Knowledge and Games
Theory and Implementation

Andreas Witzel

Knowledge and Games: Theory and Implementation

Andreas Witzel

Knowledge and Games: Theory and Implementation

ILLC Dissertation Series DS-2009-05

Institute for Logic, Language and Computation

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 904
1098 XH Amsterdam
phone: +31-20-525 6051
fax: +31-20-525 5206
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

Knowledge and Games: Theory and Implementation

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D. C. van den Boom
ten overstaan van een door het college voor promoties
ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op donderdag 3 september 2009, te 12.00 uur

door

Simon Andreas Witzel

geboren te Freiburg, Duitsland

Promotiecommissie:

Promotor:
Prof. dr. K. R. Apt

Overige leden:
Prof. dr. J. F. A. K. van Benthem
Prof. dr. D. J. N. van Eijck
Prof. dr. B. Löwe
Prof. dr. J.-J. Ch. Meyer
Prof. dr. R. Parikh
Dr. U. Endriss

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The investigations were supported by a GLoRiClass fellowship funded by the European Commission (Marie Curie Early Stage Research Training Mono-Host Fellowship MEST-CT-2005-020841).

Printed and bound by Ipskamp Drukkers.

ISBN: 90–5776–193–9

Contents

Acknowledgments

First of all, I am indebted to my supervisor Krzysztof Apt, who was always available for discussion and counsel, and who skillfully provided guidance while at the same time leaving ample freedom for my own ideas. With his help, a sufficiently coherent research plan evolved, where initially there were only vague conceptions.

I would like to thank Johan van Benthem, Jan van Eijck, Ulle Endriss, Benedikt Löwe, John-Jules Meyer, and Rohit Parikh for agreeing to be on my thesis committee, and for many insightful and pleasant discussions at various occasions. In particular, I am grateful to Rohit Parikh for being an incredibly dedicated host during a very inspiring summer at the Graduate Center in New York City; to Ulle Endriss for providing helpful guidance and interesting opportunities for teaching and collaborations; and to Benedikt Löwe for countless interesting discussions, help and suggestions for projects and collaborations, as well as his tireless administrative support as GLoRiClass project coordinator.

Speaking of administrative support, my time at ILLC, and partly at CWI, would have been impossible without Karin Gigengack, Tanja Kassenaar, Monique Laurent, Ingrid van Loon, Peter van Ormondt, and Marjan Veldhuisen, who are perpetually busy sustaining the great working environment. René Goedman was a reliable source of good mood at the entrance (unfortunately somewhat less frequently after we got outsourced to another building), and Marco Vervoort helped with the Dutch translation of the abstract of my dissertation.

At work outside the office, there were countless people I met on conferences, had interesting discussions with, or interacted with academically in various other ways. Especially I would like to thank Aaron Archer for inviting me to AT&T Shannon Research Laboratory in New Jersey, Ronald Fagin whom I visited at IBM Almaden Research Center, David Pennock who introduced me to Yahoo! Research New York City, as well as Rahul Bendre, Jelle Gerbrandy, Ethan Kennerly, Willemien Kets, Martin Magnusson, and Achim Rettinger, with whom I had helpful discussions, enjoyable collaborations, long email conversations, and entertaining activities.

In the office, many people made life enjoyable. With Cédric Dégremont, Pietro Galliani, Nina Gierasimczuk, Umberto Grandi, Nicole Immorlica, Jarmo Kontinen, Vangelis Markakis, Jakub Szymanik, Joel Uckelman, Yun Qi Xue, and Jonathan Zvesper I had awesome times at work, on conferences, and travelling. I am especially thankful to Jakub, with whom I not only shared an office during this whole adventurous time, but also many highs and lows connected to it and to life in general. Almost like a big brother, he has been a source of calmness, confidence and fun at the same time. Jonathan has been a great research and conference/travel mate, and many a night was spent discussing logic and life with his culinary support. I would like to thank Jonathan and Nina for proofreading parts of my dissertation. Finally, a very enjoyable tradition of overtime at the office has been established with Cédric, Nina, Jakub, Jarmo, Pietro and QiQi, at times leaving us with red eyes and trembling hands.

Naturally, there were also people with whom I did not work together, and still had good moments. I would not want to miss the hours I spent chatting, laughing, musicizing, learning about language, life, and myself, with Olivia Ladinig and Salvador Mascarenhas. It was great living with Olivia and the other boaties Tikitu de Jager, Olga Grigoriadou, and Stefan Bold, and, in postdiluvian times, with Inés Crespo, Craig Rea, and QiQi. I am thankful to many other ILLC people for the good times we had at parties and other occasions: Edgar Andrade-Lotero, Gaëlle Fontaine, Michael Franke, Amélie Gheerbrant, Umberto Grandi, Eric Hielscher, Daisuke Ikegami, Lauri Keskinen, Szymon Klarman, Marijn Koolen, Lena Kurzen, Henrik Nordmark, Brammert Ottens, Eric Pacuit, Petter Remen, Raul Leal Rodriguez, Olivier Roy, Federico Sangati, Merlijn Sevenster, Jonathan Shaheen, Leigh Smith, Marc Staudacher, Sarah Uckelman, and Fernando Velazquez-Quesada.

I am also very happy about the familiar faces from the past with whom I kept contact during this time and enjoyed many occasions for biking, hiking, and reality checks: Frank Blum, Anders Eriksson, Anne Fiedler, Richard Heidler, Nicola Kaiser, Philipp Pulvermüller, Rafael Say, Bernd Schlabach, Martin Schmal, Hendrik Skubch, Moritz Weeger, Till Winkler and my sister Miriam with her young family.

Probably I forgot to mention one important person or other here (sorry, I will invite you for a beer!). But most importantly, I would like to thank my parents, who have been a constant source of support in whatever I did for the past 29 years; and QiQi, who has been with me in good and in difficult times, making every day a special day—including even the stressful final days of this work.

Amsterdam/Santa Fe — Andreas Witzel
July 2009

Introduction

Does she know what he knows? And if so, *what is she going to do?*
This dissertation takes a computer science perspective on questions of knowledge and interaction and presents approaches for endowing artificial agents with corresponding reasoning capabilities.

Background

Epistemic logic

Epistemic logic is the formal study of reasoning about **knowledge**, including knowledge about knowledge (so-called *higher-order* knowledge). The modern field of epistemic logic has been initiated by von Wright [158] and Hintikka [78]. More recent treatments include work by Fagin et al. [60] and Meyer and van der Hoek [100].

The fundamental idea for a formal **model** of agents' knowledge is to consider a set of *possible worlds*, or *states* in which the actual world may be, together with *indistinguishability relations* between them, one for each agent. In each possible world, certain *atomic statements*, or *propositions*, hold. Intuitively, an agent *knows* whatever holds in all worlds which he *cannot distinguish* from (or which he *considers possible* in) the actual world. In order to properly reflect certain properties that the philosophical notion of knowledge is thought to possess, the indistinguishability relations are usually required to be equivalence relations.

This *relational semantics* is often called *Kripke semantics*, and the involved structures *Kripke structures*, due to the pioneering work in modal logic by Kripke [87].

For clarification, consider the following example.[1] Assume that for some reason, Ann knows Bob's bank PIN code. It then depends on her higher-order knowledge

[1] This example is taken from [17]. For simplicity, issues like possession of the physical bank pass or effects of involving additional agents are abstracted away from.

whether she can safely empty his bank account: she can do this only if she knows that he doesn't know that she knows the code—otherwise he would immediately suspect her. Such a "safe" situation is modeled in Figure 1. In the actual world on the left, Bob's PIN is 1234. Bob considers another world possible, where his PIN is still 1234 (he knows his PIN), but where Ann considers a world possible where his PIN is *not* 1234. In the actual world, however, Ann only considers the actual world possible. That is, Ann knows Bob's PIN, Bob doesn't know that she knows it, and Ann in turn knows that he doesn't know that she knows it.

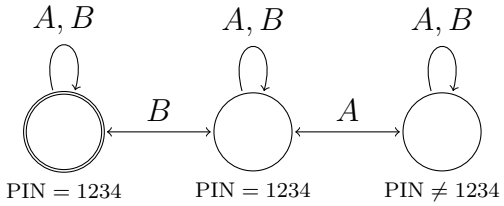


Figure 1: Model of a situation where it is safe for Ann (A) to empty Bob's (B) bank account.

In order to talk about such models formally, an epistemic **language** is used, consisting of *formulas* with a clearly defined *syntax*. In the simplest case, the language consists of letters for the atomic statements, certain logical *connectives* to combine statements to more complex ones, and a *knowledge operator* for each agent. For example, if we use p to denote that Bob's PIN is 1234, the connectives $\wedge$ and $\neg$ to denote "and" and "not", and K_A and K_B as knowledge operators for Ann and Bob, then the formula

$$K_A(p \wedge \neg K_B K_A p)$$

means that Ann knows Bob's PIN and knows that he doesn't know that she knows it. As we saw, this formula *holds* in the actual world in the model in Figure 1.

In general, one can **reason**, or **deduce**, in two fundamental ways. *Semantically*, one can say, for example, that a formula follows from another formula if the former holds in all models under consideration in which the latter holds. *Syntactically*, one can use certain *axioms* and *inference rules* to reach a desired conclusion. We are in this dissertation mostly concerned with semantic arguments.

Besides knowledge, one can also model the related notion of **belief**.[2] The difference is that, philosophically speaking, the concept of knowledge implies correctness, while beliefs may be false. On the level of models, this is reflected by using more general *accessibility relations*, which need not be equivalence relations. For example, if we replace Bob's indistinguishability between the actual and the middle world in Figure 1 by a one-way accessibility from the actual to the middle

[2]In some contexts, one then speaks of *doxastic logic*, though in other contexts this term has a more narrow definition referring to frameworks formalizing how beliefs are *revised*.

world, we obtain the situation depicted in Figure 2 where Bob (mistakenly) *believes* that Ann does *not* know his PIN.

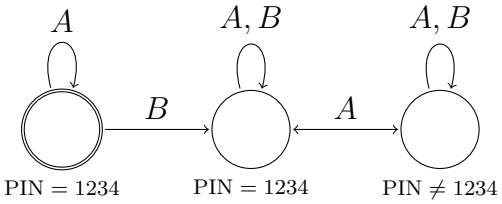


Figure 2: Model of a situation where Bob (B) mistakenly believes that Ann (A) does not know his PIN.

A concept of particular importance is **common knowledge** (or belief), which intuitively corresponds to the infinite iteration of *mutual knowledge* (or belief):

$$K_B p, \quad K_A K_B p, \quad K_B K_A K_B p, \quad K_A K_B K_A K_B p, \quad \dots$$

and so on ad infinitum. We illustrate this limit case in the upcoming subsection.

Distributed computing

Distributed computing is the formal study of programs, or processes, running simultaneously on possibly different processing units. Typically, these processes can coordinate and communicate in some way, which necessitates a study of their joint behavior and the system they form as a whole. In particular, one can ask what processes can be said to "know" about each other, each other's state, and each other's knowledge. This is relevant, for example, for sharing resources or ensuring correct or secret transmission of information. Consequently, issues around knowledge in distributed systems have been studied using formalisms similar to the one described above, among others, by Parikh and Ramanujam [116], Chandy and Misra [39], and extensively by Fagin et al. [60].

Some intuitions about the concept of common knowledge can be drawn from a classic example in distributed computing, introduced by Akkoyunlu et al. [1] and Gray [70] and later independently considered by Rubinstein [129]. The example in its most well-known version features two generals who are positioned with their armies on two hilltops with no direct line of sight (and no mobile phones). They need to launch a *coordinated attack* on their enemy in the valley. So one general, let us call him A, sends a messenger to the other general, B, proposing a time for the attack. Unfortunately, A has no way of knowing whether and when his messenger will arrive to deliver the message. Since B is aware of this fact, upon receiving the proposal he sends the messenger back to confirm. So far so good, but B has no way of knowing whether and when his confirmation will be received. And as long as it has not been received, A will not know that B received the

original proposal, and thus A will not attack. Since A is aware of this fact, he sends the messenger back to confirm... and before they know it, the generals are trapped in an infinite loop.

It can be shown that what the generals need is *common knowledge* of the original message, no finite level of mutual knowledge will suffice for their purposes. Without compromises, and without sufficient time for exchanging infinitely many confirmations, common knowledge can only be attained by **synchronous communication** such as provided by a direct line of sight (or mobile phones).[3]

Game theory

Game theory, a framework for the formal study of strategic interaction, was initiated by von Neumann and Morgenstern [104]. Many modern textbooks exist, e.g., Osborne [106]. Issues of (common) knowledge and belief are a subject of continuing interest in game theory, starting with work by Harsanyi [76] and Aumann [12]. The interface of (epistemic) logic and game theory is currently being actively studied, for example by van Benthem [18].

The basic idea of **non-cooperative** game theory is that agents, also called *players*, act and interact without any institutions that would allow them to form and enforce binding agreements for cooperation. At the heart of such a player lies a **payoff function**, which models the payoff, or *utility*, which the player derives from any given interaction. An interaction in this framework is taken to be a tuple of simultaneous *actions*, or **strategies**, one for each player. A payoff function then maps such **strategy profiles**, or **joint strategies**, to numbers representing the utility.[4] A **rational** player is assumed to aim at maximizing his utility.

For example, assume that Ann loves Indian food and isn't too fond of Mexican food, and that Bob likes Ann[5] and really wants to eat in the same restaurant as she does. So consider the actions of going to an Indian restaurant or going to a Mexican restaurant. The according payoff functions can be specified using a **payoff matrix** as in Figure 3.

Clearly, such payoff structures induce (or reflect) *preferences*: Ann (unconditionally) prefers the Indian over the Mexican restaurant, while Bob prefers Indian over Mexican if Ann chooses Indian, and Mexican over Indian if she chooses Mexican. If Bob does end up in a different restaurant than Ann, though, then he would rather eat Mexican food than Indian.

[3] There are some subtleties regarding just how synchronous communication can actually be, especially when mediated by communication devices. Monderer and Samet [101] show how common belief can be seen to approximate common knowledge as uncertainty decreases. See Chapter 1, Section 1.5, for some discussion in our context.

[4] We are not concerned with so-called *extensive-form* games, which consist of multiple actions taken in turns. Suffice it to say that they can be represented in the *normal form* we describe here.

[5] He hasn't found out about his bank account yet.

		Bob	
		Indian	Mexican
Ann	Indian	$10, 10$	$10, 5$
	Mexican	$0, 0$	$0, 10$

Figure 3: Ann's and Bob's strategies and corresponding payoffs, giving for each strategy profile first Ann's and then Bob's payoff.

In this example, going to the Mexican restaurant is **strictly dominated** for Ann: no matter what Bob does, she always gets a strictly higher payoff from choosing the Indian restaurant. The usual definition of a rational player implies that such an action will not be chosen—it can be *eliminated*, leaving us with the game in Figure 4.

		Bob	
		Indian	Mexican
Ann	Indian	$10, 10$	$10, 5$

Figure 4: The game after eliminating the Mexican restaurant choice for Ann.

It turns out that in this reduced game, Mexican is dominated by Indian for Bob. Eliminating this strategy in turn, we end up with a trivial game where both Ann and Bob are left with the single choice of going to the Indian restaurant. This process, first considered by Dekel and Fudenberg [46], is known as **iterated elimination of strictly dominated strategies** (IESDS).

So far we have taken an omniscient point of view and simply manipulated the fully specified payoff matrix. If we put ourselves in a player's shoes, however, who may have **incomplete information** about other players' preferences and knowledge, then things get more involved. In our example, for Bob to safely perform the last elimination, he needs to *know* that Ann eliminated the Mexican restaurant choice—that is, he needs to know that Ann is rational and strictly prefers the Indian restaurant.[6] If we introduce a third player, Carol, who likes Bob and wants to prevent him from being alone with Ann, then for her restaurant choice Carol needs to know whether Bob knows what Ann prefers. The whole elimination process then needs to be formulated in terms of knowledge, and epistemic logic lends itself to that end.

If the players obtain their knowledge through some sort of communication, then matters are somewhat simpler with *synchronous* communication, which, as we saw above, creates common knowledge, superseding intermediate levels of mutual knowledge and removing the need for confirmations. Such a setting is the

[6] Of course Ann could eliminate the choice for other reasons than rationality and dominance, but let us stick with this explanation here.

motivation for the main part of this dissertation, and we give some more intuition and justification in the appropriate places.

Outside the main part, we deal with two other topics from game theory: firstly **combinatorial auctions**, and secondly **coalition formation**.

Combinatorial auctions are auctions of multiple goods where the bidders can bid on *combinations* of these goods. As in conventional single-item auctions, the bidders transmit their bids to the auctioneer, who determines the winner of the auction. However, in contrast to single-item auctions, in a combinatorial setting bid representation and winner determination become rich and complex issues; see, e.g., the book compiled by Cramton et al. [43] for a recent overview of these and other topics. We consider a certain kind of combinatorial auction where services transforming goods are offered, rather than conventional atomic goods. These services then may be scheduled in such a way that the output of one transformation can be used as input for another transformation. We examine how preferences over such orderings of the offered transformations can be taken into account.

Coalition formation belongs to the field of **cooperative** game theory, which assumes that there *is* a way for players to form and enforce binding agreements. The *coalitions* resulting from such agreements constitute the focus of interest on two scales. From the individual player's point of view, the questions concern what he can, or should, get out of joining a coalition, and what coalitions he prefers over others; and on a larger scale, the dynamics of coalitions are of interest, how they form and change, and when they might be considered stable. The recent textbook by Ray [123] provides an overview of these latter issues. We address them under an operational viewpoint of merging and splitting coalitions.

Social networks

In the research field of social networks, one studies *relationships* and *interdependencies* among individuals. This field has been highly active in recent years, see, e.g., the books by Jackson [83] and Goyal [69].

We are here especially interested in *communication networks*, that is, networks which determine the possibilities for communication among the individual agents. Sharing of information in social networks has been studied in probabilistic frameworks, e.g., by Chamley [38]. Within logic, the relevance of epistemic issues in communication networks has been recognized by a number of authors, e.g., by van Benthem [20] and Pacuit and Parikh [109].

To illustrate some of the issues and subtleties involved, we give here an informal instance of the framework we set up in the main part of this dissertation. We look at communication about *preferences* among *groups* of agents, and we are interested in the evolution of *knowledge* within such "group networks", and in how the agents can use that knowledge in order to choose their *actions*.

We formalize the assumptions we make later on. For now consider Figure 5,

and remember that Bob likes Ann and Carol likes Bob, and how their choices depend on each other. Shyness and jealousy forbid them to speak to each other explicitly about coordination and preferences, so let us assume that communication is limited, for example only through observing behavior.

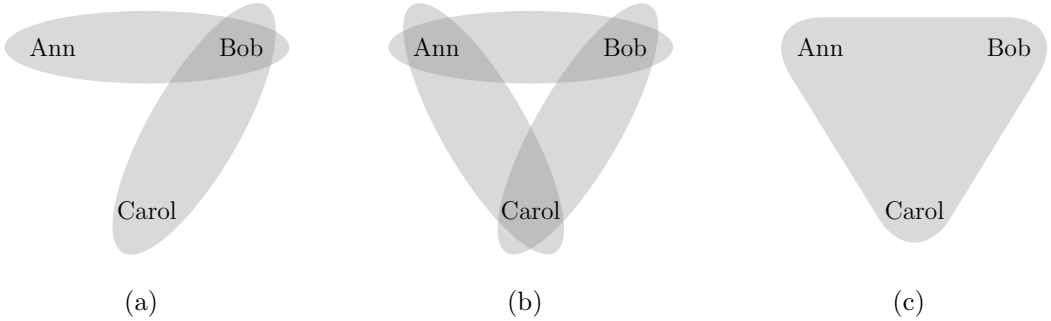


Figure 5: Three possible group configurations of Ann, Bob and Carol.

Figure 5 shows three possible group configurations of Ann, Bob and Carol. In (a), Bob shares a communication platform with Ann and another one with Carol. Through the former, he can learn about Ann's preferences, and through the latter, Carol can learn about Bob's preferences (e.g., through observations while pairwise eating at the same restaurant). However, Carol cannot learn about Ann's preferences. In (b), Carol can learn about Ann's preferences, but she cannot learn what Bob knows about Ann's preferences (and she is too jealous to ask him directly). Only in (c), where they all share the same communication platform, they can all commonly learn about each other. In particular, Carol can then learn about Ann's preferences, and also that Bob knows them. She will then be able to predict his choice of action, and choose her action accordingly.

Theory and Implementation

Epistemic logic and game theory have a strong focus on an *external* and *descriptive* point of view.

In epistemic logic, this is already reflected by the fact that there is *one* central model comprising *all* agents. It describes a modeler's perspective on *what* the agents can be said to know in a philosophical sense, and what they indeed will be able to figure out if they are perfect reasoners; but it does not necessarily say exactly *how* the agents in actuality arrive at that knowledge.[7]

Similarly, many concepts in game theory describe *what* situations rational players will end up in, but not exactly *how* they might get there.

[7]There is a sizable but as yet somewhat inconclusive literature in computer science and game theory concerning modeling of bounded rationality and reasoning capabilities; see, e.g., [59, 130]. We discuss these and related issues at various places, notably in Chapter 3, Section 3.6.

Without doubt, epistemic logic does provide elegant and general mechanisms to perform deductions and determine the truth value of statements. But those, again, feel more like a tool for the modeler, typically without any claim that they reflect what agents actually (can) do. So these mechanisms do not really try to take an agent's point of view and can, without any qualms, be arbitrarily complex.

This kind of criticism is not new. For example, Parikh [114] aims to model more closely what and how we as humans actually know and believe. However, the mainstream research focus does not lie on such approaches, nor on concrete algorithmic realizations of epistemic reasoning, for example, within artificial agents.

Also in game theory, certain algorithmic characterizations exist, the procedure of IESDS explained above being one example. However, IESDS still takes a centralized perspective and operates on the fully specified payoff matrix. Tan and Werlang [144], Brandenburger [30] and Börgers [28] have characterized mutual beliefs about each other's rationality and preferences that lead players to the outcome of IESDS, but so far it has not been examined just what players can do given some particular pieces of information, or how such information can be arrived at and processed.

In the main part of this dissertation, we want to take a *procedural* and *subjective* point of view: How might a player actually access his theoretically ascribed knowledge, and how might he obtain the game-theoretic solution?

To this end, we are interested in restricting the general theories and obtaining concrete implementations of simple fragments that can be proved correct with respect to the general theoretical foundations. The aim of this approach is to obtain practical implementations grounded in theory.

Chapter overview

Figure 6 shows how the chapters of this dissertation relate to its main concepts and to each other. The main part of the dissertation is formed by Chapters 1–3, which build upon each other and make a progression from knowledge and theory to games and implementation. The remaining satellite chapters have either been directly inspired by the main part (in the case of Chapter 4) or are related via the same main concepts (in the case of Chapters 5 and 6). In the following, we give more details on the contents of the chapters and the overall structure of the dissertation.

The basic idea of the main part (Chapters 1–3) is to view computer processes, or otherwise *distributed* programs, as players in a game-theoretic setting with incomplete information. As such, they should be able to communicate in order to obtain information, and to perform game-theoretic algorithms.

In particular, we focus on the IESDS algorithm described above, and a setting where each player initially knows only his own preferences. Players can communicate their preferences across a communication network, and as they

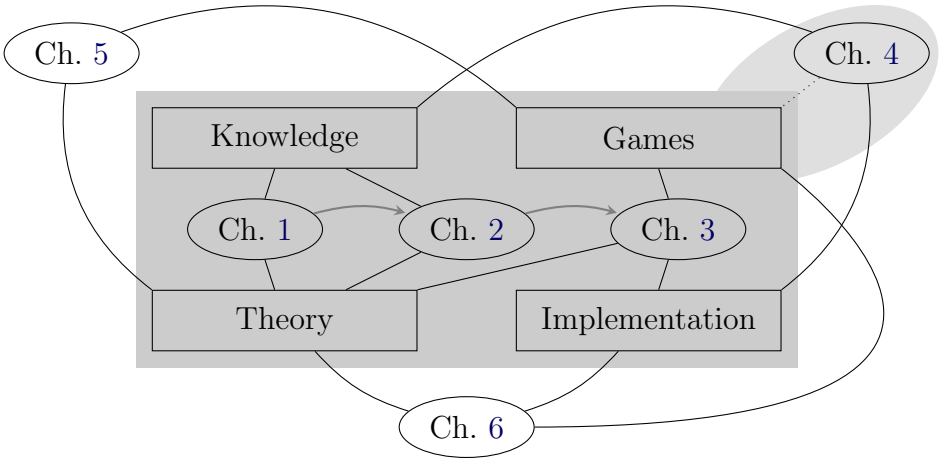


Figure 6: Structure of this dissertation.

communicate, their knowledge changes and they can obtain more conclusive elimination outcomes.

In order to obtain an actual implementation, we need to enable players to actually *compute* their knowledge so that they can *access* what they theoretically can be said to know. In order to make these computations efficient, we are especially interested in a framework where knowledge is *simple* to maintain and process. We therefore impose clear *restrictions* on the allowed communication, and, for the reasons mentioned above, focus on *synchronous* communication.

We have illustrated the underlying approach, which we call *explicit knowledge programming*,[8] in [155].

In Chapter 1 (Guarding common knowledge), we establish the technical foundations to support implementation of synchronous communication, and thus the attainment of common knowledge, among computer processes representing players. To this end, we examine dialects of a process calculus which is available in the form of programming languages.

We define a setting where processes are treated "on an equal footing", in the sense that none of them takes a special role in initiating or coordinating communication. This is a prerequisite for interpreting processes as players in arbitrary games, since the differences among them should only depend on the strategies and preferences of a particular given game, and not be predetermined through some *a priori* roles.

We then show that a certain guard construct is needed in the language in order to implement correct programs in such a symmetric setting. Since

[8] Our term *explicit knowledge programming* is somewhat related to *explicit knowledge* and *algorithmic knowledge* [113, 60]; however, it is mostly meant to contrast with *knowledge-based programs* [61]. We give some more discussion on this in Chapter 3, Section 3.6.

this construct is not commonly provided, our result practically identifies a unique programming language suitable for our purposes.

This chapter is an extended version of [154].

In Chapter 2 (Knowledge in interaction structures), we define what we call *interaction structures*, a concrete class of communication networks compatible with the findings from Chapter 1. We also specify what kind of communication scenario we focus on, consisting of possible initial situations and possible communication.

We then study properties of knowledge that results from such communication. In particular, we investigate what is the impact of common knowledge of the underlying interaction structure, and we establish that common knowledge distributes over disjunction for formulas without negation. These results can be used to simplify reasoning about knowledge in our setting.

This chapter builds on parts of [9], joint work with Krzysztof R. Apt and Jonathan A. Zvesper.

Chapter 3 (Strategies in interaction structures) then turns towards the game-theoretic and implementation-oriented side.

We study games in the presence of an interaction structure, which allows players to communicate their preferences, assuming that each player initially only knows his own preferences. We study the outcomes of IESDS that can be obtained in any given state of communication.

The insights from Chapter 2 are used in order to prove that the outcomes of IESDS which we establish indeed correctly reflect what the players know in any particular situation. Building upon Chapter 1, we then describe a distributed algorithm that implements IESDS locally in each player process.

This chapter extends unpublished joint work with Krzysztof R. Apt and Jonathan A. Zvesper.

This rounds off the main part of the dissertation and starts the more loosely related satellite chapters. The first of these has developed from the main part and is close to it in spirit, with the difference that it focuses on a *centralized* rather than a distributed approach, and that it considers *computer games* rather than games in the strict sense of game theory.

In Chapter 4 (Epistemic reasoning in computer games), we argue that reasoning about knowledge, including about each other's knowledge, plays a crucial role in real-life strategic and social interaction. We survey existing literature and games which simulate such interaction, and show that this issue is currently neglected.

We give concrete scenarios from existing computer games which could profit from incorporating such reasoning techniques and substantiate one of them by describing a simple implementation intended for experimental evaluation. Finally, we discuss a number of issues that arise when generalizing our approach, some of which go beyond the scope computer games and are of more general interest.

This chapter extends joint work with Jonathan A. Zvesper and Ethan Kennerly, published as [156, 157].

The last two satellite chapters return to game theory, and in particular to the areas of coalition formation and auctions.

Chapter 5 (Coalition formation: A generic approach) proposes an abstract approach to coalition formation that focuses on simple merge and split rules transforming partitions of a group of players.

We identify conditions under which every iteration of these rules yields a unique partition. The main conceptual tool is a specific notion of a stable partition.

The results are parametrized by a preference relation between partitions of a group of players and naturally apply to coalitional TU-games, hedonic games and exchange economy games.

This chapter is joint work with Krzysztof R. Apt, to appear as [8].

In Chapter 6 (Time constraints in mixed auctions), we extend the existing framework of mixed multi-unit combinatorial auctions to include time constraints, present an expressive bidding language, and show how to solve the winner determination problem for such auctions using an integer programming implementation.

Mixed multi-unit combinatorial auctions are auctions where bidders can offer combinations of transformations of goods rather than just simple goods. For example, a transformation might take dough and water and yield bread. This model has great potential for applications in the context of supply chain formation, which is further enhanced by the integration of time constraints.

We consider different kinds of time constraints: they may be based on either time points or intervals, they may determine a relative ordering of transformations, they may relate transformations to absolute time points, and they may constrain the duration of transformations.

This chapter is based on unpublished joint work with Ulle Endriss.

In Chapter 7 we give an outlook on possible future directions for implementing epistemic logic.

Chapter 1

Guarding common knowledge

1.1 Introduction

1.1.1 Motivation

As described in the Introduction chapter, our goal is a distributed implementation of a game-theoretic algorithm (see, e.g., [72] for a discussion of the interface between game theory and distributed computing). In this chapter, we lay the technical foundations to support such an implementation from a distributed computing point of view.

Two important issues in the domain of game theory are knowledge, especially common knowledge, and symmetry between the players, also called anonymity. We describe these issues and the connections to distributed computing in the following two paragraphs, before we motivate our choice of process calculus and the overall goal of the chapter.

Common knowledge and synchronization. The concept of *common knowledge*, first introduced by Lewis [92] building on ideas of Schelling [134], has been a topic of much research in distributed computing [73] as well as in game theory [12]. When do processes or players "know" some fact, mutually know that they know it, mutually know that they mutually know that they know it, and so on ad infinitum? And how crucial is the difference between arbitrarily, but finitely deep *mutual knowledge* and the limit case of real common knowledge?

In the area of distributed computing, the classical example showing that the difference is indeed essential is the scenario of **coordinated attack** described in the Introduction chapter. The game-theoretic incarnation of the underlying issue is the **electronic mail game** by Rubinstein [129] (see [102] for a more recent treatment).

The basic insight of these examples is that two agents who communicate through an *unreliable* channel can never achieve common knowledge, and that

their behavior under finite mutual knowledge can be strikingly different.

These issues have been analyzed in detail by Fagin et al. [60], in particular in a separately published part [73], including a variant where communication is *reliable*, but message delivery takes an unknown amount of time. Even in that variant, which has also been looked at by Parikh and Ramanujam [116], it is shown that only finite mutual knowledge can be attained.

However, in a *synchronous* communication act, sending and receiving of a message is, by definition, performed simultaneously. In that way, the agents obtain not only the pure factual information content of the message, but the sender also knows that the receiver has received the message, the receiver knows that the sender knows that, and so on ad infinitum. The communicated information immediately becomes common knowledge.

Attaining common knowledge and achieving synchronization between processes are thus closely related. Furthermore, synchronization is in itself an important subject (see, e.g., [135]).

Symmetry and peer-to-peer networks.[1] In game theory, it is usually assumed that players are *anonymous* and treated on an equal footing in the following sense: Any differences between them are only induced by the payoff structure and the information state in a given game, while their names do not play a role and no player is *a priori* distinguished from the others [106, 103]. If we want to set up a system that allows processes to incarnate players in any given game, we need to make sure that those processes are on an equal footing in a corresponding sense. Even though the interaction structures we introduce later can take any form, we need to be prepared for cases where the communication infrastructure is in a certain sense symmetric. Such symmetric settings, as we see in this chapter, constitute the crucial cases from the technical viewpoint of an implementation.

In distributed computing, a corresponding kind of *symmetry* among processes is often a desideratum. Reasons to avoid a predetermined assignment of roles to processes or a centralized coordinator include fault tolerance, modularity, and load balancing [3].

We consider symmetry on two levels. Firstly, we assume communication networks used by the processes to be symmetric to some extent in order not to discriminate single processes *a priori* on a topological level; we formalize this requirement by defining peer-to-peer networks. These networks are a special case of the interaction structures we use in Chapters 2 and 3, but our results carry over, as we discuss in Section 1.5. Secondly, processes in symmetric positions of the network should have equal possibilities of behavior; this we formalize in a semantic symmetry requirement on the possible computations.

[1] Please note that we are *not* dealing with fashionable incarnations such as file-sharing networks, but merely use this name for a mathematical notion of a network consisting of directly connected peers "treated on an equal footing", i.e., not having a client-server structure or otherwise pre-determined roles.

Communicating Sequential Processes (CSP). Since we are interested in synchronization and common knowledge, a process calculus which supports synchronous communication through primitive statements clearly has some appeal. We focus on one of the prime examples of such calculi, namely CSP, introduced by Hoare [79] and later revised [80, 136]. It allows synchronous communication by means of deterministic statements on the one hand and non-deterministic alternatives on the other hand, where the communication statements occur in so-called *guards*.

CSP has been implemented in various programming languages, among the best-known of which is *Occam* [82]. We thus have at our disposal a theoretical framework and programming tools which in principle could give us synchronization and common knowledge "for free".

However, symmetric situations are a reliable source of impossibility results, see [63] for a recent collection. There exist different dialects of CSP, which differ in what communication statements are allowed to appear in guards. We define the relevant dialects later on, for now suffice it to say that the dialect CSP_{in} which was, for implementation-related reasons [33], chosen to be the theoretical foundation of Occam is more sensitive to symmetric situations than the general form $\text{CSP}_{i/o}$. This has been proved formally by Bougé [29].

CSP_{in} has been used throughout the history of Occam, up to and including its latest variant *Occam-π* [150]. This restriction to CSP_{in} generally tends to be the case for implementations of CSP. One notable exception is a recent extension of JCSP, a Java™ implementation of CSP, by Welch et al. [151].

Some of the restrictions resulting in CSP_{in} can in practice be overcome by using helper processes such as buffers [84]. It is conceivable that such processes could be used as mediators to coordinate and establish direct and synchronous communication among the main processes. Therefore, our goal is to formalize the concepts mentioned above, extend the notion of peer-to-peer networks by allowing helper processes, and examine whether synchronization is feasible in either of these two dialects of CSP. We come to the result that, while it can (straightforwardly) be obtained in $\text{CSP}_{i/o}$, it is impossible to do so in CSP_{in}. For the setting of this dissertation, we thus need to use one of the rare more general implementations such as JCSP.

1.1.2 Related work

We extend work by Bougé [29], where a semantic characterization of symmetry for CSP is given and fundamental possibility and impossibility results for the problem of electing a leader in networks of symmetric processes are proved for various dialects of CSP. More recently, this has inspired a similar work by Palamidessi [110] on the more expressive π-calculus, but the possibility of adding helper processes is explicitly excluded.

There has been research on how to circumvent problems resulting from the

restrictions of CSP_{in}. However, solutions are typically concerned only with the factual content of messages and do not preserve synchronicity and the common knowledge creating effect of communication, for example by relaying communication through buffers [84].

The same focus on factual information holds for general research on synchronizing processes with asynchronous communication. For example, in [135] one goal is to ensure that a writing process knows that no other process is currently writing; whether this is common knowledge, is not an issue.

The problem of coordinated attack has also been studied for models in which processes run synchronously [63]; however, the interesting property of CSP is that processes run asynchronously, which is more realistic in physically distributed systems, and synchronize only at communication statements.

Since we focus on the communication mechanisms, our results carry over to other formalisms with synchronous communication facilities comparable to those of CSP.

1.1.3 Plan of the chapter

In Section 1.2 we give a short description of CSP and the dialects that we are interested in, define some basic concepts from graph theory, and recall the required notions and results for symmetric electoral systems by Bougé [29].

In Section 1.3 we set the stage by first formally defining the problem of pairwise synchronization that we examine. Subsequently, we give a formalization of peer-to-peer networks which ensures a certain kind of symmetry on the topological level, and describe in what ways we want to allow them to be extended by helper processes. Finally, we adapt a concept from [29] to capture symmetry on the semantic level.

Section 1.4 contains two positive results and the main negative result saying that pairwise synchronization of peer-to-peer networks of symmetric processes is not obtainable in CSP_{in}, even if we allow extensions through buffers or similar helper processes which might mediate between the main processes.

Section 1.5 offers a concluding discussion.

1.2 Preliminaries

We briefly review the required concepts and results from the CSP calculus in Section 1.2.1, from graph theory in Section 1.2.2, and from [29] in Section 1.2.3. For more details see [79, 117, 29].

1.2.1 CSP

A **CSP process** consists of a sequential program which can use, besides the usual *local* statements (e.g. assignments or expression evaluations involving its local variables), two *communication* statements:

- $P\,!\,message$ to send (output) the given message to process P;
- $P\,?\,variable$ to receive (input) a message from process P and store it in the given (local) variable.

Communication is *synchronous*, i.e., send and receive instructions block until their counterpart is available, at which point the message is transferred and both participating processes continue execution. Note that the communication partner P is statically defined in the program code.

There are two *control structures* containing guarded commands, see Figure 1.1. A **guard** is a Boolean expression over local variables (which, if omitted, is taken to be true), optionally followed by a communication statement. A guard is *open* if its Boolean expression evaluates to true and its communication statement, if any, can currently be performed. A guard is *closed* if its Boolean expression evaluates to false. Note that a guard can thus be neither open nor closed.

$$
\begin{array}{l}
[\ \ guard_1 \to command_1 \\
\Box\ guard_2 \to command_2 \\
\ldots \\
\Box\ guard_k \to command_k\ \ ]
\end{array}
$$

(a) Non-deterministic selection

$$
\begin{array}{l}
*[\ \ guard_1 \to command_1 \\
\ \ \Box\ guard_2 \to command_2 \\
\ \ \ldots \\
\ \ \Box\ guard_k \to command_k\ \ ]
\end{array}
$$

(b) Non-deterministic repetition

Figure 1.1: Control structures in CSP.

The selection statement *fails* and execution is aborted if all guards are closed. Otherwise execution is suspended until there is at least one open guard. Then one of the open guards is selected non-deterministically, the required communication (if any) performed, and the associated command executed.

The repetition statement keeps waiting for, selecting, and executing open guards and their associated commands until all guards are closed, and then exits normally; that is, program execution continues at the next statement.

We sometimes use the following abbreviation to denote multiple branches of a control structure (for some finite set X):

$\Box_{x \in X} guard_x \to command_x$

Various dialects of CSP can be distinguished according to what kind of communication statements are allowed to appear in guards. Specifically, in CSP_{in} only input statements are allowed, and in $CSP_{i/o}$ both input and output statements

are allowed (within the same control structure). For technical reasons, CSP_{in} has been suggested from the beginning [79] and is indeed commonly used for implementations, as mentioned in Section 1.1.1.

1.2.1. Definition. A **communication graph** (or **network**) is a directed graph without self-loops. A **process system** (or simply **system**) $\mathcal{P}$ with communication graph $G = (V, E)$ is a set of component processes $\{P_v\}_{v \in V}$ such that for all $v, w \in V$, if the program run by P_v (respectively P_w) contains an output command to P_w (respectively input command from P_v) then $(v, w) \in E$. In that case we say that G **admits** $\mathcal{P}$. We identify vertices v and associated processes P_v and use them interchangeably. □

1.2.2. Example. Figure 1.2 shows a simple network G with the vertex names written inside the vertices, and a $\mathrm{CSP}_{i/o}$ program run by two processes which make up a system $\mathcal{P} := \{P_0, P_1\}$. Obviously, G admits $\mathcal{P}$. The intended behavior is that the processes send each other, in non-deterministic order, a message containing their respective process name. Since communication is synchronous, it is guaranteed that both processes execute each communication statement synchronously at the time when the message is transmitted. In a larger context, executing this code fragment would have the effect that the participating processes synchronize, i.e., wait for each other and jointly perform the communication. In terms of knowledge, this fact as well as the transmitted message (which can of course be more interesting than just the process names) become common knowledge between the processes.

(a) Network G

$$
\begin{aligned}
&\mathit{recd} := \text{false}\\
&\mathit{sent} := \text{false}\\
*[\;&\neg \mathit{recd} \wedge P_{i+1}\,?\,x \to \mathit{recd} := \text{true}\\
\Box\;&\neg \mathit{sent} \wedge P_{i+1}\,!\,i \to \mathit{sent} := \text{true}\;\;]
\end{aligned}
$$

(b) Program of process P_i

Figure 1.2: Network and program run by P_0 and P_1 in Example 1.2.2. Addition within process names here and in all further example programs is modulo 2.

1.2.3. Definition. A **state** of a system $\mathcal{P}$ is the collection of all component processes' (local) variables together with their current execution positions. A **computation step** is a transition from one state to another, involving either one component process executing a local statement, or two component processes jointly executing a pair of matching (send and receive) communication statements. The possible next computation steps are thus determined by the current state of the system.

A **computation** is a maximal sequence of computation steps, i.e., a sequence which is not a prefix of any other sequence of computation steps. A computation

- is **properly terminated** if all component processes have completed their last instruction,
- **diverges** if it is infinite, and
- is in **deadlock** if it is finite but not properly terminated.

□

1.2.4. Example. Figure 1.3 shows a computation of the system from Figure 1.2. It is finite and both processes reach the end of their respective program, so it is properly terminated. Note that the exact order in which, for example, the processes get to initialize their local variables is non-deterministic, so there are other computations with these steps exchanged. Only certain restrictions to the order apply, e.g. that the steps within one process are ordered corresponding to its program, or that both processes must have evaluated the Boolean guards before they can participate in the subsequent communication.

P_0 : assign false to *recd*
P_1 : assign false to *recd*
P_1 : assign false to *sent*
P_0 : assign false to *sent*
P_1 : evaluate Boolean guards
P_0 : evaluate Boolean guards
P_0, P_1 : send 0 from P_0 to P_1's variable x
P_0 : assign true to *sent*
P_0 : evaluate Boolean guards
P_1 : assign true to *recd*
P_1 : evaluate Boolean guards
P_0, P_1 : send 1 from P_1 to P_0's variable x
P_1 : assign true to *sent*
P_0 : assign true to *recd*
P_0 : evaluate Boolean guards and exit repetition
P_1 : evaluate Boolean guards and exit repetition

Figure 1.3: A properly terminating computation of the system from Example 1.2.2.

1.2.2 Graph theory

We state some fundamental notions concerning directed finite graphs, from here on simply referred to as **graphs**.

1.2.5. Definition. Two vertices $a, b \in V$ of a graph $G = (V, E)$ are **strongly connected** if there are paths from a to b and from b to a consisting of edges in E. G is strongly connected if all pairs of vertices are.

Two vertices $a, b \in V$ are **directly connected** if $(a, b) \in E$ or $(b, a) \in E$; G is directly connected if all pairs of vertices are. □

1.2.6. Definition. An **automorphism** of a graph $G = (V, E)$ is a permutation σ of V such that for all $v, w \in V$,

$$(v, w) \in E \text{ implies } (\sigma(v), \sigma(w)) \in E.$$

The *automorphism group* Σ_G of a graph G is the set of all automorphisms of G. The least $p > 0$ with $\sigma^p = \mathrm{id}$ is called the **period** of σ, where by id we denote the identity function defined on the domain of whatever function it is compared to.

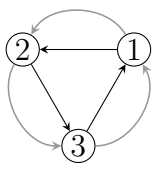
The **orbit** of $v \in V$ under $\sigma \in \Sigma_G$ is $O_v^\sigma := \{\sigma^p(v) | p \geq 0\}$. An automorphism σ is **well-balanced** if the orbits of all vertices have the same cardinality, or alternatively, if for all $p \geq 0$,

$$\sigma^p(v) = v \text{ for some } v \in V \text{ implies } \sigma^p = \mathrm{id}.$$

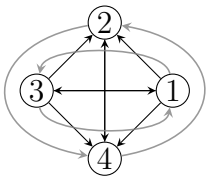
We usually consider the (possibly empty) set $\Sigma_G^{\mathrm{wb}} \setminus \{\mathrm{id}\}$ of *non-trivial* well-balanced automorphisms of a graph G, that is those with period greater than 1.

A subset $W \subseteq V$ is called **invariant** under $\sigma \in \Sigma_G$ if $\sigma(W) = W$, i.e., if W is an orbit under σ; it is called invariant under Σ_G if it is invariant under all $\sigma \in \Sigma_G$. □

1.2.7. Example. Figure 1.4 shows two graphs G and H and automorphisms $\sigma \in \Sigma_G$ with period 3 and $\tau \in \Sigma_H$ with period 2. Both are well-balanced since, e.g., $O_1^\tau = O_3^\tau = \{1, 3\}$ and $O_2^\tau = O_4^\tau = \{2, 4\}$ all have the same cardinality. We have $\Sigma_H = \{\mathrm{id}, \tau\}$, so $\{1, 3\}$ and $\{2, 4\}$ are invariant under Σ_H.



