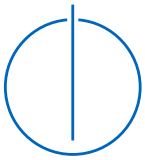
TECHNISCHE UNIVERSITÄT MÜNCHEN

FAKULTÄT FÜR INFORMATIK

Proving Noninterference in Multi-Agent Systems

Alexander Christian Müller



TECHNISCHE UNIVERSITÄT MÜNCHEN

FAKULTÄT FÜR INFORMATIK

LEHRSTUHL FÜR SPRACHEN UND BESCHREIBUNGSSTRUKTUREN

Proving Noninterference in Multi-Agent Systems

Alexander Christian Müller

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat)

genehmigten Dissertation.

Vorsitzender:

Prof. Tobias Nipkow, Ph. D.

Prüfer der Dissertation:

1. Prof. Dr. Helmut Seidl
2. Prof. Dr. Francisco Javier Esparza Estaun

Die Dissertation wurde am 18.02.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 25.05.2020 angenommen.

I confirm that this thesis is my own work and I have documented all sources and material used.

Munich, 18.02.2020 Alexander Christian Müller

Acknowledgements

Foremost, I want to express sincere gratitude to my advisor Helmut Seidl. Without his continued support and the many, many hours of productive discussions none of the work in this thesis would have been possible. I also thank my mentor Javier Esparza, who always had an open ear for my troubles, welcomed me into his group and provided a new home for the practical course we invented. Furthermore, I want to thank my co-author Bernd Finkbeiner from the university of Saarbrücken where I was not only invited several times, but even became a part of his group towards the end of my thesis. During the several trips to Saarbrücken, I was always warmly welcomed and we had fruitful discussions where many of the proofs in this work originated.

A special thanks goes to all my colleagues from TUM: This includes my co-authors Máté Kovács and Eugen Zalinescu and all the awesome colleagues from both chairs I7 and I2, who always made me feel a member of both chairs and who I spent many a coffee break with.

Beyond my research, I ended up dedicating my time toward the ICPC community, both teaching courses as well as organizing multiple years worth of programming contests and training camps. For the great memories and fun times I thank everyone involved, in particular Stefan Toman, Gregor Schwarz, Chris Pinkau, Moritz Fuchs, Philipp Meyer and Kirill Martynov.

I also want to thank my friend and office-mate Philipp Hoffmann, who provided valuable feedback on most of my papers as well as this thesis, co-wrote many fun problems for the ICPC and spent more than a dozen weeks travelling around the world with me to multiple conferences, programming contests and/or just for fun.

Last but not least, I want to thank my parents for their moral support and the certainty to always have a worry-free place to come home to as well as my friends and flatmates for the many enjoyable days distracting me from university.

Abstract

In this thesis, we show how to specify and verify safety as well as secrecy in workflow systems with arbitrarily many participating agents.

As the underlying formal model we rely on *First-order Transition Systems*, which allow us to describe the states of multi-agent systems with an unbounded number of participants by means of First-order Predicates. In order to specify secrecy properties for First-order Transition Systems, we introduce *First-order HyperLTL*, as the First-order extension of (propositional) HyperLTL. HyperLTL is a formal temporal logic which is able to express complex information flow requirements.

To verify these properties, we provide several approaches. Our first approach is to encode both the system and the property φ to be verified into First-order HyperLTL satisfiablity, which then is reduced to satisfiability of First-order LTL. This allows us to use First-order satisfiability tools like Z3 to prove the absence of counterexamples of a bounded length as well as use First-order LTL satisfiability tools to verify that φ holds. We identify classes of First-order Transition Systems for which our methods are able to effectively decide if φ holds. For cases where our algorithms are incomplete, we prove that the problem is undecidable in general. This approach is complemented by a second method which is applicable to arbitrary First-order Transition Systems, and tailored for the verification of Noninterference in presence of declassification and specifications of the agents' capabilities. In this approach, as much of the specification as possible is encoded into the transition system. This simplifies the temporal .property to be verified and allows us to verify it by means of *inductive invariants*. Again, rich sub-classes are identified where effective verification algorithms can be provided. For arbitrary Noninterference properties and First-order Transition Systems we introduce approximation methods and are still able to prove interesting systems correct.

In order to go beyond verification we extend our setting to *First-order Safety Games*. Our goal is to automatically construct *strategies* which enforce safety or noninterference. We succeed in doing so in non-trivial cases by means of approximative Second Order existential quantifier elimination.

The methods developed in this thesis have been implemented into software — the resulting tool NIWO is a fully automated solver for verifying and inferring properties of workflow systems automatically as well as for synthesizing safe strategies.

Zusammenfassung

In dieser Arbeit modellieren wir Sicherheits- sowie Geheimhaltungseigenschaften von Workflow-Systemen mit einer unbeschränkten Anzahl von beteiligten Agenten.

Als zugrunde liegendes formales Modell benutzen wir *prädikatenlogische Transitionssysteme* (First-order Transition Systems), die es uns erlauben, die Zustände eines Multi-Agenten-Systems mithilfe von Prädikaten erster Stufe zu beschreiben. Um Geheimhaltungseigenschaften für Prädikatenlogische Transitionssysteme zu formalisieren benutzen wir *First-order HyperLTL*, die prädikatenlogische Erweiterung von (propositionalem) HyperLTL. HyperLTL ist eine formale temporale Logik, die in der Lage ist komplexe Anforderungen an den Informationsfluss auszudrücken.

Um diese Eigenschaften zu verifizieren, benutzen wir verschiedene Ansätze: Unser erster Ansatz besteht darin, sowohl das System als auch die zu verifizierende Eigenschaft in First-order HyperLTL Erfüllbarkeit zu codieren und diese dann auf die Erfüllbarkeit von First-order LTL zu reduzieren. Dies erlaubt uns mithilfe von Werkzeugen wie Z3 die Abwesenheit von Gegenbeispielen begrenzter Länge nachzuweisen, sowie mithilfe von Werkzeugen für First-order LTL zu überprüfen, ob die gegebene Eigenschaft auf allen unbeschränkten Ausführungspfaden gilt. Wir identifizieren Klassen von prädikatenlogischen Transitionssystemen, für die unsere Methoden in der Lage sind, effektiv zu entscheiden, ob eine gegebene Eigenschaft gilt. Für Fälle, in denen unsere Algorithmen unvollständig sind, beweisen wir, dass das Problem generell unentscheidbar ist. Dieser Ansatz wird durch einen zweiten Ansatz ergänzt der für allgemeine prädikatenlogische Transitionssysteme anwendbar ist. Dieser konzentriert sich auf die Verifikation der Sicherheitsrichtlinie *Noninterference* unter Berücksichtigung von Deklassifikation. Bei diesem Ansatz wird so viel von der Spezifikation wie möglich innerhalb des Transitionssystems selbst kodiert. Das erlaubt uns die nun stukturell einfachere Eigenschaft mithilfe von induktiven Invarianten zu verifizieren. Auch hier identifizieren wir Klassen von Transitionssystemen, für die das Verifikationsproblem entscheidbar ist. Für allgemeine First-order Transitionssysteme benutzen wir Näherungsverfahren um noch möglichst viele interessante Systeme verifizieren zu können.

Zusätzlich zur reinen Verifikation erweitern wir unser Modell zu *First-order Safety Games*. Hierbei gilt es nicht nur herauszufinden ob eine bestimmte Eigenschaft gilt, sondern darüber hinaus automatisiert eine *Strategie* zu konstruieren, die die gewünschte Eigenschaft sicherstellt. Dies gelingt uns in nicht-trivialen Fällen durch die annähernde Elimination von Second Order Existenzquantoren.

Die in dieser Arbeit entwickelten Methoden wurden implementiert. Das Ergebnis ist NIWO — ein vollautomatischer Solver zur Verifizierung von Eigenschaften von Workflow-Systemen sowie zur Synthese sicherer Strategien.

Contents

Chapter 1

Introduction

Contents

1 Introduction

Web-based workflow management systems allow diverse groups of users to collaborate efficiently on complex tasks. This has led to their wide-spread use throughout the daily lives of people from many different fields. For example, conference management systems like EasyChair let authors, reviewers, and program committees collaborate on the organization of a scientific conference; health management systems like HealthVault let family members, doctors, and other health care providers collaborate on the management of a patient's care. Shopping sites like Amazon or Ebay let merchants, customers, as well as various other agents responsible for payment, customer service, and shipping, collaborate on the purchase and delivery of products.

Since the information maintained in such systems is often confidential, these systems must carefully manage who has access to what information in a particular stage of the workflow. For the designer of the system, this is a very hard task that has to be verified rigorously, since both programming errors as well as conceptual errors can easily happen.

As a running example, we will use the reviewing process inside a scientific conference management system akin to EasyChair. Here, members of the program committee (PC members) must first declare with which submissions they are in conflict of interest. Then the organizers of the conference assign the members of the PC to the submissions so that every submission gets reviews from multiple different PC members. After the reviews have been written, all PC members that are assigned to the same paper discuss about their reviews and eventually decide together wether to accept (and publish) the submission. Naturally, PC members should only see reviews and discussions of papers with which they have declared no conflict of interest. Authors eventually get access to reviews of their papers, but only when the process has reached the official notification stage, and without identifying information about the reviewers.

Example 1.1.

During submission of the first paper of the work in this thesis [39], one of the authors was also a member of the program committee and had to review a different paper p. Someone else, who also reviewed the same paper p unfortunately mixed up his reviews and accidentally pasted his review and discussion of paper [39] into the discussion of paper p — allowing us to learn about the internal discussion related to our work. By modeling this process beforehand, such an assignment could have been prevented.

To ensure that these kinds of information leaks can not happen in a finished system, the scientific community has come up with many different analysis methods and tools that can analyze software in a semi- or fully automated way. The approaches range from high-level conceptual approaches like attack trees [65] to low-level formal proofs using proof assistants like Isabelle [76] or Coq [25] to reason about the specific code of the system. An example for the latter approach is CoCon [56], a conference management which comes with confidentiality guarantees which have been formalized and proven in

Isabelle and then used to generate correct code for the system. However, these proofs have been hand-written and are specific to CoCon, so can not easily be reused for different systems.

In this thesis, we introduce how to formally specify secrecy requirements in workflow systems with arbitrarily many participating agents. These requirements include declassification conditions, which specify under which conditions specific agents are declassified to learn about the information. For the example of a conference management system, it might be alright for a PC member to learn about the reviews of a specific paper as long as the PC member did not specify a conflict of interest with this specific paper. In addition, secrecy policies require an *agent capability model* that specifies which parts of the state of the system can be observed by any specific agent — and that all decisions of an agent only depend on information actually available to this agent. These agents might be real people interacting via their browser or different automated systems employed by payment providers or publishers, each with their own goals, knowledge and possible interactions.

We then show how to *automatically* prove if a high-level model of the given workflow system adheres to the secrecy requirements or not. If already the high-level model of the workflow system exhibits information leaks, there is no way for the actual implementation not to leak information as well. Thus, we are able to spot mistakes already in the conceptual phase of a new system, without having to analyze the code itself.

This approach is called *model checking* [28] and has been used very successfully to prove that a given *finite* system is correct [55, 10]. The particular challenge with verifying web-based workflow systems however, is that here is no fixed set of agents participating in the workflow. Clearly, we would not like to reason about the correctness of a conference management system for every concrete instance of a particular conference, a particular program committee and a particular set of submissions and reports. Instead, we would like to prove a given system only once — for any possible instantiation and any number of PC members, submitted papers and reports. In this thesis, we thus develop methods to handle the setting of an unbounded universe.

1.1 Structure of this Thesis

In Chapter 2, we will set the stage for the rest of the thesis and recall some basic notation and semantics of First-order logics that we will use heavily throughout the rest of the thesis. From there, we will motivate and introduce *First-order Transition Systems* (FO Transition Systems) as a formal model of parameterized multi-agent workflow systems in Chapter 3. We show how to embed the examples we mention into this model, discuss alternatives to our modeling approach and show how to embed some of the alternative models into FO Transition System.

Chapter 4 then shows how to specify secrecy policies for FO Transition Systems, specifically *Noninterference*. To do so, we recall Linear Temporal Logic and its First-order variant, as well as introduce *First-order HyperLTL*, an extension that allows us to reason about multiple executions of a given system. We then give a formal definition of Noninterference with added declassification conditions as well as an observation model of the participating agents in First-order HyperLTL.

In Chapter 5, we then show how to apply techniques and ideas from the area of *model checking* to automatically verify that a given FO Transition System satisfies a First-order Hyperproperty for the restricted class of quantifier-free transition systems and apply our findings to the special case of Noninterference properties. We also show that this approach can not be extended to *general* FO Transition Systems and prove the problem undecidable.

Chapter 6 then continues to investigate incomplete methods that can still solve the general case in practice. We consider simplified setting of *invariants* for FO Transition Systems, and show how to cast Noninterference as such. We show how to verify that a given invariant is *inductive*, and provide methods on how to infer an inductive strengthening if it is not. For this, we use the abstract domain of purely universal First-order formulas. We identify classes of transition systems where invariant inference is decidable and and show how to handle the general case — which necessarily leads to an incomplete method, but which works well in practice. We then apply the results to Noninterference and compare our results to the state of the art of alternative verification methods for parameterized systems.

Chapter 7 then extends the setting to *First-order Safety Games*. These are transition systems where some decisions by the environment are not assumed to be picked malevolently, but rather benevolently. This leads us to the synthesis problem, which asks "Is there a strategy we can use to ensure that the given system is well-behaved?" We show how to formalize this question and indicate how Second Order Quantifier elimination can be used to solve it. In addition, this allows us to extract a *strategy* to ensure safety. We then adapt the methods of Chapter 6 to the setting of games and further improve the progress guarantees of the incomplete verification algorithm.

After laying the theoretical groundwork, Chapter 8 introduces NIWO, our fully automated solver for First-order Transition Systems and Games based on the methods of Chapters 5 to 7 and evaluate its performance on the examples mentioned throughout the thesis.

Finally, Chapter 9 concludes, summarizes the thesis and gives an outlook to future work.

The different chapters tackle rather diverse techniques and approaches. Accordingly, each chapter will feature its own section about related approaches and models, which will be discussed at the end of the respective chapter.

1.2 Preceding Publications

This thesis includes the content of the following four scientific publications:

[39] Bernd Finkbeiner, Helmut Seidl, and Christian Müller. "Specifying and Verifying Secrecy in Workflows with Arbitrarily Many Agents". In: *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*. Ed. by Cyrille Artho, Axel Legay, and Doron Peled. Vol. 9938. Lecture Notes in Computer Science. 2016, pp. 157–173. ISBN: 978-3-319-46519-7. DOI: 10.1007/978-3-319-46520-3_11

[37] Bernd Finkbeiner, Christian Müller, Helmut Seidl, and Eugen Zalinescu. "Verifying Security Policies in Multi-agent Workflows with Loops". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM, 2017, pp. 633–645. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134080

[69] Christian Müller, Helmut Seidl, and Eugen Zalinescu. "Inductive Invariants for Noninterference in Multi-agent Workflows". In: *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, 2018, pp. 247–261. ISBN: 978-1-5386-6680-7. DOI: 10.1109/CSF.2018.00025

[82] Helmut Seidl, Christian Müller, and Bernd Finkbeiner. "How to Win First-Order Safety Games". In: *Verification, Model Checking, and Abstract Interpretation - 21st International Conference, VMCAI 2020, New Orleans, LA, USA, January 16-21, 2020, Proceedings*. Ed. by Dirk Beyer and Damien Zufferey. Vol. 11990. Lecture Notes in Computer Science. Springer, 2020, pp. 426–448. ISBN: 978-3-030-39321-2. DOI: 10.1007/978-3-030-39322-9_20

Our automated solver NIWO has been published as a software artifact together with [82]:

[81] Helmut Seidl, Christian Müller, and Bernd Finkbeiner. *How to Win First Order Safety Games - Software Artifact*. Oct. 2019. DOI: 10.5281/zenodo.3514277

Chapters 2 and 3 introduce the general model of multi-agent systems upon which all papers are based. Chapter 4 introduces the temporal specification for Noninterference as discussed in [37, 69]. Chapter 5 is based on the results of [39, 37] and introduces model checking techniques. Chapter 6 is based on the results of [69] and uses invariants to prove safety properties. Chapter 7 generalizes the model to include synthesis questions and is based on [82]. Chapter 8 describes the architecture and experimental results of our software as described in [37, 69, 82].

Chapter 2

Preliminaries

Contents

2 Preliminaries

As stated in the introduction, we care about workflow management systems like conference management systems. Any description of these should not depend on the actual papers submitted or the members of the PC chair. Rather, it should define a template that can be instantiated for any given number of participating reviewers, submitted papers or posts in the discussion forums.

Formally, if we want to describe these systems using logic, we need to do so without specifying the specific universe for which they are instantiated. For this purpose, propositional logic is not enough. Instead, we will use *First-order Logic* throughout this thesis as it is a powerful tool that allows us to use *predicates* and *quantifiers* to describe system states and transitions between them rather than referring to (finitely many) propositional facts only.

2.1 Sorted First-order Logic

First-order Logic (FO Logic), also known as First-order Predicate Logic, is an assertion language that can form statements about the *predicates* of objects as well as *relations* between objects.

Example 2.1.

Sentences may include *quantification*, which is used to make universal statements applying to several individuals at once. An example is:

$$\forall x.\ \mathit{Human}(x) \rightarrow \mathit{Mortal}(x)$$

which says "All individuals x that are *human* are also *mortal*". Sometimes, we want quantifiers to only range over specific individuals. For this, we use *sorts* that are specified together with the quantifier:

$$\forall x\colon \mathit{Human}.\ \mathit{Mortal}(x)$$

Quantifiers can also be nested to yield more complex sentences:

$$\forall s\colon \mathit{Subject}.\ \exists x\colon \mathit{Student}.\ \mathit{Interested}(x, s)$$

which says "For each subject s, there exists a student x that is interested in studying said subject s."

To build formulas, we need to know which predicates exist and which sorts their parameters need to be. This information is given by a *signature*. A signature $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$ consists of a non-empty and finite set of sorts S, finite and disjoint sets $\mathcal{C}$ and $\mathcal{R}$ of *function symbols* and *relation symbols* (or predicate symbols). We assign to each function or relation symbol an *arity*, given by function $ar : \mathcal{C} \cup \mathcal{R} \rightarrow S^*$, where

S^* denotes the set of finite sequences of sorts. For a relation R, the sequence of sorts $ar(R)$ specifies the sequence of sorts the parameters of R should follow. For functions f, the last sort in $ar(f)$ specifies the sort of the returned value, while the rest specifies the parameter types. Functions with no parameters are called *constant*, functions and relations with a single parameter are called *unary* or *monadic* and functions and relations with two parameters are called *binary*. In this thesis, we will not consider functions beyond constants, thus $|ar(c)| = 1$ for any $c \in \mathcal{C}$.

For each sort s, we let $\mathcal{V}_s$ be a countably infinite set of *variable symbols*. We let $\mathcal{V} := \bigcup_{s \in S} \mathcal{V}_s$ be the set of all variables of all possible sorts.

Example 2.2.

A signature for formulas about a university setting is for example $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$ with

$$\begin{aligned} S &= \{Student, Professor, Subject\}, \\ \mathcal{C} &= \{Dean\}, \\ \mathcal{R} &= \{Studies, Teaches\}, \\ ar &= \left\{ \begin{aligned} Dean &: Professor \\ Studies &: (Student, Subject) \\ Teaches &: (Professor, Subject) \end{aligned} \right\} \end{aligned}$$

This enables us to build formulas over the relations *Studies* and *Teaches* and quantify over individuals or objects who are of type *Student*, *Professor* or *Subject*. We additionally have a constant *Dean* for the dean of the faculty who is of type *Professor*.

Sentences of Sorted First-order Logic are produced by the following grammar:

$$\varphi \quad := \quad \exists v : s.\ \varphi \mid \neg\varphi \mid \varphi \vee \varphi \mid t = t' \mid R(t_0, \ldots, t_k)$$

where s ranges over S, t, t' range over $\mathcal{V} \cup \mathcal{C}$ and are of the same sort, R ranges over $\mathcal{R}$ and for $ar(R) = s_0 \ldots s_k$ the t_i are of the correct sorts, i.e. t_i is of sort s_i for all i. We also let $sort(t)$ denote the sort of term t.

Here, the operators $\neg, \vee$ stand for boolean *negation* and *disjunction* while $\exists$ is called *existential* quantification. Additionally, we use abbreviations for *universal* quantification as well as the standard Boolean connectives conjunction, implication and equivalence:

$$\begin{aligned} \forall x : s.\ \varphi &:= \neg\exists x : s.\ \neg\varphi \\ \varphi_1 \wedge \varphi_2 &:= \neg(\neg\varphi_1 \vee \neg\varphi_2) \\ \varphi_1 \to \varphi_2 &:= \neg\varphi_1 \vee \varphi_2 \\ \varphi_1 \leftrightarrow \varphi_2 &:= \varphi_1 \to \varphi_2 \wedge \varphi_2 \to \varphi_1 \\ \varphi_1 \not\leftrightarrow \varphi_2 &:= \neg(\varphi_1 \leftrightarrow \varphi_2) \end{aligned}$$

We also abbreviate sequences of the same quantifier to quantification over sequences, e.g. $\exists x_1 : A_1.\ \exists x_2 : A_2$ is abbreviated to $\exists x_1 : A_1, x_2 : A_2$.

The set of *free variables* of a formula φ, that is, those that are not in the scope of some quantifier in φ, is denoted by $fv(\varphi)$. For a term $t \in \mathcal{V} \cup \mathcal{C}$, we let $fv(t) := \{t\}$ if $t \in \mathcal{V}$ and $fv(t) := \emptyset$ otherwise. A formula without free variables is called *closed*. Closed formulas

might still use constants from $\mathcal{C}$. We may drop the sort in $\exists x{:}s.\,\varphi$ when it is irrelevant or clear from the context and simply write $\exists x.\,\varphi$. To omit parentheses, we assume that unary connectives bind stronger than binary ones and quantifiers bind weaker than boolean connectives.

We also define *renaming* for formulas. For a formula φ, we denote by $\varphi[y/x]$ the formula φ where all occurrences of x have been replaced by y. We will typically use this notation to rename free variables.

Example 2.3.

Let φ be the formula $\exists z.R(x,y,z)$. Then $\varphi[y/x]$ is $\exists z.R(y,y,z)$.

2.1.1 Semantics

Given a signature we know which sorts, relations and constants exist in our setting. However, we still do not know which specific individuals or objects exist and how to interpret the relations between them. For this, we need *structures*. A particular structure provides us with information about which objects exist and how to interpret the relations and constants in any given formula. Given a structure and a formula we can then *evaluate* the formula to find out if it holds in this particular structure.

A structure s over the signature $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$ consists of a (finite or infinite) *universe* (domain) $U_s \neq \emptyset$ for every sort s from S and an *interpretation* $R^s \subseteq U_{s_1} \times \cdots \times U_{s_k}$ for each constant or relation R from $\mathcal{C} \cup \mathcal{R}$ of arity $(s_1, \ldots, s_k)$. We let U be the union of all universes, i.e. $U := \bigcup_{s \in S} U_s$.

In addition to constants, a formula can contain free variables. These are interpreted by a *valuation*. A valuation is a mapping $\nu : \mathcal{V} \to U$ with x and $\nu(x)$ of the same sort for any $x \in \mathcal{V}$. Thus, a valuation is used to map variable names to a particular element of the universe (of the correct sort), as given by a structure.

We now introduce a bit of notation to manipulate valuations. In the following, ν is a valuation, $\bar{x} = (x_1, \ldots, x_n)$ is a tuple of variables with $x_i \in \mathcal{V}_{s_i}$ and $\bar{d} = (d_1, \ldots, d_n)$ is a tuple of universe elements with $d_i \in U_{s_i}$, both for some sort s_i for each i. We write $\nu[\bar{x} \mapsto \bar{d}]$ for the valuation that maps each x_i to d_i and leaves the other variables' valuation unaltered. By $\nu(\bar{x})$ we denote the tuple $(\nu(x_1), \ldots, \nu(x_n))$. We extend this notation by applying a valuation ν also to constant symbols $c \in \mathcal{C}$, with $\nu(c) = c^s$. An *update* of a valuation of the first-order variables is defined as follows: $\nu[x \mapsto a](x) = a$ and $\nu[x \mapsto a](y) = \nu(y)$ for $x \neq y$.

Let s be a structure over the signature Σ, φ a formula over Σ and ν a valuation. Now that the signature Σ tells us which sorts and relations exist, and s provides a universe for each sort and interpretations for all relations, we can *evaluate* a closed formula φ over Σ on the structure s. We define the relation $s, \nu \models \varphi$ ("φ holds on structure s and valuation ν") inductively as follows:

$$
\begin{array}{lll}
s, \nu \models t = t' & \text{iff} & \nu(t) = \nu(t') \\
s, \nu \models R(\bar{t}) & \text{iff} & \nu(\bar{t}) \in R^s \\
s, \nu \models \neg\psi & \text{iff} & s, \nu \not\models \psi \\
s, \nu \models \psi \vee \psi' & \text{iff} & s, \nu \models \psi \text{ or } s, \nu \models \psi' \\
s, \nu \models \exists x : s.\,\psi & \text{iff} & s, \nu[x \mapsto d] \models \psi, \text{ for some } d \in U_s
\end{array}
$$

A FO Logic formula φ is said to be *satisfiable* iff there exists a structure s and a valuation ν s.t. $s, \nu \models \varphi$, i.e. a formula is satisfiable if it is true for *some* instantiation of the universes, relations and free variables. We say a closed formula φ *holds* on a structure s (also written $\mathcal{S} \models \varphi$) iff $s, \emptyset \models \varphi$.

Example 2.4.

For the university signature above from Example 2.2, a possible structure s consists of:

$$
\begin{aligned}
U_{Student} &= \{Philipp, Christian, Salomon\},\\
U_{Professor} &= \{Esparza, Seidl, Bungartz\},\\
U_{Subject} &= \{Automata, AbstractInterpretation\},\\
Dean^s &= Bungartz,\\
Studies^s &= \left\{\begin{array}{r}(Philipp, Automata),\\ (Salomon, Automata),\\ (Christian, AbstractInterpretation)\end{array}\right\}\\
Teaches^s &= \left\{\begin{array}{r}(Esparza, Automata),\\ (Seidl, AbstractInterpretation)\end{array}\right\}
\end{aligned}
$$

On this structure s, the closed formula

$$\forall s : Student.\ \exists x : Subject.\ Studies(s, x)$$

holds, as every student studies at least one subject.

We will sometimes consider that formulas are in *negation normal form*, which is obtained by pushing negation inside until it appears only in front of atomic formulas. When considering this form, the abbreviated operators $\wedge$, $\forall$ are seen as primitives. A formula is in *prenex normal form* if it is written as a sequence of quantifiers followed by a quantifier-free part. The *quantifier rank* $qr(\varphi)$ of a formula φ is the length of the sequence of quantifiers of φ in prenex normal form.

2.1.2 Fragments

For background on FO logic and known decidable subclasses, we refer to the textbook [20].

We let $\exists^*$FO Logic, $\forall^*$FO Logic, $\exists^*\forall^*$FO Logic, and $\forall^*\exists^*$FO Logic denote the fragments of FO Logic consisting of those formulas whose prenex normal form equivalent has the quantifier prefix of the respective form. We also call formulas in $\forall^*$FO Logic *purely universal*. The $\exists^*\forall^*$FO Logic fragment has also been called the Bernays-Schönfinkel-Ramsey (BSR) fragment or *effectively propositional logic* (EPR). It contains exactly the formulas of First-order Logic that have a quantifier prefix of $\exists^*\forall^*$ and do not contain function symbols that take parameters. Satisfiability of formulas in BSR is known to be decidable [20, 77].

2.1.3 Universal FO formulas

To reason about purely universal FO formulas, we often prefer to consider them in *normal form*. In particular, this allows us to expose all occurrences of literals of a particular relation A:

Lemma 2.1. *Every universal FO formula φ possibly containing occurrences of a relation A is equivalent to a formula*

$$E \wedge (\forall \bar{y}.F \vee A\bar{y}) \wedge (\forall \bar{y}'.G \vee \neg A\bar{y}') \wedge (\forall \bar{y}\bar{y}'.H \vee A\bar{y} \vee \neg A\bar{y}') \tag{2.1}$$

where E, F, G, H are universal formulas without A-literals.

The proof is by rewriting the formula:

Proof. W.l.o.g., we assume that $\varphi = \forall \bar{x}.\varphi'$ where φ' is quantifier-free and in conjunctive normal form. Let E, F', G', H' be the conjunctions of all clauses in φ' containing no occurrence of A, only positive, only negative and both positive and negative occurrences of A.

Each clause of the form $c' \vee A\bar{z}_1 \vee \ldots \vee A\bar{z}_k$ (where c' does not contain A-literals) is equivalent to

$$\forall \bar{y}.c' \vee \ (\bigwedge_{i=1}^{k} \bar{z}_i \neq \bar{y}) \vee A\bar{y}$$

where $\bar{z}_i \neq \bar{y}$ abbreviates the disjunction $\bigvee_{j=1}^{r} \bar{z}_{ij} \neq \bar{y}_j$ — given that $\bar{z}_i = z_{i1} \ldots \bar{z}_{ir}$.

Likewise, each clause of the form $c' \vee \neg A\bar{z}_1 \vee \ldots \vee \neg A\bar{z}_k$ (where again c' does not contain A-literals) is equivalent to

$$\forall \bar{y}'.c' \vee \ (\bigwedge_{i=1}^{k} \bar{z}_i \neq \bar{y}') \vee \neg A\bar{y}'$$

Finally, each clause of the form $c' \vee \neg A\bar{z}_1 \vee \ldots \vee \neg A\bar{z}_k \vee \neg A\bar{z}'_1 \vee \ldots \neg A\bar{z}'_l$ (where c' does not contain A-literals) is equivalent to

$$\forall \bar{y}\bar{y}'.c' \vee \ (\bigwedge_{i=1}^{k} \bar{z}_i \neq \bar{y}) \vee (\bigwedge_{i=1}^{l} \bar{z}'_i \neq \bar{y}') \ \vee A\bar{y} \vee \neg A\bar{y}'$$

Applying these equivalences to the clauses in the conjunctions in F', G', H', respectively, we arrive at conjunctions of clauses which all contain just the A-literal $A\bar{y}$, the A-literal $\neg A\bar{y}'$ or $A\bar{y} \vee \neg A\bar{y}'$, respectively. From these, the formulas F, G and H can be constructed by distributivity. □

In this transformation, we introduce disequalities between variables, and fresh auxiliary variables $\bar{y}$ and $\bar{y}'$ where the sequence $\bar{y}'$ is only required if both positive and negative literals of relation A occur within the same clause. In case these are missing, we say that the formula is in *simple* normal form.

Definition 2.2. *A FO formula φ possibly containing occurrences of a relation A is in* simple normal form*, if it can be written as:*

$$E \wedge (\forall \bar{y}.F \vee A\bar{y}) \wedge (\forall \bar{y}'.G \vee \neg A\bar{y}') \tag{2.2}$$

where E, F, G are universal formulas without A-literals.

The given normal form for formulas is closely related to the *Eliminationshauptform* proposed by Behmann [15] for monadic formulas — with the notable difference that we allow clauses containing occurrences both of positive and negative A-literals.

2.2 Introduced Concepts

$\exists, \forall$	Quantifiers
$\wedge, \vee, \neg, \rightarrow, \leftrightarrow, =$	Boolean connectives
$x, y, z, \ldots$	Variables for domain elements
$\bar{x}, \bar{y}, \bar{z}, \ldots$	Variables for tuples of domain elements
$x : T$	Variable x of sort T
R	Variable for predicates
φ	Variable for formulas
$fv(\varphi)$	Free variables of φ
$qr(\varphi)$	Quantifier rank (depth) of φ
$\Sigma(S, \mathcal{C}, \mathcal{R}, ar)$	Variable for signatures, consisting of sets of sorts, constants, relations and an arity function
$\mathcal{R}$	Set of predicates
$\mathcal{C}$	Set of constants
S	Set of sorts (types)
$ar(R)$	Arity of relations (predicates),functions
$\mathcal{V}_s$	Set of variable symbols of sort s
$\mathcal{V}$	Set of all variable symbols
U_s	Domain of s, set of elements in the universe of sort s
U	Universe, union of all domains
$sort(t)$	Sort of term t
s	Variable for structures, providing universe and interpretations of relations and constants for a signature
R^s, x^s	Interpretation of predicate R, variable or constant x in structure s
ν	Variable for valuations
$\nu[\bar{x} \mapsto \bar{d}]$	Update the value of $\bar{x}$ in valuation ν to $\bar{d}$
$s, \nu \models \varphi$	φ holds on structure s and valuation ν

Chapter 3

First-order Transition Systems

Contents

3 First-order Transition Systems

In this chapter, we introduce our model of multi-agent systems in which multiple actors or *agents* act in parallel and with differing knowledge to affect parts of the global state. A workflow system describes a *template* of a system, that any particular set of agents can then execute together. In the real world, this template can then be implemented in software, e.g. as a website the actors can interact with.

In Section 3.1, we define a modeling language for *workflow systems* that we use to model the interaction parts of real-world systems such as the mentioned conference management system or network protocols, where the actors are themselves automated systems.

To formally reason about workflow systems we introduce the underlying formal model of *First-order Transition Systems* in Section 3.2, together with an embedding of the specification language for workflow systems (Section 3.2.2) as well as an embedding of the RML language from [75] (Section 3.2.4) into FO Transition Systems.

3.1 The Workflow Language

We use *workflow systems* to model the interactions of multiple agents with a system, where agent interactions are recorded by relations. Here, we define a language to specify workflows. Updates of the relations describing the state are organized into *blocks*. These describe operations that a subset of the agents can choose to execute to change the contents of the relations. The most basic construct in the description of workflows is a parameterized guarded update operation to some relation. Such an update is meant to simultaneously be executed for all tuples satisfying the given guard. Some of these updates may also be *optional*, i.e. may also be omitted for some of the tuples that satisfy the guard. A block is made of several statements which add (or remove) specific tuples from a given relation depending on a guard clause.

Example 3.1.

A block consisting of one statement that says "Everyone that does not have a conflict of interest with a given paper may be assigned to review that paper." is specified as follows:

forall x: *PCMember*, p: *Paper* **may**. $\neg \mathit{Conflict}(x, p) \rightarrow \mathit{Assign} \mathrel{+}= (x, p)$

Workflows are defined over signatures $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$, with $\mathcal{R} = \mathcal{R}_{state} \cup \mathcal{R}_{input}$ (with $\mathcal{R}_{state}$ and $\mathcal{R}_{input}$ being disjoint sets of symbols). Symbols in $\mathcal{R}$ are possible relation symbols, where the symbols in $\mathcal{R}_{state}$ denote *state relations* which are updatable and the symbols in $\mathcal{R}_{input}$ denote non-updatable *input relations* that contain input from the environment to the workflow. S is the set of sorts individual variables can range over,

$\mathcal{C}$ is a set of constant terms and *ar* the respective arity function. A workflow is then produced by the following grammar:

$$
\begin{array}{lll}
w & ::= & block; w \mid \epsilon \mid \\
 & & \textbf{loop (*)}\ \{w\} \mid \textbf{choose}\ w\ \textbf{or}\ w \qquad // \quad \text{workflow program} \\
block & ::= & \textbf{forall}\ x_0 : s_0, \ldots, x_k : s_k.\{stmts\} \\
 & \mid & \textbf{forall}\ x_0 : s_0, \ldots, x_k : s_k.\ \textbf{may}\ \{stmts\} \qquad // \quad \text{block} \\
stmts & ::= & \varphi \rightarrow R \mathrel{+}= (t_1, \ldots, t_n); stmts \\
 & \mid & \varphi \rightarrow R \mathrel{-}= (t_1, \ldots, t_n); stmts \\
 & \mid & \epsilon \qquad // \quad \text{updates}
\end{array}
$$

Here, terms $t_1, \ldots, t_n$ are either agent variables $x_0, \ldots, x_k$ of sorts $s_0, \ldots, s_k$, where x_i ranges over variables in $\mathcal{V}_{s_i}$ or constant values in $\mathcal{C}$. R ranges over the predicate symbols in $\mathcal{R}$ and φ ranges over first-order formulas over the signature Σ. For a statement $\varphi \rightarrow R \pm= (t_1, \ldots, t_k)$, we require that $R \in \mathcal{R}_{state}$ and $fv(\varphi) \cup \{t_1, \ldots, t_k\} \subseteq \bar{x}$, where $\bar{x}$ is the sequence of variables appearing in the **forall** construct of the block that contains the statement. We also require the sorts of $(t_1, \ldots, t_k)$ to match the sorts of the relation to be updated, i.e. $(sort(t_1), \ldots, sort(t_k)) = ar(R)$.

The constructs **loop (*)** and **choose** specify the control flow. Loops, as given by **loop (*)** may be executed arbitrarily often before terminating (even 0 times), or not terminate at all. Nondeterministic choice constructs **choose** execute either the left or the right branch. For update statements, R denotes the predicate symbol to be updated and φ is the *guard* clause that needs to be met before performing the update. If the guard is not met, no update occurs. In order to specify deterministic/nondeterministic behavior, we use two different kinds of statements. In a normal block, all agents execute the block, i.e., the listed sequence of guarded updates. In a **may** block, only a subset of tuples of agents executes the block. This serves the intuition that agents may choose if they want to be included in the update or not (for example by clicking a button on a webpage).

Example 3.2.

