

ADVERTIMENT. L'accés als continguts d'aquesta tesi doctoral i la seva utilització ha de respectar els drets de la persona autora. Pot ser utilitzada per a consulta o estudi personal, així com en activitats o materials d'investigació i docència en els termes establerts a l'art. 32 del Text Refós de la Llei de Propietat Intel·lectual (RDL 1/1996). Per altres utilitzacions es requereix l'autorització prèvia i expressa de la persona autora. En qualsevol cas, en la utilització dels seus continguts caldrà indicar de forma clara el nom i cognoms de la persona autora i el títol de la tesi doctoral. No s'autoritza la seva reproducció o altres formes d'explotació efectuades amb finalitats de lucre ni la seva comunicació pública des d'un lloc aliè al servei TDX. Tampoc s'autoritza la presentació del seu contingut en una finestra o marc aliè a TDX (framing). Aquesta reserva de drets afecta tant als continguts de la tesi com als seus resums i índexs.

ADVERTENCIA. El acceso a los contenidos de esta tesis doctoral y su utilización debe respetar los derechos de la persona autora. Puede ser utilizada para consulta o estudio personal, así como en actividades o materiales de investigación y docencia en los términos establecidos en el art. 32 del Texto Refundido de la Ley de Propiedad Intelectual (RDL 1/1996). Para otros usos se requiere la autorización previa y expresa de la persona autora. En cualquier caso, en la utilización de sus contenidos se deberá indicar de forma clara el nombre y apellidos de la persona autora y el título de la tesis doctoral. No se autoriza su reproducción u otras formas de explotación efectuadas con fines lucrativos ni su comunicación pública desde un sitio ajeno al servicio TDR. Tampoco se autoriza la presentación de su contenido en una ventana o marco ajeno a TDR (framing). Esta reserva de derechos afecta tanto al contenido de la tesis como a sus resúmenes e índices.

WARNING. Access to the contents of this doctoral thesis and its use must respect the rights of the author. It can be used for reference or private study, as well as research and learning activities or materials in the terms established by the 32nd article of the Spanish Consolidated Copyright Act (RDL 1/1996). Express and previous authorization of the author is required for any other uses. In any case, when using its content, full name of the author and title of the thesis must be clearly indicated. Reproduction or other forms of for profit use or public communication from outside TDX service is not allowed. Presentation of its content in a window or frame external to TDX (framing) is not authorized either. These rights affect both the content of the thesis and its abstracts and indexes.

The Complexity of Angel-Daemons and Game Isomorphism

Alina García Chacón

PhD Thesis advised by Joaquim Gabarró

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

April 2012



The Complexity of Angel-Daemons and Game Isomorphism

Alina García Chacón

PhD Thesis advised by Joaquim Gabarró

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

April 2012

To my dear son Dani

*“There is a driving force
more powerful than
steam, electricity and nuclear power:
the will.”*

Albert Einstein

Abstract

The analysis of the computational aspects of strategic situations is a basic field in Computer Sciences. Two main topics related to strategic games have been developed. First, introduction and analysis of a class of games (so called angel/daemon games) designed to assess web applications, have been considered. Second, the problem of isomorphism between strategic games has been analysed. Both parts have been separately considered.

Angel-Daemon Games

A service is a computational method that is made available for general use through a wide area network. The performance of web-services may fluctuate; at times of stress the performance of some services may be degraded (in extreme cases, to the point of failure). In this thesis *uncertainty profiles* and *Angel-Daemon* games are used to analyse service-based behaviours in situations where probabilistic reasoning may not be appropriate.

In such a game, an *angel* player acts on a bounded number of “angelic” services in a beneficial way while a *daemon* player acts on a bounded number of “daemonic” services in a negative way. Examples are used to illustrate how game theory can be used to analyse service-based scenarios in a realistic way that lies between over-optimism and over-pessimism.

The resilience of an orchestration to service failure has been analysed - here angels and daemons are used to model services which can fail when placed under stress. The Nash equilibria of a corresponding *Angel-Daemon* game may be used to assign a “robustness” value to an orchestration.

Finally, the complexity of equilibria problems for *Angel-Daemon* games has been analysed. It turns out that *Angel-Daemon* games are, at the best of our knowledge, the first natural example of zero-sum succinct games.

The fact that deciding the existence of a pure Nash equilibrium or a dominant strategy for a given player is Σ_2^P -complete has been proven. Furthermore, computing the value of an *Angel-Daemon* game is EXP-complete. Thus, matching the already known complexity results of the corresponding problems for the generic families of succinctly represented games with exponential number of actions.

Game Isomorphism

The question of whether two multi-player strategic games are equivalent and the computational complexity of deciding such a property has been addressed. Three notions of isomorphisms, *strong*, *weak* and *local* have been considered. Each one of these isomorphisms preserves a different structure of the game. *Strong* isomorphism is defined to preserve the utility functions and Nash equilibria. *Weak* isomorphism preserves only the

player preference relations and thus pure Nash equilibria. *Local* isomorphism preserves preferences defined only on “close” neighbourhood of strategy profiles.

The problem of the computational complexity of game isomorphism, which depends on the level of succinctness of the description of the input games but it is independent of the isomorphism to consider, has been shown. Utilities in games can be given succinctly by Turing machines, boolean circuits or boolean formulas, or explicitly by tables. Actions can be given also explicitly or succinctly. When the games are given in *general form*, an explicit description of actions and a succinct description of utilities have been assumed. It has been established that the game isomorphism problem for general form games is equivalent to the circuit isomorphism when utilities are described by Turing Machines; and to the boolean formula isomorphism problem when utilities are described by formulas. When the game is given in explicit form, it has been proven that the game isomorphism problem is equivalent to the graph isomorphism problem.

Finally, an equivalence classes of *small* games and their graphical representation have been also examined.

Resum

L'anàlisi dels aspectes computacionals de situacions estratègiques és un camp bàsic de les Ciències de la Computació. En aquesta tesi s'han desenvolupat dos aspectes fonamentals relacionats amb els jocs estratègics. Primer, s'ha considerat la introducció i l'anàlisi d'una classe de jocs anomenada jocs d'àngel-dimoni, dissenyada per avaluar aplicacions web. Segon, s'ha analitzat el problema de l'isomorfisme entre jocs estratègics. Ambdues parts han estat considerades de manera independent.

Jocs d'Àngel-Dimoni

Un servei és un mètode de càlcul que es fa disponible a través de la xarxa per ser usat de manera general. L'acompliment dels serveis web pot fluctuar. Fins i tot, en moments d'estrès el funcionament d'alguns serveis es pot degradar, fins al punt de fallar en casos extrems. Per tant, els *perfils d'incertesa* i els jocs d'Àngel-Dimoni s'utilitzen en aquesta tesi per analitzar conductes basades en els serveis, en situacions on el raonament probabilístic pot no ser apropiat.

En aquests jocs, el jugador *àngel* actua dins d'una sèrie limitada de serveis angelicals d'una manera beneficiosa. No obstant això, el jugador *dimoni* actua dins de la sèrie limitada de serveis demoníacs d'una manera negativa. Alguns dels exemples mostrats serveixen per il·lustrar com la teoria de jocs pot utilitzar-se en l'anàlisi d'escenaris basats en serveis d'una manera realista, a mig camí entre l'optimisme i el pessimisme.

Adicionalment s'ha analitzat la resistència d'orquestracions a la manca de serveis. Aquí, els àngels i els dimonis s'empren per a modelar els serveis que poden fallar en condicions de pressió. Els equilibris de Nash del joc corresponent d'Àngel-Dimoni es poden utilitzar per assignar cert valor de "solidesa" a una orquestració. Per últim, s'ha analitzat la complexitat de problemes d'equilibri en jocs d'Àngel-Dimoni. Aquesta classe de jocs d'Àngel-Dimoni constitueix el primer exemple natural de jocs succints de suma zero, i aquest és un dels resultats d'aquesta tesi.

A més, s'ha pogut provar el fet que la decisió de l'existència d'un equilibri de Nash pur o d'una estratègia dominant per a un jugador donat és Σ_2^P completa. També, s'ha comprovat que el càlcul del valor d'un joc d'Àngel-Dimoni és EXP-complet, la qual cosa coincideix amb els coneguts resultats de complexitat dels problemes corresponents de les famílies genèriques de jocs representats de manera succinta, amb un nombre exponencial d'accions.

Isomorfisme de Jocs

En aquesta tesi també s'ha abordat la qüestió de si dos jocs de múltiples jugadors són equivalents així com la complexitat computacional de decidir aquest tipus de propietat.

Adicionalment s'han considerat tres nocions de isomorfismes, *Strong*, *Weak* i *Local*. Cadascun d'aquests isomorfismes conserva una estructura diferent del joc. L'isomorfisme *Strong* es defineix per preservar les funcions d'utilitat i els equilibris de Nash. L'isomorfisme *Weak* conserva únicament les relacions de preferència del jugador i per tant, els equilibris de Nash purs. L'isomorfisme *Local* conserva les preferències definides només en veïnatge dels perfils d'estratègia.

També s'ha demostrat que la complexitat computacional del problema de l'isomorfisme de joc depèn del nivell de la concisió de la descripció dels jocs d'entrada, però és independent de quin dels dos tipus d'isomorfismes es considera. Les utilitats en els jocs, poden venir representades de manera succinta per màquines de Turing, circuits booleans o fórmules booleans. Fins i tot, poden ésser representats explícitament per taules.

Les accions es poden representar de forma explícita o de forma succinta. Quan els jocs es troben en forma *general*, s'assumeix una descripció explícita de les accions així com una breu descripció dels serveis públics. S'ha establert que el problema de l'isomorfisme de jocs en forma general, és equivalent al isomorfisme de circuits quan les utilitats són descrites per màquines de Turing i al problema de l'isomorfisme de fórmules booleans quan les utilitats són descrites per fórmules. Quan el joc es descriu en forma explícita, s'ha pogut comprovar que el problema d'isomorfisme de joc és equivalent al problema d'isomorfisme de grafs.

Finalment, s'han examinat algunes classes d'equivalència de jocs *petits*, així com la representació gràfica dels mateixos.

Resumen

El análisis de los aspectos computacionales de situaciones estratégicas es un campo básico de las Ciencias de la Computación. Han sido desarrollados dos tópicos fundamentales relacionados con los juegos estratégicos. Primero, se ha considerado la introducción y el análisis de una clase de juegos llamada juegos de angel-demonio, diseñada para evaluar aplicaciones web. Segundo, se ha analizado el problema del isomorfismo entre juegos estratégicos. Ambas partes han sido consideradas de modo independiente.

Juegos de Angel-Demonio

Un servicio es un método de cálculo que se hace disponible a través de la red para ser usado de modo general. El desempeño de los servicios web pueden fluctuar. Incluso, en momentos de estrés el funcionamiento de algunos servicios pueden ser degradados, hasta el punto de fallar en casos extremos. Por lo tanto, los *perfiles de incertidumbre* y los juegos de *Angel-Demonio* se utilizan en esta tesis para analizar conductas basadas en los servicios, en situaciones en que el razonamiento probabilístico puede no ser apropiado.

En tales juegos, el jugador *ángel* actúa dentro de una serie limitada de servicios angelicales de una manera beneficiosa. Sin embargo, el jugador *demonio* actúa dentro de la serie limitada de servicios demoníacos de una manera negativa. Algunos de los ejemplos mostrados sirven para ilustrar como la teoría de juegos puede ser utilizada en el análisis de escenarios basados en servicios de una manera realista a medio camino entre el optimismo y el pesimismo.

Adicionalmente se ha analizado la resistencia de orquestaciones a la falta de servicios. Aquí los ángeles y los demonios se utilizan para modelar los servicios que pueden fallar en condiciones de presión. Los equilibrios de Nash del juego correspondiente de *Angel-Demonio* pueden utilizarse para asignarle cierto valor de “solidez” a una orquestación.

Por último, se ha analizado la complejidad de problemas de equilibrio en juegos de *Angel-Demonio*. Resulta que esta clase de juegos de *Angel-Demonio* son el primer ejemplo natural de juegos sucintos de suma cero, en el mejor de nuestro conocimiento.

Además se ha podido probar el hecho de que la decisión de la existencia de un equilibrio de Nash puro o de una estrategia dominante para un jugador dado es Σ_2^P completo. También, se ha comprobado que el cálculo del valor de un juego de *Angel-Demonio* es EXP-completo, lo cual coincide con los conocidos resultados de complejidad de los problemas correspondientes de las familias genéricas de juegos representados de manera sucinta, con un número exponencial de acciones.

Isomorfismo de Juegos

En esta tesis también se ha abordado la cuestión de si dos juegos de múltiples jugadores son equivalentes así como la complejidad computacional de decidir este tipo de propiedad. Adicionalmente se han considerado tres nociones de isomorfismos, *Strong*, *Weak* y *Local*. Cada uno de estos isomorfismos conserva una estructura diferente del juego. El isomorfismo *Strong* se define para preservar las funciones de utilidad y los equilibrios de Nash. El isomorfismo *Weak* conserva únicamente las relaciones de preferencia del jugador y por lo tanto los equilibrios de Nash puros. El isomorfismo *Local* conserva las preferencias definidas solamente en vecindad de los perfiles de estrategia.

También se ha demostrado que la complejidad computacional del problema del isomorfismo de juego depende del nivel de la concisión de la descripción de los juegos de entrada, pero es independiente de cuál de los dos tipos de isomorfismos se considera. Las utilidades en los juegos, pueden venir representadas de forma sucinta por máquinas de Turing, circuitos booleanos o fórmulas booleanas. Incluso, pueden representarse explícitamente por tablas.

Las acciones se pueden representar de forma explícita o de forma sucinta. Cuando los juegos se encuentran en forma *general*, se asume una descripción explícita de las acciones así como una breve descripción de los servicios públicos. Se ha establecido que el problema del isomorfismo de juegos en forma general, es equivalente al isomorfismo de circuitos cuando las utilidades son descritas por máquinas de Turing y al problema del isomorfismo de fórmulas booleanas cuando las utilidades son descritas por fórmulas. Cuando el juego se describe de forma explícita, se ha podido comprobar que el problema de isomorfismo de juego es equivalente al problema de isomorfismo de grafos.

Finalmente, se han examinado algunas clases de equivalencia de juegos *pequeños*, así como la representación gráfica de los mismos.

Acknowledgements

I have spent several years of the Software department (LSI)¹ in the ALBCOM² research group, developing this PhD thesis. Time, in which I accumulated knowledge and experience about research. Many people around have given me an unconditional support make it possible that I am here today. These words are dedicated to all these people.

Firstly, I would like to thank my advisor Joaquim Gabarró for being always very accessible. I am specially grateful for his advice and patience. In addition, for giving me the opportunity to work with him and Maria J. Serna.

Many thanks to the secretaries, who gave me advice and help with my young son. Especially thanks to Mercè Juan who has always helped with administrative tasks during these long years as a PhD student.

It is essential to devote a few words of gratitude to those institutions and projects, which economically have supported our work. First, the FPI scholarship gratefully³, which gave me four years to devoted to the study of computational game theory. Thanks as well, to the following projects:

- Técnicas de Optimización Avanzadas para Problemas Complejos (TRACER), (CYCIT TIC2002-04498-C05-03).
- Fundamental Aspects of Global Computing Systems (FLAGS), (EU IST- 2001-33116).
- Autoorganización en Sistemas de Comunicación Emergentes (ASCE), (MEC-TIN2005-09198-C02-02).
- Algorithmic principles for building efficient Overlay computers (AEOLUS), (FET pro-actives Integrated Project 15964).
- FP6 Network of Excellence CoreGRID founded by the European Commision (Contract IST-2002-004265).
- Métodos Formales y algoritmos para el diseño de sistemas (FORMALISM), (MEC-TIN2007-66523).

I am also very grateful to the eTUMOUR⁴ project, which gave me the possibility to work in the GABRMN⁵ group at the UAB⁶ university. This research group has opened

¹<http://www.lsi.upc.edu>

²ALBCOM-SGR-2013 Grup de Recerca Reconegut: Algorismes, Bioinformàtica, Complexitat i Mètodes Formals.

³Ministerio de Ciencia y Tecnología, beca FPI BES-2003-2361

⁴<http://www.etumour.net/>

⁵<http://gabrmn.uab.es/>

⁶<http://www.uab.cat/>

me the door to a very interesting world: brain tumours research. I am especially thankful to Carles Arus, the group's director, for giving me some time to finish up this thesis. Similarly, many thanks to Margarida Julià Sapé for her English language support. Moreover, to be open to the computational issues of game theory.

I am very grateful to my parents Jorge and Aída, whose infinite love and support have always been there and also, given me forces to do everything I do. Finally, I want to thank to Daniel and my son Dani, for their love, patience and constant support.

But nothing would have been possible without the help of one person, Francesc Puntas. He gave me the chance to live in Barcelona. Consequently, thanks too to this city, which has give me the opportunity to realise my dreams.

Contents

1	Algorithmic Game Theory: Angel-Daemons and Isomorphisms	1
1.1	Algorithmic Game Theory and Isomorphisms	1
1.2	Isomorphisms on Game Theory	3
1.3	Angel-Daemon Games and Web Orchestrations	4
1.4	Overview of this thesis	6
1.5	Thesis outline	8
1.6	Notes	9
2	Preliminaries on Games	11
2.1	Strategic and Extensive Games	11
2.2	Definitions and Preliminaries	15
2.3	Notes	20
I	Angel-Daemon Games	23
3	Preliminaries on Web Orchestrations	25
3.1	Web-services and Orchestration versus Choreography	25
3.2	Orchestration and Game Theory	31
3.3	Notes	31
4	Bounded Site Failures: an Approach to Unreliable Web Environments	33
4.1	Unreliable Environments and Risk Management	33
4.1.1	Orchestrations: The Number of Published Values	36
4.2	Assessing Orchestrations	38
4.3	Two Player Games: The Angel-Daemon Case	39
4.4	Maximisation and Minimisation Approaches	43
4.5	Properties of Uncertainty Profiles and Assessments	45
4.6	Notes	51

5	On the Complexity of Equilibria Problems in Angel-Daemon Games	53
5.1	Angel-Daemon Games	53
5.2	Strategic Games and Succinct Representations	54
5.3	Orc and Angel-Daemon Games	56
5.4	The Complexity of the EPN Problem	56
5.5	Computing the Value of Angel-Daemon Game	60
5.6	Deciding the Existence of Dominant Strategies	64
5.7	Notes	64
II	Computational Issues of Game Isomorphism	67
6	Preliminaries on Game Isomorphisms	69
6.1	Strong, Weak and Local Game Isomorphism	71
6.2	Classical Complexity's Problems	73
6.3	Notes	75
7	The Complexity of Game Isomorphism	77
7.1	The ISISO and ISO Problems	77
7.2	Complexity Results for Strong Isomorphisms	79
7.3	Weak Isomorphisms	99
7.4	Notes	110
8	On the Hardness of Game Equivalence Under Local Isomorphism	111
8.1	The Isomorphism Problem	111
8.2	From Strong Isomorphism to Local Isomorphism	114
8.3	From General Games to Binary Actions Games	118
8.4	From Local Isomorphism on Binary Action Games to Strong Isomorphism	126
8.5	The Complexity of Local Isomorphism	127
8.6	Notes	129
III	Conclusions and Future Work	131
9	Conclusions and Future Work	133
IV	Appendices	139
A	Arranging a Meeting using Reputation	141

B	IT System example	145
B.1	Different Failure Scenarios	148
C	Small Games. Graphic Representation	153
C.1	Local Nash Isomorphism	153
C.2	Equivalence Classes of Games	154
V	Publications and Projects	169
A	List of Publications	171
B	List of Projects	173
	Bibliography	175

Chapter 1

Algorithmic Game Theory: Angel-Daemons and Isomorphisms

Game theory is a branch of Mathematics, which was conceived as a tool to analyse situations involving conflict of interests and strategies. It served as theoretical model in the modern Economics. This theory has military applications, as well as in the fields of sociology, biology and currently, in informatics.

1.1 Algorithmic Game Theory and Isomorphisms

Game Theory. A game is defined as a competitive situation between two or more players under specified rules. Therefore, any sort of activity involving contenders, which interact according to well-defined rules, may be considered as a game. Specifically, a strategic game represents a situation where two or more players have a set of actions, by which each may gain or lose, depending on what others choose to do or not to do. These situations are uncertain, because no player knows for sure what the other player is going to decide. Consequently, Game theory studies situations where multiple contenders make decisions in order to maximise their respective gains. Of course, *chess*, *poker* and *bridge* are instances of games.

The first known discussion connected to game theory took place in 1713, when J. Waldegrave wrote a letter in which he provided a solution to a two-person version of a card game. Unfortunately, a generalised theoretical analysis of games was not further pursued until the publication of *Recherches sur les principes mathematiques de la theorie des richesses* of A. Augustin Cournot's in 1838 [26], in which Cournot considered a duopoly and presented a solution as a restricted version of what would later be defined as *Nash equilibrium*.

Nevertheless, game theory did not really exist as a unique and well-defined field until J. Neumann and O. Morgenstern published a series of papers starting in 1928, and culmi-

nating in 1944 with the book *The Theory of Games and Economic Behavior* [81]. This book presented the method for finding optimal solutions to two-person zero-sum games, where one's gain means the other's loss.

During the period 1928 to 1950, work on game theory was primarily focused in cooperative game theory, which analyse optimal strategies for groups of individuals, which have agreed on the strategies to follow. At the same time, J. Nash [80] developed a definition for an optimum strategy for multi-player games, where each player chose his best response to the choices of the other players. If all players announce their strategies simultaneously, nobody will want to reconsider his own choice and, such a situation is known as *Nash equilibrium*. Besides, Nash equilibrium is the most famous of equilibrium concepts in game theory.

From 1950 onwards, game theory experienced a flurry of activity. During this period of time, games in extensive form, and repeated games were developed. In addition, the first applications of game theory to philosophy and political science occurred.

Algorithmic Game Theory. At the beginning, mathematicians, economists and theorists never worried about computational representation of games. The classic book on *Game theory* by M. Osborne and A. Rubinstein [84] in 1994 does not explore the computability side of Nash equilibrium.

In game theory, the computability side of Nash equilibrium is a very active research field. The idea of applying computational complexity to analyse Nash equilibrium was introduced by C. Papadimitriou [85] in his well known paper *Algorithms, games, and the Internet*, at 2001. In this work, C. Papadimitriou presented his ideas about the growing interaction between game theory and, more generally, Economic theory and Theoretical Computer Science, mainly in the context of the Internet, where C. Papadimitriou invited to study into the frontier between these three research areas. Algorithmic game theory has emerged as a result of such a fusion.

The Algorithmic game theory applies analytical tools from computer science to game theory and/or economics, and also considers if the efficient computability of best response is or not a natural consideration for algorithm designer. This allow us to consider the not rational behaviour of agents, contrary to what classical economics says, and to study equilibria problems under bounded rationality. It also seeks optimal solutions or the impossibility to obtain results for specific problems, and adopts reasonable polynomial-time computational complexity as a constraint on the behaviour of systems.

This theory considers different ways to represent the input of games. However, all game representations [82] affect the complexity of problems associated with them. In this line of research, C. Álvarez et al. [8, 9] studied the computational complexity of deciding the existence of a pure Nash equilibrium in multi-player strategic games, addressing two

questions: *How can we represent a game?* and *how can we represent a game with polynomial payoff functions?* In order to resolve these questions, the authors considered different cases, depending on how are listed the set of actions and payoff functions of each player in a game. If the actions are described explicitly, by listing set of actions and by tabulating the payoff functions or more succinct representations, in which the payoff functions are described in terms of Turing machines TM, on what they called *standard form*. If the actions are described explicitly, by giving the list of the actions allowed to each player, what they called *general form*, or succinctly, by giving the length of the actions, what they called the *implicit form*. To this effect, the *Angel-Daemon* games proposed in this thesis, constitute a natural class of strategic games with a succinct representation.

The two main topics developed in this work will be presented: the *game isomorphisms* and the *Angel-Daemon games*.

1.2 Isomorphisms on Game Theory

Morphisms have been well studied in mathematics, as an abstraction of a structure-preserving mapping between two mathematical entities. Therefore, game isomorphism is a key concept that may be viewed as an appropriate model to identify structural similarity between strategic or extensive games. First, in 1947, J. von Neumann and O. Morgenstern [81] introduced a notion of strategic equivalence. Two years later, in 1949, J. C. C. McKinsey [72] was interested in the concepts of isomorphisms in games, and strategic equivalence. In 1951, J. Nash [80] defined the concept of automorphism or symmetry of a game, as a permutation of its pure strategies, if certain conditions were satisfied. More recently, in 1988, J. C. Harsanyi and R. Selton introduced other definitions of isomorphism [49] for strategic games.

One of the goals within game theory consists in considering a particular problem as a game and then restricting its space of possible solutions. In this sense, and given the emergence of this theory, another important goal has been to identify structural equivalence between games. Since it is interesting to know which transformations leave a game without changes, i.e., which transformations preserve all relations and operations in a given game, the following question could be asked: *If two strategic games are equivalent, does this equivalence subsist when both games are represented as extensive games?* Otherwise, in connection to symmetry, *do two symmetric strategic games loose their symmetry when represented as extensive games?* This idea was studied by B. Peleg, J. Rosenmüller, and P. Sudhölter [90, 108] in 2000 and A. Casajus [22, 23] in 2003 and 2006 respectively, among others.

To our knowledge, no previous studies reported on the complexity of game isomorphisms, despite this being a significant issue in computational game theory. This context

led us to ask questions like: *Whether two games are equivalent or not?* And, given the different ways of representing games, *what is the computational complexity of deciding when two games are equivalent or not?*

1.3 Angel-Daemon Games and Web Orchestrations

In order to study the computational complexity of problems on games it is fundamental to define how an input game is represented. The complexity of a problem is analysed taking into account the size of the input. Clearly, the size of a game representation depends mainly on the players number and on the action set size. In the same way, and also as part of the game description, the utility function of each player depends on the number of strategy profiles. Therefore, it is important to make clear how to describe the players set, and for each player their actions set and utility functions.

In this thesis, is defined an *Angel-Daemon* game, as an example of zero-sum succinct games, in the sense of [9, 38]. This class of games has been defined in order to analyse the behaviour of orchestrations over unreliable environments. Similarly that unreliable environments, the computational models inherent in many Web applications have been considered. These models consist in acquiring data from remote services, performing calculations with these data, and invoking other remote services with the results. Additionally, it is often necessary to rely on alternative services to execute the same calculation, in order to guard against system failures.

The first discussion about the importance of concurrency, communication and synchronisation in software systems was presented by E. Dijkstra [30], in 1965:

“The applications are these in which the activity of a computer must include the proper reaction to a possible great variety of messages that can be sent to it at unpredictable moments, a situation which occurs in process control, traffic control, stock control, banking applications, automation of information flow in large organisations, centralized computer service, and, finally, all information systems in which a number of computers are coupled to each other.”

Some clear examples of the importance of finding solutions that require *orchestration* and *choreography* of concurrent and distributed services are: the networks applications with many distributed systems available, the increasing dependency of complex workflows of data analysis, the workflows among organisations and individuals, and finally the problems associated to acquiring data from one or more remote services. All of these may fail in their components and communications. In addition, it is also necessary to be able to handle time-outs, and arbitrary delays, among others instances.

Orchestration represents control from the perspective of one party. *Choreography*

tracks the message sequences among multiple parties and sources, i.e., public message exchanges that occur between web-services.

Practical solutions exist, requiring *orchestration* of concurrent and distributed services, that may have failures in their components and communications. These solutions are based on programming languages, and rely on threads for concurrency and semaphores for synchronisation. But in spite of these the programming of concurrent systems is difficult. Theoretical and very flexible models have been developed for concurrency, for instance, π -Calculus by R. Milner [74] and C. A. R. Hoare [51], that include constructs for concurrency and channels for communication and synchronisation. Petri nets by J. L. Peterson [94], have been developed to describe reactive systems.

The orchestration language Orc by J. Misra and W. Cook [76, 58, 25, 59] is a very useful tool that allows to model concurrency. It also allows to describe orchestrations of distributed and management systems such as these used in the context of Web. An Orc language is a process calculus in which basic services, like user interaction and data manipulation, are implemented by primitive *sites*. This language provides constructs to orchestrate the concurrent invocation of sites to achieves a goal while managing timeouts, priorities, and failures of sites or communications. In addition, this language has already been used to implement a variety of traditional concurrent programming patterns, some of which overlap with the workflow patterns, as the W. van der Aalst's patterns [3].

A very important aspect to take into account when designing computational environments, Web applications, workflows, is *Risk management*. This is a well-established discipline in finance [60, 53] which has also been applied to software design [112] and web-services [62, 63]. Risk is quantifiable and, may be measured using probability. In contrast, uncertainty refers to something less tangible and, consequently, more difficult to quantify. Theoretical economists have long been trying to model uncertainty (see [6] Chapter 11). Game theory has been used in order to analyse the behaviour of distributed systems by modelling players as agents, which want to maximize their utilities. game theory, has also been used to assess some aspects of risk within computer science (e.g. an analysis of system failure is given in [77]). In this sense, in 1999, K. Eliaz introduced a notion of fault tolerance implementation [32]. In this paper he investigated the problem posed by the failure of a limited number of players. These faulty players have defined their own preferences well, but for some reason they do not behave in an optimal way, therefore not act in accordance to their preferences.

Selfishness in systems has also been studied, which may cause suboptimal computations. T. Moscibroda and S. Schmid and R. Wattenhofer published a work [77] in 2006, where they considered a *Price of Malice* in a game over a distributed system. This game modeled the containment of the spread of viruses. In such game, each node could choose whether or not to install anti-virus software. A virus started from a random node, to infect

all neighbouring nodes which did not have anti-virus software installed.

The analysis of the behaviour of distributed systems with a certain number of faulty components, which may show a kind of *malicious* behaviour, is expressed as *the Byzantine Generals Problem* [67] published by L. Lamport, R. Shostak and M. Pease in 1982. They proposed solutions in order to implement a reliable computer system.

Therefore, the *Angel-Daemon game* associated to a uncertainty profile for a given Orc expression E , allows to analyse the behaviour of orchestrations over unreliable environments. For such an analysis, the *risk profile*¹ concept [44] has been introduced, where a distinction is made between *risk* and *uncertainty* [106].

These games have two players, angel and daemon. The \mathcal{A} set represents sites that fail due to misbehaviour or other reasons, but that essentially are not malicious, and are called *angelic*. While the set \mathcal{D} represent the sites with malicious behaviour, and thus called *daemonic*. In general, each site is assumed to have probability of failure and distinct sites are assumed to be independent. As such an *Angel-Daemon game* is a zero-sum game it follows [113] that all Nash equilibrium of the game will have the same utility for the angel, which is known as the *game value*. The game value is used as a measure for assessing the expected behaviour of expression E in the environment described by the uncertainty profile [39, 44].

1.4 Overview of this thesis

The thesis objectives. This thesis has two objectives: first, to study the complexity of equilibrium problems for a class of strategic zero-sum games, called *Angel-Daemon* games. Second, to deepen the research on computational complexity of game isomorphisms and study how the game representations affect the complexity of problems.

Angel-Daemon Games. This class of zero-sum games is considered, in order to evaluate Orc expressions in an untrusted environment, by means of game theory. Was analysed the effect of a number of service failures in Web applications, modelled as Orc expressions in which sites are called to perform sub-computations. On the other hand, the complexity of deciding the existence of pure Nash equilibrium or a dominant strategy for a given player is considered. Furthermore, the complexity of computing the value of an *Angel-Daemon* game is also considered. Thus, matching with the already known complexity results of the corresponding problems for the generic families of succinctly represented games with exponential number of actions. In particular, the computational complexity of the fol-

¹The concepts "risk profile" and "uncertainty profile" are the same. From now onwards in this thesis, we will refer as "uncertainty profile".

lowing problems on angel-daemon games and on games with succinct representations are analysed.

Exists pure Nash equilibrium? (EPN). Given a game Γ , decide whether Γ has a pure Nash equilibrium.

Exists dominant strategy? (EDS). Given a game Γ , and a player i , decide whether there is a dominant strategy for player i in Γ .

Game Value (GV). Given a zero-sum game Γ , compute its value.

The results include a characterisation of the complexity of all the problems introduced above, when the input is restricted to be an *Angel-Daemon* game, showing that:

- Deciding the EPN and the EDS problem are Σ_2^P -complete, and
- The problem GV is EXP-complete.

This provides the first natural family of succinct games for which such complexity results can be established. Similar result for general families were already known for strategic games in implicit form [9] and succinct zero-sum games [38].

Game Isomorphisms. In order to understand the computational complexity of game isomorphism, two problems related to games and morphisms have been considered.

Is Game Isomorphism (ISISO). Given two games Γ, Γ' and a game mapping $\psi : \Gamma \rightarrow \Gamma'$, decide whether ψ is a game isomorphism.

Game Isomorphism (ISO). Given two games Γ, Γ' , decide whether there exists a game isomorphism between Γ and Γ' .

To study the computational aspects of isomorphism problems on strategic games, it was first needed to determine the way in which games and morphisms are represented as inputs to a program. For the representation of strategic games, the proposal given in [9] was adopted, and two representations was considered, each with a different level of succinctness: when a game is given in *general* form, the actions are listed explicitly but utilities and mappings are given by deterministic Turing machines TM. In the *explicit* case utilities are stored in tables. In both cases morphisms are always represented by tables. This is not a restriction as in polynomial time a morphism representation can be transformed by Turing machines TM into a tabular representation by tables, because the actions are explicitly given.

The main contributions are to the following problems:

- The ISISO problem is coNP-complete, for games given in general form, and belongs to NC when games are given in explicit form.

- The ISO problem belong to Σ_2^P , for games given in general form, and to NP when games are given in explicit form.
- The ISO problem is equivalent to the boolean circuit isomorphism problem, for games in general form, and to the graph isomorphism problem, for games given in explicit form.

Besides the above generic forms of representing games another particular class of strategic games was considered, that was called *formula games*. Formula games are (as it was showed) equivalent in power of representation to a subfamily of the *weighted boolean formula games* introduced in [71]. The complexity of the ISO problem when the games correspond to a general form, that is, the number of bits controlled by each player is a constant. For formula games in general form, was showed that the ISO problem is equivalent to boolean formula isomorphism. Recall that the complexity of the boolean formula isomorphism problem is the same as that of circuit isomorphism, however it is conjectured that both problems are not equivalent.

1.5 Thesis outline

This thesis consists of two parts, organized in 9 chapters. Chapter 2 is a general introduction to game theory, containing the most important definitions and notations used in this work.

PART I: *Angel-Daemon* Games.

- By way of a specific introduction to each area of research, an introductory Chapter 3 is dedicated to orchestration evaluation of Web applications using game theory aspects, and all the elements and concepts related.
- In chapter 4 *Angel-Daemon* games are evaluated, as a method of assessing Orc expressions in untrusted environments. The effect of a number of service failures during the execution of a Web orchestration is analysed according to this model.
- Chapter 5 is concerned to the complexity of equilibria problems for called *Angel-Daemon* games.

PART II: Game Isomorphisms and complexity.

- Introduction chapter 6 for this area, and with provides the terminology used in Algorithmic game theory and game isomorphisms.
- Chapters 7 and 8 refer to show the work performed in game isomorphisms. The first one is concerned to the complexity of strong and weak

game isomorphisms and the second one, studies the complexity of local game isomorphisms.

PART III: Conclusions.

- The conclusions of this work and the future lines of research, are proposed in chapter 9.

Finally, most references to related works have been moved out of the main chapter and have been placed a notes section at the end of each chapter.

1.6 Notes

Game Theory. For a general introduction to game theory, and mathematical definitions of strategic and extensive games refer to *An Introduction to the game theory* of M. Osborne and A. Rubinstein [84] published on 1994, and *Theory of Games and Economic Behavior* of John von Neuman and Oscar Morgenstern [81]. A very popular concept and also very important in this theory, is called *Nash equilibrium*. The contributions was written by John Nash when he was 21-years old. In this dissertation he presented the *Nash Equilibrium* for strategic non-cooperative games and proposed a very *compact* definition of game morphisms [80, 79]. On the formal side, the existence of a proof was one of the first applications of Kakutani's fixed-point theorem. In addition, refer to *The Nash equilibrium: a perspective* [52] by Charles A. Holt and Alvin E. Roth. It is important to read the proposed isomorphism among strategic games, by J. Harsanyi and R. Selten [49] in 1998.

Algorithmic Game Theory. Other interesting results are *When are two games the same?* by J. van Benthem [111] and *Game Transformations and Game Equivalence* by B. de Bruin [29].

In order to understand the complexity in game theory, it is crucial be familiar with [45] *Computers and Intractability: A Guide to the Theory of NP-completeness* from M. Garey and D. Johnson. From the point of view of algorithmic game theory, there are a few important references like *The complexity of pure Nash equilibria* [34] where the focus is set on congestion games, investigating multi-player games that are guaranteed to have pure Nash equilibria.

Angel-Daemon Games and Web Orchestrations. When designing a Web application, the development and deployment of a number of services is required. These include security, information, directory, resource allocation, and payment mechanisms in an open environment; and high level services for application development, execution management,

resource aggregation and scheduling. Additionally, it is often necessary to rely on alternative services to do the same calculation, in order to guard against system failures. The study of systems under failure is not new [31, 89, 32, 37]; the analysis of risk is well studied in microeconomics (chapter 6 from [70]). Finally, refer to an orchestration language Orc by J. Misra and W. Cook [76, 59] as a solution to orchestrate distributed services.

Chapter 2

Preliminaries on Games

In Game Theory, a game can be described in two different ways: by strategic and extensive representations. The first representation is not graphical and represents a game by a matrix. Specifying players strategies and their corresponding payoffs. Therefore this approach is useful in identifying Nash equilibria and the strictly dominated strategies of players. Unlike strategic representation, an extensive form is a tree representation of a game, where each non-terminal node belongs to a player. Each player may choose among the possible moves at each node. Therefore, the nodes represent every possible state of game. Additionally, as the computational complexity of the game isomorphism problem depends on the level of succinctness of the description of the input game, was defined the inputs of games considered. The utilities in games have been given succinctly by a Turing machines TM, boolean circuits or boolean formulas, or explicitly by tables. Actions have been given also explicitly or succinctly.

2.1 Strategic and Extensive Games

Game Theory provides the mathematical tools and models to analyse strategic situations in which multiple participants interact or affect each other. In the last years a huge amount of research has been devoted to explore the usefulness of Game Theory in situations arising on the Internet. In these situations, many participants interact with competing goals and therefore, the games can be modeled by strategic or cooperative games.

Strategic Games. A strategic game is a model consisting in two or more interacting *players*, each one with a set of *strategies*. For each combination of strategies, there is a numerical *payoff* for each player. In such games, each player chooses her/his best available action, which depends in general, on other players actions. In order to choose among different alternative actions, each player must keep in mind the actions that the others players may chose. We may also assume that each player has experience playing the game, so that

each player's belief is derived from her/his past experience. The other players' actions and her/his belief upon the correctness of the other players' actions.

Example 2.1 In Figure 2.1 one of the better-known strategic games is shown: the Prisoner's Dilemma, which was originally framed by M. Flood and M. Dresher, in 1950.

		Suspect B	
		Quiet	Betrays
Suspect A	Quiet	1 year, 1 year	3 years, free
	Betrays	free, 3 years	2 years, 2 years

Figure 2.1: The Prisoner's Dilemma, where the players are the suspects A and B, and 1 year means one year in prison, free means that prisoner goes free. The Nash equilibrium is in the strategy profile (*Betrays, Betrays*).

This game models a situation in which there are two prisoners accused of a major crime, which are kept into separate cells. The dilemma arises when we assume that both prisoners only care about minimizing their own quantity of years in prison. Each prisoner has two options: cooperate with his accomplice and *stay quiet*, or *betray* his accomplice and confess, in return for a lighter sentence. The outcome of each choice depends on the choice of the accomplice, but each prisoner must choose without knowing what his accomplice has chosen to do.

In order to decide what to do in strategic situations, it is usually important to predict what others will do, but this is not the case here. Knowing that the other prisoner will stay silent, the best strategy is to betray, because in such a case the betrayer, will be out free, instead of receiving the minor sentence. Otherwise, knowing that the other prisoner will betray, the best strategy is still betray, because then, the first prisoner will receive a small sentence, just like the accomplice. Since the reasoning is similar for the other prisoner, he will also choose to betray. The Prisoner's Dilemma has a Nash equilibrium, which takes place when both prisoners desert, which is the strategy profile (*Betrays, Betrays*). This strategy (*both betray*), is worse than (*both quiet*), in the sense that the total time in jail is greater for both prisoners. Nevertheless the strategy (*both quiet*) is unstable, since each of the prisoners can improve his result betraying (if his opponent maintains the strategy of *quiet*). Therefore, (*both quiet*) is not an equilibrium.

The strategic model of a games, which are used as ingredient of more complicated games, allows us to study the interaction between players (i.e., to discuss how the actions of one of them may be affected by those of the others). This model is focused upon the strategies as a whole, thus ignoring the corresponding sequence of events. On the contrary, the extensive games model is centered upon such a sequence of events.

Therefore, strategic games are the first game structure to start analysing the computational complexity of game equivalence. The combinatorial structure of a strategic game

is simple enough to allow such kind of analysis by comparison with an isomorphism in other combinatorial structures.

Extensive Games. These games may be represented as *trees* of decision nodes, where *actions* correspond to arcs, and *terminal histories* are associated to leaves. That is, each terminal history is viewed as a sequence or lists of actions, where each list specifies the actions of a set of players. Thus games may be grouped on two categories: *extensive games with perfect information* and *extensive games with imperfect information*. In the case of extensive games with *perfect information*, each player knows every other player previous actions. In the case of extensive games with *imperfect information*, players do not know the previous actions of the other players.

In Figure 2.2, the *Prisoner's Dilemma* game is represented in extensive form with *perfect information*, and with *imperfect information*. Note that in sub-figure (a), the second player knows the action taken by the first one. In (b), the second player cannot distinguish between *Q* and *B*. This game is conceptually similar to the strategic game given in Figure 2.1.

Example 2.2 *The Prisoner's Dilemma game represented in its extensive form, both, with perfect information, and with imperfect information.*

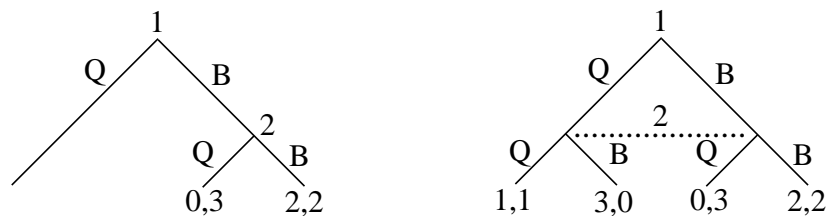


Figure 2.2: A game in extensive form, (a) with *perfect* information and (b) with *imperfect* information. The prisoners choose between the two strategies: staying Quiet (*Q*), or Betrays (*B*). The leaves are terminal histories, represents the payoff for each player.

As occurs in cases of other structures such as graphs, it is interesting to know which transformations leave a game without change; i.e., which transformations preserve all relations and operations in a given game. If two strategic games are equivalent, *does this equivalence subsist when both games are represented as extensive games?* Otherwise, in connection to symmetry, *two symmetric strategy games, loses their symmetry when they are represented as extensive games?*

Morphisms. Automorphism and isomorphism have been well studied in mathematics, as an abstraction of a structure preserving mapping between two mathematical entities.

Therefore, game isomorphism is a key concept that may be viewed as an appropriate model to identify structural similarity between strategic or extensive games.

The informal idea of *strategic equivalence* [72] has been widely discussed and explored along the history of Game Theory. Likewise, traditionally the notion of equivalence had been studied at different levels, using different types of isomorphism, depending on the family of games and the structural properties to be preserved. In 1951, J. Nash [80] gave a definition of automorphism between strategic games. More recently, S. Elmes and P. Reny [33] introduced a *strong* isomorphism of extensive games, and Peleg, Rosenmüller and Sudhölter [90] studied the relationship between strategic games and their possible representations in extensive form. They also defined a *strong* isomorphism of extensive games. But these isomorphisms are very strong and they are incompatible with the traditional representations of strategic games in extensive form. In essence, in strategic games, symmetric strategies are not symmetric in their equivalent extensive form representation. In order to resolve this conflict, they proposed [108] a method to represent strategic games as extensive forms preserving, essentially, all symmetries of the strategic games that satisfy additional axioms, so that both notions of symmetries coincide. A. Casajus [23, 22] also introduced another notion of isomorphism for extensive games, and equivalence [29] by the way of transformations to a common form.

Example 2.3 Figure 2.3, given by B. Peleg et al. in [90], presents an example of automorphism. Let us consider the strategic game in the same Figure 2.3. Consider an automorphism given by permuting the players; i.e., player 1 is transformed into player 2 and player 2 is transformed into player 1. The actions bijection are given by $\varphi_i(a_i) = 3 - a_i$ for $a_i \in A_i$, being $A_i = \{1, 2\}$.

		Player 2	
		1	2
Player 1	1	2, 1	0, 0
	2	0, 0	1, 2

Figure 2.3: For the initial matrix game Γ and applying the automorphism given by permuting the players, with the actions bijection given by $\varphi_i(a_i) = 3 - a_i$ for $a_i \in A_i = \{1, 2\}$, the same initial matrix game Γ is obtained.

Computational Game Theory. It is sometimes impossible to improve the worst-case time requirements as a function of the size of the input problem, when an algorithm is developed to solve a given computational problem. It is specifically in these cases, in which it is important to develop mathematical techniques for proving that no algorithm exists which runs faster than the current one. Therefore, the set of mathematical models and techniques for establishing the impossibility proofs are the focus of computational complexity.

In order to study the computational aspects of isomorphism problems of strategic games, we need first to determine the way in which games and morphisms are represented as inputs to a program. To represent strategic games, was adopted the proposal given in [8] and considered the following two representations, each with a different level of succinctness. When a game is given in *general* form, the actions are listed explicitly but utilities and mappings are given by deterministic Turing machines. In the *explicit* case, utilities are stored in tables. In both cases, the morphisms are always represented by tables. This is not a restriction, as in polynomial time we can transform a morphism representation by Turing machines into a tabular representation by tables, because the actions are given explicitly.

The computational issues arising from this framework, are one of the main objectives of the Algorithmic Game Theory community [85, 82, 103]. In this sense, an important issue, to be consider, is the study of complexity of isomorphisms of games, which no results in previous studies. Therefore, this thesis is concerned in the computational issues related to game equivalence.

2.2 Definitions and Preliminaries

In this section, definitions and terminology used during this thesis are provided. Firstly, strategic games and their representations. Will be presented, a discussion on game mapping and will follow, as well as the definition of the two types of isomorphism considered in this work.

Strategic Games

Definition 2.1 (Strategic Game) *A strategic game Γ is a tuple $\Gamma = \langle N, (A_i)_{i \in N}, (u_i)_{i \in N} \rangle$ where $N = \{1, \dots, n\}$ is the set of players [84]. For each player $i \in N$, A_i is a finite set of actions. For each player $i \in N$, u_i is a utility (or payoff) function, mapping $A_1 \times \dots \times A_n$ to the rational.*

The set of combined actions $A = A_1 \times \dots \times A_n$ is called the *set of strategy profiles*. Given a strategic game Γ , player i can “make a move” by selecting an action $s_i \in A_i$ (s_i is called a strategy). If player i selects a strategy s_i *independently*, then the joint *strategy profile* is $s = (s_1, \dots, s_n)$. Player i assesses the state of a game using $u_i(s)$. A strategy profile $s = (s_1, \dots, s_n)$ can be factored for player i as (s_{-i}, s_i) where $s_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$ (i.e. s_{-i} is the profile s in which has been strategy s_i removed). Strategic games model non-cooperative behaviour: a solution to a game corresponds to being able to identify pure Nash equilibria.

Definition 2.2 (Pure Nash Equilibrium) A strategy profile $s = (s_1, s_2, \dots, s_n)$ is a pure Nash equilibrium (PNE from now onwards) if for any player i and any $s'_i \in A_i$ we have $u_i(s_{-i}, s_i) \geq u_i(s_{-i}, s'_i)$, where (s_{-i}, s_i) denotes the strategy profile in which s_i is replaced by s'_i .

Definition 2.3 (Expected Utility) An expected utility $u_i(\sigma)$ for player i , where σ is a mixed strategy profile $\sigma = (\sigma_1, \dots, \sigma_n)$, can be defined as:

$$u_i(\sigma) = \sum_{(a_1, \dots, a_n) \in A_1 \times \dots \times A_n} \sigma_1(a_1) \dots \sigma_n(a_n) u_i(a_1, \dots, a_n)$$

Definition 2.4 (Mixed Strategy) A mixed strategy σ_i for player i , is a probability distribution on the A_i set. A mixed strategy profile is a tuple, $\sigma = (\sigma_1, \dots, \sigma_n)$ and the utility of $u_i(\sigma)$ for player i is the expected utility.

Definition 2.5 (Nash Equilibrium) A mixed strategy profile $\sigma = (\sigma_1, \dots, \sigma_n)$ is a Nash equilibrium if, for any player i and any σ'_i it holds that, $u_i(\sigma_{-i}, \sigma_i) \geq u_i(\sigma_{-i}, \sigma'_i)$.

Theorem 2.1 (Mixed Nash Equilibrium) A mixed strategy profile σ is a mixed Nash equilibrium iff

- for any i and $s_i \in A_i$ such that $\sigma_i(s_i) > 0$, it holds that, $u_i(s_i, \sigma_{-i}) = u_i(\sigma)$,
- and for any i and $s_i \in A_i$ such that $\sigma_i(s_i) = 0$, it holds that, $u_i(s_i, \sigma_{-i}) \leq u_i(\sigma)$.

(see Proposition 116.2 in [84]). All strategic games have a Nash equilibrium [84]; however, there are games without any pure Nash equilibrium (see Example 2.4).

Two players Zero-Sum Games

In these games, players gains or losses is exactly balanced by the losses or gains of the other player.

Definition 2.6 (Zero-Sum Games) A zero-sum game is a strategic two-player game $\Gamma = \langle \{1, 2\}, A_1, A_2, u_1, u_2 \rangle$, in which $u_1(s) + u_2(s) = 0$, for any $s = (a_1, a_2)$.

For the case of zero-sum games, all Nash equilibria (pure or mixed) are assessed [81, 113] using utility player 1. The value of this utility, $v(\Gamma)$, is called the *value of Γ* . A *pure saddle point* is a strategy profile $s = (a_1, a_2)$ such that,

$$u_1(a_1, a_2) = \max_{a'_1 \in A_1} \min_{a'_2 \in A_2} u_1(a'_1, a'_2) = \min_{a'_2 \in A_1} \max_{a'_1 \in A_2} u_1(a'_1, a'_2)$$

The set of pure saddle points (which can be empty) coincides with the set of PNE (see Example 2.4 below). A player choice of action can be defined probabilistically. A mixed

strategy for player 1 is a probability distribution $\alpha : A_1 \rightarrow [0, 1]$ and, similarly, a mixed strategy for player 2 is a probability distribution $\beta : A_2 \rightarrow [0, 1]$. A mixed strategy profile is a tuple (α, β) , where

$$u_1(\alpha, \beta) = \sum_{(a_1, a_2) \in A_1 \times A_2} \alpha(a_1)\beta(a_2)u_1(a_1, a_2)$$

Let Δ_1 and Δ_2 denote the set of mixed strategies for players 1 and 2, respectively. It is well known [81] that there is always a mixed saddle point (α, β) satisfying,

$$u_1(\alpha, \beta) = \max_{\alpha' \in \Delta_1} \min_{\beta' \in \Delta_2} u_1(\alpha', \beta') = \min_{\beta' \in \Delta_2} \max_{\alpha' \in \Delta_1} u_1(\alpha', \beta')$$

The set of saddle points (pure or mixed) coincides with the set of Nash equilibria (pure or mixed).

Example 2.4 Consider the game Γ where $A_1 = \{\text{t}, \text{b}\}$ (denoting top and bottom) and $A_2 = \{\text{l}, \text{r}\}$ (denoting left and right). If the utility of the first player is

		Player 2	
		l	r
Player 1	t	$u_1(\text{t}, \text{l}) = 0$	1
	b	1	0

then Γ has no pure Nash equilibrium, because each player will always want to change his current strategy. If (t, l) is the current strategy then player 1 will wish to change t to b (so as to increase the utility by 1). This move is schematised as $(\text{t}, \text{l}) \xrightarrow{u_1=1} (\text{b}, \text{l})$. Profile (b, l) is not stable because player 2 would wish to change l to r (thereby increasing his utility by 1). The change in strategies is schematised below:

$$\begin{aligned} (\text{t}, \text{l}) &\xrightarrow{u_1=1} (\text{b}, \text{l}) \xrightarrow{u_1=0} (\text{b}, \text{r}) \\ &\xrightarrow{u_1=1} (\text{t}, \text{r}) \xrightarrow{u_1=0} (\text{t}, \text{l}) \xrightarrow{u_1=1} \dots \end{aligned}$$

$\alpha(\text{t}) = \alpha(\text{b}) = 1/2$ and $\beta(\text{l}) = \beta(\text{r}) = 1/2$ and $v(\Gamma) = 1/4 + 1/4 = 1/2$.

Suppose that $\alpha(\text{t}) = p$ where $p > 0$ and $\alpha(\text{b}) = 1 - p$. Then $u_1(\text{t}, \beta) = u_1(\text{b}, \beta)$ and therefore $p = 1/2$. Likewise, if $\beta(\text{l}) = q > 0$ and $\beta(\text{r}) = 1 - q$, then $u_2(\alpha, \text{l}) = u_2(\alpha, \text{r})$ and therefore $q = 1/2$. Consequently, $v(\Gamma) = u_1(\alpha, \beta) = 1/2$.

The Problem Input

In the context of computational complexity, it is very important to fix how games are represented as inputs of the problem. In all the different types of representations it will

always be assumed that the actions for each player are given explicitly, by listing all their elements used in this thesis. This leads with two types of representations, depending on whether the utilities are given explicitly or succinctly.

The first representation is the generic representation of strategic games given in [7], where the payoff functions of a game are described by a deterministic Turing Machine TM.

Strategic Game in General Form. Game Γ is given by a tuple $\Gamma = \langle 1^n, A_1, \dots, A_n, M, 1^t \rangle$. The game has n players, and for each player i , where $1 \leq i \leq n$, their set of actions A_i is given by listing all their elements. Given a strategy profile and a player i , $1 \leq i \leq n$, $u_i(a)$ is the output of M on input $\langle a, i \rangle$ after t steps.

A more succinct representation of games [8] is obtained by implicitly defining the sets of actions A_i as subsets of $\{0, 1\}^m$. In such a case a game Γ is given by $\langle 1^n, 1^m, M, 1^t \rangle$, which is called *implicit form*. For reasons that will be clarified later strategic games in implicit form will not be considered.

The second representation assumes that payoff functions are given explicitly by means of a table.

Strategic Game in Explicit Form. Game Γ is given by a tuple $\Gamma = \langle 1^n, A_1, \dots, A_n, T \rangle$, where T is a table of dimensions $|A_1| \times \dots \times |A_n| \times n$. Given a strategy profile and a player i , $1 \leq i \leq n$, then $u_i(a) = T[a][i]$.

Will be considered strategic games in which utility functions are described by boolean formulas. In [16] player i has a goal φ_i to fulfil. Goals are usually described by boolean formulas. The utility of the player is binary. It is 1 if the goal is satisfied and 0 otherwise. Along the lines suggested by circuit games [102], the following family of strategic games will be considered, whose representation is close to a game given in general form [8].

Formula Game in General Form. Game Γ is given by a tuple,

$$\Gamma = \langle 1^n, A_1, \dots, A_n, 1^\ell, (\varphi_{i,j})_{1 \leq i \leq n, 0 \leq j < \ell} \rangle$$

The set of actions for player i , $1 \leq i \leq n$ is $A_i = \{0, 1\}^{m_i}$. The utilities of player i are given by the boolean formulas $\varphi_{i,j}(a_1, \dots, a_n) \in \{0, 1\}$, $0 \leq j < \ell$, and by the equation $u_i(a_1, \dots, a_n) = \sum_{0 \leq j < \ell} \varphi_{i,j}(a_1, \dots, a_n) 2^j$.

Another model for strategic games that uses a boolean formula was introduced in [71], the *weighted boolean formula games*, that is defined in the next paragraph.

Weighted Boolean Formula Game (WBFG) [71]. A game is given by a tuple,

$$\Gamma = \langle 1^n, 1^m, 1^r, 1^\ell, (\mathfrak{F}_i)_{1 \leq i \leq n} \rangle$$

where player i has the set of actions $A_i = \{0, 1\}^m$. For each player i , there is a set $\mathfrak{F}_i = \{(f_{i,1}, w_{i,1}), \dots, (f_{i,r}, w_{i,r})\}$ such that $f_{i,j} : A_1 \times \dots \times A_n \rightarrow \{0, 1\}$, $1 \leq j \leq r$, are boolean formulas and $w_{i,j} \in \{0, 1\}^\ell$, $1 \leq j \leq r$. The utility for player i is computed by the formula $u_i(a_1, \dots, a_n) = \sum_{(f,w) \in \mathfrak{F}_i} w \cdot f(a_1, \dots, a_n)$.

In the above definition, the set of actions is described implicitly. In the rest, descriptions will be restricted to WBFG, in which the set of actions is described explicitly. Following the previous notation to such games will be referred as *weighted boolean formula games in general form*. Formula games and WBFG in general form are equivalent as, given a WBFG we can build a Formula Game Γ' in polynomial time, in the size of Γ with the same utilities and *viceversa*. The details of the proof are given in the following Claim 2.1. Thus, results for Formula Games will be applied also to WBFG.

Claim 2.1 *Given a WBFG $\Gamma = \langle 1^n, 1^r, 1^\ell, (\mathfrak{F}_i)_{1 \leq i \leq n} \rangle$ we can build in polynomial time in the size of Γ , a Boolean Formula Game $\Gamma' = \langle 1^n, 1^\ell, (\varphi_{i,j})_{1 \leq i \leq n, 0 \leq j < \ell} \rangle$ with the same utilities and reciprocally.*

Proof. A WBFG has been transformed into a formula game through a sequence of steps. Let us start to consider a restricted form of utility.

Given $A = \{0, 1\}^r$, $a_i \in \{0, 1\}$ and $a = (a_1, \dots, a_r) \in A$ and given $v(a) = \sum_{1 \leq i \leq r} w_i a_i$, where each w_i has ℓ bits, we can compute in time $O(\ell r)$ formulas φ_j , $0 \leq j < \ell + \log r$, such that $v(a) = \sum_{0 \leq j < \ell + \log r} \varphi_j(a) 2^j$.

To prove the preceding fact, the ideas given in [15] are used. It is defined $x_{i,j}$ to take care in the future, of the bit j of the word w_i , eventually $x_{i,j} = w_{i,j} a_i$. Note that, $y_i = (x_{i,\ell-1}, \dots, x_{i,0})$ is like a number of ℓ bits. Using results given in [95] or [97], a non-uniform TC^0 circuit $\text{IteratedSum}(y_1, \dots, y_r)$ can easily be built in polynomial time, giving the sum of the r numbers each one of ℓ bits. Let us compute the number of outputs of such a circuit.

With ℓ bits, the biggest number written with ℓ bits has value $2^\ell - 1$. Therefore, the sum of r numbers, each one of ℓ bits, is at most $k(2^\ell - 1)$ and this sum can be written with $\ell + \log k$ bits. Therefore, $\text{IteratedSum}(y_1, \dots, y_r)$ outputs $\ell + \log k$ bits. From this circuit, it can be easily obtained a TC^0 circuit giving the bit j of this iterated sum. Since, $\text{TC}^0 \subseteq \text{NC}^1$ and circuits in NC^1 have logarithmic depth and polynomial size, in polynomial time a formula $\phi_j(y_1, \dots, y_r)$ can be found, giving the bit j of $\text{IteratedSum}(y_1, \dots, y_r)$. To obtain the final result, $x_{i,j}$ has to be substituted by $w_{i,j} a_i$ and the following expression is

obtained,

$$\varphi_j(a_1, \dots, a_r) = \phi_j((w_{1,\ell-1}a_1, \dots, w_{r,\ell-1}a_r), \dots, (w_{1,0}a_1, \dots, w_{r,0}a_r))$$

Let us consider another fact.