(a) Graph G, $\sigma \in \Sigma_G$



(b) Graph H, $\tau \in \Sigma_H$

Figure 1.4: Two graphs with non-trivial well-balanced automorphisms, indicated by gray, bent arrows.

1.2.3 Symmetric electoral systems

We take over the semantic definition of symmetry from Bougé [29]. As discussed there, syntactic notions of symmetry are difficult to formalize properly; requiring that "all processes run the same program" does not do the job. We skip some formal details which we are not going to use. The interested reader is referred to [29].

1.2.8. Definition (adapted from [29, Definition 2.2.2]). A system $\mathcal{P}$ with communication graph $G = (V, E)$ is **symmetric** if for each automorphism $\sigma \in \Sigma_G$ and each computation C of $\mathcal{P}$, there is a computation C' of $\mathcal{P}$ satisfying the following condition: For each $v \in V$, process $P_{\sigma(v)}$ performs the same steps in C' as P_v in C, modulo changing via σ the process names occurring in the computation (e.g. as communication partners). □

The intuitive interpretation of this symmetry notion is as follows. Any two processes which are not already distinguished by the communication graph G itself, i.e., which are related by some automorphism, must have equal possibilities of behavior. That is, whatever behavior one process exhibits in some particular possible execution of the system (i.e., in some computation), the other process must exhibit in some other possible execution of the system, localized to its position in the graph by appropriate process renaming. Taken back to the syntactic level, this is the case, for example, if both processes run the same program, which does not make use of any externally given distinctive features like an ordering on the process names.

1.2.9. Example. The system from Figure 1.2 is symmetric. It is easy to see that, for example, if we swap all 0s and 1s in the computation shown in Figure 1.3, we still have a computation of $\mathcal{P}$. Note that programs are allowed to access the process names, and indeed they do; however, they do not, for example, use their natural order to determine which process sends first.

1.2.10. Example. On the other hand, consider the system $\mathcal{Q} = \{Q_0, Q_1\}$ running the program in Figure 1.5. There is obviously a computation where Q_0 sends its process name 0 to Q_1; since the two vertices of the communication graph are related by an automorphism, symmetry would require that there also be a computation where Q_1 sends its process name 1 to Q_0. However, such a computation does not exist due to the use of the process name for determining the communication role, so the system is not symmetric.

$$\begin{array}{l}[\quad i = 0 \to Q_{i+1}\,!\,i \\ \Box\ i = 1 \to Q_{i+1}\,?\,x\quad]\end{array}$$

Figure 1.5: Asymmetric program run by Q_0 and Q_1 in Example 1.2.10.

We now recall a classical problem for networks of processes, pointed out by Le Lann [90].

1.2.11. Definition (from [29, Definition 1.2.1]). A system $\mathcal{P}$ is an **electoral system** if

(i) all computations of $\mathcal{P}$ are properly terminating and

(ii) each process of $\mathcal{P}$ has a local variable `leader`, and at the time of termination all these variables contain the same value, the name of some process $P \in \mathcal{P}$.

□

We now restate the impossibility result which our work builds on, combining a graph-theoretical characterization with the symmetry notion and electoral systems.

1.2.12. Theorem (from [29, Theorem 3.3.2]). *Suppose a network G admits some well-balanced automorphism σ different from* id. *Then G admits no symmetric electoral system in CSP_{in}.*

1.3 Setting the stage

1.3.1 Pairwise synchronization

Intuitively, if we look at synchronization as part of a larger system, a process is able to synchronize with another process if it can execute an algorithm such that a direct communication (of any message) between the two processes takes place. This may be the starting point of some communication protocol to exchange more information, or simply be taken as an event creating common knowledge about the processes' current progress of execution.

Communication in CSP always involves exactly two processes and facilities for synchronous broadcast do not exist, thus synchronization is inherently pairwise only. This special case is interesting nevertheless and has been used as a setting to examine knowledge-related issues, e.g., by Parikh and Krasucki [115]. Note that JCSP supports broadcasts as an extension of communication to multiple recipients, and as such can accommodate the interaction structures we deal with in Chapters 2 and 3.

Focusing on the synchronization algorithm, we want to guarantee that it allows all pairs of processes to synchronize. To this end, we require existence of a system where in all computations, all pairs of processes synchronize. Most probably, in a real system not all pairs of processes need to synchronize in all executions. However, if one has an algorithm which in principle allows that, then one could certainly design a system where they actually do; and, vice versa, if one has a system which is guaranteed to synchronize all pairs of processes, then one can

obviously use its algorithms to synchronize any given pair. Therefore we use the following formal notion.

1.3.1. Definition. A system $\mathcal{P}$ of processes **(pairwise) synchronizes** $\mathcal{Q} \subseteq \mathcal{P}$ if all computations of $\mathcal{P}$ are finite and properly terminating and contain, for each pair $P_a, P_b \in \mathcal{Q}$, at least one direct communication from P_a to P_b or from P_b to P_a. □

1.3.2. Example. The system from Figure 1.2 synchronizes $\{P_0, P_1\}$.

Note that the program considered so far is not a valid CSP$_{in}$ program, since an output statement appears within a guard. If we want to restrict ourselves to CSP$_{in}$ (for example, to implement the program in Occam), we have to get rid of that statement. Attempts to simply move it out of the guard fail since the symmetric situation inevitably leads to a system which may deadlock.

To see this, consider the system $\mathcal{P}' = \{P'_0, P'_1\}$ with the program shown in Figure 1.6. There is no guarantee that not both processes enter the second clause of the repetition at the same time, since it is now only guarded by a local variable, and then block forever at the output statement, waiting for each other to become ready for input. A standard workaround [84] for such cases is to introduce *buffer* processes mediating between the main processes, in our case resulting in the extended system $\mathcal{R} = \{R_0, R'_0, R_1, R'_1\}$ shown in Figure 1.7.

$$
\begin{array}{l}
\mathit{recd} := \text{false} \\
\mathit{sent} := \text{false} \\
*[\ \ \neg \mathit{recd} \wedge P'_{i+1}\,?\,x \to \mathit{recd} := \text{true} \\
\ \ \square\ \neg \mathit{sent} \to P'_{i+1}\,!\,i;\ \mathit{sent} := \text{true}\ \ ]
\end{array}
$$

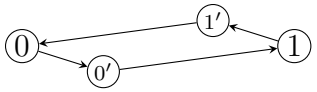
Figure 1.6: Program of process P'_i potentially causing deadlock.

$$
\begin{array}{l}
\mathit{recd} := \text{false} \\
\mathit{sent} := \text{false} \\
*[\ \ \neg \mathit{recd} \wedge R'_{i+1}\,?\,x \to \mathit{recd} := \text{true} \\
\ \ \square\ \neg \mathit{sent} \to R'_i\,!\,i;\ \mathit{sent} := \text{true}\ \ ]
\end{array}
$$

(a) Program of main process R_i

$$
\begin{array}{l}
R_i\,?\,y \\
R_{i+1}\,!\,y
\end{array}
$$

(b) Program of buffer process R'_i



(c) Underlying communication network

Figure 1.7: Extended system with main processes R_0 and R_1 and buffer processes R'_0 and R'_1 together with the underlying communication network.

While the actual data transmitted between the main processes remains the same, this system obviously cannot synchronize $\{R_0, R_1\}$, since there is not even a direct link in the communication network. This removes the synchronizing and common knowledge creating effects of communication. Even though a buffer might notify its main process when its message is delivered, then notify the communication partner about the notification, and so on, synchronicity is not restored and mutual knowledge only attained to a finite (if arbitrarily high) level, as seen in the coordinated attack example discussed in the Introduction chapter.

The obvious question now is: Is it possible to change the program or use buffer or other helper processes in more complicated and smarter ways to negotiate between the main processes and aid them in establishing direct communications?

To attack this question, in the following Section 1.3.2 we formalize the kind of communication networks we are interested in and define how they may be extended in order to allow for helper processes without affecting the symmetry inherent in the original network.

1.3.2 Peer-to-peer networks

The idea of peer-to-peer networks is to have nodes which can communicate with each other directly and on an equal footing, i.e., there is no predetermined client/server architecture or central authority coordinating the communication. We first formalize the topological prerequisites for this, and then adapt the semantic symmetry requirement to our setting.

1.3.3. Definition. A **peer-to-peer network** is a communication graph $G = (V, E)$ with at least two vertices (also called nodes) such that

(i) G is strongly connected,

(ii) G is directly connected, and

(iii) we have $\Sigma_G^{\mathrm{wb}} \setminus \{\mathrm{id}\} \neq \emptyset$.

□

In this definition,

(i) says that each node has the possibility to contact (at least indirectly) any other node, reflecting the fact that there are no predetermined client/server roles;

(ii) ensures that all pairs of nodes have a direct connection at least in one direction, without which pairwise synchronization by definition would be impossible; and

(iii) requires a kind of symmetry in the network.

This last item is implied by the more intuitive requirement that there be some $\sigma \in \Sigma_G$ with only one orbit, i.e., an automorphism relating all nodes to each other and thus making sure that they are topologically on an equal footing. The requirement we use is less restrictive and suffices for our purposes.

1.3.4. Example. See Figure 1.4 for two examples of peer-to-peer networks.

We consider extensions of a peer-to-peer network in order to allow for helper processes while preserving the symmetry inherent in the network. The intuitive background for this kind of extensions is that we view the peers, i.e., the nodes of the original network, as processors each running a main process, while the added nodes can be thought of as helper processes running on the same processor as their respective main process. Communication connections between processors are physically given, while inside a processor they can be created as necessary.

1.3.5. Definition. Let $G = (V, E)$ be a peer-to-peer network, then $G' = (V', E')$ is a **symmetry-preserving extension of** G iff there is a collection $\{S_v\}_{v \in V}$ partitioning V' such that

(i) for all $v \in V$, we have $v \in S_v$;

(ii) all $v \in V$ and $v' \in S_v \setminus \{v\}$ are strongly connected (possibly via nodes $\notin S_v$);

(iii) for all $v, w \in V$, $E' \cap (S_v \times S_w) \neq \emptyset$ iff $(v, w) \in E$;

(iv) there is, for each $\sigma \in \Sigma_G$, an automorphism $\iota_\sigma \in \Sigma_{G'}$ extending σ such that $\iota_\sigma(S_v) = S_{\sigma(v)}$ for all $v \in V$.

Note that, in general, the collection $\{S_v\}_{v \in V}$ may not be unique. When we refer to it, we implicitly fix an arbitrary one. □

Intuitively, these requirements are justified as follows:

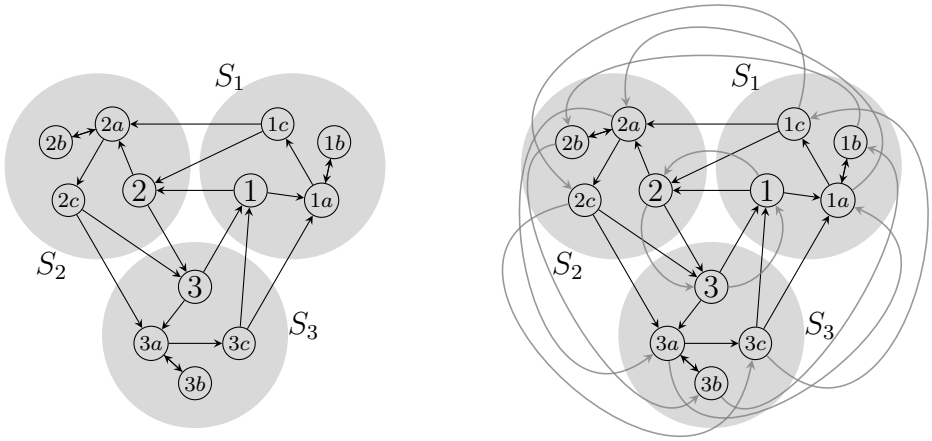
(i) Each S_v can be seen as the collection of processes running on the processor at vertex v, including its main process P_v.

(ii) The main process should be able to communicate (at least indirectly) in both ways with each helper process.

(iii) While communication links within one processor can be created freely, links between processes on different processors are only possible if there is a physical connection, that is a connection in the original peer-to-peer network; also, if there was a connection in the original network, then there should be one in the extension in order to preserve the network structure.

(iv) Lastly, to preserve symmetry, each automorphism of the original network must have an extension which maps all helper processes to the same processor as their corresponding main process.

1.3.6. Example. See Figure 1.8 for an example of a symmetry-preserving extension. Note that condition (iii) of Definition 1.3.5 is liberal enough to allow helper processes to communicate directly with processes running on other processors, and indeed, e.g. $2c$ has a link to 3. It also allows several communication links on one physical connection, reflected by the fact that there are three links connecting S_2 to S_3. Furthermore, (ii) is satisfied in that the main processes are strongly connected with their helper processes, although, as e.g. with 2 and $2c$, indirectly and through processes on other processors.



(a) Symmetry-preserving extension of the network from Figure 1.4(a).

(b) Extended automorphism ι_σ as required by Definition 1.3.5.

Figure 1.8: A symmetry-preserving extension (illustrating Definition 1.3.5).

We need the following immediate fact later on.

1.3.7. Fact. As a direct consequence of Definitions 1.3.3 and 1.3.5, any symmetry-preserving extension of a peer-to-peer network is strongly connected. □

1.3.3 G-symmetry

Corresponding to the intuition of processors with main and helper processes, we weaken Definition 1.2.8 such that only automorphisms are considered which keep the set of main processes invariant and map helper processes to the same processor as their main process. There are cases where the main processor otherwise would be required to run the same program as some helper process.

1.3.8. Definition. A system $\mathcal{P}$ whose communication graph G' is a symmetry-preserving extension of some peer-to-peer network $G = (V, E)$ is called **G-symmetric** if Definition 1.2.8 holds with respect to those automorphisms $\sigma \in \Sigma_{G'}$ satisfying, for all $v \in V$,

(i) $\sigma(V) = V$, and

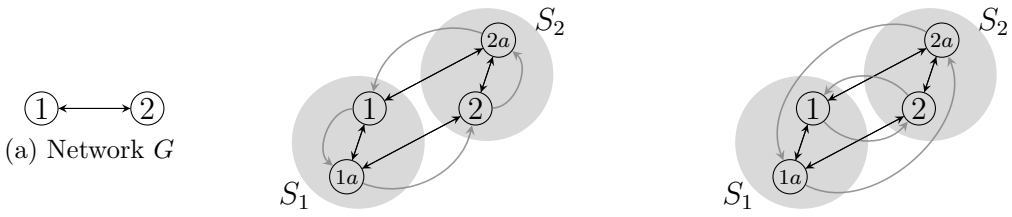
(ii) $\sigma(S_v) = S_{\sigma(v)}$.

□

This is weaker than Definition 1.2.8, since there we require the condition to hold for all automorphisms.

1.3.9. Example. To illustrate the impact of G-symmetry, Figure 1.9 shows a network G and an extension where symmetry relates all processes with each other. G-symmetry disregards the automorphism which causes this and considers only those which keep the set of main processes invariant, i.e., the nodes of the original network G, thus allowing them to behave differently from the helper processes.

Note that the main processes do not have a direct connection in the extension, which is permitted by Definition 1.3.5 although it will obviously make it impossible for them to synchronize.



(a) Network G

(b) Extension of G and an automorphism mixing main and helper processes

(c) Extension of G and the only automorphism taken into account by G-symmetry

Figure 1.9: A network G and an extension which has an automorphism mixing main and helper processes, disregarded by G-symmetry.

Now that we have formalized peer-to-peer networks and the symmetry-preserving extensions which we want to allow, we are ready to prove positive and negative results about feasibility of pairwise synchronization.

1.4 Results

1.4.1 Positive results

First, we state the intuition foreshadowed in Section 1.3.1, namely that $\text{CSP}_{i/o}$ does allow for symmetric pairwise synchronization in peer-to-peer networks.

1.4.1. Theorem. *Let $G = (V, E)$ be a peer-to-peer network. Then G admits a symmetric system pairwise synchronizing V in $CSP_{i/o}$.*

Proof. A system which at each vertex $v \in V$ runs the program shown in Figure 1.10 is symmetric and pairwise synchronizes V. Each process simply waits for each other process in parallel to become ready to send or receive a dummy message, and exits once a message has been exchanged with each other process. □

for each $w \in V$ do $sync_w := \text{false}$
$W_{in} := \{w \in V | (w, v) \in E\}$
$W_{out} := \{w \in V | (v, w) \in E\}$
$*[$
$\square_{w \in W_{in}} \neg sync_w \wedge P_w \,?\, x \rightarrow sync_w := \text{true}$
$\square_{w \in W_{out}} \neg sync_w \wedge P_w \,!\, 0 \rightarrow sync_w := \text{true}$
$]$

Figure 1.10: The program run at each vertex $v \in V$ in the proof of Theorem 1.4.1.

As a second result, we show that by dropping the topological symmetry requirement for peer-to-peer networks, under certain conditions we can achieve symmetric pairwise synchronizing systems even in CSP_{in}.

1.4.2. Theorem. *Let $G = (V, E)$ be a network satisfying only the first two conditions of Definition 1.3.3, i.e., G is strongly connected and directly connected. If G admits a symmetric electoral system and there is some vertex $v \in V$ such that $(v, a) \in E$ and $(a, v) \in E$ for all $a \in V$, then G admits a symmetric system pairwise synchronizing V in CSP_{in}.*

Proof. First, the electoral system is run to determine a temporary leader v'. When the election has terminated, v' chooses a coordinator v that is directly and in both directions connected to all other vertices, and broadcasts its name. Broadcasting can be done by choosing a spanning tree and transmitting the broadcast information together with the definition of the tree along the tree, as in the proof of [29, Theorem 2.3.1, Phase 2] (the strong connectivity which is required there holds for G by assumption). After termination of this phase, the other processes each send one message to v and then wait to receive commands from v according to which they perform direct communications with each other, while v receives one message from each other process and uses the obtained order to send out the commands.

This can be achieved by running the following program at each process P_c, $c \in V$, after having elected the temporary leader v':

- If $c = v'$, choose some $v \in V$ such that $(v, a) \in E$ and $(a, v) \in E$ for all $a \in V$, and broadcast the name v; otherwise obtain the broadcast name.
- If $c = v$:

– Receive exactly one message from each other process in some non-deterministic order and remember the order:

$W := V \setminus \{v\}$
for each $w \in W$ do $order_w := -1$
$count := 0$
$*[\ \ \square_{w \in W} order_w = -1 \wedge P_w\,?\,x \rightarrow$
$order_w := count$
$count := count + 1$
$]$

– Issue commands to the other processes according to the obtained order:

for each $a, b \in V \setminus \{v\}, a \neq b$ do
$[\ \ order_a < order_b \wedge (a,b) \in E \rightarrow$
$P_a\,!$ "contact b"
$P_b\,!$ "listen to a"
$\square\ order_a \geq order_b \vee (a,b) \notin E \rightarrow$
$P_b\,!$ "contact a"
$P_a\,!$ "listen to b"
$]$
done

otherwise (i.e., $c \neq v$):

– Send dummy message to P_v:

$P_v\,!\,0$

– Execute the commands from v until one message has been exchanged with each other process:

$num := |V \setminus \{c, v\}|$
$*[\ \ num > 0 \wedge P_v?m \rightarrow$
$[\ \ m =$ "contact w" $\rightarrow P_w\,!\,0$
$\square\ m =$ "listen to w" $\rightarrow P_w\,?\,x$
$]$
$num := num - 1$
$]$

□

1.4.3. Example. See Figure 1.11 for an example of a network which admits a symmetric system pairwise synchronizing all its vertices in CSP_{in}. The fact that the network admits a symmetric electoral system can be established as for [29, Fig. 4]. There the property is used that $\{1, 2\}$ and $\{3, 4, 5\}$ are invariant under the network's automorphism group and the associated processes can thus behave differently; this property is not affected by the edges we have added (note that the edges between the lower nodes are only in one direction).

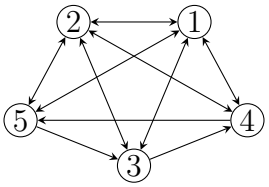


Figure 1.11: A network which by Theorem 1.4.2 admits a symmetric system pairwise synchronizing all its vertices in CSP_{in}. Note that the connections between vertices 3, 4 and 5 are only in one direction.

This result could be generalized, e.g. by weakening the conditions on v and taking care that the commands will reach the nodes at least indirectly. Since our main focus is the negative result, we do not pursue this further.

1.4.2 Negative result

In the following we establish the main result of this chapter, saying that, even if we extend a peer-to-peer network G by helper processes (in a symmetry-preserving way), it is not possible to obtain a network which admits a G-symmetric system pairwise synchronizing the nodes of G in CSP_{in}.

To this end, we derive a contradiction with Theorem 1.2.12 by proving the following intermediate steps (let G denote a peer-to-peer network and G' a symmetry-preserving extension):

- Lemma 1.4.4: If G' admits a G-symmetric system pairwise synchronizing the nodes of G in CSP_{in}, it admits a G-symmetric electoral system in CSP_{in}.
- Lemma 1.4.5: G' has a non-trivial well-balanced automorphism taken into account by G-symmetry (i.e., satisfying the two conditions of Definition 1.3.8).
- Lemma 1.4.7: We can extend G' in such a way that there exists a non-trivial well-balanced automorphism (derived from the previous result), G-symmetry is reduced to symmetry, and admittance of an electoral system is preserved.

1.4.4. Lemma. *If some symmetry-preserving extension of a peer-to-peer network $G = (V, E)$ admits a G-symmetric system pairwise synchronizing V in CSP_{in}, then it admits a G-symmetric electoral system in CSP_{in}.*

Proof. The following steps describe the desired electoral system (using the fact that under G-symmetry processes of nodes $\in V$ may behave differently from those of nodes $\notin V$):

- All processes run the assumed G-symmetric pairwise synchronization program, with the following modification for the processes in $\mathcal{P} := \{P_v | v \in V\}$ (intuitively this can be seen as a kind of knockout tournament, similar to the proof of [29, Theorem 4.1.2, Phase 1]):

  - Each of these processes has an additional local variable `winning` initialized to `true`.
  - After each communication statement with some other $P \in \mathcal{P}$, insert a second communication statement with P in the same direction:
    * If it was a "send" statement, send the value of `winning`.
    * If it was a "receive" statement, receive a Boolean value, and if the received value is `true`, set `winning` to `false`.

  Note that, since the program pairwise synchronizes V, each pair of processes associated to vertices in V has had a direct communication at the end of execution, and thus there is exactly one process in the whole system which has a local variable `winning` containing `true`.

- After the synchronization program terminates the processes check their local variable `winning`. The unique process that still has value `true` declares itself the leader and broadcasts its name; all processes set their variable `leader` accordingly. As in the proof of Theorem 1.4.2, broadcasting can be done using a spanning tree. The required strong connectivity is guaranteed by Fact 1.3.7. □

1.4.5. Lemma. *For any symmetry-preserving extension $G' = (V', E')$ of a peer-to-peer network $G = (V, E)$, there is $\sigma' \in \Sigma^{\mathrm{wb}}_{G'} \setminus \{\mathrm{id}\}$ such that $\sigma'(V) = V$ and $\sigma'(S_u) = S_{\sigma'(u)}$ for all $u \in V$.*

Proof. Take an arbitrary $\sigma \in \Sigma^{\mathrm{wb}}_{G} \setminus \{\mathrm{id}\}$ (exists by Definition 1.3.3) and let ι, to save indices, denote the ι_σ required by Definition 1.3.5. If $\iota \in \Sigma^{\mathrm{wb}}_{G'} \setminus \{\mathrm{id}\}$ we are done; otherwise we can construct a suitable σ' from ι by "slicing" orbits of ι which are larger than the period of σ into orbits of that size. See Example 1.4.6 for an illustration of the following proof.

Let p denote the period of σ and pick an arbitrary $v \in V$. For simplicity, we assume that σ has only one orbit; if it has several, the proof extends straightforwardly by picking one v from each orbit and proceeding with them in parallel.

For all $u \in S_v$ let $p_u := |O^\iota_u|$ and note that for all $t \in O^\iota_u$ we have $p_t = p_u$, and $p_u \geq p$ since ι maps each S_v to $S_{\sigma(v)}$ and these sets are pairwise disjoint. We define $\sigma' : V' \to V'$ as follows:

$$\sigma'(u) := \begin{cases} \iota^{p_u - p + 1}(u) & \text{if } u \in S_v \\ \iota(u) & \text{otherwise.} \end{cases}$$

Now we can show that

- $\sigma'(V) = V$, $\sigma' \neq \mathrm{id}$: Follows from $\iota \restriction_V = \sigma$ and $p_v = p$ and thus $\sigma' \restriction_V = \sigma$ (where $f \restriction_X$ denotes the restriction of a function f to the domain X)

- $\sigma' \in \Sigma_{G'}$: With (iv) from Definition 1.3.5 we obtain that, for $u \in S_v$, p_u must be a multiple of p, and $\sigma'(O^\iota_u \cap S_v) = \iota(O^\iota_u \cap S_v)$, thus σ' is a permutation of V' since ι is one. Furthermore, for $t, u \in S_v$, we have $\iota^{p_t(p_u-1)}(t) = t$ and $\iota^{p_u(p_t-1)}(u) = u$ and therefore

$$\begin{aligned}(\sigma'(t), \sigma'(u)) &= (\iota^{p_t-p+1}(t), \iota^{p_u-p+1}(u)) \\ &= (\iota^{p_tp_u-p+1}(t), \iota^{p_tp_u-p+1}(u)),\end{aligned}$$

  thus σ' also inherits edge-preservation from ι.

- $\sigma'(S_u) = S_{\sigma'(u)}$, σ' well-balanced: The above-mentioned fact that for all $u \in S_v$ we have $\sigma'(O^\iota_u \cap S_v) = \iota(O^\iota_u \cap S_v)$, together with (iv) from Definition 1.3.5 implies that also $\sigma'(S_u) = S_{\sigma(u)}$ for all $u \in V$. For all $v' \in V'$, well-balancedness of σ and disjointness of the S_u imply that $\sigma'^q(v') \neq v'$ for $0 < q < p$. On the other hand, since each orbit of σ has size p and contains exactly one element from S_v (namely v), we have that

$$\begin{aligned}\sigma'^p(v') &= \iota^{(p_u-p+1)+(p-1)}(v') && \text{for some } u \in O^\iota_{v'} \\ &= \iota^{p_u}(v') = \iota^{p_{v'}}(v') = v'. && \square\end{aligned}$$

1.4.6. Example. Consider the extended peer-to-peer network G' shown in Figure 1.12(a) with automorphism ι_σ as required by Definition 1.3.5. We illustrate the construction of σ' given in the proof of Lemma 1.4.5.

We have $p = 2$ (the period of $\sigma = \iota_\sigma \restriction_{\{1,2\}}$), and we pick vertex $v = 2$. For the elements of S_2, we obtain $p_2 = p = 2$ and $p_{2a} = p_{2b} = p_{2c} = 6$ since, e.g., $O^{\iota_\sigma}_{2a} = \{2a, 1a, 2c, 1b, 2b, 1c\}$. Thus σ' is defined as follows:

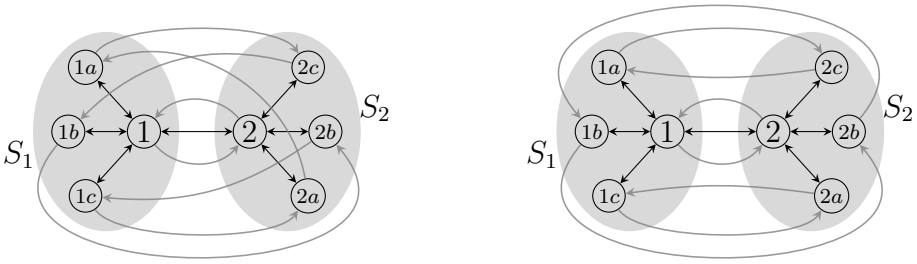
$$\sigma'(u) = \begin{cases} \iota(u) & \text{if } u = 2 \\ \iota^5(u) & \text{if } u \in S_2 \setminus \{2\} \\ \iota(u) & \text{if } u \in S_1. \end{cases}$$

This σ' is depicted in Figure 1.12(b). All orbits have the same cardinality, namely 2, and the remaining claims of Lemma 1.4.5 are also satisfied.

1.4.7. Lemma. *Any symmetry-preserving extension $G' = (V', E')$ of a peer-to-peer network $G = (V, E)$ can be extended to a network H such that*

(i) $\Sigma^{\mathrm{wb}}_H \setminus \{\mathrm{id}\} \neq \emptyset$, *and*

(ii) if G' admits a G-symmetric electoral system in CSP_{in}, then H admits a symmetric electoral system in CSP_{in}.



(a) ι_σ as required by Definition 1.3.5 (b) σ' constructed from ι_σ as in Lemma 1.4.5

Figure 1.12: An extended peer-to-peer network G' illustrating Lemma 1.4.5.

Proof. The idea is to add an "identifying structure" to all elements of V, which forces all automorphisms to keep V invariant and map the S_v to each other correspondingly (see Figure 1.13). Formally, let $K = |V'|$ and, denoting the inserted vertices by $i_{\cdot,\cdot}$, for each $v \in V$ let

$$I_v := \bigcup_{k=1}^{K} \{i_{v,k}\}$$

$$E_v := \{(v, i_{v,1})\} \cup \bigcup_{k=1}^{K-1} \{(i_{v,k}, i_{v,k+1}), (i_{v,k+1}, v)\} \cup \bigcup_{w \in S_v} \{(i_{v,K}, w)\},$$

and let

$$H := (V' \cup \textstyle\bigcup_{v \in V} I_v, E' \cup \bigcup_{v \in V} E_v).$$

Now we can prove the two claims as follows.

(i) Let $\sigma \in \Sigma^{\mathrm{wb}}_{G'} \setminus \{\mathrm{id}\}$ with $\sigma(V) = V$ and $\sigma(S_v) = S_{\sigma(v)}$ for all $v \in V$ (such a σ exists by Lemma 1.4.5). Then we have

$$\sigma \cup \textstyle\bigcup_{v \in V} \bigcup_{k=1}^{K} \{i_{v,k} \mapsto i_{\sigma(v),k}\} \in \Sigma^{\mathrm{wb}}_{H} \setminus \{\mathrm{id}\},$$

and thus $\Sigma^{\mathrm{wb}}_{H} \setminus \{\mathrm{id}\} \neq \emptyset$.

(ii) H is still a symmetry-preserving extension of G via (straightforward) extensions of the S_v. The discriminating construction (notably the fact that the vertices from V now are guaranteed to have more edges than any vertex not in V, but still the same number with respect to each other) has the effect that Σ_H consists only of extensions, as above, of those $\sigma \in \Sigma_{G'}$ for which $\sigma(V) = V$ and $\sigma(S_v) = S_{\sigma(v)}$ for all $v \in V$. Thus, any G-symmetric system with communication graph H is a symmetric system with communication graph H.

Additionally, the set of all $i_{v,k}$ is invariant under Σ_H due to the distinctive structure of the I_v, thus the associated processes are allowed to differ from those of the remaining vertices. A symmetric electoral system in CSP_{in} can thus be obtained by running the original G-symmetric electoral system on all members of G' and having each $v \in V$ inform $i_{v,1}$ about the leader, while all $i_{v,k}$ simply wait for and transmit the leader information. □

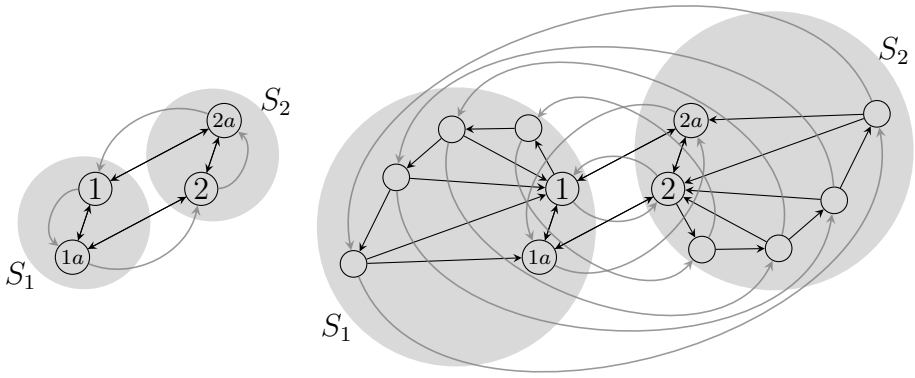


Figure 1.13: The network from Figure 1.9, shown with an automorphism disregarded by G-symmetry, and the extension given in Lemma 1.4.7 invalidating automorphisms of this kind shown with the only remaining automorphism.

Now we have gathered all prerequisites to prove our main result.

1.4.8. Theorem. *There is no symmetry-preserving extension of any peer-to-peer network $G = (V, E)$ that admits a G-symmetric system pairwise synchronizing V in CSP_{in}.*

Proof. Assume there is such a symmetry-preserving extension G'. Then by Lemma 1.4.4 it also admits a G-symmetric electoral system in CSP_{in}. According to Lemma 1.4.7, there is then a network H with $\Sigma_H^{\mathrm{wb}} \setminus \{\mathrm{id}\} \neq \emptyset$ that admits a symmetric electoral system in CSP_{in}. This is a contradiction to Theorem 1.2.12. □

1.5 Conclusions

We have provided a formal definition of peer-to-peer networks and adapted a semantic notion of symmetry for process systems communicating via such networks. In this context, we have defined and investigated the existence of pairwise synchronizing systems, which are directly useful because they achieve synchronization, but particularly in our context because they create common knowledge among processes. Focusing on two dialects of the CSP calculus, we have proved the existence of such systems in $\mathrm{CSP}_{i/o}$, as well as the impossibility

of implementing them in CSP_{in}, even when we allow additional helper processes. We have also mentioned a recent extension to JCSP to show that, while CSP_{in} is less complex and most commonly implemented, implementations of $\text{CSP}_{i/o}$ are feasible and do exist.

A way to circumvent our impossibility result is to remove some requirements. For example, we have provided a construction for non-symmetric systems in CSP_{in}. In general, if we give up the symmetry requirement, $\text{CSP}_{i/o}$ can be implemented in CSP_{in}, as proved by Bougé [29], p. 197.

Another way is to tweak the definition or the assumptions about common knowledge. Various possibilities have been discussed by Halpern and Moses [73]. By following the eager protocol they propose, common knowledge can eventually be attained, but the trade-off is an indefinite time span during which the knowledge states of the processes are inconsistent. This may not be an option, especially in systems which have to be able to act sensibly and rationally at any time. Alternatively, if messages are guaranteed to be delivered exactly after some fixed amount of time, common knowledge can also be achieved, but this may not be realistic in actual systems. Finally, possibilities to approximate common knowledge are described. Approximate common knowledge or finite mutual knowledge may suffice in settings where the impact decreases significantly as the depth of mutual knowledge increases, as discussed, e.g., by Weinstein and Yildiz [149].

However, if one is interested in symmetric systems and exact common knowledge, as is the case in our Chapters 2 and 3, then these results show that $\text{CSP}_{i/o}$ is a suitable formalism while CSP_{in} is insufficient.

Hoare [79] recognized in his initial paper on CSP that the exclusion of output guards reduces expressivity and is programmatically inconvenient. Soon it was deemed technically not justified [22, 33, 64] and removed in later versions of CSP [80, p. 227].

Some existing proposals for implementations of input and output guards and synchronous communication could be criticized for simply shifting the problems to a lower level, notably for not being symmetric themselves or for not even being strictly synchronous in real systems due to temporal imprecision [73]. However, it is often useful to abstract away from implementation issues on the high level of a process calculus or a programming language, as argued in a context similar to ours by Kurki-Suonio [88], Section 10.

We therefore identify JCSP as the most natural platform for the implementation which we discuss in Chapter 3. Since the interaction structures used there generalize our peer-to-peer networks, it is clear that our negative result Theorem 1.4.8 still holds, excluding most other implementations of CSP. Vice versa, JCSP supports broadcasts, i.e., synchronous communication with multiple recipients, and is therefore suitable to implement our positive result Theorem 1.4.1 even in that more general setting.

Chapter 2

Knowledge in interaction structures

2.1 Introduction

2.1.1 Motivation

In this chapter, we introduce a framework for reasoning about communication and knowledge within groups of agents, or players, in settings such as presented in the Introduction chapter.

We assume that each player is a member of a number of groups, and that he can communicate synchronously within each of them. These groups can be thought to have the possibility to meet or communicate privately in some way, for example by sharing a common language. Thus, there is what we call an **interaction structure**, a hypergraph of players, that determines the communication possibilities. We are interested in studying what players can learn, what impact common knowledge of the underlying hypergraph has, and what the properties of the knowledge are that results from this communication.

The framework we set up here forms the epistemic foundation for our study of reasoning about each other's preferences in Chapter 3. In order to be able to cover all the way from theory to implementation, we keep things simple. In particular, we want to obtain a framework where knowledge has a simple structure, is local in some sense and has properties which simplify reasoning about it.

For this reason, we impose clear restrictions while accommodating possibilities for generalizations of the framework. For example, while we here only allow players to communicate the information they initially have, in [9] we have lifted this assumption to allow actual flow of information.

To illustrate our framework, consider the interaction structure in Figure 2.1. It represents a situation where a student s is giving a talk to an audience $\{x, y, S, z\}$ including his supervisor S. Let us assume that the student got something wrong, and the truth, p, is only known to the supervisor. Of course, the supervisor would like to make p known to everybody in the audience. Due to the interaction

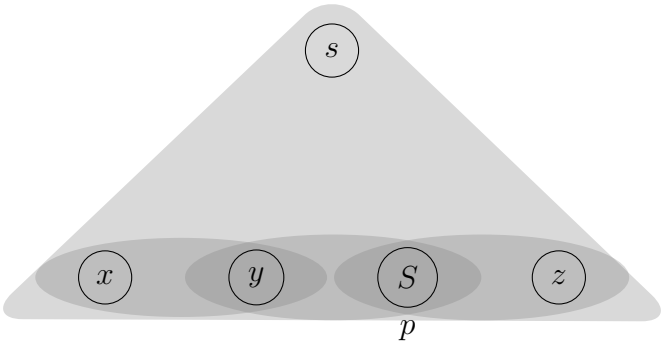


Figure 2.1: Interaction structure during a talk given by s to audience $\{x, y, S, z\}$.

structure, his only options are to either raise his hand and say p out loud, or to whisper p to his neighbors. However, the first option would embarrass the student by making his lapse common knowledge. Since the supervisor is considerate, he is only left with the second option.

In the generalization of our framework that we examine in [9], we consider it possible that y turns to x and tells him what he just learned from S. However, in the basic version of the framework that we examine in this chapter, we assume that facts learned from others are not re-told. Arguably, in real life it happens rarely that a remark made to one's neighbor proceeds to make its way being whispered through a whole chain of persons in the audience.

For more motivation and intuitive justification of the technical assumptions we make here, see Chapter 3, Section 3.1.

2.1.2 Plan of the chapter

In the following Section 2.2, we set up the framework and clarify the restrictions we impose. We examine this framework in detail in Section 2.3, where we also prove various properties of knowledge. In Section 2.4 we discuss related work, in particular two closely related frameworks from the literature, and draw some conclusions. We look at some possible extensions in Section 2.4.2.

2.2 Preliminaries

We assume the following setup to be common knowledge among the players, which is reflected in the formal semantics we introduce later on. This is roughly along the lines of *history-based models* (see, e.g., Pacuit and Parikh [109], Fagin et al. [60]).

We start with a set of players N. Each player $i \in N$ has a private set At_i of **atomic propositions** (*atoms*, or *facts*), of which we assume that only player i

initially knows whether they are true.

The truth values of these facts are represented by a **valuation**, which can be written as a set $V \subseteq \mathrm{At}$ containing those facts that are true, where $\mathrm{At} = \bigcup_{i \in N} \mathrm{At}_i$. By V_i, we denote $V \cap \mathrm{At}_i$, the restriction of V to i's facts. We allow facts to have certain *dependencies*, as is the case in Chapter 3, where the facts represent preference orderings. For example, assume that $p, q, r \in \mathrm{At}_i$, representing whether i prefers chicken over beef, beef over fish, and chicken over fish, respectively. If these interpretations, and i's being consistent, are common knowledge, all players will only consider valuations V such that from $p, q \in V$ it follows that $r \in V$. That is, for example, $\{p, q\}$ will never be considered as possible valuation. To accommodate this, we consider an arbitrary, non-empty class $\mathcal{V}$ of valuations, only subject to two conditions: valuations are *closed under intersection*, that is,

$$\text{for any } V, V' \in \mathcal{V}, \text{ we have } V \cap V' \in \mathcal{V}; \tag{v1}$$

and valuations are *independent across players*, that is,

$$\text{for any } V, V' \in \mathcal{V} \text{ and player } i, \text{ we have } V_i \cup \bigcup_{i \neq j} V'_j \in \mathcal{V}. \tag{v2}$$

An **interaction structure** H is a *hypergraph* on the set N of players, that is, a set of non-empty subsets of N, called *hyperarcs*.

Players can communicate their facts to any of their hyperarcs. We denote such a **message** as tuple (i, A, p) with $i \in A$ and $p \in \mathrm{At}_i$; that is, i communicates his fact p among the group A. A message is *truthful* with respect to a valuation V if, indeed, $p \in V$, and it is *H-compliant* if $A \in H$. If the players consider only H-compliant messages possible, then they know the underlying hypergraph H. So if the model allows only H-compliant messages, the underlying hypergraph H is common knowledge among the players; if it uses all messages, H is unknown.

A **state**, or *possible world*, (V, M) consists of a valuation $V \in \mathcal{V}$ with a set M of truthful messages. An **H-compliant** state is one where M only contains H-compliant messages.

We briefly discuss the assumptions mentioned above and made explicit in this formalization. Firstly, we only allow messages of the form $(i, \cdot, p)$ with $p \in \mathrm{At}_i$. That is, players only communicate basic facts that "belong" to them, and as such are known to them initially. Secondly, this has the consequence that our definition of truthfulness indeed entails that players can only send information they *know* to be true, since $p \in V$ with $p \in \mathrm{At}_i$ implies that i actually knows p. Thirdly, we use unordered *sets* of messages, that is, without any temporal structure, since in our setting it only matters whether a given message has been communicated or not, and not *when* it was communicated.

If one allows sending of information that is not *initially* known, one has to be more careful with the distinction of information that is *known* to be true versus information that is true, but not known to be true. Truthful communication usually allows information to be sent only in the former case. For this reason,

Pacuit and Parikh [109] define truthfulness in mutual recursion with the semantics of knowledge. In [9], we instead use the notion of an *explanation* for a message, which is a sequence of messages with the same content, originating from the player who initially knows the content. This leads to some temporal ordering on the messages, in the sense that if the message (i, A, p) occurs with $p \notin \mathrm{At}_i$, then it becomes common knowledge in A that *before* that message there must have been another message of the form $(\cdot, B, p)$ with $i \in B$, since otherwise i could not have known p.

Returning to the formal setup, by a **word** over a set $A \subseteq N$ we mean a finite sequence $w = i_1 \dots i_k$ where each $i_\ell \in A$. By A^* we denote the set of all words over A, and we write $\mathit{Set}(w)$ for the *set* of players occurring in w.

Now given a set of messages M and a word w, we introduce the following notation:

$$M_w := \{(\cdot, A, \cdot) \in M \mid \mathit{Set}(w) \subseteq A\}$$
$$\mathit{Facts}(M) := \{p \mid (\cdot, \cdot, p) \in M\}.$$

So M_i (respectively, M_w) is the subset of the set of messages M that player i received (respectively, that were communicated jointly to all the players in w; note that the order in w does not matter), and $\mathit{Facts}(M)$ is the set of facts that were communicated in the messages in M. In particular, $\mathit{Facts}(M_i)$ is the set of facts that were communicated in the messages in M that player i received. Note that (V, M) is a state if $\mathit{Facts}(M) \subseteq V$. Further, we define all set operations to act component-wise on states, e.g. $(V, M) \subseteq (V', M')$ iff $V \subseteq V'$ and $M \subseteq M'$.

In order to represent the knowledge of the players we define an **indistinguishability relation** between states:

$$(V, M) \sim_i (V', M') \text{ iff } (V_i, M_i) = (V'_i, M'_i).$$

In the semantics we present below, a player i is said to "know" a fact just if that fact is true in every state that is indistinguishable for i from the actual state. Of particular interest to us is the knowledge of *groups* $G \subseteq N$ (always assumed to be non-empty). Specifically we consider *common knowledge* among a group (cf. [60, p. 23]). These are facts that everybody in the group knows, they all know that they know, etc. To define this formally, we extend the individual indistinguishability relation to groups: for $G \subseteq N$ the relation $\sim_G$ is the transitive closure of $\bigcup_{i \in G} \sim_i$.