Workflows can be used to model possible interactions between agents and a central authority to form a coherent system. As a very simple example, here is a possible hierarchical information flow model from a university setting:

```
1  forall p: Professor, s: Student, g: Grade
2      Oracle(p,s,g) → Grading += (p,s,g)
3  forall p: Professor, t: TA, s: Student, g: Grade may.
4      isSupervisor(p,t) ∧ visitsTutorialBy(s,t) ∧ Grading(p,s,g)
5          → Msg1 += (t,s,g)
6  forall t: TA, s: Student, g: Grade may.
7      Msg1(t,s,g) → Msg2 += (s,g)
```

Here, input relations to the system ($\mathcal{R}_{input}$) are *Oracle* (the exam grading process), *isSupervisor* and *visitsTutorialBy*. In the first step, professors enters grades from the exam grading process into the system. Professors can then choose to send messages to their own teaching assistants (TAs) and give them the grades of students in their group. This is recorded in the tuples of the relation Msg_1 which

```
1   // PC members may declare conflicts
2   forall x: PCMember, p: Paper may. true → Conf += (x, p)
3   // PC members are assigned to papers
4   forall x: PCMember, p: Paper may. ¬Conf(x, p) → Assign += (x, p)
5   // PC members write reviews for papers
6   forall x: PCMember, p: Paper, r: Report.
7           Assign(x, p) ∧ Oracle(x, p, r) → Review += (x, p, r)
8   // PC members discuss about the papers
9   loop (*) {
10          // PC members read all other reviews
11          forall x: PCMember, p: Paper, r: Report, y: PCMember may.
12                  Assign(x, p) ∧ Review(y, p, r) → Read += (x, p, r)
13          // PC members may rethink their reviews
14          forall x: PCMember, p: PCMember, r: Report may.
15                  Assign(x, p) ∧ Oracle(x, p, r) → Review += (x, p, r)
16  }
```

Figure 3.1: EasyChair-like workflow.

consist of the receiving TA and a pair of (student, grade). In the third step, the TAs can then pass the grades on to the students. This is again recorded in a relation, this time called Msg_2. It consists of the informed student and the grade he was given. Depending on the choices of the professors and TAs in the end result, every single student and his grade could be included in Msg_2 or any particular subset of students.

Example 3.3.

As a modestly involved running example, we introduce a workflow to model the paper reviewing and review updating of a conference management example like EasyChair in Figure 3.1.

In this workflow, all PC members are agents who interact with papers, reviews and with other PC members. In a first step, they can declare that they have a conflict of interest with some of the papers (line 2). Then, papers are assigned to reviewers as long as they have not declared a conflict with the respective papers (line 4). Reviewers are then required to write an initial review of their assigned papers (lines 6-7). Afterwards, the discussion phase starts. Here, all reviewers of a paper are shown all the reviews other people wrote for the same paper (lines 11-12). They can then alter their review based on the information they have seen (lines 14-15). This discussion phase continues for multiple turns until the PC chair ends the phase.

In this thesis, we will provide techniques to prove multi-agent workflows and similar systems have specific desirable properties. We will now give a formal semantics to our multi-agent workflows in terms of First-order signatures and evolving structures.

3.2 First-order Transition Systems

To give multi-agent systems like the one shown in Example 3.3 precise semantics, we first define the formal model of *First-order Transition Systems*. Afterwards, we show how to embed multi-agent systems in workflow syntax into this model.

Assume that we are given a signature $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$ with finite set $\mathcal{R} = \mathcal{R}_{state} \cup \mathcal{R}_{input}$ of predicates, $\mathcal{C}$ of constants together with a set S of variable sorts and an arity function ar.

A First-order (FO) Transition System $\mathcal{S}$ (over Σ) consists of a control-flow graph (V, E, v_0) underlying $\mathcal{S}$ where V is a finite set of program points, $v_0 \in V$ is the start point and E is a finite set of edges between vertices in V. Each edge of the graph is of the form (v, θ, v') where θ signifies how a first-order structure for program point v' is determined in terms of a first-order structure at program point v. Thus, θ is defined as a mapping which provides for each state predicate $R \in \mathcal{R}_{state}$ of arity r, a first-order formula $\theta(R\bar{y})$ for a dedicated well-sorted sequence of fresh FO variables $\bar{y} = y_1 \dots y_r$. Each formula $\theta(R\bar{y})$ may use FO quantification, equality or disequality literals as well as predicates from $\mathcal{R}_{state}$. Additionally, we allow occurrences of dedicated *input* predicates from $\mathcal{R}_{input}$. An important feature of input predicates (in contrast to state predicates) is that their interpretation is not constrained and its interpretations may vary over time. Thus, querying the same input relation multiple times may produce a different result every time.

A FO Transition System also includes an assertion Init over the relations in $\mathcal{R}_{state}$ and constants $\mathcal{C}$ that specifies which *initial* structures are valid starting points from which to execute the FO Transition System.

Example 3.4.

For the example given in Example 3.3, in every loop iteration, a different set of people may choose to update their reviews. Thus, both the interpretation of the **may** construct as well as the interpretation of the relation *Oracle* may change each iteration.

For convenience, we denote a substitution θ of predicates $R_1 \dots R_n$ with formulas $\varphi_1 \dots \varphi_n$ by

$$\theta = \{R_1\bar{y}_1 \coloneqq \varphi_1 \ \dots \ R_n\bar{y}_n \coloneqq \varphi_n\}$$

where each $\bar{y}_i$ is a list of pairwise distinct variables signifying the parameters of R_i and thus may occur freely in φ_i. We also will use $R\bar{y} \mathrel{+}= \varphi$ as a shorthand for $R\bar{y} \coloneqq R\bar{y} \vee \varphi$ as well as $R\bar{y} \mathrel{-}= \varphi$ as a shorthand for $R\bar{y} \coloneqq R\bar{y} \wedge \neg\varphi$.

Example 3.5.

The FO Transition System corresponding to the workflow specification from Figure 3.1 is given in Figure 3.2. The corresponding signature $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$ is shown in Figure 3.3.

Here, the state predicates in $\mathcal{R}_{state}$ are *Conflict*, *Assign*, *Review* and *Read*, while

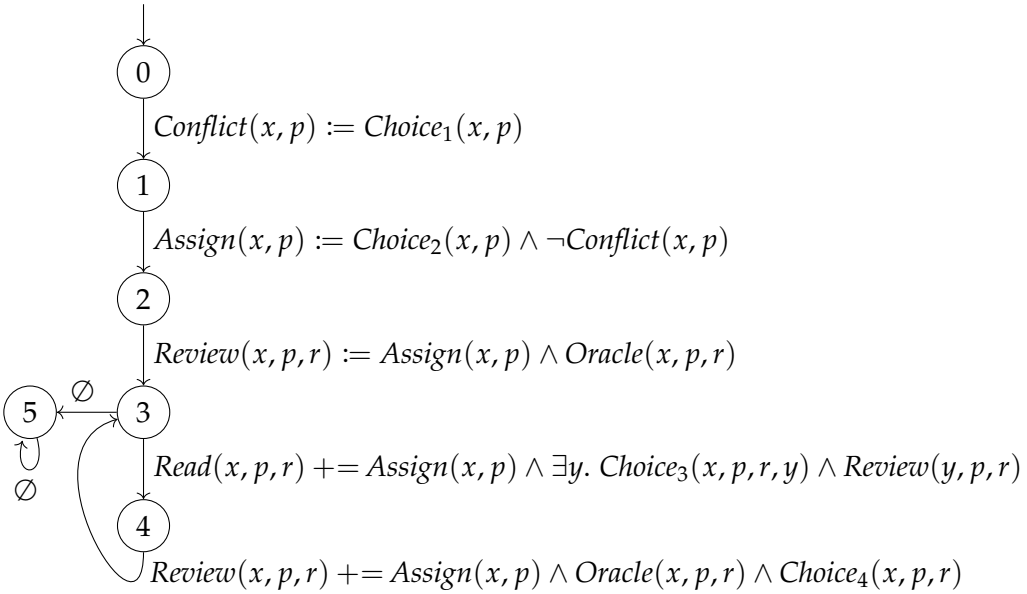


Figure 3.2: FO transition system for the running example

$$
\begin{aligned}
S &= \{PCMember, Paper, Report\},\\
\mathcal{C} &= \{\},\\
\mathcal{R}_{state} &= \{Conflict, Assign, Review, Read\},\\
\mathcal{R}_{input} &= \{Choice_1, Choice_2, Choice_3, Choice_4, Oracle\},\\
ar &= \left\{\begin{array}{rr}
Conflict: & (PCMember, Paper),\\
Assign: & (PCMember, Paper),\\
Review: & (PCMember, Paper, Report),\\
Read: & (PCMember, Paper, Report),\\
Choice_1: & (PCMember, Paper),\\
Choice_2: & (PCMember, Paper),\\
Choice_3: & (PCMember, Paper, Report, PCMember),\\
Choice_4: & (PCMember, Paper, Report),\\
Oracle: & (PCMember, Paper, Report)
\end{array}\right\}
\end{aligned}
$$

Figure 3.3: Signature of the running example

the input predicates $\mathcal{R}_{input}$ consist of $Choice_1, \ldots, Choice_4$ and *Oracle*. As there are no global constants, $\mathcal{C}$ is empty.

For the edge from node 2 to node 3, θ maps *Review* to the formula $Assign(y_1, y_2) \wedge Choice_2(y_1, y_2, y_3)$ and each other predicate R from $\mathcal{R}_{state}$ to itself (applied to a suitable list of formal parameters). Thus, θ maps, e.g., *Conflict* to $Conflict(y_1, y_2)$. For the edge from node 3 to 4, the given formula introduces existential quantification on y, as not all free variables in the corresponding workflow guard are bound by the tuple to be added to *Review*.

No node is explicitly marked as final node. We model termination of the workflow by going to state 5, from which the only outgoing edge does not change the state anymore.

A *path* in a FO Transition System is a path (sequence of edges) $e_0, e_1, \ldots$ through its corresponding control flow graph. The set of paths of a FO Transition System $\mathcal{S}$ is called $\mathsf{Paths}(\mathcal{S})$. We do not distinguish between finite and infinite paths in this thesis to ease notation. Embedding finite paths can be done as in Example 3.5 by adding a *stuttering* state that repeats the final state forever.

States and Transitions Let U be a universe specifying domains for all sorts in S. A *state* structure s (over U) contains interpretations for constants and state relations $\mathcal{C} \cup \mathcal{R}_{state}$ reached in some location $v \in V$ during an execution of the FO Transition System. To fully determine the successor structure of s, interpretations of the input predicates are needed. Thus, state structure s together with an *input* structure ω (over U) (containing interpretations for all relations in $\mathcal{R}_{input}$) determine the successor structure s'. Formally, for a structure s and an edge (v, θ, v') of the FO Transition System, there is a transition from (v, s) to (v', s') iff there exists an input structure ω such that for each predicate $R \in \mathcal{R}_{state}$ of arity $(s_1 \ldots s_n)$ together with a vector $\bar{y} = y_1 \ldots y_n$ (of sorts $ar(R)$) and an element $\bar{u} \in U_{s_1} \times \ldots \times U_{s_n}$

$$s', \nu[\bar{y} \mapsto \bar{u}] \models R\bar{y} \text{ iff } s \oplus \omega, \nu[\bar{y} \mapsto \bar{u}] \models \theta(R\bar{y})$$

holds. Additionally, s and s' need to have *rigid* constant interpretations, i.e. $c^s = c^{s'}$ for all $c \in \mathcal{C}$. Here, the operator "$\oplus$" is meant to combine the two structures into a single structure containing interpretations for both state predicates $\mathcal{R}_{state}$ as well as input predicates $\mathcal{R}_{input}$.

We call any pair (v, s) constructed in this way a *state* of the transition system and the set of all pairs $((v, s), (v', s'))$ constitute the *transition relation* $\Delta_{U,\nu}$ of $\mathcal{S}$ (relative to universe U and valuation ν). We call a state (v, s) *initial* if $v = v_0$ and s satisfies the initial condition, i.e. $s, \nu \models \mathsf{Init}$. A *trace* is a possibly infinite sequence

$$(v_0, s_0), (v_1, s_1) \ldots$$

such that (v_0, s_0) is initial and for each $i = 0, 1, \ldots$, $((v_i, s_i), (v_{i+1}, s_{i+1})) \in \Delta_{U,\nu}$ holds. We denote the set of all traces of a transition system $\mathcal{S}$ as $\mathsf{Traces}(\mathcal{S})$. The *path* $\pi(\tau)$ of a trace $\tau = (v_0, s_0), (v_1, s_1) \ldots$ is the projection to the sequence of taken edges, i.e. it is the sequence $e_0, e_1, \ldots$ connecting the sequence of nodes $v_0, v_1, \ldots$. In this thesis, we assume that this sequence is *unique*, i.e. between two nodes of V there exists at most *one* edge in E. This does not lose generality, as it is always possible to insert an auxiliary node on one of the edges.

We let $\sigma(n)$ denote the *n-th element* in a path σ, and let $\sigma[n,\infty] = \sigma(n)\sigma(n+1)\ldots$ denote the *n-th suffix* of σ while $\sigma[0,n]$ denotes the *n-th prefix* of σ. We also lift the suffix operation from paths to traces and define $\tau[n,\infty]$ be the n-suffix of a trace τ and $\tau[0,n]$ the n-th prefix.

Example 3.6.

Let us instantiate the Easychair example from Figure 3.2 for the universe U with

$$\begin{array}{lcl} U_{PCMember} & = & \{x_1, x_2\} \\ U_{Paper} & = & \{p_1, p_2\} \\ U_{Report} & = & \{r_1\} \end{array}$$

A possible structure s attainable at node 2 could have

$$\begin{array}{lcl} Conflict^s & = & \{(x_1, p_1)\} \\ Assign^s & = & \{(x_1, p_2), (x_2, p_1), (x_2, p_2)\} \\ Review^s & = & \emptyset \\ Read^s & = & \emptyset \end{array}$$

For the input structure ω with

$$Oracle^\omega = \{(x_2, p_2, r_1)\}$$

there is a transition to node 3 and a structure s' which is similar to s, but updated the relation *Review* to

$$Review^{s'} = \{(x_2, p_2, r_1)\}$$

with *Conflict* and *Assign* unchanged and *Read* still empty.

Fragments In this thesis, we sometimes consider simpler fragments of the full class of FO Transition Systems. We call a FO Transition System *acyclic* iff the underlying control flow graph is acyclic. In particular, all traces of an acyclic FO Transition System are *finite*. We call a FO Transition System *quantifier-free* iff for all edges (v, θ, v'), all formulas in θ used to update the state are quantifier-free.

Example 3.7.

The example shown in Example 3.5 is neither acyclic nor quantifier-free (because of the existential quantifier in the edge between nodes 3 and 4), but the FO Transition System version of the university example shown in Example 3.2 is both acyclic as well as quantifier-free.

3.2.1 Guards

First-order Transition Systems do not directly provide guards in the semantics. However, the formalism is rich enough to simulate guarded behavior: We add a fresh nullary predicate *error* with initial value *false* to $\mathcal{R}_{state}$ that is meant to track if the current trace has violated any guards. Given an edge (u, θ, v) in the original system that should be guarded by a FO formula φ, we transform it into an edge (u, θ', v) in the new system

where θ and θ' use the same transformations for all relations in $\mathcal{R}_{state}$ except *error* which θ' maps to $error \vee \neg\varphi$. Then, for any execution that violated any of the guards, *error* will be *true* from some point onward, which can be checked by verification conditions.

3.2.2 Embedding Workflows into First-order Transition Systems

The workflow language provides statements, e.g., of the form

---

forall $x : PCMember, p : Paper$ **may**. $\neg Conflict(x, p) \rightarrow Assign \mathrel{+}= (x, p)$

---

which is meant to add the pair (x, p) to the relation *Assign* whenever the condition $\neg Conflict(x, p)$ is satisfied and some implicit choice predicate $Choice(x, p)$ is satisfied whose elements are selected by the respective agent x. In this section, we will give precise semantics to workflows by embedding them into FO Transition Systems.

In the formalism of FO transition systems the substitution θ corresponding to this statement is the identity for all predicates R different from *Assign* and maps *Assign* to the formula

$$Assign(y_1, y_2) \vee Choice(y_1, y_2) \wedge \neg Conflict(y_1, y_2)$$

where $Choice \in \mathcal{R}_{input}$ and all other relations are in $\mathcal{R}_{state}$.

Workflows start in an initial state where all relations in $\mathcal{R}_{state}$ are empty, but with no assumptions on the input predicates. Thus, the initial condition Init is

$$\bigwedge_{R \in \mathcal{R}_{state}} \forall \bar{y}. \neg R\bar{y}$$

where $\bar{y}$ is a well-sorted tuple of correct arity for all R.

The control structures of workflows **loop (*)** and **choose** are used to specify the control flow graph of the workflow. It consists of edges (u, b, v) where u, v are nodes and b is a workflow block.

The corresponding FO Transition System uses the same graph structure and converts each edge (u, b, v) to an edge (u, θ, v). We now present how to define the substitution θ. We consider first a **may** block b, of the form

---

forall $\bar{x}{:}\,\bar{s}$ **may**. *stmts*

Since a **may** block executes only for a subset of all agents, we introduce the auxiliary predicate $Choice_i$ (where i is the index of block b) which collects all tuples for which the block should be executed. For each $R \in \mathcal{R}_{state}$, we let

$$\theta(R\bar{y}) := \begin{cases} R(\bar{y}) & \text{if } R \text{ not updated} \\ R(\bar{y}) \vee \exists \bar{x}{:}\,\bar{s}.\ \theta \wedge Choice_i(\bar{x}) \wedge (\bar{y} = \bar{u}) & \text{if } \bar{u} \text{ added to } R \\ R(\bar{y}) \wedge \neg\big(\exists \bar{x}{:}\,\bar{s}.\ \theta \wedge Choice_i(\bar{x}) \wedge (\bar{y} = \bar{u})\big) & \text{if } \bar{u} \text{ deleted from } R \end{cases}$$

where block b's statement that updates R has the form $\theta \rightarrow R \pm= \bar{u}$, i is the index of the block b in the linearisation of the workflow, and it is assumed that $\bar{x} \cap \bar{y} = \emptyset$. The definition of $\theta(R\bar{y})$ when b is a non-**may** block is similar, except that the $Choice_i(\bar{x})$ conjuncts are omitted.

If the same relation is modified multiple times in a block, the updates take place sequentially. This is expressed by building the formulas $\theta(R\bar{y})$ inductively, similarly

as above: $\theta(R\bar{y})$ up to statement j is obtained by replacing literals $R\bar{y}$ by $\theta(R\bar{y})$ up to statement $j-1$.

Example 3.8.

The execution semantics of the second block of the workflow from Example 3.3

forall x: *PCMember*, p: *Paper* **may**. $\neg Conf(x,p) \rightarrow Assign \mathrel{+}= (x,p)$

is given by the following mapping:

$$\begin{aligned} Assign(y_1,y_2) &:= Assign(y_1,y_2) \vee \\ &\quad (\exists x\text{: }PCMember, p\text{: }Paper.\ Choice_1(x,p) \wedge \\ &\quad\ \neg Conflict(x,p) \wedge y_1 = x \wedge y_2 = p)) \\ Conflict(y_1,y_2) &:= Conflict(y_1,y_2) \\ &\ldots \end{aligned}$$

The first formula of the above mapping can be rewritten into the logically equivalent formula

$$Assign(x,p) := Assign(x,p) \vee Choice_1(x,p) \wedge \neg Conflict(x,p)$$

by substituting x and p by y_1 and y_2 respectively, and then renaming y_1 and y_2 back to x and p. We note that this formula matches well the syntax of the second block in Example 3.3. The mentioned simplification cannot be performed in general, but only for a class of workflows, see Section 3.2.3. Following this transformation, the FO Transition System in Figure 3.2 is the embedding of the workflow from Example 3.3.

Since the interpretations of the $Choice_i$ predicate symbols used in the $\theta(R\bar{y})$ formulas is to be chosen by the participating agents, these are one part of the input relations $\mathcal{R}_{input}$. In addition, there is also *external* input, which is not provided by the agents themselves, which is also part of the input relations $\mathcal{R}_{input}$.

Example 3.9.

For the Easychair example from Example 3.3, the internal input relations for **may** blocks are $\{Choice_1, Choice_2, Choice_3, Choice_4\}$ while the external inputs are $\{Oracle\}$. Together, they form $\mathcal{R}_{input}$.

A **forall** $\bar{x}$ **may** block with statements of the form

$$\varphi_i \rightarrow R_i \pm= \bar{u}_i$$

can be seen as an abbreviation of a non-**may** block with statements of the form

$$Choice(\bar{x}) \wedge \varphi_i \rightarrow R_i \pm= \bar{u}_i$$

for some predicate symbol $Choice \notin \mathcal{R}_{state}$. Note also that for an atom $Oracle(\bar{t})$ occurring in some guard θ_i, the arity of *Oracle* need not be $|\bar{x}|$. We use the abbreviated **may** form to emphasize the subtle differences between the two kinds of non-workflow relations.

3.2.3 Non-omitting Workflows

We call a workflow *non-omitting* iff for each of its blocks

```
1  forall x̄: s̄ [may].
2          φ1 → R1 ±= ū1;
3          ...
4          φn → Rn ±= ūn;
```

we have $fv(\bar{u}_i) = \bar{x}$ and φ_i is quantifier-free, for each $i \in \{1, \ldots, n\}$.

For a non-omitting workflow w, we can replace all existentially quantified variables inside all formulas $\theta(R\bar{y})$ by their respective values, and remove the existential quantifiers. Thus, if for all block b, all guards of b are quantifier-free, all update formulas $\theta(R\bar{y})$ become quantifier-free for all $R \in \mathcal{R}_{state}$. Thus, non-omitting workflows with quantifier-free guards translate to quantifier-free FO Transition System.

3.2.4 Embedding RML into First-order Transition Systems

Generally, there is a growing interest in verifying infinite-state or parametric systems via a formalization in First-order Logic. An alternative formalization of such systems is the RML language introduced in [75], which has been used to model and check a variety of parameterized systems, most notably the Paxos network protocol [74]. RML is similar to the workflow language considered here in that its only data structure are (in this case finite) relations. It also contains control structures like nondeterministic choice, nondeterministic assignment, and an unbounded while loop **while** $*$ **do** that is similar to the **loop (*)** construct used in workflows. The local state is stored in first-order variables, first-order predicates and functions where typed update commands are provided to change the local state. The main difference between the workflow language and RML is that RML statements always manipulate just a single tuple in a predicate, while workflow blocks allow the manipulation of all tuples matching a particular formula.

In this section we show that our formalism of FO Transition Systems is general enough to also encode RML. In many cases FO transition systems are even more concise and allow for single statements where RML would need a loop construct.

Example 3.10.

Consider the RML program shown in Figure 3.4. The first five lines specify a signature Σ that consists of two relations R and S over a single sort where R is binary and S is unary. It then declares variables x and y which can be used throughout the program. The program then executes, consisting of a single loop that inserts some subset of tuples (x, y) into R where $S(y)$ holds.

A faithful encoding achieving the same result with a FO Transition System over signature Σ is shown in Figure 3.5. The additional predicate A is an input predicate in $\mathcal{R}_{input}$ that specifies for which (x, y) the RML loop should be executed.

Example 3.11.

For illustration, in Figure 3.6 we show a (a simplified version of) the running

```
1   sort s
2   relation R:s,s
3   relation S:s
4   variable x:s
5   variable y:s
6
7   while * do {
8           x = *;
9           y = *;
10          assume S(y);
11          R.insert(x, y);
12  }
```

Figure 3.4: Example of an RML program

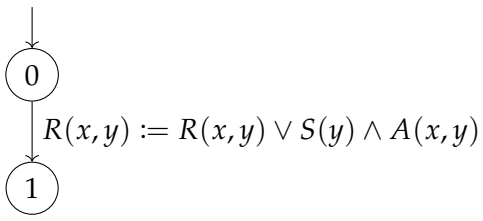
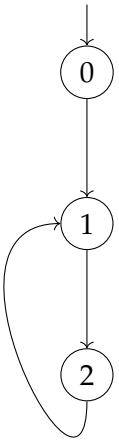


Figure 3.5: Encoding of a simple RML program



// n sends a message with its own id to m
$msg(id,m) := msg(id,m) \vee \exists n.\ id(id,n) \wedge next(n,m)$

// check all sent messages
// did someone get his own id?
$leader(n) := leader(n) \vee \exists id.\ msg(id,n) \wedge id(id,n)$

// send message on if it was higher than own id
$msg(id,m) := msg(id,m) \vee \exists n.\ msg(id,n) \wedge$
$\exists id'.\ id(id',n) \wedge \neg(id \leq id') \wedge next(n,m)$

Figure 3.6: Leader Election example

example from [75] given in the syntax of First-order Transition Systems.

It shows a network protocol for leader election for a set of nodes, organized in a ring topology. The ring is represented by the relation *next* with messages passed between different nodes, stored in the message relation *msg*. To elect a leader, all nodes send their own id to the next node in the ring. Whenever a node receives a message, it checks if it received its own id. If it did, it declares itself the leader. Otherwise, if it received an id higher than its own, it sends it on to the next node in the ring. All required properties for $\leq$, *next* and *id* are either purely universal formulas or purely existential and can be stated in Init.

The property that we are interested in is that at no point, there are two nodes that declared themselves as leaders — which can be stated by a universal formula.

RML is designed in such a way that all substitutions preserve Bernays-Schönfinkel formulas. Nonetheless, the language is Turing-complete [75]. In contrast to FO Transition Systems, it does not support input predicates that change over time.

All its features can also be encoded faithfully into FO Transition System without input predicates. To allow FO transition systems to only change relations by single tuples, we introduce an additional monadic relation X into $\mathcal{R}_{input}$ for every variable name x used in the RML program as well as the guard relation *error* mentioned before. We then translate the statement

$x = *;$

into the sequence of transitions shown in Figure 3.7.

Now $X(x)$ can be used in subsequent statements and constrains x to be a singular individual variable and all guards are in $\exists^*$FO Logic.

FO transition systems do not directly allow for functions f with arbitrary arity. These, however, can be encoded as relations R_f where the last component is the value of the function together with some universal axioms. To enable this, functions in RML come

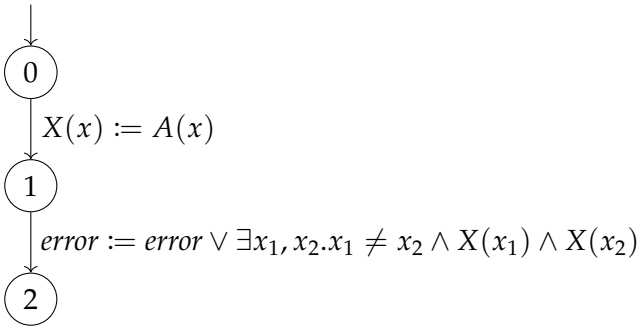


Figure 3.7: RML havoc statement as a FO Transition System

with a stratification that disallows nestings of function applications when the sort of the result is the same as one of the parameters, which excludes cyclic applications of functions. **assume** and **assert** statements, which are not directly supported by FO Transition Systems, can still be simulated by the same mechanism we used to encode guards.

3.3 Alternative Models

In terms of practical verification efforts, there has been a lot of recent interest in proving secrecy in various variants of parametric systems, oftentimes via a formalization in First-order Logic. In this chapter, we have argued for our formalism of FO Transition Systems. In the literature there are already several variants of multi-agent systems, each with their own pros and cons.

One such tool that we have already seen in Section 3.2.4 is the programming language RML [75], which we argue can be encoded into FO Transition Systems and thus profits from our results. Another recent verification approach has been made for the VeriCon system [72], which has been proposed for describing and verifying the semantics of controllers in software-defined networks. Both are based on evolving states of First-order relations and are thus close in spirit to FO Transition Systems. After presenting our verification results in Chapters 5 and 6, we will compare the technical results more closely.

Multi-agent systems are not always explicitly based on First-order Logic. A fairly recent line of work in that line is based on *population protocols*, where an unbounded number of memoryless agents move around a graph — every interaction they can take dictated by the current state they are in. For a survey, we refer to [8] as well as [19]. A population protocol can be seen as a particular class of FO Transition System where all involved relations are monadic and correspond to the states of the graph. However, FO Transition Systems allows to model structured processes rather than focusing on interactions between the participating agents. As we will show in Chapter 4, FO Transition Systems give more fine-grained control over the possible observations as well as the decisions of the participating agents.

Work on infinite-state systems also goes back a lot further. For example, a rich

body of work has been done on *abstract state machines* (ASMs) [44], i.e., state machines whose states are first-order structures. ASMs have been used to give comprehensive specifications of programming languages such as Prolog, C, and Java, and design languages, like UML and SDL (cf. [21]). A number of tools for the verification and validation of ASMs are available [22]. Known decidability results for ASMs are, however, based on strong restrictions such as sequential nullary ASMs [84]. As we will show in the upcoming chapters, FO Transition Systems seem to show more promise for automated analysis.

In AI, First-order Logic has a long tradition for representing potentially changing states of the environment [23], f.e. reachability problems that arise in robot planning. An example is GOLOG [62], which is a programming language based on a first-order language designed for representing dynamically changing systems. A GOLOG program specifies the behavior of the agents in the system. The program is then evaluated with a theorem prover, and thus assertions made in the program can be checked for validity.

Another type of multi-agent systems is represented by business processes. They are often described by BPMN diagrams [32] and formalized by Petri nets. A business process is a collection of activities, and a workflow thereof represents the flow of data items between activities. Activities are performed by users, who may need to synchronize on certain actions, but otherwise execute activities asynchronously. This is in contrast to our formalism, where steps are executed by a set of agents in parallel. Information flow in business processes has been considered, e.g., in [13]. The MASK framework for possibilistic information flow security [64] and a variant of the unwinding technique from [42] is used to prove that specifications satisfying particular constraints are safe. To our understanding, the approach is again not easily amenable to automation.

If we turn to web-based workflow management systems, there have been many different approaches of certifying secrecy for specific applications. For example, for the ConfiChair conference management system it was proven that the system provider cannot learn the contents of papers [7]. For CoCon, another conference management system, it was proven that various groups of users, such as authors, reviewers, and PC members cannot deduce certain content, such as reviews, unless certain declassification triggers, such as being a PC member without a conflict of interest, are met [57]. For the verification of security in an eHealth system, Bhardwaj and Prasad [17] assume that all agents are known at analysis time. Based on this information, the authors construct a dedicated security lattice and then apply techniques from universal information flow [51, 5].

These examples show that there is a lot of interest in automated and easily applicable tools that certify secrecy for such systems but fully automated approaches are still limited in the current state of the art.

3.4 Introduced Concepts

u, v, v'	Variable for locations in the control flow graph
$\mathcal{S}$	Variable for First-order Transition Systems
$\mathcal{R}_{state}$	State predicates, updated by the FO Transition System
$\mathcal{R}_{input}$	Input predicates, given by the outside world
θ	Variable for substitutions
$\theta(\varphi)$	Application of a substitution to a formula
$R\bar{y} := \varphi$	Shorthand for a substitution replacing literals $R\bar{y}$ by φ
$R\bar{y} \mathrel{+}= \varphi$, $R\bar{y} \mathrel{-}= \varphi$	Shorthands for $R\bar{y} := R\bar{y} \vee \varphi$ and $R\bar{y} := R\bar{y} \wedge \neg\varphi$
$s \oplus \omega$	Combination of state structure s and input structure ω
$\Delta_{U,\nu}$ of $\mathcal{S}$	Transition relation of $\mathcal{S}$ relative to universe U and valuation ν
$\mathsf{Paths}(\mathcal{S})$	Set of paths through the control flow graph of $\mathcal{S}$
σ	Variable for paths
$\sigma(n)$	n-th element of σ
$\sigma[0, n]$	Path/Trace prefix from 0 to n
$\sigma[n, \infty]$	Path/Trace suffix starting from n
(v, s)	State of a FO Transition System in node v with structure s
$(v_0, s_0), \ldots$	Trace of a FO Transition System through states $(v_0, s_0) \ldots$
$\mathsf{Traces}(\mathcal{S})$	Set of traces of $\mathcal{S}$
τ	Variable for traces
$\pi(\tau)$	Projection of a trace to its path

3.5 Conclusion

In this chapter, we have introduced a syntax for multi-agent workflow systems and have given them a formal semantics. For this, we have defined *First-order Transition Systems* which use First-order Logic formulas to describe how a system evolves over time. Given a particular universe consisting of elements of various sorts, the FO Transition System can be instantiated and executed to compute specific states where the relations are given specific interpretations over the set of elements of the universe. However, to prove that a given FO Transition System has specific properties, it is not enough to instantiate it for any number of fixed universes. Rather, we need to prove that all possible instantiations satisfy the property.

We have discussed how different languages and syntax features like multi-agent workflows, RML [73] or guards can be expressed by FO Transition Systems and introduced the classes of acyclic FO Transition System as well as quantifier-free FO Transition Systems, which can be written using entirely quantifier-free substitution formulas.

Chapter 4

Temporal Security Properties

Contents

4 Temporal Security Properties

In the last chapter, we introduced the formal model we use to express multi-agent workflow systems. We will now show how to formulate safety properties and secrecy requirements for FO Transition Systems. In our case, these are temporal properties such as "At every point in time during the execution of the transition system, formula φ holds." or "At some point in the future, formula φ will hold.".

Historically, temporal properties are expressed in *temporal logics* like Linear Temporal Logic (LTL) or CTL*. These logics are *propositional*. In FO Transition Systems, the global state is stored in relations which may be of unbounded size, which makes propositional logics ill suited to reason about FO Transition Systems.

Instead, we use First-order Linear Temporal Logic (FOLTL), an extension of Linear Temporal Logic (LTL) with First-order Quantifiers [48, 59]. We give a short introduction to FOLTL in Section 4.1. Our goal is to specify secrecy policies such as Noninterference. These are more complicated than linear temporal safety properties and are generally formalized as *hyperproperties* — properties that compare two or more possible executions of the given system.

To express these policies for FO Transition Systems, we extend First-order LTL to First-order HyperLTL in Section 4.2 and show how to embed this new logic back into First-order LTL. We then use First-order HyperLTL to formulate a version of Noninterference for FO Transition Systems with optional Declassification conditions and assumptions on the behavior of the participating agents in Section 4.3.

4.1 First-order Linear Temporal Logic

First-order Linear Temporal Logic (FOLTL) is an extension of standard Linear Temporal Logic with First-order Quantifiers. It has been studied in [48, 59] and describes how quantified properties evolve over time. For an overview, we refer to [47, Chapter 11].

Just as standard Linear Temporal Logic, FOLTL is used for statements over a timeline consisting of discrete timesteps. Instead of the atomic propositions used in LTL, FOLTL uses formulas in First-order Logic to make statements about specific points in time and uses temporal operators to connect them to a statement about the states that form a timeline.

Example 4.1.

First-order LTL connects First-order Logic and Linear Temporal Logic to form temporal first-order properties. Example properties in FOLTL include:

$$\mathsf{F}\,\mathsf{G}\,\forall x\colon \mathit{Student}, s\colon \mathit{Subject}.\ \mathit{Finished\_Studying}(x, s)$$

which specifies that from some point onward, every student will have finished studying all subjects (at which point he graduates). Temporal operators and

quantifiers can be nested freely to express more complicated properties:

$$\exists x\colon \mathit{Student}.\ \mathsf{F}\,\forall y\colon \mathit{Student}.\ (x \neq y) \rightarrow \mathit{Knows}(x, y)$$

This property expresses that there is a specific student x which will eventually get to know every other student.

FOLTL *formulas* over a signature $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$ are given by the grammar

$$\varphi ::= t = t' \mid R(t_1, \ldots, t_k) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\colon s.\ \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi$$

where s ranges over S, t, t' range over $\mathcal{V} \cup \mathcal{C}$ and are of the same sort, R ranges of $\mathcal{R}$ and for $ar(R) = s_0 \ldots s_k$ the t_i are of the correct sorts, i.e. t_i is of sort s_i for all i.

The symbols X and U denote the *Next* and *Until* LTL operators with the intuition that $\mathsf{X}\,\varphi$ means that φ should hold exactly one timestep from now, while $\varphi_1\,\mathsf{U}\,\varphi_2$ holds if φ_1 holds in every timestep until at some point in time φ_2 holds. As syntactic sugar, we again use the abbreviated operators from FO Logic $\wedge, \rightarrow, \leftrightarrow$ as well as the universal quantifier $\forall x : s$. Additionally, we use the derived temporal operators F (eventually), G (globally), W (weak until) and R (release):

$$\begin{array}{lcl} \mathsf{F}\,\varphi & := & \mathit{true}\,\mathsf{U}\,\varphi \\ \mathsf{G}\,\varphi & := & \neg\,\mathsf{F}\,\neg\varphi \\ \varphi_1\,\mathsf{W}\,\varphi_2 & := & (\varphi_1\,\mathsf{U}\,\varphi_2) \vee \mathsf{G}\,\varphi_1 \\ \varphi_1\,\mathsf{R}\,\varphi_2 & := & \neg(\neg\varphi_1\,\mathsf{U}\,\neg\psi_2) \end{array}$$

where $\mathit{true} := (c = c)$ for some $c \in \mathcal{C}$.

4.1.1 First-order LTL semantics

A *temporal structure* over signature Σ is a sequence $\bar{s} = (s_0, s_1, \ldots)$ of structures over Σ such that all structures s_i, with $i \geq 0$, have the same universe for all sorts $(U_s)_{s \in S}$, and rigid constant interpretations, i.e. $c^{s_i} = c^{s_0}$, for all $c \in \mathcal{C}$ and $i > 0$.

