

Measuring and Influencing Sequential Joint Agent Behaviours

Peter Abraham Raffensperger

A thesis presented for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

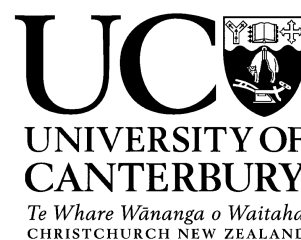
26 February 2013

Abstract

Algorithmically designed reward functions can influence groups of learning agents toward measurable desired sequential joint behaviours. Influencing learning agents toward desirable behaviours is non-trivial due to the difficulties of assigning credit for global success to the deserving agents and of inducing coordination. Quantifying joint behaviours lets us identify global success by ranking some behaviours as more desirable than others. We propose a real-valued metric for turn-taking, demonstrating how to measure one sequential joint behaviour. We describe how to identify the presence of turn-taking in simulation results and we calculate the quantity of turn-taking that could be observed between independent random agents. We demonstrate our turn-taking metric by reinterpreting previous work on turn-taking in emergent communication and by analysing a recorded human conversation. Given a metric, we can explore the space of reward functions and identify those reward functions that result in global success in groups of learning agents. We describe ‘medium access games’ as a model for human and machine communication and we present simulation results for an extensive range of reward functions for pairs of Q-learning agents. We use the Nash equilibria of medium access games to develop predictors for determining which reward functions result in turn-taking. Having demonstrated the predictive power of Nash equilibria for turn-taking in medium access games, we focus on synthesis of reward functions for stochastic games that result in arbitrary desirable Nash equilibria. Our method constructs a reward function such that a particular joint behaviour is the unique Nash equilibrium of a stochastic game, provided that such a reward function exists. This method builds on techniques for designing rewards for Markov decision processes and for normal form games. We explain our reward design methods in detail and formally prove that they are correct.

Keywords: multi-agent systems, multi-agent reinforcement learning, decentralised systems, resource allocation, turn-taking, Nash equilibria, emergent behaviour, reward functions, reward design, Markov decision processes, Markov chains.

Deputy Vice-Chancellor's Office
Postgraduate Office



Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from which the extract comes:

Chapter 3 was adapted from a previously published journal paper: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012b). A simple metric for turn-taking in emergent communication. Adaptive Behavior, 20(2):104–116

Please detail the nature and extent (%) of contribution by the PhD candidate:

The PhD candidate did the research and wrote all sections of the published manuscripts. The co-authoring supervisory team suggested research directions and offered criticisms which the candidate usually worked into the manuscripts.

Certification by co-authors: If there is more than one co-author then a single co-author can sign on behalf of them all. The undersigned certifies that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work.
- In cases where the PhD candidate was the lead author of the co-authored work, that he or she wrote the text.

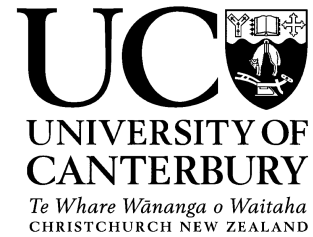
Name: Philip Bones

Signature:

A handwritten signature in black ink, appearing to read 'P. Bones', written over a light blue horizontal line.

Date: 16 November 2012

Deputy Vice-Chancellor's Office
Postgraduate Office



Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from which the extract comes:

Chapter 4 was adapted from a previously published journal paper: Raffensperger, P. A., Bones, P. J., McInnes, A. I., and Webb, R. Y. (2012a). Rewards for pairs of Q-learning agents conducive to turn-taking in medium-access games. Adaptive Behavior, 20(4):304–318

Please detail the nature and extent (%) of contribution by the PhD candidate:

The PhD candidate did the research and wrote all sections of the published manuscripts. The co-authoring supervisory team suggested research directions and offered criticisms which the candidate usually worked into the manuscripts.

Certification by co-authors: If there is more than one co-author then a single co-author can sign on behalf of them all. The undersigned certifies that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work.
- In cases where the PhD candidate was the lead author of the co-authored work, that he or she wrote the text.

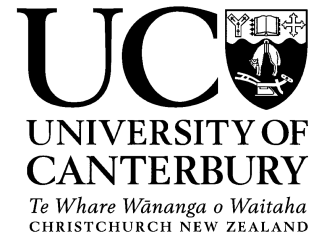
Name: Philip Bones

Signature:

A handwritten signature in black ink, appearing to read 'P. Bones', written over a light blue horizontal line.

Date: 16 November 2012

Deputy Vice-Chancellor's Office
Postgraduate Office



Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from which the extract comes:

Chapter 5 has been adapted into a journal paper that we submitted for publication in the journal of Autonomous Agents and Multi-Agent Systems.


Please detail the nature and extent (%) of contribution by the PhD candidate:

The PhD candidate did the research and wrote all sections of the published manuscripts. The co-authoring supervisory team suggested research directions and offered criticisms which the candidate usually worked into the manuscripts.

Certification by co-authors: If there is more than one co-author then a single co-author can sign on behalf of them all. The undersigned certifies that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work.
- In cases where the PhD candidate was the lead author of the co-authored work, that he or she wrote the text.

Name: Philip Bones

Signature: 

Date: 16 November 2012

Contents

Abstract	i
Acknowledgements	viii
Nomenclature	ix
1 Introduction	1
1.1 The big picture	2
1.2 The organisation of this dissertation	3
1.2.1 Research contributions	5
1.2.2 Publications	6
1.2.3 What the thesis is not about	6
1.3 Related research areas	7
1.4 Summary	9
2 Background	10
2.1 Markov decision processes & reinforcement learning	10
2.1.1 Markov chains	11
2.1.2 Markov decision processes	13
2.1.3 Reinforcement learning	15
2.1.4 Balancing exploration and exploitation	18
2.2 Multi-agent systems and game theory	19
2.2.1 Multi-agent systems	20
2.2.2 Normal form games and Nash equilibria	20
2.2.3 Stochastic games	23
2.2.4 Multi-agent reinforcement learning	24
2.3 Emergent multi-agent behaviours	26
2.3.1 What is emergence?	27
2.3.2 Emergent communication	28
2.3.3 Emergent turn-taking	28
2.4 Summary	29

3	A Simple Metric for Turn-Taking in Emergent Communication	31
3.1	Introduction	31
3.2	A simple turn-taking metric, $\tau\tau$	34
3.2.1	Resource allocation model	34
3.2.2	$\tau\tau$ is the product of fairness and usage efficiency	35
3.2.3	Choosing a resolution	36
3.3	Measuring the turn-taking of random agents	37
3.4	Application of the metric to work by Di Paolo	40
3.4.1	Defining the resource usage attempt sequences	40
3.4.2	Choosing a resolution	41
3.4.3	Results	43
3.5	Application of the metric to work by Iizuka and Ikegami	44
3.5.1	Choosing resolutions	45
3.5.2	Results	46
3.6	Application of the metric to a human conversation	48
3.6.1	Defining the resource usage attempt sequences	50
3.6.2	Choosing a resolution	51
3.6.3	Results	51
3.7	Open source software implementation	53
3.8	Discussion	53
3.9	Related work	54
3.10	Conclusion	57
4	Rewards for Pairs of Q-Learning Agents Conducive to Turn-Taking in Medium Access Games	58
4.1	Introduction	59
4.2	Simulated medium access games	60
4.2.1	Discrete interference channel model	61
4.2.2	Agent design	62
4.3	Insights gained from stationary agents	64
4.3.1	Turn-taking behaviour in agents with deterministic policies	64
4.3.2	Computing the expected rewards for agents with stationary stochastic policies	66
4.3.3	Computing the Nash equilibria of medium access games	67
4.4	Predictors for turn-taking in medium access	69
4.4.1	Q-learning algorithm set up	69
4.4.2	Classification methods	72
4.4.3	Classifier verification	74
4.4.4	Designing rewards for turn-taking	76
4.5	Discussion	76

4.6	Related work	80
4.7	Conclusion	82
5	A Method for Designing the Rewards of Stochastic Games	84
5.1	Introduction	85
5.2	Designing rewards for Markov decision processes	86
5.2.1	Formalising reward design for Markov decision processes	88
5.2.2	Rewarding an agent for following a successful policy is a design compliant reward function	88
5.3	Designing rewards for normal form games	92
5.3.1	When a deterministic joint policy is desirable	93
5.3.2	Algorithms for generating fully mixed two-agent games	94
5.3.3	Generalising to normal form games with arbitrary numbers of agents	100
5.4	Designing rewards for general stochastic games	107
5.4.1	Definitions	107
5.4.2	Rewarding agents for following a successful joint policy is a design compliant reward function	109
5.4.3	An example	110
5.5	Discussion	116
5.6	Related work	120
5.7	Conclusion	123
6	Conclusion	124
6.1	How it all fits together	124
6.2	Discussion	126
6.3	Future work	127
6.3.1	Opportunities to broaden	128
6.3.2	Opportunities to extend	129
6.3.3	Possible applications	130
6.4	Final summary	130
	References	132
A	Graph Generator Algorithms	146

Acknowledgements

Thanks to Hiroyuki Iizuka (Osaka University) and Takashi Ikegami (University of Tokyo) for data that came from their experiments (Iizuka and Ikegami, 2004) which was useful in developing and validating my turn-taking metric. I am also grateful to the thorough and genuinely helpful researchers who selflessly took the time to critique my work through the anonymous reviews of the journal articles that make up my thesis. Thanks to the University of Canterbury and the NZi3 ICT Innovation Institute for funding my research.

Special thanks to my supervisors, who are truly research ninjas of the highest quality. To Phil Bones: thank you for supervising my crazy project and for your ruthless demand for lexicographically precise discrimination between apparent synonyms. To Allan McInnes: thank you for your infectious enthusiasm when my own waned thin and for your uncanny ability to wriggle out of the tightest paper-review binds. To Russ Webb: thank you for your constant stream of ideas and for letting me tap into your $O(1)$ ability to see ideas from oblique and useful angles.

Thanks to my fellow post-grads and fellow spelunkers of the dark, occulted caves of the unknown. Thanks to my officemates, especially Alex Opie (don't worry, I'll never tell) and Firas Al-Hasani. Special thanks to A. Steven Banerjee (with the reality distortion field) and to the many afternoon coffee club participants and celebrities.

Thank you to my family and friends, who have helped me in many ways. Special thanks to my parents, for moral support, academic savoir-faire and all the Sunday lunches. To my dearest Lauren: thank you for marrying me! I love you! Thanks for proof-reading, for all your practical help and for your long-suffering support while I've been madly writing. You are wonderful and amazing!

Soli Deo gloria: to God alone—the first multi-agent system designer—be the glory!

Nomenclature

‘Agent’ An active goal-directed entity. While an agent may be a person or an animal, we refer to agents with the pronoun ‘it’ because we are mostly concerned with artificial agents.

‘Multi-agent system’ A set of related agents where each agent’s actions may affect the other agents.

\mathcal{N} The set of agents.

n A symbol to represent an agent; n is an element of \mathcal{N} , the set of agents.

m An alternative agent symbol; m is an element of \mathcal{N} , the set of agents.

t A discrete time index.

Γ A normal form game: in a single unrepeated instant each agent selects a probability distribution over its actions and receives a reward that depends on the action probability distributions of all agents.

\mathcal{G} A stochastic game: for an indefinite number of time steps, each agent selects an action based on the current state, then the state changes based on the agents’ actions & chance, and each agent receives a reward that depends on the agents’ actions and the state transition.

‘Nash equilibrium’ A joint agent policy such that no agent can increase its expected payoff by unilaterally changing its policy.

\mathcal{A}_n The set of actions of agent n .


a_n One of agent n ’s actions, an element of \mathcal{A}_n .

\mathcal{A} The set of joint actions of a group of agents, $\mathcal{A} = \times_{n \in \mathcal{N}} \mathcal{A}_n$.

\mathbf{a} A joint action, made up of an action from each agent in an understood set of agents, $\mathbf{a} \in \mathcal{A}$.

\mathbf{a}_{-n} A joint action, made up of an action from each agent in an understood set of agents except agent n , $\mathbf{a}_{-n} \in \times_{m \in \mathcal{N}, m \neq n} \mathcal{A}_m$. Note that $\mathbf{a} = (a_n, \mathbf{a}_{-n})$.

- \mathcal{A}_{-n} The set of possible values for \mathbf{a}_{-n} , that is $\times_{m \in \mathcal{N}, m \neq n} \mathcal{A}_m$.
- π_n Agent n 's policy, a time-varying mapping between agent n 's state (if agent n has state) and a probability distribution over agent n 's actions.
- ϵ An agent's exploration amount. In an ϵ -greedy exploration policy, ϵ is the probability that an agent will take a random 'exploratory' action rather than the best known 'greedy' action at any time step.
- \mathcal{S} A set of states.
- T A state transition function.
- γ An agent's discount factor, which is a measure of how much more it prefers a reward at time t to that same reward at the later time of $t + 1$.
- $R_n(t)$ The value of agent n 's reward function at time t . A reward function is a real-valued indication of an agent's performance that the agent can use to improve its policy.
- R A vector-valued joint reward function for a group of agents.
- 'Matching Pennies' A two-agent normal form game where each agent can choose to play either 'Heads' or 'Tails.' One agent earns a high reward when the agents' actions match; the other agent earns a high reward when the agents' actions do not match.
- G A graph $(\mathcal{V}, \mathcal{E})$, made up of vertices, \mathcal{V} , and edges (or, in the case of a directed graph, arcs), \mathcal{E} .
- \mathcal{E} The edges of a graph or the arcs of a directed graph; $\mathcal{E}(\cdot)$ is an operator that returns the edges (arcs) of the (directed) graph in the argument.
- \mathcal{V} The vertices of a graph; $\mathcal{V}(\cdot)$ is an operator that returns the vertices of the graph in the argument.
- 'Hamiltonian cycle' A Hamiltonian cycle is a sequence of vertices and arcs in a graph that starts and ends on the same vertex and passes through each vertex in the graph exactly once. A graph is a 'Hamiltonian graph' if it has a Hamiltonian cycle.
- 'Bipartite graph' A graph where the vertices can be separated into two sets, \mathcal{V}_1 and \mathcal{V}_2 , such that all arcs connect a vertex in \mathcal{V}_1 with a vertex in \mathcal{V}_2 .
- ' k -partite graph' A graph where the vertices can be separated into k sets, $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_k$, such no arc connects one vertex in \mathcal{V}_i with another vertex in \mathcal{V}_i , for all $1 \leq i \leq k$.
- r A turn-taking 'resolution' for the metric $\tau\tau$, discussed in Chapter 3.

- $A_n(t)$ The value of agent n 's 'usage attempt sequence' at time t .
- $\alpha_n(t, r)$ The turn-taking 'allocation' of agent n in the time range t to $t + r - 1$, see Eq. (3.1) on p. 35 in Chapter 3.
- $fairness(t, r)$ The fairness of the agents' turn-taking allocations in the time range t to $t + r - 1$, see Eq. (3.2) on p. 35 in Chapter 3.
- $efficiency(t, r)$ The efficiency of the agents' turn-taking allocations in the time range t to $t + r - 1$, see Eq. (3.3) on p. 36 in Chapter 3.
- $\tau\tau$ A turn-taking metric introduced in Chapter 3.
- $\tau\tau(t, r)$ The quantity of turn-taking present in the time range t to $t + r - 1$, for an understood group of agents' usage attempt sequences, see Eq. (3.5) on p. 36.
- $TT(N, r, P_1, P_2, \dots, P_N)$ A random variable representing the turn-taking at resolution r of N probabilistic agents, where P_n is the probability that agent n uses the shared resource on any time step.
- $TT(N, r)_{best}$ A random variable representing the maximum achievable expected turn-taking at resolution r of N agents, assuming that each agent's resource usage probability is independent of time and the other agents' actions.
- $I_n(t)$ The value of agent n 's input at time t , see Chapter 4, Eq. (4.2) on p. 62.
- $O_n(t)$ The value of agent n 's output at time t , see Chapter 4, Eq. (4.1) on p. 62.
- β An agent's learning rate. A low learning rate means that the agent changes its policy slowly, while a high learning rate means that the agent changes its policy quickly.
- ϕ The confidence level of the post-simulation turn-taking classifier in Chapter 4.
-  Nicolas Bourbaki's 'dangerous bend' symbol; used to indicate that particularly difficult mathematics follows for the remainder of a section.
- $[\cdot]_{ij}$ Matrix element selection operator; $[\mathbf{X}]_{ij}$ is the element in the i th row and the j th column of \mathbf{X} .
- e Euler's number, the base of the natural logarithms, $e \approx 2.718$.
- $E[\cdot]$ The expectation operator; $E[X]$ is the expected value of the random variable X .
- \mathbb{N} The set of natural numbers, $\{1, 2, 3, \dots\}$.
- \mathbb{Z} The set of integers, $\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$.
- \mathbb{R} The set of real numbers.

'We' This dissertation uses the pronoun 'we' to avoid constant use of the passive voice and the less formal 'I.'

Chapter 1

Introduction

The essential thesis of this research is that: **Algorithmically designed reward functions can influence groups of learning agents toward measurable desired sequential joint behaviours.** The thesis is demonstrated with research explaining how to measure a particular sequential joint behaviour, turn-taking, how to identify rewards that are conducive (or prohibitive) to turn-taking by learning agents in a simulated context and how to design rewards that incentivise arbitrary sequential joint behaviours in multi-agent stochastic games.

Informally, the thesis is about activities performed together through time by a group of agents that figure out how to do things better as they go. An agent could be a person, a robot or a computer program. We mathematically explain how to get the overall outcomes we want by telling the agents what they should individually want. Because we do this mathematically, we need to measure the things we want our group of agents to do. This dissertation explains some new ideas about how we can measure how well a group of agents is taking turns, how we can guess whether or not pairs of a certain kind of robot-like computer programs will take turns, and how we can tell individual agents what they should want so that they collectively end up doing something that we want, for some situations.

In this chapter, we introduce the thesis in detail. We explain the organisation of this dissertation, highlight the original contributions to human knowledge in this dissertation, list those parts of this dissertation that have already been published and mention some ideas that are *not* part of the thesis. We briefly discuss some important related work.

1.1 The big picture

An ‘agent’ is an active goal-directed entity. An agent could be a robot, a person, a software program, a country or an animal. A multi-agent system is a set of connected interactive agents that form a complex whole. Multi-agent systems can model communities, teams and computer networks. Machine learning is a broad family of algorithms where the function of the algorithm depends on the input, which can be useful for defining artificial agents. Pattern recognition (Theodoridis and Koutroumbas, 2009) and data mining (Witten and Frank, 2005) are two areas of machine learning. Reinforcement learning is a machine learning framework for controlling agents in stochastic environments where the goal of an agent is to maximise its real-valued ‘reward’ function (Sutton and Barto, 1998). Multi-agent reinforcement learning is a framework for studying multi-agent systems where some or all of the agents use reinforcement learning (Buşoniu et al., 2008).

In some situations, a system designer can control aspects of a multi-agent system. In multi-agent reinforcement learning, the study of system design is called the ‘design agenda’ (Gordon, 2007). In single-agent reinforcement learning, the goal is clear: maximise the reward function. However, in multi-agent systems, each agent has a goal and it may not be possible to simultaneously maximise all the agents’ reward functions (Buşoniu et al., 2008). This leaves system designers with the problem of choosing agents’ rewards so as to induce coordinated behaviour. In this dissertation, we focus on the design agenda of multi-agent reinforcement learning for cases where the system designer has complete control of the agents’ reward functions and the system designer desires a particular sequential joint behaviour.

A sequential joint behaviour is a scheme for the actions of a group of agents at successive instants. Turn-taking in conversation, rhythm in musical ensembles and robotic assembly of cars are examples of coordinated sequential joint behaviours. For convenience, we consider that the joint behaviour sequences are indexed by time. However, the methods we present are not specific to time *per se*; they apply to systems parameterised by any single-dimension quantity. For example, we could apply our techniques to a system that changes state at a number of discrete points on a line through space by replacing the time variable in our models with the appropriate spatial quantity.

How can system designers incentivise desirable sequential joint agent behaviours when multiple agents must simultaneously learn in the face of uncertain environments and possible opposition from other agents? We propose ways for system designers to algorithmically construct appropriate reward functions to influence groups of learning agents toward measurable desired sequential joint behaviours. This dissertation presents methods for measuring one kind of sequential joint behaviour, for analysing

the reward functions of multi-agent systems to predict their behaviour in simulation and for designing reward functions that influence groups of agents toward desired behaviours. This dissertation emphasises turn-taking as a potentially desirable sequential joint behaviour in order to present a concrete and detailed demonstration of our methodology. Our work in multi-agent systems focuses on machine learning in software agents, but our results may have practical applications in areas where the agents are animals, people or interactive robots.

1.2 The organisation of this dissertation

This dissertation contains sufficient research content to substantiate the thesis. Fig. 1.1 illustrates how Chapters 3–5 support different parts of the thesis and how each chapter’s results are internally verified. We assume that the reader is familiar with some notation and concepts from matrix algebra, set theory, probability and statistics. We introduce additional key technical concepts in Chapter 2: Markov chains, Markov decision processes, reinforcement learning, game theory and emergent multi-agent behaviours. Chapter 2 describes the historical and technical foundation that precedes and underlies the thesis.

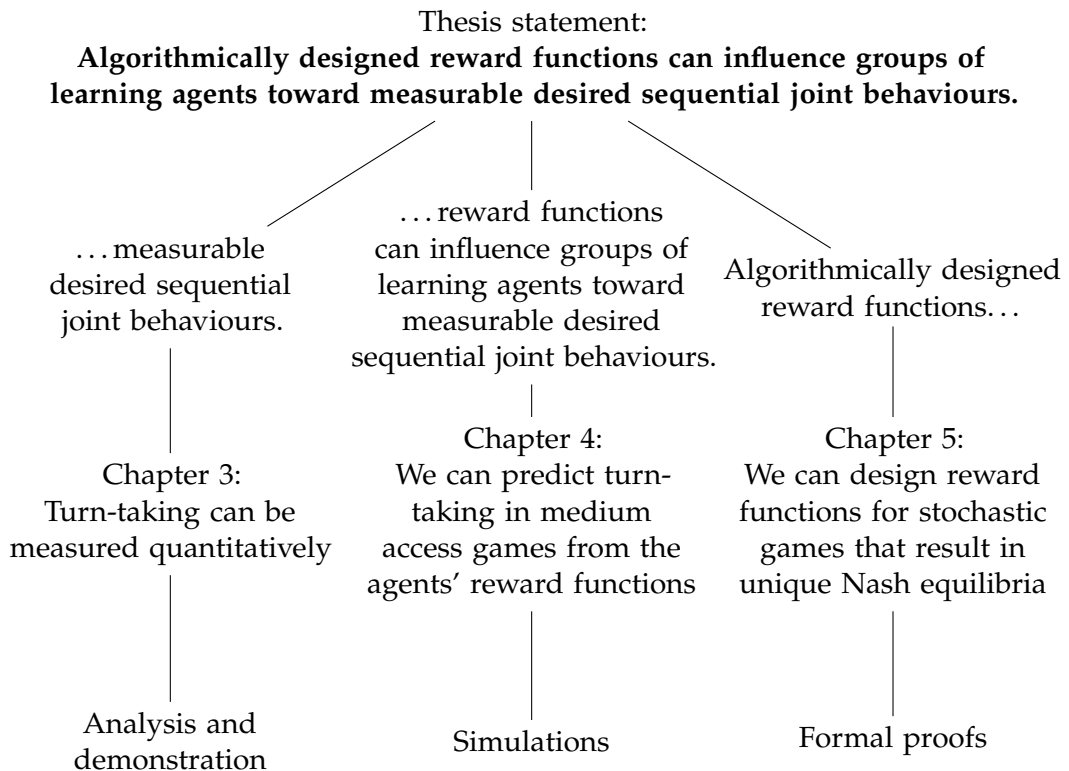


Figure 1.1: Each chapter supports a different part of the thesis and each chapter is internally supported by a different quantitative method.

The thesis requires the desired behaviour to be measurable. Chapter 3 introduces a simple metric for turn-taking. We analyse the performance of random agents, demonstrate the metric on previously reported cases of emergent turn-taking and apply the metric to a recorded human conversation. Chapter 3 supports the thesis by demonstrating how to measure a sequential joint behaviour. Chapter 3 also sets the foundation for Chapter 4. The thesis makes claims only for measurable behaviours and Chapter 4 demonstrates how turn-taking in groups of learning agents can be effected by appropriate reward functions, therefore we must have the metric from Chapter 3 in order to proceed to Chapter 4.

The thesis claims that reward functions can influence groups of learning agents towards measurable sequential joint behaviours. Chapter 4 introduces ‘medium access games,’ a class of multi-agent games that model situations where human and machine turn-taking may be a desirable joint behaviour for groups of learning agents. We analyse medium access games using Nash equilibria (Nash, 1950) and Markov chain techniques, and present simulation results showing which reward functions result in turn-taking for pairs of Q-learning agents (Watkins, 1989). Therefore, Chapter 4 demonstrates how reward functions can influence the sequential joint behaviour of pairs of learning agents toward measurable desired behaviours.

The thesis claims that we can algorithmically design reward functions to influence groups of agents toward desired behaviours. Chapter 5 extends the methodology of Chapter 4 and introduces a method for designing reward functions for stochastic games (Shapley, 1953; Littman, 1994) that result in predetermined desirable Nash equilibria. We formally prove that our method is successful and supply an example application of our method where we induce turn-taking in a three-agent stochastic game. Chapter 5 supports the thesis by demonstrating how to design appropriate reward functions that can influence the sequential joint behaviour of groups of agents.

The technical chapters (Chapters 3–5) are designed to be reasonably self-contained, at a minor expense of repetition. Therefore, rather than including an independent literature review chapter, each chapter reviews the previous work that is directly related to that chapter. We briefly outline the most important related work in Section 1.3. Each chapter introduces the main ideas used in that chapter, develops and explains research content, then proposes future work and discusses alternative approaches. Chapter 6 reiterates the key conclusions of Chapters 3–5, addresses potential criticisms and outlines the future research areas that follow from the thesis. Appendix A presents graph generator algorithms that supplement the material in Chapter 5 but are not essential to the thesis.

1.2.1 Research contributions

This dissertation embodies a number of original contributions to human knowledge. Chapter 3 makes the following contributions to the study of multi-agent systems, emergent communication and emergent turn-taking:

- We propose $\tau\tau$, a quantitative measure for turn-taking based on the intuitive concept that good turn-taking implies fair and efficient use of a shared resource.
- We derive the quantity of turn-taking achieved by random agents. We describe how to compare distributions and individual values of $\tau\tau$ against the performance of random agents.
- We demonstrate the suitability of $\tau\tau$ by re-interpreting two previously reported cases of turn-taking in simulated agents (Di Paolo, 2000; Iizuka and Ikegami, 2004). We present an application of our metric where we measure the quantity of turn-taking present in a recorded human conversation.
- We released an open source software library with an implementation of $\tau\tau$.

Chapter 4 examines reward functions for pairs of Q-learning agents that are conducive (or prohibitive) to turn-taking in simulated medium access games. Chapter 4 contributes to the multi-agent reinforcement learning approach to turn-taking in a number of ways:

- We define ‘medium access games’ to represent situations in human and machine communication where turn-taking might be a desirable joint behaviour.
- We identify which deterministic agent policies produce turn-taking behaviour in medium access games. We use Markov chains to derive the limit-average expected payoffs for agents with stationary stochastic policies, which enable us to compute the Nash equilibria of medium access games.
- We present extensive simulation results for pairs of Q-learning agents with a range of reward functions. We analyse these results using the Nash equilibria of the corresponding medium access games, demonstrating how to predict which reward functions are likely to result in turn-taking as a system behaviour.
- We consider extensions of our model and suggest ways to apply our methodology to analyse reward functions that result in sequential joint behaviours other than turn-taking.

Chapter 5 introduces a method for designing reward functions for stochastic games, making the following contributions to the design agenda of multi-agent reinforcement learning:

- We present a method for designing reward functions for Markov decision processes. We prove that our method results in a Markov decision process where the unique optimal policy is a predetermined desirable policy.
- We suggest methods for designing reward functions for normal form games. We prove that our methods result in games with unique Nash equilibria. We identify cases where a desired joint behaviour cannot be encoded as a Nash equilibrium.
- We integrate our reward design techniques for Markov decision processes and normal form games for a complete method of designing reward functions for stochastic games. We prove that this method results in a unique Nash equilibrium and we give an example demonstrating this method.

1.2.2 Publications

Parts of this dissertation have already been published as journal papers. Specifically, Chapter 3 and Chapter 4 include content published in the interdisciplinary journal *Adaptive Behavior*.

Chapter 3:

Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012b). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104–116

Chapter 4:

Raffensperger, P. A., Bones, P. J., McInnes, A. I., and Webb, R. Y. (2012a). Rewards for pairs of Q-learning agents conducive to turn-taking in medium-access games. *Adaptive Behavior*, 20(4):304–318

Other published work by the author that is not related to this dissertation is:

Raffensperger, P. A. (2012). Toward a wave digital filter model of the Fairchild 670 limiter. In *Proceedings of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, York, UK

Raffensperger, P. A. and Hayes, M. P. (2008). Motion to music interface. In *Proceedings of the Electronics New Zealand Conference*, Auckland, New Zealand

1.2.3 What the thesis is not about

Unlike much research in machine learning, the thesis is not about new machine learning algorithms. We focus on analysing multi-agent systems and designing the

reward functions that guide learning algorithms. Our work is complementary to the development of new machine learning algorithms.

We discuss emergent communication in Chapter 3 and ‘medium access games’ in Chapter 4, but the thesis is not about practical medium access control schemes (Tanenbaum, 2002) or cognitive radio (Mitola and Maguire, 1999; Haykin, 2005). While medium access control and cognitive radio are important and closely related research areas, our emphasis is on multi-agent system analysis and design rather than on building better practical communication systems.

Many mathematical models of real-world systems consider time to be a real-valued quantity. However, like much work in reinforcement learning, we make the simplifying assumption that time is discrete, that systems change state at a countable number of instants. We can represent a band-limited continuous time signal as a discrete time signal without loss of information if we sample the discrete time signal at more than twice the bandwidth of the continuous time signal (Ambardar, 1999, p. 448). Regardless of any limitations imposed by discrete time, the thesis refers only to ‘sequential’ behaviours, that is, behaviours that are fully representable in discrete time systems. While it may be possible to extend our results to continuous time systems, we make no claims about continuous time systems.

1.3 Related research areas

Here, we consider those research areas that are most closely related to the research in this dissertation. This section presents only a brief literature review. In each technical chapter, we explore in more depth the work in the literature that relates to that chapter.

The study of reinforcement learning is based on the premise that an agent can be controlled by a scalar reward function (Sutton and Barto, 1998). More recently, researchers have extended reinforcement learning to multi-agent systems (Buşoniu et al., 2008). Multi-agent reinforcement learning includes the study of how the sequential joint behaviour in groups of learning agents can be influenced by reward functions. This dissertation relates to the design agenda of multi-agent reinforcement learning proposed by Gordon (2007). We discuss reinforcement learning and multi-agent reinforcement learning in Chapter 2.

In Chapter 3, we present a novel metric for turn-taking. Iizuka and Ikegami (2004) used a turn-taking metric for their study of turn-taking in simulated mobile robots. Unlike Iizuka and Ikegami’s turn-taking metric, we can apply our metric to multi-agent systems with any number of agents and our metric can model situations where more than one agent tries to take a turn at one time. We compare and contrast our

metric with Iizuka and Ikegami's metric in more detail in Chapter 3. Our turn-taking metric incorporates metrics for fairness and efficiency. Lan et al. (2009) derive a family of fairness functions, some of which also incorporate a notion of efficiency. However, Lan et al. does not consider the passage of time, which is essential to our turn-taking metric. In spoken dialog systems, turn-taking quality is measured by the lengths of silences between speech and the interruption rates (Raux and Eskenazi, 2008; Turunen et al., 2006). However, silence lengths and interruption rates are less able to distinguish turn-taking from other arbitrary behaviours, which is one of the goals of our turn-taking metric.

Some researchers have investigated how turn-taking emerges in groups of agents. Neill (2003) describes the 'turn-taking dilemma,' a repeated two-agent game where the optimal strategy is for the agents to take turns. Neill's study bears some similarity to our work in Chapter 4 but he does not consider learning agents or specifically examine different payoffs for the agents. Helbing et al. (2005) present the results of an experiment in a computer laboratory where each person in a pair plans to drive either on a slow road or a fast road. Helbing et al. found that people sometimes take turns planning to drive on the fast road and they conjecture that the emergence of turn-taking depends on the rewards for different outcomes. Helbing et al.'s study bears some similarity to Chapter 4, but we focus on turn-taking of Q-learning agents in simulated medium access games rather than people taking turns in congestion games. Colman and Browning (2009) simulated the evolution of cooperative turn-taking in iterated static games played by Moore machines trained with a genetic algorithm. We use our turn-taking metric to evaluate the presence of turn-taking but Colman and Browning use other turn-taking performance measures, such as the agents' average payoffs.

Our reward design method for stochastic games in Chapter 5 builds on proofs of the necessary and sufficient conditions for constructing normal form games with unique Nash equilibria for pairs of agents (Kreps, 1974) and arbitrary numbers of agents (Kreps, 1981) and is closely related to a method for constructing two-agent normal form games with unique predetermined Nash equilibrium (Quintas, 1988). We go beyond the work of Kreps and of Quintas by proposing a method for designing reward functions for stochastic games with arbitrary numbers of agents that result in unique Nash equilibria.

'Mechanism design' is a part of game theory that was pioneered by Leonid Hurwicz, Eric Maskin, Roger Myerson and others. Mechanism design is about constructing games that result in some desirable outcome even when the agents act selfishly. Mechanism design can be automatically achieved (Conitzer and Sandholm, 2002; Sandholm, 2003). Our reward design method in Chapter 5 is related to automated mechanism design where the mechanism designer desires a particular outcome.

However, while mechanism design considers ways to achieve desired outcomes despite the agents' preferences, we consider that the agents have no prior preferences and that we dictate the agent's preferences entirely.

1.4 Summary

We introduce our approach to measuring and influencing sequential joint agent behaviours and we discuss how this dissertation is organised. We mention research that closely relates to the research in this dissertation. Chapters 3–5 are self-contained explorations of different aspect of the thesis: Chapter 3 introduces a metric for turn-taking; Chapter 4 describes how to analyse reward functions to predict turn-taking in medium access games played by pairs of Q-learning agents; and Chapter 5 presents a method for designing reward functions for stochastic games that result in unique predetermined Nash equilibria. We present our conclusions and suggestions for future work in Chapter 6. The research presented in this dissertation is summarised in compressed form by the thesis:

Algorithmically designed reward functions can influence groups of learning agents toward measurable desired sequential joint behaviours.

The thesis is overlaid on a background of research in machine learning and multi-agent systems, which we discuss in greater depth in Chapter 2.

Chapter 2

Background

This dissertation builds on research in multi-agent systems, machine learning, game theory and emergent multi-agent behaviours. Fig. 2.1 illustrates the relationships between those research areas and our work. Chapters 3–5 are written for an audience that is already familiar with these ideas. In this chapter, we explore the most important background ideas to this dissertation; this chapter contains no new technical results.

In Section 2.1, we discuss Markov chains, their extension to Markov decision processes and reinforcement learning, which is a method for solving Markov decision processes. We introduce relevant concepts in multi-agent systems and game theory in Section 2.2, where we also describe multi-agent reinforcement learning. We describe the concept of emergent phenomena in Section 2.3 with particular emphasis on multi-agent emergent behaviours. We summarise this chapter in Section 2.4.

2.1 Markov chains, Markov decision processes and reinforcement learning

We are particularly interested in agents situated in stochastic environments because the set of stochastic models is a superset of the set of deterministic models and stochastic models are at least as good as deterministic models. A ‘Markov chain’ is a tool for modelling stochastic processes and a ‘Markov decision process’ extends the idea of a Markov chain to include an active agent. ‘Reinforcement learning’ is an agent-based way to solve Markov decision processes. A reinforcement learning agent learns ‘online;’ the agent can start solving a Markov decision process with incomplete information and then improve over time.

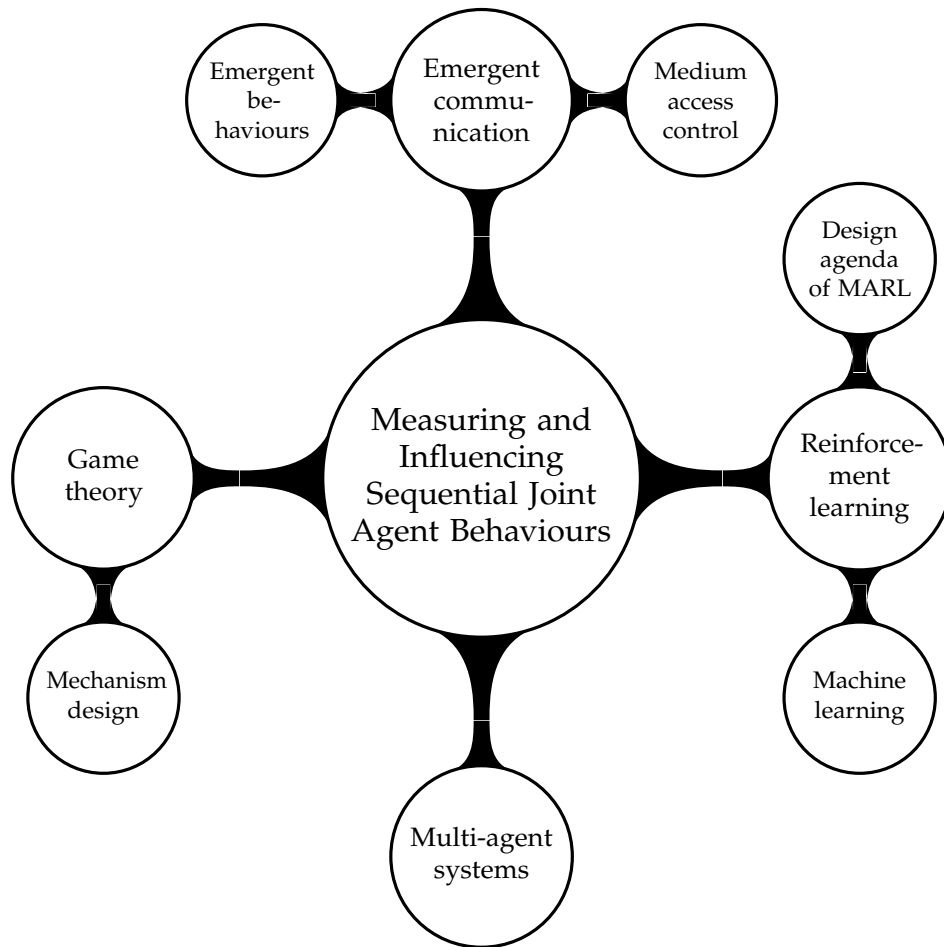


Figure 2.1: The thesis sits in the context of a number of inter-related research areas. The most important areas are multi-agent systems, reinforcement learning, emergent communication and game theory. ‘MARL’ stands for multi-agent reinforcement learning.

2.1.1 Markov chains

Our analysis of sequential joint behaviour in groups of learning agents makes extensive use of Markov chains. A Markov chain is a discrete-state stochastic process where the next state does not depend on past states, but only depends on the present state. Markov chains originated in the early 20th century work of the Russian mathematician Andrey Markov (Markov, 1906; Basharin et al., 2004).

We consider discrete time Markov chains, $t \in \mathbb{Z}$, with finite state sizes. Formally, a Markov chain is a tuple $\mathcal{C} = (\mathcal{S}, T)$ where \mathcal{S} is a set of states and T is a state transition function that indicates the probability of transitioning from one state to another, $T : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$. Consider, for example, a Markov chain, \mathcal{C}^x , with three states, $\mathcal{S} = \{s_1, s_2, s_3\}$, and a state transition function $T(s_i, s_j)$. We can represent \mathcal{C}^x as a state transition diagram, as in Fig. 2.2, where we define T graphically by the

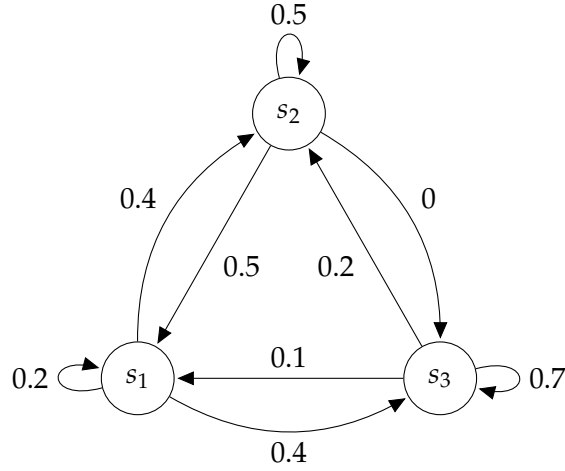


Figure 2.2: An example Markov chain, C^x , with three states. The vertices represent states and the numbers on the arcs represent the state transition probabilities.

Table 2.1: An example state trajectory for the Markov chain shown in Fig. 2.2.

t	0	1	2	3	4	5	6	...
$S(t)$	s_1	s_3	s_3	s_2	s_1	s_2	s_1	...

arcs between states. We can also represent T with a stochastic matrix, \mathbf{P} , such that $[\mathbf{P}]_{ij} = T(s_i, s_j)$. The stochastic matrix for C^x is:

$$\mathbf{P} = \begin{bmatrix} 0.2 & 0.4 & 0.4 \\ 0.5 & 0.5 & 0 \\ 0.1 & 0.2 & 0.7 \end{bmatrix} \quad (2.1)$$

Let $S(t)$ be a random variable that represents the state at time t . Given an initial state $S(0)$, we can create a trajectory of states through time by, at each time step, selecting a new state based on the probability distribution in the row of \mathbf{P} determined by the current state. Table 2.1 shows one such trajectory.

We represent a probability distribution over states at time t with a column vector $\mathbf{s}(t)$, such that $\mathbf{s}(t) = (p[S(t) = s] \text{ for } s \in \mathcal{S})^T$. Given such a vector, we can calculate the distribution of states at time $t + 1$ using \mathbf{P} :

$$\mathbf{s}(t + 1) = \mathbf{P}\mathbf{s}(t) \quad (2.2)$$

A state, s , in a Markov chain is 'periodic' with period L if and only if L is the largest possible integer such that $p[S(t + k) = s | S(t) = s] = 0$ for all k such that k is not divisible by L (Ching and Ng, 2005, p. 14). A state is 'aperiodic' if $L = 1$; if all states in a Markov chain are aperiodic, then the Markov chain is also said to be aperiodic. A

Markov chain is ‘irreducible’ if every state can be reached from every other state in a finite number of time steps with non-zero probability. If a Markov chain is irreducible and aperiodic, then there exists at least one stationary distribution of states, \bar{s} , such that:

$$\bar{s} = P\bar{s} \quad (2.3)$$

Furthermore, in an irreducible aperiodic Markov chain, all distributions of states will approach a stationary distribution of states as t approaches infinity (Ching and Ng, 2005, p. 15). We use the stationary distribution of states of Markov chain for our analysis in Chapter 4. For more information on Markov chains, see Ching and Ng (2005), Häggström (2002), or Appendix M of Filar and Vrieze (1997).

2.1.2 Markov decision processes

The idea of a Markov decision process builds on the concept of a Markov chain by considering that the state changes not only depend on the state transition probabilities but also depend on the actions of a goal-directed agent. The goal of the agents is to act so as to maximise a reward function. The idea of a Markov decision process is a powerful model for discrete stochastic control problems: we can describe desirable outcomes with the reward function in such a way that the necessary actions to maximise that reward function can be calculated.

Formally, a Markov decision process is a tuple, $\mathcal{D} = (\mathcal{S}, \mathcal{A}, T, R)$ where \mathcal{S} is a set of states, \mathcal{A} is a set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition function and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is a reward function. Fig. 2.3 shows part of an example Markov decision process. An agent’s ‘policy’ summarises its action choices, and is a stochastic mapping between states and actions, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. In the context of Markov decision processes, we use the terms ‘policy’ and ‘behaviour’ interchangeably.

Observe that if the agent fixes its policy, then the Markov decision process reduces to a Markov chain. If the agent follows policy π , where $\pi(s, a)$ represents the probability that the agent will choose action a in state s , then the system dynamics are fully described by a Markov chain with the same set of states as the original Markov decision processes and with state transition probabilities calculated as:

$$T^\pi(s, s') = \sum_{a \in \mathcal{A}} T(s, a, s') \pi(s, a) \quad (2.4)$$

for all $s \in \mathcal{S}$ and all $s' \in \mathcal{S}$, where T^π is the state transition function of the new Markov chain and T is the state transition function of the original Markov decision process.

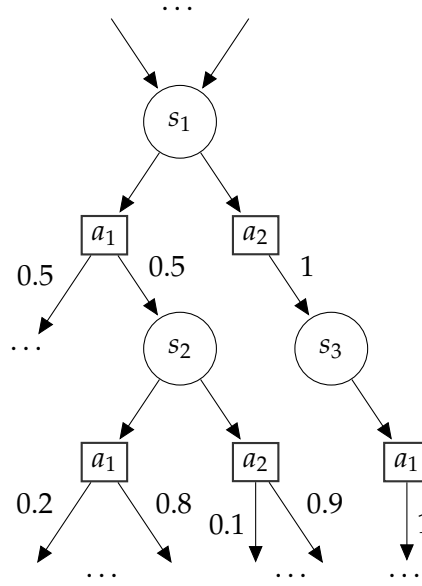


Figure 2.3: An example showing part of a Markov decision process. The circles represent states and the squares represent actions. The arrows from states to actions are agent choices while the arrows from actions to states occur with probabilities given by the state transition function.

The goal of the agent is to find an ‘optimal’ policy. A policy is optimal if it maximises the agent’s ‘payoff’ given the starting state of the system, for all starting states. We call the way that we calculate the agent’s payoff the ‘valuation scheme.’ For all the valuation schemes mentioned in this dissertation, every Markov decision process has an optimal deterministic policy. A deterministic policy is one where there exists some action a such that $\pi(s, a) = 1$ for all $s \in \mathcal{S}$.

