kuh53] In a game G if player i is not absentminded, for every behavioral strategy β of i, i has an equivalent mixed strategy σ.

So when a player is not absentminded, strategically, mixed strategies are at least as good as behavioral strategies. But the size of a mixed strategy can possibly be exponential in the size of the game tree due to a large support size. This can be costly for computations over the whole space of mixed strategies. Even though Koller have shown in [KM96] that every mixed strategy is equivalent to a mixed strategy over a small support[2] but with a possible blowup in the bit complexities of

[2]The small supports of equivalent mixed strategies of two different mixed strategies might be distinct.

the probabilities. On the other hand, the size of behavioral strategies is polynomial in the size of the tree assuming the assigned probabilities are also polynomially bounded.

The natural question to ask is if behavioral strategies are also as expressive as mixed strategies. For non-absentminded games, the answer is no as demonstrated in the following example.

Example 2.6. In Fig. 2.2b for the mixed strategy σ given by $\frac{1}{2}\{a, c\} + \frac{1}{2}\{b, d\}$ of Max there is no equivalent behavioral strategy. This is because σ induces non-zero probabilities at the terminal nodes at the end of paths ac and bd and zero to the others. On the other hand for any behavioral strategy to imitate this, it has to play all actions with non-zero probability. But any behavioral strategy of this kind would end up inducing a non-zero probability to the leaf at ad as opposed to zero induced by σ.

However, this changes when the player has perfect recall.

Theorem 2.4. *[Kuh53] In a game G if player i has perfect recall, then for every mixed strategy σ of player i there exists a behavioral strategy β of player i such that σ and β are equivalent.*

Our primary concern in this thesis is optimality, i.e. computing maxmin optimal strategies for two-player games and optimal strategies in one-player games (best response computation). Hence the natural question to ask is: what kind of strategies are sufficient for optimality? We will see that for one-player games or equivalently for best response computation, when the player is non-absentminded pure strategies are sufficient for optimality.

Lemma 2.5. In a one-player game G without absentmindedness the optimal payoff can be achieved by a pure strategy.

Proof. From Proposition 2.3 it follows that in games without absentmindedness, every behavioral strategy has an equivalent mixed strategy. Hence the optimal payoff can be achieved by a mixed strategy. Now from definition, a mixed strategy is a convex combination of the pure strategies in its support. And hence the payoff is also a convex combination of the payoff with respect to each pure strategy in the support. Hence there must exist a pure strategy in the support which gives at least the optimal payoff. □

However, when the player is absentminded, pure strategies are no longer sufficient. We see this in the following example.

Example 2.7. In Fig. 2.2d no pure strategy can reach the leaf with payoff 1, hence 0 is the maximum payoff obtained by pure strategies. Whereas the behavioral strategy $\frac{1}{2}a + \frac{1}{2}b$ gives a payoff $\frac{1}{2}$ which is the optimal payoff.

To summarize, when Min is not absentminded, and Max plays a behavioral strategy σ, Min has a pure best response against σ. On the other hand, when a player is absentminded it is possible that behavioral strategies are required to obtain optimal payoff.

Remark 2.6. It is possible that in two-player games, mixed strategies provide better maxmin payoff than behavioral strategies, but we will not discuss such cases since in this thesis we concern ourselves with optimality under behavioral strategies.

2.7 Computation of maxmin value

In this thesis, we primarily focus on the computation of the maxmin value and the maxmin strategy in a game G when both players play behavioral strategies. When Max has perfect recall and Min has A-loss recall the maxmin value as well as a maxmin strategy can be computed in PTIME by encoding the problem into a small linear program. The initial linear programming formulation was due to Koller and Megiddo [KM92] and later by Von Stengel [Ste96]. This program was further extended to the class where Min has A-loss recall. For completeness, we revisit this linear program.

2.7.1 Linear program for computing maxmin value

In this section, we revisit the linear program for computing the maxmin value when Max has perfect recall and Min has A-loss recall.

The linear program is composed of two building blocks. The first block involves the best response computation for Min using local optimality constraints. This can be equivalently perceived as computing the optimal strategy in a one-player game. And the second block is a convex polytope[3] called *realization plan* defined by linear constraints which represents the space of all the behavioral strategies of Max with perfect recall. The values in this polytope encode the probabilities of realizing different histories under a behavioral strategy of Max. We analyze each of the building blocks separately and later recombine them for the final linear program.

First, we look at the best response computation for Min. For this, we consider one-player games with Min player having A-loss recall where the goal of Min is to

[3] A polytope in n-dimension is a subset of $\mathbb{R}^n$ described by a set of linear inequation, each inequation defining a half-space

minimize the expected payoff. From Lemma 2.5 we know that in one-player games with A-loss recall the optimal strategy can be obtained by a pure strategy. Roughly speaking the payoff under a pure strategy τ can be computed in terms of local values assigned by τ to a set of nodes with the same action history. Obtaining the optimal strategy involves computing the optimal values for these local values. Since Min has A-loss recall, every node in these local sets is reached by the same sequence of actions under τ. A single local variable is attached to each of these sets, hence the number of required local value variables is bounded linearly in the size of the game.

Value equations for computing payoff under a strategy

Lemma 2.7. Let G be a one-player game Min game without absentmindedness and let Min play the pure strategy τ. Then the expected payoff $\mathbb{E}(\tau)$ to Min on playing τ in G is given by the value $val_\epsilon(\tau)$ satisfying the following equations:

$$val_h(\tau) = \sum_{o \in \mathcal{O}^h_{\mathsf{Min}}} val_{h,o}(\tau) + \sum_{\{t \in L | hist_{\mathsf{Min}}(t) = h\}} \mathcal{P}_{\mathsf{Chance}}(t)\mathcal{U}(t) \quad \forall h \in \mathcal{H}_{\mathsf{Min}} \tag{2.1}$$

$$val_{h,o}(\tau) = val_{h\tau(o)}(\tau) \qquad \forall o \in \mathcal{O}_{\mathsf{Min}}, h \in \mathcal{H}_{\mathsf{Min}}(o) \tag{2.2}$$

Proof. We will show that for an $h \in \mathcal{H}_{\mathsf{Min}}$, $val_h(\tau)$ is the expected payoff under the condition that Min plays a strategy that suggests all actions conforming to the history h and after that from that point onwards she plays following τ[4]. And $val_{o,h}(\tau)$ is the same except that immediately after history h she reaches some node with observation o and then onwards she plays according to τ. Then for the empty history ϵ, $val_\epsilon(\tau)$ is the expected payoff $\mathbb{E}(\tau)$.

We will show this by backward induction on the length of h. When h is the longest, i.e. the only nodes with history h are the leaf nodes and $\mathcal{O}^h_{\mathsf{Min}} = \emptyset$, this statement is true since the payoff is given by $\sum_{\{t \in L | hist_{\mathsf{Min}}(t) = h\}} \mathcal{P}_{\mathsf{Chance}}(t)\mathcal{U}(t) = val_h(\tau)$. Now for any other history h, h is possibly the history of several leaf nodes and also the path with history h passes through observations in $\mathcal{O}^h_{\mathsf{Min}}$. Hence $val_h(\tau)$ can be split as written in Eq. (2.1). Finally, for all nodes with history h and observation o, the game continues along $\tau(o)$, and hence the constraint in Eq. (2.2) is justified. This completes the proof. □

[4]The history h is not necessarily consistent with strategy τ.

LP for best response computation in A-loss recall

The previous lemma demonstrates the bottom-up manner in which the expected payoff can be computed in a one-player game when the strategy is fixed. Based on this, the following linear program computes the optimal payoff in a one-player game with A-loss recall.

$$\max \quad v_\epsilon$$

$$v_h = \sum_{o \in \mathcal{O}^h_{\text{Min}}} v_{h,o} + \sum_{\{t \in L | hist_{\text{Min}}(t) = h\}} \mathcal{P}_{\text{Chance}}(t)\mathcal{U}(t) \quad \forall h \in \mathcal{H}_{\text{Min}} \tag{2.3}$$

$$v_{h,o} \leq v_{ha} \qquad \forall o \in \mathcal{O}_{\text{Min}}, h \in \mathcal{H}_{\text{Min}}(o), a \in \text{Act}(o) \tag{2.4}$$

An optimal solution to the above system of equations $\{v^*\}$ satisfies

$$v^*_{h,o} = \min_{a \in \text{Act}(o)} v^*_{ha} \quad \forall\, o \in \mathcal{O}_{\text{Min}},\ h \in \mathcal{H}_{\text{Min}}(o)$$

Lemma 2.8. Let τ^* be the pure strategy given by $\forall o \in \mathcal{O}_{\text{Min}}, \tau^*(o) = arg \min_{a \in \text{Act}(o)} v^*_{\hat{h}a}$ where $\hat{h} = arg \min_{h' \in \mathcal{H}_{\text{Min}}(o)} v^*_{o,h'}$. Then $\mathbb{E}(\tau^*) = v^*_\epsilon$.

Proof. We first observe that for all histories h consistent with τ^*, we have $v^*_h = val_h(\tau^*)$ and $v^*_{h,o} = val_{h,o}(\tau^*)$. Hence, such histories will satisfy the constraints in Lemma 2.7. Since histories not conforming to τ^* are never reached, leaves with those histories do not contribute to the payoff. As a result $\mathbb{E}(\tau^*) = val_\epsilon(\tau^*)$. □

Lemma 2.9. v^*_ϵ is the optimal expected payoff of Min and τ^* is an optimal strategy of Min.

Proof. We will actually show that for any strategy τ, for a history $h \in \mathcal{H}_{\text{Min}}$, it holds that $v^*_h \leq val_h(\tau)$, and also for any $o \in \mathcal{O}^h_{\text{Min}}$, it holds that $v^*_{h,o} \leq val_{h,o}(\tau)$. Since $val_\epsilon(\tau)$ is the expected payoff under τ and we have $v^*_\epsilon = \mathbb{E}(\tau^*)$ it will follow that v^*_ϵ is the optimal expected payoff and τ^* is an optimal strategy.

The proof goes by backward induction on the length of h. Observe that for a fixed history h, for all strategies of Min, the term $\sum_{\{t \in L | hist_{\text{Min}}(t) = h\}} \mathcal{P}_{\text{Chance}}(t)\mathcal{U}(t)$ is the same. Hence when $\mathcal{O}^h_{\text{Min}} = \emptyset$, the statement is trivially true since in this case $v^* = \sum_{\{t \in L | hist_{\text{Min}}(t) = h\}} \mathcal{P}_{\text{Chance}}(t)\mathcal{U}(t) = val_h(\tau)$. Now let for some h, the statement be true for all $h' \in H_{\text{Min}}$ longer than h.

For $o \in \mathcal{O}^h_{\text{Min}}$, from the constraints we have $v^*_{h,o} \leq v^*_{h\tau(o)}$. And from induction hypothesis, we have $v^*_{h\tau(o)} \leq val_{h\tau(o)}(\tau)$. Hence it follows that $v^*_{h,o} \leq v^*_{h\tau(o)} \leq val_{h\tau(o)}(\tau) = val_{h,o}(\tau)$. This completes the proof. □

Remark 2.10. When a player doesn't have A-loss recall, Lemma 2.9 doesn't hold. For example in Fig. 2.2c, if we make the Min node a chance node with uniform distribution, then this becomes a one-player Max game without A-loss recall. The constraints in the L.P. change accordingly. For the optimal solution, we have $v^*_{d,o_1} = v^*_{dg} = 1$ and $v^*_{e,o_1} = v^*_{eh} = 1$ which makes $v^*_{\epsilon,u_1} = v^*_d$, $v^*_{\epsilon,u_2} = v^*_e$ and $v^*_\epsilon = v^*_{\epsilon,u_1} + v^*_{\epsilon,u_2}$. But this is never possible since this requires Max to play both g and h after observation o_1. Essentially the optimal solution doesn't take into account the fact that once nodes with observation o_1 are reached via both histories d and e simultaneously, in a strategy both histories should continue along the same action from o_1. In games with A-loss recall this problem never arises since two distinct histories with the same observation are never reachable simultaneously.

Realization plan polytope

As we have seen earlier, for a behavioral strategy σ computing the expected payoff involves the set of values $\mathcal{P}_\sigma(t)$ for leaf nodes. Hence expressing these values for a certain behavioral strategy is fundamental in maxmin computation. In general, it can be challenging to represent them with linear constraints. When the player has perfect recall Koller, Megiddo and Von-Stengel demonstrated a set of linear constraints that exactly captures these values generated by some behavioral strategy. These constraints define a convex bounded polytope. This is called the realization plan polytope.

For any two nodes u and v with $\mathit{hist}(u) = \mathit{hist}(v)$ and for a behavioral strategy σ of player i we always have $\mathcal{P}_\sigma(u) = \mathcal{P}_\sigma(v)$. So for the sake of convenience we write $\mathcal{P}_\sigma(h)$ for some $h \in \mathcal{H}_i$.

Linear Program for Perfect recall vs A-loss recall

$$\max \quad v_\epsilon$$

$$v_h = \sum_{o \in \mathcal{O}^h_{\text{Min}}} v_{h,o} + \sum_{\{t \in L | hist_{\text{Min}}(t) = h\}} x_h \mathcal{P}_{\text{Chance}}(t)\mathcal{U}(t) \quad \forall h \in \mathcal{H}_{\text{Min}} \tag{2.5}$$

$$v_{h,o} \leq v_{ha} \qquad \forall o \in \mathcal{O}_{\text{Min}}, h \in \mathcal{H}_{\text{Min}}(o), a \in \text{Act}(o) \tag{2.6}$$

$$x_\epsilon = 1 \tag{2.7}$$

$$x_h = \sum_{a \in \text{Act}(o)} x_{ha} \qquad \forall h \in \mathcal{H}_{\text{Max}}, o \in \mathcal{O}^h_{\text{Max}} \tag{2.8}$$

$$0 \leq x_h \leq 1 \qquad \forall h \in \mathcal{H}_{\text{Max}} \tag{2.9}$$

Lemma 2.11. For any valuation of $\{x\}$ satisfying the constraints Eq. (2.7)-Eq. (2.9) there is a strategy σ_x such that $x_h = \mathcal{P}_{\sigma_x}(h)$. Also, for any strategy σ, if we set $x_h = \mathcal{P}_\sigma(h)$ then this system will satisfy these constraints.

Proof. First, we will prove the second statement. For any strategy σ, $\mathcal{P}_\sigma(\epsilon) = 1$ since the empty history is always realized. For any h and $o \in \mathcal{O}^h_{\text{Max}}$, we have $\sum_{a \in \text{Act}(o)} \mathcal{P}_\sigma(ha) = \sum_{a \in \text{Act}(o)} \mathcal{P}_\sigma(h)\sigma(o,a) = \mathcal{P}_\sigma(h) \sum_{a \in \text{Act}(o)} \sigma(o,a) = \mathcal{P}_\sigma(h)$. Hence the second statement is true.

Now for the first statement, given a valuation $\{x\}$ satisfying the given equation, we build σ_x as follows: For all $o \in \mathcal{O}_{\text{Max}}$, since Max has perfect recall, $|\mathcal{H}_{\text{Max}}(o)| = 1$. Let that particular history be h. σ_x is given by: $\sigma_x(o,a) = \frac{x_{ha}}{x_h}$. From the constraints, it follows that this is a valid behavioral strategy. Now we will show by induction that $x_h = \mathcal{P}_{\sigma_x}(h)$ for all h. This is trivially true for $h = \epsilon$. Let it be true for h. Then for $o \in \mathcal{O}^h_{\text{Max}}$, and $a \in \text{Act}(o)$, $x_{ha} = x_h\sigma_x(o,a) = \mathcal{P}_{\sigma_x}(h)\sigma_x(o,a) = \mathcal{P}_{\sigma_x}(ha)$. This completes the proof. □

Theorem 2.12. *[KM92][Ste96][KK95][Kli02] The LP computes* $\text{MaxMin}_{\text{beh}}(G)$ *when* Max *has perfect recall and* Min *when A-loss recall.*

Proof. From Lemma 2.11 we know that a valuation $\{x\}$, gives a behavioral strategy σ_x of Max. Hence fixing σ_x gives a one-player Min game where we can treat the Max nodes as chance nodes. And in this game, we have the new probability of reaching a leaf t with history h given by $\mathcal{P}_{\sigma_x}(h)\mathcal{P}_{Chance}(t)$ where $\mathcal{P}_{\sigma_x} = x_h$. Hence from Lemma 2.9 it follows that $v_\epsilon \leq BR_{\text{Min}}(\sigma_x)$. Now since this is true for any valuation $\{x\}$ and this encompasses all behavioral strategies of Max, the linear program is essentially $\max_{\{x\}} BR_{\text{Min}}(\sigma_x)$ which is essentially the maxmin value. □

Remark 2.13. Lemma 2.11 doesn't hold when Max doesn't have perfect recall. We can have valuations $\{x\}$ satisfying the constraints which do not correspond to reaching probabilities for any behavioral strategy. For example in Fig. 2.2b , let $x_\epsilon = 1, x_a = x_b = \frac{1}{2}, x_{ac} = x_{ad} = \frac{1}{4}, x_{bc} = \frac{1}{6}$ and $x_{bd} = \frac{1}{3}$ These values satisfy the constraints but do not correspond to a behavioral strategy because the histories a and b extend to different probabilities for c and d individually. Hence in the case of A-loss recall the space of valuations satisfying the constraints is strictly larger than the set of reachable probability valuations arising out of behavioral strategies. This does not happen in perfect recall, because for any observation there is a unique history ending there.

Chapter 3

Complexity of solving imperfect recall games

In this chapter, we present our results on the complexity of the maxmin decision problem both for one-player and two-player games. When players are absent-minded we give lower bounds with respect to complexity classes associated to fragments of First Order Theory of Reals namely $\exists\mathbb{R}$, $\forall\mathbb{R}$ and $\exists\forall\mathbb{R}$. For one-player games with absentmindedness, we show that the payoff maximization problem is $\exists\mathbb{R}$-complete. On the other hand for the maxmin decision problem in two-player games with absentmindedness, we demonstrate both $\exists\mathbb{R}$-hardness and $\forall\mathbb{R}$-hardness along with $\exists\forall\mathbb{R}$ upper bound. For the maxmin problem on games without absentmindedness, we show lower bounds with respect to the Sqrt-Sum problem and co-NP. We conclude with open questions and future directions on the complexity front.

Contents

Computing the maxmin value in two-player zero-sum games is a fundamental computational problem. The complexity theoretic aspect of the maxmin decision problem is quite well studied [KM92][Ste96][HMS07][BP17]. We are concerned with computations over behavioral strategies. We present a finer picture of the complexity of the maxmin decision problem.

For studying complexity we need to define the *size* of a game given in extensive-form.

Definition 13. (Size of a game) The *size of a game* G in extensive-form denoted by $|G|$ is the sum of the bit-lengths of all the chance probabilities and the terminal payoffs in G[1].

Next, we define the maxmin decision problem for behavioral strategies.

3.1 Maxmin Decision Problem

The maxmin decision problem is the following:

Decision Problem 3.1. (MaxMinBeh$_{\geq 0}$)
Given a two-player game G, is $\mathrm{MaxMin}_{\mathrm{beh}}(G) \geq 0$?

Remark 3.1. The general version of this problem which appears in previous studies [KM92][BP17] asks if $\mathrm{MaxMin}_{\mathrm{beh}} \geq \lambda$ for $\lambda \in \mathbb{Q}$. However, the general form can be reduced to the case with $\lambda = 0$. For a game G, this can be done by introducing a chance node at the root and branching out with probability $\frac{1}{2}$ to G and another trivial game(leaf node) with payoff $-\lambda$.

Also, in the specific case when G is a one-player game with Max as the sole player, the decision problem simplifies into asking if MaxBeh$_{\geq 0}$.

Decision Problem 3.2. (MaxBeh$_{\geq 0}$)
Given a one-player game G, is $\mathrm{Max}_{\mathrm{beh}}(G) \geq 0$?

[1]This definition is justified since the number of leaves in a tree with each internal node having at least two children is at least twice the number of non-leaf nodes. Also, the number of observations is at most the number of internal nodes.

Recall of Max	*Complexity*
A-loss recall	PTIME [KM92][KK95][Kli02]
Non-absentminded	NP-Complete [KM92]

Figure 3.1: Previous complexity of $\mathrm{Max}_{\mathrm{beh}} \geq 0$

We will also make use of the strict version of this problem which is the following.

Decision Problem 3.3. ($\textsc{MaxBeh}_{>0}$)
Given a one-player game G, is $\mathrm{Max}_{\mathrm{beh}}(G) > 0$?

Since these decision problems have been studied before, we will review what was known about the complexity of these problems prior to our work.

3.1.1 Known Complexity Results

First, we will review what is known about the one-player decision problem and then continue to the two-player version.

Recall that we observed in Example 2.7, when Max is absentminded, pure strategies are not enough to ensure optimal payoff. We also saw in Lemma 2.5 that in the case of non-absentminded player, pure strategies are sufficient for this purpose. This leads us to the following two complexity results known for one-player games.

Firstly in Lemma 2.9 we saw that the optimal payoff in one-player A-loss recall games can be computed efficiently by a linear program.

Theorem 3.2. *[KM92][KK95][Kli02] The decision problem $\textsc{MaxBeh}_{\geq 0}$ is in* P *when* Max *has A-loss recall.*

And secondly, sufficiency of pure strategies for optimality places it in NP. In addition, it was also shown in [KM92] that $\textsc{MaxBeh}_{\geq 0}$ is NP-hard.

Theorem 3.3. *[KM92] The decision problem $\textsc{MaxBeh}_{\geq 0}$ is* NP*-complete when* Max *is not absentminded.*

Since pure strategies are not sufficient in games with absentmindedness, for general one-player games no specific complexity results were known that was an improvement over the NP-hard lower bound coming from non-absentminded case. The previously known complexity results for one-player games are summarized in Fig. 3.1.

Now we move on to previously known complexity results for two-player games.

Recall of Max	*Recall of* Min	*Complexity*
Perfect Recall	A-loss recall	PTIME [KM92][Ste96][KK95][Kli02]
A-loss Recall	Perfect Recall	NP-hard [CBHL18]

Figure 3.2: Previous complexity of MaxMinBeh$_{\geq 0}$

Again from the linear program in Section 2.7.1, we know that the case when Max has perfect recall and Min has A-loss recall can be solved in P.

Theorem 3.4. *[KM92][Ste96][KK95][Kli02] When* Max *has perfect recall and* Min *has A-loss recall the decision problems* MaxMinBeh$_{\geq 0}$ *is in* P.

But the problem becomes hard when Max no longer has perfect recall. In fact, it is hard even when Max has A-loss recall and Min has perfect recall.

Theorem 3.5. *[CBHL18] The problem* MaxMinBeh$_{\geq 0}$ *is* NP*-hard even when* Max *has A-loss recall and* Min *has perfect recall.*

And similar to the one-player case, this lower bound carries forward to the general case as well where players can have absentmindedness. No better bounds were known for the general case. The previously known complexity results for two-player games can be summarized in Fig. 3.2.

Now that we have stated what was known prior to our work, we will provide a snapshot of our complexity results.

3.2 Our complexity results

For dealing with games with absentmindedness we consider complexity classes associated to fragments of First Order of Theory of Reals. We define these classes and then state our contribution to the complexity picture.

3.2.1 First Order Theory of Reals

We consider the language of first order formulas over equalities and inequalities over $\mathbb{Q}$ whose validity is interpreted over $\mathbb{R}$.

Definition 14. $\mathrm{FOT}(\mathbb{R})$ is the set of all true first order formulas with connectives $\vee, \wedge, \neg$ and quantifiers $\exists, \forall$ over polynomials over the signature $(0, 1, +, *, \leq, <, =)$ and variables interpreted over $\mathbb{R}$[2].

[2]Even though the signature is over 0 and 1, they can be used to encode any rational number efficiently (see page 6 in [SS22])

Example 3.1. $\Phi = \exists x \forall y \ (y - x > 0)$ is a first order formula. However, Φ is false and hence doesn't belong to $\mathrm{FOT}(\mathbb{R})$. On the other hand, the first order formula $\Psi = \forall y \exists x \ (y - x > 0)$ is valid and hence it lies in $\mathrm{FOT}(\mathbb{R})$.

3.2.2 Complexity classes $\exists\mathbb{R}, \forall\mathbb{R}$ and $\exists\forall\mathbb{R}$

Here we focus on the fragment which contains formulas with only existential quantifiers. Existential Theory of Reals(ETR) is the set of all true formulas in $\mathrm{FOT}(\mathbb{R})$ of the form $\exists x_1 \ldots \exists x_n F(x_1 \ldots x_n)$ where F is a quantifier-free formula.

Example 3.2. The fact that the polynomial $x^2 - 5x + 6$ has two distinct real solutions can be expressed by the existential formula $\exists y \exists z \ (y^2 - 5y + 6 = 0) \wedge (z^2 - 5z + 6 = 0) \ \wedge \ (y < z)$. Hence this formula lies in ETR. On the other hand, since the polynomial $x^2 + 1$ has no root in $\mathbb{R}$, the existential formula $\exists x \ (x^2 + 1 = 0)$ is not in ETR.

For associating complexity class to ETR, the natural decision problem of deciding the membership of an existential formula in ETR is considered.

Decision Problem 3.4. (ETR)
Given an existential formula $\exists x_1 \ldots \exists x_n F(x_1 \ldots x_n)$, is it true?

Definition 15. ($\exists\mathbb{R}$)[Sch10] The complexity class $\exists\mathbb{R}$ is the set of all decision problems that has a many-one polynomial time reduction to ETR [3] .

The complexity classes $\forall\mathbb{R}$ and $\exists\forall\mathbb{R}$ are defined similarly.

Decision Problem 3.5. (UTR)
Given a universal first order formula $\forall y_1 \ldots \forall y_m F(y_1 \ldots y_m)$ where F is quantifier free, is it true?

Decision Problem 3.6. (EUTR)
Given a first order formula $\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m$
$F(x_1 \ldots x_n, y_1 \ldots y_m)$ where F is quantifier free, is it true?

Definition 16. ($\forall\mathbb{R}$)[Sch10] The complexity class $\forall\mathbb{R}$ is the set of all decision problems that has a many-one polynomial time reduction to UTR.

Definition 17. ($\exists\forall\mathbb{R}$) The complexity class $\exists\forall\mathbb{R}$ is the set of all decision problems that has a many-one polynomial time reduction to EUTR.

[3] ETR is the theory, ETR is the decision problem, $\exists\mathbb{R}$ is the complexity class

Size of a formula

Any polynomial in a formula is given in its fully expanded form. A polynomial over n variables $x_1, \ldots, x_n$ with total degree d is given in expanded form as $\sum_{d_i | d_1 + \cdots + d_n \leq d} a_{d_1,\ldots,d_n} x^{d_1} \ldots x^{d_n}$ where $a_{d_1,\ldots,d_n}$ is the coefficient of $x^{d_1} \ldots x^{d_n}$. The *size of a polynomial* with n variables and total degree d is the sum of the bit-lengths of the coefficients $a_{d_1,\ldots,d_n}$.

The *size of a formula* is defined as the sum of the size of all the polynomials in it.

Some lower bounds and upper bounds are known about these classes but the exact complexity status is still open.

Theorem 3.6. *[Sch10][Can88]* $\exists\mathbb{R}$ *contains* NP *and* $\forall\mathbb{R}$ *contains co-*NP*.*

Theorem 3.7. *[Can88] [BPR06] The classes* $\exists\mathbb{R}, \forall\mathbb{R}$ *and* $\exists\forall\mathbb{R}$ *are contained in PSPACE*[4]

Max \ Min	Non-existent	Perfect Recall	A-loss Recall	N.A.M	General
Non-existent				co-NP-hard, **co-NP** (Corollary 3.31)	$\forall\mathbb{R}$-hard, $\forall\mathbb{R}$ (Theorem 3.25)
Perfect Recall	**PTIME** [KM92] [CBHL18]				
A-loss Recall	**PTIME** [CBHL18]	NP-hard [CBHL18] SQRT-SUM-hard (Theorem 3.28)			
N.A.M	NP-hard, **NP** [KM92]				
General	$\exists\mathbb{R}$-hard (Theorem 3.18)	$\exists\mathbb{R}$ (Theorem 3.20)			$\exists\forall\mathbb{R}$ (Theorem 3.25)

Figure 3.3: Complexity of MAXMINBEH$_{\geq 0}$

For games without absentmindedness, we use another decision problem known as the SQRT-SUM problem which we define next.

3.2.3 SQRT-SUM Problem

Decision Problem 3.7 (SQRT-SUM)**.** Given k positive integers $a_1, \ldots, a_k$ and another positive integer n, is $\sum_i \sqrt{a_i} \leq n$?

A decision problem D is said to be SQRT-SUM-hard, if there is polynomial time many-one reduction from SQRT-SUM to D.

Theorem 3.8. *[ABKM09] The SQRT-SUM problem is in the 4th level of the counting hierarchy which is in PSPACE.*

[4] In fact any complexity class defined similarly with bounded number of quantifiers is in PSPACE. This is not stated explicitly but discussed in Remark 13.11 in [BPR06]

Remark 3.9. There is another version of the Sqrt-Sum problem studied where given m positive integers $a_1, \ldots, a_m$ and n positive integers $b_1, \ldots, b_n$, is $\sum_i \sqrt{a_i} \geq \sum_i \sqrt{b_i}$. Although the version we stated reduces to this version, it is still unknown whether reduction in the other direction is also possible.

3.2.4 New Complexity Picture

Now that we have defined the relevant complexity classes we present our results in a compact manner in the table in Fig. 3.3. The complexity of MaxMinBeh$_{\geq 0}$ depends on the individual memory recalls of Max and Min which can be found in the corresponding cell in Fig. 3.3. The case corresponding to non-existent instances of either player indicates one-player games(except when both are non-existent). When Min is non-existent it is a one-player game with only Max player, i.e. complexity of MaxBeh$_{\geq 0}$. And When Max is non-existent the complexity in the table corresponds to the complexity of MaxMinBeh$_{\geq 0}$ but with only Min player(or a dummy Max player with no influence on outcome). A lower-bound (i.e. hardness) for MaxMinBeh$_{\geq 0}$ for a pair of memory recall of Max and Min in a cell in Fig. 3.3 carries forward inductively to all the cells to its right and below. Similarly, an upper bound (in bold font) for MaxMinBeh$_{\geq 0}$ holds inductively to all cells to the left and above of a cell. So essentially fixing a memory recall of Max, the complexity of MaxMinBeh$_{\geq 0}$ potentially increases with a decrease in memory recall of Min(towards the right). And fixing the memory recall of Min, the complexity of MaxMinBeh$_{\geq 0}$ potentially increases with the decrease in memory recall of Max(downwards). Hence we can observe the general trend that the weaker memory of players makes the computation of the maxmin value harder.

Our contributions to the complexity picture are indicated by the corresponding theorems, which we will state and prove later in this chapter. For games with absentmindedness (and consequently the general case), we show $\exists\mathbb{R}$-completeness for the one-player case and for the two-player case where Min has A-loss recall. We also show that MaxMinBeh$_{\geq 0}$ is $\forall\mathbb{R}$-hard for the one-player absentminded case with a Min player. These lower bounds naturally extend to the two-player case, making the general case both $\exists\mathbb{R}$-hard and $\forall\mathbb{R}$-hard. We also show $\exists\forall\mathbb{R}$ upper-bound for the general case.

For the case where players are not absentminded, the question of whether MaxMinBeh$_{\geq 0}$ is Sqrt-Sum-hard was posed in [CBHL18]. We settle this question by showing that it is in fact Sqrt-Sum-hard. We also show that the one-player non-absentminded case with only Min player is co-NP-complete. The lower bound naturally extends to the 2-player non-absentminded case. These two new lower bounds for the non-absentminded case further reinforce the belief that MaxMinBeh$_{\geq 0}$ for this case is not in NP.

Next, we will see how we obtain our complexity results. We mention the relevant complete problems in the next section and the main technique that we used for these reductions.

3.3 Path to reduction

In this section, we mention the complete problem for the class $\exists\mathbb{R}$ that we use for our reduction. In the founding works on the class $\exists\mathbb{R}$, several complete problems were introduced in [SS17]. We make use of a certain complete problem that involves checking if a system of quadratic equations has a common real root. The complete decision problem for the class $\exists\mathbb{R}$ that we are concerned with is the following:

Decision Problem 3.8 (QUAD)**.**
Given a system of quadratic equations $\{Q_i(X)\}_{i\in[s]}$ over $X = \{x_1, \ldots, x_n\}$, do they have a common root in $\mathbb{R}^n$?

This problem was shown to be $\exists\mathbb{R}$-complete [SS17].

Theorem 3.10. *[SS17] QUAD is $\exists\mathbb{R}$-complete.*

The decision problem QUAD asks for the existence of a root over the whole $\mathbb{R}^n$. However in order to use this complete problem for reducing to a target problem one might need some suitable bound on the roots.

Let $B_n(0,1) = \{(x_1, \ldots, x_n) \mid \sum_i x_i^2 < 1\}$ be the unit ball around 0. It was shown in [Sch13] that QUAD remains $\exists\mathbb{R}$-complete even with the given promise that whenever a root exists for the system of equations, one can also find a root in $B_n(0,1)$. For our purpose, we will need the roots to lie in the unit hypercube of dimension n given by $H_n = [0,1]^n$. This is because we will see later that the space of behavioral strategies can be represented as a point in H_n. In fact we will use another intermediate problem where the roots are promised to exist in a specific subset of H_n namely the corner simplex.

Let $\Delta_n^c = \{(x_1, \ldots, x_n) \mid \forall i \in [n]\ x_i >= 0 \wedge \sum_i x_i \leq 1\}$ be the standard corner n-simplex. Since for any $(x_1, \ldots, x_n) \in \Delta_n^c$, we have $\forall i, 0 \leq x_i \leq 1$, it follows that $\Delta_n^c \subseteq H_n$. Using similar techniques from [Sch13] for the $B_n(0,1)$ case it was proved in [Han19] that the complexity of QUAD remains the same when the promise set is Δ_n^c.

Theorem 3.11. *[Han19] It is $\exists\mathbb{R}$-complete to decide if a system of quadratic equations $\{Q_i(X)\}_{i\in[s]}$ has a common root even with the given promise that if the system has a common root, they also have a common root in Δ_n^c.*

We modify the problem QUAD by asking for the existence of a root in H_n.

Decision Problem 3.9 (QUAD-H)**.**
Given a system of quadratic equations $\{Q_i(X)\}_{i\in[s]}$ over $X = \{x_1, \ldots, x_n\}$, do they have a common root in H_n?

Using Theorem 3.11 we show that QUAD-H is $\exists\mathbb{R}$-complete as well.

Corollary 3.12. QUAD-H is $\exists\mathbb{R}$-complete.

Proof. Firstly this problem is in $\exists\mathbb{R}$ because the existence of solution can be expressed by the existential formula $\exists x_1 \ldots x_n(\wedge_{i\in[s]} Q_i(x_1 \ldots x_n) = 0) \wedge (\wedge_{j\in[n]} 0 \leq x_j \leq 1)$.

For the hardness, we make use of Theorem 3.11 and reduce the problem of deciding existence of real roots of a system to deciding existence of root in H_n. We take a system of quadratic equations $\{Q_i(X)\}_{i\in[s]}$ which comes with the promise that if they have a common root they have one in Δ_n^c. We claim that this system has a common real root if and only iff it has a common root in H_n. For the forward direction, if the system has a common root, it has one in Δ_n^c and since $\Delta_n^c \subseteq H_n$, it follows that the root lies in H_n as well. For the other direction, if the system has a common root in H_n, we have a root of the system. As a result, the initial decision problem is reduced to checking the existence of a root in $H_n$of the same system of equations. Hence QUAD-H is $\exists\mathbb{R}$-complete. □

As we have established that QUAD-H is $\exists\mathbb{R}$-complete, next we see how the system of polynomials and the set H_n are linked to games. We will express a behavioral strategy as a set of variables taking values over H_n. Computing the expected payoff in terms of these variables will generate a *payoff polynomial*. Then we will formulate the optimization problem for games as a root finding problem for a suitable system of equations using these polynomials.

So first we construct the link between polynomials and extensive-form game payoffs. In fact we only need to consider games where at every stage players have exactly two choices.

Definition 18 (Binary decision game)**.** An extensive-form game is called a *binary decision* game if $\forall i \in \{\mathsf{Max}, \mathsf{Min}\},\ \forall o \in \mathcal{O}_i \mid \mathrm{Act}(o)| = 2$.