Let $\bar{s}$ be a temporal structure over the signature Σ, with $\bar{s} = (s_0, s_1, \ldots)$, φ a formula over Σ, and ν a valuation. We define the relation $\bar{s}, \nu \models \varphi$ inductively as follows:

$$\begin{array}{lll} \bar{s}, \nu \models t = t' & \text{iff} & \nu(t) = \nu(t') \\ \bar{s}, \nu \models R(\bar{t}) & \text{iff} & \nu(\bar{t}) \in R^{s_0} \\ \bar{s}, \nu \models \neg\psi & \text{iff} & \bar{s}, \nu \not\models \psi \\ \bar{s}, \nu \models \psi \vee \psi' & \text{iff} & \bar{s}, \nu \models \psi \text{ or } \bar{s}, \nu \models \psi' \\ \bar{s}, \nu \models \exists x.\ \psi & \text{iff} & \bar{s}, \nu[x \mapsto d] \models \psi, \text{ for some } d \in U \\ \bar{s}, \nu \models \mathsf{X}\,\psi & \text{iff} & \bar{s}[1, \infty], \nu \models \psi \\ \bar{s}, \nu \models \psi\,\mathsf{U}\,\psi' & \text{iff} & \text{for some } j \geq 0,\ \bar{s}[j, \infty], \nu \models \psi', \text{and} \\ & & \quad \bar{s}[k, \infty], \nu \models \psi, \text{ for all } k \text{ with } 0 \leq k < j \end{array}$$

A FOLTL formula φ is said to be *satisfiable* iff there exists a temporal structure $\bar{s}$ and a valuation ν s.t. $\bar{s}, \nu \models \varphi$. It is said to be *finitely satisfiable* iff there exists a temporal structure $\bar{s}$ over a finite universe U and a valuation ν s.t. $\bar{s}, \nu \models \varphi$. To omit parentheses, we assume that temporal operators bind stronger than quantifiers but weaker than the other boolean connectives. In analogue to First-order Logic, we also consider formulas in negation normal form, where R becomes a primitive operator instead of a derived one.

Example 4.2.

A simple safety property for the running conference management example from Example 3.3, we want to check that no papers are assigned to any PC member who declared a conflict of interest with that particular paper. This can be formalized as

$$\mathsf{G}\,\neg\exists x{:}\,\mathit{PCMember}, p{:}\,\mathit{Paper}.\ (\mathit{Conflict}(x,p) \wedge \mathit{Assign}(x,p))$$

4.1.2 First-order LTL Decidability

Since FOLTL subsumes First-order Logic (FO Logic), FOLTL is also undecidable in general. In this thesis, we consider formulas of a Bernays-Schönfinkel-Ramsey-like fragment of FOLTL, which we name $\exists^*FOLTL$.

To define $\exists^*$FOLTL we will consider the projection of a sorted FOLTL formula on a sort s, defined as the FOLTL formula obtained by removing all quantifiers and terms of sorts different from s. We refer to [1, Definition 17] for the formal definition of the projection, and only illustrate it here with an example:

Example 4.3.

Given the formula

$$\exists x{:}\,A.\,\forall y{:}\,B.\,\exists z{:}\,A.\,\neg(x = z) \wedge P(x,y) \wedge \mathsf{G}\,Q(y,z)$$

its projections Π_A, Π_B on the sorts A and B are the two formulas

$$\Pi_A : \exists x.\,\exists z.\,\neg(x = z) \wedge P_2(x) \wedge \mathsf{G}\,Q_1(z) \qquad \Pi_B : \forall y.\,P_1(y) \wedge \mathsf{G}\,Q_2(y)$$

The $\exists^*$FOLTL fragment of sorted FOLTL consists of those closed formulas φ in negation normal form such that, for each sort s, the projection of φ on s is a formula of the form

$$\exists x_1, \ldots, x_k.\,\varphi'_s$$

with $k \geq 0$ and φ'_s a FOLTL formula containing no existential quantifiers. This definition extends the definition of the Bernays-Schönfinkel-Ramsey-like fragments in [1, 70] from FO Logic to FOLTL,[1] and of the Bernays-Schönfinkel-Ramsey-like fragment in [59] from unsorted FOLTL[2] to sorted FOLTL. Note that the previous example formula in Example 4.3 is in $\exists^*$FOLTL.

To bring an arbitrary FOLTL formula into the mentioned form, we can use the standard transformations that put a FO Logic formula into prenex normal form, as well as the following equivalences to move existential quantifiers outside of temporal operators:

$$\varphi\,\mathsf{U}\,\exists x.\psi \equiv \exists x.\,(\varphi\,\mathsf{U}\,\psi) \quad \text{and} \quad (\exists x.\psi)\,\mathsf{R}\,\varphi \equiv \exists x.\,(\psi\,\mathsf{R}\,\varphi),$$

assuming that x does not occur free in φ. Note that in particular we have that $\mathsf{F}\,\exists x.\varphi \equiv \exists x.\,\mathsf{F}\,\varphi$. However, the previous equivalences cannot be generalized. For instance,

[1] The decidable FO Logic fragments in [1, 70] are larger than the projection of the $\exists^*$FOLTL fragment to sorted FO Logic, as they also consider function symbols.

[2] Unsorted FOLTL can be seen as sorted FOLTL with exactly one sort.

existential quantifiers cannot, in general, be moved over the G operator. Intuitively, $\mathsf{G}\,\exists x.\ \varphi$ means that "for all time points t, there exists an x such that φ holds at t". Thus, in contrast to FO Logic, not all FOLTL formulas can be put in prenex normal form.

Theorem 4.1 ($\exists^*$FOLTL Decidability)**.** *The following statements hold:*

1. *Checking satisfiability of a formula in $\exists^*$FOLTL is equivalent to checking* finite *satisfiability of the same formula.*
2. *$\exists^*$FOLTL is decidable.*

Proof. This proof follows the reasoning for decidability of the Bernays-Schönfinkel-Ramsey fragment of FO Logic, see e.g. [20].

Consider a closed formula φ in $\exists^*$FOLTL. It has the form

$$Q_1 x_1 \colon s_1 \ldots Q_k x_k \colon s_k.\ \psi$$

where $k \geq 0$, $Q_1, \ldots, Q_k$ is a sequence of quantifiers, and ψ is an FOLTL formula in negation normal form containing no existential quantifiers. We group the sequence $Q_1, \ldots, Q_k$ of quantifiers into maximal subsequences of the form $\exists^*\forall^*$. We let n be the number of such subsequences, and let ψ_i be obtained from φ be removing the first i groups of quantifiers, for $0 \leq i \leq n$. Note that $\varphi_0 = \varphi$ and $\varphi_n = \psi$.

We iteratively transform the formula φ_0 into the formulas ψ_1 to ψ_n. We also build the sets D_s^i of constant symbols of sort s, for each sort s and each i with $1 \leq i \leq n$. Consider step i, with $1 \leq i \leq n$. For each sort s, we pick a set C_s^i of constant symbols whose cardinality is given by the number of existential quantifiers over the sort s in the i-th subsequence, and such that $C_s^i \cap C_s^j = \emptyset$ for any $0 < j < i$. We let $D_s^i = C_s^i \cup D_s^{i-1}$, where D_s^0 is a singleton containing some constant of sort s. For each sort s, and each variable x of sort s bound by an existential quantifier from the i-th group, we remove the existential quantifier and we instantiate x in ψ_i by a corresponding constant from C_s^i. In this way, all existentially quantified variables in ψ_i are instantiated. Next, starting from the top-most universal quantifier in ψ_i, we iteratively replace every subformula of the form $\forall y \colon s.\ \alpha$ by the finite conjunction over elements of D_s^i, namely, $\bigwedge_{d \in D_s^i} \alpha[d/y]$. Let ψ_{i+1} be the formula obtained in this manner. Finally, we replace all subformulas of the form $\forall y \colon s.\ \alpha$ in ψ_n (recall that ψ may have universal quantifiers) by the finite conjunction over elements of D_s^n, as above. Let ψ' be the formula obtained in this manner. It is easy to see that φ is satisfiable iff ψ' is satisfiable. Furthermore, it is also clear that ψ' is satisfiable iff it is finitely satisfiable, as we can pick $U_s = D_s^n$ as the (Herbrand) universes. Note that by construction the universe U_s is non-empty even when there is no existential quantifier over the sort s; in this case $C_s^i = \emptyset$, for all i with $1 \leq i \leq n$. This ends the proof of the first statement of the theorem.

For the second statement of the theorem, note that ψ' contains no quantifiers and its literals do not contain variables. We transform ψ' into an LTL formula by taking the disjunction over all combinations of equivalence relations over D_s^n (for each sort s) of the formulas obtained by replacing each predicate $R(d_1, \ldots, d_\ell)$ in ψ' with the atomic propositions $R_{(d'_1, \ldots, d'_\ell)}$, where d'_i is the representative of the equivalence class to which d_i belongs. Furthermore, equalities $a = b$ are replaced by *true* if a and b are in the same equivalence class and by *false* otherwise. Clearly, the thus obtained formula is equisatisfiable with ψ'. We can now conclude by noting that LTL satisfiability is decidable, see e.g. [83]. □

We also note that there are very few decidability results concerning FOLTL. Besides the $\exists^*$FOLTL fragment, the only other decidable fragment we are aware of is the monodic fragment [48], which requires that temporal subformulas have at most one free variable. As will become clear in the next section, this restriction is too strong for our purposes: we cannot encode FO transition systems by FOLTL formulas in this fragment.

4.2 First-order HyperLTL

HyperLTL [29] is a recent extension of linear-time temporal logic (LTL) with *trace variables* and *trace quantifiers*. HyperLTL can relate multiple execution traces and is thus well suited to express not only trace properties, but security policies like Noninterference or Observational Determinism [42], which are often hyperproperties [30].

Since HyperLTL was introduced as a propositional logic, it cannot express properties about systems with an unbounded number of agents. We now present *First-order* HyperLTL, which extends propositional HyperLTL with first-order quantifiers. In the following, we will refer to First-order HyperLTL simply as FO HyperLTL.

Example 4.4.

In the secrecy policy called "Observational Determinism" [92], the inputs and outputs of any given program are classified as low or high security with respect to the clearance level of some particular user with the intention that that user is able to observe the low security inputs/outputs but can not observe the high security ones.

The property states that, on any two program executions, if the low inputs are always the same, then the low outputs are also always the same. That is, from the point of view of a particular user, the observable behavior of the program is only determined by the observable inputs this particular user is cleared for. This property can be formalized by a FO HyperLTL formula quantifying over two paths π,π':

$$\forall \pi, \pi'. (\mathsf{G}\, \forall x. I_\pi(x) \leftrightarrow I_{\pi'}(x)) \rightarrow (\mathsf{G}\, \forall y. O_\pi(y) \leftrightarrow O_{\pi'}(y)),$$

where $I(x)$ denotes that x is a low input to the program, while $O(y)$ denotes that y is a low output.

4.2.1 First-order HyperLTL syntax.

Let $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$ be a signature, and let Π be a set of *trace variables* disjoint from the set $\mathcal{V}$ of first-order variables. We call $\mathcal{R}_\Pi = \{R_\pi \mid R \in \mathcal{R}, \pi \in \Pi\}$ the set of *indexed predicates*. Let $\Sigma' = (S, \mathcal{C}, \mathcal{R}_\Pi, ar')$ be the signature with $ar'(R_\pi) = ar(R)$, for any $R \in \mathcal{R}$ and $\pi \in \Pi$.

FO HyperLTL extends FOLTL as follows. FO HyperLTL *formulas* over Σ and Π are generated by the following grammar:

$$\psi ::= \exists \pi.\ \psi \mid \neg\psi \mid \varphi$$

where $\pi \in \Pi$ is a trace variable and φ is a FOLTL formula over Σ'. Universal trace quantification is defined as $\forall\pi.\varphi \equiv \neg\exists\pi.\neg\varphi$. FO HyperLTL formulas thus start with a prefix of trace quantifiers consisting of at least one quantifier and then continue with a subformula that contains only first-order quantifiers, no trace quantifiers.

4.2.2 First-order HyperLTL semantics

The *semantics* of an FO HyperLTL formula ψ is given with respect to a set $\mathcal{T}$ of temporal structures, a valuation $\alpha : \mathcal{V} \to U$ of the first-order variables, and a valuation $\beta : \Pi \to \mathcal{T}$ of the trace variables.

The *satisfaction* of a FO HyperLTL formula ψ, denoted by $\mathcal{T}, \alpha, \beta \models \psi$, is then defined as follows:

$$\begin{array}{lll} \mathcal{T}, \alpha, \beta \models \varphi & \text{iff} & \bar{\beta}, \alpha \models \varphi, \\ \mathcal{T}, \alpha, \beta \models \neg\psi & \text{iff} & \mathcal{T}, \alpha, \beta \not\models \psi, \\ \mathcal{T}, \alpha, \beta \models \exists\pi.\ \psi & \text{iff} & \mathcal{T}, \alpha, \beta[\pi \mapsto \bar{s}] \models \psi, \text{ for some } \bar{s} \in \mathcal{T}, \end{array}$$

where ψ is a FO HyperLTL formula, while φ is a FOLTL formula. Formulas φ are FOLTL formulas over indexed predicates R_π where π is a trace variable. We evaluate φ as an ordinary FOLTL formula over a *combined* temporal structure $\bar{\beta}$ that interprets an indexed predicate R_π by the interpretation of R in the temporal structure that β assigns to the trace variable π, i.e. $R_\pi^{\bar{\beta}_i} = R^{\beta(\pi)_i}$ for all timepoints i.

A FO HyperLTL formula without free first-order and trace variables is called *closed*. A closed FO HyperLTL formula ψ is *satisfiable* iff there exists a set $\mathcal{T}$ of temporal structures and valuations α and β s.t. $\mathcal{T}, \alpha, \beta \models \psi$. A closed formula ψ *holds* for a FO transition system $\mathcal{S}$, denoted by $\mathcal{S} \models \psi$, iff $\mathcal{T}, \alpha, \beta \models \psi$ for the empty assignments α and β and the set $\mathcal{T} = \mathsf{Traces}(\mathcal{S})$ of traces of the transition system.

Example 4.5.

In the running conference management example from Example 3.5, there exists a trace where from some point on, all papers are reviewed. Thus, this FO HyperLTL property holds for the corresponding transition system $\mathcal{S}$, i.e.

$$\mathcal{S} \models \exists\pi.\ \mathsf{F}\,\mathsf{G}\,\forall p{:}\mathit{Paper}.\ \exists x{:}\mathit{PCMember}, r{:}\mathit{Report}.\ \mathit{Review}(x, p, r)$$

FO HyperLTL formulas in which all trace quantifiers are universal are called *universal formulas*. Satisfaction of any FOLTL formula φ can be embedded into FO HyperLTL satisfaction as $\forall\pi.\ \varphi_\pi$ where φ_π is φ where all relation symbols have been indexed with π.

We refer to [29] for the formalization in HyperLTL of other hyperproperties.

4.2.3 First-order HyperLTL Decidability

Analogously to the FOLTL case we will consider a decidable fragment of FO HyperLTL, namely $\exists^*_\pi\forall^*_\pi\exists^*$FOLTL, which consists of all formulas of the form $\exists\pi_1, \ldots \pi_k.\ \forall\pi'_1 \ldots \pi'_\ell.\ \varphi$ with $k \geq 0$, $\ell \geq 0$, and φ a FOLTL formula in $\exists^*$FOLTL.

We first remark that by seeing trace variables as first-order variables of a new sort T — the trace sort, FO HyperLTL formulas can be faithfully encoded by FOLTL formulas.

By this we mean that, for any closed FO HyperLTL formula ψ, there is a closed FOLTL formula φ such that we can translate models of ψ into models of φ and vice-versa. The formula φ is obtained by replacing trace quantification $Q\pi$ with first-order quantification $Q\pi{:}\,T$, for $Q \in \{\exists, \forall\}$, and predicates $R_\pi(\bar{u})$ with predicates $R'(\pi, \bar{u})$. Note that ψ and φ are formulas over slightly different signatures. The translation between models is straightforward. For instance, if $\bar{s}$ is a temporal structure that satisfies φ, then the corresponding set $\mathcal{T}$ of temporal structures that satisfies ψ consists of temporal structures obtained by projecting a predicate's interpretation on the predicate's non-trace arguments, for each of the values of the trace universe U_T of $\bar{s}$, i.e. $\mathcal{T} = \{\bar{s}_t \mid t \in U_T\}$ and $R^{s_{t,i}} = \{\bar{a} \mid (t, \bar{a}) \in R'^{\bar{s}_i}\}$, for each $R \in \mathcal{R}$, $t \in U_T$, and $i \in \mathbb{N}$.

As a consequence of the previous discussion, and as a corollary of Theorem 4.1, we obtain the following results.

Theorem 4.2. *The following statements hold:*

1. *Every FO HyperLTL formula can be translated into an equi-satisfiable FOLTL formula.*
2. *Every $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL formula can be translated into an equi-satisfiable $\exists^*$FOLTL formula.*
3. *Satisfiability of formulas in $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL is decidable.*

Having introduced the logics we are going to use, we will now turn to the question of how to encode desirable secrecy properties of FO Transition System into FO HyperLTL.

4.3 Noninterference

There is a vast body of work on information flow policies and associated verification techniques. We mention Goguen and Meseguer's seminal work on *noninterference* [42], Zdancewic and Myer's *observational determinism* [92], Sutherland's *nondeducability* [85], and Halpern and O'Neill's *secrecy maintenance* [45] as representative examples. See Kanav *et al.* [57] for a recent overview with a detailed discussion of the most relevant notions for the verification of multi-agent systems.

Most secrecy properties are based on a classification of the inputs and outputs of a system into either *low*, meaning not confidential, or *high*, meaning highly confidential. In this thesis, we concentrate on the property of *noninterference*: A system has the noninterference property if in any pair of traces where the low inputs are the same, the low outputs are the same as well, regardless of the high inputs.

When we are interested in the noninterference property of a single agent, it is possible to model the low and high inputs and the low and high outputs of the system (as seen by the agent) using separate predicates, for example, as I_l, I_h, O_l, O_h, respectively. The basic ingredient to noninterference is observational determinism, which is expressed as the HyperLTL formula

$$\forall\pi.\forall\pi'.\ \mathsf{G}(I_{l,\pi} \leftrightarrow I_{l,\pi'}) \ \rightarrow\ \mathsf{G}\,(O_{l,\pi} \leftrightarrow O_{l,\pi'}),$$

which states that all traces π and π' that have the same low input I_l at all times, must also have the same low output O_l at all times. This formula compares two traces of the

given system and can be refuted by giving two finite prefixes of traces of the system that do not satisfy this property. Thus, it is called a 2-hypersafety property [29].

To adapt this property to the setting of FO Transition Systems, we need to specify how an agent interacts with the modeled system and reason about his knowledge and possible interactions. In a FO Transition System, the inputs or outputs of different agents may be collected in the same predicate. Formally, we classify all sorts into agent sorts and data sorts. Moreover, we assume that the arity $(s_1, s_2, \dots)$ of every relation $R \in \mathcal{R}_{state} \cup \mathcal{R}_{low} \cup \mathcal{R}_{high}$ is non-nullary and is such that s_1 is an agent sort. This restriction, while not strictly necessary, allows us to present the results in a much cleaner way.

Example 4.6.

In the running conference management example from Example 3.5, members of the PC can use a conference management system to specify conflicts, read the reviews/reports that other members have provided, provide their own reviews, etc. As an example property, we will formalize that no member of the PC gains any information about papers that he declared a conflict of interest with. In this case, the low outputs observed by a PC member x consist of the pairs (x, p, r) for some paper p in the *Read* relation and of the tuples (x, p, r) for some *Review* relation. Low input is provided by the agents in the form of tuples of the *Choice* predicates that begin with x. Additionally, the system has high input in the form of the *Oracle* predicate which is used to produce the reviews.

Generalizing from the example, we assume there is one or more predicates of the form $O_l(x, \bar{y})$, modeling *low output* observed by the agents from the system, and one or more predicates of the form $I_l(x, \bar{y})$ modeling *low inputs* provided by the agents to the system. An output is *observable* by agent x whenever x occurs in the first position of the tuple. Likewise, an input is *controllable* by agent x whenever x occurs in the first position of the tuple. The remaining components of the tuple are denoted by the vector $\bar{y} = y_1, y_2, \dots$. For our examples, we will use predicates named *Choice* to indicate controllable input relations.

4.3.1 Declassification

Declassification [79] becomes necessary when the functionality of the system makes it unavoidable that some information is leaked. In the conference management example, a PC member x is supposed to read the reviews of the papers assigned to x. This is legitimate as long as x has not declared a conflict of interest with those papers. We assume that, in addition to the input and output predicates, there is a *declassification condition* $D_{I_h} x \bar{y}$, which indicates that agent x is allowed to learn about the high input $I_h \bar{y}$. We call a first-order transition system *noninterferent* with respect to traces π,π', iff for any agent a, his observations do not depend on the non-declassified inputs which are classified as "high" for a in any way. It is expressed as the FO HyperLTL formula

$$noninterferent_{\pi,\pi'} \quad := \quad \forall a.\ \mathsf{G}\, same\_high\_inputs_{\pi,\pi'}(a) \rightarrow \mathsf{G}\, same\_observations_{\pi,\pi'}(a)$$

where the individual parts are

$$\begin{aligned} \mathit{same\_observations}_{\pi,\pi'}(a) &:= \bigwedge_{R \in \mathcal{R}_{state}} \left(\forall \bar{z}.\, R_\pi a \bar{z} \leftrightarrow R_{\pi'} a \bar{z}\right) \\ \mathit{same\_high\_inputs}_{\pi,\pi'}(a) &:= \bigwedge_{I \in \mathcal{R}_{high}} \forall \bar{z}.\, \left(D_{I\pi} a \bar{z} \vee D_{I\pi} a \bar{z} \rightarrow (I_\pi \bar{z} \leftrightarrow I_{\pi'} \bar{z})\right) \end{aligned}$$

It expresses that on all pairs of traces where the low inputs are the same and, additionally, the high inputs are the same whenever the declassification condition is true on one of the traces, the low outputs must be the same.

Example 4.7.

In the conference management example, the input relation *Oracle* is used to produce reviews. It contains tuples of the form

$$\mathit{Oracle}(x\colon \mathit{PCMember}, p\colon \mathit{Paper}, r\colon \mathit{Report})$$

In our example, any agent a is allowed to learn about reviews for papers he has not declared conflict of interest with. Thus,

$$D_{\mathit{Oracle}}(a, x, p, r) := \neg \mathit{Conflict}(a, p)$$

We can specify the information flow policy that an agent should not receive information regarding conflicting papers as a noninterference property:

$$\begin{aligned} \mathit{same\_high\_inputs}_{\pi,\pi'}(a) &:= \forall x, p, r.\ \ (\neg \mathit{Conflict}_\pi(a, p) \vee \neg \mathit{Conflict}_{\pi'}(a, p)) \\ &\qquad\qquad \rightarrow (\mathit{Oracle}_\pi(x, p, r) \leftrightarrow \mathit{Oracle}_{\pi'}(x, p, r)) \\ \mathit{same\_observations}_{\pi,\pi'}(a) &:= \begin{array}{l} \forall p.\ \mathit{Conflict}_\pi(a, p) \leftrightarrow \mathit{Conflict}_{\pi'}(a, p) \wedge \\ \forall p.\ \mathit{Assign}_\pi(a, p) \leftrightarrow \mathit{Assign}_{\pi'}(a, p) \wedge \\ \forall p, r.\ \mathit{Review}_\pi(a, p, r) \leftrightarrow \mathit{Review}_{\pi'}(a, p, r) \wedge \\ \forall p, r.\ \mathit{Read}_\pi(a, p, r) \leftrightarrow \mathit{Read}_{\pi'}(a, p, r) \end{array} \end{aligned}$$

For this particular example, we are interested in information leaking via the *Read* relation, so for this case, we can simplify *same_observations* to only some specific relations.

$$\mathit{same\_observations}_{\pi,\pi'}(a) := \forall p, r.\ \mathit{Read}_\pi(a, p, r) \leftrightarrow \mathit{Read}_{\pi'}(a, p, r)$$

4.3.2 Causality assumptions on agents

In the conference management example from Example 3.3, it is easy to see that no PC member can directly read the reviews of papers where a conflict of interest has been declared: the PC member can only read a review if the PC member was assigned to the paper, which, in turn, can only happen if no conflict of interest was declared. It is much more difficult to rule out an indirect flow of information via reviews posted by another PC member. So far, neither the description of the FO Transition System, nor the HyperLTL specification would prevent other PC members to add arbitrarily add any review to the *Read* relation, even if it contains information the PC member has no access to. To enforce that agents can only spread the knowledge they actually have access to,

we must make assumptions about the possible behaviors of the *other* agents. For any agent, we consider two kinds of possible behavior — Agents that either *stubbornly* make the same choices, independently of their observations; or they let their choices depend on previous observations, which we call acting *causally*.

Stubborn agents. A radical restriction on the behavior of the other agents is to require that they always, stubbornly, produce the same input, independently of their own observations. An agent a acts *stubbornly* on traces π, π' if his inputs never differ:

$$stubborn_{\pi,\pi'}(a) \quad := \quad \mathsf{G}\, same\_low\_inputs_{\pi,\pi'}(a)$$

with

$$same\_low\_inputs_{\pi,\pi'}(a) \quad := \quad \bigwedge_{I \in \mathcal{R}_{low}} \left(\forall \bar{z}.\, I_\pi a \bar{z} \leftrightarrow I_{\pi'} a \bar{z}\right)$$

Causal agents. A more natural restriction on the behavior of the other agents is to require that they act causally, i.e., they only provide different inputs if they, themselves, have previously observed a difference in the states of the two traces. This forces agents to be deterministic — choosing to always give the same low input whenever they made the same observations. The *causality* of an agent a w.r.t. traces π, π' is formally described by:

$$causal_{\pi,\pi'}(a) \quad := \quad same\_low\_inputs_{\pi,\pi'}(a)\, \mathsf{W}\, \neg same\_observations_{\pi,\pi'}(a)$$

The property states that, for all agents a the inputs provided on two traces are the same — but only *until* a difference in the outputs observed by a occurs. This does not restrain the behavior of agents *after* they got access to a single bit of differentiating information. From then on, it allows them to actively spread information to anyone.

Example 4.8.

In the conference management example, stubbornness for traces π, π' uses the HyperLTL formula

$$\begin{aligned} same\_low\_inputs_{\pi,\pi'}(a) := \; & \forall p.\, Choice_{1,\pi}(a,p) \leftrightarrow Choice_{1,\pi'}(a,p) \wedge \\ & \forall p.\, Choice_{2,\pi}(a,p) \leftrightarrow Choice_{2,\pi'}(a,p) \wedge \\ & \forall p, r.\, Choice_{3,\pi}(a,p,r) \leftrightarrow Choice_{3,\pi'}(a,p,r) \wedge \end{aligned}$$

The requirement of causality for π, π' additionally uses *same_observations* as given in Example 4.7.

Since causality does not constrain behavior after a difference has been observed, any agent acting stubbornly also fits the requirements to act causally, i.e. the causality assumption allows for more behaviors. Therefore, the most general agent model is when each agent is causal, while the most restrictive model is when each agent is stubborn.

We thus instantiate a formula $agent\_model_{\pi,\pi'}$ to use in the complete formalization of noninterference for first-order transition systems. It can be instantiated with one of the following formulas:

$$\begin{aligned} agent\_model^{(c,k)}_{\pi,\pi'} &:= \exists a_1, \ldots, a_k.\, \left(\textstyle\bigwedge_{i=1}^{k} causal_{\pi,\pi'}(a_i)\right) \wedge \\ &\qquad \left(\forall a.\, \left(\textstyle\bigwedge_{i=1}^{k} a \neq a_i\right) \rightarrow stubborn_{\pi,\pi'}(a)\right) \\ agent\_model^{(s)}_{\pi,\pi'} &:= \forall a.\, stubborn_{\pi,\pi'}(a) \qquad\qquad = agent\_model^{(c,0)}_{\pi,\pi'} \\ agent\_model^{(c)}_{\pi,\pi'} &:= \forall a.\, causal_{\pi,\pi'}(a) \end{aligned}$$

where $k \geq 0$.

To wrap up, *Noninterference with Declassification and Agent model* (NDA) for First-order Transition Systems expresses that — given that all agents act according to their behavior assumptions — no agent (indirectly) observes differences in the high input on two traces π and π' in the transition system. It is expressed by the FO HyperLTL formula

$$\forall \pi, \pi'.\ agent\_model_{\pi,\pi'} \rightarrow noninterferent_{\pi,\pi'} \tag{4.1}$$

4.3.3 Control Flow

In this thesis, we assume that the control flow is chosen by external factors. This could be because a certain time has passed or the conference organizers determine that discussions about papers have come to a close. This also implies that the current location in the control flow graph does *not* depend on any secret information.

We further assume the current location in the control flow graph is not observable by any agent. This simplifies reasoning as we only have to compare the relation contents. In case we want agents to be able to observe the current location, it is possible to add fresh relations R_v of arity 1 for each location v of the underlying graph. Then, for each edge (u, θ, v), we transform it to an edge (u, θ', v) where θ' behaves like θ, but additionally moves all agents from relation R_u to R_v by setting

$$R_u(x) := false$$
$$R_v(x) := true$$

Using this transformation, all agents can always observe the current node of the control flow graph.

4.4 Related Specification Languages

Our approach is based on a First-order extension of the temporal logic HyperLTL [29]. HyperLTL has been applied in the verification of hardware systems, such as an Ethernet controller [38]. Expressing trace properties with sorted FOLTL has been initiated already in the work of Manna and Pnueli [63] and logic-based approaches are now standard in the verification of such properties. However, our extension First-order HyperLTL is to our knowledge the first temporal logic suited for the specification of information flow in systems with arbitrarily many participating agents. Logic-based approaches for non-trace properties are less common and include the ones based on epistemic temporal logics [36], SecLTL [33] or the polyadic modal μ-calculus [6]

4.5 Introduced Concepts

$\mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{U}, \mathsf{W}, \mathsf{R}$	Temporal Operators
$\bar{s}$	Variable for temporal structures
$\bar{s}, \nu \models \varphi$	Temporal formula φ holds on temporal structure $\bar{s}$ and valuation ν
$\forall \pi.\ \varphi, \exists \pi.\ \varphi$	Quantification over traces
π	Variable for traces
R_π	Predicate R on trace π
$\mathcal{T}, \alpha, \beta \models \psi$	Temporal Hyperproperty ψ holds on set of temporal structures $\mathcal{T}$ with first-order valuation α and trace valuation β
$\mathcal{S} \models \psi$	ψ holds on FO Transition System $\mathcal{S}$
$\mathcal{R}_{high}, \mathcal{R}_{low}$	Partition of $\mathcal{R}_{input}$ into predicates which contain secret information and which do not

4.6 Conclusion

In this chapter we discussed the temporal logics First-order LTL and First-order HyperLTL. We used FOLTL to specify temporal safety and liveness properties for FO Transition Systems and showed that the fragment $\exists^*$FOLTL is decidable. We then extended it to FO HyperLTL, an extension of FOLTL with path quantifiers that allows us to specify secrecy policies for FO Transition Systems. We proved that FO HyperLTL can be embedded back into FOLTL and lifted our results on decidability to FO HyperLTL. We then used FO HyperLTL to formalize Noninterference together with state-based Declassification conditions. To model the knowledge that agents possess and the interactions they can take we added one of several agent models: Agents can either behave *stubbornly* and only pass on information that they got explicitly communicated via the system or they can act *causally* where they try to spread information as soon as they are able to distinguish between different scenarios.

Chapter 5

Verification of Temporal Properties

Contents

5 Verification of Temporal Properties

In the last decades, a lot of research has been done on the verification problem for finite-state systems and their temporal properties. A prominent line of research has introduced *Model Checking*, which explores the state space of a given system and proves that all possible executions adhere to a specification, often given in Linear Temporal Logic. For an overview, we refer to [9]. This line of research has been a huge success and is now being incorporated into the design process of software and hardware across the world [26].

In this chapter, we present approaches on how to verify that a given FO Transition System satisfies a given temporal hyperproperty in FO HyperLTL by applying ideas and approaches from the area of Model Checking.

We consider the two main flavors of Model Checking: *Bounded* Model Checking — which is based on exploring executions of a given system up to a given bounded length and is often based on SAT-Solving [18], as well as *Symbolic* Model Checking, which systematically explores all states of the given finite-state system and is often based on Binary Decision Diagrams [3]. Both variants have had tremendous success and have spawned strong tool support in the form of PRISM [60], SPIN [50] and others.

In this chapter, we adapt both variants to our setting of FO Transition Systems. In Section 5.1, we present an approach based on *Bounded Symbolic Model Checking*, which takes into account all traces of the given FO Transition System that do not exceed a given length. This is achieved by encoding a bounded version of the Transition System as well as a bounded version of the property to be verified into First-order Satisfiability. In Section 5.2, we overcome the limitation of bounded traces and present an approach for general Symbolic Model Checking. Here, we encode the complete FO Transition System as well as the property to be verified into First-order LTL Satisfiability. We then identify that the verification of Noninterference stays decidable for quantifier-free FO Transition Systems and prove undecidability in case this restriction is violated.

5.1 Bounded Symbolic Model Checking

We start by presenting a bounded verification for FO Transition Systems based on the ideas from *Bounded Model Checking*. Bounded Model Checking is based on the exploration of traces of a system up to a fixed bound. For finite-state systems, this can be encoded into satisfiability queries on propositional logic formulas. As the state of a FO Transition System is encoded in FO Logic, we will use FO satisfiability instead of propositional satisfiability. Thus, our approach reduces the violation of an FO HyperLTL specification formula on the prefix of a trace of the given FO Transition System to the satisfiability of a formula in FO Logic. Since the state space of a particular FO Transition System is still unbounded, we thus arrive at a bounded symbolic model checking approach — bounded in the length of the trace, but symbolic in the exploration of the state space.

5.1.1 Bounded Satisfaction

Bounded model checking is based on a restricted notion of FO HyperLTL satisfaction where only trace prefixes of length n, for some fixed bound n are considered. Let $\mathcal{T}$ be a set of traces, $\alpha : \mathcal{V} \to U$ a valuation of the free first-order variables, and $\beta : \Pi \to \mathcal{T}$ a valuation of the trace variables.

The *n-bounded satisfaction* of a HyperLTL formula ψ, denoted by $\mathcal{T}, \alpha, \beta \models^n \psi$, is then defined as follows:

$$
\begin{array}{lll}
\mathcal{T}, \alpha, \beta \models^n \exists \pi.\ \psi & \text{iff} & \mathcal{T}, \alpha, \beta[\pi \mapsto \bar{s}] \models^n \psi, \text{ for some } \bar{s} \in \mathcal{T}, \\
\mathcal{T}, \alpha, \beta \models^n \neg \varphi & \text{iff} & \mathcal{T}, \alpha, \beta \not\models^n \varphi, \\
\mathcal{T}, \alpha, \beta \models^n R\bar{y} & \text{iff} & s, \alpha \models R\bar{y}, \\
\mathcal{T}, \alpha, \beta \models^n \varphi_1 \vee \varphi_2 & \text{iff} & \mathcal{T}, \alpha, \beta \models^n \varphi_1 \text{ or } \mathcal{T}, \alpha, \beta \models^n \varphi_2, \\
\mathcal{T}, \alpha, \beta \models^n \exists x.\ \varphi & \text{iff} & \exists a \in U.\ \alpha[x \mapsto a], \beta \models^n \varphi, \\
\mathcal{T}, \alpha, \beta \models^n \mathsf{X}\, \varphi & \text{iff} & \mathcal{T}, \alpha, \beta[1, \infty] \models^{n-1} \varphi, \text{ for } n > 0, \\
\mathcal{T}, \alpha, \beta \models^0 \mathsf{X}\, \varphi & \text{iff} & \bot, \\
\mathcal{T}, \alpha, \beta \models^n \varphi_1 \,\mathsf{U}\, \varphi_2 & \text{iff} & \exists i \geq 0 :\ \mathcal{T}, \alpha, \beta[i, \infty] \models^{n-i} \varphi_2 \text{ and} \\
 & & \forall 0 \leq j < i :\ \mathcal{T}, \alpha, \beta[j, \infty] \models^{n-j} \varphi_1, \text{ for } n > 0, \\
\mathcal{T}, \alpha, \beta \models^0 \varphi_1 \,\mathsf{U}\, \varphi_2 & \text{iff} & \mathcal{T}, \alpha, \beta[i, \infty] \models^0 \varphi_2,
\end{array}
$$

where φ, φ_1, and φ_2 are FO HyperLTL formulas, and the structure s contains the interpretations of all predicates R in the first structure of the trace named by the variable π in the trace valuation β, i.e. $R^s_\pi = R^{(\beta(\pi))_0}$.

A closed formula ψ is *n-bounded satisfied* by a FO Transition System $\mathcal{S}$, denoted by $\mathcal{S} \models^n \psi$, iff $\mathsf{Traces}(\mathcal{S}), \alpha, \beta \models^n \psi$ for the empty assignments α and β. For acyclic FO Transition Systems, satisfaction and bounded satisfaction coincide.