We can compute the agent’s payoff in different ways. For example, we could maximise the expected reward for the current time step, t :

$$v_m(s, \pi) = E_{s, \pi}[R(t)] \quad (2.5)$$

where $E_{s, \pi}[R(t)]$ is a random variable representing the reward at time t given that the system starts in state s and that the agent follows policy π . This valuation is called ‘myopic’ because the agent may earn smaller rewards in the long run by taking larger immediate rewards. Alternatively, we can make the agent focus on long-term expected rewards by setting the agent’s payoff to the limit-average expected reward, which, for state s at time t , and the agent’s policy, π , we compute as:

$$v_a(s, \pi) = \lim_{k \rightarrow \infty} \frac{1}{k} \sum_{t'=t}^{k+t} E_{s, \pi}[R(t')] \quad (2.6)$$

The myopic valuation and the limit-average valuations are extremes on the continuum between valuing rewards that are close in time to the present and valuing high rewards over a long time span. A more flexible valuation scheme is the γ -discounted valuation, where we explicitly control the degree to which the agent prefers earlier rewards to later rewards by choosing the value of γ . We compute the γ -discounted expected reward for the agent as:

$$v_\gamma(s, \pi) = (1 - \gamma) \sum_{t'=t}^{\infty} \gamma^{t'-t} E_{s, \pi}[R(t')] \quad (2.7)$$

The $1 - \gamma$ term is a normalising factor so that $|v_\gamma(s, \pi)| \leq \max\{|R(t)|\}$. Usually, we restrict γ to the range $[0, 1)$.

The best choice of valuation scheme depends on the situation. Myopic valuation is usually used to simplify situations, since it is straightforward to compute. The limit-average valuation also has a computational advantage: if the agent's policy reduces the Markov decision processing into an irreducible aperiodic Markov chain, then the limit-average valuation does not depend on the initial state. We make use of this property of the limit-average valuation in Chapter 4. The γ -discounted valuation has great flexibility: for $\gamma = 0$, the γ -discounted valuation is equivalent to the myopic valuation and the γ -discounted valuation becomes similar to the limit-average valuation as γ approaches 1. For more information on Markov decision processes, see Puterman (1994).

2.1.3 Reinforcement learning

Machine learning is the study of algorithms that change their function to suit their input. 'Learning', 'adaption' and 'evolution' are concepts borrowed from psychology and biology. While algorithms in machine learning draw inspiration from nature, it is best to think of machine learning as a broad family of computational algorithms rather than ways of producing 'artificial intelligence.' The term 'artificial intelligence' is plagued by cultural expectations and the philosophical problem of defining intelligence in a definite and measurable way. Consequently, we adopt the term 'machine learning.'

Hand-programmed algorithms have the advantage of performing well on their specific task, but there are situations where machine learning is important. Sometimes small changes in the definition of the problem can have enormous consequences on the solution. We use learning algorithms to solve problems where the algorithm itself depends on the data. Learning algorithms are useful for meta-design; if we can quantify 'good design' but are unable to search the design space efficiently by hand, then we can use adaptive algorithms to find good solutions. Adaptive algorithms are

useful for defeating uncertainty and constantly learning online algorithms are useful for solving non-stationary problems.

Machine learning encompasses a broad range of problem domains including pattern recognition (Theodoridis and Koutroumbas, 2009), data mining (Witten and Frank, 2005), natural language processing (Manning and Schütze, 1999) and optimisation (Goldberg, 1989). Reinforcement learning is the branch of machine learning that focuses on control in uncertain environments (Sutton and Barto, 1998). In particular, reinforcement learning is a way to solve Markov decision processes in an online manner. Andreae did some of the earliest work on reinforcement learning at the University of Canterbury with STeLLA (Andreae, 1963; Andreae and Cashin, 1969; Andreae, 1969) and later PURR-PUSS (Andreae, 1977, 1998).

Reinforcement learning is about choosing actions for an agent in a stochastic environment so as to maximise the agent's 'reward' function. Sutton and Barto describe reinforcement learning concisely:

Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics—trial-and-error search and delayed reward—are the two most important distinguishing features of reinforcement learning. (Sutton and Barto, 1998, pp. 3-4)

Reinforcement learning draws inspiration from the now outdated (Pinker, 1999) psychological theory of behaviourism that Skinner (1938) and other researchers pioneered. Behaviourism is the idea that we can condition animals' behaviour by positive and negative reinforcements. For example, near the start of the 20th century, the Russian scientist Pavlov conditioned dogs to salivate at the sound of a bell by ringing the bell before feeding the dogs (Todes, 2001). In a similar manner, reinforcement learning algorithms learn a policy from the scalar reward function of a Markov decision process.

Traditionally, reinforcement learning models Markov decision processes with two parts: an environment and an agent, as shown in Fig. 2.4. The agent chooses actions based on the state of the environment so as to maximise its reward function (Sutton and Barto, 1998). The environment encapsulates the set of states, the reward function and the state transition function. We assume that the agent starts without complete knowledge of the environment; the agent must learn the dynamics of the environment at the same time as it learns to maximise its reward function. We can also apply

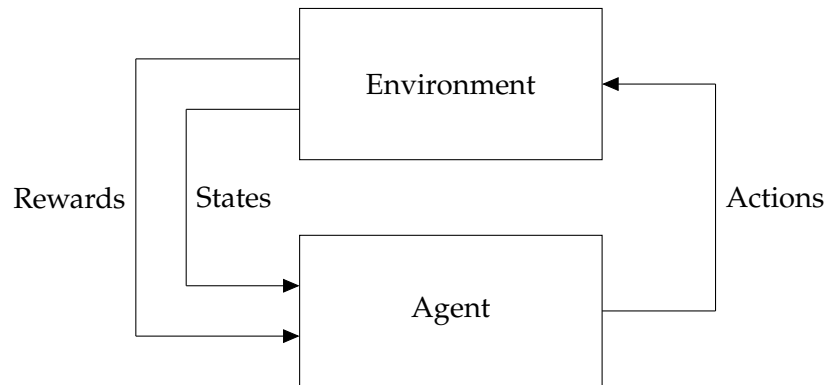


Figure 2.4: Single-agent reinforcement learning models situations as an environment and an agent.

reinforcement learning in multi-agent settings (Buşoniu et al., 2008); we discuss multi-agent reinforcement learning in Section 2.2.4, after we introduce stochastic games.

One popular single-agent reinforcement learning algorithm is Q-learning, which provably converges to the optimal policy when solving a Markov decision process (Watkins, 1989). A Q-learning agent maintains a table of state-action values, or Q-values. Initially, we set the table with some arbitrary values; usually we use large initial Q-values, for an optimistic learning bias (Sutton and Barto, 1998). When the agent takes action a in state s , the next state is s' and the agent earns reward $R(s, a, s')$. The Q-learning algorithm updates the Q-table as:

$$Q(s, a) \leftarrow (1 - \beta)Q(s, a) + \beta \left(R(s, a, s') + \gamma \max_{a' \in \mathcal{A}} \{Q(s', a')\} \right) \quad (2.8)$$

where γ is the discount factor of the Markov decision process and β is a 'learning rate' that determines how quickly the algorithm adapts to new information. This Q-learning update function assumes that the environment is a Markov decision process with the γ -discounted valuation. Q-learning is an off-policy learning method; that is, a Q-learning agent can follow a wide range of different policies and still learn the right action values. Usually, we couple Q-learning with an 'exploration policy' that describes how to select actions. We describe considerations relevant to exploration policies and give an example policy in Section 2.1.4.

Notice that the variant of Q-learning stated here stores a real value for each state-action pair. However, the number of states may be large (Sutton and Barto, 1998, §8). In general, we must address this difficulty in reinforcement learning systems, but we side-step this issue by limiting ourselves to simulations with small numbers of states. For one way to defeat the problems with Q-learning and large numbers of states, see the recent algorithm by Sutton et al. (2009).

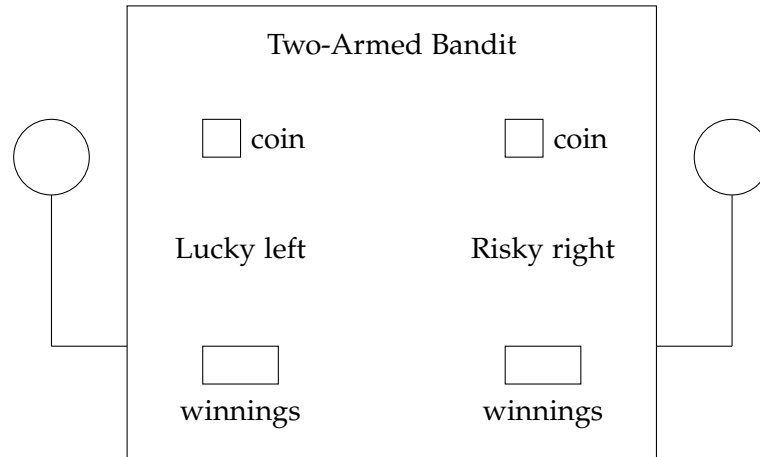


Figure 2.5: The two-armed bandit problem serves to illustrate the balance between exploitation of the best known rewards and exploration to identify the best rewards.

2.1.4 Balancing exploration and exploitation

Q-learning is a way of updating state-action values to keep a running estimate of the value of each action in each state. However, if the agent always selects the action with the highest value given the current state, then the agent may be misled if it receives an unusually low reward for the action with the highest expected reward early in its learning processes. The agent has to balance its ‘exploitation’ of known good rewards with ‘exploration’ for currently unknown good rewards. This difficulty is not specific to Q-learning; all reinforcement learning algorithms must balance exploration with exploitation.

The multi-armed bandit problem is a simple stochastic decision problem that serves to illustrate the important balance between exploration for the highest rewards and exploitation of the highest known rewards (Ishikida and Varaiya, 1994; Gittins et al., 2011). Consider a gambling machine similar to a ‘one-armed bandit’ slot machine found in casinos but with more than one arm, as illustrated in Fig. 2.5 for the case of two arms.

The agent chooses to place a coin in either the ‘Lucky left’ slot or the ‘Risky right’ slot and pulls the level for the chosen side. The two-armed bandit then releases winnings with a value randomly sampled from a distribution determined by choice of arm. The agent repeats this process a number of times, trying to maximise its winnings. Suppose that the lucky left arm has a mean value of μ_L and a standard deviation of σ_L , while the risky right arm has a mean value of μ_R and a standard deviation of σ_R . Assume that the lucky left arm is the best choice, but has a larger standard deviation

than the right arm:

$$\begin{aligned}\mu_L &> \mu_R \\ \sigma_L &> \sigma_R \\ \sigma_L &> \mu_L - \mu_R\end{aligned}\tag{2.9}$$

The agent chooses a slot for a number of trials. Assuming the agent does not know Eq. (2.9), how should the agent allocate trials to the two arms? The arm with the highest mean reward may not always produce the highest reward on each trial. Each trial provides information about the distributions of the two arms, but at no trial does the agent ever have certainty as to which arm has the highest mean value. The agent must balance trials that explore for the best arm while simultaneously exploiting the winnings offered by the arm with the highest currently estimated value.

There are a variety of strategies for balancing exploration with exploitation. One method is to allocate some trials to each arm and to allocate the remaining trials to the arm that has the highest estimated value after that point (Goldberg, 1989, p. 37). Another method is the ‘ ϵ -greedy’ policy, where the agent chooses the most valuable arm (the ‘greedy’ choice) with probability $1 - \epsilon$, and chooses a random arm with probability ϵ (Sutton and Barto, 1998). Common values for ϵ are 0.1 or 0.01. Throughout this dissertation, we use ϵ -greedy policies when we need agents to balance exploration and exploitation. To learn more about reinforcement learning and the exploration-exploitation balance, see the excellent book by Sutton and Barto (1998).

2.2 Multi-agent systems and game theory

A multi-agent system is a set of related agents where each agent’s actions may affect the other agents. While single-agent models comprise an agent and an environment, like in Fig. 2.4, a multi-agent system has multiple agents interacting with each other. In multi-agent systems, we can view the environment as a special, singleton agent. Alternatively, an agent could view all of the other agents as part of its environment. Multi-agent systems are used to model financial trading situations (Sherstov and Stone, 2005), to describe robotic soccer (Kitano et al., 1997), to automatically generate musical rhythms (Eigenfeldt, 2007) and to model many other situations (Shoham and Leyton-Brown, 2009). Each agent may have its own reward function which may or may not be related to the other agents’ reward functions. Game theory provides us with methods for analysing multi-agent systems where each agent is interested in maximising its reward function.

2.2.1 Multi-agent systems

A number of algorithms and problems in search and optimisation can be cast in a distributed way and solved with agents (Shoham and Leyton-Brown, 2009). While Moore's law has afforded us great increases in computing power (Moore, 1965), multi-agent systems have an intrinsic advantage over centralised algorithms because it is hard to increase the speed of each computation process, but it is natural to assign each agent to its own processes. Most supercomputers have abundant parallelism, but have comparable clock speeds to personal computers. Multi-agent systems have the important advantage of being easier to implement with parallel processing.

We can classify multi-agent systems based on the relationships between the agents' goals. A 'cooperative' multi-agent system is one where the agents share the same goal and each agent acts in conjunction with the other agents to achieve that goal. For example, one kind of multi-agent system can find the shortest path between two vertices in a graph (Shoham and Leyton-Brown, 2009); each agent performs its function and the goal is achieved. On the other hand, a 'non-cooperative' multi-agent system is one where each agent has its own reward function and each agent simply tries to maximise its own reward function. For example, we might model a financial trading system as a non-cooperative multi-agent system and reward each agent based on how much money it earns. To confuse matters slightly, the agents in a non-cooperative multi-agent system can be rewarded for the performance of the group which means that the agents benefit from cooperating with each other. The key feature of a non-cooperative multi-agent system is that each agent will maximise its own reward function regardless of the other agents' reward functions. Thus the agents in a non-cooperative multi-agent system are intrinsically self-interested, even if their self-interest might incidentally lead to cooperative behaviour. To learn more about multi-agent systems, see Shoham and Leyton-Brown (2009).

2.2.2 Normal form games and Nash equilibria

Game theory provides important methods for the study of multi-agent systems. The central thesis of game theory is that we can model some microeconomic situations as mathematical games (von Neumann and Morgenstern, 1944). We can view a non-cooperative multi-agent system as an economy by interpreting the agents' reward functions as having monetary values. A seminal result in non-cooperative game theory is the 'Nash equilibrium' (Nash, 1950), which describes how ideal agents might behave.

The concept of the Nash equilibrium is best illustrated in the game-theoretic framework of 'normal form games.' Rock-paper-scissors is a popular game that is often played as

a normal form game. In a normal form game, at a single instant, each agent selects an action then each agent is rewarded based on the combination of the agents' actions. In rock-paper-scissors, each of two people simultaneously displays one of three hand symbols which represent rock, paper or scissors, where rock defeats scissors, scissors defeats paper and paper defeats rock. The winner receives a high reward and the loser receives a low reward; if both people choose the same action, then both receive a medium reward. Formally, a normal form game is a tuple $\Gamma = (\mathcal{N}, \mathcal{A}, R)$ where \mathcal{N} is a set of agents, \mathcal{A} is the joint action space, \mathcal{A}_n is the action space of agent n , $R : \mathcal{A} \rightarrow \mathbb{R}^{|\mathcal{N}|}$ is the vector-valued joint reward function and $R_n : \mathcal{A} \rightarrow \mathbb{R}$ is the reward function for agent n . We consider $a_n \in \mathcal{A}_n$ to be one of agent n 's actions, $\mathbf{a} \in \mathcal{A}$ to be a joint action and $\mathbf{a}_{-n} \in \times_{m \in \mathcal{N}, m \neq n} \mathcal{A}_m$ to be a vector of actions that includes actions for all agents except agent n .

Rather than restricting agents to selecting a single action, we allow agents to select probability distributions over their actions. In the context of normal form games, we call the agent's probability distribution over its actions the agent's 'policy'; π_n represents agent n 's policy and $\pi_n(a_n)$ is the probability that agent n will chose action a_n . We assume that each agent chooses its policy to maximise its expected reward. The traditional assumptions of game theory are that the agents act 'rationally' in the sense that they prefer higher rewards to lower ones (von Neumann and Morgenstern, 1944; Arrow, 1986) and that all agents know that the other agents will act rationally. A rational agent always acts in a self-serving way and takes advantage of all the information available to it. Many multi-agent reinforcement learning algorithms fail to achieve the game-theoretic ideal of rationality (Buşoniu et al., 2008).

A Nash equilibrium is a distribution over joint agent actions that rational agents with perfect information about the reward function would play. At a Nash equilibrium, no agent can improve its reward function by unilaterally changing its policy. Formally, a Nash equilibrium is a joint policy, π , such that each agent's policy is a 'best response' to the other agent's policies. We define agent n 's best response in terms of $v_n(a_n)$, that agent's expected value for taking action a_n , given that the other agents have fixed their policies at π_m for $m \in \mathcal{N}, m \neq n$:

$$v_n(a_n) = \sum_{\mathbf{a}_{-n} \in \times_{m \in \mathcal{N}, m \neq n} \mathcal{A}_m} R_n(a_n, \mathbf{a}_{-n}) \prod_{m \in \mathcal{N}, m \neq n} \pi_m(a_m) \quad (2.10)$$

where a_m is agent m 's action in \mathbf{a}_{-n} . An agent's policy, π_n , is a best response to the other agents' policies if and only if the expected value for each action that agent n plays with non-zero probability is equal to the expected value of the most valuable action. That is, for all $a_n \in \mathcal{A}_n$, the inequality $\pi_n(a_n) > 0$ implies that:

$$v_n(a_n) = \max_{a'_n \in \mathcal{A}_n} \{v_n(a'_n)\} \quad (2.11)$$

Table 2.2: An example normal form game where the agents have conflicting interests, commonly called the ‘Prisoners’ Dilemma.’ The reward function values are shown in the form of reward for row agent, reward for column agent.

		Column agent	
		<i>C</i>	<i>D</i>
Row agent	<i>C</i>	6, 6	0, 10
	<i>D</i>	10, 0	2, 2

An agent’s policy is its best response to the other agents’ policies if that policy maximises its expected payoff. If a policy is an agent’s best response, then that policy must take all sub-optimal actions with zero probability (von Stengel, 2007, Proposition 3.1 on p. 55). At a Nash equilibrium, each agent’s policy is its best response to the other agents’ policies. A Nash equilibrium is ‘pure’ if each agent chooses an action with probability one and is ‘mixed’ otherwise. In general, a game may have more than one Nash equilibrium, but every normal form game has at least one Nash equilibrium (Nash, 1950). The Nash equilibrium of rock-paper-scissors is a mixed one where both people play each action with equal probability.

The beauty of a Nash equilibrium is that it predicts the expected outcome of a normal form game in slightly idealised conditions. In general, we may not be able to simultaneously maximise all the agents’ reward functions; the agents may have conflicted interests. A Nash equilibrium allows us to analyse situations where the agents’ reward functions can be simultaneously maximised as well as situations where the agents’ interests are conflicted. For example, consider the game Γ^{PD} where $\mathcal{N} = \{\text{Column agent, Row agent}\}$ and $\mathcal{A}_n = \{C, D\}$ for $n \in \mathcal{N}$ and with payoffs as shown in Table 2.2. This game is a variation on the ‘Prisoners’ Dilemma’ (Flood, 1958) which we can interpret as a situation where the police apprehend two criminals and each prisoner has the opportunity to either cooperate (*C*) with his accomplice or to admit guilt to the police, defecting (*D*) against his partner. The prison sentence for each criminal depends on the actions of both agents. While the sum of the agents’ reward functions is maximised when both agents cooperate, each individual agent expects a greater payoff by choosing to defect regardless of the other agents’ actions. Consequently, the Nash equilibrium for the game in Table 2.2 is pure: both agents defect.

Mechanism design (Maskin, 1985) is the study of constructing games that result in some desirable outcome as the Nash equilibrium of the game. While our research does not build directly on mechanism design, mechanism design is similar in spirit to our work in Chapter 5. For more information on game theory, normal form games, Nash equilibrium and mechanism design, see the helpful series of books edited by Aumann and Hart (1992; 1995; 2002).

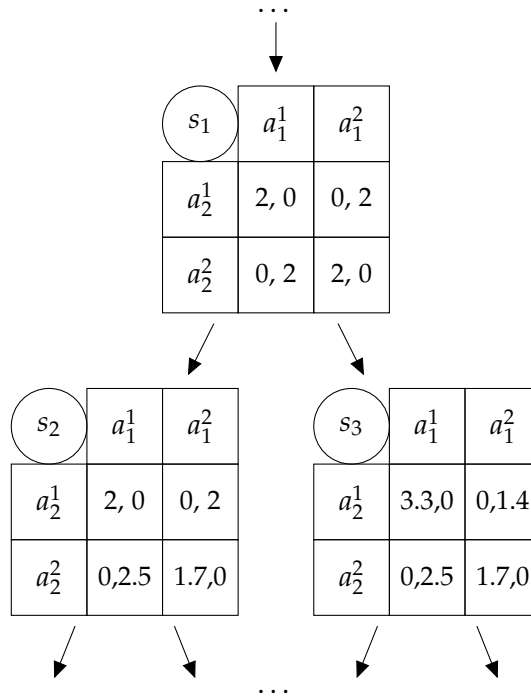


Figure 2.6: An example showing part of a stochastic game with two agents and two actions per agent. The values in the tables are the rewards to the agents for taking that combination of actions in that state.

2.2.3 Stochastic games

The idea of a ‘stochastic game’ generalises the ideas of a Markov decision process and of a normal form game. We can think of a stochastic game as a multi-agent Markov decision process or as a normal form game with states. The set of stochastic games is a subset of the set of non-cooperative multi-agent systems. We make extensive use of stochastic games in Chapter 5.

Formally, a stochastic game is defined as a tuple $\mathcal{G} = (\mathcal{N}, \mathcal{S}, \mathcal{A}, T, R)$ where \mathcal{S} is a set of states, \mathcal{A} is a joint action space, T is a state transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{N}|}$ is a joint reward function. Fig. 2.6 shows part of an example stochastic game; in essence, the agents play a normal form game at each state and the state transitions depend on the current state and the agents’ actions. The goal of each agent is to control its actions so as to maximise its expected rewards. Each agent maintains a policy, π_n , that is a mapping between states and probability distributions over actions such that $\pi_n(s, a_n) \in [0, 1]$ and $\sum_{a_n \in \mathcal{A}_n} \pi_n(s, a_n) = 1$, for all $s \in \mathcal{S}$. In the context of a stochastic game, an agent’s ‘behaviour’ is equivalent to its policy and the ‘joint behaviour’ of a group of agents is equivalent to the agents’ joint policy.

Stochastic games bear much similarity to single-agent Markov decision processes. In a stochastic game, each agent's goal is to maximise its payoffs, which we can base on the myopic valuation, the limit-average valuation or the γ -discounted valuation in the same way as in a Markov decision process. However, unlike in a Markov decision process, each agent must cater for the possibility that another agent might exploit it. This implies that the agents cannot necessarily find an optimal policy for a stochastic game in the set of deterministic policies. An agent may have to randomise between more than one action to prevent other agents from exploiting the predictability of a deterministic policy.

If we fix the agents policies', then we can reduce a stochastic game to a Markov chain. The Markov chain will have the same set of states as the stochastic game and will have a state transition function, T^π , computed as:

$$T^\pi(s, s') = \sum_{\mathbf{a} \in \mathcal{A}} T(s, \mathbf{a}, s') \prod_{n \in \mathcal{N}} \pi_n(s, a_n) \quad (2.12)$$

where a_n is agent n 's action in the joint action \mathbf{a} .

Stochastic games have Nash equilibria like normal form games. A Nash equilibrium for a stochastic game is a joint policy such that no agent can increase its payoff by unilaterally changing its policy. There exists a Nash equilibrium in the set of stationary stochastic policies for every stochastic game (Filar and Vrieze, 1997, Theorem 3.8.1, p. 130).

We consider that the agents in a stochastic game have complete knowledge of the current system state. However, a richer set of models is the set of partially observable stochastic games, where agents do not observe the full state. An agent in a partially observable stochastic game receives 'observations' which allow the agent to estimate a probability distribution over states. While the idea of partially observable stochastic games allows for accurate models, finding optimal strategies in a partially observable stochastic game is computationally intractable (Bernstein et al., 2002). We define medium access games in Chapter 4 to be partially observable, however we use the Q-learning single-agent reinforcement learning technique rather than a more complicated algorithm dedicated to partially observable stochastic games. To learn more about stochastic games, see Filar and Vrieze (1997); to learn more about partially observable stochastic games, see Kaelbling et al. (1998) and Hansen et al. (2004).

2.2.4 Multi-agent reinforcement learning

Just as we can use single-agent reinforcement learning to solve Markov decision processes, we can use multi-agent reinforcement learning to solve stochastic games.

Multi-agent reinforcement learning (MARL) is both a class of machine learning algorithms and a paradigm for applying the principles of reinforcement learning to multi-agent systems.

Multi-agent reinforcement learning has challenges inherited from single agent reinforcement learning, such as the trade-off between exploration and exploitation, as well as challenges specific to multi-agent reinforcement learning. The additional challenges are the difficulty of choosing a learning goal for the agents, the non-stationary nature of the agents' learning and the need for the agents to coordinate their actions. The learning goal of the agents must balance stability and adaptation. The overall goal of multi-agent reinforcement learning is a challenge because the agents' rewards may be correlated and so cannot be maximised independently. Agents must coordinate their actions so as to break ties between equally good actions.

Some researchers use single-agent reinforcement learning algorithms for multi-agent reinforcement learning because single-agent algorithms can be simpler. 'However, the nonstationarity [*sic*] of the MARL problem invalidates most of the single-agent RL theoretical guarantees' (Buşoniu et al., 2008, p. 10). Researchers have developed a number of multi-agent reinforcement learning algorithms, from Fictitious Play (Brown, 1951) to more recent examples:

- WoLF-PHC (Bowling and Veloso, 2002)
- AWESOME (Conitzer and Sandholm, 2003)
- Nash-Q (Hu and Wellman, 2003)
- MetaStrategy (Powers and Shoham, 2005)

A large number of other algorithms have been proposed (Sen et al., 1994; Littman, 1994; Matarić, 1997; Claus and Boutilier, 1998; Hu and Wellman, 1998; Singh et al., 2000; Littman, 2001; Jafari et al., 2001; Suematsu and Hayashi, 2002; Kapetanakis and Kudenko, 2002; Wang and Sandholm, 2002; Banerjee and Peng, 2003; Kononen, 2003; Zinkevich, 2003; Greenwald and Hall, 2003; Weinberg and Rosenschein, 2004). For a comparison of a number of algorithms, see Crandall and Goodrich (2010).

Shoham et al. (2007) outlines five different agendas for research in multi-agent reinforcement learning:

- Computational: how can we solve stochastic games in an online fashion, particularly to find Nash equilibria?
- Descriptive: how can we model living learning agents, such as animals or people?
- Normative: which sets of learning algorithms converge to a Nash equilibrium?

- Prescriptive, both cooperative and non-cooperative kinds: how should agents learn? How can we design an agent that earns high rewards when it plays against a range of other agents? (Both of Shoham et al.'s prescriptive agendas have been merged into this one bullet point.)

Gordon (2007) adds two additional agendas:

- Modelling: how can we model real-world situations? What are appropriate simplifications to a model?
- Design: 'How can we best use this flexibility [that is inherent in most multi-agent systems] to optimize criteria such as ease of learning, uniqueness of the final learned policies, or predicted welfare of one or more of the agents?' (Gordon, 2007, p. 392).

Our work in Chapters 4 and 5 relates to the design agenda of multi-agent reinforcement learning. To learn more about multi-agent reinforcement learning, see the thorough review article by Buşoniu et al. (2008).

2.3 Emergent multi-agent behaviours

We want to design multi-agent systems that exhibit turn-taking and other sequential behaviours. Sometimes system designers will deliberately induce particular desirable joint behaviours: we could consider this 'designed' behaviour. In multi-agent learning systems, sometimes the agents behave in a way that surprises the system designer. These surprising behaviours are often called 'emergent' behaviours.

Our research seems to embrace a dichotomy: we are interested in designing emergent behaviours. However, in some ways this dichotomy is only apparent and in other ways it is not a dichotomy. Chapter 3 is concerned with measuring the sequential multi-agent behaviour that is colloquially called 'turn-taking' and we develop our turn-taking metric in the context of emergent behaviours. In Chapter 4 we demonstrate techniques for analysing agents' reward functions to predict the behaviour of the agents. In some ways, Chapter 4 is concerned with 'designing emergent behaviours' but it should be considered as a way to predictably design behaviours that were previously considered emergent. We discuss this distinction further in Chapter 4. Chapter 5 is concerned with designing reward functions for stochastic games and therefore the behaviours within belong to the conceptual category of designed behaviours. By extending the range of possible joint sequential behaviours that are measurable and by providing designers with methods to design reward functions, we hope to invade the domain of emergent behaviours and stake a claim for the space of designed behaviours.

2.3.1 What is emergence?

People intuitively assume that the complexity of a system is related to the sum of the complexities of its component parts. However, ‘the whole is greater than the sum of its parts’ (Aristotle, 1924, book 8.6.1045a:8-10). In fact, the complexity of a system can increase much faster than the linear sum of its parts, often exponentially or factorially, because of the many different ways to connect each part. Because people tend to ignore the importance of the connections between parts, the difference between the perceived ‘sum of the parts’ and the behaviour of the complete system seems to have come from nowhere, to have *emerged*.

Monro (2007) explores definitions of emergence in the context of generative art in his Masters thesis. Some examples of emergence are Langton’s ant (Langton, 1986), Conway’s game of life (Gardner, 1970) and Craig Reynolds’ Boids (Reynolds, 1987). Monro considers a number of definitions of emergence, including:

- ‘Complete knowledge of the *construction* of a system does not imply complete knowledge of its *behaviour*’ (Monro, 2007, p. 18).
- ‘Simple rules give rise to complex behaviour’ (Monro, 2007, p. 20).
- ‘Many simple agents together produce complex behaviour’ (Monro, 2007, p. 21).
- ‘The fastest way to predict what the system is going to do is to simulate it (run it and see what it does, if it is a computer system)’ (Monro, 2007, p. 23).

In the context of non-cooperative multi-agent systems, we may label a behaviour ‘emergent’ when it is in fact a Nash equilibrium, or some set of behaviours close to a Nash equilibrium. Computation of a Nash equilibrium is difficult; more precisely, it belongs to the computation complexity class (Arora and Barak, 2009) of PPAD-complete problems (Chen and Deng, 2006). Consequently, a system designer may have no faster way of computing the Nash equilibrium than by observing the system, which fits with the final definition of emergence by Monro listed above.

Often engineers specifically design systems to have loosely coupled parts (Stevens et al., 1974), decreasing the complexity of the whole given constant complexity of the parts and reducing the surprising emergent behaviours of the completed system. (In software development, emergent behaviours are usually called ‘bugs.’) However, if the coupling between parts is not limited by design then the space of system behaviours is bigger. For example, the one dimensional cellular automaton known as Rule 110 is Turing complete (Cook, 2004). Since apparently complex behaviours result from simple rules, we propose exploring methods for crafting the behaviour of multi-agent systems with tight coupling between the agents.

2.3.2 Emergent communication

Our turn-taking metric in Chapter 3 is particularly designed for the context of emergent communication. The study of emergent communication is useful for understanding the origins of human communication and language and also for analysing the different ways that communication can emerge in artificial systems. Research in emergent communication relates to a number of questions:

- What good can communication do for agents? (Gmytrasiewicz and Durfee, 2001)
- How can agents gain the ability to communicate when they start with no such ability? (Quinn, 2001)
- How can mutually intelligible languages emerge? (Smith, 2002; Goldman et al., 2007)
- What conditions result in truthful communication and what conditions result in deception? (Floreano et al., 2007)

The emergence of language in the history of human life intrigues some psychologists and biologists. Our species has a survival advantage by having language (Pinker, 1995). Communication is a cooperative behaviour, so emergent communication is related to the evolution of cooperation (Axelrod and Hamilton, 1981). Researchers use computer simulations to explore multi-agent games that approximate aspects of the history of language; for example, see Cangelosi and Parisi (2002).

2.3.3 Emergent turn-taking

Human language emerged in a social, multi-agent context and conversation became the central mode of language use (Sacks et al., 1974, p. 722). Turn-taking is a key feature of conversation (Sacks et al., 1974, p. 722), so turn-taking and emergent turn-taking are of direct interest to emergent-communication simulations. Because the emergence of turn-taking is important to emergent communication, we frame Chapter 3 in the context of emergent communications. Some researchers have studied emergent turn-taking in its own right (Di Paolo, 2000; Iizuka and Ikegami, 2004), while others assume turn-taking while studying other aspects of language (Goldman et al., 2007) or ignore turn-taking entirely in their emergent-communication studies.

Emergent turn-taking bears some relation to cognitive radio and opportunistic spectrum access in wireless networks (Mitola and Maguire, 1999). We could use emergent turn-taking for medium access control in self-organising networks of heterogeneous wireless systems. Researchers have already modelled interfering wireless networks

as multi-agent cooperative or competitive games (Larsson et al., 2009; Leshem and Zehavi, 2009; Yang et al., 2009); a kind of turn-taking is the optimal solution in some cases (Larsson et al., 2009, p. 21). We can also think of medium access control in computer networks as a kind of designed turn-taking. Some stochastic medium access control methods such as slotted ALOHA display a degree of emergent turn-taking behaviour (Tanenbaum, 2002). For more on this, see the discussion in Chapter 3 regarding the quantity of turn-taking present in groups of random agents.

Some psychological studies suggest that turn-taking may be linked to musical rhythm in the human mind. Rhythmic perception and control are essential in both language and music; perhaps people even use the same neural circuitry for musical rhythm and speech prosody (Juslin and Laukka, 2003). Furthermore, Wilson and Wilson (2005) believe that their oscillator model for turn-taking could also apply to musical performance:

The model [for turn-taking in conversation] described here is intended as a more detailed specification of that proposed by Sacks et al. (1974). . . . We note the further possibility that other activities that are complex, rhythmic, and interpersonally coordinated—notably, music performance—may be governed by similar principles (see Jungers et al., 2002; Large and Jones, 1999; Large and Palmer, 2002). (Wilson and Wilson, 2005, p. 966)

Consequently, we can describe our search for the emergence of turn-taking and other sequential behaviours as working toward an artificial model for the part of the human mind that governs turn-taking in conversation and rhythmic coordination in musical performance. By developing a metric for turn-taking and techniques for analysing multi-agent systems that may or may not exhibit turn-taking behaviour, we pave the way for research on more complicated sequential joint agent behaviours such as musical rhythms.

2.4 Summary

The research areas and concepts that frame this dissertation are:

- Markov chains, Markov decision processes and reinforcement learning;
- multi-agent systems, normal form games, Nash equilibria, stochastic games and multi-agent reinforcement learning;
- and emergent phenomena, especially emergent communication and emergent turn-taking.

The thesis builds on these areas and we assume that the reader has knowledge of these concepts in later chapters. In particular, Chapter 5 makes great use of stochastic games and Nash equilibria. Chapter 4 uses Markov chains, reinforcement learning, Nash equilibria and emergent phenomena. Chapter 3 makes use of emergent phenomena, especially emergent turn-taking, and multi-agent systems.

Chapter 3

A Simple Metric for Turn-Taking in Emergent Communication

Turn-taking is a sequential joint agent behaviour. In Chapter 4, we demonstrate how to induce turn-taking in pairs of learning agents, but such a feat requires us to know when agents are taking turns. Therefore, in this chapter, we devise a simple metric for turn-taking. In addition to being useful as a demonstration of measuring sequential joint behaviour in groups of learning agents, our turn-taking metric is useful in the area of emergent communication. We analyse the turn-taking produced by random agents; if turn-taking is a desirable behaviour in a stochastic multi-agent system, then we must be able to distinguish deliberate turn-taking among the agents from random behaviour that might result in some non-zero quantity of turn-taking. To validate and demonstrate our turn-taking metric, we reinterpret two previously reported cases of turn-taking (Di Paolo, 2000; Iizuka and Ikegami, 2004) and analyse a recorded human conversation. While this chapter focuses on turn-taking as a potentially desirable sequential joint multi-agent behaviour, the methodology we use could be applied to construct metrics for other kinds of sequential joint behaviours. In particular, we propose that metrics for multi-agent sequential behaviours should consider the time scale of a behaviour (called ‘resolution’ in this chapter), the behaviour of random agents and the differences between metric results and qualitative human judgements.

3.1 Introduction

‘Turn-taking’ can refer to a wide range of human social behaviours. Turn-taking in human speech has been studied extensively (Sacks et al., 1974; Wilson and Wilson, 2005; Stivers et al., 2009). In some contexts, turn-taking occurs in groups of machines (Di Paolo, 2000; Iizuka and Ikegami, 2004) or groups comprised of people and machines

Table 3.1: Two agents sharing a resource.

t	1	2	3	4
Agent 1	Tries to use resource	Does not try	Does not try	Tries
Agent 2	Does not try	Tries	Does not try	Tries

(Turunen et al., 2006; Raux and Eskenazi, 2008). Consider two agents that share a resource that cannot be allocated to more than one agent at once. Each agent can try to use the resource at each of a number of time steps, $t = 1, 2, 3, \dots$. Suppose the history of agents trying to use the shared resource is as in Table 3.1.

To what extent are agents 1 and 2 in Table 3.1 taking turns using the resource? Chao and Thomaz’s study of the interaction between an anthropomorphic robot and people lead them to a similar question: ‘What is an appropriate metric for good turn-taking?’ (Chao and Thomaz, 2010, p. 134). We propose to answer questions like this by introducing a metric for the quantity of turn-taking in groups of identical agents that share a single resource. We narrow our scope further by assuming a particular model of how the resource behaves when more than one agent tries to take a turn at one time. Our metric is a simple and intuitive one rather than a general one. However, we believe that our metric is useful in a variety of situations where it is possible for a group of agents to take turns perfectly, fail to take turns entirely, or have some behaviour that falls between those extremes. Situations like this occur in ‘emergent communication,’ where agents are learning to communicate. Agents learning to communicate may need to take turns to be successful but may display a range of other behaviours.

Frequently, groups of agents benefit from sharing information (Gmytrasiewicz and Durfee, 2001; Floreano et al., 2007) or at least having reliable access to common information (Amato et al., 2009). However, agents may not be innately endowed with the ability to communicate. Some agents can learn to communicate, either by learning a language common to the other agents (Grim et al., 2002; Smith, 2002; Goldman et al., 2007; Ampatzis et al., 2008) or by reusing their non-communication-specific abilities to create a communications channel (Quinn, 2001; Nolfi, 2005). Emergent communication refers to cases where the ability to communicate emerges in a group of adaptive agents that originally could not communicate. Wagner et al. (2003) and Nolfi (2005) survey previous research in emergent communication and discuss the conditions required for the emergence of communication.

Sometimes communication is over an interfering channel; turn-taking is one way of defeating interference. People everywhere use turn-taking (Stivers et al., 2009) to defeat the difficulties in understanding another speaker while they themselves are talking. These difficulties relate to attention, short-term memory (Baddeley and Logie,

1999; Baddeley, 2000) and the limitations of language processing (Christoffels, 2006). Furthermore, turn-taking shapes human languages through its role in conversation:

It seems productive to assume that, given conversation as a major, if not THE major, locus of a language's use, other aspects of language structure will be designed for conversational use and, *pari passu* [Latin: equally], for turn-taking contingencies (Sacks et al., 1974, p. 722).

The origins and characteristics of human language are the starting point of many emergent-communication studies (Cangelosi and Parisi, 2002). Realistic models of the emergence of communication should therefore include turn-taking.

Some emergent-communication studies assume that the group of agents already possesses the ability to coordinate turn-taking (Goldman et al., 2007, Definition 19). Other studies have specifically examined the emergence of turn-taking (Di Paolo, 2000; Iizuka and Ikegami, 2004). One application of emergent turn-taking is medium access control for self-organising networks of heterogeneous wireless embedded systems. Sometimes researchers model interfering wireless networks as agents engaged in cooperative or competitive games (Larsson et al., 2009; Leshem and Zehavi, 2009; Yang et al., 2009). The optimal solution to some of those games is a kind of turn-taking where agents transmit one at a time (Larsson et al., 2009, p. 21). Our focus in this chapter is on describing agents' behaviours rather than controlling those behaviours. A quantitative measure of turn-taking is required to facilitate further research in emergent turn-taking. Quantifying turn-taking facilitates algorithmic optimisation and machine learning of turn-taking behaviour.

Our contributions to the field of emergent communication are:

- We propose $\tau\tau$, a quantitative measure for turn-taking that integrates the concepts of fairness and efficiency. See Section 3.2.
- We derive the turn-taking performance of random agents in Section 3.3. We consider how to evaluate distributions and individual values of $\tau\tau$ against the performance of random agents.
- We demonstrate the suitability of our turn-taking metric by re-interpreting two previously reported cases of turn-taking in simulated agents in Section 3.4 (Di Paolo, 2000) and Section 3.5 (Iizuka and Ikegami, 2004). We find that our computed values of $\tau\tau$ are consistent with the conclusions drawn by Di Paolo, and Iizuka and Ikegami. We present an application of our metric in Section 3.6 where we analyse the turn-taking behaviours present in a recorded human conversation.
- We have released an open source software implementation of $\tau\tau$, which we describe in Section 3.7.

We explore related work in Section 3.9. We discuss future work on alternative metrics and possible extensions to $\tau\tau$ in Section 3.8 and draw conclusions in Section 3.10.

3.2 A simple turn-taking metric, $\tau\tau$

We propose a simple metric for turn-taking in groups of identical agents that share a limited resource. Our metric transforms the agents' actions into a single real number representing the quantity of turn-taking present. We consider the attempts of each agent to use the shared resource to compute a real-valued turn-taking allocation for each agent, given a particular range of times. The turn-taking metric, $\tau\tau(t, r)$, is the product of the fairness and efficiency of the agents' allocation. The parameters t and r represent the time that the turn-taking value refers to and the time scale, or 'resolution,' of the turn-taking measurement, respectively. Turn-taking behaviour may or may not be valuable to the agents. In general, our turn-taking metric is not inherently related to the agents' utilities but turn-taking is a valuable system behaviour in a number of emergent-communication studies (Di Paolo, 2000; Iizuka and Ikegami, 2004).

We assume all sequences are discrete in time and that all agents in the group simultaneously select actions to take for a particular time step, t . Some of those actions may involve trying to use the shared resource. We summarise agent n 's actions with a 'usage attempt sequence', defined as $A_n(t) \in \{0, 1\}$, where $A_n(t) = 1$ indicates that agent n is attempting to use the shared resource at time t , and $A_n(t) = 0$ indicates that agent n is not attempting to use the shared resource.

3.2.1 Resource allocation model

The resource allocation model provides a set of rules for allocating the shared resource. We need a rule, for example, if more than one agent tries to use the resource at a particular time (a collision). If none of the agents attempt to use the resource at a particular time, then the resource is unallocated. When only one agent makes a resource usage attempt, that agent gets the resource for that particular time. If all the agents could use the resource on each time step, then sharing the resource through turn-taking would be unnecessary. Resource collisions must give a lower value of the turn-taking measure than alternating single uses so as to match intuition. For simplicity, we stipulate that if more than one agent tries to use the resource at once, then no agent gets the resource. This way, the turn-taking of a usage attempt sequence where every time step has a collision is the same as a usage attempt sequence where no agent ever makes a usage attempt. Table 3.2 summarises our resource allocation model.

Table 3.2: Resource allocation model summary.

Resource usage attempts	Allocations
No agent makes a resource usage attempt	No agent gets the resource
Only one agent makes a resource usage attempt	That agent gets the resource
More than one agent makes a usage attempt	No agent gets the resource

Each agent contributes to global turn-taking by taking its share of turns, that is, using the shared resource without collision on some time steps. The ratio of the number of time steps that an agent uses the shared resource without collisions to the resolution is its turn-taking ‘allocation.’ Within a set of agents, \mathcal{N} , we define agent n ’s turn-taking allocation over the time range t to $t + r - 1$ as:

$$\alpha_n(t, r) = \frac{1}{r} \sum_{t'=t}^{t+r-1} \left(A_n(t') \prod_{m \in \mathcal{N}, m \neq n} [1 - A_m(t')] \right) \quad (3.1)$$

where r and t are in \mathbb{Z} and $r > |\mathcal{N}|$. The expression $A_n(t') \prod_{m \in \mathcal{N}, m \neq n} [1 - A_m(t')]$ takes a value of 1 if agent n and only agent n is attempting to use the shared resource at time t' and equals 0 otherwise. Observe that $\alpha_n(t, r) \in [0, 1]$.

3.2.2 $\tau\tau$ is the product of fairness and usage efficiency

The allocation of the resource can only be ‘fair’ if all agents get access to it within a period of time corresponding to r . The resource usage fairness is therefore defined for a particular time, t , a particular resolution, r , and a set of agents \mathcal{N} as:

$$fairness(t, r) = \begin{cases} 0 & \text{if } \sum_{n \in \mathcal{N}} \alpha_n(t, r) = 0 \\ \frac{|\mathcal{N}| \min_n [\alpha_n(t, r)]}{\sum_{n \in \mathcal{N}} \alpha_n(t, r)} & \text{otherwise} \end{cases} \quad (3.2)$$

Thus we define fairness to be the normalised ratio of the allocation of the agent with the smallest allocation to the sum of the allocations for all agents, in the time range t to $t + r - 1$. For a truly fair situation, $fairness(t, r) = 1$; for an unfair system, when one agent misses out altogether (say), fairness is zero. In general, $fairness(t, r) \in [0, 1]$.

This definition of fairness reflects our intuitive linkage between the human concepts of taking turns and of fairness. Jain’s fairness index (Jain et al., 1984) is unsuitable in this case because it assigns non-zero fairness values in cases where one agent’s allocation is zero. Other fairness metrics without the problem in Jain’s metric are possible (Chen and Zhang, 2005; Lan et al., 2009). We discuss the possibility of using other fairness metrics in Section 3.10. Our chosen fairness metric matches our efficiency metric:

the numerator of the efficiency metric is the same as the denominator of the fairness metric to simplify the expression for the turn-taking metric.

As well as being fair, our intuition says that turn-taking must be ‘efficient.’ We define the usage efficiency to be the sum of the agents’ allocations:

$$\text{efficiency}(t, r) = \sum_{n \in \mathcal{N}} \alpha_n(t, r) \quad (3.3)$$

In effect, the expression for $\text{efficiency}(t, r)$ is the ratio of the number of steps with a single resource usage attempt to the total number of steps within the resolution. For example, if $A_n(t) = 1$ for all n and t , then $\text{efficiency}(t, r) = 0$; similarly if all $A_n(t) = 0$ then $\text{efficiency}(t, r) = 0$. On the other hand, if $A_n(t) = 1$ and $A_m(t) = 0$ for all $m \neq n$ and all t , then $\text{efficiency}(t, r) = 1$.

A different definition of efficiency could be more appropriate in some emergent-communication contexts. We could use the Pareto efficiency from economics (Musgrave and Musgrave, 1984; Brams, 2008), for example. We have chosen our definition to match our intuitive perception of turn-taking.