We are interested in properties definable by the following **epistemic language** $\mathcal{L}$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid C_G\varphi,$$

where the atoms p denote the facts in At, $\neg$, $\wedge$ and $\vee$ are the standard connectives; and C_G is a knowledge operator, with $C_G\varphi$ meaning φ is common knowledge among G. We write K_i for $C_{\{i\}}$; $K_i\varphi$ can be read "i knows that φ". The **positive**

language $\mathcal{L}^+$ is the sublanguage of $\mathcal{L}$ in which negation ($\neg$) does not occur, and a formula is **propositional** if it does not contain any knowledge operators.

The semantics for $\mathcal{L}$ is as follows:

$$\begin{array}{ll}
(V,M) \models_H p & \text{iff } p \in V, \\
(V,M) \models_H \neg\varphi & \text{iff } (V,M) \not\models_H \varphi, \\
(V,M) \models_H \varphi \vee \psi & \text{iff } (V,M) \models_H \varphi \text{ or } (V,M) \models_H \psi, \\
(V,M) \models_H \varphi \wedge \psi & \text{iff } (V,M) \models_H \varphi \text{ and } (V,M) \models_H \psi, \\
(V,M) \models_H C_G\varphi & \text{iff } (V',M') \models_H \varphi \text{ for each } H\text{-compliant } (V',M') \\
 & \qquad\qquad \text{with } (V,M) \sim_G (V',M').
\end{array}$$

We also use a special case of propositional entailment, namely, for any $P \subseteq \mathrm{At}$ and $p \in \mathrm{At}$, we write

$$P \models p \qquad \text{iff } p \in V \text{ for all } V \in \mathcal{V} \text{ with } P \subseteq V.$$

By $\models_H \varphi$ we mean that φ is a **tautology**, that is $(V,M) \models_H \varphi$ for *all* H-compliant states (V,M). Note that for propositional φ this is equivalent to saying that all valuations satisfy φ. Further note that, due to the way in which we defined valuations, more formulas can be tautological than one might think, including facts and positive propositional formulas. For example, if we consider the class of valuations $\mathcal{V} = \{\emptyset, \{q\}, \{p,q\}\}$, then $\neg p \vee q$ is a tautology. For the class $\mathcal{V} = \{\{p\}, \{p,q\}\}$, p and thus also $p \vee q$ are tautologies.

By allowing only H-compliant states in the last clause of the semantics, the underlying hypergraph H is assumed to be *common knowledge*. Assuming that the hypergraph H is *unknown* turns out to be equivalent to the case where it is common knowledge that the hypergraph H is complete, i.e., $H = 2^N \setminus \{\emptyset\}$. This might seem counter-intuitive, but it reflects the fact that if the hypergraph is unknown then every player must consider it possible that every set $A \subseteq N$ might be a hyperarc in H. To denote the corresponding semantics, we use $\models$ as abbreviation for $\models_H$ with H being the complete hypergraph.

For a word $w = i_1 \dots i_k$, we write K_w to abbreviate $K_{i_1} K_{i_2} \cdots K_{i_k}$, and we write $\sim_w$ to denote the concatenation $\sim_{i_1} \circ \cdots \circ \sim_{i_k}$.

Notice that $(V,M) \sim_G (V',M')$ iff there is $w \in G^*$ with $(V,M) \sim_w (V',M')$. So an equivalent way of specifying the semantics for C_G with non-singleton G is as follows:

$$(V,M) \models C_G\varphi \text{ iff } (V,M) \models K_w\varphi \text{ for all } w \in G^*. \qquad (\star)$$

2.3 Properties of knowledge

We now study properties of knowledge that can be attained in our framework.

Where does knowledge come from? We are interested in knowledge that is obtained through communication. Due to our setup, certain non-trivial (non-tautological) formulas are known even without any communication.

2.3.1. Example. With players $N = \{i, j, k\}$, $H = \{N\}$, $p \in \mathrm{At}_i$, $V = \{p\}$, and $\mathcal{V} = \{\emptyset, V\}$, we have

$$(V, \emptyset) \vDash_H K_j \neg K_k p.$$

Intuitively, the only hyperarc in H through which player k could learn anything from i is the one which also contains player j. So there is no way for i to tell k anything "secretly". Hence, with $M_j = \emptyset$, j also knows that $M_{ik} = \emptyset$. That is, in all states which j considers possible at (V, M), i has not told k that p, therefore in all these states k does not know p (since p is not a tautology).

Indeed, this is commonly known among N, so that not only iterated, but common knowledge exists without communication:

$$(V, \emptyset) \vDash_H C_N \neg C_N p.$$

Lastly, knowledge about thirds can be obtained from communication not concerning them:

$$(V, \{(i, \{i, j\}, p)) \vDash_H K_j \neg K_k \neg p.$$

It can be noted that all of these examples include formulas using negation. Since we want to examine knowledge obtained through communication, and since we only need positive formulas in Chapter 3, we focus on these in the following. Indeed, we establish in Theorem 2.3.10 that knowledge of non-tautological positive formulas can only result from communication.

Common knowledge of H. We now examine the relevance of common knowledge of the interaction structure H. Consider the following example.

2.3.2. Example. In Example 2.3.1, we had

$$(V, \emptyset) \vDash_H K_j \neg K_k p.$$

On the other hand, we have

$$(V, \emptyset) \nvDash K_j \neg K_k p,$$

since $(V, \emptyset) \sim_j (V, \{(i, \{i, k\}, p)\})$.

So there are formulas for which knowledge of H matters. However, for positive formulas it does not. In order to establish this, we first show the following Lemma 2.3.3, which intuitively says that if a positive formula is true in some state, then it remains true in any state where more facts are true or more communication has taken place. Remember that $\vDash$ corresponds to $\vDash_H$ with H being the complete hypergraph, so the following carries over to general (not H-compliant) states together with $\vDash$.

2.3.3. Lemma. *For any $\varphi \in \mathcal{L}^+$ and H-compliant states (V, M) and (V', M') with $(V, M) \subseteq (V', M')$,*

$$\text{if } (V, M) \models_H \varphi, \text{ then } (V', M') \models_H \varphi.$$

Proof. We proceed by structural induction on φ. The only not completely obvious case is when $\varphi = C_G\psi$ with $\psi \in \mathcal{L}^+$. We show the claim for $G = \{i\}$; the non-singleton case then follows by induction and $(\star)$.

Take an H-compliant state (V'', M'') such that $(V', M') \sim_i (V'', M'')$. Let

$$(V''', M''') := (V_i \cup \textstyle\bigcup_{j \neq i} V''_j, M_i),$$

and note that V''' is a valuation due to (v2). We have $\mathit{Facts}(M_i) \subseteq \mathit{Facts}(M) \subseteq V$, since (V, M) is a state. Also, $M_i \subseteq M'_i = M''_i \subseteq M''$, so $\mathit{Facts}(M_i) \subseteq \mathit{Facts}(M'') \subseteq V''$, since (V'', M'') is a state. Hence,

$$\mathit{Facts}(M''') = \mathit{Facts}(M_i) \subseteq V \cap V'' = \textstyle\bigcup_{i \in N}(V_i \cap V''_i) \subseteq V'''.$$

This shows that the messages in M''' are truthful with respect to V''', so (V''', M''') is a state. Moreover, $(V, M) \sim_i (V''', M''')$. Assume now $(V, M) \models_H K_i\psi$. Then we obtain $(V''', M''') \models_H \psi$. Further, we have $(V''', M''') \subseteq (V'', M'')$ since $V_i \subseteq V'_i = V''_i$ and $M_i \subseteq M'_i = M''_i$ due to $(V', M') \sim_i (V'', M'')$. Thus, by induction hypothesis we obtain $(V'', M'') \models_H \psi$. □

Now we can prove that common knowledge of H does not matter for positive formulas.

2.3.4. Theorem. *For any H-compliant state (V, M) and $\varphi \in \mathcal{L}^+$,*

$$(V, M) \models \varphi \textit{ iff } (V, M) \models_H \varphi.$$

Proof. We proceed by structural induction. The only non-trivial step is when $\varphi = C_G\psi$ with $\psi \in \mathcal{L}^+$.
($\Rightarrow$) By induction hypothesis, $(V, M) \models C_G\psi$ implies $(V, M) \models_H C_G\psi$, since each H-compliant state is a state.
($\Leftarrow$) Assume to the contrary that $(V, M) \not\models C_G\psi$. So there is a state (V', M') with $(V, M) \sim_G (V', M')$ and $(V', M') \not\models \psi$. Now let

$$M' \restriction_H := \{(\cdot, A, \cdot) \in M' \mid A \in H\}.$$

So $M' \restriction_H$ consists of all H-compliant messages in M'. Now note that $(V', M' \restriction_H) \subseteq (V', M')$, so from $(V', M') \not\models \psi$ we obtain that $(V', M' \restriction_H) \not\models \psi$ using Lemma 2.3.3 (which, as noted, also holds for general states and $\models$). Since $(V', M' \restriction_H)$ is H-compliant, the induction hypothesis yields $(V', M' \restriction_H) \not\models_H \psi$. Moreover, we also have $(V, M) \sim_G (V', M' \restriction_H)$, since (V, M) is H-compliant and $(V, M) \sim_G (V', M')$. Thus, $(V, M) \not\models_H C_G\psi$. □

In the remainder of this chapter, we are concerned with positive formulas, so in view of this result, we restrict attention to $\models$.

Distributivity of knowledge over disjunction. We now establish that C_G distributes over disjunctions of positive formulas, starting with singleton G (that is, K_i). Note that these results also hold if the disjunction is a tautology.

2.3.5. Lemma. *For any $\varphi_1, \varphi_2 \in \mathcal{L}^+$, $i \in N$, and state (V, M),*

$$(V, M) \vDash K_i(\varphi_1 \vee \varphi_2) \text{ iff } (V, M) \vDash K_i\varphi_1 \vee K_i\varphi_2.$$

Proof. To deal with the ($\Rightarrow$) implication assume that $(V, M) \nvDash K_i\varphi_1 \vee K_i\varphi_2$. Then $(V, M) \nvDash K_i\varphi_1$ and $(V, M) \nvDash K_i\varphi_2$, i.e., there are (V', M') and (V'', M'') such that

$$\begin{aligned}&(V, M) \sim_i (V', M') \text{ and } (V', M') \nvDash \varphi_1, \text{ as well as}\\ &(V, M) \sim_i (V'', M'') \text{ and } (V'', M'') \nvDash \varphi_2.\end{aligned}$$

Let now

$$(V''', M''') := (V_i \cup \textstyle\bigcup_{j \neq i}(V'_j \cap V''_j), M_i),$$

and note that V''' is a valuation due to (v1) and (v2). Then

$$\mathit{Facts}(M) \subseteq V, \quad \mathit{Facts}(M') \subseteq V', \quad \text{and} \quad \mathit{Facts}(M'') \subseteq V'',$$

since (V, M), (V', M') and (V'', M'') are states. Moreover, $M_i = M'_i$ and $M_i = M''_i$. So,

$$\begin{aligned}\mathit{Facts}(M''') &\subseteq \mathit{Facts}(M) \cap \mathit{Facts}(M') \cap \mathit{Facts}(M'')\\ &\subseteq V \cap V' \cap V''\\ &= \textstyle\bigcup_{j \in N}(V_j \cap V'_j \cap V''_j)\\ &\subseteq V'''.\end{aligned}$$

This shows that the messages in M''' are truthful with respect to V''', and since $M''' = M_i \subseteq M$, (V''', M''') is an H-compliant state.

Now since $V_i = V'_i = V''_i$ and $M_i = M'_i = M''_i$, we have $(V''', M''') \subseteq (V', M')$ and $(V''', M''') \subseteq (V'', M'')$. By Lemma 2.3.3, we obtain $(V''', M''') \nvDash \varphi_1$ and $(V''', M''') \nvDash \varphi_2$, thus $(V''', M''') \nvDash \varphi_1 \vee \varphi_2$. Furthermore $(V, M) \sim_i (V''', M''')$, so $(V, M) \nvDash K_i(\varphi_1 \vee \varphi_2)$.

Further, the ($\Leftarrow$) implication immediately holds by the semantics. □

2.3.6. Theorem. *For any $\varphi_1, \varphi_2 \in \mathcal{L}^+$, state (V, M), and $G \subseteq N$,*

$$(V, M) \vDash C_G(\varphi_1 \vee \varphi_2) \text{ iff } (V, M) \vDash C_G\varphi_1 \vee C_G\varphi_2.$$

Proof. The claim follows directly from Lemma 2.3.5 and ($\star$). □

To easily see that this result does not hold if we allow negation, consider $\varphi = K_i(p \vee \neg p)$ in the context of two players $i, j \in N$, $p \in \mathrm{At}_j$, $\mathcal{V} = \{\emptyset, \{p\}\}$, and $(V, M) = (\emptyset, \emptyset)$. We have $(V, M) \vDash \varphi$, since the disjunction is a tautology, but there is no way for i to know which disjunct is true.

Even with non-tautological disjunctions, the result does not hold, as we see in the following example.

2.3.7. Example. Consider $N = \{i, j, k\}$, $p \in \mathrm{At}_k$, and $\mathcal{V} = \{\emptyset, \{p\}\}$. With

$$\begin{aligned} V &= \{p\}, \\ M &= \{(k, \{i, k\}, p)\}, \text{ and} \\ \varphi &= K_i(K_j p \vee \neg(K_j p \vee K_j \neg p)), \end{aligned}$$

we have $(V, M) \vDash \varphi$, but again, i knows neither disjunct in (V, M). Intuitively, having privately learned that p is true, i knows that j either also learned it, or that j doesn't know whether p is true, but i does not know which of these two statements holds.

Knowledge obtained through communication. Another observation is that mutual knowledge of any fact $p \in \mathrm{At}$ can only be obtained through corresponding messages, and is thus inseparably tied to common knowledge. Note that the link between mutual and common knowledge holds even in the case of tautological facts, although then no communication is required. Remember that for $P \subseteq \mathrm{At}$ and $p \in \mathrm{At}$, by $P \vDash p$ we denote that $p \in V$ for all valuations V with $P \subseteq V$.

2.3.8. Lemma. *For any $w \in N^*$ with $|Set(w)| \geq 2$, $p \in \mathrm{At}$, and state (V, M), the following are equivalent:*

(i) $(V, M) \vDash K_w p$,

(ii) $Facts(M_w) \vDash p$,

(iii) $(V, M) \vDash C_G p$ *with* $G = Set(w)$.

Proof.
(*i*) $\Rightarrow$ (*ii*): Assume that (*ii*) does not hold. That is, there is a valuation V' such that $Facts(M_w) \subseteq V'$ and $p \notin V'$. Thus, (V', M_w) is a state and $(V', M_w) \nvDash p$.

Let $i \in N$ be such that $p \in \mathrm{At}_i$. Then $V'' := V'_i \cup \bigcup_{j \neq i} V_j$ is a valuation due to (v2). We thus get another state (V'', M_w) with $(V'', M_w) \nvDash p$.

Now fix $j \in Set(w)$ with $p \notin \mathrm{At}_j$, noting that such a j exists since $|Set(w)| \geq 2$. We then have $(V, M) \sim_j (V'', M_w)$. Since $j \in Set(w)$ and both $(V, M) \sim_k (V, M)$ and $(V'', M_w) \sim_k (V'', M_w)$ for all $k \in N$, we obtain $(V, M) \sim_w (V'', M_w)$, and thus $(V, M) \nvDash K_w p$.
(*ii*) $\Rightarrow$ (*iii*): Suppose that $G = Set(w)$ and take $m \in M_w$. Consider (V', M') such that $(V, M) \sim_G (V', M')$. This means that for a sequence $i_1, \ldots, i_k$ of players from G and some states $(V^1, M^1), \ldots, (V^k, M^k)$ we have

$$(V, M) \sim_{i_1} (V^1, M^1) \sim_{i_2} \cdots \sim_{i_k} (V^k, M^k),$$

where $(V', M') = (V^k, M^k)$. But $i_1 \in G$, so $m \in M_{i_1}$, and consequently $m \in M^1_{i_1}$. Also $i_2 \in G$, so $m \in M^1_{i_2}$, and consequently $m \in M^2_{i_2}$. Continuing this way we conclude that $m \in M^k_{i_k}$, that is, $m \in M'_{i_k}$.

This shows that $M_w \subseteq M'$. So, $Facts(M_w) \subseteq V'$, since (V', M') is a state. But by assumption, $Facts(M_w) \vDash p$, so $(V', M') \vDash p$. This proves $(V, M) \vDash C_G p$.
$(iii) \Rightarrow (i)$: By $(\star)$. □

We can extend this connection between mutual and common knowledge to arbitrary positive formulas.

2.3.9. Theorem. *For any $G \subseteq N$, $\varphi \in \mathcal{L}^+$, and state (V, M),*

$$(V, M) \vDash C_G\varphi \text{ iff } (V, M) \vDash K_w\varphi \text{ for some } w \in G^* \text{ with } Set(w) = G.$$

Proof. The direction $(\Rightarrow)$ is by $(\star)$.

For $(\Leftarrow)$, we proceed by structural induction. The base case is obtained from Lemma 2.3.8. The induction step for disjunction follows by Theorem 2.3.6, and for conjunction it follows directly by definition of the semantics. For $\varphi = K_i\psi$, the assumption $(V, M) \vDash K_w K_i\psi$ yields, by induction hypothesis, that $(V, M) \vDash C_{G'}\psi$ for $G' = Set(w) \cup \{i\} = G \cup \{i\}$, which by definition of the semantics implies that $(V, M) \vDash C_G K_i\psi$. □

Note that this result provides, for positive formulas, a simplified characterization of the common knowledge operator, as compared with $(\star)$.

Finally, we establish that common knowledge can only be attained by a group through some message (or messages) to the *whole* group. That is, unless some formula φ is inherently common knowledge due to our setup (i.e., is tautological), common knowledge of φ cannot be achieved by means of more limited communications, for example point-to-point messages.

2.3.10. Theorem. *For any $G \subseteq N$ with $|G| \geq 2$, non-tautological $\varphi \in \mathcal{L}^+$, and state (V, M),*

$$\text{if } (V, M) \vDash C_G\varphi, \text{ then there is } (\cdot, A, \cdot) \in M \text{ with } G \subseteq A.$$

Proof. By Theorem 2.3.6 and the semantics, we can transform $C_G\varphi$ into an equivalent formula consisting only of disjunctions and conjunctions over formulas of the form $C_G C_{G_1} \cdots C_{G_\ell} p$ with $G_1, \ldots, G_\ell \subseteq N$. Since $(V, M) \vDash C_G\varphi$ and φ is non-tautological, there is at least one of these formulas such that $(V, M) \vDash C_G C_{G_1} \cdots C_{G_\ell} p$ and p is non-tautological.

Fix now one such formula, and take w such that $Set(w) = G$. By $(\star)$ we obtain $(V, M) \vDash K_w C_{G_1} \ldots C_{G_\ell} p$, so by the semantics we have $(V, M) \vDash K_w p$. By Lemma 2.3.8 this implies $Facts(M_w) \vDash p$. Since p is non-tautological, this implies that M_w is non-empty, from which the claim follows. □

2.4 Conclusions

2.4.1 Related work

In this chapter we studied various aspects of common knowledge in a simple framework with synchronous communication. It is useful to clarify how our result relate to the literature.

The problem of *coordinated attack* described in the Introduction chapter demonstrates the close ties between simultaneous events and common knowledge. We have seen that in our framework, a group's attaining common knowledge of formulas which are not initially common knowledge is indeed inseparably tied to synchronous communication among that whole group.

Chandy and Misra [39] consider the flow of information in distributed systems with asynchronous communication. They study how processes "learn" about states of other processes and establish lower bounds on the number of messages required to solve certain knowledge-related problems. In some sense this resembles our work, even though we are looking for simplifications rather than lower bounds. The main difference, though, is that hypergraphs, which are important in our context, are equivalent to mere point-to-point graphs in the context of asynchronous communication: Without guarantees on the delivery time, and without temporal reasoning, the information content of receiving an asynchronous group message is the same as that of receiving just a separate private message. Sending a group message can thus be simulated by sending a separate message to each group member.

Our study concerning the consequences of the assumption whether the underlying hypergraph is commonly known among the players brings our framework somewhat closer to the area of social networks (see, e.g., Jackson [83]). Within logic, the relevance of epistemic issues in communication networks has been recognized by a number of authors, e.g. van Benthem [20]. However, to our knowledge the only work that addresses these issues is Pacuit and Parikh [109] and, to some extent, Roelofsen [126]. We now briefly discuss these frameworks and relate them to our own.

Pacuit and Parikh [109] use a history-based model to study diffusion of information in a communication graph, starting from facts initially known to individual agents. Communicative acts are assumed to consist in an agent j "reading" an arbitrary propositional formula from another agent i, with the precondition that i *knows* that the formula holds. Communicative acts are restricted to a commonly known, static, directed graph and, unlike in our case, are assumed to go *unnoticed* by i. Using our notation for messages, a communicative act in that framework can thus be denoted as (i, j, φ), and is only allowed to occur if there is an edge from i to j in the communication graph, and only in states where $K_i\varphi$ holds (that is, communication is truthful). The paper formalizes what conclusions, beyond the mere factual content of messages, can be drawn using knowledge of

the communication graph and, consequently, knowledge of the possible routes along which certain information can have flown. In this sense, it is similar to our extension [9] of the framework presented here.

Roelofsen [126] uses a model based on Dynamic Epistemic Logic (DEL) to describe how some initial epistemic model evolves in a communication situation. Communication is among subgroups and can contain arbitrary epistemic formulas. That is, a message can be denoted as (A, B, φ), where A and B are groups of players and φ is an epistemic formula, with the intuitive reading that the players in B commonly learn that the players in A commonly know φ. Here, too, truthful communication is assumed, so $C_A\varphi$ is a precondition for such a message. Further, communication is assumed to be truthful and is restricted to occur along a hypergraph. However, the hypergraph is explicitly encoded in the model, and thus (knowledge of it) is subject to change. In the spirit of DEL, this approach lends itself best to an explicit modeling of all events, including the formation of suspicions about undetected communications.

While under certain circumstances history-based modeling and DEL are equivalent [21], our approach is more in the spirit of Pacuit and Parikh [109]. All possible communications are modelled from the outset and suspicions about them are not explicitly formed, and the underlying graph (in our case hypergraph) is static and not included in the model.

On a conceptual level, our approach differs in its focus on identifying natural conditions that allow us to prove stronger results about knowledge, such as distributivity over disjunctions, or irrelevance of (common) knowledge of the underlying hypergraph. We use these results in Chapter 3.

2.4.2 Possible extensions

We conclude by listing a number of natural extensions of the considered framework that are worth to be studied further:

- By considering classes of valuations that only need to satisfy (v1) and (v2), we allow certain dependencies among atoms. Alternatively, we could equip the agents with logic theories that determine the dependencies among their atoms. These theories could be common knowledge, or only partially known to other agents.

- We could consider more complex messages than simple atomic facts, for example propositional formulas, or even epistemic formulas. (Note, however, that our messages can be more expressive than obvious at first glance. For example, with $\mathcal{V} = \{\emptyset, \{p\}, \{q\}\}$ a message containing p also entails $\neg q$. In a limited way, messages can thus already contain non-atomic information.) Also, we could study asynchronous communication, or a counterpart of the blind copy feature familiar from e-mails.

- We could assume that the players have different knowledge of the underlying hypergraph, by assuming that for all i we have $H \subseteq H_i$, where H is the underlying hypergraph and H_i is its approximation known to i, and that players learn H by exchanging messages. The messages would contain information about which hyperarcs do *not* belong to H.

- Alternatively, we could study a setup in which each player has an indistinguishability relation over hypergraphs. This would allow us to model players' partial knowledge of the underlying hypergraph.

- It would also be interesting to study the formation of interaction structures, given certain epistemic goals or other strategic considerations. For example, in the scenario discussed in Section 2.1.1, the supervisor might want to change the interaction structure and create a hyperarc containing only him and the rest of the audience, by asking the student to go and fetch his grading records at the end of the talk.

Chapter 3

Strategies in interaction structures

3.1 Introduction

3.1.1 Motivation

Our aim in this chapter is to model situations similar to the example from the Introduction chapter, and examine what players may be said to know in any particular state of communication, how they can compute this knowledge, and how they can use it to eliminate strategies.

There is a substantial amount of research within game theory on the implications of assumptions concerning players' *knowledge* and *beliefs* [16]. In particular, Tan and Werlang [144] have shown that if payoffs are commonly known and all players are *rational* in a formal sense and commonly believe in each other's rationality, they will only play strategies that survive iterated elimination of strictly dominated strategies (IESDS, as explained in the Introduction chapter). Note that we do not delve into the details of possible definitions of rationality here; in our context the relevant implication is that rational players do not choose strictly dominated strategies.

Another line of research stresses the relevance of *locality* in strategic games. For example, in *graphical games* [85] the locality assumption manifests itself in payoff functions which depend only on the strategies of players' neighbors in a graph structure over the set of players.

The framework we consider in this chapter applies a locality assumption to the *information* about payoffs, rather than to the payoffs themselves. Concretely, we use the framework of interaction structures from Chapter 2 and add the notion of a strategic game to it. We assume that the players' initial information only covers their own preferences, and that they can communicate their preferences according to the interaction structure. We study the outcomes of strategy elimination that can be obtained in any given state of communication, including the situation when all communication permitted by the interaction structure has taken place. Insights

from Chapter 2 are used in order to prove that the outcomes we establish indeed correctly reflect what the players know in any particular situation. Building upon Chapter 1, we then describe two ways to implement IESDS locally in each player process in a distributed setting.

It is important to note that we do *not* examine strategic aspects of communication here. Firstly, that means that we do not allow players to lie; and secondly, that we do not examine *why* players do, or *what* they should, communicate. Rather, we examine what happens *if* they do.

To justify this focus, we can think of settings where the strategic aspects of communication itself are not relevant. One possibility is if communication is not a deliberate act, for example, if it occurs through observing behavior. If Ann keeps going to Indian restaurants, she involuntarily communicates her food preference to anyone observing her. Such communication is certainly more difficult to control, and more laborious to fake, than mere words. In a sense it is inherently credible, and research in social learning argues along similar lines [38, Ch. 3].

This also helps to explain another assumption we make, corresponding to the framework from Chapter 2: Players only communicate their *own* preferences, since communication about *others'* preferences is either not inherently credible (if done with words) or difficult to accomplish (if conveyed through behavior). One may also assume that communicating about third parties is less common for *privacy* reasons.

Overall, the hyperarcs of the interaction structure can then be viewed as corresponding to groups who have occasions where they commonly observe each other, for example colleagues sharing lunch time.

In more proactive settings, in particular in the case of communicating processes, it may be more difficult to view communication as something not deliberate. Here, ignoring strategic aspects of communication can be interpreted as bounds on the players' rationality or reasoning capabilities—they simply lack the capabilities to deal with all implications and eventualities of an inherently rich phenomenon such as communication. This holds in particular with such simple implementations as what we obtain in Section 3.5.

Generally, strategic communication is a research topic on its own, with controversial discussions (see, e.g., [132]) and many questions wide open. Crawford and Sobel [44] have considered the topic in a probabilistic setting, and Farrell and Rabin [62] have looked at related issues under the notion of *cheap talk*. Also within epistemic logic, formalizations of the information content of strategic communication have been suggested, e.g., by Gerbrandy [65]. But the topic is unclear, and we choose not to focus on it here.

To sum up, we make the following assumptions:

- the players are rational;
- they initially know their own preferences;

- they are part of an interaction structure and can communicate their own preferences within any hyperarc they belong to;
- communication is truthful and synchronous, as in Chapter 2;
- the players have no knowledge other than what follows from these assumptions, and these are common knowledge.

These assumptions are reflected in the formalizations we make later on.

It is useful to clarify the relation between strategic games with interaction structures and *pre-Bayesian games*, introduced by Ashlagi et al. [10]. In these games, too, each player knows his payoff but does not know the payoffs of the other players. In our setup this private knowledge aspect of pre-Bayesian games can be trivially modelled by the empty interaction structure, or viewed as corresponding to our initial situation. Due to the different nature of these frameworks, however, the questions of interest are also different.

3.1.2 Plan of the chapter

This chapter is organized as follows. In the following Section 3.2, we review the basic definitions concerning strategic games, optimality notions and operators on restrictions of games. Next, in Section 3.3, we study the outcome of IESDS in the presence of an interaction structure. We first look at the outcome that is arrived at after all communication permitted in the given interaction structure has taken place, and then detail the outcomes obtained in any particular intermediate state of communication. The formulations we consider make no direct use of the notion of knowledge.

The connection with knowledge is made in Section 3.4, where we prove the outcomes we have obtained to be correct with respect to the epistemic framework from Chapter 2, in the sense that the outcomes capture exactly what the players can do given their partial knowledge of the game structure in any particular state. In Section 3.5, we describe two ways of implementing our procedures in a distributed setting. Finally, in Section 3.6, we suggest some future research directions.

3.2 Preliminaries

By a **strategic game** (in short, a **game**) for players $N = \{1, \ldots, n\}$, where $n > 1$, we mean a tuple

$$(S_1, \ldots, S_n, \succ_1, \ldots, \succ_n),$$

where for each $i \in N$,

- S_i is the non-empty, finite set of **strategies** available to player i. We write S to abbreviate the set of **strategy profiles**: $S = S_1 \times \cdots \times S_n$.

- $\succ_i$ is the **preference relation** for player i, so $\succ_i \subseteq S \times S$. We are interested in strict dominance, and therefore we consider strict orders as preference relations (possibly induced by an underlying payoff structure, as expained in the Introduction chapter).

As is customary in game theory, we denote the strategies of player i by s_i, possibly with some superscripts. We also denote i's strategy in a strategy profile $s \in S$ by s_i, and the tuple consisting of all other elements by s_{-i}, i.e.,

$$s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n).$$

Similarly, we use S_{-i} to denote $S_1 \times \cdots \times S_{i-1} \times S_{i+1} \times \cdots \times S_n$, and for $s'_i \in S_i$ and $s_{-i} \in S_{-i}$ we write (s'_i, s_{-i}) to denote $(s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n)$. Finally, we use $s'_i \succ_{s_{-i}} s_i$ as a notational alternative for $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$. That is, $s'_i \succ_{s_{-i}} s_i$ means that, if the other players choose the strategies given by s_{-i}, then player i strictly prefers to choose s'_i over s_i.

Fix now an *initial* strategic game $\mathcal{G} := (S_1, \ldots, S_n, \succ_1, \ldots, \succ_n)$. We say that $(S'_1, \ldots, S'_n)$ is a **restriction** of $\mathcal{G}$ if each S'_i is a non-empty subset of S_i. We identify the restriction $(S_1, \ldots, S_n)$ with $\mathcal{G}$.

To analyze iterated elimination of strategies from the initial game $\mathcal{G}$, we view such procedures as operators on the set of restrictions of $\mathcal{G}$. This set together with component-wise set inclusion forms a complete lattice.

For any restriction $\mathcal{G}' := (S'_1, \ldots, S'_n)$ of $\mathcal{G}$ and strategies $s_i, s'_i \in S_i$, we say that s_i is **strictly dominated by** s'_i **on** S'_{-i} if $s'_i \succ_{s'_{-i}} s_i$ for all $s'_{-i} \in S'_{-i}$.[1] Then we introduce the following abbreviations (ℓ stands for "local" and g stands for "global"; the terminology is from Apt [5]):

- $sd^{\ell}(s_i, \mathcal{G}')$ which holds iff strategy s_i of player i is not strictly dominated on S'_{-i} by any strategy from S'_i (i.e., $\neg\exists s'_i \in S'_i \, \forall s'_{-i} \in S'_{-i} \; s'_i \succ_{s'_{-i}} s_i$),
- $sd^{g}(s_i, \mathcal{G}')$ which holds iff strategy s_i of player i is not strictly dominated on S'_{-i} by any strategy from S_i (i.e., $\neg\exists s'_i \in S_i \, \forall s'_{-i} \in S'_{-i} \; s'_i \succ_{s'_{-i}} s_i$).

So in sd^g, the global version of strict dominance introduced by Chen et al. [40], it is stipulated that a strategy not be strictly dominated by a strategy *from the initial game*.

We call each relation of the form sd^{ℓ} or sd^{g} an **optimality notion**. We say then that the optimality notion ϕ used by player i is **monotonic** if for all restrictions $\mathcal{G}''$ and $\mathcal{G}'$ and strategies s_i,

$$\mathcal{G}'' \subseteq \mathcal{G}' \text{ and } \phi(s_i, \mathcal{G}'') \text{ implies } \phi(s_i, \mathcal{G}').$$

[1] We are interested in deterministic behavior, guided by deterministic knowledge, rather than the study of equilibria or randomizing players. For this reason, we do not consider *mixed strategies* here.

As noted in [32, 5], sd^g is monotonic, while sd^ℓ is not (though their respective outcomes are equivalent in finite games, as discussed in the proof of Theorem 3.3.2).

Given an operator T on a finite lattice $(D, \subseteq)$ and $k \geq 0$, by T^k we denote the k-fold iteration of T, where $T^0 = D$ (so the iterations start "at the top") and put $T^\infty := \bigcap_{k\geq 0} T^k$. We call T **monotonic** if for all $D', D'' \subseteq D$,

$$D' \subseteq D'' \text{ implies } T(D') \subseteq T(D'').$$

Finally, an **interaction structure** H, as in Chapter 2, is a *hypergraph* on N, i.e., a set of non-empty subsets of $A \subseteq N$, called *hyperarcs*.

3.3 Iterated strategy elimination

In this section we define procedures for iterated elimination of strictly dominated strategies.

Let us fix a strategic game $\mathcal{G} = (S_1, \ldots, S_n, \succ_1, \ldots, \succ_n)$ for players N, an interaction structure $H \subseteq 2^N \setminus \{\emptyset\}$, and an optimality notion ϕ. In Section 3.3.1, we look at the outcome that the players can obtain after all communication permitted by H has taken place, that is, when within each hyperarc of H all of its members' preferences have been communicated. In Section 3.3.2, we then look at the outcomes obtained in any particular intermediate state of communication.

The formulations we give here make no direct use of a formal notion of knowledge. The connection with a formal epistemic model is made in Section 3.4.

All the iterations of the considered operators start at $(S_1, \ldots, S_n)$.

3.3.1 Completed communication

Let us assume that within each hyperarc $A \in H$, all its members have shared all information about their preferences. We leave the exact definition of communication to Section 3.3.2 and the epistemic formalization to Section 3.4, and focus on an operational description for now.

For each group of players $G \in N$, let S_G denote the set of those restrictions of $\mathcal{G}$ which only restrict the strategy sets of players from G. That is,

$$S_G := \{(S'_1, \ldots, S'_n) \mid S'_i \subseteq S_i \text{ for } i \in G \text{ and } S'_i = S_i \text{ for } i \notin G\}.$$

Now we introduce an elimination operator T_G on each such set S_G, defined as follows. For each $\mathcal{G}' = (S'_1, \ldots, S'_n) \in S_G$,

$$T_G(\mathcal{G}') := (S''_1, \ldots, S''_n),$$

where for all $i \in N$,

$$S''_i := \begin{cases} \{s_i \in S'_i \mid \phi(s_i, \mathcal{G}')\} & \text{if } i \in G \\ S'_i & \text{otherwise.} \end{cases}$$

We call T_G^∞ the **outcome of iterated elimination (of non-ϕ-optimal strategies) on** G. We then define the restriction $\mathcal{G}(H)$ of $\mathcal{G}$ as

$$\mathcal{G}(H) := (\mathcal{G}(H)_1, \ldots, \mathcal{G}(H)_n),$$

where for all $i \in N$,

$$\mathcal{G}(H)_i := T_{\{i\}}\big(\textstyle\bigcap_{A:i\in A\in H} T_A^\infty\big)_i.$$

That is, the ith component of $\mathcal{G}(H)$ is the ith component of the result of applying $T_{\{i\}}$ to the intersection of T_A^∞ for all $A \in H$ containing i. We call $\mathcal{G}(H)$ the **outcome of iterated elimination (of non-ϕ-optimal strategies) with respect to** H. Note that $\mathcal{G}(H)$ implicitly depends on ϕ.

Let us "walk through" this definition to understand it better. Given a player i and a hyperarc $A \in H$ such that $i \in A$, T_A^∞ is the outcome of iterated elimination on A, starting at $(S_1, \ldots, S_n)$. The strategies of players from outside of A are not affected by this process. This elimination process is performed for each hyperarc that i is a member of. By intersecting the outcomes, i.e., by considering the restriction $\bigcap_{A:i\in A\in H} T_A^\infty$, one arrives at a restriction in which all such "group-wise" iterated eliminations have taken place. However, in this restriction some of the strategies of player i may be non-ϕ-optimal. They are eliminated using one application of the $T_{\{i\}}$ operator. We illustrate this process, and in particular this last step, in the following.

3.3.1. Example. Consider local strict dominance, sd^ℓ, in the following three-player game $\mathcal{G}$ where the payoffs of players 1 and 2 and those of players 1 and 3 respectively depend on each other's actions, but the payoffs of player 2 and 3 are independent:

		Pl. 2,3			
		L, l	L, r	R, l	R, r
Pl. 1	U	$1, 1, 1$	$0, 1, 0$	$0, 0, 1$	$0, 0, 0$
	D	$0, 1, 1$	$1, 1, 0$	$1, 0, 1$	$1, 0, 0$

That is, for example, if player 1 chooses strategy U, player 2 chooses L, and player 3 chooses r, then the payoff for player 1 is 0, player 2 gets 1, and player 3 gets 0. Now assume an interaction structure which reflects these payoff dependencies, i.e., a hypergraph $H = \{\{1, 2\}, \{1, 3\}\}$ (but note that this is just one particular example, in general the interaction structure need not reflect the payoff dependencies). We obtain $T^\infty_{\{1,2\}} = (\{U, D\}, \{L\}, \{l, r\})$ and $T^\infty_{\{1,3\}} = (\{U, D\}, \{L, R\}, \{l\})$. The restriction defined by these two outcomes is $(\{U, D\}, \{L\}, \{l\})$, and in the final step player 1 can now combine the results from his two independent interactions and eliminate his strategy D by one application of $T_{\{1\}}$. The outcome of the whole process is thus $\mathcal{G}(H) = (\{U\}, \{L\}, \{l\})$.

In this example, the outcome with respect to the given interaction structure coincides with the outcome of customary IESDS on the fully specified game matrix. We should emphasize that this is not the case in general, and the purpose of this example is simply to illustrate how the operators work. Example 3.3.3 later on shows in a different setting how the interaction structure can influence the outcome.

Note that when H consists of the single hyperarc N that contains all the players, then for each player i, $\bigcap_{A:i\in A\in H} T_A^\infty$ reduces to T_N^∞, and this is closed under application of each operator $T_{\{i\}}$. So then, indeed, $\mathcal{G}(H) = T_N^\infty$, that is, $\mathcal{G}(H)$ in this special case coincides with the customary outcome of iterated elimination of non-ϕ-optimal strategies.

In general, this customary outcome is included in the outcome w.r.t. any hypergraph H. This result is established in Theorem 3.3.2, and Example 3.3.3 shows a case where the inclusion is proper.

3.3.2. Theorem. *For $\phi \in \{sd^\ell, sd^g\}$ and for all hypergraphs H, we have $T_N^\infty \subseteq \mathcal{G}(H)$.*

Proof. First, consider $\phi = sd^g$, and $G \subseteq G' \subseteq N$. By definition, this implies that for all restrictions $\mathcal{G}'$ we have $T_{G'}(\mathcal{G}') \subseteq T_G(\mathcal{G}')$. Since ϕ is monotonic, so is the operator T_C for all $C \subseteq N$. Hence by a straightforward induction $T_N^\infty \subseteq T_G^\infty$ for all $G \subseteq N$, and consequently, for all players i,

$$T_N^\infty \subseteq \bigcap_{A:i\in A\in H} T_A^\infty. \tag{3.1}$$

Hence, for all $i \in N$,

$$T_N^\infty = T_{\{i\}}(T_N^\infty) \subseteq T_{\{i\}}(\bigcap_{A:i\in A\in H} T_A^\infty),$$

where the inclusion holds by the monotonicity of $T_{\{i\}}$. Consequently $T_N^\infty \subseteq \mathcal{G}(H)$.

We now prove the same claim for $\phi = sd^\ell$. We need to distinguish the T_C operator for $\phi = sd^\ell$ and $\phi = sd^g$. In the former case we write $T_{C,\ell}$ and in the latter case $T_{C,g}$. The reason that we use the latter operators is that they are monotonic and closely related to the former operators. Namely, as noted in [4], $T_{N,\ell}^\infty = T_{N,g}^\infty$, and the proof carries over for N replaced by an arbitrary $C \subseteq N$. Now fix an arbitrary $i \in N$, then

$$\bigcap_{A:i\in A\in H} T_{A,g}^\infty = \bigcap_{A:i\in A\in H} T_{A,\ell}^\infty,$$

and by (3.1) for $\phi = sd^g$, $T_{N,g}^\infty \subseteq \bigcap_{A:i\in A\in H} T_{A,g}^\infty$, so

$$T_{N,\ell}^\infty = T_{N,g}^\infty \subseteq \bigcap_{A:i\in A\in H} T_{A,\ell}^\infty. \tag{3.2}$$

Further, we have $T_{N,\ell}^\infty = T_{N,g}^\infty$ and $T_{N,g}^\infty = T_{\{i\},g}(T_{N,g}^\infty)$, so $T_{N,\ell}^\infty = T_{\{i\},g}(T_{N,\ell}^\infty)$. Hence, by (3.2) and monotonicity of $T_{\{i\},g}$,

$$T_{N,\ell}^\infty = T_{\{i\},g}(T_{N,\ell}^\infty) \subseteq T_{\{i\},g}(\bigcap_{A:i\in A\in H} T_{A,\ell}^\infty).$$

Also, for all $i \in N$ and all restrictions $\mathcal{G}'$ we have, by definition,

$$T_{\{i\},g}(\mathcal{G}') \subseteq T_{\{i\},\ell}(\mathcal{G}'),$$

so by the last inclusion

$$T^{\infty}_{N,\ell} \subseteq T_{\{i\},\ell}(\textstyle\bigcap_{A:i\in A\in H} T^{\infty}_{A,\ell}).$$

Consequently, $T^{\infty}_{N,\ell} \subseteq \mathcal{G}(H)$, as desired.

□

The inclusion proved in this result cannot be reversed, even when each pair of players shares a hyperarc. The following Example 3.3.3 also proves the intuition that such a graph structure can be less informative than a proper (non-degenerate) hypergraph structure.

3.3.3. Example. Consider the following strategic game with three players. The payoffs of player 1 and 2 here only depend on each other's choices, and the payoffs of player 3 only depend on the choices of player 2 and 3:

		Pl. 2 L	Pl. 2 R
Pl. 1	U	$0, 1$	$0, 0$
	D	$1, 0$	$1, 1$

Payoff of players 1 and 2

		Pl. 2 L	Pl. 2 R
Pl. 3	A	0	1
	B	1	0

Payoff of player 3

If we assume the hypergraph H that consists of the single hyperarc $\{1, 2, 3\}$, then the outcome of iterated elimination of non-ϕ-optimal strategies w.r.t. H is the customary outcome which equals $(\{D\}, \{R\}, \{A\})$. Indeed, player 1 can eliminate his strictly dominated strategy U, then player 2 can eliminate L, and subsequently player 3 can eliminate B.

In contrast, if the hypergraph consists of all pairs of players, so $H = \{\{1, 2\}, \{2, 3\}, \{1, 3\}\}$, then the outcome of iterated elimination of non-ϕ-optimal strategies w.r.t. H equals $(\{D\}, \{R\}, \{A, B\})$.

Informally, the reason for this difference is that in the latter case, player 3 can eliminate B only using the fact that player 2 eliminated L, but this information is available only to players 1 and 2.

To familiarize ourselves further with our definitions, we establish the following intuitive monotonicity result. Say that H' extends H just if for each $A \in H$ there is $A' \in H'$ such that $A \subseteq A'$.

3.3.4. Proposition. *If H' extends H and T is monotonic, then $\mathcal{G}(H') \subseteq \mathcal{G}(H)$.*

Proof. We prove the following stronger proposition: If H' extends H and for each $G \in \{\{i\} \mid i \in N\} \cup H \cup H'$, T_G is monotonic, then $\mathcal{G}(H') \subseteq \mathcal{G}(H)$.

First note that for all restrictions $\mathcal{G}'$:

$$G \subseteq G' \text{ implies } T_{G'}(\mathcal{G}') \subseteq T_G(\mathcal{G}'). \tag{3.3}$$

To see this, suppose that $G \subseteq G'$. Then for all $i \in N$, either

- $i \in G$, in which case $i \in G'$, so $T_G(\mathcal{G}')_i = T_{G'}(\mathcal{G}')_i$, or
- $i \notin G$, in which case $T_G(\mathcal{G}')_i = \mathcal{G}'_i \supseteq T_{G'}(\mathcal{G}')_i$.

In each case, $T_{G'}(\mathcal{G}')_i \subseteq T_G(\mathcal{G}')_i$.