Given $\mathfrak{F} = \{(f_1, w_1), \dots, (f_r, w_r)\}$ such that, each w_i has ℓ bits and $f_i : \{0, 1\}^n \rightarrow \{0, 1\}$, we can compute in polynomial time, in the size of \mathfrak{F} , formulas φ_j such that $u(a) = \sum_{1 \leq i \leq r} w_i f_i(a) = \sum_{0 \leq j < \ell + \log r} \varphi_j(a) 2^j$.

Let us prove it. Given $b \in \{0, 1\}^r$ consider the utility $v(b) = \sum_{1 \leq i \leq r} w_i b_i$ and using the preceding fact, find ϕ_j such that $v(b) = \sum_{0 \leq j < \ell + \log r} \phi_j(b) 2^j$. Now, we identify $b_i = f_i(a)$ and $\varphi_j(a) = \phi_j(f_1(a), \dots, f_r(a))$. Transformation can be done in polynomial time.

Finally, in order to transform WCFG into boolean formula games, the last fact is applied to each \mathfrak{F}_i .

Transforming a boolean formula game into a WCFG is easy. $w_{i,j} = 2^j$ and $f_{i,j} = \varphi_{i,j}$ are defined.

□

In case that the number of players is constant, with respect to number of actions, an explicit representation can be obtained in polynomial time from a given general form representation, otherwise transformation requires exponential time.

2.3 Notes

Game Theory. Refer to *A Course in Game Theory* of M. Osborne and A. Rubinstein [84] published on 1994, and *Theory of Games and Economic Behavior* of John von Neuman and Oscar Morgenstern [81], in order to learn Game Theory, as well as the mathematical definitions of strategic and extensive games. Note the definition of *Nash equilibrium* for non-cooperative games described in [80] by J. Nash. Finally, a good reference to know more about equilibrium of extensive form games with imperfect information is [46] by A. Gilpin and T. Sandholm.

Morphisms and Complexity in Games. In the field of game isomorphisms refer to [80] by J. Nash, where was first proposed a very *compact* definition of game morphisms. Note results of J. C. C. McKinsey [72] and J. Harsanyi and R. Selten's isomorphisms of strategic games [49] published in 1988. Other interesting reference about game equivalence, are by J. van Benthem [111] and by B. de Bruin [29].

The book [45] *Computers and Intractability: A Guide to the Theory of NP-completeness* from M. Garey and D. Johnson, it is crucial to understand the complexity of games. The reference *The complexity of pure Nash equilibria* [34], is

focused on congestion games, investigating multi-player games that are guaranteed to have pure Nash equilibria.

Part I

Angel-Daemon Games

Chapter 3

Preliminaries on Web Orchestrations

This chapter is focused on orchestrations and failures in systems, like web-services. Orchestration together with choreography, are two prominent approaches for the composition of services describing two aspects of creating business processes from composite web-services. So that, we describe some details of these two viewpoint and their relationship. Consider also, the Orc language as a useful tool that allow us to orchestrate distributed services and Web applications. Finally, we give our alternative approach based on game theory in order to analyse the behaviour of orchestrations over unreliable environments.

3.1 Web-services and Orchestration versus Choreography

There are two fundamental approaches for the composition of web-services: *orchestration* and *choreography*, describing two aspects of creating business processes. Essentially, orchestration always represents control from one perspective party, where the activities of the composed services are coordinated by a specific component, called the *orchestrator*. This orchestrator calls the composed services and collects their responses. Unlike orchestration, choreography tracks message sequences among multiple parties and sources: message exchanges occur between web-services. This last, also support a high level description of peer-to-peer interactions among services, which communicate without the participation of any orchestrator. In the figure 3.1, the relationship between orchestration and choreography at a high level, is shown.

Choreographies. Choreography languages represent an alternative approach for service composition with respect to orchestrations, due to it supports a high level description of interactions among services, which directly communicate each other. Precisely, M. Bravetty and G. Zavattaro in his work [19] addressed the question of the deployment of service compositions through choreography requirements in the context of service oriented com-

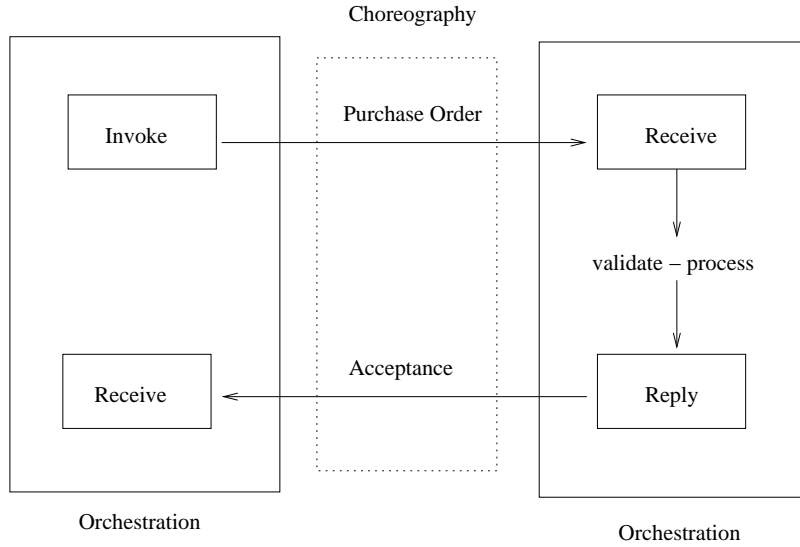


Figure 3.1: Orchestration is an executable process, which may interact with internal and external Web services. It is a process that describes how Web services can interact at the message level.

puting, and formalizing a service choreographies and service contracts through process calculi. The Example 1 described by them is considered here, as an example of service composition.

Example 3.1 [Reservation via Travel Agency [19]] Consider a travel agency service that can be invoked by a client in order to reserve both an air-plane seat and a hotel room. In order to satisfy the client's request, the travel agency contacts two separate services: one for the air-plane reservation and one for the hotel reservation. A choreographic specification of this service composition describes the possible flows of invocations exchanged among the four sites: the client, the travel agency, the airplane reservation service, and the hotel reservation service.

In order to better understanding, the proposed choreography definition 1 of [19] is also considered. That is,

Definition 3.1 (Choreographies [19]) Given a countable sets of operations, a, b, c, \dots and sites, r, s, t, \dots , be two countable sets of operation and sites names, respectively. The set of choreographies, H, L, \dots is defined by the following grammar:

$$H ::= a_{r \rightarrow s} \quad | \quad H + H \quad | \quad H; H \quad | \quad H|H \quad | \quad H^*$$

The invocations $a_{r \rightarrow s}$ means that site r invokes the operation a provided by the site s . The operators are: choice "+", sequential ";", parallel "|", and repetition "*". Exists also, two auxiliary terms: 1 and 0, which are considered in the operational semantics, in order to model the completion of a choreography.

Example 3.2 Continuing Example 3.1. Then, consider the choreography composed of these four sites: *Client*, *TravelAgency*, *AirCompany* and *Hotel*.

$$\begin{aligned} & \text{Reservation}_{\text{Client} \rightarrow \text{TravelAgency}}; \\ & ((\text{Reserve}_{\text{TravelAgency} \rightarrow \text{AirCompany}} ; \text{ConfirmFlight}_{\text{AirCompany} \rightarrow \text{TravelAgency}}) | \\ & (\text{Reserve}_{\text{TravelAgency} \rightarrow \text{Hotel}} ; \text{ConfirmRoom}_{\text{Hotel} \rightarrow \text{TravelAgency}})); \\ & \text{Confirmation}_{\text{TravelAgency} \rightarrow \text{Client}} + \text{Cancellation}_{\text{TravelAgency} \rightarrow \text{Client}} \end{aligned}$$

The *Client* initially sends a reservation request to a travel agency, that subsequently contacts in parallel an airplane company *AirCompany* and a room reservation service *Hotel* in order to reserve both the travel and the staying of the client. Then, the travel agency either confirms or cancels the reservation request of the client.

But, the researchers defined an automatic procedure, which can be used to check whether a service (in a choreography) correctly plays one of the roles described by the choreography. In the specific example of travel agency, given a choreographic specification of this, and an actual service (of the travel agency), check whether the current service behaves correctly according to the choreographic specifications.

For doing that, they proposed the combination of choreography with service contract [21] refinement, which describes the sequence of input/output operations in a session of interaction with other services.

Definition 3.2 [Choreography implementation [19]] Given the choreography H and the system P , we say that P implements H if,

- P is a correct contract composition and
- given a sequence w of labels of the kind $a_{r \rightarrow s}$, if $P \xrightarrow{w \surd} P'$, then there exists H' such that $H \xrightarrow{w \surd} H'$.

$P \xrightarrow{w \surd} P'$ denotes the sequence of transitions with successful termination.

Example 3.3 Continuing Example 3.1 and 3.2. Then, a possible implementation (see Example 2 of [19]) of choreography is as follows.

$$\begin{aligned} & \overline{[\text{Reservation}_{\text{TravelAgency}} ; \text{Confirmation}]_{\text{Client}}} || \\ & \overline{[\text{Reservation}; (\overline{[\text{Reserve}_{\text{AirCompany}} ; \text{ConfirmFlight}]_{\text{TravelAgency}}} | \\ & \overline{[\text{Reserve}; \overline{[\text{ConfirmRoom}]_{\text{Hotel}}}]_{\text{AirCompany}} ; \overline{[\text{Confirmation}_{\text{Client}}]}]_{\text{TravelAgency}}} || \\ & \overline{[\text{Reserve}; \overline{[\text{ConfirmFlight}_{\text{TravelAgency}}]}]_{\text{AirCompany}}} || \\ & \overline{[\text{Reserve}; \overline{[\text{ConfirmRoom}_{\text{TravelAgency}}]}]_{\text{Hotel}}} \end{aligned}$$

In this implementation, the travel agency always replies positively to the request of the client sending a *Confirmation* message.

Orchestration (Orc) Language. Orchestration requires the coordination among participants services. It is a coordination of multiple component services, in order to create a business process. In this sense, an orchestration language, Orc, developed by J. Misra and W. Cook [59, 76] provides uniform access to computational services, including distributed communication and data manipulation, through the set site calls. This set of sites calls can be orchestrated into a complex computation by means of an Orc expression [76] which interacts with services, and manages timeouts, priorities, and failure of sites or communication. A service provided by a site, may be unreliable because of site overuse, site failure or network congestion. In Orc, a site call either returns a result or remains in silence, corresponding this fact to a site failure.

A *site* accepts an argument and *publishes* a result value¹. A call to a search engine, $find(s)$, may publish the set of sites which *currently* offer service s . A site is *silent* if it does not publish a result. Site calls may induce *side effects*. A site call can publish *at most one response*. Although a site call may have a well-defined result, it may be the case that a call to the site, fails (silence) in an untrusted environment. Orc contains a number of inbuilt sites: 0 is always silent while $1(x)$ always publishes x . It is as well provides a special site $if(b)$, which publishes a signal if its boolean argument b is true and remains silent otherwise.

In Orc, the site calls is combined by three composition operations:

- Sequence $P > x > Q(x)$: For each output x , published by P , an instance $Q(x)$ is executed. If P publishes the stream of values, v_1, v_2, \dots, v_n , then $P > x > Q(x)$ publishes some interleaving of the set $\{Q(v_1), Q(v_2), \dots, Q(v_n)\}$. When the value of x is not needed we note $P \gg Q$.
- Symmetric Parallelism $P | Q$: Publishes *some* interleaving of the values published by P and Q .
- Asymmetric parallelism P where $x : \in Q$: In this case, P and Q are evaluated in parallel. P may become blocked by a dependency on x . The first result published by Q is bound to x , the remainder of Q 's evaluation is terminated and evaluation of the blocked residue of P is resumed.²

In Orc language, the expressions $M \gg (R|T)$ and $(M \gg R)|(M \gg T)$ are different. In the first expression, one call is made to M , while T and R sites are called after M responds. In the second one, two parallel calls are made to M , while T and R sites are called, only, after the corresponding calls respond. In this expression, M site publishes different values

¹The words “publishes”, “returns” and “outputs” are used interchangeably. The same way, the terms “site” and “service” are also used interchangeably.

²Written also, as $P < x < Q$: This expression evaluates P and Q independently, but the site calls in P (which depends on x) are suspended until x is bound to a value. The first value from Q is bound to x , evaluation of Q is then terminated and suspended calls in P are resumed. The values published by P are the ones published by $P < x < Q$.

on each call, and the sites, T and R use these values. In Figure 3.2 these two different expressions, are shown.

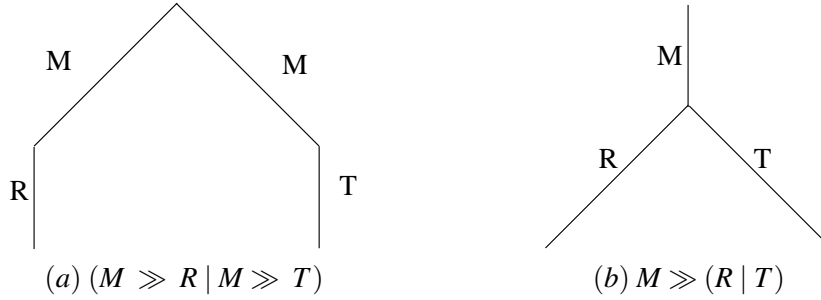


Figure 3.2: In case (a), two parallel call are made to M , while R and T are called only after the corresponding calls respond. In case (b), one call is made to M , and R and T are called after M responds. The major difference is that, in case (a), R and T sites use different values published by the M site, on each call.

Different examples from web-services are shown, in order to illustrate the useful of these composition operators.

Example 3.4 Let us consider, for instance, the following example of J. Misra [76],

$MailOnce(a) \triangleq Email(a, m)$ where $m \in (CNN(d) \mid BBC(d))$ In this example, first publication of CNN or BBC of a given date d , is sent to address a .

Definition 3.3 (Output) The number of outputs publishes by an Orc expression E is defined by $out(E)$.

Another version, of this example, is the following.

Example 3.5 [Email News]. A call to site CNN publishes a digital version of the current newspaper. A call to CNN increases by one the site counter. Site BBC is similar. A call to $Email(a, m)$ sends the email m to the address a . Publishes also, the signal sent to denote completion. We consider restricted versions like $EmailAlice(m)$ or $EmailBob(m)$. Let us consider the orchestration, where sites CNN and BBC are called in parallel, $Two_Each = (CNN \mid BBC) > x > (EmailAlice(x) \mid EmailBob(x))$ Suppose that CNN responds first and the result is emailed to Alice and Bob. Later on BBC will answer and Alice and Bob get another newspaper. The orchestration publish 4 times sent, denoted as $out(Two_Each) = 4$. In $One_Each = (CNN > x > EmailAlice(x)) \mid (BBC > x > EmailBob(x))$, Alice receives the CNN and Bob the BBC and $out(One_Each) = 2$.

An Orc expression may be naming. This allow use the expression name in its own definition, getting a recursive definition. Below, a recursion example is shown.

Example 3.6 *Recursion is allowed, as may be seen in this example given by J. Misra given in [76], where author presents an expression that may call itself,*

$$\text{MailForever}(a) \triangleq \text{MailOnce}(a) > x > \text{MailForever}(a),$$

which keeps sending newspapers to a.

Therefore, Orc language supports a set of internal sites call:

- `if(b)`: Returns a signal if its boolean arguments is true and remains silent otherwise.
- `let(x)`: Immediately publishes its argument. (*let* is often abbreviated to `1`).
- `Rtimer(t)`: (*Relative timer*), provides a timing operation via an internal site call. This site call returns a signal after t time units. *Rtimer* may be used to program timeouts.
- `Timeout`: May be used to call site s repeatedly until it returns a result.
- `Clock`: Publishes a current time at the server of this site.

An orchestration scheme allows timeouts, interruptions, and evaluations. An examples of Orc schemes allowing timeouts and `rtimer` are shown.

Example 3.7 *Let us consider the following Orc expression,*

$$\text{TimeOut} \triangleq \text{let}(z) \text{ where } z : \in (f \mid \text{Rtimer} \gg \text{let}(3))$$

This expression either publishes the first publication of f , or timeout after t units and publishes a value 3. In this expression `let` is an internal site call which publishes its argument. In this case, a value 3.

Example 3.8 *A typical programming paradigm is to call site M and publish a pair (x, b) as the value, where b is true if M publishes x before the time-out, and false if there is a time-out. In the latter case, x is irrelevant. Below, z is the pair (x, b) .*

$$\text{Refine}(M) \triangleq \text{let}(z) \text{ where } z : \in (M > x > \text{let}(x, \text{true}) \mid \text{Rtimer}(t) > x > \text{let}(x, \text{false}))$$

Example 3.9 *As more involved example, call `Refine` repeatedly starting with some initial argument, and use a publication as the argument for the next call and publish the last value (the most refined) that is received before time t . The `BestRefine(t, x)` expression implements this specification. It publishes x if there is a time-out; else it publishes `BestRefine(t, y)` where y is the value published by `Refine` before the time-out.*

$$\begin{aligned} \text{BestRefine}(t, x) &\triangleq \text{if}(b) \gg \text{BestRefine}(t, y) \mid \text{if}(\neg b) \gg \\ &\text{let}(x) \text{ where } (y, b) : \in (\text{TimeOut}(x) > y > \text{let}(y, \text{true}) \mid \text{Refine}(t, y) > y > \text{let}(y, \text{false})) \end{aligned}$$

Here the parameter t of $BestRefine(t, x)$ is an absolute time.

3.2 Orchestration and Game Theory

Failures in web-service applications can be treated through the use of timeouts and corrective action, but such defensive programming not always be easy and in some cases it is even impossible. Because of this, it is useful to have an estimate of the likelihoods of success. In practice, it may be difficult to provide a meaningful measure of site reliability and the assumption that distinct sites are independent may be too strong. By this reason, it is interesting to use game theory as alternative approach, in order to analyse the behaviour of orchestrations over unreliable environment. For doing this analysis, we are partitioned sites into two disjoint sets, angels \mathcal{A} and daemons \mathcal{D} :

- Sites in \mathcal{A} fail in such a way as to minimize damage to an application. This kind of failure is called *angelic*.
- Sites in \mathcal{D} fail in such a way as to maximize damage to the application. This kind of failure is called *daemonic*.

It is assumed that the number of possible failures in the sets \mathcal{A} and \mathcal{D} are bounded. Both sets may be viewed as players in a strategic game. If only *angels* are present then the problem is a maximisation one; if only *daemons* act then we have a minimisation problem. The interesting case lies between the extremes, when both angels and daemons act simultaneously and a competitive situation arises that may be represented by a so-called *angel-daemon* game. Here, finding a Nash equilibrium, if it exists, gives a solution that may be used to model realistic situations for unreliable web-service, where the outcome is found somewhere between over-optimism and over-pessimism.

3.3 Notes

Orchestration and Choreography. Note this interesting reference [19] about *orchestration* and *choreography*. The authors proposed an effective procedure, used to verify whether a service with a given contract can correctly play a specific site within a choreography. Another needed references about these two terms are [91, 14].

In order to read about Service Oriented Computing (SOC), you must refer to M. P. Papazoglou and D. Georgakopoulos [88]. Refer to [109], to better understanding orchestration languages such as XLANG, which consists in a messaging language with some of the expression capabilities of C#. In the same line, refer to an XML language WSFL [68], which describe both public and private process flows. Note also, an WS-BPEL [54] language, which provides support for both executable and abstract business processes.

Orc Language and Web-Services. All information about the orchestration language by J. Misra and W. R. Cook we find in [76, 58, 25, 59]. In order to learn about web-services orchestrations, note to Sidney Rosario et. al. [99]. In addition, refer to [24] by W. R. Cook and J. Bartfield and [56] by G. Kandaswamy et al., which dealing about web-services too.

Workflows. There have been developed some theoretically sound and very flexible models of concurrency, allowing synchronisation and communication patterns: The process calculi by D. Sangiorgi and D. Walker [101] includes constructs for concurrency and channels for communication and synchronisation, and Petri nets by W. Reisig [98] and J. L. Peterson [94]. Refer to [27] by M. Danelutto and M. Aldinucci, where authors proposed an extension of the set of principles to be satisfied in order to develop an efficient skeletal programming systems. Note also, the W. van der Aalst's patterns [3], and R. Milner [74] and C. A. R. Hoare [51], where may be find information about π -Calculus. In order to read about workflow pattern, refer to the *Workflow Patterns* home page [2].

Game Theory Approach. For learn more about the two lines of works of A. Stewart et al. refer to [107, 106]. A very interesting references is [77] by T. Moscibroda et al. about game theory and distributed systems, and *the reputation* of the sites [104] by G. C. Silaghi.

Chapter 4

Bounded Site Failures: an Approach to Unreliable Web Environments

The behaviour of a web-services orchestration in untrusted environments, analysed by means of game theory, is proposed in this chapter. The performance of web-services may fluctuate: some services may be degraded, while other services may maintain the capacity to withstand the effects of high demand. Here, uncertainty profiles and *angel-daemon* (AD) games are used to analyse service-based behaviours in situations where probabilistic reasoning may not be appropriate. In such a game, an *angel* player, tries to minimise damage to the application, and modifies the behaviour of a bounded number of *angelic* services in a beneficial way. In the opposite side, a *daemon* player tries to maximise damage to the application, and modifies a bounded number of *daemonic* services in a negative way.

4.1 Unreliable Environments and Risk Management

Web-services run on a variety of platforms, and provide interaction among different applications or services, making important the analysis of their possible failures. The behaviour of a services set may fluctuate, and demand for a particular service may change. The sequence and conditions in which one web-service invokes other web-services in order to perform a useful function, has been defined as orchestration [91] in Chapter 3. User-defined orchestrations may include dynamic mechanisms for altering behaviour depending on the responsiveness of underlying web-services. The robustness of an orchestration to underlying service failures is referred to as *resilience*. Therefore, an orchestration of web-services is modelled using an Orc expression [76] in which services are called upon to perform sub-computations. An Orc expression specifies how a set of service calls are orchestrated to perform a certain overall goal.

In order to analyse the expected number of results that will be published by an orches-

tration's expression, probability theory has been used. To these expressions each site S is usually assumed to have a probability of failure, and distinct sites are assumed to be independent. However, this assumption may be too strong. On the other hand, in practice it may be difficult to provide a meaningful measure of site reliability. Therefore, an alternative approach based on game theory is used in this chapter to analyse the behaviour of orchestrations over unreliable environments.

For orchestrations, the behaviour of web-services can be modelled by defining a services-based game. This game includes two players: angel A and daemon D , which control all the sites in the orchestration. Sites in A fail in such a way to minimise damage to an application, while sites in D fail in such a way to maximise damage to the application. Neither angels nor daemons can fail excessively because the number of failures, in both cases, is bounded. Thus, A and D can be viewed as players in a strategic game. If only angels are present, then the problem is of maximisation, while if only daemons act then we have a minimisation problem. Therefore, the interesting case occurs when both angels and daemons act simultaneously into a competitive situation, which can be represented by a so-called *angel-daemon*(AD) game. Here, if a Nash equilibrium exists, it gives a solution which may be used to model realistic situations for unreliable Web environments.

Risk Management and Uncertainty Profiles. To analyse the behaviour of network, when a bounded number of services fails, an *uncertainty profile* is introduced. This concept is sufficiently rich to allow the analysis of stressed Web environments, where networks may be under the influence of competing tendencies, one destructive, and the other self-correcting. In a uncertainty profile, bounds are placed on both the constructive and destructive capacities, for example, an unreliable network may have no self-correcting behaviour and no angel player.

An uncertainty profile of an orchestration is a parameterised description of a game, which specifies sets of angel and daemon services as well as, the failure capacities of both the angel and daemon players. A uncertainty profile provides an abstract, qualitative description of uncertainty (without recourse to probability). An analysis of the game associated to an uncertainty profile can be used to derive an algorithmic measure of orchestration uncertainty. To our knowledge, the use of uncertainty profiles to model stressed Web environment is new.

Web-services involve the discovery and utilisation of services. It is often the case that a service is made available by a number of providers. The performance of a provider can vary greatly over time (although service level agreements (SLAs) may provide information about "normal" expected performance). Brokers [13], are often used to monitor provider performance and to provide an interface to the "best" current provider.

It is usually the case that the performance of a provider deteriorates as demand increases, although “elastic” providers may call on extra servers in times of peak demand. Thus, in such situations performance, can conceivably improve.

In this thesis, it has been assumed that, first, providers should be treated and secondly, users behave in non-cooperative way, since Web users alternate between providers in times of high demand. Given these assumptions, it is reasonable to model the behaviour of a set of providers (resources) in a stressed environment as a strategic situation in algorithmic game theory [65], [82], [100]. The notion of a Nash equilibrium is used to derive an efficient broker allocation of providers to users, as shown in Example 4.1.

Example 4.1 [Brokering in an idealised environment]. *Consider a situation where a set of users $\{1, \dots, n\}$ submit jobs for execution to a broker. Suppose that the broker uses multiple predictable service providers (resources) $R = \{r_1, \dots, r_k\}$ to meet demand. The broker allocates services providers to jobs in such a way as to minimise user delay. This situation can be modelled by a non-cooperative game with n players in which users “move” in sequence by allocating (or relocating) their job to a provider. Providers may have modified work loads and delays as a consequence of a sequence of “moves”. A Nash equilibrium is an allocation schedule in which no user can improve their situation by making a move.*

The Web is composed of a very diverse range of resources. Such heterogeneity contributes to the complexity of a Web environment. Performance variability and sporadic unavailability of underlying networks provide further complications. Conventionally, unreliability is treated from a probabilistic viewpoint [10], [92], [93]. In contrast, a variety of provider behaviours within a stressed Web environment using non-cooperative game theory, has been investigated on this thesis.

Example 4.2 [Brokering in an idealised environment]. *Now consider a more realistic refinement of the brokering Example 4.1 where provider and network behaviour is less predictable. The following assumptions are made about stressed Web environments.*

- *stress is non-uniformly distributed across the Web;*
- *patches of stress can move dynamically in response to users moving jobs from stressed regions to more responsive providers;*
- *the performance of certain providers may be highly vulnerable to heavy workloads, while other providers may incorporate autonomic behaviour, which increases the number of servers on offer in response to increased demand (elasticity). Consequently, some providers may be associated with increased unreliability at times of stress while others may exhibit robust behaviour.*

An extended form of non-cooperative game is used to reason about brokering in stressed environments. In addition to the n users, the game also contains two extra players: a daemon player, who selects a number of sites to be stressed so as to maximise the delay associated with the game; and an angel, who selects a number of sites so as to minimise delay. Unreliability is described by the notion of an uncertainty profile which specifies, a priori, possible angel and daemon behaviours. Given an uncertainty profile, the behaviour of a broker in a stressed environment can be described by an associated $n + 2$ player game.

4.1.1 Orchestrations: The Number of Published Values

Given a complex orchestration E , it is *unrealistic to assume that there will be no site failures*, when this orchestration is executed.

Reliability assumption. Sites are unreliable and can fail. When a site fails it remains silent and delivers no result at all. When a site does not fail it delivers the correct result. Any kind of Byzantine behaviour is excluded. Any kind of behaviour delivering an *approximate* result is also excluded.

Given an orchestration E , let $\alpha(E)$ be the set of sites that are called in orchestration E and $\alpha_+(E) = \alpha(E) \setminus \{0\}$. The behaviour of the evaluation of E in this environment is given by replacing all occurrences of sites $s \in \mathcal{F}$, by 0. Let $fail_{\mathcal{F}}(E)$ denote this expression. Even though some sites fail, orchestration may still produce useful partial results. Consider the following Examples.

Example 4.3 [Email News under Failures]. Consider the Example 3.5 of Chapter 3. When there are no failures, it holds that, $out(Two_Each) = 4$, $out(One_Each) = 2$. When $F = \{CNN\}$ we have $fail_{\{CNN\}}(CNN) = 0$ and $fail_{\{CNN\}}(s) = s$ in other cases.

As $(0 \mid BBC) = BBC$ we have,

$$fail_{\{CNN\}}(Two_Each) = BBC > x > (EmailAlice(x) \mid EmailBob(x)) \text{ and} \\ out(fail_{\{CNN\}}(Two_Each)) = 2.$$

Another example of faulty orchestration is, $fail_{\{CNN, EmailAlice(x)\}}(Two_Each) = BBC > x > EmailBob(x)$ and $out(fail_{\{CNN, EmailAlice(x)\}}(Two_Each)) = 1$.

Example 4.4 [Failure Expression]. Consider the expression $(S > x > (P \mid Q))$, where S is a site. However, if S fails to publish, then this will result in a catastrophic failure in the evaluation of $S > x > (P \mid Q)$. If site T has the same functionality as S , then the orchestration $(P \mid Q)$, where $x \in (T \mid S)$, will have the same functionality as $S > x > (P \mid Q)$, while being more robust.

Example 4.5 [Multiple Job]. Consider a physicist who wishes to execute a program on a large number of data sets. The jobs will be executed in parallel on a computational Web. In order to draw conclusions, the physicist requires that at least a certain percentage of jobs return results. Suppose that the set of parallel jobs are executed on sites S_1, \dots, S_n , and that each site, has an associated reliability rating. When will the physicist will receive sufficient experimental results?

The evaluation may still have value to the orchestrator provided that a certain minimum number of results are published.

Value assumption. The evaluation of an orchestration has value even if some sites fail. For a particular failure set \mathcal{F} , the usefulness of the evaluation of E in failure environment $fail_{\mathcal{F}}(E)$, is measured by $out_{\mathcal{F}}(E)$, the number of outputs out , published by an expression.

A function out , should be such that:

- the range of out should be non-negative \mathbb{R} .
- the value of out must be 0 when all sites fail.
- $out_{\mathcal{F}}(E) \geq out_{\mathcal{F}'}(E)$ when $\mathcal{F} \subseteq \mathcal{F}' \subseteq \alpha(E)$.
- $out(E)$ and $out_{\mathcal{F}}(E)$ have a low computational complexity.

Let $out(E)$, be the number of outputs published by a non-blocking expression E . Given $E(z)$ depending on an input variable z . The behaviour of this expression is denoted by $E(\perp)$, when the value of z is undefined. This is equivalent to replacing by 0 all sub-expressions having a dependency on z . $E(z)$ is kept to represent the evaluation when z is defined.

The benefit by the number of outputs published by E is measured, where $out(E) =$ numbers of outputs published by E .

Lemma 4.1 ([39]) Let E_1, E_2 be non-blocking well formed Orc expression, the number of publications, $out(E)$,

$$\begin{aligned}
 out(0) &= 0, \quad out(s) = 1 \text{ if } s \text{ is a site } s \neq 0, \\
 out(E_1 | E_2) &= out(E_1) + out(E_2), \quad out(E_1(z) \gg E_2) = out(E_1) * out(E_2) \\
 out(E_1 \text{ where } z : \in E_2) &= \begin{cases} out(E_1) & \text{if } out(E_2) > 0 \\ out(E_1(\perp)) & \text{if } out(E_2) = 0 \end{cases}
 \end{aligned}$$

Given a non-blocking well formed Orc expression E , $out(E)$, can be computed in polynomial time with respect to the length of the expression E .

Proof. The computation bounds follow from standard techniques. It holds that, $\text{out}(0) = 0$, and $\text{out}(1) = 1$. Given a call to a single site s , it holds that, $\text{out}(s(v_1, \dots, v_k))$ is 1 if all the parameters are defined and 0 otherwise. Let E_1, E_2 be non-blocking, well formed Orc expressions, $\text{out}(E_1|E_2) = \text{out}(E_1) + \text{out}(E_2)$, and $\text{out}(E_1 > z > E_2(z)) = \text{out}(E_1) * \text{out}(E_2(z))$. Finally $\text{out}(E_1(z) < z < E_2)$ is equal to $\text{out}(E_1(z))$ if $\text{out}(E_2) > 0$ and $\text{out}(E_1(\perp))$ otherwise.

Here $E(\perp)$ denotes the behaviour of E , when z is undefined; $E(\perp)$ is found by replacing all service calls with a z -dependency by 0. Therefore, given a non-blocking, well formed Orc expression E , $\text{out}(E)$, can be computed in polynomial time with respect to the length of the expression E . \square

Example 4.6 Consider the following expression,

$$E = (M_1|M_2) > x > [(M_3|M_4) > y > (M_5(x) > z > M_6(z) | (M_7|M_8) \gg M_9(y))]$$

Then $\text{out}(E) = 12$. If site M_1 fails the benefit is $\text{out}(E') = 6$ where,

$$E' = (0|M_2) > x > [(M_3|M_4) > y > (M_5(x) > z > M_6(z) | (M_7|M_8) \gg M_9(y))]$$

4.2 Assessing Orchestrations

In this section, a method for partitioning a set of sites into angels and daemons, based on ranking is proposed.

Reliability ranking assumption. Given an Orc expression E , it is assumed that a ranking containing $\alpha(E)$ is available. This ranking is a measure (“objective” or “subjective”) of the reliability of the sites. This ranking can be independent of any orchestration E , or conversely, can depend strongly on the structure of E . Let $rk(s)$ be the rank of site s .

An orchestration assessor may use such a ranking to partition, a set of sites $\alpha(E)$, into angel and daemon sets as follows:

$$\mathcal{A} = \{S \mid S \in \alpha(E), rk(S) \geq \lambda_E\}, \quad \mathcal{D} = \{S \mid S \in \alpha(E), rk(S) < \lambda_E\}$$

λ_E is a *reliability degree parameter* fixed by the assessor, following the suggestions of the client. How λ_E is determined has not been considered in this thesis. The assessor will perform an analysis, where sites in \mathcal{A} perform *as well as possible*, and sites in \mathcal{D} perform *as badly as possible*. This is a way to perform an analysis lying between the two possible extremes “all is good” or “all is bad”. It would be possible to set up the following assumption: Sites with a rank higher than λ_E are believed by the assessor to

have non-destructive behaviour. Sites with a rank lower than λ_E are unknown entities as far as the assessor is concerned and can have highly-destructive behaviour. The assessor supposes that during an evaluation of E , a number of sites will fail:

- Let a small fraction β_E of angelic sites fail during the evaluation. The number of failing angels is $\beta_E \times \#a = f_{\mathcal{A}}$.
- Let a fraction γ_E of daemon sites fail. The number of failing daemons is $\gamma_E \times \#d = f_{\mathcal{D}}$.

In an abstract way, given an expression E , a partition of $\alpha_+(E)$ into two sets \mathcal{A} and \mathcal{D} is assumed. An analysis is conducted, where sites in \mathcal{A} perform *as well as possible*. If $\alpha(E) = \mathcal{A}$, evaluation of the behaviour can be determined by solving a maximisation problem, while sites in \mathcal{D} *maximise damage to the application*. If $\alpha(E) = \mathcal{D}$, evaluation of the behaviour can be determined by solving a minimisation problem. The uncertainty profile gives a perception of risk from an assessor's point of view.

Definition 4.1 *Given an Orc expression E , the tuple $\mathcal{U} = \langle E, \mathcal{A}, \mathcal{D}, f_{\mathcal{A}}, f_{\mathcal{D}} \rangle$ is an uncertainty profile for an orchestration E , where $\mathcal{A} \cup \mathcal{D} = \alpha_+(E)$, $\mathcal{A} \cap \mathcal{D} = \emptyset$, $f_{\mathcal{A}} \leq \#\mathcal{A}$ and $f_{\mathcal{D}} \leq \#\mathcal{D}$.*

Example 4.7 *Let us consider how to assess SEQ_of_PAR and PAR_of_SEQ under two different situations. In the first sites are ranked: $rk(P) > rk(Q) > rk(R) > rk(S)$.*

Assume that the reliability parameter is such that $\mathcal{A} = \{P, Q\}$ and $\mathcal{D} = \{R, S\}$, and moreover, $1/2$ of angelic sites will fail and $1/2$ of daemon sites will also fail (therefore $f_{\mathcal{A}} = f_{\mathcal{D}} = 1$). Then, the uncertainty profile is,

$$\mathcal{U}_1 = \langle \text{SEQ_of_PAR}, \{P, Q\}, \{R, S\}, 1, 1 \rangle, \quad \mathcal{U}_2 = \langle \text{PAR_of_SEQ}, \{P, Q\}, \{R, S\}, 1, 1 \rangle$$

Another possible rank could be, $rk'(P) > rk'(Q) > rk'(R) > rk'(S)$. Taking $\mathcal{A} = \{P, R\}$, $\mathcal{D} = \{Q, S\}$ and $f_{\mathcal{A}} = f_{\mathcal{D}} = 1$, the uncertainty profiles $f_{\mathcal{A}} = f_{\mathcal{D}} = 1$ are obtained, and

$$\mathcal{U}_3 = \langle \text{SEQ_of_PAR}, \{P, R\}, \{Q, S\}, 1, 1 \rangle, \quad \mathcal{U}_4 = \langle \text{PAR_of_SEQ}, \{P, R\}, \{Q, S\}, 1, 1 \rangle$$

Of course, other profiles are possible.

4.3 Two Player Games: The Angel-Daemon Case

Given a uncertainty profile $\mathcal{U} = \langle E, \mathcal{A}, \mathcal{D}, f_{\mathcal{A}}, f_{\mathcal{D}} \rangle$, a competitive situation arises. A strategic situation occurs when E suffers the effect of two players. Game theory [84] can be used to analyse system behaviour. Suppose that the set of failing sites is $a \cup d$,

where $a \subseteq \mathcal{A}$, $\#a = f_{\mathcal{A}}$ and $d \subseteq \mathcal{D}$, $\#d = f_{\mathcal{D}}$. System behaviour is measured by $fail_{(a,d)}(E)$. The rewards (or utilities) of the angelic player α and daemonic player δ are $u_{\alpha}(a,d) = out(fail_{(a,d)}(E))$ and $u_{\delta}(a,d) = -out(fail_{(a,d)}(E))$. This is a zero sum game as $u_{\alpha}(a,d) = -u_{\delta}(a,d)$. When $fail_{(a,d)}(E)$ is executed, player α receives $out(fail_{(a,d)}(E))$ from player δ . The strategy a is chosen (by player α) to increase the value of $u_{\alpha}(a,d)$ as much as possible, while d is chosen (by player δ) to decrease this value as much as possible.

Stable situations are Nash equilibria: a pure Nash equilibrium is a strategy (a,d) , such that player α cannot improve the utility by changing a while player δ cannot reduce the utility by changing d . When players choose strategies using probabilities, we have mixed Nash equilibria. Let (α,β) be a mixed Nash equilibrium. As in zero sum games all the Nash equilibria have the same utilities, an assessor can measure the value of a program E by the utility of $\alpha \in \mathcal{A}$ on any Nash equilibrium. Given a Nash equilibrium (α,β) , the expected benefit of an expression E is given by the *Assessment*, $v(E, rk, \lambda_E, \beta_E, \gamma_E) = out(fail_{(\alpha,\beta)}(E))$. The orchestration under uncertainty profile is assessed, as follows:

Definition 4.2 [Assessment]. *The assessment $v(\mathcal{U})$ of an uncertainty profile \mathcal{U} is defined to be a value of an angel-daemon game $\Gamma(\mathcal{U})$, which is $v(\Gamma(\mathcal{U}))$. The assessment is defined as,*

$$v(\mathcal{U}) = \max_{a \in \Delta_{\alpha}} \min_{d \in \Delta_{\delta}} u_{\alpha}(a,d) = \min_{d \in \Delta_{\delta}} \max_{a \in \Delta_{\alpha}} u_{\alpha}(a,d)$$

Let E be an Orc expression, and assume that $\alpha_+(E)$ is partitioned into two disjoint administrative domains, angel domain \mathcal{A} and daemon domain \mathcal{D} . Like angel α has the ability to control any $f_{\mathcal{A}}$ sites in \mathcal{A} and daemon δ has the ability to control any $f_{\mathcal{D}}$ sites in \mathcal{D} , both have opposed behaviour. Then, consider the following game:

Definition 4.3 [Angel-Daemon Game]. *The zero-sum angel-daemon game associated to uncertainty profile, $\mathcal{U} = \langle \mathcal{E}, \mathcal{A}, \mathcal{D}, f_{\mathcal{A}}, f_{\mathcal{D}} \rangle$, is the game $\Gamma(\mathcal{U}) = \langle \{\alpha, \delta\}, A_{\alpha}, A_{\delta}, u_{\delta}, u_{\alpha} \rangle$, with two players, angel α and daemon δ . The players have the following sets of actions.*

- The angel α chooses $p = f_{\mathcal{A}}$ different failing sites $a = \{a_1, \dots, a_p\} \subseteq \mathcal{A}$. Any call to a site in $\mathcal{A} \setminus a$ is successful. Therefore, $A_{\alpha} = \{a \subseteq \mathcal{A} \mid \#a = f_{\mathcal{A}}\}$.
- The daemon δ chooses $q = f_{\mathcal{D}}$ different failing sites $d = \{d_1, \dots, d_q\} \subseteq \mathcal{D}$. Calls to sites in $\mathcal{D} \setminus d$ are successful. Therefore, $A_{\delta} = \{d \subseteq \mathcal{D} \mid \#d = f_{\mathcal{D}}\}$.

A strategy profile is a tuple $s = (a,d)$, which defines a set of failing sites, $a \cup d = \{a_1, \dots, a_p, d_1, \dots, d_q\}$. Given E and $a \cup d$, the length of $fail_{a \cup d}(E)$ is used to define the utilities as follows: $u_{\alpha}(s) = out(fail_{a \cup d}(E))$, $u_{\delta}(s) = -u_{\alpha}(s)$. Therefore, $\Gamma(\mathcal{U})$ is a zero-sum game because $u_{\delta}(s) + u_{\alpha}(s) = 0$.

A *pure saddle point* is a strategy profile $s^* = (a^*, d^*)$ such that,

$$u_a(a^*, d^*) = \max_{a' \in A_a} \min_{d' \in A_d} u_a(a', d') = \min_{a' \in A_a} \max_{d' \in A_d} u_a(a', d')$$

A pure saddle point does not necessarily exist (see the second game showed in Figure 4.1). A mixed strategy for $a \in \mathcal{A}$ is a probability distribution $\alpha : A_a \rightarrow [0, 1]$, and similarly for $d \in \mathcal{D}$. Let Δ_a and Δ_d be the set of mixed strategies for $a \in \mathcal{A}$ and $d \in \mathcal{D}$. A mixed strategy profile is a tuple (α, β) , and the expected utility for $a \in \mathcal{A}$ is $u_a(\alpha, \beta) = \sum_{(a,d) \in A_a \times A_d} \alpha(a)\beta(d)u_a(a,b)$. It is well known [81] that there always exists a mixed saddle point (α^*, β^*) satisfying,

$$u_a(\alpha^*, \beta^*) = \max_{\alpha' \in \Delta_a} \min_{\beta' \in \Delta_d} u_a(\alpha', \beta') = \min_{\beta' \in \Delta_d} \max_{\alpha' \in \Delta_a} u_a(\alpha', \beta')$$

Moreover, all the existing saddle points give the same utility. Such a value therefore, is called the *value* of the game $\Gamma(\mathcal{U})$. Simple cases without pure saddle points are shown in Example 4.8.

Example 4.8 Consider the two well-known classical parallel workflow patterns (see [18]), *SEQ_of_PAR* and *PAR_of_SEQ*. *PAR* represents parallel execution (e.g. a farm) while *SEQ* is sequential composition. In Figure 4.1, the workflows *SEQ_of_PAR* and *PAR_of_SEQ* are analysed using angel-daemon games with the utility function *out*. A “sequential composition of parallel processes” (*SEQ_of_PAR*) is denoted by $\triangleq (P \mid Q) \gg (R \mid S)$ while a “parallel composition of sequential expressions” (*PAR_of_SEQ*) is denoted by $\triangleq (P \gg Q) \mid (R \gg S)$.

However, pure saddle points appear in the following example:

Example 4.9 [A kind of multiple job]. A farm, is an embarrassingly parallel computation defined as $FARM_n \triangleq (S_1 \mid \dots \mid S_n)$. For any $\mathcal{U} = \langle FARM_n, \mathcal{A}, \mathcal{D}, f_{\mathcal{A}}, f_{\mathcal{D}} \rangle$, it is easy to prove $v(\mathcal{U}) = n - f_{\mathcal{A}} - f_{\mathcal{D}}$. An angel failure has the same effect as a daemon failure (because of the simple structure of $FARM_n$). A sequential composition of farms [27] can be analysed using (AD) games. The sequential composition of $k > 0$ farms, is defined for $k = 1$ as $SEQ_of_FARM_{n,1} \triangleq FARM_n$ and for $k > 1$ as $SEQ_of_FARM_{n,k} \triangleq FARM_n \gg SEQ_of_FARM_{n,k-1}$. For the profile $\mathcal{U}_{SEQ} = \langle SEQ_of_FARM_{n,k}, \mathcal{A}, \mathcal{D}, f_{\mathcal{A}}, f_{\mathcal{D}} \rangle$ we have $v(\mathcal{U}_{SEQ}) = (n - f_{\mathcal{A}} - f_{\mathcal{D}})^k$.

The players can choose the actions using probabilities. A mixed strategy for the angelic player a is a probability distribution $\alpha : A_a \rightarrow [0, 1]$ such that, $\sum_{a \in A_a} \alpha(a) = 1$. Similarly, a mixed strategy for the daemon player d is a probability distribution $\beta : A_d \rightarrow [0, 1]$.

$$\begin{aligned}
& SEQ_of_PAR \triangleq (P \mid Q) \gg (R \mid S), \quad PAR_of_SEQ \triangleq (P \gg Q) \mid (R \gg S) \\
& \mathcal{U}_1 = \langle SEQ_of_PAR, \{P, Q\}, \{R, S\}, 1, 1 \rangle, \quad \mathcal{U}_2 = \langle SEQ_of_PAR, \{P, R\}, \{Q, S\}, 1, 1 \rangle \\
& \mathcal{U}_3 = \langle PAR_of_SEQ, \{P, Q\}, \{R, S\}, 1, 1 \rangle, \quad \mathcal{U}_4 = \langle PAR_of_SEQ, \{P, R\}, \{Q, S\}, 1, 1 \rangle
\end{aligned}$$

$$\begin{array}{cccc}
\begin{array}{c} \mathfrak{d} \\ R \quad S \\ \mathfrak{a} \begin{array}{c} P \\ Q \end{array} \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \end{array} &
\begin{array}{c} \mathfrak{d} \\ Q \quad S \\ \mathfrak{a} \begin{array}{c} P \\ R \end{array} \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array} \end{array} &
\begin{array}{c} \mathfrak{d} \\ R \quad S \\ \mathfrak{a} \begin{array}{c} P \\ Q \end{array} \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \end{array} &
\begin{array}{c} \mathfrak{d} \\ Q \quad S \\ \mathfrak{a} \begin{array}{c} P \\ R \end{array} \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \end{array} \\
v(\mathcal{U}_1) = 1 & \quad v(\mathcal{U}_2) = 1/2 & \quad v(\mathcal{U}_3) = 0 & \quad v(\mathcal{U}_4) = 1/2
\end{array}$$

Figure 4.1: Angel-daemon games for *SEQ_of_PAR* and *PAR_of_SEQ* in different environments. The utility $u_{\mathfrak{a}}$ is given by the number of outputs of the Orc expression. Games $\Gamma(\mathcal{U}_1)$ and $\Gamma(\mathcal{U}_3)$ have pure saddle points. $\Gamma(\mathcal{U}_2)$ and $\Gamma(\mathcal{U}_4)$ games, have mixed saddle points.

A mixed strategy profile is a tuple (α, β) and the utilities are,

$$u_{\mathfrak{a}}(\alpha, \beta) = \sum_{(a,d) \in A_{\mathfrak{a}} \times A_{\mathfrak{d}}} \alpha(a)\beta(d)u_{\mathfrak{a}}(a,b) \quad u_{\mathfrak{d}}(\alpha, \beta) = \sum_{(a,d) \in A_{\mathfrak{a}} \times A_{\mathfrak{d}}} \alpha(a)\beta(d)u_{\mathfrak{d}}(a,b)$$

As players \mathfrak{a} and \mathfrak{d} have opposing interests there is a strategic situation and the definition of Nash equilibrium is recalled as a concept solution.

Definition 4.4 [Pure Nash Equilibrium]. A pure Nash equilibrium is a pair $s = (a, d)$, such that, for any a' it holds that, $u_{\mathfrak{a}}(a, d) \geq u_{\mathfrak{a}}(a', d)$ and for any d' it holds that, $u_{\mathfrak{d}}(a, d) \geq u_{\mathfrak{d}}(a, d')$. A mixed Nash strategy is a pair (α, β) with similar conditions.

Example 4.10 Consider how to assess *SEQ_of_PAR* and *PAR_of_SEQ* under two different situations. In the first sites are ranked, $rk(P) > rk(Q) > rk(R) > rk(S)$.

Assume that the reliability parameter is, $\mathcal{A} = \{P, Q\}$ and $\mathcal{D} = \{R, S\}$ and moreover, $1/2$ of angelic sites will fail and $1/2$ of the demonic sites will also fail (therefore $f_{\mathcal{A}} = f_{\mathcal{D}} = 1$). Then, $A_{\mathfrak{a}} = \{P, Q\}$, $A_{\mathfrak{d}} = \{R, S\}$ and the bi-matrix games will be,

$$\begin{array}{ccc}
\begin{array}{c} \mathfrak{d} \\ R \quad S \\ \mathfrak{a} \begin{array}{c} P \\ Q \end{array} \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \end{array} & & \begin{array}{c} \mathfrak{d} \\ R \quad S \\ \mathfrak{a} \begin{array}{c} P \\ Q \end{array} \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \end{array} \\
SEQ_of_PAR & & PAR_of_SEQ
\end{array}$$

In both cases, any strategy profile (pure or mixed) is a Nash equilibrium and assessment,

$$v(\text{SEQ_of_PAR}, rk, 1/2, 1/2, 1/2) = 1$$

$$v(\text{PAR_of_SEQ}, rk, 1/2, 1/2, 1/2) = 0$$

For example, this assessment indicates to the client that, in the present environment, there is a reasonable expectation¹ of obtaining 1 output when executing *PAR_of_SEQ*.

Consider a second case with $rk'(P) > rk'(R) > rk'(Q) > rk'(S)$ such that $\mathcal{A} = \{P, R\}$ and $\mathcal{D} = \{Q, S\}$ with $f_{\mathcal{A}} = f_{\mathcal{D}} = 1$.

		\mathfrak{d}	
		Q	S
α	P	0	1
	R	1	0
		<i>SEQ_of_PAR</i>	

		\mathfrak{d}	
		Q	S
α	P	1	0
	R	0	1
		<i>PAR_of_SEQ</i>	

Here no game has pure Nash equilibria. There is only one mixed Nash equilibrium with $\alpha(P) = \alpha(R) = \beta(Q) = \beta(S) = 1/2$, and in this case the angel has utility $1/2$ and the daemon has utility $-1/2$. In this case, the assessment is $v(\text{SEQ_of_PAR}, rk', 1/2, 1/2, 1/2) = 1/2$, and similarly for *PAR_of_SEQ*. This assessment indicates to the client that, in the present environment, an output of 1 or 0 results (with equal likelihood) is a reasonable expectation.

Example 4.11 Consider the expression $(P | Q | R) \gg (S | T | U)$ with $\mathcal{A} = \{P, Q, S\}$ and $\mathcal{D} = \{R, T, U\}$ with $f_{\mathcal{A}} = 2$ and $f_{\mathcal{D}} = 1$.

		\mathfrak{d}		
		R	T	U
α	$\{P, Q\}$	0	2	2
	$\{P, S\}$	2	2	2
	$\{Q, S\}$	2	2	2
		$(P Q R) \gg (S T U)$		

The pure Nash equilibria are,

$(\langle \{P, S\}, R \rangle, \langle \{P, S\}, T \rangle, \langle \{P, S\}, U \rangle, \langle \{Q, S\}, R \rangle, \langle \{Q, S\}, T \rangle)$ and $\langle \{Q, S\}, U \rangle$ In this case, the assessment says that 2 outputs is the reasonable expectation.

4.4 Maximisation and Minimisation Approaches

The two extremes of behaviour can be determined through angelic and daemonic analysis. In an angelic analysis the viewpoint “world is as good as possible even when failures

¹Here “reasonable expectation” is meant in the everyday sense of the phrase and is not intended to represent a probabilistic outcome.

cannot be avoided” is adopted. In this case, the angel will control all the sites and the daemon has no power at all, formally $\mathcal{A} = \alpha_+(E)$ and $\mathcal{D} = \emptyset$. This corresponds to the uncertainty profile, $\mathcal{U} = \langle E, \alpha_+(E), \emptyset, f_{\mathcal{A}}, 0 \rangle$.

The set of eligible actions for α and \mathfrak{d} are then, $A_{\alpha} = \{a \subseteq \alpha_+(E) \mid \#a = f_{\mathcal{A}}\}$, $A_{\mathfrak{d}} = \{\emptyset\}$.

As $A_{\mathfrak{d}} = \{\emptyset\}$, the only possible choice for \mathfrak{d} is $d = \emptyset$. Given a strategy profile $s = (a, d) = (a, \emptyset)$, the utility verifies $u_{\alpha}(s) = out(fail_{a \cup d}(E)) = out(fail_a(E))$, and α fully controls the situation in $\Gamma(\mathcal{U})$. In fact, α is the only real player because \mathfrak{d} has no choice at all.

Daemonic failures are in a sense the opposite of angelic failures. In this case there is one player, the daemon \mathfrak{d} trying to maximise damage. Formally, $\mathcal{A} = \emptyset$ and $\mathcal{D} = \alpha_+(E)$. This corresponds to the uncertainty profile, $\mathcal{U} = \langle E, \emptyset, \alpha_+(E), 0, f_{\mathcal{D}} \rangle$.

The set of eligible actions for α and \mathfrak{d} are then $A_{\alpha} = \{\emptyset\}$, $A_{\mathfrak{d}} = \{d \subseteq \alpha_+(E) \mid \#d = f_{\mathcal{D}}\}$. As $a = \emptyset$, for any $s = (a, d)$ it holds that, $u_{\mathfrak{d}}(s) = -out(fail_{a \cup d}(E)) = -out(fail_d(E))$.

$u_{\mathfrak{d}}$ can be pictured as a quantity of money that \mathfrak{d} has to pay to α , and naturally \mathfrak{d} is interested in paying as little as possible.

Lemma 4.2 *Uncertainty profiles can be used to maximise or minimise the output as follows,*

- Given $A_{\alpha} = \{a \subseteq \alpha_+(E) \mid \#a = f_{\mathcal{A}}\}$ it holds that, $v(\langle E, \alpha_+(E), \emptyset, f_{\mathcal{A}}, 0 \rangle) = \max_{a \in A_{\alpha}} out(fail_a(E))$
- Given $A_{\mathfrak{d}} = \{d \subseteq \alpha_+(E) \mid \#d = f_{\mathcal{D}}\}$ it holds that,

$$v(\langle E, \emptyset, \alpha_+(E), 0, f_{\mathcal{D}} \rangle) = \min_{d \in D_{\mathfrak{d}}} out(fail_d(E))$$

Proof. Let us consider the first part. In $\Gamma = \Gamma(\langle E, \alpha_+(E), \emptyset, f_{\mathcal{A}}, 0 \rangle)$. The angel α chooses a strategy $a \in A_{\alpha}$ such that, $u_{\alpha}(a, \emptyset) = \max_{a' \in A_{\alpha}} out(fail_{a'}(E))$. Note that (a, \emptyset) is PNE because α cannot choose $a' \in A_{\alpha}$ such that, $u_{\alpha}(a, \emptyset) < u_{\alpha}(a', \emptyset)$, then $v(\Gamma) = u_{\alpha}(a, \emptyset)$. The second part is similar. \square

Example 4.12 *Let us consider the two well-known expressions, SEQ_of_PAR and PAR_of_SEQ introduced in [18].*

Let us analyse both expressions with two failures. First, consider in detail a pure angelic behaviour. Since, the player and the possible set of failures are identified, we have $\mathcal{A} = \{P, Q, R, S\}$ and the set of strategy profiles is $A = \{\{P, Q\}, \{P, R\}, \{P, S\}, \{Q, R\}, \{Q, S\}, \{R, S\}\}$. The utilities are given in the table. In order to maximise the utility, in the case of SEQ_of_PAR, the angel α has to avoid profiles $\{P, Q\}$ and $\{R, S\}$. In the case of PAR_of_SEQ, α has to take precisely $\{P, Q\}$ or $\{R, S\}$. As expected, the daemon \mathfrak{d} behaves in the opposite way.

STRATEGY PROFILES	$\{P,Q\}$	$\{P,R\}$	$\{P,S\}$	$\{Q,R\}$	$\{Q,S\}$	$\{R,S\}$
\mathcal{A}						
SEQ_of_PAR	0	1	1	1	1	0
PAR_of_SEQ	1	0	0	0	0	1
\mathcal{D}						
SEQ_of_PAR	0	-1	-1	-1	-1	0
PAR_of_SEQ	-1	0	0	0	0	-1

4.5 Properties of Uncertainty Profiles and Assessments

Uncertainty profiles can be identified with properties of games. For some restricted cases, game assessments can be made directly from uncertainty profiles. More generally, it is shown on this section that uncertainty profiles are monotonic: decreasing the number of failures in a profile improves its assessment.

Assessment and Orc Expressions. In certain situations profile uncertainty assessments can be established directly from the profile itself or from its associated orchestration. When $f_{\mathcal{A}}$ is $\#\mathcal{A}$ or 0, the angel α has no real choice. The same happens with \mathfrak{d} , when $f_{\mathcal{D}}$ is $\#\mathcal{D}$ or 0. Just as it will shown in this case, there is no conflict.

Lemma 4.3 *Given a partition of $\alpha_+(E)$ into sets \mathcal{A} and \mathcal{D} , the following holds:*

- When $f_{\mathcal{A}} = \#\mathcal{A}$, the only strategy for α is $a = \mathcal{A}$,
- When $f_{\mathcal{D}} = \#\mathcal{D}$, the only strategy for \mathfrak{d} is $d = \mathcal{D}$,
- When $f_{\mathcal{A}} = 0$, the only strategy for α is $a = 0$,
- When $f_{\mathcal{D}} = 0$, the only strategy for \mathfrak{d} is $d = 0$

Therefore, it holds that, $v(\langle E, \mathcal{A}, \mathcal{D}, 0, 0 \rangle) = out(E)$ $v(\langle E, \mathcal{A}, \mathcal{D}, \#\mathcal{A}, \#\mathcal{D} \rangle) = 0$
 $v(\langle E, \mathcal{A}, \mathcal{D}, \#\mathcal{A}, 0 \rangle) = out(fail_{\mathcal{A}}(E))$ $v(\langle E, \mathcal{A}, \mathcal{D}, 0, \#\mathcal{D} \rangle) = out(fail_{\mathcal{D}}(E))$

Proof. The first uncertainty profile corresponds to “nothing fails“. In this case $\mathcal{A}_{\alpha} = \mathcal{A}_{\mathfrak{d}} = \emptyset$ and the only possible strategy profile is $s = (\emptyset, \emptyset)$ and then, $u_{\alpha}(s) = out(fail_{\emptyset}(E)) = out(fail(E))$.

The second uncertainty profile, correspond to one of the sites fails. In this case, the only choice is $a = \mathcal{A}$ and $d = \mathcal{D}$. Then, as $\mathcal{A} \cup \mathcal{D} = \alpha_+(E)$, we get $u_{\alpha}(s) = out(fail_{\alpha_+(E)}(E)) = 0$.