We define the degree of turn-taking, $\tau\tau$, as the product of the resource usage efficiency and the fairness:

$$\tau\tau(t, r) = \text{efficiency}(t, r) \times \text{fairness}(t, r) \quad (3.4)$$

$$= |\mathcal{N}| \min_n [\alpha_n(t, r)] \quad (3.5)$$

where $\tau\tau(t, r) \in [0, 1]$. When $\tau\tau(t, r) = 1$, there is perfect turn-taking between all the agents in the time range t to $t + r - 1$; in this case each agent makes the same number of resource usage attempts, no resource usage attempts collide and the agents use the resource on every time step. When $\tau\tau(t, r) = 0$, there is no turn-taking at all; in this case at least one agent has zero allocation. We can understand the values of $\tau\tau(t, r)$ in terms of fairness and efficiency, as in Eq. (3.4), or in terms of the allocation of the agent with the smallest allocation, as in Eq. (3.5). For example, if $\tau\tau(t, r) = 0.4$ then the agent with the fewest non-colliding resource usage attempts has 40% of the allocation it requires for perfect turn-taking.

3.2.3 Choosing a resolution

It is essential to choose an appropriate resolution, r , when calculating $\tau\tau(t, r)$. Perfect turn-taking is impossible if the resolution is less than the number of agents: we must choose $r \geq |\mathcal{N}|$. If the turn-taking exhibits a periodicity, then it would be reasonable to select r as equal to the period or an integer multiple of the period. In the periodic case, $\tau\tau(t, r)$ is precise only for turn-taking periods that are perfect divisions of r .

Considering an extreme case, suppose that $\mathcal{N} = \{1, 2\}$ with

$$\begin{aligned} A_1(t) &= 1, 0, 1, 0 \text{ and} \\ A_2(t) &= 0, 1, 0, 1 \text{ for } t = 1, 2, 3, 4. \end{aligned}$$

In this case $\tau\tau(1, 4) = 1$, but $\tau\tau(1, 3) = 2/3$. For all t and r , $\text{efficiency}(t, r) = 1$, because exactly one agent attempts to use the resource at each time step. Note that $\text{fairness}(1, 4) = 1$ because both agents get 2 turns each, but $\text{fairness}(1, 3) = 2 \min[2, 1] / (2 + 1) = 2/3$. In this case, since the turn-taking has a period of 2, $\tau\tau(t, 3)$ is a less accurate measure of the turn-taking than $\tau\tau(t, 4)$.

Consider perfect turn-taking with period L , $\tau\tau(t, L) = 1$ for all t . This implies $\tau\tau(t, r) = 1$ for $r = kL$, where $k \in \mathbb{N}$. However, $\tau\tau(t, r) = 1$ is not guaranteed for $r \neq kL$. If $r \bmod |\mathcal{N}| = 0$, then $|\mathcal{N}|^{|\mathcal{N}|/r}$ usage attempt sequences have perfect turn-taking, otherwise, no usage attempt sequences have perfect turn-taking. As r increases, the accuracy of $\tau\tau(t, r)$ increases because $\tau\tau(t, r)$ averages over more periods of the turn-taking so ending alignment has less effect. Also, with larger r , $\tau\tau(t, r)$ will more accurately report turn-taking with longer periods. However, as r increases, $\tau\tau(t, r)$ also becomes less specific to a particular time t . Thus we need to trade time localisation for accuracy.

3.3 Measuring the turn-taking of random agents

After performing a stochastic experiment with agents and measuring the turn-taking of the result, we have the question ‘What value of $\tau\tau$ is sufficient to indicate that turn-taking behaviour is present rather than some other behaviour that, by chance, looks like turn-taking?’ We suggest that designed or learnt behaviour should do better than random actions, otherwise there is no value in the design or learning. Specifically, agents should significantly exceed the mean $\tau\tau(t, r)$ of all possible probabilistic agents whose probabilities are independent of time and the other agents’ actions. We argue that it is sufficient to show that the agents being considered exceed the turn-taking of the probabilistic agents with the highest mean value of $\tau\tau(t, r)$, henceforth called the ‘best’ possible probabilistic agents. Specific values of $\tau\tau(t, r)$ from an experiment should be compared against the distribution of $\tau\tau(t, r)$ values from the best probabilistic agents.

For example, suppose there is a particular system that might display turn-taking for $\mathcal{N} = \{1, 2, 3\}$ and $r = 30$. If the experimenter samples the turn-taking of the system for 100 independent trials and the mean turn-taking value is 0.3 with a standard deviation of 0.1, can the experimenter say that turn-taking has arisen? We can calculate the

probability that such turn-taking occurred by chance if we know the distribution of $\tau\tau(t,30)$ for groups of 3 random agents with the best possible usage attempt probabilities. As we will calculate formally below, for $r = 30$ and $|\mathcal{N}| = 3$, the distribution has a mean of 0.27 and a standard deviation of 0.12. Our statistical null hypothesis is that the mean $\tau\tau(t,30)$ of the system is less than or equal to the mean $\tau\tau(t,30)$ of the random agents. We test this hypothesis using a one-sided t -test with unequal variances and sample sizes (Snedecor and Cochran, 1989). The t -test assumes normal distributions, but is robust to violations of this assumption when the sample sizes are large, as is the case here. We get a one-sided p -value of 0.03 indicating that there is a 3% chance that the system actually has a lower mean $\tau\tau(t,30)$ value than the random agents. So, at the 95% confidence level, we should reject the null hypothesis, and accept the alternative hypothesis that the system has a larger mean turn-taking than we could expect due to chance.

Suppose that we have a group of probabilistic agents, where we know the probability of agent n attempting to use the shared resource, $p[A_n(t) = 1]$, for all agents. Let these probabilities be independent of the other agents' actions and of time, so $p[A_n(t) = 1] = P_n$. Without loss of generality, let us assume that $\mathcal{N} = \{1, 2, \dots, N\}$ where N is the number of agents. We view the turn-taking value of these probabilistic agents as a random variable, denoted $TT(N, r, P_1, P_2, \dots, P_N)$. The distribution of $TT(N, r, P_1, P_2, \dots, P_N)$ depends on the resolution, r , the number of agents, N , and the probabilities, P_n for $n \in \mathcal{N}$. We can calculate this distribution from all possible usage attempt sequences at some resolution, or estimate it from a random sample of usage attempt sequences.

We define the best possible probabilistic agents as the ones with the largest expected turn-taking value, $E[TT(N, r, P_1, P_2, \dots, P_N)]$. Because $\tau\tau(t, r)$ assumes that all agents are identical, a unique maximum for $E[TT(N, r, P_1, P_2, \dots, P_N)]$ will occur only when $P_n = P$ for $n \in \mathcal{N}$. Based on the resource allocation model of Eq. (3.1), each agent's average allocation is maximised if each agent takes an average of one turn per N time steps. Therefore, the maximum of $E[TT(N, r, P_1, P_2, \dots, P_N)]$ is when $P = 1/N$. If $P_n = 1/N$ for all agents, then $TT(N, r, P_1, P_2, \dots, P_N)$ is a random variable representing the turn-taking of the best possible agents and will be henceforth referred to as $TT(N, r)_{best}$. Table 3.3 shows estimates of the means and standard deviations of $TT(N, r)_{best}$ for various numbers of agents at resolutions that are various multiples of N , estimated from 10^6 samples of different possible usage attempt sequences.

We can evaluate particular values of $\tau\tau(t, r)$ for some N and r against the distribution of $TT(N, r)_{best}$. If a group of agents takes turns with some value of $\tau\tau(t, r)$, then the probability that random agents would not achieve at least that same value, $p[TT(N, r)_{best} < \tau\tau(t, r)]$, is the probability that the group of agents is taking turns by some means other than chance. However, for very small resolutions, the large

Table 3.3: The mean, $E(TT_{best})$, and standard deviation, $\sigma(TT_{best})$, turn-taking values for the best probabilistic agents for various resolutions, r , and numbers of agents, N . These values are estimates computed from a population of 10^6 samples of the possible usage attempt sequences.

A		$r = N$	$r = 2N$	$r = 10N$	$r = 100N$	$r \rightarrow \infty$
2	$E(TT_{best})$	0.13	0.23	0.37	0.46	0.5
	$\sigma(TT_{best})$	0.33	0.27	0.14	0.045	0
3	$E(TT_{best})$	0.019	0.095	0.27	0.39	0.44
	$\sigma(TT_{best})$	0.14	0.20	0.12	0.041	0
4	$E(TT_{best})$	0.0029	0.042	0.22	0.36	0.42
	$\sigma(TT_{best})$	0.054	0.14	0.11	0.038	0
5	$E(TT_{best})$	0.00044	0.018	0.19	0.34	0.41
	$\sigma(TT_{best})$	0.021	0.094	0.098	0.036	0
10	$E(TT_{best})$	0.00	0.00034	0.12	0.29	0.39
	$\sigma(TT_{best})$	0.00	0.013	0.077	0.031	0
100	$E(TT_{best})$	0.00	0.00	0.0080	0.23	0.37
	$\sigma(TT_{best})$	0.00	0.00	0.027	0.022	0

variance in the distribution of $TT(N, r)_{best}$ means that only low confidence levels can be achieved. If there is a population of turn-taking values, one can compare that population with the distribution of $TT(N, r)_{best}$. We can evaluate turn-taking at any resolution if we have a large enough sample of turn-taking values, but using single values of $\tau\tau(t, r)$ is only meaningful with large resolutions.

We now derive the $\lim_{r \rightarrow \infty} E[TT(N, r)_{best}]$ as a function of the number of agents, N . The expected allocation for random agents as $r \rightarrow \infty$ is:

$$\lim_{r \rightarrow \infty} E[\alpha_n(t, r)] = P \prod_{m \in \mathcal{N}, m \neq n} (1 - P) \quad (3.6)$$

$$= P(1 - P)^{N-1} \quad (3.7)$$

The best random agents have usage attempt probabilities $P = 1/N$:

$$\lim_{r \rightarrow \infty} E[\alpha_n(t, r)_{best}] = \frac{1}{N} (1 - 1/N)^{N-1} \quad (3.8)$$

The expected value of $TT(N, r)_{best}$ can be calculated using Eq. (3.5) in the limit as $r \rightarrow \infty$:

$$\lim_{r \rightarrow \infty} E[TT(N, r)_{best}] = N \min_n \left(\lim_{r \rightarrow \infty} E[\alpha_n(t, r)_{best}] \right) \quad (3.9)$$

$$= N \left(\frac{1}{N} (1 - 1/N)^{N-1} \right) \quad (3.10)$$

$$= \left(\frac{N-1}{N} \right)^{N-1} \quad (3.11)$$

Eq. (3.11) is at its maximum of 0.5 at $N = 2$ and monotonically decreases with increasing numbers of agents, approaching $1/e$ in the limit:

$$\lim_{N \rightarrow \infty} \lim_{r \rightarrow \infty} E[TT(N, r)_{best}] \rightarrow 1/e \approx 0.37 \quad (3.12)$$

The trend towards this limit is evident in Table 3.3. Note that our formulation of $\lim_{r \rightarrow \infty} E[TT(N, r)_{best}]$ bears some similarity to limited-contention protocols for computer network medium access control (Tanenbaum, 2002, p. 262).

3.4 Application of the metric to work by Di Paolo

Di Paolo (2000) simulated pairs of mobile agents that used acoustic signalling to approach each other. Each agent had an acoustic sensor and a motor on each half of its round body, and a single acoustic transmitter. Recurrent neural networks controlled the agents. Di Paolo examined a number of aspects of his simulation. We only expand on the interpretation of the turn-taking behaviour of his agents' acoustic signalling; we add nothing to his other results. Each of Di Paolo's agents could detect the other agent's position based on the other's simulated acoustic signalling. Di Paolo found that the agents would activate their transmitters alternately at a constant frequency and that this alternating transmission helped them move better. The acoustic channel was additive and limited in range; one agent's signalling interfered with the other's signalling.

3.4.1 Defining the resource usage attempt sequences

For our purposes, Di Paolo's agents take turns using the shared resource of the simulated acoustic channel. We focus on Fig. 13 of Di Paolo's paper that shows the alternating signalling of his two agents. We extracted the numerical values from Di Paolo's graph to make Fig. 3.1. The graphical extraction process introduces small errors but does not significantly affect our results. The domain of Di Paolo's original graph is 200 units of time. We have quantised the domain to 2000 time steps across the graph, 10 time steps for each of Di Paolo's units. Henceforth, we abandon Di Paolo's original indexing of time that ranges between 400 and 600, and refer only to our new indices that range between 0 and 2000. The signalling in Fig. 3.1 occurs after the agents' learning has converged to a solution to their coordinated motion task. Di Paolo's simulation has two agents so $\mathcal{N} = \{1, 2\}$. The agent with a dashed line in Fig. 3.1 is agent 1 and the agent with the solid line is agent 2.

The turn-taking values depend strongly on how we define the usage attempt sequences; Di Paolo does not specify a way of defining the turn boundaries. Because we cannot

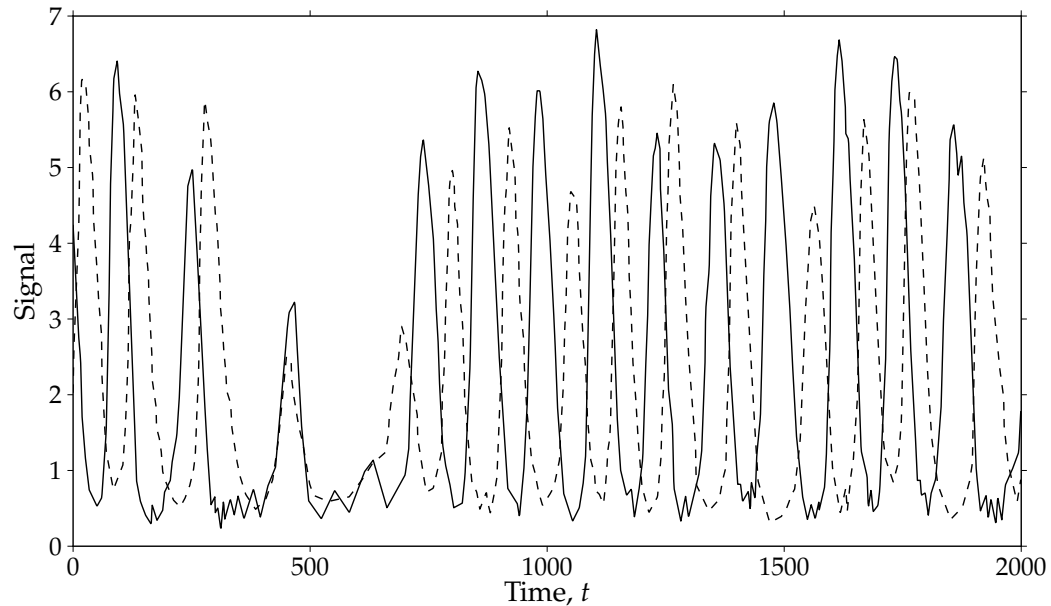


Figure 3.1: Turn-taking emerged in a simulation by Di Paolo. The solid line represents one agent's signalling and the dashed line represents the other's. We extracted the data for this graph from Fig. 13 of Di Paolo (2000).

assign specific meanings to signal features, we arbitrarily define the usage attempt sequences by thresholding the values in Fig. 3.1: for agent n , $A_n(t) = 1$ if $Signal_n(t) > 2$ and $A_n(t) = 0$ otherwise. Setting the threshold higher would result in fewer collisions but also less productive use of the channel. Fig. 3.2 shows the resulting resource allocation attempt sequences A_1 and A_2 plotted against time. Using a threshold is arbitrary and the value of the threshold is arbitrary, but the resulting usage attempt sequences serve to illustrate our turn-taking metric. Di Paolo observes that:

Since agents have no way of knowing of the presence of the other but through acoustic coupling an efficient way of doing this [completing their coordinated motion task] is by alternating the production of signals and so minimizing overlap. (Di Paolo, 2000)

Fig. 3.3 shows that the usage attempt sequences have some collisions; perhaps Di Paolo's agents try to minimise collisions but still some collisions are present.

3.4.2 Choosing a resolution

Fig. 3.3 shows that the two agents alternate their usage attempts much of the time, with small gaps and overlaps in between. Intuitively, we expect some turn-taking to be present. However, the resource is mostly unused in the range of time around

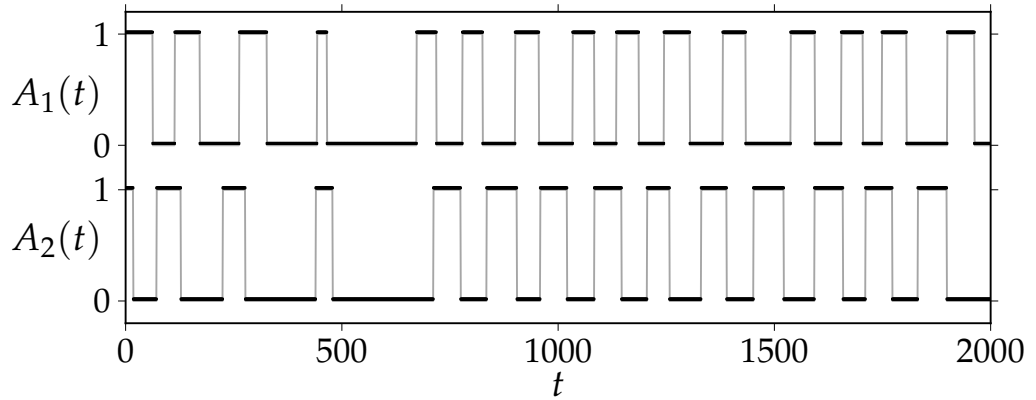


Figure 3.2: Two agents from Di Paolo (2000) share an acoustic resource. Their resource usage attempt sequences, A_1 and A_2 , are plotted against time using data from Fig. 3.1.

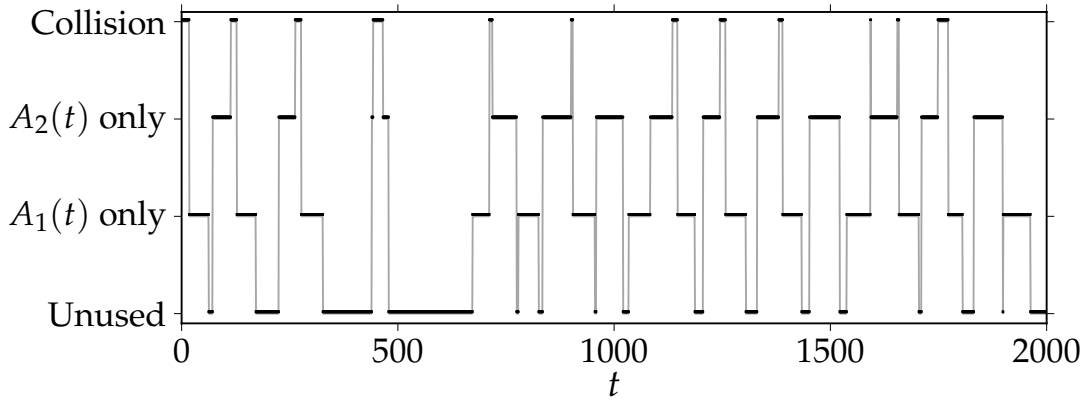


Figure 3.3: Analysis of the resource usage attempt sequences, A_1 and A_2 , from Fig. 3.2. Collisions occur when $A_1(t) = A_2(t) = 1$ and the channel remains unused when $A_1(t) = A_2(t) = 0$. Much of the time, however, only one of $A_1(t)$ or $A_2(t)$ equals 1.

$327 < t < 672$, aside from a collision around $t = 450$; this time range should get low turn-taking values. Our choice of resolution, r , should be small enough to discriminate this low turn-taking time range from the rest of the sequence.

The second half of Fig. 3.2 has 8 pulses of A_1 and 7 pulses of A_2 , giving about $7\frac{1}{2}$ periods of turn-taking in 1000 time steps. This suggests that $r = 133$ would be reasonable. Fig. 3.4 shows the mean turn-taking over time as a function of the resolution and illustrates the effects of changing the resolution described in Section 3.2.3: $\tau\tau(t, r)$ mostly increases with increasing r , but has ripples with local maxima at $r \approx r_0, 2r_0, 3r_0, \dots$ and local minima at $r \approx 1.5r_0, 2.5r_0, 3.5r_0, \dots$, where r_0 is an estimate of the period of turn-taking present in A_1 and A_2 . By inspection, $r_0 = 125$, closely matching our choice of $r = 133$.

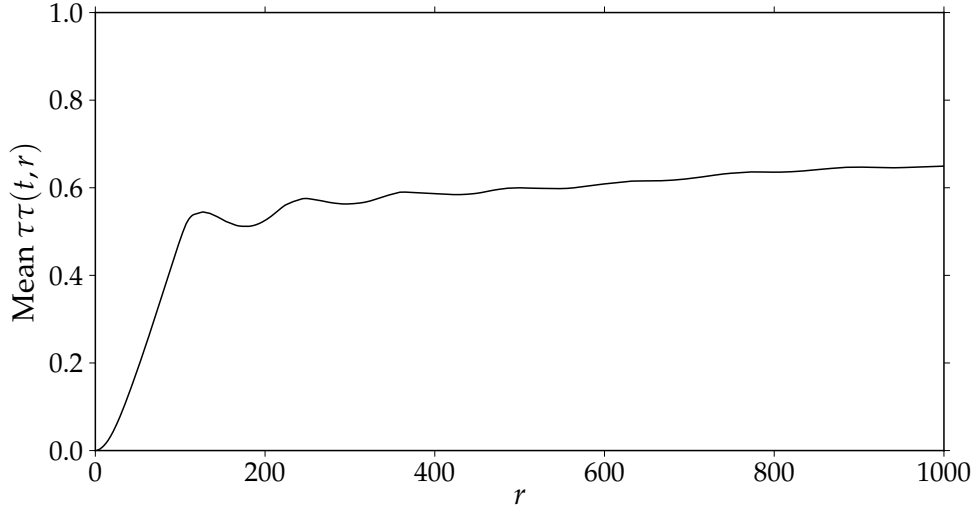


Figure 3.4: Mean turn-taking over time, $\tau\tau(t, r)$, generally increases with resolution, r , but has ripples with local maxima occurring near integer multiples of the period of turn-taking in the usage attempt sequences. The positions of these local maxima can be used to estimate the period of turn-taking.

3.4.3 Results

Fig. 3.5 shows the variation of $\tau\tau(t, 133)$ over time. The domain of Fig. 3.5 does not extend to the last 133 time steps; calculating $\tau\tau$ in the last 133 time steps would require values for the usage attempt sequences for times with $t > 2000$. Because we compute $\tau\tau(t, r)$ from the values of $A_n(t)$ from t to $t + r - 1$, $\tau\tau(t, r)$ refers to the turn-taking in the next r steps. The low turn-taking range at $327 < t < 672$ is correctly identified; $\tau\tau(t, 133) < 0.2$ for $250 < t < 599$. Because $\tau\tau(t, r)$ refers to the next r steps, it makes sense that the low turn-taking time range occurs earlier in $\tau\tau(t, r)$ than it does in $A_1(t)$ and $A_2(t)$. After $t = 599$, most of the graph exceeds 0.45, the maximum expected turn-taking at a resolution of 133 for pairs of random agents.

Table 3.4: Mean and standard deviation over time for $\tau\tau$, efficiency and fairness at $r = 133$.

Metric	Mean	Standard deviation
$\tau\tau(t, 133)$	0.54	0.29
$efficiency(t, 133)$	0.66	0.28
$fairness(t, 133)$	0.70	0.33

From Table 3.4, we can see that turn-taking is present on average over the entire sequence. Using the methods described in Section 3.3, we estimated the distribution of $TT(2, 133)_{best}$ with 10^6 samples. The probability that random agents will achieve a turn-taking value of at least 0.54 is 0.031. Therefore, for a confidence of 95%, turn-taking at a resolution of 133 is present on average over the sequence.

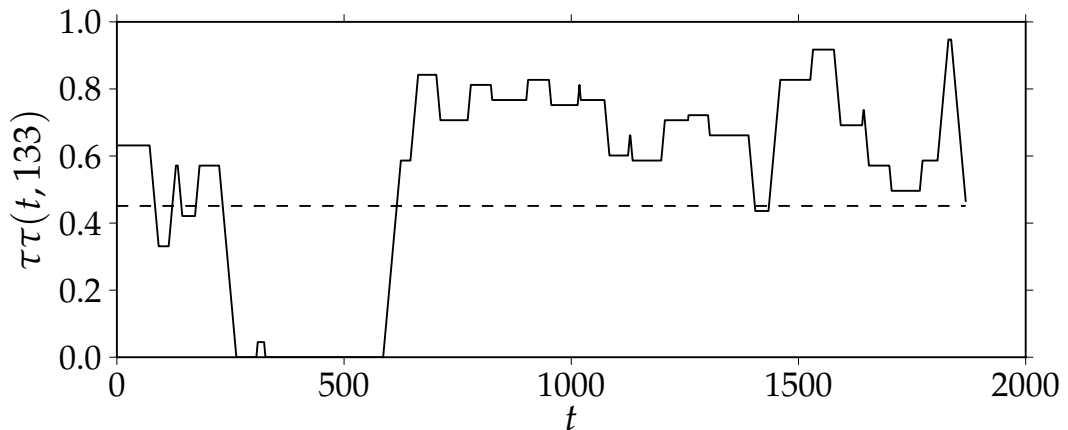


Figure 3.5: Turn-taking with $r = 133$, $\tau\tau(t, 133)$, varies with time. A period of low turn-taking is present at $277 < t < 580$. The expected turn-taking with a resolution of 133 for pairs of random agents, $E[TT(2, 133)_{best}]$, has a value of 0.45 and is plotted as a horizontal dashed line.

The large standard deviation of $\tau\tau(t, 133)$, 0.29, is due to the period of low turn-taking time range in the first half of the sequence; the mean of $\tau\tau(t, 133)$ in the second half of the sequence is 0.67, with a standard deviation of 0.12. In the time range where turn-taking does not seem to be present, $275 < t < 575$, the mean turn-taking is 0.003 indicating no turn-taking with probability $> 99\%$ as compared to random agents.

Di Paolo observed the anti-phase locking of the signalling in Fig. 3.1 and made the qualitative assessment that turn-taking was present in the signalling. He also showed that the internal activations of his agents' neural networks were strongly negatively correlated in periods of high qualitative turn-taking and only weakly negatively correlated in periods of low qualitative turn-taking (Di Paolo, 2000). Our turn-taking measure $\tau\tau(t, r)$ matches Di Paolo's qualitative judgement.

Our threshold value choice for Di Paolo's data is arbitrary; perhaps a different threshold would be better. Fig. 3.6 shows the mean turn-taking of Di Paolo's agents with a resolution of 133, $\tau\tau(t, 133)$, for different thresholds. The maximum of Fig. 3.6 is 0.55 at a threshold of 1.86. For our previously chosen threshold of 2, we have a mean turn-taking of 0.54 which is close to the maximum. Since our selected threshold is on a flat part of the graph, a slightly different threshold would produce similar results.

3.5 Application of the metric to work by Iizuka and Ikegami

Iizuka and Ikegami (2004) simulated pairs of agents that moved in a uniform, flat space and took turns chasing each other. Each agent had a circular body with a

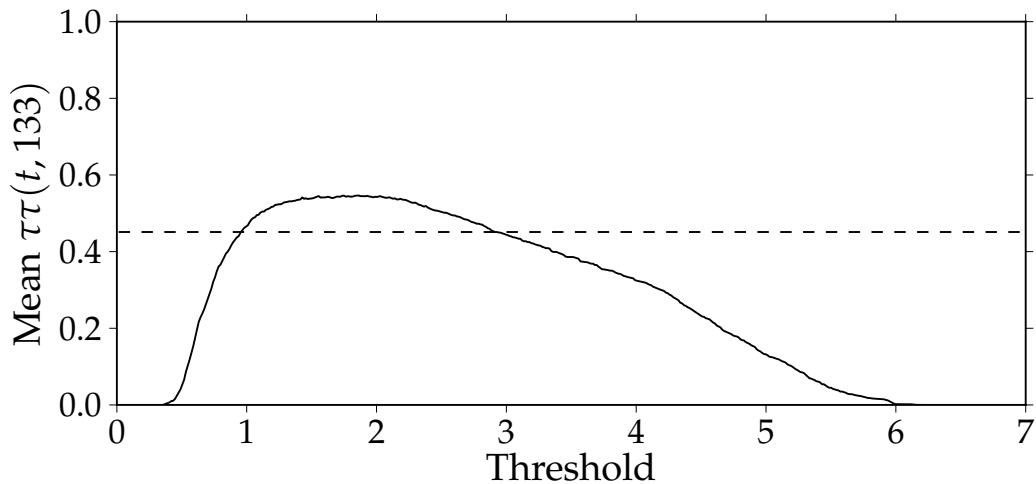


Figure 3.6: Mean turn-taking of Di Paolo’s agents at a resolution of 133, $\tau\tau(t, 133)$, as a function of the threshold for selecting $A_n(t)$ values. The expected turn-taking with a resolution of 133 for pairs of random agents, $E[TT(2, 133)_{best}]$, has a value of 0.45 and is plotted as a horizontal dashed line.

motor on either side and a recurrent neural network to control the motors. A genetic algorithm selected the weights in the agents’ neural networks; an agent’s fitness depended on how well it predicted its partner’s motion and how well it took turns chasing its partner. The fitness increased as Iizuka and Ikegami simulated more generations, so we expect greater turn-taking in later generations. Iizuka and Ikegami found that the agents successfully took turns chasing each other, moving regularly in earlier generations and chaotically in later generations.

Iizuka and Ikegami gave us unpublished data from the simulation they published in 2004, including the simulated motion paths and agent role sequences for three different pairs of agents from generations 3000, 5000 and 8000. For each generation, we define the set of agents as $\mathcal{N} = \{1, 2\}$. At each of 10^5 time steps in each generation, an agent can be the chaser, the evader or neither; an agent in the evader role is taking ‘its turn’ (Iizuka and Ikegami, 2004, p. 4). For the purposes of calculating $\tau\tau(t, r)$, these agent role sequences are the usage attempt sequences. Fig. 3.7 shows the simulated motion paths for one of the agents of each pair.

3.5.1 Choosing resolutions

For the agents from generation 3000, there is a definite periodicity of 61 to the usage attempt sequences. There are only 849 of 10^5 time steps (0.85%) where $A_1(t) \neq A_1(t + 61)$ or $A_2(t) \neq A_2(t + 61)$. This almost perfect periodicity is unsurprising, since Iizuka and Ikegami find that the turn-taking of agents from generation 3000

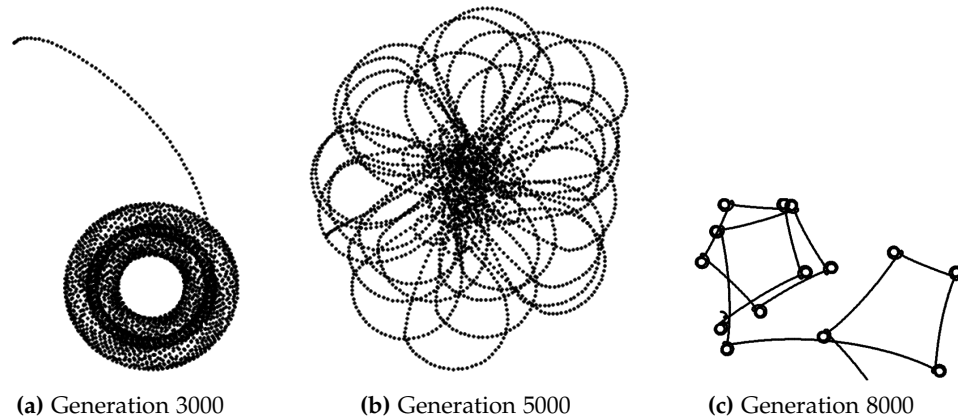


Figure 3.7: Each subfigure shows a motion trace for one agent of the pair from a particular generation of Iizuka and Ikegami’s simulation (Iizuka and Ikegami, 2004).

is ‘regular’ and immune to noise (Iizuka and Ikegami, 2004, p. 9). Fig. 3.7a shows that the agents’ locations lie on an annulus, apart from an initial trail at the top of the figure.

For the agents from generation 5000, there is a weak periodicity of around 95 in the usage attempt sequences. However, there are 0.10851×10^5 of 1×10^5 time steps (10.8%) where $A_1(t) \neq A_1(t + 95)$ or $A_2(t) \neq A_2(t + 95)$. As shown in Fig. 3.7b, the motion of the agents from generation 5000 seems to have a number of similarly shaped loops, but it lacks the definite regularity of the agents from generation 3000.

The agents from generation 8000 do not have periodic or semi-periodic usage attempt sequences because they are ‘chaotic’ turn-taking agents. ‘Beyond 6,000 generations, patterns change from regular to chaotic’ (Iizuka and Ikegami, 2004, p. 13). Fig. 3.7c shows the irregular motion of one of the agents from generation 8000. Because there is no discernible single period, we use the resolution selection method presented in Section 3.4.2 for Di Paolo’s experiment. There are 473 turn changes in 10^5 time steps, which suggests a resolution of $10^5/473 \approx 211$.

3.5.2 Results

Fig. 3.8 shows the first 1000 time steps of the usage attempt sequences and the values of $\tau\tau(t, r)$ for the three pairs of Iizuka and Ikegami’s agents from generations 3000, 5000 and 8000. The remainder of the sequences are similar to the second half of Fig. 3.8. The turn-taking is lower at the beginning of the sequences for all three generations. We believe this is because it takes a little while for the agents to start their turn-taking behaviour. The top of Fig. 3.7a shows a long trail from the start of the agents’ motion

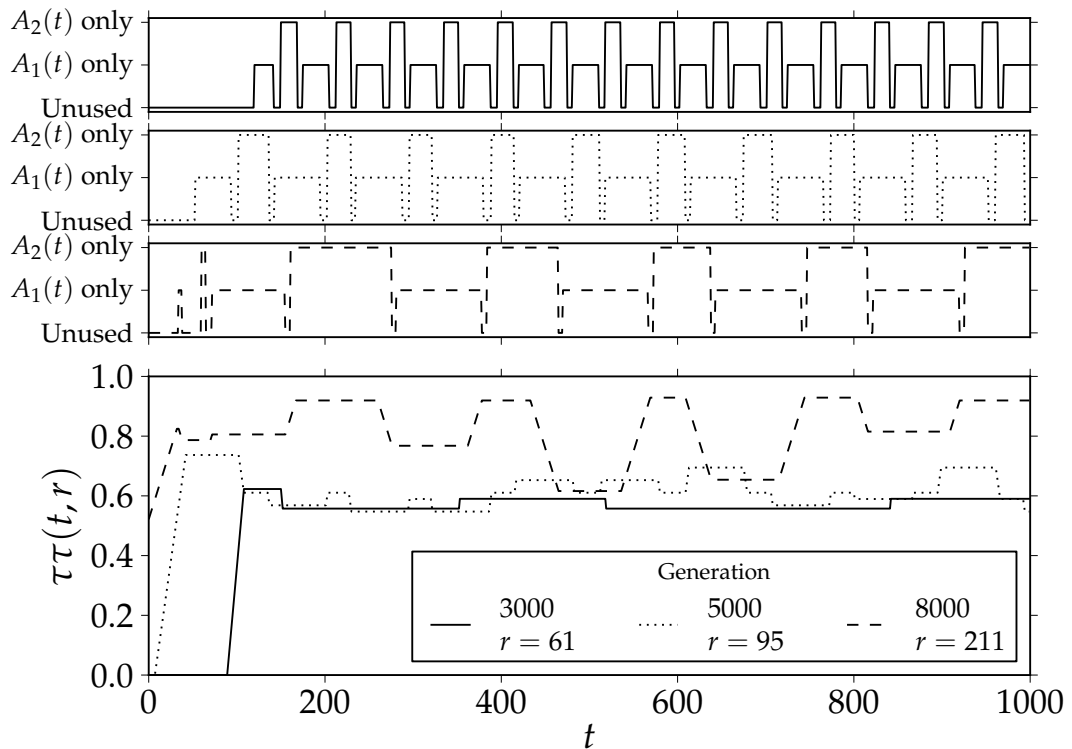


Figure 3.8: The usage attempt sequences and values of $\tau\tau(t, r)$ over time for Iizuka and Ikegami's agents from generations 3000 (solid line, $r = 61$), 5000 (dotted line, $r = 95$) and 8000 (dashed line, $r = 211$).

which leads into the regular motion; this corresponds to the low turn-taking at the start of the solid line in Fig. 3.8.

Table 3.5 compares the means and standard deviations of turn-taking values for the three different pairs of agents and for random agents at the same resolutions. The values for random agents were estimated from populations of 10^6 samples. All three generations have mean values that significantly exceed the expected turn-taking of random agents at the same resolution. With increasing generation, the turn-taking increases on average, but the spread and resolution also increase. This is consistent with the highly regular periodic behaviour of the agents in generation 3000, the less perfect periodic behaviour in generation 5000 and the chaotic behaviour in generation 8000. Not only is the mean turn-taking value of the agents from generation 8000 higher than those of the best probabilistic agents, but the standard deviation of the turn-taking is also much larger. Fig. 3.9 shows a box plot of the turn-taking values of the agents in the three different generations, illustrating the increasing median quantity of turn-taking as well as the increasing inter-quartile range.

Table 3.5: The means and standard deviations of $\tau\tau(t,r)$ over time from Iizuka and Ikegami (2004) and corresponding values for pairs of random agents at the same resolutions.

Generation	r	Iizuka and Ikegami data		Random agents	
		Mean $\tau\tau(t,r)$	Standard deviation $\tau\tau(t,r)$	$E(TT(2,r)_{best})$	$\sigma(TT(2,r)_{best})$
3000	61	0.57	0.023	0.43	0.081
5000	95	0.62	0.056	0.44	0.065
8000	211	0.76	0.19	0.46	0.044

The agents from generation 8000 only take turns in the presence of sensor noise (Iizuka and Ikegami, 2004, p. 6), but their behaviour is definitely unlike the turn-taking produced by any probabilistic agents whose probabilities are independent of time and the other agents' actions. This is unsurprising since the agents are coupled and have memories of the past. The erratic behaviour of the agents from generation 8000 makes it hard to get an intuitive grasp on how well the agents are taking turns. This highlights the strength of using a quantitative turn-taking metric. Our metric, $\tau\tau(t,r)$, identifies turn-taking even where intuition fails and our comparison to random agents reveals that the turn-taking behaviour of the agents does not arise by chance. The increased mean value of $\tau\tau(t,r)$ with increasing generation is unsurprising given that Iizuka and Ikegami designed their agents' fitness function to include a term for turn-taking behaviour. In Section 3.9, we compare $\tau\tau(t,r)$ in the context of Iizuka and Ikegami's experiment with their own turn-taking metric.

3.6 Application of the metric to a human conversation

We now apply the metric to the analysis of a recorded human conversation and we discuss how our quantitative results highlight social effects that may be otherwise overlooked. The behaviour of Di Paolo's simulated agents in Section 3.4 was amenable to intuitive analysis in his original paper and our quantitative results support his conclusions; in Section 3.5, Iizuka and Ikegami's agents behaved so as to maximise their fitness function which included an explicit term for turn-taking. On the other hand, a recorded conversation is more difficult to analyse: extracting the usage attempt sequence is more complicated, the duration of each turn varies greatly and turns do not occur regularly. Furthermore, the metric assumes that all agents are identical, as described in Section 3.2, but people are all different. While we expect turn-taking to be present in conversation in general (Sacks et al., 1974; Stivers et al., 2009), the research on turn-taking in sociology gives us no specific insights to the particular conversation we analyse here.

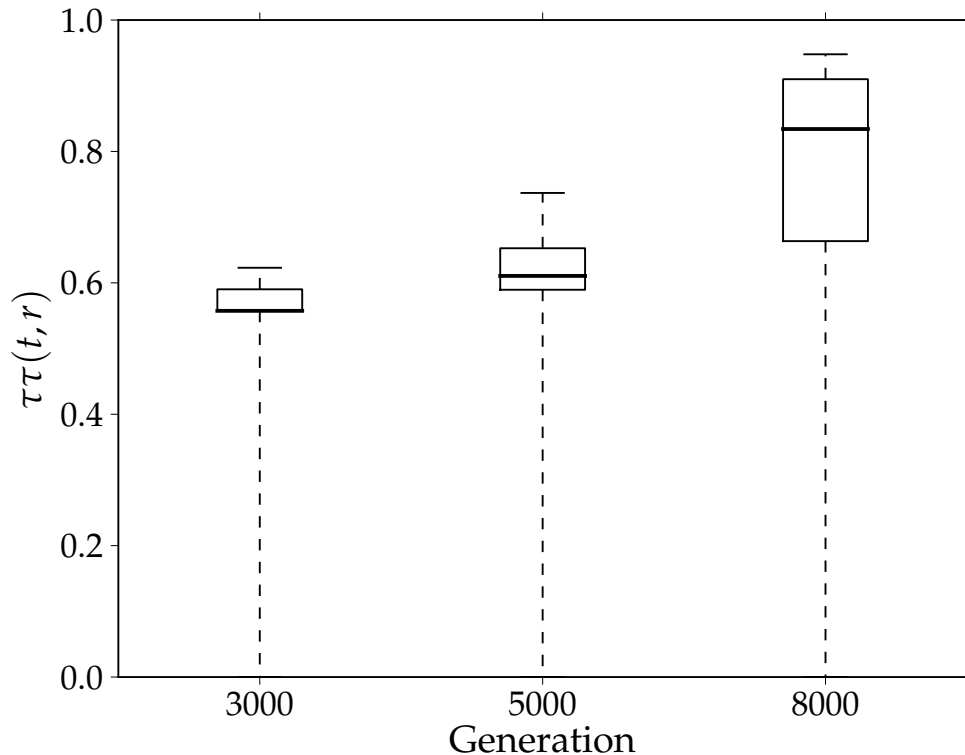


Figure 3.9: Box plot of the distributions of the turn-taking values of Iizuka and Ikegami’s agents over time. The bold centre line is the median, the box extends to the upper and lower quartiles and the whiskers span the range. For generations 3000, 5000 and 8000, $r = 61, 95$ and 211 respectively.

We measure the quantity of turn-taking present in a one minute excerpt from a corpus of speech called ‘CONversational Speech In Noisy Environments’ (COSINE). The COSINE corpus was created by Stupakov et al. (2009) to assist studies on automatic speech recognition and related topics. The corpus includes 33 conversation sessions with two to seven participants per session. Each participant wore an array of seven microphones and participated in natural conversations in noisy environments. Stupakov et al. (2009) and Stupakov et al. (2012) describe the data collection and transcription. We analyse a conversation from Session 2, starting 7 minutes 47.5 seconds into the session and ending at 8 minutes 47.5 seconds. Our source data are the speech recordings from the close-talking microphone on each person’s array, which has the least noise and the best isolation between speakers. Four speakers, two male and two female young adults, introduce themselves to each other and converse in fluent English. The set of agents is $\mathcal{N} = \{F1, M2, F3, M4\}$ so that each agent index also indicates the gender of the speaker.

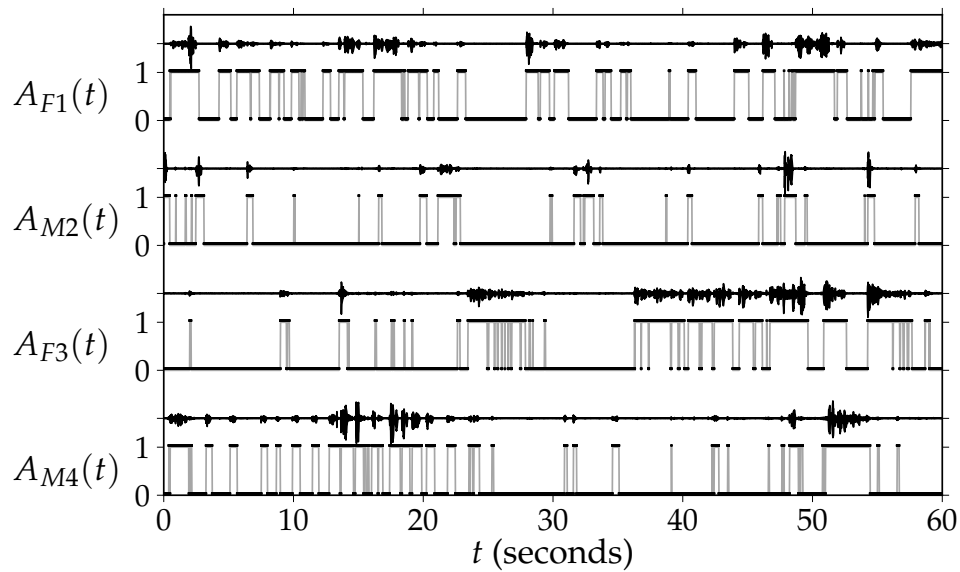


Figure 3.10: Sound amplitude waveforms and the extracted usage attempt sequences for the four speakers for the chosen excerpt from the COSINE corpus (Stupakov et al., 2009).

3.6.1 Defining the resource usage attempt sequences

As in Section 3.4, we must extract usage attempt sequences from continuous-valued signals. We want the usage attempt sequence to represent the times when each speaker is talking. Transforming a speech signal into a binary valued signal indicating when the speaker is talking is the well-studied problem of voice activity detection; for one example algorithm, see Sohn et al. (1999). We use a simpler approach: each audio signal is processed with a one-pole high pass filter with a cut-off frequency of 478 Hz followed by an envelope follower. When the envelope of the audio signal exceeds a hand selected threshold, we deem that agent to be speaking. We hand select a different threshold for each speaker to provide a balance between rejecting noise and failing to recognise speech. The resulting usage attempt sequences have some noise, but have sufficient quality for our purposes. The original audio is at 44100 Hz, but we down-sample the usage attempt sequences to 100 Hz on the assumption that a person will talk for at least 10 ms at a time and also to decrease the required computational time. Fig. 3.10 shows the waveforms of the four speakers after the high pass filter preprocessing stage and the resulting usage attempt sequences. Analysing the turn-taking in Fig. 3.10 would be difficult without a turn-taking metric.

3.6.2 Choosing a resolution

Fig. 3.10 shows that the usage attempt sequences do not form an obvious pattern. However, agent *F3* talks while the other agents are not talking for a time between about 22.5 seconds and 27.5 seconds. We expect low turn-taking for this time period while it is possible that turn-taking is present at other times. In order to discriminate this 5 second period of no turn-taking, we must use a resolution of no more than 5 seconds. On the other hand, smaller windows suffer from sensitivity to ending alignment as discussed in Section 3.2.3. With this balance in mind, we chose a resolution of 500 ten-millisecond time steps for a total window size of 5 seconds.