From (3.3) and the monotonicity of T_G and $T_{G'}$, it follows that:

$$G \subseteq G' \text{ implies } T^\infty_{G'} \subseteq T^\infty_G. \tag{3.4}$$

Now we show that if H' extends H, then $\bigcap_{A':i\in A'\in H'} T^\infty_{A'} \subseteq \bigcap_{A:i\in A\in H} T^\infty_A$: Fix $i \in N$ and take some $s_i \notin (\bigcap_{A:i\in A\in H} T^\infty_A)_i$. Then there is $A \in H$ such that $i \in A$ and $s_i \notin (T^\infty_A)_i$. Then since H' extends H, there is $A' \in H'$ such that $A \subseteq A'$, so by (3.4), $s_i \notin (T^\infty_{A'})_i \supseteq \bigcap_{A':i\in A'\in H'} T^\infty_{A'}$.

So, since each $T_{\{i\}}$ is monotonic,

$$\mathcal{G}(H')_i = T_{\{i\}}\left(\bigcap_{A':i\in A'\in H'} T^\infty_{A'}\right)_i \subseteq T_{\{i\}}\left(\bigcap_{A:i\in A\in H} T^\infty_A\right)_i = \mathcal{G}(H)_i.$$

□

3.3.2 Intermediate states

The setting considered in Section 3.3.1 corresponds to a state where within each hyperarc, all its members have shared all information about their preferences. Given the game $\mathcal{G}$ and the hypergraph H, the outcome $\mathcal{G}(H)$ defined there thus reflects, assuming the process described in Section 3.1.1, what strategies players who initially know only their own preferences can eliminate if they communicate all they can communicate in H. We now define formally what communication we assume possible, and then look at intermediate states, where only certain preferences have been communicated.

Any player i can communicate his preferences to any $A \in H$ with $i \in A$. We take a **message** by i to consist of a preference statement $s'_i \succ_{s_{-i}} s_i$ for $s_i, s'_i \in S_i$ and $s_{-i} \in S_{-i}$. We denote such a message by $(i, A, s'_i \succ_{s_{-i}} s_i)$, and require that $i \in A$ and that it is **truthful** with respect to the given game $\mathcal{G}$, that is, that indeed $s'_i \succ_{s_{-i}} s_i$ in $\mathcal{G}$. Note that the fact that i is the sender is, strictly speaking, never used. Thus, in accordance with the interpretation of communication described in Section 3.1.1, we may drop the sender and simply say "the players in A commonly observe that $s'_i \succ_{s_{-i}} s_i$". An **intermediate state** now is given by the set M of messages which have been communicated.

We now adjust the definition of an optimality notion to account for intermediate states. An **intermediate optimality notion** $\phi_{G,M}$ (derived from an optimality notion ϕ) uses only information shared among the group G in the intermediate state given by M. That is, with singleton $G = \{i\}$ only i's preferences are used, and with larger G only preferences contained in messages to G are used. For example, $sd^g_{G,M}(s_i, \mathcal{G}')$ holds iff

$$\begin{array}{ll} \neg\exists s'_i \in S_i\, \forall s_{-i} \in S'_{-i}\, s'_i \succ_{s_{-i}} s_i & \text{if } G = \{i\} \\ \neg\exists s'_i \in S_i\, \forall s_{-i} \in S'_{-i}\, M \restriction_G \models s'_i \succ_{s_{-i}} s_i & \text{otherwise,} \end{array}$$

where by $M \restriction_G \models s'_i \succ_{s_{-i}} s_i$ we mean that $s'_i \succ_{s_{-i}} s_i$ is entailed by the messages contained in M which G received. Specifically, $M \restriction_G \models s'_i \succ_{s_{-i}} s_i$ iff there are $(\cdot, G^k, s^k_i \succ_{s_{-i}} s^{k+1}_i) \in M$ for $k \in \{1, \ldots, \ell - 1\}$ such that $G^k \supseteq G$, $s^1_i = s'_i$ and $s^\ell_i = s_i$.

We can then define a generalization of the T_G operator as follows:

$$T_{G,M}(\mathcal{G}') := (S''_1, \ldots, S''_n),$$

where $\mathcal{G}' = (S'_1, \ldots, S'_n)$ and for all $i \in N$

$$S''_i := \{s_i \in S'_i \mid \phi_{G,M}(s_i, \mathcal{G}')\}.$$

Note that, as before, S'_i remains unchanged if $i \notin G$, since then $\phi_{G,M}(s_i, \mathcal{G}')$ always holds. Indeed, for it to be false, there would have to be some message $(i, G, \cdot) \in M$, which would imply $i \in G$.

Similarly, we now define the **outcome of iterated elimination (of non-ϕ-optimal strategies) with respect to** H, M to be the restriction $\mathcal{G}(H, M)$, where for $i \in N$

$$\mathcal{G}(H, M)_i := T_{\{i\},M}\left(\bigcap_{A : i \in A \in \overline{H}} T^\infty_{A,M}\right)_i.$$

Here, $\overline{H}$ denotes the closure of H under intersection. That is, $H \subseteq \overline{H}$ and if $A, A' \in \overline{H}$ then also $A \cap A' \in \overline{H}$. This is necessary because certain information may be entailed by messages sent to different hyperarcs. For example, with $(j, A, s''_j \succ_{s_{-j}} s'_j), (j, A', s'_j \succ_{s_{-j}} s_j) \in M$, the combined information that $s''_j \succ_{s_{-j}} s_j$ is available to $A \cap A'$. Notice that this formulation leaves room for optimizations. For example, one start by looking at M and only consider groups to which there actually exist messages, but we do not focus on this issue here.

It is easy to see that in the case where the players have communicated all there is to communicate, i.e., for

$$M^{\text{all}}_H := \{(i, A, s'_i \succ_{s_{-i}} s_i) \mid i \in N, A \in H,\ s_i, s'_i \in S_i \text{ with } s'_i \succ_{s_{-i}} s_i \text{ in } \mathcal{G}\},$$

the intermediate outcome coincides with the previously defined outcome:

$$\mathcal{G}(H, M^{\text{all}}_H) = \mathcal{G}(H).$$

This corresponds to the intuition that $\mathcal{G}(H)$ captures the elimination process when all possible communication has taken place. In particular, all entailed information has also been communicated in M_H^{all}, which is why we did not need to consider $\overline{H}$ in Section 3.3.1.

Again, we "walk through" the definition of $\mathcal{G}(H, M)$. First, a local elimination process is run on each hyperarc of $\overline{H}$, using only information which has been communicated there (which now no longer covers all members' preferences, but only the ones according to the intermediate state M). Then, in the final step, each player combines his insights from all hyperarcs of which he is a member, and he eliminates any strategies that he thereby learns not to be optimal.

3.3.5. Example. Consider again the game $\mathcal{G}$ from Example 3.3.1, and the initial state where $M = \emptyset$.

We have $T^{\infty}_{A,M} = \mathcal{G}$ for all $A \in \overline{H}$, that is, without communication no strategy can "commonly" be eliminated. However, players 2 and 3 can "privately" eliminate one of their strategies each, since each knows his own preferences. This fact and the effect that this elimination cannot be iterated upon by other players are captured in the final step performed by these respective players. The results of the final steps are thus

$$\begin{aligned}
T_{\{1\},M}(\textstyle\bigcap_{A:1\in A\in\overline{H}} T^{\infty}_{A,M}) &= (\{U, D\}, \{L, R\}, \{l, r\}),\\
T_{\{2\},M}(\textstyle\bigcap_{A:2\in A\in\overline{H}} T^{\infty}_{A,M}) &= (\{U, D\}, \{L\}, \{l, r\}),\\
T_{\{3\},M}(\textstyle\bigcap_{A:3\in A\in\overline{H}} T^{\infty}_{A,M}) &= (\{U, D\}, \{L, R\}, \{l\}),
\end{aligned}$$

so the overall outcome is

$$\mathcal{G}(H, M) = (\{U, D\}, \{L\}, \{l\}).$$

Consider now the intermediate state

$$M' = \{(2, \{1, 2\}, L \succ_{s_{-2}} R) \mid s_{-2} \in S_{-2}\},$$

that is, a state where player 2 has shared with player 1 the information that for any joint strategy of players 1 and 3, he prefers his strategy L over R. Then only the result of player 1 changes:

$$T_{\{1\},M'}(\textstyle\bigcap_{A:1\in A\in\overline{H}} T^{\infty}_{A,M'}) = (\{U, D\}, \{L\}, \{l, r\}),$$

while the other results and the overall outcome remain the same. If additionally player 3 communicates all his information in the hyperarc he shares with player 1, that is, if the intermediate state is

$$M'' = M' \cup \{(3, \{1, 3\}, l \succ_{s_{-3}} r) \mid s_{-3} \in S_{-3}\},$$

then player 1 can combine all the received information and obtain

$$T_{\{1\},M''}(\bigcap_{A:1\in A\in\overline{H}} T^{\infty}_{A,M''}) = (\{U\},\{L\},\{l\}).$$

This is also the overall outcome $\mathcal{G}(H,M'')$, which coincides with the outcome $\mathcal{G}(H,M^{\mathrm{all}}_H)$ where all information has been communicated.

Let us now illustrate the importance of using entailment in intermediate optimality notions and $\overline{H}$ (rather than H) in the definition of $\mathcal{G}(H,M)$.

3.3.6. Example. We look at a game involving four players, but we are only interested in the preferences of two of them. The other two players serve merely to create different hyperarcs. The strategies and payoffs of player 1 and 2 are as follows:

		Pl. 2	
		L	R
Pl. 1	A	$3,0$	$1,1$
	B	$2,0$	$1,1$
	C	$1,1$	$0,0$
	D	$0,0$	$5,1$

For player 3 and 4 we assume a "dummy" strategy each, denoted X and Y. Consider the hypergraph

$$H = \{\{1,2,3\},\{1,2,4\}\}$$

and the intermediate state

$$\begin{aligned} M = \{&(1,\{1,2,3\}, A \succ_{LXY} B),\\ &(1,\{1,2,4\}, B \succ_{LXY} C),\\ &(1,\{1,2,3\}, A \succ_{RXY} C)\}. \end{aligned}$$

The fact that player 1, independently of what the remaining players do, strictly prefers A over C is not explicit in these pieces of information, but it is *entailed* by them, since $A \succ_{LXY} B$ and $B \succ_{LXY} C$ imply $A \succ_{LXY} C$. However, this combination of information is only available to $\{1,2,3\}\cap\{1,2,4\}$.

Player 2 can make use of this fact that C is dominated, and eliminate his own strategy L. If we now look at a state where player 2 has communicated his relevant preferences,

$$M' = M \cup \{(2,\{1,2,3\}, R \succ_{\alpha XY} L) \,|\, \alpha \in \{A,B,D\}\},$$

we notice that player 1, in turn, can eliminate A and B, but only building upon the initial combination of information available to $\{1,2,3\}\cap\{1,2,4\}$. There is no single hyperarc in the original hypergraph which has all the required information available. It thus becomes clear that we need to take into account iterated elimination on intersections of hyperarcs.

3.4 Epistemic foundation

In this section, we provide an epistemic foundation for our framework. The aim is to prove that the definition of the outcome $\mathcal{G}(H, M)$ correctly captures what strategies the players can eliminate using all they "know", in a formal sense.

We proceed as follows. First, in Section 3.4.1, we briefly introduce an epistemic model formalizing the players' knowledge. We draw upon results from Chapter 2. In Section 3.4.2, we give a general epistemic formulation of strict dominance and argue that it correctly captures the notion. Section 3.4.2 contains the main result of our epistemic analysis, namely that the outcome $\mathcal{G}(H, M)$ indeed yields the outcome stipulated by the epistemic formulation.

We focus on the global version of strict dominance, sd^g, mainly because the presentation is then more concise. However, our results about its outcomes carry over to the local version due to the equivalence result mentioned in the proof of Theorem 3.3.2.

3.4.1 Epistemic language and states

Again, we assume a fixed game $\mathcal{G}$ with strategies S_i for each player i, and a hypergraph H representing the interaction structure. Analogously to Chapter 2, we use a propositional **epistemic language** with a set At of **atoms** which is divided into disjoint subsets At_i, one for each player i, where

$$\mathrm{At}_i = \{s_i' \succ_{s_{-i}} s_i \mid s_i, s_i' \in S_i, s_{-i} \in S_{-i}\}.$$

The set At describes all possible relative preferences between pairs of strategies. We consider the usual **connectives** $\wedge$ and $\vee$ (but not the negation $\neg$), and a **common knowledge** operator C_G for any group $G \subseteq N$ of players. As in Chapter 2, we write K_i for $C_{\{i\}}$. By $\mathcal{L}^+$ we denote the set of formulas built from the atoms in At using these two connectives and knowledge operators.

A **valuation** V is a subset of At, consisting of those atoms that are assumed true. A concrete game $\mathcal{G}$ *induces* exactly one valuation which correctly represents it, but since we need to model that players may not have full knowledge of the game and may consider other preference orders possible, we need to allow other options. However, since the setup is commonly known, players will only consider valuations possible which indeed may reflect the preference ordering of some game. Therefore, we require valuations V to be such that for each i and each $s_{-i} \in S_{-i}$, the restriction $V \cap \{\cdot \succ_{s_{-i}} \cdot\}$ represents a strict partial order. For example, $\{s \succ_a t\}$ is a valuation (given a game with appropriate strategy sets), while $\{s \succ_a t, t \succ_a u\}$ and $\{s \succ_a t, t \succ_a s\}$ are not. Note that such valuations satisfy the properties (v1) and (v2) from Chapter 2, so the framework discussed there can be applied.

Recall from Section 3.3.2 that a **message** from player i to $A \in H$ has the form $(i, A, s_i' \succ_{s_{-i}} s_i)$, where $i \in A$, $s_i, s_i' \in S_i$, and $s_{-i} \in S_{-i}$. Truthfulness now

depends on the particular valuation under consideration. Concretely, a message $(\cdot, \cdot, p)$ is **truthful** with respect to a valuation V if, indeed, $p \in V$.

A **state**, or **possible world**, is a pair (V, M), where V is a valuation and M is a set of truthful (with respect to V) messages.

Our setting is an instance of the framework defined in Chapter 2, and the formal **semantics** is as defined there (Section 2.2). We repeat here only the intuition that $C_G\varphi$ means that φ is *common knowledge* among G, that is, everybody in G knows φ, everybody knows that everybody knows φ, etc. In particular, $K_i\varphi$ means that player i *knows* φ. Our assumptions that player i from the beginning knows the true facts in At_i, and that the basic assumptions from Section 3.1.1 are commonly known among the players, are reflected in the setup we have described.

3.4.2 Correctness result

We here use our insights from Chapter 2 in order to prove that the T operator defined in Section 3.3 is correct with respect to an epistemic formulation of our setting.

We start by giving an epistemic formula describing the global version of iterated elimination of strictly dominated strategies. Note that, in contrast to the formulation in Section 3.2, this formula states when a strategy is *known to be* strictly dominated.

We define, for $i \in N$ and $s_i \in S_i$,

$$dom^1(s_i) := K_i \bigvee_{s_i' \in S_i} \bigwedge_{s_{-i} \in S_{-i}} s_i' \succ_{s_{-i}} s_i,$$

$$dom^{\ell+1}(s_i) := K_i \bigvee_{s_i' \in S_i} \bigwedge_{s_{-i} \in S_{-i}} \big(s_i' \succ_{s_{-i}} s_i \vee \bigvee_{j \in N \setminus \{i\}} dom^{\ell}(s_j)\big).$$

That is, in the base case, player i knows that s_i is strictly dominated if i knows that there is an alternative strategy s_i' which, for all joint strategies of the other players, is strictly preferred. Furthermore, after iteration $\ell + 1$, i knows that s_i is strictly dominated if i knows that there is an alternative strategy s_i' such that, for all joint strategies s_{-i} of the other players, either s_i' is strictly preferred or some strategy s_j in s_{-i} is already known by j to be strictly dominated after iteration ℓ.

We restrict attention to formulas $dom^{\ell}(s_i)$ with $\ell \in \{1, \ldots, \hat{\ell}\}$, where $\hat{\ell} = \sum_{i \in N} |S_i|$. By the semantics of the considered formulas, there is some ℓ within this range such that for all $\ell' \geq \ell$, $dom^{\ell'}$ is equivalent to dom^{ℓ}. To reflect the fact that this can be seen as the outcome of the iteration, we denote $dom^{\hat{\ell}}$ by dom^{∞}.

As a first connection with the T operator defined in Section 3.3, we have the following epistemic counterpart of Proposition 3.3.4. This is due to the fact that, intuitively, if we look at states where all communication allowed by a given

hypergraph has taken place, then knowledge (of positive formulas) can only grow as that hypergraph grows.

3.4.1. Proposition. *If H' extends H, then for any $i \in N$ and $s_i \in S_i$,*

$$(V, M^{\text{all}}_H) \vDash dom^\infty(s_i) \text{ implies } (V, M^{\text{all}}_{H'}) \vDash dom^\infty(s_i),$$

where M^{all} is as defined in Section 3.3.2.

Proof. Follows from Lemma 2.3.3 and the fact that $dom^\infty(s_i) \in \mathcal{L}^+$. □

We now proceed to the main result of this section. We prove that the non-epistemic formulation of iterated elimination of non-sd^g-optimal strategies, as given in Section 3.3, is correct with respect to the epistemic formulation of strict dominance.

3.4.2. Theorem. *For any strategic game $\mathcal{G}$, hypergraph H, set of messages M (truthful with respect to $\mathcal{G}$), and $i \in N$,*

$$\mathcal{G}(H, M)_i = \{s_i \in S_i \mid (V, M) \nvDash dom^\infty(s_i)\},$$

where V is the valuation induced by $\mathcal{G}$.

In order to prove this result, we need some preparatory steps.

3.4.3. Lemma. *For any $\ell \geq 1$, $i \in N$, $s_i \in S_i$, and state (V, M),*

$$\begin{aligned} &(V, M) \vDash dom^{\ell+1}(s_i) \\ \textit{iff } &(V, M) \vDash \bigvee_{s'_i \in S_i} \bigwedge_{s_{-i} \in S_{-i}} \big((K_i s'_i \succ_{s_{-i}} s_i) \vee \bigvee_{A : i \in A \in \overline{H}} \bigvee_{j \in A \setminus \{i\}} C_A dom^\ell(s_j)\big). \end{aligned}$$

Proof. We have

$$\begin{aligned} &(V, M) \vDash dom^{\ell+1}(s_i) \\ \text{iff } &\textit{(by definition)} \\ &(V, M) \vDash K_i \bigvee_{s'_i \in S_i} \bigwedge_{s_{-i} \in S_{-i}} \big(s'_i \succ_{s_{-i}} s_i \vee \bigvee_{j \in N \setminus \{i\}} dom^\ell(s_j)\big) \\ \text{iff } &\textit{(by Lemma 2.3.5)} \\ &(V, M) \vDash \bigvee_{s'_i \in S_i} \bigwedge_{s_{-i} \in S_{-i}} \big(K_i s'_i \succ_{s_{-i}} s_i \vee \bigvee_{j \in N \setminus \{i\}} K_i dom^\ell(s_j)\big) \\ \text{iff } &(V, M) \vDash \bigvee_{s'_i \in S_i} \bigwedge_{s_{-i} \in S_{-i}} \big(K_i s'_i \succ_{s_{-i}} s_i \vee \bigvee_{A : i \in A \in \overline{H}} \bigvee_{j \in A \setminus \{i\}} C_A dom^\ell(s_j)\big). \end{aligned}$$

The last step holds by Lemma 2.3.8 and Theorem 2.3.9 since $dom^\ell(s_j) = K_j(\cdots)$. □

3.4.4. Lemma. *For any $\ell \geq 1$, $i \in A \in \overline{H}$, $s_i \in S_i$, and state (V, M),*

$$s_i \notin T^{\ell}_{A,M}(S_1, \ldots, S_n)_i \text{ iff } (V, M) \vDash C_A dom^{\ell}(s_i).$$

Proof. By induction on ℓ. The base case follows straightforwardly from the definitions. Now assume the claim holds for ℓ. Then, focusing on the interesting case where $A \neq \{i\}$, we have the following chain of equivalences:

$$\begin{aligned}
& s_i \notin T^{\ell+1}_{A,M}(S_1, \ldots, S_n)_i \\
\text{iff } & \textit{(by definition)} \\
& s_i \notin T^{\ell}_{A,M}(S_1, \ldots, S_n)_i \text{ or } \neg sd^g_{A,M}(s_i, T^{\ell}_{A,M}(S_1, \ldots, S_n)) \\
\text{iff } & \textit{(by monotonicity of } sd^g\textit{)} \\
& \neg sd^g_{A,M}(s_i, T^{\ell}_{A,M}(S_1, \ldots, S_n)) \\
\text{iff } & \textit{(by definition)} \\
& \exists s'_i \in S_i \, \forall s_{-i} \in T^{\ell}_{A,M}(S_1, \ldots, S_n)_{-i} \; M \restriction_A \vDash s'_i \succ_{s_{-i}} s_i \\
\text{iff } & \exists s'_i \in S_i \, \forall s_{-i} \in S_{-i} \; M \restriction_A \vDash s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad s_{-i} \notin T^{\ell}_{A,M}(S_1, \ldots, S_n)_{-i} \\
\text{iff } & \exists s'_i \in S_i \, \forall s_{-i} \in S_{-i} \; M \restriction_A \vDash s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad \exists j \in A \setminus \{i\} \; s_j \notin T^{\ell}_{A,M}(S_1, \ldots, S_n)_j \\
\text{iff } & \textit{(by induction hypothesis)} \\
& \exists s'_i \in S_i \, \forall s_{-i} \in S_{-i} \; M \restriction_A \vDash s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad \exists j \in A \setminus \{i\} \; (V, M) \vDash C_A dom^{\ell}(s_j) \\
\text{iff } & \textit{(by Lemma 2.3.8)} \\
& \exists s'_i \in S_i \, \forall s_{-i} \in S_{-i} \; (V, M) \vDash C_A s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad \exists j \in A \setminus \{i\} \; (V, M) \vDash C_A dom^{\ell}(s_j) \\
\text{iff } & (V, M) \vDash \bigvee_{s'_i \in S_i} \bigwedge_{s_{-i} \in S_{-1}} \left(C_A s'_i \succ_{s_{-i}} s_i \vee \bigvee_{j \in A \setminus \{i\}} C_A dom^{\ell}(s_j) \right) \\
\text{iff } & \textit{(by Theorem 2.3.6)} \\
& (V, M) \vDash C_A \bigvee_{s'_i \in S_i} \bigwedge_{s_{-i} \in S_{-1}} \left(s'_i \succ_{s_{-i}} s_i \vee \bigvee_{j \in A \setminus \{i\}} dom^{\ell}(s_j) \right) \\
\text{iff } & (V, M) \vDash C_A dom^{\ell+1}(s_i).
\end{aligned}$$

□

We are now ready to prove the main result.

Proof of Theorem 3.4.2. We have:

$$
\begin{aligned}
& s_i \notin \mathcal{G}(H,M)_i \\
\text{iff } & \textit{(by definition)} \\
& s_i \notin T_{\{i\},M}(\textstyle\bigcap_{A:i\in A\in\overline{H}} T^\infty_{A,M})_i \\
\text{iff } & \neg sd^g_{\{i\},M}(s_i, \textstyle\bigcap_{A:i\in A\in\overline{H}} T^\infty_{A,M}(S_1,\ldots,S_n)) \\
\text{iff } & \exists s'_i \in S_i\, \forall s_{-i} \in \textstyle\bigcap_{A:i\in A\in\overline{H}} T^\infty_{A,M}(S_1,\ldots,S_n)_{-i}\ s'_i \succ_{s_{-i}} s_i \\
\text{iff } & \exists s'_i \in S_i\, \forall s_{-i} \in S_{-i}\ s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad s_{-i} \notin \textstyle\bigcap_{A:i\in A\in\overline{H}} T^\infty_{A,M}(S_1,\ldots,S_n)_{-i} \\
\text{iff } & \exists s'_i \in S_i\, \forall s_{-i} \in S_{-i}\ s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad \exists A : i\in A\in\overline{H}\ s_{-i} \notin T^\infty_{A,M}(S_1,\ldots,S_n)_{-i} \\
\text{iff } & \exists s'_i \in S_i\, \forall s_{-i} \in S_{-i}\ s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad \exists A : i\in A\in\overline{H}\ \exists j\in A\setminus\{i\} : s_j \notin T^\infty_{A,M}(S_1,\ldots,S_n)_j \\
\text{iff } & \textit{(by Lemma 3.4.4)} \\
& \exists s'_i \in S_i\, \forall s_{-i} \in S_{-i}\ s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad (V,M) \vDash \bigvee_{A:i\in A\in\overline{H}} \bigvee_{j\in A\setminus\{i\}} C_A dom^\infty(s_j) \\
\text{iff } & \textit{(since } s'_i \succ_{s_{-i}} s_i \in \mathrm{At}_i\textit{)} \\
& \exists s'_i \in S_i\, \forall s_{-i} \in S_{-i}\ (V,M) \vDash K_i s'_i \succ_{s_{-i}} s_i \text{ or} \\
& \qquad\qquad (V,M) \vDash \bigvee_{A:i\in A\in\overline{H}} \bigvee_{j\in A\setminus\{i\}} C_A dom^\infty(s_j) \\
\text{iff } & (V,M) \vDash \bigvee_{s'_i\in S_i} \bigwedge_{s_{-i}\in S_{-i}} \big((K_i s'_i \succ_{s_{-i}} s_i) \vee \bigvee_{A:i\in A\in\overline{H}} \bigvee_{j\in A\setminus\{i\}} C_A dom^\infty(s_j)\big) \\
\text{iff } & \textit{(by Lemma 3.4.3)} \\
& (V,M) \vDash dom^\infty(s_i).
\end{aligned}
$$

□

3.5 Distributed implementation

An epistemic model such as the one we have built and used in Chapter 2 and Section 3.4 allows us to reason about the players' knowledge. However, such a model always takes the perspective of an outside observer, the modeler (in this case us), and tells us what the players can be said to know, assuming they are perfect reasoners. It thus *ascribes* knowledge to the players, without really telling us (or them, for that matter) how exactly they might arrive at that knowledge. Similarly, the T operator from Section 3.3 is formulated in a centralized way. How does the situation look from the players' point of view, and what reasoning mechanisms might they use?

Obviously, it cannot be the case that each player simply maintains a copy of our central model, or something equivalent to it, because the central model contains information about the whole system and all players, which is not available to an individual player. Of course, there may be situations where the players would have access to such a central model; for example, a virtual world might provide an interface to the control programs of its simulated inhabitants through which they can query a centralized model maintained by the world, and thus find out what they (can be said to) know. We explore this setting in Chapter 4, but in a truly distributed system this is not possible.

The aim of this section is to "localize" both the T operator and the epistemic model to obtain algorithms which can be executed by any player i himself. We straightforwardly see that they correspond to "i's part" of the centralized versions, and are in that sense correct for the elimination outcome of i's strategies, and for i's knowledge.

So, in Section 3.5.1, we *directly implement* an iterated elimination process localized to player i, and we easily see that it coincides with the centralized version of the T operator on i's strategies. Correctness is thus implied by Section 3.4, in the sense that the implementation really eliminates all strategies a player can eliminate, given his theoretically ascribed knowledge.

In Section 3.5.2, we follow the approach we have described in [155, 157] and called *explicit knowledge programming*, which involves encapsulating epistemic information and algorithms in a *knowledge module* local to any player i. This module processes the events that the player observes and enables him to transparently evaluate a certain class epistemic formulas talking about i's knowledge, which includes the $dom^{\ell}(s_i)$ formulas. Again using the results from Section 3.4, we see that the knowledge module is correct with respect to the given class of formulas. Exploiting the restrictions of our setting, the knowledge module is computationally simple, as opposed to, for example, a general, full-blown epistemic logic theorem prover. In this sense, our approach is in accordance with Hayes [77]: such a knowledge module is a system which "[has] a logical inference structure—[is] making deductively valid inferences—without being a classical uniform theorem-prover which just 'grinds clauses together'." See Section 3.6 for some more discussion on this approach.

Common to both approaches is that the events a player observes need to be kept track of. So we assume that the program of any player i stores and can at any time access the initial strategy sets $(S_1, \ldots, S_n)$ of the given game $\mathcal{G}$, the given interaction structure H, i's own preferences $\succ_i$ induced by $\mathcal{G}$, as well as the messages M_i he has observed. For the sake of clarity, we use $C(M_i)$ to denote the *transitive closure* of the messages that have been observed by i. That is, $M_i \subseteq C(M_i)$, and if $(j, A, s''_j \succ_{s_{-j}} s'_j), (j, A', s'_j \succ_{s_{-j}} s_j) \in C(M_i)$, then also $(j, A \cap A', s''_j \succ_{s_{-j}} s_j) \in C(M_i)$.

Note that the algorithms we describe are symmetric in the sense of Chapter 1, and thus JCSP lends itself as implementation platform.

Algorithm 1: Computing $\mathcal{G}(H, M)_i$ using a T operator implementation

```
   // compute ⋂_{A:i∈A∈H̄} T^∞_{A,M}
 1 foreach A with i ∈ A ∈ H̄ do
 2     (S'_{1,A}, ..., S'_{n,A}) := (S_1, ..., S_n);
 3     repeat
 4         changed := false;
 5         (S''_{1,A}, ..., S''_{n,A}) := (S'_{1,A}, ..., S'_{n,A});
 6         foreach j ∈ A and s_j ∈ S'_{j,A} do
 7             if ∃s'_j ∈ S'_{j,A} ∀s_{-j} ∈ S'_{-j,A} (j, A, s'_j ≻_{s_{-j}} s_j) ∈ C(M_i) then
 8                 S''_{j,A} := S''_{j,A} \ {s_j};
 9                 changed := true;
10             end
11         end
12         (S'_{1,A}, ..., S'_{n,A}) := (S''_{1,A}, ..., S''_{n,A});
13     until not changed;
14 end
15 (S'_1, ..., S'_n) := ⋂_{A:i∈A∈H̄} (S'_{1,A}, ..., S'_{n,A});
   // compute T_{{i},M}
16 S''_i := S'_i;
17 foreach s_i ∈ S'_i do
18     if ∃s'_i ∈ S'_i ∀s_{-i} ∈ S'_{-i} s'_i ≻_{s_{-i}} s_i then
19         S''_i := S''_i \ {s_i};
20     end
21 end
22 return S''_i;
```

3.5.1 T operator approach

Algorithm 1 describes an implementation of the T operator as defined in Section 3.3.2. This implementation can straightforwardly be seen to execute directly the definition of the T operator; the only noteworthy change occurs in line 7, where we have M_i instead of M. This, however, does not make any difference, since with $i \in A$ we have $(j, A, s'_j \succ_{s_{-j}} s_j) \in M_i$ if and only if $(j, A, s'_j \succ_{s_{-j}} s_j) \in M$. Intuitively speaking, the evaluation of $T^{\infty}_{A,M}$ only uses information shared by all members of A, and thus locally available to each member.

The implementation is certainly not the most efficient one, for example, one might first look at M_i in order to see which hyperarcs of $\overline{H}$ even need to be considered. But for our purposes this suffices, and we now focus on the knowledge module approach.

3.5.2 Knowledge module approach

In the alternative, modular approach, the player program explicitly uses the epistemic formulation defined in Section 3.4.2, evaluating epistemic formulas of the form dom^ℓ in order to test which strategies are known to be dominated. Since we are here, as explained in Section 3.1.1, not examining exactly *why* certain communications are performed, we do not discuss the whole player program in detail. It can be thought of as a main loop consisting of communication statements and, whenever communication has taken place, tests involving dom^ℓ formulas in order to determine what strategies can currently be eliminated. Whenever the player program encounters such an epistemic formula, it calls a function to evaluate it.

This evaluation function is what we focus on here. It is provided by the player's *knowledge module*, which also keeps track of the relevant information (i.e., the observed messages). We here provide an evaluation function for such a knowledge module, using the results from Chapter 2. It correctly evaluates a more general class of formulas than only the dom^ℓ: For any $\varphi \in \mathcal{L}^+$, at any intermediate state M, it can efficiently compute whether $(V, M) \vDash K_i\varphi$, where V is as induced by $\mathcal{G}$.

Note that, even though (V, M) refers to all players, the knowledge module is only allowed to use the information available to i, that is, $\succ_i$ and M_i. For this reason, even though the formulation can be thought of as a *model checking* problem, in general it is closer to a *validity* check: i has to check whether the formula in question, $K_i\varphi$, holds in *all models compatible* with his information; or, if we represent his information as formulas $\psi_1, \ldots, \psi_\ell$, whether $\vDash \neg(\psi_1 \wedge \ldots \wedge \psi_\ell) \vee K_i\varphi$. While the exact relationship between model checking and testing for validity depends on the concrete formalism, in general, testing for validity is intractable [74]. With the specific restrictions of our scenario, however, our case indeed turns out to correspond to a rather simple instance of model checking, since all models compatible with i's information are equivalent with respect to i's knowledge, so only one of them needs to be checked.

The straightforward implementation is described in Algorithm 2. To test whether a player i knows a formula $\varphi \in \mathcal{L}^+$, he needs to execute $eval(i, \varphi)$. Recall that, for a word $w = i_1 \ldots i_\ell \in N^*$, we use K_w to abbreviate $K_{i_1} \ldots K_{i_\ell}$ and $Set(w)$ to mean $\{i_1, \ldots, i_\ell\}$.

The evaluation is done in a recursive way, directly reflecting the semantics as defined in Chapter 2. It analyzes the formula at hand and evaluates its components, collecting the K operators it sees on the way until an atom is reached, over which the chain of collected K operators is then evaluated. This procedure is possible due to the fact that K distributes over all connectives we use, as established in Chapter 2. Some comments on particular lines of the algorithm follow:

Line 3 reflects that i knows his own preferences, and is in accordance with the semantics of our model.

Algorithm 2: Knowledge evaluation function $eval(w, \varphi)$ of player i

Input: $w \in N^*$, $\varphi \in \mathcal{L}^+$
Output: true if $(V, M) \vDash K_w\varphi$; false otherwise

```
1  switch φ do
2      case p ∈ At
3          if Set(w) ⊆ {i} and p ∈ At_i then return true iff p ∈≻_i;
4          else if (·, A, p) ∈ C(M_i) with some A ⊇ Set(w) then return true;
5          else return false;
6      end
7      case φ_1 ∧ φ_2 return eval(w, φ_1) and eval(w, φ_2);
8      case φ_1 ∨ φ_2 return eval(w, φ_1) or eval(w, φ_2);
9      case K_j φ' with j ∈ N
10         if Set(w) ∪ {j} = {i} or there is A ∈ H̄ with Set(w) ∪ {j} ⊆ A then
11             return eval(w ∘ i, φ');
12         else return false;
13     end
14 end
```

Lines 4 and 5 reflect Lemma 2.3.8, with M equivalently replaced by M_i in line 5 since $i \in Set(w) \subseteq A$; intuitively, the respective message has been sent to A if and only if i has observed it, since $i \in A$.

Line 11 is correct because of Lemma 2.3.5.

Line 12 corresponds to Theorem 2.3.10 and allows the evaluation to be cut off if the set of collected K operators is not included in any $A \in \overline{H}$.

In particular, this last point has the effect that the exponential blowup caused by the recursive part of the dom^ℓ formulas depends on the hypergraph, rather than on the set of players. This makes evaluation especially efficient for hypergraphs with high locality.

While iterated elimination of strictly dominated strategies using the fully specified payoff matrix is a simple procedure [86], in a general setting of incomplete information, where arbitrary constellations of knowledge may occur, a player would have to maintain individual models for each other player, including each other player's models of each other players, and so on. With the restrictions in our scenario, these nested models collapse to one common model for each hyperarc, which allows us to apply the simpler procedures we described. The complexity of our algorithms therefore depends mainly on the hypergraph. In that sense, our framework simplifies the computation of knowledge in a similar way as the *graphical games* [85] mentioned in Section 3.1 simplify the computation of equilibria.

Pl. 1 \ Pl. 2	L	R
U	$0, 1$	$0, 0$
D	$1, 0$	$1, 1$

(a) Payoff of players 1 and 2

Pl. 3 \ Pl. 2	L	R
A	0	1
B	1	0

(b) Payoff of player 3

1. 1 concludes that U is dominated. 1's picture now:

	L	R
D		

	L	R
A		
B		

2. 2 communicates $L \succ_U R$ on $\{1, 2\}$
3. 2 communicates $L \succ_U R$ on $\{2, 3\}$
4. 2 communicates $R \succ_D L$ on $\{1, 2\}$
5. 2 communicates $R \succ_D L$ on $\{2, 3\}$
6. 1 communicates $D \succ U$ on $\{1, 2\}$
7. 1 concludes that 2 knows that 1 knows that U is dominated.
8. 1 concludes that 2 knows that L is dominated. 1's picture now:

	R
D	

	R
A	
B	

9. 2 concludes that 1 knows that U is dominated. 2's picture now:

	L	R
D		

	L	R
A		
B		

10. 2 concludes that L is dominated. 2's picture now:

	R
D	

	R
A	
B	

11. 3 communicates $B \succ_L A$ on $\{2, 3\}$
12. 3 communicates $A \succ_R B$ on $\{2, 3\}$
13. Communication is complete.

Figure 3.1: Protocol of program run for the game from Example 3.3.3. Messages are abbreviated, e.g., $L \succ_U R$ actually represents two messages: one containing $L \succ_{UA} R$ and one $L \succ_{UB} R$. When displaying a player's current picture of the game, we leave the matrix entries blank since the numerical payoffs are never communicated, only the relative preferences.

3.5.1. Example. Consider again the game from Example 3.3.3 with the hypergraph $H = \{\{1,2\},\{2,3\},\{1,3\}\}$. The game is depicted in Figure 3.1 together with a protocol of a program run in which players communicate their preferences and perform strategy elimination according to their respective current knowledge.

The outcome that the players arrive at after all communication allowed by H has taken place is, as expected, the same as in Example 3.3.3, for the reasons discussed there. While player 2 may deduce that player 3 would be able to eliminate B if player 3 knew that player 2 eliminated L, from the communicated information alone player 3 cannot deduce that.

Note that in this particular run, in line `8`, player 1 computes player 2's knowledge before player 2 actually computes it in line `10`. In that sense, player 1 for a certain amount of time ascribes knowledge to player 2 which player 2, strictly speaking, does not have. We come back to this issue at the end of Section 3.6.1.

3.6 Conclusions

In this chapter, we looked at strategic games in the presence of interaction structures. We assumed that initially the players know only their own preferences, and that they can truthfully communicate information about their own preferences within their parts of the interaction structure. We defined operators to perform iterated elimination of strictly dominated strategies in any given state of communication, along with an epistemic model, based on the framework from Chapter 2, showing that the outcome of these operators is in a certain sense correct. We also discussed distributed implementations of the resulting procedures, connecting back to Chapter 1.

3.6.1 Related work

A few more remarks may be in place about how our approach of *explicit knowledge programming* relates to related notions from the literature. The basic idea of the closely related topics of *explicit knowledge* and *algorithmic knowledge* [111, 59, 121, 75] is to restrict the "classic", logical notion of knowledge we have considered so far, in order to reflect what an agent could be said to *be able to compute*, rather than to *logically know*. One approach is to limit the accessibility relations in certain ways, reflecting the assumption that considering, or accessing, other possible worlds is computationally costly. However, these formalisms still take a modeler's point of view in order to reason about *what* such an agent can do, rather than describing exactly *how* the agent does it. The ascribing is, in a sense, taken to a higher level: Instead of knowledge itself, methods for computing knowledge are ascribed to the agents, and the resulting central model is then used, again, by the *modeler* in order to reason about what the agents can be said to be able to

compute. Naturally, the resulting logics may be more sophisticated and harder than the basic logic of knowledge.

Our approach, while also involving restrictions, differs in where these restrictions are placed. We put them on the situations we consider, on the initial knowledge and the ways in which additional knowledge can be created (e.g., communication), as well as on the class of epistemic statements we consider. This is done in such a way that the agents, within these limits, are able to compute the *unrestricted* logical notion of knowledge.

Another related approach is that of *knowledge-based programs* by Fagin et al. [61]. It resembles our approach in that epistemic statements are allowed to appear literally in the code of such programs. However, these knowledge-based programs are used exclusively for specification and verification of so-called *standard programs*, which do not contain epistemic statements. These resulting executable programs behave as required by their knowledge-based specification, but they are not assumed to actually "compute [the] knowledge in any way". One may thus say that they use their knowledge *implicitly*, and in a sense our T operator implementation in Section 3.5.1 corresponds to such a standard program.

Our approach in Section 3.5.2 differs from the knowledge-based programs of Fagin et al. [61] in that we allow epistemic statements in the actual executable programs, thus giving them *explicit* access to their knowledge, through concrete algorithms with which they can compute what they may be said to know by a modeler. Reasons for this approach include the following ones. Firstly, we believe that the abstraction level that epistemic statements provide are useful not only for the specifier of a program, but also for the actual programmer, who may not even be known at the time of specification, especially with today's extensible and open platforms. Secondly, programs containing epistemic statements can be easier to maintain than programs that behave equivalently but are formulated on a lower level, and the corresponding knowledge module (such as the one presented in Section 3.5.2) can be updated and verified separately. See Chapter 4 for an illustrating case study in the context of computer games.

It is important to note that, as we have seen in the context of Example 3.5.1. this different viewpoint also makes our notion of knowledge somewhat different from that of Fagin et al. [61]. Their notion of knowledge is defined in terms of possible runs of the whole distributed system. Being aware of the system as a whole and the exact programs of all its processes, the modeler who uses this notion to reason *about* processes would not ascribe knowledge to one process concerning knowledge of another process before ascribing that knowledge to that other process. We, on the other hand, take the subjective viewpoint of an intelligent agent and try to simulate epistemic reasoning *of* such an agent. In contrast to an external modeler, the agent may not be aware of the complete system and the exact internal workings of all other agents. An agent's reasoning about others thus has to be based on assumptions. One reasonable assumption seems to be that other intelligent agents do check and use their knowledge at some point, at the latest

when it becomes relevant—when they need to act on it, which in our case is when they finally pick their strategy. In a reasonably homogeneous society of agents, if an agent can compute another agent's knowledge, then that other agent can certainly also do it himself; in a society with greatly varying reasoning skills, the agents would need to model each other's reasoning capabilities more explicitly. In either case, the distinction whether other agents really possess certain knowledge as soon as they would in principle be able to deduce it, or whether they only possess it at the time when it becomes manifest in their actions, is interesting from a philosophical point of view but impossible to determine for an agent and irrelevant for choosing his own actions.

So knowing that some other agent knows something, strictly speaking, for us rather means concluding that that other agent will be able to figure it out whenever it is relevant. Arguably, this also reflects the best that we as humans can do with an intangible concept such as others' knowledge.

Note that this kind of temporary inconsistency across the agents' knowledge states is of a different nature than that occurring with the eager protocol discussed in Chapter 1, Section 1.5. There, the inconsistency occurs on the level of ascribed knowledge, and it may happen that agents act on inconsistent knowledge, which is why we chose not to adopt that protocol. Here, in contrast, an agent correctly ascribes "potential" knowledge to another agent, which that other agent merely has not computed yet. Under the described assumption that the discrepancy between ascribed and computed knowledge only lasts between two external actions by the respective agent (i.e., the assumption that agents do compute their knowledge by the time it becomes relevant for acting), this inconsistency never manifests itself.

3.6.2 Possible extensions

While the restrictions we have imposed may be so severe that our scenario seems almost trivial, it has still served to explain our approach, and can be used as a starting point for further considerations. In a bottom-up way, our analysis could be extended in a number of ways:

- Allowing players to send information about the preferences of other players that they learned through interaction. We have started to extend the epistemic framework from Chapter 2 into that direction, see [9].
- Allowing other forms of messages, for example, messages containing information that a strategy has been eliminated, or epistemic statements.
- Considering strategic aspects of communication, even if truthfulness is required (should a certain piece of information be sent or not?)
- Considering formation or evolution of interaction structures, given strategic advantages of certain interaction structures over others.

Ultimately, a setting of strategic communication and proactive behavior of querying and telling (possibly false) information may be envisioned, involving agents that plan under consideration of epistemic states, actions and goals. But, as mentioned in Section 3.1.1, for such an open-ended framework much foundational work remains to be done.

Chapter 4

Epistemic reasoning in computer games

4.1 Introduction

Higher-order knowledge, that is, knowledge about (someone else's) knowledge, is important in everyday social interaction.[1] That importance is well-recognized in logic and game theory (insofar as these disciplines extend to everyday life), see, e.g., the work by van Ditmarsch [49] and by Brandenburger [31]. In this chapter, we point out its relevance to computer games that incorporate simulations of social interaction, by which we mean interaction with *artificial agents* and *non-player characters* (NPCs). The most obvious examples of social interaction occur in computer *role-playing games* (RPGs), *interactive fiction* (IF) and *life simulation games* (such as The Sims™), but we also show examples from other genres.

We substantiate one example using what we call *explicit knowledge programming*, which we have proposed in [155] and used in Chapter 3 in this dissertation. It consists in implementing a well-defined restricted epistemic logic within a *knowledge module*, in order to provide epistemic statements on the programming language level and thus give programs *access* to the knowledge they can be said to have from a modeler's point of view.