For simplification of notation, in a binary decision game at some observation set, we use the notation $\{a, \bar{a}\}$ for its actions. We often denote the actions at $o \in \mathcal{O}$ by a_o and $\bar{a}_o$. [5]. Binary decision games simplify representing behavioral strategies because, for every observation o, the probability of playing a_o determines the

[5] $\bar{a}$ is not an operation on action a. Rather the actions are named a and $\bar{a}$.

probability of playing $\bar{a}_o$. So for every observation o, one variable suffices for a symbolic representation of a strategy. Next, we see the *payoff polynomial* generated by a game.

3.3.1 Payoff polynomial

Definition 19. The payoff polynomial $f_G(X)$ of a game G is a polynomial over $X = \{x_o\}_{o \in \mathcal{O}}$ given by the expected payoff when players play their respective behavioral strategies where for each o, a_o is played with probability x_o and $\bar{a}_o$ is played with probability $1 - x_o$.

A valuation of the set of variables X over $H_{|\mathcal{O}|}$ gives a behavioral strategy profile of the players and the payoff polynomial $f : H_{|\mathcal{O}|} \mapsto \mathbb{R}$ evaluates to the expected payoff under this strategy profile.

When X is understood from context we abuse notation and use f_G for the payoff polynomial of G.

Example 3.3. In Fig. 3.4, $\text{Act}(o_1) = \{a, \bar{a}\}$ and $\text{Act}(o_2) = \{b, \bar{b}\}$ respectively. Let $x = x_{o_1}$ be the probability of playing the action a after observing o_1. Hence $\bar{a}$ is played with probability $1 - x$. Similarly let $y = x_{o_2}$ be probability of playing action b. $\bar{b}$ is played with probability $1 - y$. The expected payoff in this game is the payoff polynomial of this game which is given by

$$\tfrac{1}{4}\Big(12x^2 + 0.x(1-x) + 0.(1-x) + 20xy + 0.x(1-y) + 0.(1-x) - 32y^2 + 0.y(1-y) + 0.(1-y) - 4\Big)$$
$$= 3x^2 + 5xy - 8y^2 - 1.$$

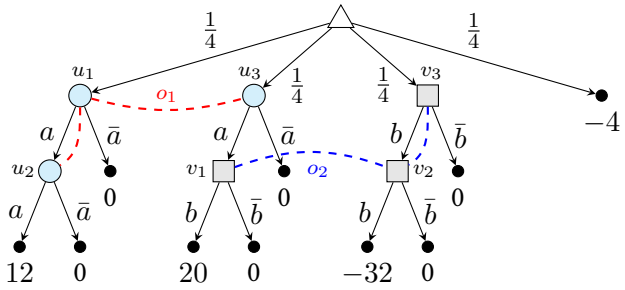


Figure 3.4: Game with payoff polynomial $3x^2 + 5xy - 8y^2 - 1$

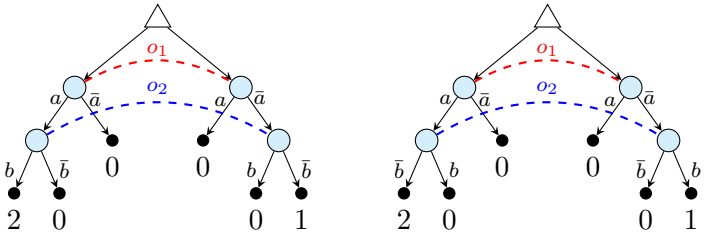
For computing the maxmin value of a binary decision game G using its payoff polynomial we make the following natural observation.

Maxmin value using payoff polynomial
Let $X = \{x_1 \dots x_n\}$ and $Y = \{y_1 \dots y_m\}$. Let G be a binary decision game with $|\mathcal{O}_{\mathsf{Max}}| = n$ and $|\mathcal{O}_{\mathsf{Min}}| = m$. $f(X, Y)$ be the payoff polynomial over $X \cup Y$ with variables in X and Y associated to observation of Max and Min respectively. Then

$$\mathrm{MaxMin}_{\mathrm{beh}}(G) = \max_{X \in H_n} \min_{Y \in H_m} f(X, Y)$$

.

Remark 3.13. Recall that an extensive-form has an underlying unlabelled game tree. The payoff polynomial is not associated to the unlabelled game tree but to the game after actions are labeled. The actions could be labeled differently on the same game tree to obtain a different payoff polynomial. For example in Fig. 3.5 both the games have the same underlying game tree but the action labels are different. The only difference is the labels b and $\bar{b}$ have been swapped in the game on the right. With $x_{o_1} = x$ and $x_{o_2} = y$, the game on the left has payoff polynomial $3xy - x - y + 1$ whereas the game on the right has payoff polynomial $2x + y - 3xy$. However, since this is a swap of actions at the same observation, the second polynomial can be obtained from the first by substituting y with $1 - y$.



(a) Game with payoff polynomial $3xy - x - y + 1$ (b) Game with payoff polynomial $2x + y - 3xy$

Figure 3.5: Two games with same game tree but different labels different payoff polynomial

So to every binary decision game G we can naturally associate a payoff polynomial f_G. But what about the other direction? If we start with a polynomial f, can we find a game G such that f_G is f? The answer is yes as we show in Theorem 3.14 which will be the crux of our reductions.

Theorem 3.14. *Given a polynomial $f(X_1, X_2)$ over two disjoint sets of variables X_1 and X_2, there exists a binary decision game G_f with the following properties.*

- *The set of observations $\mathcal{O}_{\mathsf{Max}}$ of player Max is in correspondence with the set of variables X_1 given by $\mathcal{O}_{\mathsf{Max}} = \{o_x\}_{x \in X_1}$. The same holds for observations of player Min: $\mathcal{O}_{\mathsf{Min}} = \{o_x\}_{x \in X_2}$.*
- *The payoff polynomial of G when player play action a_{o_x} with probability x is $f(X_1, X_2)$.*

Proof. Given a polynomial f the game G_f is constructed as follows. Let $|X_1 \cup X_2| = n$. Suppose $f(X_1, X_2)$ has k monomial terms $\mu_1, ..., \mu_k$ over $X_1 \cup X_2$ i,e,$f(X_1, X_2) = \sum_i c_i \mu_i$ where c_i is coefficient of μ_i. The root node r of G_f is a Chance node with k outgoing edges. For each term μ_i in $f(X_1, X_2)$, there is a node s_i and there is a transition from r to each s_i with probability $\frac{1}{k}$. For a constant term μ_l, s_l is just a leaf node. The height of each s_i is equal to the total degree D_i of μ_i. Let $\mu_i = \prod_j x_j^{d_{i,j}}$ where $d_{i,j}$ is the degree of variable x_j in μ_i and $\sum_{d_{i,j}} = D_i$. In the sub-tree under s_i there is a path from s_i to a terminal node t_i of length D_i given by $s_i^1 \dots s_i^{d_{i,1}} \dots s_i^j \dots s_i^{d_{i,j}} \dots s_i^n \dots s_i^{d_{i,n}} t_i$ where $s_i = s_i^1$. Essentially there are $d_{i,j}$ nodes on this path for each x_j in μ_i. Each of these nodes on the path has two outgoing edges of which the edge not leading to t_i goes to a terminal node with utility 0. In the terminal node t_i the utility is equal to kc_i where c_i is the coefficient of μ_i. Nodes s_i^j assigned due to variable $x_j \in X_1$ are controlled by Max whereas those assigned for variables in X_2 are controlled by Min. All the nodes s_i^j assigned due to the same variable x_j have the same observation o_{x_j}. For all nodes with observations o_{x_j} on $\mathrm{PathTo}(s_i, t_i)$ for each i, the action from these nodes are labelled a_{x_j} and the other action leading to utility 0 is labelled $\bar{a}_{x_j}$. The set of all observations is $\{o_x\}_{x \in X_1 \cup X_1}$. G_f is indeed a binary game. Now the payoff polynomial of this game is comprised only of the product of the variables on the path to t_i's since all the other utilities are 0. Given that the utility at t_i is c_i thus payoff polynomial of this game is exactly $f(X_1, X_2)$. Finally, since each leaf in the game tree represent a monomial in the polynomial with the coefficient as its payoff (including the 0 coefficient of the absent monomials) the size of the game is linear in the size of f. □

Corollary 3.15. Given a polynomial $f(X_1, X_2)$, the game G_f can be constructed in PTIME and its size is polynomially bounded in the size of f.

Proof. This statement follows from the construction described in the proof of Theorem 3.14. The construction takes PTIME and the size of the game is also polynomially bounded. □

Example 3.4. In Fig. 3.4 the game G_f constructed from $f(x, y) = 3x^2 + 5xy - 8y^2 - 1$ using the procedure described in the proof of Theorem 3.14 is demonstrated.

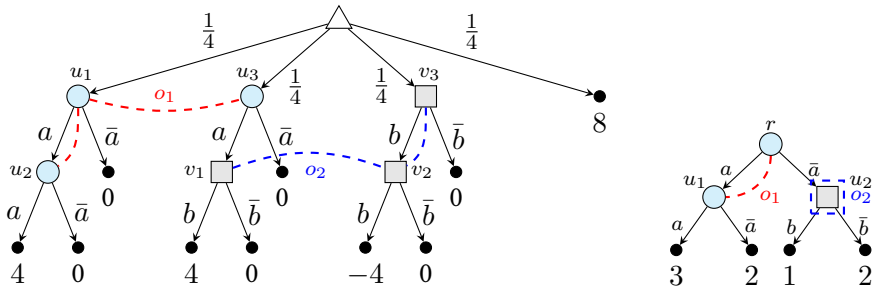


Figure 3.6: Different games with same payoff polynomial

Remark 3.16. Given a polynomial $f(X_1, X_2)$ the construction in the proof of Theorem 3.14 gives one specific game. However, this is not necessarily the only game with the same payoff polynomial. For example consider the polynomial $f(x, y) = x^2 + xy - y + 2$. Following the construction would yield the game on the left of Fig. 3.6. On the other hand, the game on the right is a different game with the same payoff polynomial.

Now we are ready to discuss how we get to our complexity results.

3.4 Proof of complexity: Games with absentminded players

First, we will prove complexity bounds for one-player games. The lower bound results for the one-player case will naturally carry forward as lower bounds to the two-player case. This is because one-player games can be thought of as trivial two-player games just with a dummy second player.

3.4.1 One player games

For general one-player games, when Max is absentminded it follows from Theorem 3.3 that MaxBeh$_{\geq 0}$ is NP-hard. However, the exact complexity of the problem depends on the precision required in describing an optimal behavioral strategy. It was shown via an example in [HMS07] that in absentminded one-player games, Max needs a behavioral strategy with irrational probabilities for optimality. This dismisses the possibility of using optimal strategies as short witnesses for inclusion in NP. Going one step further, we give a further generalization of this. It turns out that there is a class of games where not only does the optimal behavioral strategy of Max require irrational probabilities, but the optimal payoff to Max is

also irrational. We show that for positive integers n and k, there are one-player games with optimal value $n^{\frac{1}{k}}$.

Proposition 3.17. For positive integers $n, k \in \mathbb{Z}^+$, there exists a game $H_{n,k}$ where every optimal behavioral strategy of Max requires irrational probabilities and the optimal payoff to Max is $n^{\frac{1}{k}}$.

Proof. Let $f_{n,k} = \frac{nx(k+1)-n^k x^{k+1}}{k}$ be a univariate polynomial in x. Consider the game $H_{n,k} = G_{f_{n,k}}$ constructed according to Theorem 3.14. It can be shown that for $x \in [0,1]$, the maximum value of $f_{n,k}$ is attained at $x = n^{\frac{1-k}{k}}$ and the maximum value is $n^{\frac{1}{k}}$. This proves that the optimal payoff of the game $H_{n,k}$ is $n^{\frac{1}{k}}$ [1]. □

Proposition 3.17 gives an insight into the range of values required in an optimal strategy or the optimal payoff.

Next, we show $\exists\mathbb{R}$-completeness of the one-player case.

Theorem 3.18. *The problem* MaxBeh$_{\geq 0}$ *is* $\exists\mathbb{R}$*-complete.*

Proof. The inclusion in $\exists\mathbb{R}$ can be shown by encoding a behavioral strategy in a game with absentmindedness with the help of variables and constraints. The set of variables is $X = \{x_a\}_{a \in A_{\text{Max}}}$. For X to represent a behavioral strategy, the set of constraints $C(X)$ are : $\forall a \in A_{\text{Max}}, 0 \leq x_a \leq 1$ and $\forall o \in O_{\text{Max}}, \sum_{a \in \text{Act}(o)} x_a = 1$. Under these constraints the expected payoff is a polynomial[6] $F(X)$ over X with as many terms as the number of observations. Finally, with the aid of these constraints MaxBeh$_{\geq 0}$ can be expressed as an existential formula as follows:

$$\exists X, C(X) \wedge (F(X) \geq 0)$$

.

For hardness, we use the fact that QUAD-H is $\exists\mathbb{R}$-complete from Corollary 3.12. We will reduce QUAD-H to MaxBeh$_{\geq 0}$.

Given a quadratic system $\{Q_i(X)\}_{i \in [s]}$, observe that $Z \in H_n$ is a solution to the system iff $-\sum_i Q_i(Z)^2 \geq 0$. For $F(X) = -\sum_i Q_i(X)^2$ using Theorem 3.14 we can construct the game G_F with payoff polynomial $F(X)$ in polynomial time. Any point $v \in H_n$ gives a behavioral strategy in G_F and $F(v)$ is the expected payoff under this strategy. It follows that the system of equations has a common root in H_n iff MaxBeh$_{\geq 0}$ in the game G_F. This completes the proof. □

[1] Construction of $G_{n,k}$ is not polynomial in n and k. However, this becomes the case when k is constant.

[6] We don't deal with payoff polynomials here as explained in Remark 3.19

Recall of Max	*Complexity*
A-loss recall	P [KK95][Kli02]
Non-absentminded	NP-Complete [KM92]
General	$\exists\mathbb{R}$-complete (Theorem 3.18)

Figure 3.7: Complexity of $\mathrm{Max}_{\mathrm{beh}} \geq 0$

The complexity for one-player games is summarized in Fig. 3.7

Remark 3.19. In the $\exists\mathbb{R}$ upper bound proof in Theorem 3.18, note that we don't use the payoff polynomials which we used earlier in Theorem 3.14. This is because of two reasons. Firstly, we defined payoff polynomials only for binary decision games and not for general games to get rid of the constraints. Secondly, the payoff polynomial of a binary decision game in its expanded form can have exponentially many terms in the size of the game. This cannot be avoided even when the game tree is labeled differently. To see why, consider a general game G of depth n with a similar underlying structure as in the games in Fig. 3.5. The leftmost and rightmost extreme leaves are both at depth n. Each of them has a payoff of 1 and all the other leaves have a payoff of 0. Each level with a different depth has its own observation. Now no matter how we label the game tree with a_i and $\bar{a}_i$ for $i \in [n]$, at least one of the paths to the two extreme leaves will have $n/2$ labels of the type $\bar{a}_i$. Hence this path will contribute to the product at least $n/2$ terms of the form $(1 - x_i)$. But expanding this will generate $2^{o(n)}$ terms in the payoff polynomial whereas the size of the game is $O(n)$.

3.4.2 Two player games

For the two-player case, lower bounds for the one-player version apply to the two-player version as well. As a result, the problem MaxMinBeh$_{\geq 0}$ is $\exists\mathbb{R}$-hard. As shown in [KK95][Kli02] we know that when Min has A-loss recall the best response can be expressed as a linear program. We use the same linear program and the encoding of behavioral strategies in Theorem 3.18 to show that MaxMinBeh$_{\geq 0}$ is $\exists\mathbb{R}$-complete when Min has A-loss recall.

Theorem 3.20. *When player 2 has A-loss recall, the problem* MaxMinBeh$_{\geq 0}$ *is* $\exists\mathbb{R}$*-complete.*

Proof. The $\exists\mathbb{R}$-hardness follows from Theorem 3.18.

For the upper bound, we formulate this problem as a mathematical program based on the same principle as in the linear program in Section 2.7.1. We replace the realization polytope there by the encoding of behavioral strategies of Max

as done in Theorem 3.18. We keep intact the best response computation for Min using value equations as in Lemma 2.9, where $\mathcal{P}(t)$ for a leaf node t is now a polynomial over X. The maxmin value is given by the optimal value achieved by this mathematical program. MaxMinBeh$_{\geq 0}$ is reduced to an existential formula by expressing the existence of values of X which gives an optimal value of at least 0 as an ETR formula. Hence MaxMinBeh$_{\geq 0}$ is in $\exists\mathbb{R}$. □

For the general case, we will show that this problem is also $\forall\mathbb{R}$-hard. Since $\exists\mathbb{R}$ and $\forall\mathbb{R}$ are not believed to be strictly contained in one another, this is a good indication that the general problem is not in $\exists\mathbb{R}$. For this, we need some tools from algebraic geometry namely semi-algebraic sets and some relevant results from [SS17]. We briefly review semi-algebraic sets and then state the results we use.

Definition 20. A set $S \subseteq \mathbb{R}^n$ is called a semi-algebraic set if there exists a quantifier free boolean formula $\Phi(X)$ over the signature $(0, 1, +, *, \leq, <, =)$, such that $S = \{X \in \mathbb{R}^n | \Phi(X)\}$.

For a quantifier boolean formula Φ, let $S_\Phi = \{X \in \mathbb{R}^n | \Phi(X)\}$. Let the complexity of a formula Φ [SS17] be the number of bits used to represent Φ. The complexity of a semi-algebraic set S is the complexity of a formula Φ with the shortest representation such that $S = S_\Phi$.

For two vectors $X, Y \in \mathbb{R}^n$, with X_i, Y_i as ith co-ordinates, the Euclidean distance between X and Y is given by $\|X - Y\| = \sqrt{\sum_{i\in[n]}(X_i - Y_i)^2}$. The distance between two sets S_1 and S_2 is defined as $dist(S_1, S_2) = \inf_{X\in S_1, Y\in S_2} \|X - Y\|$.

The main result on semi-algebraic sets that we are concerned with is the distance between two semi-algebraic sets. For example, when two compact semi-algebraic sets are disjoint, it is a standard result that there is a positive distance between them. Our primary interest lies in a lower bound on this distance in terms of the complexity of the sets. In [JPT13] a lower bound is derived in terms of other parameters. In [SS17] an explicit lower bound is derived in terms of the complexity of the semi-algebraic sets stated as follows.

Theorem 3.21. *[SS17] S_1 and S_2 be two semi-algebraic sets in $\mathbb{R}^n$ both with complexity at most $L \geq 5n$ such that $S_1 \cap S_2 = \emptyset$ and $dist(S_1, S_2) > 0$. Then $dist(S_1, S_2) \geq 2^{-2^{L+5}}$.*

We will use this result to show that whenever the optimal payoff in a one-player game is negative, it is well below a negative constant derived from the game itself.

Lemma 3.22. For all one-player game G, if $\mathrm{Max}_{\mathrm{beh}}(G) < 0$, then $\exists \delta_G > 0$ such that $\mathrm{Max}_{\mathrm{beh}}(G) < -\delta_G$

Proof. The proof uses similar techniques as in [SS17]. Let $g(X)$ be the polynomial expressing the expected payoff in the game G when the behavioral strategy is given by the variables X. Define two sets $S_1 := \{(z, X) | z = g(X), X \in H_n\}$ and $S_2 := \{(0, X) | X \in H_n\}$. From definition, it follows that S_1 and S_2 are semi-algebraic sets. When $\mathrm{Max}_{\mathrm{beh}}(G) < 0$, $g(X)$ always attains negative values, and hence S_1 and S_2 do not intersect. Since both S_1, S_2 are compact we have $dist(S_1, S_2) > 0$. Theorem 3.21 ensures that $dist(S_1, S_2) > 2^{-2^{L+5}}$ where L is the complexity of S_1 and S_2.

Now for any two $X, Y \in H_n$ we have $\|(g(X), X) - (0, Y)\| \geq |g(X)|$. Hence, we have

$$dist(S_1, S_2) = \inf_{X,Y \in H_n} \|(g(X), X) - (0, Y)\| = \inf_{X \in H_n} |g(X)| = -\mathrm{Max}_{\mathrm{beh}}(G).$$

As a result it follows that for $\delta_G = 2^{-2^{L+5}}$, we have $\mathrm{Max}_{\mathrm{beh}}(G) < -\delta_G$. □

The last lemma tells us that for any game G, there is a $\delta_G \in \mathbb{R}^+$ such that $\mathrm{Max}_{\mathrm{beh}}(G)$ cannot lie in the interval $[-\delta_G, 0)$. Also, this δ_G can be expressed in terms of the complexity of semi-algebraic sets obtained using the payoff polynomial of G.

Next, we will show that the strict version of the one-player problem is $\exists\mathbb{R}$-hard.

Theorem 3.23. *MaxBeh$_{>0}$ is $\exists\mathbb{R}$-hard.*

Proof. We reduce MaxBeh$_{\geq 0}$ to MaxBeh$_{>0}$, by reducing the problem of checking if $\mathrm{Max}_{\mathrm{beh}}(G) \geq 0$ to checking if $\mathrm{Max}_{\mathrm{beh}}(G') > 0$ for some constructed game G'. From G we construct in polynomial time a game G' whose optimal payoff is $\mathrm{Max}_{\mathrm{beh}}(G) + \epsilon_G$ such that $0 < \epsilon_G < \delta_G$ where δ_G is derived from Lemma 3.22. It follows that $\mathrm{Max}_{\mathrm{beh}}(G) \geq 0$ if and only if $\mathrm{Max}_{\mathrm{beh}}(G) + \epsilon_G > 0$.

We now describe the construction of G'. We use the same L as used in the proof of Lemma 3.22. Recall that $\delta_G = 2^{-2^{L+5}}$. Let $t = L+5$ and let $Y = \{y_0, y_1, \ldots, y_t\}$ be a set of variables. For $i \in \{0, \ldots, t-1\}$, let $F_i(Y) := 2y_i - y_{i+1}^2$ and let $F_t(Y) := 2y_t - \frac{1}{4}$. Let $P(Y) := -\sum_i F_i^2(Y)$ and $Q(Y) := \prod_i y_i^2$. Let G_P and G_Q be the corresponding game. Let G_1 be a game constructed from G by multiplying at each leaf node of G, the payoff by a factor of 3. G' is the game as follows: Its root node is a Chance node with edges to three children each with probability $\frac{1}{3}$. To the first child, we attach the game G_1, to the second child, the game G_P and to the third child we attach the game G_Q. Information sets in G_P and G_Q follow the structure imposed by the variables, whereas information sets in G_1 are independent from the rest of the game. We set $\epsilon_G = \frac{1}{3}(\max_{Y \in H_{t+1}} P(Y) + Q(Y))$. Since scaling all payoff in a game by a positive factor, scales the maxmin value by

the same factor we have $\mathrm{Max}_{\mathrm{beh}}(G_1) = 3\,\mathrm{Max}_{\mathrm{beh}}(G)$. Hence it will follow that $\mathrm{Max}_{\mathrm{beh}}(G') = \mathrm{Max}_{\mathrm{beh}}(G) + \epsilon_G$.

Now we will show that $0 < \epsilon_G < \delta_G$.

Setting $y_t = 1/8$ and each y_i to $y_{i+1}^2/2$ for $i \in [0, \ldots, t-1]$ we have $P(Y) = 0$ and $Q(Y) > 0$. Hence $\epsilon_G > 0$.

Let $y_{t+1} = 1/2$ be a constant. $F_t(Y)$ can be rewritten as $2y_t - y_{t+1}^2$. Now consider any valuation of Y in H_{t+1}. If for some $i \in [0, \ldots, t]$, $y_i - y_{i+1}^2 \geq 0$, then we have $F_i(Y) = y_i + (y_i - y_{i+1}^2) \geq y_i$. This implies $P(Y) \leq -y_i^2$. Since $Q(Y) \leq y_i^2$, in this case $P(y) + Q(Y) \leq 0 < \delta_G$.

Otherwise for the case when $\forall i \in [0, \ldots, t], y_i < y_{i+1}^2$, we claim that $y_0 < (\frac{1}{2})^{2^t}$. We have $y_0 < y_1^2$ and inductively we can show that $\forall k > 0$, $y_0 < y_k^{2^k}$. Hence more particularly, we have have $y_0 < (y_t)^{2^t}$ but since $y_t < y_{t+1}^2 < 1/2$ it follows that $y_0 < 2^{-2^t} = \delta_G$. This implies $Q(Y) < \delta_G$. And since $P(Y) \leq 0$, in this case as well we have $P(Y) + Q(Y) < \delta_G$.

This means $P(Y) + Q(Y)$ assumes at least one positive value and all values are strictly less than δ_G. Hence $0 < \epsilon_G < \delta_G$. This completes the proof. □

Before proving the final result we will use few auxiliary decision problems.

Decision Problem 3.10. (MinBeh$_{<0}$) In a one-player game with only Min player, is the minimum payoff achievable by Min strictly negative ?

We also use the complement decision problem of MinBeh$_{<0}$.

Decision Problem 3.11. (MinBeh$_{\geq 0}$) In a one-player game with Min player, is the minimum payoff achievable by Min non-negative ?[7]

MinBeh$_{\geq 0}$ is a special case of MaxMinBeh$_{\geq 0}$ when there is no Max player (or a dummy Max player with no influence on the game outcome).

Lemma 3.24. The decision problems MaxBeh$_{>0}$ and MinBeh$_{<0}$ are polynomial time reducible to each other.

Proof. For a one-player game G with only Max player, let $\bar{G}$ be the one-player Min game with the same game tree as G, where all Max nodes in G are now Min nodes and the terminal payoffs are multiplied by a factor of -1. Since a strategy of Max in G is a strategy of Min in $\bar{G}$, it can be shown that $\mathrm{Max}_{\mathrm{beh}}(G) = -\mathrm{Min}_{\mathrm{beh}}(\bar{G})$. Hence it follows that $\mathrm{Max}_{\mathrm{beh}}(G) > 0 \iff -\mathrm{Min}_{\mathrm{beh}}(\bar{G}) < 0$. Since starting from any one-player game with Min player, we can similarly construct the corresponding Max game, it follows that MaxBeh$_{>0}$ and MinBeh$_{<0}$ can be reduced to each other in polynomial time. □

[7] This problem doesn't ask if Min can achieve a non-negative payoff, rather if the minimal payoff is non-negative.

Theorem 3.25. *The decision problem* MaxMinBeh$_{\geq 0}$ *is in* $\exists\forall\mathbb{R}$ *and is both* $\exists\mathbb{R}$*-hard and* $\forall\mathbb{R}$*-hard.*

Proof. We will just show the $\forall\mathbb{R}$-hardness since we have already shown $\exists\mathbb{R}$-hardness for one-player case in Theorem 3.18. It follows from Lemma 3.24 that since MaxBeh$_{>0}$ is $\exists\mathbb{R}$-hard, MinBeh$_{<0}$ is $\exists\mathbb{R}$-hard as well. Hence the complement problem of MinBeh$_{<0}$ which is MinBeh$_{\geq 0}$ is $\forall\mathbb{R}$-hard. But since MinBeh$_{\geq 0}$ is a special case of MaxMinBeh$_{\geq 0}$, it follows that MaxMinBeh$_{\geq 0}$ is $\forall\mathbb{R}$-hard. □

Corollary 3.26. The decision problem MaxMinBeh$_{\geq 0}$ is $\forall\mathbb{R}$-complete for the single player game with only Min player.

Proof. The $\forall\mathbb{R}$ upper-bound follows from the fact that the complement problem of MinBeh$_{\geq 0}$ is MinBeh$_{<0}$, which can be reduced to MaxBeh$_{>0}$. MaxBeh$_{>0}$ is in $\exists\mathbb{R}$ since an instance of this problem can be expressed as an existential formula. □

Next, we move on to games with non-absentminded players. Since the complexity for one-payer games in this case is settled, we consider only the two-player case.

3.5 Proof of complexity: Games without absentmindedness

For one-player games, the complexity picture is already complete as can be seen in Fig. 3.7. So we consider the maxmin problem for 2 players with non-absentminded players. For a game G with non-absentminded players, the payoff polynomial f_G is of a special kind: they are *multilinear*. A polynomial is called *multilinear* if, in every term of the polynomial, the degree of a variable is at most 1. Payoff polynomials in games without absentmindedness are multi-linear since every observation appears not more than once on any path from root to a leaf. This is not the case for absentminded games in general[8]. Also, the $\exists\mathbb{R}$-complete problems involving polynomials such as QUAD involves non-multilinear polynomials. To the best of our knowledge no suitable $\exists\mathbb{R}$-complete problems involving multilinear polynomials are known so far. Hence the proof doesn't quite extend to non-absentminded games.

The maxmin problem for non-absentminded games was known to be NP-hard [KM92] even when Max has A-loss recall and Min has perfect recall [BP17]. However it was already demonstrated in [KM92] that in such games Max might

[8]It is possible that non-linear terms lead to zero payoff or cancel each other

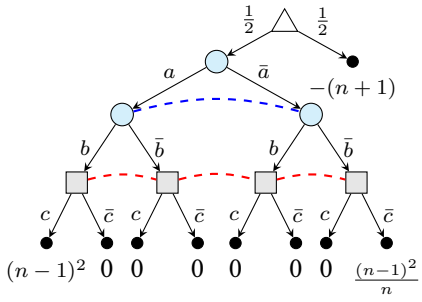


Figure 3.8: Game $G_{-\sqrt{n}}$

require irrational probabilities to achieve maxmin payoff.[9] This was also shown to be true for games with A-loss recall [CBL17]. This points towards the difficulty of demonstrating membership of this problem in NP, since optimal behavioral strategies cannot be used as short certificates. We substantiate this doubt further by showing that the maxmin problem is Sqrt-Sum-hard as well as co-NP-hard. The Sqrt-Sum lower bound holds even for games when Max has A-loss recall and Min has perfect recall. To show this we construct a general class of games where the maxmin payoff is irrational numbers[10]. More precisely for each $n \in N$ we construct a game $G_{-\sqrt{n}}$ with maxmin payoff $-\sqrt{n}$.

Before constructing $G_{-\sqrt{n}}$ we review some facts on the maxmin value of games.

Fact 1:If G is a game with maxmin value v, and G' be a game with the same underlying game tree as G but with the payoffs scaled by a positive number k, then the maxmin value of G' is vk.

Fact 2 : Let $G_1 \dots G_m$ be m games with maxmin values $v_1 \dots v_m$ respectively. G' be a new game with a chance node r as root node and m transitions out of it each leading to G_i probability $\frac{1}{m}$. Also, no pair of nodes u and v with u in G_i and v from G_j for $i \neq j$ have the same observation. Then the maxmin value of G' is $\frac{\sum_i v_i}{m}$.

Lemma 3.27. For every $n \in \mathbb{N}$, one can construct in PTIME a game $G_{-\sqrt{n}}$ whose maxmin value is $-\sqrt{n}$. The size of this game is polynomially bounded in $\log n$.

[9] In contrast to absentminded games, in one-player non-absentminded games Max always has pure optimal strategy. So the question of irrational probabilities doesn't arise for one-player case.

[10] We have constructed such games before in Proposition 3.17 but those were games with absentmindedness

Proof. We construct the game $G_{-\sqrt{n}}$ in three steps. In the first step a game G_1 is constructed whose maxmin value is $\frac{n(n+1-2\sqrt{n})}{(n-1)^2}$. By scaling the payoffs of G_1 with the positive number $\frac{(n-1)^2}{n}$ another game G_2 is constructed with maxmin value $n+1-2\sqrt{n}$. Then we take a trivial game (a terminal node) t with utility $-(n+1)$ and finally construct $G_{-\sqrt{n}}$ by taking a root vertex r as chance node and transitions with $1/2$ probability from r to G_2 and t.

We now describe the construction of the game G_1 and why it has maxmin value $\frac{n(n+1-2\sqrt{n})}{(n-1)^2}$. Fig. 3.8 depicts the whole game $G_{-\sqrt{n}}$ and the left sub-tree with payoffs scaled by $\frac{(n-1)^2}{n}$ depicts the game G_1. G_1 is a binary decision game. Its' game tree has 7 non-terminal nodes and 8 leaf nodes with payoffs. At the root node s_ϵ, there are 2 actions a and $\bar{a}$, playing which the game moves to s_0 or s_1. Then again at s_i the action b and $\bar{b}$ are available playing which the game can go to $s_{0,0}, s_{0,1}, s_{1,0}$ or $s_{1,1}$. And finally again playing action c or $\bar{c}$ the game can go to the leaf statesa$\{t_{i,j,k} | i,j,k \in \{0,1\}\}$. The nodes s_ϵ, s_0 and s_1 belongs to Max whereas the nodes $s_{i,j}$ for $i,j \in \{0,1\}$ belongs to Min. At s_ϵ Max observes o_a and at s_0 and s_1 she observes the same observation o_b. Min observes the same observation o_c at nodes $s_{0,0}, s_{0,1}, s_{1,0}$ and $s_{1,1}$. The utility at $t_{0,0,0}$ is n and the utlity at $t_{1,1,1}$ is 1. The utility at all other leaf nodes is 0.

Now we compute the maxmin value of G in terms of n. Since G_1 is a binary decision game, the maxmin value of G_1 is well expressed by its payoff polynomial. The payoff polynomial of G_1 is $nxyz + (1-x)(1-y)(1-z)$ where x, y, z corresponds to o_a, o_b, o_c respectively.

Hence the maxmin value is given by

$$\max_{x,y\in[0,1]} \min_{z\in[0,1]} nxyz + (1-x)(1-y)(1-z)$$

Since Min is not absentminded, for every strategy $(x,y) \in H_2$ of Max, Min has a pure response with $z \in \{0,1\}$. Hence the maxmin is given by:

$$\max_{x,y\in[0,1]} \min(nxy, (1-x)(1-y))$$

It turns out this value is attained when $nxy = (1-x)(1-y)$. We use this to get rid of y and finally, the maxmin reduces to:

$$\max_{x\in[0,1]} \frac{nx(1-x)}{1+(n-1)x}$$

It can be verified that the maximum of this expression in the domain $[0,1]$ is attained at $x = \frac{\sqrt{n}-1}{n-1}$. After evaluation we get $\mathrm{MaxMin}_{\mathrm{beh}}(G_1) = \frac{n(n+1-2\sqrt{n})}{(n-1)^2}$ as intended, at $x = y = \frac{\sqrt{n}-1}{n-1}$. This completes the construction of G_1.

As described earlier, using this we can construct the game $G_{-\sqrt{n}}$ as in Fig. 3.8.

The game has a constant number of leaf nodes and the terminal payoffs are all polynomially bounded in $\log n$. Hence this game has size $O(poly(\log(n))$. □

Theorem 3.28. *The problem* MaxMinBeh$_{\geq 0}$ *is* Sqrt-Sum*-hard . This holds even when* Max *has A-loss recall and* Min *has perfect recall.*

Proof. From the positive integers $a_1, ..., a_k$ and n which are the inputs to the Sqrt-Sum problem, we construct the following game $\hat{G}$. At the root, there is a chance node $\hat{r}$. From $\hat{r}$ there is a transition with probability $\frac{1}{m+1}$ to each of the games $G_{-\sqrt{a_i}}$ (as constructed in Lemma 3.27) and also a trivial game with payoff p. Now Max can guarantee a payoff 0 in $\hat{G}$ iff $\sum_{i=1}^{m} \sqrt{a_i} \leq p$.

Since in each of $G_{-\sqrt{n}}$, Max has A-loss recall and Min has perfect recall, the same holds in $\hat{G}$. Hence it is Sqrt-Sum-hard to decide the problem even when Max has A-loss recall and Min has perfect recall.

□

Next, we will show co-NP hardness of the problem MaxMinBeh$_{\geq 0}$. For this, similar to the absentminded case we will make use of the one-player problem MaxBeh$_{>0}$. We know that the one-player problem MaxBeh$_{\geq 0}$ without absentmindedness is NP-complete [KM92]. We will show that the strict the version of this problem is NP-complete as well. We use the same reduction from a 3-SAT formula Φ to a game instance G_Φ of MaxBeh$_{\geq 0}$ as used in [KM92]. It was shown there that Φ is satisfiable iff $\mathrm{Max}_{\mathrm{beh}}(G_\Phi) \geq 0$. For the same game, with the aid of a minute observation, we will show that Φ is satisfiable iff $\mathrm{Max}_{\mathrm{beh}}(G_\Phi) > \lambda$ for a constant λ depending on Φ. This approach is very similar in flavor to Lemma 3.22 used for the absentminded case.