3.6.3 Results

Fig. 3.11 shows the calculated turn-taking at a resolution of 5 seconds as a function of time. Only on two occasions does the turn-taking of the four people exceed $E[TT(4, 500)_{best}]$, the expected turn-taking of groups of four random agents. We might find the overall low quantity of turn-taking surprising for a human conversation. Sacks et al. (1974) found that some form of turn-taking is a usual feature of conversation and indeed the four speakers in our conversation follow Sacks et al.'s turn trade rules (Sacks et al., 1974, p. 704). However, Sacks et al. observed that in unstructured conversation, 'Relative distribution of turns is not specified in advance' (Sacks et al., 1974, p. 701), whereas our turn-taking metric requires a fair distribution of speaking time between each agent. The lack of fair and efficient turn-taking makes sense if we ignore the content of the conversation and focus solely on who is talking when. The conversation has a total of 6.83 seconds when no one is speaking, while a single person is speaking for a total of 33.87 seconds and at least two people are speaking at once for a total of 19.3 seconds. The mean efficiency at a resolution of 5 seconds is 0.57. However, the dominant contribution to the low turn-taking values is fairness: the mean fairness at a resolution of 5 seconds is only 0.045. We consider the possibility of alternative turn-taking metrics that use alternative fairness metrics in Section 3.8.

The results of our turn-taking metric allow us to quantify social effects. The low average value of the fairness metric reflects the disparity between agents' time speaking. Our metric assumes that all the agents are identical (see Section 3.2). The four participants in the study are socially equal: their 2008 American culture has no significant gender biases in this context, they are similar ages (19, 19, 20 and 22) and they all received the same instructions from the researchers. The differences in the agents stem from other factors, such as personality or interest in the topics of conversation. Our turn-taking metric highlights differences like these and can motivate further examination. Agent *M2* speaks significantly less than the other agents, as shown in

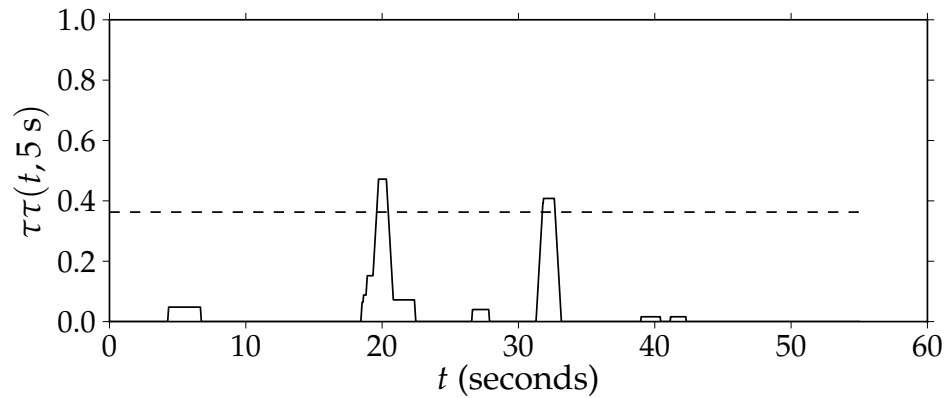


Figure 3.11: Turn-taking at a resolution of 5 seconds (500 time steps of 10 ms) plotted against time for a 60-second conversation between four young adults. The expected turn-taking at a resolution of 500 for groups of four random agents, $E[TT(4, 500)_{best}]$, has a value of 0.36 and is plotted as a horizontal dashed line.

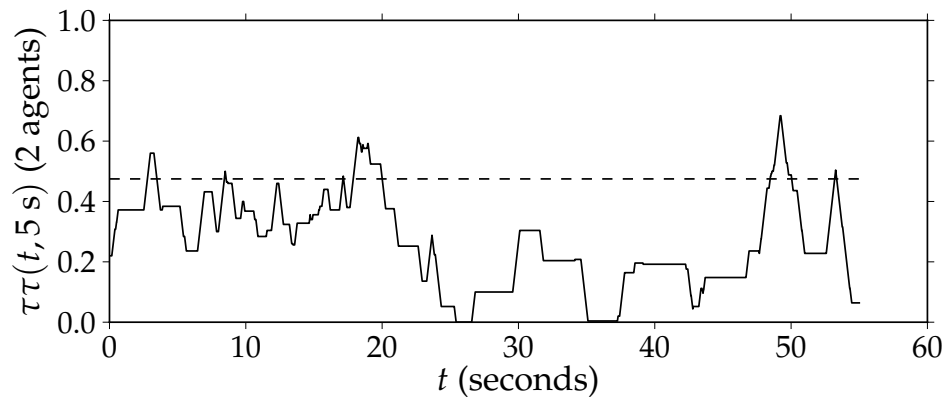


Figure 3.12: Turn-taking at a resolution of 5 seconds (500 time steps of 10 ms) plotted against time, for a human conversation considering only agents *F1* and *M4*. The expected turn-taking at a resolution of 500 for pairs of random agents, $E[TT(2, 500)_{best}]$, has a value of 0.47 and is plotted as a horizontal dashed line.

Fig. 3.10. The only two short periods where $\tau\tau(t, 500) > E[TT(4, 500)_{best}]$ in Fig. 3.11 coincide with the times that agent *M2* speaks without another agent speaking over him (see Fig. 3.10).

Agents *F1* and *M4* dominate the first half of the conversation and agents *F1* and *F3* dominate the second half of the conversation. If we consider the conversation to be between agents *F1* and *M4* only, then there is considerably more turn-taking at a resolution of 5 seconds at most times, as shown in Fig. 3.12. The quantity of turn-taking for agents *F1* and *M4* is still generally below $E[TT(2, 500)_{best}]$; this is unsurprising since there are many times when agents *F1* and *M4* stop speaking to listen to agents *M2* and *F3*.

3.7 Open source software implementation

In order to stimulate research in emergent turn-taking, we have released ‘Turn-Taking Measurement Tools,’ an open source Python (Lutz, 2006) software library that implements $\tau\tau$. To increase the library’s adoption in academia and industry, the library is licensed under the permissive new BSD license (Open Source Initiative, 1999). Turn-Taking Measurement Tools includes metric functions and tools for estimating the turn-taking present in groups of random agents as well as automated self tests and thorough documentation. To help researchers and developers, the software package also includes interactive examples, one of which is a simplified copy of the analysis that we complete in Section 3.6. Turn-Taking Measurement Tools is available online at: <http://code.google.com/p/turn-taking-measurement-tools/>

3.8 Discussion

Observe that $\tau\tau(t, r)$ is not the only possible turn-taking metric and that one could make decisions other than those we have made for $\tau\tau(t, r)$. In the following, we consider the limitations in applicability of $\tau\tau(t, r)$ and alternative decisions that would lead to other turn-taking metrics.

We assume that all agents are identical and unchanging with time. However, this will not always be the case. It is possible that agents may need the shared resource for different amounts of time, for example if one agent asks another a series of long questions with short answers. We could scale each agent’s allocation in Eq. (3.1) to account for differences in agents’ needs. Our choice of fairness metric only depends on the agent with the smallest allocation, but sometimes it is only important that some of the agents are participating in turn-taking. For example, if three people are in a room and two are conversing while the other is silent, how much turn-taking is present? Our metric resolves this question by focusing on the person who is silent: they take no turns, so the group has a turn-taking value of zero. For situations where we desire a different answer to that question, a different fairness metric would be appropriate.

Our resource allocation model is simplistic; others are definitely possible. The essential properties of a turn-taking resource allocation model are that leaving the resource unused results in low allocation and allocating to more than one agent at a time gives a lower allocation than allocating to a single agent at a time. In general, the cost and possibility of collisions will vary from situation to situation and the turn-taking metric may need to be modified. For example, the cost of a collision could be a function of the number of agents colliding at that time step, so that a collision between 100 out of

100 agents is worse than a collision between 2 of 100 agents. We could replace Eq. (3.1) on p. 35 and instead compute agent n 's turn-taking allocation as:

$$\alpha_n(t, r) = \frac{1}{r} \sum_{t'=t}^{t+r-1} A_n(t') \left(1 - \frac{\sum_{m \in \mathcal{N}, m \neq n} A_m(t')}{|\mathcal{N}| - 1} \right)^c \quad (3.13)$$

where the value of c controls the cost of collisions. Alternatively, collisions could be resolved by randomly assigning the shared resource to one of the agents that tried to use the shared resource or by deterministically assigning the shared resource according to some predefined priority scheme.

We assume binary usage attempts, but many situations involve continuous usage attempts. Alternative turn-taking metrics could represent agents trying to use only part of the shared resource, with $A_n(t)$ taking any real value between 0 and 1, rather than $A_n(t)$ equally either 0 or 1. Another possibility is to approximate a continuous resource with a binary usage attempt sequence, as in Section 3.4.1 where we use a threshold. We could have indexed $\alpha_n(t, r)$, $fairness(t, r)$, $efficiency(t, r)$ and $\tau\tau(t, r)$ as depending on $A_n(t')$ values for $t - r + 1 \leq t' \leq t$. However, our choice of indexing from t to $t + r - 1$ allows an allocation value to be calculated at $t = 0$ for finite usage attempt sequences starting at $t = 0$ without having to assume values of $A_n(t')$ for $t < 0$.

In general, turn-taking metrics must compute resource allocations given usage attempts, reflect the concepts of fairness and efficiency, and deal appropriately with issues of resolution. Our decisions leading to $\tau\tau(t, r)$ serve our goal of creating a simple metric for turn-taking but limit the scope of our metric's applicability. This limitation is intentional: while our methodology for measuring turn-taking is more general, we opt to open the discussion on measuring turn-taking with a specific, simple metric for groups of identical agents. We leave the possible extensions to $\tau\tau(t, r)$ and alternative metrics to future work.

3.9 Related work

The properties of our turn-taking metric are inspired by work in a number of existing fields, including emergent communication. Previously, authors measured emergent turn-taking intuitively, or with *ad hoc* methods such as qualitative examination of a graph or by finding negative correlation between two signals with similar power spectra (Di Paolo, 2000). Our turn-taking metric is quantitative and is only sensitive to turn-taking, unlike a correlation.

As described in Section 3.5, Iizuka and Ikegami (2004) simulated pairs of agents that moved in a uniform, flat space and took turns chasing each other. We compare $\tau\tau(t, r)$ with Iizuka and Ikegami's own turn-taking metric, F^{turn} :

$$\tau\tau(1, r) = \frac{2}{r} \min \left\{ \sum_{t=1}^r A_1(t)[1 - A_2(t)], \sum_{t=1}^r A_2(t)[1 - A_1(t)] \right\} \quad (3.14)$$

$$F^{turn} = \alpha_1 \alpha_2 \quad (3.15)$$

where $\alpha_1 = \sum_{t=1}^r A_1(t)$ and $\alpha_2 = \sum_{t=1}^r A_2(t)$, and $A_1(t) = 1$ if and only if the position of agent n is in a circular sector behind the other agent and $A_1(t) = 0$ otherwise (Iizuka and Ikegami, 2004, Eqs. 8 and 9). (Iizuka and Ikegami use different notation, with $g_a(t)$ and $g_b(t)$ in place of $A_1(t)$ and $A_2(t)$ and with T in place of r .) One key difference between the two metrics is that F^{turn} uses a resource allocation model that does not allow for the possibility of both agents attempting to use the shared resource at the same time (Iizuka and Ikegami, 2004, p. 3). If $A_1(t) = 1 \iff A_2(t) = 0$ and $A_2(t) = 1 \iff A_1(t) = 0$ then $\tau\tau(1, r) = \frac{2}{r} \min\{\alpha_1, \alpha_2\}$. Another fundamental difference is that large $\tau\tau(t, r)$ values require fairness through the $\min\{\alpha_1, \alpha_2\}$ term for all values of α_1 and α_2 . F^{turn} does not require fairness in the same way. The maximum of F^{turn} is when $\alpha_1 = \alpha_2 = r/2$, which is a fair allocation between the two agents. However, at other points in its range, F^{turn} increases with both α_1 and α_2 ; $dF^{turn}/d\alpha_1 = \alpha_2 \geq 0$ and $dF^{turn}/d\alpha_2 = \alpha_1 \geq 0$. This means that highly unfair allocations can produce large F^{turn} values. The gradient of F^{turn} could be a strength in the context of Iizuka and Ikegami (2004) if it helps the genetic algorithm find the maximum but we consider that requiring fairness is essential for a turn-taking metric. Iizuka and Ikegami do not specify a way to generalise their metric to more than two agents.

Neill (2003) describes the 'turn-taking dilemma,' a variant on the prisoners' dilemma (Flood, 1958) where the optimal strategy is for the players to alternate between defecting and cooperating. This situation appears in the natural behaviour of groups of dwarf mongoose (Rasa, 1989). Neill uses a concept of 'evolutionary dominance' (Neill, 2001) to rank strategies with memories of length 1 and 2 in the presence of noise. Neill evaluates strategies that may or may not take turns rather than evaluating histories of actions as we have done in this paper. Because a turn-taking dilemma rewards turn-taking most highly, the best strategies are good at coordinating turn-taking with other good strategies. 'Since mutual cooperation and mutual defection result in suboptimal payoffs, a failure to coordinate one player's cooperation with the other player's defection results in harm to both players' (Neill, 2003, p. 243). Neill's concept of 'self-alternating' strategies relates to our intuition that turn-taking should be fair. However, the best of Neill's strategies exploit more naïve strategies, resulting in worse turn-taking. 'Since each player is expected to defect half the time, exploitation

occurs when one player takes more than his fair share of turns, harming the other player' (Neill, 2003, p. 243). Action records with high payoffs in a turn-taking dilemma game will also have a large value of $\tau\tau(t, r)$.

Lan et al. (2009) derive a family of fairness functions based on a set of axioms. Some of Lan et al.'s metrics also incorporate a notion of efficiency. The definition of fairness in Eq. (3.2) is related to a Lan et al. fairness measure, f_β , where β is a parameter that controls certain properties of the metric. We can write Eq. (3.2) as:

$$fairness(t, r) = \frac{N}{-f_\beta(\alpha_n(t, r), 1 \leq n \leq N)} \quad (3.16)$$

where $\beta \rightarrow \infty$ so that $f_\beta(\mathbf{x}) = -\max_i\{(\sum_i x_i)/x_i\}$. Lan et al. consider unbounded fairness functions so the transformation of Eq. (3.16) is necessary to limit the range of the fairness measure to $[0, 1]$. Using Lan et al.'s Eq. 5 for the weights would mean that the axiom of irrelevance of partition does not apply to our fairness measure. However, our fairness metric satisfies the continuity, homogeneity, asymptotic saturation, and monotonicity axioms. While our fairness metric bears some similarity to one of Lan et al.'s metrics, Lan et al. make no attempt to measure turn-taking specifically or to consider the fair allocation of a resource through time.

We have considered the fair and efficient allocation of a limited shared resource through time in the context of emergent communication. Many researchers have explored efficient, fair resource allocation in general. Brams (2008) examines a number of fair division problems from a political science perspective. Bouveret and Lang (2008) investigate the computational complexity of finding efficient allocations of indivisible resources that satisfy an 'envy-free' notion of fairness. However, neither Brams nor Bouveret and Lang considers the distribution of allocations through time as we have with our turn-taking measure.

Researchers have also devised ways of measuring the quality of turn-taking in human-human or human-machine speech. Turn-taking is present in conversation when speaker changes occur and the participants follow turn trading rules (Sacks et al., 1974). Measuring turn-taking is important in spoken dialog systems, such as automated telephone answering agents, because it contributes to user satisfaction. In spoken dialog systems, turn-taking quality depends on the lengths of silences between speech and the interruption rates (Raux and Eskenazi, 2008; Turunen et al., 2006). Measuring silences and interruptions is related to our efficiency metric, which forms part of our turn-taking metric. Efficient interaction is also important in human-robot interactions. Chao and Thomaz (2010) examine turn-taking between an anthropomorphic robot and a person as the person teaches the robot to sort coloured objects; Chao and Thomaz observe that good turn-taking has 'minimal over-lapping of turns as well as minimal time spent between turns, characteristics of an efficient interaction' (Chao and

Thomaz, 2010, p. 133). The key difference between human-human or human-machine turn-taking measures and our work is that we do not assume the presence of any kind of turn-taking *a priori*. Counting silences and interruptions is a good measure of deviations from normal human-generated turn-taking but it fails to distinguish monologues from conversations because fairness is not considered. Our metric classifies a wider range of possible agent interactions, making it better suited for use in emergent communication.

3.10 Conclusion

We propose $\tau\tau(t,r)$, a quantitative metric for the degree of turn-taking present in resource usage attempt sequences by identical agents. This metric allows evaluation of turn-taking in emergent-communication experiments. We define $\tau\tau(t,r)$ as the product of the allocation efficiency and fairness to reflect our intuitive understanding of turn-taking. The concept of resolution is essential to measuring turn-taking: turn-taking may be present on some time scales but not others. We suggest that intentional turn-taking behaviour should have a value of $\tau\tau(t,r)$ that exceeds the maximum expected turn-taking that would be produced by random agents. We give mean values and standard deviations for the distributions of $\tau\tau(t,r)$ produced by random agents and methods for comparing observed values and distributions of $\tau\tau(t,r)$ with those distributions from random agents. We show that our metric is useful in practice by applying it to simulations by Di Paolo (2000) and Iizuka and Ikegami (2004), and to a recorded human conversation taken from the COSINE corpus (Stupakov et al., 2009). Our new metric enables us to make quantitative measurements about the presence of turn-taking rather than informal qualitative judgements.

The results of this chapter support the thesis by reifying the concept of measurable multi-agent sequential behaviours. We suggest that our methodology in this chapter could be applied to construct metrics for desirable sequential multi-agent behaviours in general. Specifically, multi-agent sequential behaviour metrics should consider the effects of different resolutions and different numbers of agents as well as the performance of random agents and the relationship between metric results and qualitative human judgements. We provide in $\tau\tau$ a concrete example of how to measure a potentially desirable sequential multi-agent behaviour. In Chapter 4, we apply $\tau\tau$ to a simulated multi-agent context and demonstrate how to identify reward functions that result in turn-taking as a system behaviour. Therefore, this chapter also supports the thesis by building the requisite foundation for Chapter 4.

Chapter 4

Rewards for Pairs of Q-Learning Agents Conducive to Turn-Taking in Medium Access Games

In Chapter 3, we demonstrated a simple metric for turn-taking. In this chapter, we use that metric to explore a range of possible reward functions for pairs of Q-learning agents (Watkins, 1989) in a class of stateful games, which we call ‘medium access games.’ We show that some of these reward functions are conducive to turn-taking as a system behaviour, while others are prohibitive to turn-taking. This chapter supports the thesis by showing that appropriate reward functions can influence the sequential joint behaviour of pairs of Q-learning agents toward the desired behaviour, turn-taking, which can be measured by the metric $\tau\tau$.

Medium access games are a simplified model of the interactive aspect of human and machine communication. Our goal is to develop predictors that allow us to anticipate the presence of turn-taking behaviour by the Q-learning agents in simulated medium access games by analysing the agents’ reward functions. We use the Nash equilibria of a medium access game as played by pairs of agents with stationary policies as the predictor for turn-taking in Q-learning agents. We present simulation results for an extensive range of reward functions for pairs of Q-learners and we use our Nash equilibria techniques to develop predictors for the presence and absence of turn-taking.

While our focus is on designing multi-agent reinforcement learning systems that produce intentional turn-taking, our results may also be useful for analysing emergent turn-taking behaviour. Therefore, our research expands the space of designed behaviours by encroaching on the domain of emergent turn-taking. We propose ways to apply our methodology to design reward functions for quantifiable sequential joint

behaviours besides turn-taking. In Chapter 5, we develop this chapter's methodology further, into a general method for designing rewards for stochastic games.

4.1 Introduction

How can we design incentive systems to induce coordinated turn-taking behaviour between adaptive agents in multi-agent systems? Learning agents in multi-agent systems face the challenges of interaction with the other agents as well as the intrinsic challenges of learning. Frequently, the designers of a multi-agent system desire a particular overall emergent system behaviour without knowing the exact dynamics of the agents' interaction. For example, a team of robots playing soccer (Kitano et al., 1997) tries to win the game, but the team members' interactions are highly dependent on the opposing team's strategy and the erratic movements of the ball. Multi-agent reinforcement learning (Buşoniu et al., 2008) offers a promising methodology for controlling multi-agent systems, but leaves designers with the problem of choosing agents' rewards so as to induce coordinated behaviour. We cannot assume coordination amongst cooperative, decentralised agents, especially when the agents must learn communication. Emergent coordination is also a desirable outcome in a number of purely competitive multi-agent situations, such as the Prisoners' Dilemma (Flood, 1958) and the Tragedy of the Commons (Hardin, 1968). We focus on inducing agents to exhibit a particular type of coordination: turn-taking.

We present a class of multi-agent medium access games as a framework for exploring the emergence of turn-taking in simulated Q-learning agents (Watkins, 1989). Our medium access game model is inspired by research on medium access control in wired and wireless computer networks (Tanenbaum, 2002). We simulate Q-learning agents with a wide range of reward functions playing medium access games. Some methods of rewarding the agents result in turn-taking as a system behaviour. We measure the degree of turn-taking using the metric that we develop and demonstrate in Chapter 3 (see also Raffensperger et al., 2012b).

Our goal is to provide researchers in emergent communication with ways to predict the emergence of turn-taking in multi-agent simulations. Rather than studying emergent behaviour directly, we focus on the 'design' agenda of multi-agent reinforcement learning proposed by Gordon (2007) and suggest a way to produce coordinated turn-taking in a group of agents. Therefore, the examples of turn-taking presented here are designed in the sense that we are explicitly looking for turn-taking behaviour. However, global turn-taking behaviour occurs through the synergy of agents acting with only local information, so our results are useful for analysing emergent turn-

taking. Our contributions to the multi-agent reinforcement learning approach to turn-taking are:

- We formally define a class of multi-agent games that are representative of medium access control problems in shared communication mediums. (See Section 4.2.)
- We identify the deterministic policies for agents in medium access games that produce turn-taking as a system behaviour. We use Markov chains to derive the expected payoff for stationary stochastic policy agents, which enables computation of the Nash equilibria of medium access games. (See Section 4.3.)
- We present simulation results for the emergence of turn-taking in pairs of Q-learning agents with a range of reward functions and we analyse these results using the Nash equilibria of the corresponding medium access games. We discuss how to analyse a reward function to predict the emergence of turn-taking in Q-learning agents and how to design reward functions that are either conducive or prohibitive to the emergence of turn-taking. (See Section 4.4.)
- We discuss possible extensions to our model in Section 4.5 and we give suggestions on how to apply our methodology to design rewards for coordinated multi-agent behaviours other than turn-taking.

We summarise the relevant literature in Section 4.6 and we draw conclusions in Section 4.7.

4.2 Simulated medium access games

We consider pairs of agents that share an interfering communication medium, as in Fig. 4.1. The agents have some need to transmit messages on the channel, but the interfering nature of the channel prohibits agents from successfully transmitting all at once. The agents are rewarded based on the outcome, so we describe this situation as a medium access game. The agents' interests in a medium access game may be conflicted or the agents may be incentivised to cooperate.

In our model, each agent can transmit a message on the channel at each discrete time step, $t = 1, 2, 3, \dots$. The communications channel is interfering such that an agent's message will be corrupted if another agent also transmits on the same time step. However, we assume that the agents cannot tell whether or not a message has been corrupted by a 'collision' with another message. This interfering communication channel design is a model of a variety of situations where turn-taking may be useful. For example, in half-duplex telephone conversations, a person cannot hear the other

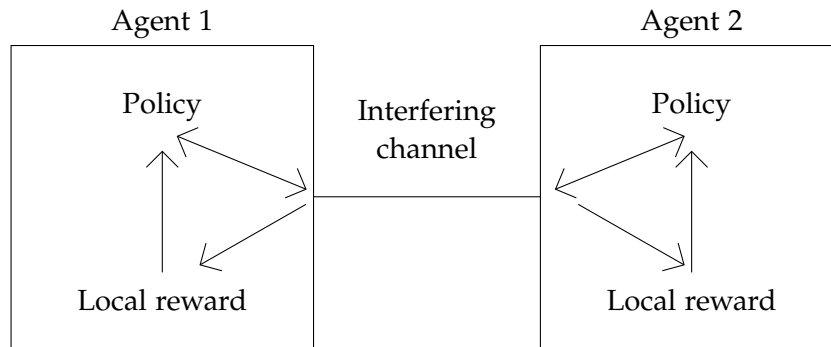


Figure 4.1: Two learning agents connected through an interfering communication channel.

speaker while he or she is speaking because of the limitations of the channel and the first speaker will not immediately know if the other person tried to speak.

The agents must coordinate their transmissions to defeat the channel's interference, if they are to successfully communicate. One way to coordinate is to use previously known medium access control protocols such as ALOHA or Ethernet (Tanenbaum, 2002). However, the collide-and-back-off approach of many medium access control protocols makes for inefficient use of the channel in high traffic situations; protocols that have a high turn-taking metric value also fairly maximise channel use among the agents, by the definition of the turn-taking metric (Raffensperger et al., 2012b). Perhaps the agents could use their successful messages on the channel to transmit information about when they needed to use the channel. However, in some cases, channel control must be independent of message content. For example, if the agents do not share a common language but have to learn to translate the messages they receive, as in an experiment by Goldman et al. (2007), then agents must simultaneously learn to play a medium access game. We consider that each agent must deduce its own control scheme based on its payoffs. The agents only have local knowledge and we compute an agent's reward solely with this local information. By using only local information, our model can be more easily applied to distributed systems. Not all methods of generating local rewards will result in good global behaviour; we identify which methods produce global turn-taking behaviour.

4.2.1 Discrete interference channel model

In the most general circumstances, agents must simultaneously learn channel coordination, language and other aspects of their joint task. However, in order to maintain our focus on learning channel coordination, and to avoid state space explosion, we use a simple interference channel and we ignore language learning and other possible

aspects of the agents' actions. We model the communications channel as a discrete-time, memoryless broadcast interference channel. We consider a set of identical agents, denoted by \mathcal{N} . Each agent $n \in \mathcal{N}$ has a single output $O_n(t)$, a binary-valued abstraction of its transmissions on the channel:

$$O_n(t) = \begin{cases} 0 & \text{if agent } n \text{ is not transmitting on the channel at time } t \\ 1 & \text{if agent } n \text{ is transmitting on the channel at time } t \end{cases} \quad (4.1)$$

We consider that each agent also has a single input, $I_n(t)$, to sense activity on the channel:

$$I_n(t) = \begin{cases} 0 & \text{if } O_m(t) = 0 \text{ for all } m \in \mathcal{N} \\ 1 & \text{otherwise} \end{cases} \quad (4.2)$$

If any agent transmits on the shared broadcast channel, then $I_n(t) = 1$. So if agent n transmits at time t , then that agent does not know whether another agent also transmitted at that time. We focus on two-agent systems where $\mathcal{N} = \{1, 2\}$.

4.2.2 Agent design

Our system consists of a single channel and a pair of agents. Each agent must act on its local information to maximise its expected payoff. Agent n 's local information comprises all past outputs and inputs, $I_n(t - t'), O_n(t - t')$ for $1 \leq t' \leq t$. An agent's current knowledge is summarised by its 'state.' The agents must have some information about their environment if they are to be reactive, so at least one input value must be included in the agents' states. Since we intend that the state is sufficient for the agents to learn turn-taking, which involves action coordination through time, we include at least one past output value. In order to make our simulations as simple as possible, we only include information from the last time step in the agents' states. Therefore, we consider that the agents' states include their last output and input:

$$S_n(t) = \langle I_n(t - 1), O_n(t - 1) \rangle \quad (4.3)$$

This kind of state has 3 possible values because $O_n(t) = 1 \implies I_n(t) = 1$ by Eq. (4.2). We discuss alternative state representations in Section 4.5.

When two Q-learning agents use the definition of state from Eq. (4.3), the next state depends on the channel properties described in Section 4.2.1, the values of both agents' states and Q-tables, and the agents' exploration policies. The definition of state in Eq. (4.3) does not have the Markov property because it does not include enough information to predict the probability distribution for the next state (Sutton and Barto, 1998, §3.5). An agent using the definition of state in Eq. (4.3) cannot always predict the probability distribution for its next state because the other agent's actions depend on

Table 4.1: One definition for agent n 's reward, $R_n(t)$, that depends on that agent's last two outputs and its last input. Each value in the rightmost column is also given a symbolic name.

$O_n(t-1)$	$O_n(t)$	$I_n(t)$	Reward value, $R_n(t)$, and symbolic name
0	0	0	$r_{unused} = 0$
0	0	1	$r_{lostturn} = 1$
0	1	1	$r_{transmit} = 1$
1	0	0	$r_{rest} = 0$
1	0	1	$r_{receive} = 1$
1	1	1	$r_{interruptedother} = 0$

what it learned on time steps $t-2, t-3, \dots$, and none of this information is included in Eq. (4.3). However, despite its minimal, non-Markovian nature, our definition of an agent's state is sufficient to demonstrate a basic form of turn-taking in pairs of agents.

On each time step, an agent can take one of two actions: either the agent chooses to transmit a message on the channel, $O_n(t) = 1$, or not, $O_n(t) = 0$. The agent maintains a policy which, in general, is a time-varying mapping between states and the probability that the agent will choose to transmit a message on the channel:

$$\pi_n(t) : \mathcal{S}_n \rightarrow [0, 1] \quad (4.4)$$

where $\pi_n(t)$ is agent n 's policy at time step t , and \mathcal{S}_n is the set of all possible states for agent n . In Section 4.4.1, we describe how Q-learning agents update their policies to maximise their expected rewards.

We calculate each agent's rewards with local information. We do not assume that the agents have prior knowledge of the structure or values of the reward function or the dynamics of the interfering channel. As with state definitions, reward functions can be defined in a multitude of ways. The reward value should be based on the agent's current action and the consequence of that action, so it should at least include the agent's current output and input. To allow for the possibility of turn-taking, we must distinguish constantly repeated actions from alternating actions; this can be achieved by including at least one past output. Therefore, we choose a simple reward function definition based on an agent's last output, current output and current input. Table 4.1 shows an example reward function of this form. We restrict our study to reward functions of this form but we vary the values in each row of the last column of Table 4.1. We consider other possible reward function definitions in Section 4.5.

We use the turn-taking metric that we develop in Chapter 3, $\tau\tau(t, r)$, to measure the quantity of turn-taking present in the results of a particular simulation (see also Raffensperger et al., 2012b). Our turn-taking metric assumes that agents are sharing a limited resource; here the interfering communications channel is the shared resource.

An agent takes a turn using the shared resource if its output is 1 while all other agents have outputs of 0. We compute the global quantity of turn-taking from each agent's allocation, as in Eq. (3.1) in Chapter 3 on p. 35. In this case, agent n' allocation is computed as:

$$\alpha_n(t, r) = \frac{1}{r} \sum_{t'=t}^{t+r-1} O_n(t') [1 - O_m(t')] \quad (4.5)$$

where $n, m \in \{\langle 1, 2 \rangle, \langle 2, 1 \rangle\}$, t is the time-step that the turn-taking calculation refers to and r is the *resolution* of the turn-taking (Raffensperger et al., 2012b, §2.3). For more details on $\tau\tau$, see Chapter 3 and the published paper (Raffensperger et al., 2012b).

4.3 Insights gained from stationary agents

While our primary goal is to understand the emergence of turn-taking in medium access games played by Q-learning agents, we gain insight by examining non-learning agents with stationary policies. Analysing agents with stationary deterministic policies enables us to understand which policies a Q-learning agent can learn that will produce turn-taking; analysing agents with stationary stochastic policies enables us to predict the average payoff that a Q-learning agent will gain by playing a particular policy against the other agent. Once we know the average payoffs of the agents for each pair of policies, we can compute the Nash equilibria of that medium access game, which we use to predict the behaviour of pairs of Q-learning agents in medium access games.

4.3.1 Turn-taking behaviour in agents with deterministic policies

We can compactly represent deterministic policies as three bits, representing the agent's output $O_n(t)$ given the three possible values of the agent's state $S_n(t)$, that is, in the form $\langle \pi_n|_{\langle 0,0 \rangle}, \pi_n|_{\langle 1,0 \rangle}, \pi_n|_{\langle 1,1 \rangle} \rangle$. Table 4.2 shows an example of a deterministic policy, $\langle 1, 0, 0 \rangle$. In game theory terms, the agent employs a 'pure strategy' for each possible state: the agent will always take the same action for any particular state. There are $2^3 = 8$ different deterministic policies with a state as defined in Eq. (4.3), so there are $8 \times 8 = 64$ different pairs of policies. Because the agents' policies are deterministic, the initial states of both agents, $S_1(0)$ and $S_2(0)$, are important. We consider the performance of a pair of policies for all possible different initial states of the agents. We can describe an initial condition with both agents' outputs, $O_1(-1)$ and $O_2(-1)$; there are therefore $2^2 = 4$ different initial conditions.

We exhaustively searched the space of pairs of deterministic policies and initial conditions and found all combinations of policies and initial conditions that produce

Table 4.2: An example of a deterministic policy. In the form $\langle \pi_1|_{\langle 0,0 \rangle}, \pi_1|_{\langle 1,0 \rangle}, \pi_1|_{\langle 1,1 \rangle} \rangle$, this is policy $\langle 1, 0, 0 \rangle$.

State components		State	The policy determines the next action
$I_1(t-1)$	$O_1(t-1)$	$S_1(t)$	$p(O_1(t) = 1) = \pi_1 _{S_1(t)}$
0	0	$\langle 0, 0 \rangle$	1
1	0	$\langle 1, 0 \rangle$	0
1	1	$\langle 1, 1 \rangle$	0

Table 4.3: The deterministic policies and initial conditions that produce turn-taking. We show the agents' policies in the form $\langle \pi_n|_{\langle 0,0 \rangle}, \pi_n|_{\langle 1,0 \rangle}, \pi_n|_{\langle 1,1 \rangle} \rangle$.

Agent 1 policy, π_1	Agent 2 policy, π_2	Initial conditions
$\langle 0, 1, 0 \rangle$	$\langle 0, 1, 0 \rangle$	Asymmetric
$\langle 1, 1, 0 \rangle$	$\langle 0, 1, 0 \rangle$	Any
$\langle 0, 1, 0 \rangle$	$\langle 1, 1, 0 \rangle$	Any
$\langle 1, 1, 0 \rangle$	$\langle 1, 1, 0 \rangle$	Asymmetric

turn-taking behaviour. We classify pairs of policies using the turn-taking metric, $\tau\tau(t, r)$ as described in Chapter 3. Some pairs of policies produce turn-taking with $\tau\tau(t, 2) = 1$ for $t > 0$ for some initial conditions. We consider those policy pairs to 'produce turn-taking behaviour.' Most strategies do not display turn-taking behaviour for any initial conditions. Those pairs of policies that do produce turn-taking must have either *non-identical policies* in the pair or the sequence of the agents' states must commence with an *asymmetric initial condition*.

Table 4.3 lists all pairs of deterministic policies that produce turn-taking. The initial conditions required for agents with policies from Table 4.3 to take turns depend on the values of the first of the three bits in the agents' deterministic policies, $\pi_1|_{\langle 0,0 \rangle}$ and $\pi_2|_{\langle 0,0 \rangle}$. If the two agents have different policies, $\pi_1|_{\langle 0,0 \rangle} \neq \pi_2|_{\langle 0,0 \rangle}$, then any initial condition produces turn-taking. However, if $\pi_1|_{\langle 0,0 \rangle} = \pi_2|_{\langle 0,0 \rangle}$, then turn-taking will only be present if the initial condition is asymmetric: either $O_1(-1) = 0, O_2(-1) = 1$ or $O_1(-1) = 1, O_2(-1) = 0$. This matches our expectation: if the agents always respond the same way to the same situation, and they start in the same situation, then they will always do the same thing at the same time and will never alternate.

We should consider the deterministic systems described in Table 4.3 to be examples of turn-taking as a *designed* behaviour rather than a learned or emergent behaviour. However, the results for deterministic systems are important for understanding systems with Q-learning agents.

4.3.2 Computing the expected rewards for agents with stationary stochastic policies

As we show in Section 4.4.2, we can use the Nash equilibria of medium access games as played by agents with stationary policies to predict turn-taking in Q-learning agents. In order to compute the Nash equilibria, we must first derive the expected rewards for stationary stochastic policies. We use Markov chain theory (Häggström, 2002) to derive the expected rewards for each agent as a function of their policy.

Suppose each agent follows a stationary stochastic policy, $\pi_n : \mathbf{S}_n \rightarrow [0, 1]$, where \mathbf{S}_n is the set of possible states. Given a reward function definition of the form given in Table 4.1, we can construct a Markov chain that fully describes the dynamics of the system. The global system state is fully described by the agents' states and the state of their reward function, see Table 4.1. We compute the global system state, $S_g(t)$, as:

$$S_g(t) = \langle O_1(t), O_2(t) \rangle \quad (4.6)$$

There are 4 different global states. For agents with stationary stochastic policies, this global system state has the Markov property.

In order to compute the agents' expected rewards for a particular pair of policies, we have to compute the probabilities that each agent will receive each different reward value, r_{unused} , $r_{receive}$, etc. These probabilities come from the global state transition probabilities and the steady state distribution of global states. The previous global state and the agents' policies determine the probability that the system will take a particular value for the next global state, S^* :

$$p[S_g(t+1) = S^*] = p[\pi_1|_{S_g(t)} = O_1^*]p[\pi_2|_{S_g(t)} = O_2^*] \quad (4.7)$$

where $\pi_n|_{S_g(t)}$ is agent n 's policy evaluated at agent n 's state, given that the global state is $S_g(t)$, and O_n^* is agent n 's output for state S^* . If we name the possible values for the global state with integers, thus $S_g(t) \in \{1, 2, 3, 4\}$, then we can construct a transition probability matrix, \mathbf{P} , such that $[\mathbf{P}]_{ij} = p[S_g(t+1) = j | S_g(t) = i]$. We compute the steady-state distribution of states directly:

$$\bar{\mathbf{S}} = \lim_{t \rightarrow \infty} \mathbf{P}^t \quad (4.8)$$

If this limit exists, then each row of $\bar{\mathbf{S}}$ is equal to the steady state probability distribution of states (Puterman, 1994, p. 593). The limit will exist when the Markov chain is irreducible and aperiodic, which is always the case in this chapter. In practice, we approximate this limit by evaluating \mathbf{P}^t with a large value of t , such as $t = 1000$.

Table 4.4: An example medium access game in normal form. We show the greedy policies for the agents in the form $\langle \pi_n|_{\langle 0,0 \rangle}, \pi_n|_{\langle 1,0 \rangle}, \pi_n|_{\langle 1,1 \rangle} \rangle$. The agents actually play the exploration policy that is a stochastic perturbation of the greedy policy. The entries of the table are of the form average payoff for row agent, average payoff for the column agent. We highlighted the pairs of policies that produce turn-taking in gray.

π_n	$\langle 0,0,0 \rangle$	$\langle 0,0,1 \rangle$	$\langle 0,1,0 \rangle$	$\langle 0,1,1 \rangle$	$\langle 1,0,0 \rangle$	$\langle 1,0,1 \rangle$	$\langle 1,1,0 \rangle$	$\langle 1,1,1 \rangle$
$\langle 0,0,0 \rangle$	0.18, 0.18	0.54, 0.10	0.24, 0.23	0.67, 0.10	0.52, 0.48	0.89, 0.10	0.54, 0.50	0.90, 0.10
$\langle 0,0,1 \rangle$	0.10, 0.54	0.30, 0.30	0.12, 0.54	0.37, 0.14	0.29, 0.71	0.49, 0.25	0.30, 0.70	0.50, 0.14
$\langle 0,1,0 \rangle$	0.23, 0.24	0.54, 0.12	0.52, 0.52	0.75, 0.13	0.62, 0.62	0.86, 0.14	0.81, 0.80	0.90, 0.14
$\langle 0,1,1 \rangle$	0.10, 0.67	0.14, 0.37	0.13, 0.75	0.18, 0.18	0.16, 0.89	0.18, 0.38	0.17, 0.90	0.19, 0.18
$\langle 1,0,0 \rangle$	0.48, 0.52	0.71, 0.29	0.62, 0.62	0.89, 0.16	0.50, 0.50	0.89, 0.11	0.54, 0.53	0.90, 0.11
$\langle 1,0,1 \rangle$	0.10, 0.89	0.25, 0.49	0.14, 0.86	0.38, 0.18	0.11, 0.89	0.32, 0.32	0.14, 0.86	0.39, 0.15
$\langle 1,1,0 \rangle$	0.50, 0.54	0.70, 0.30	0.80, 0.81	0.90, 0.17	0.53, 0.54	0.86, 0.14	0.70, 0.70	0.90, 0.14
$\langle 1,1,1 \rangle$	0.10, 0.90	0.14, 0.50	0.14, 0.90	0.18, 0.19	0.11, 0.90	0.15, 0.39	0.14, 0.90	0.18, 0.18

Let \mathbf{R}^n be the reward matrix for agent n , such that $[\mathbf{R}^n]_{ij}$ is the reward to agent n when the global state transitions from i to j . We can now compute agent n 's expected reward as the sum of all the entries in the element-wise product of \mathbf{R}^n , $\bar{\mathbf{S}}^T$ and \mathbf{P} :

$$E[r_n(t)] \Big|_{\pi_1 \cup \pi_2} = \sum_{i=1}^4 \sum_{j=1}^4 [\mathbf{R}^n]_{ij} [\bar{\mathbf{S}}^T]_{ij} [\mathbf{P}]_{ij} \quad (4.9)$$

Thus we can compute the agents' expected rewards given any possible stationary stochastic policy.

4.3.3 Computing the Nash equilibria of medium access games

Here we show how to use stationary agents' expected rewards to compute the Nash equilibria (Nash, 1950) of medium access games by expressing the game in 'normal form' as a table of payoffs given both agents' policies. Q-learning agents learn a single best action given a particular state. We refer to a Q-learning agent's current estimate of the best action to take for each state as its 'greedy policy.' However, while a Q-learning agent is learning, it must explore to discover the best rewards while also exploiting the current known good rewards. We refer to a Q-learning agent's policy as an 'exploration policy' if the policy also includes stochastic exploratory moves. A Q-learning agent's exploration policy is stochastic and non-stationary. We use Q-learning agents with ϵ -greedy exploration policies: they choose an action at random with probability ϵ , otherwise they choose the best action (Sutton and Barto, 1998, §2.2). Using the Markov chain method developed in Section 4.3.2, we can compute the expected reward for a Q-learning agent with a stable exploration policy, given a stable policy for its opponent, the reward function and the value of ϵ .

Table 4.4 shows an example of a medium access game in normal form for two agents playing ϵ -greedy exploration policies with $\epsilon = 0.2$ that are rewarded according to Table 4.1. The policies that produce turn-taking gain high rewards for both agents because the reward values in Table 4.1 were deliberately hand-chosen to induce turn-taking behaviour. Notice that Table 4.4 is symmetric; this is because the agents are identical except that they might play different strategies.

Nash (1950) built his equilibrium concept upon the idea of countering strategies, or ‘best responses.’ Suppose one agent has a fixed probability distribution over its possible policies, for example, say $\pi_1 = \langle 0, 0, 1 \rangle$ with probability 1. Then agent 2’s best response is the probability distribution over its possible policies that maximises its expected payoffs. In this case, $\pi_2 = \langle 1, 0, 0 \rangle$ with probability 1 is agent 2’s best response, giving agent 2 an expected payoff of 0.71. More formally: let Π_1 and Π_2 be policy probability column vectors for agent 1 and agent 2. Let \mathbf{V}_1 be the expected reward for agent 1 such that $[\mathbf{V}_1]_{ij} = E[R_1(t)]|_{\pi_1 \cup \pi_2}$, where π_1 is the i th policy in Π_1 and π_2 is the j th policy in Π_2 . We say that Π_1 is a best response to Π_2 if and only if Π_1 maximises $\Pi_1^T \mathbf{V}_1 \Pi_2$ (von Stengel, 2007, p. 55). A Nash equilibrium is a pair of policy probability distributions that are best responses to each other (von Stengel, 2007, p. 56).

The ‘support’ of a policy probability distribution is the set of policies that the agent plays with non-zero probability. We say that a Nash equilibrium is ‘mixed’ if the support of either agent’s policy distribution includes more than one policy; a ‘pure’ Nash equilibrium is one where each agent plays a single policy with probability 1. Unfortunately, the definition of a Nash equilibrium does not readily admit a method for finding the Nash equilibria of a game. We used a support enumeration algorithm described by von Stengel (2007) to compute all the Nash equilibria in our medium access games (von Stengel, 2007, p. 56). The game in Table 4.4 has three Nash equilibria, all of which produce turn-taking as a system behaviour. Two of those Nash equilibria are pure:

- Agent 1 plays $\langle 0, 1, 0 \rangle$ against $\langle 1, 1, 0 \rangle$ played by agent 2
- Agent 1 plays $\langle 1, 1, 0 \rangle$ against $\langle 0, 1, 0 \rangle$ played by agent 2

The third Nash equilibrium is mixed:

- Both agents play $\langle 0, 1, 0 \rangle$ with probability 0.28 and $\langle 1, 1, 0 \rangle$ with probability 0.72.

At each of these Nash equilibria, neither agent can increase its expected payoff by changing its probability distribution over policies. For the pure Nash equilibria of the game in Table 4.4, we can see that the row agent cannot increase its expected payoff by changing its policy (that is, changing the row) and also that the column agent cannot increase its expected payoff by changing its policy (that is, changing the column).

For the mixed Nash equilibrium, this informal verification of the Nash equilibria in Table 4.4 does not work because we must precisely consider probability distributions over rows and columns.

Every game is guaranteed to have at least one mixed Nash equilibrium (Nash, 1950), but pairs of Q-learning agents cannot converge to every possible mixed Nash equilibrium. Some medium access games only have mixed Nash equilibria which therefore cannot be played by pairs of Q-learning agents with ϵ -greedy exploration policies. Despite this limitation, we can use the Nash equilibria to classify and predict the emergence of turn-taking in pairs of Q-learning agents with reasonable accuracy.

4.4 Predictors for turn-taking in medium access games played by Q-learning agents

Here we identify which reward functions induce turn-taking in pairs of Q-learning agents in medium access games and we describe how to design rewards conducive to the emergence of turn-taking based on a set of predictors that we infer from our simulation results.

