TU WIEN Informatics

Extending Game-Theoretic Security of Blockchain Protocols with Compositional Reasoning and Conditional Actions

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Logic and Computation

eingereicht von

Ivana Bocevska, BSc
Matrikelnummer 11929220

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ. Prof. Dr. techn. Laura Kovács, MSc
Mitwirkung: Dipl.-Ing. Dr. techn. Sophie Rain, BSc

Wien, 1. Juni 2025

Ivana Bocevska	Laura Kovács

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Extending Game-Theoretic Security of Blockchain Protocols with Compositional Reasoning and Conditional Actions

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Logic and Computation

by

Ivana Bocevska, BSc
Registration Number 11929220

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Prof. Dr. techn. Laura Kovács, MSc
Assistance: Dipl.-Ing. Dr. techn. Sophie Rain, BSc

Vienna, June 1, 2025

Ivana Bocevska	Laura Kovács

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Ivana Bocevska, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 1. Juni 2025

Ivana Bocevska

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Laura Kovács and Sophie Rain, for their support throughout the work on this thesis and their dedication as advisors. Laura provided invaluable, continuous feedback and with her vast experience, greatly enhanced the quality of the work. Sophie generously dedicated her time, and together we engaged in numerous brainstorming and pair-programming sessions that proved to be incredibly productive.

Moreover, I would like to thank Anja Petković Komel and Michael Rawson for the numerous insightful discussions and ideas that truly broadened my perspective and helped guide my research.

Lastly, I dearly thank my close family for their unwavering support and encouragement. They have been by my side throughout my journey, always proud and excited about my achievements, and offering comfort and motivation whenever I needed it.

This work was partially supported by the Austrian Science Fund (FWF) SPyCoDe Grant 10.55776/F85.

Kurzfassung

Für die Sicherheitsanalyse von Blockchain-Technologien haben sich spieltheoretische Ansätze als nützlich erwiesen. Solche Ansätze untersuchen Protokolle aus einer wirtschaftlichen Perspektive, insbesondere durch die Erforschung der *wirtschaftlichen Anreize*, die das Nutzerverhalten steuern. Auf diese Weise wird sichergestellt, dass ein Abweichen vom ehrlichen Verhalten eines Protokolls keine finanziellen Vorteile mit sich bringt: solange sich die NutzerInnen an das Protokoll halten, kann ihnen kein finanzieller Schaden entstehen, unabhängig davon, wie sich andere verhalten.

Eine solche wirtschaftliche Analyse von Blockchain-Protokollen kann als ein automatisiertes Schlussfolgerungsproblem in der Arithmetik der reellen Zahlen in der Prädikatenlogik 1. Stufe formuliert werden, wodurch das spieltheoretische Schlussfolgerungsproblem durch die Erfüllbarkeit einer Menge von Formeln, bekannt als Satisfiability Modulo Theories (SMT) Solving, gelöst wird. Diese Tools stehen jedoch vor großen Herausforderungen in Bezug auf *Skalierbarkeit* und *Ausdruckskraft*. Das *Skalierbarkeitsproblem* entsteht, weil aktuelle Ansätze den gesamten Spielbaum, der möglicherweise Millionen von Interaktionen umfasst, auf einmal analysieren. Dabei sind die generierten Formeln sehr groß und komplex. Hinsichtlich der *Ausdruckskraft* sind bestehende Frameworks nicht in der Lage, Spiele zu modellieren und zu analysieren, die externe Faktoren einbeziehen – wie etwa Preisschwankungen oder Wechselkurse, die vom aktuellen Weltzustand abhängen und außerhalb der Kontrolle der Spieler liegen.

Um diesen Herausforderungen zu begegnen, schlagen wir eine auf dem *Teile-und-Herrsche-Prinzip* basierende Methode, welche auch die Modellierung großer Instanzen ermöglicht. Die Grundidee besteht darin, das Modell in seine Einzelteile zu zerlegen, die Sicherheitsanalyse auf diesen kleineren Teilen des Modells durchzuführen und die Ergebnisse anschließend zusammenzuführen, um das Gesamtergebnis der Analyse zu erhalten. Wir stellen eine korrekte und vollständige Automatisierung dieser Methode bereit sowie eine automatisierte Generierung von Strategien und Gegenbeispielen für die (widerlegten oder bestätigten) Sicherheitseigenschaften. Experimentelle Ergebnisse zeigen, dass diese Methode gut auf Spiele mit Millionen von Knoten skaliert und somit die Sicherheitsanalyse großer, realer Protokolle ermöglicht.

Darüber hinaus nutzen wir Teile-und-Herrsche-Schlussfolgerungstechniken, um die *Ausdruckskraft* des Frameworks zu erweitern und die Modellierung sowie Analyse von Protokollen mit durch externe Faktoren beeinflussten Aktionen zu ermöglichen. Diese nennen

wir *Spiele mit bedingten Aktionen*. Zu diesem Zweck definieren wir zentrale Konzepte wie Spielbäume, Sicherheitseigenschaften, Strategien, Gegenbeispiele, ehrliches Verhalten und verwandte Begriffe neu. Die Analyse von Spielen mit bedingten Aktionen wurde ebenfalls automatisiert und anhand von ersten Benchmarks mit nicht-trivialen Bedingungen evaluiert.

Abstract

Game-theoretic security analysis of blockchain technologies has proven highly valuable. Such analysis examines protocols from an economic perspective, specifically by exploring the *economic incentives* that drive user behavior. Thus, it ensures that deviating from the intended, honest behavior of a protocol is not financially beneficial: as long as users follow the protocol, they cannot be financially harmed, regardless of how others behave.

Such an economic analysis of blockchain protocols can be encoded as an automated reasoning problem in the first-order theory of real arithmetic, thereby reducing game-theoretic reasoning to satisfiability modulo theories (SMT) solving. However, existing frameworks face significant challenges in terms of *scalability* and *expressivity*. The *scalability* issue arises because current approaches treat the entire game tree, potentially involving millions of interactions, as a single SMT instance. In terms of *expressivity*, existing frameworks fall short in modeling and analyzing games that involve external factors – such as price fluctuations or exchange rates of currencies – that depend on the current world state and lie beyond the players' control.

Therefore, to address these challenges, we propose a *divide-and-conquer* security analysis based on *compositional reasoning* over game trees. In our approach, we decompose games into subgames, ensuring that changes to one subgame do not require re-analyzing the entire game, but only its ancestor nodes. We provide a sound and complete automation of this method, as well as automated compositonal extraction of strategies and counterexamples for the (dis)proved security properties. Experimental results show that compositional reasoning scales well to games with millions of nodes, enabling security analysis of large real-life protocols.

Additionally, we leverage compositional reasoning techniques to enhance the *expressivity* of the framework, enabling the modeling and analysis of protocols with actions influenced by external factors, which we refer to as games with *conditional actions*. To support this, we redefine key concepts such as game trees, security properties, strategies, counterexamples, honest behavior, and related notions. The analysis of games with conditional actions has also been automated and evaluated on an initial set of benchmarks involving non-trivial conditions.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Throughout the last few years, blockchain technology has experienced a significant rise in popularity. Examples include the use of cryptocurrencies, online auditing, decentralized finance, etc. As a consequence, security guarantees of such applications need to be established. Most of the existing approaches focus on cryptographic correctness, such as making sure that secret information cannot be revealed [4, 15, 16]. However, such approaches neglect the economic implications of correctness, namely, whether it is possible for a user or group of users to profit from malicious yet cryptographically correct behavior within the protocol.

In this respect, game-theoretic approaches have proven to be quite helpful. Rain *et al.* introduce a game-theoretic model in which protocols are modeled as extensive form games (EFGs) [20]. In this manner, all possible interactions between the users (players) are captured and one can analyze whether a player or a group of players can benefit from deviating from the intended (honest) behavior. As defined in [20], a protocol is secure if the following properties are satisfied:

- *Incentive compatibility:* the intended (honest) behavior is the most profitable one, i.e., users have no incentive to deviate.
- *Byzantine fault tolerance:* as long as users follow protocol instructions, they cannot be harmed, independently of how other users behave.

As a follow-up of [20], the work of [6] introduces CheckMate, an automated reasoning framework for proving the security of a protocol modeled as a game. To this end, game-theoretic security decisions are modeled as first-order arithmetic problems and solved using satisfiability modulo theory (SMT) solving.

1.2 Problem Statement

Even though game-theoretic approaches have proven to be quite helpful, they are not without limitations. In terms of *scalability*, modeling one big game is inefficient and imposes scalability issues. Consequently, SMT queries capturing the functionalities of game trees with millions of nodes present a challenge for the solver. An interesting observation is that many games contain similar or identical subgames multiple times. For instance, the model of closing a channel in the Lightning network [20] contains 8 subtrees with identical structure. Such subtrees are currently analyzed multiple times.

Furthermore, in terms of *expressivity*, there are protocols that require modeling and reasoning about uncontrollable (external) game aspects that are not controllable by any player, such as e.g., prices and exchange rates of currencies. This aspect is not adequately supported by current models.

This master thesis addresses the mentioned limitations by employing *compositional (modular) reasoning* in order to overcome the *scalability issues*, thereby allowing identical subtrees to be solved only once. Further, is also to improves *expressivity* by enabling support of uncontrollable game aspects, through a concept we call *conditional actions*.

The thesis enhances the framework presented in [6] by utilizing compositional (modular) reasoning. Simply put, the objective of the work is to introduce a divide-and-conquer approach to game-theoretic security analysis. This poses complex theoretical problems, as game-theoretic properties are not compositional in general. As a consequence, the existing C++ implementation [6] is extensively adapted in this respect.

The benefits of such an approach are manyfold. First of all, a compositional approach offers improvements in *scalability*, by allowing to solve (potentially big) subtrees independently and to reuse the obtained results in the supertree. This is especially useful when an identical subtree occurs multiple times in the main supertree; it needs to be solved only once using this approach. Furthermore, SMT queries remain compact, making them less demanding for the solver. Additionally, such an approach enhances reusability, as specific parts of the protocol (modeled as subtrees) could be effortlessly substituted with new ones.

Currently, another main restriction of the existing framework [6] is the lack of support for modeling and analyzing uncontrollable game aspects, such as price changes and exchange rates of currencies. To overcome this limitation and enhance the *expressivity* of the framework, compositionality can be employed for reasoning about uncontrollable aspects, a challenge that is also addressed in this master thesis.

1.3 State of the Art

Prospects of using compositionality in computer science, especially in economics and game theory are addressed in some recent works.

In [14], a new strain of game theory – Compositional Game Theory (CGT) – was introduced. A new representation of games, namely open games, was developed, which describe games played relative to a given environment and visualized by string diagrams. The authors define a number of operators for building up complex games from smaller simpler games. They point out that studying games compositionally is not a trivial task, since the interaction between games has to be considered. However, introduction of some new operators for certain classes of games and (full) automation of the approach are open problems. Moreover, the approach is restricted to constant numeric utilities instead of symbolic utilities and assumes rational behavior of players. In contrast to this work, we employ symbolic utilities and capture honest/rational behavior, and thus security, in games.

A compositional approach for program verification of smart contracts is presented in [23]. Instead of verifying a smart contract relative to all users, few representative users are chosen, thereby avoiding intractability due to state explosion. Each of the representatives abstracts concrete users symmetric to each other relative to the smart contract. Nonetheless, this approach requires a manually provided summary for the behavior of each representative and deals only with the verification of the Solidity language itself, hence not considering incentive compatibility and Byzantine fault tolerance.

A very prominent application of compositionality in computer science other than game-theoretic security is Facebook's static analysis method via the tools Infer and Zoncolan [8]. Program analysis of a composite program is performed based on the analysis results of its parts and by means of combining them. Through formal reasoning and compositionality, static analysis becomes scalable to large codebases that are collaboratively developed by thousands of programmers. Deployments similar to Infer's implementation exist in Amazon, Microsoft and Google [17].

In [6], a fully automated framework for reasoning about game-theoretic security of blockchain protocols is presented. In their work they point out the need of scalability solutions as well as support of conditional actions. To the best of our knowledge, there exists no extension to the framework in this respect. As previously discussed, this will be the main contribution of this master thesis.

Finally, a divide-and-conquer approach for parallelizing SMT queries is presented in [24]. The authors emphasize that solver performance is often the primary bottleneck in many cases and propose partitioning strategies to decompose an SMT problem into subproblems. The approach discussed in this master thesis differs in that it avoids generating large SMT queries altogether, focusing instead on localized aspects – specifically, smaller trees. While the method outlined in the paper is not directly applicable to our scenario, its core idea aligns with the overarching approach employed in this thesis.

1.4 Contributions

We present a method for compositional game-theoretic security, as well as a method for expressing and reasoning about uncontrollable game aspects, which we call conditional actions. To the best of our knowledge, there exist no other approaches in these regards. We implement our work as the next generation of the tool CheckMate [6] and call it CheckMate 2.0. Our experiments show that divide-and-conquer reasoning makes it possible to model and analyze complex real-world protocols with millions of nodes. Moreover, such reasoning enables support for the analysis of games that involve uncontrollable external factors.

We bring the following contributions:

- We propose a *compositional framework for game-theoretic security analysis*. This framework introduces player-dependent security properties, enabling divide-and-conquer reasoning over game trees. We decompose games into subgames, ensuring that the resulting reasoning remains both sound and complete.
- We propose the use of divide-and-conquer algorithmic reasoning to interleave compositional modeling and security analysis. This approach combines reasoning between subgames and their supergames (parent games), incorporating the security results of subgames into the leaves of their corresponding supergames.
- Our compositional framework supports the *generation of counterexamples* when security properties are violated and *extraction of game strategies* otherwise.
- We define the notion of *conditional actions* and introduce related concepts, thereby redefining game-theoretic security analysis in this context. In addition, we present algorithms for analyzing models that include conditional actions.

The material presented in Chapter 3–Chapter 5, which focuses on using compositional reasoning for scalability improvement, is based on a paper accepted the peer-reviewed venue of OOPSLA 2025 – the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, co-authored by the author of this thesis; a preprint is available at [5]. The contribution in Chapter 6, addressing conditional actions, is unpublished.

1.5 Outline

In the following chapters, we discuss our approach for compositional (modular) reasoning, as well as support for conditional actions. The rest of the thesis is organized as follows. Chapter 2 lays the groundwork by introducing essential game-theoretical concepts and security properties. Chapter 3 revises the notions of the existing security properties and redefines them compositionally. Chapter 4 presents an overview of the theory underpinning the compositional approach. In Chapter 5, the automation and evaluation

of this approach are discussed. Chapter 6 demonstrates how compositional methods can be applied to analyze models involving conditional actions. Lastly, the thesis concludes in Chapter 7.

CHAPTER 2

Preliminaries

Before exploring compositionality and discussing the application of divide-and-conquer techniques, it is essential to first understand the game-theoretic foundation of the topic, the analyzed security properties and the automation of the analysis process. In the following, we introduce the relevant concepts by adapting definitions from [6, 21] to our setting.

2.1 Game Theory

For game-theoretic security analysis, blockchain protocols (or any other relevant protocols) are modeled as games, representing the interactions between the parties involved, referred to as the *players* in the context of game theory.

A *game* is a static finite object with finitely many players. Players choose from a finite set of *actions* until the game ends, whereupon they receive a *utility*. The focus is on perfect information *Extensive Form Games* (EFGs) [18] in which the actions are chosen sequentially with full knowledge of all previous actions. Games may yield collective benefit or loss, i.e. they are not necessarily *zero-sum*.

Definition 2.1 (Extensive Form Game – EFG)**.** *An* extensive form game $\Gamma = (N, G)$ *is determined by a finite non-empty set of players* N *together with a finite tree* $G = (V, E)$*. A game path* $h = (e_1, ..., e_n)$*, with* $e_i \in E$*, that starts from the root of* G *is called a* history*. We denote the set of histories* $\mathscr{H}$*. There is a bijection between nodes* $v \in V$ *and histories* $h \in \mathscr{H}$ *that lead to these nodes.*

- *A history that leads to a leaf is called* terminal *and belongs to the set of* terminal histories $\mathscr{T} \subseteq \mathscr{H}$*. Terminal histories* t *are associated with a* utility *for each player.*

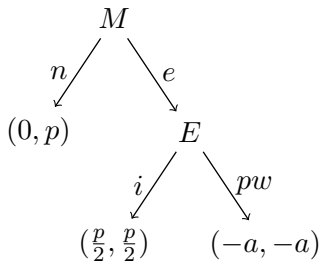

Figure 2.1: Market Entry game Γ_{me}, with $a, p > 0$. Utility tuples state M's utility first, E's second.

- Non-terminal histories *are those histories that are not terminal. Non-terminal histories h have assigned a next player denoted as $P(h) \in N$. Player $P(h)$ chooses from the set $A(h)$ of possible actions following h.*

In an EFG Γ we call a terminal history h^* *honest* if it represents the expected behavior in Γ. An EFG Γ can have many honest histories; security analysis over Γ is always performed relative to a chosen and fixed honest history.

Example 2.1 (Market Entry Game)**.** *Consider the Market Entry game Γ_{me} of Figure 2.1. In this game there are two players: M representing a new company and E an established company. At the root, it is the turn of player $P(\emptyset) = M$ to choose from actions $A(\emptyset) = \{n, e\}$. Action n represents* not *entering the market, producing a terminal history (n) where M gets 0 utility and E gets all of the profits $p > 0$. Action e represents entering the market, in which case E can respond by either ignoring this move (action i) and thus splitting profits equally, or by entering a price war (action pw) that damages both players.*

In game theory, utilities are typically represented as numeric constants. Following [6], we generalize utilities to *symbolic* terms in real arithmetic, allowing us to encode all possible values within the given constraints. Variables and numeric constants are evaluated over the real numbers extended by a finite set of *infinitesimals*, closer to zero than any real number. Infinitesimals are used to model subtle subjective preferences or (in)conveniences, not directly related to funds, such as opportunity cost. We model infinitesimals with terms over $\mathbb{R} \times \mathbb{R}$, ordered lexicographically: the first component represents the real part, the second the infinitesimal. We write real for the first projection and avoid writing pairs, using $a, b, c \ldots$ for real variables, and $\alpha, \beta, \gamma, \ldots$ for infinitesimals. The utility term $a + \alpha - \varepsilon$ is therefore represented as $(a, 0) + (0, \alpha) - (0, \varepsilon) = (a, \alpha - \varepsilon)$.

Example 2.2. *We could modify the Market Entry game from Figure 2.1 by adding an infinitesimal $\alpha > 0$ to the utility of player M at (e, i). The utility $\frac{p}{2} + \alpha$ represents half of the profit p and the additional benefit of entering the market α, as M is motivated to establish a new entity on the market.*

We now introduce notions for EFGs that are necessary for defining game-theoretic security properties, as defined in [5].

Definition 2.2 (EFG Properties)**.** *Let $\Gamma = (N, G)$ be an EFG.*

Strategy *A* strategy *σ for a group of players $S \subseteq N$ is a function mapping non-terminal histories $h \in \mathscr{H} \setminus \mathscr{T}$, where one of the players in group S has a turn $P(h) \in S$, to the possible actions $A(h)$. We write $\mathscr{S}_S$ for the set of strategies for group S, and $\mathscr{S}$ for $\mathscr{S}_N$ which we call* joint strategies*. We refer to the union of strategies with disjoint domains as a* combined strategy *and denote it as a tuple. To combine e.g. $\sigma_S \in \mathscr{S}_S$ and $\tau_{N-S} \in \mathscr{S}_{N-S}$, we write $(\sigma_S, \tau_{N-S}) \in \mathscr{S}$.*

Resulting History *The* resulting terminal history *$H(\sigma)$ of a joint strategy $\sigma \in \mathscr{S}$ is the unique history obtained by following chosen actions in σ from root to leaf.*

Following Honest History *A strategy for a player p follows the honest history h^* if, at every node along the honest history, where p is making a choice, the strategy chooses the action in h^*. For every other node, there is no constraint.*

Utility Function *The* utility function *$u_p(\sigma)$ assigns to player $p \in N$ their utility at the resulting terminal history of the joint strategy $\sigma \in \mathscr{S}$, that is $u_p(\sigma) := u_p(H(\sigma))$. We sometimes write all player utilities for a joint strategy $\sigma \in \mathscr{S}$ as $u(\sigma)$, denoting a tuple of size $|N|$.*

Subgame Subgames *$\Gamma_{|h}$ of Γ are formed from the same set N of players and a subtree of G, and are therefore identified by the history h leading to the subtree $G_{|h}$. Histories $\mathscr{H}_{|h}$ of $\Gamma_{|h}$ are histories in $\mathscr{H}$ with prefix h, strategies $\sigma_{|h} \in \mathscr{S}_{|h}$ of $\Gamma_{|h}$ are strategies restricted to the nodes in $\mathsf{G}_{|h}$ and the utility function $u_{|h}$ of $\Gamma_{|h}$ assigns each joint strategy $\sigma_{|h} \in \mathscr{S}_{|h}$ the utility of the yielded leaf $u_{|h}(\sigma_{|h}) := u(h, H(\sigma_{|h}))$. This includes trivial subgames: leaves or the entire tree Γ at the empty history.*

Supergame *If h' is a prefix of h, $\Gamma_{|h'}$ is a supergame of $\Gamma_{|h}$.*

Subtree along/off Honest History *Let h^* be the honest history. A subgame $\Gamma_{|h}$ is* along *the honest history iff h is a prefix of h^*; that is, there is a history $g \in \mathscr{H}_{|h}$ in the subtree such that $(h, g) = h^*$. Otherwise, $\Gamma_{|h}$ is* off *the honest history.*

Intuitively, a *subgame* is the part of the game that is still to be played after some actions have been taken already. A *supergame* of a subgame is any game tree that embeds the subgame as the subtree. For the sake of readability, we use *subgame/subtree* and *supergame/supertree* interchangeably. We write $u(\sigma_S, \tau_{N-S}) := u((\sigma_S, \tau_{N-S}))$ for the combined strategy $(\sigma_S, \tau_{N-S}) \in \mathscr{S}$. For history $k \in \mathscr{H}$, we write $k_{|h}$ to express the suffix of k after h, that is $(h, k_{|h}) = k$.

Example 2.3. *Consider again the Market Entry game Γ_{me} in Figure 2.1. A joint strategy τ could be M to take action n initially, and player E taking i after (e). M's strategy $\tau_M \in \mathscr{S}_M$ takes action n initially. E receives $u_E(\tau) = p$. The history resulting from τ is (n), and τ is a strategy extending history (n).*

The subgame for history (e) has players $\{M, E\}$ and a tree where E must choose between action i with utility $(\frac{p}{2}, \frac{p}{2})$ and action pw with utility $(-a, -a)$. For honest history (e, i) the subtree $\Gamma_{me|(e)}$ after action e is along *the honest history (e, i), whereas the trivial subtree $\Gamma_{me|(n)}$ after action n is* off *the honest history (e, i).*

The Market Entry game has $2 \times 2 = 4$ joint strategies as M chooses from two possible actions, and independently E picks one action out of two in the subtree $\Gamma_{me|(e)}$.

2.2 Game-Theoretic Security Properties

Once real-life protocols are modeled as extensive form games (EFGs), the game-theoretic security analysis of a protocol can be reduced to the game-theoretic security analysis of the corresponding EFG. In this context, we introduce *honest players*, who adhere strictly to the protocol's intended behavior and do not deviate from the prescribed scenario (i.e. engage in any form of cheating). Additionally, we consider *rational players* – those who may deviate from the intended behavior if incentivized, that is, if a deviation offers them a strictly better payoff. Finally, we also account for *Byzantine players*, whose behavior is entirely unpredictable; they may deviate from the protocol with the sole intention of harming others. Recall that an *honest history* is the terminal history that results from all players behaving honestly.

For a protocol to be secure, adversaries must not be able to perform an attack, whether for personal gain or to harm another party. In this regard, according to [20, 26], a protocol is considered *game-theoretically secure* if it satisfies the following two properties:

(P1) **Byzantine Fault-Tolerance.** Honest players do not suffer loss, even in the presence of adversaries. In other words, in a secure protocol an honest player will not receive negative utility, independent of others' behavior. Therefore, there are no "attacks" where somebody is harmed.

(P2) **Incentive Compatibility.** Rational agents do not deviate from the honest behavior, as it yields the best payoff. Hence, in a secure protocol, a rational "attacker" is behaving honestly and no adversary gets personal gain by deviation.

The two properties above are refined and ensured through four security properties as in [6, 20], which we discuss in the following. For all definitions, we consider a fixed arbitrary EFG $\Gamma = (N, G)$ and provide the definitions relative to it.

Definition 2.3 (Weak Immunity of a Strategy)**.** *A joint strategy $\sigma \in \mathscr{S}$ in EFG Γ is* weak immune *if all players p that follow σ always receive non-negative utility:*

$$\forall p \in N \; \forall \tau \in \mathscr{S}.\; u_p(\sigma_p, \tau_{N-p}) \geq 0 \; . \tag{wi(Γ)}$$

Weak immunity ensures that behaving honestly never causes the players to lose resources and thus captures property (P1). In some cases, weak immunity can be too restrictive. Hence, property *weaker immunity* is proposed as an alternative to ensure that (P1) holds.

Definition 2.4 (Weaker Immunity of a Strategy)**.** *A joint strategy σ in EFG Γ is* weaker immune *if all players p that follow σ always receive at least a negative infinitesimal:*

$$\forall p \in N \; \forall \tau \in \mathscr{S}.\; \mathsf{real}(u_p(\sigma_p, \tau_{N-p})) \geq 0 \; . \tag{weri(Γ)}$$

Property (P2) is ensured by *collusion resilience* and *practicality*. Collusion resilience guarantees that players do not have an incentive to form a colluding group and deviate jointly from the honest behavior.

Definition 2.5 (Collusion Resilience of a Strategy)**.** *A joint strategy σ in EFG Γ is* collusion resilient *if colluding players $S \subset N$ cannot profit from deviation:*

$$\forall S \subset N \; \forall \tau \in \mathscr{S}.\; \sum_{p \in S} u_p(\sigma) \geq \sum_{p \in S} u_p(\sigma_{N-S}, \tau_S) \; . \tag{cr}$$

Last but not least, the property practicality ensures that the honest behavior is rational. That is, even if players behave greedily, the honest behavior is the one that provides the best utility.