As a side note, we think that higher-order *desires* as well as beliefs are important, and indeed the interplay between the two. For example, in an interaction between A and B, it can be significant if A believes that B desires that A believe such-and-such. *Belief-desire-intention* (BDI) architectures [122] currently do not accommodate this kind of interactive (i.e., truly multi-agent) phenomena. Since beliefs and knowledge are, from a logical point of view, better understood than desires, we suggest to start by focusing on the former, and that is what we do in

[1] In this chapter we talk interchangeably about *higher-order knowledge* and *higher-order belief*. There are philosophical subtleties at stake here, but they are not relevant for our purposes, for which it is sufficient to stipulate that *knowledge* refers to true belief. Both have in common the *higher-order* aspect, and that is what is crucial. We use the adjective *epistemic* to mean "of (or about) beliefs (or knowledge)".

the rest of this chapter.

4.1.1 Motivation

Human beings are, on the whole, social animals: most of us derive much interest, enjoyment and drama from interacting with other people. One important feature recognized by psychologists is the so-called *theory of mind* [119], that is, the ability to represent and to empathize with the mental states and attitudes of those around us. A recent and popular theory by Baron-Cohen [15] argues that the absence of any theory of mind is what causes such obstacles for people affected with autism spectrum disorders to interact profitably with those around them.

The standard empirical test for a theory of mind, which develops in most people around the age of four, is a test of the subject's ability to represent and "correctly" form higher-order beliefs, that is, to model other people as having their own model of the world in their head, which in turn contains a model of the subject's model. This recursive notion can seem surprising, even paradoxical, however the fact is that most people's behavior is informed by some understanding of higher-order beliefs.

Much of the enjoyment of many multiplayer games actually depends on dealing with, and possibly exploiting, higher-order beliefs or knowledge. Cluedo, Diplomacy and Poker are examples of such *knowledge games*, as van Ditmarsch [49] has called them; each of them illustrates in a different way that it can be enjoyable to reason about the beliefs of others, particularly about the beliefs that they might have about your beliefs, or others' beliefs. If these games were to be played against artificial opponents, with the aim that they be enjoyable, a very natural approach would be to create an opponent who models the player, including the beliefs the player has about the opponent, etc.

So it can be on the one hand "natural", and on the other hand enjoyable, to reason about higher-order knowledge. This is known within the logic community, in fact van Ditmarsch [49] has given detailed formalizations of the epistemic mechanisms involved in real-world games of knowledge and suspicion such as Cluedo. However, so far there has not been much focus on putting such formalizations to work to support a computer simulation of a virtual game world, and in actual computer games these mechanisms have been ignored. We therefore propose to take this line of reasoning seriously, and argue that providing some facility for higher-order reasoning to NPCs would enhance the aim that many games have: being enjoyable simulations of human interaction.

The kinds of interactive situations that depend on higher-order knowledge vary greatly, from trivial to subtle. The following example is so trivial in a normal social context that it almost seems not worth mentioning: if I know that you know something, I won't tell you about it—unless I want you to know that I know. Translated into a computer game context, if Ann knows that Bob knows that the enemy is attacking Bob's base, then she will not tell him that the enemy is

attacking his base, unless she wants him to know that she knows (and is coming to support him). So even this seemingly trivial example illustrates the importance of higher-order reasoning in social interaction.

Increasingly, networked computers and game consoles have led to a rise in the number and scale of multiplayer games. However, we do not believe that this will diminish the need for social artificial intelligence in games, because in a reasonably deep virtual world, there are always "boring" roles, such as quest-givers, merchants, henchmen, and thugs. These are a few of the roles we have in mind when advocating the use of some simple higher-order knowledge reasoning.

We return to some more interesting examples and potential applications later. For now, we hope it is intuitively clear that a virtual world could be enhanced by keeping track of what its virtual inhabitants (could be said to) believe or know, and to provide a programming interface to access these beliefs and knowledge. A programmer can then use these high-level notions in order to program (in this context also called *to script*) behaviors of NPCs, or other relevant parts of the virtual world.

4.1.2 Plan of the chapter

This chapter is built up as follows. In the next Section 4.2, we review our basic approach to realizing these ideas. Then, in Section 4.3, we survey both actual computer games and related research with respect to higher-order reasoning and come to the conclusion that it is largely absent. We then give examples for possible applications of our ideas in Section 4.4, and look at a prototype implementation for one of them more closely in Section 4.5. Finally, Section 4.6 concludes.

4.2 Programming with knowledge

Our approach, proposed in [155], is based around epistemic logic, and since we have used it in Chapter 3 in this dissertation, we recall it here only briefly.

The approach, which we call *explicit knowledge programming*, involves making epistemic formulas available at the level of a programming language, for example as conditions in `if` clauses. These statements are evaluated by a *knowledge module* that processes those events in the world that affect the knowledge state of its inhabitants. It may be implemented in the programming language itself, and thus does not necessarily increase its expressivity. However, the modular approach increases succinctness and flexibility, makes it possible to develop and verify the epistemic processing of the program separately, and essentially gives the NPC scripter a "black box", so that he can directly use the familiar notions of belief or knowledge in his program. At the same time, the knowledge module is firmly grounded in a theoretical framework which gives a formal meaning to these notions.

It is important to note that we are *not* proposing to make available *every* formula that can be built from any arbitrary combination of atoms, connectives and knowledge operators. Rather, for any given concrete application, a certain subclass of formulas needs to be identified as relevant. In this way, the implementation can remain tractable and meet efficiency requirements, which are particularly strict in the case of real-time applications such as computer games.

In the case of simulating *human* agents that we are interested in here, the limits to human cognitive faculties should be taken into account. So for example, it presumably would not make sense to allow as queries to the knowledge module epistemic formulas involving complex iterations, like: "Ann believes that Bob believes that Carl doesn't believe that Ann believes that Derek believes that it's raining" (see Section 4.6 for some more discussion on this). In addition to these human limitations, in the case of NPCs like those mentioned above, for example, a merchant might have a very restricted set of "interests", and only be interested in very specific kinds of knowledge. In any case, the specification of a knowledge module ultimately includes a description of the epistemic formulas that it can evaluate.

In [155] and in Chapter 3 in this dissertation, we were dealing with *distributed* systems, where a knowledge module is instantiated for each process. We showed the implementation to be sound with respect to the formal notion of knowledge ultimately defined on the level of the underlying process calculus, CSP. Even with the simple implementations we obtained, it was desirable to show that it is in some sense "correct". To this end, we used an epistemic logic formalism based on Kripke semantics to model the occurring situations. The correctness of the implementation then proceeds in two steps, which can roughly be stated as follows:

- Argue that a particular model represents faithfully the intuitive situation which we intended to capture.
- Prove that knowledge formulas are evaluated in the same way by process a after the sequence of events σ as they are by agent a in the model after the same sequence of events.

While this idea of formal grounding remains the same, in this chapter we are dealing with a *centralized* framework. Since in a computer game there is a central place where the game world is simulated (and where other models such as a physics engine are already present), only *one* epistemic model needs to be maintained, even in a scenario with several agents. In this sense, this setting is closer in spirit to the typical viewpoint of epistemic logic, which is a central modeler's viewpoint. Rather than maintaining a separate model inside each agent, our central model is fed with the events occurring in the game world, and queried by the agents' programs. Note, however, that this design decision is not fundamental, since central models and internal models of individual agents correspond very

closely to each other [11]. For more discussion on related issues see Section 4.6 and Chapter 7.

One may question in how far a Kripke model formalism as an intermediate step is justified. However, we know of no more philosophically grounded and mathematically robust formalism with which to work in the context of reasoning about higher-order knowledge. (In order to deal with various phenomena like so-called *explicit belief*, or inconsistent beliefs, many other models have been proposed, but these are all essentially refinements or variations of Kripke models—see, e.g., the survey by Meyer and van der Hoek [100], Sections 2.4 to 2.13.)

4.3 Related work

We briefly review the current state of the art with respect to epistemic modeling in computer games, both in existing games and in (academic) research.

4.3.1 Existing games

The state of the art in commercial computer games is not easy to judge, since computer game companies are not very interested in publishing the details of their artificial intelligence (AI) implementations. So if one does not want to rely on enthusiastic slogans from marketing departments, then the best sources of information about computer game AI are private web pages like the one by Rabin [120], where observations and analyses from playing, interview quotations, and possibly the website creator's own knowledge and experience as a game AI programmer are carefully collected and presented. For an extensive overview of online resources, see the compilation by Reynolds [125].

From these resources it becomes evident that epistemic reasoning is definitely not in the focus of existing computer game AI, and we did not find any mention of higher-order reasoning. For example, the highly acclaimed Radiant AI engine is used in the RPG *The Elder Scrolls: Oblivion*™ by Bethesda Softworks [23] to make the game more lifelike. The following quotation is taken from an interview [153] during the testing phase of the game AI:

> One [non-player] character was given [by the testers] a rake and the goal "rake leaves"; another was given a broom and the goal "sweep paths," and this worked smoothly. Then they swapped the items, so that the raker was given a broom and the sweeper was given the rake. In the end, one of them killed the other so he could get the proper item.

Obviously, the characters did not mutually know their interests, or they could not make use of that knowledge. Without seeing the implementation, it is difficult to make suggestions as to how exactly one might build in a knowledge module,

and, as mentioned in the beginning, a whole architecture incorporating beliefs and desires is formally involved; however, the Radiant AI engine does seem to have some kind of goal-oriented behavior rules,[2] and it is very possible that epistemic statements would find a natural place in them.

To us it seems natural that one would use a logic-based approach in order to effectuate epistemic reasoning. Yet references in these directions are scarce. Mäkelä [96] has suggested to use logic for NPC scripting; however, higher-order epistemic reasoning is not considered, and that article is purely programmatic and apparently the ideas have not been pursued further.

The clearest statement promoting the use of epistemic reasoning comes from the famous IF writer Emily Short [140]:

> *Abstract Knowledge.* One of the artificial abilities we might like to give our NPCs, aside from the ability to wander around a map intelligently and carry out complex goals, is the ability to understand what they are told: to keep track of what items of knowledge they have so far, use them to change their plans and goals, and even draw logical inferences from what they've learned.

It is not clear whether this refers to higher-order knowledge, or whether "abstract" is just meant to imply that the implementation should be generic and encapsulated in something like a knowledge module; in any case, the currently existing IF implementation of such ideas by Eve [58] is restricted to pure facts and does not include any reference to the possibility of higher-order knowledge.

An interesting example of a game devoted to small-scale social interaction is Façade by Mateas and Stern [99], a dialog-based graphical version of interactive fiction with a detailed plot that revolves around an evening in a small group of friends. Façade is grounded in academic research, and its authors use intricate techniques to interactively generate the plot, including a behavior language [98]. That language allows to specify the behavior of the NPCs in a very flexible and general way, but does not include facilities for explicitly dealing with knowledge states. We do not suggest that this particular game would necessarily be *improved* if our approach of explicit knowledge programming were adopted, but we do think that it would make a natural addition to this language, and one that should make the programmer's job more straightforward.

4.3.2 Research

Again, where knowledge is considered, the concern seems to be exclusively *domain knowledge*, or knowledge about facts in the game world, as in work by Ponsen

[2] Note, however, that the retail version of the game may be less sophisticated; as one referee pointed out, "the version of Radiant AI that ended up in the game certainly does not have the capabilities that Bethesda aimed for."

et al. [118] and Spronck et al. [143]. A more general approach of using agent programming languages to script NPCs (e.g.,by Leite and Soares [91]) inherits the epistemic reasoning facilities of such languages—which tend to focus on facts. The closest in spirit to higher-order modeling are attempts to detect the general attitude of the human player (for example, aggressive or cautious) and to adjust the game AI accordingly. But we could find no references to explicit higher-order epistemic modeling.

The ScriptEase system by Cutumisu et al. [45] is an academic approach to NPC scripting, which was motivated by the insight that the scripting process needs to be simplified. It provides a graphical interface for generating behaviors of characters in a commercial RPG. However, knowledge statements to steer the behavior are not considered.

An interesting approach, described by da Silva and Vasconcelos [141], uses deontic logic to specify NPC behavior in a rule-based fashion. While epistemic issues are not considered there, a fusion of these two aspects could provide a highly suitable system for scripting believable social agents.

In a similar way, the work by Magnusson and Doherty [95] is a highly promising existing platform for implementing epistemic mechanisms. It consists in a virtual game world that is based on a generic theorem prover for a logic of time and action. Incorporating rules for epistemic reasoning into this framework seems like a very natural extension.

Some literature on higher-order reasoning in multi-agent systems that does *not* focus on computer games is also very relevant. Dragoni et al. [53] study the specific problem of agent *communication*, in which agents weigh costs against expected benefit of communication. The authors point out the importance of using higher-order reasoning, in the form of beliefs about beliefs, when agents make such assessments. Their particular interest is in formal representation of belief *abduction*. We do not consider abductive reasoning here, but we recognize that it is also important in our settings.

We also note that higher-order reasoning is discussed by Yin et al. [160] in the context of a Petri Net method for designing "intelligent team training systems". The authors suggest that using Petri Nets can help to overcome tractability issues in epistemic reasoning. However, they note that communication, an important ingredient in the kind of social interaction we wish to simulate, "is more complicated than Petri Nets can represent". We do not consider the Petri Net formalism further, but if progress is made in this area it could be of relevance.

Several rich platforms for multi-agent programming, including BDI architectures, have been proposed (see, e.g., the recent survey by Bordini et al. [27]). While these often do provide for explicit knowledge operators, they are never higher-order.[3] So these platforms allow for the use of conditions of the form "If the agent knows that the cup is on the table, then ...", but not "If the agent

[3]However, see [34] for a recent step towards higher-order modeling in an agent language.

knows that the other agent knows that ...".

One AI framework that does allow, and is explicitly designed for, handling higher-order beliefs is the formalism of Interactive Partially Observable Markov Decision Processes (I-POMDPs) by Gmytrasiewicz and Doshi [68]. It implements hierarchies of nested probabilistic beliefs of agents about the world and about each other,[4] including Bayesian updating of such belief hierarchies. Since we here focus on symbolic frameworks, we do not pursue this approach further.

4.4 Potential applications

In the following we illustrate how our approach could enhance potential or actually existing computer games.

Knowledge games, as described in Section 4.1.1, are an obvious application area for explicit knowledge programming, if they are to be implemented in a computer game version with intelligent computer-controlled opponents. To what extent a knowledge module here should implement the complete theoretical epistemic model, depends on performance considerations as well as a good balance to make the opponents challenging yet not super-human. We do not focus on such games here.

As mentioned before, RPGs and life simulation games naturally try to simulate realistic social interaction. Therefore, any real-life social interaction situation involving knowledge can be viewed as potential application, if one wants to script the required behavior rules into a virtual world.

For example, in real life the following behaviors might be observed in situations where Ann would get an advantage from lying to Bob:

- if she knows that he doesn't know the truth
  then she might indeed lie;
- if she knows that he does know the truth
  then she usually won't lie;
- if she doesn't know whether he knows the truth
  then her decision may depend on other circumstances.

4.4.1 Catching the Thief

To be a bit more concrete and give an example set in an actually existing computer game, we consider Thief: The Dark Project™ by Eidos Interactive [56]. This game involves an interesting special case of "social interaction": the player is a thief, and as such, the best tactic most of the time involves remaining undetected and avoiding confrontation.

[4] The framework is similar to (a finite version of) type spaces known from game theory.

```
if B(g, t_present):
    if B(g, not B(t, g_present)):
        g.ambush(t)
    elif B(g, not B(t, B(g, t_present))):
        g.attack_or_alarm_inconspicuously()
    else:
        if not alarm_active:
            g.alarm_quickly()
        else:
            g.attack(t)
```

Listing 4.1: Pseudo-code for a guard in Thief: The Dark Project. `B(x,y)` stands for "x believes y", `g` stands for guard and `t` for thief.

While there are more obvious and ubiquitous applications for higher-order reasoning generally in RPGs and life simulation games, in this somewhat unusual social interaction setting it is certainly crucial to keep track of who knows (or believes) what about whose presence, both for the player and for plausible and challenging NPCs. Essentially, the thief player exploits the, possibly false, beliefs of the guard regarding the player's presence. Due to the simplicity of the guard's control program in the commercial game, the guard's beliefs are in practice perceived as either believing that the thief is, or may be, near (having seen him or become suspicious in some other way), or not.

We conjecture that the entertainment value of a typical Thief scenario would be enhanced by a guard that acts not only depending on his own beliefs about the facts in the world, but also depending on what he believes the thief believes, including what he believes the thief believes he believes.

Imagine an NPC that embodies a guard, hostile to the thief. Among his behavior rules could be the following: If the guard believes that the thief is present, but the guard also believes that the thief does not believe that the guard is present, then the guard tries to ambush the thief. Under other conditions, the guard may instead act inconspicuously and attack by surprise, or, if all else fails, attack the thief openly. The pseudo-code in Listing 4.1 specifies several such rules, depending on higher-order beliefs. How such beliefs can come about may vary: for example, by causing or hearing noise, seeing the other one from behind, or facing each other. When scripting these behaviors, the programmer need not worry about this—it is the job of the knowledge module to take care of maintaining and updating the knowledge states, taking into account whichever events occur in the virtual world.

Once it is established exactly what kinds of epistemic formulas need to be used, and what kinds of events can take place in the virtual world, one can go about designing and implementing the knowledge module for the specific application in

question. It depends on the class of formulas and on the events how straightforward this implementation will be; but it may well turn out to be very efficient.

In Section 4.5 we return to this scenario in some more detail and describe a simple example implementation of a corresponding knowledge module.

4.4.2 Adding credence to Assassin's Creed

Assassin's Creed™ by Ubisoft [146] presents another crisp case for higher-order knowledge or beliefs. Altair (the player character) frequently has the optional objective to Save a Citizen. To do this, Altair kills the guards that are accosting the citizen. Vengeful guards nearby comment on the dead allies, which notifies Altair that they know a murder has occurred. Because Altair is present, the guards infallibly assume that Altair is the killer.

For higher-order reasoning, suppose that a guard partitions the crowd into bystanders and suspects by observing who is armed. The guard observes speech and body language of each bystander (technically voice-over callouts and animation states) to infer beliefs of the bystander. As soon as the guard observes a bystander who is responding to a murderer, the guard assumes the bystander's target is the killer. Observing the body language of the suspect (possibly Altair), the guard may be able to infer whether the suspect believes that the guard is onto him.

Listing 4.2 shows pseudo-code of this higher-order analysis. Admittedly, such a proposal would require further design, but the example illustrates reasoning yet somewhat gullible guards, which a crafty assassin may delight in deceiving.

```
if B(g, murder_happened):
    suspects = { x | B(g, is_armed(x)) }
    bystanders = { x | B(g, not is_armed(x)) }
    for b in bystanders:
        if B(g, B(b, murder_happened)):
            for s in suspects:
                if B(g, B(b, is_killer(s))):
                    if B(g, B(s, B(g, is_killer(s)))):
                        g.shout(is_killer(s))
                        g.attack(s)
                    else:
                        g.whisper(is_killer(s))
                        g.ambush(s)
                    break
```

Listing 4.2: Pseudo-code for Assassin's Creed. Again, `B(x,y)` stands for "x believes y" and `g` stands for guard.

4.5 Implementation study for Thief

In this section, we substantiate our discussion from Section 4.4.1 by describing an actual implementation of the knowledge module used in the pseudo-code in Listing 4.1.

4.5.1 Knowledge module

We consider various ways in which relevant beliefs can come about: by causing or hearing noise, seeing the other one from behind, or facing each other.

We want to emphasize again that the NPC scripter need not worry about the details, he simply uses the familiar concept of belief in order to express the rules on a high level. It is the job of the *knowledge module* to take care of maintaining and updating the agents' knowledge,[5] taking into account whichever events occur in the virtual world. As an agent's control script is executed, the knowledge module is queried and determines the truth value of any given formula.

We assume that the scene starts with thief and guard present and no events having occurred, and that agents cannot enter or leave. Put differently, the guard will not leave the scenario, and if the player does, the scenario ends. Note that this assumption is not inherent in our approach and serves mostly to avoid unnecessary complications. In the context of a computer game it is not too unnatural: game worlds often are simulated per scene, and a scene starts and ends whenever the player enters or leaves.

In our scenario, we consider the following events:

- b_t, b_g: The thief, respectively the guard, sees the other one from behind.
- n_t, n_g: The thief, respectively the guard, makes some noise. We limit this to "relevant" events in the sense that, e.g., n_t can only occur if the thief is present.
- f: Thief and guard see each other face to face.

The epistemic effects of these events intuitively are as follows:

- b_t: The thief believes that the guard is present; the guard never assumes this event to occur, so this event does not affect the guard's beliefs.
- n_t: The guard believes a thief is present; the thief believes that, if a guard is present, that guard believes that the thief is present.
- f: Thief and guard commonly believe that both are present.

[5] In our case beliefs rather than knowledge, but we stick with the name "knowledge module".

The effects of b_g and n_g are analogous, and all these effects are commonly believed. For now, we assume that events are not forgotten. Note that a consequence of all this is that after both n_g and n_t have occurred, guard and thief commonly believe that both are present.

To model this situation formally, we use a modal logic model with history-based semantics [116, 60]. We introduce two **propositions**, p_t and p_g, with the reading that the thief, respectively the guard, is present. A **valuation** is a function that assigns either true or false to each of these propositions, and can be denoted as a set V containing those propositions that are to be assigned true.

The set of **events**, as above, is $E = \{b_t, b_g, n_t, n_g, f\}$. A **history** H is a sequence of events. For a history H, by $Ag(H)$ we denote the set of agents involved in the events in H, i.e., $Ag(H) \subseteq \{t, g\}$, where naturally both t and g are involved in any of $\{b_t, b_g, f\}$, and only t (respectively g) is involved in n_t (respectively n_g). We also write $H - e$ for any $e \in E$ to denote the history obtained by removing all occurrences of e from H. The empty history is denoted by ϵ.

A pair (V, H) is a **state** (or **possible world**) if and only if $Ag(H) \subseteq V$. So the described initial situation is represented by the state $(\{p_t, p_g\}, \epsilon)$.

We now define **accessibility relations** $\dashrightarrow_t$ and $\dashrightarrow_g$ between states in our history-based model. Intuitively, $(V, H) \dashrightarrow_t (V', H')$ means that in state (V, H), t considers it possible that the state is (V', H'). According to the intuitions described above, we define these relations as follows:

$$(V, H) \dashrightarrow_t (V', H') \text{ iff } \begin{cases} t \notin V' \text{ and } H' - n_g = \epsilon & \text{if } t \notin V \\ t \in V' \text{ and } H' = H - b_g & \text{if } t \in V. \end{cases}$$

Note that in the first of these two cases, n_g is the only event which can occur in H', so the condition boils down to saying that H' may be any sequence of n_g. The second case captures the intuition that t does not assume the possibility that he is being seen from behind. The relation $\dashrightarrow_g$ is defined analogously.

We consider the doxastic language consisting of formulas of the following form:

$$\varphi ::= p_t \mid p_g \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid B_t\varphi \mid B_g\varphi.$$

The semantics is standard relational modal semantics, see [24]. Note that here we do not impose further restrictions on the doxastic language, since our scenario is simple enough for the knowledge module to evaluate all formulas efficiently.

If we merge all doxastically equivalent states, that is, all states that make the same formulas true, our model can be depicted as in Figure 4.1.[6]

Note that, in contrast to our starting point from history-based semantics, this representation is static in the sense that it does not grow (or shrink) over time, as the world evolves and events occur. It is straightforwardly represented in any

[6]This is the so-called *bisimulation contraction* of our original model. Note that in our case, if (V, H) and (V', H') are doxastically equivalent, so are (V, He) and $(V', H'e)$ for any event e.

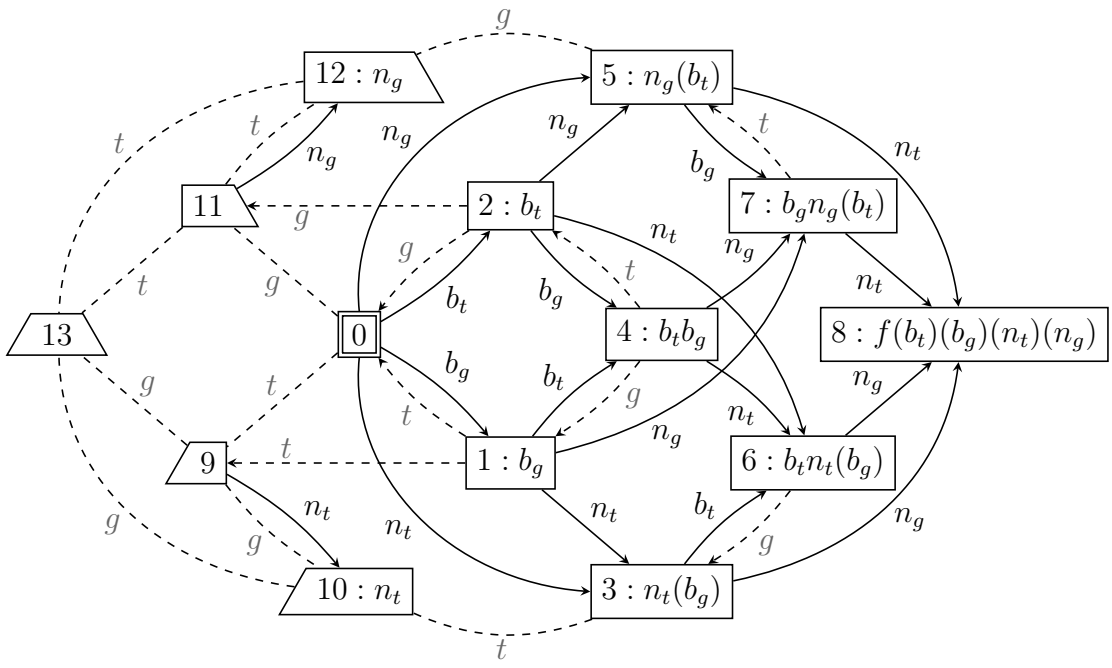


Figure 4.1: Representation of our doxastic model. The numbered nodes represent states and are annotated with one of the equivalent possibilities of what events have (optionally) taken place. Boxed states have $V = \{p_t, p_g\}$, a missing left corner means $p_g \notin V$, and a missing right corner means $p_t \notin V$. State 0 represents the initial situation. Solid arrows with black labels show event transitions, dashed arrows and edges (the latter corresponding to bidirectional arrows) with gray labels show accessibility relations. The following are omitted for clarity: reflexive transitions for each optional event at each state; f-transitions from each boxed state to 8; and reflexive accessibilities, except for g in states 2 and 6 and for t in states 1 and 7.

programming language (our implementation uses Python), and the knowledge module simply needs to keep track of which one is the current state.

Finally, note that while agents can have false beliefs, there is no need for elaborate belief revision mechanisms in our simple scenario, since there is no way for the agents to notice their false beliefs.

4.5.2 Expected impact on gameplay

In the original game, a common tactic for the thief is to shoot an arrow against a wall, in order to cause a noise which will attract the guard (who has a behavior rule along the lines of "if I hear a noise, I go there and look around"). It is questionable how innocent such a noise is, and an actually reasoning guard would

probably rather conclude that it is high time to ring the alarm instead of sniffing around in the bushes. While this was likely a conscious design decision to give the player an easy and obvious way to outwit that guard, we think that our belief-based approach provides a more flexible solution.

Instead of rigidly connecting events to resulting behaviors, beliefs in our approach act as an abstraction layer. The behaviors are defined depending on the beliefs, and these beliefs are modeled independently.

This approach removes from the scripter the burden of having to decide exactly which events cause what, and facilitates adjustments and more complex dependencies. For example, in our scenario a more clearly innocent noise such as breaking a twig will induce the guard to believe that there is a thief (who accidentally caused that noise), which will then trigger the behavior rule that says to ambush him.

We expect that our modest belief engine for the scenario we have described encourages the original game's tactics of misleading a guard. It also encourages new tactics: The thief may play stupid and let himself be seen from behind by the guard, who then again thinks he can do something sneakier than ringing the alarm.

We also expect that our scenario encourages trepidation. A guard who has actually noticed the thief without being noticed himself will act inconspicuously while preparing a counter-attack. The blackjack is the weapon for subduing, which has a short reach. A guard that pretends to not have noticed, will swing around and stab with his sword first before the player's blackjack is in range.

Overall, we expect that the player enjoyment and interest will increase. Whether or not these expectations prove true remains to be seen in an experimental evaluation, which we are planning to conduct in the future.

4.6 Conclusions

We have described a modular approach to adding (higher-order) knowledge operators to scripting languages for NPCs in various kinds of games. We have argued and given examples to show that this systematic approach would help script plausible or entertaining simulations that involve social interaction, where that last term has a broad interpretation. We have surveyed existing work, in industry and in academic research, and found that higher-order knowledge has not so far been discussed or implemented in the context of computer games. Finally, we have presented a simple implementation for one of the described example scenarios.

4.6.1 Explicit knowledge programming

The knowledge module we presented implements a centralized version of the explicit knowledge programming approach we proposed in [155] and used in

Chapter 3. The fact that it is centralized makes it simpler from the outset than a truly distributed setting: In a distributed setting, an agent would have to deduce locally whether a given formula follows from his observations, that is, whether it holds in all models which are compatible with the local observation and might thus reflect the actual situation; in our centralized setting, the game engine has the model of the actual situation available and only needs to check whether it satisfies the given formula. This together with the simplicity of our scenario removes the necessity of finding restrictions of the doxastic language or other optimizations: about any conceivable model and implementation would be trivially tractable.

However, in all its simplicity our scenario still shows the advantages of the modular approach introducing knowledge or beliefs as an abstraction layer and separating the epistemic model from the agent's control program. The knowledge module can be refined independently to cope with new events or extensions, such as probabilistic beliefs, with no need to change the behavior scripts. One extension that is easy to incorporate into our knowledge module is decaying events, or unstable states: If the guard makes some noise like whistling, he might have forgotten about that when he 10 minutes later sees the thief from behind, and therefore *not* conclude that the thief believes the guard is present. This could be modeled by a spontaneous transition from state 5 to 2 after some time.

While one could implement some ad-hoc tracking of events in the agent control program, this is error-prone and difficult to maintain, as already in our small test scenario the exact dependencies on events are getting confusing: What does "if n_t or b_g have occurred, but neither n_g nor f have, then..." mean? This quickly gets even more confusing in larger scenarios—our approach certainly scales better conceptually.

A critical issue with respect to scaling, however, is the possible explosion of the number of states in the model. Mechanisms to generate the model only locally as needed, and strategies for expanding, shrinking, or swapping out unused parts of the model may be necessary. Techniques from the area of model checking [41] will be useful for compactly representing and efficiently processing models.

4.6.2 Alternatives and extensions

The advantages of the representation we have used here is that it is straightforwardly stored using very simple data structures, and that it is static and pre-computed, so there is no expense at run-time for maintaining the model.

A very interesting alternative is building on *Dynamic Epistemic Logic* (DEL, see [52] for a recent textbook). Instead of one model representing all possible ways in which the world may evolve, one would start with a model representing only the initial situation. Events are represented as so-called *update models*, which are applied to the current model as the events occur, transforming it to reflect the resulting situation. See Chapter 7 for some discussion on this approach.

An interesting suggestion that would also simplify reasoning is to consider so-called *interaction axioms* between the belief modalities of the players. For example, Lomuscio and Ryan [93] show that (2WD) is valid on two-player *hypercubes*, i.e., models that are based on the Cartesian product of an interpreted system's state space:

$$\Diamond_1 \Box_2 p \implies \Box_2 \Diamond_1 p. \tag{2WD}$$

This can be read as "if agent 1 considers it possible that agent 2 knows p, then agent 2 knows that agent 1 considers p possible.". In a different context, van Ditmarsch and Labuschagne [51] consider interaction axioms that characterize different theories of mind, including "autistic" and "deranged". It is an interesting question whether interaction axioms might exist that capture agents that are *fun* to interact with.

Another point worth noting is that we do not distinguish between *knowledge* and *true belief*. There are many philosophical discussions concerning differences between these two notions, and formally they are generally taken to be different as well. Beliefs should really be *revisable*, for example, which is something we have not considered here. A possible future direction of research would be to use models and languages that take both belief and knowledge into account, for example along the lines of Shoham and Leyton-Brown [139], Chapter 13. Some actions would then generate knowledge, while some would (only) generate belief. This corresponds to the distinction van Benthem [19] draws between "hard" and "soft" information. For example, if you *see* something then you might be said to know it, but if somebody you do not entirely trust tells you something then you might only believe it.

Finally, it may be desirable to get rid of manually designed behavior rules altogether, and let the agents act purely based on their beliefs and some abstractly defined goals. A very long-term vision is then to have an artificial guard that by himself adopts behaviors similar to the ones we described, or to the tactics we conjectured for the human thief in Section 4.5.2.

4.6.3 Cognitive considerations

Rather than working out "bottom-up" what class of epistemic statements are needed for a certain setting or behavior, one may consider a more generic "top-down" approach. Given that we want to simulate human social interaction, the general question is: What class of formulas can humans be said to evaluate in everyday life, consciously or not?

Some results from experimental game theory about levels of strategic thinking, e.g., by Camerer [36], can be interpreted as being relevant to this question. However, these experiments do not focus on everyday social interaction. For example, the Beauty Contest game mentioned there might invoke conscious and explicit reasoning about the other agents, while we believe that in real-life social

interaction, through years of experience, the requisite higher-order reasoning processes may also occur on a more intuitive and reflexive level.

Furthermore, the experimental designs are in general not specifically concerned with knowledge, so that at best the results can give us hints about the nesting depth of knowledge operators. Clearly other criteria might define the class of knowledge formulas that are of relevance in social interactions.

From the real-life example in the beginning of Section 4.4 it is clear that formulas like

$$K_a \neg K_b p$$
$$K_a K_b p$$
$$\neg(K_a K_b p \vee K_a \neg K_b p)$$

matter. However, that does not necessarily hold for all formulas with knowledge nesting depth 2. Also, human reasoning capabilities may not be monotonic with respect to this complexity measure. For example, for a special concept like common knowledge, which in theory involves infinite depth, we may want to assume that humans are able to cope with it, while this does not hold for "intermediate" depths like 10000.

The main issue thus remains: How can we define this class of relevant formulas?

Results from experiments by Verbrugge and Mol [147] suggest that subjects use first-order *theory of mind* (beliefs about others' beliefs), but not "all kinds of reasoning possible and useful in this context". This supports the claim that the depth of knowledge operators is not the only relevant criterion. It is further reported that some subjects use second-order theory of mind, which corresponds to third-order epistemic formulas.

While it is important to look at such questions in actual experiments, thought experiments or observations from real life can also make their contribution. Work by Parikh [112] is an excellent source for enlightening examples examining what kinds of reasoning processes are going on in real life. They can be convincing enough to remove the need for abstract and reproducible lab conditions, for the benefit of being set in more natural environments, where human reasoning capabilities possibly profit from experience and training in specific social situations.

4.6.4 Final words

As mentioned earlier, in computer games the goal of incorporating epistemic reasoning may not necessarily be verisimilitude, because their ultimate objective is enjoyment and entertainment. For analogy, car racing games such as Burnout Paradise™ by Electronic Arts Inc. [57] obviously employ rigid body dynamics for traction and collision in a way that veers from verisimilitude and toward excitement. Game physics engines have evolved from algorithms originally developed for simulating physical bodies into frameworks for animating virtual toys. We hope

that an epistemics engine comprised of algorithms originally developed to simulate beliefs about the beliefs of other agents may evolve into a toolkit for cleverly (mis)informing the minds of virtual playmates.

Chapter 5
Coalition formation: A generic approach

5.1 Introduction

5.1.1 Approach

Coalition formation has been a research topic of continuing interest in the area of cooperative game theory. It has been analyzed from several points of view, starting with Aumann and Drèze [13], who considered the static situation of coalitional games in the presence of a given coalition structure (i.e., a partition of the players).

In this chapter we consider the perennial question "how do coalitions form?" by proposing a simple answer: "by means of merges and splits". This brings us to the study of a natural problem, namely under what assumptions the outcomes of arbitrary sequences of merges and splits are unique.

These considerations yield an abstract approach to coalition formation that focuses on partial comparison relations between partitions of a group of players and simple merge and split rules. These rules transform partitions of a group of players under the condition that the resulting partition is preferred. By identifying conditions under which every iteration of these rules yields a unique partition we are brought to a natural notion of a stable partition.

This approach is parametrized by a generic comparison relation. The obtained results depend only on a few simple properties, namely irreflexivity, transitivity and monotonicity, and do not require any specific model of coalitional games.

In the case of coalitional TU-games (we recall the definition in Section 5.3), the comparison relations induced by various well-known orders on sequences of reals, such as leximin or Nash order, satisfy the required properties. As a consequence our results apply to the resulting comparison relations and coalitional TU-games. We also explain how our results apply to hedonic games (games in which each player has a preference relation on the sets of players that include him) and exchange economy games.

This approach to coalition formation is indirectly inspired by the theory of abstract reduction systems (ARS, see, e.g., [145]), one of the aims of which is a study of conditions that guarantee a unique outcome of rule iterations. Apt [6] exemplified another benefit of relying on ARS by using a specific result, called Newman's Lemma, to provide uniform proofs of order independence for various strategy elimination procedures for finite strategic games.

5.1.2 Related work

Because of this different starting point underpinning our approach, it is difficult to compare it to the vast literature on the subject of coalition formation. Still, a number of papers should be mentioned even though their results have no immediate bearing on ours.

In particular, rules that modify coalitions are considered by Yi [159] in the presence of externalities and by Ray and Vohra [124] in the presence of binding agreements. In both papers two-stage games are analyzed. In the first stage coalitions form and in the second stage the players engage in a non-cooperative game given the emerged coalition structure. In this context the question of stability of the coalition structure is then analyzed.

The question of (appropriately defined) stable coalition structures often focused on hedonic games. Bogomolnaia and Jackson [26] considered four forms of stability in such games: core, Nash, individual and contractually individual stability. Each alternative captures the idea that no player, respectively, no group of players has an incentive to change the existing coalition structure. The problem of existence of (core, Nash, individually and contractually individually) stable coalitions was considered in this and other references, for example [142] and [35].

Recently, Bloch and Jackson [25] compared various notions of stability and equilibria in network formation games. These are games in which the players may be involved in a network relationship that, as a graph, may evolve. Other interaction structures which players can form were considered by Demange [47], who studied formation of hierarchies, and by Macho-Stadler et al. [94], who allowed only bilateral agreements that follow a specific protocol.

Early research on the subject of coalition formation has been discussed by Greenberg [71]. More recently, various aspects of coalition formation have been discussed in a collection by Demange and Wooders [48] and in a survey by Marini [97].

The approach we take here was studied by Apt and Radzik [7] in a limited setting of coalitional TU-games and the comparison relation induced by the utilitarian order.

5.1.3 Plan of the chapter

This chapter is organized as follows. In Section 5.2, we set the stage by introducing an abstract comparison relation between partitions of a group of players and the corresponding merge and split rules that act on such partitions. Then, in Section 5.3, we discuss a number of natural comparison relations on partitions within the context of coalitional TU-games and in Section 5.4 by using arbitrary value functions for such games.

Next, in Section 5.5, we introduce and study a parametrized concept of a stable partition and in Section 5.6 relate it to the merge and split rules. Finally, in Section 5.7 we explain how to apply the obtained results to specific coalitional games, including TU-games, hedonic games and exchange economy games, and in Section 5.8 we summarize our approach.

5.2 Comparing and transforming collections

Let $N = \{1, \dots, n\}$ be a fixed set of players called the **grand coalition**. Non-empty subsets of N are called **coalitions**. A **collection** (in the grand coalition N) is any family $C := \{C_1, \dots, C_\ell\}$ of mutually disjoint coalitions, and ℓ is called its *size*. If additionally $\bigcup_{j=1}^{\ell} C_j = N$, the collection C is called a **partition** of N. For $C = \{C_1, \dots, C_k\}$, we define $\bigcup C := \bigcup_{i=1}^{k} C_i$.

In this chapter we are interested in comparing collections. In what follows we only compare collections A and B that are partitions of the same set, i.e., such that $\bigcup A = \bigcup B$. Intuitively, assuming a comparison relation $\rhd$, $A \rhd B$ means that the way A partitions K, where $K = \bigcup A = \bigcup B$, is preferable to the way B partitions K.

To keep the presentation uniform we only assume that the relation $\rhd$ is *irreflexive*, i.e., for no collection A, $A \rhd A$ holds, *transitive*, i.e., for all collections A, B, C with $\bigcup A = \bigcup B = \bigcup C$, $A \rhd B$ and $B \rhd C$ imply $A \rhd C$, and that $\rhd$ is *monotonic* in the following two senses: for all collections A, B, C, D with $\bigcup A = \bigcup B$, $\bigcup C = \bigcup D$, and $\bigcup A \cap \bigcup C = \emptyset$,

$$A \rhd B \text{ and } C \rhd D \text{ imply } A \cup C \rhd B \cup D, \tag{m1}$$

and for all collections A, B, C with $\bigcup A = \bigcup B$ and $\bigcup A \cap \bigcup C = \emptyset$,

$$A \rhd B \text{ implies } A \cup C \rhd B \cup C. \tag{m2}$$

The role of monotonicity will become clear in Section 5.5, though property (m2) will already be of use in this section.

5.2.1. Definition. By a **comparison relation** we mean an irreflexive and transitive relation on collections that satisfies the conditions (m1) and (m2). □

A comparison relation $\rhd$ is used only to compare partitions of the *same* set of players. So partitions of different sets of players are incomparable w.r.t. $\rhd$, that is, no comparison relation is linear. This leads to a more restricted form of linearity, defined as follows. We call a comparison relation $\rhd$ *semi-linear* if for all collections A, B with $\bigcup A = \bigcup B$, either $A \rhd B$ or $B \rhd A$.

In what follows we study coalition formation by focusing on the following two rules that allow us to transform partitions of the grand coalition:

merge: $\{T_1, \ldots, T_k\} \cup P \to \{\bigcup_{j=1}^{k} T_j\} \cup P$, where $\{\bigcup_{j=1}^{k} T_j\} \rhd \{T_1, \ldots, T_k\}$

split: $\{\bigcup_{j=1}^{k} T_j\} \cup P \to \{T_1, \ldots, T_k\} \cup P$, where $\{T_1, \ldots, T_k\} \rhd \{\bigcup_{j=1}^{k} T_j\}$

Note that both rules use the $\rhd$ comparison relation "locally", by focusing on the coalitions that take part and result from the merge, respectively split. In this chapter we are interested in finding conditions that guarantee that arbitrary sequences of these two rules yield the same outcome. So, once these conditions hold, a specific *preferred* partition exists such that any initial partition can be transformed into it by applying the merge and split rules in an arbitrary order.

To start with, note that the termination of the iterations of these two rules is guaranteed.

5.2.2. Note. Suppose that $\rhd$ is a comparison relation. Then every iteration of the merge and split rules terminates.

Proof. Every iteration of these two rules produces by (m2) a sequence of partitions $P_1, P_2, \ldots$ with $P_{i+1} \rhd P_i$ for all $i \geq 1$. But the number of different partitions is finite. So by transitivity and irreflexivity of $\rhd$ such a sequence has to be finite. □

The analysis of the conditions guaranteeing the unique outcome of such iterations is now deferred to Section 5.6.

5.3 TU-games

To properly motivate the subsequent considerations and to clarify the status of the monotonicity conditions we now introduce some natural comparison relations on collections for coalitional TU-games. A **(coalitional) TU-game** is a pair (N, v), where $N := \{1, \ldots, n\}$ and the *value function* v is a function from the powerset of N to the set of non-negative reals[1] such that $v(\emptyset) = 0$.

For a coalitional TU-game (N, v) the comparison relations on collections are induced in a canonic way from the corresponding comparison relations on multisets of reals by stipulating that for collections A and B,

[1]The assumption that the values of v are non-negative is non-standard and is needed only to accomodate for the Nash order, defined below.

$$A \rhd B \text{ iff } v(A) \rhd v(B), \tag{5.1}$$

where for a collection $A := \{A_1, \ldots, A_m\}$, we let $v(A) := \dot{\{}v(A_1), \ldots, v(A_m)\dot{\}}$, denoting multisets in dotted braces.

So first we introduce the appropriate relations on multisets of non-negative reals. The corresponding definition of monotonicity for such a relation $\rhd$ is that, for all multisets a, b, c, d of reals,

$$a \rhd b \text{ and } c \rhd d \text{ imply } a \dot{\cup} c \rhd b \dot{\cup} d$$

and

$$a \rhd b \text{ implies } a \dot{\cup} c \rhd b \dot{\cup} c,$$

where $\dot{\cup}$ denotes multiset union.

Given two sequences $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_n)$ of real numbers, we define the (extended) *lexicographic order* on them by putting

$$(a_1, \ldots, a_m) >_{\text{lex}} (b_1, \ldots, b_n)$$

iff

$$\exists i \leq min(m, n)\, (a_i > b_i \wedge \forall j < i\, a_j = b_j)$$

or

$$\forall i \leq min(m, n)\, a_i = b_i \wedge m > n.$$

Note that in this order we compare sequences of possibly different length. We have, for example, $(1, 1, 1, 0) >_{\text{lex}} (1, 1, 0)$ and $(1, 1, 0) >_{\text{lex}} (1, 1)$. It is straightforward to check that it is a linear order.