In the third case the only possible strategy is $a = \mathcal{A}$ and $d = \emptyset$, and we get the same result. Case four is similar. \square

The first profile, produces a basic monotonicity property. Under restricted circumstances, the value of the assessment is easy to compute:

Lemma 4.4 *Let E and F be two expressions, then we have:*

$$\begin{aligned} v(\langle E \mid F, \alpha_+(E), \alpha_+(F), f_{\mathcal{A}}, f_{\mathcal{D}} \rangle) &= \max_{a \in A_{\alpha}} \text{out}(\text{fail}_a(E)) + \min_{d \in A_{\delta}} \text{out}(\text{fail}_d(F)) \\ v(\langle E \gg F, \alpha_+(E), \alpha_+(F), f_{\mathcal{A}}, f_{\mathcal{D}} \rangle) &= \max_{a \in A_{\alpha}} \text{out}(\text{fail}_a(E)) * \min_{d \in A_{\delta}} \text{out}(\text{fail}_d(F)) \\ v(\langle E, \mathcal{A}, \mathcal{D}, f_{\mathcal{A}}, 0 \rangle) &= \max_{a \in A_{\alpha}} \text{out}(\text{fail}_a(E)) \\ v(\langle E, \mathcal{A}, \mathcal{D}, 0, f_{\mathcal{D}} \rangle) &= \min_{d \in A_{\delta}} \text{out}(\text{fail}_d(E)) \end{aligned}$$

Therefore, in all four cases the associated game has a pure saddle point.

Proof. Given the orchestration $E \mid F$, where the angel controls E and the daemon controls F , it is easy to see that, $u_{\alpha}(a, d) = \text{out}(\text{fail}_a(E)) + \text{out}(\text{fail}_d(F))$. Similarly, given the same player domains, it follows that $E \gg F$ is assessed as $u_{\alpha}(a, d) = \text{out}(\text{fail}_a(E)) * \text{out}(\text{fail}_d(F))$. In the case that the daemon has no moves $f_{\mathcal{D}} = 0$, we have that $u_{\alpha}(a, \emptyset) = \text{out}(\text{fail}_a(E))$ and similarly, when $f_{\mathcal{A}} = 0$, we have that $u_{\mathcal{D}}(\emptyset, d) = \text{out}(\text{fail}_d(E))$. Therefore, in all the four cases, the extreme values are achievable by pure strategies and the corresponding games have a pure saddle point. \square

Monotonicity. First of all, it is assumed that \mathcal{A} and \mathcal{D} remain unchanged. The following two lemmas allow, under some restricted hypothesis, to build new strategies from the existing ones, which have adequate monotonicity properties.

Consider the behaviour of the angelic utility: when the number of failures decreases from $f_{\mathcal{A}}$ (uncertainty profile \mathcal{U}) to $f'_{\mathcal{A}}$ (uncertainty profile \mathcal{U}'), where $0 \leq f'_{\mathcal{A}} \leq f_{\mathcal{A}}$. The strategies associated with games $\Gamma(\mathcal{U})$ and $\Gamma(\mathcal{U}')$ are $A_{\alpha} = \{a \subseteq \mathcal{A} \mid \#a = f_{\mathcal{A}}\}$ and $A'_{\alpha} = \{a' \subseteq \mathcal{A} \mid \#a' = f'_{\mathcal{A}}\}$, respectively. A mapping $\text{split}(\alpha)$ is used to transform mixed strategies $\alpha \in \Delta_{\alpha}$ into mixed strategies in Δ'_{α} . $\text{split}(\alpha)$, and is constructed as follows:

- Initially, the probability of any set a is spread in an equiprobable partition over the smaller set a' , $a' \subseteq a$. As $\#a = f_{\mathcal{A}}$ and $\#a' = f'_{\mathcal{A}}$ this gives rise to a division factor of $\binom{f_{\mathcal{A}}}{f'_{\mathcal{A}}}$.
- The probability of any a' is obtained by adding the probabilities of all a contributing to a' .

Formally:

Definition 4.5 [Split Strategy]. *Suppose that $f_{\mathcal{A}} \geq f'_{\mathcal{A}} \geq 0$ and $\alpha \in \Delta_{\alpha}$. The mapping*

$\text{split}(\alpha) : A'_a \rightarrow [0, 1]$ is:

$$\text{split}(\alpha)(a') = \frac{1}{\binom{f_{\mathcal{A}}}{f'_{\mathcal{A}}}} \sum_{\{a \mid a' \subseteq a\}} \alpha(a)$$

Example 4.13 Consider $\text{split}(\alpha)$, for a given α , where $\mathcal{A} = \{1, 2, 3, 4\}$, $f_{\mathcal{A}} = 3$ and $f'_{\mathcal{A}} = 2$.

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

$$A'_a = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$$

Suppose that α is given by $\alpha(\{1, 2, 3\}) = 1/2$ and $\alpha(\{1, 2, 4\}) = \alpha(\{1, 3, 4\}) = \alpha(\{2, 3, 4\}) = 1/6$. Here $\binom{f_{\mathcal{A}}}{f'_{\mathcal{A}}} = 3$, $\{a \mid \{1, 2\} \subseteq a\} = \{\{1, 2, 3\}, \{1, 2, 4\}\}$ and so $\text{split}(\alpha)(\{1, 2\}) = 2/9$. Other cases can be calculated in a similar fashion.

Lemma 4.5 The $\text{split}(\alpha)$ mapping is a mixed strategy for \mathbf{a} . That is, $\text{split}(\alpha) \in \Delta'_a$. Moreover, $u_a(\alpha, \beta) \leq u'_a(\text{split}(\alpha), \beta)$.

Proof. Given $a \in A_a$, there are $\#\{a' \mid a' \subseteq a\} = \binom{p}{p'}$ ways to choose a' such that, $a' \subseteq a$. For any $a' \in A'_a$, $a' \subseteq a$, and $d \in A_d$, it follows that:

$$u_a(a, d) \leq u'_a(a', d) \tag{4.1}$$

(since $\text{out}(\text{fail}_{a \cup d}(E)) \leq \text{out}(\text{fail}_{a' \cup d}(E))$).

Applying (1) to each a' , where $a' \subseteq a$, we get:

$$u_a(a, d) \leq \frac{1}{\binom{p}{p'}} \sum_{\{a' \mid a' \subseteq a\}} u'_a(a', d).$$

To prove that $\text{split}(\alpha)$ is a mixed strategy, consider the construction of $\text{split}(\alpha)(a') \geq 0$. The summation $\sum_{a' \in A'_a} \sum_{\{a \in A_a \mid a' \subseteq a\}} \alpha(a)$ can be rewritten as $\sum_{a \in A_a} \sum_{\{a' \in A'_a \mid a' \subseteq a\}} \alpha(a)$ because in both cases there is a sum over the set of pairs $\{(a, a') \in A_a \times A'_a \mid a' \subseteq a\}$.

Thus,

$$\sum_{a' \in A'_a} \text{split}(\alpha)(a') = \frac{1}{\binom{p}{p'}} \sum_{a \in A_a} \left(\sum_{\{a' \in A'_a \mid a' \subseteq a\}} \alpha(a) \right) = \sum_{a \in A_a} \alpha(a) \frac{1}{\binom{p}{p'}} \left(\sum_{\{a' \in A'_a \mid a' \subseteq a\}} 1 \right) = 1$$

Finally, the inequality between u_α and u'_α is derived as follows:

$$\begin{aligned} u_\alpha(\alpha, \beta) &= \sum_{a \in A_\alpha} \sum_{d \in A_\mathfrak{D}} \alpha(a) u_a(a, d) \beta(d) \leq \sum_{a \in A_\alpha} \sum_{d \in A_\mathfrak{D}} \frac{1}{\binom{p}{p'}} \sum_{\{a' \in A'_\alpha \mid a' \subseteq a\}} \alpha(a) u'_\alpha(a', d) \beta(d) = \\ &= \sum_{a' \in A'_\alpha} \sum_{d \in A_\mathfrak{D}} \left(\frac{1}{\binom{p}{p'}} \sum_{\{a \in A_\alpha \mid a' \subseteq a\}} \alpha(a) \right) u'_\alpha(a', d) \beta(d) = \sum_{a' \in A'_\alpha} \sum_{d \in A_\mathfrak{D}} \text{split}(\alpha)(a') u'_\alpha(a', d) \beta(d) \\ &= u'_\alpha(\text{split}(\alpha), \beta) \end{aligned}$$

□

Now consider the behaviour of the utility, when the number of failures increases from $f_\mathfrak{D}$ to $f'_\mathfrak{D}$, $f_\mathfrak{D} \leq f'_\mathfrak{D}$ and $f'_\mathfrak{D} \leq \#\mathfrak{D}$. The strategies associated with the two corresponding games are, $A_\mathfrak{D} = \{d \subseteq \mathfrak{D} \mid \#d = f_\mathfrak{D}\}$ and $A'_\mathfrak{D} = \{d' \subseteq \mathfrak{D} \mid \#d' = f'_\mathfrak{D}\}$. Given a mixed strategy $\beta \in \Delta_\mathfrak{D}$, a mixed strategy $\text{joint}(\beta) \in \Delta'_\mathfrak{D}$ can be constructed as follows:

- Set d is extended to set d' by the addition of $f'_\mathfrak{D} - f_\mathfrak{D}$ services from $\mathfrak{D} \setminus d$. There are $\binom{\#\mathfrak{D} - f_\mathfrak{D}}{f'_\mathfrak{D} - f_\mathfrak{D}}$ ways to carry out the extension.
- The probability of any d' is obtained by adding the probabilities of all d contributing to d' .

Definition 4.6 [Joint Strategy]. Given $f_\mathfrak{D} \leq f'_\mathfrak{D} \leq \#\mathfrak{D}$ and $\beta \in \Delta_\mathfrak{D}$, the mapping $\text{joint}(\beta) : A'_\mathfrak{D} \rightarrow [0, 1]$ is:

$$\text{joint}(\beta)(d') = \frac{1}{\binom{\#\mathfrak{D} - f_\mathfrak{D}}{f'_\mathfrak{D} - f_\mathfrak{D}}} \sum_{\{d \mid d \subseteq d'\}} \beta(d)$$

Example 4.14 Consider the situation where, $\mathfrak{D} = \{1, 2, 3, 4\}$ with $f_\mathfrak{D} = 2$ and $f'_\mathfrak{D} = 3$.

$$A_\mathfrak{D} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$$

$$A'_\mathfrak{D} = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}\}$$

In this case $\binom{\#\mathfrak{D} - f_\mathfrak{D}}{f'_\mathfrak{D} - f_\mathfrak{D}} = 2$.

Suppose that, β is given by,

$$\beta(\{1, 2\}) = \beta(\{1, 4\}) = \beta(\{2, 4\}) = 1/9$$

$$\beta(\{1, 3\}) = \beta(\{2, 3\}) = \beta(\{3, 4\}) = 2/9$$

For $\{1, 2, 3\} \in A'_\mathfrak{D}$, we have $\{d \in A_\mathfrak{D} \mid d \subseteq \{1, 2, 3\}\} = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$ and $\text{joint}(\beta)(\{1, 2, 3\}) = 5/9$.

Lemma 4.6 The mapping $\text{joint}(\beta)$ is a mixed strategy for \mathfrak{D} , that is, $\text{joint}(\beta) \in \Delta'_\mathfrak{D}$. Moreover, $u_\alpha(\alpha, \beta) \geq u'_\alpha(\alpha, \text{joint}(\beta))$.

Proof. Any $d \in A_\delta$ contains q services. Consider a $d' \in A'_\delta$ where, $d \subseteq d'$ and d' contains $q' \geq q$ sites. A strategy d' can be developed from d by adding $q' - q$ services from the set $\mathcal{D} \setminus d$. As this set contains $n - q$ elements there are $\binom{n-q}{q'-q}$ ways to choose d' . Since $\text{out}(\text{fail}_{a \cup d}(E)) \geq \text{out}(\text{fail}_{a \cup d'}(E))$, it follows that $u_a(a, d) \geq u'_a(a, d')$. By averaging over all such d' , $d \subseteq d'$ we get:

$$u_a(a, d) \geq \frac{1}{\binom{n-q}{q'-q}} \sum_{\{d' \in A'_\delta \mid d \subseteq d'\}} u'_a(a, d')$$

To prove that $\text{joint}(\beta)$ is a mixed strategy, consider the construction of $\text{joint}(\beta)$. In a way similar to that used in Lemma 4.5 it follows that,

$$\sum_{d' \in A'_\delta} \text{joint}(\beta)(d') = \sum_{d \in A_\delta} \beta(d) \left(\frac{1}{\binom{n-q}{q'-q}} \sum_{\{d' \in A'_\delta \mid d \subseteq d'\}} \right) = 1$$

The inequality between u_a and u'_a is derived as follows,

$$\begin{aligned} u_a(\alpha, \beta) &= \sum_{a \in A_\alpha} \sum_{d \in A_\delta} \alpha(a) u_a(a, d) \beta(d) \geq \sum_{a, d} \alpha(a) \left(\frac{1}{\binom{n-q}{q'-q}} \sum_{\{d' \in A'_\delta \mid d \subseteq d'\}} u'_a(a, d') \right) \beta(d) \\ &= \sum_{a, d'} \alpha(a) \left(\frac{1}{\binom{n-q}{q'-q}} \sum_{\{d \in A_\delta \mid d \subseteq d'\}} \beta(d) \right) u'_a(a, d') = \sum_{a, d'} \alpha(a) u'_a(a, d') \text{joint}(\beta)(d') \\ &= u'_a(\alpha, \text{joint}(\beta)) \end{aligned}$$

□

Theorem 4.1 Let $\mathcal{U} = \langle E, \mathcal{A}, \mathcal{D}, p, q \rangle$ and $\mathcal{U}' = \langle E, \mathcal{A}, \mathcal{D}, p', q' \rangle$, $p' \leq p$ and $q' \leq q$, be uncertainty profiles. Then $v(\mathcal{U}) \leq v(\mathcal{U}')$.

The proof is based on the min-max and max-min characterisation of the value of a game and uses Lemmas 4.5 and 4.6.

Proof. The proof is in two parts: first consider $\mathcal{U} = \langle E, \mathcal{A}, \mathcal{D}, p, q \rangle$ and $\mathcal{U}' = \langle E, \mathcal{A}, \mathcal{D}, p', q' \rangle$, $p' \leq p$. It follows that $v(\mathcal{U}) \leq v(\mathcal{U}')$. The proof is based in the following zero-sum game characterizations:

$$v(\mathcal{U}') = \max_{\alpha' \in \Delta'_\alpha} \min_{\beta \in \Delta_\delta} u'_a(\alpha', \beta), \quad v(\mathcal{U}) = \min_{\beta \in \Delta_\delta} \max_{\alpha \in \Delta_\alpha} u_a(\alpha, \beta)$$

Now $u_a(\alpha, \beta) \leq u_{\alpha'}(\text{split}(\alpha), \beta)$, for any $\alpha \in \Delta_\alpha$ and $\beta \in \Delta_\delta$ (Lemma 4.5). Choose $\beta^*(\alpha)$ so that, $u'_a(\text{split}(\alpha), \beta^*(\alpha)) = \min_{\beta} u'_a(\text{split}(\alpha), \beta)$

As $\min_{\beta} u'_a(\text{split}(\alpha), \beta) \leq \max_{\alpha} \min_{\beta} u'_a(\alpha, \beta) = v(\mathcal{U}')$ we get $u_a(\alpha, \beta^*(\alpha)) \leq u'_a(\text{split}(\alpha), \beta^*(\alpha)) \leq v(\mathcal{U}')$, and therefore it follows that $u_a(\alpha, \beta^*(\alpha)) \leq v(\mathcal{U}')$.

Now, given the set of pairs $\{(\alpha, \beta^*(\alpha)) \mid \alpha \in \Delta_a\}$, choose α^* such that $u_a(\alpha^*, \beta^*(\alpha^*)) = \max_{\alpha} u_a(\alpha, \beta^*(\alpha))$. As $v(\mathcal{U}) = \min_{\beta} \max_{\alpha} u_a(\alpha, \beta) \leq \max_{\alpha} u_a(\alpha, \beta^*(\alpha))$ we obtain $v(\mathcal{U}) \leq u_a(\alpha^*, \beta^*(\alpha^*))$ and $v(\mathcal{U}) \leq v(\mathcal{U}')$.

Secondly, consider an increase in the number of daemon failures. Consider $\mathcal{U} = \langle E, \mathcal{A}, \mathcal{D}, p, q \rangle$, and $\mathcal{U}'' = \langle E, \mathcal{A}, \mathcal{D}, p, q' \rangle$, $q' \geq q$. It follows that, $v(\mathcal{U}) \geq v(\mathcal{U}'')$. The proof is based in the following game characterisations:

$$v(\mathcal{U}'') = \min_{\beta} \max_{\alpha} u'_a(\alpha, \beta), \quad v(\mathcal{U}) = \max_{\alpha} \min_{\beta} u_a(\alpha, \beta)$$

For any α and β , it follows that $u_a(\alpha, \beta) \geq u'_a(\alpha, \text{joint}(\beta))$ (see Lemma 4.6). First, choose $\alpha^*(\beta)$ such that, $u'_a(\alpha^*(\beta), \text{joint}(\beta)) = \max_{\alpha} u'_a(\alpha, \text{joint}(\beta))$.

As $\max_{\alpha} u'_a(\alpha, \text{joint}(\beta)) \geq \min_{\beta} \max_{\alpha} u'_a(\alpha, \beta) = v(\mathcal{U}'')$ we obtain, $u_a(\alpha^*(\beta), \beta) \geq u'_a(\alpha^*(\beta), \text{joint}(\beta)) \geq v(\mathcal{U}'')$.

Given the set $\{(\alpha^*(\beta), \beta) \mid \beta \in \Delta_{\mathcal{D}}\}$, choose β^* such that, $u_a(\alpha^*(\beta^*), \beta^*) = \min_{\beta} u_a(\alpha^*(\beta), \beta)$.

As $\max_{\alpha} \min_{\beta} u_a(\alpha, \beta) = v(\mathcal{U})$ we get, $v(\mathcal{U}) \geq u_a(\alpha^*(\beta^*), \beta^*)$ and therefore, $v(\mathcal{U}) \geq v(\mathcal{U}'')$.

Finally, consider the uncertainty profiles $\mathcal{U} = \langle E, \mathcal{A}, \mathcal{D}, p, q \rangle$, and $\mathcal{U}' = \langle E, \mathcal{A}, \mathcal{D}, p', q' \rangle$ such that, $p' \leq p$ and $q' \leq q$. Consider the uncertainty profile $\mathcal{U}'' = \langle E, \mathcal{A}, \mathcal{D}, p, q' \rangle$. By the first part of the proof it holds that, $v(\mathcal{U}'') \leq v(\mathcal{U}')$ and by the second part of the proof, $v(\mathcal{U}) \leq v(\mathcal{U}'')$. \square

Note that trading-off daemon failures for angel failures may actually increase uncertainty.

Example 4.15 Given $E = ((P|Q) \gg T)|R$, the profile $\mathcal{U} = \langle E, \{T, R\}, \{P, Q\}, 1, 1 \rangle$ has assessment $v(\mathcal{R}) = 1$. Trading a daemon failure for an angelic one makes the situation worse because the modified profile $\mathcal{U}' = \langle E, \{T, R\}, \{P, Q\}, 2, 0 \rangle$ has assessment $v(\mathcal{U}') = 0$.

Uncertainty profiles can be given best-case and worse-case bounds:

Theorem 4.2 *The best and worst case uncertainty profiles associated with an orchestration expression E in an environment in which k service failures occur are,*

$$\text{Angelic}(\mathcal{U}) = \langle E, \alpha_+(E), \emptyset, k, 0 \rangle, \quad \text{Daemonic}(\mathcal{U}) = \langle E, \emptyset, \alpha_+(E), 0, k \rangle$$

Let \mathcal{U} be an arbitrary uncertainty profile for E with k failures. Then, $v(\text{Daemonic}(\mathcal{U})) \leq v(\mathcal{U}) \leq v(\text{Angelic}(\mathcal{U}))$. Thus, the profiles $\text{Angelic}(\mathcal{U})$ and $\text{Daemonic}(\mathcal{U})$ act as bounds on \mathcal{U} .

Proof. The actions of $\Gamma(\mathcal{U})$ and $\Gamma(\text{Angelic}(\mathcal{U}))$ are, respectively,

$$\begin{aligned} A_a &= \{a \subseteq \mathcal{A} \mid \#a = p\}, \quad A_d = \{d \subseteq \mathcal{D} \mid \#d = q\} \\ A'_a &= \{a' \subseteq \alpha_+(E) \mid \#a' = p + q\}, \quad A'_d = \emptyset \end{aligned}$$

In $\Gamma(\text{Angelic}(\mathcal{U}))$, the only mixed strategy available to the daemon is $\text{one} : A'_d \rightarrow [0, 1]$, where $\text{one}(\emptyset) = 1$. Therefore, $v(\text{Angelic}(\mathcal{U})) = \max_{\alpha'} u'_a(\alpha', \text{one})$. Any profile $(a, d) \in A_a \times A_d$ can be mapped into a profile $(a \cup d, \emptyset) \in A'_a \times A'_d$ as follows, given a mixed strategy profile (α, β) in $\Gamma(\mathcal{U})$, define the mixed strategy product $(\alpha, \beta) : A'_a \rightarrow [0, 1]$ as,

$$\text{product}(\alpha, \beta)(a') = \begin{cases} \alpha(a)\beta(d) & \text{if } a' = a \cup d. \\ 0 & \text{otherwise} \end{cases}$$

Then $u_a(\alpha, \beta) = u'_a(\text{product}(\alpha, \beta), \text{one})$. Choose (α^*, β^*) such that $v(\mathcal{U}) = u_a(\alpha^*, \beta^*)$. Note that, $u_a(\alpha^*, \beta^*) = u'_a(\text{product}(\alpha^*, \beta^*), \text{one}) \leq \max_{\alpha'} u'_a(\alpha', \text{one}) = v(\text{Angelic}(\mathcal{U}))$. The daemon game $\Gamma(\text{Daemonic}(\mathcal{U}))$ has actions, $A'_a = \emptyset$, $A'_d = \{d' \subseteq \alpha_*(E) \mid \#d' = p + q\}$. Since $v(\text{Daemonic}(\mathcal{U})) = \min_{\beta'} u'_a(\text{one}, \beta')$, we have a minimization problem. Taking (α^*, β^*) as before, we have $u_a(\alpha^*, \beta^*) = u'_a(\text{one}, \text{product}(\alpha^*, \beta^*)) \geq \min_{\beta'} u'_a(\text{one}, \beta') = v(\text{Daemonic}(\mathcal{U}))$. \square

Finally, to comment that in appendix, there are two examples, which correspond with this section. The first one (see **A**), shows an example of Internet computing, where sites are interpreted as interfaces among the members of a community. This example is based on *MeetingMonitor* example given in Section 7.3 of [76]. The second one (see **B**) is focused on risk management, applied on a specific IT system and from the viewpoint of game theory.

4.6 Notes

Orchestration and Choreography. To learn about the orchestration language Orc by J. Misra and W. R. Cook, refer to [76, 58, 25]. To read about web-services orchestrations, note to Sidney Rosario et. al. [99]. References [24] by W. R. Cook and J. Bartfield and [56] by G. Kandaswamy et al., which deal about web-services, are also very instructive.

Game Theory and Fault on Systems. For a good understanding of game theory, refer to essential books *Theory of Games and Economic Behaviour* [81] by J. von Neuman and O. Morgenstern; and *A Course in game theory* [84] by M. Osborne and A. Rubinstein. In addition, refer to an another interesting book [70], *Microeconomic theory*, published on 1995, by A. Mas-Colell.

T. Moscibroda and S. Schmid and R. Wattenhofer published a work [77] in 2006, where they considered the *Price of Malice* of a game, which models the containment of the spread of viruses, over a distributed system. This is an interesting consideration about a system of selfish individuals. Another consideration about this, is the [20] reference, where authors studying the impact of altruism on systems performance in atomic congestion games, noticed that altruism can be harmful in general.

The `muskel` is founded in M. Danelutto and M. Aldinucci [27], and “the reputation” of sites by G. C. Silaghi in [104]. Note to S. Marsh [69] and Bin Yu and M. P. Singh [114] for information about reputation and trustability. Refer to K. Eliaz’s work [32], where the author introduced a notion of fault tolerance implementation, and [31, 89, 37], which also deal with “faults” and “tolerance” on systems.

Chapter 5

On the Complexity of Equilibria Problems in Angel-Daemon Games

The complexity of equilibria problems for a class of strategic zero-sum games¹, called *angel-daemon* games, is analysed in this chapter. These games were introduced to assess the goodness of a Web or Grid orchestration on a faulty environment with bounded amount of failures. Furthermore, this chapter is focused on showing first that, deciding the existence of a pure Nash equilibrium or a dominant strategy for a given player is Σ_2^P -complete, and second, that computing the value of an *angel-daemon* game is EXP-complete. Therefore, both focuses match the already known complexity results of the corresponding problems for the generic families of succinctly represented games with exponential number of actions.

5.1 Angel-Daemon Games

Failures in Web systems are well accepted. It is practically impossible to have a constant and proper system behaviour. For this reason, it would be interesting to have available an estimate of the likelihood of success. Such an analysis has been performed using a subset of the Orc [76, 75] language to describe the orchestration of sites in the Web, in the context of game theory, when some additional assumptions on the behaviour of the failed sites and on the nature and amount of expected faults are met. [39].

In order to undertake such an analysis in an environment with a bounded amount of failures, the concept of *uncertainty profile* was introduced in [44]. An uncertainty profile specifies, for a set of sites participating in an orchestration, a partition into two sets \mathcal{A} and \mathcal{D} , and two values $f_{\mathcal{A}}$ and $f_{\mathcal{D}}$. The set \mathcal{A} represents these sites that fail due to misbehaviour or other reasons, but that are essentially non-malicious, and thus called

¹Work partially supported by FET pro-active Integrated Project 15964 (AEOLUS) and by Spanish projects TIN2005-09198-C02-02 (ASCE) and MEC-TIN2005-25859-E.

angelic. The set \mathcal{D} represents these sites with malicious behaviour, and are thus called *daemonic*. The numbers $f_{\mathcal{A}}$ and $f_{\mathcal{D}}$ constitute an estimation of the number of angelic and daemonic failures. The final component is the *angel-daemon game* associated to a uncertainty profile for a given Orc expression E . In an *angel-daemon* game there are two players, the angel and the daemon. The angel can select $f_{\mathcal{A}}$ sites in \mathcal{A} to fail and the daemon has to select $f_{\mathcal{D}}$ sites in \mathcal{D} to fail. Once the set of failures is fixed, the number of outputs of expression E when it is evaluated under such failures. The angel's objective is to maximise this value, while the daemon tries to minimise it. As such, an *angel-daemon* game is a zero-sum game and it follows from [113] that the entire Nash equilibrium of the game will have the same utility for the angel, which is known as the *game value*. The game value is used as the measure for assessing the expected behaviour of the E expression in the environment described by the uncertainty profile [39, 44].

The main components of strategic games are players, their actions and the payoffs or utility functions. Each component of a game, that is, a set or a function, can be explicitly described by means of a list or a table or implicitly via computation models, for example a Turing machine. Depending on how each component is described, a different degree of succinctness [9, 47, 102] can be achieved. This chapter is therefore focused on the analysis of the computational complexity of the following problems in *angel-daemon* games and in games with succinct representations.

Exists pure Nash equilibrium? (EPN). Given a game Γ , decide whether Γ has a pure Nash equilibrium.

Exists dominant strategy? (EDS). Given a game Γ and a player i , decide whether there is a dominant strategy for player i in Γ .

Game Value (GV). Given an zero-sum game Γ , compute its value.

The standard terminology for classical complexity classes like LOGSPACE, P, NP, coNP, Σ_2^P and EXP [12, 86] has been used. Results include a characterisation of the complexity of all problems addressed introduced above when the input is restricted to be an *angel-daemon* game. The decision on whether the EPN and the EDS problems are Σ_2^P -complete, and if GV is EXP-complete are shown. It will be discussed that similar results for general families were already known for strategic games in implicit form [8] and succinct zero-sum games [38].

5.2 Strategic Games and Succinct Representations

In essence, a *strategic game* is a tuple $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$, where $N = \{1, \dots, n\}$ is the set of players. For each player $i \in N$, A_i is a finite set of actions. For each player $i \in N$,

u_i is an *utility* (or *payoff*) function, mapping $A_1 \times \dots \times A_n$ to the rational. In chapter 2 the full strategic games definition 2.1 is given, borrowed from [84].

Then, the following question may be asked: How the players play given a strategic game Γ ? Let us take a strategic game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ and the strategy s_i for player i , as a selection of an action $s_i \in A_i$, where a *strategy profile* is a tuple $s = (s_1, \dots, s_n)$. After independently selecting the joint strategy profile s , player i gets utility $u_i(s)$. Given $s = (s_1, \dots, s_n)$ and a player i , s can be factorised as (s_{-i}, s_i) , where $s_{-i} = (s_j)_{j \neq i}$. A zero-sum game is a strategic game in which, for any strategy profile s holds $u_1(s) + u_2(s) = 0$.

At this point, is defined a strategic game, and an explanation on the way players play is given. In order to study the computational complexity of problems on games, the way an input game Γ is represented has to be defined.

The complexity of a problem is always analysed with respect to the size of the input. It is clear that the size of a game representation depends mainly on the number of players and on the size of the action set. Similarly, the utility function of each player, which is also part of the game description depends on the number of strategy profiles. Because of these, it is necessary to make clear the way the set of players is described: for each player, their set of actions and utility functions, depending on the succinctness of the description of the action sets and depending on whether the pay-off functions are described implicitly by Turing machines (TM).

A strategic game in *implicit form* [9] is represented by $\langle 1^n, 1^m, M, 1^t \rangle$. This game has n players. For each player i , their set of actions is $A_i = \Sigma^m$ and $\langle M, 1^t \rangle$ is the description of the utility functions. The utility of player i on the strategy profile s is given by the output of the Turing machine M on input (s, i) after t steps.

Notice that in this case, the length of the representation is proportional to the number of players n , the length of the actions m (and then logarithmic to the number of different actions), and the computation time t of the pay-off functions. Observe that this type of representation includes the *circuit games* considered in [47, 102]. The complexity of several problems on games given in implicit form was analysed in [8, 7]. In particular, they showed that the EPN problem is Σ_2^P -complete for games given in implicit form with four players.

A *succinct*, two-person, zero-sum game [38] representation is a boolean circuit C , such that the utilities are defined by $u_1(i, j) = C(i, j)$ and $u_2(i, j) = -C(i, j)$. It is well known that computing the value of a succinct zero-sum game is EXP-complete [35, 38]. To the best of this thesis candidate's knowledge, there are no results on the complexity of the EDS problem.

5.3 Orc and Angel-Daemon Games

An orchestration is a user-defined program that uses services on a Web or Grid. As we have seen in Chapters 3 and 4, Orc services are modelled by sites which have some pre-defined semantics. Typical examples of services are: an eigensolver, a search engine or a database [107]. A *site* accepts an argument and *publishes* a result value². For example, a call to a search engine, $find(x)$, may publish the set of sites which currently offer service x . A site call is silent if it does not publish a result. Site calls may induce side effects. A site call can publish at most one response. An orchestration, which composes a number of service calls into a complex computation can be represented by an Orc expression. An orchestrator may use any service that is available on the Web. The simplest kind of Orc expression is a site (service) call. Orc has two special sites, site 1 and site 0. A call to 1 always returns exactly one signal. A call to 0 never returns and thus remains silent. The subset of Orc operations, that has been used in the Orc, will be defined next. A site call $s(v_1, \dots, v_n)$ is called *non-blocking* if it must publish a result, when v_1, \dots, v_n are well defined; otherwise s is *potentially blocking*. In Orc, the predefined site $!$ is non-blocking while $if(b)$ is potentially blocking. In this thesis only some non-blocking expressions have been considered, to ensure the use of a subset of non-blocking operators.

Taking Lemma 4.1 in Chapter 4, and given an uncertainty profile \mathcal{U} , it is straightforward to obtain a description of the game $\Gamma(\mathcal{U})$ in implicit form or in zero-sum succinct form in polynomial time. Therefore, the following result was obtained.

Lemma 5.1 *Given an uncertainty profile \mathcal{U} for a non-blocking Orc expression E , a description of the angel-daemon game $\Gamma(U)$ in implicit form or zero-sum succinct form can be obtained in polynomial time.*

5.4 The Complexity of the EPN Problem

In this section it is proved that, deciding the existence of a pure Nash equilibrium for *angel-daemon* games is Σ_2^P -complete. In this way, this problem is as hard as for the general case of strategic games given in implicit form [9].

In order to prove the hardness for Σ_2^P a restricted version of the Quantified Boolean Formula is considered i.e., the Q2SAT problem, which is Σ_2^P -complete [86]. It is also established that Φ is a Q2BF formula, when Φ has the form $\Phi = \exists \alpha_1, \dots, \alpha_n \forall \beta_1, \dots, \beta_m F(\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_m)$, where F is a Boolean formula given in conjunctive normal form (CNF). Recall that Q2SAT is defined as follows:

²The words publish, return and output are used interchangeably. The terms site and service are also used interchangeably.

Given a Q2BF formula Φ over the boolean variables $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_m$, decide whether Φ is true.

An Orc expression on $2(n + m) + 5$ sites, from a given Q2BF formula $\Phi = \exists x_1, \dots, x_n \forall y_1, \dots, y_m F(x_1, \dots, x_n, y_1, \dots, y_m)$, is constructed. For any i , $1 \leq i \leq n + 1$, there are two sites X_i and \bar{X}_i . For any j , $1 \leq j \leq m + 1$, two sites Y_j and \bar{Y}_j , are considered. Let X be the Orc expression $(X_1 | \bar{X}_1) \gg \dots \gg (X_{n+1} | \bar{X}_{n+1})$. Let Y be the expression $(Y_1 \gg \bar{Y}_1) | \dots | (Y_{m+1} \gg \bar{Y}_{m+1})$.

It is assumed that $F = C_1 \wedge \dots \wedge C_k$, where each C_λ , $1 \leq \lambda \leq k$ is a clause. An expression O_λ is constructed for each clause ($1 \leq \lambda \leq k$). This expression is formed by the parallel composition of the sites corresponding to the literals appearing in C_λ . Then, the expression O associated to formula F is defined as $O_1 \gg \dots \gg O_k$. The expression associated to the boolean formula is the following:

$$E_\Phi = Z(x) \text{ where } x : \in [(Y_{m+1} \gg X_{n+1}) | (\bar{Y}_{m+1} \gg \bar{X}_{n+1}) | (X \gg (O|Y))].$$

The associated uncertainty profile is $\mathcal{U}(\Phi) = \langle E_\Phi, \mathcal{A}, \mathcal{B}, n + 1, m + 1 \rangle$, with $\mathcal{A} = \{Z, X_1, \dots, X_{n+1}, \bar{X}_1, \dots, \bar{X}_{n+1}\}$ and $\mathcal{B} = \{Y_1, \dots, Y_{m+1}, \bar{Y}_1, \dots, \bar{Y}_{m+1}\}$.

Lemma 5.2 *The Q2BF formula Φ is true iff the angel-daemon game $\Gamma(\mathcal{U}(\Phi))$ has a pure Nash equilibrium.*

Proof. A given formula $\Phi = \exists \alpha_1, \dots, \alpha_n \forall \beta_1, \dots, \beta_m F(\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_m)$ is assumed. Before providing the proof of the lemma some notation is introduced. Given $\alpha \in \{0, 1\}^n$ denoted by A_α , the Angel strategy is defined as:

$$A_\alpha = \{X_{n+1}\} \cup \{X_i | \alpha_i = 0\} \cup \{\bar{X}_i | \alpha_i = 1\}.$$

A angelic strategy A is *consistent* when, for any $1 \leq i \leq n$, exactly one of the sites X_i or \bar{X}_i belongs to A , and the same for the Daemon, that is, given $\beta \in \{0, 1\}^m$, two Daemon strategies are defined:

$$\begin{aligned} D_\beta^+ &= \{Y_{m+1}\} \cup \{Y_j | \beta_j = 1\} \cup \{\bar{Y}_j | \beta_j = 0\} \\ D_\beta^- &= \{\bar{Y}_{m+1}\} \cup \{Y_j | \beta_j = 1\} \cup \{\bar{Y}_j | \beta_j = 0\}. \end{aligned}$$

A daemonic strategy D is *consistent* if, for any $1 \leq j \leq m + 1$, exactly one of the sites Y_j or \bar{Y}_j belongs to D . Observe that for any consistent strategy for the angel, in which Z does not fail, X produces some kind of output, and that for any non consistent strategy for the daemon, Y produces some kind of output.

In the case that Φ is true, an assignment $\alpha \in \{0, 1\}^n$ for which $F(\alpha, \beta)$ is true, for any $\beta \in \{0, 1\}^m$ is considered. The Angel strategy A_α is also considered. It is possible to

observe that, first X produces some kind of output and second, for any consistent strategy for the daemon, O produces some kind of output. In addition, for any non consistent strategy for the daemon, Y produces some kind of output. Therefore, for any strategy of the daemon, the final number of outputs is 1. Thus, any combined strategy profile (A_α, D) is a Nash equilibrium.

In the case that \mathcal{F} is not true, any strategy profile (A, D) is considered. It is also shown that (A, D) is not a Nash equilibrium. Proof is given by cases. All possibilities are divided into 4 cases and it is proven that in any of them, the strategy profile is not a Nash equilibrium.

- **Case 1:** $Z \in A$. In this case the utility obtained by the angel, is zero.

If D is not consistent, the angel can move to a consistent strategy replacing Z by X_{n+1} , and as Y produces at least one output, the angel improves its utility.

In the case that D is a consistent strategy, then either $Y_{m+1} \in D$ or $\bar{Y}_{m+1} \in D$. Then, it is assumed that $Y_{m+1} \in D$. Substituting Z by X_{n+1} or by any other not failed site, if $X_{n+1} \in A$ the angel obtains 1 instead of zero. In the case that the failed site is \bar{Y}_{m+1} , a symmetric argument shows that the strategy profile is not a Nash equilibrium.

- **Case 2:** $Z \notin A$ but both $X_{n+1}, \bar{X}_{n+1} \in A$.

If D is not consistent, the angel can move into a consistent strategy making only X_{n+1} fail. Taking into account that Y produces at least one output, the angel improves its utility.

In the case that D is a consistent strategy, either $Y_{m+1} \in D$ or $\bar{Y}_{m+1} \in D$. Then, it is assumed that $Y_{m+1} \in D$ thus, substituting X_{n+1} by another not failed site in A , the angel gets 1 instead of zero. In the case that the failed site is \bar{Y}_{m+1} , a symmetric argument shows that the strategy profile is not a Nash equilibrium.

- **Case 3:** $Z, X_{n+1}, \bar{X}_{n+1} \notin A$.

In this case, A is not consistent and therefore, X does not produce any output.

If $Y_{m+1}, \bar{Y}_{m+1} \in D$, the number of outputs is zero. Then, the angel can move to a consistent strategy and, as Y produces at least one output, improve its own utility.

If $Y_{m+1}, \bar{Y}_{m+1} \notin D$, the number of outputs is one. In this case the daemon can move to a non consistent strategy in which both Y_{m+1}, \bar{Y}_{m+1} , and as X does not produce any output, it improves its own utility.

It is assumed that knowing that either $Y_{m+1} \in D$ or $\bar{Y}_{m+1} \in D$, in this sub-case the expression produces some kind of output. Then, the angel can move to a strategy in which both Y_{m+1} and \bar{Y}_{m+1} fail and improve their own utilities.

- **Case 4:** $Z \notin A$, either X_{n+1} or \bar{X}_{n+1} belong to A . Without lack of generality, it is assumed that $X_{n+1} \in A$.

- **Case 4.1:** A is not consistent.

In this case, X does not produce any output.

If $Y_{m+1}, \bar{Y}_{m+1} \in D$, the number of outputs is zero. Then, the angel can move to a consistent strategy and, as Y produces at least one output, improve its own utility.

If $Y_{m+1}, \bar{Y}_{m+1} \notin D$, the number of outputs is one. In this case, the daemon can move to a non-consistent strategy in which both Y_{m+1}, \bar{Y}_{m+1} fail and, as X does not produce any output, improve its utility.

It is also assumed that knowing that $Y_{m+1} \in D$ and $\bar{Y}_{m+1} \notin D$, in this sub-case the expression produces some kind of output. Then, the daemon can move to a strategy in which both Y_{m+1} and \bar{Y}_{m+1} fail and improve its utility. Then, it is assumed that knowing that $Y_{m+1} \notin D$ and $\bar{Y}_{m+1} \in D$, in this sub-case the expression does not produce any output. Then, the angel can move to a consistent strategy and improve its own utility.

- **Case 4.2:** A is consistent and D is not consistent.

Let $\alpha \in \{0, 1\}^n$ be the assignment, so that $A_\alpha \subseteq A$. Since Φ is not true, there should be at least one $\beta \in \{0, 1\}^m$ such that $F(\alpha, \beta)$ is false. Let β_0 be one of such assignments.

- **Case 4.2.1:** D is not consistent.

In this case the number of outputs is one. Then the daemon can move to $D_{\beta_0}^-$ reducing the outputs to zero.

- **Case 4.2.2:** D is consistent.

Let $\beta \in \{0, 1\}^m$ be the assignment so that $D_\beta \subseteq D$. If $F(\alpha, \beta)$ is true, the number of outputs is one, but in which the daemon can change its strategy towards $D_{\beta_0}^-$, reducing the outputs to zero. If $F(\alpha, \beta)$ is false we have two cases: $Y_{m+1} \in D$, which means that the number of outputs is one, and where the daemon can replace Y_{m+1} by \bar{Y}_{m+1} , improving its payoff. However, when $\bar{Y}_{m+1} \in D$, the number of outputs is one and the angel can change strategy and get one output.

□

Lemma 5.1 and Lemma 5.2, are put together, and taking in mind that PNE problems for games given in implicit form belong to Σ_2^P [9], the following theorem is obtained.

Theorem 5.1 *Deciding whether an angel-daemon game has a pure Nash equilibrium is Σ_2^P complete.*

As a consequence of the above result, some results on the complexity of the PNE problem on succinct described games, were obtained.

Corollary 5.1 *Deciding whether a zero-sum game given in implicit form has a pure Nash equilibrium is Σ_2^P complete. Deciding whether a two-player strategic game given in implicit form has a pure Nash equilibrium is Σ_2^P complete.*

The above results settles the number of players to a minimum. It were improved the already known results as in the reduction provided in [9], where the number of players was four.

5.5 Computing the Value of Angel-Daemon Game

In this section it is proved that computing the number of outputs for a non-blocking Orc expression is P-complete, and that computing the value of an *angel-daemon* game is EXP-complete. Thus, again the VG problem is as hard as for the general case of succinct zero-sum games [35, 38].

A construction of an *angel-daemon* game associated to a succinct description of a zero-sum game, is provided. A boolean circuit C with $n + m$ inputs and $k + 1$ outputs describing a succinct zero-sum game Γ is assumed. Without lack of generality, it is assumed that the circuit has only and and or gates and that all the negations are on the inputs.

First, a non-blocking Orc expression S_C corresponding to a one output circuit C , is defined. For any i , $1 \leq i \leq n$, there are two sites X_i and \bar{X}_i . For any j , $1 \leq j \leq m$, two sites Y_j and \bar{Y}_j , are considered. These sites have been referred as the *input sites*. The Orc expression uses a construction provided in [75] to express the orchestration of a workflow on a dag (see proof of Lemma 5.3 for more details). The idea is to associate a site with each gate and a variable to each wire in the circuit. The expression makes use of the where constructor to guarantee the appropriate workflow. The input to the circuit is bound through variables to the input sites, so that when sites fail according to a consistent assignment α , $\text{out}(\text{fail}_{\mathcal{F}_\alpha}(S_C)) = C(\alpha)$, where $\mathcal{F}_\alpha = \{X_i \mid x_i = 0\} \cup \{\bar{X}_i \mid x_i = 1\} \cup \{Y_j \mid y_j = 0\} \cup \{\bar{Y}_j \mid y_j = 1\}$. Thus, the expression produces zero outputs if and only if the circuit evaluates to zero. This construction is used to devise a reduction from the *Circuit value problem*, which is P-complete [66, 48].

Theorem 5.2 *Given a non-blocking, well formed, Orc expression E , determine whether $\text{out}(E) \neq 0$ is P-complete. Given a set of failures $\mathcal{F} \subseteq \alpha_+(E)$, determine whether $\text{out}(\text{fail}_{\mathcal{F}}(E)) \neq 0$ is also P-complete.*

We go back to circuit C , which gives a succinct representation of a game. For each output gate, a different Orc expression is generated, thus there are expressions S_C^0, \dots, S_C^k .

These expressions are organised so that the final result corresponds to the value represented by the circuit. To do so, for any $0 \leq j$ the Orc expression P_j that produces 2^j outputs is considered. This is obtained by repeatedly doubling the number of outputs, $P_0 = 1$ and $P_l = P_{l-1} | P_{l-1}$ for any $0 < l \leq j$. The combination $V_C = (S_C^0 \gg P_0) | \dots | (S_C^k \gg P_k)$ produces $C(\alpha)$ outputs provided that the input variables fail according to α . The last step takes care of inconsistent sets of failures. Let X be the Orc expression $(X_1 | \bar{X}_1) \gg \dots \gg (X_n | \bar{X}_n)$ and let Y be the expression $(Y_1 \gg \bar{Y}_1) | \dots | (Y_{m+1} \gg \bar{Y}_{m+1})$. The expression associated to the circuit $E_C = X \gg (V_C | (Y \gg P_{k+1}))$, and the uncertainty profile $\mathcal{U}_C = \langle E_C, \mathcal{A}, \mathcal{D}, n, m \rangle$ are defined, where $\mathcal{D} = \{Y_1, \dots, Y_m, \bar{Y}_1, \dots, \bar{Y}_m\}$ and $\mathcal{A} = \alpha^+(E_C) \setminus \mathcal{D}$.

Lemma 5.3 *Given a succinct description of a zero-sum game Γ by means of a circuit C , the angel-daemon game $\Gamma(\mathcal{U}_C)$ has the same value as Γ .*

Proof. First, the Orc expression S_C corresponding to a boolean circuit C is provided, with $n + m$ inputs and one output gate. It is assumed that all the internal gates are two inputs and or or gates. Furthermore, the l gates are enumerated in topological order, from 1 to l , so that gate g_i has as inputs the output of lower numbered gates. The output gate g_l is assumed.

The expression Orc will have a site G_i for each gate and a variable z_{ij} whenever the output of gate i is the input of two gate j . For any $1 \leq i \leq n$, there are two variables x_i and \bar{x}_i , and for any $1 \leq j \leq m$ two more variables y_j and \bar{y}_j represent the circuit's input. An additional variable is used for the circuit's output. Gate g_i is associated to the following expression:

$$E_i(z_{i_0i}, z_{i_1i}) = \begin{cases} G_i(z_{i_0i}, z_{i_1i}) & \text{if } g_i = g_{i_0} \text{ and } g_{i_1} \\ G_i(z_{i_0i}) | G_i(z_{i_1i}) & \text{if } g_i = g_{i_0} \text{ or } g_{i_1} \end{cases}$$

Moreover, for each wire that goes from gate i to gate j , there is a variable assignment $z_{ij} : \in G_i(z_{i_0i}, z_{i_1i})$.

An order on the wire variables that corresponds to the reversed topological order of gates is defined. The output wires of a gate in the gate position, is inserted in any order. It is assumed that this gives order z^1, \dots, z^w , where $z^1 = z_l$, is reordered according to the expressions of the corresponding gates. The expression S_C is defined according to the following schema:

$$\begin{aligned} Z(z^1) \text{ where } z^1 : \in E^1 \text{ where } z^2 : \in E^2 \dots \text{ where } z^w : \in E^w \\ \text{where } x_1 : \in X_1, \dots, x_n : \in X_n, \bar{x}_1 : \in \bar{X}_1, \dots, \bar{x}_n : \in \bar{X}_n \\ \text{where } y_1 : \in Y_1, \dots, y_m : \in Y_m, \bar{y}_1 : \in \bar{Y}_1, \dots, \bar{y}_m : \in \bar{Y}_m. \end{aligned}$$

The following result follows from this definition.

Lemma 5.4 For any $\alpha \in \{0, 1\}^n$, $\beta \in \{0, 1\}^m$ we have $\text{out}(\text{fail}_{\mathcal{F}_{\alpha, \beta}}(S_C)) = C(\alpha, \beta)$, where $\mathcal{F}_{\alpha, \beta} = \{X_i \mid \alpha_i = 0\} \cup \{\bar{X}_i \mid \alpha_i = 1\} \cup \{Y_j \mid \beta_j = 0\} \cup \{\bar{Y}_j \mid \beta_j = 1\}$.

Recall that the *circuit value* problem is defined as follows: given a boolean circuit C and an assignment of truth values to its inputs α , determine whether $C(\alpha) = 1$. Given (C, α) , the expression S_C^α which is obtained by first computing S_C and then replacing by O in any site in \mathcal{F}_α , is constructed. This expression is equivalent to $\phi_{\mathcal{F}_\alpha}(S_C)$. Observe that both \mathcal{F}_α and S_C^α can be computed using only logarithmic space. This, together with the previous, show the following hardness result.

Lemma 5.5 Given a non-blocking well formed Orc expression E , determine whether $\text{out}(E) \neq 0$ is P -hard. Given a set of failures $\mathcal{F} \subseteq \alpha_+(E)$, determine whether $\text{out}(\phi_{\mathcal{F}}(E)) \neq 0$ is also P -hard.

This concludes the proof of Theorem 5.2, taking into account the result in Lemma 4.1. \square

In the general case, in which the circuit C has $k + 1$ outputs that are interpreted as a $k + 1$ -bit number (for the sake of clarity), the construction for each output is repeated, although the connection with the input variables could may be written only once,

$$V_C = (S_C^0 \gg P_0) \mid \cdots \mid (S_C^k \gg P_k).$$

From the definition of P_j , we have that for any $j \geq 0$, $\text{out}(P_j) = 2^j$, therefore taking into account Lemma 5.4.

Lemma 5.6 For any $\alpha \in \{0, 1\}^n$, $\beta \in \{0, 1\}^m$, we have that $\text{out}(\text{fail}_{\mathcal{F}_{\alpha, \beta}}(V_C)) = C(\alpha, \beta)$.

Recall that the final expression defining an *angel-daemon* game in the construction is the following: Let $X = (X_1 \mid \bar{X}_1) \gg \cdots \gg (X_n \mid \bar{X}_n)$ and let $Y = (Y_1 \gg \bar{Y}_1) \mid \cdots \mid (Y_{m+1} \gg \bar{Y}_{m+1})$. The expression associated to the circuit is $E_C = X \gg (V_C \mid (Y \gg P_{k+1}))$, and the uncertainty profile is $\mathcal{U}_C = \langle E_C, \mathcal{A}, \mathcal{D}, n, m \rangle$, where the set of daemonic sites is $\mathcal{D} = \{Y_1, \dots, Y_m, \bar{Y}_1, \dots, \bar{Y}_m\}$ and $\mathcal{A} = \alpha^+(E_C) \setminus \mathcal{D}$.

Let $\Gamma = \Gamma(\mathcal{U})$ be the associated *angel-daemon* game. For a consistent strategy a of the angel, let α_a be the truth assignment $\alpha_i = 1$ iff $X_i \notin a$, $1 \leq i \leq n$. In a similar way, for a consistent strategy d of the daemon, let β_d be the truth assignment $\beta_j = 1$ iff $Y_j \notin d$, $1 \leq j \leq m$. Observe that $\mathcal{F}_{\alpha_a, \beta_d} = a \cup d$.

From the definition of X and Y and taking into account that the circuit outputs a number with $k + 1$ bits, the following result is obtained.

Lemma 5.7

- For any non-consistent strategy a of the angel, and for any strategy d of the daemon, we have $u_a(a, d) = 0$.

- For any consistent strategy a of the angel, and any non-consistent strategy d of the daemon, we have $u_a(a, d) = 2^{k+1}$.
- For any consistent strategy a of the angel, and any consistent strategy d of the daemon, we have $u_a(a, d) = C(\alpha_a, \beta_d)$ and therefore $0 \leq u_a(a, d) < 2^{k+1}$ holds.

Finally, it is shown that a Nash equilibrium of the succinct game described by the circuit C is a Nash equilibrium in the *angel-daemon* game $\Gamma(\mathcal{U}_C)$.

Lemma 5.8 *Let (σ_a, σ_d) be a mixed Nash equilibrium of the game defined by C , then the mixed profile (σ'_a, σ'_d) defined by,*

$$\sigma'_a(a) = \begin{cases} \sigma_a(\alpha_a) & \text{if } a \text{ is consistent} \\ 0 & \text{otherwise} \end{cases} \quad \sigma'_d(d) = \begin{cases} \sigma_d(\beta_d) & \text{if } d \text{ is consistent} \\ 0 & \text{otherwise} \end{cases}$$

is a Nash equilibrium in $\Gamma(\mathcal{U}_C)$.

Proof. Note that, in the support of the strategy profile (σ'_a, σ'_d) , only consistent strategies are possible.

For any possible consistent strategy for the angel that is not in the support, using Lemma 5.7 and the fact that (σ_a, σ_d) is a Nash equilibrium, the expected benefit is equal or smaller than the one that for a strategy in the support. For any non-consistent strategy for the angel, the expected value is zero. Therefore, it is equal or smaller than the one for a strategy in the support. Thus σ'_a is a best response to σ'_d .

For any possible consistent strategy for the daemon that is not in the support, using Lemma 5.7, the expected benefit again, is equal or smaller than the one that for a strategy in the support. For any non-consistent strategy for the daemon, since all the strategies in the support of the angel are consistent, the expected value is 2^{k+1} . Therefore, it is equal or bigger than the one for a strategy in the support. Thus, σ'_d is a best response to σ'_a . □

The previous lemma shows the existence of a Nash equilibrium in game $\Gamma(\mathcal{F}_C)$ that, according to Lemma 5.7, has the value of the game associated to the original circuit. Therefore, both zero-sum games have the same value, as it was proven by Theorem 5.3.

According to Lemma 5.1, a succinct description of an *angel-daemon* game can be obtained in polynomial time. Furthermore, computing the value of a succinct zero-sum game is EXP-complete [38]. This, together with Lemma 5.3 gives the following result.

Theorem 5.3 *Computing the value of an angel-daemon game is EXP-complete.*

5.6 Deciding the Existence of Dominant Strategies

This section is focused on some considerations on the last problem considered here. The complexity of the EDS, has not been addressed before on succinctly represented games.

Note that, in the Orc expression constructed in the proof of Theorem 5.1, when the Q2BF formula Φ is true, the associated *angel-daemon* game has a dominant strategy, but when formula Φ is not true the *angel-daemon* game has no pure Nash equilibrium and therefore has no dominant strategy.

Theorem 5.4 *Deciding whether a dominant strategy exists for:*

- angel-daemon games,
- strategic games in implicit form, and
- zero-sum games in implicit form

is Σ_2^P -complete.

Proof. A player i has a dominant strategy if s_i^* exists, such that for any other s_i and s_{-i} it holds that, $u_i(s_{-i}, s_i) \leq u_i(s_{-i}, s_i^*)$. From the definition it is clear that, given a game in implicit form, the EDS problem is in Σ_2^P . In order to prove completeness, a reduction from Q2SAT to EDS, is given. A formula $\Phi = \exists \alpha_1, \dots, \alpha_n \forall \beta_1, \dots, \beta_m F(\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_m)$, is mapped into a pair with two components. The first component is the *angel-daemon* game corresponding to the uncertainty profile $\mathcal{U}(\Phi) = \langle E_\Phi, \mathcal{A}, \mathcal{B}, n+1, m+1 \rangle$ given in Section 5.4. The second component is the angel α . When Φ is true, there is $\alpha \in \{0, 1\}^n$, such that $F(\alpha, \beta)$ is true for any $\beta \in \{0, 1\}^m$. The Angel's strategy A_α is dominant, because $u_\alpha(A_\alpha, A_\beta) = 1$ and this is the maximum possible utility for the angel. When Φ is not true, according to Lemma 5.2, there is no Nash equilibrium, and therefore there is no dominant strategy for α . \square

5.7 Notes

Game Theory and Computational Complexity. For a good understanding of this chapter, refer to the books *Theory of Games and Economic Behavior* [81] by J. von Neuman and O. Morgenstern, and the *Computers and Intractability: A Guide to the Theory of NP-completeness* [45], of M. Garey and D. Johnson. Refer also to [80] about Nash equilibria, and [11] by J. L. Balcazar, A. Lozano and J. Torán about the complexity of algorithmic problems on succinct instances.

The complexity of several problems on games given in implicit form was analysed in [9, 7]. In particular, they show that the EPN problem is Σ_2^P -complete for

games given in implicit form with four players. It is well known that computing the value of a succinct zero-sum game is EXP-complete [35, 38].

Orchestration and Web-Services. Refer to Sidney Rosario et. al. [99], to learn about web-services orchestrations. Very instructive references focused on web-services are [24] by W. R. Cook and J. Bartfield and [56] by G. Kandaswamy et al.. In addition, essential references to understanding the orchestration language Orc by J. Misra and W. R. Cook are [76, 58, 25, 75].

Part II

Computational Issues of Game Isomorphism

Chapter 6

Preliminaries on Game Isomorphisms

The introduction of three notions of isomorphisms, strong, weak and local is proposed in this chapter. Each one of these isomorphisms preserves a different structure of the game. Strong isomorphisms have been defined as to preserve the utility functions and Nash equilibria. Weak isomorphisms preserve only the player preferences relations and thus pure Nash equilibria. Local isomorphisms preserve the structure of Nash dynamics. The computational complexity of the game isomorphism problem depends on the level of succinctness of the description of the input games but it is independent on which type of isomorphism is considered. Some classic computational complexity problems have been, specifically considered which are to the equivalence of two given problems, with respect to its computational model.

Game Mappings

Game mappings that provide a way to associate players and their actions in one game to players and actions in the other have been considered. Inasmuch as, usually these mappings are independent of the utilities. The notations and definitions given were adapted from [90, 108].

Definition 6.1 [Game mappings]. Given $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ and $\Gamma' = (N, (A'_i)_{i \in N}, (u'_i)_{i \in N})$, a game mapping ψ from Γ to Γ' is a tuple $\Psi = (\pi, (\varphi_i)_{i \in N})$ where π is a bijection from N to N , the player's bijection, and, for any $i \in N$, φ_i is a bijection from A_i to $A'_{\pi(i)}$, the i -th player actions bijection.

Example 6.1 The mappings between games Γ and Γ' with two players and two actions each one, are considered. As before, $A_1 = \{t, b\}$ and $A_2 = \{l, r\}$. In the second game, $A'_1 = \{t', b'\}$ and $A'_2 = \{l', r'\}$. The profiles are located as follows:

	l	r
t	(t,l)	(t,r)
b	(b,l)	(b,r)

strategy profiles

	l'	r'
t'	(t',l')	(t',r')
b'	(b',l')	(b',r')

strategy profiles

Consider a mapping $\Psi = (\pi, \varphi_1, \varphi_2)$ such that,

$$\pi = (1 \rightarrow 2, 2 \rightarrow 1), \varphi_1 = (t \rightarrow r', b \rightarrow l'), \varphi_2 = (l \rightarrow t', r \rightarrow b')$$

As players are interchanged, $\Psi(a_1, a_2) = (\varphi_2(a_2), \varphi_1(a_1))$ and the profiles are mapped as follows $(t, l) \rightarrow (t', r')$ (boldfaced in preceding games), $(t, r) \rightarrow (b', r')$, $(b, l) \rightarrow (t', l')$, and $(b, r) \rightarrow (b', l')$.

In this case $N = \{0, 1\}$ and $A_1 = A_2 = A'_1 = A'_2 = \{0, 1\}$. The mapping Ψ is,

$$\pi = (1 \rightarrow 2, 2 \rightarrow 1), \varphi_1 = (0 \rightarrow 1, 1 \rightarrow 0), \varphi_2 = (0 \rightarrow 0, 1 \rightarrow 1)$$

where $\varphi_1 = (0 \rightarrow 1, 1 \rightarrow 0)$ is equivalent to $\varphi_1 = (t \rightarrow r', l \rightarrow t')$, and $\varphi_2 = (0 \rightarrow 0, 1 \rightarrow 1)$ is equivalent to $\varphi_2 = (l \rightarrow t', r \rightarrow b')$, and profiles are mapped $(0, 0) \rightarrow (0, 1)$, $(0, 1) \rightarrow (1, 1)$, $(1, 0) \rightarrow (0, 0)$, $(1, 1) \rightarrow (1, 0)$.