Definition 2.6 (Practicality of a Strategy)**.** *A joint strategy σ in EFG Γ is* practical *if it is a subgame perfect equilibrium, i.e. a Nash equilibrium in every subgame:*

$$\forall h \in \mathscr{H} \; \forall p \in N \; \forall \tau \in \mathscr{S}_{|h}.u_{|h,p}(\sigma_{|h}) \geq u_{|h,p}(\tau_p, \sigma_{|h,N-p}) \; . \tag{pr(Γ)}$$

Definition 2.7 (Security Properties of Strategies and Histories)**.** *A* strategy *$\sigma \in \mathscr{S}$ of a game Γ is weak(er) immune, collusion resilient, or practical, iff it can serve as the witness strategy in* (wi(Γ)), (weri(Γ)), (cr), *or* (pr(Γ)), *respectively.*

A terminal history *$t \in \mathscr{T}$ of the game Γ is weak(er) immune, collusion resilient, or practical, iff there exists a strategy $\sigma \in \mathscr{S}$ that has the respective property and extends history t, that is $H(\sigma) = t$.*

Having defined the security properties of strategies and histories, we can introduce the notion of *secure history*, as proposed in [20].

Definition 2.8 (Secure History)**.** *A terminal history h^* of an EFG Γ is* secure *if there are three strategies $\sigma_1, \sigma_2, \sigma_3$ extending h^*, such that they are weak immune, collusion resilient and practical, respectively.*

Example 2.4 (Security of Market Entry game with Honest History (e, i))**.** *Let us once again recall the Market Entry game depicted in Figure 2.1 and fix the only honest history to be (e, i). In the following, we examine whether the Market Entry game with honest history (e, i) is secure, i.e. whether there are weak immune, collusion resilient and practical strategies that extend the history (e, i).*

First of all, consider weak immunity. Out of the four joint strategies that can be constructed for the Market Entry game, the only one that extends the honest history (e, i) is the strategy where M picks e at the beginning and E picks i after (e). We represent this strategy by $[\emptyset \to e,\ (e) \to i]$. We need to determine whether this strategy is weak immune, that is, no player can get a negative utility, no matter how the other players behave. We can see that this is not the case. If player M takes action e, player E might decide to deviate from the honest behavior and take action pw instead of i, causing harm to player M. In this case, M follows the honest behavior, but can still end up with a negative utility, depending on how E plays. Because of this reason, the Market Entry game with honest history (e, i) is not weak immune.

Next, we analyze collusion resilience. To achieve this, we consider all strict subgroups of players and need to check whether there is a strategy extending the honest history, such that if any strict subgroup of players chooses to jointly deviate from this strategy they will not receive a utility strictly better than the honest one by the end of the game. Recall that out of the four joint strategies that can be constructed for the Market Entry game, the only one that extends the honest history (e, i) is the strategy $[\emptyset \to e,\ (e) \to i]$, so we analyze this strategy. Since the Market Entry game features two players, the only strict subgroups of players are M and E. We inspect them individually. Starting with M, we can conclude that if M decides to deviate and takes action n instead of action e, the game will end and M will receive utility 0. The honest utility for M is $\frac{p}{2}$, where p is a positive number. As 0 is not strictly better than $\frac{p}{2}$, collusion resilience is not violated in terms of the first possible collusion group M and the honest history (e, i). Moving to the second subgroup that we need to consider, namely E, we see that if E chooses to deviate and takes action pw instead on i after non-terminal history (e), the game will end with a negative utility for E, namely $-a$, which is trivially not strictly better than the honest utility for E, namely $\frac{p}{2}$. Thus, collusion resilience is also not violated in terms of the second possible collusion group E and the honest history (e, i). Consequently, the Market Entry game with honest history (e, i) is collusion resilient.

Last, we inspect practicality. For this property, we need to also consider each subgame of the original game. The Market Entry game has two subgames - the trivial one, which is the whole game and the subgame after action (e) where only player E has a choice. We analyze the only strategy which extends the honest history, namely $[\emptyset \to e,\ (e) \to i]$. In the subtree after E, the rational decision for E is to choose action i, as this yields a utility for them which is strictly better that the one obtained after pw, namely $-a$. Next, we move one level up – at the beginning of the game. Assuming E takes action i we analyze which choice is best for M. We observe that it is not rational for M to deviate and pick n, because $0 < \frac{p}{2}$. Hence, the strategy $[\emptyset \to e,\ (e) \to i]$ is practical and consequently, the

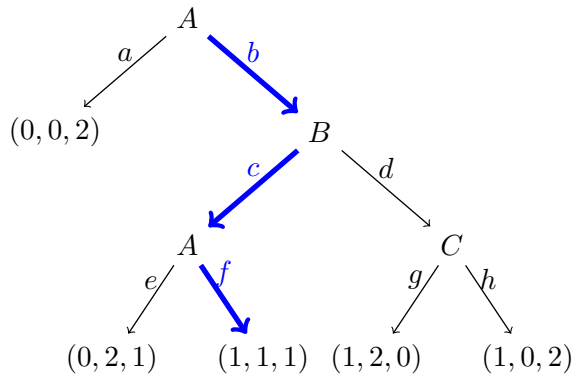

Figure 2.2: Example Γ_{cr} for the property Collusion Resilience.

Market Entry game with honest history (e, i) is practical as well.

In summary, the Market Entry game with honest history (e, i) is collusion resilient and practical, but not weak immune, and thus, not secure.

As a final point for this section, let us consider a non-trivial example for collusion resilience, namely an example with more than two players. This is a more complex example, as it allows colluding groups that are not necessarily singletons.

Example 2.5 (Security of an Example for Collusion Resilience)**.** *Consider the example given in Figure 2.2, with honest history (b, c, f). Let us inspect the strategy $[\emptyset \to b, (b) \to c, (b, c) \to f, (b, d) \to h]$. We consider all possible colluding groups and check if they can deviate in a way such that they receive a joint utility greater than the honest one. $\{A\}$ cannot deviate profitably, as it would lead to utility 0 (both after $\emptyset$ and (b, c)), which is not better than the honest utility for A, namely 1. Should $\{B\}$ choose to deviate (alone) after (b), they will get utility 0 instead of 2, because according to the strategy we fixed, C will choose action h after the deviation of B. Player C cannot deviate alone, since if A and B are honest, the game will end without C having their turn. Next, we consider the remaining possible colluding groups, namely $\{A, B\}$, $\{A, C\}$ and $\{B, C\}$. If A and B decide to collude together, they either get a total utility 0 (leaf after a), or total utility 1 (after (b, d), since C is honest and chooses the action h fixed in our strategy). In both cases, the sum of their utilities is less than the sum of the utilities they would get if not colluding. In a similar manner, it can be argued for $\{A, C\}$ and $\{B, C\}$ that they cannot deviate profitably. Hence, Γ_{cr} is collusion resilient.*

2.3 Automated Game-Theoretic Security Analysis with CheckMate

CheckMate is a framework for full automation of game-theoretic security analysis, with particular focus on blockchain technologies [6]. For this purpose, the game, as well as

the security properties are encoded as SMT constraints. In the following, we briefly describe the CheckMate framework, in particular we elaborate on its input instances, the generated SMT constraints, its components and its output, as given in [6]. We assume familiarity with standard first-order logic [22] and real arithmetic in the context of SMT solving [1, 3].

Pipeline. The pipeline of CheckMate follows the structure summarized in [21]:

- First, an input instance is given. An input instance is a tuple

$$\Pi = (\Gamma, \mathscr{O}, C, C_{\text{wi}}, C_{\text{weri}}, C_{\text{cr}}, C_{\text{pr}}).$$

  Γ is an EFG corresponding to a protocol that needs to be analyzed, whereas $\mathscr{O}$ is a set of honest histories. Next, C is the set of initial constraints on the variables occurring in the player's utilities and C_{wi}, C_{weri}, C_{cr}, C_{pr} are sets of constraints on variables to hold when checking weak immunity, weaker immunity, collusion resilience and practicality, respectively.

- After the input is parsed and preprocessed, CheckMate proceeds to the analysis of the honest behavior. If multiple honest histories are given, CheckMate will analyze one honest history at a time and report the results individually.

- For each history and property, CheckMate constructs an SMT formula ϕ such that ϕ is satisfiable iff the EFG satisfies the security property. The construction of the formulas is described in more detail below.

  If ϕ is satisfiable, a model (representing *strategy*) can be extracted and we can output the result. Otherwise, it is checked whether case splitting is needed. This occurs when some utility terms cannot be compared. In such cases, the tool considers all consistent total orders of the terms that appear in the constructed formulas. If ϕ is unsatisfiable, but further case splits can be applied, case splitting is performed again. This iterative process terminates, as CheckMate is proven to be sound and complete [6]. Otherwise, if ϕ is unsatisfiable and no case splits apply, CheckMate can list *counterexamples* witnessing why ϕ is violated, list the cases in which ϕ is violated or compute the *weakest precondition* that if added as an additional constraint, satisfies ϕ.

  CheckMate uses Z3 [7] as its underlying SMT solver to determine the satisfiability of the constructed SMT formulas.

Construction of SMT formulas. As presented in [6], so-called *action variables* are used for the construction of SMT encodings. Action variables are essentially Boolean variables, that correspond to the branches of the game tree we analyze. For example, v_a^h is an action variable capturing whether after non-terminal history h, the player whose turn it is takes action a. Depending on whether the action variable v_a^h is true or false, we can determine if the current player chooses action a after history h or not.

Joint Strategies. In order to be able to extract joint strategies from the model, one needs to ensure that at every internal node of the game, exactly one action variable is set to true. This means, that at every point in the game where some player has a turn they choose exactly one action. This can be achieved by adding a constraint stating that at every internal node at least one and at most one of the available action variables is set to true.

Honest Histories. Furthermore, when analyzing a protocol modeled as an EFG, we are interested in finding joint strategies that extend honest histories. Hence, we need to ensure that the strategy extracted from the model yields the fixed honest history we currently check. This can be done by adding a constraint that ensures that the action variables corresponding to actions along the honest history are set to true.

Properties. After encoding joint strategies and honest histories it remains to encode the security properties in order to guarantee that they hold in a model.

The encoding of weak immunity is elaborated in the following. For this property we need to guarantee that each player gets a non-negative utility at the end of the game, independent on how the other players behave. To achieve this, we need to fix one player at a time and an add an implication stating that if the action variables corresponding to the actions along a terminal history when this player has a turn are all true, then the utility of this player received upon the end of the game has to be non-negative. The final SMT formula is constructed by linking the previously described formulas for the individual players conjunctively.

For the sake of readability and compactness of this chapter, interested users are instructed to look up the SMT encoding in [6].

Example 2.6 (SMT encoding of Market Entry game with Honest History (e, i))**.** *Consider once again the Market Entry game illustrated in Figure 2.1 and let the only honest history be* (e, i)*. Constructing the constraints for joint strategies and honest histories as described above yields:*

$$(v_n^{\emptyset} \vee v_e^{\emptyset}) \wedge (\neg v_n^{\emptyset} \vee \neg v_e^{\emptyset}) \wedge (v_i^{(e)} \vee v_{pw}^{(e)}) \wedge (\neg v_i^{(e)} \vee \neg v_{pw}^{(e)}) \wedge v_e^{\emptyset} \wedge v_i^{(e)}.$$

The first four conjuncts correspond to the encoding of joint strategies. They ensure that at the beginning of the game, either action n *or* e *is picked but not both and similarly, after action* e *either action* i *or* pw *is picked, but not both. The last two conjuncts are action variables that guarantee that action* e *is picked at the beginning and action* i *after that, corresponding to the given honest history.*

Encoding weak immunity results in the following formula:

$$(v_n^{\emptyset} \to 0 \geq 0) \wedge (v_e^{\emptyset} \to \frac{p}{2} \geq 0) \wedge (v_e^{\emptyset} \to -a \geq 0) \wedge$$
$$(p \geq 0) \wedge (v_i^{(e)} \to \frac{p}{2} \geq 0) \wedge (v_{pw}^{(e)} \to -a \geq 0)$$

The formula expresses that for each player, whenever the strategy includes actions leading to a leaf, the utility the player receives at the end of the game is guaranteed to be non-negative. The first line above corresponds to the player M, whereas the second one to the player E. In the first line we ensure that M gets a non-negative utility if M chooses action n in the strategy (first conjunct), but also if M takes action e, independently on how E behaves. In the same manner, this is performed for E in the second line. Note that the first conjunct in the second line does not contain an implication. This is due to the fact, that this conjunct represents the constraint for the utility of E given terminal history (o). Along this history, player E does not get to make a choice, so we obtain an empty conjunction for the antecedent of the implication, which is trivially true and can be omitted.

CHAPTER 3

Compositional Game-Theoretic Security Properties

The game-theoretic security properties introduced by Rain *et al.* in [20] are not compositional by nature. Hence, we need to propose and prove new, but equivalent versions of the security properties, which allow compositional game analysis.

A more detailed version of this chapter, together with Chapter 4–Chapter 5 is currently accepted at the peer-reviewed venue of OOPSLA 2025 – the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, with our preprint available at [5].

3.1 Security Properties for Subgames

For the purpose of the compositional game-theoretic approach, we define the security properties *weak immunity*, *weaker immunity*, *collusion resilience*, and *practicality* for any subtree of Γ, generalizing definitions given by [20] and [6]. The definitions coincide when the entire game Γ is taken as the subtree. As before, the property Byzantine Fault-Tolerance (P1) is ensured by weak immunity or respectively by weaker immunity; and the property Incentive Compatibility (P2) by the combination of collusion resilience and practicality. In this chapter we assume a total order on symbolic utility terms, same as in [6]. Recall that whenever some utility terms cannot be compared, the algorithm of CheckMate takes care of it by performing case splitting and analyzing all possible consistent total orders.

Definition 3.1 (Weak Immunity of a Subtree)**.** *A subtree $\Gamma_{|h}$ of game Γ with honest history h^* is* weak immune*, if a strategy $\sigma \in \mathscr{S}_{|h}$ exists such that all players p following*

σ always receive non-negative utility:

$$\exists \sigma \in \mathscr{S}_{|h}. \forall p \in N \; \forall \tau \in \mathscr{S}_{|h}. \; u_p(\sigma_p, \tau_{N-p}) \geq 0 \; . \qquad (\mathrm{wi}(\Gamma_{|h}))$$

If h is along h^*, additionally $H_{|h}(\sigma) = h^*_{|h}$ has to hold.

Sometimes, weak immunity is too restrictive and we take *weaker immunity* to ensure (P1).

Definition 3.2 (Weaker Immunity of a Subtree)**.** *A subtree $\Gamma_{|h}$ of game Γ with honest history h^* is* weaker immune*, if there exists a strategy $\sigma \in \mathscr{S}_{|h}$, such that all players p that follow σ always receive at least a negative infinitesimal:*

$$\exists \sigma \in \mathscr{S}_{|h}. \forall p \in N \; \forall \tau \in \mathscr{S}_{|h}. \; \mathsf{real}(u_p(\sigma_p, \tau_{N-p})) \geq 0 \; . \qquad (\mathrm{weri}(\Gamma_{|h}))$$

If h is along h^*, additionally $H_{|h}(\sigma) = h^*_{|h}$.

Next, the property collusion resilience ensures that the honest behavior yields the best payoff, even in the presence of collusion between players.

Definition 3.3 (Collusion Resilience of a Subtree)**.** *A subtree $\Gamma_{|h}$ of the game Γ with honest history h^* is* collusion resilient *if there exists a strategy $\sigma \in \mathscr{S}_{|h}$ such that no strict subgroup of players can deviate to receive a joint utility greater than their joint honest utility:*

$$\exists \sigma \in \mathscr{S}_{|h}. \forall S \subset N \; \forall \tau \in \mathscr{S}_{|h}. \; \sum_{p \in S} u_p(h^*) \geq \sum_{p \in S} u_p(\sigma_{N-S}, \tau_S) \; . \qquad (\mathrm{cr}(\Gamma_{|h}))$$

If h is along h^*, also $H_{|h}(\sigma) = h^*_{|h}$ has to hold.

Note that the collusion resilience of a subtree according to the above definition depends on the *honest utility*, the utility resulting from the honest history in the entire game Γ. The node containing the honest utility is not necessarily part of the considered subtree.

The property practicality ensures that, for all player decisions, the honest behavior is also "greedy": when all players act to maximize their own utility, the honest choice provides the highest payoff.

Definition 3.4 (Practicality of a Subtree)**.** *A subtree $\Gamma_{|h}$ of the game Γ with honest history h^* is* practical*, if there exists a strategy $\sigma \in \mathscr{S}_{|h}$ such that no player can deviate in any subtree to receive a strictly greater utility in the subtree:*

$$\exists \sigma \in \mathscr{S}_{|h} \; \forall g \in \mathscr{H}_{|h} \; \forall p \in N \; \forall \tau \in \mathscr{S}_{|(h,g)}. \qquad (\mathrm{pr}(\Gamma_{|h}))$$
$$u_{|g,p}(\sigma_{|g}) \geq u_{|g,p}(\tau_p, \sigma_{|g,N-p}) \; .$$

If h is along h^*, also $H_{|h}(\sigma) = h^*_{|h}$ has to hold.

We finally note that every subtree $\Gamma_{|h}$ of a game Γ that is off the honest history is always practical.

Example 3.1 (Practicality of Market Entry subgame off the honest history)**.** *Let the honest history be* (n)*. Consider the Market Entry subgame in Figure 2.1 after the non-terminal history* (e)*, that is off the honest history* (n)*. We can always choose the action that yields the best utility for the current player* E*. The only way we can violate practicality is by having the best choice conflicting with the honest choice, which cannot happen when the subtree is off the honest history.*

Recall that in Definition 2.7, we introduced the security properties of strategies and histories – specifying when a strategy satisfies a given security property and what it means for a terminal history to do so. The same definition applies in the current context as well.

3.2 Total Orders

Following [6], to lift the assumption about full knowledge of the relationships between utility terms, we conduct the security analysis relative to a finite set C of *initial constraints* on the symbolic variables appearing in the utility terms and explicitly universally quantify over the variables, as follows

$$\forall \vec{x}. \left(\bigwedge_{c \in C} c[\vec{x}] \right) \rightarrow \exists \sigma \in \mathscr{S}. H(\sigma) = h^* \wedge sp(\sigma)[\vec{x}] \,, \tag{3.1}$$

where $\vec{x} = (x_1, \ldots, x_\ell)$ are the real/infinitesimal variables occurring in the utility terms T_u and $sp(\sigma)$ is the formula of a security property $sp \in \{wi, weri, cr, pr\}$ after existential quantification of the strategy: e.g. for weak immunity $wi(\sigma) = \forall p \in N \; \forall \tau \in \mathscr{S}_{|h}. \; u_p(\sigma_p, \tau_{N-p}) \geq 0$, and similarly for the other properties.

Additionally, to enable efficient comparison of symbolic utilities within an SMT solver, we reformulate the above expression by accounting for all consistent *total orders* $\preceq$ over the set T_u of utility terms present in the game Γ.

Theorem 3.1 (Game-Theoretic Security with Total Orders)**.** *For an EFG* Γ *with honest history* h^* *and a finite set of initial constraints* C*, property* (3.1) *is equivalent to*

$$\forall (\preceq, T_u) \; \exists \sigma \in \mathscr{S}. H(\sigma) = h^* \wedge \;\; \forall \vec{x}. \left(\bigwedge_{c \in C \cup \preceq} c[\vec{x}] \right) \rightarrow sp(\sigma)[\vec{x}] \,. \tag{3.2}$$

3.3 Compositional Counterexamples

If there is no joint strategy satisfying a security property (wi, weri, cr, or pr), we can investigate why *not*. Counterexamples play a crucial role by revealing potential attack

vectors, thereby highlighting weaknesses in the protocol underlying the game model. By adjusting [6], the *counterexamples* to security for each individual property are as given in the sequel.

Counterexamples to Weak(er) Immunity. For the weak(er) immunity property, a counterexample is a harmed honest player p and a partial strategy of the other players $N - p$ such that no matter what honest actions p chooses, they cannot avoid receiving a real-valued negative utility.

Definition 3.5 (Counterexamples to Weak(er) Immunity)**.** *Let* Γ *be an EFG and* h^* *the considered honest history. A* counterexample to h^* being weak(er) immune *is a player* p *together with a partial strategy* s_{N-p} *such that* s_{N-p} *extended by any strategy* σ_p *of player* p *who follows the honest history* h^**, yields a terminal history* $H(s_{N-p}, \sigma_p) = t_{\sigma_p}$ *with* $u_p(t_{\sigma_p}) < 0$ *(resp. for weaker immunity* $\mathsf{real}(u_p(t_{\sigma_p})) < 0$*) and it is* minimal *with that property.*

Minimality of the partial strategy s_{N-p} states that, if any information point $s_{N-p}(h) = a$ is removed, there exists a strategy σ_p of player p such that (σ_p, s'_{N-p}) does not yield a terminal history, where s'_{N-p} is s_{N-p} without action a. That is, when following only actions of (σ_p, s'_{N-p}), we get stuck at an internal node of the tree.

Example 3.2. *A counterexample to the weak immunity of the Market Entry game depicted in Figure 2.1 with the honest history* (e, i) *would be player* M *and a partial strategy for* E*, where they choose action* pw*. If* M *behaves honestly and chooses action* e*, they end up with the negative utility of* $-a$ *after the terminal history* (e, pw)*.*

Counterexamples to Collusion Resilience. A counterexample to collusion resilience consists of a group of deviating players S and their partial strategy $s_S \in \mathscr{S}$, such that the joint utility of S is better than the honest utility, no matter how the other players $N - S$ react, while still following the honest history.

Definition 3.6 (Counterexamples to Collusion Resilience)**.** *Let* Γ *be an EFG and* h^* *the considered honest history. A* counterexample to h^* being collusion resilient *is a set of deviating players* S *together with their strategy* s_S *such that* s_S *extended by any strategy* σ_{N-S} *of players* $N - S$*, which follows the honest history* h^**, yields a terminal history* $H(\sigma_{N-S}, s_S) = t_{\sigma_{N-S}}$ *with*

$$\sum_{p \in S} u_p(t_{\sigma_{N-S}}) > \sum_{p \in S} u_p(h^*)$$

and it is minimal with that property. The minimality of s_S *is similar to the minimality of the partial strategy for weak(er) immunity.*

Example 3.3. *In the Market Entry game shown in Figure 2.1, a counterexample to the honest history* (e, pw) *being collusion resilient is a deviating group* $\{E\}$ *with a partial strategy that takes action* i*. Since the honest player* M *can only take action* e*, the deviating utility for* E *is* $\frac{p}{2}$*, which is greater than the honest one* $-a$*.*

Counterexamples to Practicality. Intuitively, a counterexample to practicality of the honest history h^* has to provide a reason why a rational player would not follow h^*. At some point along h^* after a prefix h, there is an action a promising the current player $P(h)$ a strictly better utility than h^*. Further, in the subgame $\Gamma_{|(h,a)}$ after (h, a) all practical utilities have to be better for $P(h)$, otherwise other players could choose actions in $\Gamma_{|(h,a)}$ that would disincentivize $P(h)$ to deviate from h^*.

Definition 3.7 (Counterexamples to Practicality)**.** *For an EFG Γ and honest history h^*, a* counterexample to practicality of h^* *is a prefix h of h^* together with an action $a \in A(h)$, such that for all practical terminal histories t in the subgame $\Gamma_{|(h,a)}$ it holds that $u_{P(h)}(h^*) < u_{P(h)}((h, a, t))$.*

Example 3.4. *The Market Entry game depicted in Figure 2.1 with the honest history (n) is not practical. A counterexample to practicality is the empty prefix $h = \emptyset$ and the action e, as the practical utility in the subgame after history (e) yields $\frac{p}{2}$ for player M, which is strictly better than the 0 in the honest case.*

CHAPTER 4

Compositional Game-Theoretic Security

Our compositional approach to game-theoretic security analysis is designed through two fundamental components:

1. Stratified analysis of security properties over players, capturing player-wise security properties (Section 4.2);
2. Splitting player-wise security properties into subgames, enabling us to propagate subgame reasoning to supergames in order to derive supergame security (Section 4.3).

For simplicity, in this chapter we assume a total order on the occurring utility terms in order to relate symbolic game utilities. This assumption is relaxed in the case splitting engine of the implemented algorithm (Algorithm 5.1), as discussed in Chapter 5.

As noted, this chapter is part of our extended submission [5].

4.1 Unsound Naïve Approach to Compositionality

For a divide-and-conquer style of compositional game-theoretic security analysis, we would like to analyze a game tree by propagating security results of subtrees upwards to the parent/ancestor nodes of the supertree. However, naïvely propagating the yes/no security result of the subtree does not suffice. This is shown in the following example.

Example 4.1. *Consider the Market Entry game (from Figure 2.1) reproduced on the left-hand side of Figure 4.1. Let the honest history to be* (n)*, marked with a solid blue line in the figure. Let us first analyze whether the property weak immunity is satisfied in*

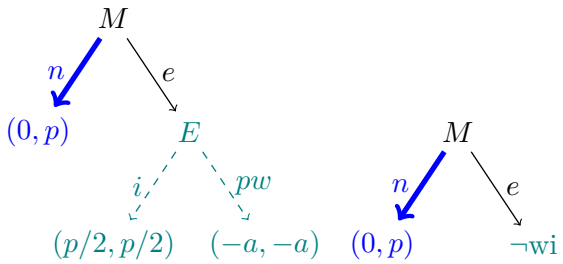

Figure 4.1: Naïve compositionality of Weak Immunity for Market Entry game, $a, p > 0$.

the usual, non-compositional way. The Market Entry game with honest history (n) *is weak immune: if* M *behaves honestly both players get a nonnegative utility; if* M *deviates via* e*, player* E *can choose action* i *and obtains a positive utility* $\frac{p}{2}$*.*

Now consider a naïve compositional approach looking at the subgame after non-terminal history (e)*, marked by teal dashed lines. Since player* E *can take action* pw *— leading to negative utility for* M *— this subtree is not weak immune. To mimic a naïve compositionality approach, we replace the subtree after* (e) *by* $\neg wi$*, shown on the right. Asked whether this* supertree *is weak immune, one would say no, as* M *could deviate from the honest history via* e*, which leads to a subtree that is not weak immune. However, this is an incorrect conclusion, because the game is weak immune, as justified in the previous paragraph. Consequently, this example shows that the naïve approach is unsound.*

The naïve approach fails because simply knowing that a subtree is not weakly immune is not enough–we must also know *who* it is not weak immune for. As seen in the example above, if weak immunity is violated only for player M, the parent can still achieve to satisfy the property if M behaves honestly. In fact, a supertree can be weak immune even if the subtree is not weak immune for any player. Similar details are needed for other properties: collusion resilience requires information about the colluding groups the subtree is collusion resilient against, and practicality requires the practical utilities resulting from the subtree, and whether the utility of the honest history is practical. We show that tracking this information enables a sound and complete compositional analysis – which can further also be implemented algorithmically.