4.4.1 Q-learning algorithm set up

A Q-learning agent adapts the actions it chooses to maximise its reward (Watkins, 1989). A Q-learner maintains a table of learned state-action values, $Q_n[S_n, O_n]$. Learning is done locally for each agent. Agent n 's Q-table is updated as:

$$Q_n[S_n(t), O_n(t)] \leftarrow Q_n[S_n(t), O_n(t)] + \beta \left(R_n(t) + \gamma \max_O Q_n[S_n(t+1), O] - Q_n[S_n(t), O_n(t)] \right) \quad (4.10)$$

where $S_n(t)$ is the state, $R_n(t)$ is the reward and $O_n(t)$ is the output produced at time t for agent n , and where β is the learning rate and γ is a factor to discount future rewards (Sutton and Barto, 1998). A Q-learner's policy is a stochastic mapping between states and outputs (Sutton and Barto, 1998, §3.7). As mentioned in Section 4.3.3, we have chosen the ' ϵ -greedy' exploration scheme (Sutton and Barto, 1998, §2.2).

We follow the simulation procedure shown in Algorithm 4.1. Table 4.5 summarises the experimental parameters and gives their values. We hold the learning rate, exploration amount, and discount factor constant over the course of each simulation. We give all state-action pairs the same initial value, $Q_0 = 100$, chosen as an optimistic learning bias to induce the agents to explore untried states and actions. We set the number of computation steps, t_{end} , to 5000.

```

1 Initialise the agents' states arbitrarily.
2 Initialise the agents' Q-tables with  $Q_0$ .
3 For  $t = 0, 1, 2, \dots, t_{end}$  do
4   For each agent  $n \in \{1, 2\}$  do
5     Agent  $n$  sets  $O_n(t)$  according to its policy,  $\pi_n(t)$ .
6   Set  $I_1(t)$  and  $I_2(t)$  according to Eq. (4.2).
7   For each agent  $n \in \{1, 2\}$  do
8     Agent  $n$  observes  $I_n(t)$  and computes its next state,  $S_n(t + 1)$ .
9     Set  $R_n(t)$  according to agent  $n$ 's reward function.
10    Agent  $n$  updates its policy according to Eq. (4.10).

```

Algorithm 4.1: Experiment procedure pseudo code.**Table 4.5:** Experimental parameter values.

Parameter name	Symbol	Domain	Value
Learning rate	β	$[0, 1]$	0.2
Exploration amount	ϵ	$[0, 1]$	0.2
Discount factor	γ	$[0, 1)$	0.9
Initial Q-table value	Q_0	\mathbb{R}	100
Computation time steps	t_{end}	\mathbb{N}	5000

The policies learned by the agents are highly dependent on the reward function. We use rewards of the form in Table 4.1, controlling the values of r_{unused} , $r_{lostturn}$, etc., in order to illustrate their effects on the resulting system behaviour. In some cases, a designer will not be able to control the reward function exactly; then our results enable the designer to predict whether or not a system will produce turn-taking behaviour. For simplicity, we restrict the value of each agent's reward function to a finite, discrete set:

$$R_n(t) \in \{0, 0.25, 0.5, 0.75, 1\} \quad (4.11)$$

We focus on two-agent systems with perfect symmetry between the two agents such that both agents have the same parameters and same reward function. Unlike for deterministic agents, this symmetry between the agents poses no barrier to the emergence of turn-taking because of the stochastic nature of the simulation.

By Eq. (4.11), the reward function can take one of five different values for each row of the six rows in Table 4.1. Therefore, there are $5^6 = 15625$ possible reward functions. However, a game is invariant for positive-affine transformations of the agents' payoff table (von Stengel, 2007, p. 54), so some of the resulting reward functions are theoretically identical. The agents' payoff table is a scaled sum of the reward values r_{unused} , $r_{lostturn}$, etc., with non-negative scale factors, so any two reward functions that differ only by an affine transformation will result in two payoff tables that also only differ by an affine transformation. Table 4.6 shows two reward functions

Table 4.6: Two theoretically identical reward functions. The values of $R_n(t)$ for Case 2 are twice the values for Case 1 plus 0.25.

Case 1		Case 2	
Reward symbol	$R_n(t)$	Reward symbol	$R_n(t)$
r_{unused}	0.25	r_{unused}	0.75
$r_{lostturn}$	0	$r_{lostturn}$	0.25
$r_{transmit}$	0.25	$r_{transmit}$	0.75
r_{rest}	0	r_{rest}	0.25
$r_{receive}$	0.25	$r_{receive}$	0.75
$r_{interruptedother}$	0	$r_{interruptedother}$	0.25

that are theoretically identical because they differ by only a scale factor and an additive constant. There are 10803 theoretically distinct reward functions.

For each reward function, we simulated an ensemble of 1000 independent systems. Each system ran for 5000 time steps. Fig. 4.2 shows the emergence of turn-taking in one system, with rewards as given in Table 4.1. This reward function gives the agents high rewards for turn-taking behaviour; the mean reward over all 1000 independent systems for agent 1 is 0.48 for the first 1000 time-steps, before the onset of turn-taking, and is 0.93 for the last 1000 time-steps. Because the agents take exploratory moves, the turn-taking is not perfect. However, the Q-learning agent's post-learning average turn-taking is substantially higher than 0.44 (see Fig. 4.2), the expected turn-taking at resolution $r = 100$ for pairs of random agents (Raffensperger et al., 2012b, §3).

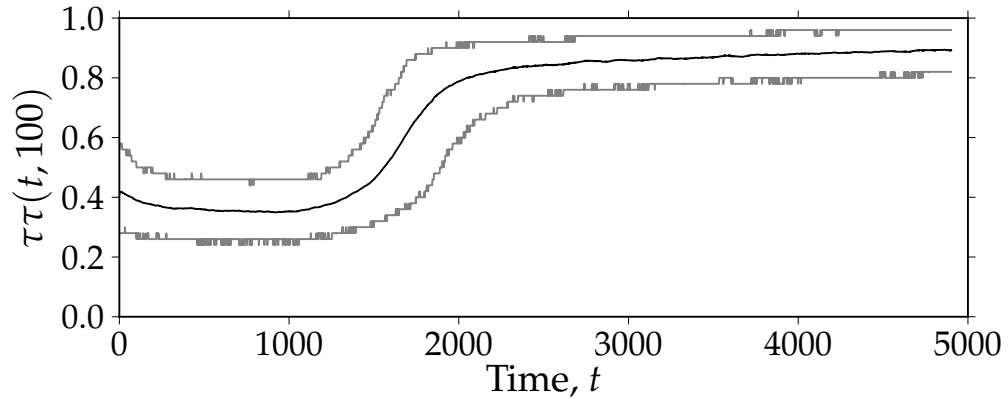


Figure 4.2: Turn-taking at a resolution of 100 as a function of time in pairs of simulated Q-learning agents rewarded as in Table 4.1, the mean of the ensemble of 1000 simulations is shown in black and the 5th and 95th percentiles are shown in gray. The turn-taking increases as the agents learn to maximise their reward.

4.4.2 Classification methods

We classify a system as producing turn-taking behaviour if the agents learn a set of greedy policies that would produce turn-taking with $\tau\tau(t,2) = 1$ when the exploration and learning are turned off, given an initial condition that is favourable towards turn-taking. We consider that the agents' exploratory moves are incidental to the underlying policy. We refer to this evaluation of the final learned policies of a system as the post-simulation turn-taking classification. Because we are looking at the average of an ensemble of stochastic simulations, we have to set a threshold for the presence or absence of turn-taking for an ensemble: we chose to identify a reward function as one that produces turn-taking if more than 95% of the systems with that reward function learned a policy that would produce turn-taking.

More formally, we evaluate the proportion of simulation runs that produce a turn-taking strategy as the final greedy policy of both agents. We define an agent's greedy policy as the action with the highest Q-table value for each state:

$$\pi_n^{greedy} = \{\arg \max_O Q_n[S, O] \text{ for all states } S\} \quad (4.12)$$

For simulation run i , we record the value of π_n^{greedy} for both agents at the end of the simulation. If both agents' policies correspond to the deterministic turn-taking policies (see Section 4.3.1), then we consider that turn-taking emerged in simulation run i , $\tau\tau_i^{policy} = 1$, otherwise we consider that turn-taking has not emerged, $\tau\tau_i^{policy} = 0$. We repeat this process for each of our 1000 independent systems for each reward function, and consider that turn-taking is a system behaviour for Q-learning agents if:

$$\frac{1}{1000} \sum_{i=1}^{1000} \tau\tau_i^{policy} > \phi \quad (4.13)$$

where ϕ is the confidence level of our post-simulation turn-taking classifier, which we set at 0.95 unless otherwise stated. By this definition, we found that 1843 out of 10803 possible reward functions produce turn-taking as a system behaviour.

We use the Nash equilibria of the medium access game defined by a given reward function to predict the post-simulation turn-taking. This way, we can predict the outcome from the reward function alone without doing the simulation. In theory, we expect turn-taking to emerge in systems whose reward function defines a medium access game that has a Nash equilibrium where both agents have a turn-taking policy. However, Q-learning agents are not guaranteed to converge to the optimal policy in non-stationary environments, as is the case in our multi-agent reinforcement learning context (Buşoniu et al., 2008, p. 10).

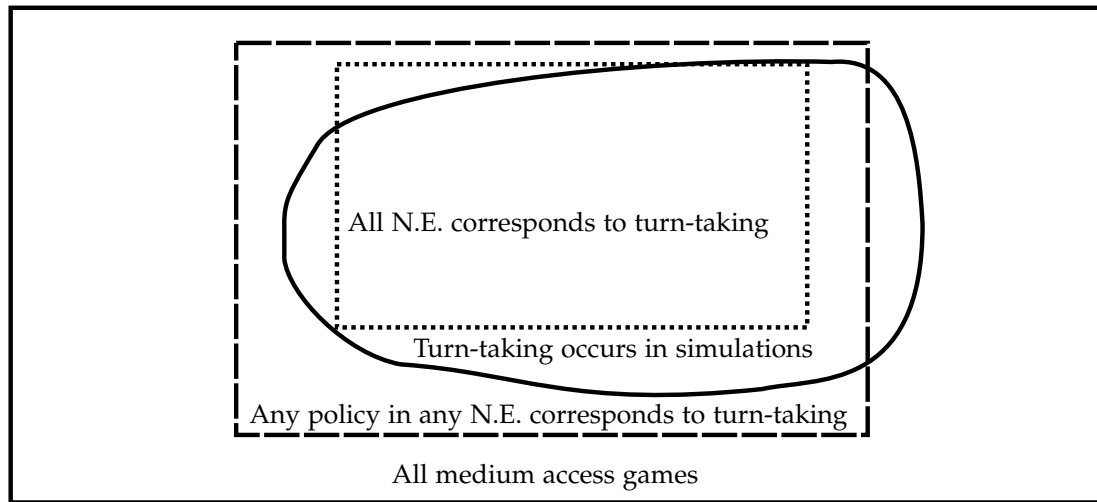


Figure 4.3: We approximate the space of turn-taking by using two predictors: the outer bound is the set of medium access games where any policy in any of its Nash equilibria (N.E.) corresponds to turn-taking and the inner bound is those games where all Nash equilibria correspond to turn-taking.

Despite the limitations of the game-theoretic approach, we can still find approximate predictors of turn-taking behaviour. For each reward function, we compute all the Nash equilibria of the medium access game as played by stationary stochastic policies, as in Section 4.3.2. If *all* the agents' policies in those Nash equilibria have support only for the four deterministic policies that produce turn-taking, then we predict that Q-learning agents will *always* learn turn-taking policies. Given the known limitations of Q-learning agents' performance in general-sum stochastic games (Zinkevich et al., 2006; Waltman and Kaymak, 2007; Buşoniu et al., 2008; Crandall and Goodrich, 2010), this method works with reasonable success: We correctly predict 1631 reward functions as producing turn-taking and 8716 as not producing turn-taking. However, we erroneously predict 244 functions as taking turns when they do not, and 212 functions as not taking turns when they do.

Similarly, we predict that turn-taking will *only* emerge if *any* policy in the support of the Nash equilibria corresponds to the four deterministic policies that produce turn-taking. This method correctly predicts 1843 reward functions as taking turns and 6858 reward functions as not taking turns, but erroneously predicts 2102 functions as taking turns that actually do not. This way, we bound the space of medium access games that show turn-taking in simulation with two predictors, as shown in Fig. 4.3. Interestingly, if we predict turn-taking when the support of at least any one Nash equilibrium corresponds to the deterministic policies that produce turn-taking, we get a predictor with performance between the one that checks all Nash equilibria and the one that checks any policy in any Nash equilibrium. We summarise these results in Table 4.7.

Table 4.7: We can use the Nash Equilibria (N.E.) to predict whether or not Q-learners will learn turn-taking policies. This table shows the number of different non-identical reward functions for different cases, depending on whether turn-taking is present in the simulation results and whether we predict turn-taking with each of three different game theoretic methods.

Turn-taking (95% confidence)	Predictor		
	All N.E.	Any N.E.	Any policy in a N.E.
Predicted and present	1631	1729	1843
Predicted but not present	244	722	2102
Not predicted and not present	8716	8238	6858
Not predicted but present	212	114	0
Turn-taking present	1843	1843	1843
Total	10803	10803	10803

An alternative way of visualising the performance of predictors is to look at their ‘receiver operating characteristics’ (Fawcett, 2006), shown in Fig. 4.4. By adjusting the confidence level of our post-simulation turn-taking classifier, ϕ in Eq. (4.13), we change the number of functions that we consider to have turn-taking present. The curves in Fig. 4.4 are parameterised by ϕ and show how well the predictors perform. We compute the true and false positive rates in Fig. 4.4 as:

$$\text{True positive rate} = \frac{\text{Number of functions correctly predicted to have turn-taking}}{\text{Total number of functions with turn-taking}} \quad (4.14)$$

$$\text{False positive rate} = \frac{\text{Number of functions erroneously predicted to have turn-taking}}{\text{Total number of functions without turn-taking}} \quad (4.15)$$

An ideal predictor would have a true positive rate of 1 and a false positive rate of 0 for all values of ϕ .

4.4.3 Classifier verification

We verify that our prediction methods generalise beyond the particular set of reward functions defined by Eq. (4.11) by testing our predictors on a new data set. We repeated the simulation method in Section 4.4.2, but this time we used random reward values that were not bounded to a finite, fixed set. Formally, we assign the reward values, r_{unused} , $r_{lostturn}$, etc., to uniformly distributed values in the range $[0, 1]$. Our results are similar to those given above in Section 4.4.2. Out of 10000 theoretically distinct, random reward functions, we found that the predictor that uses all Nash equilibria corresponding to turn-taking had a true positive rate of $1427/1580 = 0.90$ and a false positive rate of $329/8420 = 0.03$. Our predictor based on any policy in a

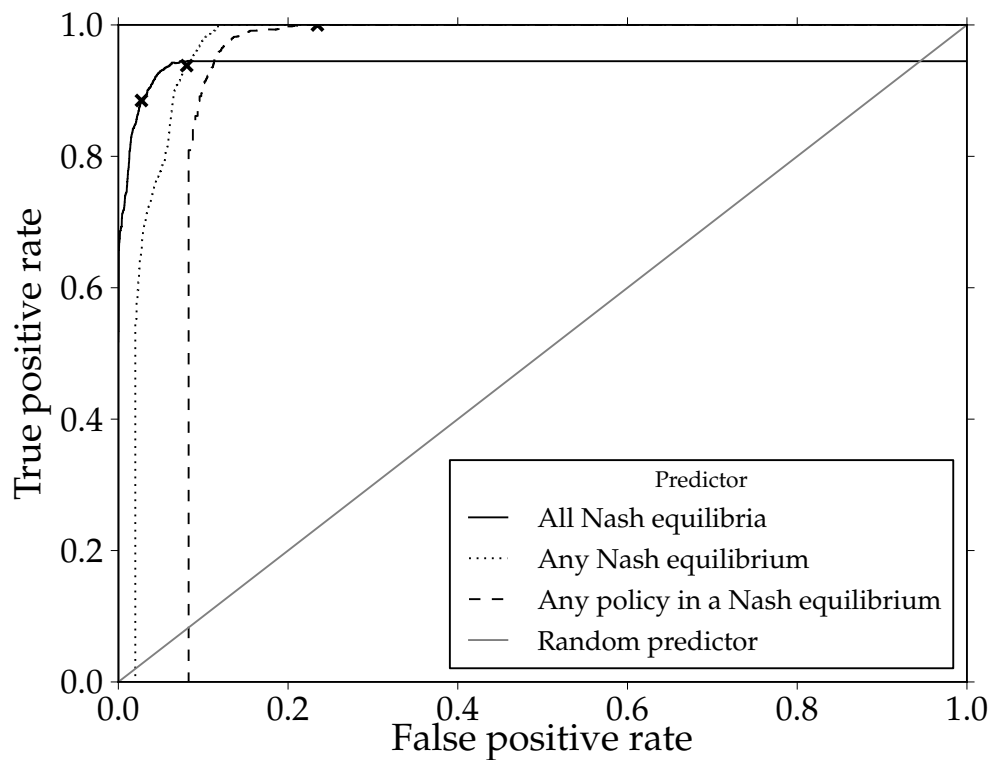


Figure 4.4: Receiver operating characteristics for the turn-taking predictors. As we vary the threshold of the post-simulation turn-taking classification from 0% in the lower left of the graph to 100% in the upper right, the false positive rates and true positive rates of the predictors change, revealing their different characteristics. The 'x's mark the 95% turn-taking threshold.

Nash equilibrium had a true positive rate of $1580/1580 = 1$ and a false positive rate of $2096/8420 = 0.25$. From these results we conclude that our predictors are not specific to the reward function set defined by Eq. (4.11) .

4.4.4 Designing rewards for turn-taking

Our classifier results can guide system designers who wish to have turn-taking as a system behaviour, or conversely, who wish to not have turn-taking. First we must express the medium access game in normal form, by computing the expected payoffs of both agents for each pair of possible policies using the Markov chain method described in Section 4.3.2. Next, we must compute the Nash equilibria of the medium access game, as in Section 4.3.3. Then we can make predictions about the emergence of turn-taking using the game theoretic predictors described in Section 4.4.2.

To create a reward function that will be conducive to turn-taking in a medium access game when played by two Q-learning agents, a reward function should be chosen such that all the Nash equilibria correspond to policies that produce turn-taking. According to our simulation results, having all Nash equilibria as ones that produce turn-taking is a good predictor for the presence of turn-taking in pairs of Q-learning agents. This is not a strict guarantee, but the search space for policies that do produce turn-taking is greatly reduced by only looking at cases where all the Nash equilibria correspond to turn-taking policies. As described in Section 4.4.3, a randomly chosen reward function has a $1580/10000 = 15.8\%$ chance of producing turn-taking, but one that comes from a population where all Nash equilibria correspond to turn-taking increases the chance to $1427/1756 = 81.7\%$. (Out of the 1756 random reward functions with all Nash equilibria corresponding to turn-taking, 1427 resulted in turn-taking as a system behaviour in simulation.) Similarly, to create a reward function that will be prohibitive to turn-taking, a reward function should be chosen such that no Nash equilibria correspond to policies that produce turn-taking.

4.5 Discussion

Our simulations reveal that the Q-learning agents frequently fail to learn any of the calculated Nash equilibria. If the agents were game-theoretic ‘rational’ agents with perfect information we would expect them to always play a Nash equilibrium. In that case, we could always expect turn-taking to emerge if all Nash equilibria correspond to turn-taking policies and we could only expect turn-taking to emerge if at least any one of the Nash equilibria corresponds to a turn-taking policy. Table 4.8 shows how

Table 4.8: Some of our simulations showed turn-taking as a system behaviour. This table shows those reward functions that produce turn-taking, broken down by what portion of the Nash equilibria (N.E.) support turn-taking policies. (The data are from the same simulation as that in Table 4.7)

Portion of N.E. policies that support turn-taking	Number of simulations with turn-taking at 95% confidence
All policies of all N.E.	1631
All policies of at least one N.E., but not all N.E.	98
At least one policy of one N.E., but no complete N.E.	114
No policy of any N.E.	0
Total	1843

many simulations produced turn-taking depending on how many policies in the Nash equilibria supported turn-taking.

Our theoretical understanding for the success of the ‘any policy in a Nash equilibrium’ predictor is lacking. However, of the 1843 simulation ensembles that showed turn-taking behaviour, 114 had Nash equilibrium with only one of the two agents’ policies having support for turn-taking, as shown in Table 4.8. This suggests that a common outcome in our simulations is for one agent to learn its policy in a Nash equilibrium while the other agent plays a different policy. We leave investigation of this phenomena to further research.

Agents in a multi-agent reinforcement learning context face a number of inherent difficulties (Buşoniu et al., 2008). Q-learning is a single agent reinforcement learning algorithm and ‘the nonstationarity of the MARL [multi-agent reinforcement learning] problem invalidates most of the single-agent RL theoretical guarantees’ (Buşoniu et al., 2008, p. 10). There are extensions to Q-learning that perform better in stochastic games (Hu and Wellman, 2003; Greenwald and Hall, 2003) but, with the exception of WoLF-PHC (Bowling and Veloso, 2002), these require agents to have knowledge of the other agents’ actions or rewards. Our formulation of a medium access game assumes that the agents cannot fully observe other agents’ actions, other agents’ rewards, or the full system state. This limitation prevents us from applying algorithms that require more information. We have opted to study Q-learning agents because of their wide use in the literature and in practice: Buşoniu et al. (2008) note that some researchers use single-agent reinforcement learning algorithms for multi-agent reinforcement learning because the single agent algorithms can be simpler.

Our focus has been exclusively on pairs of identical agents but emergent turn-taking is also useful in larger groups of agents. We believe that our methods will generalise to more than two agents. From the point of view of a particular agent, situations with more than two agents can be considered as unfair sharing between that agent

and another aggregate agent which takes up the time of all the other agents. The turn-taking metric is already suited to multiple agents, and can be extended to include unequal sharing of the channel by weighting each agent's allocation, replacing Eq. (4.5) with:

$$\alpha_n(t, r) = \frac{w_n |\mathcal{N}|}{r} \sum_{t'=t}^{t+r-1} \left(O_n(t') \prod_{m \in \mathcal{N}, m \neq n} [1 - O_m(t')] \right) \quad (4.16)$$

where smaller values of w_n indicate that agent n should use the channel on more time steps and $\sum_{n \in \mathcal{N}} w_n = 1$. Our study so far has considered identical agents engaged in rapid turn-taking, with a resolution of $r = 2$. Our choice of state representation and reward definition were focused on this situation. In order for an agent to coordinate turn-taking over a window of k time steps, we conjecture that the agent would need state information relating to at least the last k inputs and outputs. The agents' reward functions would also require more information. For example, we could compute the reward from the number of times the agent used the channel in the last k time steps, $\sigma_O = \sum_{t'=0}^k O_n(t - t')$ and the number of time steps that the channel was in use $\sigma_I = \sum_{t'=0}^k I_n(t - t')$. Such a reward function could reward an agent for using the channel once out of every k time steps while the other agent or agents use the channel the remainder of the time, if, for example, $R_n(t) = 0$ except $R_n(t) = 1$ for $\sigma_O = 1$ and $\sigma_I = k$. Whether or not this reward function would enable the agents to coordinate turn-taking depends on the agents' state representation, learning algorithm and whether or not the other agents had complementary reward functions that produce Nash equilibria corresponding to turn-taking policies.

As previously mentioned, the agents' states and rewards can be defined in many ways and our specific choices were guided by simplicity. However, our methodology can be applied to different state representations and different reward functions. The analysis of deterministic agents can be extended to any finite state representation by enumerating all policies and initial states that produce turn-taking behaviour. Turn-taking is not an achievable behaviour in all state representations, for example, turn-taking in pairs of deterministic agents is impossible if the agents have only one state. Similarly, some reward function designs preclude turn-taking. For example, if, in all cases, $O_n(t) = 1 \implies R_n(t) > r^*$ and $O_n(t) = 0 \implies R_n(t) < r^*$ for some value of reward r^* , then the agents would value the action $O_n(t) = 1$ over $O_n(t) = 0$ for all states and turn-taking would be excluded entirely. Computation of the agents' average rewards with the Markov chain method can be used so long as the agents' states can be derived from a Markovian global state, and so long as the agents' rewards are computable from a global state transition. In the absence of any simulation results for alternative state representations, reward functions, or more than two agents, we cannot be sure that our prediction methods will be accurate for larger groups of agents. Further research is required, but we hypothesise that our use of Nash equilibria to

predict the emergence of turn-taking will generalise to larger groups of agents and to agents with alternative reward functions and state representations.

Suppose that a multi-agent system designer desires a particular joint behaviour other than turn-taking. We suggest that our methodology may be more generally applicable than only to designing rewards for turn-taking between agents in medium access games. One way to apply our methodology to more general cases is as follows:

1. The desired behaviour must be quantified so that its presence or absence can be measured. The turn-taking metric $\tau\tau$ is an example of how to quantify a desired behaviour.
2. The environmental dynamics must be specified or, if the environment is physical, modelled. In the case of our study, this is the interfering channel model of Section 4.2.1.
3. The agents' reinforcement learning algorithms, state definitions and reward function should be designed with enough flexibility to encompass the desired behaviour. At this stage, the reward function is in parametric form and our goal is to find those parameters that produce the desired system behaviour.
4. The designer should enumerate all those stationary joint agent policies that exhibit the desired behaviour. Essentially, this is expressing the desired joint behaviour in terms of local policies. In our embodiment, this relies on a finite state representation. If the desired behaviour is deterministic, then the stationary policies will be deterministic as in Section 4.3.1.
5. The designer should calculate the agents' expected rewards for all possible stationary stochastic joint exploration policies using a Markov chain based on a Markovian global state representation, as we do in Section 4.3.2. This allows us to express the agents' situation as a game in normal form for every reward function parameter combination.
6. The Nash equilibria of the resulting games should be calculated. As our experimental results in Section 4.4 suggest, the desired behaviour is likely to emerge for those reward function parameter combinations that produce games where all Nash equilibria correspond to the desired behaviour. Conversely, those reward function parameter combinations that produce games where no part of any Nash equilibrium corresponds to the desired behaviour are unlikely to display the desired behaviour.

After the completion of steps 1 to 6, the search space for reward function parameters is constrained and simulations can be more accurately directed. Therefore, this method enables system designers to more effectively design reward functions to influence sequential joint behaviour in groups of learning agents, provided that the desired

behaviour can be measured. In this chapter, our analysis of this methodology is only preliminary because our study focuses on the concrete case of emergent turn-taking between Q-learning agents. Some work on emergent cooperation between Q-learners playing the prisoners' dilemma has been done by Waltman and Kaymak (2007) using a similar Markov chain method.

Chapter 5 builds on this methodology for the cases of Markov decision processes, normal form games and fully observable multi-agent stochastic games. However, the methods in Chapter 5 do not follow this method exactly, focusing instead on developing a method for designing reward functions that result in games with unique Nash equilibria. Chapter 5 is not the only possible implementation of the above methodology; alternatives are possible and may be necessary in cases where the agents cannot fully observe the system state.

4.6 Related work

Multi-agent coordination through turn-taking is especially important in emergent communication. Wagner et al. (2003) and Nolfi (2005) both survey previous research in emergent communication. We argue that turn-taking should be included in emergent-communication studies, because of the importance of turn-taking to human languages. Turn-taking is a universal feature of human language (Stivers et al., 2009) and the necessity for turn-taking influences language:

It seems productive to assume that, given conversation as a major, if not THE major, locus of a language's use, other aspects of language structure will be designed for conversational use and, *pari passu* [Latin: equally], for turn-taking contingencies (Sacks et al., 1974, p. 722).

While some emergent-communication studies involve agents with built-in turn-taking abilities (Goldman et al., 2007, Definition 19), other studies specifically examine emergent turn-taking behaviour (Iizuka and Ikegami, 2004). Di Paolo (2000) examined the emergence of turn-taking in simulated mobile agents controlled by neural networks trained with genetic algorithms. Di Paolo's agents took turns signalling each other with a shared acoustic channel, which is similar to our use of an interfering channel as a medium for examining the emergence of turn-taking. However, Di Paolo does not examine different incentives for his agents, which is the central goal of our work.

In addition to being useful in emergent-communication research, emergent turn-taking relates to a number of other research areas. Human-machine turn-taking quality is important in spoken dialog systems (Raux and Eskenazi, 2008; Turunen et al., 2006) and in human-robot interaction in a range of activities, from sorting coloured objects

(Chao and Thomaz, 2010) to musical drumming interaction (Kose-Bagci et al., 2008; Weinberg and Blosser, 2009). Turn-taking also occurs in a wide range of animal species: in some situations pairs of fish take turns as a way of resolving conflicted interests (Harcourt et al., 2010), some primate species take turns grooming each other (Manson et al., 2004), and mating pairs in some Antarctic penguins take turns foraging at sea while the other mate stays with the recently hatched chick or chicks (Trivelpiece et al., 1987, p. 355). Our research enables further research in modelling turn-taking in domains such as human-robot interaction and biology.

Neill (2003) describes the ‘turn-taking dilemma,’ a variant on the prisoners’ dilemma where the optimal strategy is for the players to alternate between one defecting and the other cooperating and one cooperating and the other defecting. This situation appears in the natural behaviour of groups of dwarf mongoose (Rasa, 1989). Neill ranks fixed strategies with memories of length 1 and 2 based on their payoffs, with and without noise. Neill’s study bears some similarity to our study of stationary policies in Section 4.3 but he does not consider learning agents or specifically examine different payoffs for the agents. While Neill uses a concept of ‘evolutionary dominance’ (Neill, 2001) to rank strategies, we use the turn-taking metric $\tau\tau$.

Emergent turn-taking could be productively applied to medium access control in self-organising networks of heterogenous wireless systems. Researchers have already modelled interfering wireless networks as multi-agent cooperative or competitive games (Larsson et al., 2009; Leshem and Zehavi, 2009; Yang et al., 2009); a kind of turn-taking is the optimal solution in some cases (Larsson et al., 2009, p. 21). Mechanism design legislation for heterogeneous wireless networks relies on knowledge of conditions required for the emergence of turn-taking or other desirable coordination amongst the participating network nodes. We enable further research into emergent turn-taking in computer networks by providing researchers with further knowledge on what incentives are required for the emergence of turn-taking in medium access games played by pairs of Q-learning agents.

Chen et al. (2010) propose a game-theoretic approach to medium access control based on ‘random access games,’ which bear some similarity to our medium access games. Chen et al. define a random access game by the agents’ utility functions and a channel contention measure, which is similar to the way we define a medium access games using agents’ reward functions and an interfering channel model. Chen et al. consider that the agents must update their channel access probabilities so as to reach the (usually) unique Nash equilibrium of the random access game, whereas we consider that agents must make a binary transmission decision at each time step and that the medium access game may have multiple Nash equilibria. Chen et al. design a near-optimal medium access protocol based on CSMA/CA using gradient play

(Flam, 2002) to update the agents' channel access probabilities while we examine the emergence of turn-taking between Q-learning agents.

Helbing et al. (2005) present the results of an experiment where people plan driving routes in a computer laboratory. Helbing et al. focus on congestion in a two road network: a slow road and a fast road between the same two destinations. For a number of iterations, two (or sometimes four) players choose between the two roads and receive payoffs that depend on both player's choices. In later iterations, a form of turn-taking sometimes emerges in the route choices. Helbing et al. conjecture that 'alternating forms of reciprocity' occur when the sum of an agent's payoffs for turn-taking behaviour is higher than for a short-sighted, greedy strategy. (In Helbing et al.'s terminology, $P_{12} + P_{21} > 2P_{11}$.) Helbing et al. reproduce the results of their human-based experiments with a novel reinforcement learning model, but they calculate their agents' payoffs with central information. We consider that fully decentralised control is an essential assumption in multi-agent reinforcement learning medium access control systems.

Colman and Browning (2009) simulate the evolution of cooperative turn-taking in iterated static games played by Moore machines trained with a genetic algorithm. Colman and Browning find that turn-taking can emerge 'without communication or insight on the part of the players, and without the use of randomization (mixed strategies)' (Colman and Browning, 2009, p. 959). Colman and Browning use the average payoffs, the proportions of each move combination, 'reciprocity' (Colman and Browning, 2009, p. 955) and the prevalence of different kinds of three-round histories as turn-taking performance measures, while we use measures derived from our turn-taking metric $\tau\tau$.

4.7 Conclusion

We define 'medium access games' as a set of multi-agent stateful games. Medium access games model situations where turn-taking is useful, like human conversations and computer networks. We examine medium access games played by pairs of agents with a minimal definition of state and identify which deterministic policies give rise to turn-taking behaviour. We use Markov chain theory to derive the expected rewards for agents with stationary stochastic policies so that we can compute the Nash equilibria of medium access games. We use the Nash equilibria calculated for stationary policies to predict the presence or absence of turn-taking in medium access games played by Q-learning agents. For Q-learning agents, we find that turn-taking emerges with a high probability in those medium access games where all Nash equilibria support only turn-taking policies and that turn-taking does not emerge in those medium access

games where no policy in any Nash equilibria supports turn-taking policies. Our results enable system designers to predict the emergence of turn-taking in medium access games played by Q-learning agents and to design rewards that are either conducive or prohibitive to turn-taking.

This chapter supports the thesis by providing an example of how sequential joint behaviour in groups of learning agents can be influenced by appropriate reward functions where the 'appropriateness' of the reward function is determined using a metric for the desired behaviour. We discuss how to apply our methodology to achieve system behaviours other than turn-taking in medium access games. In Chapter 5, we discuss an extension of this chapter's methodology for algorithmically designing reward functions for stochastic games.

Chapter 5

A Method for Designing the Rewards of Stochastic Games

As we saw in Chapter 4, constructing reward functions for even simple joint behaviours is not straightforward. However, the Nash equilibria of a system are useful predictors of the joint behaviour. In this chapter, we extend the methodology of Chapter 4 to examine how we might design reward functions that result in desirable Nash equilibria. In Chapters 3 and 4, we focus on turn-taking as an example of a sequential joint behaviour that may occur in groups of learning agents. We now tackle the more general problem of algorithmically designing rewards to influence the sequential joint behaviour of learning agents, assuming that we know the set of desired behaviours. The method that we develop can be used to construct reward functions that lead groups of agents toward a wide range of sequential joint behaviours, not only turn-taking.

Algorithmically designing reward functions for multi-agent systems is difficult because we must assign credit for a global success to the deserving agents and we may need to induce coordination. In this chapter, we model multi-agent systems as stochastic games. We start with a stochastic game that has no reward function and we find a reward function that makes a particular joint policy the unique Nash equilibrium, provided that such a reward function exists. This way, a system designer can algorithmically design a reward function that will lead to a desirable Nash equilibrium if he or she can find a desirable stationary stochastic joint policy. We build our method for stochastic games on simpler techniques that apply to Markov decision processes and normal form games. We formally prove that our methods are correct and provide an example of how to design a reward function for a three-agent system that results in turn-taking as a system behaviour.

5.1 Introduction

Multi-agent systems have a myriad of applications, from financial trading simulations (Sherstov and Stone, 2005), to robotic soccer (Kitano et al., 1997) and automatic musical rhythm generation (Eigenfeldt, 2007). Multi-agent reinforcement learning (Buşoniu et al., 2008) is a promising approach to controlling multi-agent systems, but designers still have the problem of how to choose reward functions for the agents that produce desirable behaviour. In many situations, each agent has a predetermined utility function (von Neumann and Morgenstern, 1944) that encodes its preferences over all possible outcomes. For example, a financial trading agent may want to make money for itself, regardless of the payoffs of other agents. However, in some cases, a system designer must design rewards for the agents to produce a particular joint sequential behaviour. Suppose a musician is designing a percussion ensemble of musical agents to produce rhythms of some kind. In this case, the individual reward functions of each agent are free variables that the musician can adjust to control the joint behaviour of the ensemble.

Agents in a multi-agent reinforcement learning system learn to maximise their reward function. Because each agent earns its own rewards in an uncertain environment, we apply the common model that multi-agent reinforcement learning agents are playing a ‘stochastic game’ (Shapley, 1953; Littman, 1994). Our objective is to design reward functions for each agent that will result in a desired joint behaviour. One approach is to simply reward every agent equally and in proportion to how well all the agents are producing the desired behaviour; this results in a purely cooperative game. If the agents maximise the shared reward function correctly, then, by definition, the agents produce the desired behaviour. However, from the perspective of an individual agent, a shared reward function can make learning the right actions more difficult. If an agent receives a large reward, was that because it took good actions itself or did another agent perform well? In a large group of agents, an individual agent’s contribution to the global performance may be barely noticeable. Ideally, we would only reward each agent for its own contribution to the global performance. Furthermore, agents face the difficulty of coordinating their actions. For example, having all agents drive on the right hand side of the road is nominally of equal value to having them all drive on the left, but the outcome is costly if the agents follow a mix of the two policies.

In this chapter, we assume that we know the set of behaviours that we desire for the agents. Our approach is to encode a known desired joint behaviour as the unique Nash equilibrium of a stochastic game. By designing reward functions with unique Nash equilibria, we alleviate the difficulty of each agent having to select between multiple equilibria. Designing games with unique Nash equilibria is conceptually simple, but the details of game construction can be complex for mixed Nash equilibria.

Furthermore, we cannot encode all joint behaviours as Nash equilibria in a stochastic game.

Stochastic games are a generalisation of Markov decision processes and matrix games. Fig. 5.1 illustrates the hierarchy of models that we use in this chapter. Our contributions to the design agenda of multi-agent reinforcement learning start with the special cases shown in Fig. 5.1 and progress to the general case of all stochastic games:

- In Section 5.2, we describe how to design rewards for Markov decision processes that result in some desired deterministic policy. We prove that our method results in a Markov decision process where the unique optimal policy is that desired policy.
- In Section 5.3, we explain how to design rewards for normal form games. We separately consider three cases:
 1. The desired joint action distribution is deterministic (this case is trivial).
 2. The desired joint action distribution is stochastic and the game has two agents (this case has already been addressed in the literature).
 3. The desired joint action distribution is stochastic and the game has an arbitrary number of agents (necessary and sufficient conditions for this case have been given in the literature, but, to our knowledge, our method is novel).

We prove that these three methods result in games with unique Nash equilibria. We discuss cases where it is not possible to encode a desired joint behaviour as a Nash equilibrium.

- We combine our methods for Markov decision processes and normal form games to produce a complete method for designing rewards for general case stochastic games. We prove that this method results in a unique Nash equilibrium in the space of stationary strategies and demonstrate our method with an example. (See Section 5.4.)

We discuss possible extensions to our method in Section 5.5 and we review related work in Section 5.6. We draw conclusions in Section 5.7.

5.2 Designing rewards for Markov decision processes

Before we address the general problem of designing rewards for general stochastic games, we develop our understanding by considering some special cases of stochastic games. In Section 5.3 we consider designing rewards for normal form games. In this

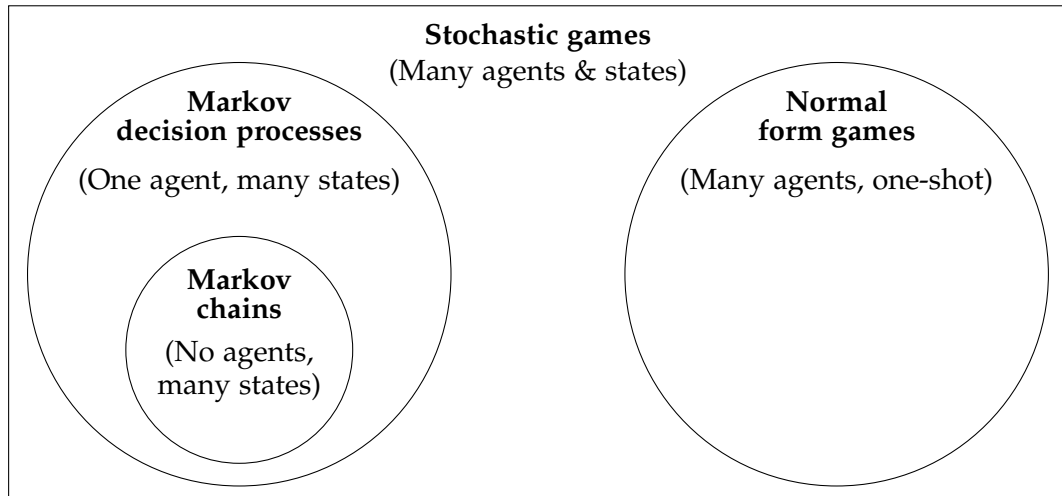


Figure 5.1: Stochastic games are a generalisation of normal form games and Markov decision processes; the latter generalise Markov chains.

section, we consider designing rewards for stochastic games for the special case of one agent: designing rewards for Markov decision processes. We demonstrate how to identify if a reward function exists for a Markov decision process that will produce a desired behaviour and we construct such a reward function for those cases where it does exist.

A Markov decision process is a tuple, $\mathcal{D} = (\mathcal{S}, \mathcal{A}, T, R)$ where \mathcal{S} is a set of states, \mathcal{A} is a set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition function and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is a reward function. The goal of the agent is to find an optimal policy, a stochastic mapping between states and actions, $\pi^* : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. A policy, π^* , is optimal if and only if, for all states $s \in \mathcal{S}$, it maximises the γ -discounted expected reward starting in state s :

$$v(s, \pi^*) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t E_{s, \pi^*} [R(t)] \quad (5.1)$$

where $R(t)$ is a random variable representing the reward at time t . Theorem 2.7.2 of Filar and Vrieze (1997) states that π^* can be a deterministic policy, that is, the agent can select a single action with probability 1 for each state in such a way that maximises the discounted expected reward for each state (Filar and Vrieze, 1997, p. 26). The set of optimal policies for a Markov decision process need not only have one member, but there exists at least one optimal deterministic policy.

5.2.1 Formalising reward design for Markov decision processes


Suppose we take the perspective of a system designer rather than the perspective of the agent. How can we design a reward function, R , to promote a particular class of behaviours? Filar and Vrieze show that we lose no generality by restricting agents to the set of stationary policies for γ -discounted Markov decision processes (Filar and Vrieze, 1997, Corollary 2.6.2 on p. 55). Therefore, we consider that an agent's 'behaviour' in a Markov decision process is its policy and that the set of desired behaviours is B , a subset of the set of all stationary policies. We can now formally express Markov decision process reward design:

Definition 5.1. *Suppose we have a Markov decision process without a reward function $\mathcal{D}^{-R} = (\mathcal{S}, \mathcal{A}, T)$ and a set of desired behaviours, B . A reward function R^\dagger is a design compliant reward function if and only if all optimal policies of the Markov decision process $(\mathcal{D}^{-R}, R^\dagger)$ are in the set of desired behaviours.*

There may be more than one possible value for R^\dagger ; in this case, we should choose R^\dagger to best assist agents that are learning from R^\dagger . Delayed rewards make learning more difficult, so a good reward function will reward agents in such a way that their myopic best actions are also optimal in the long run. Another difficulty that agents face while learning is estimating the expected reward $E[R(s, a, s')]$ for a particular action $a \in \mathcal{A}$ in a given state s . This difficulty relates to the definition of the reward function; as written, $R(s, a, s')$ depends on the current state, the current action and the next state. Since the next state is a stochastic function of the current state and the current action, the reward for the agent is also a stochastic function. A system designer can help agents learn by choosing a reward function with the following easy-learning properties:

1. The optimal policy does not change with the discount rate, so that optimal actions in the short run are also optimal in the long run.
2. The reward values do not depend on the next state, but only on the current state and the current action, so that action-value estimation is easier.

5.2.2 Rewarding an agent for following a successful policy is a design compliant reward function

 We can construct design compliant reward functions that also satisfy the two desirable properties of an easy-to-learn reward with the method described in Lemma 5.1 and Theorem 5.1, as follows:

Lemma 5.1. *Let π^0 be a deterministic policy such that $\pi^0 \in B$. If no such deterministic policy π^0 exists, then no reward function is design compliant for \mathcal{D}^{-R} .*

Proof. Suppose the contrary: some reward function is design compliant but there exists no deterministic policy in B . This implies that we can pair some reward function, R^\times , with \mathcal{D}^{-R} to form a complete Markov decision process, \mathcal{D}^\times , such that each optimal policy is also a desired behaviour. Every Markov decision process has at least one optimal deterministic policy (Filar and Vrieze, 1997, Theorem 2.3.1, p. 26). So there must be some deterministic policy that is optimal for \mathcal{D}^\times . We supposed that no deterministic policy was in B but by Definition 5.1 all optimal policies are in B . Therefore no such R^\times can exist. \square

Theorem 5.1. *Let R^0 be a reward function with no dependency on the next state such that $R^0(s, a, s') = R^0(s, a) = \pi^0(s, a) \in \{0, 1\}$, where π^0 is as given in Lemma 5.1. Then R^0 is a design compliant reward function for the reward-less Markov decision process \mathcal{D}^{-R} , if π^0 exists. Furthermore, R^0 possesses both the above easy-learning properties.*

Proof. First, we prove that π^0 is an optimal policy for $\mathcal{D}^0 = (\mathcal{D}^{-R}, R^0)$, the Markov decision process made up of \mathcal{D}^{-R} and R^0 . Then, we show that π^0 is the only optimal policy of \mathcal{D}^0 .

If π^0 is an optimal policy, then it must maximise the discounted value of each state. We can express the value of each state with $\mathbf{v}(\pi)$, the vector of state values given the agent's policy. Let $\mathbf{P}(\pi)$ be a probability transition matrix with entries:

$$[\mathbf{P}(\pi)]_{ij} = p(s^j | s^i, \pi) = \sum_{a \in \mathcal{A}} T(s^j | s^i, a) \pi(s^i, a) \quad (5.2)$$

We calculate the vector of expected rewards as:

$$\mathbf{r}(\pi) = \left(\sum_{a \in \mathcal{A}} R(s, a) \pi(s, a) \text{ for } s \in \mathcal{S} \right)^T \quad (5.3)$$

We can then calculate the vector of state values for a particular policy π as:

$$\mathbf{v}(\pi) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbf{P}^t(\pi) \mathbf{r}(\pi) \quad (5.4)$$

$$= (1 - \gamma) [\mathbf{I} - \gamma \mathbf{P}(\pi)]^{-1} \mathbf{r}(\pi) \quad (5.5)$$

See also Eqs. 2.4 and 2.5 on p. 13 of Filar and Vrieze (1997). An optimal policy is one that maximises \mathbf{v} . We can express the value of an optimal policy starting in state s as:

$$v(s) = \max_{a \in \mathcal{A}} \left\{ (1 - \gamma) R(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) v(s') \right\} \quad (5.6)$$

We now calculate $\mathbf{v}^0(\pi^0)$, the value vector for π^0 , and show that all of its elements satisfy Eq. (5.6). To find the value of $\mathbf{v}^0(\pi^0)$, we must compute values for the quantities

on the right-hand-side of Eq. (5.5). First, we calculate the value of $\mathbf{r}(\pi^0)$. We know that $\sum_{a \in \mathcal{A}} \pi^0(s, a) = 1$ because π^0 defines a probability distribution over the set of actions for each state. Also, because $\pi^0(s, a)$ is zero for all but one action for each state, we know that:

$$\sum_{a \in \mathcal{A}} \pi^0(s, a) = \sum_{a \in \mathcal{A}} \pi^0(s, a) \pi^0(s, a) \quad (5.7)$$

and, because $R^0(s, a) = \pi^0(s, a)$,

$$\sum_{a \in \mathcal{A}} R^0(s, a) \pi^0(s, a) = 1 \quad (5.8)$$

Therefore, by Eq. (5.3), $\mathbf{r}(\pi^0) = \mathbf{1}_{|S|}$.