Then, Ψ is encoded as,

$$\langle 11, \langle 0, 1 \rangle, \langle 0, 1 \rangle, \langle 0, 1 \rangle \langle 0, 1 \rangle, \langle \pi \rangle, \langle \varphi_1 \rangle, \langle \varphi_2 \rangle \rangle$$

with $\langle \pi \rangle = \langle 2, 1 \rangle$, $\langle \varphi_1 \rangle = \langle 1, 0 \rangle$, and $\langle \varphi_2 \rangle = \langle 0, 1 \rangle$.

Note that, a player bijection identifies player $i \in N$ with player $\pi(i)$, and the corresponding actions' bijection φ_i maps the set of actions of player i on to the set of actions of player $\pi(i)$. A game mapping Ψ from Γ to Γ' induces in a natural way, a bijection from $A_1 \times \dots \times A_n$ to $A'_1 \times \dots \times A'_n$, where the strategy profile (a_1, \dots, a_n) is mapped into the strategy profile (a'_1, \dots, a'_n) , defined as $a'_{\pi(i)} = \varphi_i(a_i)$, for all $1 \leq i \leq n$. Consider this mapping as $\Psi(a_1, \dots, a_n) = (a'_1, \dots, a'_n)$, overloading the use of Ψ . A mixed strategy profile $p = (p_1, \dots, p_i, \dots, p_n)$ is given by probabilities p_i on A_i (such that $\sum_{a_i \in A_i} p_i(a_i) = 1$) for $1 \leq i \leq n$. A game mapping Ψ also induces a mapping $\Psi(p_1, \dots, p_n) = (p'_1, \dots, p'_n)$, such that $p'_{\pi(i)}$ is a probability on $A'_{\pi(i)}$ defined by $p'_{\pi(i)}(\varphi_i(a_i)) = p_i(a_i)$. Thus, isomorphisms are game mappings, which fulfil some additional restrictions on utilities or preferences as will be shown, next.

In order to describe a game mapping, the less succinct approach has been considered. Note that for information on each game, only the set of actions for each player, have to be kept.

Game Mapping in Explicit Form. All components of the mapping are given explicitly. Action sets are given by listing all their elements and permutations are given by tables, that is $\psi = \langle 1^n, A_1, \dots, A_n, A'_1, \dots, A'_n, T_\pi, T_{\varphi_1}, \dots, T_{\varphi_n} \rangle$, where $T_\pi, T_{\varphi_1}, \dots, T_{\varphi_n}$ are tables such that $T_{\varphi_i}[a_i] = a'_{T_\pi[i]}$.

The description of a mapping by Turing machines,

$$\psi = \langle 1^n, A_1, \dots, A_n, A'_1, \dots, A'_n, M_\pi, M_{\varphi_1}, \dots, M_{\varphi_n}, 1^t \rangle$$

has not been considered, because in such a case, it is possible to construct an explicit coding of ψ with size bounded by $2|\psi|$ in time $|\psi|^2$.

Note that it is not trivial to consider an adequate description of mapping associated to a set with exponentially many actions. In view of that, the implicit form representation for mappings and games, has not been considered.

6.1 Strong, Weak and Local Game Isomorphism

Strong Isomorphism. Traditionally the notion of equivalence is studied at different levels using different types of isomorphism, depending on the family of games and the structural properties to be preserved. Therefore, a stronger version of isomorphism introduced by J. Nash [80] (see also [90, 108]), is defined.

Definition 6.2 [Strong Isomorphism]. *Given two strategic games $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ and $\Gamma' = (N, (A'_i)_{i \in N}, (u'_i)_{i \in N})$, a game mapping $\psi = (\pi, (\varphi_i)_{i \in N})$ is called a strong isomorphism $\psi : \Gamma \rightarrow \Gamma'$ when, for any player $1 \leq i \leq n$ and any strategy profile a , it holds that, $u_i(a) = u'_{\pi(i)}(\psi(a))$. In the particular case that Γ' is Γ , a strong isomorphism is called a strong automorphism.*

An example of strong isomorphism, is provided in Example 6.2.

Example 6.2 *Given the following games, Γ and Γ' :*

$$\begin{array}{c}
 \begin{array}{cc}
 & \text{Player 2} \\
 & l \quad r \\
 \text{Player 1 } t & \begin{array}{|c|c|} \hline 0,0 & 0,1 \\ \hline 1,1 & 1,0 \\ \hline \end{array} \\
 b & \\
 \Gamma &
 \end{array}
 \xrightarrow{\psi}
 \begin{array}{c}
 \begin{array}{cc}
 & \text{Player 2} \\
 & l' \quad r' \\
 \text{Player 1 } t' & \begin{array}{|c|c|} \hline 1,0 & 0,1 \\ \hline 0,0 & 1,1 \\ \hline \end{array} \\
 b' & \\
 \Gamma' &
 \end{array}
 \end{array}
 \end{array}$$

Consider the morphism $\psi : \Gamma \rightarrow \Gamma'$ defined as $\psi = (\pi, \varphi_1, \varphi_2)$, where $\pi = (1 \rightarrow 2, 2 \rightarrow 1)$, $\varphi_1 = (t \rightarrow l', b \rightarrow r')$ and $\varphi_2 = (l \rightarrow b', r \rightarrow t')$. This morphism maps strategy profiles as: $\psi(t, l) = (b', l')$, $\psi(t, r) = (t', l')$, $\psi(b, l) = (b', r')$ and $\psi(b, r) = (t', r')$. Therefore it is a strong isomorphism.

Given a strong isomorphism ψ between Γ and Γ' , a mixed strategy profile p is a Nash equilibrium in Γ iff $\psi(p)$ is a Nash equilibrium in Γ' . The same holds for pure Nash equilibria. Thus, the bijection induced by strong isomorphisms on to the set of mixed strategy profiles, preserves the structure of Nash equilibria. Furthermore, a strong isomorphism induces an isomorphism among the Nash dynamics graphs of both games.

Weak Isomorphism. There are several ways to relax the notion of strong isomorphism while maintaining the structure of Nash equilibria. For instance, Harsanyi and Selten [49] substitute $u_{\pi(i)}(\psi(a)) = u_i(a)$ by $u_{\pi(i)}(\psi(a)) = \alpha_i u_i(a) + \beta_i$. In order to generalize this approach the following game isomorphism [84] is considered, in which the preference relations $(\preceq_i)_{i \in N}$ induced by the utility functions are preserved. The preference relations induced by the utility functions u_i are defined as:

$$\begin{aligned} a \preceq_i a' & \text{ when } u_i(a) \leq u_i(a') \\ a \prec_i a' & \text{ when } u_i(a) < u_i(a') \\ a \sim_i a' & \text{ when } u_i(a) = u_i(a') \end{aligned}$$

Note that *strict preference* $a \prec_i a'$ iff $a \preceq_i a'$ but not $a' \preceq_i a$ as usual. Note also *indifference* by $a \sim_i a'$, as usual indifference occurs when $a \preceq_i a'$ and $a' \preceq_i a$ holds. The definition of isomorphism can be adapted to respect only preference relations instead of utility functions.

Definition 6.3 [Weak Isomorphism]. A weak isomorphism $\psi : \Gamma \rightarrow \Gamma'$ is a mapping $\psi = (\pi, (\varphi_i)_{i \in N})$, such that any triple a, a' and i verifies: $a \preceq_i a'$ iff $\psi(a) \preceq_{\pi(i)} \psi(a')$.

Example 6.3 Consider $a \preceq_i a'$ iff $u_i(a) \leq u_i(a')$. Below follows an example of weak isomorphism $\psi = (\pi, \varphi_1, \varphi_2)$:

$$\begin{array}{ccc} & \begin{array}{cc} \text{Player 2} \\ l & r \end{array} & \\ \begin{array}{cc} \text{Player 1} & t \\ & b \end{array} & \begin{array}{|cc|} \hline 0,0 & 0,1 \\ \hline 1,1 & 1,0 \\ \hline \end{array} & \xrightarrow{\psi} & \begin{array}{cc} \text{Player 2} \\ l' & r' \end{array} \\ & \Gamma & & \begin{array}{|cc|} \hline 3,3 & 2,2 \\ \hline 2,3 & 3,2 \\ \hline \end{array} \\ & & & \Gamma' \end{array}$$

where $\pi = (1 \rightarrow 2, 2 \rightarrow 1)$, and $\varphi_1 = (t \rightarrow r', b \rightarrow l')$, $\varphi_2 = (l \rightarrow t', r \rightarrow b')$. Observe that $u_i(a)$ and $u_i(\psi(a))$ are not even related by a linear function.

When games Γ and Γ' are weakly isomorphic, note that $\Gamma \sim_w \Gamma'$. In the particular case that Γ' is Γ , the weak isomorphism is called weak automorphism.

Weak isomorphisms preserve preferences for any pair of strategy profiles and any player, therefore they maintain the structure of pure Nash equilibria.

Local Isomorphism. This type of isomorphism is a more relaxed notion than the strong and weak isomorphisms. This isomorphism preserves the preferences of the player on the “close” neighborhood of the strategy profile, for each player and each strategy profile. This is an even weaker requirement, which still guarantees that the structure of Pure Nash equilibria is preserved, however the structure of mixed Nash equilibria is not.

Definition 6.4 [Local Isomorphism]. A local isomorphism $\psi : \Gamma \rightarrow \Gamma'$ is a mapping ψ , such that for any triple a, a' and i such that $a_{-i} = a'_{-i}$, $a \prec_i a'$ iff $\psi(a) \prec_{\pi(i)} \psi(a')$ and $a \sim_i a'$ iff $\psi(a) \sim_{\pi(i)} \psi(a')$ are verified. When Γ and Γ' are locally isomorphic it is noted $\Gamma \sim_\ell \Gamma'$. In the particular case that Γ' is Γ , a weak isomorphism is named a local automorphism.

Here, only preferences $a \preceq_i a'$, have been considered, such that $a_{-i} = a'_{-i}$ are preserved for any player. It is easy to see that local isomorphisms preserve pure Nash equilibria.

Example 6.4 Consider the following games, which are locally isomorphic but not strongly isomorphic.

		<i>Player 2</i>				<i>Player 2</i>	
		<i>l</i>	<i>r</i>			<i>l'</i>	<i>r'</i>
<i>Player 1</i>	<i>t</i>	0,0	0,1	$\xrightarrow{\psi}$	<i>Player 1</i>	3,3	1,0
	<i>b</i>	1,1	1,0			2,3	2,1
Γ_1				Γ_2			

Let a mapping $\psi = (\pi, \varphi_1, \varphi_2)$ with $\pi = (1 \rightarrow 2, 2 \rightarrow 1)$, $\varphi_1 = (t \rightarrow r', b \rightarrow l')$, $\varphi_2 = (l \rightarrow t', r \rightarrow b')$. We can easily check that ψ is a local morphism. For Γ_1 , $(t, l) \prec_1 (b, r)$ and $(t, l) \sim_2 (b, r)$, but for Γ_2 , $\psi(t, l) \prec_{\pi(1)} \psi(b, r)$ and $\psi(t, l) \prec_{\pi(2)} \psi(b, r)$, therefore the morphism is not weak.

It is easy to see that local isomorphisms preserve pure Nash equilibria.

6.2 Classical Complexity's Problems

In this section, the definitions of several computational problems are briefly explained. The coNP-hardness results follow from reductions of the following coNP-complete problem [45].

Validity problem (VALIDITY): Given a boolean formula F , decide whether F is satisfiable by all truth assignments.

The following problems on boolean circuits were also considered. Recall that two circuits $C_1(x_1, \dots, x_n)$ and $C_2(x_1, \dots, x_n)$ are *isomorphic*, if there is a permutation π of $\{1, \dots, n\}$ such that, for any truth assignment $x \in \{0, 1\}^n$, $C_1(x) = C_2(\pi(x))$.

Boolean circuit isomorphism problem (CIRCUITISO): Given two boolean circuits C_1 and C_2 , decide whether C_1 and C_2 are isomorphic.

A related problem is based on the notion of congruence. A *congruence* between two circuits on n variables, $C_1(x_1, \dots, x_n)$ and $C_2(x_1, \dots, x_n)$ is a mapping $\psi = (\pi, f_1, \dots, f_n)$, where π is a permutation of $\{1, \dots, n\}$ and, for any $1 \leq i \leq n$, f_i is a permutation on $\{0, 1\}$ (either the identity or the negation function). Similarly that in the case of game morphism, the image $\psi(x)$ is obtained by permuting the positions of the input bits, according to permutation π , and then applying to any bit i the permutation f_i .

Boolean circuit congruence problem (CIRCUITCONG): Given two circuits C_1 and C_2 , decide whether C_1 and C_2 are congruent.

The CIRCUITISO problem has been studied by B. Borchert, D. Ranjan and F. Stephan in [17]. Among many other results, they show that CIRCUITISO and CIRCUITCONG are equivalent. It is known that CIRCUITISO $\in \Sigma_2^P$, but M. Agrawal and T. Thierauf prove that it cannot be Σ_2^P -hard unless the polynomial hierarchy collapses (see Corollary 3.5 in [4]).

The isomorphism and congruence problems for boolean formulas have been considered. Recall that two formulas $\Phi_1(x_1, \dots, x_n)$ and $\Phi_2(x_1, \dots, x_n)$ are *isomorphic* if there is a permutation π of $\{1, \dots, n\}$ such that, for any truth assignment $x \in \{0, 1\}^n$, $C_1(x) = C_2(\pi(x))$. They are *congruent* if there is a mapping $\psi = (\pi, f_1, \dots, f_n)$, where π is a permutation of $\{1, \dots, n\}$ and, for any $1 \leq i \leq n$, f_i there is a permutation on $\{0, 1\}$ such that for any truth assignment $x \in \{0, 1\}^n$, $C_1(x) = C_2(\pi(x))$.

Boolean formula isomorphism problem (FORMULAISO): Given two boolean formulas Φ_1 and Φ_2 , decide whether Φ_1 and Φ_2 are isomorphic.

Boolean formula congruence problem (FORMULA CONG): Given two boolean formulas Φ_1 and Φ_2 , decide whether Φ_1 and Φ_2 are congruent.

B. Borchert, D. Ranjan and F. Stephan in [17] show that FORMULAISO and FORMULA CONG are equivalent. It is known that FORMULAISO $\in \Sigma_2^P$. but it cannot be Σ_2^P -hard unless the polynomial hierarchy collapses (see Corollary 3.4 in [4]).

Two graphs are *isomorphic* if there is a one-to-one correspondence between their vertices. There is an edge between two vertices of one graph, if and only if there is an edge between the two corresponding vertices in the other graph.

Graph isomorphism (GRAPHISO): Given two graphs, decide whether they are isomorphic.

It is well known that GRAPHISO is not expected to be NP-hard [61].

Notation

Some additional definitions and notation used in this thesis are given in this section. A *binary actions game* is a game in which the set of actions for each player is $\{0, 1\}$. A *binary game* is a binary actions' game in which the utility functions range is $\{0, 1\}$. One will need to construct binary actions' games associated to general games. To do this, a *binify* process on the strategies of the original game is used.

Given a strategic game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$, without loss of generality is assumed that $N = \{1, \dots, n\}$ and that, for any $i \in N$, $A_i = \{1, \dots, k_i\}$. An action $j \in A_i$ is “binify”, coding it with k_i bits, as $\text{binify}(j) = 0^{j-1}10^{k_i-j}$. In a strategy profile the “binify” process can be used such that, given $a = (a_1, \dots, a_n)$, we write $\text{binify}(a) = (\text{binify}(a_1), \dots, \text{binify}(a_n))$. Observe that by setting $k = \sum_{i \in N} k_i$, $\text{binify}(a) \in A' = \{0, 1\}^k$ is obtained. Then, $\text{good}(A') = \{\text{binify}(a) | a \in A\}$ and $\text{bad}(A') = A' \setminus \text{good}(A')$ are defined. Note that $\text{binify} : A \rightarrow \text{good}(A')$ is a bijection and therefore the inverse function is also a bijection.

Example 6.5 Given Γ with 3 players, $A_1 = A_3 = \{1, 2\}$ and $A_2 = \{1, 2, 3\}$, $\text{binify}(1, 2, 2) = (10, 010, 01) = (1, 0, 0, 1, 0, 0, 1)$ and $\text{binify}^{-1}(10, 010, 01) = (1, 2, 2)$ is obtained.

6.3 Notes

Game Theory. For mathematical definitions of strategic and extensive games, refer to *A Course in Game Theory* of M. Osborne and A. Rubinstein [84] published in 1994, and also the *Theory of Games and Economic Behavior* of John von Neuman and Oscar Morgenstern [81]. It is also very important to note the definition of *Nash equilibrium* for non-cooperative games in [80] by J. Nash.

Morphisms in Games. In the field of game isomorphisms refer to [80] by J. Nash, where a very *compact* definition of game morphisms was first proposed. Note the results of J. C. C. McKinsey [72] and J. Harsanyi and R. Selten's isomorphisms of strategic games [49], published in 1988.

Algorithmic Game Theory. Essential reference is *Computers and Intractability: A Guide to the Theory of NP-completeness* [45], by M. Garey and D. Johnson. This book features a compendium of NP-complete problems. In order to understand the complexity in game theory, it is crucial to refer to [36, 35] by J. Feigenbaum and J. Feigenbaum, D. Koller and P. Shor respectively. These researches were devoted to the study of connection between game theory and the traditional complexity class. In addition, note [64], which is devoted

to the analysis of the complexity involving in finding max-min strategies in zero-sum games represented in extensive form. Finally, very important references in algorithmic game theory are [[65](#), [87](#), [86](#)].

Chapter 7

The Complexity of Game Isomorphism

The question of whether two multi-player strategic games are equivalent and the computational complexity of deciding such a property are evaluated in this chapter. Furthermore, is focused in showing the computational complexity of the game isomorphism problem depends on the level of succinctness of the description of the input games, but it is independent on which of the two types of isomorphisms is considered. In cases where games are given in *general form*, an explicit description of actions and a succinct description of utilities is assumed.

It is as well shown that the game isomorphism problem for general form games is equivalent to circuit isomorphism when utilities are described by Turing Machines TMs, and to the boolean formula isomorphism problem when utilities are described by formulas. When a game is given in explicit form, the game isomorphism problem is shown to be equivalent to the graph isomorphism problem.

7.1 The ISISO and ISO Problems

In defining a concrete equivalence between games, it has to be kept in mind a structural properties preserved in equivalent games. A *strong* isomorphism, which preserves pure and mixed Nash equilibria is considered, and also a *weak* isomorphism that only preserves pure Nash equilibria. Each of them requires preserving less information about the relative structure of profiles while preserving still the structure of the Nash or pure Nash equilibria.

This chapter is focused in the computational complexity of deciding whether two games are equivalent. Two problems related to isomorphisms are considered.

Game Isomorphism (IsIso). Given two games Γ and Γ' and a game mapping $\psi : \Gamma \rightarrow \Gamma'$, decide whether ψ is a game isomorphism.

Game Isomorphism (Iso). Given two games Γ, Γ' , decide whether there exists a game isomorphism between Γ and Γ' .

In order to study the computational aspects of isomorphism problems on strategic games, the way in which games and morphisms are represented as inputs to a program, is the first needed to be determined. As it has been said in Chapter 1, in order to represent strategic games, the following two representations proposed in [7] are considered, each with a different level of succinctness. When a game is given in *general* form the actions are listed explicitly, but utilities and mappings are given by deterministic Turing machines. In the *explicit* case, utilities are stored in tables. In both cases morphisms are always represented by tables. This is not a restriction, since in polynomial time a morphism representation is transformed by Turing machines into a tabular representation by tables, because the actions are given explicitly.

The main contributions of this chapter are classified into the following ones:

- The ISISO problem is coNP-complete, for games given in general form, and belongs to NC when games are given in explicit form.
- The ISO problem belongs to Σ_2^P , for games given in general form, and to NP when games are given in explicit form.
- The ISO problem is equivalent to the boolean circuit isomorphism problem, for games in general form, and to the graph isomorphism problem, for games given in explicit form.

The above results hold independently of the type of isomorphism considered. Note that the boolean circuit isomorphism problem is believed not to be Σ_2^P -hard [4], and that the graph isomorphism problem is conjectured not to be NP-hard [61]. Therefore the same results are valid for the ISO problem.

Besides the above mentioned generic forms of representing games, will also be considered another particular class of strategic games, which is called *formula games*. As it will be proven, the formula games considered in this thesis are equivalent in power of representation to a subfamily of the *weighted boolean formula games* introduced in [71]. The complexity of the ISO problem when the games correspond to a general form was analysed, that is, the number of bits controlled by each player is a constant. Thus, for formula games in general form, it will be shown that the ISO problem is equivalent to the boolean formula isomorphism problem. Recall that the complexity of the boolean formula isomorphism problem is the same as that of circuit isomorphism, however it is conjectured that both problems are not equivalent.

The ISISO and ISO problems can be formulated for the strong and weak isomorphisms introduced above, and also for games in general form (strategic or boolean formula) or games in explicit form. The game isomorphism problem can also be considered for the

case in which $n = 1$. For this particular case, the isomorphism problem is computationally easy.

Theorem 7.1 *The ISO problem for games with one player is solvable in polynomial time, for strong and weak isomorphisms, for general form strategic and formula games and for explicit form games.*

Proof. A 1-player game $\Gamma(\{1\}, A_1, (u_1))$ and the vector $x = (x_1, \dots, x_m)$, where $x_i = u_1(i)$ and $m = |A_1|$, are considered. The game's characteristic vector $S(\Gamma)$ is defined as the vector obtained after sorting x in increasing order. Then, there are two 1-player games that are strongly isomorphic iff their characteristic vectors are identical.

For the case of weak isomorphism the condition is equivalent to the fact that the relative order of two consecutive elements is the same in both characteristic vectors.

The vector can be obtained in polynomial time for any of the considered game representation and thus, the problems can be solved in polynomial time.

□

Assumption. In view of the above result, it is assumed that all games have at least two players, for the rest of the work.

7.2 Complexity Results for Strong Isomorphisms

Let us start with the complexity for the ISISO problem in the case of strategic games.

Theorem 7.2 *The ISISO problem for strong isomorphisms is coNP-complete for strategic games in general form and for boolean formula games in general form. The problem belongs to NC whenever the games are given in explicit form. The strong isomorphism is given in both cases in explicit form.*

Proof. First, it is assumed that games are given in general form. In this case the input is composed of two games, $\Gamma = \langle 1^n, A_1, \dots, A_n, M_1, 1^{t_1} \rangle$ and $\Gamma' = \langle 1^n, A'_1, \dots, A'_n, M_2, 1^{t_2} \rangle$, and a game mapping between the two games $\Psi = \langle 1^n, A_1, \dots, A_n, A'_1, \dots, A'_n, T_\pi, T_{\phi_1}, \dots, T_{\phi_n} \rangle$. Then, we have $\langle \Gamma, \Gamma', \Psi \rangle \in \text{ISISO}$ iff

$$\forall (a_1, \dots, a_n) \in A_1 \times \dots \times A_n \forall i \in N \ u'_{\pi(i)}(\Psi(a_1, \dots, a_n)) = u_i(a_1, \dots, a_n).$$

Therefore ISISO belongs to coNP. Since this is enough to guess a strategy profile $a = (a_1, \dots, a_n)$ and a player i , using polynomial space, then, $u'_{\pi(i)}(\Psi(a)) \neq u_i(a)$ is checked in polynomial time.

To prove hardness two games are defined. The first one is associated to a boolean formula, and a mapping between them. Given a boolean formula F with n variables, the following game is considered:

WINWHENTRUE(F): This game has n players, $N = \{1, \dots, n\}$, and player i has $A_i = \{0, 1\}$. All the players $1 \leq i \leq n$ have the same utility $u_i(a_1, \dots, a_n) = F(a_1, \dots, a_n)$.

The game is coded in general form as $\langle 1^n, A_1, \dots, A_n, \text{Eval}, 1^{\log n(n+|F|)^2} \rangle$, where Eval is a TM that evaluates a formula in time $O((n+|F|)^2)$. Some additional time is provided, to get rid of the constant. Observe that this codification can be obtained in polynomial time given F .

ALWAYSWIN: This game has n players, $N = \{1, \dots, n\}$, and player i has $A_i = \{0, 1\}$. All players $1 \leq i \leq n$ have the same utility $u_i(a_1, \dots, a_n) = 1$.

This game can be represented in general form as $\langle 1^n, A_1, \dots, A_n, \text{One}, 1^{n+1} \rangle$, where One is a TM that, after reading the input, outputs 1 in time $n+1$. Furthermore, the representation can be computed in $O(n)$ time.

IDENTITY: This mapping combines the identity function on $N = \{1, \dots, n\}$ with the identity function on $\{0, 1\}$.

The mapping is represented by $\langle 1^n, A_1, \dots, A_n, A_1, \dots, A_n, id_\pi, id_1, \dots, id_n \rangle$, where $id_\pi(i) = i$ and $id_i(a_i) = a_i$ for $i \leq i \leq n$. Note that, such a representation can be obtained in time $O(n)$.

The mapping is claimed: $\langle \text{WINWHENTRUE}(F), \text{ALWAYSWIN}, \text{IDENTITY} \rangle \in \text{ISISO}$ iff F is valid. When F is a valid formula both games have the same utility functions. Then, the mapping IDENTITY is a strong isomorphism. When F is not valid, there exists x_1, \dots, x_n such that $F(x_1, \dots, x_n) = 0$. Therefore, the utility of this strategy profile, for player 1 in WINWHENTRUE is 0. But, the same player gets utility 1 in the ALWAYSWIN game. Then, IDENTITY is not a strong isomorphism.

When Γ_1, Γ_2 and ψ are formula games in general form, the same argumentations show that the ISISO problem is coNP-complete.

When Γ_1, Γ_2 and ψ are given in explicit form the strong isomorphism can be verified in poly-logarithmic parallel. The corresponding $a' = (a'_1, \dots, a'_n)$, have to be computed in parallel, such that $a'_{T_\pi[i]} = T_{\varphi_i}[a_i]$. Finally, it has to be tested if $T_1[a, i] = T_2[a', T_\pi[i]]$ for each player i .

□

The next theorem provides upper bounds for the complexity of the ISO problem.

Theorem 7.3 *The ISO problem for a strong morphism belongs to Σ_2^P for strategic and formula games in general form. The problem belongs to NP when the games are given in explicit form.*

Proof. First, the proof of membership is considered. A non-deterministic algorithm working in polynomial space/time is defined, depending on the representation of the input game. Given two strategic games $\Gamma_1 = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ and $\Gamma_2 = (N, (A'_i)_{i \in N}, (u'_i)_{i \in N})$ by definition, is assumed that there is a strong isomorphism between Γ_1 and Γ_2 iff,

$$\exists \Psi = (\pi, \varphi_1, \dots, \varphi_n) \forall a \in A_1 \times \dots \times A_n \forall i \in N u_i(a) = u'_{\pi(i)}(\Psi(a)),$$

where Ψ is a mapping of Γ_1 to Γ_2 . Note that it is possible to guess an isomorphism $\Psi = \langle A_1, \dots, A_n, A'_1, \dots, A'_n, T_\pi, T_{\varphi_1}, \dots, T_{\varphi_n} \rangle$ using polynomial space. Furthermore, given a strategy profile $a = (a_1, \dots, a_n)$ it is possible to compute $\Psi(a)$ in polynomial time, just doing $a'_{T_\pi[i]} = T_{\varphi_i}[a_i]$. To check the correctness of the guess, it is needed verify that, for every player i and strategy profile a , it holds that, $u_i(a) = u'_{\pi(i)}(\Psi(a))$.

When games are given in general form, the strategy profile can be represented in polynomial space and the test performed in polynomial time, both for utilities given by TM or formulas. Therefore, the ISO problem belongs to Σ_2^P . When games are given in explicit form, the number of strategy profiles is polynomial in the size of the input. Then, for all a , the condition $u_i(a) = u'_i(\Psi(a))$ can be checked in polynomial time, once the mapping has been guessed. Therefore, the ISO problem belongs to NP. □

In the following paragraph, the fact that ISO is equivalent to CIRCUITISO is proved for games in general form. This is done through a series of reductions transforming the game while preserving the existence of a strong isomorphism. Firstly, it is shown how to construct corresponding isomorphic binary actions games. Secondly, the construction from a binary action game of a binary game preserving isomorphism is illustrated. Finally, the equivalence with the Boolean circuit congruence is as well shown. All transformations presented here can be computed in polynomial time, and this will not be specified again along the rest of the chapter. Let us start with the first transformation:

A construction for the first reduction that makes use of the binify process, is defined. Let $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ be a strategic game. In this case, $k = \sum_{i \in N} k_i$ were $k_i = |A_i|$. The binify process can be used in a strategy profile, given $a = (a_1, \dots, a_n) \in A$, is written: $\text{binify}(a) = (\text{binify}(a_1), \dots, \text{binify}(a_n))$. Recall that $\text{good}(A') = \text{binify}(A)$.

$\text{BINARYACT}(\Gamma, \mu) = (N', (A'_i)_{i \in N'}, (u'_i)_{i \in N'})$, where $N' = \{1, \dots, k\}$ and, for any $i \in N'$, $A'_i = \{0, 1\}$ and thus the set of action profiles is $A' = \{0, 1\}^k$. The players are partitioned into B_1, \dots, B_n blocks. Block i is formed by k_i players.

Given $i \in B_j$, it is possible to say that i belongs to block j of players and the following expression can be written: $\text{block}(i) = j$.

The utilities are defined by,

$$u'_i(a') = \begin{cases} u_{\text{block}(i)}(\text{binify}^{-1}(a')) & \text{if } a' \in \text{good}(A'), \\ \mu & \text{if } a' \in \text{bad}(A'). \end{cases}$$

Notice that, for $a \in A$, $u'_i(\text{binify}(a)) = u_{\text{block}(i)}(a)$. Furthermore, all players in a given block have the same utility function. Each strategy profile a' in $\text{BINARYACT}(\Gamma, \mu)$ can be factorised considering the actions taken by the k players as $a' = (a'_1, \dots, a'_k)$ or by grouping the actions according to the blocks B_1, \dots, B_n as $a' = (b_1, \dots, b_n)$, where $b_i \in \{0, 1\}^{k_i}$. The value μ will be selected to create a gap on the utility that separates the profiles in $\text{BINARYACT}(\Gamma, \mu)$, which correctly separate a Γ profile, from those that do not.

Example 7.1 The transformation from Γ into $\text{BINARYACT}(\Gamma, \mu)$ is considered. The Γ game is taken as a version of BS game with non-zero utilities and setting $\mu = 0$, such that:

		Player 2	
		1	2
Player 1	1	3, 2	1, 1
	2	1, 1	2, 3

BS game

A'	u_1	u_2	u_3	u_4
1010	3	3	2	2
1001	1	1	1	1
0110	2	2	3	3
0110	1	1	1	1
$a' \in \text{bad}(A')$	0	0	0	0

In the BS game $A_1 = A_2 = \{1, 2\}$, and $\text{binify}(1) = 10$, $\text{binify}(2) = 01$. Therefore $\text{good}(A') = \{1010, 1001, 0110, 0101\}$ and $\text{bad}(A') = \{0, 1\}^4 \setminus \text{good}(A')$.

The game $\text{BINARYACT}(BS, 0)$ has $N' = \{1, 2, 3, 4\}$. The partition of players into blocks is given by $B_1 = \{1, 2\}$ and $B_2 = \{3, 4\}$.

Given a good strategy profile a and the player i , $u'_i(a)$ is computed as follows:

Suppose $a = 1010 = (\text{binify}(1), \text{binify}(1))$ and $i = 4$. As player 4 belongs to B_2 it holds that, $\text{block}(4) = 2$, $u'_4(1010) = u_4(\text{binify}(1))$, and $\text{textsf{binify}}(1) = u_{\text{block}(4)}(1, 1) = u_2(1, 1) = 2$.

Now, the reduction from the ISO problem for strong isomorphism is applied to the same problem for binary actions games.

Lemma 7.1 Let Γ_1, Γ_2 be two strategic games given in general form. Let t be $\max\{t_1, t_2\}$, where t_i , $1 \leq i \leq 2$, is the time allowed to the utility TM of the game Γ_i . There is a strong isomorphism between Γ_1 and Γ_2 iff there is a strong isomorphism between the games $\text{BINARYACT}(\Gamma_1, \mu)$ and $\text{BINARYACT}(\Gamma_2, \mu)$ where $\mu = -2^t$.

Proof. When M is a TM computing the utilities in time t the following inequalities are obtained $|u_i(a)| \leq t$ and $-2^t \leq u_i(a) \leq 2^t$. Given Γ and Γ' with utilities computed in times t and t' , and taking $t = \max\{t, t'\}$ and $\mu = -2^t$ a TM for $\text{BINARYACT}(\Gamma, \mu)$ and $\text{BINARYACT}(\Gamma', \mu)$ can be found, computing utilities in $O(t)$. Furthermore, a description of both machines can be obtained in polynomial time.

Given a strong isomorphism $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ of Γ into Γ' , let us define a mapping $\psi' = (p, f_1, \dots, f_k)$ of $\text{BINARYACT}(\Gamma, \mu)$ into $\text{BINARYACT}(\Gamma', \mu)$. Suppose that it holds that, in ψ , $\pi(i) = j$. Then, as $\varphi_i : A_i \rightarrow A'_j$ is a bijection, by construction blocks B_i and B'_j in binary games have the same cardinality and p will be considered to be a bijection $p : B_i \rightarrow B'_j$. Writing $A_i = A'_j = \{1, \dots, \ell\}$ and $B_i = \{i_1, \dots, i_\ell\}$ and $B'_j = \{j_1, \dots, j_\ell\}$, the action bijection $\varphi_i(p) = q$, $1 \leq p \leq \ell$, induces the bijection $p(i_p) = j_q$ between both blocks. This concludes the definition of p .

In ψ' all the f_i for $1 \leq i \leq k$ are taken to be identities. Let us prove that ψ' is a strong isomorphism. The proof is stated as a sequence of claims.

ψ' maps any strategy profile b_i for players in B_i into a strategy profile $b'_{\pi(i)}$ for players in $B'_{\pi(i)}$, it can be written $\psi' : B_i \rightarrow B'_{\pi(i)}$ and $\psi'(b_i) = b'_{\pi(i)}$. This is clear because p gives a bijection between B_i and $B'_{\pi(i)}$.

If $|B_i| = \ell$, profile $b_i = 0^{p-1}10^{\ell-p} = \text{binify}(p)$ maps into $\psi'(b_i) = 0^{q-1}10^{\ell-q} = \text{binify}(q)$ iff $\varphi_i(p) = q$. As all the f_i 's are identities and p is a permutation of $\{1, \dots, \ell\}$, a binified action is mapped into a binified action. Moreover as $p(i_p) = j_q$ the 1 in position p is mapped into the 1 in position q . As ψ' maps strategy profiles between blocks B_i and $B_{\pi(i)}$, $\psi'(\text{binify}(p)) = \text{binify}(\varphi_i(p))$ can be written.

Given $a = (a_1, \dots, a_n) \in A$ in Γ and $\text{binify}(a) = (\text{binify}(a_1), \dots, \text{binify}(a_n))$, it holds that, for binary action games: $\psi'(\text{binify}(a)) = (\text{binify}(a'_1), \dots, \text{binify}(a'_n))$ such that $\text{binify}(a'_{\pi(i)}) = \text{binify}(\varphi_i(a_i))$.

For any $a \in A$ in Γ , it holds that, $\psi'(\text{binify}(a)) = \text{binify}(\psi(a))$. For $a = (a_1, \dots, a_n) \in A$ in Γ , it holds that, $\psi(a) = (a'_1, \dots, a'_n)$ with $a'_{\pi(i)} = \varphi_i(a_i)$. As $\text{binify}(\psi(a)) = (\text{binify}(a'_1), \dots, \text{binify}(a'_n))$, $\text{binify}(\psi(a)) = \psi'(\text{binify}(a))$ is obtained.

For $p(i_p) = j_q$, it holds that, $u'_{j_q}(\psi'(\text{binify}(a))) = u'_{i_p}(\text{binify}(a))$. Note that, since we have $u'_{j_q}(\psi'(\text{binify}(a))) = u'_{j_q}(\text{binify}(\psi(a))) = u_j(\psi(a))$ and $u'_{i_p}(\text{binify}(a)) = u_i(a)$ and ψ is a morphism, the identity holds.

As ψ' maps bijectively bad strategy profiles, in this case utilities are the penalty payoff, and μ and ψ' are a morphism.

For the reverse implication, it is assumed that $\psi' = (p, f_1, \dots, f_k)$ is a strong isomorphism between the games $\text{BINARYACT}(\Gamma_1, \mu)$ and $\text{BINARYACT}(\Gamma_2, \mu)$ having players N'_1 and N'_2 with $N'_1 = N'_2$. The strategy profiles in both binary actions games are A'_1 and A'_2 . Now, a mapping $(\pi, \varphi_1, \dots, \varphi_n)$ of Γ_1 to Γ_2 is defined. The permutation of players π mimics the block permutation induced by p . Thus, if B_i is mapped into $B'_{p(i)}$, $\pi(i) = p(i)$ can

be set.

The i action bijection is defined as follows: The action j in A_i corresponds in $\text{BINARYACT}(\Gamma_1, \mu)$ to the profile $\text{binify}(j)$ in block B_i . As this block is mapped into $B'_{p(i)}$, the profile is mapped into another good profile $\text{binify}(j')$, and $\varphi_i(j) = j'$ is defined. The mapping $(\pi, \varphi_1, \dots, \varphi_n)$ is a strong isomorphism from Γ_1 to Γ_2 . Again, the proof is stated as a sequence of claims.

$\psi' : A'_1 \rightarrow A'_2$ induces a bijection between $\psi' : \text{bad}(A'_1) \rightarrow \text{bad}(A'_2)$. Let $a' \in \text{bad}(A'_1)$ then $u'_i(a') = \mu$, as μ is a penalty payoff and ψ' is a morphism, $u'_{p(i)}(\psi(a')) = \mu$ but this forces $\psi(a') \in \text{bad}(A'_2)$ and $\psi'(\text{bad}(A'_1)) \subseteq \text{bad}(A'_2)$. Given $a' \in \text{bad}(A'_1)$, any player gets μ and then $\psi^{-1}(a) \in \text{bad}(A'_1)$. As $\text{good}(A'_1) = A'_1 \setminus \text{bad}(A'_1)$ there is also a bijection $\psi' : \text{good}(A'_1) \rightarrow \text{good}(A'_2)$.

Note that N'_1 can be partitioned into the different blocks of players as $N'_1 = B_1 \cup \dots \cup B_n$ and $N_2 = B'_1 \cup \dots \cup B'_n$ being n the number of players in Γ and Γ' .

Given a block B_k in N'_1 and $i, j \in B_k$ with $i \neq j$, it is impossible that $p(i)$ and $p(j)$ belongs to different blocks of N'_2 . Suppose that B_k has ℓ players and $1 \leq i < j \leq \ell$. The strategy profile $b_k = \text{binify}(i) = 0^{i-1} \mathbf{1} 0^{j-i-1} \mathbf{0} 0^{\ell-j}$ for block B_k is considered. All other blocks take the corresponding $\text{binify}(1)$. Then, $a' = (b_1, \dots, b_n) \in \text{good}(A'_1)$ and $\psi'(a') \in \text{good}(A'_2)$. Therefore, we have a factorisation $\psi'(a) = (b'_1, \dots, b'_n)$ corresponding each b'_i , $1 \leq i \leq n$, to a binify process. The profile $c_k = \text{binify}(j) = 0^{i-1} \mathbf{0} 0^{j-i-1} \mathbf{1} 0^{\ell-j}$ is defined for block B_k . All other blocks keep as before $\text{binify}(1)$ then, $c = (c_1, \dots, c_n)$ is good, and $\psi(c)$ is also good and factorises as $\psi(c) = (c'_1, \dots, c'_k)$. Let us compare $\psi'(a) = (b'_1, \dots, b'_n)$ with $\psi'(c) = (c'_1, \dots, c'_k)$. Let $p(i) = i' \in B'_{k_1}$ and $p(j) = j' \in B'_{k_2}$ with $k_1 \neq k_2$. In fact, the bits in $\psi'(a)$ and $\psi'(c)$ coincide everywhere except in positions corresponding to the players $i' \in B'_{k_1}$ and $j' \in B'_{k_2}$. Suppose $|B'_{k_1}| = \ell_1$. The bijection f associated to the position i in B_k is considered: This bijection can be an identity or a negation. When f is the identity, the 1 appearing in position i of block b_k is mapped into the 1 in position i' of b'_{k_1} , as this profile is binified, we have $b'_{k_1} = 0^{i'-1} \mathbf{1} 0^{\ell_1-i'}$. Unfortunately, the 0 appearing in position i of c_k will give $c'_{k_1} = 0^{\ell_1}$ turning a valid profile into an invalid profile. When f is a negation b'_{k_1} , it has a 0 in position i' , and c'_{k_1} will have two 1's, giving a contradiction.

Permutation of p maps bijectively, each block B_i into another B'_j . Let $k \in B_i$ and $p(k) \in B'_j$ then $p(B_i) \subseteq B'_j$. Suppose that $l \in B'_j \setminus p(B_i)$ exists, then $i' = \text{block}(p^{-1}(l))$ verifies $B_i \cap B_{i'} = \emptyset$. Let f and f' the bijections associated to the positions k in B_i and $p^{-1}(l)$ in $B_{i'}$. Let us consider two cases depending on the size of $B_{i'}$.

- Case $|B_{i'}| = 1$. When f' is the identity, defining $B'_j = \text{binify}(p(k))$ we force $B_{i'} = 0$. Fulfilling all the other blocks in $\text{BINARYACT}(\Gamma_2, \mu)$ with $\text{binify}(1)$, it is obtained that ψ' maps a bad profile into a good one, but this is a contradiction. Consider the case, in which f' is a negation. In this case, taking $B'_j = \text{binify}(l)$ the similar contradiction is obtained. The same argumentation allow us to assume that $|B_i| > 1$.

- Case $|B_{i'}| > 1$ and $|B_i| > 1$. When f' is the identity, any bijection associated to a position m in B_i is a negation. Take $B_{i'} = \text{binify}(p^{-1}(l))$ and $B'_j = \text{binify}(l)$ and $B_i = \text{binify}(m)$ as good profiles which map into good profiles, then the 1 in position m in B_i is transformed into a 0 in B'_j . Therefore $B_i = \text{binify}(1) = 10^{|B_i|-1}$ will give $|B_i| - 1 > 0$ 1's in B'_j and the number of 1's in such a block will be at least 2.

Consider the case, in which f' is a negation. As B'_j has at least two positions, take a position m in a block such that $m \neq p^{-1}(l)$ and fix $B_i = \text{binify}(m)$. This profile fixes a 0 in position $p^{-1}(l)$ of $B_{i'}$ and a 1 in position l of B_j and the preceding argument is applied.

It is assumed that all the bijections f are identities. It is supposed that $p(B_i) = B'_j$. Three cases depending on the size of B_i are considered.

- Case $|B_i| = 1$. In this case, $B_{j'}$ has also one element. As good profiles map into good profiles, the bijection associated to this element has to be the identity.
- Case $|B_i| = 2$. See in detail the different possibilities: Call $B_i = \{1, 2\}$ and $B'_j = \{1', 2'\}$ and call the corresponding bijections f_1 and f_2 . There are two possibilities for p . The cases $p(1) = 1'$ and $p(2) = 2'$ are considered, and look at the different possibilities for f_i .
 - When $f_1 = f_2$ are identities, the property holds.
 - When f_1 is the identity and f_2 is a negation, a contradiction is obtained because $B_i = \text{binify}(2)$ is mapped into $B'_j = 00$. When f_1 is a negation and f_2 is the identity, the same argumentation applies with $B_i = \text{binify}(1)$.
 - When f_1 and f_2 are negations, it should be remarked that *bad profile* maps into *bad profile* because 00 maps to 11 and 11 maps to 00. In addition, *good profiles* map into *good profiles* because 10 maps to 01 and 01 maps into 10. No bad happens in this case. To obtain identities, another morphism ψ'' is defined such that $p(1) = 2'$, $p(2) = 1'$ and $f_1 = f_2$ are identities. Under ψ'' , *bad profiles* map into *bad profiles* because 00 maps to 00 and 11 maps into 11. Much better *good profiles* map into ψ' and into ψ'' similarly, because 10 maps to 01 and 01 maps to 10. Note that ψ'' and ψ' are isomorphic, therefore one can take ψ'' , where f_1 and f_2 are identities.

When $p(1) = 2'$ and $p(2) = 1'$, the proof is similar to the preceding case.

- Case $|B_i| > 2$. If all the f 's associated to B_i are identities, the property holds. Otherwise, there is a position l such that f_l is a negation. As $B_i = \text{binify}(l)$ maps to a good profile, there exists l' such that $B'_j = \text{binify}(l')$. Moreover, as f_l is a negation $l' \neq p(l)$, $f_{p^{-1}(l')}$ is also a negation, and $|B_i| > 2$. Then, there is a position

$k \notin \{l, p^{-1}(l')\}$ in B_i , and $B_i = \text{binify}(k)$ give at least two one's corresponding to positions l and $p^{-1}(l')$. Therefore, a contradiction is obtained.

To summarise, given a strong isomorphism $\psi' = (p, f_1, \dots, f_k)$ between the games $\text{BINARYACT}(\Gamma_1, \mu)$ and $\text{BINARYACT}(\Gamma_2, \mu)$, we have that p maps blocks of players bijectively and it can be assumed that all the f_i , $1 \leq i \leq k$ are identities.

Let us consider the mapping $(\pi, \varphi_1, \dots, \varphi_n)$ of Γ_1 to Γ_2 . The permutation of players π mimics the block permutation induced by p , thus if B_i is mapped to $B'_{p(i)}$, $\pi(i) = p(i)$ can be set. The i action bijection is defined as follows: The action j in A_i corresponds in $\text{BINARYACT}(\Gamma_1, \mu)$ to the profile $\text{binify}(j)$ in block B_i . As this block is mapped into $B'_{p(i)}$, the profile is mapped into another good profile: then, $\text{binify}(j')$ and $\varphi_i(j) = j'$ are defined. It is straightforward to show that the mapping $(\pi, \varphi_1, \dots, \varphi_n)$ is a strong isomorphism from Γ_1 to Γ_2 . □

Now, a binary actions' game is transformed into a binary game. There is a game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ in which $A_i = \{0, 1\}$, for any $i \in N$, and $N = \{1, \dots, n\}$. There are positive values t and m such that, for any action profile a and any player i , $|u_i(a)| \leq t$ and $m \geq \{n, t\}$. Then, it is possible to set $k = n + tn + m + 2$. Furthermore, the following game is considered.

$\text{BINARY}(\Gamma, t, m) = (N', (A'_i)_{i \in N'}, (u'_i)_{i \in N'})$, where $N' = \{1, \dots, k\}$ and, for any $i \in N'$, $A'_i = \{0, 1\}$. The set N' is partitioned into $n + 2$ consecutive intervals B_0, \dots, B_n, B_{n+1} so that the interval B_0 has exactly n players. For $1 \leq i \leq n$, block B_i has t players, and block B_{n+1} has $m + 2$ players. Inside the blocks, relative coordinates are used to identify the players. In all blocks, coordinates start at 1 except for the last block, which starts with 0. In this situation, a strategy profile a is usually factorised as $a = x b_1 \dots b_n z$, where $x = x_1 \dots x_n$, $b_i = b_{i_1} \dots b_{i_t}$ and $z = z_0 \dots z_{m+1}$. To improve readability, $a = x b_1 \dots b_n z$ is written as $a = (x, b_1, \dots, b_n, z)$. The utility function, by properties of the strategy profile, is defined and it is assumed that $a = x b_1 \dots b_n z$ is a strategy profile of $\text{BINARY}(\Gamma, t, m)$.

- In the case that for some ℓ , $0 \leq \ell \leq m + 1$, the last one ℓ bits of z are 1, all players except the last one ℓ get utility 0. The remaining players get utility 1. Observe that, in the case in which $\ell = 0$, one has that $z = 0^{m+2}$, therefore all players get utility 0.
- In the case that for some j , $1 \leq j \leq t$, the j -th bit of z is the only 1 in z , all players in blocks B_1, \dots, B_n that do not occupy position j in their block get utility 0, all players in blocks B_0 and B_{n+1} get utility 1, and all the remaining players get as utility their action.

- In the case that the 0-th bit of z is the only 1 in z , for any i , $1 \leq i \leq n$, player i in block B_0 and all the players in block B_i get utility 1 when $u_i(x) = b_i$, and 0 otherwise. All players in block B_{n+1} get utility 0.
- In the remaining cases all players get utility 1.

In a strategy profile $a = xb_1 \dots b_n z$, the parts $x = x_1 \dots x_n$, $b_i = b_{i_1} \dots b_{i_t}$ and $z = z_0 \dots z_{m+1}$ are binary words. Then, the whole profile a is also a binary string, having length $k = n + tn + m + 2$. As the utilities for all players are either 0 or 1, take all the utilities together as a binary string $u(a) = u_1 \dots u_k$.

Example 7.2 This is the continuation of the game used in Example 7.1. The game $\Gamma = \text{BINARYACT}(BS, 0)$ where actions are binary but utilities are not, is considered. The utilities values are 1, 2 and 3 obtained from the utilities in BS and 0, corresponding to the utility of any bad profile. As expressed in binary form the utilities are 00, 01, 10 and 11, with two bits sufficing. The game Γ has $n = 4$. Therefore, $t = 2$ and $m = 4$ can be taken. The game $\text{BINARY}(\Gamma, 2, 4)$ has $k = n + tn + m + 2 = 18$ players.

Set N' is partitioned into 6 blocks. Block B_0 contains 4 players, each B_i , $1 \leq i \leq 4$ has 2 players and B_5 has 6 players. A strategy profile has the format $a = xb_1 \dots b_4 z$ with $x = x_1 \dots x_4$, $b_i = b_{i_1} b_{i_2}$ for $1 \leq i \leq 4$ and $z = z_0 z_1 \dots z_4 z_5$. Utilities are coded $u(a) = u_1 \dots u_{18}$. Then, let us consider the following examples of utilities in each of the preceding four cases:

- For instance, taking $\ell = 3$. Then, $a = xb_1 \dots b_4 0^3 1^3$ and $u(a) = 0^{15} 1^3$. The block structure of the preceding utility can be displayed as,

$$u(a) = \underbrace{0 \dots 00}_{B_0} \underbrace{\dots 0000}_{B_1, \dots, B_4} \overbrace{111}^{\ell}_{B_5}$$

There are profiles such as $a = xb_1 \dots b_4 0^6$, when $\ell = 0$. In this case, all players get utility 0.

- Profile z “looks at” the second bit of each b_i , $1 \leq i \leq 4$, when $z = z_0 z_1 z_2 z_3 z_4 z_5 = 001000$. In this case,

$$u(a) = \underbrace{1 \dots 10}_{B_0} \underbrace{0b_{1_2} 0b_{2_2} 0b_{3_2} 0b_{4_2}}_{B_1, \dots, B_4} \underbrace{1 \dots 1}_{B_5}$$

Profile z points to an “out of range” position in blocks b_i , $1 \leq i \leq 4$, when $z =$

000010. In this case,

$$u(a) = \underbrace{1\dots 1}_{B_0} \underbrace{0\dots\dots 0}_{B_1, \dots, B_4} \underbrace{1\dots 1}_{B_5}$$

- The connections between strategy profiles and utilities appear when $z = 100000$. It has to be noted at this point that in the game $\Gamma = \text{BINARYACT}(BS, 0)$ it holds that,

$$u_1(1010) = u_2(1010) = 11, u_3(1010) = u_4(1010) = 10$$

Profiles starting and ending as $a = (1010, b_1, \dots, b_4, 100000)$ are considered. For instance $a = (1010, 10, 11, 00, 10, 100000)$. Since $b_1 = 10 \neq u_1(1010)$, player 1 and players in block B_1 get utility 0. Since $b_2 = 11 = u_2(1010)$, player 2 and players in block B_2 get 1 utility. Following this argumentation,

$$u(a) = \underbrace{0101}_{B_0} \underbrace{00}_{B_1} \underbrace{11}_{B_2} \underbrace{00}_{B_3} \underbrace{11}_{B_4} \underbrace{000000}_{B_6}$$

- In all the remaining cases, all players get utility 1.

Lemma 7.2 Let Γ_1, Γ_2 be two binary action games given in general form. Set $t = \max\{t_1, t_2, 3\}$, where t_i is the time allowed to the utility TM of game Γ_i , and $m = \max\{n_1, n_2\}$, where n_i is the number of players in game Γ_i . There is a strong isomorphism between Γ_1 and Γ_2 iff there is a strong isomorphism between $\text{BINARY}(\Gamma_1, t, m)$ and $\text{BINARY}(\Gamma_2, t, m)$.

Proof. Given a mapping $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ of Γ_1 into Γ_2 , the mapping $\psi' = (p, f_1, \dots, f_k)$ of $\text{BINARY}(\Gamma_1, t, m)$ into $\text{BINARY}(\Gamma_2, t, m)$ is considered. In such mapping, f_i is the identity, for any $1 \leq i \leq n$, $f_i = \varphi_i$, and, for any $i > n$. The permutation p on $B_{1,0}$ is exactly π . For any $1 \leq i \leq n$, block $B_{1,i}$ is mapped to block $B_{2,\pi(i)}$ and block $B_{1,n+1}$ is mapped to block $B_{2,n+1}$. Players inside each block are assigned preserving the relative order of positions in the block. It is straightforward to show that if ψ is an isomorphism, then ψ' is also an isomorphism.

For the reverse implication, it is assumed that $\psi' = (p, f_1, \dots, f_k)$ is a strong isomorphism between the $\text{BINARY}(\Gamma_1, t, m)$ and $\text{BINARY}(\Gamma_2, t, m)$ games. In such case, Γ_1 and Γ_2 have the same number n of players. In this case, it can be shown that a permutation p preserves blocks and relative positions inside interior blocks. Therefore, a mapping $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ is defined. In this mapping, π is the restriction of p to block $B_{1,0}$ and, for any $1 \leq i \leq n$, $\varphi_i = f_i$.

Let $a = xb_1\dots b_nz$ be a strategy profile for $\text{BINARY}(\Gamma_1, t, m)$, where $x = x_1\dots x_n$, $b_i = b_{i_1}\dots b_{i_t}$ and $z = z_0\dots z_{m+1}$ are binary words. The utilities of a are represented as a

binary string $u(a) = u_1, \dots, u_{n+tn+m+2}$. When we speak in general terms about a property of the construction subindices will not be used. However, $u_1(a)$ and $u_2(\psi(a))$ will be used to denote vector utilities for the first or second game. As usual, for a binary string w is used $|w|_1$ to denote the number of 1's present in w . Note that, for a strategy profile a , $|u_1(a)|_1 = |u_2(\psi(a))|_1$. According to the definition of utilities for $\text{BINARY}(\Gamma, t, m)$ for any profile a , we have that,

1. if $z = 0^{m+2-\ell}1^\ell$, then $|u(a)|_1 = \ell$, and at least one player in block B_{n+1} gets utility 1.
2. if $z = 00^{j-1}10^{m+1-j}$, for some $1 \leq j < t$, then $n+m+2 \leq |u(a)|_1 \leq n+t+m+2$, and all players in block B_{n+1} get utility 1.
3. if $z = 10^{m+1}$, then $|u(a)|_1$ is a multiple of $t+1$, and all players in block B_{n+1} get utility 0.
4. In the remaining cases, $|u(a)|_1 = n+tn+m+2$.

Permutation p maps the block $n+1$ of Γ_1 to the block $n+1$ of Γ_2 . Furthermore, the restriction of p to $B_{1,n+1}$ is the identity and, for any $j \in B_{1,n+1}$, f_j is the identity. The claim follows from condition 1, as this is needed to guarantee that, when $z = 0^{m+2-\ell}1^\ell$, $|u_1(a)|_1 = |u_2(\psi(a))|_1$.

Let $\text{BIT}(j)$ be the set of players that appears at the j -th position in a block B_1, \dots, B_n .

For any $1 \leq j \leq t$, permutation p maps $\text{BIT}_1(j)$ to $\text{BIT}_2(j)$. Furthermore, for any $i \in \text{BIT}_1(j)$, f_i is the identity. The rigidity of ψ on block $B_{1,n+1}$ forces that profile a , in which all players $i \in \text{BIT}_1(j)$ select action 1 and $z = 00^{j-1}10^{m+1-j}$, creating an utility string with exactly $2n+m$ ones. Therefore, the only possibility for ψ to remain as an isomorphism, is the one expressed in the claim.

As a consequence of the previous claims we have that permutation p maps the players in block $B_{1,0}$ to block $B_{2,0}$.

For any $1 \leq i \leq n$, permutation p maps block $B_{1,i}$ to block $B_{2,p(i)}$. Furthermore, for any $1 \leq j \leq n$, the player in the j -th position of $B_{1,i}$ is mapped by ψ to the j -th position of $B_{2,p(i)}$. Profile a is considered. In this profile a , $z = 10^{m+1}$ and x verify that $b_i = u_i(x)$ and, for any $\ell \neq i$, $b_\ell \neq u_\ell(x)$. The rigidity of ψ on block $B_{1,n+1}$ forces that in $|u_1(a)| = t+1$. In $u_2(\psi(a))$, it is known that the utility for player $p(i)$ has to be one and therefore, all utilities of all players in $B_{2,p(i)}$ must be one. Again, the only possibility is the one expressed in the first part of the claim. The second part follows as a consequence of the first part and the previous claim.

Putting all together, a morphism $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ can be defined, in which π is the restriction of p to block $B_{1,0}$ and, for any $1 \leq i \leq n$, $\varphi_i = f_i$. Profile a is considered, in which $z = 10^{m+1}$, and x verifies that, for any $1 \leq i \leq n$, verifies $b_{1,i} = u_i(x)$. Then

$u_1(a)$ has a one in all positions except the last $m + 2$ that hold a 0. Furthermore, $\psi(a) = \pi(x)b_{2,1} \dots b_{2,n}z$ and, for any $1 \leq i \leq n$, if $b_{2,\pi(i)} = b_{1,i}$. Therefore we have that, for any $1 \leq i \leq n$, $u_1(x) = u_2(\psi(x))$, therefore ψ is an isomorphism. \square

Given a binary game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ with n players, such that for any $1 \leq i \leq n$, utility u_i has range $\{0, 1\}$ and $A_i = \{0, 1\}$. A circuit C_Γ on $4n + 2$ variables is constructed. Recall that, when $u_i(x)$ is computed by a Turing machine in polynomial time, Ladner's construction [48] gives us a polynomial size circuit when computing the same function.

Circuit C_Γ . The variables in C_Γ are grouped in four blocks, the X -block contains the first n -variables. The Y -block is formed by these variables in positions from $n + 1$ to $2n$. The C -block contains the following $n + 2$ variables, and the D -block the remaining variables. For the sake of readability, the set of variables is split into four parts $a = (x, y, c, d)$, where $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$, $c = (c_1, \dots, c_{n+2})$, and $d = (d_1, \dots, d_n)$.

The circuit C_Γ is defined with the help of the following $n + 2$ circuits,

$$\begin{aligned} C_1(x, y, d) &= [(x_1 = \bar{d}_1) \wedge \dots \wedge (x_n = \bar{d}_n) \wedge (u_1(x) = y_1) \wedge \dots \wedge (u_n(x) = y_n)] \\ C_2(y) &= [y_1 \vee \dots \vee y_n] \\ C_{i+2}(x_i, y_i, d_i) &= [\bar{y}_i \wedge (x_i = \bar{d}_i)] \quad \text{for } 1 \leq i \leq n. \end{aligned}$$

Finally,

$$C_\Gamma(x, y, c, d) = \begin{cases} 0 & \text{if } \sum_{1 \leq i \leq n+2} c_i = 0 \text{ or } \sum_{1 \leq i \leq n+2} c_i > 1 \\ C_j & \text{if } \sum_{1 \leq i \leq n+2} c_i = 1 \text{ and } c_j = 1 \end{cases}$$

The previous construction is used to reduce the ISO problem to the CIRCUITCONG problem.