We also observe that the naïve approach fails across all security properties and directions. For instance, in Example 4.1, the subtree is not secure while the full game is secure. In our experiments, we encountered all possible cases for each of the four properties: the subgame is secure while the supergame is not, the reverse, both secure, or both not secure.

4.2 Security Properties Stratified over Players

While Example 4.1 shows that there are no implications of subtree and supertree results in general, subtrees *along the honest history* can, in fact, soundly pass negative (not

secure) results up to their parents, as formalized in the following theorem.

Theorem 4.1 (Equivalence of Non-Secure Games)**.** *A game* Γ *with honest history* h^* *violates one of the security properties of weak(er) immunity, collusion resilience, or practicality iff there exists a history* h *along the honest history* h^* *such that* $\Gamma_{|h}$ *violates the respective security property.*

Intuitively, the honest history h^* "enforces" a path down the tree Γ: when a non-secure subtree $\Gamma_{|h}$ is encountered along this path, there is no way to compensate for it. Theorem 4.1, however, only propagates non-secure properties along the honest history. To allow for analysis results propagating from subgames to supergames, we *stratify game-theoretic security analysis over individual players.*

This allows us to analyze security properties for individual players (weak immunity: Definition 4.1, practicality: Definition 4.3) or specific a player group (collusion resilience: Definition 4.2) independently, without affecting the results for others (Theorem 4.2). Before stating this theorem, we first define what it means for a game to satisfy a security property for a single player or group.

Definition 4.1 (Weak Immunity for a Player)**.** *A subgame* $\Gamma_{|h}$ *with honest history* h^* *is* weak immune for player $p \in N$*, if there exists a strategy* $\sigma \in \mathscr{S}_{|h}$ *such that no matter to which strategy* $\tau \in \mathscr{S}_{|h}$ *other players deviate,* p*'s utility will be non-negative and, if* h *is along* h^**, then also* $H_{|h}(\sigma) = h^*_{|h}$ *:*

$$\exists \sigma \in \mathscr{S}_{|h}.\ (h \text{ along } h^* \to H_{|h}(\sigma) = h^*_{|h}) \wedge \qquad (\mathrm{wi}_p(\Gamma_{|h}))$$
$$\forall \tau \in \mathscr{S}_{|h}.\ u_{|h,p}(\sigma_p, \tau_{N-p}) \geq 0\ .$$

An analogous definition applies to *weaker immunity.*

The definition for collusion resilience *against* a given player group is similar to Definition 4.1, by lifting the quantifier over the player subgroups $S \subset N$ to the front of the formula.

Definition 4.2 (Collusion Resilience against a Player Group)**.** *A subgame* $\Gamma_{|h}$ *of game* Γ *with honest history* h^* *is* collusion resilient against a group of players $S \subset N$*, if there exists a strategy* $\sigma \in \mathscr{S}_{|h}$ *such that no matter to which strategy* $\tau \in \mathscr{S}_{|h}$ *the players in* S *deviate, their joint utility will be not greater than their honest joint utility and, if* h *is along* h^**, then also* $H_{|h}(\sigma) = h^*_{|h}$*:*

$$\exists \sigma \in \mathscr{S}_{|h}.\ (h \text{ along } h^* \to H_{|h}(\sigma) = h^*_{|h}) \wedge \qquad (\mathrm{cr}_S(\Gamma_{|h}))$$
$$\forall \tau \in \mathscr{S}_{|h}.\ \sum_{p \in S} u_{|h,p}(\sigma) \geq \sum_{p \in S} u_{|h,p}(\tau_S, \sigma_{N-S})\ .$$

Defining practicality for a single player requires slight changes: instead of considering an arbitrary player p, we define practicality for that player whose turn it is in the considered subtree.

Definition 4.3 (Practicality for the Current Player)**.** *A subgame $\Gamma_{|h}$ of a game Γ with honest history h^* is* practical for the current player*, if there exists a strategy $\sigma \in \mathscr{S}_{|h}$ such that in each subtree $\Gamma_{|(h,g)}$ no matter to which strategy $\tau \in \mathscr{S}_{|(h,g)}$ the current player $P(h,g)$ deviates, the utility of $P(h,g)$ in the subtree will not increase strictly and, if h is along h^*, then also $H_{|h}(\sigma) = h^*_{|h}$:*

$$\begin{aligned} &\exists \sigma \in \mathscr{S}_{|h}. && (\mathrm{pr}_P(\Gamma_{|h})) \\ &\quad (h \text{ along } h^* \rightarrow H_{|h}(\sigma) = h^*_{|h}) \ \wedge \forall g \in \mathscr{H}_{|h}\ \forall \tau \in \mathscr{S}_{|(h,g)}. \\ &\quad u_{|(h,g),P(h,g)}(\sigma_{|g}) \geq u_{|(h,g),P(h,g)}(\tau_{P(h,g)}, \sigma_{|g,N-P(h,g)}) \ . \end{aligned}$$

We now state our first crucial result towards compositionality: stratification of security analysis over players.

Theorem 4.2 (Player-Wise Security Properties)**.** *A game Γ satisfies a security property iff it satisfies the respective security property player-wise. That is, the following equivalences hold:*

1. *Γ weak immune $\Leftrightarrow \forall p \in N.$ Γ weak immune for p*
2. *Γ weaker immune $\Leftrightarrow \forall p \in N.$ Γ weaker immune for p*
3. *Γ collusion resilient $\Leftrightarrow \forall S \subset N.$ Γ collusion resilient against S*
4. *Γ practical $\Leftrightarrow$ Γ practical for the current player*

Example 4.2 (Player-Wise Weak Immunity)**.** *Let us revisit the Market Entry game Γ_{me} with honest history (n) from Figure 4.1, considering one player at a time.*

The first player is M. The subgame $\Gamma_{me|(e)}$ after history (e) is not weak immune for M, since E could take action pw. Propagating this result to the supertree Γ_{me}, we report weak immunity for M: as M will honestly take action n, we avoid $\Gamma_{me|(e)}$.

For E, $\Gamma_{me|(e)}$ is weak immune as action i can always be chosen, yielding positive utility. Propagating this result, we conclude that Γ_{me} is weak immune for E: all choices of M (whom we do not assume to be honest in the analysis of E), lead to either non-negative utility for E or to a subtree which is weak immune for E.

As weak immunity is given for each individual player, we can conclude that Γ_{me} with honest history (n) is weak immune.

Example 4.3 (Player-Wise Collusion Resilience)**.** *Recall the example for collusion resilience shown in Figure 2.2. In Example 2.5 we showed that Γ_{cr} with honest history (b,c,f) is collusion resilient. We now analyze the different strategies $\sigma^S \in \mathscr{S}$ that are collusion resilient against S, for all $S \subset N$ and extend the honest history. Note that, since these strategies need to extend the honest history, we have the following constraints: $\sigma^S(\emptyset) = b, \sigma^S((b)) = c, \sigma^S((b,c)) = f$. The only choice that is missing and needs to be determined is $\sigma^S((b,d))$.*

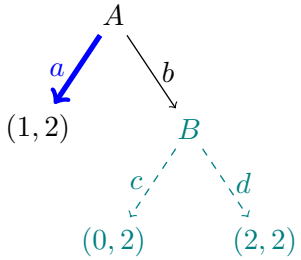

Figure 4.2: Example Γ_{pr} for Practicality

- $S = \{A\}$. *If A deviates, they will not obtain a utility strictly grater than the honest one. There are no restrictions on the choice of* $\sigma^S((b, d))$.
- $S = \{B\}$. *If B deviates, we need to make sure to set* $\sigma^S((b, d)) = h$, *so that B cannot deviate profitably.*
- $S = \{C\}$. *C cannot deviate alone, since C does not get to have a turn, if A and B are honest.*
- $S = \{A, B\}$. *For the case that both A and B collude, we need to set* $\sigma^S((b, d)) = h$, *so that they cannot deviate profitably via terminal history (b, d, g).*
- $S = \{A, C\}$. *If A and C collude either A takes action a, or B gets to play their turn and takes action c, after which a takes action e. In both cases the sum of the utilities of A and C is not greater that the sum of their honest utilities. There are no restrictions on the choice of* $\sigma^S((b, d))$.
- $S = \{B, C\}$. *If B and C deviate in any way, their joint utility will be 2 which is not strictly better than their honest joint utility (also 2).*

In conclusion, a collusion resilient strategy σ^S that is collusion resilient against all possible colluding groups has to have $\sigma^S((b, d)) = h$.

Example 4.4 (Player-Wise Practicality)**.** *Consider the example for practicality given in Figure 4.2 with honest history (a), marked by a thick blue line. We inspect whether there is a strategy $\sigma \in \mathscr{S}$ that yields the honest history and is practical for the current player. Let us start by considering the subgame after (b) marked with teal dashed lines. In this subtree, it is rational for player B to take either one of the actions c and d, as they both yield the same utility for them. Moving on to the supertree, we consider player A. By definition, history (a) is practical if there is a strategy that yields (a) such that A cannot obtain a strictly greater utility by deviating. We observe that this can be achieved by fixing action c for player B in our strategy, in case A deviates. Hence, Γ_{pr} is practical.*

4.3 Splitting and Combining Player-Wise Security Properties

Theorem 4.2 establishes that security analysis can be conducted independently for each player, without needing to consider interactions among all players. We will further demonstrate that the security of a supergame can be broken down into the security of its subgames, meaning that proving player-specific security in subgames is sufficient to establish the security of the entire supergame. This approach ensures that compositional game-theoretic security is both sound and complete.

Theorem 4.3 (Compositional Game-Theoretic Security)**.** *The game-theoretic security of an EFG Γ with honest history h^* can be computed compositionally. That is, the only information needed of a subtree $\Gamma_{|h}$, to decide whether Γ satisfies security property is*

- *for weak(er) immunity: for which players $p \in N$ the subtree $\Gamma_{|h}$ is weak(er) immune;*
- *for collusion resilience: against which player groups $S \subset N$ the subtree $\Gamma_{|h}$ is collusion resilient;*
- *for practicality:*
  - *if h is along h^*: whether $h^*_{|h}$ is practical in $\Gamma_{|h}$;*
  - *if h is not along h^*: the set $\mathbb{U}(h)$ containing all practical utilities of $\Gamma_{|h}$. A utility $u(t)$ after terminal history $t \in \mathscr{T}$ is practical in subgame $\Gamma_{|h}$ iff t is practical in $\Gamma_{|h}$.*

Theorem 4.4, Theorem 4.5 and Theorem 4.6, establish how to compositionally compute player-wise security for each security property, yielding a constructive proof of Theorem 4.3.

Theorem 4.4 (Compositional Weak Immunity)**.** *Let Γ be an EFG with honest history h^* and $p \in N$ a player. The following hold.*

1) A leaf of Γ is weak immune for p iff p's utility is non-negative:

$$\forall t \in \mathscr{T}.\, wi_p(\Gamma_{|t}) \quad \Leftrightarrow \quad u_p(t) \geq 0\,.$$

2) A branch of Γ is weak immune for p, where p is not the current player, iff all children are weak immune for p:

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\, p \neq P(h) \;\Rightarrow\; \big(wi_p(\Gamma_{|h}) \;\Leftrightarrow\; \forall a \in A(h).\, wi_p(\Gamma_{|(h,a)})\big)\,.$$

3) A branch of Γ along the honest history h^* is weak immune for the current player p, iff the child following h^* is weak immune for p. Let $a^* \in A(h)$ be the honest choice, i.e. (h, a^*) along h^*, then:

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\, p = P(h) \wedge h \text{ along } h^* \quad \Rightarrow \\ \big(wi_p(\Gamma_{|h}) \;\Leftrightarrow\; wi_p(\Gamma_{|(h,a^*)})\big)\,.$$

4) A branch of Γ off the honest history h^* is weak immune for the current player p, iff there exists a child that is weak immune for p:

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\ p = P(h) \wedge h \text{ off } h^* \quad \Rightarrow \\ \left(wi_p(\Gamma_{|h}) \ \Leftrightarrow\ \exists a \in A(h).\ wi_p(\Gamma_{|(h,a)})\right) .$$

Similar results to Theorem 4.4 hold for weaker immunity.

Example 4.5 (Compositional Weak Immunity)**.** *We revisit the Market Entry game Γ_{me} of Figure 4.1, with honest history (n). We compute that Γ_{me} is weak immune using our compositional approach, where we stratify over players first and then split Γ_{me} into subtrees.*

We start with player M. Theorem 4.4 implies that Γ_{me} is weak immune for M iff $\Gamma_{me|(n)}$ is weak immune for M; since $\Gamma_{me|(n)}$ is a leaf, we must check that the utility of M is non-negative, i.e. $0 \geq 0$. As this is true, game Γ_{me} is weak immune for M.

Next, we move to E. According to Theorem 4.4, the game Γ_{me} is weak immune iff $\Gamma_{me|(n)}$ and $\Gamma_{me|(e)}$ are weak immune for E. The subgame $\Gamma_{me|(n)}$ is weak immune for E if their utility is non-negative, i.e. $p \geq 0$, true by assumption. The subtree $\Gamma_{me|(e)}$ is now weak immune for E iff either $\Gamma_{me|(e,i)}$ or $\Gamma_{me|(e,pw)}$ is. E's utility at $\Gamma_{me|(e,i)}$ is $p/2 \geq 0$. Therefore Γ_{me} is weak immune for E, and from Theorem 4.2 it follows that Γ_{me} is weak immune.

Theorem 4.5 (Compositional Collusion Resilience)**.** *Let Γ be an EFG with honest history h^* and honest utility $u^* = u(h^*)$. The following equivalences hold.*

1) A leaf of Γ is collusion resilient against $S \subset N$ iff the honest joint utility of the deviating players $p \in S$ is greater than or equal to their joint utility at that leaf:

$$\forall t \in \mathscr{T}.\ cr_S(\Gamma_{|t}) \quad \Leftrightarrow \quad \sum_{p \in S} u^*_p \geq \sum_{p \in S} u_p(t) .$$

2) A branch of Γ, where the current player is in the deviating group $S \subset N$, is collusion resilient against S iff all children are collusion resilient against S:

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \in S \quad \Rightarrow \\ \left(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ \forall a \in A(h).\ cr_S(\Gamma_{|(h,a)})\right) .$$

3) A branch of Γ along the honest history h^*, where the current player is not in the deviating group $S \subset N$, is collusion resilient against S iff the child following h^* is collusion resilient against S. Let $a^* \in A(h)$ be the honest action, i.e. (h, a^*) along h^*, then:

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \notin S \wedge h \text{ along } h^* \quad \Rightarrow \\ \left(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ cr_S(\Gamma_{|(h,a^*)})\right) .$$

4) A branch of Γ off the honest history h^*, where the current player is not in the deviating group $S \subset N$, is collusion resilient against S iff there exists a child that is collusion resilient against S:

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \notin S \wedge h \text{ off } h^* \quad \Rightarrow \\ \left(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ \exists a \in A(h).\ cr_S(\Gamma_{|(h,a)})\right) .$$

Note that, if a player is in a deviating group, all child subtrees need to be collusion resilient even if we are along the honest history, as the deviator might choose any action and potentially harm honest players. In contrast, for an honest player and a node off honest history, there needs to merely exist one collusion resilient child that the player can choose to defend against the deviating group.

Example 4.6 (Compositional Collusion Resilience)**.** *We compositionally compute the collusion resilience of the Market Entry game Γ_{me} (Figure 4.1) with honest history (n). We have two possible colluding groups, both singletons $\{M\}$ and $\{E\}$.*

Consider $\{M\}$. At the root of Γ_{me}, since the player M is in the colluding group, all subtrees must be collusion resilient against $\{M\}$. Along the honest history we reach a leaf $\Gamma_{me|(n)}$, which is collusion resilient (it is the honest leaf). For subtree $\Gamma_{me|(e)}$ there needs to exist a collusion resilient child, which is the case in the leaf after (e, pw): utility $-a$ is strictly smaller than the honest utility 0.

Next, consider $\{E\}$. At the root, M is not in the deviating group. Hence, only the honest child $\Gamma_{me|(n)}$ need be collusion resilient against $\{E\}$, which it is, as it is the honest leaf; so the utility is equal to the honest one in part (1) of Theorem 4.5. This suffices to establish collusion resilience against $\{E\}$; checking $\Gamma_{me|(e)}$ is unnecessary.

Using Theorem 4.2, it follows that Γ_{me} is collusion resilient.

Theorem 4.6 (Compositional Practicality)**.** *Let Γ be an EFG with honest history h^* and $\mathbb{U}(h)$ be the set of practical utilities of subtree $\Gamma_{|h}$. Let u^* be the honest utility $u^* = u(h^*)$. Then the following identities and equivalences hold.*

1) In a leaf of Γ the only practical utility is that of the leaf.

$$\forall t \in \mathscr{T}.\ \mathbb{U}(t) = \{u(t)\} .$$

2) The honest utility u^* is practical in a branch of Γ along h^* iff it is practical in the child following h^* and if for every other child at least one practical utility is not greater than u^* for the current player. Let $a^* \in A(h)$ be the honest action after h, then:

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\ h \text{ along } h^* \quad \Rightarrow \quad \big(pr(\Gamma_{|h}) \quad \Leftrightarrow \\ pr(\Gamma_{|(h,a^*)}) \ \wedge\ \forall a \in A(h) \setminus \{a^*\}\ \exists u \in \mathbb{U}((h,a)).\ u^*_{P(h)} \geq u_{P(h)}\big) .$$

3) A utility is practical in a branch of Γ off the honest history h^* iff it is practical in a child and if, for every other child, at least one practical utility is not greater for the

current player.

$$\begin{aligned}\forall h \in \mathscr{H} \setminus \mathscr{T}.\ h \text{ off } h^* \quad \Rightarrow \quad & \big(\forall t \in \mathscr{T}_{|h}.\ u(t) \in \mathbb{U}(h) \quad \Leftrightarrow \\ & \exists a \in A(h).\ u(t) \in \mathbb{U}((h,a)) \wedge \\ & \forall a' \in A(h) \setminus \{a\}\ \exists u' \in \mathbb{U}((h,a')).\ u_{P(h)}(t) \geq u'_{P(h)}\big)\ .\end{aligned}$$

Example 4.7 (Compositional Practicality)**.** *To compositionally compute the practicality of the Market Entry game Γ_{me} of Figure 4.1 with honest history (n), we start with the leaves of the tree, where the practical utilities are the utilities of the leaves. Moving upwards in the tree, we look at the subtree $\Gamma_{me|(e)}$, which is off the honest history, so we take the better utility for player E, setting $\mathbb{U}(e) = \{(\frac{p}{2}, \frac{p}{2})\}$. At the root of the tree, which is along the honest history, the practical utility of the honest subtree $(0, p)$ should be practical in the entire tree. Since all practical utilities of the non-honest child (there is just one) are better for player M (as $\frac{p}{2} > 0$), the honest utility is not practical. Theorem 4.2 then implies that Γ_{me} is not practical.*

CHAPTER 5

Automation and Evaluation of Game-Theoretic Security Analysis

Chapter 4 assumed a total order $\preceq$ on game utility terms T_u. Following the explanation in Section 3.2, this section lifts this assumption and interprets the game variables in the utility terms T_u as real-valued variables $\vec{x}$. Combining Theorem 3.1 and Theorem 4.2 we observe that quantification of the variables $\vec{x}$ can be done equivalently by grouping values of $\vec{x}$ that satisfy the same total order $(\preceq, T_u)$, and moreover, security property quantifications (over players, subgames and strategies) can be pulled out of the $\vec{x}$ quantification.

More concretely, for weak immunity, the formula (3.1) (with $sp = wi$) becomes equivalent to:

$$\forall(\preceq, T_u)\ \forall p \in N\ \exists \sigma \in \mathscr{S}.\, H(\sigma) = h^* \wedge \forall \tau \in \mathscr{S}\ .$$
$$\forall \vec{x}.\, \left(\bigwedge_{c \in C \cup \preceq} c[\vec{x}] \right) \rightarrow u_p(\sigma_p, \tau_{N-p})[\vec{x}] \geq 0\,. \tag{5.1}$$

The translation of game-theoretic security from Theorem 3.1 to the player-wise security of Theorem 4.2 is key to automating compositional security, as it allows us to pass on only compact first-order expressions of the form

$$\forall \vec{x}.\, \left(\bigwedge_{c \in C \cup \preceq} c[\vec{x}] \right) \rightarrow ut_1[\vec{x}] \geq ut_2[\vec{x}]\ , \tag{5.2}$$

to an SMT solver, where ut_1 and ut_2 are term expressions over $\vec{x}$; which can be efficiently handled by SMT solvers. Unlike [6], which passes a single large SMT formula to the solver, the compositional framework passes many but much smaller formulas of the form 5.2.

To determine whether (5.2) is a theorem, the formula is first negated and then checked for satisfiability using an SMT solver. Its simplified quantified structure is particularly *automation-friendly*, as it falls within a purely existential fragment, for which efficient decision procedures exists [1, 3]. Reasoning about players and the existence of strategies σ proving player-wise security is handled separately using the compositional results of Theorem 4.4, Theorem 4.5 and Theorem 4.6, keeping this complexity out of the SMT solving.

5.1 Divide-and-Conquer Algorithms for Compositional Security

Our compositionality results from Theorem 4.2, Theorem 4.4, Theorem 4.5, and Theorem 4.6 extended by a lazy total-order approach, induce a *divide-and-conquer method for splitting and combining reasoning over game subtrees and supertrees*. Our overall divide-and-conquer framework for automating compositional game-theoretic reasoning is summarized in Algorithm 5.1. We compositionally compute the game-theoretic security of a protocol, analyzing the (protocol) game for all real-valued variables of utitilty terms, considering all total orders at once. If we fail, we split the total orders into multiple cases, unless we can conclude that the respective security property cannot be satisfied even if we restrict the values to one total order.

Algorithm 5.1: Function `SatisfiesProperty`. In Algorithm 5.1 an instance Π, which contains the game tree Γ and the set of initial constraints C (as introduced in Section 2.3), is given as input. The input to Algorithm 5.1 also contains the honest history h^*, the security property to be analyzed, and the currently considered case `case`.

The function `SatisfiesProperty` in Algorithm 5.1 is called initially with the empty case to analyze all total orders. This case can be refined throughout Algorithm 5.1, using case splits. Hence, in the first call of the function, the set `S`, representing the constraints handed to an SMT solver, contains only the initial constraints C. The relevant player groups `RelevantGroups` of security property *sp* are set according to the stratified definitions of *sp* from Section 4.2: N for $w(er)i$, as we stratify over players; $2^N \setminus \{\emptyset, N\}$ for *cr*, as we stratify over deviating subgroups; and {"none"} for *pr*.

The function `ComputeSP` in line 6 of Algorithm 5.1 stands for `ComputeWI`(Algorithm 5.2), `ComputeCR` or `ComputePR`, depending on the security property *sp*. The result of `ComputeSP` depends on whether Γ with honest history h^* satisfies property *sp* for/against pg, given the constraints in S.

In `ComputeSP`, we also keep track of utility comparisons we cannot decide. Importantly, if constraints need to be compared, but cannot, they are returned and checked as $\mathsf{split}_{\mathsf{pg}}$ later.

The loop in lines 5–12 of Algorithm 5.1 incorporates player-wise security from Theorem 4.2. It additionally provides a necessary case split if the security property is violated for a

Algorithm 5.1: Function `SatisfiesProperty` for Compositional Game-Theoretic Security Reasoning.

input : input instance $\Pi = (\Gamma, inf, C)$, honest history h^*, the name of a security property $sp \in \{wi, weri, cr, pr\}$, and the currently analyzed case (as set of SMT constraints) `case`.

output : `true` if Π satisfies sp in case `case`, `false` otherwise

```
1  S ← ∅
2  AddConstraints (S, C ∪ case)
3  result ← true
4  split ← null
5  for pg ∈ RelevantGroups(Π, sp) do
6  |  (result_pg, split_pg) ← ComputeSP (Π,h*,S,sp,pg)
7  |  if result_pg = false then
8  |  |  result ← result_pg
9  |  |  split ← split_pg
10 |  |  break
11 |  end
12 end
13 if result = true then
14 |  return true
15 end
16 if split = null then
17 |  return false
18 end
19 for constr ∈ {split, ¬split} do
20 |  if ¬SatisfiesProperty(Π, h*, sp, case ∪ {constr}) then
21 |  |  return false
22 |  end
23 end
24 return true
```

player group, if one exists. Subsequently, the respective results are returned: `true`for all groups yields `true`; `false`but nothing to split on for at least one group yields `false`; and `false`together with a split leads to further case splits (lines 19–24). If we split the total orders into multiple cases, all the cases have to return `true`for the property to be satisfied.

The security-property-specific function variants of `ComputeSP` recursively apply the compositional results of Theorem 4.3. To illustrate case splitting of total orders, we only describe function `ComputeWI` of Algortihm 5.2 below.

Algorithm 5.2: Function `ComputeWI`. The function `ComputeWI` of Algorithm 5.2 is initially called with the entire game tree Γ from function `SatisfiesProperty` of Algorithm 5.1. We then proceed recursively, according to Theorem 4.4. Note that the player group pg is just one player.

In a leaf, `GetUtility` in Algorithm 5.2 returns the utility of player pg. In line 2, we check if the constraints in S combined with the condition that the utility is negative are `unsat`. To ensure non-negative utility, we negate the constraint and check for unsatisfiability. If the implication holds, we return `true`, indicating the utility is non-negative. If not, we check if the utility can be non-negative in line 5. If it cannot (line 6), the leaf is not weak immune, so we return `false` and null, as no further case split is needed. Otherwise (line 8), weak immunity depends on the total order, requiring a case split.

In lines 10–32 of Algorithm 5.2, we check in which of the cases of Theorem 4.4 we are. We then call the function `ComputeWI` recursively on immediate subgames $\Gamma_{|(a)}$ and propagate the result accordingly. Note that, for simplicity, in line 13 of Algorithm 5.2 we do not wait for a null split that would immediately return `false`, but rather proceed with a split. However, as there are only finitely many possible case splits, we will eventually see the null split for a `false` subtree if it exists and return it to reach the correct result.