We assume below that $a = \dot{\{}a_1, \ldots, a_m\dot{\}}$ and $b = \dot{\{}b_1, \ldots, b_n\dot{\}}$, and that a^* is a sequence of the elements of a in decreasing order, and define

- the *utilitarian* order:
  $a \succ_{\text{ut}} b$ iff $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$,
- the *Nash* order:
  $a \succ_{\text{Nash}} b$ iff $\prod_{i=1}^{m} a_i > \prod_{j=1}^{n} b_j$,
- the *leximin* order:
  $a \succ_{\text{lex}} b$ iff $a^* >_{\text{lex}} b^*$.

Moulin [103] considered these relations for sequences of the same length. For such sequences, we discuss two other natural orders in Section 5.4. The intuition behind the Nash order is that when the sum $\sum_{i=1}^{m} a_i$ is fixed, the product $\prod_{i=1}^{m} a_i$ is largest when all a_is are equal. So in a sense the Nash order favours an equal distribution.

The above relations are clearly irreflexive and transitive. Additionally the following holds.

5.3.1. NOTE. The above three relations are all monotonic both in sense (m1) and (m2).

Proof. The only relation for which the claim is not immediate is $\succ_{\text{lex}}$. We only prove (m1) for $\succ_{\text{lex}}$; the remaining proof is analogous.

Let arbitrary multisets of non-negative reals a, b, c, d be given. We define, with e denoting any sequence or multiset of non-negative reals,

$$\begin{aligned} len(e) &:= \text{the number of elements in } e,\\ \mu &:= (a \dot\cup b \dot\cup c \dot\cup d)^* \text{ with all duplicates removed},\\ \nu(x,e) &:= \text{the number of occurrences of } x \text{ in } e,\\ \beta &:= 1 + \max_{k=1}^{len(\mu)}\{\nu(\mu_k, a \dot\cup b \dot\cup c \dot\cup d)\},\\ \#(e) &:= \sum_{k=1}^{len(\mu)} \nu(\mu_k, e)\cdot\beta^{-k}. \end{aligned}$$

So μ is the sequence of all distinct reals used in $a \dot\cup b \dot\cup c \dot\cup d$, arranged in a decreasing order. The function $\#(\cdot)$ injectively maps a multiset e to a real number y in such a way that in the floating point representation of y with base β, the kth digit after the point equals the number of occurrences of the kth biggest number μ_k in e. The base β is chosen in such a way that even if e is the union of some of the given multisets, the number $\nu(x,e)$ of occurrences of x in e never exceeds $\beta - 1$. Therefore, the following sequence of implications holds:

$$\begin{aligned} a^* >_{\text{lex}} b^* \text{ and } c^* >_{\text{lex}} d^* &\Rightarrow \#(a) > \#(b) \text{ and } \#(c) > \#(d)\\ &\Rightarrow \#(a) + \#(c) > \#(b) + \#(d)\\ &\Rightarrow \#(a \dot\cup c) > \#(b \dot\cup d)\\ &\Rightarrow (a \dot\cup c)^* >_{\text{lex}} (b \dot\cup d)^* \end{aligned}$$

□

Consequently, the corresponding three relations on collections induced by (5.1) are all semi-linear comparison relations.

As a natural example of an irreflexive and transitive relation on multisets of reals that does not satisfy the monotonicity condition (m1) consider $\succ_{\text{av}}$ defined by

$$a \succ_{\text{av}} b \text{ iff } (\textstyle\sum_{i=1}^m a_i)/m > (\sum_{j=1}^n b_j)/n.$$

Note that for

$$a := \dot{\{}3\dot{\}}, b := \dot{\{}2,2,2,2\dot{\}}, c := \dot{\{}1,1,1,1\dot{\}}, d := \dot{\{}0\dot{\}}$$

we have both $a \succ_{\text{av}} b$ and $c \succ_{\text{av}} d$ but not $a \dot\cup c \succ_{\text{av}} b \dot\cup d$ since $\dot\{3,1,1,1,1\} \succ_{\text{av}} \dot\{2,2,2,2,0\}$ does not hold.

Further, the following natural irreflexive and transitive relations on multisets of reals do not satisfy the monotonicity condition (m2):

- the *elitist* order:

  $a \succ_{\text{el}} b$ iff $max(a) > max(b)$,

- the *egalitarian* order:

  $a \succ_{\text{eg}} b$ iff $min(a) > min(b)$,

Indeed, we have both $\dot\{2\} \succ_{\text{el}} \dot\{1\}$ and $\dot\{2\} \succ_{\text{eg}} \dot\{1\}$, but neither $\dot\{3,2\} \succ_{\text{el}} \dot\{3,2\}$ nor $\dot\{1,0\} \succ_{\text{eg}} \dot\{1,0\}$ holds.

5.4 Individual values

In the previous section we defined the comparison relations in the context of TU-games by comparing the values (yielded by the v function) of whole coalitions. Alternatively, we could compare payoffs to individual players. The idea is that in the end, the value secured by a coalition may have to be distributed to its members, and this final payoff to a player may determine his preferences.

To formalize this approach we need the notion of an *individual value function* ϕ that, given the v function of a TU-game and a coalition A, assigns to each player $i \in A$ a real value $\phi_i^v(A)$. We assume that ϕ is *efficient*, i.e., that it exactly distributes the coalition's value to its members:

$$\sum_{i \in A} \phi_i^v(A) = v(A).$$

For a collection $C := \{C_1, \ldots, C_k\}$, we put

$$\phi^v(C) := \dot\{\phi_i^v(A) \mid A \in C, i \in A\}.$$

Given two collections $C = \{C_1, \ldots, C_k\}$ and $C' = \{C'_1, \ldots, C'_\ell\}$ with $\bigcup C = \bigcup C'$, the comparison relations now compare $\phi^v(C)$ and $\phi^v(C')$, which are multisets of $|\bigcup C|$ real numbers, one number for each player. In this way it is guaranteed that the comparison relations are *anonymous* in the sense that the names of the players do not play a role.

In this section, to distinguish between comparison relations defined only by means of v and those defined using both v and ϕ, we denote the former by $\rhd_v$ and the latter by $\rhd_\phi$.

We now examine how these two different approaches for defining comparison relations relate. To this end, we will clarify when they coincide, i.e., when given a comparison relation defined in one way, we can also obtain it using the other way, and when they are unrelated. We begin by formalizing the concept of anonymity.

5.4.1. Definition. Assume a coalitional TU-game (N, v).

- An individual value function ϕ is *anonymous* if for all v, permutations π of N, $i \in N$, and $A \subseteq N$

$$\phi_i^v(A) = \phi_{\pi(i)}^{v\circ\pi^{-1}}(\pi(A)).$$

- v is *anonymous* if for all permutations π of N and $A \subseteq N$

$$v(A) = v(\pi(A)).$$

□

Note that for all A we have $(v\circ\pi^{-1})(\pi(A)) = v(A)$. Intuitively, ϕ is anonymous if it does not depend on the names of the players and v is anonymous if it is defined only in terms of the cardinality of the argument coalition.

The following simple observation holds.

5.4.2. Note. For any v and ϕ, if $\rhd_v$ and $\rhd_\phi$ both realize the utilitarian order (as defined in Section 5.3), then for all collections C and C', we have $\phi^v(C) \rhd_\phi \phi^v(C')$ iff $v(C) \rhd_v v(C')$.

Proof. Immediate since

$$\sum_{A\in C} v(A) = \sum_{A\in C}\sum_{i\in A} \phi_i^v(A) = \sum_{A\in C, i\in A} \phi_i^v(A).$$

□

For other orders discussed in Section 5.3 no relation between $\rhd_v$ and $\rhd_\phi$ holds. In fact, we have the following results.

5.4.3. Theorem. *Given v and $\rhd_v$, it is in general not possible to define an anonymous individual value function ϕ along with $\rhd_\phi$ such that for all collections C and C', we have $\phi^v(C) \rhd_\phi \phi^v(C')$ iff $v(C) \rhd_v v(C')$. This holds even if we restrict ourselves to anonymous v.*

Proof. Consider the following game with $N = \{1, 2\}$:

$$v(\{1\}) := 1 \qquad v(\{2\}) := 1 \qquad v(\{1, 2\}) := 2,$$

and take $\rhd_v$ to be the Nash order as defined in Section 5.3. This yields both

$$v(\{\{1, 2\}\}) = \dot{\{}2\dot{\}} \rhd_v \dot{\{}1, 1\dot{\}} = v(\{\{1\}, \{2\}\})$$

and

$$v(\{\{1\}, \{2\}\}) \not\rhd_v v(\{\{1, 2\}\}).$$

However, the symmetry of the game and anonymity of ϕ forces

$$\phi^v(\{\{1,2\}\}) = \dot{\{}1,1\dot{\}} = \phi^v(\{\{1\},\{2\}\}),$$

so we have either

$$\phi^v(\{\{1,2\}\}) \rhd_\phi \phi^v(\{\{1\},\{2\}\}) \text{ and } \phi^v(\{\{1\},\{2\}\}) \rhd_\phi \phi^v(\{\{1,2\}\})$$

or

$$\phi^v(\{\{1,2\}\}) \not\rhd_\phi \phi^v(\{\{1\},\{2\}\}) \text{ and } \phi^v(\{\{1\},\{2\}\}) \not\rhd_\phi \phi^v(\{\{1,2\}\}).$$

□

5.4.4. THEOREM. *Given v, ϕ and $\rhd_\phi$, it is in general not possible to define $\rhd_v$ such that for all collections C and C', we have $v(C) \rhd_v v(C')$ iff $\phi^v(C) \rhd_\phi \phi^v(C')$. This holds even if we restrict ourselves to anonymous v, anonymous ϕ, and a Nash or leximin order (as defined in Section 5.3) for $\rhd_\phi$.*

Proof. Consider $N = \{1,\dots,4\}$ and

$$v(A) := 6 \text{ for all } A \subseteq N$$
$$\phi^v_i(A) := \frac{v(A)}{|A|}.$$

Then we have

$$\phi^v(\{\{1\},\{2,3,4\}\}) = \dot{\{}6,2,2,2\dot{\}}$$
$$\phi^v(\{\{1,2\},\{3,4\}\}) = \dot{\{}3,3,3,3\dot{\}},$$

which are distinguished by each of the mentioned $\rhd_\phi$, while

$$v(\{\{1\},\{2,3,4\}\}) = v(\{\{1,2\},\{3,4\}\}) = \dot{\{}6,6\dot{\}}.$$

□

These results suggest that the two approaches for defining comparison relations are fundamentally different and coincide only for the utilitarian order.

In the case of individual values we can introduce natural orders that have no counterpart for the comparison relations defined only by means of v. The reason is that for each partition P, $\phi^v(P)$ can be alternatively viewed as a sequence (of payoffs) of (the same) length n. Such sequences can then be compared using

- the *majority order*:
  $(k_1,\dots,k_n) \succ_m (\ell_1,\dots,\ell_n)$ iff $|\{i \mid k_i > \ell_i\}| > |\{i \mid \ell_i > k_i\}|$,

- the *Pareto order*:
  $(k_1, \ldots, k_n) \succ_p (\ell_1, \ldots, \ell_n)$ iff
  $\forall i \in \{1, \ldots, n\}\ k_i \geq \ell_i$ and $\exists i \in \{1, \ldots, n\}\ k_i > \ell_i$.

The relation $\succ_m$ is clearly irreflexive and monotonic both in sense (m1) and (m2). Unfortunately, it is not transitive. Indeed, we have both $(2, 3, 0) \succ_m (1, 2, 2)$ and $(1, 2, 2) \succ_m (3, 1, 1)$, but $(2, 3, 0) \succ_m (3, 1, 1)$ does not hold. In contrast, the relation $\succ_p$ is transitive, irreflexive, monotonic both in sense (m1) and (m2).

5.5 Stable partitions

We now return to our analysis of partitions. One way to identify conditions guaranteeing the unique outcome of the iterations of the merge and split rules is through focusing on the properties of such a unique outcome. This brings us to the concept of a stable partition.

We follow here the approach of Apt and Radzik [7], although now no notion of a game is present. The introduced notion is parametrized by means of a *defection function* $\mathbb{D}$ that assigns to each partition some partitioned subsets of the grand coalition. Intuitively, given a partition P, the family $\mathbb{D}(P)$ consists of all the collections $C := \{C_1, \ldots, C_\ell\}$ whose players can leave the partition P by forming a new, separate, group of players $\cup_{j=1}^{\ell} C_j$ divided according to the collection C. Two natural defection functions are $\mathbb{D}_p$, which allows formation of all partitions of the grand coalition, and $\mathbb{D}_c$, which allows formation of all collections in the grand coalition.

Next, given a collection C and a partition $P := \{P_1, \ldots, P_k\}$, we define

$$C[P] := \{P_1 \cap \bigcup C, \ldots, P_k \cap \bigcup C\} \setminus \{\emptyset\}$$

and call $C[P]$ the **collection C in the frame of P**. (By removing the empty set we ensure that $C[P]$ is a collection.) To clarify this concept consider Figure 5.1. We depict in it a collection C, a partition P, and C in the frame of P (together with P). Here C consists of four coalitions, while C in the frame of P consists of three coalitions.

Intuitively, given a subset S of N and a partition $C := \{C_1, \ldots, C_\ell\}$ of S, the collection C offers the players from S the "benefits" resulting from the partition of S by C. However, if a partition P of N is "in force", then the players from S enjoy instead the benefits resulting from the partition of S by $C[P]$, i.e., C in the frame of P.

To get familiar with the $C[P]$ notation, note that

- if C is a singleton, say $C = \{T\}$, then $\{T\}[P] = \{P_1 \cap T, \ldots, P_k \cap T\} \setminus \{\emptyset\}$, where $P = \{P_1, \ldots, P_k\}$,

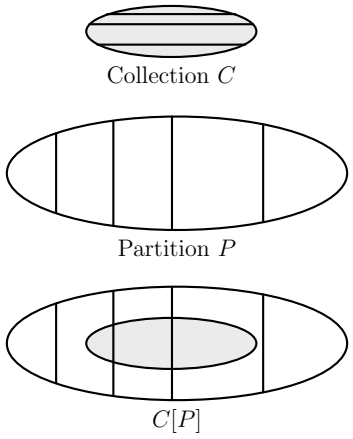


Figure 5.1: A collection C in the frame of a partition P

- if C is a partition of N, then $C[P] = P$,
- if $C \subseteq P$, that is, C consists of some coalitions of P, then $C[P] = C$.

In general, the following simple observation holds.

5.5.1. Fact. For a collection C and a partition P, $C[P] = C$ iff each element of C is a subset of a different element of P. □

This brings us to the following notion.

5.5.2. Definition. Assume a defection function $\mathbb{D}$ and a comparison relation $\rhd$. We call a partition P **$\mathbb{D}$-stable** if $C[P] \rhd C$ for all $C \in \mathbb{D}(P)$ such that $C[P] \neq C$. □

The last qualification, that is, $C[P] \neq C$, requires some explanation. Intuitively, this condition indicates that the players only care about the way they are partitioned. Indeed, if $C[P] = C$, then the partitions of $\bigcup C$ by means of P and by means of C coincide and are viewed as equally satisfactory for the players in $\bigcup C$. By disregarding the situations in which $C[P] = C$, we therefore adopt a limited viewpoint of cooperation according to which the players in C do not care about the presence of the players from outside of $\bigcup C$ in their coalitions.

The following observation holds, where we call a partition P of N *$\rhd$-maximal* if for all partitions P' of N different from P, $P \rhd P'$ holds.

5.5.3. Theorem. *A partition of N is $\mathbb{D}_p$-stable iff it is $\rhd$-maximal. In particular, a $\mathbb{D}_p$-stable partition of N exists if $\rhd$ is a semi-linear comparison relation.*

Proof. Note that if C is a partition of N, then $C[P] \neq C$ is equivalent to the statement $P \neq C$, since then $C[P] = P$. So a partition P of N is $\mathbb{D}_p$-stable iff for all partitions $P' \neq P$ of N, $P \rhd P'$ holds. □

In contrast, $\mathbb{D}_c$-stable partitions do not need to exist even if the comparison relation $\rhd$ is semi-linear.

5.5.4. Example. Consider $N = \{1, 2, 3\}$ and any semi-linear comparison relation $\rhd$ such that $\{\{1, 2, 3\}\} \rhd \{\{1\}, \{2\}, \{3\}\}$ and $\{\{a\}, \{b\}\} \rhd \{\{a, b\}\}$ for all $a, b \in \{1, 2, 3\}$ with $a \neq b$.

Then no partition of N is $\mathbb{D}_c$-stable. Indeed, $P := \{\{1\}, \{2\}, \{3\}\}$ is not $\mathbb{D}_c$-stable, since for $C := \{\{1, 2, 3\}\}$ we have $C[P] = \{\{1\}, \{2\}, \{3\}\} \not\rhd \{\{1, 2, 3\}\} = C$. Further, any other partition P contains some coalition $\{a, b\}$ and is thus not $\mathbb{D}_c$-stable either, since then for $C := \{\{a\}, \{b\}\}$ we have

$$C[P] = \{\{a, b\}\} \not\rhd \{\{a\}, \{b\}\} = C.$$

In [7] another example is given for the case of TU-games and utilitarian order. More precisely, a TU-game is defined in which no $\mathbb{D}_c$-stable partition exists, where $\rhd$ is defined through (5.1) using the utilitarian order $\succ_{\text{ut}}$.

5.6 Stable partitions and merge/split rules

We now resume our investigation of the conditions under which every iteration of the merge and split rules yields the same outcome. To establish the main theorem of this chapter and provide an answer in terms of $\mathbb{D}_c$-stable partitions, we first present the following three lemmas about $\mathbb{D}_c$-stable partitions.

5.6.1. Lemma. *Every $\mathbb{D}_c$-stable partition is closed under applications of the merge and split rules.*

Proof. To prove closure of a $\mathbb{D}_c$-stable partition P under the merge rule assume that for some $\{T_1, \ldots, T_k\} \subseteq P$ we have $\{\bigcup_{j=1}^{k} T_j\} \rhd \{T_1, \ldots, T_k\}$. $\mathbb{D}_c$-stability of P with $C := \{\bigcup_{j=1}^{k} T_j\}$ yields

$$\{T_1, \ldots, T_k\} = \{\bigcup_{j=1}^{k} T_j\}[P] \rhd \{\bigcup_{j=1}^{k} T_j\},$$

which is a contradiction by virtue of transitivity and irreflexivity of $\rhd$.

Closure under the split rule is shown analogously. □

Next, we provide a characterization of $\mathbb{D}_c$-stable partitions. Given a partition $P := \{P_1, \ldots, P_k\}$ we call here a coalition T **P-compatible** if for some $i \in \{1, \ldots, k\}$ we have $T \subseteq P_i$, and **P-incompatible** otherwise.

5.6.2. Lemma. *A partition $P = \{P_1, \ldots, P_k\}$ of N is $\mathbb{D}_c$-stable iff the following two conditions are satisfied (see Figure 5.2 for an illustration of the following coalitions):*

(i) for each $i \in \{1, \ldots, k\}$ and each pair of disjoint coalitions A and B such that $A \cup B \subseteq P_i$,

$$\{A \cup B\} \rhd \{A, B\}, \tag{5.2}$$

(ii) for each P-incompatible coalition $T \subseteq N$,

$$\{T\}[P] \rhd \{T\}. \tag{5.3}$$

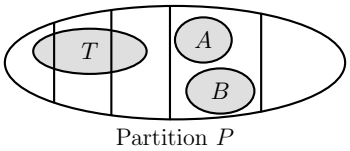


Figure 5.2: P-compatible coalitions A and B and a P-incompatible coalition T as in Lemma 5.6.2

Proof. ($\Rightarrow$) It suffices to note that for $C = \{A, B\}$ we have $C[P] = \{A \cup B\}$ and for $C = \{T\}$ we have $\{T\}[P] \neq \{T\}$ by the P-incompatibility of T. Then (i) and (ii) follow directly by the definition of $\mathbb{D}_c$-stability.

($\Leftarrow$) Transitivity, monotonicity (m2) and (5.2) imply by induction that for each $i \in \{1, \ldots, k\}$ and each collection $C = \{C_1, \ldots, C_\ell\}$ with $\ell > 1$ and $\bigcup C \subseteq P_i$,

$$\left\{\bigcup C\right\} \rhd C. \tag{5.4}$$

Let now C be an arbitrary collection in N such that $C[P] \neq C$. We prove that $C[P] \rhd C$. Define

$$D^i := \{T \in C \mid T \subseteq P_i\},$$
$$E := C \setminus \textstyle\bigcup_{i=1}^{k} D^i,$$
$$E^i := \{P_i \cap T \mid T \in E\} \setminus \{\emptyset\}.$$

Note that D^i is the set of P-compatible elements of C contained in P_i, E is the set of P-incompatible elements of C, and E^i consists of the non-empty intersections of P-incompatible elements of C with P_i.

Suppose now that $\bigcup_{i=1}^{k} E^i \neq \emptyset$. Then $E \neq \emptyset$ and consequently

$$\bigcup_{i=1}^{k} E^i = \bigcup_{i=1}^{k} (\{P_i \cap T \mid T \in E\} \setminus \{\emptyset\}) = \bigcup_{T \in E} (\{T\}[P]) \overset{\text{(m1),(5.3)}}{\rhd} E. \tag{5.5}$$

Consider now the following property:

$$|D^i \cup E^i| > 1. \tag{5.6}$$

Fix $i \in \{1, \dots, k\}$. If (5.6) holds, then

$$\left\{P_i \cap \bigcup C\right\} = \left\{\bigcup (D^i \cup E^i)\right\} \overset{(5.4)}{\rhd} D^i \cup E^i$$

and otherwise

$$\left\{P_i \cap \bigcup C\right\} = \left\{D^i \cup E^i\right\}.$$

Recall now that

$$C[P] = \bigcup_{i=1}^{k} \left\{P_i \cap \bigcup C\right\} \setminus \{\emptyset\}.$$

We distinguish two cases.

Case 1. (5.6) holds for some $i \in \{1, \dots, k\}$.

Then by (m1) and (m2)

$$C[P] \rhd \bigcup_{i=1}^{k} (D^i \cup E^i) = (C \setminus E) \cup \bigcup_{i=1}^{k} E^i.$$

If $\bigcup_{i=1}^{k} E^i = \emptyset$, then also $E = \emptyset$ and we get $C[P] \rhd C$. Otherwise by (5.5), transitivity and (m2)

$$C[P] \rhd (C \setminus E) \cup E = C.$$

Case 2. (5.6) does not hold for any $i \in \{1, \dots, k\}$.

Then

$$C[P] = \bigcup_{i=1}^{k} (D^i \cup E^i) = (C \setminus E) \cup \bigcup_{i=1}^{k} E^i.$$

Moreover, because $C[P] \neq C$, by Fact 5.5.1 a P-incompatible element in C exists. So $\bigcup_{i=1}^{k} E^i \neq \emptyset$, and by (5.5) and (m2) we get, as before,

$$C[P] \rhd (C \setminus E) \cup E = C.$$

□

In [7] the above characterization was proved for coalitional TU-games and the utilitarian order. We shall now use it in the proof of the following lemma.

5.6.3. Lemma. *Assume that P is $\mathbb{D}_c$-stable. Let P' be closed under applications of merge and split rules. Then $P' = P$.*

Proof. Suppose $P = \{P_1, \dots, P_k\}$, $P' = \{T_1, \dots, T_m\}$. Assume $P \neq P'$. Then there is $i_0 \in \{1, \dots, k\}$ such that for all $j \in \{1, \dots, m\}$ we have $P_{i_0} \neq T_j$. Let $T_{j_1}, \dots, T_{j_\ell}$ be the minimum cover of P_{i_0}. In the following case distinction, we use Lemma 5.6.2.

Case 1. $P_{i_0} = \bigcup_{h=1}^{\ell} T_{j_h}$.

Then $\{T_{j_1}, \ldots, T_{j_\ell}\}$ is a proper partition of P_{i_0}. But (5.2) (through its generalization to (5.4)) yields $P_{i_0} \rhd \{T_{j_1}, \ldots, T_{j_\ell}\}$, thus the merge rule is applicable to P'.

Case 2. $P_{i_0} \subsetneq \bigcup_{h=1}^{\ell} T_{j_h}$.

Then, for some j_h, we have $\emptyset \neq P_{i_0} \cap T_{j_h} \subsetneq T_{j_h}$, so T_{j_h} is P-incompatible. By (5.3), we have $\{T_{j_h}\}[P] \rhd \{T_{j_h}\}$, thus the split rule is applicable to P'. □

We can now present the desired result.

5.6.4. Theorem. *Suppose that $\rhd$ is a comparison relation and P is a $\mathbb{D}_c$-stable partition. Then*

(i) P is the outcome of every iteration of the merge and split rules.

(ii) P is a unique $\mathbb{D}_p$-stable partition.

(iii) P is a unique $\mathbb{D}_c$-stable partition.

Proof. (*i*) By Note 5.2.2, every iteration of the merge and split rules terminates, so the claim follows by Lemma 5.6.3.

(*ii*) Since P is $\mathbb{D}_c$-stable, it is in particular $\mathbb{D}_p$-stable. By Theorem 5.5.3, for all partitions $P' \neq P$, $P \rhd P'$ holds. So uniqueness follows from transitivity and irreflexivity of $\rhd$.

(*iii*) Suppose that P' is a $\mathbb{D}_c$-stable partition. By Lemma 5.6.1, P' is closed under applications of the merge and split rules, so by Lemma 5.6.3, $P' = P$. □

This theorem generalizes [7], where this result was established for coalitional TU-games and the utilitarian order. It was also shown that there exist coalitional TU-games in which all iterations of the merge and split rules have a unique outcome which is not a $\mathbb{D}_c$-stable partition.

5.7 Applications

The obtained results do not involve any notion of a game. In this section, we show applications to three classes of coalitional games. In each case we define a class of games and a natural comparison relation for which all iterations of the merge and split rules have a unique outcome.

5.7.1 Coalitional TU-games

To show that the obtained results naturally apply to coalitional TU-games, consider first the special case of the utilitarian order, according to which, given a coalitional

TU-game (N, v), for two collections $P := \{P_1, \ldots, P_k\}$ and $Q = \{Q_1, \ldots, Q_\ell\}$ such that $\bigcup P = \bigcup Q$, we have

$$P \rhd Q \text{ iff } \textstyle\sum_{i=1}^{k} v(P_i) > \sum_{i=1}^{\ell} v(Q_i).$$

Recall that (N, v) is called *strictly super-additive* if for each pair of disjoint coalitions A and B

$$v(A) + v(B) < v(A \cup B).$$

Further, recall from [108, p. 241] that, given a partition $P := \{P_1, \ldots, P_k\}$ of N and coalitional TU-games $(P_1, v_1), \ldots, (P_k, v_k)$, their *composition* $(N, \oplus_{i=1}^{k} v_i)$ is defined by

$$(\oplus_{i=1}^{k} v_i)(A) = \sum_{i=1}^{k} v_i(P_i \cap A).$$

We now modify this definition and introduce the concept of a *semi-union* of $(P_1, v_1), \ldots, (P_k, v_k)$, written as $(N, \overline{\oplus}_{i=1}^{k} v_i)$, and defined by

$$(\overline{\oplus}_{i=1}^{k} v_i)(A) := \begin{cases} (\oplus_{i=1}^{k} v_i)(A) & \text{if } A \subseteq P_i \text{ for some } i \\ (\oplus_{i=1}^{k} v_i)(A) - \epsilon & \text{otherwise,} \end{cases}$$

where $\epsilon > 0$.

So for P-incompatible coalitions the payoff is strictly smaller for the semi-union of TU-games than for their union, while for other coalitions the payoffs are the same. It is then easy to prove, using Lemma 5.6.2, that in the semi-union $(N, \overline{\oplus}_{i=1}^{k} v_i)$ of strictly super-additive TU-games, the partition P is $\mathbb{D}_c$-stable. Consequently, by Theorem 5.6.4, in this game, P is the outcome of every iteration of the merge and split rules.

The following more general example deals with arbitrary monotonic comparison relations, as introduced in Sections 5.3 and 5.4.

5.7.1. Example. Given a partition $P := \{P_1, \ldots, P_k\}$ of N, with $\rhd$ being one of the orders defined in Section 5.3, we define a TU-game for which P is the outcome of every iteration of the merge and split rules.

Let

$$f(x, y) := \begin{cases} x + y & \text{if } \rhd \text{ is the utilitarian order} \\ x \cdot y & \text{if } \rhd \text{ is the Nash order} \\ \max\{x, y\} & \text{if } \rhd \text{ is the leximin order} \end{cases}$$

and define

$$v(A) := \begin{cases} 1 & \text{if } |A| = 1 \\ \max\limits_{\substack{B \cup C = A \\ B, C \text{ disjoint} \\ \text{coalitions}}} \{f(v(B), v(C))\} + 1 & \text{if } |A| > 1 \text{ and } A \subseteq P_i \text{ for some } i \\ 0 & \text{otherwise.} \end{cases}$$

Then

(i) for any two disjoint coalitions A, B with $A \cup B \subseteq P_i$ for some i, we have

$$v(A \cup B) > f(v(A), v(B))$$

by construction of v, and thus

- $v(A \cup B) > v(A) + v(B)$ for utilitarian $\rhd$;
- $v(A \cup B) > v(A) \cdot v(B)$ for Nash $\rhd$;
- $v(A \cup B) > \max\{v(A), v(B)\}$ for leximin $\rhd$.

Hence, in all cases $\{A \cup B\} \rhd \{A, B\}$.

(ii) for any P-incompatible coalition $T \subseteq N$, we have

$$v(A) > 0 \text{ for all } A \in \{T\}[P], \quad \text{and } v(T) = 0.$$

Hence, $\{T\}[P] \rhd \{T\}$.

Lemma 5.6.2 now implies that P is indeed $\mathbb{D}_c$-stable, so Theorem 5.6.4 applies.

5.7.2. Example. Given a partition $P := \{P_1, \ldots, P_k\}$ of N, with $\rhd$ being one of the orders defined in Section 5.3 or the Pareto order from Section 5.4, we define a TU-game and an individual value function for which P is the outcome of every iteration of the merge and split rules.

Let

$$f(x, y) := \begin{cases} |N| \cdot \max\{x, y\} + 1 & \text{if } \rhd \text{ is leximin or Pareto} \\ x + y & \text{otherwise,} \end{cases}$$

define v as in Example 5.7.1, and define

$$\phi_i^v(A) := \frac{v(A)}{|A|}.$$

Then

(i) for any two disjoint coalitions A, B with $A \cup B \subseteq P_i$ for some i, we have

$$v(A \cup B) > f(v(A), v(B))$$

again by construction of v, and thus

- for utilitarian or Nash $\rhd$:
  $v(A \cup B) > v(A) + v(B)$, and since ϕ_i^v distributes the value evenly, in all cases $\{A \cup B\} \rhd \{A, B\}$,

- for leximin or Pareto $\rhd$:
  $v(A \cup B) > |A \cup B| \cdot \max\{v(A), v(B)\}$,
  thus $\phi_i^v(A \cup B) > \max\{v(A), v(B)\}$ for all i,
  thus $\{A \cup B\} \rhd \{A, B\}$ in all cases,

(ii) for any P-incompatible coalition $T \subseteq N$, $\{T\}[P] \rhd \{T\}$ as before.

Again, Lemma 5.6.2 implies that P is $\mathbb{D}_c$-stable, and Theorem 5.6.4 applies.

5.7.2 Hedonic games

Recall that a **hedonic game** $(N, \succeq_1, \ldots, \succeq_n)$ consists of a set of players $N = \{1, \ldots, n\}$ and a sequence of linear preorders $\succeq_1, \ldots, \succeq_n$, where each $\succeq_i$ is the *preference* of player i over the subsets of N containing i. In what follows, we do not need the assumption that the $\succeq_i$ relations are linear. We use $\succ_i$ to denote the associated irreflexive relation.

Given a partition P of N and a player i, we denote by $P(i)$ the element of P to which i belongs and call it the set of *friends of* i *in* P.

We now provide an example of a hedonic game in which a $\mathbb{D}_c$-stable partition w.r.t. to a natural comparison relation $\succ$ exists.

To this end, we assume that, given a partition $P := \{P_1, \ldots, P_k\}$ of N, each player

- prefers a larger set of his friends in P over a smaller one,
- "dislikes" coalitions that include a player who is not his friend in P.

We formalize this by putting for all sets of players that include i

$$S \succeq_i T \text{ iff } T \subseteq S \subseteq P(i),$$

and extending this order to coalitions that include player i and possibly players from outside of $P(i)$ by assuming that such coalitions are the minimal elements in $\succeq_i$. So

$$S \succ_i T \text{ iff either } T \subsetneq S \subseteq P(i) \text{ or } S \subseteq P(i) \text{ and not } T \subseteq P(i).$$

We then define, for any two partitions Q and Q' of the same set of players,

$$Q \rhd Q' \text{ iff, for } i \in \{1, \ldots, n\},\ Q(i) \succeq_i Q'(i) \text{ with at least one } \succeq_i \text{ being strict.}$$

(Note the similarity between this relation and the $\succ_p$ relation introduced in Section 5.4.) It is straightforward to check that $\rhd$ is indeed a comparison relation and that the partition P satisfies conditions (5.2) and (5.3) of Lemma 5.6.2. So by virtue of this result, P is $\mathbb{D}_c$-stable. Consequently, on the account of Theorem 5.6.4, the partition P is the outcome of every iteration of the merge and split rules.

5.7.3 Exchange economy games

Recall that an *exchange economy* consists of

- a market with k goods,
- for each player i an initial endowment of these goods represented by a vector $\vec{\omega}_i \in \mathcal{R}_+^k$,
- for each player i a transitive and linear *preference relation* $\succeq_i$, using which he can compare the bundles of goods, represented as vectors from $\mathcal{R}_+^k$.

An **exchange economy game** is then defined by first taking as the set of *outcomes* the set of all sequences of bundles,

$$X := \{(\vec{x}_1, \ldots, \vec{x}_n) \mid \vec{x}_i \in \mathcal{R}_+^k \text{ for } i \in N\},$$

i.e., $X = (\mathcal{R}_+^k)^n$, and extending each preference relation $\succeq_i$ from the set $\mathcal{R}_+^k$ of all bundles to the set X by putting, for $\vec{x}, \vec{y} \in X$,

$$\vec{x} \succeq_i \vec{y} \text{ iff } \vec{x}_i \succeq_i \vec{y}_i. \tag{5.7}$$

This simply means that each player is only interested in his own bundle.

Then we assign to each coalition S the following set of outcomes:

$$V(S) := \{\vec{x} \in X \mid \textstyle\sum_{i\in S} \vec{x}_i = \sum_{i\in S} \vec{\omega}_i \text{ and } \vec{x}_j = \vec{\omega}_j \text{ for all } j \in N \setminus S\}.$$

So $V(S)$ consists of the set of outcomes that can be achieved by trading among the members of S.

Given a partition $P = \{P_1, \ldots, P_k\}$ of $N = \{1, \ldots, n\}$, we now define a specific exchange economy game with n goods (one type of good for each player) as follows, where $i \in N$:

$$\begin{aligned} &\vec{\omega}_i := \text{characteristic vector of } P(i), \\ &\vec{x}_i \succeq_i \vec{y}_i \text{ iff } x_{i,i} \geq y_{i,i} \quad \text{and} \quad \vec{x}_i \succ_i \vec{y}_i \text{ iff } x_{i,i} > y_{i,i}, \end{aligned}$$

that is, each player's initial endowment consists of exactly one good of the type of each of his friends in P, and he prefers a bundle if he gets more goods of his own type.

Now let $A \rhd B$ iff

$$\begin{gathered} \forall A_\ell \in A \setminus B \; \exists \vec{x} \in V(A_\ell) \, \forall j \in A_\ell \\ \left[\left(\forall \vec{y} \in V(B(j)) \vec{x} \succ_j \vec{y}\right) \vee \left(\forall \vec{y} \in V(B(j)) \vec{x} \succeq_j \vec{y} \wedge |A_\ell| < |B(j)|\right)\right]. \end{gathered}$$

So a partition A is preferred to a partition B if each coalition A_ℓ of A not present in B can achieve an outcome which each player of A_ℓ strictly prefers to

any outcome of his respective coalition in B, or which he likes at least as much as any outcome of his respective coalition in B when that coalition is strictly larger than A_ℓ. The intuition is that the players' preferences over outcomes weigh most, but in case of ties the players prefer smaller coalitions.

It is easy to check that $\rhd$ is a comparison relation. We now prove that the partition P is $\mathbb{D}_c$-stable w.r.t. $\rhd$. First, note that by definition of the initial endowments, for all $\ell \in \{1, \ldots, k\}$ and coalitions $A \subseteq P_\ell$ there is an outcome $\vec{z}_A \in V(A)$ which gives exactly $|A|$ units of good j to each player $j \in A$. We have $\vec{z}_A \succeq_i \vec{x}$ for all $i \in A$ and $\vec{x} \in V(A)$. This implies that P is $\mathbb{D}_c$-stable by Lemma 5.6.2, since

(i) for each pair of disjoint coalitions A and B such that $A \cup B \subseteq P_\ell$, we have $\vec{z}_{A\cup B} \succ_i \vec{z}_A$ for each $i \in A$ and $\vec{z}_{A\cup B} \succ_i \vec{z}_B$ for each $i \in B$ since $|A \cup B| > |A|$ and $|A \cup B| > |B|$, thus $\{A \cup B\} \rhd \{A, B\}$,

(ii) for any P-incompatible $T \subseteq N$, $A \in \{T\}[P]$, $i \in A$, and $\vec{x} \in V(T)$, we have $\vec{z}_A \succeq_i \vec{x}$ (since player i can get in T at most all goods of his type from his friends in P, which are exactly the same as in A), and $|A| < |T|$, thus $\{T\}[P] \rhd \{T\}$.

Consequently, in the above game, by Theorem 5.6.4 the partition $\{P_1, \ldots, P_k\}$ is the outcome of every iteration of the merge and split rules.

5.8 Conclusions

We have presented a generic approach to coalition formation, in which the only possible operations on coalitions are merges and splits. These operations can take place when they result in an improvement with respect to some given comparison relation on partitions of the involved subset of players. Such a comparison relation needs to satisfy only a few natural properties, namely irreflexivity, transitivity and monotonicity, and we have given examples induced by several well-known orders in the context of TU-games.

We have identified natural conditions under which every iteration of merges and splits yields a unique outcome, which led to a natural notion of a stable partition. We have shown that, besides general TU-games, our approach and results also naturally apply to hedonic games and exchange economy games.

It would be interesting to extend this approach and allow other transformations, such as transfers (moving a subset of one coalition to another) or, more generally, swaps (exchanging subsets of two coalitions), as considered by Apt and Radzik [7] in the setting of TU-games and the utilitarian order.

Chapter 6

Time constraints in mixed auctions

6.1 Introduction

6.1.1 Motivation

Combinatorial auctions are auction mechanisms where bidders can bid for sets of goods rather than just single items. Despite the fact that solving a combinatorial auction, i.e., finding an allocation of goods to bidders that will maximize the sum of the prices offered by the bidders, is an intractable optimization problem, combinatorial auctions have turned out to be an extremely useful tool with many applications. See the textbook by Cramton et al. [43] for an overview.

In recent work, Cerquides et al. [37] have proposed an extension of the standard combinatorial auction model, called **mixed multi-unit combinatorial auctions** (or simply **mixed auctions** for short). In a mixed auction, bidders can offer transformations, consisting of a set of input goods and a set of output goods, rather than just plain goods. Bidding for such a transformation means declaring that one is willing to deliver the specified output goods after having received the input goods, for the price specified by the bid. Solving a mixed auction means choosing a sequence of transformations that satisfies the constraints encoded by the bids, that produces the goods required by the auctioneer from those he holds initially, and that maximizes the amount of money collected from the bidders (or minimizes the amount paid out by the auctioneer). Mixed auctions extend a number of other types of combinatorial auctions: direct auctions, reverse auctions, and combinatorial exchanges. A very promising application of mixed auctions is supply chain formation [148].

In this chapter, we extend the framework of mixed auctions by allowing bidders to specify constraints regarding the times at which they perform the transformations offered in their bids. The motivation for this extension is that, in a complex economy, the bidders (service providers) themselves may need services from others and have their own supply chains, so the bidders may have preferences

over the timing of transformations and over their relative ordering. A notion of time is already implicit in the original mixed auction framework as far as the auctioneer is concerned, because he has to build a sequence of transformations, but this is not the case for the bidders. In this work we seek to redress this imbalance.

6.1.2 Approach

Our contribution covers four types of **time constraints**:

- *Relative time points:* This is the simplest model. It associates each transformation with a time point and allows bidders to express constraints regarding their relative ordering. For example, a bidder may want to offer transformations X and Y, but only under the condition that X be executed before Y.

- *Absolute time points:* This is an extension to the first model and in addition allows for references to absolute time. For example, a bidder could specify that transformation X can only be executed at time unit 15, or that it needs to be executed at most 3 time units after transformation Y.

- *Intervals:* Alternatively, transformations may be associated with time intervals, and bidders can express constraints on whether one transformation should be executed *before*, *during*, or *after* another transformation, or whether two transformations need to be executed in *overlapping* time intervals.

- *Intervals with absolute durations:* Combining the notions of interval and absolute time, we can also express constraints on the duration of an interval. For example, a bidder may want to specify that transformation X requires at least 5 time units.

These constraint types can be freely mixed to, for instance, express an interval taking place after a time point.

We also show how to model soft constraints, allowing bidders to offer discounts in return for satisfying certain time constraints, and how to model the fact that an auctioneer may sometimes be able to quantify the monetary benefit resulting from a shorter supply chain. As in the original paper by Cerquides et al. [37], we are working with the multi-unit variant of mixed auctions, where there may be several copies of the same good available, but the restriction to single units is certainly also of interest.

We define a **bidding language** for expressing combinatorial bids over transformations with associated time constraints, we give a precise definition of the **winner determination problem**, and we present an algorithm for solving it by showing how the problem can be encoded as an **integer program**. As we will see, our approach blends nicely into the existing framework of mixed auctions,

requiring surprisingly few modifications. This facilitates the integration of time constraints with other extensions and optimizations discussed in the literature.

6.1.3 Plan of the chapter

The remainder of this chapter is structured as follows.

Section 6.2 defines a bidding language for mixed auctions with time constraints. We give a formal semantics for this language by showing how any given bid expression is mapped to a valuation function defined over sets of transformations arranged on a time line, and we discuss its expressive power.

In Section 6.3 we formulate the winner determination problem. Building on the original algorithm by Cerquides et al. [37], we show how to model it as an integer program. For relative time constraints, the extension is particularly neat and simple; to accommodate absolute time we show how to overcome a number of conceptual and technical difficulties.

While Sections 6.2 and 6.3 cover the cases of constraints over relative and absolute time points, Section 6.4 presents the extension to time intervals.

Finally, Section 6.5 concludes and briefly discusses related work.

6.2 Bidding language

In this section, we define and discuss a bidding language for mixed auctions with time constraints.

We first define transformations with time points and then valuations, which are functions used to model the preferences of bidders by mapping sets of such transformations with time points to numerical values. We then define the syntax and semantics of our bidding language. The purpose of a bidding language is to encode a bidder's valuation for transmission to the auctioneer.[1] The basic idea is to use a simple XOR-language, of the kind familiar from standard combinatorial auctions [105, 133] and mixed auctions [37], to express possible combinations of transformations and associated prices, and to use time constraints to impose conditions on the ordering of their execution.

We also add some syntactic sugar to the time constraint language which *prima facie* seems to have very limited expressive power, and show that it is actually surprisingly expressive. This is achieved by "borrowing" expressive power from the underlying bidding language into the time constraint language. We conclude this section by arguing that our bidding language is fully expressive over the class of all "reasonable" valuations.

[1] Bidders may or may not wish to transmit their *true* valuation. This is an important issue, but one that is entirely irrelevant for the *representation* of bids, as long as all necessary valuations can be expressed.

6.2.1 Transformations and time points

Let G be the finite set of all types of goods under consideration. A **transformation** is a pair

$$(\mathcal{I}, \mathcal{O}) \in \mathbb{N}^G \times \mathbb{N}^G.$$

An agent offering such a transformation declares that, when provided with the multiset of goods $\mathcal{I}$, he can deliver the multiset of goods $\mathcal{O}$.

When agents submit their bid, they should be able to put restrictions on the time points at which they offer to perform transformations. To this end, we introduce a finite (but big enough) set of **time point identifiers** $\mathcal{T}$. The time points here are to be thought of merely as identifiers, not as variables having an actual value. Agents are negotiating over sets of transformations with time point identifiers

$$\mathcal{D} \subset \mathbb{N}^G \times \mathbb{N}^G \times \mathcal{T},$$

which we can write as sets of the form

$$\{(\mathcal{I}^1, \mathcal{O}^1, \tau^1), \ldots, (\mathcal{I}^\ell, \mathcal{O}^\ell, \tau^\ell)\}.$$

For example, $\{(\{\}, \{q\}, \tau_1), (\{r\}, \{s\}, \tau_2)\}$ means that the agent in question is able to deliver q without any input at some time point τ_1, and to deliver s if provided with r at some time point τ_2. Note that this is not the same as $\{(\{r\}, \{q, s\}, \tau)\}$. In the former case, if another agent can produce r from q, we can get s from nothing; in the latter case this does not work.