Lemma 7.3 *Let Γ and Γ' be two binary games in general form with at least two players each. There is a congruence isomorphism between C_Γ and $C_{\Gamma'}$, iff there is a strong isomorphism between Γ and Γ' .*

Proof. It is assumed that $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ is a strong isomorphism from Γ to Γ' . Also, the following variable transformation preserving blocks is considered. Variable x_i is mapped to variable $x'_{\pi(i)}$. With permutation φ_i , the same happens with block D . The variable y_i is mapped to variable $y'_{\pi(i)}$ with permutation to the identity function, variable c_1 is mapped to variable c'_1 , c_2 to c'_2 , and c_{2+i} to $c'_{2+\pi(i)}$, and all the block c with permutation to the identity function.

For the reverse implication, let $\psi' = (p, f_1, \dots, f_{4n+2})$ be a congruence morphism between C_Γ and $C_{\Gamma'}$. Given $a = (a_1, \dots, a_{4n+2})$, for $1 \leq i \leq 4n+2$ the value $p(i)$ points to the image of a_i . When $\psi(a) = (a'_1, \dots, a'_{4n+2})$ it holds that, $a'_{p(i)} = f(a_i)$.

When $a = (x, y, c, d)$, we note $p(x_i)$ the position of the image of x_i and we take similar conventions for $p(y_i)$, $p(c_i)$ and $p(d_i)$ to avoid confusions. Similarly, the value of the image of x_i will be $f(x_i)$. The congruence verifies that for any truth assignment a to the variables of C_Γ , there is $C_\Gamma(a) = C_{\Gamma'}(\psi'(a))$. Congruence Ψ' allows to be proved that ψ preserves the structure of the C and Y blocks as follows:

The values of variables c_1, \dots, c_{n+2} are used to activate the different circuits C_1, \dots, C_{n+2} that form C_Γ . Each of these circuits has different properties. As before, the proof is stated as a series of claims.

Permutation pf maps to the variables in the C -block of C_Γ to the C -block of $C_{\Gamma'}$. By contradiction, it is assumed that there are $k \geq 1$ variables mapped from outside the C -block of C_Γ to the C -block of $C_{\Gamma'}$. Please see that an assignment a can be forced in which there is only one 1 in position 2 of the C -block for which $C_\Gamma(a)$ is true, while in $\psi'(a)$ there are at least two 1's, and that is impossible. The case $k = 1$ is considered in detail. As in $a = (a_1, \dots, a_{4n+2})$, one position leaves block C , there are an a_i in blocks X, Y or D entering block C' in $\psi'(a)$ and $c'_{p(i)} = f(a_i)$. Two cases based on $p(c_2)$ will be considered in the following paragraphs:

- Case $p(c_2)$ is a position in C' . In this case, $c'_{p(c_2)} = f(c_2)$. When the bijection f is the identity, $c'_{p(c_2)} = c_2$, different possible origins of a_i will be considered below:
 - When a_i is located in X we have $a_i = x_i$. Fix $x_i \in \{0, 1\}$ to the value such that $f(x_i) = 1$. It is considered $a = (x, y, c, d)$, such that $x = 0^{i-1}x_i0^{n-i}$, $y = 10^n$, $c = 010^n$ and $d = 0^n$. It holds that $C_\Gamma(a) = C_2(a) = 1$ but $C_{\Gamma'}(\psi(a)) = 0$ because C' contains at least two ones in $c'_{p(i)}$ and $c'_{p(c_2)}$.
 - When a_i is located in D we have $a_i = d_j$, for $j = i - 3n + 2$. Fix $d_j \in \{0, 1\}$ to the value, such that $f(d_j) = 1$. It is considered $a = (x, y, c, d)$, such that $x = 0^n$, $y = 10^n$, $c = 010^n$ and $d = 0^{j-1}d_j0^{n-j}$. Again, it holds that, $C_\Gamma(a) = C_2(a) = 1$ but $C_{\Gamma'}(\psi(a)) = 0$.
 - When a_i is located in Y , it holds that $y_{i-n} = a_i$. Fix the value of y_{i-n} such that $f(y_{i-n}) = 1$. As the Y block has at least two positions, there is j , such that $j \neq i - n$ and $y_j = 1$ can be fixed. Then, $a = (x, y, c, d)$ with $x = 0^n$, $y = 0 \dots 0y_{i-n}0 \dots 0y_j0 \dots 0$ (case $i - n \geq j$, other cases are similar) $c = 010^n$ and $d = 0^n$ verify $C_\Gamma(a) \neq C_{\Gamma'}(\psi(a))$.

Next, the situation in which f is a negation, $c'_{p(c_2)} = \neg c_2$ is considered. As just one position in C is mapped outside C' , there exists a j such that $p(c_j)$ is not a position

in C' , therefore $c_j \neq c_2$. Taking C also as a set, we have that for any $c_k \in C \setminus \{c_2, c_j\}$ it holds that $p(c_k)$ is located in C' . Two cases are considered next:

- For all $c_k \in C \setminus \{c_2, c_j\}$ it holds that $f(c_k)$ is the identity, $c'_{p(c_k)} = c_k$. The value $c'_{p(j)}$ can be forced to be 0, therefore for $C = 010^n$, $C' = 0^{n+2}$ is obtained. As block Y can be chosen having at least one 1, a profile a can be easily built such that $C_\Gamma(a) \neq C_{\Gamma'}(\Psi(a))$.
- There exists $c_k \in C \setminus \{c_2, c_j\}$ such that $f(c_k)$ is a negation, $c'_{p(c_k)} = \neg c_k$. Fixing $C = 010^n$, $c_k = 1$ is obtained. Forcing $c'_{p(i)}$ to be 1, block C' will have at least two 1. As Y has at least two positions, an a not fulfilling the congruence can be easily built.
- Case $p(c_2)$ is a position in X' , Y' or D' . Intuitively, c_2 leaves the C block and at the elements $c_k \in C \setminus \{c_2\}$ have to be examined. Two cases are considered next:
 - For any $c_k \in C \setminus \{c_2\}$, the bijection $f(c_k)$ is the identity. Fixing $c'_{p(i)}$ to be 0 and $C = 010^n$, $C' = 0^{n+2}$ is obtained and an a not fulfilling the congruence can be built.
 - There exists $c_k \in C \setminus \{c_2\}$ such that $f(c_k)$ is a negation. Fixing $c'_{p(i)}$ to be 1 and $C = 010^n$, C' has at least two 1 and an a not fulfilling the congruence. can be built.

This concludes the analysis of the impossibility when $k = 1$. When $k > 1$ the analysis follows the same ideas.

All functions associated to variables in the C -block are the identity. If there are more than two negations, Ψ' transforms an input with exactly one 1 in block C to a situation with two 1's in block C . If there is one negation, the situation in which all the bits in C are set to 0 is transformed into another one in which there is only one 1.

We have $p(c_1) = c'_1$ and $p(c_2) = c'_2$. Let us consider the possible misplacements for $p(c_1)$. There are two cases,

- The index $p(c_1) = c_i$ is located in one of the last n positions of C' . Then, $C = 10^{n+1}$ is mapped into $C' = 0^{i-1}10^{n+2-i}$ and the activated circuits are $C_\Gamma = C_1$ and $C_{\Gamma'} = C_i$. Fix $x'_{i-2} = y'_{i-2} = 0$ and $d'_{i-2} = 1$ and $C_{\Gamma'} = 1$. As p maps C into C' it also maps $X \cup Y \cup D$ bijectively into $X' \cup Y' \cup D'$. As $|X| + |D| \geq 4$ and $x'_{i-2} y'_{i-2} d'_{i-2}$ are fixed, only three anti-images have been fixed and there are at least one "free" position in the $X \cup D$ blocks. It is assumed that x_j is the free position (the case d_j is similar) and that is looks the corresponding d_j . If d_j is fixed, take $x_j = d_j$. If d_j is free, define $d_j = x_j = 1$. In both cases $C_1 = 0$.

- Index $p(c_1)$ points to c'_2 , that is $c_2 = c_1$. Block $C = 10^{n+1}$ is mapped to 010^n and under this situation $C_\Gamma = C_1$ and $C_{\Gamma'} = C_2$. Fixing an arbitrary bit y'_i in Y' and looking at the possible anti-image of y'_i follows that in the bijection $p : X \cup Y \cup D \rightarrow X' \cup Y' \cup D'$ only the anti-image of y'_i fixed. Suppose that the anti-image is one and is an x_j , then fix $d_j = x_j$. When the anti-image belongs to Y , choose one x_k , fix it and the corresponding d_k as well to 1. When the anti-image is d_j fix x_j to the same value. In all cases, $C_1 = 0$.

It is then proven that $p(c_1) = c'_1$. To prove $p(c_2) = c'_2$ the same train of thought has need to be followed.

Permutating p maps on to the variables in the Y -block of C_Γ to the Y -block of $C_{\Gamma'}$. Furthermore, the functions associated to the variables in the Y -block are the identity. This is a consequence of the rigidity of ψ' on c_2 and the definition of the C_2 formula.

Furthermore, the function f_i , for i in block C or Y , is the identity. This allows us to consider the permutation π on $\{1, \dots, n\}$, such that $p(c_{i+2}) = c'_{\pi(i)+2}$. Moreover, it will be proven that this permutation verifies $p(x_i) = x'_{\pi(i)}$ iff $p(d_i) = d'_{\pi(i)}$ and $p(x_i) = d'_{\pi(i)}$ iff $p(d_i) = x'_{\pi(i)}$. π is considered to be the permutation on $\{1, \dots, n\}$, such that $p(c_{i+2}) = c'_{\pi(i)+2}$.

For any $1 \leq i \leq n$, positions i in blocks X , Y and D of C_Γ are mapped to positions $\pi(i)$ of blocks X , Y and D of $C_{\Gamma'}$. Furthermore, $p(y_i) = y'_{\pi(i)}$. This result is enforced by the definition of the n formulas C_{i+2} , as each of them forces to combine the input bits x_i and y_i with d_i . The last part of this section takes into account that the Y -block of C_Γ is mapped to the Y -block of $C_{\Gamma'}$.

The above result implies that, for any $1 \leq i \leq n$, either $p(x_i) = x_{\pi(i)}$ or $p(x_i) = d_{\pi(i)}$. Furthermore, the permutation associated to x_i and d_i must be the same,

$$p(x_i) = x'_{\pi(i)} \text{ iff } p(d_i) = d'_{\pi(i)} \text{ and } p(x_i) = d'_{\pi(i)} \text{ iff } p(d_i) = x'_{\pi(i)},$$

otherwise, an input for which $C_{i+2}(x_i, y_i, d_i) = 1$ can be found, while there is also a $C_{\pi(i)+2}(\psi'(x_i, y_i, d_i)) = 0$.

The mapping $\psi'' = (p', f'_1, \dots, f'_{4n+2})$ is considered, such that the behaviour of the permutation and bijections coincides with ψ' in blocks Y and C . In blocks X and D the mapping is given by,

$$p'(x_i) = \begin{cases} x'_{\pi(i)} & \text{if } p(x_i) = x'_{\pi(i)} \\ x'_{\pi(i)} & \text{if } p(x_i) = d'_{\pi(i)} \end{cases} \text{ and } p'(d_i) = \begin{cases} d'_{\pi(i)} & \text{if } p(d_i) = d'_{\pi(i)} \\ d'_{\pi(i)} & \text{if } p(d_i) = x'_{\pi(i)} \end{cases}$$

then, $p' : X \rightarrow X'$ and $p' : D \rightarrow D'$ and the behaviour of p' is the same in both blocks, that is $p'(x_i) = x'_{\pi(i)}$ iff $p'(d_i) = d'_{\pi(i)}$. The bijections are defined as:

$$f'(x_i) = \begin{cases} f(x_i) & \text{if } p(x_i) = x'_{\pi(i)} \\ \neg f(x_i) & \text{if } p(x_i) = d'_{\pi(i)} \end{cases} \quad \text{and} \quad f'(d_i) = \begin{cases} f(d_i) & \text{if } p(d_i) = d'_{\pi(i)} \\ \neg f(d_i) & \text{if } p(d_i) = x'_{\pi(i)} \end{cases}$$

The morphism ψ'' is a congruence. This trivially happens because for any strategy profile a , $\psi'(a) = \psi''(a)$ holds.

Finally, it is easy to prove that the morphism ψ , given by $\psi = (\pi, f'_1, \dots, f'_n)$, is an isomorphism between Γ and Γ' . \square

It is easy to show that CIRCUITCONG is reducible to ISO. A game with as many players as variables, in which the utilities for all players are identical and coincide with the evaluation of the circuit is considered. Taking into account that CIRCUITCONG is equivalent to CIRCUITISO and putting all together, the following is obtained:

Theorem 7.4 *The strong isomorphism problem for strategic games in general form, is polynomially equivalent to the circuit isomorphism problem.*

The ISO problem for strong isomorphisms of games in general form remains equivalent to the ISO problem for strong isomorphisms, when games are restricted to be binary actions or binary games, since the identity trivially reduces the latest problem to the former one.

Formula games in general form are considered. The results also apply to WBFG games [71], where actions are given in explicit way. The proof follows the same steps as in the previous case. Now, a description of the games provided in the reduction as formula games will be shown, that can be computed in polynomial time.

Theorem 7.5 *The strong isomorphism problem for formula games and WBFG in general form are equivalent to the boolean formula isomorphism problem.*

Proof. The game $\text{BINARYACT}(\Gamma, \mu)$ when Γ is a formula game in general form is a formula game. The game $\text{BINARY}(\Gamma, t, m)$, when Γ is a binary action formula game in general form, is a formula game. A description in general form of the games $\text{BINARYACT}(\Gamma, \mu)$ and $\text{BINARY}(\Gamma, t, m)$ can be computed in polynomial time. Furthermore, a description of the circuit C_Γ , for a binary formula game Γ can be obtained in polynomial time. First, it is shown that the $\text{BINARYACT}(\Gamma, \mu)$ game, for a given formula game in general form $\Gamma = \langle 1^n, A_1, \dots, A_n, 1^\ell, (\varphi_{i,j})_{1 \leq i \leq n, 0 \leq j < \ell} \rangle$, as defined in Page 81, is a formula game whose description can be computed in polynomial time.

Recall that the utility functions of $\text{BINARYACT}(\Gamma, \mu)$ are defined as follows:

$$u'_i(a') = \begin{cases} u_{\text{block}(i)}(\text{binify}^{-1}(a')) & \text{if } a' \in \text{good}(A'), \\ \mu & \text{if } a' \in \text{bad}(A'). \end{cases}$$

where $\text{good}(A') = \{\text{binify}(a) \mid a \in A\}$ and $\text{binify}(j) = 0^{j-1}10^{k_i-j}$. To compute the utilities, it will be needed to show that the function binify^{-1} can be represented by a boolean formula as well as the property $a' \in \text{good}(A')$. For doing so, it will be shown that they can be computed in NC^1 , and the same argumentation was used in the proof of Claim 2.1 in Chapter 2 to construct the formulas.

For $a' \in \text{good}(A')$, it must happen that the sum of all its bits is 1, and this can be computed in NC^1 . To compute $\text{binify}^{-1}(a')$ for some $a' \in \text{good}(A')$, let $a' = 0^{j-1}10^{k_i-j}$ be assumed. The suffix sum of the bits of a' is computed, thus getting $b = 1^j0^{k_i-j}$. Then, j is the sum of the bits of b .

Finally, the set of formulas describing the utilities for $\text{BINARYACT}(\Gamma, \mu)$ in polynomial time can be constructed using the formula for $a' \in \text{good}(A')$ and the ones that compute the bits of $\text{binify}^{-1}(a')$, combined with a constant μ , and formulas describing the utilities of the player's in Γ . Therefore, according to Lemma 7.1, the ISO problem for formula games in general form is equivalent to the the ISO problem for formula games in general form with binary actions.

Now, it will be shown that given $\Gamma = \langle 1^n, A_1, \dots, A_n, 1^\ell, (\varphi_{i,j})_{1 \leq i \leq n, 0 \leq j < \ell} \rangle$, a formula game in general form with binary actions, the game $\text{BINARY}(\Gamma, t, m)$, as defined in Page 86, is a formula game whose description can be computed in polynomial time.

Recall that $\text{BINARY}(\Gamma, t, m)$ is the game $(N', (A'_i)_{i \in N'}, (u'_i)_{i \in N'})$ where $N' = \{1, \dots, k\}$ and, for any $i \in N'$, $A'_i = \{0, 1\}$ where $k = n + tn + m + 2$. The set N' is partitioned into $n + 2$ consecutive intervals B_0, \dots, B_n, B_{n+1} , so that the interval B_0 has exactly n players, for $1 \leq i \leq n$, the block B_i has t players, finally block B_{n+1} has $m + 2$ players. As before, a strategy profile a is usually factorised as $a = x b_1 \dots b_n z$ where now $x = x_1 \dots x_n$, $b_i = b_{i\ell-1} \dots b_{i0}$ and $z = z_0 \dots z_{m+1}$. Note that if the formula game uses ℓ formulas per player, then $t = \ell + 1$.

To express the utilities by a boolean formula, the following auxiliary formulas for $z = z_0, \dots, z_{m+1}$ are considered:

$$\begin{aligned} \text{FROM}_i(z) &= \left(\bigwedge_{j=0}^{i-1} \neg z_j \right) \wedge \left(\bigwedge_{j=i}^{m+1} z_j \right) \quad \text{for } 0 \leq i \leq m+1 \\ \text{ONLY}_i(z) &= \left(\bigwedge_{j=0}^{i-1} \neg z_j \right) \wedge z_i \wedge \left(\bigwedge_{j=i+1}^{m+1} \neg z_j \right) \quad \text{for } 0 \leq i \leq m+1 \end{aligned}$$

The previous formulas allow the expression of the different conditions considered in the definition of a game $\text{BINARY}(\Gamma, t, m)$.

$$\begin{aligned}\text{ONE}(z) &= \bigvee_{i=0}^{m+1} \text{FROM}_i(z) \\ \text{TWO}(z) &= \bigvee_{i=1}^t \text{ONLY}_i(z) \\ \text{THREE}(z) &= \text{ONLY}_0(z) \\ \text{FOUR}(z) &= \neg(\text{ONE}(z) \vee \text{TWO}(z) \vee \text{THREE}(z))\end{aligned}$$

Note that predicates ONE , TWO , TWO , FOUR give a partition of the strategy profiles. Recall that the utility of player i in Γ is given by the equation $u_i(a_1, \dots, a_n) = \sum_{0 \leq j < \ell} \varphi_{i,j}(a_1, \dots, a_n) 2^j$. The formula,

$$\text{EQU}_i(x, b_i) = \bigwedge_{j=0}^{\ell} (\varphi_{ij}(x) \wedge b_{ij}) \vee (\neg \varphi_{ij}(x) \wedge \neg b_{ij}),$$

which express the fact that b_i is the utility of player i in game Γ is also considered.

Below, a formula for each “type of player” that allows to compute their utility in game $\text{BINARY}(\Gamma, t, m)$ is provided.

- The utility for player α in position β of block j ($1 \leq j \leq n$). The formula is expressed as disjunction of the four cases.
 - When $\text{ONE}(z)$ holds, the utility is 0. This gives a term $\text{ONE}(z) \wedge 0$, which is equivalent to 0.
 - When $\text{TWO}(z)$ hold, there are two cases. When $\text{ONLY}_\beta(z)$ holds, the position of player α inside the block j coincides with the position of the 1 in z and then, the utility is $b_{j\beta}$. When $\text{ONLY}_\beta(z)$ is false the utility is 0. Therefore, this part contributes with a term $\text{TWO}(z) \wedge \text{ONLY}_\beta(z) \wedge b_{j\beta}$.
 - When $\text{THREE}(z)$ holds, all players in block j have the same boolean utility, defined as the value of the expression ($u_j(x) = b_j$). This part is encoded as $\text{THREE}(z) \wedge \text{EQU}_j(x, b_j)$.
 - When $\text{FOUR}(z)$ holds, the value of the utility is 1, therefore a term $\text{FOUR}(z) \wedge 1$ is obtained.

Using basic properties of boolean functions, it is obtained,

$$\Psi_\alpha(a) = (\text{TWO}(z) \wedge \text{ONLY}_\beta(z) \wedge b_{j\beta}) \vee (\text{THREE}(z) \wedge \text{EQU}_j(x, b_j)) \vee \text{FOUR}(z)$$

- The utility for player α in position β of block 0 is:

$$\Psi_\alpha(a) = \text{TWO}(z) \vee (\text{THREE}(z) \wedge \text{EQU}_\beta(x, b_\beta)) \vee \text{FOUR}(z)$$

- The utility for player α in position β of block $n + 1$ is:

$$\Psi_{\alpha}(a) = (\text{ONE}(z) \wedge \text{FROM}_{\beta}(z)) \vee \text{TWO}(z) \vee \text{FOUR}(z)$$

It is straightforward to show that the previous formulas can be written in polynomial time. Thus, using Lemma 7.2, we have that the ISO problem for formula games in general form is equivalent to the the ISO problem for binary formula games.

The last step is to show that, given a binary formula game Γ , the boolean circuit C_{Γ} as defined in Page 90 can be described by a formula. From the definition of C_{Γ} , it follows trivially that C_k ($1 \leq k \leq n + 2$) can be described by formulas as the utility for the player is given by a formula. The following formulas are considered:

$$\text{ONLY}_i(c_1, \dots, c_{n+2}) = \neg c_1 \wedge \dots \wedge \neg c_{i-1} \wedge c_i \wedge \neg c_{i+1} \wedge \dots \wedge \neg c_{n+2}, \quad 1 \leq i \leq n+2$$

$$\text{EXONE}(c_1, \dots, c_{n+2}) = c_1 \vee \dots \vee c_{n+2}$$

$$\text{MOREONE}(c_1, \dots, c_{n+2}) = \bigvee_{1 \leq i < j \leq n+2} (c_i \wedge c_j)$$

Then, C_{Γ} can be expressed as a disjunctions of the three cases. When $\neg \text{EXONE}(c)$ or $\text{MOREONE}(c)$ holds, the result is 0. Otherwise the value is computed by a disjunction of terms $\text{ONLY}_j(c) \wedge C_j(a)$. Therefore C_{Γ} is expressed as,

$$\bigvee_{j=1}^{n+2} \text{ONLY}_j(c) \wedge C_j(a).$$

It is straightforward to show that a description of the previous circuit can be computed in polynomial time. Thus, using Lemma 7.3, it results that the ISO problem for formula games in general form is equivalent to the FORMULAISO problem. □

Proving NP-completeness in the case of explicit form appears to be a difficult task. A game in explicit form can be seen as a graph with edge labels and weights. As the total number of different weights appearing in both games is polynomial, the problem can be reduced to the Graph isomorphism (GI) problem [110]. Therefore, the NP-hardness of ISO will imply the NP-hardness of GI. The opposite direction is needed to prove. Thus, the following discussion is started by constructing a game from a graph.

Given an undirected graph G , a strategic game $\Gamma(G)$ associated to this graph is defined.

Game $\Gamma(G)$. It is assumed that $G = (V, E)$ is a non-directed graph with $V = \{1, \dots, n\}$ and m edges. Given $e \in E$, it is written $e = \{i, j\}$ to denote an edge connecting i and j . The game has 4 players with $A_1 = A_2 = \{0, 1, \dots, n\}$, $A_3 = \{0, 1\}$, $A_4 = E \cup \{0\}$. Let $A = A_1 \times A_2 \times A_3 \times A_4$ and $(i, j, k, l) \in A$. The utilities are:

$$u_1(i, j, k, l) = u_2(i, j, k, l) = \begin{cases} 1 & \text{if } l = \{i, j\} \text{ and } k = 0, \\ 0 & \text{otherwise} \end{cases}$$

$$u_3(i, j, k, l) = \begin{cases} 1 & \text{if } i = 0, j \neq 0, k = 1 \text{ and } l \neq 0, \\ 0 & \text{otherwise} \end{cases}$$

$$u_4(i, j, k, l) = \begin{cases} 1 & \text{if } i = j = k = 0, \\ 0 & \text{otherwise} \end{cases}$$

Note that the graph isomorphism problem is equivalent to the problem restricted to connected graphs G and G' that have $n > 2$ vertices. Otherwise, new vertices connected to all other vertices are added, one after the other, in the graph (in both graphs) until the condition is fulfilled. This type of vertex addition preserves isomorphism. With this construction it is possible to assume that, any vertex has at least one outgoing edge.

Lemma 7.4 *Let G, G' be two connected undirected graphs, with at least two vertices. The games $\Gamma(G)$ and $\Gamma(G')$ are strongly isomorphic iff G and G' are isomorphic.*

Proof. An isomorphism p between graphs G and G' is assumed. Let $\psi = (\pi, \varphi_1, \dots, \varphi_4)$ be a game mapping defined as follows, π and φ_3 being the identity, $\varphi_1 = \varphi_2 = p$, $\varphi_4(0) = 0$ and, for any edge $\{u, v\} \in E(G)$, $\varphi_4(\{u, v\}) = \{p(u), p(v)\}$. It is straightforward to show that ψ is a strong isomorphism between $\Gamma(G)$ and $\Gamma(G')$.

Let $\psi = (\pi, \varphi_1, \dots, \varphi_4)$ be a strong isomorphism between $\Gamma(G)$ and $\Gamma(G')$. This verifies the following:

The player's permutation verifies $\pi : \{1, 2\} \rightarrow \{1, 2\}$, $\pi : \{3\} \rightarrow \{3\}$ and finally $\pi : \{4\} \rightarrow \{4\}$. Denoting the cardinality of a set by $\#$, there are $\#\{a | u_1(a) = 1\} = \#\{a | u_2(a) = 1\} = 2m$, $\#\{a | u_3(a) = 1\} = mn$ and $\#\{a | u_4(a) = 1\} = m + 1$. When $n > 2$ and $m > 2$, these sets have different cardinality. As $\#\{a | u_i(a) = 1\} = \#\{\psi(a) | u'_{\pi(i)}(\psi(a)) = 1\}$, the result is obtained.

As players 1 and 2 have the same behaviour, $\pi(1) = 1$ and $\pi(2) = 2$ are assumed, therefore π is the identity.

The action's bijection φ_3 is the identity. As π is the identity, the following bijection $\psi(A_1 \times A_2 \times \{0\} \times A_4) = A'_1 \times A'_2 \times \{1\} \times A'_4$ holds. Therefore, for any i', j', l' $u'_3(i', j', 1, l') = 0$ holds, because $u_3(\varphi_1^{-1}(i'), \varphi_2^{-1}(j'), 0, \varphi_4^{-1}(l')) = 0$. This is a contradiction because $u'_3(i', 0, 1, e') = 1$ when $1 \leq i \leq n$ and $e' \in E'$.

The action's bijections for players 1 and 2 verify $\varphi_1(0) = 0$ and $\varphi_2(0) = 0$. This is forced by the rigid the structure of u_4 . As $u_4(0, 0, 0, l) = 1$, then $u'_4(\varphi_1(0), \varphi_2(0), 0, \varphi_1(l)) = 1$ and this force $\varphi_1(0) = \varphi_2(0) = 0$. Therefore, φ_1 and φ_2 are permutations on vertices $\{1, \dots, n\}$.

The action's bijection φ_4 verifies $\varphi_4(0) = 0$. When this does not hold, as φ_4 is a bijection, there exists an e such that $\varphi_4(e) = 0$. For $j \neq 0$ it holds that $u_3(0, j, 1, e) = 1$ and, as ψ is a morphism, $u'_3(0, \varphi_2(j), 1, 0) = 1$, but this is a contradiction. Therefore, φ_4 is a permutation on the m edges.

When $\varphi_4(e) = e'$ and $e = \{i, j\}$, it holds that $e' = \{\varphi_1(i), \varphi_2(j)\}$. As $u_1(i, j, 0, \{i, j\}) = 1$ and ψ is a morphism, $u'_1(\varphi_1(i), \varphi_2(j), 0, \varphi_4(\{i, j\})) = 1$, and $\varphi_4(\{i, j\}) = \{\varphi_1(i), \varphi_2(j)\}$.

φ_1 and φ_2 are the same permutations on $\{1, \dots, n\}$. It is needed to be proven that, for all $i \in \{1, \dots, n\}$ it holds that $\varphi_1(i) = \varphi_2(i)$. Let i be a vertex, as every node has a positive grade, there exists a j such that $e = \{i, j\}$ is an edge in G . As $u_1(i, j, 0, e) = u_1(j, i, 0, e) = 1$, $\varphi_4(e) = \{\varphi_1(i), \varphi_2(j)\} = \{\varphi_1(j), \varphi_2(i)\}$ holds. There are two possibilities, $\varphi_1(i) = \varphi_2(i)$ and $\varphi_1(j) = \varphi_2(j)$ or $\varphi_1(i) = \varphi_1(j)$ and $\varphi_2(j) = \varphi_2(i)$. But $\varphi_1(i) = \varphi_1(j)$ is impossible because φ_1 is a permutation.

When ψ is an isomorphism, the mapping $\varphi_1 : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ induces a graph isomorphism. Edge $e = \{i, j\}$ in G is considered, as ψ is a game morphism $u_1(i, j, 0, e) = u'_1(\varphi_1(i), \varphi_1(j), 0, \varphi_4(e))$ and this forces $\varphi_4(e) = \{\varphi_1(i), \varphi_1(j)\}$.

□

As a consequence of previous results the following is obtained.

Theorem 7.6 *The strong isomorphism problem for games given in explicit form is equivalent to the graph isomorphism problem.*

7.3 Weak Isomorphisms

Replacing strong by weak isomorphisms does not modify complexity bounds. In this section is shown that, for the case of a weak isomorphism, the ISISO problem is coNP-complete, and the ISO problem is equivalent to the ISO problem for strong isomorphisms. The last equivalence will hold for any of the considered representations of the games.

Theorem 7.7 *The ISISO problem for a weak isomorphism is coNP-complete, for games given in general form (strategic, formula and WCFG), and it belongs to NC when the games are given in explicit form. The ISO problem belongs to Σ_2^P , when the games are given in general form (strategic, formula and WCFG), and it belongs to NP when the games are given in explicit form.*

Proof. The proofs given in Theorems 7.2 and 7.3 have adapted. Membership in coNP of the ISISO problem for weak isomorphism and for games given in explicit or general form follows from the definitions.

When the games and the morphism are given in explicit form, a direct adaptation of the proof given in Theorem 7.2 gives us that ISISO belongs to NC for weak isomorphisms.

To prove hardness, given a boolean formula F with n variables, a variation of the game $\text{WINWHENTRUE}(F)$ is defined that is, $\text{WINWHENTRUEW}(F)$ in which utilities are redefined as follows:

$$u_i(a_1, \dots, a_n) = \begin{cases} 2^n & \text{if } F(a_1, \dots, a_n) \text{ is true,} \\ \sum_{i=1}^n a_i 2^{n-i} & \text{if } F(a_1, \dots, a_n) \text{ is false.} \end{cases}$$

For any pair of strategy profiles $a \neq a'$, $a \sim_i a'$ holds when both $F(a) = F(a') = 1$. When $F(a) = F(a') = 0$, $a \prec_i a'$ if and only if $a < a'$ in lexicographic order. When $F(a) \neq F(a')$ player i prefers the satisfying assignment. On the other hand, the ALWAYSWIN game in which $a \sim_i a'$ always holds is as well considered.

The IDENTITY morphism is a weak isomorphism between ALWAYSWIN and $\text{WINWHENTRUEW}(F)$ games iff F is valid. Thus, the ISISO problem for weak isomorphisms and games in general form is coNP-complete.

Finally, it is observed that a description in general form of the $\text{WINWHENTRUEW}(F)$ and the ALWAYSWIN games can be computed in polynomial time, when the utility functions are described by Turing machines or by formulas. □

When considering a weak isomorphism, it is shown first that the ISO problem is equivalent to the boolean circuit isomorphisms, second that the formula games in general form are equivalent to the boolean formula isomorphism, and third that the strategic games in explicit form are equivalent to the graph isomorphism problem. Before proving these results, a series of game transformations that preserve weak isomorphism is provided, establishing equivalence with the strong isomorphism. Later on will, it will be shown that these transformations are indeed reduction in polynomial time for the considered representations.

The $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ game is assumed. It is a binary game, where $N = \{1, \dots, n\}$. The following game is considered.

$\text{CHECKW}(\Gamma) = (N', (A'_i)_{i \in N'}, (u'_i)_{i \in N'})$, where $N' = \{1, \dots, n, n+1\}$ and, for any $1 \leq i \leq n$, $A'_i = \{0, 1\}$ and $A'_{n+1} = \{0, 1, 2, 3\}$. The utilities are defined as follows, for a player i , $1 \leq i \leq n$,

$$u'_i(a') = \begin{cases} 1 & \text{if } u_i(a'_1, \dots, a'_n) = (a'_{n+1} \bmod 2), \\ 0 & \text{otherwise,} \end{cases}$$

For the last player,

$$u'_{n+1}(a') = a'_{n+1}.$$

The equality $(u_i(a'_1, \dots, a'_n) = (a'_{n+1} \bmod 2))$ is taken as a boolean expression with values $\{0, 1\}$. Under this point of view it can be written shortly, $u'_i(a') = (u_i(a'_1, \dots, a'_n) = (a'_{n+1} \bmod 2))$.

Note that Γ is a binary game (both, actions and utilities are binary). $\text{CHECKW}(\Gamma)$ is not a binary game, neither a binary action game due to the last player. Player $n + 1$ in $\text{CHECKW}(\Gamma)$ has four actions and u_{n+1} takes four values.

Lemma 7.5 *Let Γ_1 and Γ_2 be two binary games. Γ_1 and Γ_2 are strongly isomorphic iff the games $\text{CHECKW}(\Gamma_1)$ and $\text{CHECKW}(\Gamma_2)$ are weakly isomorphic.*

Proof. Let $\Gamma'_1 = \text{CHECKW}(\Gamma_1)$ and $\Gamma'_2 = \text{CHECKW}(\Gamma_2)$. A strong isomorphism $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ between Γ_1 and Γ_2 is assumed. The mapping $\psi' = (p, f_1, \dots, f_{n+1})$ is defined for $1 \leq i \leq n$, $p(i) = \pi(i)$ and $f_i = \varphi_i$, $p(n + 1) = n + 1$, and f_{n+1} is the identity function. Let us prove that ψ' is a strong (therefore also a weak) isomorphism between Γ'_1 and Γ'_2 .

Let $a' = (a_1, \dots, a_n, a_{n+1})$ be a strategy profile in $\text{CHECKW}(\Gamma_1)$. It is written $a' = (a, a_{n+1})$ with $a = (a_1, \dots, a_n)$. By definition of ψ' it holds that $\psi'(a') = (\psi(a), a_{n+1})$, because $f_{n+1}(a_{n+1}) = a_{n+1}$. Let us prove that ψ' is a strong isomorphism. For $1 \leq i \leq n$, $u_{\pi(i)}(\psi(a)) = u_i(a)$, since ψ is a strong isomorphism, and therefore,

$$u'_{p(i)}(\psi'(a')) = (u_{\pi(i)}(\psi(a)) = (a'_{n+1} \bmod 2)) = (u_i(a) = (a'_{n+1} \bmod 2)) = u'_i(a').$$

It remains the case $n + 1$. As $p(n + 1) = n + 1$ and f_{n+1} is the identity, $u'_{n+1}(\psi'(a)) = a'_{n+1} = u_{n+1}(a)$.

Now, $\psi' = (p, f_1, \dots, f_{n+1})$ is assumed a weak isomorphism between Γ'_1 and Γ'_2 . As p maps between players having the same number of actions, and the only player with 4 actions is the last one we are forced to have $p(n + 1) = n + 1$. Let ψ be ψ' restricted to players $1, \dots, n$, that is $\psi = (\pi, \varphi_1, \dots, \varphi_n)$, such that for $1 \leq i \leq n$, $\pi(i) = p(i)$ and $\varphi_i = f_i$. For any $a' = (a, a'_{n+1})$ with $a = (a_1, \dots, a_n)$, we have that $\psi'(a') = (\psi(a), f_{n+1}(a'_{n+1}))$. The definition of the preference relation of player $n + 1$ forces $f_{n+1} = Id$ (see Claim 7.1). Let ψ be ψ' restricted to players $1, \dots, n$, that is $\psi = (\pi, \varphi_1, \dots, \varphi_n)$. Then, $\pi(i) = p(i)$ and $\varphi_i = f_i$, for $1 \leq i \leq n$ and $\psi'(a') = (\psi(a), f_{n+1}(a'_{n+1}))$. In $\text{CHECKW}(\Gamma_1)$ player $n + 1$ has the following chain of strict preferences,

$$(a, 0) \prec_{n+1} (a, 1) \prec_{n+1} (a, 2) \prec_{n+1} (a, 3)$$

As ψ' is a weak morphism, preferences of player $n + 1$ in $\text{CHECKW}(\Gamma_2)$ verify that,

$$(\psi(a), f_{n+1}(0)) \prec_{n+1} (\psi(a), f_{n+1}(1)) \prec_{n+1} (\psi(a), f_{n+1}(2)) \prec_{n+1} (\psi(a), f_{n+1}(3))$$

This forces $u'_{n+1}(f_{n+1}(0)) < u'_{n+1}(f_{n+1}(1)) < u'_{n+1}(f_{n+1}(2)) < u'_{n+1}(f_{n+1}(3))$. Therefore,

$$u'_{n+1}(f_{n+1}(0)) = 0, u'_{n+1}(f_{n+1}(1)) = 1, u'_{n+1}(f_{n+1}(2)) = 2, u'_{n+1}(f_{n+1}(3)) = 3$$

The only possibility to fulfill the preceding equalities is to take $f_{n+1}(a'_{n+1}) = a'_{n+1}$ for $a'_{n+1} \in \{0, 1, 2, 3\}$.

After that, it has a factorisation $\psi'(a') = (\psi(a), a'_{n+1})$. Note that for any $a' = (a, a_{n+1})$ it holds that, $u'_{n+1}(\psi'(a')) = u'_{n+1}(\psi(a), a_{n+1}) = a_{n+1} = u'_{n+1}(a')$. Given $a = (a'_1, \dots, a'_n)$, since Γ_1 is a binary game, $u_i(a) \in \{0, 1\}$ holds. It is defined $\bar{u}_i(a) = 1 - u_i(a_i)$ and in this case, $u_i(a) = (u_i(a) \bmod 2)$ and $\bar{u}_i(a) = (\bar{u}_i(a) \bmod 2)$. In $\text{CHECKW}(\Gamma_1)$, given $a = (a'_1, \dots, a'_n)$ for any player $1 \leq i \leq n$ it holds that,

$$\begin{aligned} u'_i(a, \bar{u}_i(a)) &= (u_i(a) = (\bar{u}_i(a) \bmod 2)) = (u_i(a) = \bar{u}_i(a)) = 0 \\ u'_i(a, u_i(a)) &= (u_i(a) = (u_i(a) \bmod 2)) = (u_i(a) = u_i(a)) = 1 \end{aligned}$$

Therefore, $(a, \bar{u}_i(a)) \prec_i (a, u_i(a))$. As ψ' is a weak isomorphism, $\psi'(a, \bar{u}_i(a)) \prec_{\pi(i)} \psi'(a, u_i(a))$, since f_{n+1} is the identity $(\psi(a), \bar{u}_i(a)) \prec_{\pi(i)} (\psi(a), u_i(a))$. This forces $u'_{\pi(i)}(\psi(a), \bar{u}_i(a)) < u'_{\pi(i)}(\psi(a), u_i(a))$ and consequently $u'_{\pi(i)}(\psi(a), \bar{u}_i(a)) = 0$ and $u'_{\pi(i)}(\psi(a), u_i(a)) = 1$. According to the definition of $u'_{\pi(i)}$, it holds,

$$u'_{\pi(i)}(\psi(a), u_i(a)) = (u_{\pi(i)}(\psi(a)) = (u_i(a) \bmod 2)) = (u_{\pi(i)}(\psi(a)) = u_i(a)) = 1.$$

Therefore, $u_{\pi(i)}(\psi(a)) = u_i(a)$ for any $1 \leq i \leq n$, and ψ is a strong isomorphism. \square

Claim 7.1 Let $\psi' = (p, f_1, \dots, f_{n+1})$ be a weak isomorphism $\psi' : \text{CHECKW}(\Gamma_1) \rightarrow \text{CHECKW}(\Gamma_2)$ then, $f_{n+1} = \text{Id}$ holds.

Proof. Let ψ be a ψ' mapping restricted to players $1, \dots, n$, that is $\psi = (\pi, \varphi_1, \dots, \varphi_n)$, as in the proof of Lemma 7.5. Then, $\pi(i) = p(i)$, $\varphi_i = f_i$ for $1 \leq i \leq n$ and $\psi'(a') = (\psi(a), f_{n+1}(a'_{n+1}))$. In $\text{CHECKW}(\Gamma_1)$, player $n + 1$ has the following chain of strict preferences: $(a, 0) \prec_{n+1} (a, 1) \prec_{n+1} (a, 2) \prec_{n+1} (a, 3)$. Since ψ' is a weak morphism, preferences of player $n + 1$ in $\text{CHECKW}(\Gamma_2)$ verify,

$$(\psi(a), f_{n+1}(0)) \prec_{n+1} (\psi(a), f_{n+1}(1)) \prec_{n+1} (\psi(a), f_{n+1}(2)) \prec_{n+1} (\psi(a), f_{n+1}(3))$$

This forces, $u'_{n+1}(f_{n+1}(0)) < u'_{n+1}(f_{n+1}(1)) < u'_{n+1}(f_{n+1}(2)) < u'_{n+1}(f_{n+1}(3))$ and therefore, $u'_{n+1}(f_{n+1}(0)) = 0$, $u'_{n+1}(f_{n+1}(1)) = 1$, $u'_{n+1}(f_{n+1}(2)) = 2$, $u'_{n+1}(f_{n+1}(3)) = 3$. The only possibility to fulfil the preceding equalities is to take, $f_{n+1}(a'_{n+1}) = a'_{n+1}$ for $a'_{n+1} \in \{0, 1, 2, 3\}$.

□

As it has been explained in the previous section, a transformation to a binary action game is defined. The construction of the game follows the same lines as in the $\text{BINARYACT}(\Gamma)$ (see Page 81). But now, an adequate preference relation has been guaranteed for each player. The game $\Gamma = (N, A_1, \dots, A_n, (u_i)_{1 \leq i \leq n})$, it is assumed. In addition, without loss of generality, it is assumed that utilities are non-negative. If this happens, it is sufficient to add a “big positive number”. When utilities are computed by a TM with time t , 2^t can be added.

$\text{BINARYACTW}(\Gamma) = (N', (A'_i)_{i \in N'}, (u'_i)_{i \in N'})$, where $N' = \{1, \dots, k\}$ and, for any $i \in N'$, $A'_i = \{0, 1\}$ and thus the set of action profiles is $A' = \{0, 1\}^k$. A block B_i is associated to A_i of $k_i = |A_i|$ players, each one taking care of one bit. Thus, $k = k_1 + \dots + k_n$. A strategy profile a' is split into n blocks. Thus, $a' = (b_1, \dots, b_n)$ where $b_i \in \{0, 1\}^{k_i}$. a'_j has been kept to refer to the strategy of player j . Recall that, if $A^i = \{0, 1\}^{k_i}$, $\text{good}(A^i) = \{\text{binify}(a) \mid a \in A_i\}$, where $\text{binify}(j) = 0^{j-1}10^{k_i-j}$, $\text{good}(A') = \{\text{binify}(a_1) \cdots \text{binify}(a_n) \mid a_1 \in A_1, \dots, a_n \in A_n\}$ and that, for $a' \in \text{good}(A')$, $\text{binify}^{-1}(a') = (\text{binify}^{-1}(b_1), \dots, \text{binify}^{-1}(b_n))$. For a player α occupying position j in block B_i , the player partitions A' in the following sets are:

$$\begin{aligned} X_0(\alpha) &= \{a' \mid b_i \notin \text{good}(A^i)\} \\ X_1(\alpha) &= \{a' \mid b_{ij} = 0 \text{ and } b_i \in \text{good}(A^i) \text{ and } a' \in \text{bad}(A')\} \\ X_2(\alpha) &= \{a' \mid b_{ij} = 1 \text{ and } b_i \in \text{good}(A^i) \text{ and } a' \in \text{bad}(A')\} \\ X_3(\alpha) &= \text{good}(A') \end{aligned}$$

and the utility function is defined as

$$u'_\alpha(a') = \begin{cases} 0 & \text{if } a' \in X_0(\alpha), \\ 1 & \text{if } a' \in X_1(\alpha), \\ 2 & \text{if } a' \in X_2(\alpha), \\ 3 + u_i(\text{binify}^{-1}(a')) & \text{if } a' \in X_3(\alpha), \end{cases}$$

When $|A_i| = k_i = 1$, we have $A^i = \{0, 1\}$, $\text{binify}(1) = 1$ and B_i has just one player. Let α be such a player, in this case, $X_1(\alpha) = \emptyset$. When $k_i > 1$, all sets $X_0(\alpha), \dots, X_3(\alpha)$ are non empty.

Player α prefers profiles in $X_3(\alpha)$ to profiles in $X_2(\alpha)$, profiles in $X_2(\alpha)$ to profiles in $X_1(\alpha)$, and profiles in $X_1(\alpha)$ to profiles in $X_0(\alpha)$. Moreover, player α is indifferent among two profiles belonging to the same set $X_0(\alpha), X_1(\alpha)$, or $X_2(\alpha)$. For profiles a'_1 and a'_2 , both in $X_3(\alpha)$, player α keeps the preferences of player i in Γ among the profiles $\text{binify}^{-1}(a'_1)$ and $\text{binify}^{-1}(a'_2)$.

Lemma 7.6 *Let Γ_1 and Γ_2 be two strategic games. Γ_1 and Γ_2 are weakly isomorphic iff $\text{BINARYACTW}(\Gamma_1)$ and $\text{BINARYACTW}(\Gamma_2)$ are weakly isomorphic.*

Proof. Let $\Gamma'_1 = \text{BINARYACTW}(\Gamma_1)$ and $\Gamma'_2 = \text{BINARYACTW}(\Gamma_2)$.

It is assumed that $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ is a weak isomorphism between Γ_1 and Γ_2 . The mapping $\psi' = (p, f_1, \dots, f_k)$ is considered where, for $1 \leq i \leq n$, p maps the bits in block i of Γ'_1 to the bits in block $\pi(i)$ of Γ'_2 so that the j -th bit of B_i goes to bit $\varphi_i(j)$ of $B_{p(i)}$, and, for $1 \leq j \leq k$, f_j is the identity function. It is straightforward to show that ψ' is a weak isomorphism between Γ'_1 and Γ'_2 .

The reverse part is considered. It is assumed that $\psi' = (p, f_1, \dots, f_k)$ is a weak isomorphism between Γ'_1 and Γ'_2 . As it will be proven later on, all the f_α , $1 \leq \alpha \leq k$ are identities and p induces a permutation into the blocks.

If $\psi' = (p, f_1, \dots, f_k)$ is a weak isomorphism between the games, $\Gamma'_1 = \text{BINARYACTW}(\Gamma_1)$ and $\Gamma'_2 = \text{BINARYACTW}(\Gamma_2)$. The proof is stated as a series of claims.

Given players $\alpha \in B_i$ and $\alpha' \in B_j$ with $i \neq j$, it holds $X_0(\alpha) \neq X_0(\alpha')$. When $k_i \neq k_j$, the proof is direct, because $X_0(\alpha)$ has a cardinality $2^{k-k_i}(2^{k_i} - k_i)$, which is different from the cardinality of $X_0(\alpha')$. The case $k_i = k_j$ is considered. It is assumed that block i precedes block j , and the profile schematised is considered as follows,

$$a = (b_1, \dots, b_{i-1}, \text{bad}_i, b_{i+1}, \dots, b_{j-1}, \text{good}_j, b_{j+1}, \dots, b_n)$$

where bad_i is a bad profile in A^i and good_j is a good profile in A^j . It holds that $a \in X_0(\alpha)$ but $a \notin X_0(\alpha')$.

Given the permutation p and a player α , sets $X_0(\alpha)$ and $X_0(p(\alpha))$ are both non-empty. Player α occupies forcedly a position into a block. It is supposed that B_i is such a block. As $\text{good}(A^i)$ has $k_i \geq 1$ elements, the set $A^i \setminus \text{good}(A^i)$ contains $2^{k_i} - k_i > 0$ elements. By the same reason $X_0(p(\alpha))$ is not empty.

It holds that $\psi'(X_0(\alpha)) = X_0(p(\alpha))$ for any player α . First note that $\psi'(X_0(\alpha)) \subseteq X_0(p(\alpha))$. Otherwise, there is $a' \in \psi'(X_0(\alpha)) \setminus X_0(p(\alpha))$ and $a'' \in X_0(p(\alpha))$ (because $X_0(p(\alpha))$ is not-empty), such that $a'' \prec_{p(\alpha)} a'$. Then $\psi'^{-1}(a'') \prec_\alpha \psi'^{-1}(a')$, but

this is impossible because $\psi'^{-1}(a') \in X_0(\alpha)$ and therefore, $\psi'^{-1}(a')$ is a less preferred element. It is supposed that $\psi'(X_0(\alpha)) \neq X_0(p(\alpha))$. Let $a' \in X_0(p(\alpha)) \setminus \psi'(X_0(\alpha))$ and $\psi'^{-1}(a')$ is considered. If $\psi'^{-1}(a')$ belongs to $X_0(\alpha)$, a contradiction is obtained. If $\psi'^{-1}(a') \notin X_0(\alpha)$ is assumed, there exists an $b \in X_0(\alpha)$ such that, $b \prec_\alpha \psi'^{-1}(a')$. Then, $\psi(b) \prec_{p(\alpha)} a'$, but this is impossible because a' is a less preferred element.

It holds that $p(B_{\text{block}(\alpha)}) = B_{\text{block}(p(\alpha))}$ for all α . Let α and α' be players in block B_i , that is $\text{block}(\alpha) = \text{block}(\alpha') = i$. As $X_0(\alpha) = X_0(\alpha')$ it holds that $\psi(X_0(\alpha)) = X_0(p(\alpha)) = X_0(p(\alpha'))$. Both $X_0(p(\alpha)) = X_0(p(\alpha'))$ iff $\text{block}(p(\alpha)) = \text{block}(p(\alpha'))$.

Thus, ψ' induces a permutation π on $\{1, \dots, n\}$ such that $\pi(B_i) = B_{\pi(i)}$ moreover, $\pi(\text{block}(\alpha)) = \text{block}(p(\alpha))$. For a player α in position j inside block B_i , let $\varphi(j)$ be the position of player $p(\alpha)$ in block $\pi(i)$. Therefore, a mapping $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ is defined.

It holds that $\psi'(X_1(\alpha)) = X_1(p(\alpha))$ for any player α . There are two cases depending on the values of the k_i corresponding to the block containing α . First, the case $k_i = 1$ is considered. In this case, $B_i = \{0, 1\}$ and $X_i(\alpha) = \emptyset$. One has $p(B_i) = B_{\pi(i)} = \{0, 1\}$ and $X_1(p(\alpha)) = \emptyset$. The case $k_i > 1$ is considered. As ψ' is a bijection between strategy profiles and, there is a bijection between $X_0(\alpha)$ and $X_0(p(\alpha))$ we have $\psi(X_1(\alpha)) \subseteq X_1(p(\alpha)) \cup X_2(p(\alpha)) \cup X_3(p(\alpha))$. If there exists an $a' \in X_1(\alpha)$ such that $\psi'(a') \in X_2(p(\alpha)) \cup X_3(p(\alpha))$, there exists a $b \in X_1(p(\alpha))$ such that $b \prec_{p(\alpha)} \psi'(a')$. Therefore, $\psi'^{-1}(b) \prec_\alpha a'$, but this is impossible because $\psi'^{-1}(b)$ cannot be an element of $X_0(\alpha)$. Therefore, $\psi(X_1(\alpha)) \subseteq X_1(p(\alpha))$. Since $\psi(X_1(\alpha))$ and $X_1(p(\alpha))$ have the same number of elements, it is concluded that $\psi(X_1(\alpha)) = X_1(p(\alpha))$.

It holds that $\psi'(X_2(\alpha)) = X_2(p(\alpha))$ for any player α . One has $\psi'(X_2(\alpha)) \subseteq X_2(p(\alpha)) \cup X_2(p(\alpha))$ and by similar argumentation the equality is concluded.

It holds that $\psi'(X_3(\alpha)) = X_3(p(\alpha))$ for any player α . As ψ' is a bijection and $\psi'(X_2(\alpha)) \subseteq X_2(p(\alpha))$, this forces equality.

It holds that, f_α is the identity for any player α . Note that α belongs to a block of players B_i , $i = \text{block}(\alpha)$ having A^i as the corresponding alphabet. Two cases depending on the size of A^i are considered. First, it is considered the case such that A^i has just one element. In this case $\text{good}(A^i) = \{1\}$. As $\psi(\text{good}(A^i)) = \text{good}(A^{\pi(i)}) = \{1\}$ this forces f_α to be the identity. It is considered the case where A^i contains more than one element. It is supposed that α occupies the position j in B_i , and the profile $a' = (b_{-i}, 0^{j-1} 10^{k_i-j})$ is considered, belonging to $X_2(\alpha)$, as $\psi(a')$ belongs to $X_2(p(\alpha))$. A factorization $\psi(a') = (\psi(b_{-i}), 0^{\varphi_i(j)-1} 10^{k_i-\varphi_i(j)})$ is needed, and this forces to f_α to be the identity.

Given a strategy profile a in Γ_1 , it holds that $\psi'(\text{binify}(a)) = \text{binify}(\psi(a))$. Note that $A_i = \{1, \dots, k_i\}$ and for $j \in A_i$ we have $\text{binify}(j) = 0^{j-1} 10^{k_i-j} \in A^i$. As $p(B_i) = B_{\pi(i)}$, we have $\psi(\text{binify}(j)) = 0^{\varphi_i(j)-1} 10^{k_i-\varphi_i(j)} \in A^{\pi(i)}$ and the result is concluded.

Given two profiles a, a' and a player i in Γ_1 and a player α in Γ'_1 such that, $\text{block}(\alpha) = i$, it holds that $a \prec_i a'$ iff $\text{binary}(a) \prec_\alpha \text{binary}(a')$. This happens because there are equalities

like $u_\alpha(\text{binary}(a)) = u_i(a) + 3$. The same property holds for Γ_2 and Γ'_2 .

The mapping $\Psi = (\pi, \varphi_1, \dots, \varphi_n)$ is a weak morphism between Γ_1 and Γ_2 . It is supposed that in Γ_1 there are $a \prec_i a'$. Let in Γ'_1 a player α such that $\text{block}(\alpha) = i$, then it holds that $\text{binify}(a) \prec_\alpha \text{binify}(a')$. As Ψ' is a weak morphism $\Psi'(\text{binify}(a)) \prec_{p(\alpha)} \Psi'(\text{binify}(a'))$ and therefore, changing Ψ' into Ψ , $\text{binify}(\Psi(a)) \prec_{p(\alpha)} \text{binify}(\Psi(a'))$. Then it holds that $\Psi(a) \prec_{\text{block}(\pi(\alpha))} \Psi(a')$, as $\text{block}(\pi(\alpha)) = \pi(\text{block}(\alpha)) = \pi(i)$. Finally, $\Psi(a) \prec_{\pi(i)} \Psi(a')$ is obtained.

Therefore, a player permutation π on $\{1, \dots, n\}$ is considered. Let $\varphi(j)$ be the position of player $p(\beta)$ in block $\pi(i)$, for a player α in position j inside block B_i . It holds that $(\pi, \varphi_1, \dots, \varphi_n)$ is a weak isomorphism between Γ_1 and Γ_2 . □

The next step is to transform weakly isomorphic games into strongly isomorphic games. The transformation consists in coding precedence relations into utilities. Given a binary actions game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$, where $N = \{1, \dots, n\}$ and $A_i = \{0, 1\}$, the following game is considered:

$\text{FLIPW}(\Gamma) = (N, (A'_i)_{i \in N}, (u'_i)_{i \in N})$, where $A'_i = \{0, 1\}^2$. Let $a' = (a_1 b_1, \dots, a_n b_n)$ be a strategy profile in game $\text{FLIPW}(\Gamma)$, and $\text{driver}(a') = (a_1, \dots, a_n) = a$ and $\text{flipper}(a') = (b_1, \dots, b_n) = b$ are defined. Note shortly $a' = a \uparrow b$. For $xy \in \{0, 1\}^2$ is defined,

$$\text{flip}(xy) = \begin{cases} x & \text{if } y = 0, \\ \bar{x} & \text{if } y = 1. \end{cases}$$

Let $a' = (a_1 b_1, \dots, a_n b_n)$ be a strategy profile in game $\text{FLIPW}(\Gamma)$, the following are defined:

$$\text{flip}(a') = (\text{flip}(a_1 b_1), \dots, \text{flip}(a_n b_n)).$$

Note that, $\text{flip}(a')$ is a strategy profile in game Γ . Given a strategy profile $a' = (a_1 b_1, \dots, a_n b_n)$, for any player i , $1 \leq i \leq n$, the following are defined,

$$u'_i(a') = \begin{cases} 5 & \text{if } u_i(\text{flip}(a')) < u_i(\text{driver}(a')) \text{ and } b_i = 1 \\ 4 & \text{if } u_i(\text{flip}(a')) = u_i(\text{driver}(a')) \text{ and } b_i = 1 \\ 3 & \text{if } u_i(\text{flip}(a')) > u_i(\text{driver}(a')) \text{ and } b_i = 1 \\ 2 & \text{if } u_i(\text{flip}(a')) < u_i(\text{driver}(a')) \text{ and } b_i = 0 \\ 1 & \text{if } u_i(\text{flip}(a')) = u_i(\text{driver}(a')) \text{ and } b_i = 0 \\ 0 & \text{if } u_i(\text{flip}(a')) > u_i(\text{driver}(a')) \text{ and } b_i = 0 \end{cases}$$

Example 7.3 The Γ game is considered,

		<i>Player 2</i>	
		0	1
<i>Player 1</i>	0	0,0	1,0
	1	0,1	0,0
		Γ	

It holds that $(1,0) \prec_1 (0,1)$. It will be shown how this preference is coded as an utility in $\text{FLIPW}(\Gamma)$. To transform $a = (1,0)$ into $(0,1)$, both bits in $(1,0)$ have to be flipped, therefore the flipper is $b = (1,1)$ and the transformation in $\text{FLIPW}(\Gamma)$ with the strategy profile $a' = (11,01) = (1,0) \uparrow (1,1) = a \uparrow b$ is coded. There are $\text{driver}(a') = (1,0)$, $\text{flipper}(a') = (1,1)$ and $\text{flip}(a') = (0,1)$. To compute $u'_1(a')$, one has to look at the flipper of first player. Since $b_1 = 1$ and $u_1(1,0) < u_1(0,1)$, $u'_1(a') = 3$ is obtained. A case of indifference is considered, for instance $(0,0) \sim_2 (0,1)$. The driver is $a = (0,0)$, the flipper is $b = (0,1)$ and $a' = (00,01) = a \uparrow b$. Since $b_2 = 1$, $u'_2(a') = 4$ is obtained.

Lemma 7.7 Let Γ_1 and Γ_2 be two binary actions games. Γ_1 and Γ_2 are weakly isomorphic, if and only if the games, $\text{FLIPW}(\Gamma_1)$ and $\text{FLIPW}(\Gamma_2)$ are strongly isomorphic.

Proof. Let $\Gamma'_1 = \text{FLIPW}(\Gamma_1)$ and $\Gamma'_2 = \text{FLIPW}(\Gamma_2)$.

Let $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ be a mapping between two binary actions games Γ_1 and Γ_2 . Let $\psi' = (\pi, f_1, \dots, f_n)$ be a mapping between Γ'_1 and Γ'_2 , verifying $f_i(a_i b_i) = \varphi_i(a_i) b_i$ for $1 \leq i \leq n$. It is taken also $\mu = (\pi, id_1, \dots, id_n)$, for any $a' = a \uparrow b$, then $\psi'(a') = \psi(a) \uparrow \mu(b)$ holds.