Example 5.1. *We simulate the execution of `ComputeWI` (Algorithm 5.2) on the Market Entry game (Figure 4.1), assuming only $a > 0$ and letting $p \in \mathbb{R}$. We check weak immunity for player M with honest history (e, i) on the full game tree Γ_{me}. Since the root is along the honest history, the function jumps to line 19 and recursively calls `ComputeWI` for the honest subtree $\Gamma_{me|(e)}$. There, E is the current player (not M), so we follow line 10 and evaluate both actions pw and i. For pw we recursively compute weak immunity for the leaf after (e, pw) in line 12. Algorithm 5.2 will execute lines 1 and 2, and since the utility of player M is 0, which is a non-negative number, the check in line 2 will be unsat, so the function returns (`true`, null). For the other action i, we recursively compute (line 12 of the algorithm) the weak immunity for the leaf after (e, i). The function `GetUtility`$(\Gamma_{me|(e,i)}, M)$ will return $\frac{p}{2}$, for which we cannot decide whether it is non-negative, as there are no initial constraints on p. Both checks in lines 2 and 5 fail, so we return (`false`, $\frac{p}{2} \geq 0$) in line 8. Back in $\Gamma_{me|(e)}$, since the result is `false`, we return (`false`, $\frac{p}{2} \geq 0$) in line 14.*

Algorithm 5.2: Function `ComputeWI` for Weak Immunity.

input : game tree Γ, honest history h^*, set S containing initial constraints and current case, player group pg.

output : $(\mathsf{result}, \mathsf{split})$, where result states whether Γ is weak immune for pg, given S, and split a crucial utility comparison we cannot decide.

```
1  if isLeaf(Γ) then
2  |  if Check(S, GetUtility(Γ, pg) < 0) = unsat then
3  |  |  return (true, null)
4  |  end
5  |  if Check(S, GetUtility(Γ, pg) ≥ 0) = unsat then
6  |  |  return (false, null)
7  |  end
8  |  return (false, GetUtility(Γ, pg) ≥ 0)
9  end
10 if CurrentPlayer(Γ) ≠ pg then
11 |  for a ∈ Actions(Γ) do
12 |  |  (result, split) ← ComputeWI(Γ_|(a), h*, S, pg)
13 |  |  if result = false then
14 |  |  |  return (result, split)
15 |  |  end
16 |  end
17 |  return (true, null)
18 end
19 if AlongHonest(Γ, h*) then
20 |  a* ← HonestAction(Γ, h*)
21 |  return ComputeWI(Γ_|(a*), h*, S, pg)
22 end
23 newsplit ← null
24 for a ∈ Actions(Γ) do
25 |  (result, split) ← ComputeWI(Γ_|(a), h*, S, pg)
26 |  if result = true then
27 |  |  return (true, null)
28 |  else if split ≠ null then
29 |  |  newsplit ← split
30 |  end
31 end
32 return (false, newsplit)
```

Theorem 5.1 (Correctness of 5.1)**.** *The compositional approach to compute the game-theoretic security of an input instance* Π *for honest history* h^* *described in Algorithm 5.1*

is sound and complete. That is, $\texttt{SatisfiesProperty}(\Pi, h^*, sp, \emptyset) = \texttt{true}$ *iff* Π *with honest history* h^* *satisfies the property* sp*. Otherwise, it returns* `false`*.*

In addition to compositional security via Algorithm 5.1, our work supports additional features to debug a protocol and better understand its structure. Those include (i) strategy extraction in case the considered security property was satisfied (Section 5.2), (ii) finding counterexamples (Section 5.3), and (iii) providing weakest preconditions to make the game secure otherwise. Computing preconditions in our compositional setting can be done via collecting all cases in which the security property is violated, and then conjoining and negating them afterwards.

5.2 Extracting Compositional Strategies

The compositional security analysis in Algorithm 5.1 is designed in a way that it efficiently carries the necessary information to compute witnesses in a simple and intuitive manner.

Theorem 5.2 (Weak(er) Immune Strategies)**.** *For a weak(er) immune game* Γ*, with honest history* h^**, strategy* σ *is honest and weak(er) immune where*

$$\sigma := (\sigma^{p_1}, \ldots, \sigma^{p_{|N|}}) ,$$

and $\sigma^{p_i} \in \mathscr{S}_{p_i}$ *is a strategy for player* p_i*. Strategy* σ^{p_i} *picks the honest choice along the honest history, whereas at other nodes, where it is* p_i*'s turn, it picks an arbitrary action* a *that yields a weak(er) immune for* p_i *subtree after action* a*.*

Theorem 5.2 outlines a constructive algorithm for extracting a weak(er) immune strategy. For each player pg, the function `ComputeWI` (and `ComputeWERI`) operates as follows: when it is pg's turn after history h, h off h^*, and a weak(er) immune choice is identified, the action is recorded as part of a potential weak(er) immune and honest strategy σ. If the game is weak(er) immune for all players, we can compute σ by collecting all the recorded choices throughout the tree.

Example 5.2. *We compute the weak immune strategy of the Market Entry game from Figure 2.1 with honest history* (n)*, which was analyzed in Example 4.5. The strategy* σ^M *for player* M *has to choose the honest action* n *at the root, which is the only choice point for* M*. The strategy* σ^E *for player* E *needs to choose one weaker immune subtree after history* (e)*. Since the subtree after history* (e, i) *is the only candidate, we set* $\sigma^E(e) = i$*. The strategy* $\sigma = (\sigma^M, \sigma^E)$ *is the desired weak immune strategy.*

Collusion resilience and practicality also admit elegant methods for deriving compositional strategies in a similar manner. Detailed description can be found in [5].

5.3 Finding Compositional Counterexamples

Counterexamples to the security properties, as defined in Definition 3.5, Definition 3.6 and Definition 3.7, serve the important purpose of providing attack vectors and thus pinpointing the weaknesses of a protocol underlying the considered game model. We use the following *pseudo-algorithms* to compute counterexamples compositionally.

Compositional Counterexamples to Weak(er) Immunity. We first store information during Algorithm 5.2: When analyzing the weak(er) immunity for a player p, whenever it is not p's turn and there exists an action leading to a not weak(er) immune subtree (line 14 with split = null in Algorithm 5.2), we store the action, the current history and the player p.

Secondly, after the analysis terminated and the result was not weak(er) immune, we generate a counterexample to the weak(er) immunity of player p by walking through the tree again. Assume the current history is h and we proceed from the root as follows.

- If p is the current player and h is along the honest history, we follow the honest action to a subtree. This is sufficient since an honest p follows the honest history.
- If it is p's turn but h is not along the honest history, all choices had to lead to not weak(er) immune for p subtrees for the current tree to be not weak(er) immune for p. We, therefore, have to follow all choices to compute a counterexample.
- Otherwise, if it is not p's turn, we check our stored data for a choice a that is not weak(er) immune for p. By construction and using Theorem 4.4, it has to exist. We add it to our partial strategy s_{N-p}, i.e. $s_{N-p}(h) = a$. Then, we continue at history (h, a).
- At a leaf nothing has to be considered. A leaf that is not weak(er) immune for p contains a negative (real) utility for p.

According to Theorem 4.4, the steps outlined above provide a player p and a partial strategy s_{N-p} for the other players $N - p$, no matter how the honest p behaves off the honest history. It also yields only negative utilities for p and it thus provides a counterexample to the weak(er) immunity of p and, therefore, a counterexample to the weak immunity of the game with the considered honest history.

It is also possible to compute *all* counterexamples to weak(er) immunity. This can be done by simply storing *all* actions that lead to not weak(er) immune subtrees.

Example 5.3. *Let us adapt the Market Entry game from Figure 2.1 by changing the initial constraint on the variable p to $p < 0$. The honest history (n) is not weak immune for player E, as they get a negative utility $p < 0$ in the honest leaf. We can thus construct the counterexample as follows: starting from the root, it is not E's turn and the not weak*

immune choice is (n)*, so we add the action* n *to the partial strategy for player* M*. We then continue at history* (n)*, which is a leaf, so we are done.*

Counterexamples to collusion resilience and practicality can also be computed in a similar manner. Detailed description can be found in [5].

5.4 Experimental Evaluation

We implemented the compositional security approach of Chapter 4 by exploiting its divide-and-conquer reasoning nature from Section 5.1, Section 5.2 and Section 5.3. Our implementation is available online in the CheckMate 2.0 tool[1].

Experimental Setup. We evaluated our tool using a machine with 2 AMD EPYC 7502 CPUs clocked at 2.5GHz with 32 cores and 1TB RAM using 16 game-theoretic security benchmarks. Our dataset contains the 15 examples analyzed in [21], which include realistic models of real-world blockchain protocols along with game scenarios of various sizes. Additionally, we detail later in this section one large example, named 4-Player Routing, in order to showcase the impact of interleaved sub- and supertree reasoning. Our experiments also compare CheckMate 2.0 to CheckMate.

Experimental Results. Table 5.1, Table 5.2 and Table 5.3 summarize our experiments. We report both on the results of CheckMate 2.0 and CheckMate; the respective columns on times, nodes, and calls detail these comparisons. In particular, the columns "Nodes evaluated" and "Nodes evaluated (reps)" indicate the number of game tree nodes visited during the security analysis without and, respectively, with repetitions. The "Calls" column of Table 5.1 shows the number of calls made to the SMT solver while proving the security property listed in column 4.

Experimental Analysis. Table 5.1 demonstrates that the compositional approach of CheckMate 2.0 significantly outperforms the non-compositional CheckMate setting in execution time across nearly all benchmarks. The scalability of CheckMate 2.0 is especially evident in the Tic Tac Toe benchmark, which involves a substantial 548,946 nodes. In this example, for the properties weak immunity (wi), weaker immunity (weri), and collusion resilience (cr), CheckMate 2.0 completes the security analysis in approximately 5 seconds, whereas CheckMate requires between 255 and 287 seconds. When proving practicality (pr) of Tic Tac Toe, the conventional CheckMate fails to terminate within 8 hours while CheckMate 2.0 succeeds in less than 37 seconds.

In some benchmarks, where a security property is not satisfied, CheckMate 2.0 explores significantly fewer nodes, see 3-Player Routing for weak immunity and collusion resilience, the Pirate game for weak immunity, and Auction for weak immunity and collusion resilience.

[1] https://github.com/apre-group/checkmate/tree/CCS25

Game	Nodes	Players	Security property	Secure yes/no	Time	Nodes evaluated CheckMate 2.0/CheckMate	Nodes evaluated (reps)	Calls
$\text{Splits}_{\text{wi}}$	5	2	wi	y	0.010 / 0.018	5	18 / 10	10 / 3
(q)			weri	y	0.010 / 0.018	5	18 / 10	10 / 3
			cr	y	0.010 / 0.015	4 / 5	6 / 10	3 / 1
			pr	y	0.011 / 0.017	5	25 / 5	19 / 3
$\text{Splits}_{\text{cr}}$	5	2	wi	y	0.011 / 0.019	4 / 5	6 / 10	3 / 1
(n)			weri	y	0.011 / 0.019	4 / 5	6 / 10	3 / 1
			cr	y	0.011 / 0.018	5	16 / 10	10 / 3
			pr	y	0.011 / 0.018	5	15 / 5	20 / 3
Market Entry	5	2	wi	n	0.011 / 0.014	5	8 / 10	5 / 1
(e, i)			weri	n	0.010 / 0.014	5	8 / 10	5 / 1
			cr	y	0.010 / 0.014	5	8 / 10	4 / 1
			pr	y	0.010 / 0.015	5	5	2 / 1
G	5	2	wi	n	0.010 / 0.012	5 / 5	28 / 10	18 / 4
(r_A, l_B)			weri	n	0.009 / 0.012	5 / 5	28 / 10	18 / 4
			cr	n	0.008 / 0.010	2 / 5	2 / 10	2 / 1
			pr	n	0.009 / 0.010	5 / 5	9 / 5	5 / 1
Simplified Closing	8	2	wi	y	0.009 / 0.012	8	10 / 16	8 / 1
(H)			weri	y	0.009 / 0.011	8	10 / 16	8 / 1
			cr	y	0.008 / 0.011	7 / 8	9 / 16	6 / 1
			pr	n	0.009 / 0.012	8	8	8 / 1
(C_h, S)			wi	n	0.008 / 0.012	3 / 8	3 / 16	2 / 1
			weri	n	0.008 / 0.011	3 / 8	3 / 16	2 / 1
			cr	y	0.008 / 0.012	8	11 / 16	7 / 1
			pr	y	0.009 / 0.013	8	8	6 / 1
Simplified Routing	17	5	wi	n	0.008 / 0.012	7 / 17	7 / 85	2 / 1
$(S_H, L, L, L, L, U, U, U, U)$			weri	y	0.009 / 0.011	17	77 / 85	28 / 1
			cr	n	0.010 / 0.017	16 / 17	105 / 510	24 / 1
			pr	y	0.009 / 0.012	17	17	8 / 1
Centipede	19	3	wi	n	0.044 / 0.051	19	602 / 57	345 / 18
$(C, C, C, C, C, C, C, C, C)$			weri	n	0.033 / 0.052	19	602 / 57	345 / 18
			cr	n	0.044 / 0.038	19	534 / 114	305 / 9
			pr	n	0.011 / 0.028	19	103 / 19	39 / 7
EBOS	31	4	wi	n	0.009 / 0.013	28 / 31	38 / 124	21 / 1
$(Mine, Mine, Mine, Mine)$			weri	n	0.008 / 0.013	28 / 31	38 / 124	21 / 1
			cr	n	0.039 / 0.021	31	476 / 434	304 / 4
			pr	n	0.019 / 0.024	31	167 / 31	184 / 5
Pirate	79	4	wi	n	0.010 / 0.015	10 / 79	10 / 316	5 / 1
(y, n, n, n, y, y)			weri	n	0.009 / 0.016	10 / 79	10 / 316	5 / 1
			cr	n	0.041 / 0.029	79	622 / 1106	368 / 4
			pr	n	0.036 / 0.049	79	482 / 79	554 / 8
Auction	92	4	wi	n	0.012 / 0.033	16 / 92	16 / 368	9 / 1
(E, E, I, I)			weri	y	0.016 / 0.027	90 / 92	229 / 368	162 / 1
			cr	n	0.018 / 0.030	66 / 92	128 / 1,288	103 / 1
			pr	y	0.021 / 0.145	92	92	188 / 1
Closing	221	2	wi	y	0.011 / 0.024	20 / 221	22 / 442	16 / 1
(H)			weri	y	0.010 / 0.021	20 / 221	22 / 442	16 / 1
			cr	y	0.012 / 0.023	44 / 221	46 / 442	36 / 1
			pr	n	0.097 / 0.346	221	568 / 221	1454 / 1
(C_h, S)			wi	y	0.011 / 0.024	33 / 221	36 / 442	25 / 1
			weri	y	0.011 / 0.020	33 / 221	36 / 442	25 / 1
			cr	y	0.013 / 0.023	60 / 221	63 / 442	48 / 1
			pr	y	2.144 / 0.345	221	14353 / 221	38220 / 1
3-Player Routing	21,688	3	wi	n	0.248 / 0.984	16 / 21,688	16 / 65,064	9 / 1
(S_H, L, L, U, U)			weri	y	0.514 / 1.008	7,084 / 21,688	7,570 / 65,064	5,441 / 1
			cr	n	0.272 / 1.886	430 / 21,688	474 / 130,128	299 / 1
			pr	n	33.162 / 34.717	21,688	416,156 / 21,688	569,418 / 13
Unlocking Routing	36,113	5	wi	n	0.621 / 2.121	1,184 / 36,113	1,184 / 180,565	714 / 1
(U, U, U, U)			weri	y	1.525 / 1.625	32,429 / 36,113	55,090 / 180,565	27,897 / 1
			cr	n	0.584 / 15.247	319 / 36,113	373 / 1,083,390	60 / 1
			pr	y	2.848 / 4.382	36,113 / 36,113	36,113 / 36,113	46,636 / 1
Tic Tac Toe Concise	58,748	2	wi	y	0.557 / 6.372	1,345 / 58,748	1,355 / 117,496	698 / 1
$(CM, LU, RU, LD, LM$			weri	y	0.541 / 6.373	1,345 / 58,748	1,355 / 117,496	698 / 1
$RM, CU, CD, RD)$			cr	y	0.543 / 7.352	1,345 / 58,748	1,355 / 117,496	698 / 1
			pr	y	3.937 / 227.807	58,748 / 58,748	58,748 / 58,748	57,250 / 1
Tic Tac Toe	549,946	2	wi	y	5.276 / 255.368	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
$(CM, RU, LU, RD, RM,$			weri	y	5.256 / 255.600	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
$LM, CU, CD, LD)$			cr	y	5.302 / 286.574	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
			pr	y	36.530 / TO	549,946 / TO	549,946 / TO	527,198 / TO

Table 5.1: Experimental results of game-theoretic security, using the compositional CheckMate 2.0 approach and the non-compositional CheckMate setting of [21]. Runtimes are given in seconds, with a timeout (TO) after 8 hours. For each game, columns 2–3 list the size (tree nodes and game players) of the game from column 1. Column 4 shows the game-theoretic security property we analyzed and (dis)proved, as indicated in column 5. Columns 6–9 present the results of CheckMate 2.0 compared to CheckMate, using the slash / sign.

Game	Property	Time (one CE) CheckMate 2.0/CheckMate	Time (all CE) CheckMate 2.0/CheckMate
Market Entry	wi	0.010 / 0.016	0.010 / 0.023
(e, i)	weri	0.010 / 0.017	0.009 / 0.019
G	wi	0.010 / 0.013	0.010 / 0.020
(r_A, l_B)	weri	0.010 / 0.014	0.009 / 0.020
	cr	0.008 / 0.011	0.008 / 0.013
	pr	0.009 / 0.016	0.009 / 0.028
Simplified Closing (H)	pr	0.009 / 0.019	0.008 / 0.019
(C_h, S)	wi	0.009 / 0.014	0.008 / 0.016
	weri	0.009 / 0.014	0.008 / 0.014
Simplified Routing	wi	0.009 / 0.014	0.010 / 0.033
$(S_H, L, L, L, L, U, U, U, U)$	cr	0.010 / 0.023	0.016 / 0.096
Centipede	wi	0.046 / 0.049	0.080 / 0.495
$(C, C, C, C, C, C, C, C, C)$	weri	0.034 / 0.050	0.061 / 0.495
	cr	0.045 / 0.047	0.078 / 0.538
	pr	0.012 / 0.062	0.022 / 0.400
EBOS	wi	0.010 / 0.015	0.011 / 0.058
$(Mine, Mine, Mine, Mine)$	weri	0.010 / 0.015	0.010 / 0.057
	cr	0.040 / 0.028	0.057 / 10.760
	pr	0.020 / 0.032	0.021 / 0.032
Pirate	wi	0.010 / 0.020	0.157 / 9.465
(y, n, n, n, y, y)	weri	0.009 / 0.020	0.157 / 9.495
	cr	0.041 / 0.039	3.232 / 79.839
	pr	0.037 / 0.064	7.414 / 35.227
Auction	wi	0.012 / 0.048	0.025 / 4.172
(E, E, I, I)	cr	0.018 / 0.066	0.036 / 15.106
Closing (H)	pr	0.096 / 0.650	2.204 / 8.846
3-Player Routing	wi	0.251 / 1.925	5.909 / 110.716
(S_H, L, L, U, U)	cr	0.279 / 5.619	1.657 / 7.815
	pr	33.561 / 46.480	291.236 / 3 033.784
Unlocking Routing	wi	0.602 / 5.219	2.090 / 1 988.997
(U, U, U, U)	cr	0.564 / 116.906	3.562 / error

Table 5.2: Experiments on counterexample (CE) generation using our CheckMate 2.0 approach and the non-compositional CheckMate tool of [21]. Runtimes are given in seconds; *error* means we encountered an exception thrown from CheckMate's Z3 backend.

We note that CheckMate 2.0 requires considerably more SMT solving calls. Notable examples include the Closing Game (38,220 CheckMate 2.0 calls vs. 1 CheckMate call for practicality), 3-Player Routing (546,418 vs. 13 calls for practicality), and Tic Tac Toe (10,694 vs. 1 call for weak(er) immunity and collusion resilience). Despite the higher number of SMT calls in CheckMate 2.0, the SMT queries generated by CheckMate 2.0 are considerably smaller than the ones of CheckMate. Moreover, CheckMate 2.0 calls inhabit a quantifier-free fragment, easing reasoning significantly as reflected in the improved execution times.

In general, CheckMate 2.0 analysis may occasionally result also in suboptimal splits, leading to longer execution times. This issue is exemplified in the Closing game when analyzing practicality of the honest history (C_h, S). Additionally, analyzing collusion resilience can sometimes take longer, particularly when more players are involved, for

example in the Pirate game. This might be explained by the very large number of colluding groups combined with a small game resulting in many trivial SMT calls compared to CheckMate.

Counterexamples. Table 5.2 presents the CheckMate 2.0 runtimes to generate counterexamples compared to CheckMate. It reports the execution time required to find one counterexample (for one case split) as well as finding all counterexamples in all cases for violated security properties. The former is useful for quickly identifying scenarios where the property is not met, while the latter proves particularly helpful when revising and refining a protocol.

The use of compositionality in CheckMate 2.0 demonstrates notable improvements in execution time, particularly when retrieving all counterexamples. Additionally, the execution times for compositional analysis with and without counterexample extraction are quite similar, indicating that CheckMate 2.0 enables counterexample extraction with minimal overhead. The counterexamples to collusion resilience for the Pirate game show this clearly. While CheckMate 2.0 requires slightly more time for property analysis compared to CheckMate, we note that the new CheckMate 2.0 identifies all counterexamples across all cases in approximately 3 seconds, whereas CheckMate takes almost 80 seconds. Similarly, in the case of the 3-Player Routing game, CheckMate 2.0 retrieves all counterexamples for all cases within 291 seconds, while it takes CheckMate over 3,000 seconds (50 minutes).

Strategies. We also compared the two approaches in terms of efficiency for strategy extraction and report the results in Table 5.3. The findings closely mirror the results observed for counterexamples. Firstly, strategy extraction in the compositional approach outperforms the previous method across nearly all benchmarks. Secondly, the compositional approach incurs almost no additional overhead for strategy extraction.

One benchmark that stands out is Tic Tac Toe, where the additional overhead for strategy extraction is clearly noticeable for collusion resilience and practicality. However, strategy extraction for these properties is still achievable within reasonable time, namely 18 seconds for collusion resilience and 276 seconds for practicality. This represents a significant improvement over the non-compositional approach, which takes 347 seconds for collusion resilience and fails to terminate within the 8-hour time limit for practicality.

Sub- and Supertree Reasoning. One of the most significant contributions of the compositional reasoning is that CheckMate 2.0 enables analyzing subtrees independently and integrating only their security results in the supertree. This feature of CheckMate 2.0 is particularly beneficial in larger models. For instance, the 3-Player Routing and Routing Unlocking benchmarks based on the routing protocol [19] are generated using a script, as it is not feasible to model protocols of this size manually. Modeling the routing protocol for 3 players results in a game with 21,688 nodes (3-Player Routing), taking 20 MB on disk.

Next, we detail a more challenging routing example with 4 players, *4-Player Routing*, which has 144,342,306 nodes. This example exceeds our 200 GB of allocated disk space, and thus could not even be created fully. However, by leveraging compositionality, we intertwine model generation and analysis, making it possible to discard generated subtrees after the results of security analysis have been obtained. Specifically, during the game generation process, each subtree corresponding to a specific phase of the protocol called unlocking phase (a total of 1440 subtrees) is analyzed on the fly, with only the results kept. The final outcome, the *4-Player Routing* game, is a supertree with 396 regular nodes and 1440 nodes representing subtrees, or 1,836 nodes in total. The supertree has a size of about 60 MB and in it all subtrees for the unlocking phase have already been solved. This allows us to directly apply CheckMate 2.0 to the supertree. Using CheckMate compositionally, we conclude that 4-Player Routing is weaker immune, but not weak immune, nor collusion resilient, nor practical.

Game	Property	Time CheckMate 2.0/CheckMate
$\text{Splits}_{\text{wi}}$	wi	0.010 / 0.019
(q)	weri	0.010 / 0.018
	cr	0.010 / 0.015
	pr	0.011 / 0.017
$\text{Splits}_{\text{cr}}$	wi	0.011 / 0.018
(n)	weri	0.011 / 0.018
	cr	0.011 / 0.019
	pr	0.011 / 0.018
Market Entry	cr	0.010 / 0.014
(e, i)	pr	0.010 / 0.014
Simplified Closing	wi	0.009 / 0.012
(H)	weri	0.009 / 0.011
	cr	0.009 / 0.012
(C_h, S)	cr	0.010 / 0.012
	pr	0.010 / 0.014
Simplified Routing	weri	0.010 / 0.011
$(S_H, L, L, L, L, U, U, U, U)$	pr	0.010 / 0.012
Auction	weri	0.017 / 0.028
(E, E, I, I)	pr	0.022 / 0.153
Closing	wi	0.011 / 0.025
(H)	weri	0.011 / 0.022
	cr	0.023 / 0.025
(C_h, S)	wi	0.012 / 0.025
	weri	0.011 / 0.021
	cr	0.024 / 0.025
	pr	2.185 / 0.0364
3-Player Routing	weri	0.539 / 1.163
(S_H, L, L, U, U)		
Unlocking Routing	weri	2.194 / 4.233
(U, U, U, U)	pr	4.241 / 5.718
Tic Tac Toe Concise	wi	0.556 / 7.003
$(CM, LU, RU,$	weri	0.561 / 7.780
$LD, LM, RM,$	cr	1.883 / 8.894
$CU, CD, RD)$	pr	8.644 / 219.689
Tic Tac Toe	wi	5.509 / 276.333
$(CM, RU, LU,$	weri	5.507 / 306.763
$RD, RM, LM,$	cr	18.608 / 347.093
$CU, CD, LD)$	pr	276.719 / TO

Table 5.3: Experiments on strategy extraction using our CheckMate 2.0 approach and the non-compositional CheckMate tool of [21]. Runtimes are given in seconds, with a timeout (TO) after 8 hours.

CHAPTER 6

Game-Theoretic Security for Games with Conditional Actions

In terms of expressivity, there are protocols that require modeling and reasoning about uncontrollable game aspects, such as changes in prices and exchange rates of currencies. Intuitively, these aspects represent the world state at a given moment and, as such, are not controllable by any party participating in the protocol (or player in the EFG corresponding to the protocol). A player's possible actions following an uncontrollable event depend on the event's outcome. As a result, actions become conditional – hence, we refer to such uncontrollable aspects of the game as *conditional actions*. To the best of our knowledge, current frameworks and models, particularly CheckMate, do not adequately support the modeling and reasoning of games with conditional actions.

6.1 Games with Conditional Actions

So far, in the games we have seen and analyzed, players have always had a set of actions they could take at each point in the game. Furthermore, at certain points, namely along the honest history, the current player always had exactly one honest action to choose. In the context of conditional actions, however, this is different. Depending on which condition is satisfied (i.e., the current world state), the current player may have different actions to choose from. Additionally, the honest (intended) behavior may vary across different conditions, depending on which condition is satisfied. We will clarify this further in this section by presenting two games involving conditional actions.