We assume that each time point identifier is used at most once, even across bidders, and thus indeed identifies one particular transformation of one particular bidder.

Note that this does not allow two transformations to be offered at the same time point identifier. If a bidder wants to do that, he can instead offer a single transformation with combined $\mathcal{I}$ and $\mathcal{O}$.

6.2.2 Valuations

Since bidders may have preferences over the time points at which they perform transformations, we first define a **time line** Σ of transformations to be a finite sequence of transformations and "clock ticks" c. The latter stand for time passing without any transformations allocated to the bidder. That is,

$$\Sigma \in (\mathbb{N}^G \times \mathbb{N}^G \cup \{c\})^*,$$

where * denotes words as defined in Chapter 2, Section 2.2.

A **valuation** v maps a time line Σ to a real number p. Intuitively, $v(\Sigma) = p$ means that an agent with valuation v is willing to make a payment of p for getting the task of performing transformations according to the time line Σ (intuitively, p

is usually negative, meaning that the agent is actually *being paid* for performing the transformations). We write $v(\Sigma) = \bot$ if v is *undefined* for Σ, i.e., the agent would be unable to accept the corresponding deal.

For example, the valuation v given by

$$\begin{aligned} v((\{oven, dough\}, \{oven, cake\})) &= -2 \\ v((\{oven, dough\}, \{oven, cake\}); (\{\}, \{bread\})) &= -3 \\ v((\{\}, \{bread\}); (\{oven, dough\}, \{oven, cake\})) &= \bot \end{aligned}$$

expresses that for two dollars I could produce a cake if given an oven and dough, also returning the oven; for another dollar I could do the same and afterwards give you a bread without any input; but I could not do it the other way round (which should make you wonder whether you might be giving away too much dough in the first case).

We say that a valuation v uses **relative time** if for all Σ we have that $v(\Sigma) = v(\Sigma - c)$, where $\Sigma - c$ stands for Σ with all clock ticks c removed. Otherwise v is said to use **absolute time**.

That is, in the context of relative time, valuations depend only on the relative ordering of the transformations, while with absolute time valuations may depend on the absolute time point at which a transformation occurs within a given sequence. In the latter case, we interpret each step in the sequence as a **time unit** (one second, one day, one week, ...).

Note that the fact that we are dealing with a sequence agrees with the assumption above that no two transformations can be offered at the same time point identifier. Again, from the bidder's side this can be overcome by simply combining the two transformations that he would like to offer concurrently. In Section 6.3, we get back to this issue from the auctioneer's point of view, where it has slightly more serious conceptual consequences. Note also that all transformations are assumed to have the same duration of one time unit; in Section 6.4 we look at how to deal with transformations of variable durations.

6.2.3 Bids

We are now ready to define the bidding language. An **atomic bid** $\textsc{bid}(\mathcal{D}, p)$ specifies a finite set of finite transformations with time points and a price. Intuitively, submitting such a bid expresses the bidder's willingness to perform the given transformations for price p.

Constraints on the time points at which these transformations should be scheduled will be added on top of this bid. But before we turn to that, we define the rest of our bidding language. We restrict ourselves to the XOR-*language*, which is known to be fully expressive for standard (single-unit) combinatorial auctions [105], and which, for multi-unit mixed auctions, has been shown to subsume many other intuitively useful language constructs and to fully express a

class of valuations comprising most (if not all) intuitively sensible ones [37]. In particular, it is not difficult to extend the auction framework presented in this chapter to also handle the so-called OR-operator [105, 37].

So we consider **complex bids** in XOR normal form,

$$Bid = \text{BID}(\mathcal{D}_1, p_1) \text{ XOR} \ldots \text{XOR BID}(\mathcal{D}_n, p_n),$$

with the intuitive interpretation that the bidder is willing to perform *at most one* of the $\mathcal{D}_j$ and pay the associated p_j. Note again that the used time point identifiers are unique, also across the $\mathcal{D}_j$.

6.2.4 Time constraints

We now define two variants of a time constraint language. The **atomic constraints** for *relative time* are all of the form $\tau < \tau'$. For *absolute time*, we allow the following (where $\tau, \tau' \in \mathcal{T}$, $\xi, \xi' \in \mathbb{N}$):

$$\begin{array}{ccc} \tau = \xi & \tau < \xi & \tau > \xi \\ \tau + \xi < \tau' + \xi' & & \tau + \xi = \tau' + \xi' \end{array}$$

The reason why we distinguish between the two variants, even though the latter may seem to subsume the former, is that the former is conceptually "safer", at least in our formalism which is based on sequences: Concepts like "before" and "after" always make intuitive sense, while with absolute time we have to agree on the time unit of a sequence step and take timing issues into account. This is not always without difficulties, as noted in several places in this chapter; see, e.g., the discussion in Section 6.3.3.

As an example, the atomic bid with time constraint

$$\begin{aligned} &\text{BID}(\{\, (\{oven, dough\}, \{oven, cake\}, \tau_1), \\ &\qquad (\{\}, \{bread\}, \tau_2)\}, -3) \\ &\tau_1 < \tau_2 \end{aligned}$$

expresses the above fact that I am willing to sell you the bread only *after* I have sold you the cake.

We allow time **constraint formulas** of the form

$$\varphi = \gamma_1 \wedge \ldots \wedge \gamma_\nu$$

with atomic constraints γ_ι. A bidder submits a bid *Bid* together with a time constraint formula φ (the atomic constraints of which refer to time point identifiers occurring in *Bid*). Intuitively, this expresses that he is willing to perform according to *Bid*, but *only under the condition* that φ is satisfied.

This condition is *hard*: the bidder will only accept if it is met. In Section 6.2.6 we give a method to express *soft* time constraints (associated with costs), and we also show that *disjunctions* over time constraints can be expressed.

6.2.5 Semantics

Syntactically, we are thus dealing with *complex bids with time points* together with *constraint formulas* over these time points:

$$\begin{aligned}&\text{BID}(\mathcal{D}_1, p_1) \text{ XOR} \ldots \text{XOR BID}(\mathcal{D}_n, p_n)\\ &\gamma_1 \wedge \ldots \wedge \gamma_\nu.\end{aligned}$$

In order to make the intuitive meanings explained above explicit, we now specify a formal semantics. In the following, let Σ be a time line (clock ticks allowed), let $\tau, \tau' \in \mathcal{T}$, $\xi, \xi' \in \mathbb{N}$, and let φ and φ' be time constraint formulas. Let $\tau \in \Sigma$ denote the fact that τ is associated with some transformation in Σ, and let $\Sigma(\tau)$ denote the sequence number (starting from 1) of the transformation associated with τ, if $\tau \in \Sigma$. For clarity, we may include the time point identifiers in the sequence. For example, if $\Sigma = ((\mathcal{I}^1, \mathcal{O}^1, \tau^1); \ldots)$, then $\tau^1 \in \Sigma$ and $\Sigma(\tau^1) = 1$.

We inductively define a **satisfaction relation** $\vDash$ as follows:

$\Sigma \vDash \tau \circ \xi$	iff $\tau \notin \Sigma$ or $\Sigma(\tau) \circ \xi$, for $\circ \in \{=, <, >\}$
$\Sigma \vDash \tau + \xi < \tau' + \xi'$	iff $\tau' \notin \Sigma$ or $\tau \in \Sigma$ and $\Sigma(\tau) + \xi < \Sigma(\tau') + \xi'$
$\Sigma \vDash \varphi \wedge \varphi'$	iff $\Sigma \vDash \varphi$ and $\Sigma \vDash \varphi'$

Relative time constraints are covered by omitting the $+\,\xi$ and $+\,\xi'$, and $\tau + \xi = \tau' + \xi'$ is defined to be an abbreviation for

$$\tau + \xi < \tau' + (\xi' + 1) \wedge \tau' + \xi' < \tau + (\xi + 1).$$

According to this semantics, time constraints over time point identifiers that are fully included in Σ are interpreted as expected. Constraints over time point identifiers not in Σ are simply ignored (they are always satisfied). Note that the choice of semantics for constraints such as $\tau < \tau'$ is somewhat arbitrary in case only one of the time points being compared occurs in Σ. As an intuitive justification for this detail of the semantics, τ may be thought of as a precondition for τ', for instance, because some outcome of the first transformation is needed for the second. In the case of $\tau + \xi = \tau' + \xi'$, this has the effect that either none of the two mentioned transformations is included, or both are and must have the specified distance. However, the exact details do not matter all that much, since the bidding language allows specifying in all detail which transformations can occur together and which cannot.

Using a more technical justification, we prefer this interpretation of constraints because it turns out that it has a straightforward translation to integer constraints, which we need for the implementation described in Section 6.3.3.

We say that a set of transformations $\mathcal{D}$ **permits** Σ if Σ consists of *exactly* the transformations in $\mathcal{D}$ (and optionally clock ticks). In contrast to this definition,

in [37], different assumptions concerning *free disposal* are distinguished. Informally, free disposal means that participants are always happy to accept *more* goods than they strictly require; if they really have absolutely no use for them (or are even bothered by them), they can dispose of them *for free*. Free disposal makes intuitive sense for most every-day goods; however it is not as appropriate for certain "goods" like nuclear waste. We do not delve further into this issue here and continue without any free-disposal assumptions; however, we emphasize that this is purely for the sake of clarity, and these assumptions could be built in with only minuscule changes. In particular, the issue of free disposal as far as bidders are concerned has no impact on the winner determination problem discussed in Section 6.3; it only affects the definition of the semantics of the bidding language.

We now define the valuation expressed by an atomic bid $Bid = (\mathcal{D}, p)$ together with a time constraint formula φ to be:

$$v_{Bid,\varphi}(\Sigma) = \begin{cases} p & \text{if } \mathcal{D} \text{ permits } \Sigma \text{ and } \Sigma \vDash \varphi \\ \bot & \text{otherwise.} \end{cases}$$

Accordingly, the valuation expressed by a complex bid $Bid = \textsc{xor}_{j=1}^{n} Bid_j$ together with a time constraint formula φ is (interpreting $\bot$ as $-\infty$):

$$v_{Bid,\varphi}(\Sigma) = \max\{v_{Bid_j,\varphi}(\Sigma) \mid j \in \{1, \dots, n\}\}.$$

That is, out of all the applicable atomic bids Bid_j (i.e., where $v_{Bid_j,\varphi}(\Sigma) \neq \bot$), the auctioneer is allowed to choose the one giving him maximum profit.

6.2.6 Syntactic sugar

The time constraint language seems rather limited at first glance, allowing only conjunctions of atomic constraints. However, it turns out that additional expressive power can be "borrowed" from the bidding language. We discuss two extensions to the time constraint language, which can be rewritten into the core language by using additional bid expressions.

Disjunctive time constraints. If a bidder wants to offer $(\mathcal{I}^1, \mathcal{O}^1)$, $(\mathcal{I}^2, \mathcal{O}^2)$ and $(\mathcal{I}^3, \mathcal{O}^3)$ with price p, where the third should take place after the second *or* after the first, it seems natural to write

$$\begin{aligned} &\textsc{bid}(\{(\mathcal{I}^1, \mathcal{O}^1, \tau^1), (\mathcal{I}^2, \mathcal{O}^2, \tau^2), (\mathcal{I}^3, \mathcal{O}^3, \tau^3)\}, p) \\ &\tau^1 < \tau^3 \vee \tau^2 < \tau^3, \end{aligned}$$

with the obvious meaning of the disjunction $\vee$. This is not directly possible in our time constraint language. However, it can be translated into

$$\begin{aligned} &\textsc{bid}(\{(\mathcal{I}^1, \mathcal{O}^1, \vartheta^1), (\mathcal{I}^2, \mathcal{O}^2, \vartheta^2), (\mathcal{I}^3, \mathcal{O}^3, \vartheta^3)\}, p) \\ &\quad \textsc{xor}\, \textsc{bid}(\{(\mathcal{I}^1, \mathcal{O}^1, \zeta^1), (\mathcal{I}^2, \mathcal{O}^2, \zeta^2), (\mathcal{I}^3, \mathcal{O}^3, \zeta^3)\}, p) \\ &\vartheta^1 < \vartheta^3 \wedge \zeta^2 < \zeta^3. \end{aligned}$$

The choice which of the disjuncts to satisfy has been moved into the bid expression and is determined by picking one of the atomic bids. Since their variables are disjoint, this pick makes one conjunct of the transformed time constraint formula vacuously true, while the other conjunct still needs to be satisfied. Since it may perfectly well happen that both of the original disjuncts are satisfied in the end, disjunction is the right notion here, even though it is translated into an XOR of bids.

For a general formulation, we allow a bid expression in XOR normal form together with a time constraint formula in disjunctive normal form:

$$\operatorname*{XOR}_{j=1}^{n} Bid_j \qquad\qquad \bigvee_{\iota=1}^{\nu} \varphi_\iota,$$

where the φ_ι are standard (conjunctive) time constraint formulas. The bidder can thus conveniently express, e.g., several alternative partial orders over his transformations. Let now σ_ι for $\iota \in \{1, \ldots, \nu\}$ be substitutions (with disjoint ranges), each mapping all variables occurring in the bid to fresh (used nowhere else) ones. The resulting translation is

$$\operatorname*{XOR}_{\iota=1}^{\nu} \operatorname*{XOR}_{j=1}^{n} Bid_j\sigma_\iota \qquad\qquad \bigwedge_{\iota=1}^{\nu} \varphi_\iota\sigma_\iota.$$

This may seem surprising, because in the original formulation the auctioneer has two choices (which of the time constraint disjuncts to satisfy and which bid to pick), and in the translation he loses the choice among the time constraints. However, in return he gets the freedom to choose over the outer XOR. As illustrated in the example above, this boils down to choosing one of the fresh variable spaces, which corresponds to choosing one of the original disjuncts. All the rest of the transformed time conjunction does not have any effect, because it talks about variables which do not occur in the chosen sub-bid. The auctioneer then proceeds to pick a bid from the inner XOR, just as before.

Soft time constraints. Soft constraints are constraints with associated costs. Intuitively, such a constraint does not have to be satisfied, but if it is, then the price of the bid is modified by the given cost (usually a discount to the auctioneer).

For example, if a bidder wants to bid on $(\mathcal{I}^1, \mathcal{O}^1)$ and $(\mathcal{I}^2, \mathcal{O}^2)$ for price p and offer a discount, i.e., raise his bid by δ, if he gets to do the first before the second, then he could write

$$\begin{aligned}&\textsc{bid}(\{(\mathcal{I}^1, \mathcal{O}^1, \tau^1), (\mathcal{I}^2, \mathcal{O}^2, \tau^2)\}, p)\\ &(\tau^1 < \tau^2, \delta).\end{aligned}$$

Again, this expression can be translated:

$$\begin{aligned}&\textsc{bid}(\{(\mathcal{I}^1,\mathcal{O}^1,\vartheta^1),(\mathcal{I}^2,\mathcal{O}^2,\vartheta^2)\},p)\\&\quad\textsc{xor}\,\textsc{bid}(\{(\mathcal{I}^1,\mathcal{O}^1,\zeta^1),(\mathcal{I}^2,\mathcal{O}^2,\zeta^2)\},p+\delta)\\&\zeta^1<\zeta^2.\end{aligned}$$

In general, we can allow discounts depending on time constraint *formulas* rather than only on single constraints, and also the possibility to specify several *alternative* such discount options. Only one of these options should be applicable (possible combinations can be expressed as separate options), which is why we use XOR to denote the alternatives, analogously to the XOR of the bidding language. We thus consider a bid expression in XOR normal form together with a soft time constraint formula,

$$\bigXOR_{j=1}^{n}(\mathcal{D}_j,p_j)\qquad\qquad\bigXOR_{\iota=1}^{\nu}(\varphi_\iota,\delta_\iota),$$

where the φ_ι are time constraint formulas and the $\delta_\iota\in\mathbb{R}$ (possibly 0). Again, let σ_ι for $\iota\in\{1,\dots,\nu\}$ be substitutions, each mapping all variables occurring in the bid to fresh ones. The resulting translation then is

$$\bigXOR_{\iota=1}^{\nu}\bigXOR_{j=1}^{n}(\mathcal{D}_j\sigma_\iota,p_j+\delta_\iota)\qquad\qquad\bigwedge_{\iota=1}^{\nu}\varphi_\iota\sigma_\iota.$$

Note that the two translations we discussed are completely analogous. The difference is that the "exclusive" behavior of the bidding language XOR carries over to soft constraints in the sense that at most one discount will have an effect, while it does not carry over to disjunctive constraints, since multiple disjuncts may well end up being satisfied. Note also that the transformations can be combined. For example, a soft time constraint could have a disjunctive condition.

The *blowup* resulting from the transformations is straightforwardly seen to be linear in the number of disjuncts or of alternative discounts, respectively.

6.2.7 Expressive power

We say a valuation is **finite** if it has a finite domain (i.e., yields non-$\bot$ for finitely many time lines only) consisting of finite sequences of finite transformations (i.e., with finite input and output). Having seen how versatile time constraints really are, it may not be surprising that XOR bids with time constraints are **fully expressive** for finite valuations.

Concretely, XOR bids with relative time constraints can express all finite valuations that use relative time; XOR bids with absolute time constraints can express all finite valuations. To see this, simply take an XOR bid with one atomic bid $\textsc{bid}(\mathcal{D},p)$ for each Σ in the domain of v, with $\mathcal{D}$ set to permit Σ and p set

to $v(\Sigma)$, and impose the order corresponding to Σ using time constraints (note that there may be several atomic bids with the same transformations in $\mathcal{D}$, but different time points).

For a discussion of the expressive power of mixed-auction bidding languages (without time constraints but) with free disposal we refer to [37].

6.3 Winner determination

We now study the winner determination problem (WDP). This is the problem, faced by the auctioneer, of determining which transformations to award to which bidder, so as to maximize (minimize) the sum of payments collected (made), given the bids of the bidders expressed in our bidding language. This may be interpreted as computing a solution that maximizes revenue for the auctioneer, or social welfare for the collective of bidders (if we interpret prices offered as reflecting bidder utility). Note that we are interested in the *algorithmic* aspects of the WDP. Game-theoretical considerations, such as how to devise a more sophisticated pricing rule that would induce bidders to bid truthfully, are orthogonal to the algorithmic problem addressed here. (We briefly comment on mechanism design issues in Section 6.5, but this is not the topic of this chapter.)

For symmetry between bidders and auctioneer, we do *not* assume free disposal for the auctioneer (just like for the bidders), i.e., he does not want to end up with any goods except the required ones. Note, however, that the formulations are easily adapted to allow free disposal (and we point out the necessary changes along the way).

After formulating the WDP, we give an integer program [137] solving it and discuss some further topics. We aim at keeping the descriptions short and focus on the changes compared to the version from [37]. For more background and intuitive explanations, see there. The advantage of this approach, besides showing how few modifications are necessary and thereby how powerful the original framework already is, is that it is modular and can (hopefully) be combined without too much effort with other extensions or optimizations.

6.3.1 WDP with time constraints

The *input* to the WDP consists of

- a bid expression Bid_i in XOR normal form together with a conjunction of time constraints φ_i, for each bidder i;
- a multiset $\mathcal{U}_{in}$ of goods the auctioneer holds in the beginning;
- and a multiset $\mathcal{U}_{out}$ of goods the auctioneer wants to end up with.

Let Bid_{ij} denote the jth atomic bid $\text{BID}(\mathcal{D}_{ij}, p_{ij})$ occurring within Bid_i, let t_{ijk} be a unique label for the kth transformation in $\mathcal{D}_{ij}$ (for some arbitrary but fixed ordering of $\mathcal{D}_{ij}$), and let τ_{ijk} be the time point identifier associated with transformation t_{ijk}. Let $(\mathcal{I}_{ijk}, \mathcal{O}_{ijk})$ be the actual transformation labelled with t_{ijk}. Finally, let T be the set of all t_{ijk}.

An **allocation sequence** Σ resembles the time line we introduced before, but can only contain transformations actually offered by some bidder, and each one at most once. That is, Σ now is a permutation of a subset of T, possibly interspersed with clock ticks c.

We write $t_{ijk} \in \Sigma$ to say that the kth transformation in the jth atomic bid of bidder i has been selected, and we write $\Sigma(t_{ijk})$ to denote the sequence number of t_{ijk} (starting from 1) if $t_{ijk} \in \Sigma$.

By Σ_i we denote the **projection** of Σ to bidder i, that is, Σ with each t_{ijk} replaced by $(\mathcal{I}_{ijk}, \mathcal{O}_{ijk}, \tau_{ijk})$ and all $t_{i'jk}$ replaced by c for $i' \neq i$.

By $(\mathcal{I}^m, \mathcal{O}^m)$ we denote the mth transformation in Σ. Thus, we have two ways of referring to a selected transformation: by its position in the received bids (t_{ijk}) and by its position m in the allocation sequence.

Given Σ, we can inductively define the bundle of goods held by the auctioneer after each step (let $g \in G$ be any good, and let $\mathcal{M}^0 = \mathcal{U}_{in}$):[2]

$$\mathcal{M}^m(g) = \mathcal{M}^{m-1}(g) + \mathcal{O}^m(g) - \mathcal{I}^m(g) \tag{6.1}$$

under the condition that

$$\mathcal{M}^{m-1}(g) \geq \mathcal{I}^m(g). \tag{6.2}$$

Given a multiset $\mathcal{U}_{in}$ of goods available to the auctioneer, a multiset $\mathcal{U}_{out}$ of goods required by the auctioneer, and a set of bids Bid_i with time constraints φ_i, an allocation sequence Σ is a **valid solution** if:

(i) For each bidder i, some $\mathcal{D}_{ij}$ permits Σ_i, or $\Sigma_i \in \{c\}^*$.

(ii) For each bidder i, $\Sigma_i \vDash \varphi_i$.

(iii) Equations (6.1) and (6.2) hold for each transformation $(\mathcal{I}^m, \mathcal{O}^m) \in \Sigma$ and each good $g \in G$.

(iv) For each good $g \in G$, $\mathcal{M}^{|\Sigma|}(g) = \mathcal{U}_{out}(g)$.[3]

The **revenue** for the auctioneer associated with a valid solution Σ is the sum of the prices of the selected atomic bids:

$$\sum \{p_{ij} \mid \exists k : t_{ijk} \in \Sigma\}.$$

[2] Given a multiset $\mathcal{S} \in \mathbb{N}^G$ and an item $g \in G$, we write $\mathcal{S}(g)$ to denote the number of copies of g in $\mathcal{S}$.

[3] Replace $=$ by $\geq$ to model free disposal.

Given multisets $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ of initial and required goods and a set of bids with time constraints, the **winner determination problem** (WDP) consists in finding a valid solution that maximizes the auctioneer's revenue.

We now show how to solve the WDP using integer programming (IP). To this end, we first review the original formulation from [37] and then discuss the modifications required for dealing with time constraints.

6.3.2 Original integer program

In this part, we closely follow [37]. The main issue is to decide, for each offered transformation, whether it should be selected for the solution sequence, and if so, at which position. Thus, we define a set of binary decision variables $x^m_{ijk} \in \{0, 1\}$, each of which takes on value 1 if and only if the transformation t_{ijk} is selected at the mth position of the solution sequence.

The position number m ranges from 1 to an upper bound M on the solution sequence length. For the time being, we take $M = |T|$, the overall number of transformations, accommodating all sequences that can be formed using only transformations (and not clock ticks).

Further, i ranges over all bidders; j ranges for each bidder i from 1 to the number of atomic bids submitted by i; and k ranges for each atomic bid j of bidder i from 1 to the number of transformations in that bid.

We use the following auxiliary binary decision variables: x^m takes on value 1 if and only if any transformation is selected at the mth position; x_{ijk} takes on value 1 if and only if transformation t_{ijk} is selected at all; and x_{ij} takes on value 1 if and only if any of the transformations in the jth atomic bid of bidder i are selected.

The following set of constraints define a valid solution *without* taking time constraints into account (i.e., neglecting (ii) in the valid solution definition above):

(1) Select either all or no transformations from an atomic bid (cf. (i) above):

$$x_{ij} = x_{ijk} \quad (\forall ijk)$$

(2) Select at most one atomic bid from each XOR normal form bid (cf. (i) above):

$$\sum_j x_{ij} \leq 1 \quad (\forall i)$$

(3) Select each transformation at most for one position:

$$x_{ijk} = \sum_m x^m_{ijk} \quad (\forall ijk)$$

(4) For each position, select at most one transformation:

$$x^m = \sum_{ijk} x^m_{ijk} \quad (\forall m)$$

(5) There should be no gaps in the sequence:

$$x^m \geq x^{m+1} \quad (\forall m)$$

Note that this is strictly speaking not required; indeed we drop this constraint later on in order to allow clock ticks between transformations.

(6) Treating each $\mathcal{M}^m(g)$ as an integer decision variable, ensure that necessary input goods are available (cf. (iii) above):

$$\mathcal{M}^m(g) = \mathcal{U}_{in}(g) + \sum_{\ell=1}^{m} \sum_{ijk} x^\ell_{ijk} \cdot (\mathcal{O}_{ijk}(g) - \mathcal{I}_{ijk}(g))$$

$$\mathcal{M}^m(g) \geq \sum_{ijk} x^m_{ijk} \cdot \mathcal{I}_{ijk}(g) \qquad (\forall g \in G, \forall m)$$

(7) In the end, the auctioneer should have the bundle $\mathcal{U}_{out}$ (cf. (iv) above):[4]

$$\mathcal{M}^M(g) = \mathcal{U}_{out}(g) \quad (\forall g \in G)$$

Solving the WDP now amounts to solving the following integer program:

$$\max \sum_{ij} x_{ij} \cdot p_{ij}, \quad \text{subject to constraints (1)–(7)}$$

A valid solution is then obtained by making transformation t_{ijk} the mth element of the solution sequence Σ exactly when $x^m_{ijk} = 1$.

6.3.3 Modified integer program

To implement time constraint handling (thus obeying (ii) in the definition of valid solution given above), we first introduce an additional set of auxiliary binary decision variables $y^m_{ijk} \in \{0, 1\}$, taking on value 1 if and only if transformation t_{ijk} is selected at the mth position *or earlier* in the solution sequence. This can be achieved by adding the following constraint:

(8) y^m_{ijk} should be 1 iff $t_{ijk} \in \Sigma$ and $\Sigma(t_{ijk}) \leq m$:

$$y^m_{ijk} = y^{m-1}_{ijk} + x^m_{ijk} \quad (\forall ijkm),$$

with $y^0_{ijk} = 0$.

We now give implementations for our two variants of time constraints.

[4] With free disposal, $=$ would become $\geq$.

Relative time. Each bidder i's time constraint formula is a conjunction of atomic time constraints, and all bidders' time constraints need to be satisfied. The following set of integer constraints takes care of this.

(9a) For each $\tau_{ijk} < \tau_{ij'k'}$ occurring in $\bigwedge_i \varphi_i$:

$$y^m_{ijk} \geq y^{m+1}_{ij'k'} \quad (\forall m).$$

In accordance with the time constraint semantics, if neither t_{ijk} nor $t_{ij'k'}$ occurs in the solution sequence, this requirement is vacuously satisfied since both sides stay 0. If $t_{ij'k'}$ *does* occur, then $y^m_{ij'k'}$ will become 1 at some point m. In this case, the requirement boils down to y^{m-1}_{ijk} being 1 as well, so t_{ijk} must have occurred already.

Solving the WDP with relative time constraints thus amounts to the same optimization as before, but subject to constraints (1)–(8) and (9a).

Absolute time. In order to have an absolute notion of time, we need some way of mapping points of a possible solution sequence to an absolute time line. The simplest way is to interpret each sequence point itself as a time unit (a minute, a day, a week, ...), and this is the approach we take.

Before giving the formalization, we need to discuss some conceptual details. If we interpret steps in the sequence as absolute time units, some issues arise which did not matter before.

Firstly, while it may be acceptable to break time down into discrete steps of equal duration, it is not so easy to defend that any transformation that can possibly be offered should have exactly that duration.

Secondly, there is no reason why the auctioneer should wait for one transformation to end before commissioning the next transformation, which may be offered by a different, idling bidder, unless the output of the former is needed as input to the latter.

To some extent, these issues can be addressed by a purely conceptual extension presented in Section 6.4. However, we leave it to future work to design frameworks which handle time in a more flexible way and truly optimize for effective parallelizations (which is a research field on its own). For our purposes, we simply assume that the auctioneer is busy when he is delivering or receiving goods of some particular transformation, and cannot deal with several bidders simultaneously.

To start the formalization, first of all we drop constraint (5). As mentioned, it is not strictly speaking necessary anyway, and since now the bidders can refer to arbitrary absolute time points, we actually might have to accept gaps in the sequence.

Now a technical issue arises: The length of possible solution sequences is no longer bounded by $|T|$. While it may be possible to find a correct bound by looking at all numbers occurring in the bidders' time constraints, we settle for a

different solution: The auctioneer manually specifies M, the maximum length of the solution sequence.

At first glance this seems like a pure loss of generality; however the auctioneer may profit from having some control over the size of the WDP he has to solve, and he can always iterate over different values for M in his search for a good solution. Economically speaking, it also makes sense that the auctioneer wants some control over the length of his supply chain, rather than allowing an arbitrary length. Indeed, he might have graded preferences over the time his supply chain takes; in Section 6.3.4 we show how he can accomplish this, turning the ostensible loss of generality into a feature.

We now give the integer constraints for handling absolute time constraints.

(9b) For each $\tau_{ijk} + \xi < \tau_{ij'k'} + \xi'$ occurring in $\bigwedge_i \varphi_i$:

$$y_{ijk}^{m+\xi'} \geq y_{ij'k'}^{m+\xi+1} \quad (\forall m);$$

for each $\tau_{ijk} + \xi = \tau_{ij'k'} + \xi'$ occurring in $\bigwedge_i \varphi_i$:

$$y_{ijk}^{m+\xi'} = y_{ij'k'}^{m+\xi} \quad (\forall m)$$

(10) For each $\tau_{ijk} \circ \xi$, with $\circ \in \{=, <, >\}$, occurring in $\bigwedge_i \varphi_i$:

$$x_{ijk}^{m} = 0 \quad (\forall m \not\circ \xi).$$

Constraint (9b) requires some explanation. First of all, note that (9a), the version for relative time, is covered as a special case. As indicated by the semantics, the absolute time variant is thus an extension of the relative time variant. Secondly, note that the second half of (9b) can be obtained from the first half if interpreted as an abbreviation, as in Section 6.2.5. Now consider the case where $\xi' = 0$. Intuitively speaking, the time constraint then says that t_{ijk} must take place at least $\xi + 1$ time steps before $t_{ij'k'}$. That is, whenever $t_{ij'k'}$ is selected, t_{ijk} must already have been selected for at least $\xi + 1$ time steps. In terms of the integer program, this means that, for all positions m, $y_{ij'k'}^{m}$ must be 0 *unless* $y_{ijk}^{m-\xi-1}$ was already 1. Now it is only a small step to the formulation in (9b).

Solving the WDP with absolute time constraints amounts to the same optimization as before, but subject to constraints (1)–(4), (6)–(8), (9b) and (10).

A valid solution is then obtained by making transformation t_{ijk} the mth element of the solution sequence Σ if and only if $x_{ijk}^{m} = 1$, and using a clock tick c as mth element when there is no x_{ijk}^{m} which equals 1 (i.e., when $x^m = 0$).

6.3.4 Valuation for the auctioneer

Given that we decided to require the auctioneer to specify the maximum length M of the solution sequence (for the absolute-time variant of the framework), we

may also want to give him the possibility to express more detailed preferences over durations. This turns out to be achievable in a neat way, also enabling the auctioneer to express graded preferences over the final bundles.

So let us assume that the auctioneer derives a certain value from a given supply chain, depending on its overall duration and on its outcome, the bundle of goods he owns in the end. Note that this discussion assumes absolute time; with relative time, preferences over *durations* do not make much sense, but the results can easily be adjusted to only model preferences over *outcomes*.

We thus assume the auctioneer's valuation is a function

$$u : \mathbb{N} \times \mathbb{N}^G \to \mathbb{R} \cup \{\bot\},$$

mapping duration/outcome pairs to a value or $\bot$, meaning the duration/outcome pair is not acceptable. This valuation can be incorporated into the WDP in the following way.

After receiving the bids, the auctioneer decides on a maximum duration M and creates an additional bid under an unused bidder identity:

$$\operatorname*{XOR}_{\{(m,\mathcal{U}) \mid u(m,\mathcal{U}) \neq \bot\}} \text{BID}(\{(\mathcal{U}, \{\circledcirc\}, \tau_{m,\mathcal{U}})\}, u(m,\mathcal{U})),$$

where $\circledcirc$ is a special token that does not occur as a good in any other bid, together with time constraints:

$$\bigwedge_{\{(m,\mathcal{U}) \mid u(m,\mathcal{U}) \neq \bot\}} \tau_{m,\mathcal{U}} = m.$$

The transformations in this bid are to be thought of as terminal transformations: they denote the possible time points and outcomes at which a solution sequence may end, and the associated values for the auctioneer. The idea is that using this method, the auctioneer's valuation can be expressed with almost no change to the integer program.

Let us look at the requirements needed for the auctioneer's valuation to work.

- The terminal transformations should only be used at the respective intended positions in the sequence; this is ensured by the given time constraints.
- At *most* one of them should be used; this is ensured by the XOR (and strictly speaking also follows from the last point below).[5]
- At *least* one[6] of them should be used; this can be ensured by setting $\mathcal{U}_{out} = \{\circledcirc\}$.

[5]Even more strictly speaking, it also follows from the next requirement and the fact that we assume no free disposal; we include it nevertheless for conceptual clarity and in order to accommodate a possible free disposal assumption.

[6]This could read "*exactly* one", but again, we want to accommodate a possible free disposal assumption.

- The unique terminal transformation which is used should indeed constitute the end of the solution sequence.[7]

For this last point, we need an additional integer constraint:

(11) No transformations are scheduled after a terminal transformation:

$$x_{ijk}^{m+1} \leq 1 - y_{-1j'k'}^{m} \ (\forall ijkj'k'm)$$

(-1 being the auctioneer's "bidder identity").

While the remaining requirements could also be encoded more directly and more efficiently into the integer program, for clarity we here restrict ourselves to this version using the high-level features of the bidding language.

Many further extensions and optimizations along these lines are conceivable. We do not try to exhaust them here, but sketch only one example. The auctioneer might want to extract some goods $\mathcal{U}$ from the supply chain by some intermediate time point ξ, not necessarily at its end. To express this, he can add a transformation $(\mathcal{U}, \{\diamond\}, \tau)$ with time constraint $\tau < \xi + 1$ to his bid, and add $\diamond$ to $\mathcal{U}_{out}$. Dropping constraint (11) for this transformation, he makes it non-terminal. He can also make this a soft requirement by including another transformation that yields $\diamond$ from no input and attaching appropriate prices to the corresponding bids.

6.3.5 Computational complexity

The (decision problem underlying the) WDP for mixed auctions with time constraints is NP-complete. NP-hardness follows from NP-hardness of the WDP for standard combinatorial auctions [127]. NP-membership follows from the fact that the validity of a given allocation sequence can clearly be verified in polynomial time. That is, in terms of (abstract) computational complexity, the integration of time constraints does not have too much of an impact when compared to the original mixed-auction model [37].

This is also the reason why time constraints seem to fall into place so easily in our case, which may be somewhat unexpected given the amount of research dedicated to handling them (see, e.g., [138]): Most of those research efforts are directed towards tractability, which in our context does not have such a high priority since even without any optimizations, time constraints do not increase the complexity already inherent in the underlying framework.

Consequently, not much has changed with respect to the complexity of the integer programming formulation. While there is room for optimizations, the number of variables we introduce is of the same order as in the original formulation: $O(n^2)$, where n is the number of transformations occurring in the bids submitted.

[7] As a last remark, this requirement could be dropped if we *did* assume free disposal and all bids' prices were positive.

The most recent work on winner determination algorithms for mixed auctions has tried to reduce the number of decision variables needed so as to improve performance [67, 107]. Due to the modular nature of our approach, we are optimistic that it will be possible to take advantage of these optimizations and integrate them with the extensions for handling time constraints presented here. In this chapter, our focus has been on presenting the basic model of mixed auctions with time constraints, and on demonstrating the feasibility of the approach.

6.4 Intervals

As discussed before, it is desirable to allow transformations to overlap or take place during other transformations, and to allow transformations to have different durations (meaningful mostly in the variant with absolute time).

One step towards this goal, which can be expressed in our framework without any modifications, is to allow for a transformation to specify an **interval** during which it takes place, rather than only a single time point identifier.

Interval handling is pure syntactic sugar in our framework. To represent a transformation that takes place during an interval, we include two time point identifiers: start time and end time. Internally, intervals are reduced to two transformations with single time points and an appropriate time constraint:

$$(\mathcal{I}, \mathcal{O}, [\tau, \tau']) \quad \rightsquigarrow \quad \begin{array}{l} (\mathcal{I}, \emptyset, \tau), (\emptyset, \mathcal{O}, \tau') \\ \tau < \tau' \end{array}$$

Since the replacement takes place within a single atomic bid, it is guaranteed that either both the start and end transformations will be selected, or neither. That is, the interval transformation remains intact.

The usual interval relations (see the interval calculus by Allen [2]; due to sequentiality we consider only the strict relations) can be defined as macros yielding standard time constraints:

$$\begin{aligned} [\tau_1, \tau_1'] \text{ BEFORE } [\tau_2, \tau_2'] &\rightsquigarrow \tau_1' < \tau_2 \\ [\tau_1, \tau_1'] \text{ OVERLAPS } [\tau_2, \tau_2'] &\rightsquigarrow \tau_1 < \tau_2 \wedge \tau_1' < \tau_2' \\ [\tau_1, \tau_1'] \text{ DURING } [\tau_2, \tau_2'] &\rightsquigarrow \tau_2 < \tau_1 \wedge \tau_1' < \tau_2 \end{aligned}$$

With absolute time, absolute restrictions on the durations can also be implemented:

$$\begin{aligned} \mathit{duration}([\tau, \tau']) > \xi &\rightsquigarrow \tau + \xi < \tau' \\ \mathit{duration}([\tau, \tau']) < \xi &\rightsquigarrow \tau' < \tau + \xi \\ \mathit{duration}([\tau, \tau']) = \xi &\rightsquigarrow \tau' = \tau + \xi \end{aligned}$$

Note that expressions like $\mathit{duration}(\cdot) > \mathit{duration}(\cdot)$ are not so straightforwardly expressible in our framework, but arguably also much less useful in the context of specifying bids.

6.5 Conclusions and related work

We have presented an extension to the existing framework of mixed multi-unit combinatorial auctions [37], enabling bidders to impose time constraints on the transformations they offer.

In the original framework, the auctioneer is free to schedule the offered transformations in any way suitable to achieve his desired outcome, while bidders are left with no control over this process. Our work redresses this asymmetry, thus representing an important step towards a more realistic model of supply chain formation, where bidders themselves may have supply chains or other factors restricting the possible schedules for performing certain transformations.

Starting from a very basic core language for expressing time constraints, we have given various extensions, many purely syntactic, showing the somewhat unexpected power inherent to the core language.

We have also extended the integer program given in [37] to handle time constraints. Our extensions are modular in a way that will facilitate combining them with other extensions and optimizations for mixed auctions, such as [67, 107].

6.5.1 Related work

Time constraints have been applied to different types of combinatorial auctions in the literature. For example, Hunsberger and Grosz [81] extend an existing algorithm for winner determination in combinatorial auctions to allow precedence constraints when bidding on roles in a prescribed action plan ("recipe"). Collins [42] permits relative time constraints in a combinatorial reverse auction over combinations of tasks, and the efficiency of various approaches to solving the winner determination problem is tested.

Auction frameworks involving time have also been fruitfully applied to problems of distributed scheduling. In the work of Wellman et al. [152], time constraints do not enter separately, but rather time slots are the actual objects being auctioned, and game-theoretic properties and mechanism design issues are discussed.

While it would be interesting to examine whether the insights about efficiency and alternative approaches to handling time could be applied to our framework, the roles, tasks, and time slots being auctioned in those contributions are "atomic", and the formulations and results do not easily translate to transformations in the context of mixed auctions.

6.5.2 Possible extensions

Concerning mechanism design, the remarks by Cerquides et al. [37] still apply. The bottom line is that, with finite valuations, the incentive-compatibility of the Vickrey-Clarke-Groves (VCG) mechanism carries over from standard combinatorial auctions to mixed multi-unit combinatorial auctions with time constraints. It

is a question of independent interest, whether and how this can be extended to non-finite valuations when still allowing only finite bids.

Other topics for future work include the exact interplay between the various syntactic extensions we have given, defining a uniform general language, and determining whether some of the features can be implemented in more direct (and efficient) ways than through the translation to the core language used in this work. The same holds for the underlying bidding language, where operators such as OR may be executed more efficiently than through translation to XOR. Finally, an empirical analysis needs to be performed, including testing and optimizing our integer program.

Chapter 7

Outlook

One main topic of this dissertation has been what we called *explicit knowledge programming*: giving artificial agents access to the knowledge one can theoretically ascribe to them, by way of providing epistemic statements in a programming language, made feasible by considering scenarios with well-defined restrictions. We believe that this approach is worth exploring in a more systematic way.

As illustrated in Chapter 4, simulations of virtual worlds, and in particular computer games, form a rich playground and testbed for formalisms and implementations. In the artificial intelligence (AI) community, this has been realized, among others, by Laird and van Lent [89]. Such environments therefore are a natural starting point for exploring a practical perspective on reasoning about knowledge in interaction, about which we give some thoughts in this chapter.

We start by briefly describing a logic framework which seems highly suitable for this endeavour.

Dynamic Epistemic Logic (DEL). DEL is a sophisticated and intuitive formalism for reasoning about epistemic situations and change. It was initiated by Gerbrandy and Groeneveld [66] and Baltag et al. [14]. The book by van Ditmarsch et al. [52] provides a recent overview.

The fundamental idea of DEL is to add a dynamic aspect to epistemic logic, which can be represented on the level of models. Besides the (main) model describing a given situation as illustrated in the Introduction chapter, DEL has *action models* describing actions (or *events*).

An action model has a similar structure as the main model, but its nodes represent possible actions, including an actual action, and its edges represent the agents' uncertainties with respect to these possible actions. For example, if Ann sees Bob showing the ace of spades to Eve, but Ann cannot see which card it is, she will also consider it possible that he is showing her the queen of hearts (or any other card).

An action model can be applied to a model using a product update operation

which yields a new model reflecting the situation after the corresponding action has occurred. Actions which not only affect the epistemic situation but change actual facts in the world have been examined [50], but the more widely studied purely epistemic actions already suffice for many interesting scenarios, for example knowledge games as described in Chapter 4.

Using DEL in simulated worlds. One way to employ DEL (or logic systems in general) is through its *axioms*. These fundamental truths and rules can be used to deduce any valid formula. A possible approach then is to combine a virtual world that uses deductive planning, such as the one by Magnusson and Doherty [95], with axioms for epistemic reasoning. In this way, the planning agents would be enabled to reason about the epistemic preconditions and consequences of their actions and the epistemic parts of their goals.

However, as throughout this dissertation, we here focus on a model-based viewpoint. In the following, we describe some issues and possibilities on the way towards realistic implementations.

There are a number of reasons why we believe that DEL, and in particular a model-based approach, is very suitable for introducing epistemic reasoning to computer-simulated worlds:

- A centralized model comprising all agents lends itself naturally to simulated worlds, where there already is a central controller providing agents with information such as current vision and other physical senses.[1]

- Importantly for real-time applications, a model-based approach can allow for more efficient evaluation of epistemic formulas than a deduction-based approach.

- The DEL model is updated as events occur to reflect the current state of affairs. It does not retain information about past states, as other, for example history-based, frameworks do; this keeps the model manageable.

- Focusing on the model, rather than the axioms, is more intuitive (in particular in the case of DEL), which makes it easier to find suitable optimizations and restrictions of the logic. A model-based implementation can also be more straightforward to inspect and debug than a heap of formulas and deductions.

- The data structures for DEL models and the operations for updates are straightforwardly represented in any programming language.

[1] However, the centralized viewpoint is not vital to our aim of implementing DEL, and any results and techniques obtained for a centralized model are likely transferable to autonomous agents with internal epistemic models, due to close correspondences between these viewpoints [11].

As illustrated in Chapter 4, artificial agents acting in a simulated world can then, for example, use epistemic conditions to branch over alternative program paths. Whenever an epistemic formula is encountered, the *knowledge module*, which maintains the DEL model, is queried to evaluate it. Thus, agents gain access to their knowledge and beliefs just like they receive information about their physical senses from the central simulation controller.