Theorem 3.29. *The decision problem* MaxBeh$_{>0}$ *is* NP*-complete. This holds even when* Max *is non-absentminded.*

Proof. Since we are concerned with one-player games, we know from Lemma 2.5 that there exists a pure optimal strategy for Max. Hence this optimal strategy is a short witness for true instances of MaxBeh$_{>0}$ and this proves the membership of MaxBeh$_{>0}$ in NP.

Now for the NP-hardness, we reduce 3-SAT to MaxBeh$_{>0}$. Given a 3-SAT formula Φ with n clauses C_i for $i \in [n]$ over a set of m boolean variables $X = \{x_1, \ldots, x_m\}$, a one-player Max game G_Φ is constructed as follows: The root node r is a chance node from which there are n transitions each leading with probability $1/n$ to Max player node u_i for $i \in [n]$ corresponding to each clause C_i. At each u_i is rooted a sub-game, a binary tree of depth 3 (one level for each variable in

C_i), where at each level Max chooses a valuation for each variable. Formally for a clause $C_i = (y_1 \vee y_2 \vee y_3)$ with $y_i \in X$, $u_i = v_{y_1}^{C_i}$, with actions $(y_1, 0)$ and $(y_1, 1)$ leading to $v_{y_2}^{C_{i,y_1=b}}$ for $b \in \{0, 1\}$ respectively. The tree is extended in the same fashion with y_2 in the next level and then y_3 in the final level leading to leaf nodes. The leaf node $t^{C_{i,y_1=b_1,y_2=b_2,y_3=b_3}}$ for $b_i \in \{0, 1\}$ has payoff 0 if $b_1 \vee b_2 \vee b_3$ is true, otherwise the payoff is -1. For any variable $x_j \in X$, all the nodes of the form $v_{x_j}^{C_{k,\cdots}}$ have the same observation o_{x_j} and this is to ensure that Max chooses the same valuation for x_j in every clause.

For the game, G_Φ, we will show that $\mathrm{Max}_{\mathrm{beh}}(G_\Phi) + 1/n > 0$ iff the formula is satisfiable. Assuming we have shown this, we can construct a new game G'_Φ, with a chance node at the root node with two equiprobable actions, one leading to G_Φ and another leading to a trivial leaf node with payoff $1/n$. In this new game $\mathrm{Max}_{\mathrm{beh}}(G'_\Phi) > 0$ iff Φ is satisfiable. The size of G'_Φ is polynomial since n is the number of clauses and n can be written using $\log(n)$ bits. As a result, this will prove that MaxBeh$_{>0}$ is NP-hard.

Now we will proceed to show $\mathrm{Max}_{\mathrm{beh}}(G_\Phi) + 1/n > 0$ iff Φ is satisfiable Since we know that there is always a pure optimal strategy we will only deal with pure strategies. Any assignment of boolean values 0 or 1 to variables x_i gives a pure strategy in the game and vice versa. When Φ is satisfiable, it follows from the construction that for each clause C_i the sub-game at u_i bags a payoff of 0 which proves the backward direction.

For the forward direction, if the formula Φ is not satisfiable, then for any assignment of values, at least one clause bags a payoff -1 contributing $-1/n$ ($1/n$ from chance probability) to the total payoff. So even assuming that all the other clauses bags the maximum possible payoff of 0, the total payoff cannot exceed $-1/n$. Hence Φ is not satisfiable implies $\mathrm{Max}_{\mathrm{beh}}(G_\Phi) \leq -1/n$. This completes the proof. □

Corollary 3.30. MinBeh$_{\geq 0}$ is co-NP complete. This holds even when Min is non-absentminded.

Proof. It follows from Lemma 3.24 that MinBeh$_{<0}$ reduces to MaxBeh$_{>0}$ in PTIME. Hence MinBeh$_{\geq 0}$, the complement problem of MinBeh$_{<0}$ is co-NP-complete. □

Corollary 3.31. MaxMinBeh$_{\geq 0}$ is co-NP hard. This holds even when Max is non-existent or has trivial moves and Min is non-absentminded.

Proof. Since MinBeh$_{\geq 0}$ is a special case of MaxMinBeh$_{\geq 0}$, the problem MaxMinBeh$_{\geq 0}$ is co-NP hard. □

The complexity of MaxMinBeh$_{\geq 0}$ for two-players can be summarized by the table in Fig. 3.9.

Recall of Max	*Recall of* Min	*Complexity*
Perfect Recall	A-loss recall	P [KM92] [KK95][Kli02]
A-loss Recall	Non-absentminded	NP-hard [BP17], Sqrt-Sum-hard (Theorem 3.28), co-NP-hard (Corollary 3.31), $\exists\mathbb{R}$ (Theorem 3.20)
General	A-loss recall	$\exists\mathbb{R}$-Complete (Theorem 3.20)
General	General	$\exists\mathbb{R}$-hard, $\forall\mathbb{R}$-hard, $\exists\forall\mathbb{R}$ (Theorem 3.18) (Theorem 3.25)

Figure 3.9: Complexity of MaxMinBeh$_{\geq 0}$

3.6 Conclusion

In this chapter, we provided several new complexity bounds for the maxmin problem both for the two-player case and one-player case. The complexity results are summarized in Fig. 3.3.

Even though we settle the complexity of the one-player case for the known classes of recalls, the exact complexity of the two-player case remains open.

Open Question 3.6.1

Is MaxMinBeh$_{\geq 0}$ $\exists\forall\mathbb{R}$-hard ?

Also, for the non-absentminded, there is a bigger gap between the lower and the upper bounds. The $\exists\forall\mathbb{R}$ upper-bound comes from the general case. Since for the case with trivial Max player against non-absentminded Min, we show co-NP hardness, we think the next natural class to investigate would be Max with perfect recall against non-absentminded Min player.

The linear program in Section 2.7.1 can be extended to the case when Max has perfect recall and Min is non-absentminded as suggested in [CBHL18] but with an exponential blowup in the number of constraints for best response computation. Unlike for the case with A-loss recall where Max needs irrational probabilities in a maxmin optimal behavioral strategy, Max with perfect recall will always have such a strategy with rational probabilities. But it is still unclear if Max can avoid exponential blow-up in the probabilities for optimal payoff. In case this can be done, the problem can be shown to lie in Σ_2.

Open Question 3.6.2

When Max has perfect recall and Min is non-absentminded is MaxMinBeh$_{\geq 0}$ in Σ_2 ?

Also, with the co-NP hardness lower bound, it would also be interesting to check if for non-absentminded games, MaxMinBeh$_{\geq 0}$ is Σ_2 hard.

Chapter 4

Simplifying non-absentminded games

In this chapter we identify a new class of one-player non-absentminded games called games with *A-loss recall shuffle*. This class is an extension of the class of A-loss recall games. We know that one player A-loss recall games can be solved in PTIME. We also know that in two player games when Max has perfect recall and Min has A-loss recall, the maxmin value can be computed in PTIME as well. In this chapter we show that any game of this larger class can be solved by simplifying it into an A-loss recall game of same size which we call an *A-loss recall shuffle*. As a result one player games of this class can be solved in PTIME. As a consequence we also obtain a polynomial time algorithm for computing the maxmin value in two player games where Max has perfect recall and Min has A-loss recall shuffle. However unlike A-loss recall and perfect recall, given a game it is not immediate to check from the definition if a player has A-loss recall shuffle. We also provide an efficient procedure to identify games with A-loss recall shuffle.

We also generalize the notion of A-loss recall shuffle to *A-loss recall span* and show that all games have an A-loss recall span. Games not having A-loss recall shuffle can be solved by simplifying them to an A-loss recall game but with possible blowup in size which we call an *A-loss recall span*. We give an algorithm to compute the size of the smallest A-loss recall span but with exponential worst-case runtime. Using the size of the smallest A-loss recall span as a parameter we provide an algorithm for computing maxmin value. We also give games for which the smallest A-loss recall span is exponential in the original game size. We show that the decision problem related to computing smallest A-loss recall span is in NP.

We also discuss limitations of using the same idea of span for perfect recall

and show that it doesn't work similarly. Finally we discuss another heuristic of game simplification using the payoff polynomial. We show how to build a perfect recall game from a class of payoff polynomials and give a characterization of such games. We also demonstrate a side application of this in multi-linear optimization.

Contents

In this chapter we take up the task of simplifying games with non-absentminded players via *game equivalence*. For establishing this equivalence we re-examine extensive form games in their *sequence form* representation and use payoff polynomials associated to them. In Chapter 3 we provided complexity bounds for the maxmin decision problem for both one player and two player cases, which pointed to the difficulty of computing optimal strategies for some classes of games. Here we further refine the complexity picture but towards a positive direction. We identify a new fragment of non-absentminded games called *Aloss recall shuffle*, for which the optimal payoff and the optimal strategy in the one player version can be computed in PTIME. We will briefly recall the complexity picture from the last chapter and give a preview of our results to appear in this chapter.

4.1 Complexity Picture

The complexity of the maxmin decision problems that we have seen so far can be summarized in a compact manner in Fig. 4.1. Recall that the complexity lower

bounds (hardness) carry forward recursively to the right and downwards in the table with decreasing degree of recall of either players, whereas complexity upper-bounds (in bold text) carry forward to the left and upwards.

<table>
<tr><th>Max \ Min</th><th>Non-existent</th><th>Perfect Recall</th><th>A-loss Recall</th><th>N.A.M</th><th>General</th></tr>
<tr><td>Non-existent</td><td></td><td colspan="2"></td><td>co-NP-hard, co-NP</td><td>$\forall\mathbb{R}$-hard, $\forall\mathbb{R}$</td></tr>
<tr><td>Perfect Recall</td><td colspan="3">PTIME</td><td></td><td></td></tr>
<tr><td>A-loss Recall</td><td>PTIME</td><td>NP-hard SQRT-SUM-hard</td><td></td><td></td><td></td></tr>
<tr><td>N.A.M</td><td>NP-hard, NP</td><td></td><td></td><td></td><td></td></tr>
<tr><td>General</td><td>$\exists\mathbb{R}$-hard</td><td colspan="2">$\exists\mathbb{R}$</td><td></td><td>$\exists\forall\mathbb{R}$</td></tr>
</table>

Figure 4.1: Complexity of MAXMINBEH$_{\geq 0}$ so far

Next we will state the contribution of the current chapter to this complexity picture. For this we will define the class *A-loss recall shuffle*. The class *A-loss recall shuffle* heavily depends on the terminal sequences of a game and the interplay between them. Hence we will review a well-known representation of extensive form games called *sequence form* [Ste96].

Sequence Form

In the sequence form representation, instead of a game tree, the game is defined by the set of all terminal histories. Each terminal history comes with a utility augmented with the chance probability to follow this history.

Recall that a binary decision game is a game where at every node there are two actions, i.e. for every observation $o \in \mathcal{O}$, $\mathrm{Act}(o) = 2$. In this chapter we will restrict our discussion to binary decision games. We will see later in Appendix A why it is enough to consider binary decision games for our purpose. As stated earlier we also restrict our discussion to non-absentminded games.

Let A_{Max} and A_{Min} be the set of actions of Max and Min respectively. Recall that in binary decision games we use the notation a and $\bar{a}$ for denoting two actions at the same node. In this chapter we will always assume that for two actions of the form a and $\bar{a}$, there is some unique observation o such that $\mathrm{Act}(o) = \{a, \bar{a}\}$. Let $A = A_{\mathsf{Max}} \cup A_{\mathsf{Min}}$ be the set of all actions and let A^* be the set of all finite sequences over A. For action $a \in A$ and a sequence of actions $s \in A^*$, let $a \in s$ signify that action a is present in sequence s. Since we deal with non-absentminded games we consider sequences s over A such that

1. Any action in A appears at most once in s

2. s doesn't contain both a and $\bar{a}$ for any a

For $A' \subseteq A$, let $Seq(A')$ be the set of sequences over A' satisfying these two conditions. A *sequence set* S over A' is a subset of $Seq(A')$. The set of terminal histories of G form a sequence set over the set of actions.

However if we start from an action set A, an arbitrary sequence set $S \subseteq Seq(A)$ doesn't necessarily correspond to some extensive form game. For example for $A = \{a, \bar{a}, b, \bar{b}\}$, the sequence set $S = \{ab, \bar{a}\bar{b}\}$ does not correspond to any extensive form game with action set A. This is because, if we assume that there is a corresponding extensive form game, then after history a, at a node where action b is played, the action $\bar{b}$ is also present leading to some other leaf with distinct terminal history. But this history is not present in S. We call sequence sets that correspond to an extensive form game, a *rulebook*.

In an extensive form game G with set of terminal nodes L, let $\mathcal{H}_L = \{hist(t) \mid t \in L\}$ be the set of all terminal histories of G.

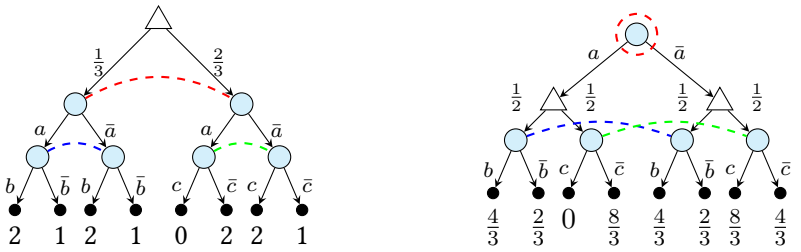
Definition 21 (Rulebook)**.** A sequence set $S \subseteq Seq(A)$ is called a rulebook, if there is a game G with action set A such that $S = \mathcal{H}_L$.

Now we can represent a game in sequence form by starting with a rulebook and assigning utilities augmented with chance probabilities to every action sequence in the rulebook.

Definition 22 (Sequence form)**.** A game G in extensive form is represented in the *sequence form* given as (S, Λ) where $S = \mathcal{H}_L$ is a rulebook and $\Lambda : S \mapsto \mathbb{Q}$ is the augmented payoff function given by $\Lambda(s) = \sum_{t|hist(t)=s} \mathcal{U}(t)\mathcal{P}_{\mathsf{Chance}}(t)$.

Example 4.1. In Fig. 4.2 both the games have the same sequence form representation. The rulebook is given by $S = \{ab, a\bar{b}, \bar{a}b, \bar{a}\bar{b}, ac, a\bar{c}, \bar{a}c, \bar{a}\bar{c}\}$ and the augmented payoffs Λ is given by $\{ab : \frac{2}{3},\ a\bar{b} : \frac{1}{3},\ \bar{a}b : \frac{2}{3},\ \bar{a}\bar{b} : \frac{1}{3},\ ac : 0,\ a\bar{c} : \frac{4}{3},\ \bar{a}c : \frac{4}{3},\ \bar{a}\bar{c} : \frac{2}{3}\}$. For the sequence ab, $\Lambda(ab) = \frac{2}{3}$ since in the game in Fig. 4.2a this is $\frac{1}{3}.2$ and in the game in Fig. 4.2b this is $\frac{1}{2}.\frac{4}{3}$.

Remark 4.1. We don't take into account the chance nodes and edges out of them in the sequence form representation. Rather the product of chance probabilities $\mathcal{P}_{\mathsf{Chance}}(t)$ at a leaf node t is absorbed into the augmented payoff $\Lambda(t)$. This doesn't necessarily give a unique game but with respect to sequence form representation it gives a class of games which will be sufficient for our purpose. This is exactly the case for the two games with same sequence form in Fig. 4.2. So essentially by a slight abuse of notation by (S, Λ) we denote a set of games. We will see later in Proposition 4.9 how to construct an extensive form game from a rulebook. In fact that construction will give the game on the right in Fig. 4.2.



(a) (b)

Figure 4.2: Sequence Form: Both the games above have the same sequence form with rulebook $S = \{ab, a\bar{b}, \bar{a}b, \bar{a}\bar{b}, ac, a\bar{c}, \bar{a}c, \bar{a}\bar{c}\}$

Recalls in Sequence Form

We will redefine A-loss recall and perfect recall with respect to sequence form. We will not restrict ourselves to rulebooks and define these recalls for arbitrary sequence sets.

For a sequence $s \in A$, let s_{Max} be the sequence restricted to actions from A_{Max}. For $S \subseteq Seq(A)$, $S_{\mathsf{Max}} = \{s_{\mathsf{Max}} \mid s \in S\}$ be the set of sequences in S restricted to actions in A_{Max}. For actions a and b let $(a + b)$ denote the set $\{a, b\}$. For sequences s_1, s_2 and actions $a, b \in A$, let $s_1(a + b)s_2$ denote the set $\{s_1as_2, s_1bs_2\}$. In general, for sequence sets S_1, S_2 and actions $a, b \in A$, let $S_1(a + b)S_2 = \{s_1as_2 \mid s_1 \in S_1, s_2 \in S_2\} \cup \{s_1bs_2 \mid s_1 \in S_1, s_2 \in S_2\}$.

Let s_{Min} and S_{Min} be defined similarly. We give definitions with respect to player Max, but they can be extended to Min in a similar manner.

Definition 23. (Perfect Recall in sequence form)

Let $S \subseteq Seq(A)$ be a sequence set and s_1, s_2 be two sequences in $Seq(A_{\mathsf{Max}})$. We say s_1 is *perfect recall equivalent* to s_2, denoted by $s_1 \sim_P s_2$ if $\forall s_1', s_2' \in Seq(A_{\mathsf{Max}})$ and $a, \bar{a} \in A_{\mathsf{Max}}$ such that $s_1 \in s_1'(a + \bar{a})Seq(A_{\mathsf{Max}})$ and $s_2 \in s_2'(a + \bar{a})Seq(A_{\mathsf{Max}})$, we have $s_1' = s_2'$.

S_{Max} is said to have perfect recall if $\forall s_1, s_2 \in S, s_1 \sim_P s_2$.

Definition 24. (A-loss recall in sequence form)

Let $S \subseteq Seq(A)$ be a sequence set and s_1, s_2 be two sequences in $Seq(A_{\mathsf{Max}})$. We say s_1 is *A-loss recall equivalent* to s_2, denoted by $s_1 \sim_A s_2$ if

$\forall s_1', s_2' \in Seq(A_{\mathsf{Max}})$ and $a, \bar{a} \in A_{\mathsf{Max}}$ such that $s_1 \in s_1'(a + \bar{a})Seq(A_{\mathsf{Max}})$ and $s_2 \in s_2'(a + \bar{a})Seq(A_{\mathsf{Max}})$ at least of one of the following is true:

(i) $s_1' = s_2'$

(ii) $\exists\ s' \in Seq(A_{\mathsf{Max}})$, $b, \bar{b} \in A_{\mathsf{Max}}$, such that $s_1' = s'b_1s_1''$ and $s_2' = s'b_2s_2''$ for some $s_1'', s_2'' \in Seq(A_{\mathsf{Max}})$ and $b_i \in \{b, \bar{b}\}$ with $b_1 \neq b_2$.

S_{Max} is said to have A-loss recall if $\forall s_1, s_2 \in S, s_1 \sim_A s_2$.

Observe that from the first condition in the above definition it follows that A-loss recall equivalence implies perfect recall equivalence.

Example 4.2. Let $S_1 = \{abc, a\bar{b}\}, S_2 = \{abc, \bar{a}c\}, S_3 = \{ba, ca\}$ be sequence sets over $A_{\mathsf{Max}} = \{a, \bar{a}, b, \bar{b}, c, \bar{c}\}$. S_1 has perfect recall but S_2 doesn't. On the other hand S_2 has A-loss recall but S_3 doesn't.

It can be verified that the above definitions of recalls are equivalent to their corresponding definitions in the extensive form.

Proposition 4.2. Let G be a game in extensive form and (S, Λ) be representation of G in the sequence form. Then Max (Min) has A-loss recall (perfect recall) in G iff S_{Max} (S_{Min}) has A-loss recall (perfect recall).

4.1.1 Our contribution

First we define A-loss recall shuffle and then give a preview of our primary contribution in this chapter to the complexity picture.

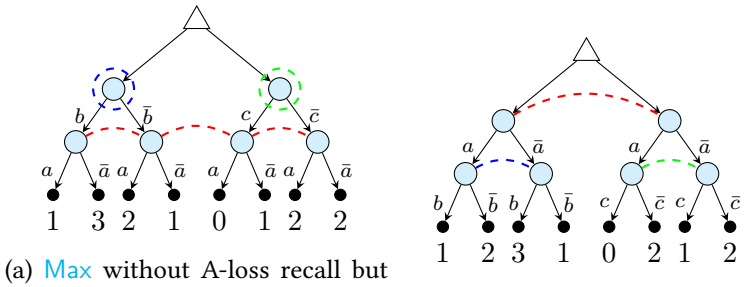
For an action set A and a sequence of actions $s = a_1 \dots a_n$ with $a_i \in A$, a *permutation* $s^\dagger$ of s is a sequence over A given by $s^\dagger = \rho(a_1) \dots \rho(a_n)$ where $\rho : \{a_1 \dots a_n\} \mapsto \{a_1 \dots a_n\}$ is a bijective function.

Definition 25 (Shuffle)**.** Given two sets $S, S' \subseteq Seq\ (A_{\mathsf{Max}})$, S and S' are called *shuffles* of each other, if for every $s \in S$, there exists $s' \in S'$ such that s' is a permutation of s and vice versa.

Definition 26 (A-loss recall shuffle)**.** A sequence set $S \subseteq Seq\ (A_{\mathsf{Max}})$ is said to have *A-loss recall shuffle* if there exists $S^\dagger \subseteq Seq\ (A_{\mathsf{Max}})$ such that $S^\dagger$ is a shuffle of S and $S^\dagger$ has A-loss recall. $S^\dagger$ is called an A-loss recall shuffle of S.

In a game $G = (S, \Lambda)$ in sequence form with rulebook S, Max is said to have A-loss recall shuffle if S_{Max} has an A-loss recall shuffle $S^\dagger$.

Example 4.3. In Fig. 4.3 the game on the left has the sequence set $S = \{ba, b\bar{a}, \bar{b}a, \bar{b}\bar{a}, ca, c\bar{a}, \bar{c}a, \bar{c}\bar{a}\}$ and doesn't have A-loss recall. This is because the sequences ba and ca are not A-loss recall equivalent. However this game has A-loss recall shuffle and the evidence to this is the game on the right. The game on the right has sequence set $S' = \{ab, a\bar{b}, \bar{a}b, , \bar{a}\bar{b}, ac, a\bar{c}, \bar{a}c, \bar{a}\bar{c}\}$ which has A-loss recall. S' is a shuffle of S since for any sequence in S, a permutation of that sequence is present in S'.



(a) Max without A-loss recall but with A-loss recall shuffle

(b) Max with A-loss recall

Figure 4.3: A-loss recall shuffle

Max \ Min	Non-existent	Perfect Recall	A-loss Recall	ALR Shuffle	N.A.M	General
Non-existent					co-NP-hard, **co-NP**	$\forall\mathbb{R}$-hard, $\forall\mathbb{R}$
Perfect Recall	**PTIME**					
A-loss Recall	**PTIME**	NP-hard SQRT-SUM-hard				
N.A.M	NP-hard, **NP**					
General	$\exists\mathbb{R}$-hard	$\exists\mathbb{R}$				$\exists\forall\mathbb{R}$

Figure 4.4: Complexity of MAXMINBEH$_{\geq 0}$ w.r.t A-loss recall(ALR) shuffle

In a game if Max has A-loss recall, then she also has A-loss recall shuffle. Hence the class of games where Max has A-loss recall shuffle subsumes the class of games where Max has A-loss recall. Also, we will see later that sometimes Max can be non-absentminded but still not have A-loss recall shuffle. This implies A-loss recall shuffle is a strict extension of A-loss recall and a strict sub-class of non-absentmindedness.

Our finer complexity analysis in terms of A-loss recall shuffle is given in Fig. 4.4.

Our primary contribution consists of extending all complexity bounds from the class A-loss recall to A-loss recall shuffle. For the class of one-player Max games with A-loss recall shuffle the maxmin decision problem i.e. MAXBEH$_{\geq 0}$ is now in PTIME. For two player games when Max has perfect recall and Min has A-loss recall shuffle the maxmin decision problem can be solved in PTIME.

To view this in the light of degree of recall, A-loss recall shuffle may appear to have less degree of recall compared to just A-loss recall but in terms computation it doesn't make much difference.

We also provide a generalization of the class A-loss recall called A-loss recall span.

4.2 Why A-loss recall shuffle? Simplification via sequences

In this section we will see that having A-loss recall shuffle can simplify computation of the maxmin value. To show this, we first define a notion of strategic equivalence between games.

4.2.1 Strategic equivalence of games

Let $\mathcal{S}_i(G)$ be the set of all behavioral strategies of player i in the game G.

Definition 27 (Game Equivalence)**.** Two games G and G' given in extensive form are said to be strategically equivalent denoted by $G \sim G'$ if for each player i there is a bijective function $\phi_i : \mathcal{S}_i(G) \to \mathcal{S}_i(G')$ such that $\forall\, \sigma_{\mathsf{Max}} \in \mathcal{S}_{\mathsf{Max}}$, $\forall\, \sigma_{\mathsf{Min}} \in \mathcal{S}_{\mathsf{Min}}$

$$\mathbb{E}[\sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}}] = \mathbb{E}[\phi_{\mathsf{Max}}(\sigma_{\mathsf{Max}}), \phi_{\mathsf{Min}}(\sigma_{\mathsf{Min}})]$$

Strategic equivalence can be used as a tool for simplifying games. When a game G has an equivalent simpler game G', with the aid of an efficiently computable bijective mapping ϕ, computing optimal strategies in G' would be enough to solve G. In essence maxmin strategies of one game can be mapped to that of the equivalent game.

Proposition 4.3. If for two game G and G' we have $G \sim G'$ which is witnessed by the maps $\phi_i : \mathcal{S}_i(G) \to \mathcal{S}_i(G')\ \forall i \in \{\mathsf{Max}, \mathsf{Min}\}$ then the following statements are true:

- $\mathrm{MaxMin}_{\mathrm{beh}}(G) = \mathrm{MaxMin}_{\mathrm{beh}}(G')$.
- σ is a maxmin strategy in G iff $\phi_{\mathsf{Max}}(\sigma)$ is a maxmin strategy in G'

Proof. The proof simply follows from the definition of strategic equivalence. □

Remark 4.4. Since we are concerned with computation over behavioral strategies we restrict ourselves to strategic equivalence with respect to behavioral strategies.

Recall that to each binary decision game with action set A, we associate a payoff polynomial which is formed from the probability variables assigned to each action. When the variables associated to actions are not explicitly mentioned, for an action a we assign the variable x_a and for $\bar{a}$ we assign the variable $1 - x_a$. Sometimes we use $\bar{x}$ to denote $1 - x$.

We observe that payoff polynomials can actually determine game equivalence.

Proposition 4.5. Let G and G' be two binary decision games with the same action set A. If the payoff polynomials $f_G(X, Y)$ and $f'_G(X, Y)$ are equal then $G \sim G'$.

Proof. Consider the function ϕ_i which maps a strategy σ in G to a strategy σ' in G' where for every $o \in \mathcal{O}_i$, $a \in \mathrm{Act}(o)$, $\sigma(o, a) = \sigma'(o, a)$. Since the payoff of a game is given by evaluating the payoff polynomial, we have $\mathbb{E}[\sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}}] = \mathbb{E}[\phi_{\mathsf{Max}}(\sigma_{\mathsf{Max}}), \phi_{\mathsf{Min}}(\sigma_{\mathsf{Min}})]$. Hence $G \sim G'$. □

Example 4.4. In Fig. 4.3 associating the variables x, y and z to actions a, b and c respectively, we obtain the polynomial $f(x, y, z) = xy - xz + 2x + y - z + 5$ for both the games. Hence these two games are equivalent.

Remark 4.6. The converse of Proposition 4.5 doesn't hold in general. Two equivalent games G and G' over the same action set A can have different payoff polynomials. Recall that in Remark 3.13 we saw two games in Fig. 3.5 with same action set but different payoff polynomials. Since the labels b and $\bar{b}$ are swapped, these two games are still equivalent.

The payoff polynomial of a game can be seen as a linear combination over individual terms induced at each leaf. The individual terms depend on the terminal histories, hence we define these terms for individual sequences of actions.

Definition 28 (Leaf monomial)**.** For a sequence set $S \subseteq Seq(A)$ and a sequence $s \in S$, the monomial $\mu(s)$ is the product of all variables assigned to actions in s. For a set S, let $\mu(S) = \{\mu(s) \mid s \in S\}$ be the set of leaf monomials of sequences in S.

Note that from definition, for the empty sequence ϵ, $\mu(\epsilon) = 1$.
In an extensive form game, for a leaf node t we will often abuse notation by using $\mu(t)$ in order to denote $\mu(\mathit{hist}(t))$. The payoff polynomial can then be expressed as $\sum_{t \in T} \mathcal{P}_{\mathsf{Chance}}(t)\mathcal{U}(t)\mu(t)$ which in the sequence form translates to $\sum_{s \in S} \Lambda(s)\mu(s)$.

Example 4.5. In Fig. 4.3 if we assign the variables x, y and z to actions a, b and c in either of the games, the bottom left-most leaf has monomial xy whereas the bottom right-most leaf has monomial $\bar{x}\bar{z}$. Even though the sequences at these leaves are distinct in the two games, the corresponding monomials at a leaf are the same.

Now we will see what A-loss recall shuffle implies for binary decision games. We first define the size of a game in sequence form.

Definition 29 (Size of a game in sequence form)**.** The size of a game $G = (S, \Lambda)$ in sequence form denoted by $|G|$ is the sum of the bit-lengths of $\Lambda(s)$ of all s in S[1].

[1]This definition is the same as that of extensive form.

We deal with arbitrary sequence sets that do not necessarily come from any extensive form game. So we will also define the size of sequence sets.

Definition 30 (Size of sequence set)**.** The *size* of a sequence set S, denoted by $|S|$, is the number of sequences in S.[2]

Theorem 4.7. *Let G be a one-player game where Max has A-loss recall shuffle. Then G has an equivalent game G' of size $O(|G||A|)$ where Max has A-loss recall. As a consequence, the optimal value in G can be computed in PTIME.*

Theorem 4.8. *Let G be a two player game where Max has perfect recall and Min has A-loss recall shuffle. Then G has an equivalent game G' of size $O(|G||A|)$ where Max has perfect recall and Min has A-loss recall. As a consequence, the maxmin value in G can be computed in PTIME.*

We will shortly prove the above two theorems. Since maxmin computation in one-player games is a special case of maxmin computation in two-player games, it follows that Theorem 4.8 implies Theorem 4.7. We will still prove the two theorems separately. We will construct the equivalent game G' in both the cases provided we have the A-loss recall shuffle. Note that any sequence set is not a rulebook and hence an A-loss recall shuffle is not a priori a rulebook either. We will first see how to get a rulebook from any sequence set via a characterization of rulebooks. Since our construction of the game G' uses the rulebook, we will then see how to construct an extensive form game from a rulebook.

We observe a property of rulebooks called completeness and then show that this condition is sufficient for a sequence set to be a rulebook.

Definition 31 (Complete set)**.** A set of sequences $S \subseteq Seq(A)$ is called complete if for every sequence of the form $sas' \in S$ where $s, s' \in Seq(A)$ and $a \in A$, there exists a sequence $s\bar{a}s'' \in S$ for some $s'' \in Seq(A)$.

From definition a rulebook is already complete. In an extensive form game, at a node v with some history s, all actions in $\mathrm{Act}(\mathrm{Obs}(v))$ at v lead to some terminal history.

Example 4.6. The sequence set $S_1 = \{a, \bar{a}b, \bar{a}\bar{b}\}$ is a complete set. On the other hand the set $S_2 = \{ab, \bar{a}\bar{b}\}$ is not complete since for the prefix a, the sequence $ab \in S_2$ but no sequence exists in S_2 starting with $a\bar{b}$.

Proposition 4.9 (Rulebook to Game)**.** A set of sequences $S \subseteq Seq(A)$ is a rulebook iff S is complete.

[2]The length of sequences in S also matters in the descriptive size of S but since the length of a sequence is at most $|A|$ we ignore this $|A|$ factor in the definition of size without much loss in the complexity.

Proof. Consider a sequence sas' in a rulebook S. For some vertex v, $\mathit{hist}(v) = s$ and let $\mathrm{Act}(v) = \{a, \bar{a}\}$. Hence the action $\bar{a}$ from v will lead to some leaf l such that $\mathit{hist}(l) = s\bar{a}s''$ for some s''. Hence a rulebook is complete.

For the other direction, let $S \subseteq Seq(A)$ be a sequence set which is complete. We will construct an extensive form game G_S with $\mathcal{H}_L = S$. Since this is a property of the sequence set, we will only give the game tree without the utility function. Any assignment of utility to leaves will give a complete extensive form game corresponding to the sequence set S.

First we construct the game tree with actions labelled from A. The actions will determine the observations of a node, i.e. any node with action a and $\bar{a}$ out of it will have observation o_a.

We do this inductively on the size of S. When $S = \{\epsilon\}$, the corresponding game G_S is the trivial game with a single leaf node. Now let's assume for all complete set S of size m, it holds that S is the rulebook of game G_S. Consider a complete set S of size $m + 1$. Let $A_S = \{a \in A \mid as' \in S\}$ and for $a \in A_S$ let $S_a = \{s' \mid as' \in S\}$. Note that since S is complete $a \in A_S$ if and only if $\bar{a} \in A_S$. Also, it can be seen that for each $a \in A_S$, S_a and $S_{\bar{a}}$ are complete sets with size $\leq m$. By induction hypothesis for each of these one can construct corresponding games G_{S_a} and $G_{S_{\bar{a}}}$. Now there are two cases, either $|A_S| = 2$ or $|A_S| > 2$.

Case 1: When $|A_S| = 2$ i,e. $A_S = \{a, \bar{a}\}$, we construct the game G_S as follows. At the root node we have a player node v with two action a and $\bar{a}$ leading to G_{S_a} and $G_{S_{\bar{a}}}$ respectively.

Case 2: When $|A_S| > 2$ then $A_S = \{b_1, \bar{b}_1 \ldots b_k, \bar{b}_k\}$. In this case we have at the root a chance node c with k outgoing edges each with probability $1/k$. $\forall b_i$ there is an edge from c to a node v_i. From v_i there are actions b_i and $\bar{b}_i$ leading to $G_{S_{b_i}}$ and $G_{S_{\bar{b}_i}}$ respectively.

The control of a node v as well as the observation is determined the action labels out of v. This completes the construction of the game. □

Corollary 4.10. Given a game $G = (S, \Lambda)$ in its sequence form, a game G in extensive form with sequence set S can be constructed in PTIME.

Proof. Given S we construct the complete game tree following the procedure described in proof of Proposition 4.9. All that is left to do is to assign the utility to leaves. For any leaf t, the utility is $\Lambda(t)$ factored by the product of the chance probabilities in the constructed game. □

Remark 4.11. If a sequence set S over $A = A_{\mathsf{Max}} \cup A_{\mathsf{Min}}$ is complete then using the arguments in the proof of Proposition 4.9 we can show that S_{Max} and S_{Min} are also complete sets over A_{Max} and A_{Min} respectively. The converse is however not

true. Take for example the sequence set $S = \{ab, \bar{a}\bar{b}\}$ over $A_{\mathsf{Max}} = \{a, \bar{a}\}$ and $A_{\mathsf{Min}} = \{b, \bar{b}\}$. S is not complete since it contains no sequence starting with $a\bar{b}$.

We saw how to construct a game in extensive form from a rulebook. Now what about the case when a sequence set is not complete. Can we start from an incomplete sequence set and extend it to a rulebook without blowing up the size of the final sequence set by too much? It turns out this can be achieved simply by extending S with the addition of only a single action to every sequence contradicting completeness. We call this set the *closure*.

Definition 32 (Closure of a set of sequences)**.** For a sequence set $S \subseteq Seq(A)$ the closure of S denoted by $cl(S)$ is defined as $cl(S) = S \cup \{s\bar{a} \mid sas' \in S, s\bar{a}A^* \cap S = \emptyset\} \cup \{sa \mid s\bar{a}s' \in S, saA^* \cap S = \emptyset\}$

Observe that when $S_1 = \{a\}$ and $S_2 = \{\bar{a}\}$ for some $a, \bar{a} \in A$, $cl(S_1) = cl(S_2) = \{a, \bar{a}\} = (a + \bar{a})$. For a and $\bar{a}$ we will often use the notation $cl(a)$ and $cl(\bar{a})$ to denote $(a + \bar{a})$.