Example 6.1 (Liquidation Phase of FAsset)**.** *The Flare Network is a technology designed to provide decentralized finance (DeFi) services and help different blockchains communicate and work together [12]. Users can create wrapped tokens, FAssets [13], for their assets like Bitcoin (BTC) [2], Dogecoin (DOGE) [9] and XRP [25]. For instance, the wrapped*

token for Bitcoin in the FAsset context is called FBTC. These tokens can then be used in Flare's DeFi ecosystem or be returned in order to reclaim the original assets.

To ensure the security of the network, a so-called FAssets collateral *is locked in contracts every time FAssets are minted to ensure that FAssets can always be redeemed for the corresponding assets or compensated by the collateral [11]. If one of the agents that manages the minting and redemption misbehaves in some way, then the assets are reimbursed to the user from the FAsset collateral fund.*

If the value of the collateral drops below a certain threshold due to market fluctuations or agent's misconduct, the agent is being liquidated. Users are then encouraged to return their FAssets and get paid by the agent's collateral [10]. In this process, some actions are only possible if the agent is in liquidation, making them conditional with respect to the market's prices. Hence, the EFG model considered in the preceding chapters needs to be extended if we want to model and analyze the Liquidation Phase of FAsset.

In the following, we revisit some notions that we need to adapt to be able to analyze games involving (external) uncontrollable aspects.

Game Trees. The game trees involving conditional actions have a very similar structure to the game trees we considered so far and introduced in Section 2.1. The only difference now is, that at any node, the choices that the current player can take are grouped into conditions. Each condition represents a possible world state and contains a set of possible actions. Players can only choose from the set of actions (choices) corresponding to the condition satisfied in the moment, i.e. the current world state.

We use $\mathcal{C}(h)$ to refer to the set of conditions after history h. We denote by $A_c(h)$ the set of possible actions following history h that the player $P(h)$ can take if condition $c \in \mathcal{C}(h)$ is satisfied.

Definition 6.1 (Conditional EFG (CEFG))**.** *A* conditional EFG *(CEFG) is an EFG as defined in Definition 2.1, with the following differences:*

- *Non-terminal histories $h \in \mathscr{H} \setminus \mathscr{T}$ have assigned a next player denoted as $P(h) \in N$ and a set of conditions $\mathcal{C}(h)$. Player $P(h)$ chooses from the set $A_c(h)$ of possible actions following h for each condition $c \in \mathcal{C}(h)$.*
- *For terminal histories $h \in \mathscr{T}$, we define $\mathcal{C}(h) = \{true\}$.*
- *For every history $h \in \mathscr{H}$, the set of possible conditions $\mathcal{C}(h)$ satisfies the following requirements:*

  (R1) *Conditions in $\mathcal{C}(h)$ are mutually exclusive:*
  $\forall c_1 \in \mathcal{C}(h), c_2 \in \mathcal{C}(h).\, c_1 \neq c_2 \rightarrow (c_1 \wedge c_2$ *is unsatisfiable).*

  (R2) *Conditions in $\mathcal{C}(h)$ are collectively exhaustive (i.e. span the whole subspace):*
  $\neg(\bigvee_{c \in \mathcal{C}(h)} c)$ *is unsatisfiable.*

(R3) *Conditions* along $h = (a_1, \ldots, a_n)$ *are non-contradictory:*
$\left(\bigwedge_{c \in \mathcal{C}((a_1,\ldots,a_{i-1})) \wedge a_i \in A_c((a_1,\ldots,a_{i-1}))) \text{ for } i \in \{1,\ldots n\}} c \right)$ *is satisfiable.*

In other words, requirement R3 states that if we descend down a path in the tree the set of collected conditions that need to hold can only be refined, but can never get inconsistent. We note that we do not lose any generality by enforcing R3.

For better readability, we will represent the conditions as edges leading to black squares (■). Note that these squares are, in fact, for visualization purposes only and not part of the tree. Hence, they can be removed from the tree by adding the corresponding condition to every edge that is outgoing from a black square.

Honest history. As defined in Section 2.1, the honest history, typically denoted by h^*, has previously been a path in the tree, starting at the root and ending at a leaf. However, this concept of honest history evolves in the context of games with conditional actions. Specifically, when conditional actions are involved, the honest action that is indented to be taken can differ depending on the condition that is satisfied. For each condition $c \in \mathcal{C}(h)$ available after a non-terminal history $h \in \mathscr{H} \setminus \mathscr{T}$, there exists exactly one honest action $a_c^* \in A_c(h)$. As a result, the honest history no longer represents a simple path, but instead takes the form of a tree within the game tree, i.e. it is a set of edges of the game tree (respectively actions in the game) which do not form a simple path.

Note that even though the honest behavior is not a history in the strict sense anymore, we will still use the term honest history to refer to the honest behavior in the remaining part of this thesis.

Definition 6.2 (Honest Behavior for Conditional Actions)**.** *Let $G = (V, E)$ be a game with conditional actions. The honest behavior $h^* \subset E$ is described as follows:*

- *At the root node, namely for $h = \emptyset$, for every possible condition $c \in \mathcal{C}(h)$, we fix one action $a_c^* \in A_c(h)$, $a_c^* \in h^*$ that is meant to be the action that the current player $P(h)$ is intended to take if the world state satisfies condition c.*
- *In each subgame $\Gamma_{|\emptyset,c}$ restricted to a condition $c \in \mathcal{C}(h)$ we proceed by repeating the previous step for every condition $c' \in \mathcal{C}((a_c^*))$, i.e. the game that is still to be played after each previously picked honest action.*

The properties of EFGs introduced in Definition 2.2 can be extended to CEFGs as follows.

Definition 6.3 (Properties of CEFGs)**.** *Let $\Gamma = (N, G)$ be a CEFG.*

Strategy A *strategy* σ for a group of players $S \subseteq N$ is a function mapping non-terminal histories $h \in \mathscr{H} \setminus \mathscr{T}$, where one of the players in group S has a turn $P(h) \in S$, to a

possible action $a \in A_c(h)$ for every condition $c \in \mathcal{C}(h)$. We write $\mathscr{S}_S$ for the set of strategies for group S, and $\mathscr{S}$ for $\mathscr{S}_N$ which we call *joint strategies*. We refer to the union of strategies with disjoint domains as a *combined strategy* and denote it as a tuple. To combine e.g. $\sigma_S \in \mathscr{S}_S$ and $\tau_{N-S} \in \mathscr{S}_{N-S}$, we write $(\sigma_S, \tau_{N-S}) \in \mathscr{S}$. We use $\sigma_c(h) = a$ for $a \in A_c(h)$ to denote the action a is fixed to be taken by strategy σ after history h in the case of condition c.

World State A *world state* or *world model* is a function mapping each $h \in \mathscr{H}$ to a satisfied condition $c \in \mathcal{C}(h)$. The set of all world states is denoted by $\mathcal{M}$. Given a fixed world state $m \in \mathcal{M}$ and a (sub)game $\Gamma_{|h}$, the game restricted to only conditions satisfied in m is denoted by $\Gamma_{|m,h}$.

Resulting History The *resulting terminal history* $H_m(\sigma)$ of a joint strategy $\sigma \in \mathscr{S}$ and world state $m \in \mathcal{M}$ is the unique history obtained by following chosen actions $\sigma_c(h)$ from root to leaf, taking into consideration which condition c is satisfied in the current (or fixed) world state m.

Following Honest History A strategy for a player p follows the honest history h^* if, at every node along the honest history, where p is making a choice, the strategy chooses the action in h^* for every possible condition $c \in \mathcal{C}(h)$. For every other node, there is no constraint.

Utility Function The *utility function* $u_{m,p}(\sigma)$ assigns to player $p \in N$ their utility at the resulting history of the joint strategy $\sigma \in \mathscr{S}$, corresponding to world state $m \in \mathcal{M}$. If we are dealing with a terminal history, i.e. a unique path in the tree which already takes the current world state $m \in \mathcal{M}$ into consideration we can also write $u_p(H_m(\sigma))$ for $u_{m,p}(\sigma)$. We sometimes write all player utilities for a joint strategy $\sigma \in \mathscr{S}$ as $u_m(\sigma)$ (or $u(H_m(\sigma))$), denoting a tuple of size $|N|$.

Subgame *Subgames* $\Gamma_{|h}$ of Γ are formed from the same set N of players and a subtree of G, and are therefore identified by the history h leading to the subtree. Histories $\mathscr{H}_{|h}$ of $\Gamma_{|h}$ are histories in $\mathscr{H}$ with prefix h, strategies $\sigma_{|h} \in \mathscr{S}_{|h}$ of $\Gamma_{|h}$ are strategies restricted to the nodes in $G_{|h}$ and the utility function $u_{|h}$ of $\Gamma_{|h}$ assigns each joint strategy $\sigma_{|h} \in \mathscr{S}_{|h}$ the utility of the yielded leaf $u_{|h}(H_m(\sigma_{|h})) := u(h, H_m(\sigma_{|h}))$. Additionally, a *subgame* $\Gamma_{|h}$ of Γ can be restricted to the chosen or currently satisfied condition $c \in \mathcal{C}(h)$ and this is denoted by $\Gamma_{|h,c}$.

Prefix of Honest History A history $h = (a_1, \ldots, a_n)$ is a prefix of the honest history (behavior) h^* iff for each action $a_i \in h$ along h is holds that the action is an honest action, that is, $a_i \in h^*$. Recall that the honest behavior is a tree within the game tree now and represented by a set of edges.

Supergame A game $\Gamma_{|h'}$ is a *supergame* of a game $\Gamma_{|h}$ if h' is a prefix of h.

Subtree along/off Honest History Let h^* be the honest history. A subgame $\Gamma_{|h}$ is *along* the honest history iff all actions along h are in h^*; otherwise, $\Gamma_{|h}$ is *off* the honest history.

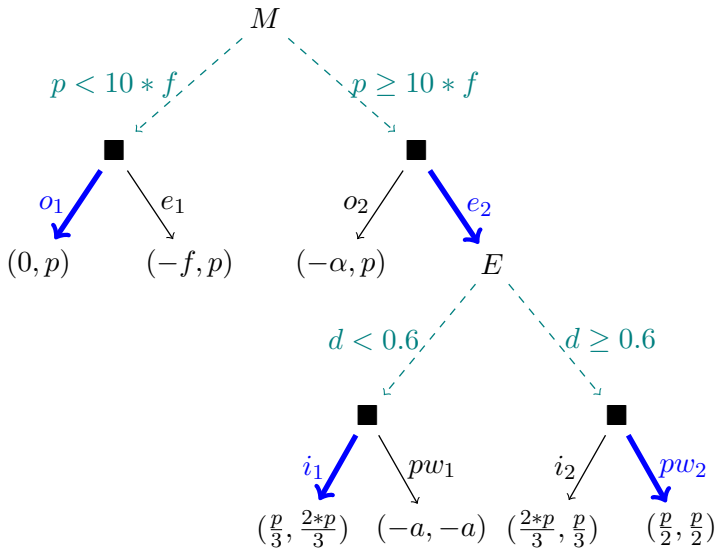

Figure 6.1: Market Entry game with Conditional Actions Γ_{me_ca}, with $a, p, f > 0$, $0 \leq d \leq 1$ and infinitesimal α. Utility tuples state M's utility first, E's second. Conditions are represented with teal dashed lines and the honest history with thick blue lines.

The newly revised notions for the game trees and the honest history can be better understood through Example 6.2.

Example 6.2 (Market Entry Game with Conditional Actions)**.** *Consider the Market Entry game with conditional actions Γ_{me_ca} given in Figure 6.1. We have the two players: M, representing a new company that is considering entering the market and E – an established company. At the root, it is the turn of player M. Now, there is a condition depending on the values of the profit that can be obtained from the market (p) and the fee that one needs to invest to enter the market (f). Namely, $\mathcal{C}(\emptyset) = \{p < 10 * f, p \geq 10 * f\}$.*

If the world state satisfies the condition $p < 10 * f$, M can choose one of the actions $A_{p<10*f}(\emptyset) = \{o_1, e_1\}$; namely staying out of the market (action o_1) or entering the market (action e_1). The intended behavior (honest history) at this point of the game is to take action o_1, after which M gets utility 0 and all the profit p remains for the established company E. If M deviates and takes action e_1, they lose the fee f they paid and therefore obtain utility $-f$, whereas the utility of E remains p.

On the other hand, if the condition $p \geq 10 * f$ is satisfied, M can choose from $A_{p\geq 10*f}(\emptyset) = \{o_2, e_2\}$. If M chooses action o_2 they receive utility $-\alpha$, where α is infinitesimal representing opportunity cost. Otherwise, M takes action e_2 and it is E's turn.

At this point of the game, there is another condition, that determines the possible actions for player E. This condition takes the degree d into account: a value between 0 and 1 measuring the potential and quality of M's product.

If the condition $d < 0.6$ is satisfied, we consider M's product to be inferior to E's product. Hence, out of the actions $A_{d<0.6}(e_2) = \{i_1, pw_1\}$ we fix the intended behavior for E to be ignoring M and taking action i_1. In this case E still gets the majority of the profit, namely $\frac{2*p}{3}$, whereas M gets $\frac{p}{3}$. Otherwise, E can also choose to start a price war and take action pw_1, which causes both companies to lose some assets (a).

If the condition $d \geq 0.6$ is satisfied, we consider M's product to be superior to E's product. Thus, out of the actions $A_{d\geq 0.6}(e_2) = \{i_2, pw_2\}$ we fix the intended behavior for E to be staring a price war (pw_2) to make themselves more appealing to consumers. If E only ignores M instead and takes action i_2, E will only get one-third of the market's profit and the majority (two-thirds) will remain for M.

From Figure 6.1 (thick blue lines) and from the description above we can also observe how the honest history is not a path anymore, but forms in tree.

We will revisit this example later to analyze whether the Market Entry game with conditional actions satisfies the game-theoretic security properties weak(er) immunity, collusion resilience and practicality.

In the following, we examine the requirements imposed on Conditional EFGs, introduced in Definition 6.1, and justify these choices through examples.

In the next example, we discuss why the Market Entry game with conditional actions shown in Figure 6.1 satisfies the requirements in Definition 6.1.

Example 6.3 (Conditional EFG Requirements for Market Entry Game with Conditional Actions)**.** *Looking at the Market Entry game with conditional actions Γ_{me_ca} given in Figure 6.1 we can show that the requirements (R1-R3) given in Definition 6.1 are satisfied.*

After history $\emptyset$ we have the conditions $p < 10 * f$ and $p \geq 10 * f$ for which we can easily observe that they are mutually exclusive and collectively exhaustive, thereby complying with R1 and R2. Analogously, the same holds for the conditions $d < 0.6$ and $d \geq 0.6$ after history (e_2).

Moreover, there are some histories along which there are multiple conditions. For example, along history (e_2, i_2) we have the conditions $p \geq 10 * f$ and $d < 0.6$. These are not contradictory, so R3 is not violated. The same conclusion can be reached for the other histories along which we have more than one condition.

In addition, we justify requirements R1-R3 with the following two examples.

Example 6.4 (Requirements R1-R2 for Conditional Actions)**.** *Consider the very simple example depicted in Figure 6.2, in which player A can either buy (actions b_1, b_2) or sell (actions s_1, s_2) some assets. Depending on the value of the price coefficient p in the current world state, A can either win and receive a utility 1 or lose and receive utility 0. However, we observe that this game does not satisfy requirement R1, because the conditions at the root, namely $\mathcal{C}(\emptyset) = \{p \leq 0.5, p \geq 0.5\}$ are not mutually exclusive. Imagine a world*

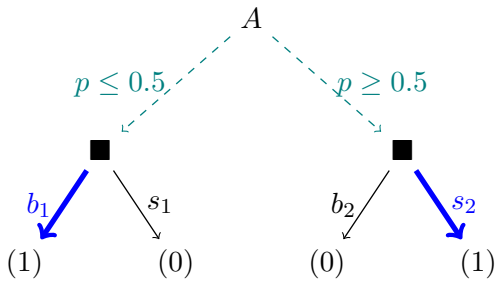

Figure 6.2: Example Γ_{ca_R1} for requirement R1 of games with conditional actions, $0 \leq p \leq 1$.

state where $p = 0.5$. In this case, it is neither clear what the intended behavior of player A is, nor what is the obtained utility based on the actions that A takes. Hence, reasoning over this game is impossible.
Similarly, imagine the conditions are strict inequalities, i.e. $\mathcal{C}(\emptyset) = \{p > 0.5, p < 0.5\}$. This variation of the game violates requirement R2. Again, in the case when $p = 0.5$, we are stuck in the game without reaching a leaf. In other words, there is no adequate part of the game which depicts this world state and we cannot analyze the game.

Example 6.5 (Requirement R3 for Conditional Actions)**.** *Consider the Market Entry game with conditional actions Γ_{me_ca} given in Figure 6.1 but instead of $\mathcal{C}((e_2)) = d < 0.6, d \geq 0.6$ let $\mathcal{C}((e_2)) = p < 10 * f, p \geq 10 * f$. Both $\mathcal{C}(\emptyset)$ and $\mathcal{C}((e_2))$ satisfy requirement R1 and requirement R2.*
However, we can observe that along history (e_2, i_1) there are contradictory conditions, namely $p \geq 10 * f$ (condition at the root) and $p < 10 * f$ (condition after history (e_2)). This is problematic from two different aspects. First of all, from an intuitive point of view, the history (e_2, i_1) and the leaf related to it correspond to actions which are taken in a world state where both $p \geq 10 * f$ and $p < 10 * f$ hold which is impossible. Secondly, as discussed later in Section 6.2, we sometimes want to make sure that for a history h for all conditions $c \in \mathcal{C}(h)$ the subgame $\Gamma_{|h,c}$ is secure. Having contradictory conditions along a history compromises the security analysis as they do not comply with the world state and are trivially not possible to satisfy the security properties, which is why we opt to model games in a way they satisfy requirement R3.

As a final point in this section, we note that the games we have considered so far can be observed as games with conditional actions, where at each node of the tree the only available condition is the trivial condition `true`.

Theorem 6.1. *Every EFG (Definition 2.1) is a CEFG (Definition 6.1), where $\forall h \in \mathscr{H}. \mathcal{C}(h) = \{true\}$.*

Proof. An EFG $\Gamma = (N, G)$ can be identified with an EFG $\Gamma^c = (N, \Gamma^c)$ where Γ^c is a game tree involving conditional actions. The game trees considered in Chapter 2-Chapter 5 can

be naturally extended to game trees with conditional actions by associating every history $h \in \mathscr{H}$ with the set consisting of only the trivial condition `true`, i.e. $\mathcal{C}(h) = \{true\}$. Additionally, we set $A_{\texttt{true}}(h) = A(h)$. Note that the extension of Γ to a game with trivial conditions Γ^c satisfies requirements R1-R3 from Definition 6.1.

The notion and format of honest history h^* coincides in both cases. To construct an honest history for the newly constructed game tree with trivial conditions, we need to start at the root and fix an honest action for each possible condition. Since there is only one possible condition, we fix one honest action $a \in A(\emptyset)$ and need to repeat the construction recursively for the subtree after the picked action. Since we always have only one condition, the resulting honest history is a simple path, just like for game trees with no conditional actions.

The remaining properties either coincide or can be extended analogously. For example, the notions of *Strategy* and *Following Honest History* for games with conditional actions differ from the ones without conditional actions in the sense that they require that some properties are satisfied for all possible conditions. Since our newly constructed game trees with conditional actions only have one condition at each inner node, the notions coincide. The notion of *Resulting History* requires to follow conditions which are satisfied in the current world state. In the newly constructed game trees the only condition we have is `true` and it is trivially satisfied in every world state.

□

6.2 Security Properties for Conditional EFGs

For analyzing games involving conditional actions, we define adaptations of the game-theoretic security properties *weak(er) immunity*, *collusion resilience* and *practicality*. We assume a fixed total order on symbolic utility terms, as in Chapter 3–Chapter 4. This assumption is lifted in Section 6.4 in the same manner as in Section 5.1 – if some utility terms cannot be compared, the algorithm takes care of it by performing case splitting and analyzing all possible consistent total orders of the real/infinitesimal variables occurring in the utility terms.

Games restricted to different conditions at a node are independent from each other (recall requirements given in Definition 6.1), which allows us to define the security properties in a compositional manner. Moreover, in the sequel, we follow our compositional approach from the previous chapters and employ player-wise reasoning, as well as propagation of results from subtrees to supertrees.

As we have seen in Section 6.1, there can be one or more conditions $c \in \mathcal{C}(h)$ after a history $h \in \mathscr{H}$. Hence, for the analysis of games with conditional actions we introduce two notions of the security properties: the *weak notion of the security properties* and the *strong notion of the security properties*.

As we will see in the sequel, the strong notion takes into account all possible consistent world states. While this approach ensures a high degree of generality, it may prove to

be overly restrictive or impractical in certain contexts. For instance, there could be a large amount of possible world states and there could be one among them such that the probability of this one specific possible world state happening may be negligible. Therefore, we also introduce the weak notion to offer a more tractable and context-sensitive alternative. In reality, it is often sufficient to have one particular scenario in which security properties are ensured to hold.

Weak Notion of Security Properties. The intuition behind the weak notion of the security properties is that for a (sub)game, there exists a world state $m \in \mathcal{M}$ such that if it is satisfied, the (sub)game satisfies the security property. Note that in the (sub)game only actions consistent with m are allowed.

We start by defining how satisfiability of weak(er) immunity for a fixed player and a game restricted to one condition can be determined compositionally, following the approach from Theorem 4.4.

Definition 6.4 (Conditional Weak Immunity for a Player – Weak Notion)**.** *Let $\Gamma_{|h,c}$ be a subgame of a CEFG Γ with honest behavior h^*. We define weak immunity of $\Gamma_{|h,c}$ for player $p \in N$ in terms of the weak notion – denoted by $wi_{w,p}(\Gamma_{|h,c})$ – recursively:*

1) If $\Gamma_{|h,c}$ is a leaf, then it is weak immune for p in terms of the weak notion iff p's utility is non-negative:

$$\forall t \in \mathscr{T}.\, wi_{w,p}(\Gamma_{|t,c}) \quad \Leftrightarrow \quad u_p(t) \geq 0\,.$$

2) If $\Gamma_{|h,c}$ is a branch and p is not the current player, then it is weak immune for p, iff all children $\Gamma_{|(h,a)}$ for $a \in A_c(h)$ can be restricted to a condition $c_a \in \mathcal{C}((h,a))$ such that $\Gamma_{|(h,a),c_a}$ is weak immune for p in terms of the weak notion:

$$\forall h \in \mathscr{H}\backslash\mathscr{T}.\, p \neq P(h) \;\Rightarrow\; (wi_{w,p}(\Gamma_{|h,c}) \;\Leftrightarrow\; \forall a \in A_c(h).\, \exists \boldsymbol{c_a \in \mathcal{C}((h,a))}.\, wi_{w,p}(\Gamma_{|(h,a),c_a}))\,.$$

3) If $\Gamma_{|h,c}$ is a branch along the honest history and p is the current player p, then it is weak immune for p, iff the child $\Gamma_{|(h,a^*)}$ following h^* restricted to some condition $c_{a^*} \in \mathcal{C}((h,a^*))$ is weak immune for p in terms of the weak notion. Note that $a^* \in A_c(h)$ represents the honest choice, i.e. (h,a^*) along h^*:

$$\begin{aligned} &\forall h \in \mathscr{H} \setminus \mathscr{T}.\, p = P(h) \wedge h \text{ along } h^* \quad \Rightarrow \\ &(wi_{w,p}(\Gamma_{|h,c}) \;\Leftrightarrow\; \exists \boldsymbol{c_{a^*} \in \mathcal{C}((h,a^*))}.\, wi_{w,p}(\Gamma_{|(h,a^*),c_{a^*}}))\,. \end{aligned}$$

4) If $\Gamma_{|h,c}$ is a branch off the honest history and p is the current player p, then it is weak immune for p iff there exists a child $\Gamma_{|(h,a)}$ after some $a \in A(h)$ such that $\Gamma_{|(h,a)}$ restricted to some condition $c_a \in \mathcal{C}((h,a))$ is weak immune for p in terms of the weak notion:

$$\begin{aligned} &\forall h \in \mathscr{H} \setminus \mathscr{T}.\, p = P(h) \wedge h \text{ off } h^* \quad \Rightarrow \\ &\quad (wi_{w,p}(\Gamma_{|h,c}) \;\Leftrightarrow\; \exists a \in A_c(h).\, \exists \boldsymbol{c_a \in \mathcal{C}((h,a))}.\, wi_{w,p}(\Gamma_{|(h,a),c_a}))\,. \end{aligned}$$

Definition 6.5 (Conditional Weak Immunity – Weak Notion)**.** *A subgame $\Gamma_{|h,c}$ of a CEFG Γ with honest behavior h^* is weak immune in terms of the weak notion iff $\Gamma_{|h,c}$ is weak immune in terms of the weak notion for every player $p \in N$, that is:*

$$wi_w(\Gamma_{|h,c}) \Leftrightarrow \forall p \in N.\ wi_{w,p}(\Gamma_{|h,c}). \qquad (\mathrm{wi}_w(\Gamma_{|h,c}))$$

Further, the subgame $\Gamma_{|h}$ of a CEFG Γ not restricted to any condition is weak immune in terms of the weak notion of this security property, iff there exists a condition $c \in \mathcal{C}(h)$ such that $\Gamma_{|h,c}$ is weak immune in terms of the weak notion, that is:

$$wi_w(\Gamma_{|h}) \Leftrightarrow \exists \boldsymbol{c \in \mathcal{C}(h)}.\ wi_w(\Gamma_{|h,c}). \qquad (\mathrm{wi}_w(\Gamma_{|h}))$$

Recall that by definition, for a terminal history $h \in \mathscr{T}$, we have defined $\mathcal{C}(h) = \{true\}$. Hence, checking if $wi_w(\Gamma_{|h})$ holds for a terminal history h refers to determining whether the leaf is weak immune.

For weaker immunity, $weri_{w,p}(\Gamma_{|h,c})$, $weri_w(\Gamma_{|h,c})$ and $weri_w(\Gamma_{|h})$ are defined analogously to Definition 6.4 and Definition 6.5.

Example 6.6 (Conditional Weak Immunity – Weak Notion)**.** *Consider the Market Entry game with conditional actions given in Figure 6.2 and let us examine whether weak immunity is satisfied in terms of the weak notion of the property.*

We consider one player at a time, starting with player M. For the weak notion of security properties for games with conditional actions there should be at least one condition such that the subtree restricted to this condition is weak immune. Recall that since M is the player who has a turn and we are along the honest history, we need to choose a condition where the honest choice leads to a subtree weak immune for M. We see that for the condition $p < 10 * f$, there exists a strategy in which the honest history o_1 is taken and the game ends with a non-negative utility for M. Hence, weak immunity is satisfied for M in terms of the weak notion.