Issues. The biggest problem of a model-based approach is that the model to be maintained can be very large. The number of possible configurations of atoms is exponential in the number of atoms, and in order to keep track of higher-order knowledge, the same configuration of atoms may even need to be embodied in several states. In DEL, update operations exacerbate this problem further: the product operation *prima facie* increases the size of the model exponentially.

Correspondingly, implementations of DEL are rare. There exists one generic implementation of DEL models and operations (DEMO, [54]). However, without precautions the models quickly become too large to handle, and it is unclear to what extent realistic scenarios stay manageable in the long run.

Real-time simulation environments, having to simulate concurrently all parts and aspects of the world, make efficiency of both representation and processing a top priority, especially in computer games aimed at customary home computers.

Thus, the crucial point is to find compact ways of representing models, examine how exactly updates affect the size of the model in the long run, and find conditions on actions and epistemic formulas under which an implementation becomes feasible. In order to obtain results that are relevant for general applications, this needs to be done in a more generic and systematic way than what we have done in Chapters 3 and 4.

In the following, we discuss some starting points for tackling these issues.

Representation and evaluation of models. Besides ensuring that the maintained model is always the smallest of all equivalent ones,[2] the model needs to be represented in a way that can be stored efficiently. This is one of the central points in the area of model checking [41], and that area can thus serve a starting point for finding techniques for compact representation and efficient evaluation.

However, DEL brings an additional twist in the form of update operations. These should operate directly on the compact representation of the model, without having to blow up the model in some intermediate steps. Since this adds a requirement to the representation techniques, they need to be revisited and adapted where necessary.

Restrictions on events. One way to avoid uncontrolled exponential growth of the model as events occur is to impose restrictions on the classes of events that

[2] This is the so-called *bisimulation contraction*.

will be taken into account in the simulation. For example, the events considered in Chapter 4 have an effect only the first time they occur. This can be seen by noting that the order in which different events occur does not matter, and that any event occurring twice in a row has the same effect as only one occurrence.

Along similar lines, more general classes of events which are commutative and idempotent in an adequate sense may be of relevance. It may also be possible to identify other technical conditions which make updates "well-behaved" in the long run. It is then necessary to examine what these conditions imply on the logical level, what configurations of knowledge they give rise to, and whether the resulting classes of events are useful in practice.

In the DEL community there is increasing interest in issues such as what events are equivalent, how updates behave in the long run, and whether models ultimately stabilize. This line of research, initiated by Ruan [128], Sadzik [131], and van Eijck et al. [55], is closely related to the aim of finding classes of events that are suitable for implementations.

Restrictions on epistemic formulas. Naturally, the class of epistemic formulas that are to be evaluated by the knowledge module has a direct impact on its efficiency. Imposing restrictions, for example, on the nesting depths of knowledge and belief operators allows to bound the evaluation time of formulas, and to prune down the model: information that is never going to be queried does not need to be modeled. Note also that there is some interplay between the different kinds of restrictions. For example, similarly to what we have seen in Chapter 2, certain restrictions on initial situations and events make certain formulas equivalent, since models discriminating them will never occur. In this sense, there are corresponding restrictions of events and formulas.

Here, too, the update operation should be specialized to these restrictions and operate directly on pruned models. That is, rather than generating a temporary model and pruning it down, it should, if possible, avoid generating unnecessary information in the first place.

In the proposed context of virtual worlds, the ultimate goal is to maximize efficiency while retaining realism, or enjoyability in the case of computer games. Therefore, as described in Chapter 4, Section 4.6.3, insights about human cognition need to be used in order to find suitable epistemic languages. Conversely, technical restrictions that allow for efficient implementations can be examined on a logical level to find possible application scenarios.

Conclusion. Overall, we believe that an implementation-oriented perspective can provide new and useful stimuli for future directions of the mentioned lines of research.

Bibliography

[1] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber. Some constraints and tradeoffs in the design of network communications. In *Proceedings of the fifth ACM symposium on Operating systems principles (SOSP-5)*, pages 67–74, Austin, Texas, 1975. ACM. Cited on p. 3.

[2] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. Cited on p. 133.

[3] Gregory R. Andrews. *Concurrent Programming: Principles and Practice*. Addison Wesley, 1991. Cited on p. 14.

[4] Krzysztof R. Apt. Relative strength of strategy elimination procedures. *Economics Bulletin*, 3(21):1–9, 2007. Cited on p. 57.

[5] Krzysztof R. Apt. The many faces of rationalizability. *The B.E. Journal of Theoretical Economics*, 7(1):Article 18, 2007. Cited on pp. 54 and 55.

[6] Krzysztof R. Apt. Uniform proofs of order independence for various strategy elimination procedures. *The B.E. Journal of Theoretical Economics*, 4(1): Article 5, 2004. Cited on p. 96.

[7] Krzysztof R. Apt and Tadeusz Radzik. Stable partitions in coalitional games. arXiv.org, http://arxiv.org/abs/cs.GT/0605132, 2006. Cited on pp. 96, 104, 106, 108, 109, and 114.

[8] Krzysztof R. Apt and Andreas Witzel. A generic approach to coalition formation. *International Game Theory Review*, 2009. To appear. Cited on p. 11.

[9] Krzysztof R. Apt, Andreas Witzel, and Jonathan A. Zvesper. Common knowledge in interaction structures. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge (TARK XII)*, 2009. To appear. Cited on pp. 10, 37, 38, 40, 48, and 75.

[10] Itai Ashlagi, Dov Monderer, and Moshe Tennenholtz. Resource selection games with unknown number of players. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 819–825, Hakodate, Japan, 2006. ACM Press. Cited on p. 53.

[11] Guillaume Aucher. *Perspectives on Belief and Change*. PhD thesis, Université de Toulouse and University of Otago, 2008. Cited on pp. 81 and 138.

[12] Robert J. Aumann. Agreeing to disagree. *The Annals of Statistics*, 4(6): 1236–1239, 1976. Cited on pp. 4 and 13.

[13] Robert J. Aumann and Jacques H. Drèze. Cooperative games with coalition structures. *International Journal of Game Theory*, 3:217–237, 1974. Cited on p. 95.

[14] Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. The logic of public announcements, common knowledge, and private suspicions. In Itzhak Gilboa, editor, *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98)*, pages 43–56, Evanston, Illinois, 1998. Morgan Kaufmann. Cited on p. 137.

[15] Simon Baron-Cohen. *Mindblindness*. The MIT Press, 1995. Cited on p. 78.

[16] Pierpaolo Battigalli and Giacomo Bonanno. Recent results on belief, knowledge and the epistemic foundations of game theory. *Research in Economics*, 53(2):149–225, 1999. Cited on p. 51.

[17] Johan van Benthem. Logical dynamics of information and interaction. Book in preparation, 2009. Cited on p. 1.

[18] Johan van Benthem. Logic in games. Book in preparation, 2009. Cited on p. 4.

[19] Johan van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17(2):129–155, 2007. Cited on p. 92.

[20] Johan van Benthem. One is a lonely number: Logic and communication. In Zoé Chatzidakis, Peter Koepke, and Wolfram Pohlers, editors, *Logic Colloquium 2002*, volume 27 of *Lecture Notes in Logic*, pages 96–129. A K Peters, 2006. Cited on pp. 6 and 47.

[21] Johan van Benthem, Jelle Gerbrandy, and Eric Pacuit. Merging frameworks for interaction: DEL and ETL. In Dov Samet, editor, *Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge (TARK XI)*, pages 72–81, Brussels, Belgium, 2007. ACM. Cited on p. 48.

[22] Arthur Bernstein. Output guards and nondeterminism in "Communicating Sequential Processes". *ACM Transactions on Programming Languages and Systems*, 2(2):234–238, 1980. Cited on p. 35.

[23] Bethesda Softworks. The Elder Scrolls: Oblivion. http://www.elderscrolls.com/games/oblivion_overview.htm, 2006. Cited on p. 81.

[24] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001. Cited on p. 88.

[25] Francis Bloch and Matthew Jackson. Definitions of equilibrium in network formation games. *International Journal of Game Theory*, 34(3):305–318, 2006. Cited on p. 96.

[26] Anna Bogomolnaia and Matthew O. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002. Cited on p. 96.

[27] Rafael H. Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah Seghrouchni, Jorge J. Gomez-Sanz, Joao Leite, Gregory O'Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for Multi-Agent systems. *Informatica*, 30(1):33–44, 2006. Cited on p. 83.

[28] Tilman Börgers. Weak dominance and approximate common knowledge. *Journal of Economic Theory*, 64(1):265–276, 1994. Cited on p. 8.

[29] Luc Bougé. On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. *Acta Informatica*, 25(2): 179–201, 1988. Cited on pp. 15, 16, 21, 22, 28, 29, 30, and 35.

[30] Adam Brandenburger. Knowledge and equilibrium in games. *The Journal of Economic Perspectives*, 6(4):83–101, 1992. Cited on p. 8.

[31] Adam Brandenburger. The power of paradox: some recent developments in interactive epistemology. *International Journal of Game Theory*, 35(4): 465–492, 2007. Cited on p. 77.

[32] Adam Brandenburger, Amanda Friedenberg, and H. Jerome Keisler. Fixed points for strong and weak dominance, 2006. Working paper. Cited on p. 55.

[33] Gael N. Buckley and Abraham Silberschatz. An effective implementation for the generalized Input-Output construct of CSP. *ACM Transactions on Programming Languages and Systems*, 5(2):223–235, 1983. Cited on pp. 15 and 35.

[34] Nils Bulling and Koen V. Hindriks. Communicating rational agents: Semantics and verification. Technical Report IfI-09-02, Institut für Informatik, Technische Universität Clausthal, 2009. Cited on p. 83.

[35] Nadia Burani and William S. Zwicker. Coalition formation games with separable preferences. *Mathematical Social Sciences*, 45(1):27–52, 2003. Cited on p. 96.

[36] Colin F. Camerer. Behavioural studies of strategic thinking in games. *Trends in Cognitive Sciences*, 7(5):225–231, 2003. Cited on p. 92.

[37] Jesús Cerquides, Ulle Endriss, Andrea Giovannucci, and Juan A. Rodríguez-Aguilar. Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In Manuela Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 1221–1226, Hyderabad, India, 2007. Cited on pp. 115, 116, 117, 120, 122, 125, 127, 132, and 134.

[38] Christophe P. Chamley. *Rational herds: Economic models of social learning*. Cambridge University Press, 2004. Cited on pp. 6 and 52.

[39] K. Mani Chandy and Jayadev Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986. Cited on pp. 3 and 47.

[40] Yi-Chun Chen, Ngo Van Long, and Xiao Luo. Iterated strict dominance in general games. *Games and Economic Behavior*, 61(2):299–315, 2007. Cited on p. 54.

[41] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999. Cited on pp. 91 and 139.

[42] John Edgar Collins. *Solving combinatorial auctions with temporal constraints in economic agents*. PhD thesis, University of Minnesota, 2002. Cited on p. 134.

[43] Peter Cramton, Yoav Shoham, and Richard Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006. Cited on pp. 6 and 115.

[44] Vincent P. Crawford and Joel Sobel. Strategic information transmission. *Econometrica*, 50(6):1431–1451, 1982. Cited on p. 52.

[45] Maria Cutumisu, Curtis Onuczko, Matthew McNaughton, Thomas Roy, Jonathan Schaeffer, Allan Schumacher, Jeff Siegel, Duane Szafron, Kevin Waugh, Mike Carbonaro, Harvey Duff, and Stephanie Gillis. ScriptEase: a generative/adaptive programming paradigm for game scripting. *Science of Computer Programming*, 67(1):32–58, 2007. Cited on p. 83.

[46] Eddie Dekel and Drew Fudenberg. Rational behavior with payoff uncertainty. *Journal of Economic Theory*, 52(2):243–267, 1990. Cited on p. 5.

[47] Gabrielle Demange. On group stability in hierarchies and networks. *Journal of Political Economy*, 112(4):754–778, 2004. Cited on p. 96.

[48] Gabrielle Demange and Myrna Wooders. *Group Formation in Economics: Networks, Clubs, and Coalitions*. Cambridge University Press, 2005. Cited on p. 96.

[49] Hans van Ditmarsch. *Knowledge Games*. PhD thesis, Groningen University, 2000. Cited on pp. 77 and 78.

[50] Hans van Ditmarsch and Barteld Kooi. Semantic results for ontic and epistemic change. In Giacomo Bonanno, Wiebe van der Hoek, and Michael Wooldridge, editors, *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, volume 3 of *Texts in Logic and Games*, pages 87–117, Amsterdam, 2008. Amsterdam University Press. Cited on p. 138.

[51] Hans van Ditmarsch and Willem Labuschagne. My beliefs about your beliefs: a case study in theory of mind and epistemic logic. *Synthese*, 155(2):191–209, 2007. Cited on p. 92.

[52] Hans van Ditmarsch, Wiebe Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Springer, 2007. Cited on pp. 91 and 137.

[53] Aldo Franco Dragoni, Paolo Giorgini, and Luciano Serafini. Mental states recognition from communication. *Journal of Logic and Computation*, 12(1): 119–136, 2002. Cited on p. 83.

[54] Jan van Eijck. DEMO—a demo of epistemic modelling. In Johan van Benthem, Benedikt Löwe, and Dov Gabbay, editors, *Interactive Logic: Selected Papers from the 7th Augustus de Morgan Workshop, London*. Amsterdam University Press, 2008. Cited on p. 139.

[55] Jan van Eijck, Ji Ruan, and Tomasz Sadzik. Action emulation. Draft paper, 2008. Cited on p. 140.

[56] Eidos Interactive. Thief: The Dark Project. http://www.eidosinteractive.com/games/info.html?gmid=34, 1998. Cited on p. 84.

[57] Electronic Arts Inc. Burnout Paradise. http://burnout.ea.com, 2008. Cited on p. 93.

[58] Eric Eve. Epistemology, Version 4. *Extension for Inform 7: A Design System for Interactive Fiction Based on Natural Language.* http://www.inform-fiction.org/I7Downloads/Extensions/Eric%20Eve/Epistemology, 2007. Cited on p. 82.

[59] Ronald Fagin and Joseph Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1987. Cited on pp. 7 and 73.

[60] Ronald Fagin, Joseph Y. Halpern, Moshe Y. Vardi, and Yoram Moses. *Reasoning about knowledge.* MIT Press, 1995. Cited on pp. 1, 3, 9, 14, 38, 40, and 88.

[61] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, 1997. Cited on pp. 9 and 74.

[62] Joseph Farrell and Matthew Rabin. Cheap talk. *The Journal of Economic Perspectives*, 10(3):103–118, 1996. Cited on p. 52.

[63] Faith Fich and Eric Ruppert. Hundreds of impossibility results for distributed computing. *Distributed Computing*, 16(2):121–163, 2003. Cited on pp. 15 and 16.

[64] Robert E. Filman and Daniel P. Friedman. *Coordinated Computing: Tools and Techniques for Distributed Software.* McGraw-Hill, Inc, 1984. Cited on p. 35.

[65] Jelle Gerbrandy. Communication strategies in games. *Journal of Applied Non-Classical Logics*, 17(2):197–211, 2007. Cited on p. 52.

[66] Jelle Gerbrandy and Willem Groeneveld. Reasoning about information change. *Journal of Logic, Language and Information*, 6(2):147–169, 1997. Cited on p. 137.

[67] Andrea Giovannucci, Meritxell Vinyals, Juan A. Rodríguez-Aguilar, and Jesús Cerquides. Computationally efficient winner determination for mixed multi-unit combinatorial auctions. In *Proc. 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008).* IFAAMAS, 2008. Cited on pp. 133 and 134.

[68] Piotr J. Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in Multi-Agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005. Cited on p. 84.

[69] Sanjeev Goyal. *Connections: An introduction to the economics of networks.* Princeton University Press, 2007. Cited on p. 6.

[70] Jim Gray. Notes on data base operating systems. In Rudolf Bayer, Robert M. Graham, and Gerhard Seegmüller, editors, *Operating Systems, An Advanced Course*, volume 60 of *Lecture Notes in Computer Science*, pages 393–481. Springer-Verlag, 1978. Cited on p. 3.

[71] Joseph Greenberg. Coalition structures. In Robert J. Aumann and Sergiu Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 2 of *Handbook of Game Theory with Economic Applications*, chapter 37, pages 1305–1337. Elsevier, 1994. Cited on p. 96.

[72] Joseph Y. Halpern. A computer scientist looks at game theory. *Games and Economic Behavior*, 45(1):114–131, 2003. Cited on p. 13.

[73] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. Cited on pp. 13, 14, and 35.

[74] Joseph Y. Halpern and Moshe Y. Vardi. The complexity of reasoning about knowledge and time. I. Lower bounds. *Journal of Computer and System Sciences*, 38(1):195–237, 1989. Cited on p. 70.

[75] Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. Algorithmic knowledge. In *Proceedings of the 5th conference on Theoretical aspects of reasoning about knowledge (TARK V)*, pages 255–266, Pacific Grove, California, 1994. Morgan Kaufmann Publishers Inc. Cited on p. 73.

[76] John C. Harsanyi. Games with incomplete information played by "Bayesian" players, I-III. *Management Science*, 14(3, 5, 7):159–182, 320–334, 486–502, 1967. Cited on p. 4.

[77] Patrick J. Hayes. In defence of logic. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI 1977)*, pages 559–565, Cambridge, MA, USA, 1977. Cited on p. 68.

[78] Jaakko Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, 1962. Cited on p. 1.

[79] C. Antony R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978. Cited on pp. 15, 16, 18, and 35.

[80] C. Antony R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc, 1985. Cited on pp. 15 and 35.

[81] Luke Hunsberger and Barbara J. Grosz. A combinatorial auction for collaborative planning. In *Proc. 4th International Conference on MultiAgent Systems*. IEEE Computer Society, 2000. Cited on p. 134.

[82] INMOS Ltd. *occam 2 Reference Manual*. Prentice-Hall, 1988. Cited on p. 15.

[83] Matthew O. Jackson. *Social and Economic Networks*. Princeton University Press, 2008. Cited on pp. 6 and 47.

[84] Geraint Jones. On guards. In Traian Muntean, editor, *Parallel Programming of Transputer Based Machines*, pages 15–24, Amsterdam, 1988. IOS Press. Cited on pp. 15, 16, and 23.

[85] Michael Kearns, Michael L. Littman, and Satinder Singh. Graphical models for game theory. In Jack S. Breese and Daphne Koller, editors, *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 253–260, Seattle, Washington, 2001. Morgan Kaufmann. Cited on pp. 51 and 71.

[86] Donald E. Knuth, Christos H. Papadimitriou, and John N. Tsitsiklis. A note on strategy elimination in bimatrix games. *Operations Research Letters*, 7(3):103–107, 1988. Cited on p. 71.

[87] Saul Kripke. Semantical analysis of modal logic I: Normal modal propositional calculi. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963. Cited on p. 1.

[88] Reino Kurki-Suonio. Towards programming with knowledge expressions. In *Proceedings of the 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 140–149, St. Petersburg Beach, Florida, 1986. ACM Press. Cited on p. 35.

[89] John E. Laird and Michael van Lent. Human-level AI's killer application: Interactive computer games. *AI Magazine*, pages 15–25, 2001. Cited on p. 137.

[90] Gérard Le Lann. Distributed systems, towards a formal approach. *Information Processing*, 77:155–160, 1977. Cited on p. 22.

[91] João Leite and Luís Soares. Evolving characters in role playing games. In *Cybernetics and Systems, Proceedings of the 18th European Meeting on Cybernetics and Systems Research (EMCSR 2006)*, volume 2, pages 515–520, Vienna, 2006. Cited on p. 83.

[92] David Lewis. *Convention: A Philosophical Study*. Harvard University Press, 1969. Cited on p. 13.

[93] Alessio Lomuscio and Mark Ryan. Ideal agents sharing (some!) knowledge. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98)*, page 557–561. John Wiley & sons, 1998. Cited on p. 92.

[94] Inés Macho-Stadler, David Pérez-Castrillo, and Nicolás Porteiro. Sequential formation of coalitions through bilateral agreements in a cournot setting. *International Journal of Game Theory*, 34(2):207–228, 2006. Cited on p. 96.

[95] Martin Magnusson and Patrick Doherty. Logical agents for language and action. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE-08)*, Stanford, 2008. Cited on pp. 83 and 138.

[96] Sami Mäkelä. NPC Scripting and Reasoning about the NPC behaviour. *WorldForge: The Original Open Source MMO Project.* http://www.worldforge.org/project/newsletters/November2001/NPC_Scripting, 2001. Cited on p. 82.

[97] Marco Marini. An overview of coalition & network formation models for economic applications. Working Papers 0712, University of Urbino Carlo Bo, Department of Economics, 2007. Cited on p. 96.

[98] Michael Mateas and Andrew Stern. A behavior language: Joint action and behavioral idioms. In *Life-like Characters: Tools, Affective Functions and Applications*. Springer, 2004. Cited on p. 82.

[99] Michael Mateas and Andrew Stern. Façade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference, Game Design track*, 2003. Cited on p. 82.

[100] John-Jules Ch. Meyer and Wiebe van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, 1995. Cited on pp. 1 and 81.

[101] Dov Monderer and Dov Samet. Approximating common knowledge with common beliefs. *Games and Economic Behavior*, 1(2):170–190, 1989. Cited on p. 4.

[102] Stephen Morris. Coordination, communication, and common knowledge: A retrospective on the electronic-mail game. *Oxford Review of Economic Policy*, 18(4):433–445, 2002. Cited on p. 13.

[103] Hervé Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988. Cited on pp. 14 and 99.

[104] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. Cited on p. 4.

[105] Noam Nisan. Bidding languages for combinatorial auctions. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*. MIT Press, 2006. Cited on pp. 117, 119, and 120.

[106] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, New York, 2003. Cited on pp. 4 and 14.

[107] Brammert Ottens and Ulle Endriss. Comparing winner determination algorithms for mixed multi-unit combinatorial auctions. In *Proc. 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2008)*. IFAAMAS, 2008. Cited on pp. 133 and 134.

[108] Guillermo Owen. *Game Theory*. Academic Press, New York, third edition, 2001. Cited on p. 110.

[109] Eric Pacuit and Rohit Parikh. Reasoning about communication graphs. In Johan van Benthem, Benedikt Löwe, and Dov Gabbay, editors, *Interactive Logic*, volume 1 of *Texts in Logic and Games*, pages 135–157, London, 2007. Amsterdam University Press. Cited on pp. 6, 38, 40, 47, and 48.

[110] Catuscia Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical Structures in Computer Science*, 13(05):685–719, 2003. Cited on p. 15.

[111] Rohit Parikh. Knowledge and the problem of logical omniscience. In *Proceedings of the Second International Symposium on Methodologies for intelligent systems*, pages 432–439, Charlotte, North Carolina, 1987. North-Holland Publishing Co. Cited on p. 73.

[112] Rohit Parikh. Levels of knowledge, games, and group action. *Research in Economics*, 57(3):267–281, 2003. Cited on p. 93.

[113] Rohit Parikh. *Logical omniscience*, volume 960 of *Lecture Notes in Computer Science*, pages 22–29. Springer, Berlin, 1995. Cited on p. 9.

[114] Rohit Parikh. Sentences, belief and logical omniscience, or what does deduction tell us? *The Review of Symbolic Logic*, 1(04):459–476, 2008. Cited on p. 8.

[115] Rohit Parikh and Paul Krasucki. Communication, consensus, and knowledge. *Journal of Economic Theory*, 52(1):178–189, 1990. Cited on p. 22.

[116] Rohit Parikh and Ramaswamy Ramanujam. *Distributed processes and the logic of knowledge*, volume 193 of *Lecture Notes in Computer Science*, pages 256–268. Springer, Berlin, 1985. Cited on pp. 3, 14, and 88.

[117] Gordon D. Plotkin. An operational semantics for CSP. In Dines Bjørner, editor, *Formal Description of Programming Concepts – II*, pages 199–225, Amsterdam, 1983. North-Holland. Cited on p. 16.

[118] Marc Ponsen, Pieter Spronck, Héctor Muñoz-Avila, and David W. Aha. Knowledge acquisition for adaptive game AI. *Science of Computer Programming*, 67(1):59–75, 2007. Cited on p. 83.

[119] David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and Brain sciences*, 1(4):515–526, 1978. Cited on p. 78.

[120] Steve Rabin. AI Wisdom. http://www.aiwisdom.com/, 2009. Cited on p. 81.

[121] Ramaswamy Ramanujam. A discussion on explicit knowledge. In Krister Segerberg, editor, *The Parikh project: Seven papers in honour of Rohit*, volume 1996:18 of *Uppsala prints and preprints in philosophy*, pages 92–101. Filosofiska Institutionen, Uppsala Universitet, 1996. Cited on p. 73.

[122] Anand S. Rao and Michael P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 312–319, 1995. Cited on p. 77.

[123] Debraj Ray. *A Game-Theoretic Perspective on Coalition Formation*. Oxford University Press, 2008. Cited on p. 6.

[124] Debraj Ray and Rajiv Vohra. Equilibrium binding agreements. *Journal of Economic Theory*, 73(1):30–78, 1997. Cited on p. 96.

[125] Craig Reynolds. Game Research and Technology. *Reynolds Engineering & Design.* http://www.red3d.com/cwr/games/, 2007. Cited on p. 81.

[126] Floris Roelofsen. Exploring logical perspectives on distributed information and its dynamics. Master's thesis, ILLC, University of Amsterdam, 2005. Cited on pp. 47 and 48.

[127] Michael H. Rothkopf, Aleksandar Pekeč, and Ronald M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8): 1131–1147, 1998. Cited on p. 132.

[128] Ji Ruan. Exploring the update universe. Master's thesis, University of Amsterdam, 2004. Cited on p. 140.

[129] Ariel Rubinstein. The electronic mail game: Strategic behavior under "Almost common knowledge". *The American Economic Review*, 79(3): 385–391, 1989. Cited on pp. 3 and 13.

[130] Ariel Rubinstein. *Modeling Bounded Rationality*. The MIT Press, 1998. Cited on p. 7.

[131] Tomasz Sadzik. Exploring the iterated update universe. Technical report PP-2006-26, ILLC, University of Amsterdam, Amsterdam, 2006. Cited on p. 140.

[132] David Sally. Can I say "bobobo" and mean "There's no such thing as cheap talk"? *Journal of Economic Behavior & Organization*, 57(3):245–266, 2005. Cited on p. 52.

[133] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1–2):1–54, 2002. Cited on p. 117.

[134] Thomas C. Schelling. *The Strategy of Conflict*. Harvard University Press, 1960. Cited on p. 13.

[135] Fred B. Schneider. Synchronization in distributed programs. *ACM Transactions on Programming Languages and Systems*, 4(2):125–148, 1982. Cited on pp. 14 and 16.

[136] Steve Schneider. *Concurrent and Real Time Systems: The CSP Approach*. John Wiley and Sons, Inc, 1999. Cited on p. 15.

[137] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986. Cited on p. 125.

[138] Eddie Schwalb and Lluís Vila. Temporal constraints: A survey. *Constraints*, 3(2):129–149, 1998. Cited on p. 132.

[139] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009. Cited on p. 92.

[140] Emily Short. Conversation. *Emily Short's Interactive Fiction*. http://emshort.wordpress.com/writing-if/my-articles/conversation, 2007. Cited on p. 82.

[141] Flávio S. Corrêa da Silva and Wamberto W. Vasconcelos. Rule schemata for game artificial intelligence. In *Advances in Artificial Intelligence - IBERAMIA-SBIA 2006*, Ribeirão Preto, Brazil, 2006. Springer. Cited on p. 83.

[142] Tayfun Sönmez, Suryapratim Banerjee, and Hideo Konishi. Core in a simple coalition formation game. *Social Choice and Welfare*, 18(1):135–153, 2001. Cited on p. 96.

[143] Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma. Adaptive game AI with dynamic scripting. *Machine Learning*, 63(3):217–248, 2006. Cited on p. 83.

[144] Tommy Chin-Chiu Tan and Sérgio Ribeiro da Costa Werlang. The bayesian foundations of solution concepts of games. *Journal of Economic Theory*, 45 (2):370–391, 1988. Cited on pp. 8 and 51.

[145] Terese. *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science 55. Cambridge University Press, 2003. Cited on p. 96.

[146] Ubisoft. Assassin's Creed. http://assassinscreed.us.ubi.com, 2007. Cited on p. 86.

[147] Rineke Verbrugge and Lisette Mol. Learning to apply theory of mind. *Journal of Logic, Language and Information*, 17(4):489–511, 2008. Cited on p. 93.

[148] William E. Walsh, Michael P. Wellman, and Fredrik Ygge. Combinatorial auctions for supply chain formation. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, 2000. Cited on p. 115.

[149] Jonathan Weinstein and Muhamet Yildiz. Impact of higher-order uncertainty. *Games and Economic Behavior*, 60(1):200–212, 2007. Cited on p. 35.

[150] Peter Welch. An occam-pi Quick Reference, 1996–2008. https://www.cs.kent.ac.uk/research/groups/sys/wiki/OccamPiReference. Cited on p. 15.

[151] Peter Welch, Neil Brown, James Moores, Kevin Chalmers, and Bernhard Sputh. Integrating and extending JCSP. In Alistair A. McEwan, Steve Schneider, Wilson Ifill, and Peter Welch, editors, *Communicating Process Architectures*. IOS Press, 2007. Cited on p. 15.

[152] Michael P. Wellman, William E. Walsh, Peter R. Wurman, and Jeffrey K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35(1–2):271–303, 2001. Cited on p. 134.

[153] Wikipedia. The Elder Scrolls IV: Oblivion. http://en.wikipedia.org/w/index.php?title=The_Elder_Scrolls_IV:_Oblivion&oldid=96078473#Radiant_A.I., 2006. (The relevant section was removed in later revisions, since according to the 'discussion' page it concerned "Radian [sic] AI's behavior prior to the game release. Listing all these examples is beyond the scope of this article."). Cited on p. 81.

[154] Andreas Witzel. Symmetric and synchronous communication in peer-to-peer networks. In Philippe Audebaud and Christine Paulin-Mohring, editors, *Proceedings of the 9th International Conference on Mathematics of Program Construction (MPC'08)*, volume 5133 of *Lecture Notes in Computer Science*, pages 404–421, Marseille, France, 2008. Springer. Cited on p. 10.

[155] Andreas Witzel and Jonathan A. Zvesper. Epistemic logic and explicit knowledge in distributed programming (short paper). In Lin Padgham, David Parkes, Jörg P. Müller, and Simon Parsons, editors, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Estoril, Portugal, 2008. Cited on pp. 9, 68, 77, 79, 80, and 90.

[156] Andreas Witzel and Jonathan A. Zvesper. Higher-order knowledge in computer games. In *Proceedings of the AISB'08 Symposium on Logic and the Simulation of Interaction and Reasoning*, volume 9 of *AISB 2008 Convention*, pages 68–72, Aberdeen, Scotland, 2008. The Society for the Study of Artificial Intelligence and Simulation of Behaviour. Cited on p. 11.

[157] Andreas Witzel, Jonathan A. Zvesper, and Ethan Kennerly. Explicit knowledge programming for computer games. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2008)*, Stanford, California, 2008. AAAI Press. Cited on pp. 11 and 68.

[158] Georg Henrik von Wright. *An essay in modal logic*. Studies in logic and the foundations of mathematics. North-Holland Publishing Company, Amsterdam, 1951. Cited on p. 1.

[159] Sang-Seung Yi. Stable coalition structures with externalities. *Games and Economic Behavior*, 20:201–237, 1997. Cited on p. 96.

[160] Jianwen Yin, Michael S. Miller, Thomas R. Ioerger, John Yen, and Richard A. Volz. A knowledge-based approach for designing intelligent team training systems. In *Agents*, pages 427–434, 2000. Cited on p. 83.

Index

Samenvatting

Weet zij wat hij weet? En zo ja, *wat gaat ze doen?*

Dit proefschrift biedt een informatica-perspectief op vragen over kennis en interactie, en presenteert manieren om kunstmatige agents uit te rusten met de bijbehorende redeneringsvermogens. We beperken algemene kaders van epistemische logica en speltheorie, om praktische implementaties te verkrijgen gefundeerd in de theorie.

Het fundamentele idee van het hoofddeel van het proefschrift (hoofdstukken 1 tot 3) is om computer processen, of anderszins *gedistribueerde* programma's, te zien als spelers in een speltheoretische setting met onvolledige informatie. Als zodanig zouden zij kunnen communiceren om informatie te verkrijgen en speltheoretische algoritmen uit te voeren.

In hoofdstuk 1 leggen we de technische grondslagen voor de uitvoering van synchrone communicatie, en dus het bereiken van gemeenschappelijke kennis, tussen computer-processen die spelers vertegenwoordigen. Hiervoor bestuderen we dialecten van de proces-calculus CSP die beschikbaar is in de vorm van programmeertalen. We betogen dat voor onze doeleinden het proces-systeem een bepaalde symmetrie dient te hebben, en tonen aan dat om aan deze eis te voldoen er een bepaalde "guard" constructie in de taal aanwezig moet zijn. Omdat deze constructie niet vaak aanwezig is, is ons resultaat praktisch voldoende om een unieke programmeertaal te identificeren geschikt voor onze doeleinden.

In hoofdstuk 2 definiëren we wat wij "interaction structures" noemen, een concrete klasse van communicatie-netwerken. We geven aan wat voor soort communicatie-scenario's we ons op richten, en bestuderen eigenschappen van de kennis die het resultaat is van dit soort communicatie. Deze eigenschappen kunnen worden gebruikt voor het vereenvoudigen van redeneren over kennis in onze setting.

In hoofdstuk 3 bestuderen we spellen met een interaction structure die het spelers mogelijk maakt hun voorkeuren te communiceren, ervan uitgaande dat elke speler aanvankelijk alleen zijn eigen voorkeuren kent. We bekijken de resultaten

van herhaalde eliminatie van strikt gedomineerde strategieën die kunnen worden verkregen in elke gegeven communicatie-toestand. De inzichten uit de vorige hoofdstukken worden gebruikt om een epistemische basis voor onze resultaten te geven en een gedistribueerd algoritme te laten zien dat de procedures lokaal in elke speler-proces implementeert.

Na dit hoofddeel van het proefschrift gaan we verder met meer losjes gerelateerde satelliet-hoofdstukken.

Hoofdstuk 4 ligt kwa idee dicht bij het hoofddeel van het proefschrift, met dit verschil dat het zich richt op een *gecentraliseerde* in plaats van een gedistribueerde benadering, en dat het *computerspellen* beschouwd in plaats van spellen in de strikte speltheoretische zin. We betogen dat redeneren over kennis, ook over elkaars kennis, een cruciale rol speelt in strategische en sociale interactie in het echte leven. We bekijken bestaande literatuur en spellen die dergelijke interactie simuleren, en laten zien dat dit aspect momenteel wordt verwaarloosd. We geven concrete scenario's uit bestaande computerspellen die zouden kunnen profiteren van de integratie van dergelijke redenerings-technieken, en onderbouwen één daarvan met een beschrijving van een eenvoudige uitvoering bestemd voor experimentele evaluatie.

In hoofdstuk 5 doen we een voorstel voor een abstracte benadering van coalitievorming die zich richt op eenvoudige regels voor het samenvoegen en splitsen van groepen. We identificeren omstandigheden waaronder elke reeks van deze regels een unieke opsplitsing oplevert. Ons belangrijkste conceptuele instrument is een specifieke definitie van een stabiele opsplitsing. De resultaten worden geparametriseerd door een voorkeurs-relatie tussen de opsplitsingen van een groep van spelers, en zijn op natuurlijke wijze toepasbaar op coalitional TU-games, hedonic games en exchange economy games.

In hoofdstuk 6 breiden we het bestaande kader van mixed multi-unit combinatorial auctions uit met tijdsbeperkingen, presenteren we een expressieve biedings-taal, en laten we zien hoe het probleem is op te lossen van het bepalen van de winnaar van dergelijke veilingen, middels een integer-programmering implementatie. Mixed multi-unit combinatorial auctions zijn veilingen waar bieders kunnen bieden op combinaties van transformaties van goederen in plaats van alleen maar simpelweg op goederen. Bijvoorbeeld, een transformatie kan deeg en water nemen en brood opleveren. Dit model heeft veel potentieel voor toepassingen in het gebied van het formeren van toeleveringsketens.

Ten slotte geven we in hoofdstuk 7 een kijkje op mogelijke toekomstige richtingen voor de implementering van epistemische logica.

Abstract

Does she know what he knows? And if so, *what is she going to do?*
This dissertation takes a computer science perspective on questions of knowledge and interaction and presents approaches for endowing artificial agents with corresponding reasoning capabilities.

To this end, we restrict general frameworks of epistemic logic and game theory in order to obtain practical implementations grounded in theory.

The basic idea of the main part of the dissertation (Chapters 1–3) is to view computer processes, or otherwise *distributed* programs, as players in a game-theoretic setting with incomplete information. As such, they should be able to communicate in order to obtain information, and to perform game-theoretic algorithms.

In Chapter 1, we establish the technical foundations to support implementation of synchronous communication, and thus the attainment of common knowledge, among computer processes representing players. To this end, we examine dialects of the process calculus CSP, which is available in the form of programming languages. We argue that for our purposes the process system needs to exhibit a certain symmetry, and show that to satisfy this requirement we need a certain guard construct in the language. Since this construct is not commonly provided, our result practically identifies a unique programming language suitable for our purposes.

In Chapter 2, we define what we call interaction structures, a concrete class of communication networks. We specify what kind of communication scenario we focus on, and study properties of the knowledge that results from such communication. These properties can be used to simplify reasoning about knowledge in our setting.

In Chapter 3, we study games in the presence of an interaction structure, which allows players to communicate their preferences, assuming that each player initially only knows his own preferences. We study the outcomes of iterated elimination of strictly dominated strategies that can be obtained in any given

state of communication. The insights from the previous chapters are used in order to provide an epistemic basis for our results and to show a distributed algorithm that implements the procudures locally in each player process.

After this main part of the dissertation, we continue with more loosely related satellite chapters.

Chapter 4 is close to the main part in spirit, with the difference that it focuses on a *centralized* rather than a distributed approach, and that it considers *computer games* rather than games in the strict sense of game theory. We argue that reasoning about knowledge, including about each other's knowledge, plays a crucial role in real-life strategic and social interaction. We survey existing literature and games which simulate such interaction, and show that this issue is currently neglected. We give concrete scenarios from existing computer games which could profit from incorporating such reasoning techniques, and substantiate one of them by describing a simple implementation intended for experimental evaluation.

In Chapter 5, we propose an abstract approach to coalition formation that focuses on simple merge and split rules transforming partitions of a group of players. We identify conditions under which every iteration of these rules yields a unique partition. The main conceptual tool is a specific notion of a stable partition. The results are parametrized by a preference relation between partitions of a group of players and naturally apply to coalitional TU-games, hedonic games and exchange economy games.

In Chapter 6, we extend the existing framework of mixed multi-unit combinatorial auctions to include time constraints, present an expressive bidding language, and show how to solve the winner determination problem for such auctions using an integer programming implementation. Mixed multi-unit combinatorial auctions are auctions where bidders can offer combinations of transformations of goods rather than just simple goods. For example, a transformation might take dough and water and yield bread. This model has great potential for applications in the context of supply chain formation, which is further enhanced by the integration of time constraints.

Finally, in Chapter 7 we give an outlook on possible future directions for implementing epistemic logic.

Titles in the ILLC Dissertation Series:

ILLC DS-2001-01: **Maria Aloni**
Quantification under Conceptual Covers

ILLC DS-2001-02: **Alexander van den Bosch**
Rationality in Discovery - a study of Logic, Cognition, Computation and Neuropharmacology

ILLC DS-2001-03: **Erik de Haas**
Logics For OO Information Systems: a Semantic Study of Object Orientation from a Categorial Substructural Perspective

ILLC DS-2001-04: **Rosalie Iemhoff**
Provability Logic and Admissible Rules

ILLC DS-2001-05: **Eva Hoogland**
Definability and Interpolation: Model-theoretic investigations

ILLC DS-2001-06: **Ronald de Wolf**
Quantum Computing and Communication Complexity

ILLC DS-2001-07: **Katsumi Sasaki**
Logics and Provability

ILLC DS-2001-08: **Allard Tamminga**
Belief Dynamics. (Epistemo)logical Investigations

ILLC DS-2001-09: **Gwen Kerdiles**
Saying It with Pictures: a Logical Landscape of Conceptual Graphs

ILLC DS-2001-10: **Marc Pauly**
Logic for Social Software

ILLC DS-2002-01: **Nikos Massios**
Decision-Theoretic Robotic Surveillance

ILLC DS-2002-02: **Marco Aiello**
Spatial Reasoning: Theory and Practice

ILLC DS-2002-03: **Yuri Engelhardt**
The Language of Graphics

ILLC DS-2002-04: **Willem Klaas van Dam**
On Quantum Computation Theory

ILLC DS-2002-05: **Rosella Gennari**
Mapping Inferences: Constraint Propagation and Diamond Satisfaction

ILLC DS-2002-06: **Ivar Vermeulen**
A Logical Approach to Competition in Industries

ILLC DS-2003-01: **Barteld Kooi**
Knowledge, chance, and change

ILLC DS-2003-02: **Elisabeth Catherine Brouwer**
Imagining Metaphors: Cognitive Representation in Interpretation and Understanding

ILLC DS-2003-03: **Juan Heguiabehere**
Building Logic Toolboxes

ILLC DS-2003-04: **Christof Monz**
From Document Retrieval to Question Answering

ILLC DS-2004-01: **Hein Philipp Röhrig**
Quantum Query Complexity and Distributed Computing

ILLC DS-2004-02: **Sebastian Brand**
Rule-based Constraint Propagation: Theory and Applications

ILLC DS-2004-03: **Boudewijn de Bruin**
Explaining Games. On the Logic of Game Theoretic Explanations

ILLC DS-2005-01: **Balder David ten Cate**
Model theory for extended modal languages

ILLC DS-2005-02: **Willem-Jan van Hoeve**
Operations Research Techniques in Constraint Programming

ILLC DS-2005-03: **Rosja Mastop**
What can you do? Imperative mood in Semantic Theory

ILLC DS-2005-04: **Anna Pilatova**
A User's Guide to Proper names: Their Pragmatics and Semanics

ILLC DS-2005-05: **Sieuwert van Otterloo**
A Strategic Analysis of Multi-agent Protocols

ILLC DS-2006-01: **Troy Lee**
Kolmogorov complexity and formula size lower bounds

ILLC DS-2006-02: **Nick Bezhanishvili**
Lattices of intermediate and cylindric modal logics

ILLC DS-2006-03: **Clemens Kupke**
Finitary coalgebraic logics

ILLC DS-2006-04: **Robert Špalek**
Quantum Algorithms, Lower Bounds, and Time-Space Tradeoffs

ILLC DS-2006-05: **Aline Honingh**
The Origin and Well-Formedness of Tonal Pitch Structures

ILLC DS-2006-06: **Merlijn Sevenster**
Branches of imperfect information: logic, games, and computation

ILLC DS-2006-07: **Marie Nilsenova**
Rises and Falls. Studies in the Semantics and Pragmatics of Intonation

ILLC DS-2006-08: **Darko Sarenac**
Products of Topological Modal Logics

ILLC DS-2007-01: **Rudi Cilibrasi**
Statistical Inference Through Data Compression

ILLC DS-2007-02: **Neta Spiro**
What contributes to the perception of musical phrases in western classical music?

ILLC DS-2007-03: **Darrin Hindsill**
It's a Process and an Event: Perspectives in Event Semantics

ILLC DS-2007-04: **Katrin Schulz**
Minimal Models in Semantics and Pragmatics: Free Choice, Exhaustivity, and Conditionals

ILLC DS-2007-05: **Yoav Seginer**
Learning Syntactic Structure

ILLC DS-2008-01: **Stephanie Wehner**
Cryptography in a Quantum World

ILLC DS-2008-02: **Fenrong Liu**
Changing for the Better: Preference Dynamics and Agent Diversity

ILLC DS-2008-03: **Olivier Roy**
Thinking before Acting: Intentions, Logic, Rational Choice

ILLC DS-2008-04: **Patrick Girard**
Modal Logic for Belief and Preference Change

ILLC DS-2008-05: **Erik Rietveld**
Unreflective Action: A Philosophical Contribution to Integrative Neuroscience

ILLC DS-2008-06: **Falk Unger**
Noise in Quantum and Classical Computation and Non-locality

ILLC DS-2008-07: **Steven de Rooij**
Minimum Description Length Model Selection: Problems and Extensions

ILLC DS-2008-08: **Fabrice Nauze**
Modality in Typological Perspective

ILLC DS-2008-09: **Floris Roelofsen**
Anaphora Resolved

ILLC DS-2008-10: **Marian Counihan**
Looking for logic in all the wrong places: an investigation of language, literacy and logic in reasoning

ILLC DS-2009-01: **Jakub Szymanik**
Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language

ILLC DS-2009-02: **Hartmut Fitz**
Neural Syntax

ILLC DS-2009-03: **Brian Thomas Semmes**
A Game for the Borel Functions

ILLC DS-2009-04: **Sara L. Uckelman**
Modalities in Medieval Logic