Moreover, it is proven that $\text{flip}(\psi'(a')) = \psi(\text{flip}(a'))$, given $a' = (a_1 b_1, \dots, a_n b_n) = a \uparrow b$, with $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$. In all mappings ψ , μ and ψ' , the bijection function π maps player i into player $\pi(i)$. There is $\psi(a) = (\hat{a}_1, \dots, \hat{a}_n)$ with $\hat{a}_{\pi(i)} = \varphi(a_i)$, for all i . There is also $\mu(b) = (\hat{b}_1, \dots, \hat{b}_n)$ with $\hat{b}_{\pi(i)} = b_i$, for all i . Moreover, $\psi'(a') = (\hat{a}_1 \hat{b}_1, \dots, \hat{a}_n \hat{b}_n)$ with $\hat{a}_i \hat{b}_i = \varphi_i(a_i) b_i$, for all i . Therefore, $\psi'(a') = (\hat{a}_1, \dots, \hat{b}_n) \uparrow (\hat{b}_1, \dots, \hat{b}_n) = \psi(a) \uparrow \mu(b)$.

The behaviour of the flips is considered. Given $a' = a \uparrow b$, note that $\psi(\text{flip}(a')) = \text{flip}(\psi'(a'))$ is equivalent to $\psi(\text{flip}(a \uparrow b)) = \text{flip}(\psi(a) \uparrow \mu(b))$. The component wise this equality corresponds to $\varphi_i(\text{flip}(a_i b_i)) = \text{flip}(\varphi_i(a_i) b_i)$. Since flip , as $\text{flip}(xy) = x\bar{y} + \bar{x}y$ is a boolean function, the preceding equality is $\varphi_i(a_i \bar{b}_i + \bar{a}_i b_i) = \varphi_i(a_i) \bar{b}_i + \varphi_i(\bar{a}_i) b_i$. Since φ_i is a permutation in $\{0, 1\}$, the only possibilities are $\varphi(a_i) = a_i$ or $\varphi(a_i) = \bar{a}_i$. Equation trivially holds for identity. It is enough to check $\neg(a_i \bar{b}_i + \bar{a}_i b_i) = \bar{a}_i \bar{b}_i + a_i b_i$, when φ_i is a negation.

It is assumed that $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ is a weak isomorphism between games, Γ_1 and Γ_2 . Then, games Γ'_1 and Γ'_2 , and a morphism $\psi' = (\pi, f_1, \dots, f_n)$, where $f_i(a_i b_i) = \varphi_i(a_i) b_i$ for $1 \leq i \leq n$, are considered.

Now, a strong isomorphism ψ' between Γ'_1 and Γ'_2 , is considered. It will be proven later on that $u'_i(a') = u'_{\pi(i)}(\psi'(a'))$. Utilities have 6 different values. It is assumed that there exists an a' , such that $u_i(a') = 6$. This is equivalent to $u_i(\text{flip}(a')) < u_i(a)$, factorising $a' = a \uparrow b$. It holds that, $u_{\pi(i)}(\psi(\text{flip}(a'))) < u_{\pi(i)}(\psi(a))$, because ψ is a weak isomorphism. Therefore, $\psi(\text{flip}(a')) = \text{flip}(\psi'(a'))$ and $\psi'(a') = \psi(a) \uparrow \mu(b)$. Since $u_{\pi(i)}(\text{flip}(\psi'(a'))) < u_{\pi(i)}(\psi(a))$, $u'_{\pi(i)}(\psi'(a')) = 6$ is concluded. For other utility values, the proofs are similar.

If $\psi' = (\pi, f_1, \dots, f_n)$ is a strong isomorphism between Γ'_1 and Γ'_2 , the definition of utility functions of player i forces to $f_i(a_i b_i) = \varphi_i(a_i) b_i$, for a permutation φ_i in $\{0, 1\}$ and any action $a_i b_i$. A morphism $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ and a profile $a' = (a_1 b_1, \dots, a_n b_n)$ are considered. Observe that $\text{flip}(\psi'(a')) = \psi(\text{flip}(a'))$.

This fact is taken and also ψ' , which preserves utilities, then a weak isomorphism ψ between Γ_1 and Γ_2 can be shown. Sets $\text{Zero}(i) = \{a' | a' = (a'_{-i}, a_i 0)\}$ and $\text{One}(i) = \{a' | a' = (a'_{-i}, a_i 1)\}$ are considered. Note that, $a' \in \text{Zero}(i)$ iff $u'_i(a') \in \{0, 1, 2\}$ and $a' \in \text{One}(i)$ iff $u'_i(a') \in \{3, 4, 5\}$. Moreover, the following holds:

- For $n > 0$, every set $\text{Zero}(i)$ and $\text{One}(i)$ contains $2^{2n-1} > 0$ elements each one.
- $\text{Zero}(i) \cap \text{One}(i) = \emptyset$ and $\text{Zero}(i) \cup \text{One}(i) = A'_1$ are disjoint.
- It holds that, $\psi'(\text{Zero}(i)) = \text{Zero}(\pi(i))$. If this fact is false, $\psi'(\text{Zero}(i)) \cap \text{One}(\pi(i)) \neq \emptyset$. Therefore, there exists an $a' \in \text{Zero}(i)$ such that $u'_i(a') \in \{0, 1, 2\}$, but utilities $u'_{\pi(i)}(\psi'(a')) \in \{3, 4, 5\}$. This cannot be happen because ψ' is a strong isomorphism.
- Similarly $\psi'(\text{One}(i)) = \text{One}(\pi(i))$.
- Player's bijections verify $f_i(a_i b_i) = \varphi_i(a_i) b_i$, for a permutation φ_i on $\{0, 1\}$. This is just another way to write $\psi'(\text{Zero}(i)) = \text{Zero}(\pi(i))$ and $\psi'(\text{One}(i)) = \text{One}(\pi(i))$.

A weak isomorphism ψ will be proven. It is assumed that in Γ_1 , there exists an a and \hat{a} , such that $u_i(a) < u_i(\hat{a})$. It will be proven that $u_{\pi(i)}(\psi(a)) < u_{\pi(i)}(\psi(\hat{a}))$. The inequality $u_i(a) < u_i(\hat{a})$ is given. Let b be the flipper such that, $\text{flip}(a \uparrow b) = \hat{a}$. It is assumed that $b_i = 0$, and take care that case $b_i = 1$ is similar. Then, in $\text{FLIPW}(\Gamma_1)$, $u'_i(a \uparrow b) = 0$ holds. Since ψ' is a strong isomorphism, it holds that $u'_i(a \uparrow b) = u'_{\pi(i)}(\psi'(a \uparrow b)) = 0$. Therefore, $u_{\pi(i)}(\text{flip}(\psi'(a \uparrow b))) > u_{\pi(i)}(\text{driver}(\psi'(a \uparrow b)))$.

Since $f_i(a_i b_i) = \varphi_i(a_i) b_i$, $\text{flip}(\psi'(a \uparrow b)) = \psi(\text{flip}(a \uparrow b)) = \psi(\hat{a})$ holds, by the following Claim.

Claim 7.2 *Let $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ be a mapping between binary action games, Γ_1 and Γ_2 . Let $\psi' = (\pi, f_1, \dots, f_n)$ be a mapping between $\text{FLIPW}(\Gamma_1)$ and $\text{FLIPW}(\Gamma_2)$ such that, $f_i(a_i b_i) = \varphi_i(a_i) b_i$, for $1 \leq i \leq n$. A mapping $\mu = (\pi, id_1, \dots, id_n)$ is taken, for any $a' = a \uparrow b$. Then, $\psi'(a') = \psi(a) \uparrow \mu(b)$ holds. Moreover, $\text{flip}(\psi'(a')) = \psi(\text{flip}(a'))$.*

Proof. Given $a' = (a_1b_1, \dots, a_nb_n) = a \uparrow b$ with $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$. In all mappings ψ , μ and ψ' , the bijection function π maps player i into player $\pi(i)$. There is $\psi(a) = (\hat{a}_1, \dots, \hat{a}_n)$ with $\hat{a}_{\pi(i)} = \varphi(a_i)$ for all i . There is also $\mu(b) = (\hat{b}_1, \dots, \hat{b}_n)$, with $\hat{b}_{\pi(i)} = b_i$ for all i . Moreover, $\psi'(a') = (\hat{a}_1\hat{b}_1, \dots, \hat{a}_n\hat{b}_n)$ with $\hat{a}_i\hat{b}_i = \varphi_i(a_i)b_i$ for all i . Therefore, $\psi'(a') = (\hat{a}_1, \dots, \hat{b}_n) \uparrow (\hat{b}_1, \dots, \hat{b}_n) = \psi(a) \uparrow \mu(b)$ holds.

Behaviour of the flips, is now considered. Given $a' = a \uparrow b$, note that $\psi(\text{flip}(a')) = \text{flip}(\psi'(a'))$ is equivalent to $\psi(\text{flip}(a \uparrow b)) = \text{flip}(\psi(a) \uparrow \mu(b))$. The component wise equality corresponds to $\varphi_i(\text{flip}(a_i b_i)) = \text{flip}(\varphi_i(a_i) b_i)$. Since a boolean function is $\text{flip}(xy) = x\bar{y} + \bar{x}y$, the preceding equality is $\varphi_i(a_i\bar{b}_i + \bar{a}_i b_i) = \varphi_i(a_i)\bar{b}_i + \varphi_i(\bar{a}_i)b_i$.

Since φ_i is a permutation on $\{0, 1\}$, the only possibilities are $\varphi(a_i) = a_i$ or $\varphi(a_i) = \bar{a}_i$. The equation trivially holds for identity. When φ_i is a negation, it is enough to check that $\neg(a_i\bar{b}_i + \bar{a}_i b_i) = \bar{a}_i\bar{b}_i + a_i b_i$. \square

Moreover, as $\psi'(a \uparrow b) = \psi(a) \uparrow \mu(b)$, $\text{driver}(\psi'(a \uparrow b)) = \psi(a)$ also holds. Finally, $u_{\pi(i)}(\psi(\hat{a})) > u_{\pi(i)}(\pi(a))$ is obtained. Other cases are similar. \square

It has to be pointed that, all previous results are taken into account together, and it remains to show that previous transformation can be performed in polynomial time, when an input and output game representation is fixed to be one of the considered in this thesis, as stated in the following complexity equivalence.

Theorem 7.8 *For strategic games given in general form, the ISO problem for a weak isomorphism is equivalent to the circuit isomorphism problem. For formula games given in general form, the ISO problem for a weak isomorphism is equivalent to the boolean formula isomorphism problem. For strategic games given in explicit form, the ISO problem for a weak isomorphism is equivalent to the graph isomorphism problem.*

Proof. It is straightforward to show that, for a strategic game in general form, games constructed in this section can be computed in polynomial time. The same happens when an original and a target representation, are a formula games in general form, or a games in explicit form. In consequence, in this section, all games constructions show reductions in polynomial time, between different isomorphism problems.

Lemma 7.5 reduces a strong isomorphism for binary games to a weak isomorphism. Lemma 7.6 reduces a weak isomorphism to a weak isomorphism for binary actions games. Lemma 7.7 reduces a weak isomorphism to a strong isomorphism. Finally, Lemmas 7.1 and 7.2 establish a reduction from a strong isomorphism to a strong isomorphism for binary games. Therefore, all problems, for same games representations, are polynomially equivalent. Therefore, according to the complexity's equivalences stated in Section 7.2, the claim follows. \square

7.4 Notes

Game Theory. Refer to *A Course in Game Theory* of M. Osborne and A. Rubinstein [84] published on 1994. Refer also to the *Theory of Games and Economic Behavior* of John von Neuman and Oscar Morgenstern [81], for a general introduction to game theory, and mathematical definitions of strategic and extensive games, as well as the reference [80] for a definition of *Nash equilibrium*.

Algorithmic Game Theory. It is crucial to note the results from C. Álvarez et al. [8, 9] published on 2005. They studied the existence of a pure Nash equilibrium in multi-player strategic games, and proposed a brief description of games depending of explicit representations.

Refer to the well known book *Computers and Intractability: A Guide to the Theory of NP-completeness* from M. Garey and D. Johnson [45] in order to read more about coNP-complete problems.

Note explicitly, references like [71] by M. Mavronicolas, B. Monien and K. W. Wagner about weighted boolean formula games, and [16] published in 2006 by E. Bonzon, M. C. Lagasquie-Schiex, J. Lang and B. Zanuttini about boolean games.

Chapter 8

On the Hardness of Game Equivalence Under Local Isomorphism

This chapter is focused on a type of isomorphism among strategic games that has been called *local isomorphism*. Local isomorphism is a weaker version of the notions of strong and weak game isomorphism. In a local isomorphism it is required that, for any player, the player's preferences are preserved on the sets of strategy profiles that differ only in the action selected by this player. It will be shown that the game isomorphism problem for local isomorphism is equivalent to the same problem for either strong or weak isomorphisms for strategic games given in general, extensive and formula general form. As a consequence of results shown in Chapter 7 (see [41]), this implies that the local isomorphism problem for strategic games is equivalent to (a) the circuit isomorphism problem for games given in general form, (b) the boolean formula isomorphism problem for formula games in general form, and (c) the graph isomorphism problem for games given in explicit form.

8.1 The Isomorphism Problem

Just as it has been said before in defining a concrete equivalence between strategic games, it is crucial to pay attention to the structural properties that are preserved in equivalent games. In this sense, the *strong* and *weak* isomorphisms, which preserve the structure of Nash equilibria at different levels were introduced in the previous Chapter 7. Isomorphisms are defined on the basis of *game mappings* formed by a bijection among players, and for each player a bijection among its action set. In this chapter, a weaker concept of isomorphism, which has been called *local isomorphism* is considered. This isomorphism is a mapping that preserves, for each player and each strategy profile, the preferences of the player on the “close” neighborhood of the strategy profile. This condition still guarantees that the structure of Pure Nash equilibria is preserved, however the structure of mixed

Nash equilibria is not.

Accordingly, the objective is to analyse the computational complexity of the isomorphism problem. The same computational problems related to games and morphisms, which were analysed in the Chapter 7, have been considered here.

Strong Isomorphism problem (STRONGISO). Given two strategic games Γ, Γ' , decide whether $\Gamma \sim_s \Gamma'$.

Local Isomorphism problem (LOCALISO). Given two strategic games Γ, Γ' , decide whether $\Gamma \sim_\ell \Gamma'$.

Due the importance of fixing the way that games are represented as problem inputs in the context of computational complexity, two of the game representations considered in the previous chapter are also considered here. Providing a list of the sets of actions allowed to each player and their respective utilities, it is possible to give a game Γ . The two representations differ on the form in which utilities are given. In the *general form* utilities are given implicitly by a Turing Machine (TM). In the *explicit form*, utilities are provided explicitly by giving the value corresponding to each profile. Also, another succinct representation of games has been considered: *formula game in general form*. In those games, the player's utility is defined by a collection of boolean formulas, each one providing a part of the player's utility. This is one of the many ways in which games have been described in terms of formulas. In [16], player i has a goal φ_i to fulfill. Goals are usually described by boolean formulas. The utility of the player is binary. It is 1 if the goal is satisfied and 0 otherwise. Another model for strategic games, which uses boolean formula was introduced in [71], the *weighted boolean formula games*. Along the lines suggested by circuit games [102], the formula strategic for, whose representation is close to a game given in general form but with utilities defined by formulas, was introduced in Chapter 7 (see [41]). Therefore, the representations considered are as following.

- *Explicit form* by a tuple $\langle 1^n, A_1, \dots, A_n, (T_{i,a})_{1 \leq i \leq n, a \in A} \rangle$. Γ has n players, and for each player i , $1 \leq i \leq n$, their set of actions A_i is given by listing all its elements. The utility $u_i(a)$, for player i of strategy profile a , is the value $T_{i,a}$.
- *General form* by a tuple $\langle 1^n, A_1, \dots, A_n, M, 1^t \rangle$. Γ has n players, and for each player i , $1 \leq i \leq n$, their set of actions A_i is given by listing all its elements. The utility $u_i(a)$, for player i of strategy profile a , is the output of M on input $\langle a, i \rangle$ after t steps.
- *Formula general form* by a tuple $\langle 1^n, A_1, \dots, A_n, 1^\ell, (\varphi_{i,j})_{1 \leq i \leq n, 0 \leq j < \ell} \rangle$. The set of actions for player i , $1 \leq i \leq n$, is $A_i = \{0, 1\}^{m_i}$. The utility of player i is given by the boolean formulas $\varphi_{i,j}(a_1, \dots, a_n) \in \{0, 1\}$, $0 \leq j < \ell$, by the equation $u_i(a_1, \dots, a_n) = \sum_{0 \leq j < \ell} \varphi_{i,j}(a_1, \dots, a_n) 2^j$.

Previously, a classification of the complexity of the game isomorphism problem, according to the level of succinctness of the games representation has been obtained. These results show that the isomorphism problem for strong and weak isomorphisms is equivalent to the circuit isomorphism problem for games given in general form, the boolean formula isomorphism problem for formula games in general form, and the graph isomorphism problem for games given in explicit form. Consequently, the fact that the classification differs depending on the game representation, has been shown.

Wherefore, in this chapter, the computational equivalence between the isomorphism problem for strong isomorphism and the isomorphism problem for local isomorphism is shown. This equivalence is proved for any of the three game representation considered above. Thus, the fact that the local isomorphism problem for strategic games has the same classification as the other stronger notions of isomorphism, will be shown. The proof shows the equivalence through a series of steps.

Firstly, the StrongIso problem for *binary games*, games whose actions and utilities are binary, is reduced to the LocalIso problem. Then, it is shown how to reduce the LocalIso problem to the LocalIso problem for *binary actions games*, games whose actions are binary. Finally, the LocalIso problem for binary actions games is reduced to the StrongIso problem for binary games. The reductions are polynomial time computable, when the given games and the output games are written in explicit, general, and formula general form.

For the rest of this chapter, it is assumed all games with 2 or more players. Note that, in case the number of players is constant with respect to the number of actions, an explicit representation in polynomial time from a given general form representation, can be obtained. Otherwise, the transformation requires exponential time.

Theorem 8.1 (Chapter 7 (see [41])) *The STRONGISO problem is polynomially equivalent to*

- *the CIRCUITISO problem, for strategic games given in general form,*
- *the FORMULAISO problem, for strategic games given in formula general form, and*
- *to the GRAPHISO problem, for strategic games given in explicit form.*

The equivalence is also valid for binary games and binary actions games.

In the following sections, the computational equivalence among the two game isomorphism problems have been proved. For doing so, a series of game transformations is provided, which preserve local isomorphism or strong isomorphism. Finally, it will be shown that for any of these transformations, given an strategic game in form F , a description in form F of the transformed game can be obtained in polynomial time, when F is any of the three game representations considered.

8.2 From Strong Isomorphism to Local Isomorphism

Game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ is assumed as a binary game, where $N = \{1, \dots, n\}$. For $a_i \in \{0, 1\}$, $\bar{a}_i = 1 - a_i$ has been defined.

$\text{CHECKL}(\Gamma) = (N', (A'_i)_{i \in N'}, (u'_i)_{i \in N'})$ has $n + 1$ players, that is $N' = \{0, 1, \dots, n\}$. Player 0 has $A'_0 = \{0, 1\}$ and for $i > 0$ the set of actions is $A'_i = \{0, 1\}^2$. A profile factor is defined as $c = (a_0, a')$ with $a_0 \in A'_0$, $a' = (a_1 b_1, \dots, a_n b_n)$ with $a_i b_i \in A'_i$. Note that the part $a = (a_1, \dots, a_n)$ extracted from c is a profile in Γ .

The utilities for any player i , $0 \leq i \leq n$, and any profile $c = (a_0, b)$, have been defined. The utility of player 0 is defined as $u'_0(c) = a_0$. Note that it depends only on a_0 , and player 0 prefers 1 to 0. In order to define u'_i for $i > 0$, the cases $a_0 = 1$ and $a_0 = 0$, have been considered separately. When $a_0 = 1$, we set $u'_i(c) = 7$ when $b_i = 0$, and $u'_i(c) = 8$ when $b_i = 1$. When $a_0 = 0$, there are three cases:

- When $u_i(a) \neq u_i(a_{-i}, \bar{a}_i)$, the following can be defined:

$$u'_i(c) = \begin{cases} 0 & \text{when } b_i = 0 \text{ and } u_i(a) = 1, \\ 1 & \text{when } b_i = 0 \text{ and } u_i(a) = 0, \\ 2 & \text{when } b_i = 1 \text{ and } u_i(a) = 0, \\ 3 & \text{when } b_i = 1 \text{ and } u_i(a) = 1. \end{cases}$$

- When $u_i(a) = u_i(a_{-i}, \bar{a}_i) = 0$, $u'(c) = 4$ is defined.
- When $u_i(a) = u_i(a_{-i}, \bar{a}_i) = 1$, set

$$u'_i(c) = \begin{cases} 5 & \text{when } b_i = 0, \\ 6 & \text{when } b_i = 1. \end{cases}$$

Example 1 $\text{CHECKL}(\Gamma)$ for the following game Γ , is constructed

		Player 2	
		0	1
Player 1	0	0, 1	0, 1
	1	1, 0	1, 0

Γ_1

Since Γ has two players, the game $\text{CHECKL}(\Gamma_1)$ has three players $N' = \{0, 1, 2\}$. Any profile in CHECKL factors as $c = (a_0, a_1 b_1, a_2 b_2)$, and utilities are defined by two tables: one corresponds to $a_0 = 0$ and the other to $a_0 = 1$.

When $a_0 = 0$, as $u_1(0, 0) = 0 \neq u_1(1, 0)$ and for any $a \in \{0, 1\}$ the first player utilities are defined as,

$$u'_1(0, 0b_1, 0a) = \begin{cases} 1 & \text{if } b_1 = 0 \\ 2 & \text{if } b_1 = 1 \end{cases} \quad u'_1(0, 1b_1, 0a) = \begin{cases} 0 & \text{if } b_1 = 0 \\ 3 & \text{if } b_1 = 1 \end{cases}$$

When $a_2 = 0$, the analysis is similar and for any $a \in \{0, 1\}$ the following is obtained,

$$u'_1(0, 0b_1, 1a) = \begin{cases} 1 & \text{if } b_1 = 0 \\ 2 & \text{if } b_1 = 1 \end{cases} \quad u'_1(0, 1b_1, 1a) = \begin{cases} 0 & \text{if } b_1 = 0 \\ 3 & \text{if } b_1 = 1 \end{cases}$$

Utilities for the second player, are written. When $a_1 = 0$, since $u_2(0, 0) = u_2(0, 1) = 1$ the utility of u'_2 depends on b_2 . Nevertheless, when $a_1 = 1$ as $u_2(1, 0) = u_2(1, 1) = 0$ the utility is independent of b_2 . Thus, for any $a, b, c \in \{0, 1\}$,

$$u'_2(0, 0a, bb_2) = \begin{cases} 5 & \text{if } b_2 = 0 \\ 6 & \text{if } b_2 = 1 \end{cases} \quad u'_2(0, 1a, bc) = 4$$

Collecting all this information, the first table is obtained.

		Player 2			
		00	01	10	11
Player 1	00	0, 1, 5	0, 1, 6	0, 1, 5	0, 1, 6
	01	0, 2, 5	0, 2, 6	0, 2, 5	0, 2, 6
	10	0, 0, 4	0, 0, 4	0, 0, 4	0, 0, 4
	11	0, 3, 4	0, 3, 4	0, 3, 4	0, 3, 4

Player 0 chooses 0

When $a_0 = 1$, utilities u'_1 and u'_2 are defined as follows. For any $a, b, c \in \{0, 1\}$,

$$u'_1(1, ab_1, bc) = \begin{cases} 7 & \text{if } b_1 = 0 \\ 8 & \text{if } b_1 = 1 \end{cases} \quad u'_2(1, ab, cb_2) = \begin{cases} 7 & \text{if } b_2 = 0 \\ 8 & \text{if } b_2 = 1 \end{cases}$$

and the following table is obtained,

		Player 2			
		00	01	10	11
Player 1	00	1, 7, 7	1, 7, 8	1, 7, 7	1, 7, 8
	01	1, 8, 7	1, 8, 8	1, 8, 7	1, 8, 8
	10	1, 7, 7	1, 7, 8	1, 7, 7	1, 7, 8
	11	1, 8, 7	1, 8, 8	1, 8, 7	1, 8, 8

Player 0 chooses 1

Before showing the correctness of this transformation, some ad-hoc notation will be introduced in order to deal with profiles in $\text{CHECKL}(\Gamma)$. Given $a' = (a_1b_1, \dots, a_nb_n)$ there is $a' = a \uparrow b$ with $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$. Given a player i , we factor a profile c in $\text{CHECKL}(\Gamma)$ as $c = (a_0, a_{-i} \uparrow b_{-i}, a_ib_i)$, adopting here the criterion $-i = N' \setminus \{0, i\}$ ($-i$ is the adversary team of i in game Γ). Given ψ and defining $\mu = (\pi, id_1, \dots, id_n)$, using that for $i > 0$ player i maps into player $\pi(i)$, it holds that,

$$\psi'(a_0, a_{-i} \uparrow b_{-i}, a_ib_i) = (a_0, \psi(a_{-i}) \uparrow \mu(b_{-i}), \phi(a_i)b_i).$$

Lemma 8.1 *Let Γ_1 and Γ_2 be two binary games such that, $\Gamma_1 \sim_s \Gamma_2$, then $\text{CHECKL}(\Gamma_1) \sim_\ell \text{CHECKL}(\Gamma_2)$.*

Proof. Let $\Gamma'_1 = \text{CHECKL}(\Gamma_1)$ and $\Gamma'_2 = \text{CHECKL}(\Gamma_2)$. Let also $\psi : \Gamma_1 \rightarrow \Gamma_2$ be a strong isomorphism, then it is assumed that $\psi = (\pi, \varphi_1, \dots, \varphi_n)$. The mapping $\psi' : \text{CHECKL}(\Gamma_1) \rightarrow \text{CHECKL}(\Gamma_2)$, $\psi' = (p, f_0, \dots, f_n)$ has been considered where,

- $p(0) = 0$ and $p(i) = \pi(i)$, for $i > 0$, and,
- $f_0(a_0) = a_0$ and $f_i(a_i b_i) = \varphi_i(a_i) b_i$, for $i > 0$.

The fact that mapping ψ' verifies $u_i(c) = u_{p(i)}(\psi'(c))$ will be shown.

Player $i = 0$ is considered. Since, $p(0) = 0$ and $f_0(a_0) = a_0$ given $c = (a_0, \dots)$ it holds that, $u_0(c) = u_{p(0)}(\psi'(c)) = a_0$.

Case $i > 0$ with $c = (0, a_{-i} \uparrow b_{-i}, a_i b_i)$ is considered. It holds that, $u_i(c) = u_{p(i)}(\psi'(c))$. The proof is shown by case analysis.

- When $u_i(a) \neq u_i(a_{-i}, \bar{a}_i)$ it holds that, $u_{\pi(i)}(\psi(a)) \neq u_{\pi(i)}(\psi(a_{-i}), \overline{\varphi(a_i)})$. Since ψ is a strong morphism, $u_i(a) = u_{\pi(i)}(\psi(a))$ and $u_i(a_{-i}, \bar{a}_i) = u_{\pi(i)}(\psi(a_{-i}), \overline{\varphi(a_i)})$ are obtained. As $a_i \in \{0, 1\}$ and φ_i is a bijection, $\varphi_i(\bar{a}_i) = \overline{\varphi_i(a_i)}$ holds. Then, the inequality is concluded. The case $u'_i(c) = 0$ is considered (other cases are similar). When $u'_i(c) = 0$, the profile c verifies $b_i = 0$ and $u_i(a) = 1$. Since $\mu_i(b_i) = b_i$ and $u_{\pi(i)}(\psi(a)) = 1$, it holds that $u'_{p(i)}(\psi'(c)) = u'_i(c) = 0$.
- When $u_i(a) = u_i(a_{-i}, a_i) = 0$ it holds that, $u_{\pi(i)}(\psi(a)) = u_{\pi(i)}(\psi(a_{-i}), \overline{\varphi(a_i)}) = 0$. Therefore, $u'_{p(i)}(\psi'(c)) = u'_i(c) = 0$. The proof is similar to the preceding case.
- When $u_i(a) = u_i(a_{-i}, a_i) = 1$ it holds that, $u_{\pi(i)}(\psi(a)) = u_{\pi(i)}(\psi(a_{-i}), \overline{\varphi(a_i)}) = 1$. As $\mu_i(b_i) = b_i$ it holds that, $u'_{p(i)}(\psi'(c)) = u'_i(c) \in \{5, 6\}$.

Case $i > 0$ with $c = (1, a_{-i} \uparrow b_{-i}, a_i b_i)$ is considered. Since $\mu_i(b_i) = b_i$, $u'_{p(i)}(\psi'(c)) = u'_i(c) \in \{7, 8\}$ holds.

Therefore, it is possible conclude that ψ' is a strong isomorphism, and indeed a local isomorphism. □

Now, the reverse implication will be proven.

Lemma 8.2 *Let Γ_1 and Γ_2 be two binary games. If $\text{CHECKL}(\Gamma_1) \sim_\ell \text{CHECKL}(\Gamma_2)$, then $\Gamma_1 \sim_s \Gamma_2$.*

Proof. It is assumed that $\psi' : \Gamma'_1 \rightarrow \Gamma'_2$ is a local isomorphism with $\psi' = (p, f_0, \dots, f_n)$. Since, all f_i for $0 \leq i \leq n$ are bijections, players' bijection p has to map player 0 into player 0, because player 0 is the only one with 2 actions. Therefore, $p(0) = 0$ and players

$\{1, \dots, n\}$ are bijectively mapped into $\{1, \dots, n\}$ by p , so $\pi(i) = p(i)$ can be defined for $1 \leq i \leq n$, and a player's bijection has been obtained.

Two profiles $c_1 = (0, a'_1)$ and $c_2 = (1, a'_2 \dots)$ are considered, and it holds that $c_1 \prec_0 c_2$. In addition, $f_0(a_0) = \bar{a}_0$ is assumed. Since $p(0) = 0$, $\psi'(c) = (1, \dots)$, $\psi'(\hat{c}) = (0, \dots)$ and $\psi'(c) \prec_0 \psi'(\hat{c})$ holds. Therefore, a contradiction is obtained, and it holds that f_0 is the identity.

Now it will be proven that, $f_i(a_i 0) = \varphi_i^0(a_i)0$ and $f_i(a_i 1) = \varphi_i^1(a_i)1$, for $1 \leq i \leq n$.

Given $a'_{-i} = a_{-i} \uparrow b_{-i}$ it holds that $(1, a'_{-i}, a_i 0) \prec_i (1, a'_{-i}, a_i 1)$, because $u'_i(1, a'_{-i}, a_i 0) = 7$ and $u'_i(1, a'_{-i}, a_i 1) = 8$. As ψ' is a local isomorphism and $f_0(1) = 1$ there is necessarily a $(1, \psi'(a'_{-i}), f_i(a_i 0)) \prec_{p(i)} (1, \psi'(a'_{-i}), f_i(a_i 1))$. This forces factorisations $f_i(a_i 0) = \varphi_i^0(a_i)0$ and $f_i(a_i 1) = \varphi_i^1(a_i)1$.

Finally, the fact that the mapping $\psi^0 = (\pi, \varphi^0, \dots, \varphi^0)$ between Γ_1 and Γ_2 is a strong isomorphism will be proven. It is proved by cases that $u_i(a) = u_{\pi(i)}(\psi^0(a))$.

Case 1: $a = (a_{-i}, a_i)$ with $u_i(a_{-i}, a_i) \neq u_i(a_{-i}, \bar{a}_i)$.

In the following, $b = (0, \dots, 0)$ is used. The array containing $n - 1$ zeros.

The proof for $u_i(a_{-i}, a_i) = 1$ and $u_i(a_{-i}, \bar{a}_i) = 0$, has been given. For $u_i(a_{-i}, a_i) = 0$ and $u_i(a_{-i}, \bar{a}_i) = 1$ the proof is similar. Since,

$$\begin{aligned} u'_i(0, a_{-i} \uparrow b, a_i 0) &= 0, \quad u'_i(0, a_{-i} \uparrow b, \bar{a}_i 0) = 1 \\ u'_i(0, a_{-i} \uparrow b, \bar{a}_i 1) &= 2, \quad u'_i(0, a_{-i} \uparrow b, a_i 1) = 3 \end{aligned}$$

Then, $(0, a_{-i} \uparrow b, a_i 0) \prec_i (0, a_{-i} \uparrow b, \bar{a}_i 0) \prec_i (0, a_{-i} \uparrow b, \bar{a}_i 1) \prec_i (0, a_{-i} \uparrow b, a_i 1)$.

Since ψ' is a local morphism,

$$\begin{aligned} (0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(a_i)0) &\prec_{\pi(i)} (0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(\bar{a}_i)0) \\ &\prec_{\pi(i)} (0, \psi^0(a_{-i}) \uparrow b, \varphi_i^1 \bar{a}_i 1) \prec_{\pi(i)} (0, \psi^0(a_{-i}) \uparrow b, \varphi_i^1(a_i)1) \end{aligned}$$

This forces us to the values,

$$\begin{aligned} u'_{\pi(i)}(0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(a_i)0) &= 0, \quad u'_{\pi(i)}(0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(\bar{a}_i)0) = 1 \\ u'_{\pi(i)}(0, \psi^0(a_{-i}) \uparrow b, \varphi_i^1 \bar{a}_i 1) &= 3, \quad u'_{\pi(i)}(0, \psi^0(a_{-i}) \uparrow b, \varphi_i^1(a_i)1) = 4 \end{aligned}$$

Since $u_{\pi(i)}(0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(a_i)0) = 0$ and $u_{\pi(i)}(0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(\bar{a}_i)0) = 1$, the difference in values forces that, we must have $u_{\pi(i)}(\psi^0(a_{-i}), \varphi_i^0(a_i)) = u_{\pi(i)}(\psi^0(a)) = 0$ in Γ_2 , and $u_{\pi(i)}(\psi^0(a_{-i}), \varphi_i^0(\bar{a}_i)) = 1$.

Case 2: $a = (a_{-i}, a_i)$ with $u_i(a_{-i}, a_i) = u_i(a_{-i}, \bar{a}_i) = 1$. As before, $b_{-i} = (0, \dots, 0)$ is used.

In this case utilities verify:

$$\begin{aligned} u'_i(0, a_{-i} \uparrow b, a_i 0) &= u'_i(0, a_{-i} \uparrow b, \bar{a}_i 0) = 5 \\ u'_i(0, a_{-i} \uparrow b, \bar{a}_i 1) &= u'_i(0, a_{-i} \uparrow b, a_i 1) = 6 \end{aligned}$$

Then, $(0, a_{-i} \uparrow b, a_i 0) \sim_i (0, a_{-i} \uparrow b, \bar{a}_i 0) \prec_i (0, a_{-i} \uparrow b, \bar{a}_i 1) \sim_i (0, a_{-i} \uparrow b, a_i 1)$.

Since ψ' is a local isomorphism,

$$\begin{aligned} (0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(a_i) 0) &\sim_{\pi(i)} (0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(\bar{a}_i) 0) \\ &\prec_{\pi(i)} (0, \psi^0(a_{-i}) \uparrow b, \varphi_i^1 \bar{a}_i 1) \sim_{\pi(i)} (0, \psi^0(a_{-i}) \uparrow b, \varphi_i^1(a_i) 1) \end{aligned}$$

This preorder structure forces the value of utilities, in particular we have that $u'_{\pi(i)}(0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(a_i) 0) = 5$, but then $u_{\pi(i)}(\psi^0(a)) = 1$.

Case 3: $a = (a_{-i}, a_i)$ with $u_i(a_{-i}, a_i) = u_i(a_{-i}, \bar{a}_i) = 0$. In this case there is

$$(0, a_{-i} \uparrow b, a_i 0) \sim_i (0, a_{-i} \uparrow b, \bar{a}_i 0) \sim_i (0, a_{-i} \uparrow b, \bar{a}_i 1) \sim_i (0, a_{-i} \uparrow b, a_i 1).$$

This forces $u'_{\pi(i)}(0, \psi^0(a_{-i}) \uparrow b, \varphi_i^0(a_i) 0) = 4$, but then $u_{\pi(i)}(\psi^0(a)) = 0$.

Then, the proof has been concluded. □

8.3 From General Games to Binary Actions Games

The next step is to transform a strategic game into a binary actions game preserving local isomorphism. The game construction follows the same lines as in the BINARYACT in previous Chapter 7, but now, definition has been adapted, in order to guarantee an adequate local preference relation for each player.

Given a strategic game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$, $N = \{1, \dots, n\}$ is assumed without loss of generality and that, for any $i \in N$, $A_i = \{1, \dots, k_i\}$ for suitable values. Given $A_i = \{1, \dots, k_i\}$ we “binify” an action $j \in A_i$ coding it with k_i bits, as $\text{binify}(j) = 0^{j-1} 1 0^{k_i-j}$. Thus, $\text{binify}(A_i) \subseteq A^i = \{0, 1\}^{k_i}$. The binify process can be used in a strategy profile, given $a = (a_1, \dots, a_n)$, $\text{binify}(a) = \text{binify}(a_1) \cdots \text{binify}(a_n)$ is written. Note that, by setting $k = \sum_{i \in N} k_i$, there is $\text{binify}(a) \in A' = \{0, 1\}^k = A^{k_1} \times \cdots \times A^{k_n}$. Thus, $\text{good}(A') = \{\text{binify}(a) \mid a \in A\}$ and $\text{bad}(A') = A' \setminus \text{good}(A')$ have been defined. Note that, $\text{binify} : A \rightarrow \text{good}(A')$ is a bijection and therefore, its inverse function is also a bijection. Notice that, for $a' \in \text{good}(A')$, $\text{binify}^{-1}(a') = \text{binify}^{-1}(b_1) \cdots \text{binify}^{-1}(b_n)$. It is assumed that, $\Gamma = (N, A_1, \dots, A_n, (u_i)_{1 \leq i \leq n})$. Then, the following game is constructed.

BINARYACTL(Γ) = $(N', (A'_i)_{i \in N'}, (u'_i)_{i \in N'})$, where $N' = \{1, \dots, k\}$ and, for any $i \in N'$, $A'_i = \{0, 1\}$ and thus, the set of action profiles is $A' = \{0, 1\}^k$.

For a player i with $|A_i| > 2$, a block B_i is associated to A_i , and a block C_i of $k_i = |A_i|$ players in each block, each player takes care of one bit. For a player i with $|A_i| \leq 2$, a block B_i is associated to player i , formed by three players, $k_i = 3$. Thus, $k = 2(k_1 + \cdots + k_n)$.

A strategy profile a' is decomposed into $2n$ blocks, 2 blocks per player, so that $a' = (b_1, \dots, b_n, c_1, \dots, c_n)$ where $b_i, c_i \in \{0, 1\}^{k_i}$, often $a' = (b, c)$ will be also decomposed.

We keep a'_j , to refer to the strategy of player j in the profile a' . But sometimes, we refer to its strategy by the position inside its corresponding B or C block.

For a player α , which occupies position j in block B_i , the utility function is defined as,

$$u'_\alpha(a') = \begin{cases} 0 & \text{if } b_i \in \text{bad}(A^i), \\ 1 & \text{if } b_i \in \text{good}(A^i) \text{ and } b \in \text{bad}(A'), \\ 2 & \text{if } b \in \text{good}(A'). \end{cases}$$

For a player β , which occupies position j in block C_i , the utility function is defined as,

$$u'_\beta(a') = \begin{cases} 1 - a'_\beta & \text{if } b_i \neq c_i \text{ and } b \in \text{bad}(A'), \\ 3 - a'_\beta & \text{if } b_i = c_i \text{ and } b \in \text{bad}(A'), \\ 4 + u_i(\text{binify}^{-1}(b)) & \text{if } a'_\beta = 1 \text{ and } b \in \text{good}(A'), \\ 4 + u_i(\text{binify}^{-1}(b)_{-i,j}) & \text{if } a'_\beta = 0 \text{ and } b \in \text{good}(A'). \end{cases}$$

The case with alphabets having a small number of actions, will be commented. When $A_i = \{1\}$, by definition B_i contains 3 players, $A^i = \{0,1\}^3$ and $\text{good}(A^i) = \{\text{binify}(1)\} = \{100\}$. When $A_i = \{1,2\}$, there is $A^i = \{0,1\}^3$ and $\text{good}(A^i) = \{\text{binify}(1), \text{binify}(2)\} = \{100, 010\}$. Finally, when $A_i = \{1,2,3\}$ there is also $A^i = \{0,1\}^3$ but $\text{good}(A^i) = \{100, 010, 001\}$.

Example 2 A version game Γ of rock-paper-scissors has been considered, where 1 is added to all utilities, to obtain non negative values. The set of actions is $A_i = \{1,2,3\}$, where 1 corresponds to rock, 2 to paper and 3 to scissors.

		Player 2		
		1	2	3
Player 1	1	1, 1	0, 2	2, 0
	2	2, 0	1, 1	0, 2
	3	0, 2	2, 0	1, 1

Γ

$\text{BINARYACTL}(\Gamma)$ has 12 players having binary actions. The strategy profiles are $a' = (a_1, \dots, a_{12}) = (b_1, b_2, c_1, c_2)$, with $b_i, c_i \in \{0,1\}^3$. In this case,

$$\text{good}(A^3) = \{100, 010, 001\}, \quad \text{bad}(A^3) = \{0,1\}^3 \setminus \text{good}(A^3)$$

Since the number of strategy profiles is 2^{12} some examples are given.

The profile $a' = (b_1, b_2, c_1, c_2) = (000, 010, 100, 010)$ is considered. The utility u'_α for a player α in block B_1 or B_2 , depends only on the b part of the profile. As, $b_1 = 000 \in \text{bad}(A^3)$, $b_2 = 010 \in \text{good}(A^3)$ and $b = 000010 \in \text{bad}(A^6)$ therefore,

$$u'_\alpha(a') = \begin{cases} 0 & \text{if } \alpha \in \{1,2,3\} \\ 1 & \text{if } \alpha \in \{4,5,6\} \end{cases}$$

for a player β , which occupies position j in block C_i ,

$$u'_\beta(a') = \begin{cases} 1 - a'_\beta & \text{if } \beta \in \{10, 11, 12\} \\ 3 - a'_\beta & \text{if } \alpha \in \{13, 14, 15\} \end{cases}$$

These results can be summarised in the following table,

player	β	7	8	9	10	11	12
action	a'_β	1	0	0	0	1	0
utility	u'_β	0	1	1	3	2	3

When $a' = (010, 100, 011, 010)$, $b = 010100 \in \text{good}(A^6)$ and $u'_\alpha(a') = 2$ for $\alpha \in \{1, \dots, 6\}$. Furthermore, setting $a = \text{binify}^{-1}(010100) = (2, 1) \in A$, that is associated to an action profile in Γ . Then,

$$u'_\beta(a') = \begin{cases} 4 + u_1(a_{-1}, 1) = 4 + u_1(1, 1) = 5 & \text{if } \beta = 7 \\ 4 + u_1(a) = 4 + u_1(2, 1) = 6 & \text{if } \beta \in \{8, 9\} \\ 4 + u_2(a_{-2}, 1) = 4 + u_1(2, 1) = 5 & \text{if } \beta = 10 \\ 4 + u_2(a) = 4 + u_2(2, 1) = 5 & \text{if } \beta = 11 \\ 4 + u_1(a_{-2}, 3) = 4 + u_2(2, 3) = 6 & \text{if } \beta = 12 \end{cases}$$

Which results is in the following table,

player	β	7	8	9	10	11	12
block number	i	1	1	1	2	2	2
position into the block	j	1	2	3	1	2	3
action	a'_β	0	1	1	0	1	0
utility	u'_β	5	6	6	5	5	6

Given a player α in $\text{BINARYACTL}(\Gamma)$, local indifference set of α has been defined as $I(\alpha) = \{a'_{-\alpha} | (a'_{-\alpha}, 0) \sim_\alpha (a'_{-\alpha}, 1)\}$.

Lemma 8.3 Let Γ_1 and Γ_2 be two strategic games, such that $\Gamma_1 \sim_\ell \Gamma_2$, then $\text{BINARYACTL}(\Gamma_1) \sim_\ell \text{BINARYACTL}(\Gamma_2)$.

Proof. Let $\Gamma'_1 = \text{BINARYACTL}(\Gamma_1)$ and $\Gamma'_2 = \text{BINARYACTL}(\Gamma_2)$. It is assumed that $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ being a local isomorphism between Γ_1 and Γ_2 . The mapping $\psi' = (p, f_1, \dots, f_k)$ is considered such that, for $1 \leq i \leq n$, p maps bits in block B_i of Γ'_1 to bits in block $B_{\pi(i)}$ of Γ'_2 , and bits in block C_i of Γ'_1 to bits in block $C_{\pi(i)}$ of Γ'_2 , so that bit j goes to bit $\varphi_i(j)$, and for $1 \leq \alpha \leq k$, f_α is the identity function. It is straightforward to show that ψ' is a strong (therefore also a local) isomorphism between Γ'_1 and Γ'_2 . \square

Before proving the reverse implication, some properties of the constructed game have been analysed. Let Γ be a game and let $\Gamma' = \text{BINARYACTL}(\Gamma)$. As usual, $n > 1$ is assumed. A player α is considered. This player occupies position j in block B_i , for $u = 0, 1, 2$, and define $X_u(\alpha) = \{a' | u'_\alpha(a') = u\}$.

Lemma 8.4 *Given an α player belonging to a B-block, it holds that $|X_0(\alpha)| > |X_1(\alpha)| > |X_2(\alpha)|$.*

Proof. Notice that, any strategy a' factors $a' = (b, c)$. Since the utility $u'_\alpha(a')$ only depends on the b part, the c part gives a common multiplicative constant. The b part of a' , is only analysed. Suppose that α occupies position j in block B_i and we factor $b = (b_{-i}, b_i)$.

First, $|X_0(\alpha)| > |X_1(\alpha)|$ will be proven. When $a' \in X_0(\alpha)$, the profile factors $a' = (b, c)$, such that $b_i \in \text{bad}(A^i)$ and there are no special restrictions on b_{-i} . When $\hat{a}' \in X_1(\alpha)$, the profile factors $\hat{a}' = (\hat{b}, \hat{c})$, such that $\hat{b}_i \in \text{good}(A^i)$ and $\hat{b}_{-i} \in \text{bad}(A^i)$. Since $|\text{bad}(A^i)| = 2_i^k - k_i$ and $|\text{good}(A^i)| = k_i$ and $k_i > 3$, it holds that $|\text{bad}(A^i)| > |\text{good}(A^i)|$. As there are $2^{k/2-k_i}$ profiles b_{-i} with no special restrictions, note that, $k/2 - k_i = k_1 + \dots + k_{i-1} + k_{i+1} + \dots + k_n$ and $|\text{bad}(A^i)| = 2^{k/2-k_i} - k/2 - k_i$, the inequality also holds for the second part. Since there are more elements in both parts (parts are i and $-i$) in the first case, inequality holds.

It remains to be proven that $|X_1(\alpha)| > |X_2(\alpha)|$. There is $a' \in X_1(\alpha)$, iff $a' = (b, c)$ and b factors as (b_{-i}, b_i) with $b_i \in \text{good}(A^i)$ and $b_{-i} \in \text{bad}(A^{-i})$. There is $\hat{a}' \in X_2(\alpha)$, iff $\hat{a}' = (\hat{b}, \hat{c})$ and \hat{b} factors as $(\hat{b}_{-i}, \hat{b}_i)$ with $\hat{b}_i \in \text{good}(A^i)$ and $\hat{b}_{-i} \in \text{good}(A^{-i})$. As $\text{bad}(A^{-i}) > \text{good}(A^{-i})$ inequality holds

□

Given a player α in Γ' , *local indifference set* $I(\alpha)$ for this player is the set $I(\alpha) = \{a'_{-\alpha} | (a'_{-\alpha}, 0) \sim_\alpha (a'_{-\alpha}, 1)\}$.

Lemma 8.5 *Let α be a player belonging to a B-block and, let β be a player belonging to a C-block. It holds that,*

- *the indifference set to $I(\alpha)$ has 2^{k-3} elements,*
- *the indifference set to $I(\beta)$ has at most $k_1 \dots k_n 2^{k/2-1}$ elements and*
- *$|I(\beta)| < |I(\alpha)|$ holds.*

Proof. A player α , which occupies position j in block B_i , is considered. Notice that for any strategy profile a' of Γ' , the following is obtained:

- if $(a'_{-\alpha}, 1) \in X_2(\alpha)$ then $(a'_{-\alpha}, 0) \in X_0(\alpha)$, since we are eliminating the unique 1 in B_i ,
- if $(a'_{-\alpha}, 0) \in X_2(\alpha)$ then $(a'_{-\alpha}, 1) \in X_0(\alpha)$, since B_1 will have two 1's after the transformation,
- if $(a'_{-\alpha}, 1) \sim_\alpha (a'_{-\alpha}, 0)$ iff $(a'_{-\alpha}, 1), (a'_{-\alpha}, 0) \in X_0(\alpha)$. Note that, profile b_i corresponding to block B_i in a'_α factors $b_i = b_{i_1} \dots b_{i_j} \dots b_{i_{k_i}}$. Profile $a'_{-\alpha}$ ignores the

action b_{i_j} corresponding to player α . Ignoring player α in b_i we write $(b_i)_{-\alpha} = b_{i_1} \dots b_{i_{j-1}} b_{i_{j+1}} \dots b_{i_{k_i}}$, the preceding indifference condition is equivalent to $1 \in \{b_{i_1}, \dots, b_{i_{j-1}}\}$ and $1 \in \{b_{i_{j+1}}, \dots, b_{i_{k_i}}\}$. Since $|A^i| \geq 3$ we always have at least 3 positions.

Now, $I(\alpha)$ will be analysed. Since any profile $(b_i)_{-\alpha}$ contains $k_i - 1$ players and (at least) two players have to choose 1, $k_i - 3$ “free” choices remain profile $a'_{-\alpha}$ contains a profile c and this give us $k/2$ possibilities. Finally, profile $a'_{-\alpha}$ contains also a part b_{-i} and this gives us $k/2 - k_i$ extra choices. Based on that, the cardinal of $I(\alpha)$ is 2^{k-3} . Thus, the number of $a'_{-\alpha}$, which give raise to an indifferent pair for player α is 2^{k-3} . Note that this value is independent of the chosen block, B_i .

A player β , which occupies position j in block C_i and the set $I(\beta)$ have been considered now. The cardinal of $I(\beta)$ is at most $k_1 \dots k_n 2^{k/2-1}$ because, in order to give raise to an indifferent pair for player α profile a' must verify $b \in \text{good}(A')$. Since $n > 1$ it holds that $|I(\beta)| < |I(\alpha)|$. \square

Given $\Gamma'_1 = \text{BINARYACTL}(\Gamma_1)$ and $\Gamma'_2 = \text{BINARYACTL}(\Gamma_2)$ let $\psi' = (p, f_1, \dots, f_k)$ be a local morphism between Γ'_1 and Γ'_2 .

Lemma 8.6 *Local morphisms map bijectively indifference into indifference sets, that is:*

- given a player α_1 in Γ'_1 it holds $\psi'(I(\alpha_1)) = I(p(\alpha_1))$ and
- given a player α_2 in Γ'_2 it holds $\psi'^{-1}(I(\alpha_2)) = I(p^{-1}(\alpha_2))$.

Proof. A player α_1 in Γ_1 has been taken. A profile $a'_{-\alpha_1}$ belongs to $I(\alpha_1)$, iff $(a'_{-\alpha_1}, 0) \sim_{\alpha_1} (a'_{-\alpha_1}, 1)$. Since, ψ' is a local morphism, then $\psi'(a'_{-\alpha_1}, 0) \sim_{p(\alpha_1)} \psi'(a'_{-\alpha_1}, 1)$. As player α_1 is mapped into $p(\alpha_1)$, the factorisations $\psi'(a'_{-\alpha_1}, 0) = (\psi'(a'_{-\alpha_1}), f_{\alpha}(0))$ and $\psi'(a'_{-\alpha_1}, 1) = (\psi'(a'_{-\alpha_1}), f_{\alpha}(1))$ are obtained. Since f_{α_1} is a bijection in $\{0, 1\}$, then $\psi'(a'_{-\alpha_1}) \in I(p(\alpha_1))$ and therefore, $\psi'(I(\alpha_1))$ is included in $I(p(\alpha_1))$. It is supposed that inclusion is strict, then $a'_{-p(\alpha_1)} \in I(p(\alpha_1)) \setminus \psi'(I(\alpha_1))$ exists. It is straightforward to prove that $\psi'^{-1}(a'_{-p(\alpha_1)})$ belongs to $I(\alpha_1)$, getting a contradiction. Case ψ'^{-1} is similar. \square

Lemma 8.7 *Let $\psi' = (p, f_1, \dots, f_k)$ be a local morphism between Γ'_1 and Γ'_2 ,*

- ψ' preserves B-blocks and f_{α} is the identity, when α is a B-player.
- ψ' preserves C-blocks and f_{β} is the identity, when β is a C-player.
- The permutation π induced in B-blocks and C-blocks by p is the same.

- Let φ_i be a mapping p restricted to B_i , and let $\hat{\varphi}_i$ be a mapping p restricted to C_i . It holds that $\varphi_i = \hat{\varphi}_i$.

Proof. The proof proceeds through a sequence of claims. *The local morphism ψ' preserves B -blocks.* Block B_i in Γ'_1 has been supposed the only partially mapped into block B_ℓ in Γ'_2 . There exists player α in B_i and another B -player α' such that, $p(\alpha)$ and $p(\alpha')$ belong to B_ℓ . It is supposed that player α occupies position j in B_i . In Γ'_1 , a profile $a' = (b, c)$ is chose such that, $b_i = 0^{j-1}10^{k_i-j}$. In addition, b_{-i} is chosen in such a way that, in $\psi'(b) = (b_{-\ell}, b_\ell)$ the profile b_ℓ is bad. This is always possible because we are free to set the action belonging to α' in such a way that, $f_{\alpha'}(a_{\alpha'})$ forces b_ℓ to be bad. Note that, $u'_\alpha(a') \in \{1, 2\}$ but $u'_{p(\alpha)}(\psi'(a')) = 0$ and $a' = (a'_{-\alpha}, 1)$. Profile $\hat{a}' = (a'_{-\alpha}, 0)$ factors as $\hat{a}' = (\hat{b}, \hat{c})$ with $b_i \in \text{bad}(A^i)$, because $b_i = 0^{k_i}$. Therefore, $u'_\alpha(\hat{a}') = 0$. It holds that $a' = (a'_{-\alpha}, 1) \succ_\alpha (a'_{-\alpha}, 0) = \hat{a}'$. Since ψ' is a local we should have $\psi'(a') \succ_{p(\alpha)} \psi'(a'_{-\alpha}, 0)$. This cannot be true, because $\psi'(a')$ is a minimal element among preferences (in view of $u'_{p(\alpha)}(\psi'(a')) = 0$). Therefore, ψ' preserves B -blocks.

For any B -player, α mapping f_α is the identity. Note that, functions f_α can only be identities or negations. The case analysis has been done based on the number of negations.

- **Case with 1 negation.** f_α has been supposed the only negation among players in B_i . In addition, it is supposed that α plays position j in the block. We consider $a' = (b, c)$ with $b = (b_{-i}, b_i)$ and also $b_i = 0^{j-1}10^{k_i-j}$ then, $a' = (a'_{-\alpha}, 1)$. Consider $\hat{a}' = (a'_{-\alpha}, 0) = (\hat{b}, \hat{c})$ with $\hat{b}_i = 0^{k_i}$. Then, $a' \succ_\alpha \hat{a}'$ but $\psi'(b_i) = 0^{k_i}$, therefore a contradiction is obtained.
- **Case with 2 negations.** It is supposed that only α and α' have negating player functions. Note that, $I(\alpha) = I(p(\alpha))$. In B_i there are at least three players. Therefore, γ exists such that f_γ is the identity. A b_i has been considered such that actions corresponding to α , α' and γ are 1, and all other actions are 0. Clearly, such b_i is bad. Let $a' = (b, c)$ be a profile containing the preceding b_i . Clearly, $a'_{-\alpha} \in I(\alpha)$ because the part of the profile corresponding to b_i has two ones (the one corresponding to player α is missing). Profile $\psi'(a'_{-\alpha})$ contains just one 1. Therefore, $\psi'(a'_{-\alpha}) \notin I(p(\alpha))$, because player $p(\alpha)$ can force a good (choosing 0) or a bad (choosing 1) block. Then, $(\psi'(a'_{-\alpha}), 0) \not\prec_{p(\alpha)} (\psi'(a'_{-\alpha}), 1)$.
- **Case with at least 3 negations.** Let α be having a negation for f_α . Similarly to the case with one negation, a profile $a' = (a'_{-\alpha}, 1)$ containing $b_i = 0^{j-1}10^{k_i-j}$, is considered. Then, $u'_\alpha(a')$ is 1 or 2. Profile $\psi'(a'_{-\alpha}, 1) = (\hat{b}, \hat{c})$ contains at least two ones in the corresponding B -block. Therefore, $u'_{p(\alpha)}(\psi'(a'_{-\alpha}, 1)) = 0$. A contradiction is obtained.

All f_β in the C -blocks are identities. It is supposed that there is a player $\beta \in C_i$ such

that, f_β is a negation. Let be C_ℓ a block such that $p(\beta) \in C_\ell$. There are two exclusive possibilities for the other players in C_ℓ ,

- there is at least one player γ such that, f_γ is the identity and $p(\gamma) \in C_\ell$,
- all players γ mapped into C_ℓ have functions f_γ corresponding to negations.

The first possibility is considered. An $a'_{-\beta}$ is fixed such that all B -blocks are fixed to 1, and all C -blocks are fixed to 0. Note that, independently of $a'_\beta \in \{0, 1\}$, $c_i \neq b_i$ holds. Since $u'_\beta(a'_{-\beta}, 0) > u'_\beta(a'_{-\beta}, 1)$ it holds that $(a'_{-\beta}, 0) \succ_\beta (a'_{-\beta}, 1)$. Since γ enters C_ℓ and f_γ is the identity, profile $\Psi'(a'_{-\beta})$ corresponding to the team $-p(\beta)$ verifies $b_\ell \neq c_\ell$, independently of the value of the action chosen by $p(\beta)$. Additionally, $u'_{p(\beta)}(\Psi'(a'_{-\beta}, 0)) = u'_{p(\beta)}(\Psi'(a'_{-\beta}, 1)) = 0$ and $u'_{p(\beta)}(\Psi'(a'_{-\beta}, 1)) = 1$. Then, a contradiction is obtained.

Now, the second possibility is considered. A profile $a'_{-\beta}$ is chosen such that all players in B -blocks choose 1 and all players in C blocks choose 0, except player γ choosing 1. The analysis follows in a similar way.

The players bijection p induces a bijection between C -blocks and moreover, this bijection coincides with the bijection for the B -blocks. By π a bijection has been denoted, which is induced by the B -blocks. If this does not hold, there exists an i such that players in C_i are only partially mapped into $C_{\pi(i)}$. Let β be a player mapped into C_ℓ with $\ell \neq \pi(i)$. A profile $a'_{-\beta}$ taken such that, b_i and $(c_i)_{-\beta}$ contains only ones in $(a'_{-\beta}, 1)$, then $b_i = c_i$ and $u'_\beta(a'_{-\beta}, 1) = 2$. As in $(a'_{-\beta}, 0)$ there are $b_i \neq c_i$ and $u'_\beta(a'_{-\beta}, 1) = 1$. Therefore, $(a'_{-\beta}, 0) \prec_\beta (a'_{-\beta}, 1)$. Since, B_i is mapped into $B_{\pi(i)}$, there is no intersection between $B_{\pi(i)}$ and B_ℓ . The profile corresponding to players C_ℓ in $\Psi'(a'_{-\beta}, 0)$ is denoted by \hat{c}_ℓ . Note that in such a profile, the action chosen by player β is 0. Profile $b_{\pi^{-1}(\ell)}$ corresponding to $B_{\pi^{-1}(\ell)}$, has been fixed in $a'_{-\beta}$ such that $\Psi'(b_{\pi^{-1}(\ell)}) = \hat{c}_\ell$. Notice that $u'_{p(\beta)}(\Psi'(a'_{-\beta}, 0)) = 3$, because profiles belonging to B_ℓ and C_ℓ coincide. Note also that $u'_{p(\beta)}(\Psi'(a'_{-\beta}, 1)) = 0$, because profiles in B_ℓ and C_ℓ are different. Profiles for player $p(\beta)$ factor as $\Psi'(a'_{-\beta}, 0) = (\Psi'(a'_{-\beta}), 0)$ and $\Psi'(a'_{-\beta}, 1) = (\Psi'(a'_{-\beta}), 1)$. Therefore, $(\Psi'(a'_{-\beta}), 0) \succ_{p(\beta)} (\Psi'(a'_{-\beta}), 1)$ is obtained and leads to a contradiction.