Moving to E, at the root we again look for one condition where the game restricted to this condition is weak immune for E. Since E is not the player who has a turn at the root, all the actions in the chosen condition need to be weak immune for E. Again, we can see that for the condition $p < 10 * f$ this holds: no matter whether M chooses o_1 or e_1, the game will end with non-negative utility for E. Thus, weak immunity is satisfied for E in terms of the weak notion.

Note that in this example, we used the condition $p < 10 * f$ twice as a "witness" for the satisfiability of weak immunity (one time for each player). In general, the conditions justifying the satisfiability might differ across players/player groups.

Collusion resilience against a deviating group $S \subset N$ ($cr_{w,S}(\Gamma_{|h,c})$) is defined analogously to the one in Definition 6.4, following Theorem 4.5.

Definition 6.6 (Conditional Collusion Resilience against a Player Group – Weak Notion)**.** *Let $\Gamma_{|h,c}$ be a subgame of a CEFG Γ with honest behavior h^*. We define collusion resilience of $\Gamma_{|h,c}$ against player group $S \subset N$ in terms of the weak notion – denoted by $cr_{w,S}(\Gamma_{|h,c})$ – recursively:*

1) If $\Gamma_{|h,c}$ is a leaf, then it is collusion resilient against $S \subset N$ in terms of the weak notion iff the honest joint utility of the deviating players $p \in S$ is greater than or equal to their joint utility at that leaf:

$$\forall t \in \mathscr{T}.\ cr_{w,S}(\Gamma_{|t,c}) \quad \Leftrightarrow \quad \sum_{p \in S} u_p^* \geq \sum_{p \in S} u_p(t)\ .$$

2) If $\Gamma_{|h,c}$ is a branch and the current player is in the deviating group $S \subset N$, it is collusion resilient against S iff all children $\Gamma_{|(h,a)}$ for $a \in A_c(h)$ can be restricted to a condition $c_a \in \mathcal{C}((h,a))$ such that $\Gamma_{|(h,a),c_a}$ is collusion resilient against S in terms of the weak notion:

$$\begin{aligned} &\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \in S \quad \Rightarrow \\ &\quad (cr_{w,S}(\Gamma_{|h,c}) \ \Leftrightarrow\ \forall a \in A_c(h).\, \boldsymbol{\exists c_a \in \mathcal{C}((h,a))}.\, cr_{w,S}(\Gamma_{|(h,a),c_a}))\ . \end{aligned}$$

3) If $\Gamma_{|h,c}$ is a branch along the honest history h^* and the current player is not in the deviating group $S \subset N$, it is collusion resilient against S iff the child $\Gamma_{|(h,a^*)}$ following h^* restricted to some condition $c_{a^*} \in \mathcal{C}((h,a^*))$ is collusion resilient against S in terms of the weak notion. Note that $a^* \in A(h)$ represents the honest action, i.e. (h,a^*) along h^*, then:

$$\begin{aligned} &\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \notin S \wedge h \text{ along } h^* \quad \Rightarrow \\ &(cr_{w,S}(\Gamma_{|h,c}) \ \Leftrightarrow\ \boldsymbol{\exists c_{a^*} \in \mathcal{C}((h,a^*))}.\, cr_{w,S}(\Gamma_{|(h,a^*),c_{a^*}}))\ . \end{aligned}$$

4) If $\Gamma_{|h,c}$ is a branch off the honest history h^*, and the current player is not in the deviating group $S \subset N$, it is collusion resilient against S iff there exists a child $\Gamma_{|(h,a)}$ after some $a \in \mathcal{C}(h)$ such that $\Gamma_{|(h,a)}$ restricted to some condition $c_a \in \mathcal{C}((h,a))$ is collusion resilient against S in terms of the weak notion:

$$\begin{aligned} &\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \notin S \wedge h \text{ off } h^* \quad \Rightarrow \\ &\quad (cr_{w,S}(\Gamma_{|h,c}) \ \Leftrightarrow\ \exists a \in A_c(h).\, \boldsymbol{\exists c_a \in \mathcal{C}((h,a))}.\, cr_{w,S}(\Gamma_{|(h,a),c_a}))\ . \end{aligned}$$

Definition 6.7 (Conditional Collusion Resilience – Weak Notion)**.** *A subgame $\Gamma_{|h,c}$ of a CEFG Γ with honest behavior h^* is collusion resilient in terms of the weak notion iff $\Gamma_{|h,c}$ is collusion resilient in terms of the weak notion for every player group $S \subset N$, that is:*

$$cr_w(\Gamma_{|h,c}) \Leftrightarrow \forall S \subset N.\ cr_{w,S}(\Gamma_{|h,c}). \qquad (\mathrm{cr}_w(\Gamma_{|h,c}))$$

Further, the subgame $\Gamma_{|h}$ of a CEFG Γ not restricted to any condition is collusion resilient in terms of the weak notion of this security property, iff there exists a condition $c \in \mathcal{C}(h)$ such that $\Gamma_{|h,c}$ is collusion resilient in terms of the weak notion, that is:

$$cr_w(\Gamma_{|h}) \Leftrightarrow \boldsymbol{\exists c \in \mathcal{C}(h)}.\ cr_w(\Gamma_{|h,c}). \qquad (\mathrm{cr}_w(\Gamma_{|h}))$$

Conditional practicality can be determined similarly, following Theorem 4.6. This is formally stated in the next two definitions.

Definition 6.8 (Conditional Practicality – Weak Notion)**.** *Let $\Gamma_{|h,c}$ be a subgame of a CEFG Γ with honest behavior h^* and $\mathbb{U}(h)$ be the set of practical utilities of subtree $\Gamma_{|h}$ in terms of the weak notion. We define practicality of $\Gamma_{|h,c}$ denoted by $pr_w(\Gamma_{|h,c})$ – recursively:*

1) If $\Gamma_{|h,c}$ is a leaf, that is $h \in \mathscr{T}$, it is always practical and the only practical utility is that of the leaf. Then:

$$pr_w(\Gamma_{|h,c}) \Leftrightarrow \texttt{true}$$
$$\mathbb{U}(\Gamma_{|h}) = \{u(h)\} .$$

2) The honest history is practical in terms of the weak notion in a branch of Γ along h^* restricted to a condition c iff it is practical in terms of the weak notion in the subgame following h^* and for every other child in the condition c and every practical utility u^* in the subgame along h^*, at least one compatible practical utility is not greater than u^* for the current player. Here, compatibility (comp) refers to the requirement that the conditions along the paths leading to the corresponding leaves–where the utilities are compared–must be satisfiable.
Let $a^* \in A_c(h)$ be the honest action after h complying with condition c, then:

$$\begin{aligned}\forall h \in \mathscr{H} \setminus \mathscr{T}.\ h \text{ along } h^*.\ \forall c \in \mathcal{C}(h) \quad \Rightarrow \quad & \big(pr_w(\Gamma_{|h,c}) \quad \Leftrightarrow \\ & pr_w(\Gamma_{|(h,a^*)}) \wedge \\ & \forall a \in A_c(h) \setminus \{a^*\}\ \forall u^* \in \mathbb{U}((h,a^*))\ \exists u \in \mathbb{U}((h,a))\ comp(u,u^*) \wedge u^*_{P(h)} \geq u_{P(h)}\big) .\end{aligned}$$

3) A utility is practical in terms of the weak notion in a branch of Γ off the honest history h^* restricted to condition c iff it is practical in terms of the weak notion in a child and if, for every other child, at least one compatible practical utility is not greater for the current player. The definition of compatibility is the same as in the previous item.

$$\begin{aligned}\forall h \in \mathscr{H} \setminus \mathscr{T}.\ h \text{ off } h^* \quad \Rightarrow \quad & \big(\forall t \in \mathscr{T}_{|h}.\ u(t) \in \mathbb{U}(h) \quad \Leftrightarrow \\ & \exists a \in A_c(h)\ u(t) \in \mathbb{U}((h,a)) \wedge \\ & \forall a' \in A_c(h) \setminus \{a\}\ \exists u' \in \mathbb{U}((h,a'))).\ comp(u,u') \wedge u_{P(h)}(t) \geq u'_{P(h)}\big) .\end{aligned}$$

Definition 6.9 (Conditional Practicality – Weak Notion)**.** *A subgame $\Gamma_{|h}$ of a CEFG Γ with honest behavior h^* not restricted to any condition is practical in terms of the weak notion of this security property, iff there is a condition $c \in \mathcal{C}(h)$ such that the game restricted to this condition is practical:*

$$pr_w(\Gamma_{|h}) \Leftrightarrow \exists \boldsymbol{c \in \mathcal{C}(h)}.\ pr_w(\Gamma_{|h,c}) \qquad (\mathrm{pr}_w(\Gamma_{|h}))$$

Providing support for the analysis of games involving conditional actions feels intuitive and natural, as they seamlessly build on top of the game structure we previously established.

They also come at a low cost, since we can leverage the security properties stratified over players and their associated splitting and combining techniques (Theorem 4.2, Theorem 4.4, Theorem 4.5, Theorem 4.6). Note that, we apply the techniques for splitting and combining (Section 4.3) in an adapted manner, namely to subtrees of the game resulting when restricting the game to one condition at a time. This is further elaborated in the following example, as well as in Section 6.4.

Example 6.7 (Conditional Practicality – Weak Notion)**.** *We revisit the Market Entry game with conditional actions given in Figure 6.2–this time to analyze whether it satisfies the weak notion of practicality.*

This game is not a single leaf, so by Definiton 6.9, it follows that we need to find one condition at the root, for which the game restricted to this condition is practical. We can first check if the game restricted to condition $p < 10 * f$ at the root is practical in terms of the weak notion. We observe that both actions in $\Gamma_{me\_ca_{p<10*f,\emptyset}}$ lead to leaves. Leaves are always practical the practical utilities are simply the utilities contained in them. So, we set $\mathbb{U}((o_1)) = \{(0, p)\}$ and $\mathbb{U}((e_1)) = \{(-c, p)\}$. Moving upwards, we apply Definition 6.8, in particular item 1), as the root is along the honest history. We know that the subgame after history (o_1) is practical (because it is a leaf) and its only practical utility is $(0, p)$. Furthermore, for the other action possible in this condition, namely (e_1) it holds that its only practical utility is $(-c, p)$, which is compatible to $(0, p)$, as they both comply with condition $p < 10 * f$, that is, the conditions needed to be satisfied to reach these leaves are not unsatisfiable. Moreover, we know that $0 \geq -f$, since we have $f > 0$ as initial constraint. Thus, $pr_w(\Gamma_{me\_ca_{p<10*f,\emptyset}})$ holds and the game is practical in terms of the weak notion of this property.

After defining the weak notions of all four game-theoretic security properties, we proceed with a theorem about their relation to security properties for games without conditional actions.

Theorem 6.2 (Equivalence of Weak Notion of Security Properties of CEFGs and Security Properties of EFGs)**.** *Let Γ be an EFG and Γ^c its equivalent CEFG, as in Theorem 6.1. Then for $sp \in \{wi, weri, cr, pr\}$ the following equivalence holds:*

$$sp(\Gamma) \Leftrightarrow sp_w(\Gamma^c),$$

where $sp(\Gamma)$ is according to Definition 2.3–Definition 2.6, and $sp_w(\Gamma^c)$ is according to Definition 6.5, Definition 6.7 and Definition 6.9.

Proof. Consider the property weak immunity, the empty history $h = \emptyset$ and Definition 6.5. According to this definition, Γ^c satisfies the security property if there is any condition $c \in \mathcal{C}(\emptyset)$ for which the subgame restricted to this condition satisfies the security property for every player $p \in N$. Moreover, by construction of Γ^c, we know that $\mathcal{C}(h) = \{true\}$ for any history h. Thus, Γ^c satisfies the weak notion of the security property iff $\Gamma^c_{|\emptyset,true}$ satisfies the weak notion of the security property. Proceeding recursively for the subgames

and always having only the one trivial condition, we can conclude that this is the only condition which can be used for showing that the weak notion of the security property for Γ^c is satisfied. Hence, checking whether the weak notion of a security property is satisfied for a game with only trivial conditions essentially boils down to checking whether the security property is satisfied in Γ. This can be argued for the rest of the properties analogously. □

Strong Notion of Security Properties. Contrary to the weak notion, security in regard to the strong notion of the security properties ensures that a (sub)game always satisfies a security property independently on the condition that is satisfied (i.e. the current or fixed world state).

Definition 6.10 (Conditional Weak Immunity – Strong Notion)**.** *A subgame $\Gamma_{|h}$ of a CEFG Γ with honest behavior h^* satisfies the strong notion of the security property weak immunity – denoted by $wi_s(\Gamma_{|h})$ – iff $\Gamma_{|h}$ satisfies a modified version of $wi_w(\Gamma_{|h})$ where the existential quantification over $c \in \mathcal{C}(h)$ is replaced by universal quantification. Specifically, in the definition of the strong notion of weak immunity, every instance of existential quantification emphasized in blue bold font in Definition 6.4 and Definition 6.5 is replaced by universal quantification.*

The strong notion of weaker immunity is defined analogously to the one for weak immunity given in Definition 6.10.

Definition 6.11 (Conditional Collusion Resilience – Strong Notion)**.** *A subgame $\Gamma_{|h}$ of a CEFG Γ with honest behavior h^* satisfies the strong notion of the security property collusion resilience – denoted by $cr_s(\Gamma_{|h})$ – iff $\Gamma_{|h}$ satisfies a modified version of $cr_w(\Gamma_{|h})$ where the existential quantification over $c \in \mathcal{C}(h)$ is replaced by universal quantification. Specifically, in the definition of the strong notion of collusion resilience, every instance of existential quantification emphasized in blue bold font in Definition 6.6 and Definition 6.7 is replaced by universal quantification.*

Definition 6.12 (Conditional Practicality – Strong Notion)**.** *A subgame $\Gamma_{|h}$ of a CEFG Γ with honest behavior h^* satisfies the strong notion of the security property practicality – denoted by $pr_s(\Gamma_{|h})$ – iff $\Gamma_{|h}$ satisfies a modified version of $pr_w(\Gamma_{|h})$ where the existential quantification over $c \in \mathcal{C}(h)$ is replaced by universal quantification. Specifically, in the definition of the strong notion of collusion resilience, every instance of existential quantification emphasized in blue bold font in Definition 6.8 and Definition 6.9 is replaced by universal quantification.*

Example 6.8 (Conditional Weak Immunity – Strong Notion)**.** *We revisit the Market Entry game with conditional actions given in Figure 6.1 and analyze whether weak immunity is satisfied in terms of the strong notion of this property.*

Starting with player M we see that it is M's turn in the beginning. For each condition $c \in \mathcal{C}(\emptyset) = \{p < 10 * f, p \geq 10 * f\}$ we need to check whether the subgame restricted to

this condition $\Gamma_{|\emptyset,c}$ is weak immune. All conditions need to be checked, as we are dealing with the strong notion of the security properties. Since M is the player that has a turn and we are along the honest history, it means that we need to check that the honest action in each condition leads to a subgame that is weak immune for M.

In $\Gamma_{|p<10*f,\emptyset}$ the subgame after the honest action o_1 is a leaf with non-negative utility for M and thus weak immune for M.

In $\Gamma_{|p\geq 10*f,\emptyset}$ the subgame after the honest action e_2 leads to a leaf where player E has a turn. We again need to check all conditions. Moreover, since the player we are checking weak immunity for is M and it is E's turn, we need to make sure that all actions in each condition lead to subtrees that are weak immune for M. We can easily see that this is not the case, as if condition $d < 0.6$ is satisfied, E can take action pw_1 and the game ends with a negative utility for M. Hence, Γ_{me_ca} is not weak immune for player M in terms of the strong notion of weak immunity.

In a similar manner, we can analyze weak immunity for player E and conclude that this property is satisfied for player E in terms of the strong notion.

In summary, we state that the Market Entry game with conditional actions is not weak immune in terms of the strong notion of this security property (because it is not weak immune for player M).

Example 6.9 (Conditional Practicality – Strong Notion)**.** *In Example 6.7 we analyzed practicality in terms of the weak notion for the Market Entry game with conditional actions given in Figure 6.2 and concluded that the weak notion of practicality is satisfied.*

We now revisit this game and are interested in analyzing whether the strong notion of practicality is satisfied as well. For this purpose, we need to make sure, that at every node, for every condition, practicality is satisfied. Beginning with the root, we observe that there are two conditions, namely, $p < 10 * f$ and $p \geq 10 * f$. The game restricted to the first condition was analyzed in Example 6.7, so we focus on the game restricted to the second condition, that is $\Gamma_{me\_ca_{|p\geq 10*f,\emptyset}}$.

Let us start with the leaves in $\Gamma_{me\_ca_{|p\geq 10*f,\emptyset}}$. Each leaf is practical so we set $\mathbb{U}((e_2, i_1)) = \{(\frac{p}{3}, \frac{2*p}{3})\}$, $\mathbb{U}((e_2, pw_1)) = \{(-a, -a)\}$, $\mathbb{U}((e_2, i_2)) = \{(\frac{2*p}{3}, \frac{p}{3})\}$ and $\mathbb{U}((e_2, pw_2)) = \{(\frac{p}{2}, \frac{p}{2})\}$. Moving upwards, we look at the subtree $\Gamma_{me\_ca_{|(e_2)}}$. We apply the adapted notion Definition 6.9 for strong practicality. Hence, we need to make sure that the subgame after (e_2) is practical in terms of the strong notion for each condition. Applying Definition 6.8, in particular item 1), we observe that $\frac{2*p}{3} \geq -a$ and $\frac{p}{2} \geq \frac{p}{3}$, so the honest behavior is also the one that is "greedy" for the current player, namely E. Hence, the subgame after (e_2) is practical in terms of the strong notion and we set $\mathbb{U}((e_2)) = \{(\frac{p}{3}, \frac{2*p}{3}), (\frac{p}{2}, \frac{p}{2})\}$. Moving up one more time, we apply Definition 6.8 and need to compare utilities in $\mathbb{U}((e_2))$ with $\mathbb{U}((o_2)) = \{(-\alpha, p)\}$ for the player M. We observe that both, $\frac{p}{3} \geq -\alpha$ and $\frac{p}{2} \geq -\alpha$ hold. Thus, $\Gamma_{me\_ca_{|p\geq 10*f,\emptyset}}$ is practical as well.

Finally, we can conclude that, the Market Entry game with conditional actions is practical in terms of the strong notion of this property.

Theorem 6.3 (Equivalence of Weak and Strong Notion of Security Properties of CEFGs)**.** *Let* Γ *be an EFG and* Γ^c *its equivalent CEFG, as in Theorem 6.1. Then for* $sp \in \{wi, weri, cr, pr\}$ *the following equivalence holds:*

$$sp_w(\Gamma^c) \Leftrightarrow sp_s(\Gamma^c),$$

where $sp_w(\Gamma^c)$ *is according to Definition 6.5, Definition 6.7 and Definition 6.9 and* $sp_s(\Gamma^c)$ *is according to Definition 6.10, Definition 6.11 and Definition 6.12.*

Proof. By observing Definition 6.10, Definition 6.11 and Definition 6.12 we can see that the weak and strong notion of a security property for a game involving conditional actions only differ in the quantifier used to stratify over the possible conditions $\mathcal{C}(h)$ after a history h. Consider the whole game, i.e. let $h = \emptyset$. Since Γ^c features only trivial conditions the only available condition at the root is `true`, i.e. $\mathcal{C}(\emptyset) = \{true\}$. Thus, the existential and the universal quantification over this singleton coincide and therefore the weak and the strong notion are equivalent. □

Corollary 6.1 (Equivalence of Strong Notion of Security Properties of CEFGs and Security Properties of EFGs)**.** *Let* Γ *be an EFG and* Γ^c *its equivalent CEFG, as in Theorem 6.1. Then for* $sp \in \{wi, weri, cr, pr\}$ *the following equivalence holds:*

$$sp(\Gamma) \Leftrightarrow sp_s(\Gamma^c),$$

where $sp(\Gamma)$ *is according to Definition 2.3–Definition 2.6, and* $sp_s(\Gamma^c)$ *is according to Definition 6.10, Definition 6.11 and Definition 6.12.*

Proof. Follows from Theorem 6.2 and Theorem 6.3. In more details, Theorem 6.2 shows that the fulfillment of a security property of a game with no conditional actions is equivalent to the fulfillment of the weak notion of this security property when extending this game to a game with only trivial conditions, whereas Theorem 6.3 shows that the weak and strong notions of a security property coincide for games with only trivial conditions. Consequently, we can conclude that the fulfillment of a security property of a game with no conditional actions is equivalent to the fulfillment of the strong notion of this security property when extending this game to a game with only trivial conditions. □

Lastly, we present one more corollary which states how (super)game security can be reduced to subgame security. This means, that a decision for the fulfillment of a security property in a (super)game can be made if some necessary information is provided for the subgames.

Corollary 6.2 (Compositional Property Analysis for Conditional Actions)**.** *Let* Γ *be a CEFG with honest history* h^**. Then, it can be computed compositionally whether the weak (strong) notion of a security property is satisfied. The only information needed of a subtree* $\Gamma_{|h}$*, to decide whether* Γ *satisfies the weak (strong) notion of a security property is:*

- *for weak(er) immunity: for which players $p \in N$ the subtree $\Gamma_{|h}$ is weak(er) immune in terms of the weak (strong) notion;*
- *for collusion resilience: against which player groups $S \subset N$ the subtree $\Gamma_{|h}$ is collusion resilient in terms of the weak (strong) notion;*
- *for practicality:*
  - *if h is along h^*: whether $h^*_{|h}$ is practical in $\Gamma_{|h}$ in terms of the weak (strong) notion;*
  - *if h is not along h^*: the set of all practical utilities of $\Gamma_{|h}$ in terms of the weak (strong) notion and the conditions which need to be satisfied (i.e. the world state) to obtain those utilities. A utility $u(t)$ after terminal history $t \in \mathscr{T}$ is practical in subgame $\Gamma_{|h}$ in terms of the weak (strong) notion iff t is practical in $\Gamma_{|h}$ in terms of the weak (strong) notion.*

Proof. This corollary is an analogous version of Theorem 4.3 and follows directly from the definitions of the security properties (Definition 6.5–Definition 6.12). □

6.3 Conditional Security

Having extended the notions of the security properties for games with conditional actions, we can proceed to define security for CEFGs. In particular, we define *weak* and *strong conditional security*. We note that reasoning in terms of the weak notion of the security properties necessitates the existence of consistent world states $m \in \mathcal{M}$ for which weak(er) immunity, collusion resilience and practicality are satisfied, where as the strong notions of the security properties require the properties to be satisfied for every consistent world state $m \in \mathcal{M}$.

Definition 6.13 (Weakly Secure CEFG)**.** *A CEFG Γ with honest behavior h^* is* weakly secure *if it is weak(er) immune, collusion resilient and practical in terms of the weak notions of these security properties.*

Definition 6.14 (Strongly Secure CEFG)**.** *A CEFG Γ with honest behavior h^* is* strongly secure *if it is weak(er) immune, collusion resilient and practical in terms of the strong notions of these security properties.*

Finally, we can observe how the weakly and strongly secure honest histories relate to each other, but also to the notion of a secure honest history for a game with no conditional actions.

Theorem 6.4 (Equivalence of Weak and Strong Conditional Security)**.** *Let $\Gamma = (N, G)$ be an EFG with honest history h^*. Furthermore, let Γ^c be the extension of Γ using only trivial conditions. Then the following hold:*

Algorithm 6.1: Function `ComputeWI_CA` for Conditional Weak Immunity.

input : game tree Γ, honest history h^*, set S containing initial constraints and current case, player group pg, weak notion of security properties weakCA.

output : $(\mathsf{result}, \mathsf{split})$, where result states whether Γ is weak immune for pg in terms of the weak or strong notion, depending on whether weakCA is true or not, given S, and split a crucial utility comparison we cannot decide.

```
if isLeaf(Γ) then
    if Check(S, GetUtility(Γ, pg) < 0) = unsat then
        return (true, null)
    end
    if Check(S, GetUtility(Γ, pg) ≥ 0) = unsat then
        return (false, null)
    end
    return (false, GetUtility(Γ, pg) ≥ 0)
end
if CurrentPlayer(Γ) ≠ pg then
    return ComputeWI_CA_differentPlayer(Γ, h*, S, pg, weakCA)
end
if AlongHonest(Γ, h*) then
    return ComputeWI_CA_alongHonest(Γ, h*, S, pg, weakCA)
end
return ComputeWI_CA_notAlongHonest(Γ, h*, S, pg, weakCA)
```

1) Γ *is secure iff* Γ^c *is weakly secure;*

2) Γ^c *is weakly secure iff it is strongly secure.*

Proof. From Theorem 6.1 we know that Γ^c can be obtained from Γ and the honest behavior remains the same.

The proof for 2) follows from Definition 6.13 and Definition 6.14 by taking into consideration that the weak and strong notion of the security properties coincide for Γ^c (Theorem 6.3).