Theorem 5.1. *Let $\mathcal{S}$ be a FO Transition System where all paths are of length at most n. For all HyperLTL formulas ψ, it holds that $\mathcal{S} \models \psi$ iff $\mathcal{S} \models^n \psi$.*

5.1.2 Bounded Model Checking

We now translate a FO Transition System $\mathcal{S}$ together with a given FO HyperLTL formula ψ with only universal trace quantifiers for a given bound n into a formula $\Psi^n_{\mathcal{S},\neg\psi}$ of First-order Logic such that $\Psi^n_{\mathcal{S},\neg\psi}$ is satisfiable iff $\mathcal{S} \not\models^n \psi$. Since ψ is universal, its negation is of the form $\exists \pi_1, \ldots, \pi_k.\ \varphi$, where φ does not contain any trace quantifiers. In $\Psi^n_{\mathcal{S},\neg\psi}$, we use for every predicate $R \in \mathcal{R}$ several copies $R_{\pi,l}$, one per trace variable $\pi \in \{\pi_1, \ldots, \pi_k\}$ and time point l, $0 \leq l \leq n$. The formula $\Psi^n_{\mathcal{S},\neg\psi} = [\![\mathcal{S}]\!]^n \wedge [\![\varphi]\!]^n_0$ is a conjunction of two formulas in FO logic, the first being the unfolding $[\![\mathcal{S}]\!]^n$ of the transition system $\mathcal{S}$ and the second being the unfolding $[\![\varphi]\!]^n_0$ of the FO HyperLTL formula φ.

For a FO Transition System $\mathcal{S}$ and a bound $n \geq 0$ let Paths^n be the (finite) set of paths of $\mathcal{S}$ of length exactly n. For formulas φ such as the initial condition Init or the update formula $\theta(R\bar{y})$ to a relation R, we use $\varphi_{\pi,l}$ to mean φ where all relations P have been replaced with $P_{\pi,l}$. The *unfolding* $[\![\mathcal{S}]\!]^n$ of the transition system is defined as follows:

$$
[\![\mathcal{S}]\!]^n = \bigwedge_{(v_0,\theta_0,v_1),\ldots,(v_{n-1},\theta_{n-1},v_n) \in \mathsf{Paths}^n(\mathcal{S})} \ \bigwedge_{\pi \in \{\pi_1,\ldots,\pi_k\}} \mathsf{Init}_{\pi,0} \wedge \bigwedge_{l=0}^{n-1} \bigwedge_{R \in \mathcal{R}_{state}} R_{\pi,l+1} \leftrightarrow (R\theta_l)_{\pi,l}
$$

As discussed in Section 4.3.3, this definition of unfolding uses the same path through $\mathcal{S}$ to instantiate all trace variables and does not use an exponential blowup by comparing different paths. This relates to the fact that the control flow in a FO Transition Systems is assumed to be under external control.

For a FO HyperLTL formula φ without trace quantifiers and a bound $n \geq 0$, the *unfolding* $[\![\varphi]\!]_l^n$ is defined as follows:

$$\begin{array}{lcl}
[\![\neg\varphi]\!]_l^n & = & \neg[\![\varphi]\!]_l^n, \\
[\![\psi]\!]_l^n & = & \psi_l, \\
[\![\varphi_1 \wedge \varphi_2]\!]_l^n & = & [\![\varphi_1]\!]_l^n \wedge [\![\varphi_2]\!]_l^n, \\
[\![\exists x.\ \varphi]\!]_l^n & = & \exists x.\ [\![\varphi]\!]_l^n, \\
[\![\mathsf{X}\,\varphi]\!]_l^n & = & [\![\varphi]\!]_{l+1}^{n-1} \text{ for } n > 0, \\
[\![\mathsf{X}\,\varphi]\!]_l^0 & = & \bot, \\
[\![\varphi_1 \,\mathsf{U}\, \varphi_2]\!]_l^n & = & [\![\varphi_2]\!]_l^n \vee ([\![\varphi_1]\!]_l^n \wedge [\![\varphi_1 \,\mathsf{U}\, \varphi_2]\!]_{l+1}^{n-1}) \text{ for } n > 0, \\
[\![\varphi_1 \,\mathsf{U}\, \varphi_2]\!]_l^0 & = & [\![\varphi_2]\!]_l^0
\end{array}$$

where φ, φ_1, and φ_2 are FO HyperLTL formulas, ψ is a formula in FO logic over indexed predicates $P_{i,\pi}$ and ψ_l is the same formula with all occurrences of an indexed predicate $P_{i,\pi}$ replaced by the predicate $P_{i,\pi,l}$, i.e. we use the interpretation of predicate P_i on trace π at timepoint l.

Theorem 5.2. *For a FO Transition System S, a FO HyperLTL formula ψ and a bound $n \geq 0$, it holds that $S \models^n \psi$ iff $[\![\mathcal{S}]\!]^n \wedge \neg[\![\varphi]\!]_0^n$ is unsatisfiable.*

Combining Theorems 5.1 and 5.2, we obtain the corollary that bounded model checking is a complete verification technique for acyclic FO Transition Systems.

Corollary 5.3. *Let $\mathcal{S}$ be an acyclic FO Transition System with paths of length at most n. For all FO HyperLTL formulas ψ, it holds that $\mathcal{S} \models \psi$ iff $[\![\mathcal{S}]\!]^n \wedge \neg[\![\varphi]\!]_0^n$ is unsatisfiable.*

5.1.3 Decidability

We now identify cases where the satisfiability of the predicate logic formulas constructed by the verification method of the previous section are decidable.

Theorem 5.4. *Consider a FO HyperLTL property φ on a quantifier-free FO Transition System $\mathcal{S}$ with a given bound of n under the assumption that all agents behave* stubborn. *Assume that*

$$\forall \pi_1, \ldots, \pi_r.\ \mathit{agent\_model}^{(s)} \rightarrow \varphi$$

denotes a FO HyperLTL formula where $[\![\neg\varphi]\!]_0^n$ is a sorted Bernays-Schönfinkel formula, i.e., the prenex form of ψ' has a quantifier sequence of the form $\exists^*\forall^*$. Then it is decidable whether $\mathcal{S} \models^n \forall \pi_1, \ldots, \pi_r.\ \mathit{agent\_model}^{(s)} \rightarrow \varphi$.

Proof. To deal with the assumption about the agent model, we transform the unfolding $[\![\mathcal{S}]\!]_0^n$ to include the restrictions on it. As all agents are stubborn, the low input predicates $\mathit{Choice}_{i,\pi_j}$ are equivalent for $j = 1, \ldots, r$. Accordingly, we may replace all $\mathit{Choice}_{i,\pi_j}(s_1, \ldots, s_k)$ with $\mathit{Choice}_{i,\pi_1}(s_1, \ldots, s_k)$ in μ without changing the satisfiability of μ under the assumption of $\mathit{agent\_model}^{(s)}$. This allows us to remove the agent model from the verification condition.

It remains to check if $\mathcal{S} \models_n \varphi$, which is done by checking the satisfiability of $[\![\mathcal{S}]\!]^n \wedge \neg[\![\varphi]\!]_0^n$. Since $\mathcal{S}$ is quantifier-free, all formulas $\theta(R\bar{y})$ along any path are quantifier-free. Thus, the unfolding $[\![\mathcal{S}]\!]^n$ is also quantifier-free. The satisfiability of $\neg[\![\varphi]\!]_0^n$ is also decidable — which thus implies the the theorem. □

Theorem 5.4 can be extended to more general classes of FO Transition Systems, given that the predicates $R_{\pi_j,l}$ occur only positively or only negatively in ψ'. Noninterference amounts to stating that (under certain conditions) no distinction is observable between some $R_{\pi_j,l}$ and $R_{\pi_{j'},l}$. Logically, indistinguishability is expressed by equivalence, which thus results in both positive and negative occurrences of the predicates in question. However, this is useful for the encoding of the guards, as the fresh *error* relation only occurs positively.

Theorem 5.5. *Consider a FO HyperLTL property φ on a quantifier-free FO Transition System $\mathcal{S}$ with a bound n under the assumption that all agents behave* causal. *Assume that*

$$\forall \pi_1, \ldots, \pi_r.\ agent\_model^{(c)} \rightarrow \varphi$$

is a temporal formula where the prenex form of $\psi' \equiv [\![\neg\varphi]\!]_0^n$ is purely existential. Then it is decidable whether $\mathcal{S} \models^n \forall \pi_1, \ldots, \pi_r.\ agent\_model^{(c)} \rightarrow \varphi$.

Proof. The argument for causal agents is somewhat more complicated and accordingly leads to decidability only for a smaller fragment of HyperLTL formulas. Removal of the temporal operators and skolemization of the formula $agent\_model^{(c)}$ describing causality yields a conjunction of clauses of the form

$$S(x, f_1(x), \ldots, f_r(x)) \vee Choice_{i,\pi_{j_1}}(x, \bar{z}) \vee \neg Choice_{i,\pi_j}(x, \bar{z}) \tag{5.1}$$

or

$$S(x, f_1(x), \ldots, f_r(x)) \vee \neg Choice_{i,\pi_{j_2}}(x, \bar{z}) \vee Choice_{i,\pi_j}(x, \bar{z}) \tag{5.2}$$

for $j_1, j_2 < j$, where the disjunction S refers to predicates which depend on $Choice_{i',\_}$ predicates from $\mathcal{R}_{low}$ for $i' < i$ only. In order to perform ordered resolution, we put an ordering upon predicates so that $Choice_{i,\pi_j}$ receives a higher priority than $Choice_{i',\pi_{j'}}$ if $i' < i$ or, if $i = i'$, $j' < j$. Moreover all predicates in S have lower priorities than the *Choice* predicates. Accordingly, the highest priority literal in each clause of $agent\_model^{(c)}$ contains all free variables of the clause.

Let us first consider the case $r = 2$. Then resolution of two clauses with a positive and negative occurrence of the same highest-priority literal will result in a tautology and therefore is useless. According to our assumption on ψ', the clauses obtained from $[\![\neg\varphi]\!]_0^n$ are all closed. Resolution of such a clause with a clause of $agent\_model^{(c)}$ for some $Choice_{i,\pi_2}$ will again return a closed formula. We obtain a set of new closed clauses with occurrences of predicates $Choice_{i',\pi_{j'}}, i' < i$, only. As a consequence, for every i, there is a bounded number of new clauses derivable by means of clauses from $agent\_model^{(c)}$ with highest priority predicate $Choice_{i,\pi_2}$. Altogether, we therefore obtain only a bounded number of closed clauses which are derivable by means of ordered resolution. Hence, it is decidable whether a contradiction is derivable or not. This concludes the proof.

The argument for $r > 2$ is similar, only that resolution of any two such clauses originating from $agent\_model^{(c)}$ with $j_1 \neq j_2$ upon the literal $Choice_{i,\pi_j}(x, \bar{z}))$ will again

result in a clause of the given form. In particular, no further literals are introduced. Therefore, saturation of $agent\_model^{(c)}$ by ordered resolution will eventually terminate. Then the argument for termination proceeds analogously to the case $r = 2$ where $agent\_model^{(c)}$ is replaced with the saturation of $agent\_model^{(c)}$. □

Theorem 5.5 can be extended to formulas φ where $\bar{\psi}'$ obtained from $[\![\neg\varphi]\!]_0^n$ is a Bernays-Schönfinkel formula at least in restricted cases.

Consider the clauses of the form Equations (5.1) and (5.2) as obtained from $agent\_model^{(c)}$ after skolemization. In case that the disjunction S is empty, we call the corresponding clause *simple*, otherwise *complex*. Now assume that complex clauses from the saturation of $agent\_model^{(c)}$ are always resolved with clauses (originating from the skolemization of Ψ') upon a *closed* literal. Then the same argument as in the proof of Theorem 5.5 applies to show that saturation by resolution will eventually terminate.

Applying these results to the Noninterference formula from the previous chapter, we get:

Corollary 5.6. *Given a quantifier-free FO Transition System $\mathcal{S}$ and a bound n it is decidable whether Noninterference with Declassification and Agent Model holds on all traces of $\mathcal{S}$ up to length n, given that all declassification conditions are quantifier-free for agent models $agent\_model^{(s)}$ as well as $agent\_model^{(c)}$.*

We have now presented a sound verification approach for quantifier-free FO Transition Systems based on Bounded Model Checking that is able to prove Noninterference properties for traces of bounded length. For FO Transition System that are both quantifier-free and acyclic, this technique is complete, as all traces are of bounded length.

In the next section, we have a look at cyclic FO Transition System that inherently exhibit traces of unbounded length and present a technique that is still able to prove that all traces of the system satisfy a given property.

5.2 Symbolic Model Checking

In this section, we will introduce methods for symbolic model checking for FO Transition Systems.

For finite-state systems, symbolic Model Checking encodes the state space of the system into propositional logic and often uses binary decision diagrams or similar technology to reason about reachable parts of the state space. To apply this approach to our setting of states in FO Logic, we instead use FOLTL to encode the state space. Compared to the bounded approach, instead of unrolling the transition relation, we will encode the complete semantics of a given FO Transition System into FOLTL, which allows us to encode possibly infinite traces and prove safety for all traces, not just the set of bounded prefixes of traces.

5.2.1 Formalizing FO Transition Systems in FOLTL

Control Flow Given a FO Transition System $\mathcal{S}$, we consider its underlying control flow graph. Let (V, E) be the CFG of $\mathcal{S}$. We abuse notation, and we use a fresh proposition

(i.e. nullary predicate) v to express whether the current state is in node $v \in V$. We denote by $\mathcal{R}_{cfg}$ the set of these predicate symbols. The transition relation of the control flow graph of $\mathcal{S}$ is then expressed by the formula:

$$\texttt{cfg}(\mathcal{S}) := \mathsf{G}\ \Big(\bigwedge_{u \in V} u \rightarrow \mathsf{X}\Big(\bigvee_{v \in succ(u)} v\Big)\Big)$$

where $succ(u)$ is the set of the successors of the node u in the CFG. Furthermore, a trace can never be in two states at once:

$$\texttt{sanity}(\mathcal{S}) := \mathsf{G}\ \Big(\bigwedge_{u,v \in V, u \neq v} \neg(u \wedge v)\Big)$$

Termination In our case, traces need not be finite, and thus the execution of $\mathcal{S}$ need not terminate. As discussed, terminating behavior could be imposed by adding a fresh node v_{final} whose only outgoing edge is a self-loop which does not change any relation in $\mathcal{R}_{state}$. Then by requiring that v_{final} is eventually reached the trace effectively terminates by repeating the last state forever. The FOLTL formula ensuring this is $\mathsf{F}\, v_{final}$. We do not impose this requirement here.

Initial state Every trace executes sequentially, starting in the initial node v_0 of the CFG of $\mathcal{S}$. There, the initial condition Init of $\mathcal{S}$ holds. Formally, this is expressed by the following formula:

$$\texttt{init}(\mathcal{S}) := v_0 \wedge \mathsf{Init}$$

The sanity requirement then forces all other v_l different from v_0 to be false.

State Transformation An edge in the execution of a trace determines the new state of all relations in $\mathcal{R}_{state}$. We formalize this execution by characterizing with an FOLTL formula the interpretation of a predicate at the next time point based on the interpretation of the relations at the current time point.

For an edge $e = (u, \theta, v)$ and every relation symbol $R \in \mathcal{R}_{state}$, $R(\bar{y})$ holds after execution of e iff $\theta(R\bar{y})$ holds before the execution of e, with $|\bar{y}| = |ar(R)|$. A single edge (u, θ, v) is then represented by the following formula:

$$\texttt{exec}_{\mathcal{S}}(u, \theta, v) := (u \wedge \mathsf{X}\, v) \rightarrow \bigwedge_{R \in \mathcal{R}_{state}} \big(\forall \bar{y}.(\mathsf{X}\, R\bar{y}) \leftrightarrow \theta(R\bar{y})\big)$$

The execution semantics of the FO Transition System is then captured by the following formula:

$$\texttt{exec}(\mathcal{S}) := \mathsf{G} \bigwedge_{(u,\theta,v) \in E} \texttt{exec}_{\mathcal{S}}(u, \theta, v).$$

where E is the set of edges of $\mathcal{S}$.

Complete Specification The complete specification $\texttt{ts}(\mathcal{S})$ of the FO Transition System is a conjunction of the several parts described previously — the control flow graph, the initial state, and the semantics of the transitions between time points.

$$\texttt{ts}(\mathcal{S}) := \texttt{cfg}(\mathcal{S}) \wedge \texttt{sanity}(\mathcal{S}) \wedge \texttt{init}(\mathcal{S}) \wedge \texttt{exec}(\mathcal{S})$$

Note that the formula $\texttt{ts}(\mathcal{S})$ is expressed over the signature Σ' obtained from Σ by extending it with relation symbols $v \in \mathcal{R}_{cfg}$.

By construction, for any given FO Transition System $\mathcal{S}$ over a signature Σ, its traces $\mathsf{Traces}(\mathcal{S})$ consist of all temporal structures $\bar{s}$ over Σ' that satisfy $\texttt{ts}(\mathcal{S})$. We have:

Theorem 5.7. *Given a FO Transition System $\mathcal{S}$, a FOLTL formula $\varphi_{\mathcal{S}}$ can be built in polynomial time so that for every FOLTL formula φ, it holds that $\mathcal{S} \models \varphi$ iff $\varphi_{\mathcal{S}} \rightarrow \varphi$ is valid.*

In fact, as such $\varphi_{\mathcal{S}}$ we may choose the formula $\mathtt{ts}(\mathcal{S})$.

5.2.2 FOLTL Model Checking

For a quantifier-free FO Transition System $\mathcal{S}$, all update formulas are quantifier-free. For an initial condition Init which is in $\exists^*$FOLTL, It follows that the formula $\mathtt{ts}(\mathcal{S})$ can be brought into the $\exists^*$FOLTL fragment. Additionally, $\mathtt{ts}(\mathcal{S})$ contains no existential quantifiers and all its universal quantifiers are either not under a temporal operator (in the case of the $\mathtt{init}(\mathcal{S})$ subformula) or under the G temporal operator (in the case of the $\mathtt{exec}(\mathcal{S})$ subformula). Therefore the simplified $\mathtt{ts}(\mathcal{S})$ formula can be put in prenex normal form having a quantifier prefix consisting of only universal quantifiers. As a side remark, this means that $\neg\mathtt{ts}(\mathcal{S})$ can also be brought into $\exists^*$FOLTL.

Theorem 5.8. *It is decidable for a quantifier-free FO Transition System $\mathcal{S}$ and a formula φ in $\exists^*$FOLTL whether or not $\mathcal{S} \models \neg\varphi$ holds.*

This means that if the set of all *bad* behaviors can be expressed by a formula φ in $\exists^*$FOLTL, then absence of bad behaviors can be checked for quantifier-free FO Transition Systems. The theorem follows from Theorems 5.7 and 5.8. Indeed, it is sufficient to check whether $\mathtt{ts}(\mathcal{S}) \wedge \varphi$ is unsatisfiable. This can be done, since both conjuncts can be brought into $\exists^*$FOLTL, and thus the conjunction itself too.

The following theorem shows that the decidability result from Theorem 5.8 cannot be lifted to arbitrary FO Transition Systems.

Theorem 5.9. *It is undecidable for a general FO Transition System $\mathcal{S}$ and a formula φ in $\exists^*$FOLTL whether or not $\mathcal{S} \models \neg\varphi$ holds.*

Proof. We prove the theorem by reducing the *periodic tiling problem* to our setting. The tiling problem was first mentioned in [87] and has first been shown undecidable by Berger in [16]. Its closely related variant, the periodic tiling problems has also been proven undecidable by multiple authors — for an overview see [54]. We now briefly recall the definition of the problem.

Given a set of k tile types $T = \{T_i \mid 0 \leq i < k\}$ as well as horizontal and vertical compatibility relations x-comp $\subseteq T \times T$ and y-comp $\subseteq T \times T$, a *tiling* is a function $f(x, y) : \mathbb{N} \times \mathbb{N} \rightarrow T$ such that whenever two tiles are adjacent, they have to respect the compatibility relations:

$$\forall x, y.\ x\text{-comp}(f(x, y), f(x + 1, y)),$$
$$\forall x, y.\ y\text{-comp}(f(x, y), f(x, y + 1)).$$

A tiling is *periodic* if there exist horizontal and vertical periods p_x and p_y, such that

$$\forall x, y.\ f(x, y) = f(x + p_x, y),$$
$$\forall x, y.\ f(x, y) = f(x, y + p_y).$$

The periodic tiling problem is to find out for a given set of tile types and its compatibility relations, if there exists a periodic tiling. [1]

We will now proceed to show how to encode this problem into a FO Transition System. To find a periodic tiling, it is enough to find the periods p_x, p_y, and the values $f(x,y)$ for $0 \leq x < p_x$ and $0 \leq y < p_y$, such that they are also compatible at borders:

$$\forall y.\ x\text{-comp}(f(p_x,y), f(0,y)),$$
$$\forall x.\ y\text{-comp}(f(x,p_y), f(x,0)).$$

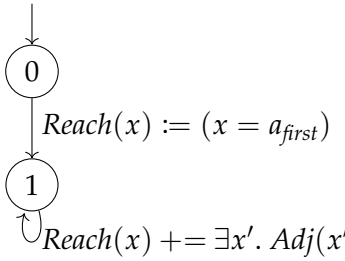
We thus interpret a periodic tiling as a table with rows referring to points on the y-axis and columns referring to points on the x-axis.

We build next a FO Transition System $\mathcal{S}$ and a formula φ such that $\mathcal{S} \models \varphi$ iff there is a periodic tiling for $(T, x\text{-comp}, y\text{-comp})$. We use the following signature:

$$\Sigma = \big(\{A\}, \{a_{first}, a_{last}\}, \{Q, Adj, Reach, T'_0, \ldots, T'_{k-1}\}, ar\big)$$

Intuitively, time points refer to the rows of the tiling, while agents refer to its columns. We explain next the role of the constant and relation symbols. The k unary relations T'_i, with $0 \leq i < k$, are used to encode the tiling function as follows: if $T'_i(a_j)$ holds at time point t, for some particular agent a_j, then the tiling function is $f(t,j) = T_i$. How the agent a_j is determined is explained later. There are two constant agents a_{first} and a_{last} which are used to name the first and last row of the tiling. The nullary relation Q encodes the last column of the tiling. The predicate $Adj(x,x')$ expresses that the row named by x' is directly below the row named by x. Only *Reach* is a state relation and initially (i.e. at time point 0) it is empty. There is a single sort, the agent sort A.

We let $\mathcal{S}$ be the following FO Transition System. It is used to compute all reachable parts of the adjacency relation *Adj* starting from the initial agent a_{first}:



The initial condition Init of $\mathcal{S}$ is $\forall x.\ \neg Reach(x)$. To encode the rest of the tiling requirements, we use a conjunction of $\exists^*$FOLTL formulas, where i, j implicitly range over the elements in $\{0, \ldots, k-1\}$:
All agents always have exactly one tile assigned at each point in time (Equations (5.3) and (5.4)).

$$\mathsf{G}\,\forall x.\ \bigvee_i T'_i(x) \tag{5.3}$$

[1] The original formulation used just a single period p in both directions. We use independent periods to have less complicated constructions. We note that given a periodic tiling t with periods p_x and p_y it is easy to construct a periodic tiling t' with $p'_x = p'_y = (p_x * p_y)$. The original problem also did not consider compatibility relations, but edges of the same color. Again this makes our constructions easier and is easily transformed into a solution of the original setting.

$$\mathsf{G}\,\forall x.\ \bigwedge_{i \neq j} T'_i(x) \rightarrow \neg T'_j(x) \tag{5.4}$$

A time point will be reached state where Q holds (Equation (5.5)) and it will only hold once (Equation (5.6)).

$$\mathsf{X}\,\mathsf{F}\,Q \tag{5.5}$$

$$\mathsf{G}\,(Q \rightarrow \mathsf{X}\,\mathsf{G}\,\neg Q) \tag{5.6}$$

Two adjacent time points need to be assigned x-compatible tiles (Equation (5.7)). Also, the right border of the tiling should be x-compatible to the left, i.e. the time point where Q holds should be compatible to the starting time point (Equation (5.8)).

$$\forall x.\ \mathsf{G}\,(\bigvee_{i,j:\, x\text{-comp}(T_i,T_j)} T'_i(x) \wedge \mathsf{X}\,T'_j(x)) \tag{5.7}$$

$$\forall x.\ \bigvee_{i,j:\, x\text{-comp}(T_i,T_j)} T'_j(x) \wedge \mathsf{F}\,\big(Q \wedge T'_i(x)\big) \tag{5.8}$$

Two adjacent agents need to be assigned y-compatible tiles (Equation (5.9)). The last agent should be reachable from the first via *Adj* relations. We cannot express this fact in pure $\exists^*$FOLTL, so we will use the relation *Reach* computed by the FO Transition System (Equation (5.10)). The last agent should also be y-compatible to the first (Equation (5.11)).

$$\mathsf{G}\,\forall x, x'.\ (\mathsf{F}\,\mathit{Adj}(x, x')) \rightarrow \bigvee_{i,j:\, y\text{-comp}(T_i,T_j)} T'_i(x) \wedge T'_j(x') \tag{5.9}$$

$$\mathsf{F}\,\mathit{Reach}(a_{\mathit{last}}) \tag{5.10}$$

$$\mathsf{G}\,(\bigvee_{i,j:\, y\text{-comp}(T_i,T_j)} T'_i(a_{\mathit{last}}) \wedge T'_j(a_{\mathit{first}})) \tag{5.11}$$

Let φ be the conjunction of Equations (5.3) to (5.11). Note that φ can be brought in $\exists^*$FOLTL. We show next that $\mathcal{S} \models \varphi$ iff there is a periodic tiling for $(T, x\text{-comp}, y\text{-comp})$. Let $\bar{s} \in \mathsf{Traces}(\mathcal{S})$ such that $\bar{s} \models \varphi$. We construct a tiling as follows. Since $\bar{s}$ satisfies the formula (5.10) as well as the formulas `exec` encoding the semantics of the first two edges of $\mathcal{S}$, it follows that there is a sequence $(t_0, \ldots, t_n)$ of time points with $n > 0$ and $t_0 = 0$, and a sequence $(a_0, \ldots, a_n)$ of elements of the universe such that $a_0 = a_{\mathit{first}}$, $a_n = a_{\mathit{last}}$, $\mathit{Reach}(a_i)$ holds at time point t_i, for all i with $0 \leq i \leq n$, and $\mathit{Adj}(a_i, a_{i+1})$ holds at time point t_i, for all i with $0 \leq i < n$. Then, we set p_x to the time point where Q holds and p_y to n. For $0 \leq t < p_x$ and $0 \leq j < p_y$, let $f(t, j)$ be T_i iff $T'_i(a_j)$ holds at time point t. Then f satisfies the compatibility relations and any given tiling can be transformed into a model of φ. □

5.2.3 From FOLTL properties to Hyperproperties

The results for FOLTL carry over to the FO HyperLTL case, where decidability is given for the case of quantifier-free FO Transition System, but absent for the general case.

Theorem 5.10. *Let $\mathcal{S}$ be a FO Transition System and ψ a FO HyperLTL formula. Then the following statements hold.*

1. *An FOLTL formula ψ' can be constructed in polynomial time so that $\mathcal{S} \models \neg\psi$ iff ψ' is unsatisfiable.*
2. *If $\mathcal{S}$ is quantifier-free with a purely universal initial condition and ψ is in $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL, then it is decidable whether or not $\mathcal{S} \models \neg\psi$ holds.*

Proof. Assume ψ has the form $Q_1\pi_1 \dots Q_k\pi_k.\, \varphi$, where the trace quantifiers are partitioned into a set E of existential quantifiers and a set A of universal quantifiers. Then $\mathcal{S} \models \neg\psi$ is equivalent with the validity of the following FO HyperLTL formula

$$\bar{Q}_1\pi_1 \dots \bar{Q}_k\pi_k. \Big(\bigwedge_{\bar{Q}_i \in E} \mathtt{ts}(\mathcal{S})_{\pi_i} \Big) \wedge \Big(\Big(\bigwedge_{\bar{Q}_i \in A} \mathtt{ts}(\mathcal{S})_{\pi_i} \Big) \rightarrow \neg\varphi \Big),$$

where $\mathtt{ts}(\mathcal{S})_\pi$ is $\mathtt{ts}(\mathcal{S})$ with each predicate symbol R replaced by the predicate symbol R_π, and $\bar{Q}$ is $\exists$ if Q is $\forall$ and vice-versa. The formula ψ' is then the FOLTL encoding of the following FO HyperLTL formula

$$Q_1\pi_1 \dots Q_k\pi_k. \Big(\bigwedge_{Q_i \in A} \mathtt{ts}(\mathcal{S})_{\pi_i} \Big) \rightarrow \Big(\Big(\bigwedge_{Q_i \in E} \mathtt{ts}(\mathcal{S})_{\pi_i} \Big) \wedge \varphi \Big).$$

From Theorem 4.2, to prove the second statement, it is sufficient to show that the previous FO HyperLTL formula, which we call ψ_1, can be brought in the $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL. By assumption, we have that the trace quantifier prefix of ψ_1 is of the form $\exists^*_\pi \forall^*_\pi$ and that φ is in the $\exists^*$FOLTL fragment. Also, since $\mathcal{S}$ is quantifier-free, then both $\mathtt{ts}(\mathcal{S})$ and $\neg\mathtt{ts}(\mathcal{S})$ can be brought in the $\exists^*$FOLTL fragment, as remarked in Section 3.2.3. Thus all conjuncts in the following FOLTL formula can be brought in the $\exists^*$FOLTL fragment

$$\Big(\bigwedge_{Q_i \in A} \neg\mathtt{ts}(\mathcal{S})_{\pi_i} \Big) \vee \Big(\Big(\bigwedge_{Q_i \in E} \mathtt{ts}(\mathcal{S})_{\pi_i} \Big) \wedge \varphi \Big)$$

This means that the formula itself can be put into $\exists^*$FOLTL and thus ψ_1 can be brought into $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL . □

5.2.4 From HyperLTL to Noninterference

As we have seen in Section 4.3, Noninterference is best formulated as a 2-*hypersafety* property [30], that is, a property of pairs of traces.

In our application, the sequence of edges traversed by an execution of the transition system is determined *externally*, i.e., independent of any oracle or choice predicate. For instance for the case of the conference management system, it is up to the PC chair to decide when a particular stage is complete and which next stage to execute. This means that we are only interested in the noninterference property where the considered two

traces follow the *same* control flow path, but may differ in the sequences of attained states. This assumption is formalized by the following formula:

$$\mathit{same\_paths}_{\pi,\pi'} := \mathsf{G} \bigwedge_{v \in \mathcal{R}_{cfg}} v_\pi \leftrightarrow v_{\pi'}$$

A FO Transition System $\mathcal{S}$ has the NDA property, iff

$$\forall \pi, \pi'. \left(\mathtt{ts}(\mathcal{S})_\pi \wedge \mathtt{ts}(\mathcal{S})_{\pi'} \wedge \mathit{same\_paths}_{\pi,\pi'} \wedge \mathit{agent\_model}_{\pi,\pi'} \right) \rightarrow \mathit{noninterferent}_{\pi,\pi'} \tag{5.12}$$

Thus, our approach to certify NDA for $\mathcal{S}$ consists of showing the (un)satisfiability of Equation (5.12).

Declassification In general, all external input relations $I_h \in \mathcal{R}_{input}$ come with a declassification condition $D_{I_h} x \bar{y}$. The declassification conditions appear in the formula *same_high_inputs* under negation inside a universal quantifier, so to bring the complete formula into $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL, declassification conditions should use existential quantifiers only.

Agent Models For the different agent models, the causal agent model $\mathit{agent\_model}^{(c)}$ subsumes the stubborn agent model and is less constraining on the behavior of the individual agents, which leads to more intricate information flow violations. However, the formula $\mathit{agent\_model}^{(c)} := \forall a. \mathit{causal}(a)$ is not expressible in $\exists^*$FOLTL, as it has an $\forall\exists$ quantifier structure. In case, however, that we consider a fixed upper bound on the number of causal agents, the corresponding formula $\mathit{agent\_model}^{(c,k)}$ is in the $\exists^*$FOLTL fragment.

Example 5.1.

For at most two agents, $\mathit{agent\_model}^{(c,2)}$ is the following formula:

$$\exists a_1, a_2. \mathit{causal}(a_1) \wedge \mathit{causal}(a_2) \wedge \forall a. \ (a \neq a_1) \wedge (a \neq a_2) \rightarrow \mathit{stubborn}(a)$$

which can be brought into $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL.

Considering an upper bound on the number of causal agents is a realistic setting, as it allows to verify the system for attacks by coalitions up to a given size. We obtain that NDA can be checked for quantifier-free FO Transition Systems with a bounded number of causal agents.

Theorem 5.11. *For any quantifier-free FO Transition System with a purely universal initial condition, it is decidable to check whether it satisfies Noninterference with Declassification for a finite number of causal agents and an unbounded number of stubborn agents, as long as for each formula D_{I_h} expressing a declassification condition, the negation normal form of $\neg D_{I_h}$ contains no existential quantifier.*

Given the discussed assumptions on declassification and initial condition, we can now check that the negation of Noninterference can indeed be brought into $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL. Then, as $\mathcal{S}$ is quantifier-free, the result follows directly from by Theorem 5.10.

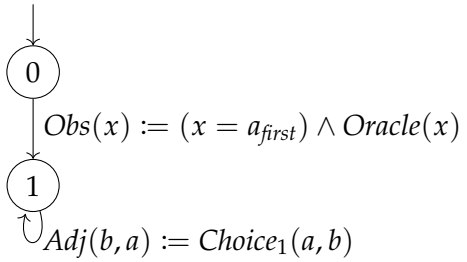
In Theorem 5.5, we have shown that for acyclic FO Transition System, it is possible to check NDA even when *all* agents behave in a causal way. This is no longer the case for FO Transition Systems containing loops:

Theorem 5.12. *The problem of checking for a given quantifier-free FO Transition System $\mathcal{S}$ whether it satisfies Noninterference with Declassification for an* unbounded *number of causal agents is undecidable, even if for all formulas D_{I_h} expressing a declassification condition, the negation normal form of D_{I_h} contains no existential quantifier.*

Proof. As in the proof for Theorem 5.9, we present a reduction from the periodic tiling problem. We will consider a FO Transition System $\mathcal{S}$ over a signature Σ with

$$\Sigma = \big(\{A\}, \{a_{first}\}, \{Q, Oracle, Obs, Adj, T'_0, \ldots, T'_{k-1}\}, ar\big)$$

where A, a_{first}, Q, and $T'_0, \ldots, T'_{k-1}$ are as in proof for Theorem 5.9, and they fulfill the same purposes. The *Adj* symbols denote again a vertical adjacency relation, but here it is not filled with input data, but rather computed stepwise by the transition system. The relations denoted by *Oracle* and *Obs* contain an initial secret that differs on both traces and spreads along the adjacency relation *Adj*. The symbols $Q, T'_0, \ldots, T'_{k-1}$ denote again high-input relations containing input data with a declassification of *true* (i.e. they are always equal in both traces). We use the following FO Transition System with the initial condition $\forall a, b.\ \neg Adj(a, b)$:



We add the rest of the tiling requirements to the declassification condition of *Oracle*, so that there only is an information flow violation in case that all formulas hold.

As we again use time as the x-axis, we reuse Equations (5.3) to (5.8). We also use a_{last} as one representative agent of the bottom-most row, so we reuse Equation (5.11) to specify that a_{last} is compatible to a_{first}. This time a_{last} is not part of the signature, but we will call the outermost agent of the non-interference condition a_{last}, so all declassification conditions can use the variable.
Two adjacent agents need to be assigned y-comp tiles (Equation (5.13)).

$$\forall a, b.\ (\mathsf{F}\, Adj(b, a)) \rightarrow \mathsf{G} \bigvee_{y\text{-comp}(i,j)} T'_i(a) \wedge T'_j(b) \tag{5.13}$$

The last agent should be reachable from the first via *Adj* relations. This is expressed by specifying that a_{last} can observe different tuples on traces π, π' (the non-interference property.). Let ψ be the conjunction of equations Equations (5.3) to (5.8) and (5.13). Let the declassification conditions be:

$$D_{Oracle} = \neg\psi,\ D_Q = true,\ D_{T'_i} = true$$

We then verify the Noninterference property given above.

There exists a satisfying model for the negation of the Noninterference property of $\mathcal{S}$ iff there is a periodic tiling for $(T, x\text{-comp}, y\text{-comp})$. If a_{last} observes different

Table 5.1: A counterexample to non-interference.

block	relation	π	π'
(e_1)	*Conflict*	(a_1, p_1)	
(e_2)	*Assign*	$(a_1, p_2), (a_2, p_2), (a_2, p_1)$	
(e_3)	*Review*	(a_2, p_1, r_{21}) (a_2, p_2, r_{22})	 (a_2, p_2, r_{22})
(e_4)	*Read*	(a_1, a_2, p_2, r_{22}) (a_2, a_2, p_2, r_{22}) (a_2, a_2, p_1, r_{21})	(a_1, a_2, p_2, r_{22}) (a_2, a_2, p_2, r_{22})
(e_5)	*Review*	(a_2, p_2, r_{21})	
(e_4)	*Read*	(a_1, a_2, p_2, r_{21}) (a_2, a_2, p_2, r_{21})	

low outputs (tuples in *Adj*), then either he is the same agent as a_{first} and y-compatible to himself or there exists a chain of causal agents spreading the tuples along *Adj* to a_{last}. Since a_{first} is the only one able to read *Oracle*, every possible differences can only originate in *Obs*. Thus, there is a chain of y-compatible causal agents a_i starting with a_{first} that reaches a_{last}. We can construct the tiling from any satisfying model in exactly the same way as in the proof of Theorem 5.9. □

Example 5.2.