Example 4.7. For the incomplete set $S = \{ab, \bar{b}\}$, the closure is given by $cl(S) = \{ab, \bar{a}, a\bar{b}, b, \bar{b}\}$.

Lemma 4.12. For a set $S \subseteq Seq(A)$, $cl(S)$ is complete and $|cl(S)|$ is $O(|A||S|)$.

Proof. The completeness of $cl(S)$ follows from the definition of closure.

Since for every $s \in S$ we are adding at most $|s|$ sequence in $cl(S)$ it follows that $|cl(S)| \leq |A||S|$. Hence size of $cl(S)$ is $O(|A||S|)$. □

Another important thing to check is if properties like A-loss recall or perfect recall are preserved under the closure. For this we list out certain properties with respect to preservation of recall properties.

Lemma 4.13. Let $S \subseteq Seq(A_{\mathsf{Max}})$ be a sequence set of Max and for $sas' \in S$, let $S' = S \cup \{s\bar{a}\}$. Then the following is true:

1. S has A-loss recall $\implies$ S' has A-loss recall.
2. S has perfect recall $\implies$ S' has perfect recall.

Proof. For the first part, since S has A-loss recall, then for S' to not have A-loss recall, there must exist some $s_0 \in S$ such that $s_0 \not\sim_A s\bar{a}$. Observe that $\sim_A$ is closed under prefixes. That is any proper prefix of s_0 and sas' are A-loss compatible. This means $s_0 \sim_A sa$. Hence $s_0 \sim_A s(a + \bar{a})$, more particularly $s_0 \sim_A s\bar{a}$ which is a contradiction. Hence S' has A-loss recall.

Since $\sim_P$ is also closed under prefixes the same holds for perfect recall. □

Corollary 4.14. If a set $S \subseteq Seq(A_{Max})$ has A-loss (perfect) recall, so does $cl(S)$.

Proof. The closure $cl(S)$ is the addition of sequences of the form $s\bar{a}$ to S. Since A-loss (perfect) recall is preserved after every addition according to Lemma 4.13, this follows. □

Now we are ready to prove Theorem 4.7.

Proof of Theorem 4.7. Let $G = (S, \Lambda)$ be a one player game where S has an A-loss recall shuffle $S^\dagger$. Let $S' = cl(S^\dagger)$. We will give the equivalent game $G' = (S', \Lambda')$ by defining the augmented utility function Λ'. For every newly added sequence in the closure the utility is 0 i.e., for $s \in S' \setminus S^\dagger$, $\Lambda'(s) = 0$. And for sequences already in $S^\dagger$ we set the utility to be same as its permutation in S i.e., for $s^\dagger \in S^\dagger$, $\Lambda'(s^\dagger) = \Lambda(s)$ where $s^\dagger$ is the permutation of s.

From Lemma 4.12 it follows that S' is a rulebook of size at most $|A||S|$. Also Corollary 4.14 tells us that S' has A-loss recall. We just need to show G and G' are equivalent. Observe that for a sequence s and its permutation $s^\dagger$, $\mu(s) = \mu(s^\dagger)$. Hence $\mu(S) = \mu(S^\dagger)$. The payoff polynomial of G' is $f'_G = \sum_{s \in S'} \Lambda'(s)\mu(s) = \sum_{s \in S} \Lambda(s)\mu(s) = f_G$. This is because $\Lambda'(s) = 0$ for newly added sequences. Since the payoff polynomial of G and G' are the same, it follows from Proposition 4.5 that G and G' are equivalent. Now since $|S'| = |A||S|$ and Λ' assigns 0 to sequences in $S' \setminus S^\dagger$, we have $|G'| \leq |G| + |S' \setminus S^\dagger|$ which is $O(|G||A|)$. This completes the proof.

□

For example in Fig. 4.3 the game on the left doesn't have A-loss recall. However the game on the right is an equivalent game with A-loss recall.

Remark 4.15. By careful analysis, the size of the equivalent game G' in Theorem 4.7 can be improved to $|G|$. This is because we show later in Lemma 4.27 that A-loss recall shuffles of complete sets are also complete. Hence for the A-loss recall shuffle $S^\dagger$ of S, we will have $cl(S^\dagger) = S^\dagger$ and this will essentially make $S' \setminus S^\dagger = \emptyset$ in the proof.

Stitch

Definition 33. Given action sets A_{Max} and A_{Min}, and a relation $R \subseteq Seq(A_{\text{Max}}) \times Seq(A_{\text{Min}})$, $\textsc{Stitch}(R) \subseteq Seq(A_{\text{Max}} \cup A_{\text{Min}})$ is defined as
$\textsc{Stitch}(R) = \{s_1 s_2 | (s_1, s_2) \in R\}$.

For a sequence s, let s_{Max} and s_{Min} be sub-sequences of s restricted to A_{Max} and A_{Min} respectively.

Lemma 4.16. Let $G = (S, \Lambda)$ be a two player game with rulebook S and let $R_s = \{(s_{\mathsf{Max}}, s_{\mathsf{Min}}) \mid s \in S\}$. Let $G' = (S', \Lambda')$ be a game with $S' = cl(\textsc{Stitch}(R_S))$ and Λ' as follows:

$$\begin{aligned} \Lambda'(s_1 s_2) &= \Lambda(s) && \text{when } s_1 = s_{\mathsf{Max}}, s_2 = s_{\mathsf{Min}} \\ &= 0 && s_1 s_2 \notin \textsc{Stitch}(R_s) \end{aligned}$$

Then G is equivalent to G'.

Proof. For any sequence $s \in S$, the sequence $s_{\mathsf{Max}} s_{\mathsf{Min}} \in S'$ and $\mu(s) = \mu(s_{\mathsf{Max}} s_{\mathsf{Min}})$. Also for any sequence $s \notin \textsc{Stitch}(R_s)$, $\Lambda'(s') = 0$. Hence using similar arguments to the proof of Theorem 4.7, it follows that G and G' have the same payoff polynomial. This completes the proof. □

Lemma 4.16 tells us that any two player game in the extensive form is equivalent to another game where all actions of Max precede all actions of Min.

Now we will prove Theorem 4.8.

Proof of Theorem 4.8. Let $G = (S, \Lambda)$ be a two player game where S_{Max} has perfect recall and S_{Min} has A-loss recall shuffle $S^{\dagger}_{\mathsf{Min}}$. Let $R = \{(s_{\mathsf{Max}}, s^{\dagger}_{\mathsf{Min}}) | s \in S, s^{\dagger}_{\mathsf{Min}} \in S^{\dagger}_{\mathsf{Min}}$ is permutation of $s_{\mathsf{Min}}\}$ and let $S' = cl(\textsc{Stitch}(R))$.

Let $G = (S', \Lambda')$ be a game where Λ' is defined as follows: For $s \in S' \setminus \textsc{Stitch}(R)$, $\Lambda'(s) = 0$ and for $s' \in \textsc{Stitch}(R)$, $\Lambda'(s') = \Lambda(s)$ where s' is the permutation of s. Using similar arguments to the proof of Lemma 4.16, it can be shown that G and G' have the same payoff polynomial. Hence G and G' are equivalent. Also from Lemma 4.12 it follows that since we S' has at most $|S||A|$ sequences, $|G'|$ is $O(|G||A|)$.

Now we just need to show that S'_{Max} has perfect recall and S'_{Min} has A-loss recall. Since all actions of Max precede all actions of Min in every sequence s, observe that S'_{Max} is essentially $cl(S_{\mathsf{Max}})$. And since S_{Max} has perfect recall, it follows from Corollary 4.14 that S'_{Max} also has perfect recall.

Now for Min, S'_{Min} is formed by adding at each step, sequences of the form $s\bar{a}$ to $S^{\dagger}_{\mathsf{Min}}$, for some sas' in $S^{\dagger}_{\mathsf{Min}}$. Hence it follows from Lemma 4.13 that S'_{Min} has A-loss recall. □

Given a game without prior knowledge about the recalls of players, it can be efficiently verified if a certain player has perfect recall or A-loss recall. We can do this by verifying pairwise compatibility of every pair of sequences by an end to end string comparison. However for A-loss recall shuffle, the definition doesn't immediately provide a similar algorithm. A first approach would be to find permutations of sequences that are pairwise compatible. However any pair

of A-loss recall incompatible sequences can always be shuffled to turn them A-loss recall compatible, but making one pair compatible might make another pair incompatible.

Example 4.8. Let $S = \{ab, bc, ca\}$. For the rulebook $cl(S)$ to be have A-loss recall, its subset S needs to have A-loss recall shuffle as well. Now in any shuffle of S that has A-loss, the sequences ab and bc must be present in the form ba and bc. But then the sequence ca cannot be permuted to make it A-loss recall compatible with ba. Hence S, and consequently $cl(S)$ cannot have A-loss recall shuffles.

Now when we use the arguments described in Example 4.8 to see if a set has A-loss recall shuffle, in the process of turning a particular incompatible pair into compatible forms can possibly offer lots of permutations of the pairs to choose from. We will list out exactly the set of possible compatible forms for a pair.

For a sequence $s \in S$, let $\mathrm{Act}(s) \subseteq A$ be the set of all actions that appear in s.

Lemma 4.17. Let $S \subseteq Seq(A_{\mathsf{Max}})$ be a sequence set and s_1, s_2 be two sequences in S. Let $\textsc{com} = \mathrm{Act}(s_1) \cap \mathrm{Act}(s_2)$ be the set of all common actions in s_1 and s_2, for $i, j \in \{1, 2\}$ and $i \neq j$, let $\textsc{co-act}_{s_i} = (\mathrm{Act}(s_i) \cap cl(\mathrm{Act}(s_j))) \setminus \textsc{com}$ be the set of all the actions in s_i which has a co-action in s_j and $\textsc{rest}_{s_i} = \mathrm{Act}(s_i) \setminus cl(\mathrm{Act}(s_j))$ be all the other actions in s_i. Let s_1' and s_2' be permutations of s_1 and s_2 respectively, such that $s_1' \sim_A s_2'$. Then s_1' and s_2' must satisfy the following:

- If $\textsc{co-act}_{s_i} = \emptyset$, then $s_1' = sw_1$ and $s_2' = sw_2$ where $\mathrm{Act}(s) = \textsc{com}$ and $\mathrm{Act}(w_i) = \textsc{rest}_{s_i}$
- If $\textsc{co-act}_{s_i} \neq \emptyset$, then $s_1' = sa_1w_1$ and $s_2' = sa_2w_2$ where $\mathrm{Act}(s) \subseteq \textsc{com}$, $a_i \in \textsc{co-act}_{s_i}, cl(a_i) = \{a_1, a_2\}$ and $\mathrm{Act}(w_i) = (\textsc{com} \setminus \mathrm{Act}(s)) \cup (\textsc{co-act}_{s_i} \setminus \{a_i\}) \cup \textsc{rest}_{s_i}$

Proof. Notice that $\textsc{co-act}_{s_1} = \emptyset$ iff $\textsc{co-act}_{s_2} = \emptyset$. In this case, for $s_1 \sim_A s_2$ to hold, s_1 and s_2 must satisfy the first condition in Definition 24 and hence this follows. Now when $\textsc{co-act}_{s_i}$ are non-empty, s_1 and s_2 can satisfy either of the conditions for $\sim_A$. In both the cases, this satisfies the definition. □

Hence starting from a sequence set, in order for the whole set to have an A-loss recall shuffle, every pair of sequences induces some constraints on each of the sequences in the shuffled form. It is needless to say that checking all possible combinations of permutations of pairs of sequences to check which one works for the whole set is a costly procedure. So our next step is to identify sequence sets with A-loss recall shuffle efficiently. From Theorem 4.7 and Theorem 4.8 we can deduce that for identifying A-loss recall shuffles it is sufficient to work with

sequence sets of one player. So given a sequence set S over action set A, we have two objectives: First we want to detect if S has an A-loss recall shuffle. Next if S has an A-loss recall shuffle then we want to compute the A-loss recall shuffle so that we can build the equivalent game. In the next section we will provide one single algorithm that will achieve both these goals efficiently.

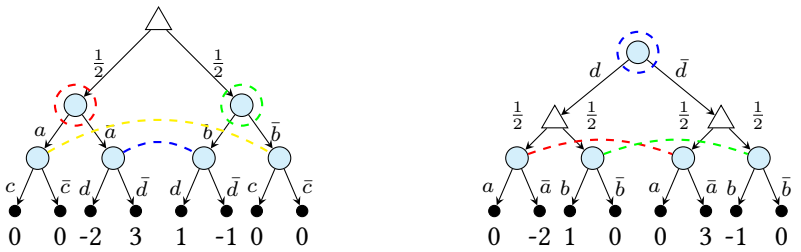


Figure 4.5: A-loss recall shuffle using sequences with non-zero payoff

Remark 4.18. Given a rulebook, in order to find A-loss recall shuffles and construct the equivalent game, we considered all sequences from the rulebook. This can be slightly optimized by taking sequences only with non-zero utility in the original sequence form. For example in Fig. 4.5 the rulebook of the game on the left doesn't have A-loss recall shuffle. This is because from Lemma 4.17 it follows that the pair ad and bd has the form da and db in the shuffle, and ac and $\bar{b}\bar{c}$ has the form ca and $\bar{c}\bar{b}$. But then $da \not\sim_A ca$. However if we take only the sequences with non-zero Λ values, then the resulting set has an A-loss recall shuffle. The closure of this shuffle gives us the game on the right on using the construction in Proposition 4.9. Note that in the game on the left, Max has an incentive to go to the zero payoff leaves since there are negative payoffs in the middle part. However they can be avoided for computational purposes.

Remark 4.19. In regards to the last remark, for an arbitrary sequence set S and its closure $cl(S)$, we can deduce that if $cl(S)$ has an A-loss recall shuffle then S also has an A-loss recall shuffle, since $S \subseteq cl(S)$. But the other direction is not true in general. For example, if $S = \{ba, ca\}$ then $\{ab, ac\}$ is an A-loss recall shuffle of S. But the closure $cl(S) = \{\bar{b}, ba, b\bar{a}, \bar{c}, ca, c\bar{a}\}$ has no A-loss recall shuffle. This is because, again from Lemma 4.17, ba and ca must be in the form ab and ac in the shuffle. But then $\bar{b} \not\sim_A ab$.

4.3 Finding A-loss recall shuffles

In this section we study properties of sequence sets for which an A-loss recall shuffle exists and devise an algorithm to identify such sets and eventually find the shuffle. Since we are working with one player sequence sets, we will only consider sequence sets S over $A = A_{\mathsf{Max}}$.

First we set up some notations that we will use for the rest of this chapter. Recall that for $s \in S$, $\mathrm{Act}(s)$ is the set of all actions in s. For $S \subseteq Seq(A)$ and $a \in A$, let $S_a = \{s \in S \mid a \in \mathrm{Act}(s)\}$, $S_{\bar{a}} = \{s \in S \mid \bar{a} \in \mathrm{Act}(s)\}$, $S_{\not a} = S \setminus (S_a \cup S_{\bar{a}})$. For a set S, let $\mathrm{Act}(S) = \{a \mid a \in \mathrm{Act}(s) \text{ for some } s \in S\}$. For a sequence s and $A' \subseteq A$, let $s[A']$ be the sequence s with only the actions restricted to A' and let $S[A'] = \{s[A'] \mid s \in S\}$. Let $s[\backslash A'] = s[A \setminus A']$. $S[\backslash A']$ is defined similarly. We use the notation $s[\backslash a]$ and $S[\backslash a]$ when $A' = \{a\}$. Also for $a \in A$ let $aS = \{as \mid s \in S\}$.

How do sequence sets with A-loss recall look like?
Before finding A-loss recall shuffles of sequence sets, we will have some insight about the structure of sets with A-loss recall. First we will define the notion of connected sets that will be useful in breaking down the problem of finding A-loss recall shuffle into smaller instances.

Definition 34 (Connectedness)**.** For two sequences s_1 and s_2 from $Seq(A)$, s_1 and s_2 are said to be connected if $cl(\mathrm{Act}(s_1)) \cap cl(\mathrm{Act}(s_2)) \neq \emptyset$.

For sequence sets S_1 and S_2, S_1 and S_2 are said to be connected if there exist $s_1 \in S_1$ and $s_2 \in S_2$ such that s_1 and s_2 are connected.

A sequence set S is called connected if for all disjoint partitions S_1, S_2 of S, i.e., $S = S_1 \uplus S_2$, S_1 and S_2 are connected.

Lemma 4.20. A sequence set S can be uniquely decomposed into disjoint sets $\{S_i\}$ such that each S_i is connected.

Proof. Consider an undirected graph where the vertices are sequences from S and there is an edge between two nodes, if the corresponding sequences are connected. Then the connected components of this graph give the desired decomposition. □

Each set S_i in the decomposition is called a *connected component* of S.

Example 4.9. The set $S = \{ab, b, \bar{b}c, ac, d\bar{e}, ef\}$ is a disconnected set whose connected components are $S_1 = \{ab, b, \bar{b}c, ac\}$ and $S_2 = \{d\bar{e}, ef\}$. $cl(\mathrm{Act}(S_1)) \cap cl(\mathrm{Act}(S_2))$ are disjoint. S_2 is connected since $d\bar{e}$ and ef are connected. S_1 is connected because $ab, \bar{b}c, ac$ are all mutually connected and b is connected to both ab and $\bar{b}c$.

Equipped with the notion of connectedness, there is now a simpler interpretation of the relation $\sim_A$. For two sequences s_1 and s_2, when $s_1 \sim_A s_2$, either $(s_1, s_2) = (ss_1', ss_2')$ with s_1' and s_2' being disconnected or $(s_1, s_2) = (sas_1', s\bar{a}s_2')$ for some $a, \bar{a} \in A$.

Now we will state some simple observations on the A-loss recall compatibility of two sequences.

Lemma 4.21. Let s_1 and s_2 be two sequences in $Seq(A)$. Then the following statements are true.

1. For any $a, \bar{a} \in A$ such that $\mathrm{Act}(\{s_1, s_2\}) \cap cl(a) = \emptyset$, it holds that $as_1 \sim_A \bar{a}s_2$.
2. If $s_1 \sim_A s_2$ and for some $a, \bar{a} \in A$, $\mathrm{Act}(\{s_1, s_2\}) \cap cl(a) = \emptyset$, then $as_1 \sim_A as_2$ and $\bar{a}s_1 \sim_A \bar{a}s_2$
3. If $s_1 \sim_A s_2$ such that $a \in \mathrm{Act}(s_1)$ and $a \in \mathrm{Act}(s_2)$ then $s_1[\backslash a] \sim_A s_2[\backslash a]$.
4. If $s_1 \sim_A s_2$ such that $a \in \mathrm{Act}(s_1)$ and $cl(a) \cap \mathrm{Act}(s_2) = \emptyset$, then $s_1[\backslash a] \sim_A s_2$.

Proof. The first statement follows from the definition of $\sim_A$. The second statement is true since adding common prefixes doesn't interfere with the rest of the sequence for A-loss recall compatibility.

For the third statement, since $s_1 \sim_A s_2$, either $(s_1, s_2) = (ss_1', ss_2')$ with s_1' and s_2' being disconnected or $(s_1, s_2) = (sbs_1', s\bar{b}s_2')$ for some $b \in A$. In the first case a can only be present in s and deletion of a preserves the form, hence preserving A-loss recall compatibility. In the second case a can be in s or both in s_1' and s_2'. In either case deletion of a still preserves the form and consequently A-loss recall compatibility. Hence in both case $s_1[\backslash a] \sim s_2[\backslash a]$.

For the fourth statement, again either $(s_1, s_2) = (ss_1', ss_2')$ with s_1' and s_2' being disconnected or $(s_1, s_2) = (sbs_1', s\bar{b}s_2')$ for some $b \in A$. Since $cl(a) \cap \mathrm{Act}(s_2) = \emptyset$, it follows that $a \in s_1'$. Hence removing a preserves the structure for $\sim_A$ compatibility.

□

Proposition 4.22. Let $S \subseteq Seq(A)$ be a sequence set with A-loss recall. Then the following statements are true:

1. If S is connected, then $S = aS_1 \uplus \bar{a}S_2$ for some $a \in A$ where each of S_1 and S_2 has A-loss recall.
2. If S is disconnected and let $S = \uplus_i S_i$ be the decomposition of S into disjoint connected components, then each S_i has A-loss recall.

Proof. Since any subset of a set with A-loss recall also has A-loss recall, the second statement follows. We will now prove the first statement.

Let $s = as'$ be a sequence in S. Consider the sets S_a and $S_{\bar{a}}$. Now from the definition of $\sim_A$, we cannot have a sequence of the form $s''as'''$ in S_a with non-empty s'', since then $as' \not\sim_A s''as'''$. Hence all sequences in S_a start with a and let $S_a = aS_1$. Similar statement holds for $\bar{a}$ and $S_{\bar{a}}$. Let $S_{\bar{a}} = \bar{a}S_2$. For any $as_1, as_2 \in S_a$, we have $as_1 \sim_A as_2$. From Lemma 4.21 it follows that $s_1 \sim_A s_2$. Hence S_1 has A-loss recall. By similar argument S_2 also has A-loss recall. Now if $S_{\not a} = \emptyset$, we are done. Suppose this is not true. Then since S is connected, $(S_a \cup S_{\bar{a}})$ and $S_{\not a}$ are connected. Then w.l.o.g. there is a sequence $s_0 = as_0'$ and another sequence $s_3 = s_3'bs_3''$ in S with $b \notin cl(a)$ but $b \in cl(\text{Act}(s_0'))$. But then from definition of $\sim_A$ it follows that $s_0 \not\sim_A s_3$ which contradicts the fact that S has A-loss recall. Hence $S_{\not a} = \emptyset$. Therefore $S = aS_1 \uplus \bar{a}S_2$ and each of S_1 and S_2 has A-loss recall. □

Proposition 4.22 gives a recursive characterization of sequences sets with A-loss recall. Now we will exploit this recursive structure to characterize sequence sets with A-loss recall shuffle and compute it.

Finding A-loss recall shuffles

We start with an arbitrary sequence set $S \subseteq Seq(A)$ and our goal is to check if S has an A-loss recall shuffle.

Firstly we see that, looking for A-loss recall shuffle of a sequence set can be decomposed in finding A-loss recall shuffle of each connected component.

Proposition 4.23. Let S be a disconnected set and $S = \uplus_i S_i$ be the decomposition of S into disjoint connected components. Then S has an A-loss recall shuffle iff each connected component S_i has an A-loss recall shuffle.

Proof. For the forward direction let S^+ be an A-loss recall shuffle of S. For each $s \in S$, let s^+ be the permutation of s in S^+. For S_i, let $S_i^+ = \{s_+ \mid s \in S_i\}$. For any two disconnected sequences s_1 and s_2, s_1^+ and s_2^+ are also disconnected. Hence $\{S_i^+\}$ is the decomposition of S^+ into connected components. It follows from Proposition 4.22 that each S_i^+ had A-loss recall. Hence for each i, S_i^+ is an A-loss recall shuffle of S_i^+. This proves the statement.

For the other direction let S_i^+ be an A-loss recall shuffle of each connected component S_i of S. Let $S^+ = \cup_i S_i^+$. We claim that S^+ is an A-loss recall shuffle of S. For $s \in S$, $s \in S_i$ for some i. Let s^+ be the permutation of s in S_i^+. Hence every sequence s has a permutation in S^+. Now for $s_1, s_2 \in S_i^+$ it is already true that $s_1 \sim_A s_2$. If $s_1 \in S_i^+$ and $s_2 \in S_j^+$ for distinct i and j, we know that s_1 and

s_2 are disconnected. Hence $s_1 \sim_A s_2$ in this case as well. Hence S^+ has A-loss recall and it is an A-loss recall shuffle of S. This completes the proof. □

Corollary 4.24. For a set S, let S_i^+ be an A-loss recall shuffle of each connected component S_i of S. Then $S^+ = \cup_i S_i^+$ is an A-loss recall shuffle of S.

Now we will see a recursive condition for a connected sequence set to have an A-loss recall shuffle.

Proposition 4.25. Let S be a connected set.

1. S has an A-loss recall shuffle $\implies$ $\exists a, \bar{a} \in A$ such that $S_{\not a} = \emptyset$
2. Suppose $\exists a, \bar{a} \in A$ such that $S_{\not a} = \emptyset$. Then S has an A-loss recall shuffle iff both $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ have A-loss recall shuffles.

Proof.
(1)

Let S^+ be an A-loss recall shuffle of S. S^+ is also connected. It follows from Proposition 4.22 that for some a and $\bar{a}$, $S = aS_1 \uplus \bar{a}S_2$ where each of S_1 and S_2 has A-loss recall. For action $a, \bar{a}$ it follows that $S_{\not a} = \emptyset$.
(2)

We first prove the backward direction. Let S_1 and S_2 be A-loss recall shuffles of $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ respectively. Let $S^+ = aS_1 \uplus \bar{a}S_2$. We claim that S^+ is an A-loss recall shuffle of S. First we show that S^+ is a shuffle of S. Since S_1 is a shuffle of $S_a[\backslash a]$, it follows that aS_1 is a shuffle of S_a. Similarly $\bar{a}S_2$ is also a shuffle of $S_{\bar{a}}$. Hence S^+ is indeed a shuffle of S.

Now we will show that S^+ has A-loss recall. For any two sequences $as_1, as_2 \in aS_1$, since $s_1 \sim_A s_2$, it follows from Lemma 4.21 that $as_1 \sim as_2$. Hence aS_1 has A-loss recall. By similar arguments $\bar{a}S_2$ has A-loss recall as well. Also since $aS_1 \cap \bar{a}S_2 = \emptyset$, it follows that S^+ is an A-loss shuffle of S.

Now for the forward direction let S^+ be an A-loss recall shuffle of S. We have $S^+ = S_a^+ \uplus S_{\bar{a}}^+$. We observe that S_a^+ and $S_{\bar{a}}^+$ are A-loss recall shuffles of S_a and $S_{\bar{a}}$ respectively. Consider the sets $S_a^+[\backslash a]$ and $S_{\bar{a}}^+[\backslash \bar{a}]$. We claim that these are A-loss recall shuffles of $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ respectively. We observe that $S_a^+[\backslash a]$ is a shuffle of $S_a[\backslash a]$. It remains to show that this set has A-loss recall. Let s_1 and s_2 be any two sequences in $S_a^+[\backslash a]$. There exists s_1' and s_2' in S_a^+ such that $s_1 = s_1'[\backslash a]$ and $s_2 = s_2'[\backslash a]$. S_a^+ has A-loss recall and hence $s_1' \sim_A s_2'$. It follows from Lemma 4.21 that $s_1 \sim_A s_2$. Hence $S_a^+[\backslash a]$ has A-loss recall shuffle. Similarly it can be shown that $S_{\bar{a}}^+[\backslash \bar{a}]$ also has A-loss recall. Hence $S_a^+[\backslash a]$ and $S_{\bar{a}}^+[\backslash \bar{a}]$ are A-loss recall shuffles of $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ respectively. This completes the proof. □

Corollary 4.26. Let S be a connected set with $S_{\not a} = \emptyset$ for some $a, \bar{a} \in A$. Let S_1 and S_2 be A-loss recall shuffles of $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ respectively. Then $S^+ = aS_1 \cup \bar{a}S_2$ is an A-loss recall shuffle of S.

Proposition 4.25 tells us how to find A-loss recall shuffle of connected sets. The first statement provides a necessary condition for a connected sequence set S to have A-loss recall shuffle. It says that one must be able to find $a, \bar{a}$ such that $S_{\not a} = \emptyset$. Once we have found any such $a, \bar{a}$, the second statement gives us a necessary and sufficient condition. The only way S can have A-loss recall shuffle is when each of the sets $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ has A-loss recall shuffles. Hence they are recursively checked for A-loss recall shuffles. Furthermore, if there are multiple actions a satisfying $S_{\not a} = \emptyset$, it does not matter in which order we pick them for the recursive call.

Example 4.10. In Fig. 4.5, the game in Fig. 4.5a has sequence set $S = \{ac, a\bar{c}, \bar{a}d, \bar{a}\bar{d}, bd, b\bar{d}, \bar{b}c, \bar{b}\bar{c}\}$ which doesn't have A-loss recall shuffle. This is because this set is connected but it fails the first condition in Proposition 4.25. On the other hand, if we take the sub-set of S, $S' = \{\bar{a}d, \bar{a}\bar{d}, bd, b\bar{d}\}$ which doesn't have A-loss recall, we can find the A-loss recall shuffle $\{d\bar{a}, \bar{d}\bar{a}, db, \bar{d}\bar{b}\}$ from which we can construct the game in Fig. 4.5b.

Using Proposition 4.23 and Proposition 4.25 we can also deduce a nice property of A-loss recall shuffles of complete sets.

Lemma 4.27. Let $S \subseteq A_{\mathsf{Max}}$ be a complete sequence set and let $S^\dagger$ be an A-loss recall shuffle of S. Then $S^\dagger$ is also a complete set.

Proof. We show this by induction on the size of A_{Max}. This is trivial for the case when A_{Max} is of the form $\{a, \bar{a}\}$. Now suppose the statement is true when $|A_{\mathsf{Max}}| = 2k$. Let $|A_{\mathsf{Max}}| = 2k + 2$. If S is disconnected with connected components S_i, then each of S_i is complete since $cl(S_i) \cap cl(S_j) = \emptyset$ for $i \neq j$. And since number of actions in each component is strictly less, by induction hypothesis and application of Proposition 4.23 each of the shuffles $S_i^\dagger$ are also complete. This implies S is complete.

Now in the case when S is connected, since S has A-loss recall, we know from Proposition 4.25 that for some $a, \bar{a}$, $S^\dagger = aS_1 \uplus \bar{a}S_2$ where S_1 and S_2 are A-loss recall shuffles of $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ respectively. For our purpose, it is sufficient to show that each of the sets $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ are complete. This is because from induction hypothesis S_1 and S_2 would also be complete and for any complete set S' and any action a', $a'S'$ is also a complete set. Suppose $S_a[\backslash a]$ is not complete. Then for some sequence $sbs' \in S_a[\backslash a]$, $S_a[\backslash a]$ has no sequence of the form $s\bar{b}s''$. Also some sequence $s_0 \in S$, $sbs' = s_0[\backslash a]$. In s_0, a exists either before b or after

b. a cannot be after b since then from completeness of S, there is a sequence of form $s\bar{b}s''' \in S$ and $s\bar{b}s'''[\backslash a] \in S_a[\backslash a]$. The other possibility is a is before b in s_0, i.e $s_0 = s_1 a s_2 b s'$. But in this case some $s_1 a s_2 \bar{b} s'''$ exists in S which would imply $s\bar{b}s''' \in S_a[\backslash a]$ which is a contradiction. Hence $S_a[\backslash a]$ is also complete. Similarly we can show that $S_{\bar{a}}[\backslash \bar{a}]$ is also complete. This completes the proof.

□

Based on our observations so far we provide an algorithm to check if a set S has A-loss recall shuffle. The algorithm returns the shuffle if S has an A-loss recall shuffle and returns -1 otherwise.

Algorithm 1 Compute A-loss recall shuffle

```
1:  Input : S
2:  function A-loss Recall Shuffle(S)
3:      if S is connected then
4:          if ∃a such that S_a' = ∅ then
5:              S_1 ← A-loss Recall Shuffle(S_a[\a])
6:              S_2 ← A-loss Recall Shuffle(S_ā[\ā])
7:              S^+ ← aS_1 ∪ ā S_2
8:              return S^+
9:          else
10:             return -1 (A-loss recall shuffle not possible)
11:     else
12:         S = ⊎S_i where each S_i is connected
13:         S^+ = ∪A-loss Recall Shuffle(S_i)
14:         return S^+
```

We use the previous propositions to establish the correctness and running time of Algorithm 1.

Theorem 4.28. *Given a set sequence set S the function A-loss Recall Shuffle in Algorithm 1 runs in polynomial time constructing an A-loss recall shuffle of S if there exists one or otherwise returns -1.*

Proof. For correctness, first when S is disconnected following Proposition 4.23, the function A-loss Recall Shuffle is called recursively on each connected component in Line 13 applying Corollary 4.24. When S is connected based on Proposition 4.25, if an action a satisfying the condition is not found in Line 4 then A-loss recall shuffle doesn't exist as returned in Line 10. If such an action a is found, then again based on Proposition 4.25 and Corollary 4.26 the function

A-loss Recall Shuffle is called upon $S_a[\backslash a]$ and $S_{\bar{a}}[\backslash \bar{a}]$ recursively in Line 5 and 6 respectively.

For the running time analysis, observe that recursive calls in Lines 5 and 6 as well as in Line 13 are called on disjoint non-overlapping subsets. Finding connected component of a set takes linear time using graph. Checking the condition in Line 4 also takes linear time. Hence Algorithm 1 runs in linear time in the input size. □

Algorithm 1 can be used to solve a two player game given Max has perfect recall and Min has A-loss recall shuffle following Theorem 4.8. Recall that the operation Stitch defined in Definition 33 produces a sequence set over A following Lemma 4.16.

Algorithm 2 Simplify Game with shuffle

1: Input : G in extensive form where Max has perfect recall
2: $(S, \Lambda) \leftarrow G$ in sequence form
3: $S_1 \leftarrow S_{\text{Max}}$
4: $S_2 \leftarrow S_{\text{Min}}$
5: **if** A-loss Recall Shuffle(S_2) returns -1 **then**
6: No simplification of G possible with shuffles
7: **return** G
8: **else**
9: $S_2^+ \leftarrow$ A-loss Recall Shuffle(S_2)
10: $R = \{(s_{\text{Max}}, s^+_{\text{Min}}) | s \in S\}$
11: $S' \leftarrow cl(\text{Stitch}(R))$
12: $G' \leftarrow (S', \Lambda')$ in sequence form
13: **return** G'

Now it is possible that a sequence set S doesn't have an A-loss recall shuffle. So to deal with any sequence set in general we generalize the notion of A-loss recall shuffle to *A-loss recall span* and devise an algorithm to compute them.

4.4 Generalizing A-loss recall shuffle: A-loss recall span

In this section we will generalize the notion of A-loss recall shuffle. The key observation in this process is that in the case of A-loss recall shuffles, the games G and G' become equivalent via payoff polynomials because the set of monomials generated by the two games are the same. But actually in order to have the same payoff polynomial, the exact same set of monomials is not required. Any set of monomials that can linearly combine and generate the other payoff polynomial is

sufficient. In fact any set of monomials that can generate each monomial in the other set suffices. For this we will see the notion of spanning sets and later *A-loss recall Span*.

We define spanning sets over the action set A_{Max} of player Max.

Definition 35 (Spanning sets)**.** For two sequence sets $S, S' \subseteq Seq(A_{\mathsf{Max}})$, S is said to span S' denoted by $S' \trianglelefteq S$, if for each sequence s' in S', $\mu(s')$ can be expressed as a linear combination of monomials of sequences in S. In other words, $S' \trianglelefteq S$ iff $\forall s' \in S', \mu(s') = \sum_{s \in S} k_s^{s'} \mu(s)$ where $k_s^{s'} \in \mathbb{R}$.

For sets S, S' such that $S' \trianglelefteq S$, the span matrix of S' w.r.t. S denoted by $\mathcal{M}_S^{S'}$ is a $|S'| \times |S|$ matrix indexed by $\{(s', s)\}_{s' \in S',\ s \in S}$, where the entry at (s', s) denoted by $\mathcal{M}_{S\ [s',s]}^{S'} = k_s^{s'}$.

Example 4.11. Over the set of actions $A = \{a, \bar{a}, b, \bar{b}\}$, the sequence set $S = \{ab, a\bar{b}, \bar{a}b, \bar{a}\bar{b}\}$ spans the sequence set $S' = \{\bar{a}, b, ba\}$. This is because $x_a x_b$ is already present in $\mu(S)$, $x_b = x_a x_b + (1 - x_a)x_b$ and $(1 - x_a) = (1 - x_a)x_b + (1 - x_a)(1 - x_b)$. The span matrix $\mathcal{M}_S^{S'}$ is the following:

$$\begin{array}{c} \\ \bar{a} \\ b \\ ba \end{array} \begin{array}{c} \begin{array}{cccc} ab & a\bar{b} & \bar{a}b & \bar{a}\bar{b} \end{array} \\ \begin{bmatrix} 0 & 0 & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Now we are ready to define A-loss recall span.