Similarly, 1) follows from Theorem 6.2. □

6.4 Automation of Conditional Security

In this section, we discuss the automation of security analysis for CEFGs. The setting is very similar to the one presented in Section 5.1: we compute the protocol's security by

Algorithm 6.2: Function `ComputeWI_CA_differentPlayer` for Conditional Weak Immunity.

input : game tree Γ, honest history h^*, set S containing initial constraints and current case, player group pg, weak notion of security properties weakCA.

output : (result, split), where result states whether Γ is weak immune for pg, in terms of the weak or strong notion, depending on whether weakCA is true or not, given S, and split a crucial utility comparison we cannot decide.

```
1  compatibleConditionExists ← false
2  newsplit ← null
3  for c ∈ Conditions(Γ) do
4  |  compatible ← CheckCompatibility(S, c)
5  |  if compatible = false then
6  |  |  continue
7  |  end
8  |  compatibleConditionExists ← true
9  |  S' ← S ∪ c
10 |  nonSecureChoiceFound ← false
11 |  for a ∈ Actions(Γ) do
12 |  |  (result, split) ← ComputeWI_CA(Γ_|(a),c, h*, S', pg, weakCA)
13 |  |  if split ≠ null then
14 |  |  |  newsplit ← split
15 |  |  end
16 |  |  if result = false ∧ ¬weakCA then
17 |  |  |  return (result, split)
18 |  |  end
19 |  |  if result = false ∧ weakCA then
20 |  |  |  nonSecureChoiceFound ← true
21 |  |  |  break
22 |  |  end
23 |  end
24 |  if weakCA ∧ ¬nonSecureChoiceFound then
25 |  |  return (true, null)
26 |  end
27 end
28 if weakCA ∧ compatibleConditionExists then
29 |  return (false, newsplit)
30 end
31 return (true, null)
```

Algorithm 6.3: Function `ComputeWI_CA_alongHonest` for Conditional Weak Immunity.

input : game tree Γ, honest history h^*, set S containing initial constraints and current case, player group pg, weak notion of security properties weakCA.

output : (result, split), where result states whether Γ is weak immune for pg, in terms of the weak or strong notion, depending on whether weakCA is true or not, given S, and split a crucial utility comparison we cannot decide.

```
1  compatibleConditionExists ← false
2  newsplit ← null
3  for c ∈ Conditions(Γ) do
4  |  compatible ← CheckCompatibility(S, c)
5  |  if compatible = false then
6  |  |  continue
7  |  end
8  |  compatibleConditionExists ← true
9  |  S' ← S ∪ c
10 |  a* ← HonestAction(Γ, h*)
11 |  (result, split) ← ComputeWI_CA(Γ_{|(a*),c}, h*, S', pg, weakCA)
12 |  if split ≠ null then
13 |  |  newsplit ← split
14 |  end
15 |  if result = false ∧ ¬weakCA then
16 |  |  return (result, split)
17 |  end
18 |  if result = true ∧ weakCA then
19 |  |  return (true, null)
20 |  end
21 end
22 if weakCA ∧ compatibleConditionExists then
23 |  return (false, newsplit)
24 end
25 return (true, null)
```

analyzing one game-theoretic property and one honest history at a time, considering all possible total orders.

For analyzing games involving conditional actions, we are able to reuse Algorithm 5.1, introduced previously in Section 5.1. Note that the actual implementation of the algorithm has been adapted in the following manner:

Algorithm 6.4: Function `ComputeWI_CA_notAlongHonest` for Conditional Weak Immunity.

input : game tree Γ, honest history h^*, set S containing initial constraints and current case, player group pg, weak notion of security properties weakCA.

output : (result, split), where result states whether Γ is weak immune for pg, in terms of the weak or strong notion, depending on whether weakCA is true or not, given S, and split a crucial utility comparison we cannot decide.

```
1  newsplit ← null
2  compatibleConditionExists ← false
3  for c ∈ Conditions(Γ) do
4  |  compatible ← CheckCompatibility(S, c)
5  |  if compatible = false then
6  |  |  continue
7  |  end
8  |  compatibleConditionExists ← true
9  |  S' ← S ∪ c
10 |  secureChoiceFound ← false
11 |  for a ∈ Actions(Γ) do
12 |  |  (result, split) ← ComputeWI_CA(Γ_|(a),c, h*, S', pg, weakCA)
13 |  |  if result = true ∧ weakCA then
14 |  |  |  return (true, null)
15 |  |  end
16 |  |  if result = true ∧ ¬weakCA then
17 |  |  |  secureChoiceFound ← true
18 |  |  end
19 |  |  if result = false ∧ split ≠ null then
20 |  |  |  newsplit ← split
21 |  |  end
22 |  end
23 |  if ¬weakCA ∧ ¬secureChoiceFound then
24 |  |  return (false, newsplit)
25 |  end
26 end
27 if weakCA ∧ compatibleConditionExists then
28 |  return (false, newsplit)
29 end
30 return (true, null)
```

- The honest history h^* given as input has a new format, as described in Section 6.1.
- Additionally, we give as an input a boolean `weakCA`. If its value is `true`, we analyze the underlying game in terms of the weak notion of the security properties, otherwise in terms of the strong notion.
- The function `ComputeSP` in line 6, which previously stood for `ComputeWI`, `ComputeCR` and `ComputePR` now stands for `ComputeWI_CA`(Algorithm 6.1), `ComputeCR_CA` and `ComputePR_CA`.

Algorithm 6.1: Function `ComputeWI_CA`. The function `ComputeWI_CA` of Algorithm 6.1 is initially called with the entire game tree Γ from `SatisfiesProperty` of Algorithm 5.1. We then proceed by checking weak immunity recursively for pg, which is one player.

In a leaf, the algorithm works exactly as Algorithm 5.2. Otherwise, it is checked whether the player pg is the current player and whether we are along the honest history (recall Definition 6.4). One of the functions `ComputeWI_CA_differentPlayer`(Algorithm 6.2), `ComputeWI_CA_alongHonest`(Algorithm 6.3) or `ComputeWI_CA_notAlongHonest` (Algorithm 6.4) is called accordingly.

Algorithm 6.2: Function `ComputeWI_CA_differentPlayer`. The function `ComputeWI_CA_differentPlayer` of Algorithm 6.2 is called whenever the current player is not the one that we analyze. Recall that according to Definition 6.4, a subgame $\Gamma_{|h,c}$ after history h restricted to a condition c is weak immune for pg in terms of the weak(strong) notion $wi_{w(s),p}(\Gamma_{|h,c})$ iff every action that the current player $P(h)$ can take leads to a subgame that is weak immune for the player we are checking in terms of the weak(strong) notion.

In lines 3–27 of Algorithm 6.2 we iterate over the possible conditions and check each of them. For the weak notion of the security properties, we need to make sure that there is one condition where all actions lead to weak immune trees, whereas for the strong notion of the security properties we need to make sure that this holds for all conditions.

It is important to note that we only consider and analyze conditions c that are compatible with our set S of initial constraints and current case, i.e. $\mathsf{S} \cup \{c\}$ has to be satisfiable (lines 4–7). Moreover, we add the currently considered condition c to the set of initial constraints and current case as it is relevant for the analysis of the subtree restricted to it (line 9).

Functions `ComputeWI_CA_alongHonest` and `ComputeWI_CA_notAlongHonest` presented in Algorithm 6.3 and Algorithm 6.4, respectively, follow the same reasoning.

The algorithms for other security properties can be adapted analogously. Note that for the property practicality we previously propagated the practical utilities of a node, whereas now we additionally need to keep track of the sets of conditions that need

to be satisfied for them. Moreover, when comparing two utilities to determine which one is practical, we need to ensure that we only compare utilities whose corresponding conditions are not incompatible (i.e. unsatisfiable).

Next, we present two lemmas needed to show the soundness and completeness of our approach in Theorem 6.5.

Lemma 6.1. *Let* Π *be an input instance,* h^* *an honest history, sp a security property,* weakCA *a boolean variable determining whether to choose the weak or strong notion of the conditional security properties,* S *the set of initial constraints* C *and a case* case*. Moreover, let* $\vec{x} = (x_1, \ldots, x_l)$ *be the real/infinitesimal variables occurring in the utility terms. If there exists a player group* pg *such that*

$$\texttt{ComputeSP}(\Pi, h^*, \mathsf{S}, sp, \mathsf{pg}, \mathsf{weakCA}) = (\texttt{false}, \mathsf{null}) \ ,$$

then

$$\forall \vec{x}.\ \forall c \in C \cup \texttt{case}.\ c[\vec{x}] \rightarrow \neg sp_w(\Gamma, h^*)[\vec{x}]$$

if weakCA $=$ `true` *and similarly,*

$$\forall \vec{x}.\ \forall c \in C \cup \texttt{case}.\ c[\vec{x}] \rightarrow \neg sp_s(\Gamma, h^*)[\vec{x}]$$

if weakCA $=$ `false`*.*

Proof. We prove the lemma for the weak notion of the security properties. The strong notion can be proven analogously.

Let weakCA $=$ `true`. We prove the lemma for the individual properties and the algorithms used to analyze them, starting with the property weak immunity. Let Π be an arbitrary but fixed input instance, h^* the honest behavior and S the set of initial constraints C and a case case. There exists a player group pg such that `ComputeSP` returns (`false`, null), iff `ComputeWI_CA` returns (`false`, null) for pg. We fix a player group pg and assume `ComputeWI_CA` returns (`false`, null). To show $\forall \vec{x}.\ \forall c \in C \cup \texttt{case}.\ c[\vec{x}] \rightarrow \neg sp_w(\Gamma, h^*)[\vec{x}]$, we proceed by structural induction on Γ. For the base case, assume Γ is a leaf. Observing Algorithm 6.1, in particular lines 1–9, one can see that (`false`, null) is returned if $C \cup \texttt{case} \cup u_{\mathsf{pg}} < 0$ is satisfiable but $C \cup \texttt{case} \cup u_{\mathsf{pg}} \geq 0$ is unsatisfiable, where u is the utility of the leaf. Consequently, for all $\vec{x}$ that satisfy $C \cup \texttt{case}$ the inequality $u_{\mathsf{pg}} < 0$ holds. Therefore, $\neg sp_w(\Gamma, h^*)$ for all such $\vec{x}$.

Next, in the induction step, Γ is an inner node (a branch) and by our induction hypothesis all subtrees of Γ satisfy the property. We proceed by case distinction:

- Case 1: Assume pg is not the current player in Γ. Then (`false`, null) was returned by `ComputeWI_CA_differentPlayer` in line 29. Hence, there is no condition where the subgame after each action is weak immune. Consider an arbitrary

condition. Applying the induction hypothesis, we know that for all $\vec{x}$ satisfying the constraints in S there is an action a such that $\Gamma_{|(a)}$ is not weak(er) immune for pg. Let us now fix such an $\vec{x}$ arbitrarily. Following Definition 6.4, we know that for values $\vec{x}$ the game Γ restricted to the fixed condition is not weak(er) immune for pg. Since $\vec{x}$ was chosen arbitrarily, Γ restricted to the fixed condition is not weak(er) immune for pg for all $\vec{x}$ that satisfy the constraints in S. As the condition was fixed arbitrarily, this holds for every condition and the whole game is not weak immune in terms of the weak notion for all $\vec{x}$ that satisfy the constraints in S, Definition 6.5.

- Case 2: Assume pg is the current player and Γ is along h^* and `ComputeWI_CA` returned $(\texttt{false}, \mathsf{null})$. Observing Algorithm 6.1 this can only happen in line 14, i.e. when $(\texttt{false}, \mathsf{null})$ is returned by `ComputeWI_CA_alongHonest` in line 23. By induction hypothesis, we know that for every condition, for all $\vec{x}$ satisfying the constraints in S the subgame $\Gamma_{|(a^*)}$ is not weak(er) immune for pg. By Definition 6.4 it follows that also Γ is not weak(er) immune for pg, for all $\vec{x}$ satisfying the constraints in S for each possible condition. Hence, Γ is not weak(er) immune in terms of the weak notion for pg, for all $\vec{x}$ satisfying the constraints in S following Definition 6.5.

- Case 3: Lastly, let pg be the current player and Γ is not along h^* and `ComputeWI_CA` returned $(\texttt{false}, \mathsf{null})$. Observing Algorithm 6.1 this can only happen in line 16, i.e. when $(\texttt{false}, \mathsf{null})$ is returned by `ComputeWI_CA_alongHonest` in line 28. This means that in each condition, all children are not weak immune in terms of the weak notion for all $\vec{x}$ satisfying the constraints in S. Let us fix an arbitrary condition. Applying the induction hypothesis and Definition 6.4, it follows that also Γ is not weak(er) immune for pg for all $\vec{x}$ satisfying $C \cup \texttt{case}$ in this condition. As the condition was chosen arbitrarily, this holds for all conditions. Hence, Γ is not weak(er) immune in terms of the weak notion for pg, for all $\vec{x}$ satisfying the constraints in S following Definition 6.5.

For the remaining properties, as well as for the case when $\mathsf{weakCA} = \texttt{false}$, it can be shown analogously that the property holds. Note that for practicality, `ComputeSP` can return $(\texttt{false}, \mathsf{null})$ iff the honest utilities are not practical, implying that Γ is along the honest history. For the setting of games with conditional actions, the proof from [5] applies (with slight extension to incorporate conditions and accommodate the strong and weak notion of practicality). □

Lemma 6.2. *Let* Π *be an input instance,* h^* *an honest history, sp a security property,* weakCA *a boolean variable determining whether to choose the weak or strong notion of the conditional security properties,* S *the set of initial constraints* C *and a case* `case`*. Moreover, let* $\vec{x} = (x_1, \ldots, x_l)$ *be the real/infinitesimal variables occurring in the utility terms. If for all player groups* pg

$$\texttt{ComputeSP}(\Pi, h^*, \mathsf{S}, sp, \mathsf{pg}, \mathsf{weakCA}) = (\texttt{true}, \mathsf{split}) \ ,$$

independent of what split *is, then*

$$\forall \vec{x}.\ \forall c \in C \cup \mathsf{case}.\ c[\vec{x}] \rightarrow sp_w(\Gamma, h^*)[\vec{x}]$$

if weakCA = `true` *and similarly,*

$$\forall \vec{x}.\ \forall c \in C \cup \mathsf{case}.\ c[\vec{x}] \rightarrow sp_s(\Gamma, h^*)[\vec{x}]$$

if weakCA = `false`.

Proof. We prove the lemma for the weak notion of the security properties. The strong notion can be proven analogously.

Let weakCA = `true`. Let Π be an arbitrary but fixed input instance, h^* the honest behavior and S the set of initial constraints C and a case case. Further, let pg be a fixed player group and assume sp is wi. We assume `ComputeSP`$(\Pi, h^*, \mathsf{S}, sp, \mathsf{pg}, \mathsf{weakCA})$ returns (`true`, split). We once again prove the lemma by structural induction on the game tree Γ. In the base case, Γ is a leaf and the return value is (`true`, split). According to Algorithm 6.1, this can only happen if $C \cup$ `case` together with the negation of the property inequality of the leaf utility u is unsatisfiable, that is, only if $C \cup \mathsf{case} \cup \{u_{\mathsf{pg}} < 0\}$ is unsat. According to Definition 6.4, this implies that for all $\vec{x}$ satisfying $C \cup$ `case` the game Γ satisfies the security property for player pg.

Next, in the induction step, Γ is an inner node (a branch) and by our induction hypothesis all subtrees of Γ satisfy the property. We proceed by case distinction:

- Case 1: Assume pg is not the current player in Γ. Then (`true`, split) was returned by `ComputeWI_CA_differentPlayer` in line 25 or line 31. If this was returned in line 25, we know we have found a condition c such that for the subgame after each action a available in this condition `ComputeWI_CA`$(\Gamma_{(a),c}, h^*, \mathsf{S}, sp, \mathsf{pg}, \mathsf{weakCA})$ returns (`true`, split). Applying the induction hypothesis, we know that for all $\vec{x}$ satisfying the constraints in S and for every available action a the game $\Gamma_{|(a)}$ is weak immune for pg. Let us now fix such an $\vec{x}$ arbitrarily. Following Definition 6.4, we know that for values $\vec{x}$ the game Γ restricted to the fixed condition is weak immune for pg. Since $\vec{x}$ was chosen arbitrarily, Γ restricted to the fixed condition is weak immune for pg for all $\vec{x}$ that satisfy the constraints in S. Finally, the fixed condition can be used as a witness to conclude that the whole game is weak immune in terms of the weak notion for all $\vec{x}$ that satisfy the constraints in S, Definition 6.5.

  In the following, we consider the case when the return happened in line 31. Recall that according to R2 from Definition 6.1, the conditions at a node are collectively exhaustive i.e. span the whole subtree. That is, if the return happened in line 31, it has to be the case that the constraints in S are unsatisfiable, and the property trivially holds, as the left side of the implication is false for any $\vec{x}$.

- Case 2: Assume pg is the current player and Γ is along h^* and `ComputeWI_CA` returned (`true`, split). Observing Algorithm 6.1 this can only happen in line 14, i.e. when (`true`, split) is returned by `ComputeWI_CA_alongHonest` in line 19 or line 25. Let us first consider the case when it is returned in line 19. This means that we have found a condition c such that for the honest action a^* available in condition c it holds that `ComputeWI_CA`$(\Gamma_{(a^*),c}, h^*, \mathsf{S}, sp, \mathsf{pg}, \mathsf{weakCA})$ returns (`true`, split). By induction hypothesis, we know that for all $\vec{x}$ satisfying the constraints in S the subgame $\Gamma_{|(a^*)}$ is weak immune for pg. By Definition 6.4 it follows that also Γ is weak immune for pg, for all $\vec{x}$ satisfying the constraints in S for condition c. Hence, Γ is weak immune in terms of the weak notion for pg, for all $\vec{x}$ satisfying the constraints in S following Definition 6.5. The argument for the case when the return is in line 25 is identical to the argument for line 31 in Case 1 above.
- Case 3: Lastly, let pg be the current player and Γ is not along h^* and `ComputeWI_CA` returned (`true`, split). Observing Algorithm 6.1 this can only happen in line 16, i.e. when (`true`, split) is returned by `ComputeWI_CA_alongHonest` in line 14 or line 30. Assume the return happens in line 14. This means that there is a condition c, such that for an action a available in this condition `ComputeWI_CA`$(\Gamma_{(a),c}, h^*, \mathsf{S}, sp, \mathsf{pg}, \mathsf{weakCA})$ returns (`true`, split). Further from the induction hypothesis we can infer that for all $\vec{x}$ satisfying the constraints in S the subgame $\Gamma_{|(a)}$ is weak immune for pg. Applying Definition 6.4, it follows that also Γ is weak immune for pg for all $\vec{x}$ satisfying $C \cup$ `case` in this condition. Hence, Γ is weak immune in terms of the weak notion for pg, for all $\vec{x}$ satisfying the constraints in S following Definition 6.5. The argument for the case when the return is in line 30 is identical to the argument for line 31 in Case 1 above.

The argument above shows that for all $\vec{x}$ that satisfy the constraints in S, Γ with honest history h^* satisfies sp for player pg. The player pg was fixed arbitrarily and thus, this holds for all players. Moreover, the quantification over the player group and the $\vec{x}$ quantification are independent, so their order can be switched. Finally, by Definition 6.4, we can conclude that for all $\vec{x}$ that satisfy the constraints in S, the $sp(\Gamma, h^*)$ holds.

For $sp = weri$ and $sp = cr$ the reasoning is the same. On the other hand, for $sp = pr$, we can follow the argument for the algorithm for games with no conditional actions, presented in [5]. □

Theorem 6.5 (Correctness of Automation of Conditional Actions)**.** *The method for computing game-theoretic security of an input instance* Π *for honest history* h^* *in terms of weak or strong security depending on parameter* weakCA *is sound and complete. That is,* `SatisfiesProperty` $(\Pi, h^*, sp, \emptyset,$ weakCA$)$ = `true` *iff* Π *with honest history* h^* *satisfies the property sp in terms of the weak or strong notion (depending on whether* weakCA *is true or false). Otherwise, it returns* `false`*.*

Proof. We prove the theorem for the weak notion of the security properties following the approach presented in [5]; the proof of the strong notion can be done analogously.

We first prove direction "⇐" by contraposition. For this purpose, assume that the return value of `SatisfiesProperty` $(\Pi, h^*, sp, \emptyset, \mathsf{weakCA})$ is `false`. We need to show that Π with honest history h^* does not satisfy security property sp in terms of the weak notion. From the assumption and Algorithm 5.1, we observe that due to the return value being `false`, there has to exist an ordering of utility terms `case` such that `SatisfiesProperty` $(\Pi, h^*, sp, \emptyset, \mathsf{weakCA}) =$ `false`. Further, this means that there exists a player (group) pg such that `ComputeSP` $(\Pi, h^*, \mathsf{S}, sp, pg, \mathsf{weakCA})$ returns $(\texttt{false}, \mathsf{null})$, where S is composed of all constraints from C and `case`.

We now apply Lemma 6.1 and infer that for all values of $\vec{x}$ that satisfy the (satisfiable) set of constraints in S, the game Γ (of Π) with honest history h^* violates security property sp for player group pg in terms of the weak notion. The set `case` can be extended to a total order $\preceq$ on the utility terms T_u. Since all $\vec{x}$ that satisfy $C \cup \preceq$ also satisfy $C \cup \mathsf{case}$, it follows that for all $\vec{x}$ that satisfy $C \cup \preceq$ security property sp does not hold for player group pg in terms of the weak notion, which means that sp does not hold. Recall Theorem 3.1 which applies to the setting of CEFGs as well and allows us to infer this claim. Hence, the input instance Π violates the security property sp in terms of the weak notion.

Next, we prove direction "⇒". Assume that the return value of `SatisfiesProperty` $(\Pi, h^*, sp, \emptyset, \mathsf{weakCA})$ is `true`. Observing Algorithm 5.1, we see that this implies that all final cases returned `true`. Moreover, we also know that all considered cases are pair-wise disjoint, and their disjunction is a tautology. Let `case` be one such arbitrary, but fixed case. Then, for all player groups pg it has to hold `ComputeSP` $(\Pi, h^*, \mathsf{S}, sp, pg, \mathsf{weakCA})$ returns $(\texttt{true}, \mathsf{split})$, where split is irrelevant and S is composed of all constraints from C and `case`.

We can apply Lemma 6.2 and infer that for all $\vec{x}$ that satisfy C and `case` security property sp holds in terms of the weak notion in game Γ with honest history h^*. Similarly as before, this implies that for all total orders $\preceq$ extending `case`, the security property holds for all pg. Further, the disjunction of all total orders that extend `case` is equivalent to `case` itself. Thus, this holds for all considered cases and those cases span the whole universe, so it follows that for all total orders $\preceq$ holds that all $\vec{x}$ that satisfy $C \cup \preceq$ satisfy the security property for all player groups. Hence, the input instance Π with honest history h^* satisfies the security property in terms of the weak notion.

At last, we prove termination by showing that `SatisfiesProperty` always returns either `true` or `false`. This holds because the `Compute<SP>_CA` functions traverse the finite game tree only once, and all SMT queries are decidable (as they involve unquantified non-linear real arithmetic). Moreover, function `SatisfiesProperty` splits only on utility comparisons of players/groups of players, which are finitely many. Thus, the algorithm terminates. □

6.5 Conditional Counterexamples

As discussed previously in Section 3.3, if a security property (wi, weri, cr, pr) is not satisfied, we are interested in finding out why *not*. In the sequel, we extend the notions of compositional counterexamples presented in Section 3.3 to conditional counterexamples for games with conditional actions. Specifically, we introduce *weak* and *strong conditional counterexamples*, which refer to violation of weak and strong conditional security, respectively. Note that we use the terms *counterexamples* and *conditional counterexamples* interchangeably when dealing with CEFGs.

Conditional Counterexamples to Weak(er) Immunity. For the property weak(er) immunity, a counterexample in terms of the weak notion is a harmed honest player p and a partial strategy of the other players $N - p$ such that no matter what honest actions p chooses, they cannot avoid receiving a (real-valued) negative utility in any consistent world state. Recall that the weak notion of weak(er) immunity is satisfied if for each player there is a consistent world state such that the game in this world state is weak(er) immune for the fixed player.

Definition 6.15 (Weak Counterexamples to Weak(er) Immunity)**.** *Let Γ be a CEFG with honest behavior h^*. A* weak counterexample to h^* being weak(er) immune *is a player p together with a partial strategy s_{N-p} such that for any consistent world state $m \in \mathcal{M}$, s_{N-p} extended by any strategy σ_p of player p who follows the honest history h^*, yields a terminal history $H_m(s_{N-p}, \sigma_p) = t_{\sigma_p}$ with $u_p(t_{\sigma_p}) < 0$ (resp. for weaker immunity* $\mathsf{real}(u_p(t_{\sigma_p})) < 0$*) and it is* minimal *with that property.*

The *minimality* of the partial strategy s_{N-p} follows the definition from Section 3.3.

Example 6.10 (Weak Counterexamples for Weaker Immunity)**.** *Consider the Market Entry game with conditional actions given in Figure 6.1, but let the utilities after history (e_1) and (o_2) be $(-c, -p)$ and $(-\alpha, -p)$, respectively (the profit p is negated). A weak counterexample for the weaker immunity of the modified game is then player E together with a partial strategy s_M for M such that $s_{M_{p<10*c}}(\emptyset) = e_1$ and $s_{M_{p\geq 10*c}}(\emptyset) = o_2$. We observe that no matter what strategy E chooses, the obtained utility will be negative, that is $-p$.*

A counterexample for weak(er) immunity in terms of strong conditional security, is a harmed honest player p, a world state m and a partial strategy of the other players $N - p$ such that no matter what honest actions p chooses, they cannot avoid receiving a real-valued negative utility in the world state m. Recall that the strong notion of weak(er) immunity requires that for every player every world state is weak immune.

Definition 6.16 (Strong Counterexamples to Weak(er) Immunity)**.** *Let Γ be a CEFG with honest behavior h^*. A* strong counterexample to h^* being weak(er) immune *is a world state $m \in \mathcal{M}$ and a player p together with a partial strategy s_{N-p} such that*

s_{N-p} extended by any strategy σ_p of player p who follows the honest history h^*, yields a terminal history $H_m(s_{N-p}, \sigma_p) = t_{\sigma_p}$ with $u_p(t_{\sigma_p}) < 0$ (resp. for weaker immunity $\mathsf{real}(u_p(t_{\sigma_p})) < 0$) and it is minimal *with that property.*

Example 6.11 (Strong Counterexamples for Weaker Immunity)**.** *We argued in Example 6.8 that the Market Entry game with conditional actions given in Figure 6.1 is not weak immune (for player M) in terms of the strong notion. A strong counterexample, as defined in Definition 6.16, is player M, world state $\{p \geq 10 * c, d < 0.6\}$ and a partial strategy s_E for E such that $s_{E_{p \geq 10*c}}(e_2) = pw_1$. Since the condition $p \geq 10 * f$ is true at the root and M is considered an honest player, M takes action e_2 in any strategy σ_M (which follows the honest history), after which E can take action pw_1 according to the fixed s_E and M receives a negative utility in the case of the terminal history yielded in the fixed world state, that is history (e_2, pw_1).*

Conditional Counterexamples to Collusion Resilience. For the property collusion resilience, a counterexample in terms of weak conditional security consists of a group of deviating players S and their partial strategy $s_S \in \mathscr{S}$, such that the joint utility of S is better than the honest utility in every consistent world state, no matter how the other players $N - S$ react, while still following the honest history. We recall that the weak notion of collusion resilience is satisfied if for each possible colluding group there exist a consistent world state such that the game in this world state is collusion resilient against the fixed group.

Definition 6.17 (Weak Counterexamples to Collusion Resilience)**.** *Let Γ be a CEFG with honest behavior h^*. A* weak counterexample to h^* being collusion resilient *is a set of deviating players S together with their partial strategy s_S such that in any consistent world state $m \in \mathcal{M}$, s_S extended by any strategy σ_{N-S} of players $N - S$, which follows the honest history h^*, yields a terminal history $H_m(\sigma_{N-S}, s_S) = t_{\sigma_{N-S}}$ with*

$$\sum_{p \in S} u_p(t_{\sigma_{N-S}}) > \sum_{p \in S} u_p(H_m(h^*))$$

and it is minimal with that property. The minimality of s_S is similar to the minimality of the partial strategy for weak(er) immunity.