Coming back to the conference management example Example 3.3, we check if the transition system satisfies noninterference. The specific example is not quantifier-free, so we adapt it slightly by adding the author of the review to be read into the *Review* relation, which is now 4-ary. We also omit the terminating state, as we do not enforce termination of the FO Transition System. The adapted example is shown in Figure 5.1.

When all agents are stubborn, we find that the system satisfies noninterference. This indicates that there is no way for any agent to learn confidential information without having a conspirator helping him.

The result is different when there is at least one causal agent. Assume two PC members a_1 and a_2 where a_1 is stubborn and a_2 is causal. The non-interference property is stated for a_1. There are two papers p_1 and p_2. First, a_1 declares a conflict with p_1, so in the rest of the workflow he should not be able to observe a difference between two executions of the workflow, regardless of which reviews p_1 receives. Both agents get assigned to p_2. In addition, a_2 gets assigned to p_1 and writes a review for it. At this point, a_2 can observe at least one review for p_1, so he can deviate his behavior on the two executions. The next step is the discussion phase. In the first step, a_2 reads all reviews of p_1. In the next step, a_2 adjusts his reviews of p_2 to mirror the reviews of p_1. Then, in the next iteration, a_1 will read the differing reviews of p_2 and learn about the result of p_1, the paper he initially declared a conflict with.

Table 5.1 formalizes the counterexample. It shows the tuples that are added to the updated relation after the execution of each edge. The reviews for p_2 cannot

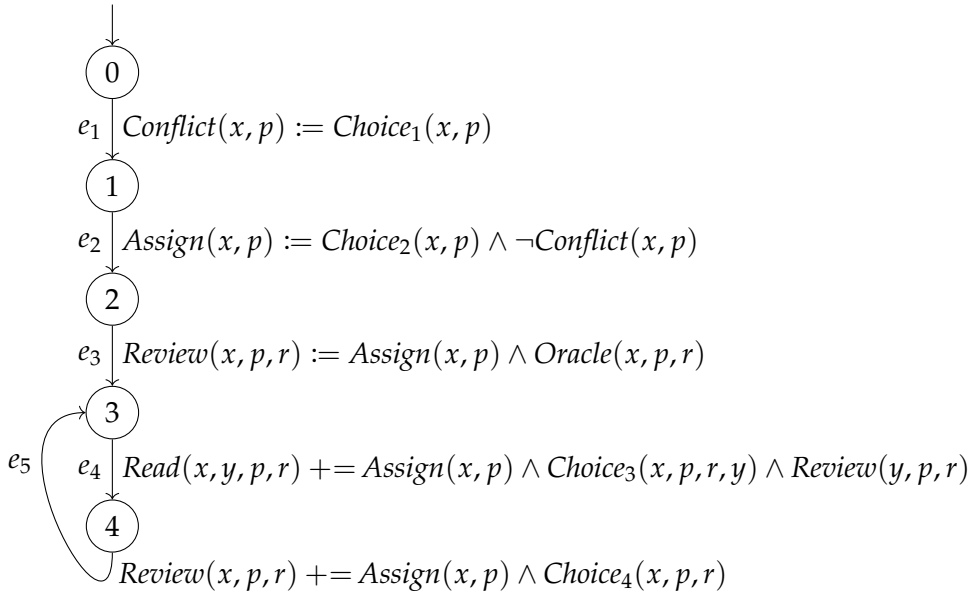


Figure 5.1: Quantifier-free version of Example 3.3

differ (in the two traces) directly after the execution of the edge (e_3) since the declassification condition states that tuples in *Oracle* can only differ when they are of the form (x, p_1, r). However, as a_2 can observe his own reviews for p_1, his choices can start to differ after (e_3) is executed; concretely, they will differ when edge (e_5) is executed. In the last two rows, any value for r (except r_{22}) would result in a counter-example; we use r_{21} to suggest that a_2 could simply replace its review for p_2 with the review for p_1. This attack represents someone copy-pasting his review for the wrong paper into one of his reviews, which is exactly the attack we mentioned in the introduction in Example 1.1.

To combat this attack, the example should be changed to only allow disjunct reviewing groups — whenever a reviewer r is assigned to a paper p, no one else that has a conflict with the other assigned papers of r can be assigned to p.

5.3 Introduced Concepts

$\mathcal{T}, \alpha, \beta \models^n \psi$	Temporal Hyperproperty ψ holds on the first n steps of the set of traces $\mathcal{T}$ with first-order valuation α and trace valuation β (n-bounded satisfaction)
$\mathcal{S} \models^n \psi$	ψ holds on $\mathcal{S}$ with all traces bounded to n steps
$R_{\pi,l}$	Relation R on trace π at timepoint l
$[\![\mathcal{S}]\!]^n$	Formula representing the n-bounded unfolding of FO Transition System $\mathcal{S}$
$[\![\psi]\!]^n_0$	Formula for the n-bounded unfolding of FOLTL formula ψ

5.4 Conclusion

In this chapter we have seen how to apply the widespread approach of model checking to the setting of quantifier-free FO Transition Systems. We have developed variants of both bounded model checking as well as symbolic model checking that can be applied to prove that a given quantifier-free FO Transition System satisfies the specification or find counter-examples that show why it does not. We have extended our methods to the setting of Hyperproperties and applied them to show that a quantifier-free FO Transition System satisfies Noninterference for all agent models. An interesting distinction is that *bounded* model checking is still decidable for the agent model where all agents behave in a causal way, while symbolic model checking which takes all traces into account is undecidable. This is due to the possibly infinite chains that could be created by agents passing information to each other.

For the case where the given FO Transition System is *not* quantifier-free, we have shown that the problem is undecidable in general, even for the specific property of Noninterference.

Chapter 6

Invariants for FO Transition Systems

Contents

6 Invariants for FO Transition Systems

In Chapter 5, we have developed techniques for the verification of quantifier-free FO Transition Systems. Many practically useful models, though, are *not* quantifier-free, like the unmodified Easychair example from Example 3.5, where y is existentially quantified in the tuple $Read(y, p, r)$ in the update formula. In general, statements which use auxiliary variables for expressing guards, naturally introduce quantifiers. A simple and common example is a workflow statement that computes the transitive closure R of some relation E when put inside a loop:

1 **forall** x,y,z. $E(x,y) \wedge R(y,z) \rightarrow R \mathrel{+}= (x,z)$

Converting this block into the setting of FO Transition Systems, we get an edge with

$$\theta = \{R(x,z) := \exists y.\ E(x,y) \wedge R(y,z)\}$$

Therefore, in this chapter we want to go beyond the results from Chapter 5, and develop automated proof strategies for more general FO Transition Systems. We have already shown that for quantifier-free FO Transition Systems and an unbounded number of *causal* agents, Noninterference with Declassification is undecidable, while for general FO Transition Systems undecidability already occurs for the simpler agent model of *stubborn* agents. In that sense, the results of Chapter 5 are tight. In this chapter, we nonetheless aim at lifting this restriction, and provide methods for analyzing general FO Transition Systems with unboundedly many agents, causal or stubborn.

Due to the undecidability results, any verification approach is necessarily incomplete and/or introduces further restrictions. We therefore concentrate on NDA in particular (instead of more general temporal hyper-properties) with declassification conditions depending on the current state only. To do so, we will use *invariants* — annotations that tell us specific facts that always hold at specific locations of the FO Transition System. Invariants have been very successfully used for verification problems where general undecidability prohibits other approaches. Examples of this include reasoning about object-oriented code [11], pointer analysis or multi-threaded behavior [41].

In the following section, we will present approaches on how to use invariants to prove that a given general FO Transition System satisfies Noninterference.

In Section 6.1 we show that pairs of execution traces complying with the assumptions on agent behavior and with the declassification conditions can be encoded into a single transformed FO Transition System. The specification of Noninterference then boils down to checking that all observations of a particular agent coincide on the two traces. Indistinguishability of such traces can then be cast as a *universal invariant*, i.e., a mapping from locations of a given FO Transition System to universally quantified formulas.

In Section 6.2, we show that proving that a given FO Transition System satisfies an invariant can be done by providing a strengthening of the invariant which is *inductive*. We find that it is decidable whether an assignment of formulas to the nodes of a control flow graph of the resulting FO Transition System is inductive or not — whenever all formulas in question are universal, and the weakest precondition calculation for each control flow edge only introduces non-nested occurrences of quantifiers. This

provides us with a means for proving noninterference with declassification for general FO Transition Systems.

Checking an invariant for being inductive, is one thing; *inferring* an inductive invariant which is sufficiently strong for proving the initial invariant, is another. In Section 6.3, we show how the latter can be cast as finding a solution to a constraint system for *weakest preconditions* of the invariant to be verified [31].

In Section 6.4, we discuss the fragment of monadic FO Transition Systems, where every predicate has at most one argument of an unbounded sort. We show that inductivity is always decidable and under which conditions inductive invariants can successfully be inferred. For cases outside these conditions, we prove undecidability. In Section 6.5, we go in a different direction and instead restrict ourselves to purely universal invariants rather than monadic ones. Again, we find that inductivity can always be decided. Moreover, the inferred invariant always stays purely universal for the class of *guarded* FO Transition Systems. When combined with the class of *stratified* FO Transition Systems, we show Section 6.6 show that weakest inductive strenghtenings can always be inferred. In combination we find that universal invariants for stratified and guarded FO Transition Systems are decidable. In Section 6.7, we generalize our methods to unrestricted FO Transition Systems. To do so, we introduce abstraction techniques that are able to deal with the undecidable formulas appearing during the computation. Still, the underlying lattice for solving the constraint system can have infinitely descending chains — implying that our methods may not always terminate. We then identify cases in which termination can be guaranteed. To solve as many cases as possible, we compute the *weakest* inductive invariant, i.e., the inductive invariant making the least assumptions at each location of the FO Transition System. Finally, we apply our findings to Noninterference in Section 6.8.

6.1 Encoding Agent Models and Declassification

The key issue of NDA is to verify that only the same observations are possible — on every trace satisfying the given sanity requirements. The structure of this property is rather complex — including trace quantifiers, temporal operators as well as implications of nested quantifiers for both agent model and declassification condition. In contrast to the model checking approaches in Chapter 5, we propose here not to include these assumptions on agent behavior and declassification into the specification of Noninterference.

Instead, we will transform the given FO Transition System $\mathcal{S}$ so that the resulting FO Transition System now tracks *pairs* of execution traces of $\mathcal{S}$ that comply with both the assumptions on agents and declassification. This simplifies the property that we want to prove by removing all trace quantifiers, all temporal operators as well as the nested quantifiers. Proving that a FO Transition System satisfies NDA then amounts to verifying that a purely universal formula holds in every state of the transformed FO Transition System.

6.1.1 Encoding of Causal Agents and Declassification

Here, we instantiate this idea for causal agents. To encode the sanity requirements into the FO Transition System, we introduce a fresh agent variable a, whose point of view we use to encode the declassification constraints. Assume we are given a FO Transition System $\mathcal{S}$ and a variable a, corresponding to the agent of concern for noninterference for the agent model $\mathit{agent\_model}^{(c)}$. We then construct the new FO Transition System $\mathcal{T}_a^{(c)}\mathcal{S}$ as follows.

Encoding pairs of traces Consider traces

$$\tau = (v_0, s_0), (v_1, s_1), \dots$$
$$\tau' = (v_0, s_0'), (v_1, s_1'), \dots$$

of the given FO Transition System over signature $\Sigma = (S, \mathcal{C}, \mathcal{R}, ar)$ and universe U, which agree in their control flow paths, i.e., $\pi(\tau) = \pi(\tau')$. To consider both traces in parallel, we combine them into a single trace $\tau \otimes \tau'$ that combines the structures attained in τ and τ'. In order to differentiate between the two parts of the state space attained in $\tau \otimes \tau'$, we introduce a copy $\mathcal{R}' = \{R' \mid R \in \mathcal{R}\}$ of the predicates in $\mathcal{R}$ and assume that the states s_i' are expressed by means of the predicates in $\mathcal{R}'$. Thus, τ is now still over the same signature Σ, while τ' is now over the signature $\Sigma' = (S, \mathcal{C}, \mathcal{R}', ar')$ where $ar'(R') = ar(R)$. Since the names of the relations in τ and τ' are now distinct, we can combine pairs of structures s_i, s_i' into a single structure $s_i \otimes s_i'$ over the signature $\Sigma'' = (S, \mathcal{C}, \mathcal{R} \cup \mathcal{R}', ar \cup ar')$ and universe U. Applying this combination along the two traces, we get

$$\tau \otimes \tau' = (v_0, s_0 \otimes s_0'), (v_1, s_1 \otimes s_1'), \dots$$

In essence, $\tau \otimes \tau'$ is now a trace of the *self-composition* of the transition system [12].

Example 6.1.

Consider again the conference management example from Example 3.5, Let τ, τ' be the two prefixes of traces:

$$\begin{aligned} \tau = &\quad (0, s_0), (1, s_1), \dots \\ \tau' = &\quad (0, s_0'), (1, s_1'), \dots \end{aligned}$$

where all structures range over the universe U with

$$\begin{aligned} U_{\mathit{PCMember}} &= \{x_1, x_2\} \\ U_{\mathit{Paper}} &= \{p_1, p_2\} \\ U_{\mathit{Report}} &= \{r_1\} \end{aligned}$$

Since both s_0 and s_0' are initial, the interpretations of all relations empty because of the initial condition Init. In the first step, authors declare conflicts of interest with some papers so let

$$\begin{aligned} \mathit{Conflict}^{s_1} &= \{(x_1, p_1)\} \\ \mathit{Conflict}^{s_1'} &= \{(x_1, p_1), (x_1, p_2)\} \end{aligned}$$

with all other relations still empty on both s_1 and s_1'. Then $s_1 \otimes s_1'$ is over the same universe U but a modified signature which contains both the relation *Conflict* and

Conflict′ (as well as two versions of all other relations) with

$$\begin{aligned} \mathit{Conflict}^{(s_1 \otimes s_1')} &= \{(x_1, p_1)\} \\ \mathit{Conflict}'^{(s_1 \otimes s_1')} &= \{(x_1, p_1), (x_1, p_2)\} \end{aligned}$$

Thus, $\tau \otimes \tau'$ contains the interpretations of all relations in τ as well the interpretations of all relations in τ', but renames the latter ones to avoid name clashes.

We can now embed properties from FO HyperLTL universally quantifying over traces π, π' into FOLTL by replacing every occurrence of a relation R_π by R and $R_{\pi'}$ by R' for any relation R in the desired property.

6.1.2 Constructing a composed FO Transition System

In the last section we showed how to combine two separate traces into a single trace. The set of all combined traces can then be used to embed FO HyperLTL properties back into FOLTL. In this section we construct a single FO Transition System whose traces are the combined traces of the initial transition system. We do this by similarly changing the signature to include copies of all relations and transforming the edges of the given FO Transition System to always update both copies of a relation at the same time.

Let $\mathcal{R}'$ denote the set of primed predicates R' corresponding to the predicates R from $\mathcal{R}$ used by $\mathcal{S}$. For a first-order formula φ with predicates from $\mathcal{R}$, let $[\varphi]'$ denote the formula obtained from φ by replacing each predicate $R \in \mathcal{R}$ with the corresponding predicate R' in $\mathcal{R}'$. Then each edge (u, θ, v) of $\mathcal{S}$ gives rise to an edge $(u, \theta_a^{(c)}, v)$ in $\mathcal{T}_a^{(c)}\mathcal{S}$. We will use $\theta_a^{(c)}$ to handle the updates to both copies of the state relations from $\mathcal{R}_{state}$ as well as update auxiliary information corresponding to the agent model. To define the individual formulas $\theta_a^{(c)}(R\bar{y})$, we make a case distinction depending on which relation symbols appear in $\theta R\bar{y}$.

Parallel Updates For an update $\theta(R\bar{y})$ in $\mathcal{S}$ that only uses predicate symbols from $\mathcal{R}_{state}$, we produce an update in $\mathcal{T}_a^{(c)}\mathcal{S}$ that updates both R and R' using the unprimed and primed copies of the relations in $\mathcal{R}_{state}$. Thus the updates in $\theta_a^{(c)}$ are:

$$\begin{aligned} R\bar{y} &:= \theta(R\bar{y}) \\ R'\bar{y} &:= [\theta(R\bar{y})]' \end{aligned}$$

Declassification Since the transformed system $\mathcal{T}_a^{(c)}\mathcal{S}$ has explicit access to both copies of the relations, it is possible to directly handle to declassification requirements in $\mathcal{T}_a^{(c)}\mathcal{S}$ and simplify the property. For update formulas $\theta(R\bar{y})$ that use a *high* input relation I_h from $\mathcal{R}_{high}$, we want $\theta_a^{(c)}(R\bar{y})$ to update R and R' in such a way that the declassification conditions are respected. This is the case if I_h and I_h' coincide for all tuples where D_{I_h} holds in one of the traces. The explicit update formulas in $\theta_a^{(c)}$ are

$$\begin{aligned} R\bar{y} &:= \theta(R\bar{y}) \\ R'\bar{y} &:= \big(D_{I_h} a\bar{x} \vee [D_{I_h}]' a\bar{x}\big) \wedge [\theta(R\bar{y})]'' \vee \\ &\quad \neg \big(D_{I_h} a\bar{x} \vee [D_{I_h}]' a\bar{x}\big) \wedge [\theta(R\bar{y})]' \end{aligned}$$

where $\bar{x}$ is the tuple in $\theta(R\bar{y})$ that is applied to I_h. In case $\bar{x}$ uses quantified variables from $\theta(R\bar{y})$, these have to be pulled outwards so that their scope includes the case distinction.

$[\theta(R\bar{y})]''$ is $\theta(R\bar{y})$ in which all relation symbols except I_h have been replaced with their primed counterparts. This ensures that if declassification holds, the updates to R and R' both use the predicate I_h, while in the other case, they use I_h and I'_h respectively. The parameter a of the declassification formula is considered as a *constant* in the transformed FO Transition System $\mathcal{T}_a^{(c)}\mathcal{S}$. For update formulas that use more than one high input relation, the transformation is similar and uses a case distinction over which subset of declassification conditions holds to update R'.

Causality The remaining case is when an update formula $\theta(R\bar{y})$ uses an agent-controlled *low* input predicate I_l from $\mathcal{R}_{low}$. In $\mathcal{T}_a^{(c)}\mathcal{S}$, agents should be forced to act causally, i.e. they should only be able to take different choices when they have already observed a difference in the considered traces. In order to capture this constraint, we introduce a new unary predicate $\mathit{Informed}(x)$, which is used to record all agents x that have already made observations depending on secret information — so their choices may diverge.

We then use a similar case distinction as in the transformation for declassification, only this time we check if the first agent mentioned in the input relation is informed. The corresponding update formulas in $\theta_a^{(c)}$ are

$$\begin{array}{lcl} R\bar{y} & := & \theta(R\bar{y}) \\ R'\bar{y} & := & \mathit{Informed}(x) \wedge [\theta(R\bar{y})]' \vee \\ & & \neg\ \mathit{Informed}(x) \wedge [\theta(R\bar{y})]'' \end{array}$$

where x is the first agent of the tuple that is applied to I_l. Again, if x is a quantified variable in $\theta(R\bar{y})$, the quantifier has to be moved outwards to include the case distinction in its scope. Here, $[\theta(R\bar{y})]''$ is $\theta(R\bar{y})$ in which all relation symbols except I_l have been replaced with their primed counterparts. This ensures that for agents that are not part of *Informed*, the unprimed version I_h is used in both traces, which ensures that the controllable inputs do not differ on the two traces.

To ensure that the new predicate *Informed* is updated correctly, we check for any pair of relations R and R' that might differ observably after the current step. Let $\mathcal{R}_{upd}$ be the set of relations updated in θ, i.e. all relations R where $\theta(R\bar{y})$ is different from $R\bar{y}$.

$$\mathit{Informed}(x) \mathrel{+}= \bigvee_{R \in \mathcal{R}_{upd}} \exists \bar{y}.\ \theta_a^{(c)} R x \bar{y} \not\leftrightarrow \theta_a^{(c)} R' x \bar{y}$$

Example 6.2.

Consider the edge $e = (2, \theta, 3)$ from Example 3.3 with

$$\theta = \{\mathit{Review}(x, p, r) := \mathit{Assign}(x, p) \wedge \mathit{Oracle}(x, p, r)\}$$

where the declassification condition for *Oracle* allows PC members to gain information about any paper they do not have declared a conflict of interest with:

$$D_{Oracle}(a, x, p, r) = \neg \mathit{Conflict}(a, p)$$

Then the transformation $\mathcal{T}_a^{(c)}e$ yields an edge $(2, \theta_a^{(c)}, 3)$ with $\theta_a^{(c)}$ being:

$$\begin{array}{lcl} \mathit{Review}(x,p,r) & := & \mathit{Assign}(x,p) \wedge \mathit{Oracle}(x,p,r) \\ \mathit{Review}'(x,p,r) & := & \mathit{Assign}'(x,p) \wedge \\ & & \left(\begin{array}{l} (\neg \mathit{Conflict}(a,p) \vee \neg \mathit{Conflict}'(a,p)) \wedge \mathit{Oracle}(x,p,r) \vee \\ (\mathit{Conflict}(a,p) \wedge \mathit{Conflict}'(a,p)) \wedge \mathit{Oracle}'(x,p,r) \end{array} \right) \\ \mathit{Informed}(x) & \mathrel{+}= & \exists p, r.\ \theta_a^{(c)}\mathit{Review}(x,p,r) \not\leftrightarrow \theta_a^{(c)}\mathit{Review}'(x,p,r) \end{array}$$

Here, the update for *Review* always uses the unprimed relation *Oracle*. The update for *Review*′ does a case distinction if D_{Oracle} holds for a on one of the two versions of *Conflict*. If it does, the tuples in *Oracle* are not confidential, so the update for *Review*′ uses the same relation *Oracle*. If it does not, the update for *Review*′ uses the unconstrained relation *Oracle*′.

In case an update formula uses both kinds of input predicates, we use a combined case distinction over the declassification for the high predicates and the informedness for the low predicates.

The transformation for stubborn agents, denoted $\mathcal{T}_a^{(s)}\mathcal{S}$, is similar, but for update formulas using a low input predicate I_l from $\mathcal{R}_{low}$, it omits the fresh predicate *Informed* and uses the unprimed copy I_l for both updates. For illustrative purposes, we show $\mathcal{T}_a^{(s)}\mathcal{S}$ in Figure 6.1 for the conference management example $\mathcal{S}$ from in Example 3.5.

The transformed FO Transition System $\mathcal{T}_a^{(c)}\mathcal{S}$ (or $\mathcal{T}_a^{(s)}\mathcal{S}$) captures all pairs of traces of $\mathcal{S}$ that satisfy the causal (or stubborn) agent model together with declassification, relative to a.

For $\mathit{agent\_model}^{(c,k)}$, we introduce k constants $a_1 \dots a_k$ into the FO Transition System and use a slight adaptation of the updates to the informedness predicate of $\mathcal{T}_a^{(c)}\mathcal{S}$ to ensure that only agents $a_1 \dots a_k$ are added to the *Informed* relation. This yields transformation $\mathcal{T}_a^{(c,k)}$.

Theorem 6.1. *Let $\mathcal{S}$ be a FO Transition System. Then for each $m \in \{s, c, (c,k) \mid k \in \mathbb{N}\}$ and for every FOLTL formula ψ using indexed predicates from $\mathcal{R}_{state}$ possibly mentioning a, the following statements are equivalent:*

- $\mathcal{S} \models \forall \pi, \pi'.\ \mathit{agent\_model}^{(m)} \rightarrow \forall a.\ \big(\mathsf{G}\, \mathit{same\_high\_inputs}(a)\big) \rightarrow \psi(a)$
- $\mathcal{T}_a^{(m)}\mathcal{S} \models [\psi(a)]'$

where $[\psi(a)]'$ replaces indexed predicates R_π by R and $R_{\pi'}$ by R' to yield a FOLTL formula using predicates from $\mathcal{R}_{state} \cup \mathcal{R}_{state}{}'$.

The proof is by establishing a simulation relation between states $s \otimes s'$ attained during a pair of traces of $\mathcal{S}$ satisfying $\mathit{agent\_model}^{(m)} \wedge \mathsf{G}\, \mathit{same\_high\_inputs}(a)$, and states $\bar{s}$ attained during a corresponding trace of $\mathcal{T}_a^{(m)}\mathcal{S}$.

Applying any one of these transformations eases the task of proving Noninterference for a given FO Transition System immensely. Since the transformation takes care of the assumptions on the agent model and declassification conditions, the only property left to prove on the transformed FO Transition System is indistinguishability for a.

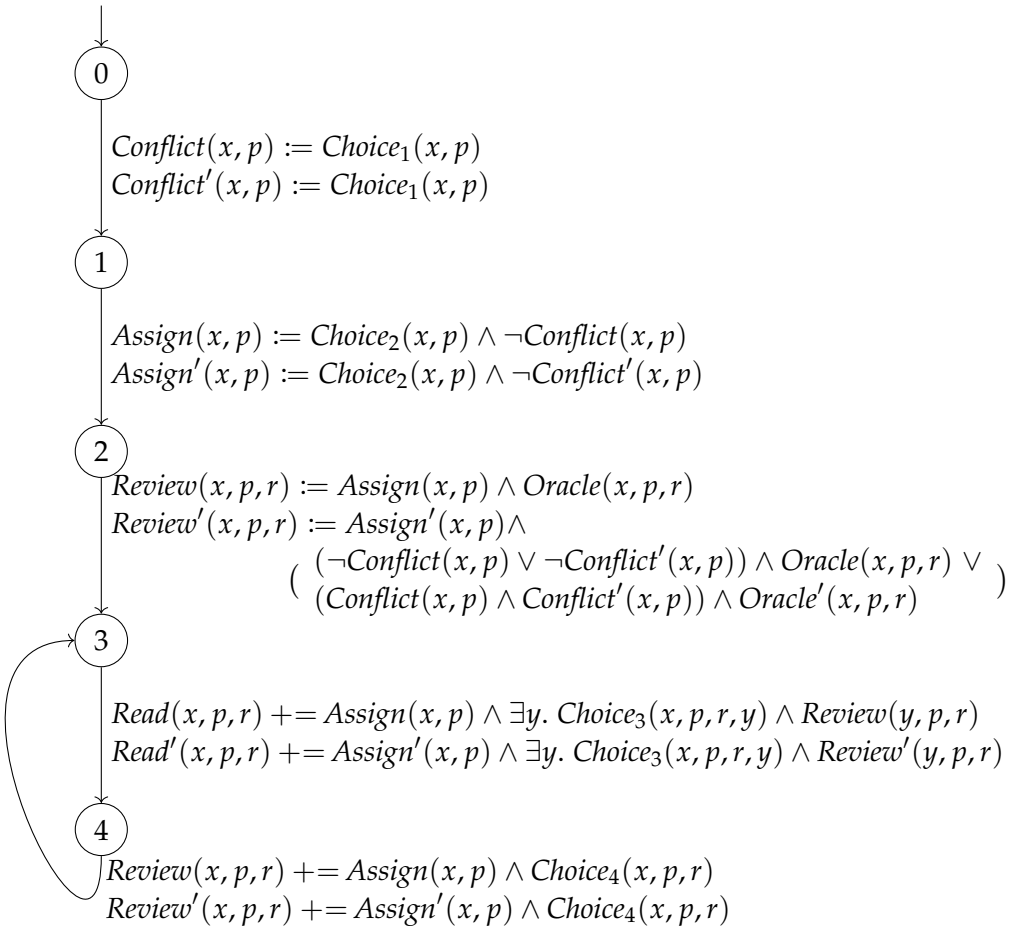


Figure 6.1: Self-composition of the FO transition system from Example 3.5

Corollary 6.2. *A given FO Transition System $\mathcal{S}$ satisfies NDA for agent model $m \in \{s, c, (c,k) \mid k \in \mathbb{N}\}$ iff*

$$\mathsf{G} \bigwedge_{R \in \mathcal{R}_{state}} \forall \bar{y}.\, Ra\bar{y} \leftrightarrow R'a\bar{y}$$

holds on $\mathcal{T}_a^{(m)}\mathcal{S}$ for a fresh agent constant a.

This allows us to remove the agent model and multi-trace aspects from the formula and focus on the analysis of a (now slightly more complicated) FO Transition System for a purely universal property with a very simple temporal structure.

6.2 Verification of Invariants

We have now transformed NDA into a universal invariant on the transformed FO Transition System. In following sections we will now focus on the problem of proving that a universal invariant holds for a given FO Transition System. In this section, we show that for any given universal invariant, we can prove or disprove inductiveness.

Weakest Preconditions For a given FO Transition System $\mathcal{S}$ over signature Σ we define a *weakest precondition* operator. It takes a formula ψ in FO Logic over Σ and a path π and gives us a formula specifying the minimum assumptions that need to hold at the beginning of π so that ψ holds at the end. For every (finite) path π in the control-flow graph of a FO Transition System $\mathcal{S}$ ending in program point v, and some property ψ, we define the weakest precondition $[\![\pi]\!]^\top \psi$ of ψ by induction on the length of π: If $\pi = \epsilon$, then $[\![\pi]\!]^\top \psi = \psi$. Otherwise, $\pi = e\pi'$ for some edge $e = (u, \theta, v')$. Then

$$[\![\pi]\!]^\top \psi := \ \forall \bar{A}_e.\theta([\![\pi']\!]^\top \psi)$$

where $\bar{A}_e$ is the set of input relations I from $\mathcal{R}_{input}$ where literals $I\bar{x}$ are introduced by the application of θ.

For an edge $e = (u, \theta, v)$, the weakest precondition $[\![e]\!]^\top \psi$ of ψ along e thus is obtained from ψ by replacing each occurrence of an atomic formula $R\bar{z}$ in ψ by the formula $\theta(R\bar{z})$. If θ uses input predicates, that substitution may have introduced occurrences of predicates from $\mathcal{R}_{input}$ which are then quantified using universal *Second Order Quantification*. We now formalize the intuition behind the $[\![\cdot]\!]^\top$ operator.

Lemma 6.3. *Let ν be a valuation of the first-order constants in $\mathcal{C}$, let τ be a finite trace in FO Transition System $\mathcal{S}$ starting in (v_0, s) and ending in (v', s'). Then $s', \nu \models \psi$ iff $s, \nu \models [\![\sigma(\tau)]\!]^\top(\psi)$.*

The proof is by repeated application of the weakest precondition operator. This gives us the weakest formula that has to hold at v_0 so that ψ holds at v'.

Invariants An *invariant* is an assignment Ψ which maps each location u of the FO Transition System to a closed first-order formula $\Psi[u]$ over predicates in $\mathcal{R}_{state}$. An invariant Ψ is *valid* (also called Ψ *holds*) for a given FO Transition System $\mathcal{S}$ iff $\Psi[u]$ holds whenever u is reached on a trace of $\mathcal{S}$. Formally, Ψ holds iff for all traces $(v_0, s_0) \dots (u, s)$ of $\mathcal{S}$ (with universe U and constant valuation ν),

$$s, \nu \models \Psi[u]$$

An invariant Ψ of $\mathcal{S}$ is called *inductive* iff for every edge (u, θ, v) of $\mathcal{S}$

$$\Psi[u] \rightarrow [\![(u, \theta, v)]\!]^{\top}(\Psi[v])$$

holds. Intuitively, an inductive invariant always carries enough information to prove that it still holds after the next step.

To prove a general invariant valid, it is necessary to reason about the possibly infinite set of reachable states. To prove an *inductive* invariant valid, it is enough to show that it holds in every initial state. This is the case iff

$$\mathsf{Init} \rightarrow \Psi[v_0]$$

Thus, an initial strategy to prove that a general invariant holds is by first proving that it is inductive and then showing that it holds in every initial state.

Example 6.3.

For our running conference management example from Example 3.3, the initial condition specifies that all relations R in $\mathcal{R}_{state}$ are empty, i.e.,

$$\bigwedge_{R \in \mathcal{R}_{state}} \forall \bar{y}. \neg R\bar{y}$$

where we assume that the sorts of the sequence of variables $\bar{y}$ matches the arity of the corresponding predicate R. An example invariant specifies that a PC member should never be assigned to a paper she declared conflict of interest with. This particular assertion should hold everywhere, so we have for every u

$$\Psi[u] = \forall x, p, r. \neg(\mathit{Conflict}(x, p) \wedge \mathit{Assign}(x, p, r)) \tag{6.1}$$

For the edge $e = (1, \theta, 2)$ from Example 3.3 with

$$\theta = \{\mathit{Assign}(x, p) := \mathit{Choice}_2(x, p) \wedge \neg \mathit{Conflict}(x, p)\}$$

and formula $\Psi[2] = \forall x, p.\ \neg(\mathit{Conflict}(x, p) \wedge \mathit{Assign}(x, p))$, we have that

$$[\![e]\!]^{\top}\Psi[2] = \forall \mathit{Choice}_2.\ \forall x, p.\ \neg(\mathit{Conflict}(x, p) \wedge \mathit{Choice}_2(x, p) \wedge \neg \mathit{Conflict}(x, p))$$

which can be simplified to

$$[\![e]\!]^{\top}\Psi[2] = \top$$

Since $\Psi[1] \rightarrow [\![e]\!]^{\top}\Psi[2] = \mathit{true}$, we have proven that whenever e is executed from a state satisfying $\Psi[1]$, it always leads to a state where $\Psi[2]$ holds. Since this implication holds not only for the edge from location 1 to location 2 but for all other edges as well, Ψ is inductive.

It is particularly convenient when for all paths π, $[\![\pi]\!]^{\top}$ introduces no alternating first-order quantifiers. In the context of a specific invariant Ψ and FO Transition System $\mathcal{S}$, a predicate is called *positive* iff it occurs only positively in Ψ as well as all substitution formulas of $\mathcal{S}$.

Definition 6.4 (Guard-Restricted FO Transition System)**.** *We call a FO Transition System* $\mathcal{S}$ guard-restricted *if all substitution formulas* $\theta(R)$ *are in the* $\exists^*$*FO Logic fragment. In the context of a specific invariant* Ψ *containing* positive *predicates,* $\theta(R)$ *is allowed to be in the alternation-free fragment of FOL.*

Guard-restrictedness holds for our running examples and is preserved by the transformation for stubborn agents introduced in Section 6.1 (assuming that all declassification conditions are in $\exists^*$FO Logic). Therefore, it is a reasonable assumption in our context. This leads us to our first decidability result for guard-restricted FO Transition Systems:

Theorem 6.5. *Let* $\mathcal{S}$ *be a guard-restricted FO Transition System and* Ψ *a universal invariant. It is decidable whether or not* Ψ *is inductive and valid.*

We remark that the restriction to *guard-restricted* FO Transition Systems can be lifted by means of the abstraction techniques provided in the next section — leading to an incomplete verification method for universal invariants on unrestricted FO Transition Systems.

Proof. The proof works by showing that all formulas fall into a decidable fragment of FO Logic, namely the Bernays-Schönfinkel-Ramsey fragment. Equivalently, we check whether $\mathsf{Init} \wedge \neg\Psi[u_0]$ as well as $\Psi[u] \wedge \neg[\![e]\!]^\top(\Psi[v])$ are unsatisfiable, for each edge e of the FO Transition System. We focus on the latter formula, as the other case is similar. Since $\mathcal{S}$ is guard-restricted and $\Psi[v]$ is a universal formula, $[\![e]\!]^\top(\Psi[v])$ is of the form $\forall \bar{A}_e.\forall \bar{y}.\psi$ where ψ has only existential first-order quantifiers, i.e., is contained in $\exists^*$FO Logic. Therefore, $\neg[\![e]\!]^\top(\Psi[v])$ is given by

$$\exists \bar{A}_e.\exists \bar{y}.\neg\psi$$

where $\neg\psi$ is in $\forall^*$FO Logic. Since the only second-order quantifiers in $\Psi[u] \wedge \neg[\![e]\!]^\top(\Psi[v])$ are the existential quantifiers for the variables in $\bar{A}_e$, satisfiability of this formula is equivalent with satisfiability of a first-order-formula consisting of two conjuncts, one in $\forall^*$FO Logic (namely, $\Psi[u]$) and the other in $\exists^*\forall^*$FO Logic. Both fragments are decidable, so their unsatisfiability can be effectively checked [20]. □

6.2.1 Guard-restrictedness in presence of guards

In Section 3.2.1, we showed how to encode *guards* into the semantics of a given FO Transition System $\mathcal{S}$. We did so by introducing a fresh predicate *error* which took care of evaluating the guards and added $\neg$*error* to the initial condition Init. Let $\mathcal{S}'$ be the resulting FO Transition System including the updates to *error*.

Example 6.4.

For an edge $e = (u, \theta, v)$ that should only be taken if a guard g is enabled, we replace θ by an updated θ' where $\theta(R\bar{y}) = \theta(R\bar{y})'$ for all relations R except *error* with

$$error := error \vee \neg\theta(g)$$

Consider an invariant Ψ on $\mathcal{S}$. We are then only interested if Ψ holds for $\mathcal{S}$ for the traces of $\mathcal{S}$ that satisfy all guards they encounter. Let the invariant Ψ' on $\mathcal{S}'$ be

$$\Psi'[u] := \neg error \rightarrow \Psi[u]$$

Then Ψ' holds on $\mathcal{S}'$ iff Ψ holds on all traces of $\mathcal{S}$ satisfying all appearing guards.