Definition 36 (A-loss recall Span)**.** For a sequence set $S \subseteq Seq(A_{\mathsf{Max}})$, an A-loss recall span $S^\dagger$ of S is a sequence set such that $S^\dagger$ has A-loss recall and $S \trianglelefteq S^\dagger$.

In Example 4.11, S is an A-loss recall span of S' since S has A-loss recall. Observe that S' doesn't have an A-loss recall shuffle.

It is needless to say that A-loss recall shuffle is a special kind of A-loss recall span where $\mu(S) = \mu(S')$. We saw earlier that for an arbitrary sequence set, an A-loss recall shuffle does not always exist. However it turns out that every sequence set always has an A-loss recall span. The caveat here is that the size of the A-loss recall span is not guaranteed to be polynomially bounded in the size of the initial sequence set. So the following task is to find small A-loss recall spans of a set. First we will see how to generate an A-loss recall span of any set. And then we will see how to optimize the size of A-loss recall spans.

Since we deal with sequence sets of single player, once again we assume that $A = A_{\mathsf{Max}}$.

Theorem 4.29. *Let $S \subseteq Seq(A)$ be a sequence set, then there exists a sequence set $S^\dagger$ such that $S^\dagger$ is an A-loss recall span of S.*

Proof. Given a sequence set S we need to find a set $S^\dagger$ such that $S \trianglelefteq S^\dagger$ and $S^\dagger$ has A-loss recall. We will directly construct the set $S^\dagger$.

Fix any ordering $a_1, \bar{a_1}, \ldots, a_n, \bar{a_n}$ of the actions in A. For a subset $Z = \{i_1, \ldots, i_k\} \subseteq [n]$ with $|Z| = k$, let $S_Z = (a_{i_1} + \bar{a}_{i_1}) \ldots (a_{i_k} + \bar{a}_{i_k})$ be the set of all sequences of length k over action $A_Z = \bigcup_{i_j} cl(a_{i_j})$, and each of them following the ordering of actions.

We claim that $S^\dagger = S_{[n]}$ is an A-loss recall span of S.

By induction on the length of sequences using Lemma 4.21 it can be shown that for any $Z \subseteq [n]$, S_Z has A-loss recall. We need to show that $\mu(S^\dagger)$ can generate any monomial in $\mu(S)$[3].

First we make a simple observation. For any $Z \subseteq [n]$ it can be verified that $\sum_{s \in S_Z} \mu(s) = 1$.

Now for any sequence $s \in Seq(A)$, we claim that the following is true:

$$\mu(s) = \sum_{s' \in S^\dagger, \mathrm{Act}(s) \subseteq \mathrm{Act}(s')} \mu(s')$$

This is because if $cl(s) = A_Z$ for suitable $Z \subseteq [n]$, then

$$\sum_{s' \in S^\dagger, \mathrm{Act}(s) \subseteq \mathrm{Act}(s')} \mu(s') = \mu(s) \sum_{s'' \in S_{[n] \setminus Z}} \mu(s'') = \mu(s)$$

Hence in the spanning matrix $\mathcal{M}^S_{S^\dagger}$, the entry $\mathcal{M}^S_{S^\dagger\ [s,s']} = 1$ if $\mathrm{Act}(s) \subseteq \mathrm{Act}(s')$, otherwise it is 0. Hence $S^\dagger$ is an A-loss recall span of S. □

Example 4.12. In Fig. 4.6a the game G doesn't have A-loss recall. $S = \{a, b, c, abc\}$ is the set of sequences with non-zero payoff, since sequences with zero payoff doesn't contribute to the final payoff, it is sufficient to find an A-loss recall span of S.
$S^\dagger = \{abc, ab\bar{c}, a\bar{b}c, a\bar{b}\bar{c}, \bar{a}\bar{b}\bar{c}, \bar{a}b\bar{c}, \bar{a}\bar{b}c, \bar{a}\bar{b}\bar{c}\}$ is the A-loss recall span of S obtained using the construction in Theorem 4.29. The span matrix $\mathcal{M}^S_{S^\dagger}$ is obtained in this process is given in Fig. 4.6b.

[3] In fact it can be shown that $\mu(S^\dagger)$ is a basis of the vector space of all multi-linear polynomials over $\{x_{a_i}\}$ with $\mathbb{R}$ as the underlying field

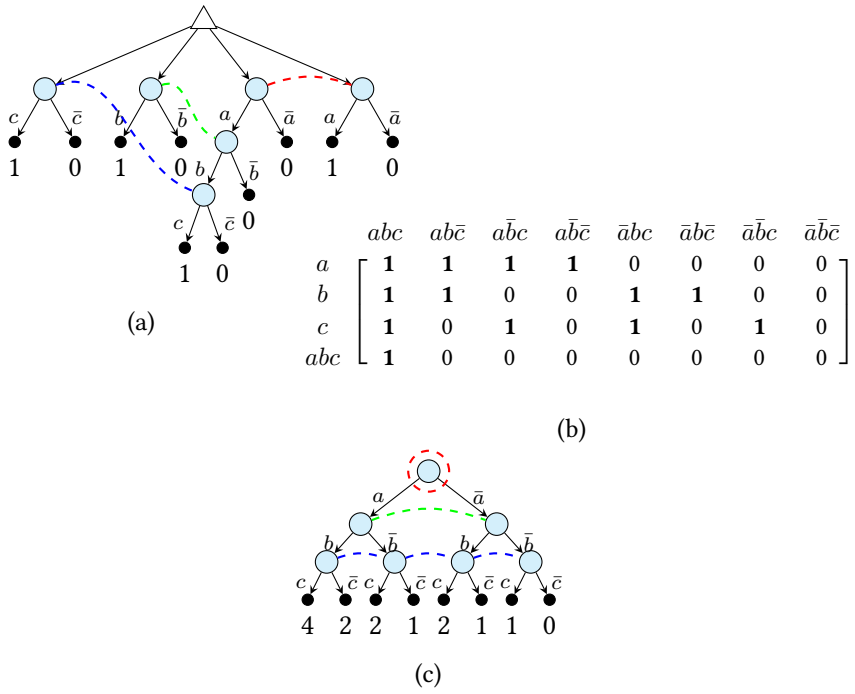


Figure 4.6: The game G in Fig. 4.6a doesn't have A-loss recall and $S = \{a, b, c, abc\}$ is the set of sequences with non-zero payoffs. The matrix in Fig. 4.6b is the span matrix $\mathcal{M}^S_{S^\dagger}$ correspnding to the A-loss recall span $S^\dagger = \{abc, ab\bar{c}, a\bar{b}c, a\bar{b}\bar{c}, \bar{a}bc, \bar{a}b\bar{c}, \bar{a}\bar{b}c, \bar{a}\bar{b}\bar{c}\}$ obtained in Theorem 4.29 for set S. The game in Fig. 4.6c is a game G' which is equivalent to G constructed using $\mathcal{M}^S_{S^\dagger}$

A game G' which is equivalent to G with sequence set $S^\dagger$ constructed from $\mathcal{M}^S_{S^\dagger}$ following Theorem 4.29 is given in Fig. 4.6c.

Now that we have established every sequence set has an A-loss recall span, similar to A-loss recall shuffle we will see how to generate equivalent games using spans. We will generalize the assignment of payoffs that we did in the case of A-loss recall shuffle. Starting from a game $G = (S, \Lambda)$ and an A-loss recall shuffle $S^\dagger$ of S we gave the equivalent game $G' = (S^\dagger, \Lambda')$. Λ' assigned an augmented utility to s equal to that given to its permutation by Λ. Now given a span we will assign utility as a linear combination of monomials that generate it.

Theorem 4.30. *Let $G = (S, \Lambda)$ be a one-player game with $S \subseteq Seq(A)$ and let $S^\dagger$ be an A-loss recall span of S. Then there exists a game $G' = (S', \Lambda')$ with A-loss recall such that $G \sim G'$ and $|S'| = O(|A||S^\dagger|)$.*

Proof. Given $S^\dagger$, an A-loss recall span of S, let $S' = cl(S^\dagger)$ be the rulebook given by the closure of $S^\dagger$. It follows from Corollary 4.14 and the definition of A-loss recall span that $S \trianglelefteq S'$ and S' has A-loss recall. Any game with S' as rulebook will have A-loss recall by definition. Given $G = (S, \Lambda)$ we will construct the payoff function Λ' in order to give the equivalent game $G' = (S', \Lambda')$. Now since $S \trianglelefteq S'$, for every $s \in S$ we have $\mu(s) = \sum_{s' \in S'} k^s_{s'} \mu(s')$. Then the payoff function Λ' over S' is given by:

$$\Lambda'(s') = \sum_{s \in S} k^s_{s'} \Lambda(s)$$

Now for equivalence we will show that they have the same payoff polynomial. The payoff polynomial in G' is given by

$$\begin{aligned}
\sum_{s' \in S'} \mu(s')\Lambda'(s') &= \sum_{s' \in S'} \mu(s') \sum_{s \in S} k^s_{s'} \Lambda(s) \\
&= \sum_{s' \in S'} \sum_{s \in S} k^s_{s'} \mu(s) \Lambda(s) \\
&= \sum_{s \in S} \sum_{s' \in S'} k^s_{s'} \mu(s) \Lambda(s) \\
&= \sum_{s \in S} \Lambda(s) \sum_{s' \in S'} k^s_{s'} \mu(s') \\
&= \sum_{s \in S} \mu(s) \Lambda(s)
\end{aligned}$$

This is the payoff polynomial of G and hence $G \sim G'$.

Also it follows from Lemma 4.12 that $|S'| = O(|A||S^\dagger|)$. This completes the proof. □

Remark 4.31. We don't consider the size of the game G', because we will see later in Lemma 4.41 that for our purposes this adds only an extra polynomial factor to the size S'.

Theorem 4.32. *Let $G = (S, \Lambda)$ be a two player game where Max has perfect recall and let $S^\dagger_{\text{Min}}$ be an A-loss recall span of S_{Min}. Then there is a game $G' = (S', \Lambda')$ with $|S'| = O(|G||S^\dagger_{\text{Min}}||A|)$ where Max has perfect recall, Min has A-loss recall and $G \sim G'$.*

Proof. Let $G = (S, \Lambda)$ be a two player game where S_{Max} has perfect recall and S_{Min} has A-loss recall span $S^\dagger_{\text{Min}}$.

For a sequence s, let s_{Max} and s_{Min} be sub-sequences of s restricted to A_{Max} and A_{Min} respectively. Let $S'' = S_{\mathsf{Max}}S^{\dagger}_{\mathsf{Min}} = \{s_1s_2 | s_1 \in S_{\mathsf{Max}}, s_2 \in S^{\dagger}_{\mathsf{Min}}\}$ and let $S' = cl(S'')$ be the closure of S''.

Since $S^{\dagger}_{\mathsf{Min}}$ spans S_{Min} for every $s \in S$ we have $\mu(s_{\mathsf{Min}}) = \sum_{s' \in S^{\dagger}_{\mathsf{Min}}} k_{s'}^{s_{\mathsf{Min}}} \mu(s')$.

Let $G' = (S', \Lambda')$ be a game where Λ' is defined as follows: For $s \in S, s' \in S^{\dagger}_{\mathsf{Min}}$, $\Lambda'(s_{\mathsf{Max}}s') = k_{s'}^{s_{\mathsf{Min}}}\Lambda(s)$ and for $s'' \in S' \setminus S''$, $\Lambda'(s'') = 0$.

Using similar arguments as in the proof of Theorem 4.30, it can be shown that G and G' has the same game polynomial. Hence G and G' are equivalent and $|S'|$ is $O(|G||S^{\dagger}_{\mathsf{Min}}||A|)$.

Now we just need to show that S'_{Max} has perfect recall and S'_{Min} has A-loss recall. Since all actions of Max precede all actions of Min in every sequence s, observe that S'_{Max} is essentially $cl(S_{\mathsf{Max}})$. And since S_{Max} has perfect recall, it follows from Corollary 4.14 that S'_{Max} also has perfect recall.

Now for Min, S'_{Min} is formed by adding at each step sequences of the form $s\bar{a}$ to $S^{\dagger}_{\mathsf{Min}}$, for some sas' in $S^{\dagger}_{\mathsf{Min}}$. Hence it follows from Lemma 4.13 that S'_{Min} has A-loss recall. □

Theorem 4.32 tells us that any game G where Max has perfect recall and Min is non-absentminded, one can find an equivalent game G' where Max has perfect recall and Min has A-loss recall. Now the A-loss recall span used to prove Theorem 4.29 is exponential in the size of the initial sequence set. This naturally leads to the question: what is the smallest size of such an A-loss recall span. Essentially given a sequence set S we want to find the minimal size $S^{\dagger}$ such that $S \trianglelefteq S^{\dagger}$ and $S^{\dagger}$ has A-loss recall.

Remark 4.33. The size of the game S' in Theorem 4.32 can be improved by replacing $|G||S^{\dagger}_{\mathsf{Min}}|$ with the number of non-zero entries in the span matrix $\mathcal{M}^{S_{\mathsf{Min}}}_{S^{\dagger}_{\mathsf{Min}}}$. This can be achieved as follows: $S'' = \{s_{\mathsf{Max}}s' | s \in S, s' \in S^{\dagger}_{\mathsf{Min}}, k_{s'}^{s_{\mathsf{Min}}} \neq 0\}$ and $S' = cl(S'')$. This doesn't affect the way Λ' is defined since all the redundant sequences are no longer present. This might significantly improve the size of G' in the case when $\mathcal{M}^{S_{\mathsf{Min}}}_{S^{\dagger}_{\mathsf{Min}}}$ is a sparse matrix even though $S^{\dagger}_{\mathsf{Min}}$ is large.

4.4.1 Finding minimal A-loss recall span

For a sequence set S, the optimal A-loss recall span of S need not be an A-loss recall shuffle. In this section we provide an exact algorithm to compute the optimal A-loss recall span. For this we make some key observations borrowing insights from the algorithm for finding A-loss recall shuffle.

Firstly, in similar flavor to Proposition 4.23 for A-loss recall shuffles, we show that it is sufficient to find optimal A-loss recall spans of each connected component of a sequence set. To show this, first we note that to span a sequence set S we don't require actions out of $cl(\mathrm{Act}(S))$.

For $x \in X$ and some $b \in \mathbb{R}$, let $f[x := b]$ be the resulting polynomial by substituting x with b in f.

Lemma 4.34. Let S be a sequence set, and $S^\dagger$ an optimal A-loss recall span of S. Then $\mathrm{Act}(S^\dagger) \subseteq cl(\mathrm{Act}(S))$.

Proof. Suppose this is not true and let $a \in A \setminus cl(A')$ such that $cl(a) \cap \mathrm{Act}(S^\dagger) \neq \emptyset$.

Consider the sets $S^\dagger_a, S^\dagger_{\bar{a}}$ and $S^\dagger_{\not a}$ that partition $S^\dagger$.

Pick any $s \in S$ and consider the polynomial $f = \mu(s)$. f doesn't contain the variable x_a.

Since $S^\dagger$ spans S, we have $f = \sum_{s' \in S^\dagger} c_{s'}\mu(s') = x_a P + \bar{x}_a Q + R$ where $x_a P = \sum_{s' \in S^\dagger_a} c_{s'}\mu(s')$, $\bar{x}_a Q = \sum_{s' \in S^\dagger_{\bar{a}}} c_{s'}\mu(s')$ and $R = \sum_{s' \in S^\dagger_{\not a}} c_{s'}\mu(s')$. Each of P,Q and R are independent of x_a. Hence we have $f[x_a := 0] = f = Q + R$ and $f[x_a := 1] = f = P + R$.

Also since $S^\dagger$ has A-loss recall, it follows from Lemma 4.21 that each of the sets $S^\dagger_a[\backslash a] \cup S^\dagger_{\not a}$ and $S^\dagger_{\bar{a}}[\backslash \bar{a}] \cup S^\dagger_{\not a}$ has A-loss recall. Hence each of the sets, $S^\dagger_a[\backslash a] \cup S^\dagger_{\not a}$ and $S^\dagger_{\bar{a}}[\backslash \bar{a}] \cup S^\dagger_{\not a}$ are A-loss recall spans of S. But then if at least one of $S^\dagger_a$ or $S^\dagger_{\bar{a}}$ is non-empty, this gives an A-loss recall span strictly smaller than $S^\dagger$. This is a contradiction and hence $\mathrm{Act}(S^\dagger) \subseteq cl(A')$.

□

Next we will see that, in order to find optimal A-loss recall span of a set S, it is enough to find optimal spans of each connected component.

Proposition 4.35. For a sequence set S, let $S = \uplus_i S_i$ be a decomposition of S into disjoint connected sets. For each i, let $S^\dagger_i$ be an optimal A-recall span of S_i. Then $S^\dagger = \cup_i S^\dagger_i$ is an optimal A-loss recall span of S

Proof. Since, each S_i is spanned by $S^\dagger_i$, it follows that $S^\dagger$ spans S. From Lemma 4.34 it also follows that for any distinct i and j, $S^\dagger_i$ and $S^\dagger_j$ are disconnected.

Lets assume that there is an A-loss recall span S' of S such that $|S'| < |S^\dagger| = \sum_i |S^\dagger_i|$. For each i, let S'_i be a minimal subset of S' that spans S_i. Since S'_i is minimal, from Lemma 4.34 it follows that $\mathrm{Act}(S'_i) \subseteq cl(\mathrm{Act}(S_i)$. This implies $S'_i \cap S'_j = \emptyset$ for any two distinct i and j. But then since $|S'| = \sum_i |S'_i|$, it must hold that for some i we have $|S'_i| < |S^\dagger_i|$. But this contradicts the optimality of $S^\dagger_i$. □

Using Proposition 4.35 we can break down the problem of finding optimal A-loss recall span of a set to its connected components. We make a small observation about the optimal A-loss recall span of a connected set.

Lemma 4.36. Let S be a connected sequence set and $S^\dagger$ be an optimal A-loss recall span of S. Then $S^\dagger = aS_1 \uplus \bar{a}S_2$ for some $a, \bar{a}, S_1, S_2$.

Proof. Given a connected sequence set S, any optimal A-loss recall span $S^\dagger$ of S is also connected. This is because if one could find a partition $S^\dagger = S_1^\dagger \uplus S_2^\dagger$, such that $S_1^\dagger$ and $S_2^\dagger$ are disconnected, then the subsets of S individually spanned by these two sets would be disconnected as well.

Since an A-loss recall span is a set with A-loss recall, it follows from Proposition 4.22 that an optimal A-loss recall span of a connected set would have the form $aS_1 \uplus \bar{a}S_2$ for some a. □

So for finding optimal span of connected sets we need to find some a and $\bar{a}$ that can lead every sequence of an optimal A-loss recall span. For this we can fix some a, $\bar{a}$ and then find the smallest optimal A-loss recall span that starts with either of these. We notice some optimal sub-structure in this method.

Proposition 4.37. Let S be a connected sequence set and let $a \in Act(S)$. Let $S_1^\dagger$ and $S_2^\dagger$ be some optimal A-loss recall spans of $S_a[\backslash a] \cup S_{\not a}$ and $S_{\bar{a}}[\backslash \bar{a}] \cup S_{\not a}$ respectively. Then $S^\dagger = aS_1^\dagger \cup \bar{a}S_2^\dagger$ is an A-loss recall span of S.

Furthermore, any A-loss recall span of S with all sequences starting with a or $\bar{a}$ has size at least $|S^\dagger|$.

Proof. First, we show that $S^\dagger$ is an A-loss recall span of S. Since each of $S_1^\dagger$ and $S_2^\dagger$ has A-loss recall, it follows from Lemma 4.21 that $S^\dagger$ also has A-loss recall. Now since $S_1^\dagger$ spans $S_a[\backslash a]$, it follows that $aS_1^\dagger$ spans S_a. Similarly, $aS_2^\dagger$ spans $S_{\bar{a}}$. Finally, since each of $S_1^\dagger$ and $S_2^\dagger$ spans $S_{\not a}$, for any sequence $s \in S_{\not a}$, we have $\mu(s) = \sum_{s_1' \in S_1^\dagger} c_{s_1'} \mu(s_1')$ and also $\mu(s) = \sum_{s_2' \in S_2^\dagger} c_{s_2'} \mu(s_2')$. As a result we have $\mu(s) = x_a \sum_{s_1' \in S_1^\dagger} c_{s_1'} \mu(s_1') + (1 - x_a) \sum_{s_2' \in S_2^\dagger} c_{s_2'} \mu(s_2') = \sum_{s_1' \in S_1^\dagger} c_{s_1'} \mu(as_1') + \sum_{s_2' \in S_2^\dagger} c_{s_2'} \mu(\bar{a}s_2')$. It follows that $S^\dagger$ spans $S_{\not a}$ as well. Hence, $S^\dagger$ is indeed an A-loss recall span of S.

Now for the second part, let us assume that S has an A-loss recall span S', with all sequences starting with a and $\bar{a}$ such that $|S'| < |S^\dagger|$.

Consider the sets S'_a and $S'_{\bar{a}}$ that partition S'.

For any $s \in S$, the polynomial $f = \mu(s)$ can be written as follows: $f = x_a P + \bar{x}_a Q$ where $x_a P = \sum_{s' \in S'_a} c_{s'} \mu(s')$ and $\bar{x}_a Q = \sum_{s' \in S'_{\bar{a}}} c_{s'} \mu(s')$.

When $s \in S_a$, since $\mu(s)$ contains x_a, we have $f[x_a := 0] = Q = 0$. This implies $f = x_a P$. Again when $s \in S_{\not a}$, we have $f[x_a := 1] = P = f$. In either of these cases, $s[\backslash a]$ is spanned by $S'_a[\backslash a]$. Hence the set $S_a[\backslash a] \cup S_{\not a}$ is spanned by $S'_a[\backslash a]$.

Also since S'_a has A-loss recall it follows from Lemma 4.21 that $S'_a[\backslash a]$ also has A-loss recall. This implies that $S'_a[\backslash a]$ is an A-loss recall span of $S_a[\backslash a] \cup S_{\not a}$. By similar arguments it can be shown that $S'_{\bar a}[\backslash \bar a]$ is an A-loss recall span of $S_{\bar a}[\backslash \bar a] \cup S_{\not a}$.

Now since $|S'_a[\backslash a]| + |S'_{\bar a}[\backslash \bar a]| = |S'| < |S^\dagger| \leq |S^\dagger_1| + |S^\dagger_2|$, either $|S'_a[\backslash a]| < |S^\dagger_1|$ or $|S'_{\bar a}[\backslash \bar a]| < |S^\dagger_2|$. But this contradicts the optimality of either $S^\dagger_1$or $S^\dagger_2$. Hence $|S'| \geq |S^\dagger|$. □

Proposition 4.37 gives us a way to find the optimal A-loss recall span of a connected set once we fix a leading action of a span. We can try all possible leading actions and check which one gives an optimal A-loss recall span. However there are cases, where we can correctly guess the leading action of an optimal span immediately without the need to try all possibilities. This is the case when for some action $a, \bar a$, $S_{\not a} = \emptyset$.

Proposition 4.38. Let S be a connected sequence set such that for some a, $S_{\not a} = \emptyset$. Let $S^\dagger_1$ and $S^\dagger_2$ be some optimal A-loss recall spans of $S_a[\backslash a]$ and $S_{\bar a}[\backslash \bar a]$ respectively. Then $S^\dagger = aS^\dagger_1 \cup \bar a S^\dagger_2$ is an optimal A-loss recall span of S.

Proof. We will show that there is an optimal A-loss recall span of S in which each sequence starts with a or $\bar a$. Then it follows from Proposition 4.37 that $S^\dagger$ is an optimal A-loss recall span of S.

Let $A' = cl(\mathrm{Act}(S))$. We prove by induction on the size of $A' \subseteq A$.

For $A' = \{a, \bar a\}$, the statement is trivially true. Suppose the statement is true for any $A' \subseteq A$ with $|A'| = 2k$ for $k < m$. Now let S_0 be a connected sequence set over A' with $|A'| = 2m$ and let $S^\dagger_0$ be an optimal A-loss recall span of S_0. Since S_0 is connected, it follows from Lemma 4.36 for some $b \in A'$ all sequences in an optimal A-loss recall span of S_0 begins with an action in $cl(b)$. Now applying Proposition 4.37 let $S^\dagger_0$ be an optimal A-loss recall span of S_0 of the form $bS_1 + \bar b S_2$ such that S_1, S_2 are optimal A-loss recall spans of $S_b[\backslash b] \cup S_{\not b}$ and $S_{\bar b}[\backslash \bar b] \cup S_{\not b}$ respectively. If $b = a$ we are done.

Otherwise since every sequence in each of $S_b[\backslash b] \cup S_{\not b}$ and $S_{\bar b}[\backslash \bar b] \cup S_{\not b}$ has a or $\bar a$ in it, from induction hypothesis we can assume that all sequences in S_1 and S_2 start with a or $\bar a$. So all sequences in $S^\dagger_0$ starts with one of $ba, b\bar a, \bar b a$ or $\bar b \bar a$. Now by applying Lemma 4.21 it follows that each of ${S^\dagger_0}_a[\backslash a]$ and ${S^\dagger_0}_{\bar a}[\backslash \bar a]$ has A-loss recall respectively. Again applying Lemma 4.21, it follows each of $a{S^\dagger_0}_a[\backslash a]$ and

$\bar{a}S_0^{\dagger}{}_{\bar{a}}[\backslash\bar{a}]$ has A-loss recall respectively. Finally again applying Lemma 4.21 one last time it follows that $aS_0^{\dagger}{}_{a}[\backslash a] \cup \bar{a}S_0^{\dagger}{}_{\bar{a}}[\backslash\bar{a}]$ has A-loss recall. This set is just a shuffle of $S_0^{\dagger}$ and hence it is the desired the A-loss recall span of S_0. This completes the proof.

□

Exponential size optimal A-loss recall span

We will provide a class of sequence sets for which the optimal A-loss recall span is of exponential size ignoring some polynomial factors.

Proposition 4.39. For every $n > 0$, there exists a sequence set $S_n \subseteq Seq(A_n)$ with $|A_n| = 2n$ and $|S_n| = O(n^2)$ such that the size of an optimal A-loss recall span of S_n is $O(2^n)$.

Proof. For $n > 0$, let $A_n = cl(\{a_1, \ldots, a_n\})$ and let $S_n = \{\epsilon, a_1, \ldots, a_n\} \cup \{a_i a_j \mid i < j\}$. Since $\mu(S_n)$ is symmetric w.r.t. all a_i, w.l.o.g we can assume that it has an optimal A-loss recall span starting with a_n. It follows from Proposition 4.37 that if S' and S'' are optimal A-loss recall spans of $S_{a_n}[\backslash a_n] \cup S_{\not a_n}$ and $S_{\bar{a}_n}[\backslash \bar{a}_n] \cup S_{\not a_n}$ respectively, then $a_n S' \cup \bar{a}_n S''$ is an optimal A-loss recall span of S. Now we have $S_{\not a_n} = \{\epsilon, a_1, \ldots, a_{n-1}\} \cup \{a_i a_j \mid i < j < n\} = S_{n-1}$. As a result both $S_{a_n}[\backslash a_n] \cup S_{\not a_n}$ and $S_{\bar{a}_n}[\backslash \bar{a}_n] \cup S_{\not a_n}$ are essentially S_{n-1}.

Let T_n be the size of an optimal A-loss recall span of S_n. From previous arguments we have the recurrence $T(n) = 2T(n-1)$. Hence we have $T(n) = O(2^n)$. □

Now we provide a recursive algorithm to compute an optimal A-loss recall span.

Algorithm 3 Compute optimal A-loss recall spanning set

Input : S

if $S = \{\epsilon\}$ **then**
  $S' \leftarrow \{\epsilon\}$
  $\mathcal{M}' \leftarrow$ Empty matrix with row-column co-ordinates S, S'
  $\mathcal{M}'_{[\epsilon,\epsilon]} = 1$
  return $(S', \mathcal{M}')$
else
  if S is connected **then**
    if $\exists a$ such that $S_{\not a} = \emptyset$ **then**
      $(S_1, \mathcal{M}_1) \leftarrow$ A-loss Recall Span$(S_a[\backslash a])$
      $(S_2, \mathcal{M}_2) \leftarrow$ A-loss Recall Span$(S_{\bar{a}}[\backslash \bar{a}])$
      $S' \leftarrow aS_1 \cup \bar{a}S_2$
      $\mathcal{M}' \leftarrow$ Empty matrix with row-column co-ordinates S, S'
      for $s \in S, s' \in S'$ **do**
        if $s \in S_a, s' \in aS_1$ **then**
          $\mathcal{M}'_{[s,s']} = \mathcal{M}_{1\ [s[\backslash a], s'[\backslash a]]}$
        else if $s \in S_{\bar{a}}, s' \in \bar{a}S_2$ **then**
          $\mathcal{M}'_{[s,s']} = \mathcal{M}_{2\ [s[\backslash \bar{a}], s'[\backslash \bar{a}]]}$
        else $\mathcal{M}'_{[s,s']} = 0$
      return $(S', \mathcal{M}')$
    else
      for $a \in \mathrm{Act}(S)$ **do**
        $(S_1^a, \mathcal{M}_1^a) \leftarrow$ A-loss Recall Span$(S_{\not a} \cup S_a[\backslash a])$
        $(S_2^a, \mathcal{M}_2^a) \leftarrow$ A-loss Recall Span$(S_{\not a} \cup S_{\bar{a}}[\backslash \bar{a}])$
      $a = argmin_a \min_{a \in \mathrm{Act}(S)} |S_1^a| + |S_2^a|$
      $S' \leftarrow aS_1^a \cup \bar{a}S_2^a$
      $\mathcal{M}' \leftarrow$ Empty matrix with row-column co-ordinates S, S'
      for $s \in S, s' \in S'$ **do**
        if $s \in S_a \cup S_{\not a},\ s' \in aS_1^a$ **then**
          $\mathcal{M}'_{[s,s']} = \mathcal{M}^a_{1\ [s[\backslash a], s'[\backslash a]]}$
        else if $s \in S_{\bar{a}} \cup S_{\not a},\ s' \in \bar{a}S_2^a$ **then**
          $\mathcal{M}'_{[s,s']} = \mathcal{M}^a_{2\ [s[\backslash \bar{a}], s'[\backslash \bar{a}]]}$
        else $\mathcal{M}'_{[s,s']} = 0$
      return $(S', \mathcal{M}')$
  else
    $S = \uplus S_i$ where each S_i is connected
    $(S'_i, \mathcal{M}_i) \leftarrow$ A-loss Recall Span(S_i)
    $\mathcal{M}' \leftarrow$ Empty matrix with row-column co-ordinates S, S'
    $S' \leftarrow \cup S'_i$
    for $s \in S, s' \in S'$ **do**
      if $s \in S_i, s' \in S'_i$ **then**
        $\mathcal{M}'_{[s,s']} = \mathcal{M}_{i[s,s']}$
      else $\mathcal{M}'_{[s,s']} = 0$
    return $(S', \mathcal{M}')$

Theorem 4.40. *The function A-loss Recall Span in Algorithm 3 on input S returns an optimal A-loss recall span $S^\dagger$ of S and the corresponding span matrix $\mathcal{M}^S_{S^\dagger}$.*

Proof. The trivial case where only ϵ is present in the set is taken care of in line 1 returning the one-dimensional identity matrix as the span matrix.

When S is disconnected, from Proposition 4.23 it follows that the computation can be decomposed into computing smallest A-loss recall spans of each connected component. In line 35, the algorithm finds the optimal A-loss recall span and corresponding span matrix of individual connected components. The final A-loss recall span is the union of all individual A-loss recall spans. Since two distinct connected components are disconnected, it follows from Lemma 4.34 that their optimal A-loss recall spans are also disconnected. Hence in the final span matrix, the values corresponding to a component S_i only concerns the sequences with actions in $\mathrm{Act}(S_i)$ and doesn't touch other blocks in the matrix. This is why for $s \in S_i, s' \in S'_i$ we do $\mathcal{M}'_{[s,s']} = \mathcal{M}_{i[s,s']}$ and for two distinct connected components S_i and S_j, and for sequences $s_i \in S_i$ and $s_j \in S'_j$, $\mathcal{M}'_{[s_i,s_j]} = 0$.

Now when S is connected and for some a, $cl(a)$ has intersection with all sequences, it follows from Proposition 4.38 there is an optimal A-loss recall in which all sequences start with actions from $cl(a)$. This case is taken care in line 8 by recursively calling the function. The value of span matrix for the part S_a and $S_{\bar{a}}$ are filled independently.

The only case remaining is when S is connected but $S_{\not a}$ is non-empty for every a. From Lemma 4.36 it follows that the A-loss recall span must start with actions from $cl(a)$ for some a. It also follows that the size of such an A-loss recall span would be $|S^a_1| + |S^a_2|$. The algorithm iterates over all actions to pick the action with minimum size A-loss recall span in line 24. Following Proposition 4.37, for each a as a possible candidate, it computes the optimal A-loss recall span starting with $cl(a)$ recursively. Once the actions a is found in this process, the final spanning set in constructed as in Proposition 4.37. In the final span matrix, for a sequence $s \in S_{\not a}$, two position are filled in $\mathcal{M}'$ since to span it we need terms from both S^a_1 and S^a_2. Otherwise for terms in either of S_a or $S_{\bar{a}}$ only one position in $\mathcal{M}'$ corresponding to sequence from either of S^a_1 or S^a_2 is filled.

□

Now we will see the worst possible size of a game constructed from an optimal A-loss recall span in terms of the size of the original game.

Lemma 4.41. Let $G = (S, \Lambda)$ be a one-player Max game and let $G' = (S', \Lambda')$ be the equivalent game to G with $S = cl(S^\dagger)$ where $S^\dagger$ is an optimal A-loss recall span of S. Then $|G'| = O(|S^\dagger| \log K)$ where $K = \sum_{s \in S} \Lambda(s)$.

Proof. It follows from Algorithm 3 that the span matrix only contains 0 or 1 entries. It also follows that $S^\dagger$ is a complete set. Hence following the construction of G' in Theorem 4.30, the Λ' value of any $s' \in S'$ can be at most $K = \sum_{s \in S} \Lambda(s)$. Hence $|G'|$ is at most $|S^\dagger| \log K$. □

We know that the size of a game $G = (S, \Lambda)$ is $\sum_{s \in S} \log \Lambda(s)$. Lemma 4.41 justifies the approach of finding the optimal A-loss recall span and constructing the equivalent A-loss recall game since it just leaves out a polynomial factor in the size of the original game.

Worst Case Runtime Example

We will now show a class of sequence sets for which Algorithm 3 takes exponential time to terminate.

Proposition 4.42. For every $n > 0$, there exists a sequence set $S_n \subseteq Seq(A_n)$ with $|A_n| = 2n$, $|S_n| = O(n)$ such that S_n has an optimal A-loss recall span of size $O(n^2)$ but Algorithm 3 takes exponential time to find an optimal A-loss recall span of S_n.

Proof. Consider the action set $A_n = \{a_1, \bar{a}_1, \ldots, a_n, \bar{a}_n\}$.

Let $S_n = \{\epsilon, a_1, \ldots, a_n, a_1 \ldots a_n\}$ over A_n. First we show that this has an A-loss recall span of size $O(n^2)$.

Since $\mu(S_n)$ is symmetric with respect to all a_i, there is an optimal A-loss recall span of S_n starting with each a_i. Let $S_n^\dagger$ be an optimal A-loss recall span of S_n starting with a_n. It follows from Proposition 4.37 that if S' and S'' are optimal A-loss recall spans of $S_{a_n}[\backslash a_n] \cup S_{\not a_n}$ and $S_{\bar{a}_n}[\backslash \bar{a}_n] \cup S_{\not a_n}$ respectively, then $a_n S' \cup \bar{a}_n S''$ is an optimal A-loss recall span of S.

Now we have $S_{a_n}[\backslash a_n] \cup S_{\not a_n} = S_{n-1}$ and $S_{\bar{a}_n}[\backslash \bar{a}_n] \cup S_{\not a_n} = \{\epsilon, a_1, \ldots, a_{n-1}\}$. The set $S_{\bar{a}_n}[\backslash \bar{a}_n] \cup S_{\not a_n}$ already has A-loss recall and hence it is the optimal A-loss recall span of itself, i.e. $|S''| = n$.