Second, we evaluate $[\mathbf{I} - \gamma \mathbf{P}(\pi^0)]^{-1}$. Because T is unknown, we cannot calculate $\mathbf{P}(\pi^0)$ entirely. Fortunately, we only need to calculate the row-sums of $[\mathbf{I} - \gamma \mathbf{P}(\pi^0)]^{-1}$, because we know that $\mathbf{r}(\pi^0)$ equals 1 for each state. Because the rows of $\mathbf{P}(\pi^0)$ are probability distributions, we know that

$$\sum_{j=1}^{|S|} [\mathbf{P}(\pi^0)]_{ij} = 1 \quad (5.9)$$

for all i , so

$$\sum_{j=1}^{|S|} [\mathbf{I} - \gamma \mathbf{P}(\pi^0)]_{ij} = 1 - \gamma \quad (5.10)$$

for all i . Also, the following is true for all invertible matrices:

If the sum of the elements in each row of a square matrix is k , then the sum of the elements in each row of the inverse matrix is $1/k$ (Wilansky, 1951).

Since $(\mathbf{I} - \gamma \mathbf{P}[\pi^0])$ is invertible, this implies that

$$\sum_{j=1}^{|S|} [(\mathbf{I} - \gamma \mathbf{P}[\pi^0])^{-1}]_{ij} = \frac{1}{1 - \gamma} \quad (5.11)$$

for all i . Now we can calculate $\mathbf{v}^0(\pi^0)$:

$$\mathbf{v}^0(\pi^0) = (1 - \gamma) (\mathbf{I} - \gamma \mathbf{P}[\pi^0])^{-1} \mathbf{1}_{|S|} \quad (5.12)$$

$$= (1 - \gamma) \frac{1}{1 - \gamma} \mathbf{1}_{|S|} \quad (5.13)$$

$$= \mathbf{1}_{|S|} \quad (5.14)$$

That is, $\mathbf{1}_{|S|}$ is an eigenvector of $(1 - \gamma) (\mathbf{I} - \gamma \mathbf{P}[\pi^0])^{-1}$ with eigenvalue 1.

For $\mathbf{v}^0(\pi^0)$ to be a value vector of an optimal policy, Eq. (5.6) must remain satisfied for all s if $v^0(s)$ replaces $v(s)$. As demonstrated in Eq. (5.14), for all $s \in \mathcal{S}$:

$$v^0(s) = 1 \quad (5.15)$$

For π^0 to be optimal, the following must be true for all $s, s' \in \mathcal{S} \times \mathcal{S}$:

$$v^0(s) = \max_{a \in \mathcal{A}} \left\{ (1 - \gamma)R^0(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a)v^0(s') \right\} \quad (5.16)$$

Since $p(s'|s, a)$ is a probability distribution and $v^0(s')$ is known and is equal to 1 for all values of s' , we know that $\sum_{s' \in \mathcal{S}} p(s'|s, a)v^0(s') = 1$ and therefore:

$$v^0(s) = \max_{a \in \mathcal{A}} \left\{ (1 - \gamma)R^0(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a)v^0(s') \right\} \quad (5.17)$$

$$= \max_{a \in \mathcal{A}} \{ (1 - \gamma)R^0(s, a) + \gamma \} \quad (5.18)$$

Because $R^0(s, a)$ has a unique maximum at $R^0(s, a) = \pi^0(s, a) = 1$:

$$v^0(s) = (1 - \gamma) + \gamma \quad (5.19)$$

$$= 1 \quad (5.20)$$

which is the same value as computed in Eq. (5.14). Therefore, Eq. (5.6) is satisfied for all states, $\mathbf{v}^0(\pi^0) \geq \mathbf{v}$ (element-wise) for all possible value vectors \mathbf{v} and π_0 is an optimal policy of \mathcal{D}^0 for $\gamma \in [0, 1)$. Because R^0 is optimal for all discount rates, R^0 satisfies the first of the easy-learning properties. Also, because R^0 is not a function of the next state, it satisfies the second easy-learning property.

For R^0 to be a design compliant reward function, *all* optimal policies must be in the set of desired behaviours. To complete this proof, we now prove that there are no other optimal policies. If there were another optimal policy, π^\times , then it would need to achieve a payoff vector that is equal to the known optimal payoff vector: $\mathbf{v}^\times(\pi^\times) = \mathbf{v}^0(\pi^0)$. This would imply that there is some state s^\times where the agent earns the same mean immediate reward by taking some action other than that specified by π^0 . However, by the definition of R^0 , all actions other than that specified by π^0 earn strictly smaller immediate rewards. Consequently, π^0 is the unique optimal policy for \mathcal{D}^0 and therefore R^0 is a design compliant reward function for \mathcal{D}^{-R} . \square

5.3 Designing rewards for normal form games

Here we consider how to design rewards for ‘normal form games,’ another special case of stochastic games. A normal form game is a stochastic game with no states. Formally, a normal form game is a tuple $\Gamma = (\mathcal{N}, \mathcal{A}, R)$ where \mathcal{N} is a set of agents, \mathcal{A} is a joint action space, \mathcal{A}_n is the action space of agent n , $R : \mathcal{A} \rightarrow \mathbb{R}^{|\mathcal{N}|}$ is a vector-valued joint reward function and $R_n : \mathcal{A} \rightarrow \mathbb{R}$ is the reward function for agent n .

We use the following notation in our discussion of normal form games:

- Let a_n represent an element in \mathcal{A}_n (a_n is an action for some agent n) and let \mathbf{a} represent an element in \mathcal{A} (\mathbf{a} is a joint action).
- Let \mathbf{a}_{-n} represent a vector of actions for all agents except agent n , and $(a_n, \mathbf{a}_{-n}) = \mathbf{a}$ and let \mathcal{A}_{-n} be the set of possible values for \mathbf{a}_{-n} , where $\mathcal{A}_{-n} = \times_{m \in \mathcal{N}, m \neq n} \mathcal{A}_m$.
- Let π_n represent a probability distribution over agent n 's actions.
- Let $\pi_n(a_n)$ be the probability that agent n selects action a_n .
- Let π represent a vector of each agent's probability distribution over its actions: $\pi = (\pi_n \text{ for all } n \in \mathcal{N})$. We refer to π as a ‘joint policy.’ Notice that a joint policy defines a probability distribution over joint actions, but that not every probability distribution over joint actions can be written as a joint policy.

As in Section 5.2, we take the perspective of a system designer. How can we design a reward function to promote a particular class of joint behaviours? In a normal form game, a joint behaviour is a joint policy. We consider that the game designer has a set of desirable joint policies, B . If some joint policy π is in B , then that joint policy is a desired outcome. Normal form game reward design can be expressed formally:

Definition 5.2. *Suppose we have a set of agents \mathcal{N} and a set of desirable joint policies, B . A reward function R^\dagger is a design compliant reward function if and only if \mathcal{N} and R^\dagger comprise a normal form game Γ^\dagger such that every Nash equilibrium of Γ^\dagger is a desirable joint policy.*

In the remainder of this section, we examine reward design methods for three classes of normal form games:

- First, in Section 5.3.1, we consider normal form games for which a deterministic joint policy is in B . This case is trivial.
- Then, in Section 5.3.2, we consider two-agent normal form games where only stochastic joint policies are in B . Quintas (1988) presented an alternative method for this case; we discuss Quintas' method in Section 5.6.

- Finally, we extend our methods to stochastic joint policies with any number of agents, in Section 5.3.3. This case is difficult. Our method draws on our techniques for the two-agent case.

For each case, we propose one possible method for designing rewards that results in a single Nash equilibrium. We present the latter two methods as algorithms.

5.3.1 When a deterministic joint policy is desirable

Suppose that π^D is a desirable deterministic joint policy so that π^D is in B . In this case, we can satisfy the normal form game reward design criteria with a game that has only pure Nash equilibria. We now construct a reward function that determines a game where π^D is the one and only Nash equilibrium. For all $n \in \mathcal{N}$, the deterministic distribution $\pi_n^D(a_n) = 1$ for some action $a_n \in \mathcal{A}_n$ and $\pi_n^D(a'_n) = 0$ for all other actions $a'_n \neq a_n$.

In order to help the agents learn their reward function better (and to simplify our analysis), we decouple the reward function for agent n from the reward functions of all other agents, $m \neq n$. This way, the reward for agent n for a particular joint action \mathbf{a} , $R_n(\mathbf{a})$, depends only on agent n 's action. This assumption limits this method to games with only pure Nash equilibria. Let R^0 be a joint reward function with each agent's reward function defined as:

$$R_n^0(a_n, \mathbf{a}_{-n}) = \begin{cases} 1 & \text{if } \pi_n^D(a_n) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.21)$$

for all $a_n \in \mathcal{A}_n$, all $\mathbf{a}_{-n} \in \mathcal{A}_{-n}$ and all $n \in \mathcal{N}$.

Theorem 5.2. *If π^D is a deterministic joint policy where $\pi^D \in B$, and R^0 is constructed as in Eq. (5.21), then R^0 is a design compliant reward function for $\Gamma^D = (\mathcal{N}, R^0)$.*

Proof. First, we show that π^D is a Nash equilibrium of Γ^D , then we show that Γ^D has no other Nash equilibria. In order for π^D to be a Nash equilibrium, the best response condition must hold for all the agents. The best response condition is defined in terms of $v_n(a_n)$, agent n 's expected value for taking action a_n , which we compute as:

$$v_n(a_n) = \sum_{\mathbf{a}_{-n} \in \mathcal{A}_{-n}} R_n(a_n, \mathbf{a}_{-n}) \prod_{m \in \mathcal{N}, m \neq n} \pi_m(a_m) \quad (5.22)$$

where a_m is agent m 's action in \mathbf{a}_{-n} and given that all other agents have fixed their policies at π_m for $m \in \mathcal{N}, m \neq n$. If agent n 's policy, π_n , is its best response to the

other agents' policies, then the following holds for all a_n where $\pi_n(a_n) > 0$:

$$v_n(a_n) = \max_{a'_n \in \mathcal{A}_n} \{v_n(a'_n)\} \quad (5.23)$$

To show that π^D is a Nash equilibrium of Γ^D , we must demonstrate that each agent's policy is a best response to the other agents' policies. By Eq. (5.21), $\pi_n^D(a_n) > 0$ implies that $R_n^0(a_n, \mathbf{a}_{-n}) = 1$. Because π_m^D is a probability distribution for all m , we know that $\sum_{\mathbf{a}_{-n} \in \mathcal{A}_{-n}} \prod_{m \in \mathcal{N}, m \neq n} \pi_m^D(a_m) = 1$; therefore $\pi_n^D(a_n) > 0$ implies $v_n(a_n) = 1$. Because 1 is the maximum value of R_n^0 , we know that $\pi_n^D(a_n) > 0$ implies $\max_{a'_n \in \mathcal{A}_n} \{v_n(a'_n)\} = 1$ and therefore Eq. (5.23) is satisfied for all agents. Because π^D satisfies the best response condition for all agents $n \in \mathcal{N}$, we know that π^D is a Nash equilibrium of Γ^D .

Suppose that Γ^D had another Nash equilibrium, $\pi^\times \neq \pi^D$. This would imply that Eq. (5.23) holds when we replace π in Eq. (5.22) with π^\times . However, $\max_{a'_n \in \mathcal{A}_n} \{v_n(a'_n)\}$ still takes the same value, because Eq. (5.21) removes all dependencies on other agents' actions from each agent's reward function. Therefore, $\pi_n^\times(a_n) > 0$ implies that $R_n^0(a_n, \mathbf{a}_{-n}) = 1$, but this means that $\pi^\times = \pi^D$. Therefore, no such π^\times can exist. Because π^D is the only Nash equilibrium and $\pi^D \in B$, we have shown that R^0 is a design compliant reward function. \square

5.3.2 Algorithms for generating fully mixed two-agent games

Suppose that no deterministic joint policy is in the set of desired behaviours. In this case, we can only satisfy the normal form game reward design criteria with a game that has no pure Nash equilibria. We now consider designing reward functions for two-agent games that result in a unique desired Nash equilibrium. We start by examining the game of 'Matching Pennies' where each agent has two possible actions and a single mixed Nash equilibrium. We generalise the game of Matching Pennies to two agent games with any number of actions. We present an algorithm that generates two agent games with any desired mixed Nash equilibrium and prove that the algorithm always produces games with the desired equilibrium. Our algorithm is not the only way to produce games with unique mixed Nash equilibria so we comment on alternative methods in Section 5.5.

The two-agent game of Matching Pennies is defined as:

$$\mathcal{N} = \{\text{Column agent, Row agent}\} \quad (5.24)$$

$$\mathcal{A}_{\text{Column agent}} = \mathcal{A}_{\text{Row agent}} = \{H, T\} \quad (5.25)$$

with a reward function as shown in Table 5.1. One interpretation of Matching Pennies is that each of two children have a coin, which they place on a table with either heads

Table 5.1: The game ‘Matching Pennies.’ We show the reward function values in the form of reward for row agent, reward for column agent.

		Column agent	
		H	T
Row agent	H	1, 0	0, 1
	T	0, 1	1, 0

		H	T
H	1, 0	0, 1	
T	0, 1	1, 0	

Figure 5.2: The preferences of two agents in Matching Pennies are intransitive. The vertical lines represent preferences for the row agent and the horizontal lines represent preferences for the column agent.

facing up (H) or tails facing up (T). If the coins match, one child receives a prize; if the coins do not match, the other child receives the prize.

The unique Nash equilibrium of Matching Pennies is for both agents to play H and T with equal probability. In order to extend this game to more than two actions per agent, we must build a better intuition as to *why* Matching Pennies has only one mixed Nash equilibrium. Every normal form game has at least one (possibly mixed) Nash equilibrium (Nash, 1950), but some games have only pure Nash equilibria (like the Prisoners’ Dilemma in Table 2.2 on p. 22). In order for a game to have a pure Nash equilibrium, the game must have some set of actions where each agent is not unilaterally incentivised to change its action. On the other hand, if we are to design a game with no pure Nash equilibria, then, for each joint action, at least one of the agents must prefer an alternative joint action. In other words, the agents’ preferences over the game outcomes must be an intransitive operator (von Neumann and Morgenstern, 1944, §4.4); we call such games ‘intransitive.’ Matching Pennies is an intransitive game. Fig. 5.2 illustrates how the agents’ preferences form a loop: the row agent prefers $R_{\text{Row agent}}(H, H)$ to $R_{\text{Row agent}}(T, H)$ but then the column agent prefers $R_{\text{Column agent}}(H, T)$ to $R_{\text{Column agent}}(H, H)$, and so on, for all four outcomes of Matching Pennies.

By considering the ordering of agent preferences, constructing an intransitive normal form game with an arbitrary number of actions for each agent is equivalent to constructing a graph with certain properties. Suppose we have a pair of agents $\mathcal{N} = \{1, 2\}$ and a set of actions \mathcal{A}_n for each agent $n \in \mathcal{N}$, where the number of actions available to each agent is equal, $|\mathcal{A}_1| = |\mathcal{A}_2|$. We base our game around

a graph, $G = (\mathcal{V}, \mathcal{E})$, with vertices for each action of each agent, $\mathcal{V}(G) = \bigcup_{n \in \mathcal{N}} \mathcal{A}_n$. For example, if $\mathcal{A}_1 = \{a_1^1, a_1^2, a_1^3, a_1^4\}$ and $\mathcal{A}_2 = \{a_2^1, a_2^2, a_2^3, a_2^4\}$, then the vertices of G will be $\mathcal{V}(G) = \{a_1^1, a_1^2, a_1^3, a_1^4, a_2^1, a_2^2, a_2^3, a_2^4\}$. In a slight abuse of notation, we refer to vertices of G with elements of \mathcal{A}_1 and \mathcal{A}_2 . The graph must be directed. An arc in $\mathcal{E}(G)$ from some $a_n \in \mathcal{A}_n$ to $a_m \in \mathcal{A}_m$ means that agent m values its action a_m more highly than any of its other actions, if agent n plays action a_n . Because this definition of the arcs only makes sense when $n \neq m$, the graph is a bipartite graph: there are no arcs from the vertices in \mathcal{A}_1 back to the vertices in \mathcal{A}_1 . Furthermore, we consider that each vertex has a single arc to it and a single arc from it. For the game to be intransitive, agent n 's best action must change whenever another agent m changes action, for all agents n and $m \neq n$. Therefore, the graph G must be a Hamiltonian graph. In summary, a graph can be transformed into an intransitive two-agent normal form game if it meets the following criteria:

1. Each agent has the same number of actions.
2. The graph has vertices for each action of each agent.
3. Each vertex has a single arc to it and a single arc from it.
4. There are no arcs leading from one of agent n 's actions to another of agent n 's actions, for all $n \in \mathcal{N}$.
5. The graph has a Hamiltonian cycle.

Input : A directed bipartite graph, G , with vertices $\mathcal{V}(G) = \mathcal{A}_1 \cup \mathcal{A}_2$ and with an arc to each vertex and an arc from each vertex such that the graph has a Hamiltonian cycle.

Output: Reward functions, R_1 and R_2 , for an intransitive two-agent normal form game.

- 1 Set $R_n(a_n, a_m) \leftarrow 0$ for all $a_n, a_m \in \mathcal{A}_n \times \mathcal{A}_m$ where $\langle n, m \rangle \in \{\langle 1, 2 \rangle, \langle 2, 1 \rangle\}$.
- 2 Choose an arbitrary starting vertex a_n in $\mathcal{V}(G)$.
- 3 **While** at least one vertex has not been visited **do**
- 4 Let a_m be the direct successor to a_n .
- 5 Set $R_n(a_n, a_m) \leftarrow 1$.
- 6 Advance to the next vertex, i.e. $a_n \leftarrow a_m$.
- 7 **Return** R_1 and R_2 .

Algorithm 5.1: A method for converting a directed bipartite graph with a Hamiltonian cycle into an intransitive two-agent game.

Given an action preference graph that meets the above criteria, G , we can construct an intransitive normal form game with Algorithm 5.1. Fig. 5.3 shows an example graph (5.3a) and the game that results from applying Algorithm 5.1 to the graph (5.3b). We generated Fig. 5.3a with Algorithm A.1 in Appendix A which generates an

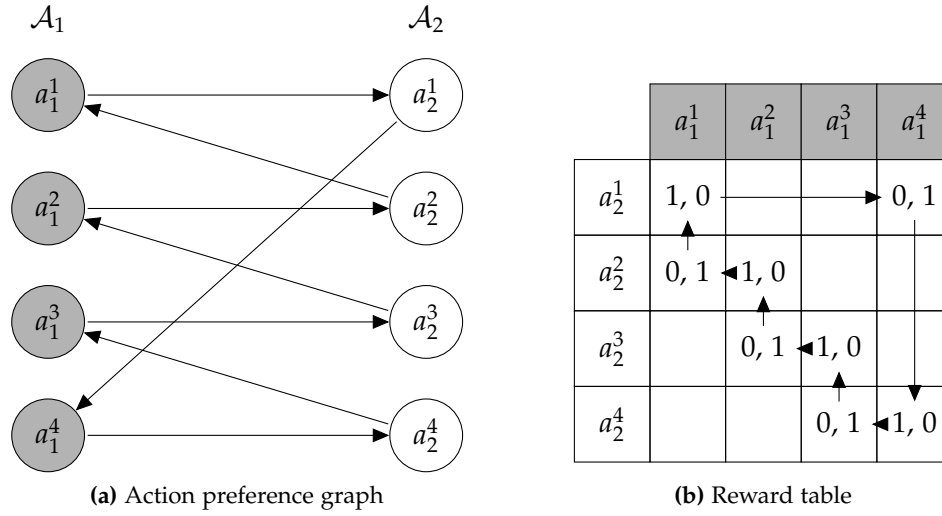


Figure 5.3: An intransitive two-agent game where each agent has four actions. The intransitive nature of the game can be represented as a bipartite graph with a Hamiltonian cycle (a). The table on the right (b) shows the reward functions with the agent preferences marked as arrows. The blank cells in (b) have a value of 0 to both agents.

action preference graph that meets the above criteria for any number of agent actions. Because the game in Fig. 5.3b is intransitive, it cannot have a pure Nash equilibrium; therefore it has at least one mixed Nash equilibrium. The game in Fig. 5.3b is a generalisation of Matching Pennies to four actions per agent and has a single Nash equilibrium where each agent plays each action with equal probability. We can use graphs other than that in Fig. 5.3a to make four action generalisations of Matching pennies; for example, we can permute the actions of either agent and still have a feasible solution. If both agents have the same number of actions, $|\mathcal{A}_1| = |\mathcal{A}_2|$, then we can construct a normal form game that has a single Nash equilibrium where each agent plays each action with equal probability by running Algorithm 5.1 with any graph that meets criteria 1-5, as discussed above.

Theorem 5.3. *A two-agent game constructed with Algorithm 5.1 has a single, fully mixed Nash equilibrium where the agents play each action with equal probability.*

Proof. ‘A necessary and sufficient condition for the existence of a bimatrix game that has (\bar{x}, \bar{y}) as its unique equilibrium point is that $k(\bar{x}) = k(\bar{y})$,’ (Kreps, 1974, p. 115) where $k(\cdot)$ represents the size of the support of a policy. Therefore, if we can show that Algorithm 5.1 constructs two-agent games with fully mixed Nash equilibria, then we know that Algorithm 5.1 constructs games with unique equilibria.

In a Nash equilibrium, each agent’s policy, π_n for all $n \in \mathcal{N}$, must meet the best response condition described in Eqs. (5.22) and (5.23). If a game has a fully mixed Nash equilibrium where the agents play each action with equal probability, then the

probabilities $\pi_n(a_n) = 1/|\mathcal{A}_n|$ for all $a_n \in \mathcal{A}_n$ and all $n \in \mathcal{N}$ must satisfy the best response condition. We check that $\pi_1(a_1) = 1/|\mathcal{A}_1|$ for all $a_1 \in \mathcal{A}_1$ is a best response to $\pi_2(a_2) = 1/|\mathcal{A}_2|$ for all $a_2 \in \mathcal{A}_2$. The following holds for all of agent 1's actions, $a_1 \in \mathcal{A}_1$:

$$v_1(a_1) = \sum_{a_2 \in \mathcal{A}_2} R_1(a_1, a_2) \pi_2(a_2) \quad (5.26)$$

If $\pi_2(a_2) = 1/|\mathcal{A}_2|$ for all $a_2 \in \mathcal{A}_2$, then Eq. (5.26) can be written as:

$$v_1(a_1) = \frac{\sum_{a_2 \in \mathcal{A}_2} R_1(a_1, a_2)}{|\mathcal{A}_2|} \quad (5.27)$$

for all $a_1 \in \mathcal{A}_1$. We know that $R_1(a_1, a_2) = 1$ for one and only one action $a_2^* \in \mathcal{A}_2$ because each action in \mathcal{A}_2 is a vertex of G in Algorithm 5.1 and each vertex is visited exactly once in the while loop starting at line 3. The other values of R_1 are zero. Therefore, for all $a_1 \in \mathcal{A}_1$, we know that $\sum_{a_2 \in \mathcal{A}_2} R_1(a_1, a_2) = 1$ and $v_1(a_1) = \max_{a_2^* \in \mathcal{A}_2} v_1(a_1, a_2^*) = 1/|\mathcal{A}_2|$. Consequently, the fully mixed uniformly distributed π_1 is a best response to the fully mixed uniformly distributed π_2 . By symmetry, π_2 is a best response to π_1 . Therefore, π_1 and π_2 comprise a fully mixed Nash equilibrium. \square

In this section, our goal is to design reward functions that produce a single desired Nash equilibrium for two-agent normal form games. Our graph method generalises Matching Pennies to any number of actions, but only if both agents have the same number of actions and only if the desired Nash equilibrium is the one where each action has equal probability. We need to generalise our graph method to non-equal probabilities, but fortunately we have no need to consider Nash equilibria with unequal support sizes. For non-degenerate two-agent games, the size of the support of a mixed Nash equilibrium is equal for both agents (von Stengel, 2007, Proposition 3.3 on p. 56). Consequently, we can design a square $k \times k$ game for $|\mathcal{A}_n| = k$ and $|\mathcal{A}_m| > k$ actions, then replace agent m 's missing actions with zero rewards for both agents. Algorithm 5.2 uses Algorithm 5.1 to create a two-agent normal form game with any single fully mixed Nash equilibrium.

Theorem 5.4. *A two-agent game constructed with Algorithm 5.2 has a single fully mixed Nash equilibrium where the agents play each action with the probabilities specified by π_n for $n \in \{1, 2\}$.*

Proof. A game constructed with Algorithm 5.2 has at least one Nash equilibrium (Nash, 1950), and if it has a fully mixed Nash equilibrium, then it has only one mixed Nash equilibrium. This is because Algorithm 5.2 merely modifies the values of the reward functions produced by Algorithm 5.1, so we can apply the reasoning given in the proof of Theorem 5.3.

Input : The probabilities that each agent will take each action at the Nash equilibrium, where $\pi_n(a_n)$ is the probability that agent n takes action a_n , for each agent $n \in \{1, 2\}$ and each action $a_n \in \mathcal{A}_n$. We assume that $\pi_n(a_n) > 0$ for both agents and all actions and that both agents have the same number of actions.

Output: Reward functions for both agents, R_1 and R_2 , that comprise a game with one Nash equilibrium where agents play actions at the given probabilities.

- 1 Let G be a graph that meets the intransitive two-agent game graph criteria 1-5.
- 2 Let R'_1 and R'_2 be the reward functions produced by Algorithm 5.1 using G .
- 3 **For each** $a_1, a_2 \in \mathcal{A}_1 \times \mathcal{A}_2$ **do**
- 4 Set $R_1(a_1, a_2) \leftarrow R'_1(a_1, a_2) / \pi_2(a_2)$.
- 5 Set $R_2(a_1, a_2) \leftarrow R'_2(a_1, a_2) / \pi_1(a_1)$.
- 6 **Return** R_1 and R_2 .

Algorithm 5.2: A reward generator for two-agent games with a given fully mixed Nash equilibrium.

To complete the proof, we must show that the fully mixed policies π_1 and π_2 are best responses to each other. We prove that π_1 is a best response to π_2 by showing that π_1 satisfies Eq. (5.23) for all actions $a_n \in \mathcal{A}_n$ when $n = 1$. Because π_1 is fully mixed, the following holds for all of agent 1's actions:

$$v_1(a_1) = \sum_{a_2 \in \mathcal{A}_2} R_1(a_1, a_2) \pi_2(a_2) \quad (5.28)$$

As mentioned in the proof of Theorem 5.3, for a given a_1 , we know that $R'_1(a_1, a_2)$ in Algorithm 5.2 equals 1 for one and only one action in \mathcal{A}_2 . So, for a given a_1 , this implies that $R_1(a_1, a_2) \neq 0$ for one and only one action $a_2^* \in \mathcal{A}_2$; in fact, $R_1(a_1, a_2^*) = 1 / \pi_2(a_2^*)$, where a_2^* depends on a_1 . For a given a_1 , all other values of $R_1(a_1, a_2) = 0$ for $a_2 \neq a_2^*$. Therefore:

$$v_1(a_1) = \sum_{a_2 \in \mathcal{A}_2} R_1(a_1, a_2) \pi_2(a_2) = R_1(a_1, a_2^*) \pi_2(a_2^*) \quad (5.29)$$

for all $a_1 \in \mathcal{A}_1$. Because $R_1(a_1, a_2^*) = 1 / \pi_2(a_2^*)$, we find that $v_1(a_1) = \pi_2(a_2^*) / \pi_2(a_2^*) = 1 = \max_{a'_1 \in \mathcal{A}_1} v_1(a'_1)$ for all $a_1 \in \mathcal{A}_1$. (Because π_2 is fully mixed, $\pi_2(a_2) \neq 0$ for all a_2 .) Consequently, the fully mixed π_1 satisfies Eq. (5.23) and is a best response to the fully mixed π_2 . By symmetry, π_2 is a best response to π_1 . Therefore, π_1 and π_2 comprise the one and only Nash equilibrium, which is fully mixed. \square

Theorem 5.4 leads us to the following conclusion about designing two-agent normal form games:

Corollary 5.1. *If π^M is a fully mixed joint policy where $\pi^M \in B$ and R^0 is the joint reward function constructed by Algorithm 5.2 with input π^M , then R^0 is a design compliant reward function for two-agent games where both agents have the same number of actions.*

We can construct two-agent normal form games that satisfy the normal form game reward design criteria where the agents have a different number of actions by adding actions to one of the agents, so long as the agents take those actions with zero probability. However, the normal form game reward design criteria cannot be satisfied for two-agent games if $\pi \in B$ implies that the supports of the agents' policies are unequal. This is because both agents must have supports of equal size in a Nash equilibrium of a non-degenerate game (von Stengel, 2007, p. 56). The only exception to this rule would be a degenerate game with a design compliant reward function. This is possible, but degenerate games have the disadvantage of having an infinite number of Nash equilibria, which is likely to hinder agent learning because it makes coordination more difficult. We do not give further consideration to games with degenerate Nash equilibria.

5.3.3 Generalising to normal form games with arbitrary numbers of agents

We generalise our two-agent results to normal form games with any number of agents. Our goal is to construct a game with a single Nash equilibrium where the agents take their actions with probabilities that we specify in advance. First, we generalise our graph method from Section 5.3.2 to construct multi-agent games where each game has a single fully mixed Nash equilibrium where each agent plays each action with equal probability. Then, we discuss how to transform normal form games with such Nash equilibria into games with the full range of possible unique Nash equilibria.

As we show in Section 5.3.2, we can construct two-agent normal form games with any fully mixed Nash equilibrium if both agents have support for the same number of actions. Here we consider the analogous constraint for games with more than two agents. We can construct a normal form game with some fully mixed joint policy as the unique Nash equilibrium if and only if the following constraint on the agents' numbers of actions holds (Kreps, 1981, p. 126):

$$\max_{n \in \mathcal{N}} \{|\mathcal{A}_n|\} = |\mathcal{A}_k| \leq 2 - |\mathcal{N}| + \sum_{m \in \mathcal{N}, m \neq k} |\mathcal{A}_m| \quad (5.30)$$

We choose to construct a generalisation of Matching Pennies for more than two agents. As in Section 5.3.2, we construct a directed graph, $G = (\mathcal{V}, \mathcal{E})$, that represents the ordering of the agents' preferences. Again, the graph has a vertex for each action of each agent: $\mathcal{V}(G) = \bigcup_{n \in \mathcal{N}} \mathcal{A}_n$. Here, the meaning of an arc is subtly different than in

Section 5.3.2; an arc in $\mathcal{E}(G)$ from a_n to a_m means that agent m prefers to switch to action a_m immediately after agent n has switched to action a_n , given whatever actions the other agents are playing. No agent can be incentivised to change its action twice in succession. Therefore, instead of being bipartite, as in the two-agent case, the graph must be $|\mathcal{N}|$ -partite, such that it has no arcs from any element in \mathcal{A}_n back to another element in \mathcal{A}_n for all $n \in \mathcal{N}$. As in Section 5.3.2, each vertex must have a single arc to it and a single arc from it and the graph must have a Hamiltonian cycle. In summary, a graph can be transformed into a multi-agent intransitive game if it meets the following ‘multi-agent action preference graph criteria:’

1. The agents’ numbers of actions must satisfy Eq. (5.30).
2. The graph has a vertex for each action of each agent.
3. Each vertex has a single arc to it and a single arc from it.
4. There are no arcs leading from one of agent n ’s actions to another of agent n ’s actions, for all $n \in \mathcal{N}$.
5. The graph has a Hamiltonian cycle.
6. The graph must satisfy the ‘action path dependency criterion:’ Suppose we have a (u, v) -path such that $u, v \in \mathcal{V}(G) \times \mathcal{V}(G)$ and $u \neq v$. Let \mathcal{P}_n be the subset of all vertices on the path between u and v (but not including v) such that every element in \mathcal{P}_n is also an element in \mathcal{A}_n . A graph satisfies the action path dependency criterion if, for all such paths, there exists some $n \in \mathcal{N}$ such that:

$$0 < |\mathcal{P}_n| < |\mathcal{A}_n| \quad (5.31)$$

Note that all the conditions except 1 and 6 are exactly the same as in the two-agent case presented in Section 5.3.2. Algorithm A.2 in Appendix A generates one possible graph for any number of agents with any number of actions, provided that the number of actions for each agent satisfies Eq. (5.30).

Given a graph that satisfies the multi-agent action preference graph criteria, we can use Algorithm 5.3 to construct a normal form game with a unique Nash equilibrium where each agent plays its actions with equal probability. Generating intransitive games for an arbitrary number of agents is more complex than generating two-agent intransitive games, therefore Algorithm 5.3 is more complex than Algorithm 5.1. While the joint reward function for a two-agent game is two-dimensional, a general joint reward function is $|\mathcal{N}|$ -dimensional. The graph concisely represents a closed path through the $|\mathcal{N}|$ -dimensional joint reward function space. The action path dependency criterion prevents this path from visiting the same point twice. Each point on the path

Input : A directed graph, G , that satisfies the multi-agent action preference graph criteria.

Output: A joint reward function, R , for an intransitive multi-agent normal form game.

- 1 Set $R_n(\mathbf{a}) \leftarrow 0$ for all joint actions $\mathbf{a} \in \mathcal{A}$ and all $n \in \mathcal{N}$.
- 2 Let \mathbf{a} be an arbitrary joint action.
- 3 // First, visit each vertex to initialise \mathbf{a} with a meaningful value.
- 4 Choose an arbitrary starting vertex a_n in $\mathcal{V}(G)$.
- 5 **While** not all vertices have been visited **do**
- 6 Set $\mathbf{a} \leftarrow (a_n, \mathbf{a}_{-n})$.
- 7 Advance to the next vertex: $a_n \leftarrow$ the direct successor of a_n .
- 8 // Second, visit each vertex again to set the appropriate values of R .
- 9 **While** not all vertices have been visited **do**
- 10 Set $\mathbf{a} \leftarrow (a_n, \mathbf{a}_{-n})$.
- 11 Set $R_n(\mathbf{a}) \leftarrow 1$.
- 12 Advance to the next vertex: $a_n \leftarrow$ the direct successor of a_n .
- 13 **Return** R .

Algorithm 5.3: A method for converting a directed partite graph that satisfies the multi-agent action preference graph criteria into an intransitive multi-agent game.

has some joint action and a reward value of one for some agent, and the path leaves each point by making a rotation to an orthogonal direction.

Algorithm 5.3 starts by setting all values of $R_n(\mathbf{a})$ to zero (line 1) and choosing an arbitrary starting joint action, \mathbf{a} . Lines 5–7 serve to initialise \mathbf{a} with a meaningful starting value by visiting each vertex of G and updating \mathbf{a} with the changes that each vertex implies; after visiting each vertex, \mathbf{a} is guaranteed to have a valid value. In lines 9–12, each vertex on the graph is visited again, this time setting the appropriate reward function value to 1 while also keeping track of the current value of \mathbf{a} .

Notice that a complete representation of R requires $|\mathcal{N}| \prod_{n \in \mathcal{N}} |\mathcal{A}_n|$ real numbers, but that Algorithm 5.3 generates a reward function that we can represent with $\sum_{n \in \mathcal{N}} |\mathcal{A}_n|$ non-zero values. This sparse reward function representation is beneficial. We can efficiently store reward functions for large games. In general, computation of the Nash equilibria of a normal form game is PPAD-complete (Chen and Deng, 2006), but we may be able to exploit our prior knowledge of the sparse representation to more efficiently compute the Nash equilibria of these games. For the two-agent case, the Nash equilibria of games generated with Algorithm 5.3 can be computed in linear time (Codenotti et al., 2006).

Before we show that Algorithm 5.3 successfully constructs a normal form game with the specified Nash equilibrium, we demonstrate the importance of the action path dependency criterion.

Lemma 5.2. *If G satisfies all the multi-agent action preference graph criteria, then no value of \mathbf{a} occurs more than once in the while loop in Algorithm 5.3 on lines 9–12.*

Proof. In order for a value of \mathbf{a} to occur twice in the while loop in Algorithm 5.3 on lines 9–12, some set of agents must change their actions, then change their actions back to their original values, while the remainder of the agents leave their actions unchanged.

Because of the multi-agent action preference graph criterion number 4, we can be sure that no agent changes its action on one vertex and then reverts that change on the next vertex. Furthermore, because each agent has a single vertex for each of its actions (criteria 2), the only way for an agent to repeat one of its actions is for all of its actions to be on the same path. So for a joint action to repeat in the loop, there must be a path that meets the following conditions:

- At least one agent has all of its actions on the path.
- All agents that do not have all their actions on the path have no actions on the path.
- The path does not include all vertices in the graph.

However, no such path can exist because we consider that the graph satisfies the action path dependency criterion, so all paths that do not include all the vertices have some agent which has some number $|\mathcal{P}_n|$ of actions on the path such that $0 < |\mathcal{P}_n| < |\mathcal{A}_n|$. Because no path can result in all agents with actions on that path switching all their actions, no joint action can repeat in the while loop on lines 9–12. \square

Theorem 5.5. *A normal form game constructed with Algorithm 5.3, Γ , has a unique, fully mixed Nash equilibrium where the agents play each action with equal probability.*

Proof. First we show that Γ has a fully mixed Nash equilibrium, then we show that the equilibrium is unique.

Let π be the joint policy where the agents play each action with equal probability and let R be the reward function produced by Algorithm 5.3. We now show that all games constructed with Algorithm 5.3 have π as a Nash equilibrium. In order for π to be a Nash equilibrium, the best response condition must hold for all agents. Because π is fully mixed, the following holds for all of agent n 's actions:

$$v_n(a_n) = \sum_{\mathbf{a}_{-n} \in \mathcal{A}_{-n}} R_n(a_n, \mathbf{a}_{-n}) \prod_{m \in \mathcal{N}, m \neq n} \pi_m(a_m) = \max_{a'_n \in \mathcal{A}_n} v_n(a'_n) \quad (5.32)$$

for all $n \in \mathcal{N}$ and where a_m is agent m 's action in \mathbf{a}_{-n} . Because π is fully mixed and uniformly distributed, we know that $\pi_m(a_m) = 1/|\mathcal{A}_m|$ and therefore $\prod_{m \in \mathcal{N}, m \neq n} \pi_m(a_m) = 1 / \prod_{m \in \mathcal{N}, m \neq n} |\mathcal{A}_m|$.

By Lemma 5.2, each value of \mathbf{a} occurs exactly once in the loop on lines 9–12. Therefore, for each a_n there exists some \mathbf{a}_{-n}^* such that $R_n(a_n, \mathbf{a}_{-n}^*)$ equals 1 and $R_n(a_n, \mathbf{a}_{-n})$ equals zero for all other \mathbf{a}_{-n} . Consequently, $\sum_{\mathbf{a}_{-n} \in \mathcal{A}_{-n}} R_n(a_n, \mathbf{a}_{-n}) = 1$ for all a_n and all \mathbf{a}_{-n} , and Eq. (5.32) holds for all a_n and all $n \in \mathcal{N}$ with:

$$v_n(a_n) = \frac{1}{\prod_{m \in \mathcal{N}, m \neq n} |\mathcal{A}_m|} = \max_{a'_n \in \mathcal{A}_n} v_n(a'_n) \quad (5.33)$$

Therefore, π , the joint policy where each agent plays each action with equal probability, is a fully mixed Nash equilibrium of Γ .

'Let x^* be a completely mixed strategy N -tuple. A necessary and sufficient condition for the existence of a finite non-cooperative game Γ that has x^* as its unique equilibrium point is that [Eq. (5.30) holds]' (Kreps, 1981, p. 126). We know that the game Γ has a unique equilibrium provided that Γ has a fully mixed Nash equilibrium. Because we have shown that Γ has a fully mixed Nash equilibrium, π , we know that π is the unique Nash equilibrium of Γ . \square

Our method for constructing reward functions for stochastic games with arbitrary numbers of agents has two parts. The first part (presented above) constructs games with unique fully mixed Nash equilibria and the second part extends the games constructed by the first part by adding all actions that an agent takes with zero probability and all agents that take an action with probability one to the desired equilibrium. Here we consider that second part. Let $\overline{\pi}_n$ denote the size of the support of π_n , i.e. $\overline{\pi}_n$ is the number of actions in π_n such that $\pi_n(a_n) > 0$. We can construct a design compliant reward for a normal form game such that the game has a unique Nash equilibrium if the following holds for all agents $n \in \mathcal{N}$:

$$\max_{n \in \mathcal{N}} \{\overline{\pi}_n\} = \overline{\pi}_k \leq 2 - |\mathcal{N}| + \sum_{m \in \mathcal{N}, m \neq k} \overline{\pi}_m \quad (5.34)$$

That is, the portion of the joint policy that has non-zero action probabilities must satisfy Eq. (5.30). If a joint policy satisfies Eq. (5.34), then we can use Algorithm 5.4 to construct a normal form game where that joint policy is the unique Nash equilibrium.

Theorem 5.6. *A normal form game constructed with Algorithm 5.4 has a unique Nash equilibrium where the agents play each action with the probabilities specified by π and the expected reward for each agent is one.*

Input : The probabilities that each agent will take each action at the Nash equilibrium, π , where $\pi_n(a_n)$ is the probability that agent n takes action a_n , for each agent $n \in \mathcal{N}$ and each action $a_n \in \mathcal{A}_n$. We assume that Eq. (5.34) is satisfied.

Output: A joint reward function for all agents, R , that defines a game with one Nash equilibrium where agents play actions at the given probabilities.

- 1 //Reward agents as in Eq. (5.21) if they follow an action with probability 1.
- 2 Set $R_n(\mathbf{a}) \leftarrow 0$ for all $n \in \mathcal{N}$ and all $\mathbf{a} \in \mathcal{A}$.
- 3 Let $\mathcal{M} \leftarrow \emptyset$.
- 4 Let $\mathcal{A}' \leftarrow \emptyset$.
- 5 **For each** $n \in \mathcal{N}$ **do**
- 6 **If** there exists some a_n such that $\pi_n(a_n) = 1$ **then**
- 7 Set $R_n(a_n, \mathbf{a}_{-n}) \leftarrow 1$ for all \mathbf{a}_{-n} .
- 8 **Else**
- 9 $\mathcal{M} \leftarrow \mathcal{M} \cup \{n\}$.
- 10 $\mathcal{A}' \leftarrow \mathcal{A}' \cup \{\text{the set of actions in } \mathcal{A}_n \text{ such that } \pi_n(a_n) \in (0, 1)\}$.
- 11 //We now have a reduced game Γ' with agents \mathcal{M} and actions \mathcal{A}' .
- 12 //Create a fully mixed Nash equilibrium for Γ' .
- 13 Let G be a graph with vertex set \mathcal{A}' that meets the multi-agent action preference graph criteria.
- 14 Let R' be the joint reward function produced by Algorithm 5.3 using G .
- 15 **For each** $\mathbf{a} \in \mathcal{A}$ **do**
- 16 Set $q \leftarrow \prod_{m \in \mathcal{M}} \pi_m(a_m)$ where the values of a_m come from the joint action \mathbf{a} .
- 17 **For each** $m \in \mathcal{M}$ **do**
- 18 Set $R_m(\mathbf{a}) \leftarrow \pi_m(a_m) R'_m(\mathbf{a}_{\mathcal{M}}) / q$ where a_m is agent m 's action in joint action \mathbf{a} and where $\mathbf{a}_{\mathcal{M}}$ is the vector of actions in \mathbf{a} for the agents in \mathcal{M} .
- 19 **Return** R .

Algorithm 5.4: A generator for multi-agent normal form games with a given Nash equilibrium.

Proof. Algorithm 5.4 splits the game into two parts: a deterministic part where $\pi_n(a_n)$ is either 0 or 1, and stochastic part where $\pi_n(a_n)$ lies in the open set $(0, 1)$. On lines 1–10, the algorithm identifies the stochastic part of the game and fills in the reward function for the deterministic part.

Each agent in the deterministic part is unaffected by the actions of the other agents and can gain an expected payoff of 1. If the game is entirely deterministic, then $\mathcal{M} = \emptyset$ and Algorithm 5.4 reduces to an implementation of Eq. (5.21) and the veracity of this theorem follows from Theorem 5.2. The remainder of this proof considers the case where $\mathcal{M} \neq \emptyset$.

The agents not in \mathcal{M} are dummy agents; their actions do not affect the outcome for other agents. Furthermore, for the agents in \mathcal{M} , their reward function is always smaller for actions not in \mathcal{A}' than it is for actions in \mathcal{A}' . Therefore, the agents in \mathcal{M}

will only play actions in \mathcal{A}' and a Nash equilibrium can only include actions in \mathcal{A}' for the agents in \mathcal{M} . To complete this proof, we must show that the reduced game $\Gamma' = (\mathcal{M}, \mathcal{A}')$ has a unique fully mixed Nash equilibrium that corresponds to the stochastic part of the game.

The remainder of this proof is similar to the proof of Theorem 5.4, with details changed to reflect the generalisation to an arbitrary number of agents. If the reduced game Γ' constructed with Algorithm 5.4 has a fully mixed Nash equilibrium, then that equilibrium is unique. This is because the reward function of Γ' in Algorithm 5.4 is only a modification of the reward functions produced by Algorithm 5.3, so we can apply the reasoning given in the proof of Theorem 5.5.

Let $\pi_{\mathcal{M}}$ be the stochastic part of π . Because $\pi_{\mathcal{M}}$ excludes any deterministic actions, we know that $\pi_{\mathcal{M}}$ is a fully mixed joint policy. To complete the proof, we must show that $\pi_{\mathcal{M}}$ comprises agent probability distributions π_m for $m \in \mathcal{M}$ that are best responses to the other agents' policies. We prove that π_m is a best response to the other agents' policies in $\pi_{\mathcal{M}}$ by showing that π_m satisfies Eq. (5.23). Because $\pi_{\mathcal{M}}$ is fully mixed, the following holds for all of agent m 's actions in \mathcal{A}'_m :

$$v_m(a_m) = \sum_{\mathbf{a}_{-m} \in \mathcal{A}_{-m}} R_m(a_m, \mathbf{a}_{-m}) \prod_{n \in \mathcal{M}, n \neq m} \pi_n(a_n) \quad (5.35)$$

for all $m \in \mathcal{M}$.