Let φ_i be a mapping p restricted to B_i . Let $\hat{\varphi}_i$ be a mapping p restricted to C_i . Then, $\varphi_i = \hat{\varphi}_i$ holds. Note that, $\varphi_i : B_i \rightarrow B_{\pi(i)}$ and $\hat{\varphi}_i : C_i \rightarrow C_{\pi(i)}$. It is supposed that, $\varphi_i(r) = s$ and $\hat{\varphi}_i(r) = t$ with $s \neq t$. A profile $a' = (b, c)$ is considered. $b = (b_{-i}, b_i)$ is fixed with $b_i = \text{binify}(r)$, and the b_{-i} filled with zeroes in order to get a bad b . Then, $c = (c_{-i}, c_i)$ with $c_i = \text{binify}(r)$ has been considered. Let β be the player belonging to the C_i block and controlling position r . Then, $a' = (a'_{-\beta}, 1)$. Moreover, $u_\beta(a'_{-\beta}, 1) = 2$ because b is bad, $b_i = c_i$, and $a'_\beta = 1$. Also $u_\beta(a'_{-\beta}, 0) = 1$ because b is bad, $b_i \neq c_i$, and $a'_\beta = 0$. Finally, $(a'_{-\beta}, 1) \succ_\beta (a'_{-\beta}, 0)$. Profile $\Psi'(a') = (\hat{b}, \hat{c})$ is considered. Since B_i is mapped into $B_{\pi(i)}$ we factor $\hat{b} = (\hat{b}_{-\pi(i)}, \hat{b}_{\pi(i)})$ with $b_{\pi(i)} = \text{binify}(s)$. Similarly, $\hat{c} = (\hat{c}_{-\pi(i)}, \hat{c}_{\pi(i)})$ with

$\hat{c}_{\pi(i)} = \text{binify}(t)$. In Ψ' , player β is mapped into $p(\beta)$, controlling position t in $C_{\pi(i)}$. Moreover, $\Psi'(a') = (\Psi'(a')_{-p(\beta)}, 1)$ and $u'_{p(\beta)}(\Psi'(a')_{-p(\beta)}, 1) = 0$. As $u'_{p(\beta)}(\Psi'(a')_{-p(\beta)}, 0) = 1$, $(\Psi'(a')_{-p(\beta)}, 1) \prec_{p(\beta)} (\Psi'(a')_{-p(\beta)}, 0)$ is obtained, and this leads to a contradiction. \square

Lemma 8.8 *Let Γ_1 and Γ_2 be two strategic games. $\Gamma_1 \sim_\ell \Gamma_2$, if and only if $\text{BINARYACTL}(\Gamma_1) \sim_\ell \text{BINARYACTL}(\Gamma_2)$.*

Proof. Let $\Gamma'_1 = \text{BINARYACTL}(\Gamma_1)$ and $\Gamma'_2 = \text{BINARYACTL}(\Gamma_2)$ and let $\Psi' = (p, f_1, \dots, f_k)$ be a local isomorphism between Γ'_1 and Γ'_2 . Let α be a player belonging to a B -block and let β be a player belonging to a C -block. As Ψ' preserves local indifference sets and $|I(\beta)| < |I(\alpha)|$, players α and β cannot be mixed (look at Lemmas 8.4, 8.5, and 8.6). It must happen that $p(\alpha)$ belongs to a B -block in Γ'_2 and $p(\beta)$ belongs to a C -block in Γ'_2 . Moreover, the structure of Ψ' verifies conditions established in Lemma 8.7. Notice that Ψ' induces a permutation π on $\{1, \dots, n\}$. Furthermore, for a player α in position j inside block B_i , mapping $\varphi(j)$ gives us the position of player $p(\beta)$ in block $\pi(i)$.

Let us show that the mapping $\Psi = (\pi, \varphi_1, \dots, \varphi_n)$ is a local isomorphism between Γ_1 and Γ_2 .

It is supposed that $(a_{-i}, r) \prec_i (a_{-i}, s)$ with $r, s \in \{1, \dots, k_i\}$. Then, $u_i(a_{-i}, r) < u_i(a_{-i}, s)$. $a' = (b, c)$ is considered such that $b = \text{binify}(a_{-i}, r)$ and $c = (c_{-i}, c_i)$ with c_{-i} is all filled with zeros and $c_i = \text{binify}(s)$. Let β be the player controlling the 1 in c_i (player β controls the position s in C -block i). Then, it holds that, $u'_\beta(a') = 4 + u_i(a_{-i}, r)$. Factorising player β we have $a' = (a'_{-\beta}, 1)$. Now, player β can change the action from 1 to 0 getting the profile $(a'_{-\beta}, 0)$ such that $u'_\beta(a'_{-\beta}, 0) = 4 + u_i(a_{-i}, s)$. Therefore, there is $(a'_{-\beta}, 1) \prec_\beta (a'_{-\beta}, 0)$. Thus, Ψ' is a local morphism it holds that $\Psi'(a'_{-\beta}, 1) \prec_{p(\beta)} \Psi'(a'_{-\beta}, 0)$.

Now, it is proven that $(\Psi(a_{-i}), \varphi_i(r)) \prec_{\pi(i)} (\Psi(a_{-i}), \varphi_i(s))$ holds. As i is mapped to the $\pi(i)$ block, C_i is mapped to block $C_{\pi(i)}$. Actions for player i are mapped as $\varphi_i(r) = r'$ and $\varphi_i(s) = s'$. The B -blocks are mapped as follows:

$$\Psi'(b) = \Psi'(\text{binify}(a_{-i}, r)) = (\text{binify}(\Psi(a_{-i})), \text{binify}(\varphi_i(r))) = (\hat{b}_{-\pi(i)}, \hat{b}_{\pi(i)})$$

C -blocks are mapped as well, $\Psi'(c) = (\hat{c}_{-\pi(i)}, \hat{c}_{\pi(i)})$ with $\hat{c}_{-\pi(i)}$ filled with zeroes and $\hat{c}_{\pi(i)} = \Psi'(\text{binify}(s)) = \text{binify}(s')$. Note that $p(\beta)$ controls the unique one in $\text{binify}(s')$. Then, $\Psi'(a'_{-\beta}, 1) = (\hat{b}, \hat{c})$ and $u'_{p(\beta)}(\Psi'(a'_{-\beta}, 1)) = 4 + u_{\pi(i)}(\Psi(a_{-i}), \varphi_i(r))$. When player $p(\beta)$ changes its action corresponding to 1 in $\text{binify}(s')$ into a 0, $u'_{p(\beta)}(\Psi'(a'_{-\beta}, 0)) = 4 + u_{\pi(i)}(\Psi(a_{-i}), \varphi_i(s))$ is obtained. Therefore, $\Psi'(a'_{-\beta}, 1) \prec_{p(\beta)} \Psi'(a'_{-\beta}, 0)$, $u_{\pi(i)}(\Psi(a_{-i}), \varphi_i(r)) < u_{\pi(i)}(\Psi(a_{-i}), \varphi_i(s))$ is obtained. Then, the result is concluded.

It is possible then to conclude that the mapping $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ is a local isomorphism between Γ_1 and Γ_2 .

□

8.4 From Local Isomorphism on Binary Action Games to Strong Isomorphism

Then, next step is to transform two binary-action, locally isomorphic games into two strongly isomorphic games.

$\text{FLIPL}(\Gamma) = (N, (A'_i)_{i \in N}, (u'_i)_{i \in N})$ is defined as follows: given a binary action's game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ where $N = \{1, \dots, n\}$ and $A_i = \{0, 1\}$. Actions are $A'_i = \{0, 1\}$ and, for $a' \in A'$ and $i \in N$, $\text{flip}_i(a') = (a'_{-i}, 1 - a_i)$ is defined. Utilities are, \prec_i and \sim_i are defined using u_i in Γ):

$$u'_i(a') = \begin{cases} 2 & \text{if } a' \prec_i \text{flip}_i(a'), \\ 1 & \text{if } a' \sim_i \text{flip}_i(a'), \\ 0 & \text{if } \text{flip}_i(a') \prec_i a'. \end{cases}$$

Example 3 The flip_i function over A' for $i \in \{1, 2\}$ is given by

A'	(0,0)	(0,1)	(1,0)	(1,1)
flip_1	(1,0)	(1,1)	(0,0)	(0,1)
flip_2	(0,1)	(0,0)	(1,1)	(1,0)

Next follows an example of $\text{FLIPL}(\Gamma)$:

		Player 2				Player 2	
		0	1			0	1
Player 1	0	0,1	1,1	Player 1	0	1,1	0,1
	1	0,0	0,1		1	1,2	2,0
Γ				$\text{FLIPL}(\Gamma)$			

Lemma 8.9 If Γ_1 and Γ_2 are two binary action's games. $\Gamma_1 \sim_\ell \Gamma_2$ iff $\text{FLIPL}(\Gamma_1) \sim_s \text{FLIPL}(\Gamma_2)$.

Proof. Let $\Gamma'_1 = \text{FLIPL}(\Gamma_1)$ and $\Gamma'_2 = \text{FLIPL}(\Gamma_2)$.

Given a game mapping $\psi' = (p, f_1, \dots, f_n)$ between Γ'_1 and Γ'_2 the equality $\psi'(\text{flip}_i(a')) = \text{flip}_{p(i)}(\psi'(a'))$ holds. To prove it, note first that as f_i is the identity or a negation $f_i(1 - a_i) = 1 - f(a_i)$ holds. Therefore,

$$\psi'(\text{flip}_i(a')) = \psi'(a'_{-i}, 1 - a'_i) = (\psi'(a'_{-i}), f_i(1 - a'_i)) = \text{flip}_{p(i)}(\psi'(a'))$$

Now, it is assumed that $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ is a local isomorphism between games Γ_1 and Γ_2 . Mapping $\psi' = (p, f_1, \dots, f_n)$ between Γ'_1 and Γ'_2 defined by $p = \pi$ and for all players $f_i = \varphi_i$ is considered. The fact that ψ' is a strong isomorphism will be proven. Let a' be a profile in Γ'_1 and suppose $u'_i(a') = 2$. Other cases are similar. Condition $u'_i(a') = 2$ forces $a' \prec_i \text{flip}_i(a')$. Then ψ is a local morphism and a' is also a profile in Γ_1 . It holds that $\psi(a') \prec_{\pi(i)} \psi(\text{flip}_i(a'))$. Since ψ' behaves identically to ψ , the relationship $\psi'(a') \prec_{p(i)} \text{flip}_{p(i)}(\psi'(a'))$ holds and $u'_{p(i)}(\psi'(a')) = 2$.

It is assumed that $\psi' = (p, f_1, \dots, f_n)$ is a strong isomorphism between games Γ'_1 and Γ'_2 . Let $\psi = (\pi, \varphi_1, \dots, \varphi_n)$ be a copy of ψ' . Note that a local preference in Γ_1 like $a = (a_{-i}, a_i) \prec_i (a_{-i}, 1 - a_i)$ can be rewritten in Γ'_2 (formally $a' = a$) as $a' \prec_i \text{flip}_i(a')$, and $u'_i(a') = 2$. As a result, ψ' is a strong isomorphism and $u'_{p(i)}(\psi'(a')) = 2$ holds, but this forces $(\psi(a_{-i}), \varphi_i(a_i)) \prec_{\pi(i)} (\psi(a_{-i}), \varphi_i(1 - a_i))$. Therefore, ψ is a local isomorphism. \square

8.5 The Complexity of Local Isomorphism

In this section, game transformations defined in the previous sections will be shown which can be computed in polynomial time. This settles the complexity classification of the Locallso problem.

Lemma 8.10 *Let Γ be a game given in explicit form. A description in explicit form of the games $\text{BINARYACT}(\Gamma)$ and $\text{FLIPL}(\Gamma)$ when Γ is a binary actions game, and $\text{CHECKL}(\Gamma)$ when Γ is a binary game, can be computed in polynomial time.*

Proof. Recall that a representation in explicit form of a game is

$$\Gamma = \langle 1^n, A_1, \dots, A_m, (T_{i,a})_{1 \leq i \leq n, a \in A} \rangle.$$

The computation of the representation in explicit form of the games $\text{BINARYACT}(\Gamma)$, $\text{FLIPL}(\Gamma)$, for a given binary actions game Γ , and $\text{CHECKL}(\Gamma)$, for a given binary game Γ will be shown.

Note that in the three constructions, the number of players increases at most polynomially in the size of Γ . Furthermore, for any player, the size of the set of actions is polynomial in the size of the set of actions of players in the original game. In consequence, the number of strategy profiles is polynomial in the size of Γ . Therefore, tabulated utility functions can be computed in polynomial time. \square

Lemma 8.11 *Let Γ be a game given in general form. A description in general form of the games $\text{BINARYACT}(\Gamma)$ and $\text{FLIPL}(\Gamma)$ when Γ is a binary actions game and $\text{CHECKL}(\Gamma)$ when Γ is a binary game, can be computed in polynomial time.*

Proof. When game Γ is given in general form, $\Gamma = \langle 1^n, A_1, \dots, A_m, M, 1^t \rangle$ in all three constructions we have that both the number of players and the size of the sets of actions are polynomial. Now, a Turing machine TM that computes utility functions in each case will be constructed. From the definitions of games, the codification of the TM in polynomial time, given a description of machine M is straightforward. \square

Lemma 8.12 *Let Γ be a game given in formula general form. A description in formula general form of the games $\text{BINARYACT}(\Gamma)$ and $\text{FLIPL}(\Gamma)$ when Γ is a binary actions game and $\text{CHECKL}(\Gamma)$ when Γ is a binary game, can be computed in polynomial time.*

Proof. Now, it will be shown how to compute a description in formula general form of the game $\text{CHECKL}(\Gamma)$, when Γ is a binary game. It is assumed that a game is given as $\Gamma = \langle 1^n, A_1, \dots, A_m, 1^\ell, (\varphi_i)_{1 \leq i \leq n} \rangle$, where $A_i = \{0, 1\}$. In the following table we recall the definition of utility functions of game $\Gamma' = \text{CHECKL}(\Gamma)$, as defined in Page 114, and reformulate its utility functions so as to describe the formula required to compute a bit of the utility value.

Recall that a profile for Γ' has form $c = (a_0, a')$ where $a' = (a_1 b_1, \dots, a_n b_n)$. We also set $a = (a_1, \dots, a_n)$ as original strategy profile in Γ . The range of utility values is $\{0, \dots, 8\}$ and therefore, 4 formulas are needed as there are 9 different utility values, ϕ_i^j , for $0 \leq i \leq n$ and $0 \leq j \leq 3$. In the following table, the utility functions are given, according to definitions, rewritten so to make clear the conditions for each value and the utility value written in binary.

		$u'_i(c)$	ϕ_i^3	ϕ_i^2	ϕ_i^1	ϕ_i^0
Player 0						
		a_0	0	0	0	a_0
Player $i > 0$						
$a_0 = 0 \wedge$ $u_i(a) \neq u_i(a_{-i}, \bar{a}_i)$	$b_i = 0 \wedge u_i(a) = 1$	0	0	0	0	0
	$b_i = 0 \wedge u_i(a) = 0$	1	0	0	0	1
	$b_i = 1 \wedge u_i(a) = 0$	2	0	0	1	0
	$b_i = 1 \wedge u_i(a) = 1$	3	0	0	1	1
$a_0 = 0 \wedge u_i(a) = u_i(a_{-i}, \bar{a}_i) = 0$		4	0	1	0	0
$a_0 = 0 \wedge$ $u_i(a) = u_i(a_{-i}, \bar{a}_i) = 1$	$b_i = 0$	5	0	1	0	1
	$b_i = 1$	6	0	1	1	0
$a_0 = 1$	$b_i = 0$	7	0	1	1	1
	$b_i = 1$	8	1	0	0	0

Using the above table it is easy to write an expression of boolean formulas, for each player, as a disjunction of corresponding minterms. Indeed, these formulas have a constant number of minterms and thus can be written in polynomial time.

For the remaining constructions working in a similar (but slightly more involved) way it can be shown the claimed result. \square

Theorem 8.2 *The LOCALISO problem and the STRONGISO problem are polynomially equivalents for strategic games given in general form, formula general form and explicit form. The equivalence holds even for binary games.*

As a consequence of results in Theorem 8.1 and the previous one, the following result is obtained.

Theorem 8.3 *The LOCALISO problem is polynomially equivalent to*

- *the CIRCUITISO problem, for strategic games given in general form,*
- *the FORMULAISO problem, for strategic games given in formula general form, and*
- *to the GRAPHISO problem, for strategic games given in explicit form.*

The equivalence is also valid for binary games.

As a corollary of the above result we have that the LOCALISO problem belongs to Σ_2^P for games given in general form or formula general form and to NP for games in explicit form. In any of the three cases the problem is not expected to be hard.

8.6 Notes

Game Theory. Refer to *A Course in Game Theory* of M. Osborne and A. Rubinstein [84] published in 1994, for mathematical definitions of strategic and extensive games. Also, refer to the *Theory of Games and Economic Behavior* of John von Neuman and Oscar Morgenstern [81]. Additionally, it is very important to note the definition of *Nash equilibrium* for non-cooperative games in [80] by J. Nash.

Morphisms in Games. Refer to [80] by J. Nash, where a very *compact* definition of game morphisms was first proposed in the field of game isomorphisms. Note the results of J. C. C. McKinsey [72] and J. Harsanyi and R. Selten's isomorphisms of strategic games [49], published in 1988.

Algorithmic Game Theory. Consider the book *Computers and Intractability: A Guide to the Theory of NP-completeness* [45], by M. Garey and D. Johnson. This book features a compendium of NP-complete problems. It is crucial to refer to [36, 35] by J. Feigenbaum and J. Feigenbaum, D. Koller and P. Shor respectively, in order to understand the complexity in game theory. These researches were devoted to the study of connection between game theory and the traditional complexity class. Note also [64], which is devoted to the analysis of the complexity involving in finding max-min strategies in zero-sum games represented in extensive form. Finally, [65, 87, 86] are very important references in algorithmic game theory.

Graphical Games. Examples of graphical representations are the *games networks* of La Mura [78], and *graphical models for game theory* of M. Kearns, M. Littman, and S. Singh [57]. Note also [55] in order to learn about correlated equilibria in graphical games, as a succinct representation for multi-player games. In addition, refer to [28] by C. Daskalakis, P. Goldberg, and C. Papadimitriou. In this work, the authors concern their hardness result for computing a Nash equilibrium and also present an interesting graph showing what players affect other players' payoffs.

Part III

Conclusions and Future Work

Chapter 9

Conclusions and Future Work

This thesis has studied several problems arising from algorithmic game theory and orchestration analysis. The behaviour of orchestrations over unreliable environments has been analysed in the first part. Moreover, the complexity of equilibrium problems for a class of strategic zero-sum games, called *Angel-Daemon* games, has been studied. It has been found established that this class of games provides the first natural family of succinct games for which such complexity results can be established. Additionally, the effect of a number of service failures in Web applications, modelled as orchestrations expressions, has been analysed through this class of games. The second part of this thesis is focused on computational complexity of game isomorphisms and is devoted to the study of how games' representations affect the complexity of problems.

Angel-Daemon Games. The well known characteristic of Grid programming is its dynamicity. The Grid users have significantly less control over resources employed than in traditional scenarios: sites used for the execution of application components may fail. Thus, an important consideration for practical Grid applications is the provision of an assessment of the quality of an application based on the expected performance of its constituent execution sites. In terms of an Orc expression E , used to model the application, an ordered list for $\alpha(E)$ is needed. How this list should be built is unclear and is perhaps a controversial question: the likely behaviour of a site may depend on subjective perceptions of its qualities. In drawing up the list, two distinct classes of considerations may be identified:

- Aspects independent of the application. A “stand-alone” quality of a site has been considered. Taking, also into account the “reputation” of sites [104].
- Aspects depending on the application. The designer has *a priori* knowledge of available potential sites \mathcal{S} . From \mathcal{S} , the designer has to choose $\alpha(E)$. Once $\alpha(E)$ has been determined, the orchestration has to be developed.

In many cases, I show that the development of an application and the rank of $\alpha(E)$ are inextricably linked. For instance, a site used only as a “thread” in a parallel search may fail with little consequence for the application; failure of a site which forms a constituent of a sequential backbone of an application will be catastrophic for the application.

The ordering of $\alpha(E)$ depends also on risk perception. Different people have different perceptions of risk and will rank sites accordingly. For instance, consider a database D with no back-up available. Assume that D is crucial for the application, so that a failure in D significantly harms the application. There are two possibilities for ranking D :

- Moderate optimism. As a failure of D harms the application, an optimistic view will rank D among the angels. In this way the angel will try to avoid having D fail, but if it does fail, then the outcome will fall far short of expectation.
- Safe pessimism. Since D is crucial to the application, D is ranked among the daemons, so that the outcome (likely failure of D), although far from ideal, is at least predictable and uncertainty is removed.

Therefore, failure of Grid sites is a reality of Grid computing and forms a significant part of its challenge. Assessing the likelihood of success of an application requires both an evaluation of the quality of its constituent sites and a means of combining the results to measure the quality of the assembly. This thesis proposes the use of Orc together with game theory as a way of addressing the latter point; the assessment of individual sites and the establishment of a ranking among them remain open questions, touching as they do upon issues such as degree of trust and perception of risk, issues which remain largely subjective.

One of the contributions of this thesis is the question of the management of risk in orchestrations. An *ex-ante* vision of the risk in an uncertainty profile has been captured, applying this idea to two different types of orchestrations. Additionally, the results include a characterisation of the complexity of all problems introduced on Chapter 5, when problems inputs are restricted to be an *Angel-Daemon* games. For doing such an analysis, a standard terminology for classical complexity classes like LOGSPACE, P, NP, coNP, Σ_2^P and EXP has been used.

Furthermore, the problem of deciding the existence of a pure Nash equilibrium or a dominant strategy for a given player has been shown to be Σ_2^P -complete. Moreover, computing the value of an *Angel-Daemon* game, has been shown to be EXP-complete, thus matching the already known complexity results of the corresponding problems for generic families of succinctly represented games with exponential number of actions. Similar results for general families were already known for strategic games in implicit form [9] and succinct zero-sum games [38].

Future Work and Open Questions. Several open questions remain:

- The establishment of a taxonomy or uncertainty profiles for large families of orchestrations could be envisioned.
- It also seems possible to define the behaviour in relation to risk attitudes.
- The behaviour of an orchestration under failures can also be studied using a probabilistic approach [106].
- The relationship between the two approaches also merits investigation.

Game Isomorphisms. The second part of this thesis is focused on the computational complexity of deciding whether two multi-player strategic games are equivalent. For doing such an analysis, three notions of isomorphisms have been introduced, where each of them preserves different structures of a game. *Strong* isomorphisms are defined to preserve the utility functions and Nash equilibria. *Weak* isomorphisms preserve only the player's preference relations and thus preserve only pure Nash equilibria. *Local* isomorphisms preserve only the structure of Nash dynamics. This last notion of isomorphism retains a minimum requirement on partially maintaining the structure of Nash equilibria.

Therefore, one contribution of this thesis, are the following classification of problems:

- The ISISO problem is coNP-complete, for games given in general form, and belongs to NC when games are given in explicit form.
- The ISO problem belongs to Σ_2^P , for games given in general form, and to NP when games are given in explicit form.
- The ISO problem is equivalent to the boolean circuit isomorphism problem, for games in general form, and to the graph isomorphism problem, for games given in explicit form.

The above results hold independently of the type of isomorphism considered, although they depend on the level of succinctness of the description of the input games. Note that the boolean circuit isomorphism problem is believed not to be Σ_2^P -hard [4], and the graph isomorphism problem is conjectured not to be NP-hard [61]. Therefore, the same results are valid for the ISO problem.

Besides the above generic forms of representing games, another particular class of strategic games, which are called *formula games*, has been also considered. The formula games are equivalent in power of representation to a subfamily of the *weighted boolean formula games* introduced in [71]. Furthermore, the complexity of the ISO problem has been analysed, when games correspond to a general form. That is, the number of bits controlled by each player is a constant. For formula games in general form, it has been shown

that the ISO problem is equivalent to the boolean formula isomorphism problem. Recall that the complexity of the boolean formula isomorphism problem is the same as that of circuit isomorphism. However, it is conjectured that both problems are not equivalent.

In this area, a second line of research was to obtain more information about the classification of strategic games with the same number of players, according to the structure of classes induced by the type of isomorphism: pure Nash equilibria. In order to do this analysis, small games were defined. For these games, all equivalence classes have been provided, giving information about the possible structure of Nash equilibria under local isomorphism. In addition, a graphical representation of these classes of equivalence, fulfilling some conditions about preferences, has been proposed.

Under *Local* isomorphism, a first naive approach is to consider games as equivalent if they have the same number of Nash equilibria. The count of pure Nash equilibria has been undertaken via probabilistic analysis by I. Y. Powers [96]. She studied the limit distributions of the number of pure strategies of Nash equilibria for n players' strategic games. Further results can be found in [73]. The sole estimation of the number of pure Nash equilibria is different from *Strong* and *Weak* isomorphisms since, the notions provide a finer classification.

Future Work and Open Question. A further line of interest could be to extend the notion of game isomorphisms to other game families, in particular for strategic games given in implicit form. The main difficulty here is to select a suitable succinct representation of permutations on the set $\{0, 1\}^k$ that are able to represent a morphism. It is expected that the ISO problem for games in implicit form (with utilities given by TM, circuits or formulas) is computationally harder than for the case of games given in general form. Observe that for strategic games in implicit form, the reductions in this work will not longer be computable in polynomial time as the number of strategy profiles will be exponential in the size of the representation.

Another family of interest is extensive games. It would be of interest to study the isomorphism problem for such games avoiding the use of strategic forms. An interesting open question is whether a suitable definition of game isomorphisms for games without *perfect* information can be found.

On the other hand, there are several fields in computer science which develop games, strategic or extensive, that can be used to attain different goals in the Semantic Web. One clear example in this direction are *games with a purpose* approach [5], [105], and [50]. These games are used for instance, to label an image, thus facilitating the acquisition of terms for the semantic Web. Games, strategic or extensive, are used in this approach to learn from the strategic behaviour of the players. Games are defined in such a way that term agreement provides higher utility. Observe that, in this setting, games designed

by different research teams might lead to different definitions of the game corresponding to the label of a single image. To assess the validity of the final results, the equivalence among games should be checked. This might lead to notions of equivalence which differ from those presented in this thesis. It is expected that the results on this work will provide the basis for the analysis of the complexity of equivalence of such games and other Web games.

Finally, it is worth remarking that another area of interest in the field of game theoretical analysis are the graphical models for multi-player games, and the complexity of these classes of concisely represented games [28]. Compared to strategic and extensive representations of games, some graphical models are more structured and compact games' representations, as the case of *game networks* of La Mura [78], and a *Graphical Models For Game Theory* of Kearns and Littman [57].

Overall, this thesis has aimed at complexity study of *Angel-Daemon* games and game isomorphism, which provides that complexity results depends on the problems inputs rather than the classification type of isomorphisms.

Part IV

Appendices

Appendix A

Arranging a Meeting using Reputation

This section is an example of Internet computing where, sites are interpreted as interfaces among the members of a community. Consider a university rector who wishes to consult his staff before making a decision. He sends an email to professors in his university to arrange a meeting. Some professors answer; others do not. Assume that professors have a reputation metrics. For example, reputation may be based on position. After a time ΔT , the rector has to decide if the meeting will take place or not. The decision will be based on the *average reputation of the professors replying*. The Orc expression *Meeting* is based on *MeetingMonitor* (see section 7.3 in [76]). It is assumed that a call to a professor's site p , with a message m , is denoted by $p(m)$. The call $p(m)$, either returns the reputation of the professor or is silent (this is represented as $p(m) > r$). See table A.1.

PROFESSOR	REPUTATION					
Rector	0.3	$r(a, d)$	E	H	P	$r_{\mathcal{A}}(a)$
Ex-Rector	0.3	R	0.13...	0.16...	0.2	0.1
Head of Dept.	0.2	I	0.2	0.23...	0.26...	0.3
Professor	0.1	$r_{\mathcal{D}}(d)$	0.15	0.2	0.25	
Instructor	0.1					
TOTAL	1.0					

Figure A.1: The first table gives the professors' reputations, and the second one reflects the reputation values associated to the strategies in the game associated to the uncertainty profile $\langle \mathcal{P}, \{R, I\}, \{E, H, P\}, 1, 1 \rangle$.

The expression $AskFor(L, m, t, \Delta T)$ gives the number and the total reputation of the answers received. In the expression, $L = (h, t)$ is the list of professors, m is the message, t the suggested meeting time, and ΔT the maximum waiting time for responses. $AskFor$ publishes a pair $(count, total_reputation)$. The average reputation is $r = reputation/count$. As the reputation r_i of any professor i satisfies $0 < r - i < 1$, it holds that $0 < r < 1$. Following [69, 114] a lower $0 \leq \omega \leq 1$ threshold was defined, in order to classify the rel-

evance of data. The meeting will take place when the reputation is good enough: $\omega \leq r$; otherwise, the meeting is cancelled. The final expression is *ProceedOrCancel*, where $L_{\mathcal{P}}$ is the email list of all professors in the university.

$$\begin{aligned}
AskFor([], m, t, \Delta T) &\triangleq let(0, 0) \\
AskFor(H : T, m, t, \Delta T) &\triangleq let(count, total_reputation) \quad \text{where} \\
&\quad count : \in add(u.count, v.count) \\
&\quad total_reputation : \in add(u.reputation, v.reputation) \\
&\quad u : \in \{h(m) > r > let(1, r) \mid Rtimer(\Delta T) \gg let(0, 0)\} \\
&\quad v : \in AskFor(T, m, t, \Delta T)
\end{aligned}$$

$$\begin{aligned}
ProceedOrCancel(c, total_r, \omega) &\triangleq average(c, total_r) > r > \\
&\quad (if(\omega \leq r) \gg let("do")) \mid if(\omega > r) \gg let("cancel"))
\end{aligned}$$

$$Meeting(\mathcal{P}, \omega, t, \Delta T) \triangleq AskFor(L_{\mathcal{P}}, m, t, \Delta T) > (c, r) > ProceedOrCancel(c, r, \omega)$$

Consider how to assess the meeting. To do this, consider a strongly divided university with a set of professors \mathcal{P} such that, $\mathcal{P} = \mathcal{A} \cup \mathcal{D}$. As before, the easy way to analyse this situation is by introducing an uncertainty profile.

Definition A.1 *For the meeting problem, an uncertainty profile contains information about the reputation and the threshold, $\mathcal{U} = \langle \mathcal{P}, (r_i)_{i \in N}, \omega, \mathcal{A}, \mathcal{D}, f_{\mathcal{A}}, f_{\mathcal{D}} \rangle$.*

This profile contains elements in common with those given in definition 4.1, just replacing $\alpha_+(E)$ by \mathcal{P} . Let $a = \{a_1, \dots, a_p\}$ be the failing sites (do not respond) for $\mathfrak{a} \in \mathcal{A}$, and $d = \{d_1, \dots, d_q\}$ the failing sites for $\mathfrak{d} \in \mathcal{D}$ and, as before, the strategy profile is $s = (a, d)$. The set of sites answering the email (successful sites) is, $S_a = \mathcal{A} \setminus a$ and $S_d = \mathcal{D} \setminus d$. Given a strategy profile (a, d) , the average reputations of $\mathfrak{a} \in \mathcal{A}$ and $\mathfrak{d} \in \mathcal{D}$ and the average reputation of the strategy profile are defined as follows,

$$r_{\mathfrak{a}}(a) = \frac{\sum_{s \in S_a} r_s}{\#S_a}, r_{\mathfrak{d}}(d) = \frac{\sum_{s \in S_d} r_s}{\#S_d}, r(a, d) = \frac{\sum_{s \in S_a \cup S_d} r_s}{\#(S_a \cup S_d)}$$

The angel $\mathfrak{a} \in \mathcal{A}$ is happy when the meeting takes place and unhappy otherwise. The

daemon $\mathfrak{d} \in \mathcal{D}$ behaves in the opposite direction.

$$u_{\mathfrak{a}}(a, d) = \begin{cases} +r_{\mathfrak{a}}(a) & \text{if } \omega \leq r(a, d) \\ -r_{\mathfrak{a}}(a) & \text{otherwise} \end{cases} \quad u_{\mathfrak{d}}(a, d) = \begin{cases} -r_{\mathfrak{d}}(d) & \text{if } \omega \leq r(a, d) \\ +r_{\mathfrak{d}}(d) & \text{otherwise} \end{cases}$$

Based on these utilities, the *angel-daemon* game is adapted to this situation and, we obtain the strategic game $\Gamma(\mathcal{U}) = \langle \mathcal{A}, \mathcal{D}, u_{\mathfrak{a}}, u_{\mathfrak{d}} \rangle$ called the Meeting game. Note that the Meeting game is not a zero-sum game.

Example A.1 A university with $N = 5$ professors $\mathcal{P} = \{R, E, H, P, I\}$ partitioned as $\mathcal{A} = \{R, I\}$, $\mathcal{D} = \{E, H, P\}$ and $f_{\mathcal{A}} = f_{\mathcal{D}} = 1$. Several examples of the meeting game for different values of w are given in the Figure A.2. Observe that, for $w = 0.1, 0.18$ and 0.3 , the meeting game has a pure Nash equilibrium. However, when $w = 0.25$, the game does not have a Nash equilibrium.

		\mathfrak{d}					\mathfrak{d}		
		E	H	P			E	H	P
\mathfrak{a}	R	0.1, -0.15	0.1, -0.2	0.1, -0.25	\mathfrak{a}	R	-0.1, 0.15	-0.1, 0.2	0.1, -0.25
	I	0.3, -0.15	0.3, -0.2	0.3, -0.25		I	0.3, -0.15	0.3, -0.2	0.3, -0.25
		$w = 0.1$					$w = 0.18$		
		\mathfrak{d}					\mathfrak{d}		
		E	H	P			E	H	P
\mathfrak{a}	R	-0.1, 0.15	-0.1, 0.2	-0.1, 0.25	\mathfrak{a}	R	-0.1, 0.15	-0.1, 0.2	-0.1, 0.25
	I	-0.3, 0.15	-0.3, 0.2	0.3, -0.25		I	-0.3, 0.15	-0.3, 0.2	-0.3, 0.25
		$w = 0.25$					$w = 0.3$		

Figure A.2: Some examples of the meeting game for the case in Example A.1, for different values of w . That is, $w \in \{0.1, 0.18, 0.25, 0.3\}$.

Theorem A.1 Given $\mathcal{U} = \langle \mathcal{N}, (r_i)_{i \in N}, \omega, \mathcal{A}, \mathcal{D}, f_{\mathcal{A}}, f_{\mathcal{D}} \rangle$, let players $\mathfrak{a} \in \mathcal{A}$ and $\mathfrak{d} \in \mathcal{D}$, and $A = A_{\mathfrak{a}} \times A_{\mathfrak{d}}$. Define $\delta = \min_{(a,d) \in A} r(a, d)$, $\mu = \max_{(a,d) \in A} r(a, d)$, $\delta_{\mathfrak{a}} = \min_{a \in A_{\mathfrak{a}}} r_{\mathfrak{a}}(a)$ and finally, $\mu_{\mathfrak{a}} = \max_{a \in A_{\mathfrak{a}}} r_{\mathfrak{a}}(a)$.

- $\Gamma(\mathcal{U})$ has a Nash equilibrium iff either (1) $\exists a \in A_{\mathfrak{a}}$ with $r_{\mathfrak{a}}(a) = \mu_{\mathfrak{a}}$ such that $\forall d \in A_{\mathfrak{d}} r(a, d) \geq \omega$, or (2) $\exists d \in A_{\mathfrak{d}}$ with $r_{\mathfrak{d}}(d) = \mu_{\mathfrak{d}}$ such that, $\forall a \in A_{\mathfrak{a}} r(a, d) < \omega$.
- If $\omega \leq \delta$ or $\omega > \mu$, game $\Gamma_{\mathcal{P}}(\mathcal{U})$ has a Nash equilibrium.

Furthermore, in any Nash equilibrium (a, d) of the game (if one exists), either the meeting holds and $u_{\mathfrak{a}}(a, d) = \mu_{\mathfrak{a}}$ and $u_{\mathfrak{d}}(a, d) = -\delta_{\mathfrak{d}}$ or the meeting is cancelled and $u_{\mathfrak{a}}(a, d) = -\delta_{\mathfrak{a}}$ and $u_{\mathfrak{d}}(a, d) = \mu_{\mathfrak{d}}$.

Proof. Observe that function r has some monotonicity properties. In the case that $r(a, d) \geq \omega$, for any $a' \in A_{\mathfrak{a}}$ with $r_{\mathfrak{a}}(a') \geq r_{\mathfrak{a}}(a)$, we have that $r(a', d) \geq \omega$. Let (a, d) be a Nash equilibrium, in which the meeting holds. That is, $r(a, d) \geq \omega$. This means that $r_{\mathfrak{a}}(a_1) \geq 0$

and $r_\delta(d_1) \leq 0$. Therefore, taking into account the monotonicity, we have that $r_a(a_1) = \mu_a$ and $r_\delta(d_1) = \delta_\delta$, otherwise a_1 will not be a best response to d_1 and *vice versa*. But in such a case, condition (1) holds. In the case that condition (1) holds, we have that a is the best response to any action of the daemon d' . Furthermore, the best response to a happens in the daemon strategy with minimum r_δ value. Therefore the game has a Nash equilibrium in which the meeting holds. Symmetrically, if $r(a, d) < \omega$, for any $d' \in A_\delta$ with $r_\delta(d') \leq r_\delta(d)$, we have that $r(a, d') < \omega$. Reversing the arguments, we obtain the equivalent in which Nash equilibrium exists and the meeting does not hold and condition (2).

Observe that $\omega \leq \delta$ implies condition (1) and $\omega > \mu$ implies condition (2). □

Appendix B

IT System example

Organisations manage and process large amounts of information, and use automated information technology (IT) systems to process it. Risk assessment is also used to determine the extent of the potential threats and the risks associated with an IT system, so that, risk assessment is the first process in the risk management methodology [1].

Risk is a function of the *likelihood* of a given threat-sources exercising a particular potential vulnerability, and the resulting impact of that adverse event on the organisation.

There are many vulnerability studies and their causes, associated to companies, organisations, and IT system environments. In fact, there are institutes ¹ and organisations dedicated to management and assessments of risks. There are also tools and techniques for managing organisational risk, because it plays a critical role in protecting an organisation's information, and therefore its mission, from IT-related risk.

Risk management involves three processes: risk assessment, risk mitigation, and evaluation and assessment. These processes include identification of the potential threat, setup of appropriate controls for reducing or eliminating risks, as well as prioritisation, evaluation and implementation of the appropriate risk-reducing controls recommended from the risk assessment process. Therefore, security protocols ² have been designed and implemented to minimise the effects of potential threats.

In this section an example about the risk assessment will be given, through the IT system implemented at the GABRMN group, and from the game theory viewpoint.

¹<http://www.theirm.org/>

²<http://www.nist.gov/index.html>

GABRMN IT System. The GABRMN group³ manages different databases that give service to the international scientific community. These databases contain relevant clinical data from brain tumour research, which have been the result of previous national and European projects. The group also processes large amounts of biological data. Mainly, these are magnetic resonance spectroscopy data, both in-vivo and ex-vivo, from clinical (human) and preclinical (mouse and rat) models. Therefore, the conservation of all information and data of the group is utmost importance. These data are constantly being acquired, therefore, it is necessary to perform backups at regular intervals, and to have a large storage capacity available.

A well designed IT system must take in mind an estimation of the likelihood of success of all their hosted services and applications. Therefore, a particular IT system has been modelled through expressions of orchestrations and its behaviour, over unreliable environments. Thus, an analysis based on an alternative approach via game theory has been performed, where games have been used to risk asses the IT system.

An identification of potential risks to IT systems is one of the main tasks in risk assessment. Risks occur when vulnerabilities in the IT system or its environment can be exploited by natural, human, or environmental threats.

Likelihood Determination and Risk Impact Analysis. To obtain an overall likelihood rating, in order to indicate the probability that a possible vulnerability may be exercised within the associated threat environment, the following factors must be considered: threat motivation and capability, nature of the vulnerability, and existence and effectiveness of current controls. The likelihood can be described as high, moderate, and low, as they been previously described in Table B.1.

Likelihood Level	Likelihood Definition
High	0-25% chance of successful exercise of threat during a one year.
Moderate	25-75% chance of successful exercise of threat during a one year.
Low	75-100% chance of successful exercise of threat during a one year.

Figure B.1: The likelihood that a potential vulnerability could be exercised by a given threat can be described as high, moderate, or low.

Identifying risk for an IT system requires understanding the system's processing environment. Therefore, to conduct a risk assessment, system-related information must be

³Aplicacions Biomèdiques de la Ressonància Magnètica Nuclear (GABRMN), que pertany a l'Unitat de Bioquímica i el Departament de Biologia Molecular ubicat a la Facultat de Biociència de l'Universitat Autònoma de Barcelona (UAB), accessible des de <http://gabrmn.uab.es>.

collected, usually classified as: hardware, software, system interfaces, data and information among others. On the other hand, it is necessary to bear in mind the most common threats for an IT system like floods, earthquakes, electrical storms, network based attacks, malicious software upload, unauthorised access to confidential information, long term power failure, pollution, and chemicals.

In this example, hardware components and softwares have been considered as threats. Technical failures in hardware components (HW) such as memory dimms, networks, and HDD can lead to system instability as well as its partial or catastrophic failure. Therefore, the set of hardware components considered are defined in general as $HW = \{h_1, \dots, h_p\}$, where each h_p is considered a hardware component.

The other threat to be considered is the softwares' set. That is, the necessary software to the proper behaviour of servers, for instance: Operating System (OS), Tomcat Apache, Php, Mysql, and also database applications, Mail service, Dirvish, Subversion. Therefore, the set of software is defined as $SW = \{s_1, \dots, s_i\}$, where s_i corresponds to each installed application. See Example B.1 below.

Example B.1 [Failures services] *Suppose that a server hosts several database applications: one Oracle database and two MySQL databases over a Linux system. There are some services such as the Operating System (OS), Oracle, Apache Tomcat, MySQL and Php behind these database applications. Failure of one service will affect other services. Suppose that the Oracle service fails. This failure causes the failure of the Oracle databases, however the MySQL databases remain active. Moreover, if an hdd fails, then the server may fail and a full crash must be considered. But if any memory Dimm fails, the system will be unstable and recovery is possible by replacing the failing memory Dimm.*

Any of the possible selected threats can cause failure to the provided services. Therefore, an Orc expression was defined. This expression asks first if hardware devices are alive, in order to ask next whether the needed services are *alive* or not,

$$AskForAlive(h_1, \dots, h_p, t) \triangleq let(x_1, \dots, x_n) > (x_1, \dots, x_n) > if(x_1 \wedge \dots \wedge x_n)$$

where

$$x_1 := (h_1 \ggg let(true)|RTimer(t) \ggg let(false))$$

⋮

$$x_n := (h_p \ggg let(true)|RTimer(t) \ggg let(false))$$

In this expression, *AskForAlive* publishes a tuple (x_1, \dots, x_n) of true values if all *hardware* devices (h_1, \dots, h_p) are *alive*.

$$\begin{aligned}
& AskForAlive(b_1, \dots, b_n, t) \triangleq AskForAlive(h_1, \dots, h_p, t) \gg \\
& let(x_1, \dots, x_n) > (x_1, \dots, x_n) > if(x_1 \wedge \dots \wedge x_n) \\
& where \\
& x_1 : \in (b_1 \gg let(true)|RTimer(t) \gg let(false)) \\
& \vdots \\
& x_n : \in (b_n \gg let(true)|RTimer(t) \gg let(false))
\end{aligned}$$

Here, *AskForAlive* expression publishes a tuple (x_1, \dots, x_n) of true values, if all basic services (b_1, \dots, b_n) are *alive*.

The relevant data to brain tumour research managed and served by the group are stored in different databases. These data are constantly being acquired, for which it is necessary to have backups and storage capacity. The databases were designed in MySQL and Oracle. Also, some applications for the group's research are stored on an applications' server. Besides these servers, in order to maintain the proper functioning of the research group, there are the *Proxy* and, also the mail server. Therefore, the IT system is composed of a few servers, which host the different services provided.

- The databases server:

$$SW(Databases) = \{OS, Tomcat, MySQL, Apache, Oracle, Php, eT, eT_f, uabDB, I, I_f\}$$

- The backups server:

$$SW(Backups) = \{OS, Rsync, Apache, OpenSSH, Dirvish, Subversion, NXFree\}$$

- The applications server:

$$SW(Applications) = \{OS, Java, Apache, OpenSSH, LcModel, IDL, SC, Definiens, NXFree\}$$

- The Firewall with Proxy:

$$SW(Proxy) = \{OS, Firewall, Apache, Drupal, MySQL, Php, WebGroup\}$$

- The Mail and Mirror servers:

$$BS(Mail) = BS(Mirror) = \{OS, Postfix, Courier, Spamassassin, Amavis, Php, Apache, MySQL, Mail service, Squirrelmail, Postfixadmin\}$$

B.1 Different Failure Scenarios

It has been shown that the IT system in this example is composed by independent servers with their own characteristics of *hardware components* and *software*. The correct behaviour of the system depends first on the *Proxy* server. This server is the face of all

servers, and all applications and databases are accessible through this one. If this server fails, the entire system will fail. On the other hand, the proper behaviour of the mail service depends on the synchronisation and orchestration of both involved nodes: *Mail* and *Mirror*. Thus, the system *IT – system* was defined by an expression Orc as follows:

$$IT - system \triangleq Proxy \gg [Mail \gg Mirror] | Applications | Backups | Databases$$

Outputs (out) of the Orc expression were considered. The number of outputs published by the expression measure the benefit of the expression itself. One simple way to analyse the Orc expression was to consider whether each site on the expression was alive or not. Then, the maximum number of outputs would be $out(IT - system) = 5$. In case that one server failed, for instance the *Applications* server, the output of the expression would be $out(IT - system) = 4$. Moreover, if the *Proxy* server failed, the other nodes would be inaccessible and then, $out(IT - system) = 0$.

In this way, the importance of the *Proxy* server for the IT system is easy to understand, knowing that is crucial to mitigate any threat to the *Proxy* server. Since a *Proxy*'s failure means the whole system's failure. Moreover, another way to analyse the IT system's behaviour under failures, by defining the estimated impact for each risk, and the impact rating assigned to the risk.

First Possible Scenario. For such an analysis, an *angel-daemon* game associated to an Orc expression, which models the considered system *IT – system*, was provided.

An *uncertainty profile* defined in [39] as $\mathcal{U} = \langle E, A, D, f_A, f_D \rangle$, were considered. In this expression, E is the *IT – system* expression. \mathcal{A} and \mathcal{D} , are the angel and the daemon players respectively, where f_A and f_D are their own failures, such that: $f_A \leq A$ and $f_D \leq D$.

A strategic game of *uncertainty profile* \mathcal{U} was considered, that is $\Gamma(U) = \langle N = \{a, d\}, A_a, A_d, u_a, u_d \rangle$, where $N = \{a, d\}$ are the angel and the daemon players. The player actions are A_a and A_d , selected from nodes composing the system. The strategic profile $s = (a, d)$ is a set of failures associated with *IT – system*. The utilities are defined as $u_a(s) = out(fail_{a \cup d}(E))$ and $u_d(s) = -u_a(s)$.

In order to construct a strategic game, the angel and the daemon sets were defined. These sets must be selected between nodes composing the IT system. To assign the actions (servers) to each player, the system administrator's opinion about the behaviour of processes and applications inside each server was considered. The system administrator's opinion must be based on whether the system will have recovery capacity or not, as well as on the cost of the recovery process or the mitigation of vulnerabilities. Thus, two actions set for both players were defined.

By nature, the Angel modifies the behaviour of a bounded number of "angelic" services in a beneficial way (e.g. to model service elasticity, or network resilience), while

a *daemon* player modifies a bounded number of “daemonic” services in a negative way (e.g. to model the effects of over-demand), *Angel* = {*Mail*, *Mirror*, *Applications*} and *Daemon* = {*Databases*, *Backups*, *Proxy*}.

Risk Determination. To assess the risk level to the IT system, the risk determination is derived by multiplying the ratings assigned to threat likelihood and threat impact. The rationale for this justification can be explained in terms of the probability assigned to each threat is likelihood level and the value assigned to each impact level, to assess the level of risk to the IT system, as shown in Figure B.2.

Risk Impact			
Risk Likelihood	Low (10)	Moderate (50)	High (100)
High (1.0)	Low (10 * 1.0 = 10)	Moderate (50 * 1.0 = 50)	High (100 * 1.0 = 100)
Moderate (0.5)	Low (10 * 0.5 = 5)	Moderate (50 * 0.5 = 25)	High (100 * 0.5 = 50)
Low (0.1)	Low (10 * 0.1 = 1)	Moderate (50 * 0.1 = 5)	High (100 * 0.1 = 10)

Figure B.2: This table shows the overall risk levels: high, medium, and low.

Assign a risk rating to each server, in the IT system, with their own *hardware* components and *softwares*. The data entry in the following table is derived from the table B.2. Then, the overall risk rating was defined as shown in the Table B.3.

A failure in the *Proxy* is regarded as non-accessibility to all applications and services. In addition, the proper behaviour of the *Mail* servers and the *Mirror* has been assumed as a single publication, since the *Mirror* is activated only if the other is down. Then, a strategic game was obtained, as shown in Figure B.4.

Overall Risk Ratings			
Node (Risk)	Risk Likelihood (Rating)	Risk Impact (Rating)	Risk Overall (Rating)
Proxy	1.0	100	100
Databases	1.0	100	100
Mail	1.0	100	100
Mirror	1.0	10	10
Applications	0.5	100	50
Backups	0.5	50	25

Figure B.3: Overall Risk Ratings.

		Daemon		
		Databases	Backups	Proxy
Angel	Mail	4 / 100	4 / 25	0 / 100
	Mirror	5 / 100	5 / 25	0 / 100
	Applications	4 / 100	4 / 50	0 / 100

Figure B.4: Strategic game for IT – system expression.

This game has three pure Nash equilibria on the strategic profiles, which consider to *Proxy* server. What does this mean? Precisely, the *Proxy* is the “door” of the system. A failure on this server does not allow access to the rest of the IT system. Therefore, the *Proxy* server is always subject to possible threats, being a point of access to the whole system.

For each strategic profile, table B.4 shows two values: first, the number of system outputs. Second, the overall Risk, where the values were calculated as a public service for strategic profiles. This calculation is explained through an example. The strategic profile (*Mail, Database*) has been taken. The *Overall Risk (OR)* in case of *Mail* server failures is $OR(Mail) = RiskLikelihood(RL) * RiskImpact(RI) = 1.0 * 100 = 100$, but since it is known that *Mirror* server was active, therefore, the true Risk Impact (RI) for *Mail* server decrease to a value of 10. However, the Overall Risk for *Databases* server, $OR(Databases) = RiskLikelihood(RL) * RiskImpact(RI) = 1.0 * 100 = 100$. Added the values of the Overall Risk, the following is obtained $SUM(OR(Mail), OR(Databases)) = SUM(10, 100) = 110$.

This means that when deciding the strategic profile (*Mail, Database*), the IT system becomes vulnerable and failures have high risk of impact. It is very important to correct this situation immediately. This can be achieved having a mirror to the mail server, which solves the problem immediately, because it automatically activates the mirror, minimising the risk impact. However, the problem will persist for the system manager. The analysis of failures in the server database is similar. It may be catastrophic and, above all, with high risk of impact. In addition, correcting a failure or some type of threat in a database could be a lengthy and costly process.

Second Possible Scenario. Another possible scenario has been considered, with the following action sets for the Angel and the Daemon players, $Angel = \{Mail, Applications, Proxy\}$ and $Daemon = \{Mirror, Databases, Backups\}$. Figure B.5 shows the corresponding strategic game. This game has not PNE. What conclusion can be obtained? The analysis is similar to the previous one. In this game, as seen above, a failure in any strategy profile is critical, with respect to the *Proxy* server. A failure is also critical for the strategic profiles (*Mail, Databases*) and (*Mail, Mirror*). Correcting the problem for both strategic profiles may be costly and complex, with a

high impact for system users.

		Daemon		
		Mirror	Databases	Backups
Angel	Mail	4/200	4/200	4/35
	Applications	4/60	3/150	3/75
	Proxy	0/110	0/200	0/125

Figure B.5: Another strategic game for *IT – system*. This game has not PNE.

Third Possible Scenario. For the third possible scenario, the action's sets for the Angel and the Daemon players has been partitioned as follows: $Angel = \{Mail, Databases, Applications\}$ and $Daemon = \{Mirror, Backups, Databases\}$. The *angel-daemon* game obtained considers a slightly different scenario. However it is essentially the same situation as the above scenarios. The overall Risks obtained for each strategic profile are similar in all cases. For instance, in these strategic profiles where there is a *Proxy* server, the overall Risk is maximum and the system became non-accessible. Moreover, faults in the other server profile strategy might complicate IT system recovery and threat mitigation.

		Daemon		
		Mirror	Backups	Proxy
Angel	Mail	4/200	4/35	0/110
	Databases	4/110	3/125	0/200
	Applications	4/60	3/75	0/150

Figure B.6: Another strategic game for *IT – system*. This AD game, like the game in Figure B.4, has 3 pure Nash equilibria. The strategic profiles consider the Proxy as Daemon's action.

Conclusions. The *angel-daemon* games designed for the three scenarios, consider whether each node is active or not. This implies a very simple model, which is especially unrealistic, because there are always several applications running at the same time in a single node. Some of them may be active, while others may not be available, which would result in partial performance of the IT system. This scenario is not taken into account under the point of view of this example.

It would be more realistic to create AD games for each node and to define their uncertainty profiles. Then, for every strategic game, their mixed strategies should be calculated in order to obtain the players' utilities of each strategic profile of the whole game, therefore, modeling an entire IT system.

Appendix C

Small Games. Graphic Representation

C.1 Local Nash Isomorphism

In Chapter 8 the local notion of isomorphism has been introduced. This local isomorphism preserves preferences defined only on the “close” neighborhood of strategy profiles. More precisely, as it was already showed *local* isomorphisms only preserve pure Nash equilibria [41].

Definition C.1 [Local isomorphism]. A local isomorphism $\psi : \Gamma \rightarrow \Gamma'$ is a mapping ψ , such that for any triple a, a' and i , such that $a_{-i} = a'_{-i}$, $a \prec_i a'$ iff $\psi(a) \prec_{\pi(i)} \psi(a')$ and $a \sim_i a'$ iff $\psi(a) \sim_{\pi(i)} \psi(a')$ are verified. When Γ and Γ' are locally isomorphic it is noted as $\Gamma \sim_\ell \Gamma'$. In the particular case that Γ' is Γ , a weak isomorphism is named a local automorphism.

Here, only preferences $a \preceq_i a'$ have been considered, such that $a_{-i} = a'_{-i}$ are preserved for any player. It is easy to see that local isomorphisms preserve pure Nash equilibria.

Example C.1 An example of local isomorphic games is provided. Games Γ and Γ' are given.

$$\begin{array}{c}
 \begin{array}{cc}
 & l & r \\
 t & \begin{array}{|c|c|} \hline 0,0 & 0,1 \\ \hline \end{array} \\
 b & \begin{array}{|c|c|} \hline 1,1 & 1,0 \\ \hline \end{array} \\
 \Gamma & &
 \end{array}
 \xrightarrow{\psi}
 \begin{array}{c}
 \begin{array}{cc}
 & l' & r' \\
 t' & \begin{array}{|c|c|} \hline 3,3 & 1,0 \\ \hline \end{array} \\
 b' & \begin{array}{|c|c|} \hline 2,3 & 2,1 \\ \hline \end{array} \\
 \Gamma' & &
 \end{array}
 \end{array}$$

A mapping $\psi = (\pi, \varphi_1, \varphi_2)$ is also given, such that $\pi = (1 \rightarrow 2, 2 \rightarrow 1)$, $\varphi_1 = (t \rightarrow r', b \rightarrow l')$, and $\varphi_2 = (l \rightarrow t', r \rightarrow b')$. It can easily be checked that ψ is a local isomorphism. For game Γ , $(t, l) \prec_1 (b, r)$ and $(t, l) \sim_2 (b, r)$, but for game Γ' , $\psi(t, l) \prec_{\pi(1)} \psi(b, r)$ and $\psi(t, l) \prec_{\pi(2)} \psi(b, r)$. Therefore, the isomorphism is not weak.

Given two strong isomorphic games $\Gamma \sim_s \Gamma'$, it holds that Γ and Γ' are weakly isomorphic, $\Gamma \sim_w \Gamma'$, because the second one is less restrictive than the first one. Since local isomorphisms are always less restrictive than weak isomorphisms, the following holds:

Lemma C.1 *Given two games Γ and Γ' , the following holds:*

- if $\Gamma \sim_s \Gamma'$, then $\Gamma \sim_w \Gamma'$.
- if $\Gamma \sim_w \Gamma'$, then $\Gamma \sim_\ell \Gamma'$.

However, the implication is not true in the opposite sense. If two games Γ and Γ' are locally isomorphic, it does not mean that they are also weakly isomorphic.

Local Nash Isomorphism. A local Nash isomorphism will be considered, as a particular case of local isomorphisms with more restrictive conditions. This case has the minimum requirement that the structure of Nash equilibria should be partially preserved.

Definition C.2 [Local Nash isomorphism]. *A local Nash isomorphism is a mapping ψ , such that for any strategy profile a in Γ , $a \in \text{PNE}$ iff $\psi(a) \in \text{PNE}$.*

Example C.2 *Given two games Γ_1 and Γ_2 , such that there is an isomorphism under a local Nash isomorphism, but not under a local isomorphism,*

	l	r		l'	r'	
t	1,1	0,0		t'	0,0	0,0
b	0,0	1,1		b'	0,0	1,1
	Γ_1			Γ_2		

where utilities corresponding to the Nash equilibria are boldfaced.

Moreover, it is possible to write $\Gamma \sim_{\ell N} \Gamma'$, when two games Γ and Γ' are Nash isomorphic locally.

Note that, different morphisms are just refinements: $\sim_s \implies \sim_w \implies \sim_\ell \implies \sim_{\ell N}$. That is, $\Gamma \sim_s \Gamma'$, $\Gamma \sim_w \Gamma'$, $\Gamma \sim_\ell \Gamma'$, $\Gamma \sim_{\ell N} \Gamma'$. But this is not true for the number of Nash equilibria ($\sim_{\#N}$), $\Gamma \sim_{\#N} \Gamma'$.

C.2 Equivalence Classes of Games

This section is focused in classifying games considering their structure as pure Nash equilibria. In order to have a classification of strategy games with the same number of players according to the structure of pure Nash equilibria, a first naive approach is to consider

games as equivalent, if they have the same number of Nash equilibria. Just counting pure Nash equilibria is different from a Local Nash isomorphism, as it can be seen in the following example.

Example C.3 Two two-player games Γ and Γ_1 are given. Both of them have two pure Nash equilibria. But they are not isomorphic under a local Nash isomorphism,

	l	r		l'	r'	
t	1,0	0,0		t'	1,0	0,0
b	1,0	1,1		b'	0,1	0,0
	Γ_3			Γ_4		

where utilities corresponding to Nash equilibria are boldfaced.

Given two weak games Γ and Γ' , it is possible to define an identity mapping between both games, so that these games are defined as weakly-identical.

Definition C.3 Two games Γ and Γ' are called weakly-identical, iff the identity mapping is a weak isomorphism between both games. It is possible to write $\Gamma \sim_w^I \Gamma'$.