Let $T(n)$ be the the size of the A-loss recall span of S_n. It follows that T_n follows the recursion $T(n) = T(n-1) + O(n)$. Solving this gives $T(n) = O(n^2)$.

However the function A-loss Recall Span in Algorithm 3 takes exponential time to find the optimal A-loss recall span. This is because since the set is symmetric, it explores recursively each of the $n!$ sequences of a_i's to find the minimal span of S_n. □

Example 4.13. The game G in Fig. 4.7a doesn't have A-loss recall and $S = \{a, b, c, abc\}$ is the set of sequences with non-zero payoffs.

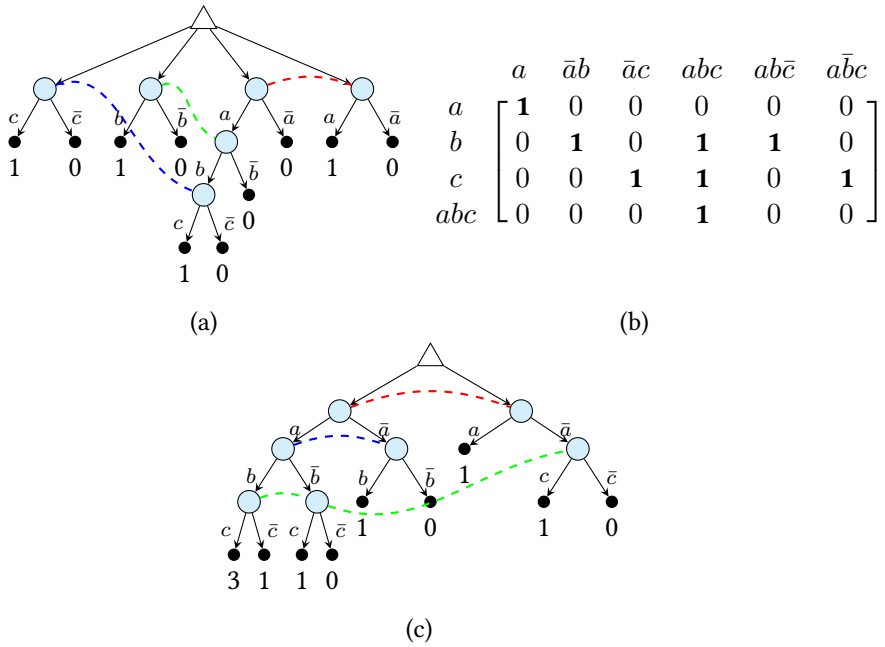


Figure 4.7: The game G in Fig. 4.7a doesn't have A-loss recall and $S = \{a, b, c, abc\}$ is the set of sequences with non-zero payoffs. $S^\dagger = \{a, \bar{a}b, \bar{a}c, abc, ab\bar{c}, a\bar{b}c\}$ is an optimal A-loss recall span of S and the matrix in Fig. 4.7b is the span matrix $\mathcal{M}^S_{S^\dagger}$. The game in Fig. 4.7c is a game G' which is equivalent to G constructed using $\mathcal{M}^S_{S^\dagger}$

$S^\dagger = \{a, \bar{a}b, \bar{a}c, abc, ab\bar{c}, a\bar{b}c\}$ is an optimal A-loss recall span of S obtained using Algorithm 3 and the matrix in Fig. 4.7b is the corresponding span matrix $\mathcal{M}^S_{S^\dagger}$. The game in Fig. 4.7c is a game G' which is equivalent to G constructed using $\mathcal{M}^S_{S^\dagger}$. Any leaf with payoff zero in the G' is a sequence that was added in the closure.

Decision Problem 4.1 (A-Loss Span)**.** Given a sequence set $S \subseteq Seq(A_{\mathsf{Max}})$ and a positive integer k, does there exist a sequence set $S' \subseteq Seq(A_{\mathsf{Max}})$ such that S' is an A-loss recall span of S and $|S'| \leq k$?

We will show that this decision problem is in NP. For that we will use a kind of well-structured set.

Definition 37 (Strongly branching set)**.** The trivial set $S = \{\epsilon\}$ is a strongly branching set. $S \subseteq Seq(A)$ is a strongly branching set if there exists $a, \bar{a} \in A$ such that there is a partition of $S = aS_1 \uplus \bar{a}S_2$ such that (i) aS_1 and $\bar{a}S_2$ are non-empty (ii) S_1 and S_2 are each strongly branching sets.

Lemma 4.43. Let S be a sequence set with A-loss recall. Then $\sum_{s\in S}\mu(s) = 1$ iff S is a strongly branching set.

Proof. First we will prove the backward direction[4] by induction on the number of actions. When $S = \{\epsilon\}$ recall that $\mu(\epsilon) = 1$. Now for any non-trivial strongly branching set S, from definition we have $S = aS_1 \uplus \bar{a}S_2$ where each of S_1 and S_2 are strongly branching. We have $\sum_{s\in S}\mu(s) = x_a(\sum_{s\in S_1}\mu(s)) + (1-x_a)(\sum_{s\in S_2}\mu(s))$. By induction hypothesis the statement holds for S_1 and S_2 and hence this equals 1. Hence this extends to S as well.

For the the forward direction we will again use induction on the number of actions. The statement is true when $A = \emptyset$. Now suppose S be a sequence set over non-trivial action set.

For any set S', if $\sum_{s\in S'}\mu(s)$ is a constant value then this value is at least 1 due to the fact that any $\mu(s)$ has either no constant terms or 1 in its full expansion. If S has at least two connected components S_1 and S_2, since $\mu(S_1)$ and $\mu(S_2)$ have no variable in common, $\sum_{s\in S}\mu(s)$ would become strictly more than 1. Hence we can conclude that S must be connected.

Since S is connected and has A-loss recall, from Proposition 4.22 it follows that $S = aS_1 \uplus \bar{a}S_2$ where each of S_1 and S_2 has A-loss recall. Let $Z_i = \sum_{s\in S_i}$. Then we have $\sum_{s\in S}\mu(s) = x_aZ + (1-x_a)Z' = 1$, which implies each of Z and Z' must equal 1. Hence by induction hypothesis each of S_1 and S_2 are strongly branching sets. As a result S is also a strongly branching set.

□

Proposition 4.44. The decision problem A-Loss Span is in NP.

Proof. To prove this we will use an optimal A-loss recall span $S^\dagger$ of S and the span matrix $\mathcal{M}^S_{S^\dagger}$ as a short certificate. If there exists an A-loss recall span S' of size at most k, we observe that $\mathcal{M}^S_{S^\dagger}$ is of size at most $k|S|$ which is polynomially bounded in k and $|S|$.

Now we need to provide an efficient algorithm to verify that $S^\dagger$ has A-loss recall and $S^\dagger$ indeed spans S. The first requirement can be verified in polynomial time by checking if for every pair of sequences $s_1, s_2 \in S^\dagger$, it holds that $s_1 \sim_A s_2$. For the second objective, we make some observations on the span matrix. Observe that it follows from Algorithm 3 that $\mathcal{M}^S_{S^\dagger}$ only has 0 and 1 as entries. So for

[4] Observe that any strongly branching set always has A-loss recall. A-loss recall assumption is redundant in this case

every sequence $s \in S$ we have to check if $\mu(s)$ equals sum of all $\mu(s')$ such that $\mathcal{M}^{S}_{S^\dagger\ [s,s']} = 1$. Now verifying this via full expansion might produce exponentially many terms. Instead we will exploit the structure of the spanning sequences. Let $Supp(s) \subseteq S^\dagger$ be defined as $Supp(s) = \{s' | \mathcal{M}^{S}_{S^\dagger\ [s,s']} = 1\}$.

From the construction of $Supp(s)$ in Algorithm 3 it follows that $\mathrm{Act}(s) \subseteq \mathrm{Act}(s')$ for every $s' \in Supp(s)$. Hence for every $s' \in Supp(s)$, $\mu(s') = \mu(s)\mu(s'')$ for some s''. Let $S' = Supp(s)[\setminus \mathrm{Act}(s)]$ be the sequences in $Supp(s)$ removing all actions from s. We just need to check if the $\sum_{s \in S'} \mu(s) = 1$. Since $Supp(s)$ has A-loss recall, by repeated application of Lemma 4.21 it follows that S' also has A-loss recall. Now from Lemma 4.43 we need to check if S' is strongly branching. Checking if a set is strongly branching can be done recursively in PTIME. This completes the proof. □

4.5 A word on perfect recall spans and shuffles

So far we have seen A-loss recall shuffles and A-loss recall spans of sequence sets. Similar question on sequence sets can be asked with respect to perfect recall as well. In this section we address these questions.

First as we did for A-loss recall, we observe some properties of sequence sets with perfect recall.

Structure of sets with perfect recall

Proposition 4.45. Let S be a sequence set over A with perfect recall. Then the following statements are true:

1. If S is connected, then $S = aS_1 \uplus \bar{a}S_2$ for some $a \in A$ such that S_1 and S_2 are not connected and each of S_1 and S_2 has perfect recall.
2. If $S = \uplus_i S_i$ be a decomposition of S into disjoint connected components then, each S_i has perfect recall

Proof. Since any subset of a set with perfect recall also has perfect recall, the second statement follows.

Now we prove the first statement. Without loss of generality let $s = as'$ be a sequence in S for some $a \in A$. Consider the sets $S_a, S_{\bar{a}}$ and $S_{\not a}$. Since S is connected $\mathrm{Act}(S_a \cup S_{\bar{a}}) \cap \mathrm{Act}(S_{\not a}) \neq \emptyset$. But this violates perfect recall when $S_{\not a} \neq \emptyset$ since sequences in $S_{\not a}$ do not start with a or $\bar{a}$. Also we have $\mathrm{Act}(S_a[\setminus a]) \cap \mathrm{Act}(S_{\bar{a}}[\setminus \bar{a}]) = \emptyset$ because otherwise this also violates perfect recall. Finally similar to Lemma 4.21 it can be shown that aS' has perfect recall implies S' also has perfect recall. Hence S_1 and S_2 also have perfect recall. □

We saw in Theorem 4.29 that any sequence set S has an A-loss recall span. The natural question with respect to perfect recall is : does every sequence set has a perfect recall span?

Also in the case of A-loss recall, we saw that shuffles are special cases of spans and there exists sequence sets (Example 4.8) with no A-loss recall shuffle. We first examine if spans are more powerful than shuffles in the case of perfect recall. It turns out that perfect recall spans are no move advantageous than perfect recall shuffles.

Proposition 4.46. For a sequence set S, if S has a perfect recall span, then S also has a perfect recall shuffle.

Proof. We prove a stronger statement. We show that any optimal perfect recall span $S^\dagger$ of S is in fact a perfect recall shuffle of S. For $s \in S$, let $S_s \subseteq S^\dagger$ be a minimal subset that spans $\{s\}$, i.e., no proper subset S'' of S_s generates s. We have $\mu(s) = \sum_{s' \in S_s} c_{s'} \mu(s')$. Let $f = \mu(s)$. For any action $a \in \mathrm{Act}(s)$, we have $f[x_a := 0] = 0$. Hence x_a is a factor of each $\mu(s')$. Consequently $\mu(s)$ is a factor of each $\mu(s')$. Now suppose for some $s' \in S_s$ and $b \in \mathrm{Act}(s')$, $b \notin \mathrm{Act}(s)$. Then $f[x_b := 0]$ is still $\mu(s)$ but $\mu(s')[x_b := 0] = 0$. But then $\mu(s)$ can be written as combination of monomials of terms in $S \setminus \{s'\}$ which contradicts the minimality of S_s. Hence no sequence has actions outside $\mathrm{Act}(s)$. From minimality of S_s, this implies $S_s = \{s^\dagger\}$ where $s^\dagger$ is a permutation of s. Since $S^\dagger$ is optimal it follows that it is a shuffle of S. This completes the proof. □

Hence the question of existence of perfect recall span of an arbitrary sequence set boils down to asking existence of just perfect recall shuffle. But we have already seen example of sequence sets with no A-loss recall shuffles (Example 4.8). Since perfect recall shuffles are also A-loss recall shuffle, it follows that there exist sequence sets with no perfect recall shuffle (or span).

In fact we will further strengthen this statement. We will show that any one-player rulebook with imperfect recall cannot have a perfect recall shuffle.

Proposition 4.47. S be a rulebook over A without perfect recall. Then S cannot have a perfect recall shuffle.

Proof. Since S doesn't have perfect recall, there exist $s_1, s_2 \in S$ with $s_1 \not\sim_P s_2$. W.L.O.G. we can say that for some $a \in \mathrm{Act}(S)$, $s_1 = s_1' a s_1''$, $s_2 = s_2' a s_2''$ and there exist another action $b \in \mathrm{Act}(s_1')$ such that $b \notin \mathrm{Act}(s_2')$. Since S is complete, one can also find a sequence $s_3 = s_1' \bar{a} s_3''$ in S.

Let $S^\dagger$ be a perfect recall shuffle of S and let $s_1^\dagger$ be the permutation of s_1 present in $S^\dagger$. By using the same reasoning as Lemma 4.17 it follows from the definition

of perfect recall that any two sequences in $S^\dagger$ are of the form ss_0 and ss_0' where $cl(\mathrm{Act}(s_0)) \cap cl(\mathrm{Act}(s_0')) = \emptyset$.

Now there can be two cases.

Case 1: $b \notin \mathrm{Act}(s_2)$

In this case, enforced by perfect recall constraints between s_1 and s_2, a must precede b in $s_1^\dagger$. On the other hand for similar constraints between s_1 and s_3, b precedes a in s_1 which is a contradiction. Hence S cannot have a perfect recall shuffle in this case.

Case 2: $b \in \mathrm{Act}(s_2)$

In this case, since $b \notin s_2'$, it must be that $b \in s_2''$, i.e. $s_2 = s_2' a w_2 b w_2''$. Since S is complete one can find $s_4 = s_2' a w_2 \bar{b} w_4$. Again enforced by perfect recall constraints between s_1 and s_4, a must precede b in $s_1^\dagger$. And for the same reason, for s_1 and s_3, b precedes a in s_1. Hence for this case also S cannot have a perfect recall shuffle.

□

Proposition 4.47 tells us we cannot use perfect recall shuffles to simplify games from their sequence sets. However as seen for A-loss in Remark 4.18, this doesn't rule out completely the possibility of using perfect recall shuffles as a heuristic. Since sequences with zero payoff doesn't contribute to the payoff polynomial, one can work only with the sequences with non-zero payoff and end up with a perfect recall shuffle of this subset. In essence, this is a way to get rid of the sequences in the rulebook that makes it impossible to have a perfect recall shuffle.

In the next section we will see how to simplify games by directly using the payoff polynomial.

4.6 Simplification via payoff polynomials

So far in order to simplify the structure of a game we worked with sequence sets because the sequence set generates the set of leaf monomials that linearly combine to give the payoff polynomial. We didn't take into account the payoff attached to each sequence. So it can be seen as simplification of the game at a structural level.

In this section, we will start directly from the payoff polynomial and work our way towards finding the structure of a possible game with this payoff polynomial. A polynomial can be the payoff polynomial of a game with perfect recall and at the same time of a game with imperfect recall. We see this in the following example.

Example 4.14. Consider the game $G = (S, \Lambda)$ in Fig. 4.8a with sequence set $S = \{ca, c\bar{a}, \bar{c}, ba, b\bar{a}, \bar{b}\}$ and $\Lambda = \{ca : 2, c\bar{a} : 1, \bar{c} : 1, ba : 1, b\bar{a} : 3, \bar{b} : 3\}$. S doesn't have A-loss recall. Also, S is connected and it follows from Proposition 4.25

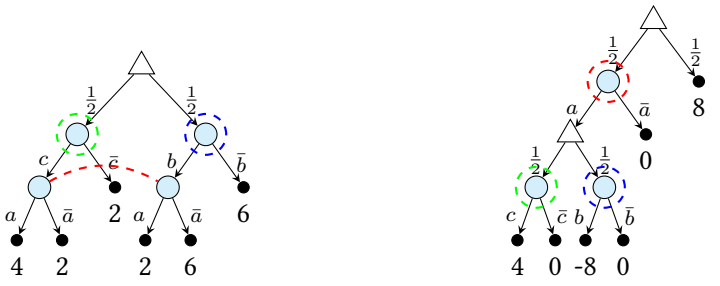


Figure 4.8: Two games for the same polynomial $x_a x_c - 2x_a x_b + 4$

that S does not have an A-loss recall shuffle either. The payoff polynomial of G is $x_a x_c - 2x_b x_a + 4$. It is also the payoff polynomial of a perfect recall game G' which is given in Fig. 4.8b.

If we take the sequence set S in Example 4.14 and apply Algorithm 3, it follows from Proposition 4.47 that we would never end up with a sequence set with perfect recall. However, the payoff polynomial $x_a x_c - 2x_b x_a + 4$ reveals more about the game and one can construct an equivalent perfect recall game.

Now while working with payoff polynomials, a possible approach might be to take the set of individual monomials in the polynomial (ignoring the payoffs), and work with the corresponding sequence set for simplification. But this approach fails in general as this sequence set obtained from the monomials might not have perfect recall shuffle. This is possible even when the monomials linearly combine to form the payoff polynomial of a game with perfect recall. We see this in the following example.

Example 4.15. Consider the polynomial $f = x_a + 2x_b - 2x_a x_b$ and its' set of monomials $\{x_a, x_a x_b, x_b\}$. The corresponding sequence set is $\{a, ab, b\}$ and it does not have perfect recall shuffle. However the polynomial $f = x_a + 2x_b - 2x_a x_b$ comes from a perfect recall game. This is because f can be rewritten as $x_a + 2(1 - x_a)x_b$ and one can see that this is generated by the sequence set $\{a, \bar{a}b\}$ that has perfect recall.

In Example 4.15, it is not evident from the expanded form of a polynomial, whether it comes from a game with perfect recall. Our goal will be to characterize polynomials that arise from one-player perfect recall games.

In this section we will only work with one-player games. Also since here we are concerned only with non-absentminded games, the payoff polynomials will

be multi-linear and here we will only deal with such polynomials. It is possible that in the process of simplifying polynomials, we will require to compute the full expansion of polynomials. For this reason this method might not give an efficient procedure since the expanded form can possibly have exponentially many terms. However this opens the path for finding heuristics for game simplification.

4.6.1 Turning some games into games with perfect-recall

In this section we will give a characterization of all polynomials that come from one-player games with perfect recall.

Recall that $\bar{x}$ denotes $1 - x$. For a polynomial g let $var(g)$ denote the set of all variables in g. For a polynomial written in the expanded form, a term is a monomial along with its' co-efficient. For a single term t in a polynomial, $var(t)$ is defined similarly.

Definition 38 (x-decomposition)**.** Given a polynomial $f(X)$ over X and $x \in X$, an *x-decomposition* of f is a re-writing of f in the form $xf_0(X_0)+\bar{x}f_1(X_1)+f_2(X_2)$ such that $x \notin X_0 \cup X_1 \cup X_2$.

An x-decomposition of f given by $xf_0(X_0) + \bar{x}f_1(X_1) + f_2(X_2)$ is said to be *disconnected* if X_0, X_1 and X_2 are mutually disjoint.

Example 4.16. For $f = 3xy + 2xz - 2z + 1$, $x(3y + z) - \bar{x}(z) + (2 - z)$ is an x-decomposition of f which is not disconnected due to z. On the other hand $x(3y) - \bar{x}(2z) + 1$ is a disconnected x-decomposition of f.

Canonical x-decomposition

For a polynomial f over X, we will provide a canonical x-decomposition of f with respect to some $x \in X$. In the expanded form, f can be uniquely written as $Ax + B$, where Ax is the sum of all the terms in f containing x and B is the sum of all the terms not containing x.

Let f_1 be the sum of all terms t in B such that $-t$ is also a term in A. Let $f_0 = A+f_1$ and $f_2 = B - f_1$.

Then we have $A = f_0 - f_1, B = f_2 + f_1$ and hence $f = Ax + B = xf_0 + \bar{x}f_1 + f_2$.

Claim : $xf_0 + \bar{x}f_1 + f_2$ is an x-decomposition of f.

Proof. $x \notin var(B)$, hence $x \notin var(f_1)$. Since $x \notin var(A)$, it follows that $x \notin var(f_0)$. Finally, since $var(f_2) \subseteq var(B)$, $x \notin var(f_2)$. Hence $xf_0 + \bar{x}f_1 + f_2$ is an x-decomposition of f. □

The x-decomposition $xf_0 + \bar{x}f_1 + f_2$ obtained in this manner is a canonical x-decomposition of f.

In Example 4.16 the canonical x-decomposition of f is $x(3y) - \bar{x}(2z) + 1$.

Next we will define recursively a class of polynomials called *perfect recall polynomials* without mentioning games.

A *trivial polynomial* is a constant polynomial without any variables.

Definition 39 (Perfect recall polynomials)**.** Every trivial polynomial is a perfect recall polynomial. A polynomial f over variable set X is a perfect recall polynomial if there exists an $x \in X$ and an x-decomposition $xf_0(X_0) + \bar{x}f_1(X_1) + f_2(X_2)$ of f such that (1) it is disconnected and (2) each $f_i(X_i)$ is a perfect recall polynomial.

Recall that for $x \in X$ and some $b \in \mathbb{R}$, $f[x := b]$ is the resulting polynomial by substituting x with b in f. For $x, y \in X$ and $b \in \{0, 1\}$, x is said to *cancel* y in f with b if y vanishes in $f[x := b]$.

Definition 40 (Polynomial Cancellation Graph)**.** A polynomial cancellation graph (PCG) of a polynomial f over variable set X denoted by Γ_f is a directed weighted graph over vertex set X, where there is an edge (x, y) with weight $b \in \{0, 1\}$ iff x cancels y with b. [5]

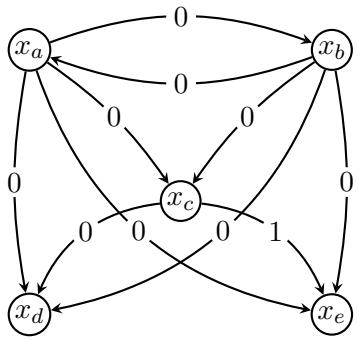
Example 4.17. The polynomial cancellation graph of the polynomial $f = 2x_a x_b x_c x_d + x_a x_b x_e - x_a x_b x_c x_e$ is given in Fig. 4.9a and that of $g = x_a x_b + x_a x_c - 2x_a x_b x_c$ is given in Fig. 4.9c.

Now we will see equivalent characterization of payoff polynomials of perfect recall games in terms of perfect recall polynomials and polynomial cancellation graph.

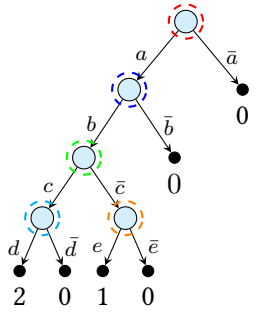
Theorem 4.48. *The following statements are equivalent.*

1. *$f(X)$ is a perfect recall polynomial*
2. *There is a one-player game G in extensive form where Max has perfect recall such that $f(X)$ is the payoff polynomial of G.*
3. *For any $x, y \in X$ such that xy is present in some monomial in the expanded form of $f(X)$, Γ_f has a weighted edge between x and y in at least one direction.*

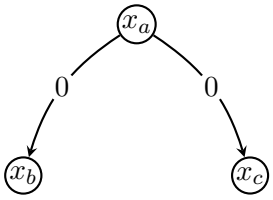
[5] PCG has similarity with information set forest used in [KM92] for analyzing structure of observations in games with perfect recall. Essentially each variable comes from an observation and PCG is the construction of that forest from the payoff polynomial.



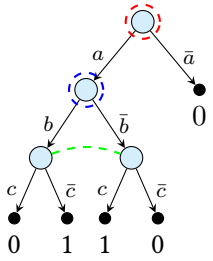
(a) Γ_f for the polynomial $f = 2x_ax_bx_cx_d + x_ax_bx_e - x_ax_bx_cx_e$



(b) A perfect recall game with payoff polynomial $f = 2x_ax_bx_cx_d + x_ax_bx_e - x_ax_bx_cx_e$



(c) Γ_g for the polynomial $g = x_ax_b + x_ax_c - 2x_ax_bx_c$



(d) A game without perfect recall for the polynomial $g = x_ax_b + x_ax_c - 2x_ax_bx_c$

Figure 4.9: Polynomial Cancellation Graph of polynomials and corresponding games

Before proceeding to prove this theorem we will prove some lemmas.

Lemma 4.49. Let $f(X)$ be a multi-linear polynomial with $x, y \in X$. Then x cannot cancel y with both 0 and 1 in f.

Proof. f can be uniquely written as $yf_1 + f_2$, where yf_1 is the sum of all the terms in f containing y and f_2 sum of those not containing y. Suppose x cancels y with both 0 and 1. We know that for a polynomial g, $g[x := b] = 0$ iff $(x - b)$ is a factor of g. Hence both x and $1 - x$ are factors of f_1. But then f_1 is no longer multi-linear which contradicts the fact that f is multi-linear. This completes the proof. □

Lemma 4.50.
Let $xf_0(X_0) + \bar{x}f_1(X_1) + f_2(X_2)$ be an x-decomposition of f. Then this decomposition is disconnected iff for $b \in \{0, 1\}$, $X_b = \{y \mid x \text{ cancels } y \text{ with } b \text{ in } f\}$.

Proof. Suppose $xf_0(X_0) + \bar{x}f_1(X_1) + f_2(X_2)$ is disconnected. Then X_i's are mutually disjoint. Hence with b, x cancels exactly the set variables X_b.

Now for the other direction consider an x-decomposition $xf_0(X_0)+\bar{x}f_1(X_1)+f_2(X_2)$ (which is not necessarily disconnected to start off with) such that $X_b = \{y \mid x \text{ cancels } y \text{ with } b \text{ in } f\}$. Using Lemma 4.49 it follows that $X_0 \cap X_1 = \emptyset$. We know that $x \notin X_2$ by definition of the x-decomposition. If some $y \in X_0 \cap X_2$ then x cannot cancel y since $y \in X_1$. This shows that $X_0 \cap X_2 = \emptyset$. Similar argument also shows that $X_1 \cap X_2 = \emptyset$. Hence the decomposition is disconnected. □

Proof of Theorem 4.48.

(1) $\implies$ (2)
We construct the game by induction on the number of variables. For a trivial polynomial c the game is a trivial game with a leaf node with payoff c.

Now consider a perfect recall polynomial with multiple variables. Consider the x-decomposition $xf_0(X_0) + \bar{x}f_1(X_1) + f_2(X_2)$ which witnesses the perfect recall. Each X_i has fewer variables since x is not present. By induction, there are perfect recall games G_0, G_1, G_2 whose payoffs are given by f_0, f_1, f_2 respectively. Construct game G with the root being a Chance node with two transitions each with probability $\frac{1}{2}$. To the right child attach the game G_2. The left child is a control node with left child being game G_0 and the right child being G_1. This node corresponds to variable x, with the left action corresponding to x and right to $\bar{x}$. Finally multiply all payoffs at the leaves with 2. The payoff of this game is given by $xf_0(X_0) + \bar{x}f_1(X_1) + f_2(X_2)$. Since the decomposition is disconnected, the constructed game also has perfect recall. In fact this construction gives us a perfect information game.

(2) $\implies$ (3)
G be a one-player game with perfect recall whose payoff polynomial is $f(X)$. Consider two variables x and y which appear together in f. W.L.O.G there must exist two nodes u and v in G such that $u \in \mathrm{PathTo}(v)$ and $\mathrm{Obs}(u) = o_x$ and $\mathrm{Obs}(v) = o_y$. But since G has perfect recall, for any w with $\mathrm{Obs}(w) = o_x$, the left subtree and right subtree has no observation in common. Hence all nodes with observation o_y is completely in some sub-tree. Hence in f, if o_y is in sub-tree x then x appears with all y terms and x cancels y with 0 in f. For the other case $\bar{x}$ appears with all y terms and x cancels y with 1 in f. Hence the edge (x, y) exists in Γ_f with the value it cancels y.

(3) $\implies$ (1)
We prove by induction on the number of variables. The statement is true for trivial polynomials. Suppose this statement be true for all polynomials over a set of k variables. Now let f be a polynomial over $k + 1$ variables.

Consider the SCC-decomposition of Γ_f and let x be a vertex in the source component of the SCC decomposition. We will give a disconnected x-decomposition $xf_0(X_0) + \bar{x}f_1(X_1) + f_2(X_2)$ of f such that each of $\Gamma_{f_0}, \Gamma_{f_1}$ and Γ_{f_2} satisfy the condition. Then from induction hypothesis it will follow that each f_i is a perfect recall polynomial and consequently this will prove that f is a perfect recall polynomial.

Now for $b \in \{0, 1\}$, let $X_b = \{y \mid x \text{ cancels } y \text{ with } b \text{ in } f\}$ and let X_2 be the set of all variables x doesn't cancel. X_2 is disjoint from $X_1 \cup X_2$. Also it follows from Lemma 4.49 that $X_0 \cap X_1 = \emptyset$. Hence X_0, X_1 and X_2 are all mutually disjoint. Also $X_0 \cup X_1 \cup X_2 = X \setminus \{x\}$.

Let $xf_0 + \bar{x}f_1 + f_2$ be the canonical x-decomposition of f with respect to x as described in Section 4.6.1. For some $y \in X_1$, since x cancels y with 1, $\bar{x}$ must be a factor of sum of all the terms containing y. Hence it follows from the construction of f_1 that $y \in var(f_2)$. Consequently we have $X_1 \subseteq var(f_2)$ and $var(f_0) \cap X_1 = \emptyset$. Also for any $z \in X_0$, x is a factor of sum of all the terms containing z. Hence $z \notin var(f_1)$ and $z \in var(f_0)$. So $X_0 \subseteq var(f_0)$ and $var(f_1) \cap X_0 = \emptyset$.

Now for some $w \in X_2$, suppose $w \in var(f_0)$. This means a term has the factor xw and from hypothesis, one of them should cancel the other. x doesn't cancel w then w must cancel x. But this is a contradiction since x is from the source component of SCC[6]. Hence $X_2 \cap var(f_0) = \emptyset$. By similar argument for f_1, we have $var(f_1) \cap X_2 = \emptyset$. Hence $X_2 \subseteq var(f_2)$. But since X_0, X_1 and X_2 forms a partition of $X \setminus \{x\}$ we have $var(f_i) = X_i$. From Lemma 4.50 it follows that $xf_0 + \bar{x}f_1 + f_2$ is a disconnected x-decomposition of f.

Now for two variables $y, z \in X_i$, if yz is present in some term, y cancels z with b in f iff y cancels z with b in f_i. So Γ_{f_i} is just the graph Γ_f restricted to variables in X_i. Hence if yz is present in some term f_i, there is an edge between y and z in Γ_{f_i}. From our induction hypothesis, this implies that each of f_i is a perfect recall polynomial. Hence $xf_0 + \bar{x}f_1 + f_2$ is a disconnected x-decomposition of f where each of f_i are perfect recall polynomials. Therefore it follows from definition of perfect recall polynomial that f is a perfect recall polynomial. □

Example 4.18. The PCG Γ_g of the polynomial $g = x_a x_b + x_a x_c - 2x_a x_b x_c$ is given in Fig. 4.9c. x_b and x_c is present in a term but there is no edge between them in Γ_g. Hence g is not a perfect recall polynomial and consequently cannot have a corresponding perfect recall game. An imperfect recall game for g is given in Fig. 4.9d.

[6] In fact since cancellation is a transitive relation between variables, it can be verified that each SCC component is a directed complete graph.

The PCG Γ_f of the polynomial $f = 2x_ax_bx_cx_d + x_ax_bx_e - x_ax_bx_cx_e$ is given in Fig. 4.9a. For every pair of variables present in some term in f, there is an edge between them. Hence f is a perfect recall polynomial and a corresponding perfect recall game (in fact perfect information) is given in Fig. 4.9b. x_a lies in the source component of the SCC-decomposition of Γ_f, hence the x_a decomposition, $x_a(2x_bx_cx_d + x_bx_e - x_bx_cx_e)$ is disconnected. On the other hand, for x_c which is not in the source component, the x_c decomposition, $x_c(2x_ax_bx_d) + \bar{x}_c(x_ax_bx_e)$ is not disconnected.

Next we will use Theorem 4.48 to give an algorithm to detect perfect recall polynomials and construct a corresponding perfect recall game.

Theorem 4.51.

1. *Given a polynomial f in its expanded form, it can be decided in time polynomial in the size of f whether f is a perfect recall polynomial.*

2. *Given a perfect recall polynomial f, a perfect recall one-player game G with payoff f can be constructed in time polynomial in the size of f.*

Proof. (1)
Given a polynomial f, the PCG Γ_f of f can be constructed in PTIME. Using Theorem 4.48, we check for every pair of variables x and y, if whenever xy is present in the expanded form of f, there is an edge between them in Γ_f. If we encounter some x, y for which this is not true, then f is not a perfect recall polynomial. Otherwise we can declare that f is a perfect recall polynomial. This whole process can be done in PTIME.

(2)
Given a perfect recall polynomial, we will construct the game recursively using the schema in the proof (1) $\implies$ (2) of Theorem 4.48. For this we need to find an x such that a disconnected x-decomposition of f can be found bearing witness to perfect recall of f. Again from the part (3) $\implies$ (2) of Theorem 4.48, we see that for any x in the source component of the SCC-decomposition of Γ_f the canonical x-decomposition of f is such an x-decomposition of f. Also we need to compute PCG only once, since PCG of the polynomial restricted to variables $Y \subseteq X$, is exactly Γ_f restricted to variables Y. Hence this algorithm runs in PTIME. □

Remark 4.52. There is one major drawback of simplifying games using our methods via polynomials as opposed to simplification via sequences. Starting from a game without perfect recall, this method cannot always be used efficiently to compute an equivalent perfect recall game. This is because the initial polynomial need not be in its expanded form and expanding it might produce exponentially many terms in the size of the input game in the process of simplification.

4.6.2 Turning any game into games with A-loss recall

Unlike perfect recall, any multi-linear polynomial can come from a game with A-loss recall. This is not surprising since Theorem 4.29 tells that any sequence set has an A-loss recall span.

Theorem 4.53. *For every polynomial $f(x)$, there is a game G with A-loss recall such that f is the payoff polynomial of G.*

Proof. Consider the rulebook of S of a f-game. It follows from Theorem 4.29 that there is rulebook $\hat{S}$ such that $\hat{S}$ has A-loss recall and $S \trianglelefteq \hat{S}$. Hence the statement follows. □

Here naturally the question arises, what is the size of the smallest A-loss recall game with payoff polynomial f. We leave this question as future research work.

4.7 Discussion

We will see an application of perfect recall polynomials in multi-linear optimization.

4.7.1 Applications in multi-linear optimization

Definition 41 (Multi-linear optimization decision problem)**.** The multi-linear optimization decision problem is given a multi-linear polynomial f over variable set $X = \{x_1, \ldots, x_n\}$ and $\lambda \in \mathbb{R}$, is there a valuation v over $[0, 1]^n$ such that $f(v) \geq \lambda$?

Theorem 4.54. *[KM92] The optimum of a multi-linear polynomial over H_n is attained at a vertex and the multi-linear optimization decision problem is* NP*-complete.*

Theorem 4.55. *The multi-linear optimization problem can be decided in* P *for perfect recall polynomials.*

Proof. Given a polynomial f, following Theorem 4.48 a perfect information game can be constructed in PTIME which can be solved efficiently. □

Remark 4.56. For polynomials that are not perfect recall, the problem can be solved in polynomial time in the size of the smallest A-loss recall game. We don't address this question in this thesis.

4.8 Conclusion

We show some nuances in the complexity of the maxmin problem for the class of games where Max has perfect recall and Min is non-absentminded. We show that every such game can be reduced to a game where Max has perfect recall and Min has A-loss recall. The size of the perfect recall component i.e. number of Max nodes remains the same whereas the size of the A-loss Min component, which is essentially the size of the A-loss Recall Span depends on the Min component in the original game. We identify new classes of games for which the size of the A-loss Recall span is the same as that of the original Min component, which we call A-loss shuffle. Since we know perfect vs A-loss games can be solved efficiently, we use this to provide an algorithm for solving perfect recall vs A-loss shuffle in PTIME.

For the general perfect recall vs non-absentminded case, this method gives us an algorithm to solve the maxmin problem in the size of the smallest A-loss recall span of the Min component. We take up the problem of finding the smallest A-loss recall span of a sequence set and give an exact algorithm to compute it. We show that the related decision problem is NP. This raises the following open question.