As shown in the proof of Theorem 5.3, for each a_m there exists some \mathbf{a}_{-m}^* such that $R'_m(a_m, \mathbf{a}_{-m}^*)$ in Algorithm 5.4 equals 1, while $R'_m(a_m, \mathbf{a}_{-m})$ equals zero for all other \mathbf{a}_{-m} . Consequently:

$$\sum_{\mathbf{a}_{-m} \in \mathcal{A}_{-m}} R_m(a_m, \mathbf{a}_{-m}) = \frac{1}{\prod_{n \in \mathcal{M}, n \neq m} \pi_n(a_n)} \quad (5.36)$$

for all a_m and all \mathbf{a}_{-m} , and where a_n is agent n 's action in \mathbf{a}_{-m} . Therefore, Eq. (5.35) holds for all a_m and all $m \in \mathcal{M}$ with:

$$v_m(a_m) = 1 = \max_{a'_m \in \mathcal{A}'_m} v_m(a'_m) \quad (5.37)$$

Therefore, the fully mixed joint action distribution $\pi_{\mathcal{M}}$ is the one and only Nash equilibrium of the reduced Γ' and each agent expects to earn a reward of one. Consequently, both the deterministic and the stochastic parts of the game form a unique Nash equilibrium where the agents follow their policies in the joint policy π and each agent earns an expected reward of one. \square

Theorem 5.6 leads us to the following conclusion about designing multi-agent normal form games:

Corollary 5.2. *If π^0 is a joint policy that satisfies Eq. (5.34), where π^0 is in B , the set of desired joint policies, and R^0 is the joint reward function constructed by Algorithm 5.4 with input π^0 , then R^0 is a design compliant reward function.*

5.4 Designing rewards for general stochastic games

We now combine our methods for designing rewards for Markov decision processes and normal form games to form a method for designing rewards for stochastic games. A stochastic game is defined as a tuple $\mathcal{G} = (\mathcal{N}, \mathcal{S}, \mathcal{A}, T, R)$ where \mathcal{S} is a set of states, \mathcal{A} is a joint action space, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition function and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{N}|}$ is a joint reward function. In addition, we consider that \mathcal{A}_n is the action space of agent n such that $\mathcal{A} = \times_{n \in \mathcal{N}} \mathcal{A}_n$, and that $R_n : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function for agent n . The goal of each agent is to control its actions so as to maximise its payoff. Each agent maintains a policy, π_n , that is a mapping between states and probability distributions over actions, that is, $\pi_n(s, a_n) \in [0, 1]$ and $\sum_{a_n \in \mathcal{A}_n} \pi_n(s, a_n) = 1$, for all $s \in \mathcal{S}$. As system designers, our goal is to design a reward function that results in a stochastic game where all Nash equilibria are desirable behaviours. For stochastic games, we quantify the concept of a ‘behaviour’ as a particular stationary stochastic joint policy, $\pi = (\pi_n \text{ for all } n \in \mathcal{N})$, where π_n is the policy of agent n .

5.4.1 Definitions

If we fix the agents’ joint policy, then a stochastic game collapses to a Markov chain. Before we begin our discussion of designing rewards for stochastic games, we require some definitions that relate to the dynamics of a stochastic game when we fix the agents’ joint policy at π :

- The state transition probability from state s to state s' is represented as $p(s'|s, \pi)$ and is calculated as:

$$p(s'|s, \pi) = \sum_{\mathbf{a} \in \mathcal{A}} T(s'|s, \mathbf{a}) \prod_{n \in \mathcal{N}} \pi_n(s, a_n) \quad (5.38)$$

such that a_n is agent n ’s component of the joint action vector \mathbf{a} .

- The state transition probability matrix $\mathbf{P}(\pi)$ has entries:

$$[\mathbf{P}(\pi)]_{ij} = p(s^j|s^i, \pi) \quad (5.39)$$

- Agent n 's expected immediate reward for each state is described by the vector:

$$\mathbf{r}_n(\pi) = \left(\sum_{\mathbf{a} \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s'|s, \mathbf{a}) R_n(s, \mathbf{a}, s') \prod_{m \in \mathcal{N}} \pi_m(s, a_m) \text{ for } s \in \mathcal{S} \right)^T \quad (5.40)$$

such that a_m is agent m 's component of the joint action vector \mathbf{a} . If the joint reward function has no dependency on the next state, s' , then the expected immediate reward vector can be simplified to:

$$\mathbf{r}_n(\pi) = \left(\sum_{\mathbf{a} \in \mathcal{A}} R_n(s, \mathbf{a}) \prod_{m \in \mathcal{N}} \pi_m(s, a_m) \text{ for } s \in \mathcal{S} \right)^T \quad (5.41)$$

- The value of state s to agent n is represented as $v_n(s, \pi)$ and is calculated as:

$$v_n(s, \pi) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t E_{s, \pi} [R_n(t)] \quad (5.42)$$

where, in an abuse of notation, $E_{s, \pi} [R_n(t)]$ is the expected reward to agent n at time t , starting from state s .

- The state values for an agent are described by the value vector:

$$\mathbf{v}_n(\pi) = (v_n(s, \pi) \text{ for } s \in \mathcal{S})^T \quad (5.43)$$

Given the definitions above, we can calculate the γ -discounted value vector for each agent from the expected immediate reward vector and the state transition matrix:

$$\mathbf{v}_n(\pi) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbf{P}^t(\pi) \mathbf{r}_n(\pi) \quad (5.44)$$

$$= (1 - \gamma) [\mathbf{I} - \gamma \mathbf{P}(\pi)]^{-1} \mathbf{r}_n(\pi) \quad (5.45)$$

Let π_{-n} represent an almost-joint policy that includes policies for all agents except agent n , and let $\pi = (\pi_n, \pi_{-n})$. A Nash equilibrium in a stochastic game is defined as a joint policy, π^* , such that no agent can increase its value vector by unilaterally changing its policy, i.e. the following holds element-wise:

$$\mathbf{v}_n[(\pi_n^*, \pi_{-n}^*)] \geq \mathbf{v}_n[(\pi_n, \pi_{-n}^*)] \quad (5.46)$$

for all π_n and all $n \in \mathcal{N}$. See also Definition 4.6.1 on p. 217 of Filar and Vrieze (1997). We can now formally define the reward design for stochastic games:

Definition 5.3. Suppose we have a stochastic game without a reward function, $\mathcal{G}^{-R} = (\mathcal{N}, \mathcal{S}, \mathcal{A}, T)$, and a set of desired behaviours, B . A reward function R^\dagger is design compliant

if and only if all stationary Nash equilibria of the stochastic game $(\mathcal{G}^{-R}, R^\dagger)$ are in the set of desired behaviours.

5.4.2 Rewarding agents for following a successful joint policy is a design compliant reward function

We can generate rewards for stochastic games by examining the normal form game at each state. By constructing a design compliant reward function for the normal form game at each state, and by setting the expected payoffs of each agent to one for each state, we can create a design compliant joint reward function for a stochastic game.

Theorem 5.7. *Let \mathcal{G}^{-R} be a stochastic game without a reward function and let $\pi^0 \in B$ be a desired joint policy such that Eq. (5.34) is satisfied when $\pi(s)$ replaces π , for all $s \in \mathcal{S}$. Let R^0 be a joint reward function with no dependency on the next state such that $R_n^0(s, \mathbf{a}, s') = R_n^0(s, \mathbf{a})$ for all $n \in \mathcal{N}$. If we set the values of R^0 as the output of Algorithm 5.4 with input $\pi(s)$ for each state, then R^0 is a design compliant reward function.*

Proof. We can be sure that at least one Nash equilibrium exists, since:

In a general-sum, discounted stochastic game Γ_β , there exists a Nash equilibrium in stationary strategies. (Filar and Vrieze, 1997, Theorem 3.8.1, p. 130)

First, we show that π^0 is a Nash equilibrium of $\mathcal{G} = (\mathcal{G}^{-R}, R^0)$, then we show that π^0 is the only Nash equilibrium of \mathcal{G} in the set of stationary strategies.

By Theorem 5.6, the reward functions produced by Algorithm 5.3 result in normal form games with a unique Nash equilibrium at the desired joint policy. We can infer the Nash equilibria of a stochastic game from the state transition function and the Nash equilibria of the agents' reward functions at each state. By Theorem 4.6.5 on p. 220 of Filar and Vrieze (1997), the joint policy π^0 is a Nash equilibrium of \mathcal{G} if the agents' policies at that state constitute a Nash equilibrium of a normal form game with joint reward function $R^{NFG(s)}$ for each state $s \in \mathcal{S}$. The reward functions of these normal form games, $R^{NFG(s)}$, can be calculated as:

$$R_n^{NFG(s)}(\mathbf{a}) = (1 - \gamma)R_n^0(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi^0) v_n(s', \pi^0) \quad (5.47)$$

for each state $s \in \mathcal{S}$, each agent $n \in \mathcal{N}$ and each joint action $\mathbf{a} \in \mathcal{A}$. (Filar and Vrieze state this theorem for two-agent games, but it can be extended to stochastic games with an arbitrary number of agents at the expense of more complex notation.)

To simplify Eq. (5.47), we apply similar reasoning as in the proof of Theorem 5.1 for Markov decision processes where we showed that the value at each state is equal to

a constant. Then, we show that $R_n^{NFG(s)}(\mathbf{a})$ is some affine transformation of $R_n^0(s, \mathbf{a})$. By Theorem 5.6, the expected reward for each agent is one, if all agents take actions according to the Nash equilibrium, that is, $\mathbf{r}_n(\pi^0) = \mathbf{1}_{|\mathcal{S}|}$ for all $n \in \mathcal{N}$. Therefore, by Eq. (5.45) and exactly the same reasoning as in the proof of Theorem 5.1, we know that $\mathbf{v}_n(\pi^0) = \mathbf{1}_{|\mathcal{S}|}$ for all $n \in \mathcal{N}$. Because $\sum_{s' \in \mathcal{S}} p(s'|s, \pi^0) v_n(s', \pi^0) = 1$, we can rewrite Eq. (5.47) as:

$$R_n^{NFG(s)}(\mathbf{a}) = (1 - \gamma)R_n(s, \mathbf{a}) + \gamma \quad (5.48)$$

If γ is a constant in $[0, 1)$, then we know that $R_n^{NFG(s)}(\mathbf{a})$ differs from $R_n(s, \mathbf{a})$ only by an affine transformation. Because Nash equilibria are invariant under affine transformations, this demonstrates that there exists a Nash equilibrium of \mathcal{G} where each agent takes actions according to π^0 .

To complete this proof, we show that π^0 is the only Nash equilibrium of \mathcal{G} . Suppose that \mathcal{G} has another Nash equilibrium $\pi^\times \neq \pi^0$. If π^\times is a Nash equilibrium, then Eq. (5.46) must be satisfied when we substitute π^\times in place of π^* . However, if $\pi^\times \neq \pi^0$, then there must be at least one state, s^\times , where at least one agent n can increase its expected payoff by changing its policy to π^0 . If there exists some a_n such that $\pi_n^0(s^\times, a_n) = 1$, then agent n can only decrease its expected payoff by deviating from π^0 . Alternatively, if no such a_n exists, then deviating from π^0 implies that the other agents can exploit agent n , causing agent n to receive an expected payoff of less than one. Because agent n cannot increase its expected payoff in some state and hence its value in that state, Eq. (5.46) cannot be satisfied when we substitute any value for π^* except π^0 . Therefore, π^0 is the only Nash equilibrium of \mathcal{G} in the set of stationary strategies. Because all stationary Nash equilibria of \mathcal{G} are in B , the set of desired behaviours, R^0 is a design compliant reward function. \square

Because R^0 in Theorem 5.7 does not depend on the next state and results in π^0 as the unique Nash equilibrium for all discount rates $\gamma \in [0, 1)$, R^0 satisfies the stochastic game analogues of the easy-learning properties given in Section 5.2.1. Therefore, we can encode desirable stationary stochastic joint policies as the unique Nash equilibrium of a stochastic game. This reduces the problem of choosing agent reward functions in stochastic games to finding a desirable joint behaviour.

5.4.3 An example

Here we demonstrate our method for designing rewards for stochastic games with an example. Suppose we have a multi-agent system with three agents, $\mathcal{N} = \{1, 2, 3\}$, where the agents play a variation of the medium access games described in Chapter 4. We model this multi-agent system as a stochastic game and provide an example reward function designed as in Theorem 5.7 that incentivises turn-taking. Unlike the

Table 5.2: One possible joint policy for a three-agent multi-agent system that results in high values of the turn-taking metric $\tau\tau$. The table shows the probability of the agent taking action b_n . Because each agent has two possible actions, the probability of action a_n is $\pi_n(s, a_n) = 1 - \pi_n(s, b_n)$.

State, s	Desired action probabilities		
	$\pi_1(s, b_1)$	$\pi_2(s, b_2)$	$\pi_3(s, b_3)$
$\langle a_1, a_2, a_3 \rangle$	0.5	0.5	0.5
$\langle a_1, a_2, b_3 \rangle$	0	1	0
$\langle a_1, b_2, a_3 \rangle$	1	0	0
$\langle a_1, b_2, b_3 \rangle$	0.5	0.5	0.5
$\langle b_1, a_2, a_3 \rangle$	0	0	1
$\langle b_1, a_2, b_3 \rangle$	0.5	0.5	0.5
$\langle b_1, b_2, a_3 \rangle$	0.5	0.5	0.5
$\langle b_1, b_2, b_3 \rangle$	0.5	0.5	0.5

medium access games of Chapter 4, in this situation we consider the state to be fully observable by the agents, in keeping with our formulation of a stochastic game.

On an indefinite number of discrete time steps, each agent can take one of two actions: $\mathcal{A}_n = \{a_n, b_n\}$ for all $n \in \mathcal{N}$, where action b_n means that agent n is transmitting on a shared communications channel and action a_n means that the agent is not transmitting. We define the system state as the joint action of all agents on the previous time step; therefore $\mathcal{S} = \mathcal{A} = \{a_1, b_1\} \times \{a_2, b_2\} \times \{a_3, b_3\}$ and the state transition probabilities are:

$$T(s, \mathbf{a}, s') = \begin{cases} 1 & \text{if } s' = \mathbf{a} \\ 0 & \text{otherwise} \end{cases} \quad (5.49)$$

for all $s, s' \in \mathcal{S} \times \mathcal{S}$ and all $\mathbf{a} \in \mathcal{A}$. Between, \mathcal{N} , \mathcal{S} , \mathcal{A} and T , we have a stochastic game without a reward function, \mathcal{G}^{-R} . Using Theorem 5.7, we can construct a reward function that results in any joint policy as the unique Nash equilibrium.

Suppose that we want the agents to take turns transmitting on the shared communications channel, that is, when we measure the mean turn-taking of the agents' actions using the turn-taking metric described in Chapter 3 and a resolution of $r = 3$, we should get a quantity of turn-taking that is greater than the expected turn-taking of a group of three random agents. So far, in this chapter, we assume that we know the set of desired behaviours. Therefore we consider that the desired behaviour is as described in Table 5.2, which shows the probability that agent n will take action b_n in state s , $\pi_n(s, b_n)$, for all states and all three agents. The joint policy shown in Table 5.2 results in $\lim_{t \rightarrow \infty} \tau\tau(t, 3) = 1$, for all starting states. We discuss constructing desirable joint behaviours from metrics or other non-explicit representations in Section 5.5.

In order to incentivise turn-taking with a unique Nash equilibrium of the stochastic game, we construct a reward function according to the scheme in Theorem 5.7 when

Table 5.3: Reward function values that result in turn-taking, written as $R_a(s, \mathbf{a}), R_b(s, \mathbf{a}), R_c(s, \mathbf{a})$.

State, s	Joint action, \mathbf{a}							
	$\langle a_1, a_2, a_3 \rangle$	$\langle a_1, a_2, b_3 \rangle$	$\langle a_1, b_2, a_3 \rangle$	$\langle a_1, b_2, b_3 \rangle$	$\langle b_1, a_2, a_3 \rangle$	$\langle b_1, a_2, b_3 \rangle$	$\langle b_1, b_2, a_3 \rangle$	$\langle b_1, b_2, b_3 \rangle$
$\langle a_1, a_2, a_3 \rangle$	0, 4, 0	0, 0, 4	4, 0, 0	0, 0, 0	0, 0, 0	4, 0, 0	0, 0, 4	0, 4, 0
$\langle a_1, a_2, b_3 \rangle$	1, 0, 1	1, 0, 0	1, 1, 1	1, 1, 0	0, 0, 1	0, 0, 0	0, 1, 1	0, 1, 0
$\langle a_1, b_2, a_3 \rangle$	0, 1, 1	0, 1, 0	0, 0, 1	0, 0, 0	1, 1, 1	1, 1, 0	1, 0, 1	1, 0, 0
$\langle a_1, b_2, b_3 \rangle$	0, 4, 0	0, 0, 4	4, 0, 0	0, 0, 0	0, 0, 0	4, 0, 0	0, 0, 4	0, 4, 0
$\langle b_1, a_2, a_3 \rangle$	1, 1, 0	1, 1, 1	1, 0, 0	1, 0, 1	0, 1, 0	0, 1, 1	0, 0, 0	0, 0, 1
$\langle b_1, a_2, b_3 \rangle$	0, 4, 0	0, 0, 4	4, 0, 0	0, 0, 0	0, 0, 0	4, 0, 0	0, 0, 4	0, 4, 0
$\langle b_1, b_2, a_3 \rangle$	0, 4, 0	0, 0, 4	4, 0, 0	0, 0, 0	0, 0, 0	4, 0, 0	0, 0, 4	0, 4, 0
$\langle b_1, b_2, b_3 \rangle$	0, 4, 0	0, 0, 4	4, 0, 0	0, 0, 0	0, 0, 0	4, 0, 0	0, 0, 4	0, 4, 0

π^0 is the desired joint policy in Table 5.2. Since Table 5.2 has four unique rows, we run Algorithm 5.4 with four different inputs. Table 5.3 shows the output of Algorithm 5.4 for each state. If we set R^0 to the entries of Table 5.3, then R^0 is a design compliant reward function for the reward-less stochastic game \mathcal{G}^{-R} , by Theorem 5.7.

If the agents play their actions according to Table 5.2 and they are rewarded according to Table 5.3, then each agent earns an expected payoff of 1 for each state (see Theorem 5.6). For the states $\langle a_1, a_2, b_3 \rangle$, $\langle a_1, b_2, a_3 \rangle$ and $\langle b_1, a_2, a_3 \rangle$, notice that Table 5.2 specifies that the agents should choose their actions deterministically and Table 5.3 shows that each agent can earn a reward of 1 by choosing its correct action, regardless of the other agents' actions. For the other states, Table 5.2 specifies that the agents should randomise equally between their two actions, so each of the eight joint actions occur with probability $1/8$. Therefore, the sum of each agent's reward function in Table 5.3 must be 8 for each of these states in order for each agent to earn an expected reward of 1. Indeed, each agent can earn a reward of 4 for two different outcomes in states $\langle a_1, a_2, a_3 \rangle$, $\langle a_1, b_2, b_3 \rangle$, $\langle b_1, a_2, b_3 \rangle$, $\langle b_1, b_2, a_3 \rangle$ and $\langle b_1, b_2, b_3 \rangle$. However, because the normal form games at these states are intransitive, no joint action can give a non-zero reward to all agents.

We simulated groups of three agents engaged in this medium access game-like stochastic game. We did two simulations: one with ideal agents that follow the stochastic stationary policies shown in Table 5.2 and one with Q-learning agents (Watkins, 1989) that follow an ϵ -greedy exploration policy (Sutton and Barto, 1998). We configured the Q-learning agents with the parameters shown in Table 5.4. Chapter 2 describes Q-learning and ϵ -greedy exploration policies in detail.

Table 5.5 shows an example sequence of actions for three ideal agents. The agents are ideal in the sense that their policies are determined by the Nash equilibrium and we show their performance so that we can compare them to Q-learning agents. The ideal agents move randomly between states until time step 4, when they settle into perfect turn-taking for all later time steps. Fig. 5.4 shows the mean turn-taking at a

Table 5.4: Q-learning agent parameter values.

Parameter name	Domain	Value
Learning rate	$[0, 1]$	0.2
Exploration amount, ϵ	$[0, 1]$	0.01
Discount factor, γ	$[0, 1)$	0.9
Initial Q-table value	\mathbb{R}	100

Table 5.5: An example sequence of actions for three ideal agents.

t	State, s	Joint action, \mathbf{a}	Rewards, R_a, R_b, R_c	$\tau\tau(t, 3)$
0	$\langle a_1, a_2, a_3 \rangle$	$\langle a_1, a_2, a_3 \rangle$	0, 4, 0	0.0
1	$\langle a_1, a_2, a_3 \rangle$	$\langle b_1, b_2, a_3 \rangle$	0, 0, 4	0.0
2	$\langle b_1, b_2, a_3 \rangle$	$\langle b_1, b_2, a_3 \rangle$	0, 0, 4	0.0
3	$\langle b_1, b_2, a_3 \rangle$	$\langle b_1, a_2, b_3 \rangle$	4, 0, 0	0.0
4	$\langle b_1, a_2, b_3 \rangle$	$\langle b_1, a_2, a_3 \rangle$	0, 0, 0	1.0
5	$\langle b_1, a_2, a_3 \rangle$	$\langle a_1, a_2, b_3 \rangle$	1, 1, 1	1.0
6	$\langle a_1, a_2, b_3 \rangle$	$\langle a_1, b_2, a_3 \rangle$	1, 1, 1	1.0
7	$\langle a_1, b_2, a_3 \rangle$	$\langle b_1, a_2, a_3 \rangle$	1, 1, 1	1.0

resolution of 3, $\tau\tau(t, 3)$, and the mean reward for 1000 simulation runs of the ideal agents that play the policy shown in Table 5.2. Notice that the mean reward is close to 1 for the ideal agents, which is what we expect because Theorem 5.6 guarantees that the mean reward at each state is 1 if the agents play the Nash equilibrium.

Fig. 5.5 shows similar plots for simulations with Q-learning agents. Notice that the x-axis limit for the time variable is much larger in Fig. 5.5 than in Fig. 5.4; the ideal agents converge to perfect turn-taking faster than the Q-learning agents, on average. The mean turn-taking of the Q-learning agents increases as they learn to maximise their reward functions. As noted in Chapter 4, Q-learning agents are not guaranteed to converge to the Nash equilibrium in stochastic games (Buşoniu et al., 2008). In fact, the Q-learning agents have not learned the policy shown in Table 5.2, but they have learned the deterministic parts in states $\langle a_1, a_2, b_3 \rangle$, $\langle a_1, b_2, a_3 \rangle$ and $\langle b_1, a_2, a_3 \rangle$ that are necessary to coordinate turn-taking. Even if the Q-learning agents may not have learned the Nash equilibrium, we can be confident that the Q-learning agents have indeed learned turn-taking because their mean turn-taking values greatly exceed 0.019, the expected turn-taking at a resolution of 3 for groups of three random agents. Unlike the examples in Chapter 3, we are measuring the quantity of turn-taking present at a resolution equal to the number of agents. This generally implies that the turn-taking of random agents will be small, as shown in Table 3.3 in Chapter 3 on p. 39.

This example serves to illustrate how our method can work in practice to design rewards that result in desirable joint sequential behaviours in groups of agents. By using Theorem 5.7 to construct a reward function that results in turn-taking behaviour as the unique Nash equilibrium, we can influence the outcome produced by groups

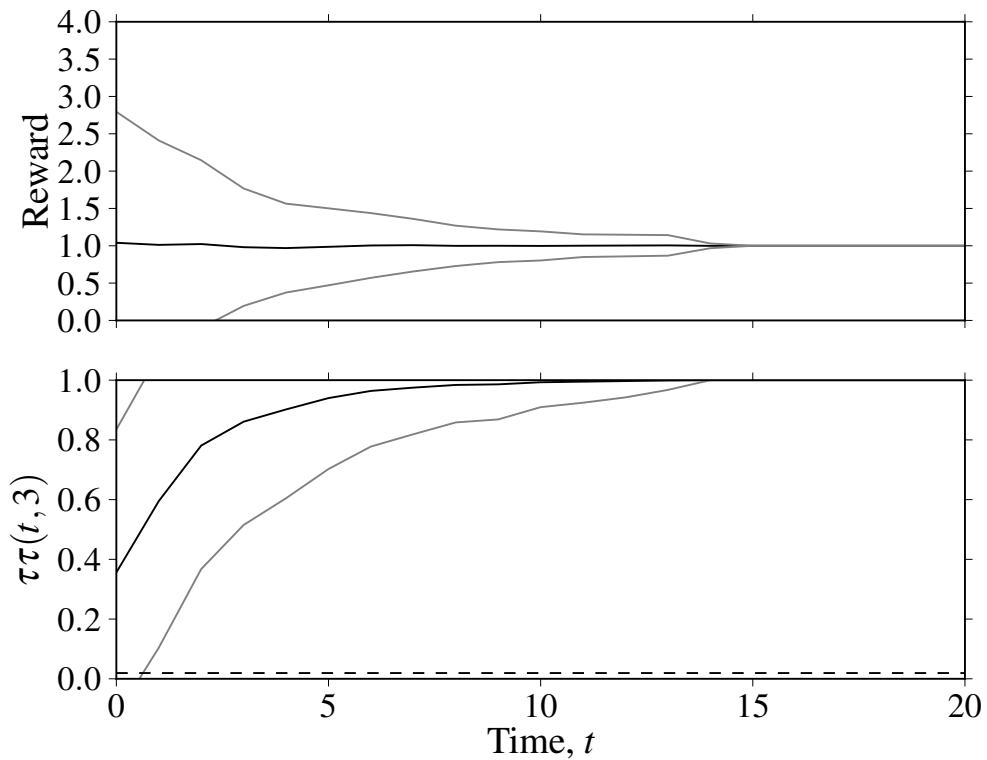


Figure 5.4: The reward (top) and turn-taking at a resolution of 3 (bottom) for groups of three simulated ideal agents playing the policy shown in Table 5.2 that are rewarded as in Table 5.3. This figure shows mean values for 1000 simulation runs in black and the mean plus and minus one standard deviation in gray. A dashed line shows the expected turn-taking at a resolution of 3 for groups of three random agents, which is 0.019. Note that the x-axis limit for the time variable is 20, much smaller than in Fig. 5.5.

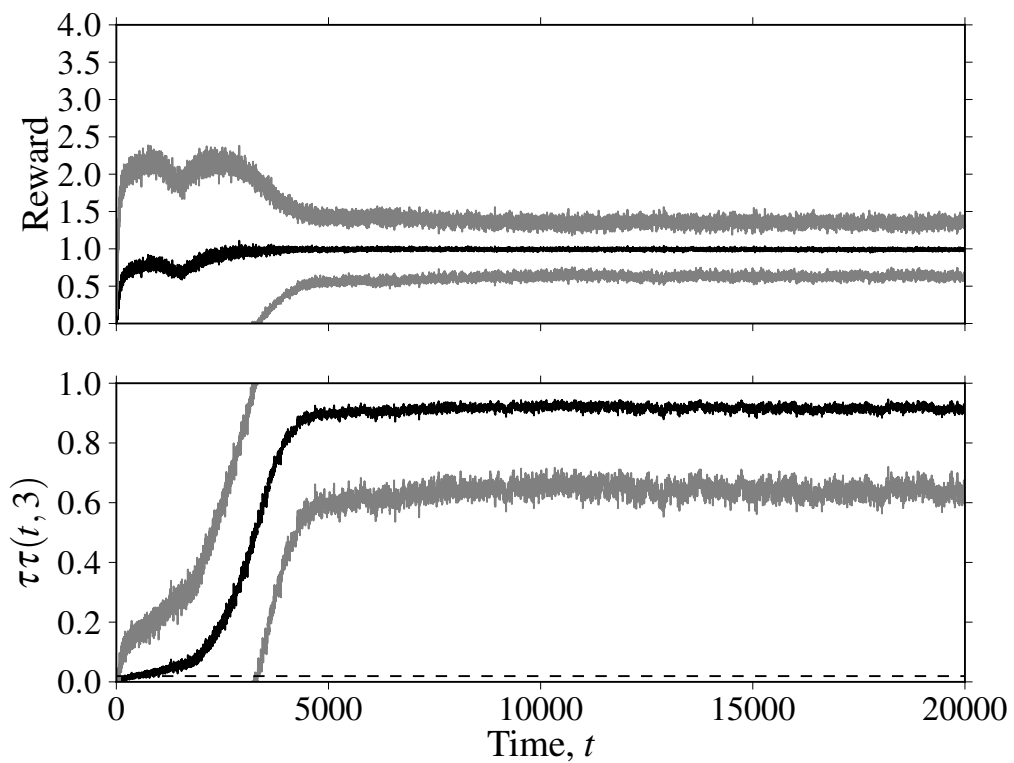


Figure 5.5: The reward (top) and turn-taking at a resolution of 3 (bottom) for groups of three simulated Q-learning agents rewarded as in Table 5.3. This figure shows mean values for 1000 simulation runs in black and the mean plus and minus one standard deviation in gray. A dashed line shows the expected turn-taking at a resolution of 3 for groups of three random agents, which is 0.019. Note that the x-axis limit for the time variable is 20000, much larger than in Fig. 5.4.

of Q-learning agents, as illustrated in Fig. 5.5. Notice that this example is similar to the work in Chapter 4; however, one important difference is that the medium access games in Chapter 4 are partially observable so the method in this chapter, which is for fully observable stochastic games, does not directly apply. We discuss the general differences between the methods of this chapter and those of Chapter 4 in Section 5.5.

5.5 Discussion

Our method is one of several different possible ways to design reward functions that result in games with unique Nash equilibria. While we generalise on the normal form game of Matching Pennies, we could instead design reward functions based on rock-paper-scissors. Rock-paper-scissors is a zero-sum game that people play by simultaneously presenting a hand gesture that symbolises an action. The first three rows and columns of Table 5.6 show the payoffs of rock-paper-scissors. Rock crushes scissors, but is covered by paper, while scissors cuts paper, forming an intransitive loop of preferences. One popular generalisation of rock-paper-scissors to five actions is the game invented by Sam Kass and Karen Bryla called ‘Rock-paper-scissors-spock-lizard’ (Kass, 2012). Spock (Roddenberry, 1966) dominates rock and scissor but is poisoned by the lizard and is disproven by paper while the lizard eats paper but is killed by rock or scissors.

Table 5.6: Generalised rock-paper-scissors with an odd number of actions. The reward function values are shown in the form of reward for row agent, reward for column agent.

		Column agent					
		Rock	Paper	Scissors	...	Spock	Lizard
Row agent	Rock	0, 0	-1, 1	1, -1	...	-1, 1	1, -1
	Paper	1, -1	0, 0	-1, 1	...	1, -1	-1, 1
	Scissors	-1, 1	1, -1	0, 0	...	-1, 1	1, -1
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	Spock	1, -1	-1, 1	1, -1	...	0, 0	-1, 1
	Lizard	-1, 1	1, -1	-1, 1	...	1, -1	0, 0

We can extend rock-paper-scissors to any odd number of actions with payoffs as given in Table 5.6 by putting any even number of extra rows and columns in place of the dots. Each action must defeat half of the total actions, be defeated by the other half of the actions and tie against itself. Generalised rock-paper-scissors has a single Nash equilibrium where both agents choose each action with equal probability. Many different possible reward functions share the same set of Nash equilibria; which reward functions are most learnable? This is an open question. We have a number of clues, including the easy-learning properties we introduce here, in addition to some

previously investigated measures, such as ‘factoredness’ and ‘learnability’ defined by Agogino and Tumer (2008).

A key assumption of our method is that we know the set of desired system behaviours and that the desirability of a given behaviour is binary. Sometimes a system designer will have to construct the set of desired system behaviours from a less explicit representation of the desired behaviour, perhaps using techniques such as:

- Linear temporal logic (Pnueli, 1977; Kröger and Merz, 2008; Wongpiromsarn et al., 2012).
- A classifier function for Markov chains where all policies that reduce the Markov decision process into a Markov chain that exhibits particular properties are in B .
- A continuous-valued results-based metric, such as the turn-taking metric $\tau\tau$ described by Raffensperger et al. (2012b) and in Chapter 3.

Given a description of the desired behaviour, a system designer would have to construct the set of desired policies. However, some system behaviours are measured with real values rather than a binary absence or presence. One approach is to create a binary abstraction of the real value by finding some satisfactory threshold for the real value, for example, accepting a behaviour if the quantity of turn-taking measured by $\tau\tau(t,r)$ is greater than some value. Alternatively, a system designer could find desirable system behaviours with a black-box optimisation technique such as a genetic algorithm (Goldberg, 1989).

Our method constructs games where the desired behaviour is a Nash equilibrium. However, the desired behaviour will only be present in the final system if all the agents converge to the Nash equilibrium. Not all reinforcement learning algorithms are guaranteed to converge to a Nash equilibrium, for example, Q-learning with an ϵ -greedy exploration policy (Sutton and Barto, 1998) is unable to converge to a mixed Nash equilibrium. One area of future work is to test the empirical learnability of reward functions produced with our method.

An important focus of multi-agent reinforcement is to design learning algorithms that guarantee convergence to a Nash equilibrium under some conditions, such as self-play (Buşoniu et al., 2008). Some research has been done on the conditions required for learning agents to converge to a Nash equilibrium (Chen and Gazzale, 2004), while other research focuses on situations involving people (Aumann and Brandenburger, 1995; Mailath, 1998). A potentially interesting research question is: given a set of multi-agent reinforcement learning algorithms, how can we predict the distribution of outcomes without performing a complete simulation? Some work has already been done examining the dynamics of Q-learning (Tuyls et al., 2006); perhaps a

similar methodology could be applied to other algorithms, possibly leading to better algorithms.

In some situations, the system designer is not entirely in control of the agents' reward functions. A constraint on one or more of the agents' reward functions will decrease the range of possible system behaviours. One area of future work is to investigate designing rewards for desirable behaviours in the presence of constraints on the agents' reward functions. In the limiting case where the agents' reward functions are fully specified and may be unknown, a system designer must resort to algorithmic mechanism design (Conitzer and Sandholm, 2002; Sandholm, 2003). We discuss mechanism design in more detail in Section 5.6.

In a number of practical situations, each agent can only partially observe the system state. For example, urban search and rescue teams usually possess a small fraction of the total state information; consequently robot design and modelling for urban search and rescue must consider partial state observability (Davids, 2002; Carpin et al., 2007). In general, the theoretical framework of partially observable stochastic games leads to more accurate models than would be found in the set of fully observable stochastic games. However, optimal algorithms for partially observable stochastic games are computationally intractable in general (Bernstein et al., 2002). In Chapter 4 and in the published article by Raffensperger et al. (2012a), we discuss designing rewards conducive for agents playing medium access games, which are partially observable games. In an extensive search of the reward function space, the best predictor for the desired behaviour was the Nash equilibrium of the analogous fully observable game. Designing learnable rewards for partially observable games is a future research area. As suggested by our work in Chapter 4, it may be possible to simply ignore partial state observability in some cases. Agents may be able to converge to a Nash equilibrium without fully observing the state. A productive area of research may be to systematically investigate the consequences of ignoring partial state observability when designing multi-agent reward functions.

In Chapter 4 and in the article by Raffensperger et al. (2012a), we outline a methodology for designing multi-agent rewards. The method here builds on that methodology, but there are some differences. In Chapter 4, we used Q-learning with an ϵ -greedy exploration policy and we explicitly limited the range of possible joint agent policies to a finite set. We used the turn-taking metric $\tau\tau$ (Raffensperger et al., 2012b) to identify the set of desirable policies in an exhaustive search. We used Markov chain techniques to calculate the agents' limit average rewards for all possible stationary policies, given that the agents' used ϵ -greedy exploration policies. At this point, our method for designing rewards for stochastic games differs, because we use the γ -discounted valuation in this chapter, rather than the limit average valuation. The limit average valuation is similar to the γ -discounted valuation when γ approaches 1 (Filar and

Vrieze, 1997, See Theorem 4.3.18 on p. 195) and the choice in Chapter 4 is empirically justified by the results in that chapter. However, optimal stationary strategies do not always exist when we use limit average valuation for stochastic games, for example, see the stochastic game called ‘The Big Match’ (Gillette, 1957; Blackwell and Ferguson, 1968). Consequently, we have chosen to base our general method on the γ -discounted valuation because it always lets us find optimal strategies in the set of stationary policies.

An important part of the method in Chapter 4 is to account for the effects of the agents’ exploration policies when computing the agents’ valuations of different policies. We could generalise our reward design method to explicitly cater for the agents’ exploration policies. For example, we could modify our method to suit Markov decision processes where the agent follows an ϵ -greedy policy. An agent that follows an ϵ -greedy exploration policy takes its best known action with probability $1 - \epsilon$ and takes a random action otherwise. To cater for an ϵ -greedy policy with constant ϵ , we could redefine the set of desired behaviours for the reward-less Markov decision process with a state transition function calculated as:

$$T^\epsilon(s, a, s') = (1 - \epsilon)T(s, a, s') + \frac{\epsilon}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} T(s, a', s') \quad (5.50)$$

for all $s, s' \in \mathcal{S} \times \mathcal{S}$ and all $a \in \mathcal{A}$. When we construct the reward function, we must also consider how the agent’s reward function will be affected:

$$R^\epsilon(s, a) = (1 - \epsilon)R(s, a) + \frac{\epsilon}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} R(s, a') \quad (5.51)$$

We control the reward function $R(s, a)$, but we want to dictate the optimal policy when the agent is rewarded according to $R^\epsilon(s, a)$.

In a similar way, we could modify our reward design method to suit stochastic games where each agent follows an ϵ -greedy policy. We would re-write the state transition function as:

$$T^\epsilon(s, \mathbf{a}, s') = \sum_{\mathcal{M} \in \wp(\mathcal{N})} \left[\binom{|\mathcal{N}|}{|\mathcal{M}|} \frac{(1 - \epsilon)^{|\mathcal{M}|} \epsilon^{|\mathcal{N}| - |\mathcal{M}|}}{\prod_{m \in \mathcal{M}} |\mathcal{A}_m|} \sum_{\mathbf{a}_{\mathcal{M}} \in \times_{m \in \mathcal{M}} \mathcal{A}_m} T[s, (\mathbf{a}_{-\mathcal{M}}, \mathbf{a}_{\mathcal{M}}), s'] \right] \quad (5.52)$$

for all $s, s' \in \mathcal{S} \times \mathcal{S}$ and all $\mathbf{a} \in \mathcal{A}$ where $\wp(\mathcal{N}) = \{\emptyset, \{n\}, \{m\}, \dots, \mathcal{N}\}$ is the power set of \mathcal{N} , $\mathbf{a}_{\mathcal{M}}$ is a vector of agent actions for all agents in \mathcal{M} and $\mathbf{a}_{-\mathcal{M}}$ is a vector of agent actions in \mathbf{a} for all agents not in \mathcal{M} . The agents’ exploratory actions are independent, so the total number of exploratory actions has a binomial distribution. In essence, Eq. (5.52) reweights the state transition function by examining each possible combination of agents that could be making exploratory moves on each state transition and weighting each combination of exploratory moves with a probability taken from

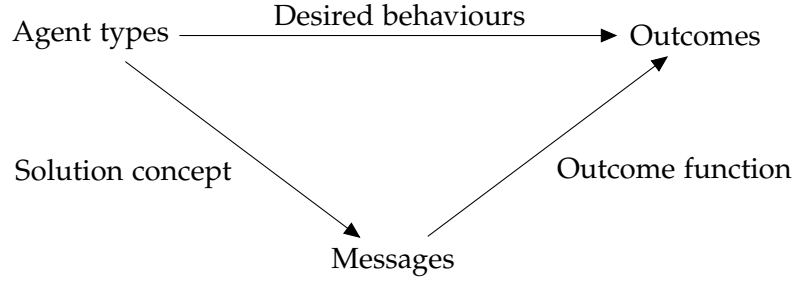


Figure 5.6: A Mount-Reiter diagram illustrates the processes of mechanism design (Mount and Reiter, 1974).

a binomial distribution. When we design a reward function for a stochastic game played by agents with ϵ -greedy policies, we can control $R(s, \mathbf{a})$ but the agents learn according to a modified reward function:

$$R^\epsilon(s, \mathbf{a}) = \sum_{\mathcal{M} \in \wp(\mathcal{N})} \left[\binom{|\mathcal{N}|}{|\mathcal{M}|} \frac{(1-\epsilon)^{|\mathcal{M}|} \epsilon^{|\mathcal{N}|-|\mathcal{M}|}}{\prod_{m \in \mathcal{M}} |\mathcal{A}_m|} \sum_{\mathbf{a}_{\mathcal{M}} \in \times_{m \in \mathcal{M}} \mathcal{A}_m} R[s, (\mathbf{a}_{-\mathcal{M}}, \mathbf{a}_{\mathcal{M}})] \right] \quad (5.53)$$

In principle, it is not more complicated if each agent has different value of ϵ , but the equations become unwieldy. We leave the derivation of reward design methods that account for the agents' exploratory moves to future work.

Our method for designing rewards for stochastic games is an important step in fully generalising the methodology given in Chapter 4, but further research is required for a complete methodology for designing multi-agent rewards. In particular, a better method for designing multi-agent rewards would account for partial state observability and agent exploration. Such a reward design method should be married to systematic techniques for measuring and specifying desirable multi-agent behaviours, a generic method for designing agent states and actions, and a meta-learning algorithm to select agent learning algorithms. Furthermore, despite the theoretical guarantees given here, empirical verification of our reward design method is essential.

5.6 Related work

'Mechanism design' is the field in game theory that was the subject of the 2007 Nobel prize in economics awarded to Leonid Hurwicz, Eric Maskin and Roger Myerson. Mechanism design is the study of constructing games that result in some desirable outcome, even when the agents act selfishly. Mechanism design is similar to our methods for designing agent reward functions, but starts from different assumptions. Unlike mechanism design, our methods assume that the agents have no intrinsic preferences. The process of mechanism design is illustrated by the Mount-Reiter

diagram (Mount and Reiter, 1974) in Fig. 5.6. Each agent has a particular ‘type’ that represents the agent’s utility function, which may be private to the agent. Each agent takes actions called ‘messages,’ these may be bids in an auction, for example. The ‘solution concept’ is a way of determining the expected outcome that results from competition among the agents; sometimes this is a form of Nash equilibrium, but other solution concepts can be used. The mechanism designer wants to enforce a particular class of desirable joint behaviours, for example, the designer may want to maximise the sum of the agents’ utilities (the social welfare). The mechanism designer constructs an outcome function that assigns outcomes based on the agents’ messages such that the Nash equilibrium (or other solution concept) of the game results in desirable agent behaviours (Maskin, 1985; Palfrey, 2002).

One important result in mechanism design is the ‘revelation principle:’ in general, we can design mechanisms where the agents’ best policies are to report their type truthfully (Mas-Colell et al., 1995). This implies that the agents cannot manipulate the mechanism by sending a message that lies about their type. The Vickrey-Clarke-Groves auction is an important mechanism that maximises social welfare by motivating agents to be truthful and has applications, for example, in public choice problems and single-good auctions (Vickrey, 1961; Clarke, 1971; Groves, 1973). For each agent, the Vickrey-Clarke-Groves mechanism re-aligns the agent’s value of each outcome with the social welfare in that outcome by penalising agents for their negative effect on the other agents’ values. Mechanism design can be automatically achieved in a wide range of application areas (Conitzer and Sandholm, 2002; Sandholm, 2003) and algorithms can be implemented with mechanism design principles (Nisan and Ronen, 1999). Our reward design method is most closely related to automated mechanism design by a self-interested designer that desires a particular outcome or distribution over outcomes. However, rather than being indifferent to the agents’ preferences, we consider that the agents have no prior preferences and that we dictate the agent’s preferences entirely. Our method is applicable to areas like designing teams of autonomous robots where a special case of automated mechanism design could be applied, but perhaps at a high computational cost (Conitzer and Sandholm, 2002).

Our method for constructing two-agent games with unique arbitrary Nash equilibria is to perturb a generalisation of Matching Pennies and subsequently add whatever dominated actions are required. Some extensions of Matching Pennies are discussed by von Neumann and Morgenstern, but only for two-agent games with two or three actions per agent (von Neumann and Morgenstern, 1944, §18.4-18.5). Kreps gave necessary and sufficient conditions for constructing normal form games with unique Nash equilibria for pairs of agents (Kreps, 1974) and arbitrary numbers of agents (Kreps, 1981), but Kreps did not give any methods for constructing such games. Quintas (1988) describes how to construct a two-agent game with a single arbitrary

Nash equilibrium using the extreme action probability points characterised by Marchi and Quintas (1987) and carefully chosen bijective functions to control the agents' action preferences. Quintas's action preference functions are analogous to our agent action preference graphs. Consequently, our graph method in Section 5.3.2 describes the set of functions that make Quintas's method work. Quintas suggests that his results can be generalised to games with more than two agents, but gives no explicit method. We go beyond Quintas's results for two-agent games by constructing reward functions that result in arbitrary unique Nash equilibria in games with an arbitrary number of agents and to stochastic games.

Wolpert et al. (1999) demonstrate the Collective INtelligence (COIN) method of designing agents' reward functions in multi-agent reinforcement learning systems. The COIN method is to design local rewards for each agent based on the 'Wonderful Life' utility which tries to account for the interactions between agents and requires minimal communication between agents. The Wonderful Life utility is named after the 1946 movie directed by Frank Capra where James Stewart plays the protagonist who is showed what his town would be like if he had never lived (Capra, 1946). In a similar way, the Wonderful Life utility shapes an agent's reward function based on what the group of agents would achieve without the influence of that agent; in this way the Wonderful Life utility bears some similarity to the Vickrey-Clarke-Groves mechanism. The COIN method belongs to the larger field of reward shaping techniques. Reward shaping has been productively applied to air traffic control (Tumer and Agogino, 2007), opportunistic spectrum access wireless communications (NoroozOliaee et al., 2012) and coordinated, scalable robust multi-agent systems (Tumer, 2006). Another reward shaping technique is to use a Kalman filter to estimate each agent's own contribution to the reward signal based on the assumption that the agent's reward signal is a linear sum of a useful reward signal and an irrelevant, random reward (Chang et al., 2004). Like our methods, the COIN method and related reward shaping techniques assume that the agents' reward functions can be directly manipulated by the system designer. Our method is to explicitly design a joint reward function that results in a desirable Nash equilibrium while reward shaping techniques modify each agents' reward function so that the emergent behaviour of the agents maximises some system-level reward function.

Boutilier (1999) considers a way of giving agents the ability to explicitly reason about 'coordination mechanisms' for resolving sequential coordination problems in cooperative multi-agent systems. By encoding the probabilities of coordination and mis-coordination into the agents' states (given some coordination mechanism in each agent), the state-action values can be more accurately computed and the agents can explicitly reason about coordination with the dynamic programming algorithm that Boutilier describes. Rather than adjusting the agents' learning algorithms, our

approach is to induce coordination by designing a reward function for each agent such that the resulting game has a single Nash equilibrium, eliminating the need for the agents to select between multiple equilibria.