Therefore, by definition C.3 if $\Gamma \sim_w^I \Gamma'$, Γ and Γ' games are taken as two “representations” of the same game. Thus, players and preferences are the same in both games, but preferences are encoded with different utility values. The encoding $\langle u \rangle = (\langle u_1 \rangle, \dots, \langle u_n \rangle)$ in Γ has a well defined block structure because, for each $1 \leq i \leq n$, the block corresponding to the utilities of player i is $\langle u_i \rangle = \langle u_i(0), \dots, u_i(k-1) \rangle$. This block structure is used to consider $\langle u_i \rangle$ in Γ and $\langle u'_i \rangle$ in Γ' . In the following lemma, both blocks have the same number of different values, as it will be proven.

Lemma C.2 Game Γ is given. Let n be the number of players, and k the number of strategy profiles, and for $0 \leq i \leq n$, $0 < \delta_i < k$ exists, such that for any $\Gamma' \sim_w^I \Gamma$ it holds that $\delta_i = \#\{u'_i(a) | a \in A\}$.

Proof. Given a game Γ , the pre-order induced by u_i is noted as \prec_i and \sim_i , that is $s \prec_i s'$ iff $u_i(s) < u_i(s')$, and $s \sim_i s'$ iff $u_i(s) = u_i(s')$. Let $s_1 r_i^1 s_2 r_i^2 \dots s_l r_i^l \dots r_i^{k-1} s_k$ be the ordering of all the strategy profiles belonging to A , where relations $r_i^j \in \{\prec_i, \sim_i\}$ are based on u_i . Let δ_i the number of different values in the preceding chain. Since $\Gamma' \sim_w^I \Gamma$, the identity is a weak isomorphism. Also, the utilities of Γ and Γ' verify $u_i(s) < u_i(s')$ iff $u'_i(s) < u'_i(s')$ and $u_i(s) = u_i(s')$ and $u'_i(s) = u'_i(s')$. Let \hat{r} be the relationship based in u'_i , it holds that $s r_i s'$ is equivalent to $s \hat{r}_i s'$ and therefore the number of different values in the chain $s_1 \hat{r}_i^1 s_2 \hat{r}_i^2 \dots s_l \hat{r}_i^l \dots \hat{r}_i^{k-1} s_k$ is also δ_i . □

The lexicographic order of games Γ and Γ' is such that $\Gamma \sim_w^J \Gamma'$ and $\Gamma \neq \Gamma'$ is taken. This order is based on the block structure taken by game encoding. Both games have the same number of players and the same set of actions. Consequently, the encoding of Γ and Γ' differs only in the encoding of utilities, that is $(\langle u_1 \rangle, \dots, \langle u_n \rangle) \neq (\langle u'_1 \rangle, \dots, \langle u'_n \rangle)$. The leftmost player i , such that $\langle u_i \rangle \neq \langle u'_i \rangle$ is considered. Both utilities are expanded as follows: $\langle u_i \rangle = \langle u_i(0), \dots, u_i(k-1) \rangle$ and $\langle u'_i \rangle = \langle u'_i(0), \dots, u'_i(k-1) \rangle$. The leftmost profile l , such that $u_i(l) \neq u'_i(l)$ is taken. It is defined $\Gamma <_{\text{lex}} \Gamma'$ if $u_i(l) < u'_i(l)$. Otherwise $\Gamma >_{\text{lex}} \Gamma'$. In case that it is not possible to find such i and l , both games coincide and $\Gamma =_{\text{lex}} \Gamma'$ is defined.

Definition C.4 A game Γ is given. Then,

- $\text{minWeak}(\Gamma) = \text{lexMin}\{\Gamma' \mid \Gamma' \sim_w^J \Gamma\}$ is defined as a weakly-identical game with lexicographically minimal game encoding.
- $\text{minLocal}(\Gamma) = \text{lexMin}\{\Gamma' \mid \Gamma' \sim_\ell^J \Gamma\}$ is defined as a local-identical game with lexicographically minimal game encoding.

The following lemma shows that, the $\text{minWeak}(\Gamma)$ is the encoding of Γ , using values as small as possible.

Lemma C.3 Given a game $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$, the pre-orders induced by u_i are \prec_i and \sim_i , that is $s \prec_i s'$ iff $u_i(s) < u_i(s')$, and $s \sim_i s'$ iff $u_i(s) = u_i(s')$. Then, the following conditions are equivalent:

1. $\Gamma' = \text{minWeak}(\Gamma)$.
2. $\Gamma' = (N, (A_i)_{i \in N}, (u'_i)_{i \in N})$. Suppose $|A| = k$ and let $s_1 r_i^1 s_2 r_i^2 \dots s_l r_i^l \dots r_i^{k-1} s_k$ be an ordering of all strategy profiles belonging to A . Relations $r_i^j \in \{\prec_i, \sim_i\}$ are based on u_i . The utility u'_i of player i in Γ' verifies $u'_i(s_1) = 0$, and for $0 < l < k$, $u'_i(s_{l+1}) = u'_i(s_l)$ if $s_l \sim_i s_{l+1}$, and $u'_i(s_{l+1}) = 1 + u'_i(s_l)$ if $s_l \prec_i s_{l+1}$.
3. $\Gamma' = (N, (A_i)_{i \in N}, (u'_i)_{i \in N})$, such that $\Gamma' \sim_w^J \Gamma$ and, for any $i \in N$, $u'_i(a) \in \{0, \dots, \delta_i - 1\}$, where $\delta_i = \#\{u_i(a) \mid a \in A\}$.

Proof. The fact that point (1) implies (2), will be proven next. Let $s_1 r_i^1 s_2 r_i^2 \dots r_i^l s_{l+1} r_i^{l+1} s_{l+2} \dots r_i^{k-1} s_k$ be the pre-order obtained by (2) and $s_1 \hat{r}_i^1 s_2 \hat{r}_i^2 \dots \hat{r}_i^l s_{l+1} \hat{r}_i^{l+1} s_{l+2} \dots \hat{r}_i^{k-1} s_k$ be the pre-order obtained by lexMin . Recall that u'_i are the utilities corresponding to (2), and \hat{u}_i are called the utilities corresponding to lexMin .

The proof proceeds by induction using the length l of r_i^l . Initially $u_i(s_1) = \hat{u}_i(s_1) = 0$, because 0 is both a starting value in (2) and the smallest value in lexicographical order. It

is supposed that utilities both coincide until s_l . Let us consider s_{l+1} . It is also assumed that $u'_i(s_l) = u'_i(s_{l+1})$ in Γ' and, since equivalences are maintained through identity, $\hat{u}_i(s_{l+1}) = \hat{u}_i(s_l)$ in Γ' is obtained. Now, it is supposed that $u'_i(s_{l+1}) = 1 + u'_i(s_l)$. Then, we are forced to $\hat{u}_i(s_{l+1}) > \hat{u}_i(s_l)$, and the best lexicographical choice is $\hat{u}_i(s_{l+1}) = 1 + \hat{u}_i(s_l)$.

In order to prove that (2) implies (3) note that, by construction, the game Γ' given in (2) is weakly identical to the initial Γ . Moreover, ascending values given to the utilities force the chain $0 < \dots < \delta_i - 1$.

Let us prove that (3) implies (1). Let $s_1 r_i^1 s_2 r_i^2 \dots r_i^l s_{l+1} r_i^{l+1} s_{l+2} \dots r_i^{k-1} s_k$ be the pre-order obtained by (3), and $s_1 \hat{r}_i^1 s_2 \hat{r}_i^2 \dots \hat{r}_i^l s_{l+1} \hat{r}_i^{l+1} s_{l+2} \dots \hat{r}_i^{k-1} s_k$ be the pre-order obtained getting lexMin .

Now, u'_i is the utility corresponding to (3) and \hat{u}_i is an utility corresponding to lexMin . Since in Γ , δ_i values are needed to encode utilities, we are forced to start with the smallest one. The smallest value is 0, otherwise there will not be enough increasing values. Moreover, a lexicographical order forces the first utility to be 0. Then, $u_i(s_1) = \hat{u}_i(s_1) = 0$ is obtained.

Let us deal with profile s_{l+1} . There is no problem when $u'_i(s_l) = u'_i(s_{l+1})$. The case $u'_i(s_{l+1}) > u'_i(s_l)$ is considered. Since in Γ' we have the number of strictly needed values to encode utilities, we are forced to take $u'_i(s_{l+1}) = 1 + u'_i(s_l)$. Moreover, the best choice under lexicographical order forces $\hat{u}_i(s_{l+1}) = 1 + \hat{u}_i(s_l)$. □

Example C.4 Games Γ and Γ' , such that $\Gamma \sim_w^I \Gamma'$, are considered.

$$\begin{array}{c}
 \begin{array}{cc}
 & \begin{array}{cc} l & r \end{array} \\
 \begin{array}{c} t \\ b \end{array} & \begin{array}{|cc|}
 \hline
 3, 2 & 0, 2 \\
 \hline
 0, 0 & 3, 0 \\
 \hline
 \end{array} \\
 \Gamma
 \end{array}
 \xrightarrow{I}
 \begin{array}{c}
 \begin{array}{cc}
 & \begin{array}{cc} l' & r' \end{array} \\
 \begin{array}{c} t' \\ b' \end{array} & \begin{array}{|cc|}
 \hline
 1, 1 & 0, 1 \\
 \hline
 0, 0 & 1, 0 \\
 \hline
 \end{array} \\
 \Gamma'
 \end{array}
 \end{array}$$

The utilities of Γ are encoded by the string $(\langle u_1 \rangle, \langle u_2 \rangle) = 30032200$. Since $3 = u_1(t, l) = u_1(b, r) > u_1(t, r) = u_1(b, l) = 0$, the utilities of player 1 encode preferences $(t, l) \sim_1 (b, r) \succ_1 (t, r) \sim_1 (b, l)$. The same preferences can be encoded with values 1 and 0. In the case of player 2, utilities encode preferences $(t, l) \sim_2 (t, r) \succ_2 (b, l) \sim_2 (b, r)$. As before, two values are necessary for encoding such a preference. Encoding preferences with values as small as possible are obtained: $\text{minWeak}(\Gamma) = \Gamma'$, such that $(\langle u'_1 \rangle, \langle u'_2 \rangle) = 10011100$.

Lemma C.4 An encoding of $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$ is given. Then, the encoding of $\text{minWeak}(\Gamma)$ can be computed in time $O((k_1 + \dots + k_n) \log(k_1 + \dots + k_n))$.

Proof. This lemma can be seen as an efficient implementation of the algorithm sketched in Lemma C.3. The input is a tuple $\langle u \rangle = (\langle u_1 \rangle, \dots, \langle u_n \rangle)$, such that for $1 \leq i \leq n$, $\langle u_i \rangle = \langle u_i(0), \dots, u_i(k-1) \rangle$. It is assumed that sorting algorithms are stable. Construction goes through several steps.

- Scan $\langle u \rangle$ and fill a table $\langle U \rangle = (\langle U_1 \rangle, \dots, \langle U_n \rangle)$, such that “segment” i corresponds to $\langle U_i \rangle = \langle U_i(0), \dots, U_i(k-1) \rangle$, where for $1 \leq l \leq n$, $U_i(i) = (u_i(l), i, l)$ is defined. This table can be built in time $O(k_1 + \dots + k_n)$.
- Order $\langle U \rangle$ according to the values u of utilities in elements (u, i, l) . This is achieved in time $O((k_1 + \dots + k_n) \log(k_1 + \dots + k_n))$.
- Scan $\langle U \rangle$ and build a table $\langle V \rangle$, such that each $\langle V_i \rangle$ contains all values of $\langle U_i \rangle$ ordered in a non-decreasing order of utilities. Using the process identifier i in elements (u, i, l) , can be done in time $O(k_1 + \dots + k_n)$.
- Scan $\langle V \rangle$ and build a table $\langle R \rangle$ containing the reduced utility values for each block $\langle R_i \rangle$. Suppose that the scanning block is started $\langle V_i \rangle$. Zero is assigned as first value of the utility. The first time that the utility (arbitrary) increases, utility 1 is assigned to the corresponding item of $\langle V_i \rangle$. Proceed in this way until the end of the block. The corresponding items are called (v, i, l) . The entire process can be performed in time $O(k_1 + \dots + k_n)$. Note that preferences are coded with minimal values, but strategies are not well ordered inside each block.
- Sort $\langle V \rangle$ according to profile value l into a new $\langle Z \rangle$. As usual, time is $O((k_1 + \dots + k_n) \log(k_1 + \dots + k_n))$.
- Scan $\langle Z \rangle$ according to i and fill $\langle u' \rangle$. This is done in $O(k_1 + \dots + k_n)$. Therefore, new utilities are ordered according to profiles in each block $\langle u_i \rangle$.

□

Lemma C.5 For any pair of games Γ and Γ' , it holds that $\Gamma \sim_w \Gamma'$ iff $\min\text{Weak}(\Gamma) \sim_s \min\text{Weak}(\Gamma')$.

Proof. First, it is proven that $\Gamma \sim_w \Gamma'$ implies $\min\text{Weak}(\Gamma) \sim_s \min\text{Weak}(\Gamma')$. Since $\Gamma \sim_w \min\text{Weak}(\Gamma)$ and $\Gamma' \sim_w \min\text{Weak}(\Gamma')$, $\min\text{Weak}(\Gamma) \sim_w \min\text{Weak}(\Gamma')$ holds. It is known that $A = A'$, and it is supposed that $|A| = k$. The ordering given by the utilities u_i in Γ of all the strategy profiles is considered. Look at Lemma C.3. Let $s_1 r_i^1 s_2 r_i^2 \dots s_l r_i^l \dots r_i^{k-1} s_k$ be an ordering of all strategy profiles belonging to A , where $r_i^j \in \{\prec_i, \sim_i\}$. For instance, it is supposed that $s_1 \prec_i s_2 \sim_i s_3 \prec_i s_4 \dots$, then $u_i(s_1) = 0$, $u_i(s_2) = u_i(s_3) = 1$, $u_i(s_4) = 2$.

Since $\min\text{Weak}(\Gamma) \sim_w \min\text{Weak}(\Gamma')$, there exists an isomorphism Ψ between both games, and $\Psi(s_1) r_{\pi(i)}^1 \Psi(s_2) r_{\pi(i)}^2 \dots \Psi(s_l) r_{\pi(i)}^l \dots r_{\pi(i)}^{k-1} \Psi(s_k)$. For instance, if

we begin with the same example, $\Psi(s_1) \prec_{\pi(i)} \Psi(s_2) \sim_{\pi(i)} \Psi(s_3) \prec_i \Psi(s_4) \cdots$. This forces (by Lemma C.3) the following values of utilities: $u'_{\pi(i)}(\Psi(s_1)) = 0$, $u'_{\pi(i)}(\Psi(s_2)) = u'_{\pi(i)}(\Psi(s_3)) = 1$, and $u'_{\pi(i)}(\Psi(s_4)) = 2$. In general, $u'_i(s) = u'_{\pi(i)}(\Psi(s))$ and $\text{minWeak}(\Gamma) \sim_s \text{minWeak}(\Gamma')$.

It remains to be proven that $\text{minWeak}(\Gamma) \sim_s \text{minWeak}(\Gamma')$ implies $\Gamma \sim_w \Gamma'$. By Lemma C.1, \sim_s implies \sim_w . Therefore, $\text{minWeak}(\Gamma) \sim_w \text{minWeak}(\Gamma')$ holds and the result follows. \square

Families of Small Games

Two families of games have been considered: the family of games under weak isomorphisms \mathcal{F}_w , and the family of games under local isomorphisms, \mathcal{F}_ℓ . The weak family was introduced by I. Young [96] to model “random”- n person ordinal games. We have to look at utilities u_i as encoding preferences \prec_i , since weak isomorphisms deal with ordinal preferences between players. Therefore, as $|A| = k$, $k = k_1 \dots k_n$ the longest chain of strict preferences \prec_i contains at most k different elements. Using the techniques in Lemma C.3, it is possible to re-encode utilities with values in $\{0, \dots, k-1\}$. Then, given a Γ game, there exists an $\Gamma' = (N, (A_i)_{i \in N}, (u'_i)_{i \in N})$, such that $\Gamma' \sim_w^I \Gamma$ and $u'_i(a) \in \{0, \dots, k-1\}$ for any i and a .

Definition C.5 [96] *The family $\mathcal{F}_w(n, k_1, \dots, k_n)$ contains all strategic games $\Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N})$, such that $N = \{1, \dots, n\}$, $A_i = \{0, \dots, k_i - 1\}$, for all $0 \leq i \leq n$. Moreover, $u_i(a) \in \{0, \dots, k-1\}$, and for any player $1 \leq i \leq n$ and profile $0 \leq a \leq k$, where $k = k_1 \dots k_n$.*

Note that $|\mathcal{F}_w(n, k_1, \dots, k_n)| = k^{nk}$. Therefore, \mathcal{F}_w contains many encodings for a given ordinal preference, as Example C.5 shows. In the same way, the family of local isomorphisms games $\mathcal{F}_\ell(n, k_1, \dots, k_n)$ must be defined.

Definition C.6 *Given a game Γ , a $\text{maxLocalNash}(\Gamma) = \text{lexMax}\{\Gamma' | \Gamma' \sim_{\ell_N}^I \Gamma\}$ is defined as local-Nash identical game with lexicographically maximal game encoding.*

It is well known that the number of parameters to be specified grows exponentially with the set of players' actions. Additionally, a strategic game may fail to capture the structure present in the players' interaction. However, it helps in order to understand the game and the computation of its equilibria, as it has been shown before. This section is focused in defining and classifying small games accordingly to the structure of PNE, and a graphical model to represent small games is proposed as well. The graphical model should be regarded as the way in which relationships among players under local isomorphisms are captured and exploited. Then, a small game is defined as follows.

Definition C.7 [Small Game]. A small game is defined as a game with a set of players $N = \{1, \dots, n\}$, where n is at most 3. Each player $i \in N$, has a finite set of actions A_i , such that $A_i = \{0, \dots, k_i - 1\}$ for all $0 \leq i \leq n = 3$, where a player i has k_i actions. For $i = \{1, \dots, n\}$ and $a \in A_1 \times \dots \times A_n$, the utilities of players are $u_i(a) = \{0, \dots, |A_1 \times \dots \times A_n| - 1\}$. Then, the family $\mathcal{F}_w(n, k_1, \dots, k_n)$ is defined for $n \leq 3$, and $k_i \leq 3$. The classes $\mathcal{F}_w(2, 2, 2)$, $\mathcal{F}_w(2, 2, 3)$, $\mathcal{F}_w(2, 3, 3)$, $\mathcal{F}_w(3, 2, 2, 2)$ are identified with the family of small games.

Example C.5 The class C of games $\Gamma \in \mathcal{F}_w(2, 2, 2)$, such that players 1 and 2 are indifferent about the choice of any strategy profile, is considered. A game Γ belongs to C iff exists $0 \leq v_1, v_2 < 4$, such that for all a , $u_1(a) = v_1$ and $u_2(a) = v_2$. Therefore, $|C| = 16$ and $\text{minLocal}(C)$ is the game, such that $u_1(a) = u_2(a) = 0$ for any profile a .

Family Game $F_w(2, 2, 2)$. In order to give a graphical representation of small games, a brief analysis about their equivalence relationships is first needed. In order to perform such analysis, a family of games $\mathcal{F}_w(2, 2, 2)$ is considered. This family has two players with two actions $A_i = \{1, 2\}$, $i \in \{0, 1\}$. Since $A = A_1 \times A_2 = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$, the length of any chain of \preceq_i is at most 4; for instance, $(1, 1) \prec_1 (1, 0) \prec_1 (0, 1) \prec_1 (0, 0)$. Therefore, $u_i(a) \in \{0, 1, 2, 3\}$ is enough to encode all possible preferences. For example, the preceding preference has been encoded as $u_1(3) = 0$, $u_1(2) = 1$, $u_1(1) = 2$, $u_1(0) = 3$. Then, for this class of games, in order to obtain a complete classification it is required that $4^8 = 65536$ games are explored. In order to compute all games, an easy way to encode them was found.

Definition C.8 [Game's encoding]. A small game $\Gamma \in F_w(2, 2, 2)$ is shortly encoded as the following string g ¹ of 8 characters: $g = u_1(0)u_1(1)u_1(2)u_1(3)u_2(0)u_2(1)u_2(2)u_2(3)$ where $u_i(1) = \{0, 1, 2, 3\}$.

Applying this definition, the encoding of any small game $\Gamma \in \mathcal{F}_w(2, 2, 2)$ should follow the following pattern: $\langle 11, \langle 0, 1, 2, 3 \rangle, \langle 0, 1, 2, 3 \rangle, \langle 0, 1, 2, 3 \rangle, \langle 0, 1, 2, 3 \rangle, \langle u_1(0)u_1(1)u_1(2)u_1(3) \rangle, \langle u_2(0)u_2(1)u_2(2)u_2(3) \rangle \rangle$. All games contain a common prefix, encoding the number of players and actions. This prefix can be avoided, and it is just needed that the part corresponding to utilities is kept. Moreover, in order to represent small games, the utilities are encoded as a unique string. For instance, game Γ given in Example C.4 is encoded as $g=30032200$.

Counting PNE in the Family Games $\mathcal{F}_w(2, 2, 2)$. The usual algorithms for counting the number of PNE in bi-matrix games, with players having n actions, takes time $O(n^2)$ [83].

¹Note that the g string is the encoding of game Γ . In this section “ g ” and “ h ” instead of Γ and Γ' are used.

I. Y. Powers [96] however, chose a different strategy, which consisted in counting pure Nash equilibria via probabilistic analysis. She studied the limit distributions of number of pure Nash equilibria strategies in \mathcal{F}_w .

In order to deal with small games, the following ad-doc compact code, is proposed:

```
int countNash(string g){
    int n=0;
    if((g[0]>=g[2])and(g[4]>=g[5])) ++n;
    if((g[1]>=g[3])and(g[5]>=g[4])) ++n;
    if((g[2]>=g[0])and(g[6]>=g[7])) ++n;
    if((g[3]>=g[1])and(g[7]>=g[6])) ++n;
    return n;
}
```

This simple approach has limitations, because games with different structures may have the same number of Nash equilibria. Therefore, it is needed that a better answer is provided to the question: *When are two games the same?* Obviously, a mathematical setting to deal with equivalence is through the notions of strong, weak and local isomorphisms. In a first classification, small games of the $F_w(2,2,2)$ family can be grouped, according to the number of Nash equilibria, as it is in the following table.

# PNE	0	1	2	3	4	total
# Games	2592	29376	27936	5376	256	65536

Nevertheless, this first classification remains wide. Then, to reduce the number of equivalence classes, games have been analysed from the point of view of their local isomorphisms.

Definition C.9 [minLocalGame]. *Given a small game g , $\text{minLocalGame}(game)$ is a game with the same local preference relationships as g , but encoded with numbers as small as possible.*

Example C.6 *Consider a game $\Gamma = 30032200$. Then, $\text{minLocalGame}(30032200) = 10010000$.*

Therefore, applying the `minLocalGame` definition below, the following snipped code

computes the `minLocalGame(game)` of each game in the $F_w(2,2,2)$ family.

```
string minLocalGame(string game){
    string s="";
    char a, b, c, d, e, f, g, h;
    //first player
    if(game[0]==game[2]){a=c='0';}
        else if(game[0]<game[2]){a='0'; c='1';}
    else{a='1'; c='0';}
    if(game[1]==game[3]){b=d='0';}
        else if(game[1]<game[3]){b='0'; d='1';}
    else{b='1'; d='0';}
    //second player
    ...
    s=s+a+b+...;
    return s;
}
```

The following table summarises the number of local isomorphisms for this class of games, $F_w(2,2,2)$. As it is shown in the following table, one type of game with 0 PNE, and 4 types of games with 1 PNE are obtained. Therefore, the equivalence classes contain the least amount of different games in each class.

Types of Local Isomorphism					
# PNE	0	1	2	3	4
# Types	1	4	7	2	1

Graphical Representations

Family Game $F_w(2,2,2)$. The set of strategy profiles $\{(0,0), (0,1), (1,0), (1,1)\}$ in the $F_w(2,2,2)$ family can be represented as a set of nodes in a square, where edges represent preferences. For instance, $(0,0) \leftrightarrow (0,1)$ means $(0,0) \sim_2 (0,1)$ and $(0,0) \rightarrow (0,1)$, iff $(0,0) \prec_2 (0,1)$ (when dealing with local preferences, the sub-index 2 can be avoided). Figure C.1 shows this setting.

This notation allows us the representation of local preferences. To the best of this thesis candidate's knowledge, this approach goes back to Harsanyi and Selten ([49], page 150). Therefore, each class of equivalence is represented by a square fulfilling some conditions about preferences. Tables C.1, C.2, and C.3 provide equivalence classes giving information about the possible structure of Nash equilibria under a local isomorphism. Graphs that appear in these tables are precisely the Nash dynamics graphs defined in [34].

Local Types				
1PNE, 29376 games				0PNE, 2592 games
00010101	00010110	00110101	00110110	01101001

Table C.1: This table shows the local isomorphic types for cases with 0 PNE and 1 PNE, for this family of games. For instance, cases having 0 PNE have 1 local isomorphic type, a 01101001 game, which is a local reduced minimum, common to all games with the same PNE.

Local Type: 2PNE, 27936 games			
00000101	00000110	00010001	00010010
00011001	00011010	01100110	

Table C.2: This table shows the local isomorphic types for cases with 2 PNE, for this family of games. There are 7 local isomorphic types.

Local Types		
3PNE, 5376 games		4 PNE, 256 games
00000001	00011000	00000000

Table C.3: This table shows the local isomorphic types for cases with 3 PNE and 4 PNE, for this family of games.

Local Types					
0 PNE	1 PNE	2 PNE		3 PNE	4 PNE
01101001	00110110	01100110	00011010	00000001	00000000

Table C.4: This table shows the local isomorphic types for cases with 3 PNE and 4 PNE, for this family of games.

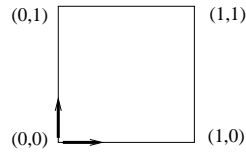


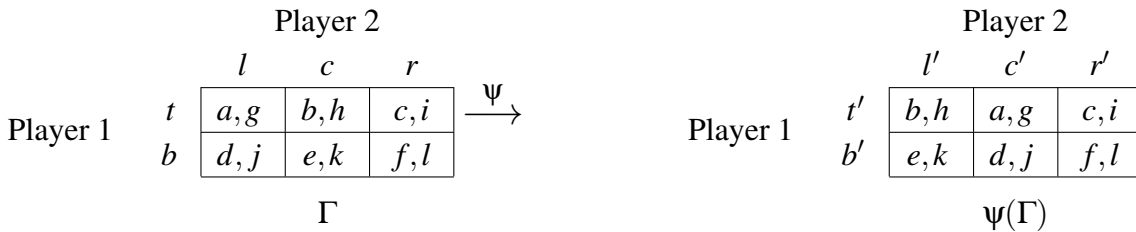
Figure C.1: Vertical edges represent preferences of one player, while horizontal edges represent preferences of the other player.

Family Game $F_w(2,2,3)$. A small game $\Gamma \in F_w(2,2,3)$ is a game having two players with different number of actions: $A_1 = \{0, 1\}$ and $A_2 = \{0, 1, 2\}$, such that $A = A_1 \times A_2 = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2)\}$. Therefore, $u_i(a) \in \{0, 1, 2, 3, 4, 5\}$ is enough to encode all possible preferences. Games are described using one table for both players, as the following table shows.

		Player 2		
		0	1	2
Player 1	0	a, g	b, h	c, i
	1	d, j	e, k	f, l
		Γ		

In this game utilities are, for instance, $u_1(0,0) = a$ and $u_2(0,0) = g$. Coding each matrix, we have $T = abcdefghijkl$, and $\Gamma = \langle 1^2, \{0, 1\}, \{0, 1, 2\}, abcdefghijkl \rangle$.

Note that, in a mapping $\Psi = (\pi, \phi_1, \phi_2)$ for this class of games, π does not swap players since they have different numbers of actions.



In order to explore all games in this family, a total of $(6^2)^6 = 2176782336$ games has been needed to be analysed, since the cardinal of $\{u_i(a) | a \in A\}$ is $6^6 = 46656$. However, as this computation takes too long, it was decided to *reduce* utilities values of players, without lost information. Accordingly to this, one player takes value $u_1(a) = \{0, 1\}$, and the second one takes values $u_2(a) = \{0, 1, 2\}$. Then, 46656 different games were generated. Nevertheless, still a high number of different types of games according to PNE resulted. For instance, if games with 6PNE: 24 different games under strong isomorphism were obtained, 6 under weak isomorphisms, and only one under local isomorphisms.

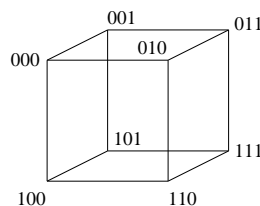
Therefore, each class of equivalence is represented by a graph, fulfilling some conditions about preferences. In these graphs, vertical edges represent preferences of one

player. The remaining edges represent preferences of the other player. Tables C.5 and C.6 show the class of equivalences for games in $F_w(2, 2, 3)$ without pure Nash equilibria, under a local isomorphism. As it will be shown, there are 17 local types for 0PNE. Table C.7 shows the case of 6PNE, where 72 small games have only one local type.

Local Types: 0PNE, reduced games: 17			
(001010010001, 108)	(001010010012, 72)	(001010010102, 72)	(001010021012, 12)
(001010021102, 24)	(001010120102, 12)	(001110110001, 54)	(001110110012, 36)
(001110120001, 36)	(001110120012, 12)	(001110120102, 12)	(001110010001, 108)

Table C.5: Class of equivalences for family games $F_w(2, 2, 3)$ with 0PNE, under a local isomorphism.

Family Game $F_w(3, 2, 2, 2)$. Now, games in $F_w(3, 2, 2, 2)$ having three players with two actions, are considered. The set of strategy profiles for these games is $A_1 \times A_2 \times A_3 = \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}$. This set can be represented as a set of nodes in a cube. Vertices represents a strategy profile for players, and each edge the movements that they make. Therefore, the following is adopted: player 1 moves between points $\{000, 100\}$, $\{001, 101\}$, $\{010, 110\}$ and $\{011, 111\}$. Player 2 moves between points $\{000, 010\}$, $\{001, 011\}$, $\{100, 110\}$ and $\{101, 111\}$. Player 3 moves between points $\{000, 001\}$, $\{010, 011\}$, $\{100, 101\}$ and $\{110, 111\}$.



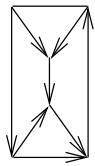
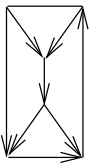
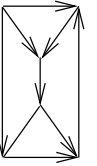
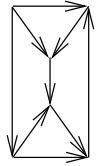
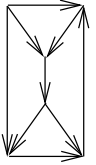
Local Types: 0PNE, reduced games: 17			
			
(001110010012, 36)	(001110010102, 36)	(001110021001, 36)	(001110021012, 12)
			
(001110021102, 12)			

Table C.6: Continuation of the previous table C.5. Class of equivalences for family games $F_w(2, 2, 3)$ with 0PNE, under a local isomorphism.

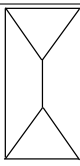
Local Types: 6PNE, reduced games: 1

(000000000000, 72)

Table C.7: Class of equivalences for family of games $F_w(2, 2, 3)$ with 6PNE, under a local isomorphism.

As before, games with ordinal preference relations \preceq_i are considered. Since $a \in A_1 \times A_2 \times A_3 = \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}$, the length of any chain of \preceq_i is at most 8. Therefore, $u_i(a) = \{0, 1, 2, 3, 4, 5, 6, 7\}$ for $i = \{1, 2, 3\}$ is enough in order to encode all possible preferences.

The following example shows a strategic game with three players and two actions each. Players' actions are $A_i = \{0, 1\}$, for $i \in \{0, 1, 2\}$. In these tables, $u_1(0, 0, 0) = a$, $u_2(0, 0, 0) = i$ and $u_3(0, 0, 0) = q$. Giving a code to each matrix, a game $\Gamma = \langle 1^3, \{0, 1\}, \{0, 1\}, \{0, 1\}, abcdefghijklmnopqrstuvw \rangle$ is obtained.

Example C.7 A strategic game with three players and two actions.

		<i>Player 2</i>				<i>Player 2</i>	
		0	1			0	1
<i>Player 1</i>	0	<i>a, i, q</i>	<i>c, k, s</i>	<i>Player 1</i>	0	<i>b, j, r</i>	<i>d, l, t</i>
	1	<i>e, m, u</i>	<i>g, o, w</i>		1	<i>f, n, v</i>	<i>h, p, x</i>
		<i>Player 3, action (0)</i>				<i>Player 3, action (1)</i>	

The following notation is adopted: player 1 with actions (t, b) (top, bottom), player 2 with actions (l, r) (left, right), and player 3 with actions (F, B) (Forward, Backward). Additionally, each player has 2 actions, therefore it is possible to swap players' actions.

Applying the mapping $\Psi = (\pi, \varphi_1, \varphi_2, \varphi_3)$, where $\pi = (1 \rightarrow 3, 2 \rightarrow 1, 3 \rightarrow 2)$, $\varphi_1 = (t \rightarrow F, b \rightarrow B)$, $\varphi_2 = (l \rightarrow t, r \rightarrow b)$, and $\varphi_3 = (F \rightarrow l, B \rightarrow r)$ then, 3PNE have been acquired in the strategic profiles $(1, 0, 0)$, $(0, 0, 1)$ and $(1, 1, 1)$.

	<p>Player 2</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td></td> <td style="text-align: center;">l</td> <td style="text-align: center;">r</td> <td></td> </tr> <tr> <td style="text-align: center;">t</td> <td style="border: 1px solid black; padding: 5px;">0,0,0</td> <td style="border: 1px solid black; padding: 5px;">0,1,1</td> <td></td> </tr> <tr> <td style="text-align: center;">b</td> <td style="border: 1px solid black; padding: 5px;">1,1,1</td> <td style="border: 1px solid black; padding: 5px;">1,0,0</td> <td></td> </tr> </table> <p style="text-align: center;">Player 3, action (F)</p>		l	r		t	0,0,0	0,1,1		b	1,1,1	1,0,0		<p>Player 2</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td></td> <td style="text-align: center;">l</td> <td style="text-align: center;">r</td> <td></td> </tr> <tr> <td style="text-align: center;">t</td> <td style="border: 1px solid black; padding: 5px;">1,0,1</td> <td style="border: 1px solid black; padding: 5px;">0,1,0</td> <td></td> </tr> <tr> <td style="text-align: center;">b</td> <td style="border: 1px solid black; padding: 5px;">1,0,0</td> <td style="border: 1px solid black; padding: 5px;">0,0,1</td> <td></td> </tr> </table> <p style="text-align: center;">Player 3, action (B)</p>		l	r		t	1,0,1	0,1,0		b	1,0,0	0,0,1		
	l	r																									
t	0,0,0	0,1,1																									
b	1,1,1	1,0,0																									
	l	r																									
t	1,0,1	0,1,0																									
b	1,0,0	0,0,1																									
$\xrightarrow{\Psi}$	<p>Player 2'</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td></td> <td style="text-align: center;">l'</td> <td style="text-align: center;">r'</td> <td></td> </tr> <tr> <td style="text-align: center;">t</td> <td style="border: 1px solid black; padding: 5px;">0,0,0</td> <td style="border: 1px solid black; padding: 5px;">0,1,1</td> <td></td> </tr> <tr> <td style="text-align: center;">b</td> <td style="border: 1px solid black; padding: 5px;">1,1,0</td> <td style="border: 1px solid black; padding: 5px;">1,0,0</td> <td></td> </tr> </table> <p style="text-align: center;">Player 3', action F'</p>		l'	r'		t	0,0,0	0,1,1		b	1,1,0	1,0,0		<p>Player 2'</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td></td> <td style="text-align: center;">l'</td> <td style="text-align: center;">r'</td> <td></td> </tr> <tr> <td style="text-align: center;">t</td> <td style="border: 1px solid black; padding: 5px;">1,1,1</td> <td style="border: 1px solid black; padding: 5px;">0,0,1</td> <td></td> </tr> <tr> <td style="text-align: center;">b</td> <td style="border: 1px solid black; padding: 5px;">0,0,1</td> <td style="border: 1px solid black; padding: 5px;">0,1,0</td> <td></td> </tr> </table> <p style="text-align: center;">Player 3', action B'</p>		l'	r'		t	1,1,1	0,0,1		b	0,0,1	0,1,0		
	l'	r'																									
t	0,0,0	0,1,1																									
b	1,1,0	1,0,0																									
	l'	r'																									
t	1,1,1	0,0,1																									
b	0,0,1	0,1,0																									

Like in the previous family of games, the cardinal of $\{u_i(a) | a \in A\}$ was computed and it is 8^8 . Since, there are three players, it is needed that a total of $(8^3)^8$ games are analysed. However, since this computation could take too long, it was that decided that utility values of players would be reduced, without losing information. In this way, $(2^3)^8$ games were obtained. For instance, in a first classification, these games can be grouped according to the number of PNE, obtaining 4096 different games with 8PNE.

The classification of this family of games was possible through reduction of player' utilities. According to isomorphism types, games were analysed and classified. For instance, 144 different "reduced" strong isomorphic games, and only 1 local isomorphic game for games with 8PNE were acquired. Therefore, it is possible to represent these game taking into account a local isomorphism. For example, Figure C.2 shows a graphical representation of the previous Example C.7. Note that it is easy to detect the 3PNE in the game represented.

Finally, as an example of graphs for this family of games, two graphs of local isomorphisms are shown in Figure C.2. Case (a) shows the graph for games with 0PNE, while the second graph, games with 8PNE.

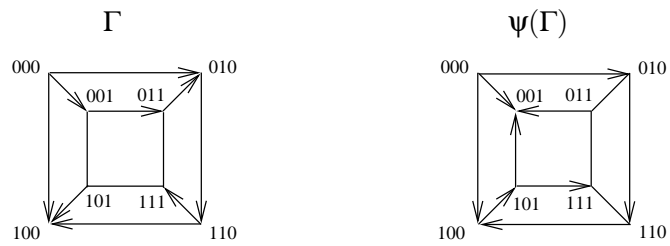


Table C.8: Graph representation of the previous Example game C.7. The game $\Gamma = abcdefghijklmnopqrstuvwxyz$, where an identity mapping is $tblrFB$, and written briefly as $\psi(\Gamma) = imjnkolpqrsvwtxaebfcgdh$.

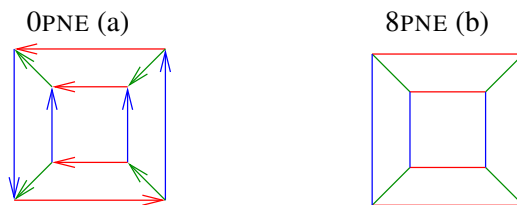


Figure C.2: Case (a) shows the graph for games with 0PNE, and case (b), games with 8PNE.

Part V

Publications and Projects

Appendix A

List of Publications

Publications that have been written during the development of the research here presented,

[41] J. Gabarro, A. Garcia and M. Serna. On the complexity of game isomorphism. *Mathematical Foundations of Computer Science, MFCS:LNCS 4708*:559-571, 2007. http://dx.doi.org/10.1007/978-3-540-74456-6_50

[39] M. Clint, P. Kilpatrick, A. Stewart, J. Gabarro, A. García. Bounded site failures: an approach to unreliable grid environments. In "Making Grids Work", Proceedings of the CoreGRID Workshop on Programming Models and P2P System Architecture, Grid Systems and Environments. Edited by M. Danelutto, P. Fragopoulou, V. Getov. Springer, 175-187, 2008. <http://www.springerlink.com/content/uk21g31059737777/>

[44] Gabarro, J; Garcia, A.; Serna, M; Stewart, A; Kilpatrick, P.;, Analysing Orchestrations with Risk Profiles and Angel-Daemon Games. In "Grid Computing Achievements and Propects", CoreGRID Integration Workshop 2008. Edited by S. Gorlatch, P. Fragopoulou, Th. Priol. Springer 121-132, 2008. <http://www.springerlink.com/content/m13372778r57p5n8/>

[42] Joaquim Gabarro and Alina Garcia and Maria J. Serna, On the Complexity of Equilibria Problems in Angel-Daemon Games. COCOON-2008: LNCS 5092:31-40. <http://www.springerlink.com/content/dvkv7753nh514203/>

[43] Joaquim Gabarro, Alina Garcia, Maria J. Serna, The complexity of game isomorphism. *Theoretical Computer Science*, Vol. 412(48): 6675–6695, 2011. <http://www.sciencedirect.com/science/article/pii/S0304397511006323>

[40] Joaquim Gabarro, Alina Garcia, Maria J. Serna, On the Hardness of game equivalence under Local Isomorphism. *Theoretical Informatics and Applications*. (Submitted).

Appendix B

List of Projects

Projects in which I participated during the development of the research here presented,

- Técnicas de Optimización Avanzadas para Problemas Complejos (TRACER), (CYCIT TIC2002-04498-C05-03).
- Fundamental Aspects of Global Computing Systems (FLAGS), (EU IS- 2001-33116).
- Autoorganización en Sistemas de Comunicación Emergentes (ASCE), (MEC-TIN2005-09198-C02-02).
- Algorithmic principles for building efficient Overlay computers (AEOLUS), (FET pro-actives Integrated Project 15964).
- FP6 Network of Excellence CoreGRID founded by the European Commission, (Contract IST-2002-004265).
- Métodos Formales y algoritmos para el diseño de sistemas (FORMALISM), (MEC-TIN2007-66523).

Bibliography

- [1] Risk management guide for information technology systems recommendations of the national institute of standards and technology. Technical report, USA/NIST.
- [2] Workflow Patterns home page. Also available as: <http://www.workflowpatterns.com/>.
- [3] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
- [4] M. Agrawal and T. Thierauf. The Formula Isomorphism Problem. *SIAM Journal on Computing*, 30(3), 2000.
- [5] Luis von Ahn. Games with a purpose. *IEEE Computer*, 39:92–94, 2006.
- [6] George Akerlof and Robert Shiller. *Animal spirits how human psychology drives the economy, and why it matters for global capitalism*. Princeton University Press, 2009.
- [7] Carme Alvarez, Joaquim Gabarro, and Maria J. Serna. Polynomial Space Sufices for Deciding Nash Equilibria Properties for extensive games with large trees. In *ISAAC, Lecture Notes in Computer Science, LNCS*, pages 634–643. Springer, 2005.
- [8] Carme Alvarez, Joaquim Gabarro, and Maria J. Serna. Pure Nash Equilibria in Games with a Large Number of Actions. *Electronic Colloquium on Computational Complexity (ECCC)*, (031), 2005.
- [9] Carme Alvarez, Joaquim Gabarro, and Maria J. Serna. Pure Nash Equilibria in Games with a Large Number of Actions. In Joanna Jedrzejowicz and Andrzej Szepietowski, editors, *Mathematical Foundations of Computer Science, 30th International Symposium, MFCS*, volume 3618 of *Lecture Notes in Computer Science, LNCS*, pages 95–106. Springer, 2005. 10.1007/11549345_10.
- [10] Moshe Babaioff, Robert Kleinberg, and Christos H. Papadimitriou. Congestion games with malicious players. In Jeffrey K. MacKie-Mason, David C. Parkes,

- and Paul Resnick, editors, *Proceedings of the 8th ACM conference on Electronic commerce*, EC'07, pages 103–112, New York, USA, 2007. ACM.
- [11] José L. Balcázar, Antoni Lozano, and Jacobo Torán. The complexity of algorithmic problems on succinct instances. In *Computer Science*, pages 351–377, New York, USA, 1992. Plenum Press.
- [12] José Luis Balcázar, Joseph Díaz, and Joaquim Gabarro. *Structural complexity 2*. Springer-Verlag, New York, USA, 1990.
- [13] Ranieri Baraglia, R. Ferrini, Nicola Tonellotto, D. Adami, S. Giordano, and Ramin Yahyapour. A Study on Network Resources Management in Grids. In *Proc. Core-GRID Integration Workshop*, Cracow, 2006.
- [14] A.P. Barros, M. Dumas, and P. Oaks. A Critical Overview of the Web Services Choreography Description Language (WS-CDL). *BPTrends*, 2005.
- [15] Amos Beimel and Enav Weinreb. Monotone circuits for monotone weighted threshold functions. *Information Processing Letters*, 97(1):12–18, 2006.
- [16] Elise Bonzon, Marie-Christine Lagasquie-Schiex, Jérôme Lang, and Bruno Zanuttini. Boolean games revisited. In *Proceeding of the 2006 conference on ECAI*, pages 265–269, Amsterdam, The Netherlands, 2006. IOS Press.
- [17] B. Borchet, D. Ranjan, and F. Stephan. On the computational complexity of some classical equivalence relations on boolean functions. *Theory of Computing Systems*, 31:679–693, 1998.
- [18] L. Bougé. Le modèle de programmation à parallélisme de données: une perspective sémantique. 12:541–562, 1993. Techniques et Science Informatiques.
- [19] Mario Bravetti and Gianluigi Zavattaro. Towards a unifying theory for choreography conformance and contract compliance. In *Proceedings of the 6th international conference on Software Composition*, SC'07, pages 34–50, Berlin, Heidelberg, 2007. Springer-Verlag.
- [20] I. Caragiannis, C. Kaklamanis, P. Kanellopoulos, M. Kyropoulou, and E. Papaioannou. The impact of altruism on the efficiency of atomic congestion games. volume 6084 of *Lecture Notes in Computer Science, LNCS*, pages 172–188. In Proc. of the 5th Symposium on Trustworthy Global Computing (TGC'10), 2010.
- [21] Samuele Carpineti and Cosimo Laneve. A basic contract language for web services. In Peter Sestoft, editor, *Programming Languages and Systems, 15th European Symposium on Programming (ESOP)*, volume 3924 of *Lecture Notes in Computer Science, LNCS*, pages 197–213. Springer-Verlag, 2006.

-
- [22] André Casajus. Weak isomorphism of extensive games. *Mathematical Social Sciences*, 46(3):267–290, 2003.
- [23] André Casajus. Super weak isomorphism of extensive games. *Mathematical Social Sciences*, 51(1):107–116, 2006.
- [24] William R. Cook and Janel Barfield. Web Services versus Distributed Objects: A Case Study of Performance and interface design. In *ICWS'06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 419–426, Washington, DC, USA, 2006. IEEE Computer Society.
- [25] William R. Cook, Sourabh Patwardhan, and Jayadev Misra. Workflow Patterns in Orc. In *COORDINATION*, pages 82–96, 2006.
- [26] A. A. Cournot. *Recherches sur les principes mathématiques de la théorie des richesses*. L. Hachette, Paris, 1838.
- [27] Marco Danelutto and Marco Aldinucci. Algorithmic skeletons meeting grids. *Parallel Comput.*, 32(7):449–462, 2006.
- [28] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. In *STOC'06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 71–78, New York, USA, 2006. ACM.
- [29] Boudewijn P. de Bruin. Game transformations and game equivalence. Technical report, 1999.
- [30] Edsger W. Dijkstra. *Cooperating sequential processes*, pages 65–138. Springer-Verlag, New York, USA, 2002.
- [31] C Dwork, D Peleg, N Pippenger, and E Upfal. Fault tolerance in networks of bounded degree. In *STOC'86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 370–379, New York, USA, 1986. ACM.
- [32] Kfir Eliaz. Fault Tolerant Implementation. *Review of Economic Studies*, 69(3):589–610, 2002.
- [33] Susan Elmes and Philip J. Reny. On the strategic equivalence of extensive form games. *Journal of Economic Theory*, 62(1):1–23, 1994.
- [34] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC'04, pages 604–612, New York, USA, 2004. ACM.

- [35] J. Feigenbaum, D. Koller, and P. Shor. A game-theoretic classification of interactive complexity classes. In *SCT'95: Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, page 227, Washington, USA, 1995. IEEE Computer Society.
- [36] Joan Feigenbaum. Games, complexity classes, and approximation algorithms. In *Proceedings of the International Congress of Mathematicians, Vol. III*, number Extra Vol. III, pages 429–439, 1998.
- [37] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
- [38] Lance Fortnow, Valentine Kabanets, Russell Impagliazzo, and Christopher Umans. On the complexity of succinct zero-sum games. In *IEEE Conference on Computational Complexity*, pages 323–332, 2005.
- [39] Joaquim Gabarro, Alina Garcia, Maurice Clint, Peter Kilpatrick, and Alan Stewart. Bounded Site Failures: an Approach to Unreliable Grid Environments. In V. Getov M. Danelutto, P. Fragopoulou, editor, *CoreGRID Workshop on A Grid Programming Model, A Grid and P2P Systems Architecture, A Grid Systems Tools and Environments*, pages 175–187. Springer, 2007.
- [40] Joaquim Gabarro, Alina Garcia, and Maria Serna. On the hardness of game equivalence under local isomorphism. In *Theoretical Informatics and Applications*.
- [41] Joaquim Gabarro, Alina Garcia, and Maria Serna. On the complexity of game isomorphism. In Ludek Kucera and Antonin Kucera, editors, *Mathematical Foundations of Computer Science, 32nd International Symposium, MFCS 2007*, volume 4708 of *Lecture Notes in Computer Science, LNCS*, pages 559–571. Springer, 2007.
- [42] Joaquim Gabarro, Alina Garcia, and Maria Serna. On the complexity of equilibria problems in angel-daemon games. In Xiaodong Hu and Jie Wang, editors, *Computing and Combinatorics*, volume 5092 of *Lecture Notes in Computer Science, LNCS*, pages 31–40. Springer, 2008.
- [43] Joaquim Gabarro, Alina Garcia, and Maria Serna. The complexity of game isomorphism. *Theoretical Computer Science*, 412(48):6675 – 6695, 2011.
- [44] Joaquim Gabarro, Alina Garcia, Maria Serna, Peter Kilpatrick, and Alan Stewart. Analysing Orchestrations with Risk Profiles and Angel-Daemon Games. In Sergei Gorlatch, Paraskevi Fragopoulou, and Thierry Priol, editors, *Grid Computing*, pages 121–132. Springer, 2008.

-
- [45] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H.Freeman and Co., San Francisco, 1979.
- [46] Andrew Gilpin and Tuomas Sandholm. Finding equilibria in large sequential games of imperfect information. In *EC'06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 160–169, New York, USA, 2006. ACM.
- [47] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Pure Nash Equilibria: Hard and Easy Games. In *Journal of Artificial Intelligence Research*, pages 215–230. ACM Press, 2003.
- [48] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, Inc., New York, USA, 1995.
- [49] John C. Harsanyi and Reinhard Selten. *A General Theory of Equilibrium Selection in Games*. MIT Press, Cambridge, MA, 1988. With a foreword by Robert Aumann.
- [50] Martin Hepp and Katharina Siorpaes. On to game: Weaving the Semantic Web by Online Games. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008*, pages 751–766, 2008.
- [51] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International (UK) Ltd., 1985.
- [52] Charles A. Holt and Alvin E. Roth. The Nash equilibrium: A perspective. *Proceedings of the National Academy of Sciences of the United States of America*, 101(12):3999–4002, 2004.
- [53] John C. Hull. *Risk Management and Financial Institutions*. Prentice Hall International (UK), 2006.
- [54] Diane Jordan (IBM) and John Evdemon (Microsoft). Technical Committee: OASIS Web Services Business Process Execution Language (WSBPEL) TC Chair(s), 2007.
- [55] Sham Kakade, Michael Kearns, John Langford, and Luis Ortiz. Correlated equilibria in graphical games. In *Proceedings of the 4th ACM conference on Electronic commerce, EC'03*, pages 42–47, New York, USA, 2003. ACM.
- [56] G. Kandaswamy, L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon. Building web services for scientific grid applications. *IBM J. Res. Dev.*, 50(2-3):249–260, 2006.

- [57] Michael J. Kearns, Michael L. Littman, and Satinder P. Singh. Graphical Models for Game Theory. In *UAI'01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 253–260, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
- [58] David Kitchin, William R. Cook, and Jayadev Misra. A Language for Task Orchestration and Its Semantic Properties. In Christel Baier and Holger Hermanns, editors, *Concurrency Theory, 17th International Conference, CONCUR 2006*, volume 4137 of *Lecture Notes in Computer Science, LNCS*, pages 477–491. Springer, 2006.
- [59] David Kitchin, Adrian Quark, William R. Cook, and Jayadev Misra. The Orc programming language. In David Lee, Antónia Lopes, and Arnd Poetzsch-Heffter, editors, *Proceedings of FMOODS/FORTE 2009*, volume 5522 of *Lecture Notes in Computer Science, LNCS*, pages 1–25. Springer, 2009.
- [60] Frank H. Knight. *Risk, Uncertainty and Profit*. Houghton Mifflin, 1921. Electronic acces at <http://www.econlib.org/library/Knight/knRUP.html>.
- [61] J. Kobler, U. Schoning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser Verlag, Basel, Switzerland, 1993.
- [62] Natallia Kokash. Risk Management for Service-Oriented Systems. In Luciano Baresi, Piero Fraternali, and Geert-Jan Houben, editors, *Web Engineering*, volume 4607 of *Lecture Notes in Computer Science, LNCS*, pages 563–568. Springer, 2007.
- [63] Natallia Kokash and Vincenzo DAndrea. Evaluating Quality of Web Services: A Risk-Driven Approach. In Witold Abramowicz, editor, *Business Information Systems*, volume 4439 of *Lecture Notes in Computer Science, LNCS*, pages 180–194. Springer, 2007.
- [64] D. Koller and N. Megiddo. The complexity of two person zero-sum games in extensive form. *Games and Economic Behavior*, (4):528–552, 1992.
- [65] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *16th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science, LNCS*, pages 404–413. Springer-Verlag, Berlin, 1999.
- [66] Richard E. Ladner. The circuit value problem is log space complete for p. *ACM Special Interest Group on Algorithms and Computation Theory, SIGACT News*, 7(1):18–20, 1975.

- [67] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.
- [68] Frank Leymann. Web Services Flow Language (WSFL 1.0). Technical report, IBM, 2001.
- [69] Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994.
- [70] A. Mas-Colell, M.D. Whinston, and J.R. Green. *Microeconomic Theory*. Oxford University Press, New York, 1995.
- [71] Marios Mavronicolas, Burkhard Monien, and Klaus Wagner. Weighted boolean formula games. In Xiaotie Deng and Fan Graham, editors, *Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science, LNCS*, pages 469–481. Springer, 2007.
- [72] J.C.C. McKinsey. Isomorphism of games, and strategic equivalence. *Annals of Mathematics Study (24)*, pages 117–130, 1950.
- [73] A. McLennan and J. Berg. Asymptotic expected number of Nash equilibria of two-player normal form games. *Games and Economic Behavior*, 51:264–295, 2005.
- [74] Robin Milner. *Communication and concurrency*. Prentice Hall International (UK) Ltd., 1995.
- [75] Jayadev Misra. A Programming Model for the Orchestration of Web Services. In *SEFM'04: Proceedings of the Software Engineering and Formal Methods, Second International Conference*, pages 2–11, Washington, DC, USA, 2004. IEEE Computer Society.
- [76] Jayadev Misra and William Cook. Computation Orchestration: A Basis for Wide-Area Computing. *Software and Systems Modeling (SoSyM)*, 6(1):83–110, 2007.
- [77] Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. When selfish meets evil: byzantine players in a virus inoculation game. In *PODC'06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 35–44, New York, USA, 2006. ACM.
- [78] Pierfrancesco La Mura. Game networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 335–342. UAI, 2000.
- [79] John Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, 1950.

-
- [80] John Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):pp. 286–295, 1951.
- [81] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.
- [82] Noam Nisan, T. Roughgarden, E Tardos, and V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, USA, 2007.
- [83] M. J. Osborne. *An Introduction to Game Theory*. Oxford University Press, 2003.
- [84] Martin J Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
- [85] Christos Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, STOC’01, pages 749–753, New York, USA, 2001. ACM.
- [86] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994.
- [87] Christos H. Papadimitriou. Game theory and mathematical economics: a theoretical computer scientist’s introduction. In *42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, pages 4–8. IEEE Computer Soc., Los Alamitos, CA, 2001.
- [88] Mike P. Papazoglou and Dimitrios Georgakopoulos. Introduction: Service-oriented computing. *Communication of the ACM*, 46(10):24–28, 2003.
- [89] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of ACM*, 27:228–234, 1980.
- [90] B. Peleg, J. Rosenmuller, and P. Sudholder. The Canonical Extensive Form of a Game Form - Part I - Symmetries, 1996. Institute of Mathematical Economics, Bielefeld University, Working Paper No.253.
- [91] Chris Peltz. Web Services Orchestration and Choreography. *Computer*, 36(10):46–52, 2003.
- [92] Michal Penn, Maria Polukarov, and Moshe Tennenholtz. Congestion games with failures. In *Proceedings of the 6th ACM conference on Electronic commerce*, EC’05, pages 259–268, New York, USA, 2005. ACM.

- [93] Michal Penn, Maria Polukarov, and Moshe Tennenholtz. Congestion games with load-dependent failures: identical resources. In *Proceedings of the 8th ACM conference on Electronic commerce*, EC'07, pages 210–217, New York, USA, 2007. ACM.
- [94] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
- [95] Nicholas Pippenger. The complexity of computations by networks. *IBM Journal of Research and Development*, 31(2):235–243, 1987.
- [96] Imelda Young Powers. Limiting Distributions of the Number of Pure Strategy Nash Equilibria in n-Person Games. *International Journal of Game Theory*, 19:277–286, 1990.
- [97] John H. Reif and Stephen R. Tate. On threshold circuits and polynomial computation. *SIAM Journal on Computing*, 21(5):896–908, 1992.
- [98] Wolfgang Reisig. *Petri Nets: an Introduction*. Springer-Verlag, New York, USA, 1985.
- [99] S. Rosario, A. Benveniste, S. Haar, and C. Jard. Probabilistics QoS and soft contracts for transaction based Web services orchestrations. *Services Computing, IEEE Transactions on*, 1(4):187–200, 2008.
- [100] R.W. Rosenthal. A Class of Games Possessing Pure-Strategy Nash Equilibria. *International Journal of Game Theory*, 2:65–67, 1973.
- [101] Davide Sangiorgi and David Walker. *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.
- [102] Grant Schoenebeck and Salil Vadhan. The computational complexity of Nash equilibria in concisely represented games. In *EC'06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 270–279, New York, USA, 2006. ACM.
- [103] Yoav Shoham. Computer science and game theory. *Communication of the ACM*, 51:74–79, 2008.
- [104] G. C. Silaghi, A. Arenas, and L. Silva. Reputation-based trust management systems and their applicability to grids. Technical report, Institute on Programming Models, CoreGRID-Network of Excellence, 2007.
- [105] Katharina Siorpaes and Martin Hepp. Games with a Purpose for the Semantic Web. *IEEE Intelligent Systems*, 23:50–60, 2008.

- [106] Alan Stewart, Maurice Clint, Terry Harmer, Peter Kilpatrick, Ron Perrott, and Joaquim Gabarro. Assessing the reliability and cost of web and grid orchestrations. In *Proceedings of the The Third International Conference on Availability, Reliability and Security, ARES 2008*, volume 0, pages 428–433, Los Alamitos, CA, USA, 2008. IEEE Computer Society.
- [107] Alan Stewart, Joaquim Gabarro, Maurice Clint, Terence J. Harmer, Peter Kilpatrick, and R. Perrott. Managing Grid Computations: An ORC-Based Approach. In Minyi Guo, Laurence Tianruo Yang, Beniamino Di Martino, Hans P. Zima, Jack Dongarra, and Feilong Tang, editors, *Parallel and Distributed Processing and Applications, 4th International Symposium, ISPA 2006*, volume 4330 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 2006.
- [108] P. Sudholter, J. Rosenmuller, and B. Peleg. The Canonical Extensive Form of a Game Form - Part II - Representation. *Journal of Mathematical Economics*, 33(3):299–338, 2000.
- [109] S. Thatte. XLANG: Web Services for Business Process Design. 2001.
- [110] J. Torán. Personal Communication.
- [111] J. van Benthem. *When are two games the same?* ILLC Scientific Publications. Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 1999.
- [112] Denis Verdon and Gary McGraw. Risk Analysis in Software Design. *IEEE Security and Privacy*, 2:79–84, 2004.
- [113] J Watson. *Strategy: An Introduction to Game Theory*. W. W. Norton & Company, 2002.
- [114] Bin Yu and Munindar P. Singh. An Evidential Model of Distributed Reputation Management. In *Proceedings of First International Conference on Autonomous Agents and MAS*, pages 294–301. ACM Press, 2002.