Open Question 4.8.1

Is the decision problem A-Loss Span NP-hard ?

Moving on to payoff polynomials, we gave a characterization of games whose payoff polynomial comes from games with perfect recall. Although it is still not quite clear if it is possible to use this method for solving games efficiently, this provides a better understanding of multi-linear polynomials arising from games with perfect recall.

In this regard, an interesting question to ask would be: Given a multi-linear polynomial, what is the size of the smallest A-loss recall game with this polynomial as its payoff polynomial?

Chapter 5

Bridge bidding game

In this chapter, we present some preliminary results of our study of the Bidding phase of the popular card game, Bridge. We start by quickly recalling the rules of Bridge and then introduce a suitable model for studying bridge bidding which is based on the *Double Dummy evaluation* of Bridge hands. With the goal of understanding maxmin computation in bridge bidding, we look into a special version of Bridge: the two-player zero-sum version where cards can have only a single *suit* (instead of the usual 4). To discuss various aspects of the model, we keep switching between two representations: the extensive-form representation and the much more succinct *Par Table* representation.

We study the maxmin problem with respect to various restrictions on strategies such as (i) pure strategies and (ii) bounded *overbidding* strategies : strategies where players cannot bid more than a fixed number of times. We demonstrate with examples that these restricted classes of strategies are not optimal in general for achieving the maxmin value. We also show that the complexity of computing the maxmin value depends on the number of *overbids* necessary for maxmin optimality. To complement this we give a bound on the number of bids necessary for maxmin optimality with respect to the *beliefs* of players.

Contents

Bridge is a card game with imperfect information which is played in two phases: the bidding phase and the trick taking phase. It is played between 2 teams each with 2 players traditionally named $\{N, S\}$ and $\{E, W\}$. Ideally rational players can remember their actions and observe the action of other players throughout the game. Hence we assume that each player has perfect recall. Most modern day Bridge programs use the Double Dummy evaluation of bridge *hands*. The double dummy evaluation of a hand evaluates the maximum number of tricks a player can take, when the game is played assuming all players have perfect information. The trick taking phase has already been studied also from a complexity theoretic point of view [BJS13] [BS16]. However to the best of our knowledge no game theoretical study of the Bidding phase of Bridge has been conducted. Our primary goal is to study the Bidding phase with a simple model.

5.1 A crash course on Bridge

Bridge, commonly known as Contract Bridge is a trick taking card game played between two teams with two players in each team [WBF]. Conventionally, the players are called N, S, E, W where N and S play in one team, and E, W play another. Typically the game is played in several rounds with each round of the game consisting of two phases: *the bidding phase* and *the trick taking phase*. At the beginning of each round each player is privately dealt a hand of 13 cards from a deck of 52 cards. Each card has two features: a suit from $\{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$ and a rank from the ordered set $\{2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A\}$.

In the bidding phase, based on their private hands the players bid in turns in order to win a suitable *contract* for the team. A bid consists of choosing a number from $\{1, 2, 3, 4, 5, 6, 7\}$ and a suit (or No Trump) from $\{\clubsuit, \diamondsuit, \heartsuit, \spadesuit, NT\}$ in the order as shown in Fig. 5.1 with $1\clubsuit$ being the smallest. In each turn a player can either overbid the last bid or *pass*. The bidding terminates when one but every other player passes. The player who made the last bid before the terminal sequence

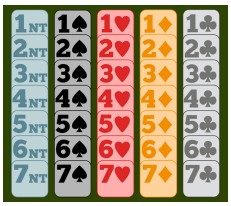


Figure 5.1: Bridge Bidding Order (Source : Funbridge)

of passes wins the contract for her team. The contract of the round is equal to the number in her bid. The player from this team who initiated bidding with the suit of the contract is called the *declarer* and the other player is called *dummy*. After the bidding is done, the dummy player reveals her hand to everyone and the declarer plays both her and her partner's turn in the next phase.[1] In the trick taking phase the goal of each team is to win as many *tricks* possible among the 13 possible tricks with suit of the contract as the trump. At the end of the trick taking phase each team is awarded points based on the number of tricks they have won. The declarer's team need to make at least 5 tricks more than her contract in order to gain positive points. If they fail, they are penalized with negative points. In essence the declaring team plays to gain positive points and the defending team plays to prevent that.

5.1.1 Why study Bridge Bidding?

Developing intelligent programs that can play Bridge has been a challenge in the AI community for a long time starting from Ginsberg's GIB [Gin99][Gin01] to JackBridge[Jac], WBridge5 [WBr] along with it's recent improved version [VCTT17]. In World Computer Bridge Championship [WCB], Bridge programs developed by different teams have been competing against each other annually since 1996. It has been observed that, the bots do quite well in the trick taking phase of Bridge. On the other hand, the main challenge seems to arise from the bidding phase. Ginsberg himself has remarked on his program GIB '...the weakest part of GIB's game is bidding' [Gin01]. Research has already been carried out to improve implementations of the bidding phase most notably using learning algorithms [AM06][YHL18][TZWW18][RQA19][GJT19]. However, understanding the theoretical complexity of bridge bidding as a separate imperfect information game in itself is yet to be explored. This is our primary motivation to dissect the Bridge game and study only the Bidding phase using a suitable model.

[1] In Bridge, dummy refers to placing hands face-up, i.e. revealing one's complete hand.

5.2 Bridge Bidding Model

We introduce our model to study a restricted version of the bidding phase based on the Double Dummy Evaluation of all initial Bridge hands. Before defining our model we provide a quick review of Double Dummy Analysis.

5.2.1 Double Dummy Analysis of Bridge Hands

Double Dummy Bridge is a variant of Bridge played with perfect information. This game follows the usual Bridge rules except that every player observes each others' cards. Since this is a perfect information game, for a given dealing of hands there is an optimal strategy for the team and this strategy can be computed by backward induction following Zermelo's ideas [Zer13]. The double dummy evaluation of a bridge hand is the optimal number of tricks possible to win starting from a player with a given trump suit. Double Dummy Analysis (DDA) is used heavily for evaluating how a bridge bot performs in the playing phase. Since the state space of Bridge is huge, backward induction on the game tree is practically infeasible. More efficient algorithms using better search techniques called *partition search* have been devised in [Gin01] with later improvements in [Bel17]. An implementation can be found here [Han].

The DDA evaluation of a hand is often called the *par score* or *par contract* of the hand. A par table of Bridge is a table consisting of all possible Bridge hands along with their par score. Ideally if a Bridge bot had access to the whole par table at once, it could evaluate a bridge hand before the bidding phase and bid accordingly. In other words, given the par table a Bridge bot could devise an optimal strategy to bid for a given hand. This is the cornerstone of our Bridge Bidding Model. We devise our model assuming we have access to a par table and we wish to compute optimal strategies for bidding in this model.

Now we are ready to define our Bridge Bidding model.

5.2.2 General Bridge Bidding Model

In our initial study of bridge bidding we will consider a simpler version of Bridge: the two player zero-sum version with a single suit. However we will define a generalized Bridge Bidding model here and later see how our simpler version is a special case of this model.

A Bridge Bidding Game abbreviated as BBG is defined as follows:

Definition 42. (Bridge Bidding Game)
A Bridge Bidding Game $BBG_m(r, t, \{H_i\}, \delta, \Theta)$ is a team zero-sum game played between two teams T_1 and T_2 each containing m players.

- m is the number of players in each team. For $m = 2$, $T_1 = \{N, S\}$, $T_2 = \{E, W\}$ and for $m = 1$, $T_1 = \{N\}, T_2 = \{E\}$.
- t is the number of trump suits and r is rank of the highest bid. The set of all possible bids is $[r] \times [t]$ ordered by the usual dictionary order: $(r_i, t_i) < (r_j, t_j)$ iff $r_i < r_j$ or ($r_i = r_j$ and $t_i < t_j$). There is also a special bid called pass which we denote by 0.
- H_i is the set of all possible secrets (hands) player i can receive. $H = H_N \times H_E \times H_S \times H_W$ is the set of all deals of hands and $(h_N, h_E, h_S, h_W) \in H$ is a deal of hands to each player where player i gets secret h_i.
- $\delta \in \Delta(\prod_i H_i)$ is a distribution over the set of all possible deals
- For each $i \in T_1 \cup T_2$, $\Theta_i : H \times [t] \mapsto [r] \cup \{0\}$ is the par function of player i. For a player $i \in T_1 \cup T_2$, Θ_i gives the maximum number of tricks i can take with suit $s \in [t]$ as the trump suit in an optimal play under perfect information corresponding to a deal of hands $(h_N, h_E, h_S, h_W) \in H$. The payoff is decided by the par function which we describe next in the game play. Θ is used to denote the collection of the functions Θ_i.

Gameplay and payoff

The game proceeds as follows: Player i receive secret h_i according to the distribution δ. Players bid in turn by choosing a bid from $([r] \times [t]) \cup \{0\}$ starting with N and going in the order $N, E, S, W, N, \ldots$ ($N, E, N, \ldots$ for $m = 1$). A player's bid can either be 0 or a bid strictly greater the last non-zero bid (bids not equal to 0). The play ends in two ways: i) Passed Out (P.O.): All players start with 0 bid and the game ends in one round. ii) There are consecutive 3 passes (1 pass in the case $m = 1$). In this case if (r_{last}, t_{last}) is the last non-zero bid, r_{last} is called the *contract* of this play. If a player from T_i played the bid (r_{last}, t_{last}), then the first player in T_i to play a bid of the form (r', t_{last}) is called the *declarer*. The payoff at the end is decided by the function Θ_D where D is the declarer. If the game ended in P.O. both teams get payoff 0. Otherwise if D is the declarer from team T_D, T_D gets payoff r_{last} iff $r_{last} \leq \Theta_D(h)$ and $-(r_{last} - \Theta_D(h))$ otherwise. Being a zero-sum game the other team gets negative of what T_D gets.

Contract Bridge Bidding in our model

In Contract Bridge, r is 7 and the bidding goes from $(1, \clubsuit)$ to $(7, NT)$ with the order between trump suits being $\clubsuit < \diamondsuit < \heartsuit < \spadesuit < NT$. Hence Bridge Bidding in contract bridge, is exactly the game given by $BBG_2(7, 5, \{H_i\}, \delta, \Theta)$ where $h_i \subset [13] \times [4]$, such that in any deal, $|h_i| = 13$ for all i and $h_i \cap h_j = \emptyset$. δ is

	$u, x\ (\frac{1}{4})$	$u, y\ (\frac{1}{4})$	$v, x\ (\frac{1}{4})$	$v, y\ (\frac{1}{4})$
Θ_N	0	0	0	2
Θ_E	0	1	1	2

Figure 5.2: A Bridge Bidding Game

the uniform distribution over all deals and Θ is the par function given by Double Dummy Analysis of bridge deals.

Remark 5.1. The payoff function in our model is proportional to the number of tricks won or the penalty due to failure in making the contract. In real Bridge, the scoring rule is more involved which may depend on players doubling, redoubling, etc and other conventions which we don't take into account in our simplified model. Our model can be adapted to any scoring rule based on plays by modifying the payoff function studied in the respective setting.

2 player bridge bidding with one suit

In our initial study we will consider the two player version with unique trump suit and no restrictions on h_i. More precisely we will consider the version $BBG_1(r, 1, \{H_i\}, \delta, \Theta)$ with player N and E, $H = H_N \times H_E$ and since $t = 1$, we have $\Theta : H \times \{N, E\} \mapsto [r] \cup \{0\}$. So according to the rules of general game, whenever N starts with a non-zero bid, the game terminates when some player bids 0. The other player becomes the declarer.

A Bridge Bidding Game can be represented completely by par table giving the values of Θ_i and the distribution δ. We often use the term par table instead of Θ.

Example 5.1. Fig. 5.2 demonstrates a Bridge Bidding Game with $H_N = \{u, v\}$, $H_E = \{x, y\}$, $t = 1$ and $r = 2$. δ is the uniform distribution over $H_N \times H_E$. When N receives v and E receives x, then the bid sequence $(1, 0)$ gives payoff -1 to N and 1 to E. On the other hand when N receives v and E receives y, for the bid sequence $(0, 1, 2, 3, 0)$, N receives 1 and E gets -1.

Remark 5.2. There exists a two-player variant of Bridge called Double Dummy Bridge where one player from each team reveals their hands at the beginning of the game. The two player Bridge Bidding Game can be interpreted as the bidding phase of this variant. For the two player version the par values are known to be computable in PTIME [Wäs05][BJS13]. Another difference is, in Double Dummy Bridge the initial dealing distribution is not independent whereas we consider all kinds of distributions.

5.2.3 Computing Maxmin strategy: Implications

Let Θ_{Bridge} be the par table in Contract Bridge.
Let $BBG_{Bridge} = BBG_2(7, 5, \{H_i\}, \delta, \Theta_{Bridge})$ denote the Bridge Bidding Game for Contract Bridge. The maxmin strategy in BBG_{Bridge} would be the optimal strategy for playing the bidding phase in Bridge. In this regard, a study of the maxmin problem on the general Bridge Bidding model would provide insights into the computational difficulty of getting close to such a strategy. In order to have an initial idea, we initiate the study on restrictions of the general bidding model. This might bear two kinds of results. Firstly efficiently computing maxmin strategies in the restricted model would give insights to computing maxmin strategies in the generalized model. And secondly proving lower bounds in the restricted model would demonstrate the hardness of the general computational problem. In our initial study we show some preliminary results with respect to computing the optimal strategy which we discuss in the next section.

5.3 Studying Restrictions

Two-player Bridge Bidding Game with Single Suit

As preliminary work we will restrict our study to a fragment of the BBG model where there are two players N and E and 1 suit. In our usual notation this means the game $BBG_1(r, 1, \{H_i\}, \delta, \Theta)$. Simplifying the notation for 1-suit, we will denote these games by $BBG(r, \{H_i\}, \delta, \Theta)$.

Next we review the representation of strategies in Bridge Bidding. Let G be given by $BBG(r, \{H_i\}, \delta, \Theta)$.

Bid Sequence

A *bid sequence* is a sequence of legal bids played by N and E from the start of the game. A bid sequence that doesn't denote the end of play is called a *non-terminating bid sequence*. Otherwise it is a terminating bid sequence.

Let Seq be the set of all possible non-terminating bid sequences in G. Essentially Seq is the set of all monotonically increasing sequences on $\{0\} \cup [r]$.

Seq_N (Seq_E respectively) be the set of all non-terminating sequences where N (E respectively) has to make the next bid. Equivalently Seq_N contains all non-terminating sequences of even length and Seq_E contains the ones with odd length.

Remark 5.3. We don't refer to bid sequences as histories because here we are just concerned with the bid number and not from which state it was played. For example, in some bidding game let's say after two sequences 12 and 13, the bid

4 is a possible action for N. But in the extensive-form game, 12 and 13 lead to two nodes with distinct observations of N and hence the action 4 just doesn't reflect the observation of the state from which it was played. Strictly speaking the actions 4 from the two are not exactly the same action in these cases.

For a sequence $s \in Seq$, let s_{last} be the last bid in s. Note that for any non-empty sequence $s \in Seq$, $s0$ is a terminating sequence. The only non-terminating bid sequence ending with 0 is $s = s_{last} = 0$ i.e. when N initiate bidding with 0.

Strategy
A behavioral strategy σ of player N is given by $\sigma(H_N \times Seq_N) \to \Delta(\{0\} \cup [r+1])$ such that $\sigma(h_N, s) \in \Delta(\{0, s_{last}+1, \ldots, r+1\})$. A strategy τ of E is defined similarly. As usual, a pure strategy is a behavioral strategy where σ maps only to Dirac distributions.

A pair of secrets (h_N, h_E) and a strategy profile (σ, τ) determines a play which results in a terminating sequence.

Remark 5.4. Notice that in the strategies in bidding, the bids are bounded by $r+1$. This is because even if we allow bids of arbitrary size, no rational player has an incentive to make a bid strictly greater than $r+1$. This is a consequence of the fact that the payoffs are designed to penalize players for bidding beyond the strength of their hands.

5.3.1 Restriction on Strategies

First we will discuss the size of a bidding game in different representations.

Size in par table representation : The complexity of representing Bridge Bidding using par table is the total complexity of representing Θ and δ. Since there are $|H_N||H_E|$ values in Θ, each value bounded by r, the size of representing in this form is $O(|H_N||H_E| \log r) + Size(\delta)$ where $Size(\delta)$ is the sum of complexities of probabilities in δ.

However when expressed in the extensive-form, the size becomes exponential.
Size in extensive-form: The size in the extensive-form is mainly decided by the number of leaf nodes in this representation. Since players observe every action in bidding game. each pair of secrets (h_N, h_E) contributes to as many leaf Nodes as the the number of terminating sequences. The set of all terminating sequence is $\{s0 \mid s \in Seq\}$. Since $r+1$ is the maximum bid possible we restrict all sequences in Seq to contain bids at most $r+1$. A non-terminating sequence is

an increasing sequence over $\{0\} \cup [r+1]$ and hence $|Seq| = O(2^{r+2})$. Also the payoffs lie between $-(r+1)$ and $r+1$. Hence the the size in the extensive-form is $O(|H_N||H_E|2^r \log r) + Size(\delta)$ where $Size(\delta)$ is the size of probabilities in δ.

5.3.2 Computing maxmin value over pure strategies

In this section we consider the maxmin decision problem over pure strategies in the extensive form representation.

Decision Problem 5.1 ($BBGMaxminPure \geq \lambda$)**.** Given a BBG G and $\lambda \in \mathbb{Q}$ is the maxmin value of N over pure strategies at least λ ?

Maxmin computation by enumerating strategies

The maxmin computation problem for games in extensive-form over pure strategies even when players have perfect recall is NP-complete [BML93]. Hence one way to find the maxmin would be to enumerate over all pure strategies of N. A best response of E to a strategy of N can be computed in PTIME in the size of the game tree. Hence the size of the game tree is important here, both for having an estimate of the pure strategies of N and also for best response computation of E.

The size in extensive-form is $K = O(|H_N||H_E|2^r \log r)$. Hence the number of pure strategies of N can be bounded by $O(2^K)$.

However in maxmin computation by enumeration we consider the whole game tree. We ignored the possibility that for maxmin optimality N might not need to play more than a fixed number of times against any strategy of E. Next we will discuss this kind of strategies.

Bounded overbidding

An *overbidding* is a non-zero bid played by a player after a non-empty bid sequence . We seek a bound on the number of times N needs to overbid in a maxmin strategy. We quantify this by defining *overbidding number* of a strategy of N and classify strategies accordingly.

Remark 5.5. Note that in our definition we don't call the very first bid of N as a overbid, because it doesn't outbid a previous bid. On the contrary the very first bid of E if that is a non-zero bid, it counts as an over bid of E since it overbids the last bid of N.

Definition 43 (Overbidding number)**.**
The overbidding number of a strategy σ of N is the maximum number of times N overbids in any play conforming to σ.

The overbidding number of a strategy τ of E is defined similarly.

Definition 44 (k-overbidding strategy)**.** A strategy with overbidding number k is called a k-overbidding strategy.

We use the term *non-overbidding* strategy to denote a 0-overbidding strategy. Note that a non-overbidding strategy of E is one where E always passes.

While representing strategies we interchangeably use $-$ and 0 for denoting a pass move.

Example 5.2. In the Bridge Bidding Game in Fig. 5.2 a non-overbidding strategy of N is $\{a : \{\epsilon : 0, 01 : -, 02 : -, 03 : -\}, b : \{\epsilon : 2, 23 : -\}\}$. On the other hand the strategy $\{a : \{\epsilon : 0, 01 : -, 02 : -, 03 : -\}, b : \{\epsilon : 0, 01 : 2, 0123 : -, 02 : 3, 03 : -\}\}$ is 1-overbidding strategy since for the sequence 02, N plays a non-zero bid and then passes.

Size of extensive-form for k-overbidding strategies
When N overbids k times, we no longer need to deal with the whole extensive-form game tree since a play will always end after k bids of E. Hence we will find the size of this smaller game tree. It is enough to find the number of sequences possible. The sequences will be of length at most $2k$. Hence the number of sequences can be given by $\sum_{i=1}^{2k} \binom{r}{i}$ which is $O(r+1)^{2k}$. As the result the size of the tree is $O(|H_N||H_E|r^{2k})$.

Maxmin computation over pure non-overbidding strategies
Now for computing maxmin by enumeration, we don't need to consider the whole game tree. The number of non-overbidding strategies of N is given by $(r+2)^{|H_N|}$. Fixing such a strategy E computes her best response on a tree of size at most $O(|H_E||H_N|r^2)$

Hence for the class of non-overbidding strategies of N we obtain the following.

Proposition 5.6. In a game $BBG(r, \{H_i\})$ the maxmin over pure non-overbidding strategies can be computed in time $O(|H_N||H_E|r^2(r+2)^{|H_N|})$.

So we have a good incentive to get some hold on the overbidding number of a maxmin strategy. For this we observe how N's strategy evolves with the *belief* of E which we define next.

5.3.3 Belief and maxmin strategies

In maxmin computation as usual we assume that N declares her strategy ex-ante to E. Before the game starts, E can make no informed guess about the secret

received by N. But having complete knowledge about N's strategy and observing N's moves, E can refine her guess about N's private secrets as the play proceeds gradually. To quantify E's guess about N's secrets, we will involve a well known concept in game theory called *beliefs*.

For a bid sequence s, let s_i be the ith bid in s and $s_{\preceq i}$ be the prefix of s up to s_i inclusive. Recall that s_{last} is the last bid in s. Also recall that a bid sequence s is said to be consistent with strategy σ if $\forall$ prefix $s_{\preceq i} \in Seq_N$ of s, $\sigma(s_{\preceq i}) = s_{i+1}$. Now we will define the belief of E after a bid sequence s when E knows that N plays strategy σ.

For a set A, let 2^A denote the power set of A.

Definition 45 (Belief)**.** Given a pure strategy σ of N in Σ_N and a sequence $s \in Seq$ consistent with σ, the belief of E after sequence s under strategy σ of N denoted by $\mathcal{B}_E(\sigma, s)$ is given by function $\mathcal{B}_E : \Sigma_N \times Seq \mapsto 2^{H_N}$ defined as:

$$\mathcal{B}_E(\sigma, s) = \{h_N \in H_N \mid \sigma(h_N, s_{\preceq i}) = s_{i+1} \; \forall \, i \text{ such that } s_{\preceq i} \in Seq_N\}$$

In essence belief $\mathcal{B}_E(\sigma, s)$ of E is the set of private secrets of N which are consistent with the sequence s under the strategy σ of N. In other words it is the set of possible secrets N could have received based on E's observation. It follows from definition that the initial belief $\mathcal{B}_E(\sigma, \epsilon) = H_N$, i.e. intially E considers every secret of N to be a possible secret. E has complete knowledge of σ, hence the belief of E refines gradually as the play progresses.

Remark 5.7. The belief can also be defined as a distribution over the set of possible secrets but since we are concerned only with pure strategies effectively we define it is as the set of all possible secrets that N could have received with a non-zero probability.

Example 5.3. Let BBG be a game with $H_N = \{u, v, x, y\}$, $r = 4$ and any arbitrary H_E, Θ and δ. Consider the strategy $\sigma = \{u : \{\epsilon : 1, (1,2) : 3, (1,b) : - \forall b \in \{3,4,5\}, (1,2,3,4) : 5,\}, \; v : \{\epsilon : 1, (1,2) : 3, (1,b) : - \forall b \in \{3,4,5\}, (1,2,3,4) : 0,\}, \; x : \{\epsilon : 2, (2,b) : - \forall b \in \{3,4,5\}\}, \; y : \{\epsilon : 2, (2,b) : - \forall b \in \{3,4,5\}\}\}$.

For σ, the initial belief of E before the game starts is $\mathcal{B}(\sigma, \epsilon) = \{u, v, x, y\}$. After N's first bid E's belief refines. $\mathcal{B}(\sigma, 1) = \{u, v\}$ and $\mathcal{B}(\sigma, 2) = \{x, y\}$. So if E observes bid 1 she can deduce from σ that N has been dealt either u or v. On the other hand if she observes 2 she knows N has been dealt either x or y. When N plays 1 and E plays 2, N plays 3 for both u and v giving $\mathcal{B}(\sigma, (1,2,3)) = \{u, v\}$. So after bid sequence $(1,2,3)$, E still can't differentiate between the two secrets. Next

when E plays 4, N plays 5 for u and passes for v. Hence $\mathcal{B}(\sigma,(1,2,3,4,5)) = \{u\}$. This indicates that after observing the bid sequence $(1,2,3,4,5)$, E can tell exactly what secret N was dealt in the beginning.

Next we see how beliefs of E can influence N's overbidding in a pure maxmin strategy.

For $s \in S$ and $b \in [r+1]$, let sb be the sequence obtained by playing bid b after s.

Definition 46 (Belief non-overbidding strategy)**.** A strategy σ of N is called belief non-overbidding if $\forall s \in Seq_E$ consistent with σ and $\forall\ b$ with $s_{last} < b \leq r+1$, $\exists h_N \in \mathcal{B}_E(\sigma, s)$ such that $\sigma(h_N, sb) = 0$.

In a belief non-overbidding strategy, after any bid of E, N passes for at least one of the secrets in the belief.

Example 5.4. Consider the strategy discussed in Example 5.3 given by $\sigma = \{u : \{\epsilon : 1, (1,2) : 3, (1,b) : -\ \forall b \in \{3,4,5\}, (1,2,3,4) : 5,\},\ v : \{\epsilon : 1, (1,2) : 3, (1,b) : -\ \forall b \in \{3,4,5\}, (1,2,3,4) : 0,\},\ x : \{\epsilon : 2, (2,b) : -\ \forall b \in \{3,4,5\}\},\ y : \{\epsilon : 2, (2,b) : -\ \forall b \in \{3,4,5\}\}\}$. σ is not belief non-overbidding because for $1 \in Seq_E$, $\mathcal{B}(\sigma, 1) = \{u, v\}$ but when E plays 2, N plays 3 for both u and v.

On the other hand, let σ_1 be a strategy of N by a minor modification of σ where she passes for secret v on seeing E's bid 2. $\sigma_1 = \{u : \{\epsilon : 1, (1,2) : 3, (1,b) : -\ \forall b \in \{3,4,5\}, (1,2,3,4) : 5,\},\ v : \{\epsilon : 1, (1,b) : -\ \forall b \in \{2,3,4,5\}\},\ x : \{\epsilon : 2, (2,b) : -\ \forall b \in \{3,4,5\}\},\ y : \{\epsilon : 2, (2,b) : -\ \forall b \in \{3,4,5\}\}\}$. σ_1 is a belief non-overbidding strategy.

Proposition 5.8 (Belief non-overbidding maxmin strategy)**.** In a bidding game G, N has a maxmin strategy which is belief non-overbidding.

Proof. Let σ be a pure maxmin strategy of N which is not belief non-overbidding. We will iteratively modify σ to eventually end up with a belief non-overbidding strategy. Since σ is not belief non-overbidding then for some $s \in Seq_E$ consistent with σ for some $b > s_{last}$ it holds that $\forall h_N \in \mathcal{B}(\sigma, s), \sigma(h_N, sb) \neq 0$. Let $s = s_0 s_{last}$.

At each iteration we modify σ to a new strategy σ' defined as follows.

$$
\begin{aligned}
\sigma'(h_N, s_{0\preceq i}) &= \sigma(h_N, s_{0\preceq i}) && \forall h_N \in \mathcal{B}(\sigma, s), \forall i < |s_0|, s_{0\preceq i} \in Seq_N \\
\sigma'(h_N, s_0) &= \sigma(h_N, sb) && \forall h_N \in \mathcal{B}(\sigma, s) \\
\sigma'(h_N, s_0 s') &= \sigma(h_N, sbs') && \forall h_N \in \mathcal{B}(\sigma, s), \forall s', sbs' \in Seq_N \\
\sigma'(h_N, s'') &= \sigma(h_N, s'') && \forall h_N \notin \mathcal{B}(\sigma, s)
\end{aligned}
$$

Essentially for every $h_N \in \mathcal{B}(\sigma, s)$, σ' imitates σ till s_0 and then skips one bid of each player(total two) and continues imitating σ onwards. For any other h_N, σ' behaves same as σ. We claim that the best response payoff against σ' is no less than the maxmin value, i.e. the best response payoff against σ.

This is because if τ' is a best response strategy of E against σ' with $\mathbb{E}(\sigma', \tau')$ strictly less than maxmin value, then E can force the same outcome against τ by playing b after sequence s and then imitate τ'. Hence σ' is also a maxmin strategy.

So the violation of the lemma for σ is due to sequence s and bid b. By construction of σ', the sequence s is no longer consistent with σ', and hence it cannot be a witness for the lemma violation. If there is a violating witness in σ', we continue by making the same modification.

To prove termination of these iterations, for a strategy σ consider the value $\sum o_{h_N}(\sigma)$ where $o_{h_N(\sigma)}$ is the number of times N over-bids for secret h_N in σ. This value strictly decreases for strategies obtained in each iteration. Hence eventually we will end up with a maxmin strategy that is belief non-overbidding. □

Remark 5.9. In the proof of Proposition 5.8, we could modify the strategy σ to σ' without losing payoff because after b, N plays non-zero bid. We cannot update a strategy when N passes for some secret after b, because this would change the outcome of the terminating play and hence might affect the payoff. In Example 5.3 if we follow the proof, the strategy σ can be modified for the secrets u and v (keeping unchanged x and y) as follows: $\{u : \{\epsilon : 3, (3,5) : -, (3,4) : 5\}, v : \{\epsilon : 3, (3,b) : - \; \forall b \in \{4,5\}\}$. N wouldn't lose anything with this new strategy since any outcome forced by E against this strategy can also be simulated by E against the old strategy. On the other hand if N had modified it for u and v to : $\{u : \{\epsilon : 5\}, v : \{\epsilon : -\}$, this doesn't entail the same outcome on v since now the last non-zero bid is no longer 3.

Proposition 5.8 throws some light on the required number of overbiddings for a maxmin optimal strategy. It gives a bound on the number of overbiddings necessary.

Corollary 5.10. N has a maxmin strategy with overbidding number $|H_N|$.

Proof. The size of the beliefs of E strictly decreases after every bid of E. Hence N will only need to bid until E's belief is non-empty. Since the initial belief is H_N, the bound follows. □

Corollary 5.11. When N gets the trivial hand i.e. $|H_N| = 1$, N has a non-overbidding maxmin strategy.

The above corollary can also be interpreted as follows: When N reveals both her strategy and her secrets to E, then N has no incentive to overbid in a maxmin strategy. However in general, for maxmin optimality it turns out that non-over bidding strategies are not sufficient.

5.3.4 Non-optimality of non-overbidding strategies

In this section we will demonstrate with an example that non-overbidding strategies are not optimal for obtaining maxmin value over pure strategies.

For proving this, we need some small lemmas. From Corollary 5.11 we know that when $|H_N| = 1$, N has a maxmin strategy which is non-overbidding. We will show that for general games, the maxmin over non-overbidding strategies can be computed by individually computing maxmin for restricted games corresponding to individual secrets of N.

For a game $G = BBG(r, \{H_i\}, \delta, \Theta)$, and $H'_i \subseteq H_i$, let $G_{[H'_N, H'_E]}$ be the game $BBG(r, \{H'_i\}, \delta', \Theta')$ where Θ' is the projection of Θ on H'_i's and δ' is the restriction of δ to H'_i and normalized, i.e. $\delta'(h_N, h_E) = \frac{\delta(h_N, h_E)}{\sum_{h_1 \in H'_N, h_2 \in H'_E} \delta(h_1, h_2)}$

We will first see that the task of computing the best response strategy of E can be broken down based on the first bid of N.

Lemma 5.12. In a bidding game G, let σ be a strategy of N. For some first bid b played by N in σ, let $G_b = G_{[\mathcal{B}(\sigma,b), H_E]}$ and let τ_b be the best response strategy of E to σ restricted to G_b in the game G_b. Then the strategy τ of E given by $\tau(h_E, bs) = \tau_b(h_E, bs)$ is a best response strategy of E to σ in G.

Proof. Any strategy of E in response to σ is a function of the first bid b of N and E's secret h_E. Since on observing the first bid b of N, E can infer the set of secrets $\mathcal{B}(\sigma, b)$, E playing after observing b is equivalent to E playing in the game G_b. Hence a global best response of E can be built from the local best response strategies in the games G_b's as stated. □

Since we can compute the best response by breaking down the game, next we will see how to compute the best response of a trivial game.

Lemma 5.13. Let G be a trivial bidding game, with $H_N = \{h_N\}, H_E = \{h_E\}$ and $\Theta_N(h_N, h_E) = q_N, \Theta_E(h_N, h_E) = q_E$. When N plays the opening bid b, then the payoff to N when E plays her best response strategy, denoted by $BR_E(b)$

is given by:

$$\begin{aligned} BR_E(b) &= -q_E && \text{when } b < q_E \\ &= -(b - q_N) && \text{else when } b > q_N \\ &= b && \text{else when } b \in [q_E, q_N] \wedge q_E = 0 \\ &= b + 1 - q_E && \text{else when } b \in [q_E, q_N] \wedge q_E \neq 0 \end{aligned}$$

Proof. This follows from the definition of the payoff function in a bridge bidding game. Whenever $b < q_E$, E will bid q_E. Otherwise when $b > q_N$, E will pass. Otherwise when $q_E = 0$, E will also pass. The only case left is when, $q_E \leq b \leq q_N$, and $q_E \neq 0$. In this case E will always bid $b + 1$ because passing will give N a payoff of b, but we have $b \geq b + 1 - q_E$ when $q_E > 0$. □

Now we will demonstrate the existence of games where overbidding is necessary for optimal payoff.

Proposition 5.14. There exists a bidding game where N cannot achieve the maxmin value over pure strategies by playing non-overbidding strategies.

Proof. Consider the game given by the par table in Fig. 5.2.

We claim that the maxmin value of this game over pure strategies is $-\frac{1}{4}$ and this value can be achieved only by some overbidding strategy of N. First we will show that, the maxmin value over non-over bidding strategies is $-\frac{1}{2}$. Observe that any best response strategy of E to a non-overbidding strategy is a 1-overbidding strategy of E.

Now consider any non-overbidding strategy σ of N. There can be two possible cases:
Case 1: $\sigma(u, \epsilon) = \sigma(v, \epsilon)$. In this case, for any first bid b played by N, $\mathcal{B}(\sigma, b) = \{u, v\}$. N can play one of the bids from 0, 1 or 2. When N plays 0, E's best response strategy $\{x : \{0 : -\}, y : \{0 : 1\}$ gives a payoff $-\frac{1}{2}$ to N. When N plays 1. E's best response strategy $\{x : \{1 : -\}, y : \{1 : 2\}$ gives payoff $-\frac{3}{4}$ to N. And when N plays 2, E's best response strategy $\{x : \{2 : -\}, y : \{2 : -\}$ gives payoff -1 to N. Hence in this case N cannot gain more than $-\frac{1}{2}$.

Case 2: $\sigma(u, \epsilon) \neq \sigma(v, \epsilon)$. In this case, the belief sets are singletons and hence it follows from Lemma 5.12 that it is enough to compute best response in the trivial sub-games. For this we will use Lemma 5.13. From Lemma 5.13 we deduce that for the sub-game $G_{[\{v\}, H_E]}$, the maxmin value of N is $-\frac{1}{2}$ when N opens with bid 2. Similarly the maxmin value in the game $G_{[\{u\}, H_E]}$ is $-\frac{1}{2}$ when N opens her bid with 0 or 1. Hence in this case N cannot obtain more than $-\frac{1}{2}$.

Now we will give a maxmin strategy giving maxmin value of $-\frac{1}{4}$. Consider the strategy of N, $\sigma = \{u : \{\epsilon : 0, 01 : -, 02 : -, 03 : -\}, v : \{\epsilon : 0, 01 : 2, 0123 : -, 02 : 3, 03 : -\}\}$. This is an overbidding strategy since for secret v, N overbids for histories 01 and 02. Since N opens her bid with 0 for both her secrets, E has no knowledge about N's secret after N's first bid. N only overbids on seeing bid 1 from E. So in the case when E's strategy is not starting with 1 for either of her secrets, N passes. This is same as N playing a non-overbidding strategy with first bid 0 and E's strategy restricted to not playing 1. In this restricted setting E's best response is $\{x : \{0 : -\}, y : \{0 : 2\}$ which gives N a payoff of $-\frac{1}{4}$.