Corollary 6.6 extends the setting of Theorem 6.5 to FO Transition Systems with guards and shows that guards in $\forall^*$FO Logic or $\exists^*$FO Logic do not break guard-restrictedness.

Corollary 6.6. *Let $\mathcal{S}$ be a guard-restricted FO Transition System and Ψ a universal invariant. Let $\mathcal{S}'$ be $\mathcal{S}$ extended with guards in the alternation-free fragment of FO logic and let $\Psi'[u]$ be $\neg error \rightarrow \Psi[u]$ for all u. Then $\mathcal{S}'$ is also guard-restricted and it is decidable whether or not Ψ' is inductive and valid for $\mathcal{S}'$.*

Proof. The newly introduced predicate *error* is *fresh*, i.e. it does not appear in Ψ. Thus, *error* is *positive* in $\mathcal{S}'$ and guard-restrictedness allows updates to *error* in the alternation-free fragment of FO logic. □

To summarize, in this section we have developed a proof strategy for purely universal safety properties on guard-restricted FO Transition Systems:

1. Formulate the property as a universal invariant Ψ.
2. Check that Ψ is inductive and valid.

After transforming the initial system $\mathcal{S}$ to account for the agent model and declassification conditions, NDA becomes a purely universal safety property of the transformed FO Transition System $\mathcal{T}_a^{(c)}\mathcal{S}$ and can be cast as a universal invariant for step (1). In Corollary 6.6 we have proven that step (2) is decidable for guard-restricted FO Transition Systems. Thus, this approach yields a fully automated way to prove NDA for guard-restricted FO Transition Systems. The approach is incomplete, since valid universal invariants need not necessarily be inductive, but already gives us a useful approach to prove properties of non-quantifier-free FO Transition Systems.

In the following sections, we will build on this approach and extend it to situations where the invariant is not naturally inductive.

6.3 Inferring Inductive Invariants

Unfortunately, not every invariant is naturally inductive. Thus, in this section we show how to automatically infer a strengthening of a given invariant that is inductive. Assume that we want to verify an invariant I for an arbitrary FO Transition System $\mathcal{S}$, i.e., verify that I holds whenever the location u of $\mathcal{S}$ is reached. Then a certificate can be obtained from an inductive invariant Ψ of $\mathcal{S}$ so that

1. $\Psi[u] \rightarrow I[u]$ for all program points u; and
2. $\mathsf{Init} \rightarrow \Psi[v_0]$ for the start point v_0 of $\mathcal{S}$.

In case that $I[u] = \psi$ for all u, we then have verified that $\mathsf{G}\,\psi$ holds for the given FO Transition System $\mathcal{S}$.

Example 6.5.

For the conference management example from Example 3.5, consider the invariant I that assigns to each location:

$$\forall x\colon \mathit{PCMember}, p\colon \mathit{Paper}, r\colon \mathit{Report}.\ \neg(\mathit{Review}(x,p,r) \wedge \mathit{Conflict}(x,p))$$

It specifies that at no point in time should somebody be able to review a paper that he declared a conflict of interest with.

This particular invariant is *not* inductive: Without any additional information on the contents of the *Assign* relation, the invariant can not guarantee that the edge $(2,\theta,3)$ with

$$\theta(\mathit{Review}(x,p,r)) \coloneqq \mathit{Assign}(x,p) \wedge \mathit{Oracle}(x,p,r)$$

does not add tuples to the *Review* predicate that might violate I. There is however a strengthening Ψ of I that assigns to each location:

$$\forall x\colon \mathit{PCMember}, p\colon \mathit{Paper}.\ \neg(\mathit{Assign}(x,p) \wedge \mathit{Conflict}(x,p)) \wedge$$
$$\forall x\colon \mathit{PCMember}, p\colon \mathit{Paper}, r\colon \mathit{Report}.\ \neg(\mathit{Review}(x,p,r) \wedge \mathit{Conflict}(x,p))$$

In contrast to I, Ψ is inductive, in particular $\Psi[2]$ implies $[\![(2,\theta,3)]\!]^{\top}(\Psi[3])$. Ψ is also valid, since it holds when *Assign*, *Conflict* and *Review* are empty predicates. Thus, Ψ is an inductive strengthening of I and can be used as a certificate to prove that I holds for the example FO Transition System.

However, the required inductive invariant Ψ (if it exists) may be complicated and not easy to guess. Therefore, we are interested in automatic methods to construct such certificates. In light of requirement (1), it suffices to determine the *weakest* inductive invariant Ψ satisfying requirement (2). Given that this invariant exists and is computable, then invariant I can be certified by means of an inductive invariant iff $\mathsf{Init} \to \Psi[v_0]$.

In order to determine Ψ, we put up the following constraint system $\mathcal{C}$:

$$X[u] \to I[u] \tag{6.2}$$
$$X[u] \to [\![e]\!]^{\top}(X[v]) \quad \text{for edge } e = (u,\theta,v) \text{ of } \mathcal{S} \tag{6.3}$$

where the unknown $X[u]$ represents the potential formula assigned to program point u. By definition, any assignment X satisfying all constraints (6.3) is inductive where requirement (1) for X is expressed by the constraint (6.2).

Solutions for constraint system $\mathcal{C}$ can be computed by a fixpoint approach. To make this explicit, we define the assignment $\Psi^{(h)}$ for each $h \geq 0$ of program points u to formulas by

$$\begin{array}{lcl} \Psi^{(0)}[u] & = & I[u] \\ \Psi^{(h)}[u] & = & \Psi^{(h-1)}[u] \ \wedge\ \bigwedge_{e=(u,\theta,v)\in \mathit{out}(u)} [\![e]\!]^{\top}(\Psi^{(h-1)}[v]) \text{ for } h > 0 \end{array} \tag{6.4}$$

In case that computing $\Psi^{(h)}$ with increasing values of h indeed reaches a fixpoint, we found a solution for X — which in turn allows us to prove the initial invariant I. Any solution for X is then an inductive strengthening of the invariant I. Checking if I holds can then be simplified to checking implications just for the starting point.

Theorem 6.7. *For a FO Transition System $\mathcal{S}$ and invariant I on $\mathcal{S}$, I holds iff $\mathsf{Init} \to \Psi^{(h)}[v_0]$ holds for all $h \geq 0$.*

The proof is by induction on the lengths h of considered paths and captures the notion that $\Psi^{(h)}$ includes the weakest precondition of the invariant along all paths of length at most h. Formally, we verify that

$$\Psi^{(h)}[v] = \bigwedge \{ [\![\pi]\!]^\top I \mid \pi \text{ path starting at } v, |\pi| \leq h \}$$

for all $h \geq 0$.

In general, it is however not guaranteed that fixpoint iteration to compute a solution $\mathcal{C}$ will always terminate. However, there are classes FO Transition Systems for which termination is guaranteed: For example, for the special case of *acyclic* FO Transition Systems, the constraint system $\mathcal{C}$ will eventually stabilize.

Theorem 6.8. *For an acyclic FO Transition System $\mathcal{S}$ and an invariant I on $\mathcal{S}$, there exists some $m > 0$ such that $\Psi^{(m)} = \Psi^{(m+k)}$ holds for each $k \geq 0$. Thus, $\bigwedge_{h \geq 0} \Psi^{(h)}[u] = \Psi^{(m)}[u]$ for all u.*

Since in acyclic FO Transition Systems all paths are of finite length and there are only finitely many paths, all conjunctions of $\Psi^{(h)}[u]$ for all locations u are finite conjunctions, so $\Psi^{(h)}$ can be computed.

Interestingly, we also find that if the weakest inductive invariant is FO definable, the iteration is also guaranteed to terminate. We have:

Theorem 6.9. *Assume that for all program points u and $h \geq 0$, $\Psi^{(h)}[u]$ is FO definable as well as the infinite conjunction $\bigwedge_{h \geq 0} \Psi^{(h)}[u]$. Then there exists some $m \geq 0$ such that $\Psi^{(m)} = \Psi^{(m+k)}$ holds for each $k \geq 0$. Thus, $\bigwedge_{h \geq 0} \Psi^{(h)}[u] = \Psi^{(m)}[u]$ for all u.*

Proof. Let φ_u denote the first-order formula which is equivalent $\bigwedge_{h \geq 0} \Psi^{(h)}[u]$. In particular, this means that $\varphi_u \to \Psi^{(h)}[u]$ for each $h \geq 0$. On the other hand, we know that $\bigwedge_{h \geq 0} \Psi^{(h)}[u]$ implies φ_u. Since φ_u as well as each $\Psi^{(h)}[u]$ are assumed to be FO definable, it follows from Gödel's compactness theorem for FO logic that there is a *finite* subset $J \subseteq \mathcal{N}$ such that $\bigwedge_{h \in J} \Psi^{(h)}[u]$ implies φ_u. Let m be the maximal element in J. Then, $\bigwedge_{h \in J} \Psi^{(h)}[u] = \Psi^{(m)}[u]$ since the $\Psi^{(h)}[u]$ form a sequence of formulas that are decreasing, i.e. each formula implies its successor. In summary, we have proven that φ_u is equivalent to $\Psi^{(m)}[u]$. □

This refines our approach on how to prove a given universal safety property:

1. Formulate the property as a universal invariant I.
2. Compute any fixpoint of the strengthening $\Psi^{(h)}$.
3. Check that $\Psi^{(h)}$ is inductive and valid.

The challenges of this approach are two-fold: First, for unrestricted FO Transition Systems, formulas $\Psi^{(h)}$ may introduce arbitrarily complex quantifier structures which need not stay decidable — even for guard-restricted FO Transition Systems. Second, $\Psi^{(h)}$ might not necessarily have a fixpoint expressible by a finite formula.

In the next sections, we will explore how to overcome these two challenges. To tackle the first challenge, we examine fragments of FO Transition Systems where all formulas appearing during the computation of $\mathcal{C}$ naturally stay decidable in Sections 6.4 and 6.5. For the second challenge, we then introduce the fragment of *stratified* FO Transition Systems where fixpoints for $\Psi^{(h)}$ always exist and can be computed in Section 6.6. We then turn to abstraction techniques that allow us to tackle unrestricted FO Transition Systems in Section 6.7.

6.4 Invariant Inference for Monadic FO Transition Systems

Assuming that the universe is finite and bounded in size by some $h \geq 0$, then FO Transition Systems reduce to a finite *propositional* transition systems by encoding the state of all relations into a state of the finite transition system. So at least in principle, checking if a given invariant holds is always decidable.

A more complicated scenario arises already when each predicate has *at most one* argument which takes elements of an unbounded sort (but may have arbitrary many parameters of sorts of bounded size).

Example 6.6.

In the conference management example shown in Example 3.3, a possible scenario would be to assume that PC members and papers constitute disjoint sorts of bounded cardinalities, while the number of (versions of) reviews is unbounded.

By encoding tuples of elements from finite sorts into predicate names, we obtain FO games where all predicates are either nullary or *monadic*. FO Transition Systems where all state and input predicates are monadic are called *monadic FO Transition Systems*.

Example 6.7.

For the conference management example, let us consider at most 2 papers p_1, p_2 and at most 2 PC members x_1, x_2, but unboundedly many reviews. The predicate *Review* containing tuples (x, p, r) for a PC member x, a paper p and a report r would be encoded with 4 new monadic predicates

$$\mathit{Review}_{x_1,p_1}(r), \mathit{Review}_{x_2,p_1}(r), \mathit{Review}_{x_1,p_2}(r), \mathit{Review}_{x_2,p_2}(r)$$

Monadic FO logic is remarkable since satisfiability of formulas in monadic logic is always decidable, and monadic SO quantifiers can always be eliminated [15, 90].

Due to Theorem 6.8 we therefore conclude for *acyclic* monadic FO Transition Systems that checking if an invariant (of arbitrary quantifier structure) holds is decidable:

Theorem 6.10. *For a given finite, monadic FO Transition System $\mathcal{S}$ and an invariant I on $\mathcal{S}$, it is decidable if I holds on $\mathcal{S}$.*

For monadic FO Transition Systems which are not finite, the situation is more involved. Monadic FO Transition Systems turn out to be close in expressive power to *multi-counter machines*, for which reachability is undecidable [49, 78].

Theorem 6.11. *For monadic FO Transition Systems and a universal invariant I, it is undecidable if I holds when there are input relations in $\mathcal{R}_{input}$ as well as substitutions with equalities or disequalities.*

The proof is by using monadic predicates to simulate the counters of a multi-counter machine. The part about equalities follows from the observation that in this simulation, it is necessary to add new, unique elements to the predicate representing the counter.

Proof. We prove the statement by showing how to construct a monadic transition system $\mathcal{S}$ such that a multi-counter automaton M has a run, starting with empty counters and reaching some designated state term iff $\mathcal{S}$ is not safe. The states $q \in \{1, \ldots, n\}$ of M are encoded into flags $f_1, \ldots, f_n$ where f_1 and $f_n = \mathsf{term}$ correspond to the initial and final states, respectively. The invariant I is given by $\neg\mathsf{term}$. Each counter c_i of M is represented by a monadic predicate P_i. Incrementing the counter means to add exactly one element to P_i. In order to do so, we set all flags f_i to *false* whenever the simulation was faulty. Accordingly, we use as initial condition

$$f_1 \wedge \ \bigwedge_{j>1} \neg f_j \wedge \bigwedge_i \forall x. \neg P_i x$$

Consider a step of M which changes state f_l to $f_{l'}$ and increments counter c_i. The simulation is split into two steps, where the first step uses an input predicate A to add some elements to P_i and the second step checks that exactly one element was added. The first step uses the substitution:

$$\begin{array}{lcll} P_i y & \mapsto & P_i y \vee A y, & \\ f_{l''} & \mapsto & \begin{cases} f_l \wedge (\exists x. Ax \wedge \neg P_i x) & \text{if } l'' = l' \\ \mathit{false} & \text{if } l'' \neq l', \end{cases} & \\ P' y & \mapsto & P y & \end{array}$$

where P' is meant to record the values of the predicate P_i before the transition. By this transition, some flag $f_{l''}$ is set only when the new predicate P_i has received some new element. By the subsequent second transition, we check if the predicate A in the previous step, has more than one element outside P_i. If it does, we set all $f_{l''}$ to *false*.

$$\begin{array}{lcl} P_{i'} y & \mapsto & P_{i'} y, \\ f_{l''} & \mapsto & f_{l''} \wedge \forall x_1 x_2. P_i x_1 \vee \neg P' x_1 \vee P_i x_2 \vee \neg P' x_2 \vee x_1 = x_2 \\ P' y & \mapsto & \mathit{false} \end{array}$$

Decrementing a counter can be simulated analogously. Since counters can also be checked for 0 by checking emptiness of the corresponding predicate, we find that the FO Transition System $\mathcal{S}$ is safe iff either the simulation of the counters was erroneous or it is is not possible to reach term. Accordingly, Theorem 6.11 follows. □

Together with Theorem 6.9, this implies that the infinite conjunction of preconditions from Section 6.3 is not always FO definable — otherwise decidability would follow.

Interestingly, if a given monadic FO Transition System does not contain both input relations and equalities/disequalities, its safety can in fact be effectively decided. To prove this, let us first consider monadic FO Transition Systems without input predicates (i.e. $\mathcal{R}_{input} = \emptyset$) where both equalities and disequalities are allowed. Then, an invariant holds iff there is no universe and control-flow such that the invariant is violated at some point. For this case, we show that the intersection of preconditions from Section 6.3 necessarily stabilizes.

Theorem 6.12. *Assume that $\mathcal{S}$ is a monadic FO Transition System, possibly containing equalities and/or disequalities with $\mathcal{R}_{input} = \emptyset$. Further, let I be an invariant for $\mathcal{S}$. Then for some $h \geq 0$, $\Psi^{(h)} = \Psi^{(h+1)}$. Therefore, it is decidable if I holds for $\mathcal{S}$.*

Theorem 6.12 relies on the observation that when applying substitutions alone, i.e., without additional SO quantification, the number of equalities and disequalities in which a FO variable is involved, remains bounded. For the proof of Theorem 6.12, we rely on a technique similar to the *Counting Quantifier Normal Form* (CQNF) as introduced by Behmann in [15] and picked up in [89]. A *counting quantifier* $\exists^{\geq n} x.\varphi(x)$ expresses that at least n individuals exist for which φ holds, i.e.

$$\exists^{\geq n} x.\varphi \equiv \exists x_1 \dots x_n.\ \textstyle\bigwedge_{1\leq i\leq n} \varphi[x_i/x] \wedge \bigwedge_{i<j\leq n} x_i \neq x_j$$

A main theorem of [89] is: A monadic FO formula φ is said to be in *liberal counting quantifier normal form* (liberal CQNF) iff φ is a Boolean combination of *basic formulas* of the form:

- $\exists^{\geq n} x. \bigwedge\limits_{1\leq i\leq m} L_i(x)$

  where $n \geq 1$, $m \geq 0$, and the $L_i(x)$ are pairwise different and pairwise non-complementary positive or negative literals with unary predicates applied to the individual variable x, and dis-equalities $x \neq a$ for free variables a,
- nullary predicates P,
- $P(x)$, where P is a unary predicate and x is a global variable,
- $x = x'$, where x, x' are global variables.

φ is in *strict counting quantifier normal form* (strict CQNF) if it is in liberal CQNF and additionally does not have dis-equalities of bound FO variables with free FO variables. We remark that the notion of strict CQNF has been called just CQNF in [89]. We have:

Theorem 6.13 (CQNF for Monadic FO Formulas [89, 15])**.** *From each monadic FO formula φ equivalent FO formulas φ_1, φ_2 can be constructed such that*

1. *φ_1 is in liberal CQNF;*
2. *φ_2 is in strict CQNF;*
3. *all FO variables and predicates in φ_1, φ_2 also occur in φ.* □

We remark that the construction of φ_1 in liberal CQNF follows the same lines as the construction of φ_2 where only the step of eliminating dis-equalities between bound variables and free variables is omitted.

Example 6.8.

We illustrate the transformation into strict CQNF:

$$\begin{aligned}
&\exists y.\ \exists x.\ px \wedge x \neq y && \equiv\\
&\exists y.\ (\exists^{\geq 1} x.\ px) \wedge ((\exists^{\geq 2} x.\ px) \vee \neg py) && \equiv\\
&(\exists^{\geq 1} x.\ px) \wedge ((\exists^{\geq 2} x.\ px) \vee \exists y.\neg py) && \equiv\\
&(\exists^{\geq 1} x.\ px) \wedge ((\exists^{\geq 2} x.\ px) \vee \exists^{\geq 1} y.\neg py)
\end{aligned}$$

For a monadic FO formula φ in liberal or strict CQNF, the *quantifier rank* $qr(\varphi)$ equals the maximal k such that $\exists^{\geq k}$ occurs in φ. Likewise, for a substitution θ where all images of predicates are in strict CQNF, $qr(\theta)$ equals the maximal rank of a formula in the image of θ. For the rest of this subsection, we assume that for all substitutions, all formulas in their images are in strict CQNF. We now state our results for such substitutions on monadic FO formulas in CQNF.

Lemma 6.14. *Given a monadic FO formula φ in liberal CQNF and a substitution θ, the quantifier rank of $\theta(\varphi)$ in liberal CQNF is at most the maximum of $qr(\varphi)$ and $qr(\theta)$.*

Proof. Since φ is in liberal CQNF and $\theta(R)$ for $(R \in \mathcal{R}_{state})$ are all in strict CQNF, none of them contain equalities between bound variables, and all quantifier scopes $\exists^{\geq k} x.$ contain only literals that mention x. While in φ these scopes may contain inequalities $x \neq a$ for free variables a, this is not allowed in the $\theta(R)$. In particular, there are no dis-equalities between y and a bound FO variable. Thus, we can write $\theta(R)$ in the form

$$\bigvee_{j=1}^{l_R} \psi_{R,j}(y) \wedge \psi'_{R,j} \vee \psi''_R$$

where each $\psi_{R,j}(y)$ is a *quantifier-free* boolean combination of literals applied to y, equalities or dis-equalities of y with further free variables, and all $\psi'_{R,j}$, ψ''_R do not contain y. Applying θ to a literal $L(a)$, where a is a free variable, does not introduce new nested quantifiers. Now consider a quantified basic formula

$$\exists^{\geq k} x.\ (\textstyle\bigwedge_{i=1}^{l_1} L_i(x)) \wedge (\bigwedge_{i=1}^{l_2} \neg L'_i(x)) \wedge D(x)$$

of φ where $D(x)$ is a conjunction of disequalities with free variables of φ. Application of θ yields a formula which is a boolean combination of

- basic formulas from θ without occurrences of y since these can be extracted out of the scope of any quantifier of φ;
- basic formulas from φ without occurrences of predicates;
- basic formulas arising of the CQNF of a formula

$$\exists^{\geq k}(\bigwedge_{j=1}^{m} \psi_{R_j,i_j}[x/y]) \wedge \neg\psi_{R,i}[x/y] \wedge D(x)$$

for some predicates R_j, R and indices i_j, i. By construction, each formula $\psi_{R_j,i_j}[x/y]$ as well as formula $\neg\psi_{R,i}[x/y]$ is quantifier-free. Therefore, it is equivalent to a boolean combination of basic formulas of rank at most k.

Altogether, the rank of $\theta(\varphi)$ is thus bounded by the maximum of the ranks of φ and θ. □

Lemma 6.15. *For any monadic FO formula φ in liberal CQNF and a sequence of substitutions $\theta_0, \ldots, \theta_n$ in strict CQNF, it holds that*

$$qr(\theta_0 \ldots \theta_n(\varphi)) \leq \max(qr(\varphi), qr(\theta_0), \ldots, qr(\theta_n))$$

The proof follows from the repeated application of Lemma 6.14. Now that we proved the intermediate steps, we can prove the initial Theorem 6.12.

Proof of Theorem 6.12. For all $h \geq 0$ and nodes v, $\Psi^{(h)}[v]$ is a conjunction of sequences of substitutions θ from E applied to some FO formula $I[v']$. Thus, $qr(\Psi^{(h)}[v])$ is at most

$$\max(\{qr(\theta) \mid (v, \theta, v') \in E\} \cup \{qr(I[v']) \mid v' \in V\})$$

Let r be this maximum. For a given set of constants, there are only finitely many formulas of fixed quantifier rank (up to logical equivalence). Thus, fixpoint computation as given necessarily terminates. According to the proof of Theorem 7.24, a FO Transition System $\mathcal{S}$ is safe iff for all $h \geq 0$, $\mathsf{Init} \to \Psi^{(h)}[v_0]$. Therefore, Theorem 6.12 follows. □

Furthermore, decidability is also retained for invariant I that only contain disequalities, if no equalities between bound variables are introduced during the weakest precondition computation.

Theorem 6.16. *Assume that $\mathcal{S}$ is a monadic FO Transition System and let I be an invariant for $\mathcal{S}$. Then if*

1. *there are no disequalities between bound variables in I and*
2. *in all (positive or negative) equalities $x = y$ or $x \neq y$ in* Init *and substitutions θ at least one of x, y is a constant (from $\mathcal{C}$),*

it is decidable whether I holds for $\mathcal{S}$.

The proof is based on the following observation: Assume that $\mathcal{C}$ is a set of cardinality d, and formulas φ_1, φ_2 are closed (but might use constants from $\mathcal{C}$). If φ_1, φ_2 contain no disequalities between bound variables, then φ_1, φ_2 are equivalent for all models and all valuations ν iff they are equivalent for models and valuations with *multiplicity* exceeding d. Here, the *multiplicity* $\mu(s)$ of a model s is the minimal cardinality of a non-empty equivalence class of U w.r.t. *indistinguishability*. We call two elements u, u' of the universe U indistinguishable in a model s iff $(s, \{x \mapsto u\} \models Rx) \leftrightarrow (s, \{x \mapsto u'\} \models Rx)$ for all relations R. Then, when computing $\Psi^{(h)}$, we can instead use an abstraction by formulas without equalities, which we show to be a *weakest strengthening*.

Lemma 6.17. *Let φ be a closed FO formula possibly containing equalities or disequalities between bound variables. We construct a closed formula $\varphi^\sharp$ with neither positive nor negative equalities between bound variables such that the following holds:*

1. $\varphi^\sharp \to \varphi$;
2. *If $\psi \to \varphi$ holds for any other monadic formula ψ without (dis-)equalities between bound variables, then $\psi \to \varphi^\sharp$.*
3. *There exists some $d \geq 0$ such that for a model s of multiplicity at least d and a valuation ρ, $s, \rho \models \varphi^\sharp$ iff $s, \rho \models \varphi$.*

If the assumptions of Lemma 6.17 are met, $\varphi^\sharp$ is called the *weakest strengthening* of φ by formulas without equalities.

Proof. Assume that φ is in prenex normal form and that the quantifier-free part φ' is in disjunctive normal form. By transitivity of equality, we may assume that in each monomial m of φ' for each occurring equality $x = y$ one of the following properties holds:

- both x, y are free variables; or
- x is free and y occurs in the quantifier prefix; or
- neither x nor y are free, x is different from y and the leftmost variable in the quantifier prefix which is transitively equal to y.

Next, m is rewritten in such a way that additionally no variable y on a right side of an equality is existentially quantified. As a result, each remaining right side of an equality literal is either free in φ' (in which case the left side is also free) or universally quantified. Now consider any model s such that $\mu(s) \geq d$ for some $d > 0$ exceeding the number of free variables plus the length of the quantifier prefix of φ. Then we verify for each universally quantified variable y (by induction on the number of universally quantified variables occurring in a quantifier prefix Qz), that $s, \rho \models \forall y\, Qz.\varphi'$ iff $s, \rho \models \forall y\, Qz.\varphi''$ where φ'' is obtained from φ' by replacing each occurrence of an equality $x = y$ with $x \sim_{\mathcal{C}} y$, defined as $\bigvee_{c \in \mathcal{C}} x = c \wedge y = c$.

Accordingly, we construct $\varphi^\sharp$ from φ by replacing all equalities $x = y$ where y is universally quantified with $x \sim_{\mathcal{C}} y$. Then $\varphi^\sharp$ satisfies statements (1) and (3). In order to prove statement (2), we first observe that $\psi \to \varphi$ also holds for all models s with $\mu(s) \geq d$ for all values of d exceeding the cardinality of $\mathcal{C}$. By property (3), we therefore have that $\psi \to \varphi^\sharp$ in all models s and all valuations ρ where $\mu(s)$ is sufficiently large. Since (dis-)equalities in $\varphi^\sharp$ and in ψ are not applied to pairs of bound variables, the assertion follows. □

We conclude:

Corollary 6.18. *Assume that φ, φ' are monadic FO formulas with positive occurrences of equality only. Then*

1. $(\varphi \wedge \varphi')^\sharp = \varphi^\sharp \wedge (\varphi')^\sharp$, *and*
2. $(\forall A.\varphi)^\sharp = (\forall A.\varphi^\sharp)^\sharp$

With this, we can now prove the initial Theorem 6.12.

Proof of Theorem 6.12. Let $\Psi^{(h)}$ denote the h-th iteration of the weakest precondition (6.4) as defined in Section 6.3. Due to SO Quantifier Elimination as in [15], each formula $\Psi^{(h)}[v]$ is equivalent to a monadic FO formula. If neither I nor θ contain equalities, $\Psi^{(h)}[v]$ has positive occurrences of equalities only.
The sequence $\Psi^{(h)}[v]$ for $h \geq 0$ still need not stabilize as more and more FO variables may be introduced. Let $\Psi_0^{(h)}$ denote the h-th iteration of the *abstraction* of the weakest precondition:

$$\begin{aligned} \Psi_0^{(0)}[v] &= I[v] \\ \Psi_0^{(h)}[v] &= \Psi_0^{(h-1)} \wedge \\ &\qquad \bigwedge_{(v,\theta,v')\in E} (\forall A_e.(\Psi_0^{(h-1)}[v']\theta))^\sharp \quad \text{for } h > 0 \end{aligned}$$

where the abstraction operator $(\cdot)^\sharp$ returns the weakest strengthening by means of a monadic FO formula without equality. Recall from Corollary 6.18 that the abstraction operator commutes with conjunctions. Also, we have that $(\forall A_e.\varphi)^\sharp = (\forall A_e.\varphi^\sharp)^\sharp$ for each monadic FO formula with positive occurrences of equality only. By induction on h, we find that $\Psi_0^{(h)}[v] = (\Psi^{(h)}[v])^\sharp$ holds for all $h \geq 0$. Since Init does not contain equalities, we therefore have for all $h \geq 0$, that $\mathsf{Init} \to \Psi^{(h)}[v_0]$ iff $\mathsf{Init} \to \Psi_0^{(h)}[v_0]$. Since (up to equivalence) the number of monadic formulas without equalities or disequalities is finite, the sequence $\Psi_0^{(h)}$ for $h \geq 0$ eventually stabilizes. This means that there is some $h' \geq 0$ such that for each program point v, $\Psi_0^{(h')}[v] = \Psi_0^{(h'+1)}[v]$. Thus, FO Transition System $\mathcal{S}$ is safe iff $\mathsf{Init} \to \Psi_0^{(h')}[v_0]$. Since the implication is decidable, the theorem follows. □

The monadic version of the running example mentioned in Example 6.6 does use input relations, but no equality or disequality literals and thus stays indeed decidable.

In this section we have shown that for *monadic* FO Transition Systems, all formulas appearing during the computation of the strengthening of an invariant I are indeed decidable. This allows us to always decide if the current strengthening is inductive, even for invariants of arbitrary quantifier structure.

However, already for monadic FO Transition Systems computation of the fixpoint may fail, as we have proven monadic FO Transition Systems undecidable in general. We then examined two classes of monadic FO Transition Systems that do not introduce an unbounded number of equality literals between quantified variables and showed that for these, indeed every invariant I that holds can be strengthened to a valid inductive invariant, yielding an effective decision procedure to check if I holds for a given monadic FO Transition System.

6.5 Universal Formulas as Abstract Domain

For general, non-monadic FO Transition Systems, the formulas appearing during the fixpoint iteration might still contain arbitrary FO quantifiers and thus easily be undecidable. To still be able to show inductivity, we thus have to restrict our approach in some way. In this section, instead of restricting ourselves to monadic predicates, we instead

restrict ourselves to the domain of *universal* First-order formulas: We require that all invariants are purely universal formulas. Again, we are interested in the question: Can an invariant I be certified by such a universal inductive invariant?

6.5.1 Eliminating Universal Second-Order Quantification

Consider an edge e occurring in a FO Transition System $\mathcal{S}$ which introduces input literals of some input predicate from $\mathcal{R}_{input}$. Then the weakest precondition operator $[\![e]\!]^\top$ introduces universal quantification over the input predicate. To be able to perform the invariant strengthening given in Section 6.3, we want to eliminate the universal quantifier to end up back in the fragment of purely universal FO formulas.

In the following, we provide methods for eliminating such second order quantifiers.

Fact 6.5.1. The clause

$$\forall A.\ A\bar{z}_1 \vee \ldots \vee A\bar{z}_r \vee \neg A\bar{z}'_1 \vee \ldots \vee \neg A\bar{z}'_s \tag{6.5}$$

for sequences $\bar{z}_i, \bar{z}'_j$ of variables is equivalent to

$$\bigvee_{i=1}^{r} \bigvee_{j=1}^{s} \bar{z}_i \doteq \bar{z}'_j \tag{6.6}$$

where the equality between sequences of variables equals the conjunction of the equalities between corresponding variables.

Proof. Let us fix some values for the occurring first-order variables. First assume that the formula (6.6) holds (w.r.t. that variable assignment). Then there are some i, j so that the conjunction of equalities $\bar{z}_i \doteq \bar{z}'_j$ holds. Then $A\bar{z}_i \vee \neg A\bar{z}'_j$ is equivalent to *true* for every predicate A. Therefore, formula (6.5) holds as well.

For the reverse implication, assume that (6.6) does not hold for the given variable assignment. Then for all i, j, the sequences z_i, z'_j are different. Then some predicate A exists so that $A\bar{z}_i$ is *false* for all i, and $A\bar{z}'_j$ is *true* for all j. For that particular predicate A and the given variable assignment, the clause $A\bar{z}_1 \vee \ldots \vee A\bar{z}_r \vee \neg A\bar{z}'_1 \vee \ldots \vee \neg A'\bar{z}'_s$ is *false*. Therefore, formula (6.5) evaluates to *false* as well, thus proving the reverse implication. □

Universal quantification generally satisfies the following laws:

$$\forall A.\ \varphi_1 \wedge \varphi_2 = \quad (\forall A.\varphi_1) \wedge (\forall A.\varphi_2) \tag{6.7}$$

$$\forall A.\ \varphi_1 \vee \varphi_2 = \ \varphi_1 \vee (\forall A.\ \varphi_2) \text{ if } A \text{ does not occur in } \varphi_1 \tag{6.8}$$

Therefore, fact 6.5.1 gives rise to an effective second-order quantifier elimination in case none of the A-literals are applied to existentially quantified variables. In particular, universal second order quantifiers can always be eliminated from purely universal formulas.

Example 6.9.

$$\begin{aligned}
&\forall A.\ \forall .x, y, z.\ R(x,y) \wedge A(x,y) \vee \neg A(y,z) \vee \neg A(x,z) \\
\equiv &\forall x, y, z.\ R(x,y) \wedge \forall A.\ A(x,y) \vee \neg A(y,z) \vee \neg A(x,z) \\
\equiv &\forall x, y, z.\ R(x,y) \wedge (x = y \wedge y = z \vee x = x \wedge y = z) \\
\equiv &\forall x, y, z.\ R(x,y) \wedge y = z
\end{aligned}$$

Theorem 6.19. *Assume that ψ is a quantifier-free formula. Then a quantifier-free formula ψ' can be constructed so that $\psi' \leftrightarrow \forall A.\ \psi$ holds.*

Proof. Assume, w.l.o.g., that ψ is in conjunctive normal form. Since universal quantification distributes over conjunctions, we may apply quantifier elimination to each clause c of ψ separately. Any given clause c can be written as $c_1 \vee c_2$ where c_1 does not contain occurrences of A and c_2 collects all literals with predicate A. Then $\forall A.\ c$ is equivalent to $c_1 \vee c_2'$ where c_2' is determined from c_2 according to fact 6.5.1. This completes the construction. □

Example 6.10.

Consider the invariant $I = \forall x, p, r. \neg(\mathit{Conflict}(x,p) \wedge \mathit{Review}(x,p,r))$ and substitution θ from the edge between program points 2 and 3 in Figure 3.2, given by

$$\theta = \left\{\ \mathit{Review}(x,p,r) := \mathit{Assign}(x,p) \wedge \mathit{Oracle}(x,p,r)\ \right\}$$

Since $\theta(I)$ contains only negative occurrences of *Oracle*, SO universal quantifier elimination is particularly simple:

$$\begin{aligned}
\forall A_3.\ (I\theta) &= \forall x, p, r.\ \forall \mathit{Oracle}.\ \neg \mathit{Conflict}(x,p) \vee \neg \mathit{Assign}(x,p) \vee \neg \mathit{Oracle}(x,p,r) \\
&= \forall x, p, r.\ \neg \mathit{Conflict}(x,p) \vee \neg \mathit{Assign}(x,p)
\end{aligned}$$

The proposed procedure for second-order quantifier elimination is not new (Isabelle, e.g., easily verifies fact 6.5.1). Implicitly, our procedure can be considered as a particular instance of the SCAN algorithm [71, 40]. When multiple predicates are universally quantified, some of them may even be eliminated without introducing new equalities into the inner formula.

In addition to just eliminating universal SO quantifiers from a purely universal formula, sometimes it is possible to eliminate universal SO quantifiers together with the existential FO quantifiers introduced during a step of the fixpoint iteration from Section 6.3.

Theorem 6.20. *Consider a universal formula ψ with an arbitrary amount of free variables of the form:*

$$\bigvee_{i=1}^{n} (\exists \bar{z}_i.\ A\bar{y}_i\bar{z}_i \wedge \varphi_i) \vee \bigvee_{j=1}^{m} (\neg A\bar{x}_j' \vee \varphi_j')$$

for some $n, m \in \mathbb{N}$ where the φ_i, φ'_j are FO formulas. Then ψ is equivalent to

$$\bigvee_{i=1}^{n} \bigvee_{j=1}^{m} (\varphi'_j \vee \varphi_i[\bar{z}'_{ij}/\bar{z}_i] \wedge (\bar{y}_i = \bar{y}'_{ij}))$$

where $\bar{x}'_j$ is split into $\bar{y}'_{ij}\bar{z}'_{ij}$ where $\bar{z}'_{ij}$ is a suffix of $\bar{x}'_j$ of length $|\bar{z}_i|$ and $\bar{y}'_{ij}$ is the remaining prefix.

Proof. The split of $\bar{x}'_j$ changes depending on the length of $\bar{z}_i$, so that the substitutions and equalities always talk about vectors of variables of the same length, even for varying lengths of the vectors of the existentially quantified variables $\bar{z}_i$. Thus, all $\bar{y}_i$ are of the same length as all $\bar{y}'_{ij}$ and all $\bar{z}_i$ are of the same length as all $\bar{z}'_{ij}$. We prove the equivalence of the negations of both formulas.