Example 6.12 (Weak Counterexamples for Collusion Resilience)**.** *Consider the Market Entry game with conditional actions given in Figure 6.1, but let the utilities after history (e_1) and (o_2) be (p, p) and (p, p), respectively ($-f$ and α have been replaced by p). A weak counterexample for the collusion resilience of the modified game is then the singleton colluding group $S = \{M\}$ and a partial strategy s_S for S such that $s_{M_{p<10*c}}(\emptyset) = e_1$ and $s_{M_{p \geq 10*c}}(\emptyset) = o_2$. We can observe that in any world state $m \in \mathcal{M}$, s_S extended by any strategy σ_{N-S} (for E), which follows h^*, yields a terminal history which has a strictly better utility for S then the one corresponding to h^*. We discuss this in more detail for every consistent world state in the following.*

Let the world state be such that it corresponds to $\{ p < 10 * f, d < 0.6 \}$ *or* $\{ p < 10 * f, d \geq 0.6 \}$. *According to* s_S, M *chooses action* e_1 *and the game ends with utility* p *for* M. *The honest utility corresponding to both world states (that we consider at once in this example) is the leaf after action* o_1, *that is,* 0 *for* M. *Recall that we have the additional constraint* $p > 0$ *in the definition of the Market Entry game with conditional actions. Thus, the sum of the utilities of the players in the colluding group* S *(in this case only the utility of* M, *because* S *is a singleton) is strictly greater than the corresponding sum if the strategy is extended by any strategy for* E *that follows* h^*.

Now let us consider world states $\{ p \geq 10 * f, d < 0.6 \}$ *and* $\{ p \geq 10 * f, d \geq 0.6 \}$. *According to* s_S, M *chooses action* o_2 *and the game ends with utility* p *for* M, *independent on the strategy of* E, *whereas the utility that* M *would obtain corresponding to* h^* *is either* $\frac{p}{3}$ *or* $\frac{p}{2}$, *depending on the world state. We observe that* $p > \frac{p}{3}$ *and* $p > \frac{p}{2}$.

For the property collusion resilience, a counterexample in terms of strong conditional security consists of a group of deviating players S, a world state m and their partial strategy $s_S \in \mathscr{S}$, such that in world state m the joint utility of S is better than the honest utility, no matter how the other players $N - S$ react, while still following the honest history. Recall that the strong notion of collusion resilience requires that for every player group every world state is collusion resilient against it.

Definition 6.18 (Strong Counterexamples to Collusion Resilience)**.** *Let* Γ *be a CEFG with honest behavior* h^*. *A* strong counterexample to h^* being collusion resilient *is a consistent world state* $m \in \mathcal{M}$ *and a set of deviating players* S *together with their partial strategy* s_S *such that in world state* m, s_S *extended by any strategy* σ_{N-S} *of players* $N - S$, *which follows the honest history* h^*, *yields a terminal history* $H_m(\sigma_{N-S}, s_S) = t_{\sigma_{N-S}}$ *with*

$$\sum_{p \in S} u_p(t_{\sigma_{N-S}}) > \sum_{p \in S} u_p(H_m(h^*))$$

and it is minimal with that property. The minimality of s_S *is similar to the minimality of the partial strategy for weak(er) immunity.*

Example 6.13 (Strong Counterexamples for Collusion Resilience)**.** *Consider the Market Entry game with conditional actions given in Figure 6.1, but let the utilities after history* (e_1) *and* (o_2) *be* (p, p) *and* (p, p), *respectively* ($-f$ *and* α *have been replaced by* p). *A strong counterexample for the collusion resilience of the modified game is then the singleton colluding group* $S = \{M\}$, *a partial strategy* s_S *for* S *such that* $s_{M_{p<10*c}}(\emptyset) = e_1$ *and* $s_{M_{p \geq 10*c}}(\emptyset) = o_2$ *and the world state* $\{ p < 10 * f, d < 0.6 \}$. *In the picked world state,* M *chooses action* e_1 *according to* s_S *and the game ends with utility* p *for* M, *independent on how* s_S *is extended by a strategy for* E. *The honest utility corresponding to the fixed world state is the leaf after action* o_1, *that is,* 0 *for* M *and we know that* $p > 0$ *holds according to the definition of the Market Entry game with conditional actions.*

Counterexamples to Practicality. Counterexamples to practicality of the honest history h^* have to provide a reason why a rational player would not follow h^*. By definition, the weak notion of practicality holds if there is a world state such that at any point at the game, the subgame corresponding the condition that is satisfied in the world state is practical for the current player. Hence, for a counterexample, we need to show that for every possible state m, there is a prefix h of h^* compliant with this state, such that there is an action a after h in the condition c satisfied by m which promises the current player $P(h)$ a strictly better utility than h^*. Moreover, in the subgame $\Gamma_{|(h,a),c}$ after (h, a) all practical utilities complying with m have to be better for $P(h)$, otherwise other players could choose actions in $\Gamma_{|(h,a),c}$ that would disincentivize $P(h)$ to deviate from h^*.

Definition 6.19 (Weak Counterexamples to Practicality)**.** *For a CEFG Γ with honest behavior h^*, a* weak counterexample to practicality of h^* *is defined as a mapping from the consistent world states in $\mathcal{M}$ to $\mathscr{H}$,*

$$m \mapsto (h, a),$$

where

- *h along h^*, all conditions along h satisfied in m;*
- *$a \in A_c(h)$ where $c = m(h)$;*
- *$\forall t \in \mathscr{T}_{|(h,a),c}.\, t$ weakly practical in $\Gamma_{|(h,a),c} \rightarrow\ u_{P(h)}(H_m(h^*)) < u_{P(h)}((h, a, t))$.*

Example 6.14 (Weak Counterexamples to Practicality)**.** *Consider the Market Entry game with conditional actions given in Figure 6.1, but let the utilities after history (e_1) and (o_2) be (p, p) and (p, p), respectively ($-f$ and α have been replaced by p). We construct a weak counterexample for the practicality of the modified game as follows.*

- *For every world state $m \in \mathcal{M}$, pick the empty history, i.e. $h = \emptyset$. Note that h is trivially a prefix of h^* and is trivially compliant to any m.*
- *For world states $\{\, p < 10 * f,\, d < 0.6 \,\}$ and $\{\, p < 10 * f,\, d < 0.6 \,\}$ pick action e_1, and for world states $\{\, p \geq 10 * f,\, d < 0.6 \,\}$ and $\{\, p \geq 10 * f,\, d < 0.6 \,\}$ pick action o_2. Both actions are compatible (that is available) with the conditions in the corresponding world states.*

Let us first consider $\{\, p < 10 * f,\, d < 0.6 \,\}$ and $\{\, p < 10 * f,\, d < 0.6 \,\}$. For both world states we picked $h = \emptyset$ and $a = o_2$. We observe that the subgame $\Gamma_{me\_ca_{|(o_2)}}$ is a leaf and thus $\mathbb{U}((o_2)) = \{(p, p)\}$. We analyzed the game after action e_2 in Example 6.9 and concluded that the honest utilities are the practical ones. Now we observe that the utility after o_2, namely p, is strictly better than both honest utilities, so the conditions for a counterexample are satisfied.

Next, consider world states $\{ p \geq 10 * f,\, d < 0.6 \}$ *and* $\{ p \geq 10 * f,\, d < 0.6 \}$*. For both world states we picked* $h = \emptyset$ *and* $a = e_1$*. We can easiliy observe that the conditions for a counterexample are satisfied here as well, since* $p > 0$*.*

Recall that the strong notion of practicality requires that all consistent world states are practical for the current player. In this sense, a counterexample is one particular world state, where this is violated.

Definition 6.20 (Strong Counterexamples to Practicality)**.** *For a CEFG* Γ *with honest behavior* h^**, a* strong counterexample to practicality of h^* *is defined as a tuple:*

$$(m, h, a),$$

where

- $m \in \mathcal{M}$*,* m *is consistent;*
- $h \in \mathscr{H}$*,* h *along* h^**, all conditions along* h *satisfied in* m*;*
- $a \in A_c(h)$ *where* $c = m(h)$*;*
- $\forall t \in \mathscr{T}_{|(h,a),c}.\, t$ *weakly practical in* $\Gamma_{|(h,a),c} \rightarrow\ u_{P(h)}(H_m(h^*)) < u_{P(h)}((h, a, t))$*.*

Example 6.15 (Strong Counterexamples to Practicality)**.** *Consider the Market Entry game with conditional actions given in Figure 6.1, but let the utilities after history* (e_1) *and* (o_2) *be* (p, p) *and* (p, p)*, respectively (*$-f$ *and* α *have been replaced by* p*). We construct a strong counterexample for the practicality of the modified game as follows.*

We fix world state $\{ p < 10 * f,\, d < 0.6 \}$*, pick the empty history, i.e.* $h = \emptyset$ *and action* e_1*. Note that* h *is trivially a prefix of* h^* *and is trivially compliant to any* m*. In the fixed world state the honest utility for* M *is 0, whereas if* M *behaves "greedily" and picks action* e_1*,* M *obtains utility* p*, which is strictly greater than the honest one.*

In the following, we present a theorem about how weak and strong counterexamples for all security properties relate to each other and to counterexamples for conventional EFGs.

Theorem 6.6 (Equivalence of (Conditional) Counterexamples)**.** *Let* Γ *be an EFG with honest behavior* h^* *and* Γ^c *the extension of* Γ *to a CEFG using only trivial conditions (as in Theorem 6.1). Then the following hold:*

1) A counterexample to Γ *satisfying property* $sp \in \{wi, weri, cr, pr\}$ *is also a weak counterexample to* Γ^c *satisfying* sp *and vice versa.*

2) A weak counterexample to Γ^c *satisfying property* $sp \in \{wi, weri, cr, pr\}$ *is also a strong counterexample to* Γ^c *satisfying* sp *and vice versa.*

Proof. Consider the security property weak immunity. With respect to 1), by Definition 6.15 we observe that a weak counterexample to weak immunity is independent on the world state, i.e. it is a counterexample for any world state. For games without conditional actions there is only one possible world state, with only trivial conditions. Hence, by definition, the counterexample provides an honest player p with a partial strategy s_{N-p} such p can be harmed. This coincides with the notion of a counterexample for games without conditional actions presented in Definition 3.5. In other words, 1) follows from the fact that the game trees of Γ and Γ^c coincide.

For 2), observing Definition 6.15 and Definition 6.16, we see that strong counterexamples deal with one particular fixed world state, whereas weak counterexamples are independent on the world state. Since there is only one world state for games with only trivial conditions, namely the function which assigns the condition `true` to every history of the game, weak and strong counterexamples for weak immunity coincide for games with only trivial conditional actions.

The argument for the remaining security properties follows the same reasoning. □

Finally, we note that the automated finding of counterexamples is left as a direction for future work.

6.6 Experimental Evaluation

Our compositional security approach for CEFGs has been automated following the algorithms in Section 6.4. Our implementation is available online in the CheckMate 2.0 tool[1].

Experimental Setup. We tested the framework with a small set of benchmarks and we present and discuss the results in the following. The current set of benchmarks includes games with non-trivial conditions and although both the benchmark set and the individual games remain relatively modest in size, they constitute an important first step toward the systematic analysis of games involving conditional actions. As such, this collection serves as a foundational reference for the development of more comprehensive benchmark sets in future work.

One of the games we evaluated our framework on is the *Market Entry game* with conditional actions, which is given in Figure 6.1 and discussed on several occasions throughout the examples in this section. Apart from this game, we also evaluated the framework on adaptations of the games *Simplified Closing*, *Auction*, *EBOS* and *Centipede*, analyzed in Section 5.4. The games have been extended to games with non-trivial conditions and can be found online in the CheckMate 2.0 tool[2]. For example,

[1] `https://github.com/apre-group/checkmate/tree/conditional-actions`

[2] `https://github.com/apre-group/checkmate/tree/conditional-actions/examples/conditional_actions`

Game	Nodes	Players	Security property	Secure yes/no	Time	Nodes evaluated	Nodes evaluated (reps)	Calls
Market Entry CA	9	2	wi_w	n	0.009	3	5	5
$[o_1[], e_2[i_1[], pw_2[]]]$			weri_w	y	0.009	3	5	5
			cr_w	y	0.010	3	5	14
			pr_w	y	0.010	9	9	10
			wi_s	n	0.010	8	12	16
			weri_s	n	0.009	8	12	16
			cr_s	y	0.011	9	14	49
			pr_s	y	0.010	9	9	10
Simplified Closing CA	27	2	wi_w	n	0.014	15	15	29
$[C_h[S_1[], S_2[], S_3[]],$			weri_w	n	0.011	15	15	26
$C'_h[S'_1[], S'_2[], S'_3[]]]$			cr_w	y	0.609	7	10	44
			pr_w	y	0.152	27	183	411
			wi_s	n	0.010	3	3	5
			weri_s	n	0.009	3	3	4
			cr_s	y	0.630	27	36	236
			pr_s	y	0.152	27	183	411
$[H[], H'[]]$			wi_w	y	0.014	8	10	14
			weri_w	y	0.013	8	10	12
			cr_w	y	0.010	6	8	16
			pr_w	n	0.050	27	69	191
			wi_s	y	0.018	27	30	54
			weri_s	y	0.017	27	30	48
			cr_s	y	0.016	27	30	89
			pr_s	n	0.043	27	69	162
Auction CA	97	3	wi_w	n	0.013	34	34	44
$[L_1[E_1[I[]], E_2[I[]]],$			weri_w	n	0.012	34	34	42
$L_2[E_1[I[]], E_2[I[]]]]$			cr_w	n	0.027	49	119	360
			pr_w	y	0.020	97	97	336
			wi_s	n	0.011	8	8	9
			weri_s	n	0.011	8	8	9
			cr_s	n	0.028	49	122	396
			pr_s	y	0.020	97	97	336
EBOS CA	89	4	wi_w	n	0.012	46	79	90
$[MINE_1[MINE_2[MINE_3[MINE_4[]]],$			weri_w	n	0.012	46	79	90
$MINE_9[MINE_{10}[MINE_{11}[]]]],$			cr_w	n	0.073	61	427	1444
$MINE_{1'}[MINE_{2'}[MINE_{3'}[MINE_{4'}[]]],$			pr_w	y	0.162	89	2870	8697
$MINE_{9'}[MINE_{10'}[MINE_{11'}[]]]]]$			wi_s	n	0.013	61	97	114
			weri_s	n	0.013	61	97	114
			cr_s	n	0.032	53	183	569
			pr_s	n	0.067	89	1046	3361
Centipede CA	17	2	wi_w	y	0.011	12	17	26
$[C_1[C_2[C_3[], C_4[]]], E_5[]]$			weri_w	y	0.010	12	17	23
			cr_w	y	0.010	7	11	28
			pr_w	y	0.022	17	85	221
			wi_s	n	0.009	3	3	5
			weri_s	n	0.009	3	3	4
			cr_s	y	0.011	14	20	64
			pr_s	n	0.022	17	85	218

Table 6.1: Experimental results of game-theoretic security for CEFGs using the compositional CheckMate 2.0 approach. Runtimes are given in seconds. For each game, columns 2–3 list the size (tree nodes and game players) of the game from column 1. Column 4 shows the game-theoretic security property we analyzed and (dis)proved, as indicated in column 5. The subscripts w and s refer to the weak and strong notions of the security properties, respectively. Columns 6–9 present the results of CheckMate 2.0 in terms of execution time, number of evaluated notes with and without repetition and the number of calls to the SMT solver.

Simplified Closing has been extended to include conditions involving several variables that represent price factors, which also appear in the utilities now. Likewise, *Centipede* can now be interpreted as a game where inheritance is divided between two parties and

depends on their age.

The evaluation has been conducted using a machine with 2 AMD EPYC 7502 CPUs clocked at 2.5GHz with 32 cores and 1TB RAM.

Experimental Results. The results of the evaluation for games with conditional actions are presented in Table 6.1. The first column lists each game along with the corresponding analyzed honest behavior. Note that, for instance, for the *Market Entry game* with conditional actions the honest behavior is denoted by $[o_1[], e_2[i_1[], pw_2[]]]$, meaning that at the root, the honest actions for the two available conditions are o_1 and e_2 and in the subtree restricted to the condition where e_2 is an available action, the honest actions for the available conditions are i_1 and pw_2. Columns 2 and 3 report the number of nodes and the number of players in the respective game trees. Column 4 shows the game-theoretic security property we analyzed and (dis)proved, as indicated in column 5. The subscripts w and s refer to the weak and strong notions of the security properties, respectively. Columns 6—9 present the results of CheckMate 2.0 in terms of execution time, number of evaluated notes with and without repetition and the number of calls to the SMT solver.

Experimental Analysis. Our framework correctly analyzed all benchmarks in terms of the weak and strong notions of the security properties. As it can be seen from Table 6.1, the execution time for all instances and properties was below one second, given that they are relatively simple benchmarks. It is not possible to establish definitive relationships between the number of evaluated nodes and function calls based solely on whether we are verifying the weak or strong versions of the properties, or whether those properties hold. This is because such factors depend heavily on the game's structure and the required case splits. The only consistent pattern observed is that, when one of the properties *wi*, *weri*, or *cr* is satisfied in terms of both the weak and strong notion, verifying the weak notion involves fewer evaluated nodes and calls than verifying the strong one.

Current Limitations and Future Work. Given their relatively limited size, the discussed benchmarking set is intended primarily for illustrative and verification purposes within the context of the proposed approach. It will be especially interesting to find out how this approach scales and if CheckMate 2.0 with support for conditional actions can analyze big real-life models in feasible time. In particular, we would like to model and analyze the Liquidation Phase of FAsset presented in Example 6.1.

Moreover, the extraction of counterexamples needs to be automated and accordingly evaluated, following the definitions of conditional counterexamples (Section 6.5). Automatic extraction of strategies for CEFGs similar to the approach for EFGs discussed in Section 5.2 is also left as a direction for future work.

Lastly, another interesting direction for future research is to include large benchmarks, where we can intertwine the generation and the analysis, similarly as for the 4-Player Routing in Section 5.4.

CHAPTER 7

Conclusion

Game-theoretic approaches have proven to be quite helpful and valuable for analyzing blockchain technologies from an economic standpoint, particularly by examining the behavior of players. However, existing frameworks face challenges related to scalability and expressivity.

In this thesis, we discussed the first approach for compositional analysis of security properties within game-theoretic models. We defined player-dependent notions of security and developed a method for decomposing games into subgames. Moreover, we interleave subgame and supergame reasoning such that the security result of a subgame can be added as a leaf and used in the respective supergame. This is particularly advantageous for subgames that occur multiple times in the supertree.

Our framework supports the generation of counterexamples when a property is violated, as well as the extraction of strategies when a property is satisfied, all in a compositional manner. This approach offers these capabilities with minimal overhead.

By combining our theoretical results with SMT-based techniques, we automated compositional reasoning in a sound and complete way. We validated our implementation using benchmarks from prior approaches, alongside new ones and observed significant improvements in scalability, especially when applied to real-world protocols with millions of nodes and actions.

Moreover, we enhanced the expressivity of the existing framework by enabling the modeling and analysis of games influenced by external uncontrollable factors. The extension of the framework enables users to model various conditions, with the game adapting based on which condition is satisfied – reflecting the current state of the world. For this purpose, we redefined strategies, honest histories, counterexamples and related notions. In addition, we introduced the concepts of weak and strong conditional security, which have also been automated. This approach has been evaluated on a small set of

benchmarks with non-trivial conditions, which can serve as a foundational reference for the development of more comprehensive benchmark sets in future work.

Despite these advancements, our work has some limitations. Notably, while we already support analysis of games with conditional actions, there is significant potential to further extend this functionality. Promising directions for future work include evaluating our implementation on a broader range of games with conditional actions – particularly real-world scenarios – as well as providing support for automated strategy extraction and counterexample generation for games involving conditional actions.

Overview of Generative AI Tools Used

- DeepL Translate, https://www.deepl.com/
- Grammarly: Free AI Writing Assistance, https://app.grammarly.com/

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Clark Barrett and Cesare Tinelli. „Satisfiability Modulo Theories“. In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. UC Berkeley: Springer International Publishing, 2018, pp. 305–343. ISBN: 978-3-319-10575-8. DOI: 10.1007/978-3-319-10575-8_11.

[2] *Bitcoin homepage*. https://bitcoin.org/. [Online; accessed March 10, 2025].

[3] Nikolaj Bjørner and Lev Nachmanson. „Arithmetic Solving in Z3“. In: *Computer Aided Verification: 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24–27, 2024, Proceedings, Part I*. Montreal, QC, Canada: Springer-Verlag, 2024, pp. 26–41. ISBN: 978-3-031-65626-2. DOI: 10.1007/978-3-031-65627-9_2.

[4] Bruno Blanchet. „Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif“. In: *Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures*. Ed. by Alessandro Aldini, Javier Lopez, and Fabio Martinelli. Cham: Springer International Publishing, 2014, pp. 54–87. ISBN: 978-3-319-10082-1. DOI: 10.1007/978-3-319-10082-1_3.

[5] Ivana Bocevska, Anja Petković Komel, Laura Kovács, Sophie Rain, and Michael Rawson. *Divide and Conquer: a Compositional Approach to Game-Theoretic Security*. EasyChair Preprint 15785, https://easychair.org/publications/preprint/kxKK. 2025.

[6] Lea Salome Brugger, Laura Kovács, Anja Petkovic Komel, Sophie Rain, and Michael Rawson. „CheckMate: Automated Game-Theoretic Security Reasoning“. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. CCS '23. Copenhagen, Denmark: Association for Computing Machinery, 2023, pp. 1407–1421. ISBN: 9798400700507. DOI: 10.1145/3576915.3623183.

[7] Leonardo De Moura and Nikolaj Bjørner. „Z3: an efficient SMT solver“. In: *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. TACAS'08/ETAPS'08. Budapest, Hungary, 2008, pp. 337–340. ISBN: 3540787992. DOI: 10.1007/978-3-540-78800-3_24.

[8] Dino Distefano, Manuel Fähndrich, Francesco Logozzo, and Peter W. O'Hearn. „Scaling static analyses at Facebook". In: *Commun. ACM* 62.8 (2019), pp. 62–70. ISSN: 0001-0782. DOI: 10.1145/3338112.

[9] *Dogecoin homepage*. https://dogecoin.com/. [Online; accessed March 10, 2025].

[10] *Flare Developer Hub, Liquidation*. https://dev.flare.network/fassets/liquidation. [Online; accessed March 10, 2025].

[11] *Flare Developer Hub, Overview*. https://dev.flare.network/fassets/overview. [Online; accessed March 10, 2025].

[12] *Flare homepage*. https://flare.network/. [Online; accessed March 10, 2025].

[13] *Flare homepage, FAsset*. https://flare.network/products/fassets. [Online; accessed March 10, 2025].

[14] Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. „Compositional Game Theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: Association for Computing Machinery, 2018, pp. 472–481. ISBN: 9781450355834. DOI: 10.1145/3209108.3209165.

[15] Nadim Kobeissi, Georgio Nicolas, and Mukesh Tiwari. „Verifpal: Cryptographic Protocol Analysis for the Real World". In: *Progress in Cryptology – INDOCRYPT 2020*. Ed. by Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran. Cham: Springer International Publishing, 2020, pp. 151–202. ISBN: 978-3-030-65277-7. DOI: 10.1007/978-3-030-65277-7_8.

[16] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. „The TAMARIN Prover for the Symbolic Analysis of Security Protocols". In: *Proceedings of the 25th International Conference on Computer Aided Verification - Volume 8044*. CAV 2013. Saint Petersburg, Russia: Springer-Verlag, 2013, pp. 696–701. ISBN: 9783642397981. DOI: 10.1007/978-3-642-39799-8_48.

[17] Peter W. O'Hearn. „Continuous Reasoning: Scaling the impact of formal methods". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: Association for Computing Machinery, 2018, pp. 13–25. ISBN: 9781450355834. DOI: 10.1145/3209108.3209109.

[18] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. Cambridge, USA: The MIT Press, 1994.

[19] Joseph Poon and Thaddeus Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*. 2016.

[20] Sophie Rain, Georgia Avarikioti, Laura Kovács, and Matteo Maffei. „Towards a Game-Theoretic Security Analysis of Off-Chain Protocols". In: *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*. 2023, pp. 107–122. DOI: 10.1109/CSF57540.2023.00003.

[21] Sophie Rain, Lea Salome Brugger, Anja Petković Komel, Laura Kovács, and Michael Rawson. „Scaling CheckMate for Game-Theoretic Security“. In: *Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning*. Ed. by Nikolaj Bjørner, Marijn Heule, and Andrei Voronkov. Vol. 100. EPiC Series in Computing. Stockport, UK: EasyChair, 2024, pp. 222–231. DOI: 10.29007/llnq.

[22] Raymond M Smullyan. *First-Order Logic*. New York: Dover Publications, 1995.

[23] Scott Wesley, Maria Christakis, Jorge A. Navas, Richard Trefler, Valentin Wüstholz, and Arie Gurfinkel. „Compositional Verification of Smart Contracts Through Communication Abstraction“. In: *Static Analysis: 28th International Symposium, SAS 2021, Chicago, IL, USA, October 17–19, 2021, Proceedings*. Chicago, IL, USA: Springer-Verlag, 2021, pp. 429–452. ISBN: 978-3-030-88805-3. DOI: 10.1007/978-3-030-88806-0_21.

[24] Amalee Wilson, Andres Noetzli, Andrew Reynolds, Byron Cook, Cesare Tinelli, and Clark Barrett. „Partitioning strategies for distributed SMT solving“. In: *Proceedings of the 23rd Conference on Formal Methods in Computer-Aided Design – FMCAD 2023*. Ed. by Alexander Nadel and Kristin Yvonne Rozier. 2023, pp. 199–208. DOI: 10.34727/2023/isbn.978-3-85448-060-0_28.

[25] *XRP homepage*. https://xrpl.org/. [Online; accessed March 10, 2025].

[26] Paolo Zappalà, Marianna Belotti, Maria Potop-Butucaru, and Stefano Secci. „Game Theoretical Framework for Analyzing Blockchains Robustness“. In: *Leibniz International Proceedings in Informatics (LIPIcs)*. Vol. 209. Leibniz International Proceedings in Informatics (LIPIcs). Freiburg, Germany: Schloss Dagstuhl, Oct. 2021, 42:1–42:18. DOI: 10.4230/LIPIcs.DISC.2021.42.