Now over the set of strategies of E where she plays 1, for x, E should play 1 since for (u, x) she gets a big payoff. And for y, she can play 0 since playing 1, the gain she makes at (u, y) is nullified by the loss at (v, y). Hence a best response strategy of E is $\{x : \{0 : 1, 012 : -\}, y : \{0 : -, \}\}$ with payoff $-\frac{1}{4}$ to N.

Hence the maxmin value over pure strategies is $-\frac{1}{4}$ achieved by a non-overbidding strategy. In fact it can be shown that $-\frac{1}{4}$ is the value of the games with pure optimal strategies exists for both N and E.

□

Next we will demonstrate via an example that for maxmin optimality in bidding games, pure strategies are less powerful than behavioral strategies.

5.3.5 Non-optimality of pure strategies

So far we have only considered pure strategies. The next natural question that arises here is: are pure strategies sufficient for maxmin optimality? We will prove by providing arguments based on an example that pure strategies are not sufficient for this purpose.

	Θ	
	h_1 $(\frac{1}{4})$	h_2 $(\frac{3}{4})$
N	1	1
E	0	2

Figure 5.3: Pure strategy not sufficient for maxmin optimality of N

Example 5.5. In the bidding game given in Fig. 5.3, N has two secrets whereas E doesn't have any secret (or trivial secret). N has the same par value which is 1 for both her secrets. On the other hand E has par values less than 1 and more than 1 for the two secrets respectively.

Now in a pure strategy played by N, if N reveals her secret by playing distinct opening bids for two secrets, then E can play the best response corresponding to

each secret. In this case following Lemma 5.12 and Lemma 5.13 N's best strategy is $\{h_1 : \{\epsilon : 1, 12 : -, 13 : -\}, h_2 : \{\epsilon : 2, 23 : -\}\}$ where E's best response is to pass. This gives N a payoff of $-\frac{1}{2}$. On the other hand, when N doesn't reveal her secret, she opens her bid with the same bid. Now in this case when N bids 2, E passes. But when N bids 0 or 1, E follows up with 2. Even though this incurs a loss to E on h_1, since the probability of seeing h_2 is three times that of h_1, for E the gain at h_2 heavily compensates for the loss at h_1. These strategies give N a payoff of -1.

However N can do better with behavioral strategies. When N plays 1 deterministically for both secrets E plays 2 because her gain for h_2 outweighs her loss at h_1. But if N can play 1 with some non-zero probability for both secrets so that with the final distribution E's loss at h_1 gets more weight than gain at h_2, then this can give a better payoff to N. This is exactly the case when N plays the behavioral strategy $\{h_1 : \{\epsilon : 1, 12 : -, 13 : -\}\}, h_2 : \{\epsilon : (\frac{1}{9})1 + (\frac{8}{9})2), 12 : -, 13 : -, 23 : -\}\}$. Here when E sees the bid 1 of N, whatever bid she plays, the final weight on E's payoff at h_2 is $\frac{3}{4}.\frac{1}{9} = \frac{1}{12}$ compared to $\frac{1}{4}$ at h_1. The best E can do as response to 1 is playing 2 or 0 and either of them gives N $\frac{1}{3}$. On the other hand when N plays 2, E knows for sure the secret is h_2 and for her best response strategy of passing gives N $-\frac{2}{3}$. As a result the final payoff to N turns out to be $-\frac{1}{3}$ which is an improvement over $-\frac{1}{2}$ obtained from deterministic strategy. We can conclude that behavioral strategies gives better maxmin value than pure strategies.

In fact it can be further shown that for this game the minmax value over deterministic strategy is $-\frac{1}{4}$ and the value of the game is $-\frac{1}{3}$ which is achieved by a pair of behavioral strategy.

Hence we have the following proposition.

Proposition 5.15. There exist bidding games where the maxmin value over pure strategies is strictly less than maxmin value over behavioral strategies. In fact there exist such games where E has trivial secrets.

Remark 5.16. From Fig. 5.3 we can construct another bidding game with uniform distribution of secrets of N that also requires behavioral strategy for maxmin optimality. This can be done by splitting the distribution of h_2 into three distinct secrets and copying the par values of h_2. In this new game, if N plays as if she cannot distinguish between these new secrets, then she gains nothing more or less from the original game. On the other hand she also gains nothing more by playing differently for these secrets, since the par values are the same.

Code

A preliminary implementation of the bridge bidding model in Python along

with few classic algorithms for computing maxmin value over pure strategies as well as behavioral strategies tailored to our model can be found on https://github.com/paulsse/bridge-bidding-game.

5.4 Conclusion

We initiated the study of Bridge bidding on a simple version of our model and investigated the sufficiency of restricted strategies for maxmin optimality. We showed with examples that none of the restricted strategies are sufficient.

Several questions remain unanswered in our preliminary study which we consider for carrying forward this work.

- What is the minimum number of times N needs to overbid for maxmin optimality ?
- What is the complexity of the maxmin problem in the par-table representation? Using extensive-form, we have an EXPTIME upper bound. But can we do better by cleverly iterating over belief non-overbidding strategies as a consequence of Proposition 5.8?

Hence the following decision problems arise naturally.

Decision Problem 5.2 (Maxmin-Bid$_{\geq}$)**.** Given a game G and $\lambda \in \mathbb{Q}$, is the maxmin value of N at least λ ?

Decision Problem 5.3 (Maxmin$_{\geq}$-k-overBid)**.** Given a game G and $\lambda \in \mathbb{Q}$ is the maxmin value obtained by a k-overbidding strategy at-least λ?

Decision Problem 5.4 (Overbid-Opt)**.** Given a game G and $k > 0$, can N attain the maxmin value by playing a k-overbidding strategy?

In the future we wish to investigate the complexity of these problems both in the extensive-form and par-table representations.

Chapter 6

Conclusion

In this chapter, first we a provide a summary of our contribution in this thesis and then discuss possible directions to proceed in future.

Our Contribution

Our main contribution in this thesis lies in providing a finer picture of the complexity of the maxmin decision problem in imperfect information games in extensive-form. For one player games in general, we showed $\exists\mathbb{R}$-completeness of the maximum value decision problem. The hardness heavily depends on the absentminded property of the player. For two player games when Max is absentminded the lower bound result carries forward automatically. When player Max is absentminded and Min has A-loss recall, we show $\exists\mathbb{R}$-completeness. In the general case where both players can be absentminded we show both $\exists\mathbb{R}$-hardness and $\forall\mathbb{R}$-hardness. We also observe $\exists\forall\mathbb{R}$ upper bound of the maxmin problem.

On the other hand for two player games when players are not absentminded we show Square-Root-Sum-hardness. In fact we show this for the sub-class of games where Max has A-loss recall and Min has perfect recall. We also show co-NP-hardness for this problem even for the case when Max is a trivial player and Min in non-absentminded.

In the proofs of our complexity results we work with the payoff polynomial of a game when players play behavioral strategies. Computing the maxmin value of a game reduces to optimizing the payoff polynomial of a game.

Making further use of payoff polynomials we give a notion of game equivalence by observing that two games with different structure can have the same payoff polynomial. As a result solving one would result in solving the other. Based on this we give a class of games called games with A-loss recall shuffle which has same

payoff polynomial to that of an A-loss recall game of same size. Using the linear program to solve one player games with A-loss recall we provide an algorithm to detect games with A-loss recall shuffle and also compute the optimal value in PTIME. Hence we provide a new class of imperfect recall which strictly extends the class of A-loss recall and which can solved in PTIME. Using this we also show that in games where Max has perfect recall and Min has A-loss recall shuffle, the maxmin problem can be solved in PTIME.

Generalizing this approach with payoff polynomial we in fact show that any non-absentminded game can have an equivalent A-loss recall game but possibly of exponential size. We call this A-loss recall span and present A-loss recall shuffle as a special class of A-loss recall span. We also give an exact algorithm to compute the minimal A-loss recall span but with possibly exponential worst case running time.

Also given a polynomial we give an algorithm to check if there exists a perfect recall game generating this polynomial as payoff. We characterize such polynomials and also construct the perfect recall game. As a small side application we show that the optimization problem over the class of multi-linear perfect recall polynomial is solvable in PTIME which is NP-complete in general.

Finally we present our preliminary work on Bridge Bidding. We motivate the study of Bridge Bidding and introduce a general Bridge Bidding model to study the bidding phase. We show that solving maxmin problem the naive way can take exponential time in our representation. We show that it is possible that N doesn't need to bid many times for maxmin optimality which we capture with overbidding number. We show that when N declares his strategy to E, she has no incentive to play a strategy which doesn't change the belief of E. We use this to give a bound on the overbidding strategies. We also show that pure strategies and non-overbidding strategies are not enough for maxmin optimality.

Future Directions

Complexity

For maxmin decision problem we were not able to fully settle the complexity of the problem in general. Complexity classes corresponding to fragments of $\exists\forall\mathbb{R}$ has been studied [DLNO21] for establishing the complexity of numerical decision problem. We believe that the general maxmin problem is complete for one of these classes. Another popular decision problem in these kinds of games is the existence of Nash Equilibrium in a two player game. It would be worth exploring if our techniques extend to this problem and produce any kind of lower bound. In general this problem has $\exists\forall\mathbb{R}$ upper bound and only NP-hardness is known so far.

Although we have shown that some games can have possibly exponential size A-loss recall span, we don't quite understand what structure in games causes this blow-up. It would be interesting to know the exact origin of the complexity of solving one player games in the view of A-loss recall spans.

Another challenge is to give better algorithms for computing the optimal A-loss recall span and also give the exact complexity of computing the smallest A-loss recall span.

Bridge

We presented a preliminary version of our ongoing work and there are lots of things to explore with respect to our Bridge Bidding model. The first task is to provide the complexity of the maxmin problem for 2 players, 4 players and also other Bridge specific parameters both in extensive form and par-table representation.

The final goal would be to implement our Bridge Bidding model and compute near optimal strategies over it. A possible direction could be using Monte-Carlo methods to work with a subset of hands and then compare the outcomes with existing Bridge Bidding bot. Hopefully these investigations would slowly take us towards bridging the gap between Bridge bots and human Bridge experts.

Bibliography

[ABKM09] Eric Allender et al. “On the Complexity of Numerical Analysis”. In: *SIAM J. Comput.* 38.5 (2009), pp. 1987–2006.

[AM06] Asaf Amit and Shaul Markovitch. “Learning to bid in bridge”. In: *Mach. Learn.* 63.3 (2006), pp. 287–327. DOI: 10.1007/s10994-006-6225-2. URL: https://doi.org/10.1007/s10994-006-6225-2.

[Aum74] Robert J. Aumann. “Subjectivity and correlation in randomized strategies”. In: *Journal of Mathematical Economics* 1.1 (1974), pp. 67–96. ISSN: 0304-4068. DOI: https://doi.org/10.1016/0304-4068(74)90037-8. URL: https://www.sciencedirect.com/science/article/pii/0304406874900378.

[BBC18] BBC. *David McNamara: Referee banned for rock, paper, scissors kick-off to appeal.* Accessed: 2022-09-2. 2018. URL: https://web.archive.org/web/20221215151119/https://www.bbc.com/sport/football/46250404.

[Bel17] P. Beling. “Partition Search Revisited”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 9.01 (Jan. 2017), pp. 76–87. ISSN: 1943-0698. DOI: 10.1109/TCIAIG.2015.2505240.

[Bet21] Paul Bethe. “Advances in computer bridge: techniques for a partial-information, communication-based game”. In: 2021.

[BH22a] Marie Louisa Tølbøll Berthelsen and Kristoffer Arnsfelt Hansen. “On the Computational Complexity of Decision Problems About Multi-Player Nash Equilibria”. In: 66.3 (June 2022), pp. 519–545. ISSN: 1432-4350. DOI: 10.1007/s00224-022-10080-1. URL: https://doi.org/10.1007/s00224-022-10080-1.

[BH22b] Manon Blanc and Kristoffer Arnsfelt Hansen. "Computational Complexity of Multi-player Evolutionarily Stable Strategies". In: *Computer Science Symposium in Russia*. 2022.

[BJS13] Édouard Bonnet, Florian Jamain, and Abdallah Saffidine. "On the Complexity of Trick-Taking Card Games". In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. IJCAI '13. Beijing, China: AAAI Press, 2013, pp. 482–488. ISBN: 9781577356332.

[BM16] Vittorio Bilò and Marios Mavronicolas. "A Catalog of $\exists\mathbb{R}$-Complete Decision Problems About Nash Equilibria in Multi-Player Games". In: *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*. Ed. by Nicolas Ollinger and Heribert Vollmer. Vol. 47. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, 17:1–17:13.

[BM21] Vittorio Bilò and Marios Mavronicolas. "$\exists\mathbb{R}$-complete Decision Problems about (Symmetric) Nash Equilibria in (Symmetric) Multi-player Games". In: *ACM Trans. Economics and Comput.* 9.3 (2021), 14:1–14:25. DOI: 10.1145/3456758. URL: https://doi.org/10.1145/3456758.

[BML93] Jean R. S. Blair, David Mutchler, and Cheng Liu. "Games with Imperfect Information". In: 1993.

[BP17] Karel Horák Branislav Bosanský Jiri Cermak and Michal Pechoucek. "Computing Maxmin Strategies in Extensive-form Zero-sum Games with Imperfect Recall". In: *Proceedings of the Ninth International Conference on Agents and Artificial Intelligence*. ICAART'17. SciTePress, 2017, pp. 63–74.

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[BS15] Noam Brown and Tuomas Sandholm. "Simultaneous Abstraction and Equilibrium Finding in Games". In: *International Joint Conference on Artificial Intelligence*. 2015.

[BS16] Édouard Bonnet and Abdallah Saffidine. "The Importance of Rank in Trick-Taking Card Games". In: 2016.

[BS17] Noam Brown and Tuomas Sandholm. "Libratus: The Superhuman AI for No-Limit Poker". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 5226–5228. DOI: 10.24963/ijcai.2017/772. URL: https://doi.org/10.24963/ijcai.2017/772.

[BS19] Noam Brown and Tuomas Sandholm. "Superhuman AI for multiplayer poker". In: *Science* 365.6456 (2019), pp. 885–890. DOI: 10.1126/science.aay2400. URL: https://www.science.org/doi/abs/10.1126/science.aay2400.

[Can88] John Canny. "Some Algebraic and Geometric Computations in PSPACE". In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. STOC '88. New York, USA: ACM, 1988, pp. 460–467.

[CBHL18] Jiıé Cermák et al. "Approximating maxmin strategies in imperfect recall games using A-loss recall property". In: *Int. J. Approx. Reasoning* 93 (2018), pp. 290–326.

[CBL17] Jiri Cermak, Branislav Bosanský, and Viliam Lisý. "An Algorithm for Constructing and Solving Imperfect Recall Abstractions of Large Extensive-Form Games". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. IJCAI'17. Melbourne, Australia: ijcai.org, 2017, pp. 936–942.

[CD06] Xi Chen and Xiaotie Deng. "Settling the Complexity of Two-Player Nash Equilibrium". In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. 2006, pp. 261–272. DOI: 10.1109/FOCS.2006.69.

[ČLB20] Jiří Čermák, Viliam Lisý, and Branislav Bošanský. "Automated construction of bounded-loss imperfect-recall abstractions in extensive-form games". In: *Artificial Intelligence* 282 (2020), p. 103248. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2020.103248. URL: https://www.sciencedirect.com/science/article/pii/S0004370220300126.

[Dem01] Erik D. Demaine. "Playing Games with Algorithms: Algorithmic Combinatorial Game Theory". In: *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science*. MFCS '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 18–32. ISBN: 3540424962.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. "The Complexity of Computing a Nash Equilibrium". In: *SIAM Journal on Computing* 39.1 (2009), pp. 195–259. DOI: 10.1137/070699652.

[DLNO21] Julian D'Costa et al. "On the Complexity of the Escape Problem for Linear Dynamical Systems over Compact Semialgebraic Sets". In: *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*. Ed. by Filippo Bonchi and Simon J. Puglisi. Vol. 202. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 33:1–33:21. DOI: 10.4230/LIPIcs.MFCS.2021.33. URL: https://doi.org/10.4230/LIPIcs.MFCS.2021.33.

[EY05] Kousha Etessami and Mihalis Yannakakis. "Recursive Markov Decision Processes and Recursive Stochastic Games". In: *Proceedings of the Thirty Second International Conference on Automata, Languages and Programming*. ICALP'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 891–903.

[EY10] Kousha Etessami and Mihalis Yannakakis. "On the Complexity of Nash Equilibria and Other Fixed Points". In: *SIAM J. Comput.* 39.6 (2010), pp. 2531–2597.

[Gin01] Matthew L. Ginsberg. "GIB: Imperfect Information in a Computationally Challenging Game". In: 14.1 (2001). ISSN: 1076-9757.

[Gin99] Matthew L. Ginsberg. "GIB: Steps Toward an Expert-Level Bridge-Playing Program". In: *International Joint Conference on Artificial Intelligence*. 1999.

[GJT19] Qucheng Gong, Yu Jiang, and Yuandong Tian. "Simple is Better: Training an End-to-end Contract Bridge Bidding Agent without Human Knowledge". In: 2019.

[GMVY15] Jugal Garg et al. "ETR-Completeness for Decision Versions of Multi-player (Symmetric) Nash Equilibria". In: *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*. Ed. by Magnús M. Halldórsson et al. Vol. 9134. Lecture Notes in Computer Science. Springer, 2015, pp. 554–566.

[GS07] Andrew Gilpin and Tuomas Sandholm. "Lossless Abstraction of Imperfect Information Games". In: 54.5 (2007). ISSN: 0004-5411. URL: https://doi.org/10.1145/1284320.1284324.

[Gua22] The Guardian. *Artificial intelligence beats eight world champions at bridge*. 2022. URL: https://www.theguardian.com/technology/2022/mar/29/artificial-intelligence-beats-eight-world-champions-at-bridge.

[Han] Bo Hangkund. URL: https://privat.bahnhof.se/wb758135/bridge/index.html.

[Han19] Kristoffer Arnsfelt Hansen. "The Real Computational Complexity of Minmax Value and Equilibrium Refinements in Multi-player Games". In: *Theory Comput. Syst.* 63.7 (2019), pp. 1554–1571. DOI: 10.1007/s00224-018-9887-9. URL: https://doi.org/10.1007/s00224-018-9887-9.

[HMS07] Kristoffer Hansen, Peter Miltersen, and Troels Sørensen. "Finding Equilibria in Games of No Chance". In: July 2007, pp. 274–284. ISBN: 978-3-540-73544-1. DOI: 10.1007/978-3-540-73545-8_28.

[HMS10] Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. "The Computational Complexity of Trembling Hand Perfection and Other Equilibrium Refinements". In: *Proceedings of the Third International Conference on Algorithmic Game Theory*. SAGT'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 198–209.

[IBM02] IBM. "Deep Blue". In: *Artificial Intelligence* 134.1 (2002), pp. 57–83. ISSN: 0004-3702.

[Jac] JackBridge. *JackBridge*. URL: http://www.jackbridge.com/eindex.htm.

[JPT13] Gabriela Jeronimo, Daniel Perrucci, and Elias P. Tsigaridas. "On the Minimum of a Polynomial Function on a Basic Closed Semialgebraic Set and Applications". In: *SIAM J. Optim.* 23.1 (2013), pp. 241–255. DOI: 10.1137/110857751. URL: https://doi.org/10.1137/110857751.

[KK95] Mamoru Kaneko and J Jude Kline. "Behavior strategies, mixed strategies and perfect recall". In: *International Journal of Game Theory* 24 (1995), pp. 127–145.

[Kli02] J Jude Kline. "Minimum memory for equivalence between ex ante optimality and time-consistency". In: *Games and Economic Behavior* 38.2 (2002), pp. 278–305.

[KM92] Daphne Koller and Nimrod Megiddo. "The complexity of two-person zero-sum games in extensive-form". In: *Games and Economic Behavior* 4.4 (1992), pp. 528–552. ISSN: 0899-8256.

[KM96] Daphne Koller and Nimrod Megiddo. "Finding mixed strategies with small supports in extensive-form games". In: *International Journal of Game Theory* 25 (1996), pp. 73–92.

[KS14] Christian Kroer and Tuomas Sandholm. "Extensive-Form Game Abstraction with Bounds". In: New York, NY, USA: Association for Computing Machinery, 2014. ISBN: 9781450325653. DOI: 10.1145/2600057.2602905. URL: https://doi.org/10.1145/2600057.2602905.

[Kuh53] H. W. Kuhn. *Extensive games and the problem of information*. Ed. by H. W. Kuhn and A. W. Tucker. Princeton, NJ: Princeton University Press, 1953. Chap. 2, pp. 193–216.

[MSBL17] Matej Moravčík et al. "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker". In: *Science* 356.6337 (2017), pp. 508–513. DOI: 10.1126/science.aam6960.

[Nas50] John F. Nash. "Equilibrium Points in N-Person Games." In: *Proceedings of the National Academy of Sciences of the United States of America* 36 1 (1950), pp. 48–9.

[Neu28] J. von Neumann. "Zur Theorie der Gesellschaftsspiele". In: *Mathematische Annalen* 100 (1928), pp. 295–320. URL: http://eudml.org/doc/159291.

[NM47] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1947.

[Osb09] Martin J. Osborne. *Introduction to Game Theory: International Edition*. OUP Catalogue 9780195322484. Oxford University Press, 2009. ISBN: ARRAY(0x58a8d690). URL: https://ideas.repec.org/b/oxp/obooks/9780195322484.html.

[PR97] Michele Piccione and Ariel Rubinstein. "On the Interpretation of Decision Problems with Imperfect Recall". In: *Games and Economic Behavior* 20.1 (1997), pp. 3–24. ISSN: 0899-8256. DOI: https://doi.org/10.1006/game.1997.0536. URL: https://www.sciencedirect.com/science/article/pii/S0899825697905364.

[RQA19] Jiang Rong, Tao Qin, and Bo An. "Competitive Bridge Bidding with Deep Neural Networks". In: *Proceedings of the Eighteenth International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS'19. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 16–24.

[Sch10] Marcus Schaefer. "Complexity of Some Geometric and Topological Problems". In: *Graph Drawing*. Ed. by David Eppstein and Emden R. Gansner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 334–344. ISBN: 978-3-642-11805-0.

[Sch13] Marcus Schaefer. "Realizability of Graphs and Linkages". In: *Thirty Essays on Geometric Graph Theory*. New York, NY: Springer New York, 2013, pp. 461–482.

[Sha50] Claude E. Shannon. "XXII. Programming a computer for playing chess". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41.314 (1950), pp. 256–275. DOI: 10.1080/14786445008521796.

[SHMG16] David Silver et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: 529 (2016).

[SHSA18] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144. DOI: 10.1126/science.aar6404. URL: https://www.science.org/doi/abs/10.1126/science.aar6404.

[SL01] Jiefu Shi and Michael L. Littman. "Abstraction Methods for Game Theoretic Poker". In: *Computers and Games*. Ed. by Tony Marsland and Ian Frank. 2001.

[SS17] Marcus Schaefer and Daniel Stefankovic. "Fixed Points, Nash Equilibria, and the Existential Theory of the Reals". In: *Theory Comput. Syst.* 60.2 (2017), pp. 172–193.

[SS22] Marcus Schaefer and Daniel Stefankovic. "Beyond the Existential Theory of the Reals". In: *CoRR* abs/2210.00571 (2022). DOI: 10.48550/arXiv.2210.00571. arXiv: 2210.00571. URL: https://doi.org/10.48550/arXiv.2210.00571.

[Ste96] Bernhard von Stengel. "Efficient Computation of Behavior Strategies". In: *Games and Economic Behavior* 14.2 (1996), pp. 220–246. ISSN: 0899-8256.

[TZWW18] Zheng Tian et al. "Learning Multi-agent Implicit Communication Through Actions: A Case Study in Contract Bridge, a Collaborative Imperfect-Information Game". In: *ArXiv* abs/1810.04444 (2018).

[VCTT17] Veronique Ventos et al. “Boosting a Bridge Artificial Intelligence”. In: *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. 2017, pp. 1280–1287. DOI: 10.1109/ICTAI.2017.00193.

[Wäs05] Johan Wästlund. “Two-Person Symmetric Whist”. In: *The Electronic Journal of Combinatorics* 12.R44 (2005).

[WBF] WBF. *LawsofDuplicateBridge*. URL: http://www.worldbridge.org/wp-content/uploads/2020/02/2017LawsofDuplicateBridge-gender_neutral_1.pdf.

[WBr] WBridge5. *WBridge5*. URL: http://www.wbridge5.com/.

[WCB] WCBC. *World Computer Bridge Championship*. URL: https://bridgebotchampionship.com.

[YHL18] Chih-Kuan Yeh, Cheng-Yu Hsieh, and Hsuan-Tien Lin. “Automatic Bridge Bidding Using Deep Reinforcement Learning”. In: *IEEE Transactions on Games* 10.4 (2018), pp. 365–377. DOI: 10.1109/TG.2018.2866036.

[Zer13] Ernst Zermelo. *Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. InProceedings of the Fifth International Congress of Mathematicians II*. 1913.

[ZJBP07] Martin Zinkevich et al. “Regret Minimization in Games with Incomplete Information”. In: NIPS’07. Vancouver, British Columbia, Canada: Curran Associates Inc., 2007. ISBN: 9781605603520.

Appendix A

Why binary decision games are enough?

In Chapter 4, we discussed simplification of games in extensive form using binary decision games. Here we will show that working with binary decision games doesn't take away any generality with respect to our analysis of games in sequence form. We briefly extend the vocabulary of concepts for binary decision games to general games and then prove our claims.

Extension to general games

Let G be a two-player game in extensive form. Let A_{Max} and A_{Min} be the set of actions of Max and Min in G respectively and $A = A_{\text{Max}} \cup A_{\text{Min}}$ be the set of all actions. Recall that $\mathcal{O}$ is the set of all observations in G. Then $A = \uplus_{o \in \mathcal{O}} \text{Act}(o)$ is a partition of A.

For two actions a_1 and a_2 in A, a_1 and a_2 are called *co-actions* if $\exists o \in \mathcal{O}$ such that $a_1, a_2 \in \text{Act}(o)$. Let $Part(A)$ be the set of all $\text{Act}(o)$, i.e. $Part(A) = \{\text{Act}(o) \mid o \in \mathcal{O}\}$ and we call this the co-action partition of A. In the specific case of binary decision games, each set in $Part(A)$ has size 2. We denote actions from a node with observation o and k actions out it by $\text{Act}(o) = \{a_1^o \dots a_k^o\}$. For any $a_i^o, cl(a_i^o) = \text{Act}(o)$.

$Seq(A) = \{s \in A^* \mid \forall a \in A, a$ appears at most once in s and $a \in s \implies \text{Act}(s)$ has no co-action of $a\}$. We consider sequences only from $Seq(A)$.

The closure of S, $cl(S) = S \cup \{sa' \mid sas' \in S, a'$ is a co-action of $a, sa'A^* \cap S = \emptyset\}$

Recall that two sequences s_1 and s_2 are called connected when $cl(\text{Act}(s_1)) \cap$

$cl(\mathrm{Act}(s_2)) \neq \emptyset$.

We also recall the definition of A-loss recall compatibility of sequences using connectedness. For two sequences s_1 and s_2, $s_1 \sim_A s_2$, when either $(s_1, s_2) = (ss_1', ss_2')$ with s_1' and s_2' being disconnected or $(s_1, s_2) = (sa_1s_1', sa_2s_2')$ where a_1 and a_2 are co-actions. A set S has A-loss recall when every pair of sequences are A-loss recall compatible.

A behavioral strategy is given by attaching to action a_i the variable x_{a_i} that can take values from $[0, 1]$ such that for any $a_i \in A$, $\sum_{a_j \in cl(a_i)} x_{a_j} = 1$. These constraints are the strategy constraints. The payoff polynomial is a polynomial over variables in x_{a_i}.

The definition of leaf monomials remains the same.

For two sequence sets S_1 and S_2, S_1 spans S_2, i.e. $S_1 \trianglelefteq S_2$, if for each $s \in S$, there exists $k^s_{s'} \in \mathbb{R}$ for each $s' \in S_2$, such that $\mu(s) - \sum_{s' \in S_2} k^s_{s'} = 0$ under the strategy constraints. Under this definition of span, A-loss spans are defined as usual.

A game G in extensive form gives a sequence set S. Conversely using the same construction used in Proposition 4.9, given a sequence set S over A and co-action partition $Part(A)$ one can construct a corresponding game.

So for discussing results on games in extensive form we will interchangeably use its sequence form.

For converting a sequence set over general action set to sequence set over binary actions, we will show that we can find an equivalent sequence set which uses at least one action less than it had before. By repeated application of this, we will get to a binary decision game.

Lemma A.1. (Action elimination lemma) Let $A = A_{\mathsf{Max}} \cup A_{\mathsf{Min}}$ along with co-action partition $Part(A)$. For some $a_1 \in A_{\mathsf{Max}}$, let $cl(a_1) = \{a_1, \ldots . a_k\}$ with $k > 2$. Let $A'_{\mathsf{Max}} = A_{\mathsf{Max}} \cup \{\bar{a}_1\}$ and $A' = A'_{\mathsf{Max}} \cup A_{\mathsf{Min}}$ with co-action partition, $Part(A') = (Part(A) \setminus \{cl(a_1)\}) \cup \{\{a_1, \bar{a}_1\}, cl(a_1) \setminus \{a_1\}\}$.

For a sequence set S over A, let $\textsc{Branch}_{a_1}(S)$ be the sequence set over A' constructed from S by replacing every occurrence of a_i for $i \in \{2, \ldots, k\}$ in every sequence in S by $\bar{a}_1 a_i$[1]. Then the following statements are true.

1. S is complete $\implies$ $\textsc{Branch}_{a_1}(S)$ is complete.
2. S_{Max} has perfect recall. $\implies$ $\textsc{Branch}_{a_1}(S_{\mathsf{Max}})$ has perfect recall.
3. S_{Max} has A-loss recall. $\implies$ $\textsc{Branch}_{a_1}(S_{\mathsf{Max}})$ has A-loss recall.

[1]Essentially in the game, this groups the decision of choosing one of the actions $a_2, \ldots, a_k$ into $\bar{a}_1$ and delays their choice in the next observation after choosing $\bar{a}_1$.

4. $S^{\dagger}_{\mathsf{Max}}$ is an A-loss recall shuffle of $S \implies \textsc{Branch}_{a_1}(S^{\dagger}_{\mathsf{Max}})$ is an A-loss recall shuffle of $\textsc{Branch}_{a_1}(S)$.

5. If $S^{\dagger}_{\mathsf{Max}}$ is an A-loss recall span of $S_{\mathsf{Max}} \implies \textsc{Branch}_{a_1}(S^{\dagger}_{\mathsf{Max}})$ is an A-loss recall span of $\textsc{Branch}_{a_1}(S_{\mathsf{Max}})$

Proof. (1) Suppose $\textsc{Branch}_{a_1}(S)$ is not complete. Then for some $sas' \in \textsc{Branch}_{a_1}(S)$, there is some $a' \in cl(a)$ such that $\textsc{Branch}_{a_1}(S)$ has no sequence of form $sa's''$. It must be that $\mathrm{Act}(sas') \cap cl(a_1) \neq \emptyset$, since sequences in $\textsc{Branch}_{a_1}(S) \cap S$ over $A \setminus cl(a_1)$ do not contradict completeness. a cannot be a_1 or $\bar{a}_1$ since for any sa_1s' there is $s\bar{a}_1s''$ and vice versa. Then the only possibility is $a \in \{a_2, \ldots, a_k\}$. But since $k > 2$ for any $s\bar{a}_1a_is'$ in S_b, we have another $s\bar{a}_1a_js'' \in S_b$. Hence $\textsc{Branch}_{a_1}(S)$ is also complete.

(3) We will prove the case for A-loss recall. The proof of (2) is similar. We can that assume $a_1 \in A_{\mathsf{Max}}$. Suppose for $s_1, s_2 \in S'_{\mathsf{Max}}$, $s_1 \not\sim_A s_2$. Then at least one of s_1 and s_2 is of the form $s\bar{a}_1a_is'$ since these are new additions in $\textsc{Branch}_{a_1}(S_{\mathsf{Max}})$. W.L.O.G, let that one be s_1. Now either $s_2 = s_0\bar{a}_1a_j1s''$ or s_2 doesn't contain $\bar{a}_1$. But then, in the latter case $s_2 \not\sim_A s_0a_js''$ which is a contradiction. And in the first case $sa_is' \not\sim s_0a_js''$ which is also a contradiction since both of these are sequences form S_{Max}. Hence $\textsc{Branch}_{a_1}(S_{\mathsf{Max}})$ has to have A-loss recall.

(5) We will prove the case for A-loss recall span and the proof of (4) will follow similarly.

Let x_a and y_a be the variables associated to some action a in A and A' respectively. For A we have the strategy constraint $\sum_{i\in[k]} x_{a_i} = 1$, For A' we have the constraints $y_{a_1} + y_{\bar{a}_1} = 1$ and $\sum_{1<j\leq k} y_j = 1$. Hence we have $y_{a_1} + \sum_{1<j\leq k} y_{\bar{a}_1}y_j = 1$. But since every leaf monomial in $\mu(S'_{\mathsf{Max}})$ containing $y_{\bar{a}_1}$ also contains a y_{a_j} for some $j > 1$, one can obtain leaf monomials of S_{Max} by replacing every occurrence of $y_{\bar{a}_1}y_j$ for $j > 1$ by x_{a_j} and any other y_a by x_a. As a result the set of polynomials generated by $\mu(S_{\mathsf{Max}})$ and $\mu(S'_{\mathsf{Max}})$ are identical under this substitution. Hence $\textsc{Branch}_{a_1}(S^{\dagger}_{\mathsf{Max}})$ also spans $\textsc{Branch}_{a_1}(S_{\mathsf{Max}})$. □

Now we convert any game into an equivalent binary decision game, preserving the same conditions we saw in Lemma A.1.

Proposition A.2 (Binary decision game)**.** Let $G = (S, \Lambda)$ be a game in sequence form with non-absentminded players. Then there exists a binary decision game $G' = (S', \Lambda')$ with $G \sim G'$ and $|G| = |G'|$ that can be constructed in PTIME such that

1. Max (Min) has A-loss (perfect) recall in $G \implies$ Max (Min) has A-loss (perfect) recall in G'.

2. Max (Min) has A-loss recall shuffle in $G \implies$ Max has A-loss recall shuffle in G'.

3. If S_{Max} has an A-loss recall span, then S'_{Max} has an A-loss recall span of the same size.

Proof. First we will construct the game G' and then prove the three properties. If $G = (S, \Lambda)$ is already a binary decision game, G' is just G. Otherwise, we run an iterative process generating a new game at each step until we get a binary decision. If A is the action set of S, there is set in $Part(A)$ with size more than 2. Let a be an action in this set. Then our new A is $A \cup \{\bar{a}\}$ and new S is $\textsc{Branch}_a(S)$. And the new Λ doesn't change, since we don't add or remove any sequence. We do this until all sets in $Part(A)$ has size 2. Since the value $\sum_{B|\ B \in Part(A), |B|>2} |B|$ strictly decreases at each iteration, this process is bound to stop and we end up with a binary decision game $G' = (S', \Lambda')$. Now since Λ' has the same payoffs as Λ it follows that $|G| = |G'|$. To see that G and G' are equivalent, we will show this after every iteration. Let for $a_1 \in A$ with $cl(a_1 = \{a_1, \ldots, a_k\}$, G' is formed by taking $S' = \textsc{Branch}_{a_1}$. Then to a behavioral strategy of player i, β_i we associate the strategy β'_i where$\beta'_i(a_1) = \beta(a_1)$, $\beta'_i(\bar{a}_1) = 1 - \beta(a_1)$ and $\forall a_i \in cl(a), i > 1, b'_i(a_i) = (1 - \beta(a_1))\beta(a_i)$. Since the new strategies induce the same probability distribution on sequences, this shows that $G \sim G'$.

Now we will prove the three properties.

(1) This follows by applying inductively Lemma A.1 at every step of iteration.

(2) Again, this follows by applying inductively Lemma A.1.

(3) This also follows from repeated application of Lemma A.1.

□

Since reducing to binary decision games doesn't affect the size of an optimal A-loss recall span, we lose nothing by working with binary decision games. A strategy in the binary decision games can be mapped back to the corresponding strategy in the original game.