1. We start by showing that

$$\begin{aligned}
&\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \neg\varphi'_j \vee \neg\varphi_i[\bar{z}'_{ij}/\bar{z}_i] \vee (\bar{y}_i \neq \bar{y}'_{ij}) \\
\rightarrow \\
&\exists A.\ (\bigwedge_{i=1}^{n} \forall \bar{z}_i.\ \neg\varphi_i \vee \neg A\bar{y}_i\bar{z}_i) \wedge \\
&\qquad (\bigwedge_{j=1}^{m} A\bar{x}'_j \wedge \neg\varphi'_j)
\end{aligned}$$

   Given a model $\mathcal{M}$ and a variable assignment ν for the free variables for the first formula, we proceed by building a relation A that satisfies the second formula in the same universe. Let A be the relation where $A\bar{y}'_{ij}\bar{z}'_{ij} = \top$ for all $\bar{y}'_{ij}$ with the $\bar{z}'_{ij}$ being the interpretations of the $\bar{z}'_{ij}$ in $\mathcal{M}$ and $\bot$ otherwise. Then A is a witness for the existential SO quantifier and $\mathcal{M}, \nu$ together with A is a model satisfying the second formula since it is $\bot$ for all $\bar{y}_i$ as they are necessarily different from the $\bar{y}'_{ij}$.

2. We now turn to the reverse implication. For that, we transform the second formula by rewriting it to a conjunction over both i and j:

$$\begin{aligned}
\exists A.\ \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \forall \bar{z}_i.(&\neg A\bar{y}_i\bar{z}_i \wedge A\bar{y}'_{ij}\bar{z}'_{ij}) \vee \\
&(\neg A\bar{y}_i\bar{z}_i \wedge \neg\varphi'_j) \vee \\
&(A\bar{y}'_{ij}\bar{z}'_{ij} \wedge \neg\varphi_i) \vee \\
&(\neg\varphi_i \wedge \neg\varphi'_j)
\end{aligned}$$

   Then for a given $i < n$ and $j < m$, if neither $\neg\varphi_i[\bar{z}'_{ij}/\bar{z}_i]$ nor $\neg\varphi'_j$ hold, the only possibility to satisfy this formula is if the first disjunct holds. This can only be the case if $\bar{y}_i \neq \bar{y}'_{ij}$ which proves the implication.

□

6.5.2 Guarded FO Transition System

We can now lift Theorem 6.20 to apply to substitutions in FO Transition System. For this, we define a class of substitutions that always lead to formulas where Theorem 6.20 can be applied.

Definition 6.21. *A substitution θ in a FO Transition System $\mathcal{S}$ is called* guarded *or* strictly guarded *iff θ modifies at most one predicate $R \in \mathcal{R}_{state}$ and $\theta(R\bar{y})$ is of one of the forms:*

$$\textit{Update:} \qquad R\bar{y} \;:=\; R\bar{y} \vee \exists \bar{z}.\, A\bar{y}\bar{z} \wedge \psi \tag{6.9}$$

$$\textit{Reset:} \qquad R\bar{y} \;:=\; \varphi \vee \exists \bar{z}.\, A\bar{y}\bar{z} \wedge \psi \tag{6.10}$$

where formulas φ, ψ are quantifier-free formulas in FO Logic and A is an input predicate [1]*. Just as for guard-restrictedness, in the presence of a specific invariant I, we can weaken the definition for updates of* positive *predicates R in I, for which formulas φ can be in the purely universal fragment $\forall^*$FO Logic instead.*

An FO Transition System is called guarded *iff every update formula $\theta(R\bar{y})$ occurring at some edge is guarded.*

For guarded fragments, the existential quantifiers introduced during the computation of the weakest precondition can always be eliminated together with the Second Order quantifiers. Thus, computing the weakest precondition of a universal formula always yields a universal formula:

Corollary 6.22. *Consider a closed purely universal FO formula Ψ and a guarded substitution θ. Let $A_1, \ldots, A_k$ be the set of relations guarding the existential quantifiers in θ. Then $\forall A_1, \ldots, A_k.\, \theta(\Psi) \equiv \Psi'$ for some purely universal FO formula Ψ'.*

Proof. The proof follows from Theorem 6.20. Consider a clause of the CNF of Ψ and one specific universally quantified relation A. Then $\forall A.\, \theta(\Psi)$ consists of conjunctions of the form:

$$\begin{array}{l} H \vee \\ \bigvee_i \exists \bar{z}_i . A\bar{y}_i\bar{z}_i \wedge \varphi_i \vee \\ \bigvee_j \forall \bar{z}'_j . \neg A\bar{y}'_j\bar{z}'_j \vee \neg\varphi'_j \end{array}$$

where i ranges only over the R_i where R_i is different from $R_i\bar{y}$ and contains an occurrence of A. All other literals (possibly containing R_i literals as well as second order variables) are bundled into the terms H, φ_i and φ'_j. Then the universal quantifier $\forall A$ can be pushed inwards and Theorem 6.20 can be applied. □

Guarded FO Transition Systems ensure that universal inductive invariants can be effectively computed and are expressive enough to decide if a given universal invariant is valid.

Theorem 6.23. *Let $\mathcal{S}$ be a guarded FO Transition System and let Ψ be a purely universal FO invariant. Assume that for some h*

$$\Psi^{(h)} = \Psi^{(h+1)}$$

Then $\Psi^{(h)}$ is an inductive invariant certifying Ψ and Ψ is valid iff $\mathsf{Init} \rightarrow \Psi^{(h)}[v_0]$.

[1] This is closely related to the *guarded* fragment of FO Logic, for which decidability is known. For an overview, we refer to [1]. In our case, the distinction that A has to be an input predicate is important, as this means it will always be eliminated during the computation of the weakest precondition of θ.

The proof is by showing that all introduced quantifiers during the computation of $\Psi^{(h)}$ can be eliminated due to Theorem 6.20 or Theorem 6.19. If we could additionally infer that the fixpoint iteration stabilizes, Theorem 6.23 would directly lead to a decision procedure. However, stabilization does not directly follow from guardedness.

6.6 Stratified Guarded FO Transition Systems

Many systems are designed with *hierarchical* information flow in mind. For the running example from Example 3.5, information always flows from the *Review* to the *Read* relation, never in the opposite direction. To capture this intuition, we introduce a notion of *stratification* to reason about directed information flow. We will then show that for universal invariants on stratified and guarded FO Transition Systems, the constraint system $\mathcal{C}$ inferring an inductive invariant always possess a finite FO logic fixpoint.

A stratification of the set $\mathcal{R}_{state}$ of predicates of the FO Transition System $\mathcal{S}$ is an (injective) mapping $\lambda : \mathcal{R}_{state} \to \mathbb{N}$ which assigns a stratum to each predicate where the lowest stratum is 0. A FO Transition System $\mathcal{S}$ is called *stratified* if for every substitution θ occurring at some edge $\theta(R\bar{y})$ is a boolean combination of $R\bar{y}$ or formulas φ where for all occurring predicates R' from $\mathcal{R}_{state}$ appearing in φ, $\lambda(R') < \lambda(R)$ holds.

Example 6.11.

The conference management example shown in Example 3.5 is stratified and guarded. A possible stratification λ is

$$\begin{aligned} \lambda(\mathit{Conflict}) &= 0 \\ \lambda(\mathit{Assign}) &= 1 \\ \lambda(\mathit{Review}) &= 2 \\ \lambda(\mathit{Read}) &= 3 \end{aligned}$$

The given stratification captures the intuition that information always flows from a lower level to a higher level and does not spread along cycles of unbounded length. For example, no information flows from the *Assign* relation back to the *Conflict* relation.

Consider again a FO Transition System $\mathcal{S}$ together with a purely universal FO assertion Ψ, and let $\Psi^{(h)}$ be defined as in Section 6.3. Together with the fragment of *guarded* FO Transition Systems, a stratification ensures that $\Psi^{(h)}$ always stabilizes:

Theorem 6.24. *Consider a FO transition system $\mathcal{S}$ where all substitutions are guarded and stratified. Then for every universal invariant Ψ, the weakest inductive invariant is again universal and can effectively be computed.*

Proof. In this proof, we use the notation $\Phi \ni \forall \bar{x}.\, c$ for a universal FO formula Φ, a clause c, and a list $\bar{x}$ of distinct variables so that for the prenex CNF $\forall \bar{z}.\, c_1 \wedge \ldots \wedge c_m$ of Φ, c occurs among the c_j, and $\bar{x}$ is the subsequence of variables in $\bar{z}$ which occur in c[2]. We rely on the lemma:

[2] "c is a clause of Φ"

Lemma 6.25. *Assume that c is a clause and θ a stratified reset or stratified guarded update with input predicate A which substitutes a predicate R with $\lambda(R) = s$. Let c' be a clause with $\forall A.\, \theta(c) \ni \forall \bar{x}.\, c'$ where $\bar{x}$ is the list of newly introduced variables in c'. Then either $c = c'$ and $\bar{x}$ is empty, or the number of literals at level s of c' is less than the corresponding number of c.*

Proof. Assume that the clause c is of the form

$$c_0 \vee R\bar{y}_1 \vee \ldots \vee R\bar{y}_n \vee \neg R\bar{y}'_1 \vee \cdots \vee \neg R\bar{y}'_m$$

where c_0 does not contain the predicate R. If θ is a reset, all literals containing R are eliminated. Therefore, the assertion of the lemma trivially holds. Since θ is a guarded update it is of the form $R\bar{y} := R\bar{y} \vee \exists \bar{z}.(A\bar{y}\bar{z} \wedge \varphi)$. Then by Theorem 6.20,

$$\begin{array}{rcl} \forall A.\, \theta(R_i\bar{y}) & \longleftrightarrow & c_0 \vee \bigvee_{j=1}^{m} \neg R\bar{y}'_j \wedge (\bigvee_{i=1}^{n} (\bar{y}_i = \bar{y}'_j) \vee \neg\psi[\bar{y}'_j/\bar{y}]) \\ & \longleftrightarrow & \bigwedge_{J\subseteq[1,m]} \forall \bar{z}_J.\, (c_0 \vee \bigvee_{j\notin J} \neg R\bar{y}'_j \vee \\ & & \qquad \bigvee_{j\in J} \bigvee_{i=1}^{n} (\bar{y}_i = \bar{y}'_j) \vee \neg\psi[\bar{y}'_j/\bar{y}, \bar{z}_j/\bar{z}]) \end{array}$$

where $\bar{z}_j$ is a fresh list of FO variables of the same length as $\bar{z}$, and $\bar{z}_J$ is the concatenation of all lists $\bar{z}_j, j \in J$. In particular for $J = \emptyset$, $\bar{z}_J$ is empty and the corresponding clause equals c. If on the other hand $J \neq \emptyset$, the number of negated literals occurring in the clause has decreased. □

By Lemma 6.25, the number of literals at level s therefore either decreases, or the clause stays the same. Let Θ denote a finite set of stratified guarded substitutions where all updates in Θ are strictly guarded, and let c_0 denote any clause. Consider a sequence $(\theta_t, \forall \bar{x}_t.c_t), t \geq 1$, where for all $t \geq 1$, $\theta_t \in \Theta$ with some input predicate A_t, and $\forall A_t.\, (\theta_t c_{t-1}) \ni \forall \bar{x}_t.\, c_t$ holds. We claim that then there is some $t' \geq 1$ so that $c_{t'} = c_{t''}$ and $\bar{x}_{t''}$ is empty for all $t'' > t'$.

In order to prove that claim, we introduce for $t \geq 1$, the vector $v_t = (v_{t,L}, \ldots, v_{t,1}) \in \mathbb{N}^L$ where L is the maximal level of a predicate in $\mathcal{R}_{state}$, and $v_{t,i}$ is the number of literals with predicates of level i. By lemma 6.25, it holds for all $t \geq 0$, that either $c_t = c_{t+1}$ and $\bar{z}_t$ is empty, or $v_t > v_{t+1}$ w.r.t. the lexicographic order on $\mathbb{N}^L$. Since the lexicographical ordering on $\mathbb{N}^L$ is well-founded, the claim follows. We conclude that the set of quantified clauses $\forall \bar{z}.c$ with $\Psi^{(h)}[u] \ni \forall \bar{z}.c$ for any u and h, is finite. From that, the statement of the theorem follows. □

The running conference management example from Example 3.5 is both stratified and guarded. Thus, Theorem 6.24 applies and proves that universal invariants are in fact decidable for this particular case. Stratification captures the intuition of a hierarchical model well and, combined with guardedness, allows the use of quantifiers in the substitution formulas of the FO Transition System while still yielding a decidability result for universal invariants. In Section 6.8, we will pick up these results and apply them to certify Noninterference.

6.7 Universal Invariants for Unrestricted FO Transition Systems

Even with the developments of the previous sections, some FO Transition System are still outside of our known decidable fragments. In this section, we propose *abstractions*,

which will still allow us to apply our techniques for the domain of universal invariants.

In order to come up with practical means for unrestricted FO Transition Systems, where we intend to solve the constraint system $\mathcal{C}$ from Section 6.3, it becomes necessary to be able to deal with arbitrary existential FO quantifiers introduced during the fixpoint iteration. To do so, we introduce an abstraction technique of existential FO quantifiers. This will allow us to further simplify the right-hand sides in the constraint system $\mathcal{C}$ and enable us to stay in the domain of universal invariants.

6.7.1 Approximating First-order Existential Quantification

Consider an edge (u, θ, v) occurring in a FO Transition System $\mathcal{S}$. Then the substitution θ may introduce fresh existential as well as fresh universal quantifiers. Since we are interested in universal inductive invariants only, our goal is to systematically remove the occurring existential quantifiers. We find:

Theorem 6.26. *For every closed formula ψ, a formula $\psi^\sharp$ can be constructed using universal quantification only so that the following two properties are satisfied:*

1. $\psi^\sharp \to \psi$;
2. *If ψ is in $\forall^*\exists^*$FO Logic, then $\varphi \to \psi^\sharp$ holds for every closed universal formula φ such that $\varphi \to \psi$.*

In light of the second statement of Theorem 6.26, the formula $\psi^\sharp$ can be seen as the uniquely determined weakest strengthening of formulas ψ in $\forall^*\exists^*$FO Logic to a universal formula. The formula $\psi^\sharp$ is called the *universal abstraction* of ψ.

Proof. We construct $\psi^\sharp$ by replacing all existential quantifiers by a disjunction over all constants and universally quantified variables which are in scope. Induction on the structure of ψ shows that $\psi^\sharp$ implies ψ. Statement (2) then follows by spelling out what it means for $\varphi \wedge \neg\psi$ to be unsatisfiable.

W.l.o.g., let us assume that ψ is in negation normal form (i.e., using conjunction and disjunction as only boolean connectives, and negation only applied to atomic propositions), and that nested universally bound variables are distinct.

We proceed by induction on the structure of ψ: For that, we introduce a transformation $[\,.\,]^\sharp_{X'}$ on subformulas of ψ, for a set X' of variables intended to be those free variables in the current subformula which are universally quantified in ψ. Then $\psi^\sharp$ is defined as $[\psi]^\sharp_{\mathcal{C}}$. The transformation $[\,.\,]^\sharp_{X'}$ is defined as follows:

$$\begin{array}{lcll}
[\forall x.\psi']^\sharp_{X'} & = & \forall x.[\psi']^\sharp_{X'\cup\{x\}} & \\
[\exists y.\psi']^\sharp_{X'} & = & \bigvee_{x\in X'} [\psi']^\sharp_{X'}[x/y] & \\
[\psi_1 \vee \psi_2]^\sharp_{X'} & = & [\psi_1]^\sharp_{X'} \vee [\psi_2]^\sharp_{X'} & \\
[\psi_1 \wedge \psi_2]^\sharp_{X'} & = & [\psi_1]^\sharp_{X'} \wedge [\psi_2]^\sharp_{X'} & \\
[\psi']^\sharp_{X'} & = & \psi' & \text{otherwise}
\end{array}$$

We claim that $\psi^\sharp$ implies ψ. For that, we prove for each finite subset X' of variables X' and each subformula ψ', that $[\psi']^\sharp_{X'} \to \psi$ holds. The proof is by induction on the structure of ψ'. The only interesting case is when ψ' is of the form $\exists y.\,\psi''$. By induction

hypothesis, $[\psi'']^\sharp_{X'} \to \psi''$ holds. Thus, also $[\psi'']^\sharp_{X'}[x/y] \to \exists y.\, \psi''$ holds for each $x \in X'$. From that the claim follows for ψ'.

For statement (2), consider any formula φ with $\varphi \to \psi$. Then $\varphi \wedge \neg\psi$ is unsatisfiable. Let ψ equal $\forall x_1, \ldots, x_r.\exists y_1, \ldots, y_s.\psi'$ with ψ' quantifier-free. Then

$$\exists x_1 \ldots x_r.\, (\varphi \wedge \forall y_1 \ldots y_s.\neg\psi')$$

must be unsatisfiable. Since that formula is equisatisfiable with

$$\exists x_1 \ldots x_r.\, (\varphi \wedge \textstyle\bigwedge_{i_1=1}^{r} \ldots \bigwedge_{i_s=1}^{r} \neg\psi'[x_{i_1}/y_1, \ldots, x_{i_s}/y_s])$$

this implies that also $\varphi \wedge \neg(\psi^\sharp)$ is unsatisfiable. Consequently, $\varphi \to \psi^\sharp$ holds. □

Applying Theorem 6.26 to the setting of FO Transition Systems, we find:

Corollary 6.27. *Assume that e is an edge in a FO Transition System $\mathcal{S}$ so that $[\![e]\!]^\top(\psi')$ is of the form $\forall A_h, \ldots, A_1.\ \psi''$ for some formula ψ'' in $\forall^*\exists^*$FO Logic. Then for all formulas ψ, it follows that $\psi \to [\![e]\!]^\top(\psi')$ iff $\psi \to \forall A_h, \ldots, A_1.(\psi'')^\sharp$ holds.*

Proof. It suffices to prove that $\psi \wedge \psi''$ is unsatisfiable iff $\psi \wedge (\psi'')^\sharp$ is unsatisfiable. That, however, follows from Theorem 6.26. □

6.7.2 Fixpoint Iteration with Abstraction

Now that we can construct the *weakest* universal strengthening of any given formula, we turn back to the problem of inferring inductive invariants for FO Transition Systems.

Now given a FO Transition System $\mathcal{S}$, we first introduce an *abstract weakest precondition operator* $[\![e]\!]^\sharp\Psi$ of a formula Ψ along an edge e that functions similarly to the weakest precondition operator $[\![e]\!]^\top\Psi$, but yields the weakest universal strengthening instead.

For every (finite) path π in the control-flow graph of $\mathcal{S}$ ending in program point v, and some property ψ, we define the *abstract* weakest precondition $[\![\pi]\!]^\sharp\psi$ of ψ: If $\pi = \epsilon$, then $[\![\pi]\!]^\top\psi = \psi^\sharp$. Otherwise, $\pi = e\pi'$ for some edge $e = (u, \theta, v')$. Then

$$[\![\pi]\!]^\sharp\psi := \ \forall \bar{A}_e.(\theta([\![\pi']\!]^\sharp\psi))^\sharp$$

where $\bar{A}_e$ is the set of input relations I from $\mathcal{R}_{input}$ where literals $I\bar{x}$ are introduced by the application of θ. According to Corollary 6.27, for a given edge e and universal formula Ψ $[\![e]\!]^\sharp\Psi$ is the weakest universal strengthening of $[\![e]\!]^\top\Psi$.

Then given a universal invariant I, we introduce the abstracted constraint system $\mathcal{C}^\sharp$. It is very similar to the initial constraint system $\mathcal{C}$ from Section 6.3 for inferring invariants, but uses the abstract weakest precondition to stay in the realm of universal formulas.

$$X[u] \to I[u] \tag{6.11}$$

$$X[u] \to [\![e]\!]^\sharp(X[v]) \quad \text{for edge } e = (u, \theta, v) \text{ of } \mathcal{S} \tag{6.12}$$

According to our assumptions, every $I[u]$ is a universal formula using the state predicates from $\mathcal{R}_{state}$ from $\mathcal{S}$ only. By applying the abstraction of existentials, followed by second-order quantifier elimination, each evaluation of a right-hand side of $\mathcal{C}^\sharp$ on a given assignment returns a universal first-order formula. Again, for $h \geq 0$, let $\Psi^{(h)}$

denote the assignment of program points to formulas which is attained after h rounds of fixpoint iteration, i.e., for $h > 0$,

$$\begin{array}{lcl} \Psi^{(0)}[u] & = & I[u] \\ \Psi^{(h)}[u] & = & \Psi^{(h-1)}[u] \ \wedge \ \bigwedge_{e=(u,\theta,v)\in out(u)} [\![e]\!]^\sharp(\Psi^{(h-1)}[v]) \text{ for } h > 0 \end{array} \tag{6.13}$$

For an edge (u, θ, v), let $\theta^{(j)}$ denote the substitution corresponding to (u, θ, v) where the predicate variables $\bar{A}_e$ are substituted with the predicates $\bar{A}_j$ (of appropriate arity). We will use the indexed versions to rename input predicates A to fresh numbered predicates A_j to avoid name clashes if the same input predicate is used multiple times.

Due to the abstraction, all appearing second-order quantifiers during the fixpoint iteration can be eliminated due to Theorem 6.19. Thus the only reason why fixpoint iteration for constraint system $\mathcal{C}^\sharp$ may not terminate, is that an ever growing number of first-order universally quantified variables is introduced. We obtain:

Theorem 6.28. *Let $\mathcal{S}$ be a FO Transition System and let I be an initial invariant on $\mathcal{S}$. Assume further that during the fixpoint iteration for the constraint system $\mathcal{C}^\sharp$ only finitely many universally quantified first-order variables are introduced. Then the iteration terminates with an assignment Ψ such that the following holds:*

1. *Ψ is a universal inductive invariant of $\mathcal{S}$ with $\Psi[u] \to I[u]$ for all program points u of $\mathcal{S}$.*
2. *If $\mathcal{S}$ is guard-restricted, then Ψ is the* weakest *assignment with property (1).*

The first statement follows by definition of the abstract weakest precondition operator, while the second statement follows from Corollary 6.27, applied at every iteration step.

Again, Theorem 6.28 can be extended to include guards as per Corollary 6.6. The set of universal formulas (modulo semantic equivalence) forms a *lattice*, when implication is seen as the ordering relation $\sqsubseteq$. W.r.t. this ordering, the greatest and least elements $\top$ and $\bot$ are represented by the formulas *true* and *false*, respectively. Likewise, the greatest lower bound of a finite set of formulas is given by their conjunction. The lattice is, however, not a *complete* lattice, i.e., not all sets of formulas necessarily have a greatest lower bound. In particular, it may have infinite decreasing chains — at least if there are two or more binary predicates E and T. As an example, consider the formulas

$$\begin{array}{rl} c_k = & (E(x_0, x_1) \wedge \ldots \wedge E(x_{k-1}, x_k)) \to T(x_0, x_k) \\ \varphi_k = & \forall x_0, \ldots, x_k.\ c_0 \wedge \ldots \wedge c_k \end{array}$$

for $k \geq 1$, where c_k forces T to at least contain the k-fold composition of relation E. All formulas φ_k are pairwise inequivalent, while at the same time $\varphi_{k+1} \to \varphi_k$ for all $k \geq 1$ holds. In general, it is thus not guaranteed that a greatest solution (corresponding to the weakest inductive invariant) exists. On the bright side, Theorem 6.9 still applies and yields that in case the weakest inductive invariant is definable in FO Logic, fixpoint iteration as solution strategy for $\mathcal{C}^\sharp$ will be able to compute it and finally terminate.

For unrestricted FO Transition Systems, this means that even though it might be undecidable to check any given universal invariant for inductivity, our fixpoint approach of computing $\mathcal{C}^\sharp$ allows us to infer a universal invariant that is inductive *by construction* iff one exists.

6.8 Application to Noninterference

Over the last few sections, we developed various techniques to prove a given invariant I for different classes of FO Transition Systems. We will now apply them to the initial setting of proving that a given FO Transition System satisfies Noninterference.

6.8.1 Stubborn Agents

Let us consider stubborn agents only and a FO Transition System $\mathcal{S}$ that is stratified.

For a given input predicate classified as *high security* $I_h \in \mathcal{R}_{high}$, inserting the declassification conditions D_{I_h} into substitutions using I_h might contradict the strata given in λ. Let $\mathcal{R}_{I_h}$ be the set of relations R where $\mathcal{S}$ contains a substitution θ where I_h appears in $\theta(R\bar{y})$. If for all relations R' appearing in D_{I_h} $\lambda(R') < \max(\{\lambda(R) \mid R \in \mathcal{R}_{I_h}\})$, we say that the declassification condition D_{I_h} *adheres* to λ. If all declassification conditions adhere to a stratification of $\mathcal{S}$, the transformed FO Transition System $\mathcal{T}_a^{(s)}\mathcal{S}$ which takes care of stubbornness of agents and declassification relative to a is again stratified.

Accordingly, we obtain:

Lemma 6.29. *Consider a FO Transition System $\mathcal{S}$ which is stratified w.r.t. stratification λ, where all declassification conditions are quantifier-free and adhere to λ as well as an agent variable a. Then $\mathcal{T}_a^{(s)}\mathcal{S}$ is again stratified.*

Lemma 6.30. *Consider a FO Transition System $\mathcal{S}$ which is guard-restricted and where all declassification conditions are quantifier-free together with an agent variable a. Then $\mathcal{T}_a^{(s)}\mathcal{S}$ is again guard-restricted.*

Unfortunately, due to the updates using the declassification conditions, $\mathcal{T}_a^{(s)}\mathcal{S}$ does not retain guardedness as defined in Definition 6.21, even if $\mathcal{S}$ is guarded. However, in a follow-up work to this thesis [68], we show that it is indeed possible to extend the notion of guardedness so that it is retained under a very similar transformation for stubborn agents, which splits simultaneous assignments to happen consecutively. Together with Lemma 6.29, this leads to decidability of NDA for stratified and guarded transition systems for $agent\_model^{(s)}$.

6.8.2 Bounded number of causal agents

Let us next consider the situation when only a *fixed* bounded number of agents behaves causally, while all others behave stubbornly. Assume that at most $k \geq 0$ behave causally, while all other agents are stubborn. Our goal is again to modify the FO Transition System in order to take care of the agent model and declassification. In case of at most k causal agents, the informedness predicate may receive only finitely many values. These finite values therefore can be encoded into locations of the transformed FO Transition System. Updates to informedness then show up as *guards* on control flow edges, which are encoded using a fresh predicate *error*.

Let $y_1, \ldots, y_k$ denote a sequence of k distinct fresh variables. Consider an edge $e = (u, \theta, v)$ of $\mathcal{S}$. Let $Y, Y' \subseteq \{y_1, \ldots, y_k\}$, $Y \subseteq Y'$, denote the subsets of agents which are

informed before and after executing the edge, respectively. Let $\mathcal{R}$ be the set of relations updated by θ. $\mathcal{T}^{(c)}_{Y,Y'}e$ is defined analogously to $\mathcal{T}^{(c)}e$ — with the major difference that now a guard is introduced to take care of the required update of informedness. In case e does not use the informedness predicate, all updates to relations $R \in \mathcal{R}$ and their copies R' in $\mathcal{T}^{(c)}_{Y,Y'}e$ are as in $\mathcal{T}^{(c)}_{a}e$ (for some agent a). In case e uses the informedness predicate, (i.e. θ uses predicates $\bar{A}_e$ from $\mathcal{R}_{low}$), $\mathcal{T}^{(c)}_{Y,Y'}e = (u, \theta', v)$ where for each substitution $R\bar{y} := \varphi$ in θ, θ' contains

$$R\bar{y} := \varphi$$
$$R'\bar{y} := [\varphi]' \wedge (x \notin Y) \vee [\varphi]'' \wedge (x \in Y)$$

where $[\varphi]'$ is φ where all state relations $R \in \mathcal{R}_{state}$ have been replaced by their primed counterpart R', but leaves input relations $I \in \mathcal{R}_{low}$ unprimed. $[\varphi]''$ is the analogue where both state relations as well as input relations are replaced by their primed counterpart. The expressions $x \notin Y$ and $x \in Y$ are shortcuts for $\bigwedge_{y_i \in Y} x \neq y_i$ and $\bigvee_{y_i \in Y} x = y_i$, respectively.

In both cases, we add an additional update to *error* that takes care of the required update of informedness.

$$error := error \vee \bigvee_{j=1\ldots k} y_j \in Y' \wedge y_j \notin Y \wedge \bigwedge_{R \in \mathcal{R}} \forall \bar{z}. \theta' R y_j \bar{z} \leftrightarrow \theta' R' y_j \bar{z}$$

The new FO Transition System $\mathcal{T}^{(c,k)}\mathcal{S}$ then consists of all locations $\langle u, Y \rangle$ for locations u of $\mathcal{S}$ and set of informed agents Y and all control flow edges

$$(\langle u, Y \rangle, \mathcal{T}^{(c)}_{Y,Y'}(u, \theta, v), \langle v, Y' \rangle)$$

for edges (u, θ, v) of $\mathcal{S}$ and $Y, Y' \subseteq \{y_1, \ldots, y_k\}$ with $Y \subseteq Y'$. The size of the resulting FO Transition System has increased by a factor of 2^k. All occurrences of the predicate *Informed*, on the other hand, have disappeared. The correctness of the transformation can be proven along the same lines as Theorem 6.1. The resulting FO Transition System $\mathcal{T}^{(c,k)}_{a}\mathcal{S}$ is now stratified iff $\mathcal{S}$ is stratified. We obtain:

Lemma 6.31. *Consider a FO Transition System $\mathcal{S}$ which is stratified w.r.t. stratification λ, where all declassification conditions are quantifier-free and adhere to λ as well as an agent variable a. Then $\mathcal{T}^{(c,k)}_{a}\mathcal{S}$ is stratified.*

Lemma 6.32. *Consider a FO Transition System $\mathcal{S}$ which is guard-restricted and where all declassification conditions are quantifier-free together with an agent variable a. Then $\mathcal{T}^{(c,k)}_{a}\mathcal{S}$ is guard-restricted.*

Again, the declassification conditions and updates fall outside the fragment of guarded FO Transition Systems but we conjecture that the widened results for guarded FO Transition Systems from [68] apply, leading to decidability of NDA for stratified and guarded FO Transition Systems for $agent\_model^{(c,k)}$ for any fixed k.

6.8.3 Causal Agents

We would like to apply the same strategy to certify noninterference also for an unbounded number of causal agents. This time, the informedness predicate is not constrained to a fixed amount of agents, so the previous transformation does not work again.

Again, we introduce a variant of the transformation $\mathcal{T}_a^{(c)}$ that additionally allows to construct a stratification if the initial FO Transition System $\mathcal{S}$ is stratified. Let $\lambda : \mathcal{R}_{state} \to \mathbb{N}$ be a stratification for $\mathcal{S}$ where no relation R in $\mathcal{R}_{state}$ is assigned the stratum 0 and all declassification conditions adhere to λ. We then build a transformation $\mathcal{T}_{(\lambda,a)}^{(c)}\mathcal{S}$ from $\mathcal{T}_a^{(c)}\mathcal{S}$ that retains stratification. The construction follows the same idea as the initial transformation $\mathcal{T}_a^{(c)}$. However, in $\mathcal{T}_a^{(c)}$, the updates to the predicate *Informed* break stratification, since they are used in every update and are themselves updated from other predicates, which forms a loop. The enhanced transformation $\mathcal{T}_{(\lambda,a)}^{(c)}$ uses the fact that updates to *Informed* are always monotonic, i.e. that no agent will ever be removed from *Informed*. This allows us to encode the updates to *Informed* by using the environment predicates instead of the state predicates.

In addition to *Informed*, $\mathcal{T}_{(\lambda,a)}^{(c)}$ will use a fresh unary input predicate I_{inf} and a flag *error*. For all updates to state predicates $R \in \mathcal{R}_{state} \cup \mathcal{R}_{state}{}'$, the update in $\mathcal{T}_{(\lambda,a)}^{(c)}\mathcal{S}$ will be the same as in $\mathcal{T}_a^{(c)}\mathcal{S}$. The difference is in how we update the informedness predicate: Instead of checking the contents of all predicates in $\mathcal{T}_a^{(c)}\mathcal{S}$, we will instead update *Informed* with the contents of the fresh input predicate I_{inf}. Since the predicate I_{inf} is chosen by the environment, we also introduce guards to track that I_{inf} only contains agents that can actually observe differences between the two runs of $\mathcal{S}$. As before, we encode the guards by a fresh, positive predicate *error*.

Consider an edge (u, θ, v) in $\mathcal{S}$ updating state predicates R_1 to R_k:

$$\begin{aligned} R_1\bar{y} &:= \varphi_1 \\ &\dots \\ R_k\bar{y} &:= \varphi_k \end{aligned}$$

It gives rise to two edges (u, θ_1', l), (l, θ_2', v) for a fresh location l in $\mathcal{T}_{(\lambda,a)}^{(c)}\mathcal{S}$, where θ_1' updates the same $R_1 \dots R_k$ and θ_2' takes care of the update to informedness and the guard (by updating the error flag *error*). Thus, $\theta_1' := \mathcal{T}_a^{(c)}(u, \theta, v)$ and θ_2' is:

$$\begin{aligned} \mathit{Informed}(x) &\mathrel{+}= I_{inf}(x) \\ \mathit{error} &\mathrel{+}= \exists x.\ I_{inf}(x) \wedge \textstyle\bigwedge_{1\dots k}^{i} \forall \bar{y}.\ R_i x\bar{y} \leftrightarrow R_i' x\bar{y} \end{aligned}$$

Example 6.12.

Consider the edge $(1, \theta, 2)$ for Example 3.5 with θ being

$$\mathit{Assign}(x, p) := \mathit{Choice}_2(x, p) \wedge \neg \mathit{Conflict}(x, p)$$

together with a stratification λ. Then $\mathcal{T}_{(\lambda,a)}^{(c)}(1, \theta, 2)$ is $(1, \theta_1', 6), (6, \theta_2', 2)$ (for a fresh location 6) with the substitutions being

$$\theta_1' := \left\{ \begin{array}{lcl} \mathit{Assign}(x, p) & := & \mathit{Choice}_2(x, p) \wedge \neg \mathit{Conflict}(x, p) \\ \mathit{Assign}'(x, p) & := & \neg \mathit{Conflict}'(x, p) \wedge \left(\begin{array}{l} \neg \mathit{Informed}(x) \wedge \mathit{Choice}_2(x, p) \vee \\ \mathit{Informed}(x) \wedge \mathit{Choice}_2'(x, p) \end{array} \right) \end{array} \right\}$$

$$\theta_2' := \left\{ \begin{array}{lcl} \mathit{Informed}(x) & \mathrel{+}= & I_{inf}(x) \\ \mathit{error} & \mathrel{+}= & \exists x.\ I_{inf}(x) \wedge \forall p.\ \mathit{Assign}(x, p) \leftrightarrow \mathit{Assign}'(x, p) \end{array} \right\}$$

For nullary predicates, $\mathit{error} \mathrel{+}= \varphi$ is still an abbreviation for $\mathit{error} := \mathit{error} \vee \varphi$.

Then $\mathcal{T}^{(c)}_{(\lambda,a)}$ with the additional assumption that *error* is false (see Section 3.2.1 for the general encoding of guards) exhibits the same traces as $\mathcal{T}^{(c)}_{a}\mathcal{S}$ does.

Lemma 6.33. *For a given FO Transition System $\mathcal{S}$, for all traces π in $\mathsf{Traces}(\mathcal{T}^{(c)}_{a}\mathcal{S})$, there exists a trace π' in $\mathsf{Traces}(\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S})$ s.t. π is a subsequence of π'.*

Proof. A trace π in $\mathcal{T}^{(c)}_{a}\mathcal{S}$ correctly keeps track of the predicate *Informed* using the negation of the error conditions in $\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S}$. For any given step in π, let S denote the set of newly informed agents. Since I_{inf} is an input predicate, there exists a trace in $\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S}$ where I_{inf} is chosen as S.

The other direction however, does not hold. Since I_{inf} can be chosen arbitrarily, $\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S}$ allows for more traces, f.e. never adding any agent to *Informed* or immediately adding every participating agent (and adding all those x to *error* in the process). □

Lemma 6.34. *For a given FO Transition System $\mathcal{S}$, and an invariant I of $\mathcal{T}^{(c)}_{a}\mathcal{S}$, I is valid for $\mathcal{T}^{(c)}_{a}\mathcal{S}$ if I' is valid for $\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S}$ where $I'[u] \coloneqq \neg error \rightarrow I[u]$ for all u.*

W.l.o.g. let λ be such that no relation R in $\mathcal{R}_{state}$ is assigned the stratum 0. Let N be a natural number that is higher than all other strata appearing in λ. From this, we construct a stratification λ_c for $\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S}$. We set $\lambda_c(R) = \lambda_c(R') = \lambda(R)$ for all state relations $\mathcal{R}_{state}$ of $\mathcal{S}$. Additionally, $\lambda_c(error) = N$ and $\lambda_c(Informed) = 0$. Then all substitutions of $\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S}$ follow the stratification λ_c and thus $\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S}$ is stratified. In case $\mathcal{S}$ is guard-restricted, all updates to *error* are alternation-free. Since *error* is fresh, it is positive w.r.t. invariants on $\mathcal{S}$ which ensures that $\mathcal{T}^{(c)}_{(\lambda,a)}\mathcal{S}$ is guard-restricted whenever $\mathcal{S}$ is.

Lemma 6.35. *Consider a FO Transition System $\mathcal{S}$ which is stratified w.r.t. stratification λ, where all declassification conditions are quantifier-free and adhere to λ as well as an agent variable a. Then $\mathcal{T}^{(c)}_{\lambda,a}\mathcal{S}$ is again stratified.*

Lemma 6.36. *Consider a FO Transition System $\mathcal{S}$ which is guard-restricted and where all declassification conditions are quantifier-free together with an agent variable a. Then $\mathcal{T}^{(c)}_{\lambda,a}\mathcal{S}$ is again guard-restricted.*

Again, stratification is retained for the agent model of only causal agents. However, this transformation depends on the existential quantifier in the update of the error predicate, which falls outside of all known versions of the guarded fragment of FO Transition Systems. This matches the undecidability results shown in Chapter 5 for an unbounded number of causal agents. Still, our approach remains optimal within the realm of universal invariants.