Agogino and Tumer (2008) present a method for visualising and analysing agent rewards which can be used to speed up agent learning and enable coordination. Agogino and Tumer explicitly measure the ‘factoredness’ of a reward function, or the extent to which an agent’s reward function aligns its interests with some global reward function, and the ‘learnability’ of a reward function, or the extent to which an agent’s reward function depends on that agent’s own state. Agogino and Tumer help system designers find better reward functions by demonstrating how to visualise the factoredness and learnability of reward functions. While Agogino and Tumer are also motivated by trying to develop better tools for reward design in multi-agent systems, we take a different approach. We focus on algorithmic techniques for designing rewards that achieve particular game-theoretic ends, with some interest in reward learnability, rather on developing ways to analyse and visualise multi-agent reward functions.

5.7 Conclusion

We describe a method for designing joint reward functions for stochastic games that encode desirable sequential multi-agent behaviours. Given a set of desirable stationary stochastic joint agent policies and a stochastic game without a reward function, our method constructs a joint reward function that results in a desirable joint agent policy as the unique Nash equilibrium of the resulting stochastic game. Therefore, our method reduces the problem of designing reward functions for stochastic games to identifying a desirable multi-agent behaviour. Our method builds on simpler methods for designing rewards for Markov decision processes and for designing rewards for normal form games. We formally prove that our methods are correct and we discuss alternative techniques.

This chapter supports the thesis by providing a method for algorithmically designing rewards for stochastic games, assuming that the set of desired behaviours is known. While learning agents do not always converge to the Nash equilibrium, a unique Nash equilibrium represents the fixed point in the field of influence that a reward function exerts. Therefore, this chapter demonstrates a way to algorithmically design reward functions to influence groups of learning agents toward desirable sequential joint behaviours.

Chapter 6

Conclusion

This dissertation comprises a number of novel contributions to human knowledge that together demonstrate that:

Algorithmically designed reward functions can influence groups of learning agents toward measurable desired sequential joint behaviours.

These contributions to multi-agent systems and machine learning are as follows:

- In Chapter 3, we introduce a quantitative measure for turn-taking, $\tau\tau$. We derive the quantity of turn-taking present in groups of random agents, demonstrate $\tau\tau$ on previously reported examples of emergent turn-taking and apply $\tau\tau$ to a recorded human conversation.
- In Chapter 4, we define ‘medium access games,’ analyse these games and present simulation results to demonstrate how to identify reward functions that are conducive (or prohibitive) to the emergence of turn-taking in pairs of Q-learning agents.
- In Chapter 5, we propose a method for designing reward functions for Markov decision processes, normal form games and stochastic games that results in a desirable predetermined unique Nash equilibrium (or optimal policy). We formally prove that our methods are successful and show an example of algorithmically designing a reward function to achieve a desired sequential joint behaviour.

6.1 How it all fits together

The idea of designing reward functions to influence groups of learning agents towards desired sequential joint behaviours is built on research in reinforcement learning,

multi-agent systems, game theory and emergent phenomena. While an agent may be a person or an animal, we focus on artificial agents that use machine learning to maximise a scalar reward function. We suggest ways to measure and influence sequential joint agent behaviours to address difficulties in designing the agents' reward functions in multi-agent reinforcement learning systems. Therefore, the thesis belongs to the design agenda of multi-agent reinforcement learning (Gordon, 2007).

We demonstrate the thesis in three parts. In Chapter 3, we develop a metric for turn-taking, one class of sequential joint agent behaviour. The thesis relates to measurable behaviours in general; we pay special attention to turn-taking in order to present a detailed, concrete example rather than presenting a more abstract analysis of the general case of sequential joint behaviours. We validate our turn-taking metric by analysing previously reported cases of emergent turn-taking and we demonstrate our metric on a recorded human conversation. Chapter 3 supports the thesis by demonstrating how to measure an example sequential joint behaviour, but, more importantly, Chapter 3 lays the foundations for Chapter 4.

Chapter 4 continues the focus on turn-taking, and demonstrates how appropriate reward functions can influence the sequential joint behaviour in pairs of learning agents toward a desired behaviour. Chapter 4 introduces 'medium access games' to model a class of situations where human and machine turn-taking may be a useful joint sequential behaviour. We use extensive simulations to show that groups of learning agents can be incentivised to produce turn-taking behaviour by reward functions where the Nash equilibria of the resulting medium access games correspond to joint policies that would produce turn-taking. Chapter 4 presents a methodology for analysing the reward functions of multi-agent systems. We can use this methodology to speed up searches for reward functions that result in desirable sequential joint behaviours.

In Chapter 5, we focus on developing the methodology from Chapter 4 and we introduce a method for designing reward functions for stochastic games that result in unique Nash equilibria. In the process of developing a reward design method for stochastic games, we introduce methods for designing reward functions for Markov decision processes and normal form games. While Chapter 4 supports the thesis by connecting reward functions to desired multi-agent behaviours, Chapter 5 presents a way to algorithmically design reward functions that incentivise agents to produce arbitrary desirable joint sequential behaviours.

Chapters 3–5 support the thesis with original research. The ideas in Chapters 3–5 are internally supported by analysis, simulation and formal proofs. This dissertation as a whole demonstrates the validity of the thesis, with particular focus on turn-taking in

Chapters 3 and 4 while also addressing the general case of arbitrary sequential joint behaviours in Chapter 5.

6.2 Discussion

The discussion sections of Chapters 3, 4 and 5 consider alternative approaches and extensions of the ideas introduced in each chapter. In this section, we discuss the significance and limitations of the work as a whole.

In some cases, if we knew what we wanted the agents to do, then we would not need learning agents. However, learning agents are necessary in some situations even when we thoroughly specify the desired behaviour of the agents. Sometimes the details of a situation or environment change in such a way that the agents must change what they do in order to maintain the same outcome. For example, if three identical robot agents are tasked with cleaning a certain area and one robot breaks down, then the other two robots might be able to succeed at the joint goal if they can adapt what they do to their new situation.

In multi-agent systems, we often find that reward functions cannot completely control agents and that agents do not always converge to the Nash equilibrium. Indeed, in Chapter 4 we find that Q-learning agents sometimes do not converge to the Nash equilibrium. However, the thesis does not claim that algorithmically designed reward functions can completely control the behaviour of agents but rather that such reward functions can influence agents towards desired behaviours. The conditions required for convergence to the Nash equilibrium have been studied previously (Aumann and Brandenburger, 1995; Mailath, 1998; Chen and Gazzale, 2004) and convergence is an important goal for designers of multi-agent reinforcement learning algorithms (Buşoniu et al., 2008). We are optimistic that the algorithms employed by future machine learning agents will be more able to converge to Nash equilibrium than present algorithms.

In some cases, we may have a global utility function, so all we need is for the agents to maximise that function. There are existing methods for shaping agents' reward functions that make the agents maximise a global utility function (Wolpert et al., 1999; Tumer, 2006; Tumer and Agogino, 2007; NoroozOliaee et al., 2012). In this dissertation, we focus on those cases where the system designer is not endowed with a scalar global reward function and cases where that global reward function cannot be efficiently communicated to all the agents. In Chapter 4, we consider how to design local reward functions that result in turn-taking as a system behaviour. Turn-taking may be necessary to mediate communication between agents in some cases and therefore

any knowledge of a global reward function would need to be transmitted between the agents after they have learned turn-taking.

For Chapters 3 and 4, rather than approaching the general case of sequential joint behaviour in an abstract sense, we have chosen to use a concrete example. Turn-taking is a non-homogenous coordinated sequential joint behaviour that is intuitively comprehensible, making it an ideal test case for general sequential joint behaviours. Furthermore, turn-taking is an interesting behaviour in linguistics (Sacks et al., 1974; Stivers et al., 2009), in emergent communication (Di Paolo, 2000; Iizuka and Ikegami, 2004) and in human-machine interaction (Turunen et al., 2006; Raux and Eskenazi, 2008; Chao and Thomaz, 2010, 2012). However, the thesis refers to sequential joint behaviour in general, therefore Chapter 5 addresses the case of arbitrary sequential joint behaviours.

Rather than developing complicated models, we study simpler models in more detail. Simple models do not necessarily imply simple behaviour because the coupling between many simple parts can result in complex behaviour. For example, the cellular automaton known as Rule 110 is Turing complete (Cook, 2004). In Chapter 3, we focus on developing a simple but usable metric for turn-taking rather than a general metric. In Chapter 4, we focus on the aspects of human and machine interaction that are most relevant to turn-taking in order to simplify our analysis and to make it comprehensible. We ignore the possibility of partially observable states in Chapter 5 to simplify our methods and proofs. In each case, we focus on the details that we deem most important to the situation.

There may exist ways to influence the sequential joint behaviour in groups of learning agents even when the desired behaviour is *not* measurable. However, we have chosen to limit the scope of the thesis to measurable behaviours. Our methodology addresses the possibility of currently unmeasurable behaviours by suggesting the construction of new metrics. In particular, we demonstrate how to quantify and measure turn-taking with a simple metric.

6.3 Future work

The work presented in this dissertation opens a range of new research opportunities. In Chapters 3, 4 and 5, we mention opportunities for future work that relate to each chapter's content. Here, we discuss future research directions that relate to the dissertation as a whole. These directions are opportunities to broaden the research that comprises this dissertation, opportunities to extend that research and possible ways to apply that research.

6.3.1 Opportunities to broaden

We could generalise the simple turn-taking metric $\tau\tau$ in a number of ways. The resource allocation model used to compute the agents' turn allocations considers that collisions are equally bad, regardless of the number of agents that attempt to use the shared resource at the same time. A resource allocation model that penalises collisions based on the number of agents colliding might better fit applications with large numbers of agents. As currently defined, $\tau\tau$ uses a fairness metric that focuses on the agent with the smallest allocation. However, it might be more appropriate to use a different fairness metric in situations with large numbers of agents where it is likely that one of the agents might break and stop taking turns. In general, we may want one agent to take more than its fair share of turns, so we could extend the metric by weighting the agents allocations according to some predefined scheme.

Further research is required to understand why the predictors for turn-taking in Chapter 4 worked as well as they did. Specifically, we do not have a good theoretical explanation for why the 'at least one policy of one Nash equilibrium' predictor was so successful. For some reason, a likely outcome observed in our simulations was that one agent would learn its policy in the Nash equilibrium while the other agent learned a different policy that was not part of the Nash equilibrium. We could investigate a number of medium access games with slightly different rules in order to identify those features of the multi-agent system that result in these deviations from the Nash equilibrium.

Constructing a reward function for a stochastic game is an under-defined problem: there exist many possible reward functions that result in a desired behaviour as the unique Nash equilibrium of the game. Our method in Chapter 5 simply chooses one possible reward function, without considering other possibilities. A system designer might want the reward function to depend on some global action criteria. For example, if at least one agent does some special action, then one reward function could be used, but if no agent does that action, then another reward function could be used. It may be possible to use such global action criteria to choose between the many possible reward functions that result in a desirable action as the unique Nash equilibrium. The method for designing rewards of stochastic games in Chapter 5 results in reward functions where each agent values each state equally when all agents play the Nash equilibrium. In some situations, a system designer may know that some states are beneficial to his or her desired outcome. Perhaps we could extend the methods of Chapter 5 to include information about which states the agents should value more highly than others.

6.3.2 Opportunities to extend

One useful tool for multi-agent system designers would be an automatic method for sequential multi-agent behaviour metric design. We could use some of the methodology of Chapter 3 in the context of automated metric design, such as comparisons with the performance of random agents. Automated metric design is related to pattern recognition (Theodoridis and Koutroumbas, 2009) and data mining (Witten and Frank, 2005). However, we may be able to use knowledge of when the agents took exploratory moves in order to reduce the information content of the history of agent actions. We could also use the policies that agents learn as summaries of their action histories. For particularly challenging metrics, we could use a semi-automated human-in-the-loop technique to learn human judgements.

In Chapter 5, we demonstrate our method for designing rewards for stochastic games with a single example. One important area for future work is to experimentally verify the learnability of the reward functions that our method in Chapter 5 produces. This would involve developing a range of Markov decision processes, normal form games and stochastic games and a range of desirable behaviours, then testing the learnability of the reward functions that our methods produce by running simulations with a range of multi-agent reinforcement learning algorithms. Because there are many different games, potentially desirable behaviours and learning algorithms, the simulations may be computationally expensive and the analysis and interpretation of the results may be challenging.

In addition to performing systematic testing, we plan to apply our reward design methods to constructing multi-agent musical systems to generate musical rhythms. Musical rhythms are related to turn-taking in conversation (Wilson and Wilson, 2005) and there are metrics that relate to musical rhythms (Toussaint, 2004). Some music has hierarchical structure so musical agents may need to have hierarchical learning algorithms, such as MAX-Q (Dietterich, 1998). Hierarchical learning algorithms often ignore information and this may affect whether or not agents that use hierarchical algorithms can converge to a Nash equilibrium.

The algorithms in Chapter 5 use graphs with certain properties to construct reward functions for intransitive normal form games. Appendix A presents algorithms that generate graphs; we hypothesise that these algorithms produce graphs with the required properties but this has not been formally proven. We intend to study those graph generator algorithms further and prove that they produce the correct output.

We focus on the agents' reward functions. However, other aspects of multi-agent systems can also be flexible. Our methods for designing reward functions could be married to new methods for designing system states, state transition functions, agent

action sets and meta-learning algorithms to design agents for a unified algorithmic method for multi-agent reinforcement learning system design. One challenge would be to create a unified system design method that maintains sufficient generality but is detailed enough to be practically useful.

6.3.3 Possible applications

The turn-taking metric $\tau\tau$ introduced in Chapter 3 could be useful in improving human-machine interaction, for example in spoken dialog systems (Raux and Eskenazi, 2008), and also in improving human-human interactions. For example, people with autism might benefit from real-time quantitative feedback on the quality of their turn-taking in social settings. Automated conversation quality analysis might also provide useful feedback for professionals and salespeople who want to improve the outcomes of their social interactions.

Biological conservation efforts relate to time-varying joint outcomes for groups of animals. Unique ecological environments such as those found in New Zealand or Australia present a number of challenges in biology and conservation. In some cases, scientific observations of empirical animal behaviour data have been fitted to turn-taking models (Harcourt et al., 2010) or game-theoretic models (Sinervo and Lively, 1996). The research in Chapters 4 and 5 suggests a way to create models for the underlying incentives that generate particular time-varying joint outcomes in multi-agent systems. By creating theoretical models for animal behaviours and outcomes, we may be able to better inform conservationists and policy makers.

The reward design techniques in Chapters 4 and 5 suggest ways to guide coordination in heterogeneous groups of agents. We could apply these techniques to design reward functions for robots that would lead toward desirable behaviours when the robots interact with people. Effective urban search and rescue robots would be invaluable in emergency situations, like the destructive 2011 earthquake in Christchurch, New Zealand, but such robot systems face theoretical and practical challenges (Davids, 2002; Carpin et al., 2007). We could use our reward design methods for planning the motions of autonomous urban search and rescue robots.

6.4 Final summary

This dissertation presents an original body of research in multi-agent systems and machine learning that makes a number of contributions to the human knowledge of measuring and influencing sequential joint agent behaviours: we propose a metric for turn-taking and demonstrate its functioning; we define and analyse medium access

games then present simulation results that demonstrate how to predict the emergence of turn-taking in pairs of Q-learning agents; and we introduce a method for designing rewards for stochastic games that result in desirable unique Nash equilibria and prove that our method is successful. We ground our research in previous work, address potential criticisms and suggest a number of possibilities for future work. Taken as a whole, this dissertation demonstrates the thesis:

Algorithmically designed reward functions can influence groups of learning agents toward measurable desired sequential joint behaviours.

References

- Agogino, A. K. and Tumer, K. (2008). Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17(2):320–338.
- Amato, C., Bernstein, D. S., and Zilberstein, S. (2009). Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21(3):293–320.
- Ambardar, A. (1999). *Analog and Digital Signal Processing*. CL Engineering, 2nd edition.
- Ampatzis, C., Tuci, E., Trianni, V., and Dorigo, M. (2008). Evolution of signaling in a multi-robot system: Categorization and communication. *Adaptive Behavior*, 16(1):5.
- Andreae, J. H. (1963). STeLLA, a scheme for a learning machine. In Broida and Butterworths, editors, *Proceeding of the 2nd IFAC Congress, Basle*, pages 497–502, London, UK.
- Andreae, J. H. (1969). Learning machines: A unified view. In Meetham, A. and Hudson, R., editors, *Encyclopedia of Information, Linguistics, and Control*, pages 261–270. Pergamon, Oxford, UK.
- Andreae, J. H. (1977). *Thinking with the Teachable Machine*. Academic Press, London, UK.
- Andreae, J. H. (1998). *Associative Learning: For a Robot Intelligence*. Imperial College Press, London, UK.
- Andreae, J. H. and Cashin, P. M. (1969). A learning machine with monologue. *International Journal of Man-Machine Studies*, 1:1–20.
- Aristotle (1924). *Metaphysics*. Clarendon Press, Oxford, UK. Translated by W. D. Ross.
- Arora, S. and Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press, Cambridge, UK.
- Arrow, K. J. (1986). Rationality of self and others in an economic system. *The Journal of Business*, 59(4):S385–S399.

- Aumann, R. and Brandenburger, A. (1995). Epistemic conditions for Nash equilibrium. *Econometrica*, 63(5):1161–1180.
- Aumann, R. J. and Hart, S., editors (1992). *Handbook of Game Theory with Economic Applications*, volume 1. North Holland.
- Aumann, R. J. and Hart, S., editors (1995). *Handbook of Game Theory with Economic Applications*, volume 2. North Holland, 1st edition.
- Aumann, R. J. and Hart, S., editors (2002). *Handbook of Game Theory with Economic Applications*, volume 3. North Holland, 1st edition.
- Axelrod, R. and Hamilton, W. D. (1981). The evolution of cooperation. *Science*, 211(4489):1390–1396.
- Baddeley, A. (2000). The episodic buffer: A new component of working memory? *Trends in Cognitive Sciences*, 4(11):417–423.
- Baddeley, A. D. and Logie, H. R. (1999). Working memory: The multiple-component model. In Miyake, A. and Shah, P., editors, *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control*, pages 28–61. Cambridge University Press, Cambridge, UK.
- Banerjee, B. and Peng, J. (2003). Adaptive policy gradient in multiagent learning. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-agent Systems*, pages 686–692.
- Basharin, G. P., Langville, A. N., and Naumov, V. A. (2004). The life and work of A. A. Markov. *Linear Algebra and Its Applications*, 386:3–26.
- Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840.
- Blackwell, D. and Ferguson, T. S. (1968). The big match. *The Annals of Mathematical Statistics*, 39(1):159–163.
- Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 478–485.
- Bouveret, S. and Lang, J. (2008). Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. *Journal of Artificial Intelligence Research*, 32(1):525–564.
- Bowling, M. and Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250.

- Brams, S. J. (2008). *Mathematics and Democracy: Designing Better Voting and Fair-Division Procedures*. Princeton University Press, Princeton, NJ, USA.
- Brown, G. (1951). Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13(1):374–376.
- Buşoniu, L., Babuška, R., and De Schutter, B. (2008). A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172.
- Cangelosi, A. and Parisi, D. (2002). *Simulating the Evolution of Language*. Springer-Verlag, New York, NY, USA, London.
- Capra, F. (1946). *It's a wonderful life* (Film). First distributed by RKO Radio Pictures.
- Carpin, S., Lewis, M., Wang, J., Balakirsky, S., and Scrapper, C. (2007). Bridging the gap between simulation and reality in urban search and rescue. *Robocup 2006: Robot Soccer World Cup X*, pages 1–12.
- Chang, Y.-H., Ho, T., and Kaelbling, L. P. (2004). All learning is local: Multi-agent learning in global reward games. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems*, volume 16, Cambridge, MA. MIT Press.
- Chao, C. and Thomaz, A. L. (2010). Turn taking for human-robot interaction. In *Dialog with Robots: Papers From the AAAI Fall Symposium*, pages 132–134. Association for the Advancement of Artificial Intelligence.
- Chao, C. and Thomaz, A. L. (2012). Timing in multimodal turn-taking interactions: Control and analysis using timed petri nets. *Journal of Human-Robot Interaction*, 1(1):4–25.
- Chen, L., Low, S. H., and Doyle, J. C. (2010). Random access game and medium access control design. *IEEE/ACM Transactions on Networking (TON)*, 18(4):1303–1316.
- Chen, X. and Deng, X. (2006). Settling the complexity of two-player Nash equilibrium. In *47th Annual IEEE Symposium on Foundations of Computer Science, 2006. FOCS '06*, pages 261–272.
- Chen, Y. and Gazzale, R. (2004). When does learning in games generate convergence to Nash equilibria? The role of supermodularity in an experimental setting. *The American Economic Review*, 94(5):1505–1535.
- Chen, Z. and Zhang, C. (2005). A new measurement for network sharing fairness. *Computers & Mathematics with Applications*, 50(5-6):803–808.

- Ching, W.-K. and Ng, M. K. (2005). *Markov Chains: Models, Algorithms and Applications*. Springer, 1st edition.
- Christoffels, I. (2006). Listening while talking: The retention of prose under articulatory suppression in relation to simultaneous interpreting. *European Journal of Cognitive Psychology*, 18(2):206–220.
- Clarke, E. H. (1971). Multipart pricing of public goods. *Public Choice*, 11(1):17–33.
- Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 746–752.
- Codenotti, B., Leoncini, M., and Resta, G. (2006). Efficient computation of Nash equilibria for very sparse win-lose bimatrix games. In Azar, Y. and Erlebach, T., editors, *Algorithms – ESA 2006*, volume 4168 of *Lecture Notes in Computer Science*, pages 232–243. Springer Berlin / Heidelberg.
- Colman, A. M. and Browning, L. (2009). Evolution of cooperative turn-taking. *Evolutionary Ecology Research*, 11(6):949–963.
- Conitzer, V. and Sandholm, T. (2002). Complexity of mechanism design. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 103–110.
- Conitzer, V. and Sandholm, T. (2003). AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In Fawcett, T. and Mishra, N., editors, *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, pages 83–90, Washington, DC, USA. AAAI Press.
- Cook, M. (2004). Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40.
- Crandall, J. W. and Goodrich, M. A. (2010). Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. *Machine Learning*, 82(3):281–314.
- Davids, A. (2002). Urban search and rescue robots: From tragedy to technology. *IEEE Intelligent Systems*, 17(2):81–83.
- Di Paolo, E. A. (2000). Behavioral coordination, structural congruence and entrainment in a simulation of acoustically coupled agents. *Adaptive Behavior*, 8(1):27–48.
- Dietterich, T. G. (1998). The MAXQ method for hierarchical reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning*, pages 118–126. Morgan Kaufmann.

- Eigenfeldt, A. (2007). The creation of evolutionary rhythms within a multi-agent networked drum ensemble. In *Proceedings of the International Computer Music Conference, Copenhagen*, pages 267–270. MPublishing, University of Michigan Library, Ann Arbor, MI, USA.
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.
- Filar, J. A. and Vrieze, K. (1997). *Competitive Markov Decision Processes*. Springer, New York, NY, USA.
- Flam, S. (2002). Equilibrium, evolutionary stability and gradient dynamics. *International Game Theory Review*, 4(4):357–370.
- Flood, M. M. (1958). Some experimental games. *Management Science*, 5(1):5–26.
- Floreano, D., Mitri, S., Magnenat, S., and Keller, L. (2007). Evolutionary conditions for the emergence of communication in robots. *Current Biology*, 17(6):514–519.
- Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway’s new solitaire game “Life”. *Scientific American*, 223(4):120–123.
- Gillette, D. (1957). Stochastic games with zero stop probabilities. *Contributions to the Theory of Games*, 3:179–187.
- Gittins, J., Glazebrook, K., and Weber, R. (2011). *Multi-Armed Bandit Allocation Indices*. Wiley, 2nd edition.
- Gmytrasiewicz, P. J. and Durfee, E. H. (2001). Rational communication in Multi-Agent environments. *Autonomous Agents and Multi-Agent Systems*, 4(3):233–272.
- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional.
- Goldman, C., Allen, M., and Zilberstein, S. (2007). Learning to communicate in a decentralized environment. *Autonomous Agents and Multi-Agent Systems*, 15(1):47–90.
- Gordon, G. J. (2007). Agendas for multi-agent learning. *Artificial Intelligence*, 171(7):392–401.
- Greenwald, A. and Hall, K. (2003). Correlated Q-learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML ’03)*, Washington, DC, USA.
- Grim, P., Denis, P. S., and Kokalis, T. (2002). Learning to communicate: The emergence of signaling in spatialized arrays of neural nets. *Adaptive Behavior*, 10(1):45.
- Groves, T. (1973). Incentives in teams. *Econometrica: Journal of the Econometric Society*, 41(4):617–631.

- Häggström, O. (2002). *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press, 1st edition.
- Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the National Conference on Artificial Intelligence*, pages 709–715.
- Harcourt, J., Sweetman, G., Manica, A., and Johnstone, R. (2010). Pairs of fish resolve conflicts over coordinated movement by taking turns. *Current Biology*, 20(2):156–160.
- Hardin, G. (1968). The tragedy of the commons. *Science*, 162(3859):1243–1248.
- Haykin, S. (2005). Cognitive radio: Brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications*, 23(2):201–220.
- Helbing, D., Schoenhof, M., Stark, H., and Holyst, J. (2005). How individuals learn to take turns: Emergence of alternating cooperation in a congestion game and the prisoner’s dilemma. *Advances in Complex Systems*, 8(1):87–116.
- Hu, J. and Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the 15th International Conference on Machine Learning (ICML '98)*, pages 242–250, San Francisco, CA, USA. Morgan Kaufmann.
- Hu, J. and Wellman, M. P. (2003). Nash Q-Learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4(Nov):1039–1069.
- Iizuka, H. and Ikegami, T. (2004). Adaptability and diversity in simulated turn-taking behavior. *Artificial Life*, 10(4):361–378.
- Ishikida, T. and Varaiya, P. (1994). Multi-armed bandit problem revisited. *Journal of Optimization Theory and Applications*, 83(1):113–154.
- Jafari, A., Greenwald, A. R., Gondek, D., and Ercal, G. (2001). On no-regret learning, fictitious play, and nash equilibrium. In *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, pages 226–233, San Francisco, CA, USA. Morgan Kaufmann.
- Jain, R., Chiu, D., and Hawe, W. (1984). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, Digital Equipment Corporation, Hudson, MA, USA.
- Jungers, M. K., Palmer, C., and Speer, S. R. (2002). Time after time: The coordinating influence of tempo in music and speech. *Cognitive Processing*, 1(2):21–35.

- Juslin, P. N. and Laukka, P. (2003). Communication of emotions in vocal expression and music performance: Different channels, same code? *Psychological Bulletin*, 129(5):770–814.
- Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134.
- Kapetanakis, S. and Kudenko, D. (2002). Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 326–331. AAAI Press.
- Kass, S. (2012). Rock paper scissors Spock lizard. <http://www.samkass.com/theories/RPSSL.html> (accessed 24 August 2012).
- Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. (1997). RoboCup: the robot world cup initiative. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 340–347, New York, NY, USA. ACM.
- Kononen, V. (2003). Asymmetric multiagent reinforcement learning. In *IEEE/WIC International Conference on Intelligent Agent Technology*, pages 336–342.
- Kose-Bagci, H., Dautenhahn, K., and Nehaniv, C. (2008). Emergent dynamics of turn-taking interaction in drumming games with a humanoid robot. In *The 17th IEEE International Symposium on Robot and Human Interactive Communication, 2008*, pages 346–353.
- Kreps, V. L. (1974). Bimatrix games with unique equilibrium points. *International Journal of Game Theory*, 3(2):115–118.
- Kreps, V. L. (1981). Finite N-person non-cooperative games with unique equilibrium points. *International Journal of Game Theory*, 10(3):125–129.
- Kröger, F. and Merz, S. (2008). *Temporal Logic and State Systems*. Texts in Theoretical Computer Science. An EATCS Series. Springer Publishing Company, Incorporated, 1st edition.
- Lan, T., Kao, D., Chiang, M., and Sabharwal, A. (2009). An axiomatic theory of fairness. Technical report, Princeton University, Princeton, NJ, USA.
- Langton, C. G. (1986). Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena*, 22(1-3):120–149.
- Large, E. W. and Jones, M. R. (1999). The dynamics of attending: How people track time-varying events. *Psychological Review*, 106(1):119–159.
- Large, E. W. and Palmer, C. (2002). Perceiving temporal regularity in music. *Cognitive Science*, 26(1):1–37.

- Larsson, E. G., Jorswieck, E., Lindblom, J., and Mochaourab, R. (2009). Game theory and the flat fading gaussian interference channel. *IEEE Signal Processing Magazine*, 26(5):18–27.
- Leshem, A. and Zehavi, E. (2009). Game theory and the frequency selective interference channel. *IEEE Signal Processing Magazine*, 26(5):28–40.
- Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann.
- Littman, M. L. (2001). Value-function reinforcement learning in Markov games. *Cognitive Systems Research*, 2(1):55–66.
- Lutz, M. (2006). *Programming Python*. O'Reilly Media, Inc., Sebastopol, CA, USA.
- Mailath, G. J. (1998). Do people play Nash equilibrium? Lessons from evolutionary game theory. *Journal of Economic Literature*, 36(3):1347–1374.
- Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.
- Manson, J. H., David Navarrete, C., Silk, J. B., and Perry, S. (2004). Time-matched grooming in female primates? New analyses from two species. *Animal Behaviour*, 67(3):493–500.
- Marchi, E. and Quintas, L. G. (1987). About extreme equilibrium points. *Mathematical Social Sciences*, 13(3):273–276.
- Markov, A. (1906). Extension of the law of large numbers to dependent quantities [in Russian]. *Proceedings of the Physical and Mathematical Society at the University of Kazan, series 2*, 15:135–156.
- Mas-Colell, A., Whinston, M. D., and Green, J. R. (1995). *Microeconomic Theory*. Oxford University Press, USA.
- Maskin, E. (1985). The theory of implementation in Nash equilibrium: A survey. *Social Goals and Social Organization: Essays in Memory of Elisha Pazner*, pages 173–204.
- Matarić, M. (1997). Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83.
- Mitola, J. and Maguire, G. Q. (1999). Cognitive radio: Making software radios more personal. *IEEE Personal Communications*, 6(4):13–18.
- Monro, G. (2007). *The Concept of Emergence in Generative Art*. Masters thesis, Sydney Conservatorium of Music, University of Sydney, Sydney, Australia.

- Moore, G. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8):33–35.
- Mount, K. and Reiter, S. (1974). The informational size of message spaces. *Journal of Economic Theory*, 8(2):161–192.
- Musgrave, R. A. and Musgrave, P. B. (1984). *Public Finance in Theory and Practice*. McGraw-Hill, New York, 4th edition.
- Nash, J. F. (1950). Equilibrium points in N-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49.
- Neill, D. B. (2001). Optimality under noise: Higher memory strategies for the alternating prisoner’s dilemma. *Journal of Theoretical Biology*, 211(2):159–180.
- Neill, D. B. (2003). Cooperation and coordination in the turn-taking dilemma. In Tennenholtz, M., editor, *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 231–244, New York. ACM.
- Nisan, N. and Ronen, A. (1999). Algorithmic mechanism design (extended abstract). In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing, STOC ’99*, pages 129–140, New York, NY, USA. ACM.
- Nolfi, S. (2005). Emergence of communication in embodied agents: Co-adapting communicative and non-communicative behaviours. *Connection Science*, 17(3):231–248.
- NoroozOliaee, M., Hamdaoui, B., and Tumer, K. (2012). Efficient objective functions for coordinated learning in large-scale distributed OSA systems. *IEEE Transactions on Mobile Computing*, pre-print(99):1.
- Open Source Initiative (1999). The BSD license. Technical report, Open Source Initiative. <http://opensource.org/licenses/BSD-3-Clause> (accessed 24 August 2012).
- Palfrey, T. R. (2002). Implementation theory. *Handbook of Game Theory with Economic Applications*, 3:2271–2326.
- Pinker, S. (1995). *The Language Instinct: How the Mind Creates Language*. Perennial (Harper Collins), 1st edition.
- Pinker, S. (1999). *How the Mind Works*. W. W. Norton & Company.
- Pnueli, A. (1977). The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Providence, RI, USA.

- Powers, R. and Shoham, Y. (2005). New criteria and a new algorithm for learning in multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 17, pages 1089–1096.
- Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. New York, NY, USA.
- Quinn, M. (2001). Evolving communication without dedicated communication channels. In Kelemen, J. and Sosik, P., editors, *Advances in Artificial Life: 6th European Conference*, volume 2159 of *Lecture Notes in Computer Science*, pages 357–366, Berlin. Springer.
- Quintas, L. G. (1988). Constructing bimatrix games with unique equilibrium points. *Mathematical Social Sciences*, 15(1):61–72.
- Raffensperger, P. A. (2012). Toward a wave digital filter model of the Fairchild 670 limiter. In *Proceedings of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, York, UK.
- Raffensperger, P. A., Bones, P. J., McInnes, A. I., and Webb, R. Y. (2012a). Rewards for pairs of Q-learning agents conducive to turn-taking in medium-access games. *Adaptive Behavior*, 20(4):304–318.
- Raffensperger, P. A. and Hayes, M. P. (2008). Motion to music interface. In *Proceedings of the Electronics New Zealand Conference*, Auckland, New Zealand.
- Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012b). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104–116.
- Rasa, O. A. E. (1989). The costs and effectiveness of vigilance behaviour in the dwarf mongoose: Implications for fitness and optimal group size. *Ethology Ecology & Evolution*, 1(3):265–282.
- Raux, A. and Eskenazi, M. (2008). Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. In Schlangen, D. and Hockey, B. A., editors, *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 1–10, Morristown, NJ. Association for Computational Linguistics.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 25–34.
- Roddenberry, G. (1966). *Star Trek* (TV show). Produced by Desilu Productions, first telecast on NBC.
- Sacks, H., Schegloff, E. A., and Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735.

- Sandholm, T. (2003). Automated mechanism design: A new application area for search algorithms. In Rossi, F., editor, *Principles and Practice of Constraint Programming – CP 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 19–36. Springer Berlin / Heidelberg.
- Sen, S., Sekaran, M., and Hale, J. (1994). Learning to coordinate without sharing information. In *Proceedings of the National Conference on Artificial Intelligence*, pages 426–426.
- Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095.
- Sherstov, A. and Stone, P. (2005). Three automated stock-trading agents: A comparative study. In Faratin, P. and Rodriguez-Aguilar, J., editors, *Agent-Mediated Electronic Commerce VI. Theories for and Engineering of Distributed Mechanisms and Systems*, volume 3435 of *Lecture Notes in Computer Science*, pages 173–187. Springer Berlin / Heidelberg.
- Shoham, Y. and Leyton-Brown, K. (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, Cambridge, UK.
- Shoham, Y., Powers, R., and Grenager, T. (2007). If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377.
- Sinervo, B. and Lively, C. M. (1996). The rock-paper-scissors game and the evolution of alternative male strategies. *Nature*, 380(6571):240–243.
- Singh, S., Kearns, M., and Mansour, Y. (2000). Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 541–548.
- Skinner, B. F. (1938). *The Behavior of Organisms: An Experimental Analysis*. Copley Publishing Group, Acton, MA, USA.
- Smith, K. (2002). The cultural evolution of communication in a population of neural networks. *Connection Science*, 14(1):65.
- Snedecor, G. W. and Cochran, W. G. (1989). *Statistical Methods*. Iowa State University Press, 8th edition.
- Sohn, J., Kim, N. S., and Sung, W. (1999). A statistical model-based voice activity detection. *IEEE Signal Processing Letters*, 6(1):1–3.
- Stevens, W. P., Myers, G. J., and Constantine, L. L. (1974). Structured design. *IBM Systems Journal*, 13(2):115–139.

- Stivers, T., Enfield, N. J., Brown, P., Englert, C., Hayashi, M., Heinemann, T., Hoymann, G., Rossano, F., Ruiter, J. P. D., Yoon, K. E., et al. (2009). Universals and cultural variation in turn-taking in conversation. *Proceedings of the National Academy of Sciences*, 106(26):10587.
- Stupakov, A., Hanusa, E., Bilmes, J., and Fox, D. (2009). COSINE – a corpus of multi-party CONversational Speech In Noisy Environments. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009.*, pages 4153–4156.
- Stupakov, A., Hanusa, E., Vijaywargi, D., Fox, D., and Bilmes, J. (2012). The design and collection of COSINE, a multi-microphone in situ speech corpus recorded in noisy environments. *Computer Speech & Language*, 26(1):52–66.
- Suematsu, N. and Hayashi, A. (2002). A multiagent reinforcement learning algorithm using extended optimal response. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, pages 370–377.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning*. MIT Press, Cambridge, MA.
- Sutton, R. S., Szepesvari, C., and Maei, H. R. (2009). A convergent $O(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 21, pages 1609–1616, Red Hook, NY, USA. Curran Associates, Inc.
- Tanenbaum, A. S. (2002). *Computer Networks*. Prentice Hall, New Jersey, 4th edition.
- Theodoridis, S. and Koutroumbas, K. (2009). *Pattern Recognition*. Elsevier/Academic Press, Waltham, MA, USA.
- Todes, D. P. (2001). *Pavlov's Physiology Factory: Experiment, Interpretation, Laboratory Enterprise*. The Johns Hopkins University Press, 1st edition.
- Toussaint, G. (2004). A comparison of rhythmic similarity measures. In *Proceedings of the 5th International Conference on Music Information Retrieval*, pages 242–245.
- Trivelpiece, W., Trivelpiece, S., and Volkman, N. (1987). Ecological segregation of adelic, gentoo, and chinstrap penguins at King George Island, Antarctica. *Ecology*, 68(2):351–361.
- Tumer, K. (2006). Designing agent utilities for coordinated, scalable and robust multi-agent systems. In Scerri, P., Vincent, R., and Mailler, R., editors, *Coordination of Large-Scale Multiagent Systems*, pages 173–188. Springer USA.

- Tumer, K. and Agogino, A. (2007). Distributed agent-based air traffic flow management. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07*, pages 255:1–255:8, New York, NY, USA. ACM.
- Turunen, M., Hakulinen, J., and Kainulainen, A. (2006). Evaluation of a spoken dialogue system with usability tests and long-term pilot studies: Similarities and differences. In *9th International Conference on Spoken Language Processing*, pages 1057–1060. ISCA.
- Tuyls, K., Hoen, P., and Vanschoenwinkel, B. (2006). An evolutionary dynamical analysis of multi-agent learning in iterated games. *Autonomous Agents and Multi-Agent Systems*, 12(1):115–153.
- Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37.
- von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press, 1st edition.
- von Stengel, B. (2007). Equilibrium computation for two-player games in strategic and extensive form. In Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V., editors, *Algorithmic Game Theory*, pages 53–78. Cambridge University Press, Cambridge, UK.
- Wagner, K., Reggia, J. A., Uriagereka, J., and Wilkinson, G. S. (2003). Progress in the simulation of emergent communication and language. *Adaptive Behavior*, 11(1):37.
- Waltman, L. and Kaymak, U. (2007). A theoretical analysis of cooperative behavior in multi-agent Q-Learning. In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007*, pages 84–91.
- Wang, X. and Sandholm, T. (2002). Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *Advances in Neural Information Processing Systems*, volume 15, pages 1571–1578.
- Watkins, C. (1989). *Learning From Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, UK.
- Weinberg, G. and Blosser, B. (2009). A leader-follower turn-taking model incorporating beat detection in musical human-robot interaction. In *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, pages 227–228, La Jolla, California, USA. ACM.
- Weinberg, M. and Rosenschein, J. S. (2004). Best-response multiagent learning in non-stationary environments. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2 of *AAMAS '04*, pages 506–513, Washington, DC, USA. IEEE Computer Society.

- Wilansky, A. (1951). The row-sums of the inverse matrix. *The American Mathematical Monthly*, 58(9):614–615.
- Wilson, M. and Wilson, T. P. (2005). An oscillator model of the timing of turn-taking. *Psychonomic Bulletin & Review*, 12(6):957.
- Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Burlington, MA, USA.
- Wolpert, D. H., Wheeler, K. R., and Tumer, K. (1999). General principles of learning-based multi-agent systems. In *Proceedings of the 3rd Annual Conference on Autonomous Agents*, AGENTS '99, pages 77–83, New York, NY, USA. ACM.
- Wongpiromsarn, T., Ulusoy, A., Belta, C., Frazzoli, E., and Rus, D. (2012). Incremental temporal logic synthesis of control policies for robots interacting with dynamic agents. *Arxiv Preprint arXiv:1203.1180*.
- Yang, J., Klein, A. G., and Brown, D. R. (2009). Natural cooperation in wireless networks. *IEEE Signal Processing Magazine*, 26(5):98–106.
- Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, pages 928–936, Washington, DC, USA. AAAI Press.
- Zinkevich, M., Greenwald, A., and Littman, M. (2006). Cyclic equilibria in Markov games. In Y. Weiss, B. S. and Platt, J., editors, *Advances in Neural Information Processing Systems*, volume 18, pages 1641–1648, Cambridge, MA. MIT Press.

Appendix A

Graph Generator Algorithms

In Chapter 5, we present algorithms that transform graphs with particular properties into intransitive normal form games. Here, we present Algorithm A.1, which generates graphs with the requisite properties for the two-agent case and Algorithm A.2, which generates graphs for the multi-agent case. Note that Quintas' work on constructing normal form games for two agents suggests a particular pair of functions that correspond to the output of Algorithm A.1 (Quintas, 1988). We conjecture that these algorithms function correctly, but we leave proofs to future work. The truth of the thesis statement does not depend on these particular algorithms. They are presented as potentially useful companions to the algorithms in Chapter 5 but are kept separate to maintain the focus of Chapter 5.

Input : A set of actions \mathcal{A}_n for each agent $n \in \{1, 2\}$, assuming that $|\mathcal{A}_n| > 1$ for all $n \in \mathcal{N}$.

Output: A directed bipartite graph, G , with vertices $V(G) = \bigcup_{n \in \mathcal{N}} \mathcal{A}_n$ and with an arc to each vertex and an arc from each vertex such that the graph has a Hamiltonian cycle.

- 1 Let $G \leftarrow (V, E)$ be a graph with arcs $E \leftarrow \emptyset$ and vertices $V \leftarrow \bigcup_{n \in \mathcal{N}} \mathcal{A}_n$.
- 2 Let $\Omega_n : \mathcal{A}_n \rightarrow \{1, 2, 3, \dots, |\mathcal{A}_n|\}$ be an arbitrary bijective ordering of agent n 's actions, for all $n \in \mathcal{N}$.
- 3 **For each** $a_1 \in \mathcal{A}_1$ **do**
- 4 Let $i \leftarrow \Omega_1(a_1)$.
- 5 Find a_2 such that $\Omega_2(a_2) = i$.
- 6 Create an arc from a_1 to a_2 and add the arc to E .
- 7 **For each** $a_2 \in \mathcal{A}_2$ **do**
- 8 Let $i \leftarrow \Omega_2(a_2) + 1$.
- 9 **If** $i > |\mathcal{A}_1|$ **then**
- 10 Set $i \leftarrow 1$.
- 11 Find a_1 such that $\Omega_1(a_1) = i$.
- 12 Create an arc from a_2 to a_1 and add the arc to E .
- 13 **Return** G .

Algorithm A.1: One possible method for constructing a directed bipartite graph with a Hamiltonian cycle, given a set of actions for each of two agents.

Input : A set of joint actions \mathcal{A} , assuming that Eq. (5.30) on p. 100 is satisfied and, without loss of generality, assuming that $|\mathcal{A}_1| \geq |\mathcal{A}_2| \geq \dots \geq |\mathcal{A}_{|\mathcal{N}|}|$.

Output: A directed $|\mathcal{N}|$ -partite graph, G , with vertices $V(G) = \bigcup_{n \in \mathcal{N}} \mathcal{A}_n$ and with an arc to each vertex and an arc from each vertex such that the graph has a Hamiltonian cycle.

```

1 Let  $G \leftarrow (V, E)$  be a graph with arcs  $E \leftarrow \emptyset$  and vertices  $V \leftarrow \bigcup_{n \in \mathcal{N}} \mathcal{A}_n$ .
2 Let  $remaining(1)$  be the number of vertices in  $\mathcal{A}_1$  that have no arcs from them.
3 Let  $remaining(-1)$  be the number of vertices in  $V(G)$  but not in  $\mathcal{A}_1$  that have
  no arcs from them.
4 Set  $start \leftarrow$  an arbitrary vertex in  $|\mathcal{A}_1|$ .
5 Set  $current \leftarrow start$ .
6 While  $remaining(-1) > remaining(1)$  do
7   If  $remaining(-1) = 1$  then
8      $next \leftarrow start$ 
9   Else if  $current \notin \mathcal{A}_1$  and  $remaining(-1) = remaining(1) + 1$  then
10     $next \leftarrow$  an arbitrary vertex in  $\mathcal{A}_1$  that has no arc from it.
11  Else
12    Find  $n$  such that  $current \in \mathcal{A}_n$ .
13     $n' \leftarrow (n + 1) \bmod |\mathcal{N}|$ 
14    While every vertex in  $\mathcal{A}_{n'}$  has an arc from it do
15       $n' \leftarrow (n' + 1) \bmod |\mathcal{N}|$ 
16     $next \leftarrow$  an arbitrary vertex in  $\mathcal{A}_{n'}$  that has no arc from it.
17  Create an arc from  $current$  to  $next$  and add the arc to  $E$ .
18   $current \leftarrow next$ .
19  Update the values of  $remaining(1)$  and  $remaining(-1)$ .
20 While  $remaining(1) > 0$  do
21   $next \leftarrow$  an arbitrary vertex in  $V(G)$  but not in  $\mathcal{A}_1$  that has no arc from it.
22  Create an arc from  $current$  to  $next$  and add the arc to  $E$ .
23   $current \leftarrow next$ .
24  If  $remaining(1) = 1$  then
25     $next \leftarrow start$ .
26  Else
27     $next \leftarrow$  an arbitrary vertex in  $\mathcal{A}_1$  that has no arc from it.
28  Create an arc from  $current$  to  $next$  and add the arc to  $E$ .
29   $current \leftarrow next$ .
30  Update the values of  $remaining(1)$  and  $remaining(-1)$ .
31 Return  $G$ .

```

Algorithm A.2: One possible method for constructing a directed $|\mathcal{N}|$ -partite graph with a Hamiltonian cycle, given a set of actions for each agent in the set of agents, \mathcal{N} .