6.8.4 Abstraction for Noninterference

As shown in the previous sections Sections 6.8.1 to 6.8.3, guard-restrictedness and stratification is retained by the transformations for Noninterference. This allows us to apply the abstracted fixpoint iteration from Section 6.7.2 to certify Noninterference for guard-restricted FO Transition Systems.

Theorem 6.37. *Consider a FO Transition System $\mathcal{S}$ with an initial condition* Init *in the Bernays-Schönfinkel-Ramsey fragment where all agents participating in $\mathcal{S}$ behave according to some agent model (m) in $\{(s), (c,k), (c)\}$ for some $k \in \mathbb{N}$. Assume that all declassification conditions are quantifier-free. Let a be an agent variable and I an assignment of the locations of $\mathcal{S}$ to universal formulas using predicates from $\mathcal{R}_{state} \cup \mathcal{R}_{state}{}'$ and free variable a.*

Assume further that the abstract fixpoint iteration stabilizes for $\mathcal{T}_a^{(m)}\mathcal{S}$ and assignment I with some $\Psi^{(h)}$ for some h. Then

1. *$\Psi[u] \to I[u]$ for all locations u of $\mathcal{S}$.*
2. *If $\mathcal{S}$ is guard-restricted, the invariant $\Psi^{(h)}$ is the weakest universal invariant with this property.*

The proof follows from Theorem 6.28.

Corollary 6.38. *Consider a guard-restricted FO Transition System $\mathcal{S}$ with an initial condition* Init *in the Bernays-Schönfinkel-Ramsey fragment where all agents behave according to some agent model (m) (in $\{(s), (c,k), (c)\}$ for some $k \in \mathbb{N}$) and where abstract fixpoint iteration according to $\mathcal{C}^\sharp$ terminates. Then it is decidable if there exists a universal inductive invariant certifying NDA for $\mathcal{S}$ under agent model (m).*

This enables us to deal with the running conference management example and many other practically relevant cases.

Example 6.13.

The running conference management example from Figure 5.1 is both guard-restricted and stratified. The declassification for *Oracle* is $\neg \mathit{Conflict}(x, p)$ and adheres to the stratification given in Example 6.11 (which sets $\lambda(\mathit{Conflict}) < \lambda(\mathit{Assign}) < \lambda(\mathit{Review}) < \lambda(\mathit{Read})$). As was already found in Chapter 5, this particular FO Transition System does not satisfy Noninterference.

However, the results for guard-restricted FO Transition System allow us to prove a fixed version safe. To exclude the counterexample shown in Table 5.1, we fix the paper assignment to only allow a group of people with the same set of conflicts to review the same paper. The fixed version is shown in Figure 6.2, where we inserted a fresh node $2'$ and edge from 2 to $2'$ to subtract problematic tuples from the *Assign* relation. The shown example is outside of the classes that could be handled by the approaches of Chapter 5, but is both stratified and guard-restricted and can thus be handled by the invariant approach from Corollary 6.38 and is indeed proven safe by a universal invariant for an unbounded number of causal agents.

For general, unrestricted FO Transition Systems, our methods can still be applied and yield an incomplete method that, if it terminates, can prove NDA for any of the given agent models. In case there is a universal inductive FO invariant implying NDA, Theorem 6.9 still shows that the algorithm is guaranteed to terminate.

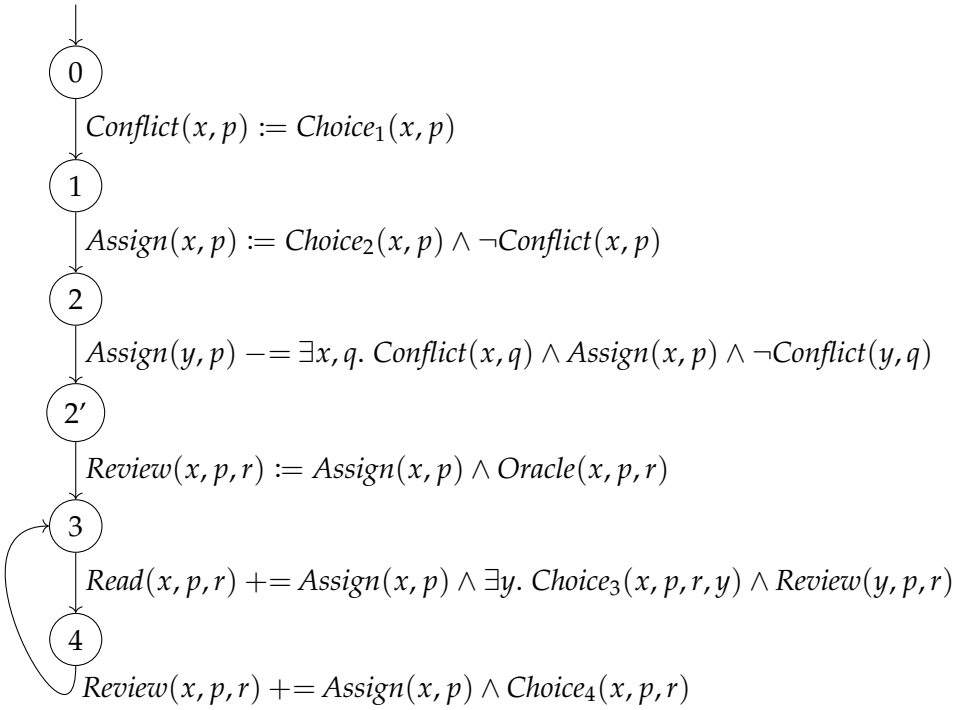


Figure 6.2: Fixed version of the running example

6.9 Forcing Stratification for General FO Transition System

In general, FO Transition Systems need not be stratified. However, we find that it is possible to adapt the approach to deal with non-stratified $\mathcal{T}_a^{(c)}\mathcal{S}$ from Section 6.8.3 to enforce stratification for general FO Transition System at the cost of adding additional input relations and further abstraction.

Consider a general FO Transition System $\mathcal{S}$. From it, we construct a new FO Transition System $\mathcal{T}^{strat}\mathcal{S}$ that will be stratified, but no longer guard-restricted (even if $\mathcal{S}$ was guard-restricted). $\mathcal{T}^{strat}\mathcal{S}$ will use the same control flow graph and state relations $\mathcal{R}_{state}$ as $\mathcal{S}$. However, in addition to the input predicates $\mathcal{R}_{input}$ from $\mathcal{S}$, we add input predicates I_R for all $R \in \mathcal{R}_{state}$ with the same arity as R, as well as a fresh 0-ary predicate *error* to encode guards. Consider an edge (u, θ, v) in $\mathcal{S}$ updating state relations R_1 to R_k:

$$\begin{array}{lcl} R_1\bar{y} & := & \varphi_1 \\ \ldots & & \\ R_k\bar{y} & := & \varphi_k \end{array}$$

It gives rise to an edge (u, θ', v) in $\mathcal{T}^{strat}\mathcal{S}$ updating the same R_i as well as *error*:

$$\begin{array}{lcl} R_1\bar{y} & := & I_{R_1}\bar{y} \\ \ldots & & \\ R_k\bar{y} & := & I_{R_k}\bar{y} \\ error & := & error \vee \bigwedge_{i=1\ldots k} \exists \bar{y}.\ I_{R_i}\bar{y} \not\leftrightarrow \varphi_i \end{array}$$

The difference between the setting for this transformation and the transformation $\mathcal{T}^{(c)}_{(\lambda,a)}$ for already stratified FO Transition System is that not all relations R need to change monotonically. Here, we need to completely replace R, instead of only adding new tuples to it. This changes the necessary update condition for *error*, which now features a $\not\leftrightarrow$ — implying that even for φ_i in $\exists^*$FO Logic, $\theta(error)$ contains quantifier alternations and now falls outside the class of guarded as well as guard-restricted FO Transition System.

Still, $\mathcal{T}^{strat}\mathcal{S}$ exhibits the same traces $\mathcal{S}$ does — under the condition that $\neg error$ holds along the traces of $\mathcal{T}^{strat}\mathcal{S}$. The only relation in $\mathcal{R}_{state}$ where state relations appear in update formulas $\theta(R\bar{y})$ now is *error*. We set $\lambda(error) = 2$ and $\lambda(R) = 1$ for all other relations $R \in \mathcal{R}_{state} \setminus \{error\}$. Then, $\mathcal{T}^{strat}\mathcal{S}$ is stratified w.r.t λ.

Theorem 6.39. *Consider a FO Transition System $\mathcal{S}$ and an invariant I of $\mathcal{S}$.*
Then $\mathcal{T}^{strat}\mathcal{S}$ is stratified.

In comparison to the construction we used to enforce stratification for the relation *Informed* in the transformed transition system, this construction loses the precision guarantee and does not preserve guardedness or guard-restrictedness. Still, it allows to transform a general FO Transition System into a stratified FO Transition System with the same traces — at the cost of lost precision during the computation.

6.10 Alternative First-order Logic based approaches

As mentioned in Section 3.3, the programming language RML [75], which we argue can be encoded into FO Transition Systems in Section 3.2.4, is another formal model based on evolving states of First-order relations which is fairly close to our FO Transition Systems. Verification for RML is implemented in the Ivy tool. As in our work, the language restricts its statements in such a way that checking whether universal invariants are inductive reduces to checking the satisfiability of an $\exists^*\forall^*$FO Logic formula and finds similar results to the results of Section 6.2. Since the interpretation of all external relations is fixed in the beginning, there is no need for Second Order Quantifier elimination. In addition to checking invariants for inductiveness, their work focuses on the automated extraction and visualization of counterexamples to a given universal invariant. In Chapter 7, we will show how to extract similar counterexamples for FO Transition Systems. In contrast to our approaches based on weakest preconditions, Ivy uses UPDR [58] to infer inductive invariants which provides weaker termination guarantees.

Another model with work on automatic verification is the VeriCon system [72], which has been proposed for describing and verifying the semantics of controllers in software-defined networks. The semantics of the underlying language is again specified in terms of First-order relations. Their goal is to prove various kinds of network invariants (topology invariants, safety invariants and transition invariants), which are checked with the automated theorem prover Z3 and iteratively strengthened similar to the methods shown in Section 6.3. In [73], the difficulty of inferring universal inductive invariants is investigated for classes of transition systems whose transition relation is expressed by FO logic formulas over theories of data structures. The authors show that inferring universal

inductive invariants is decidable when the transition relation is expressed by formulas with unary predicates and a single binary predicate restricted by the theory of linked lists and becomes undecidable as soon as the binary symbol is not restricted by background theory. By excluding the binary predicate, this result is closely related to our result for transition systems with monadic predicates, equality and disequality, but neither $\mathcal{A}$- nor $\mathcal{B}$-predicates. In our work, the termination argument relies on also imposing constraints on the structure of the transition system, rather than on the formulas alone. In [58], an inference method is provided for universal invariants s an extension of Bradley's PDR/IC3 algorithm for inference of propositional invariants [24]. The method is applied to variants of FO transition systems within a fragment of FO logic which enjoys the finite model property and is decidable. Whenever it terminates, it either returns a universal invariant which is inductive or a counter-example. From a counter-example, they obtain a minimal universal formula to exclude that example together with a family of related examples — which they use for strengthening the candidate invariant. This should be contrasted to our approach where in each iteration, the candidate invariant is more agressively strengthened — while still preserving optimality w.r.t. universal formulas. In Chapter 7, we will show a slight variation of the counter-example search that even excludes *all* counter-examples up to a given size in one go.

6.11 Introduced Concepts

$\tau \otimes \tau'$	Composition of trace τ and τ' (with disjoing predicate names in τ and τ').
$\mathcal{T}_a^{(m)}\mathcal{S}$	Self-composed FO Transition System from $\mathcal{S}$ that includes agent model (m) with respect to fresh agent a.
$\mathcal{T}_{\lambda,a}^{(c)}\mathcal{S}$	Self-composed FO Transition System from $\mathcal{S}$ that includes agent model (c) with respect to fresh agent a and retains stratification λ.
$\mathcal{T}^{strat}\mathcal{S}$	Transformed FO Transition System from $\mathcal{S}$ that is always stratified.
$[\![\pi]\!]^\top \varphi$	Weakest precondition of φ along the path π.
I	Variable for invariants (assignment of locations of a FO Transition System to formulas).
Ψ	Variable for inductive strengthenings of invariants.
$\Psi^{(h)}$	Inductive strengthening of an invariant by h steps using fixpoint iteration.
$\varphi^\sharp$	Universal abstraction of φ (purely universal formula implying φ).

6.12 Conclusion

The goal of this chapter was to provide methods for verifying the property of Noninterference for more general FO Transition Systems than the rather restricted fragment of quantifier-free FO Transition Systems. We proceeded in two steps. First, we simplified the NDA property by encoding execution of two traces together with both the agent model and declassification conditions into the FO Transition System itself. This allowed us to use inductive universal invariants to prove the now structurally simpler Noninterference property.

We applied our methods to more and more general classes of FO Transition Systems and described where our methods lead to decidability or to relative completeness.

For the case of *monadic* FO Transition Systems, we gave a complete characterization into decidable and undecidable fragments for invariants of arbitrary quantifier structure. For more general classes of FO Transition Systems and invariants, we found it useful to *abstract* arbitrary formulas by universal formulas. Together with a method for second-order quantifier elimination which results in purely universal formulas for guarded FO Transition Systems, this allowed us to not only check a given invariant for inductivity but even infer an inductive invariants in general FO Transition Systems. For the class of stratified and guarded FO Transition Systems we succeeded to prove termination of our approach and could compute the *best*, i.e., weakest universal invariant which is inductive. This leads to decidability of invariants for guarded and stratified FO Transition Systems and to relative completeness of NDA for guard-restricted FO Transition Systems.

Chapter 7

First-order Safety Games

Contents

7 First-order Safety Games

In the preceding chapters we assumed all input to be under external, potentially malicious control. In the real world however, not all input relations are necessarily chosen maliciously. Rather, sometimes we are interested in finding out which of our possible choices for some predicates lead to a correct system. In terms of our examples:

1. Given the leader election protocol shown in Example 3.11, can we automatically find which messages to send on along the ring so that the protocol is correct in the end?

2. Given the conference management system from Example 3.5, can we help the PC chair by automatically finding a paper assignment so that no PC member will be able to obtain illegitimate information about any of the reports?

In this chapter, we study an extension of FO Transition Systems where some of the input predicates are chosen with the intention to uphold the desired properties of the system rather than violate them. These question are studied extensively for traditional finite-state transition systems and are known as the problem of *synthesis* (see e.g. [28, chapter 27]).

Example 7.1.

Figure 7.1 shows a slightly simplified version of the network leader election protocol from Example 3.11, introduced as a running example in [75] — turned into a First-order Safety Game. The topology of the network, here a ring, is given by the predicates *next* and $\leq$, which are appropriately axiomatized. The participating agents communicate via messages through the predicate *msg* but are only allowed to send messages to the next agent in the ring topology. In the first step, agents can send any message (determined via the input predicate B) to their neighbor. Afterwards they check if they have received a message containing their own id. If so, they declare themselves leader and add themselves to the *leader* relation. Then, a subset of processes determined by the input predicate A decides to send any id to their next neighbor that they have received which is not exceeded by their own.

At no point more than one process should have declared itself leader — regardless of the size of the ring. This property is enforced, e.g., if the initial message to be sent is given by the id of the sending process itself, i.e., $B(a, i, b)$ is given by the literal $(i = a)$.

In the following sections, we will handle these questions and discuss how to automatically infer strategies that tell the benevolent actors how to choose their input predicates in such a way that the overall system becomes safe. In Section 7.1, we introduce the formal setting and show how to extend FO Transition Systems to FO Safety Games. We introduce how strategies work in this scenario and relate them to Second Order Logic. In Section 7.2, we discuss how Noninterference translates to the FO Safety Game setting and lift the self-composition construction for Noninterference from Section 6.1 to FO

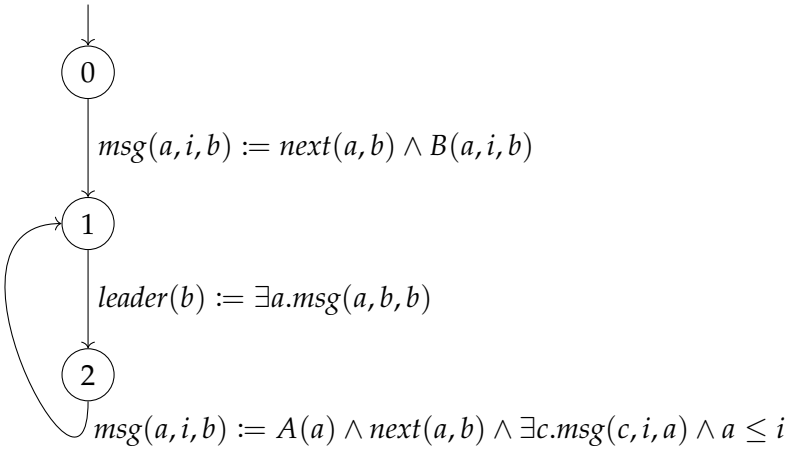


Figure 7.1: FO Safety Game for the leader election example

Safety Games. This leads us to invariant inference for FO Safety Games. We proceed as in the last chapter and first examine in which cases the problem is decidable. We first discuss how invariants can be used to solve *monadic* FO Safety Games in Section 7.3 before turning to more general classes of FO Safety Games. In Section 7.4, we show under which conditions inductivity of invariants can still be shown. We then show how to use an inductive invariant to extract winning strategies in Section 7.5. In Section 7.6, we again resort to approximation to infer inductive invariants and extract winning strategies for more general FO Safety Games and improve on the guarantees of the approximative approach for FO Transition Systems. Finally, we discuss how to use the extracted winning strategies in the context of Noninterference in Section 7.7.

7.1 First-order Safety Games

We now introduce the model of *FO Safety Games* — 2-player games where reachability player $\mathcal{A}$ aims at violating a given invariant I while safety player $\mathcal{B}$ tries to establish I as an invariant. To do so, we extend the definition of FO Transition Systems and give each player control of some of the input predicates. Accordingly, we partition the set of input predicates $\mathcal{R}_{input}$ into subsets $\mathcal{R}_{\mathcal{A}}$ and $\mathcal{R}_{\mathcal{B}}$. While player $\mathcal{B}$ controls the valuation of the predicates in $\mathcal{R}_{\mathcal{B}}$, player $\mathcal{A}$ has control over the valuations of predicates in $\mathcal{R}_{\mathcal{A}}$ as well as over the universe and the exact initial state. This includes the valuation of the state predicates as well as the constants in $\mathcal{C}$.

Example 7.2.

A FO Safety Game version of the Easychair running example is shown in Figure 7.2. Player $\mathcal{A}$, representing the set of malicious agents, has control over the conflict relation and the contents of the reviews, while player $\mathcal{B}$ represents the PC chair and is given control only over the assignment of papers to reviewers. Thus

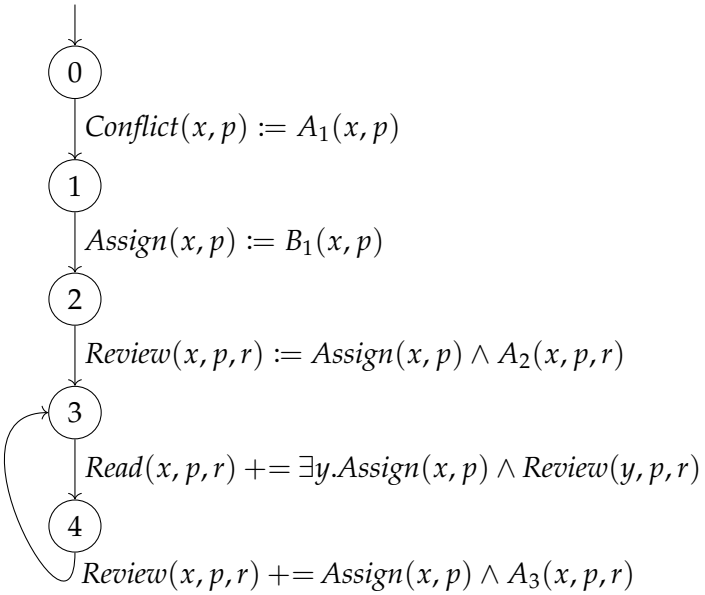


Figure 7.2: FO Safety Game Example

$\mathcal{R}_{\mathcal{A}} = \{A_1, A_2, A_3\}$ and $\mathcal{R}_{\mathcal{B}} = \{B_1\}$.

For notational convenience, we assume that each substitution θ in the control-flow graph contains at most one input predicate, and that all these are distinct[1]. We also consider a partition of the set E of edges into the subsets $E_{\mathcal{A}}$ and $E_{\mathcal{B}}$ where the substitutions at edges from $E_{\mathcal{A}}$ may use state predicates from $\mathcal{R}_{state}$ and an input predicate from $\mathcal{R}_{\mathcal{A}}$, while edges $E_{\mathcal{B}}$ may use state predicates and an input predicate from $\mathcal{R}_{\mathcal{B}}$. Edges without input predicate are by default assigned to player $\mathcal{A}$ and are thus part of $E_{\mathcal{A}}$. Edges in $E_{\mathcal{A}}$ or $E_{\mathcal{B}}$ will also be called $\mathcal{A}$-edges or $\mathcal{B}$-edges, respectively.

For a particular universe U and valuation ν, a trace τ starting in some (v_0, s) with $s, \nu \models \mathsf{Init}$ is considered a *play*. Player $\mathcal{A}$ *wins* a given play τ if the given invariant is violated in some state of the play, i.e. $s, \nu \not\models I[v]$ for some state (v, s) in τ. If the invariant holds in every state of τ, player $\mathcal{B}$ wins τ.

A *strategy* σ for player $\mathcal{B}$ is a mapping which for any given $\mathcal{B}$-edge $e = (u, \theta, v)$ with input predicate B_e of some arity r, universe U, valuation ν, play τ reaching u with state (u, s), returns a relation $B' \subseteq U^r$ to be used as interpretation for the input predicate B_e. Thus, σ provides a choice of the input predicate under the control of player $\mathcal{B}$ for each universe, the history of the play and the upcoming edge in the graph. σ is called *positional* or *memoryless* if it does not depend on the entire history of the play, but only on the current state of the play, i.e. if it depends on the universe U, the valuation ν, the state s and the upcoming $\mathcal{B}$-edge e only.

A play τ *conforms* to a strategy σ for safety player $\mathcal{B}$ if all interpretations of input

[1] In general, edges may use multiple input predicates of the same type. This can be simulated by a sequence of edges that stores the contents of the input relations in fresh auxiliary state predicates from $\mathcal{R}_{state}$ one by one, before realizing the substitution of the initial edge by means of the auxiliary predicates. This enforces that the game is a two-player perfect information game, as otherwise the choices of both players would be hidden and affect the game state at the same time, leading to a game with partial information.

relations under the control of safety player $\mathcal{B}$ occurring in τ are chosen according to σ. The strategy σ is *winning* for $\mathcal{B}$ if $\mathcal{B}$ wins all plays that conform to σ. A FO Safety Game can be *won* by player $\mathcal{B}$ iff there exists a winning strategy for $\mathcal{B}$. In this case, the game is *safe*.

Example 7.3.

In the conference management FO Safety Game shown in Figure 7.2, player $\mathcal{A}$ wants to reach a state where the safety invariant from Example 6.3 is violated: a state where someone reads a review to his own paper before the official release. Formally, for all states u, the invariant $I[u]$ is

$$\forall x, p, r. \neg(\mathit{Conflict}(x, p) \wedge \mathit{Read}(x, p, r))$$

She has control over the predicates A_1, A_2 and A_3 and in turn provides the values for the predicates *Conflict* and *Review* and also determines how often the loop body is iterated. Player $\mathcal{B}$ only has control over predicate B_1 which is used to determine the value of predicate *Assign*. This particular game is *safe*, and player $\mathcal{B}$ has several winning strategies including e.g.,

$$\begin{array}{lcll} B_1(x, p) & := & \neg \mathit{Conflict}(x, p) & \text{or} \\ B_1(x, p) & := & \mathit{false} & \end{array}$$

The second choice is rather trivial. The first choice, on the other hand, which happens to be the *weakest* possible, represents a meaningful strategy.

These definitions are very similar to their counterparts in propositional games — we mention parity games [43] which are already being used in propositional verification and synthesis settings. The game-theoretic aspects of FO Safety Games are similar to their propositional counterparts: They are also 2-player games with perfect information that are positionally determined.

Theorem 7.1. *If there exists a winning strategy for player $\mathcal{B}$, then there also exists a winning strategy that is positional.*

For a fixed universe, a FO Safety Game game can be encoded into a propositional game. Since every propositional game is positionally determined, the FO Safety Game can be won by a combination strategy that does a case distinction over every possible universe:

Proof. Once a universe U is fixed, together with a valuation ν of the globally free variables, a FO Safety Game $\mathcal{G}$ turns into a reachability game $\mathcal{G}_{U,\nu}$ where the positions are given by all pairs $(v, s) \in V \times \mathsf{States}_U$ (controlled by reachability player $\mathcal{A}$) together with all pairs $(s, e) \in \mathsf{States}_U \times E$ controlled by safety player $\mathcal{B}$ if $e \in E_{\mathcal{B}}$ and by $\mathcal{A}$ otherwise. For an edge $e = (v, \theta, v')$ in $\mathcal{G}$, $\mathcal{G}_{U,\nu}$ contains all edges $(v, s) \to (s, e)$, together with all edges $(s, e) \to (v', s')$ where s' is a successor state of s w.r.t. e and ν.

Let $\mathsf{Init}_{U,\nu}$ denote the set of all positions (v_0, s) where $s, \nu \models \mathsf{Init}$, and $I_{U,\nu}$ the set of all positions (v, s) where $s, \nu \models I[v]$ together with all positions (s, e) where $s, \nu \models I[v]$ for $\iota e = v$. Then $\mathcal{G}_{U,\nu}$ is safe iff safety player $\mathcal{B}$ has a strategy $\sigma_{U,\nu}$ to force each play started in some position $\mathsf{Init}_{U,\nu}$ to stay within the set $I_{U,\nu}$. Assuming the axiom of choice for set

theory, the set of positions can be well-ordered. Therefore, the strategy $\sigma_{U,\nu}$ for safety player $\mathcal{B}$ can be chosen positionally, see, e.g., Lemma 2.12 of [66]. Putting all positional strategies $\sigma_{U,\nu}$ for safety player $\mathcal{B}$ together, we obtain a single positional strategy for safety player $\mathcal{B}$ in $\mathcal{G}$. □

For propositional games, strategies only have a finite state space: They map one of the states of the transition system to one of the finitely many actions enabled at this state. For FO Safety Games, strategies are more complicated objects: They can depend on the unbounded size and structure of the universe and all possible instantiations of predicates. This makes representation of strategies very hard in general. In this thesis, we are thus interested in strategies that can be represented in FO Logic. This has the benefit that the strategy can be included into the FO Safety Game itself: Given a FO definable strategy σ, we can replace all occurrences of an input predicate B from $\mathcal{R}_\mathcal{B}$ with the FO Logic formula $\sigma(B)$ and get a FO Transition System that follows the game's semantics for the particular strategy σ.

Theorem 7.1 shows that memoryless strategies suffice to win any FO Safety Game. However, it does not prove whether any safe FO Safety Game allows for a winning strategy which is both memoryless and can also be expressed in FO logic. On the contrary — in general, strategies expressible in FO Logic are not enough to win all safe FO Safety Games.

Theorem 7.2. *There exist safe FO Safety Games where no winning strategy is expressible in FO logic.*

Proof. Consider a game with $\mathcal{R}_{state} = \{E, R_1, R_2\}$, $\mathcal{R}_\mathcal{A} = \{A_1, A_2\}$ and $\mathcal{R}_\mathcal{B} = \{B_1\}$, performing three steps in sequence:

$$\begin{array}{lcl} E(x,y) & := & A_1(x,y); \\ R_1(x,y) & := & B_1(x,y); \\ R_2(x,y) & := & A_2(x,y) \end{array}$$

In this example, reachability player $\mathcal{A}$ chooses an arbitrary relation E, then safety player $\mathcal{B}$ chooses R_1 and player $\mathcal{A}$ chooses R_2. The invariant I ensures that at the endpoint both R_1 and R_2 are at least the transitive closure of E and R_1 is smaller or equal to R_2:

$$\neg \mathsf{closure}(R_2, E) \vee \mathsf{closure}(R_1, E) \wedge \forall x, y. R_1(x,y) \rightarrow R_2(x,y)$$

where $\mathsf{closure}(R, E)$ is given by

$$\forall x, y. R(x,y) \leftrightarrow E(x,y) \ \vee \exists z. R(x,z) \wedge E(z,y)$$

The only winning strategy for safety player $\mathcal{B}$ (computing R_1) is to select the smallest relation satisfying I, which is exactly the transitive closure of E. In this case, no matter what reachability player $\mathcal{A}$ chooses for R_2, safety player $\mathcal{B}$ wins. This winning strategy is however not expressible in FO logic: There is no FO Logic formula that defines the transitive closure of a given relation [53]. □

Despite this negative result, we would like to come up with effective means of computing safe strategies, given that FO definable strategies exist. For this, similar to our approach for FO Transition Systems shown in Chapter 6, we use a *weakest precondition* operator to

characterize the reachable state space along a given path. The difference in the weakest precondition operators for FO Transition Systems and FO Safety Games is the handling of $\mathcal{B}$-edges that are not present in FO Transition Systems. Again, for each path π in the control-flow graph of a first-order safety game $\mathcal{G}$ ending in program point v, and some property Ψ, we define the weakest precondition $[\![\pi]\!]^\top\Psi$ of Ψ by induction on the length of π.

If $\pi = \epsilon$, then $[\![\pi]\!]^\top\Psi = \Psi$. Otherwise, $\pi = e\pi'$ for some edge $e = (v_1, \theta, v')$. Then

$$[\![\pi]\!]^\top\Psi = \begin{cases} \forall A_e.\theta([\![\pi']\!]^\top\Psi) & \text{if } e \text{ } \mathcal{A}\text{-edge} \\ \exists B_e.\theta([\![\pi']\!]^\top\Psi) & \text{if } e \text{ } \mathcal{B}\text{-edge} \end{cases}$$

Then, for any path through a FO Safety Game $\mathcal{G}$ chosen by player $\mathcal{A}$, we can use the weakest precondition to decide if player $\mathcal{B}$ can enforce that a given invariant holds.

Lemma 7.3. *Let π be a path and* Init *some initial condition and Ψ an assertion about the endpoint of π. Safety player $\mathcal{B}$ has a strategy that upholds Ψ for all plays that follow π iff* $\mathsf{Init} \to [\![\pi]\!]^\top\Psi$.

Proof. We proceed by induction on the length of π. If $\pi = \epsilon$, then safety player $\mathcal{B}$ wins all games in universes U with valuations ν starting in states s with $s, \nu \models \mathsf{Init}$ iff $\mathsf{Init} \to \Psi$, and the invariant holds. Now assume that $\pi = e\pi'$ where $e = (u, \theta, v)$. Let $\Psi' = [\![\pi']\!]^\top\Psi$. By inductive hypothesis, safety player $\mathcal{B}$ has a winning strategy for π' with initial condition Ψ'. This means that she can force to arrive at the end point in some state s such that $s, \nu \models \Psi$, given that she can start in some state s' with $s', \nu \models \Psi'$. By case distinction on whether edge e is an $\mathcal{A}$- or a $\mathcal{B}$-edge, this is the case whenever the play starts in some s with $s, \nu \models \mathsf{Init}$.

For the reverse direction, assume that for every universe U and valuation ν chosen by reachability player $\mathcal{A}$, safety player $\mathcal{B}$ can force to arrive at the end point of π by means of the strategy $\sigma_{U,\nu}$ in a state s such that $s, \nu \models \Psi$ whenever the play starts in some state s_0 with $s_0, \nu \models \mathsf{Init}$. Assume that s_0 is an initial state with $s_0, \nu \models \mathsf{Init}$. Again, we perform a case distinction on the first edge e. First assume that e is a $\mathcal{B}$-edge. Let B denote the relation selected by strategy $\sigma_{U,\nu}$ for e. We construct the successor state s_1 corresponding to edge e and relation B. Since safety player $\mathcal{B}$ can win the game on π' when starting in s_1, we conclude by inductive hypothesis that $s_1, \nu \models \Psi'$. This means that $s_0 \oplus \{B_e \mapsto B_1\}, \nu \models \Psi'\theta$ and therefore $s_0, \nu \models \exists B_e.\Psi'\theta$. If e is an $\mathcal{A}$-edge, then for every choice A of reachability player $\mathcal{A}$, we obtain a successor state s_1 such that by inductive hypothesis, $s_1, \nu \models \Psi'$. This means that for all A, $s_0 \oplus \{A_e \mapsto A\}, \nu \models \Psi'\theta$ and therefore also $s_0, \nu \models \forall A_e.\Psi'\theta$. In both cases, $s_0, \nu \models [\![e]\!]^\top([\![\pi']\!]^\top\Psi)$ and the claim follows. □

Now that we know how to deal with single paths, we can generalize this notion to multiple paths. Similar to the initial constraint system for FO Safety Games shown in Section 6.3, we directly intersect the weakest preconditions of all paths starting in a particular node. Let $\mathcal{G}$ denote a game and I an invariant on $\mathcal{G}$. Exactly as for FO Transition Systems in Section 6.3, we use the weakest precondition operator to strengthen I: For $h \geq 0$, we define the assignment $\Psi^{(h)}$ of program points v to formulas by

$$\begin{array}{lcl} \Psi^{(0)}[u] & = & I[u] \\ \Psi^{(h)}[u] & = & \Psi^{(h-1)}[u] \ \wedge \ \bigwedge_{e=(u,\theta,v)\in out(u)} [\![e]\!]^\top(\Psi^{(h-1)}[v]) \text{ for } h > 0 \end{array} \tag{7.1}$$

Just as for FO Transition Systems, this strengthening of the initial invariant contains the preconditions along all possible paths of $\mathcal{G}$ which means the game is safe iff every initial state is inside the strengthening:

Theorem 7.4. *A FO safety game $\mathcal{G}$ is safe iff* $\mathsf{Init} \to \Psi^{(h)}[v_0]$ *holds for all* $h \geq 0$.

Proof. By induction, we verify that

$$\Psi^{(h)}[v] = \bigwedge \{ [\![\pi]\!]^\top I \mid \pi \text{ path starting at } v, |\pi| \leq h \}$$

for all $h \geq 0$. Accordingly, safety player wins on all games of length at most h starting at v_0 iff $\mathsf{Init} \to \Psi^{(h)}[v_0]$ holds. From that, the assertion of the lemma follows. □

The characterization of safety due to Theorem 7.4 is precise — but may require to construct infinitely many $\Psi^{(h)}$.

Whenever, though, the safety game $\mathcal{G}$ is *finite*, i.e., the underlying control-flow graph of G is acyclic, then $\mathcal{G}$ is safe iff $\mathsf{Init} \to \Psi^{(h)}[v_0]$ where h equals the length of the longest path in the control-flow graph of G starting in v_0. As a result, we get that *finite* first-order safety games are as powerful as Second Order logic.

Theorem 7.5. *Deciding a finite FO safety game with predicates from $\mathcal{R}_{state}$ is inter-reducible to satisfiability of SO formulas with predicates from $\mathcal{R}_{state}$.*

Proof. By using the characterization $\Psi^{(h)}$ For the reverse implication, consider an arbitrary closed formula φ in SO Logic. W.l.o.g., assume that φ has no function symbols and is in prenex normal form where no SO Quantifier falls into the scope of a FO quantifier [61]. Thus, φ is of the form $Q_1 C_1 \dots Q_n C_n.\ \psi$ where all Q_n are SO quantifiers and ψ is a relational formula in FO logic.

We then construct a FO safety game $\mathcal{G}$. The set $\mathcal{R}_{state}$ of relations consists of all relations that occur freely in φ together with copies R_i' of all quantified relations C_i. The control-flow graph consists of $n+1$ nodes $v_0, \dots, v_n$, together with edges (v_{i-1}, θ_i, v_i) for $i = 1, \dots, n$. Thus, the maximal length of any path is exactly n — the number of SO quantifiers in φ.

An edge $e_i = (v_{i-1}, \theta_i, v_i)$ is used to simulate the quantifier $Q_i C_i$. The substitution θ_i is the identity on all predicates from $\mathcal{R}_{state}$ except R_i' which is mapped to C_i. If Q_i is a universal quantifier, C_i will be in $\mathcal{R}_{\mathcal{A}}$, and e_i will be an $\mathcal{A}$-edge. Similarly, if Q_i is existential, C_i will be in $\mathcal{R}_{\mathcal{B}}$ and e_i will be a $\mathcal{B}$-edge. Assume that ψ' is obtained from ψ by replacing every relation R_i with R_i'. As FO invariant I, we then use $I[v_i] = \mathit{true}$ for $i = 0, \dots, n-1$ and $I[v_n] = \psi'$. Then $\Psi^{(m)}[v_n] = \varphi$ for all $m \geq n$. Accordingly for $\mathsf{Init} = \mathit{true}$, player $\mathcal{B}$ can win the game iff φ is universally true. □

Theorem 7.5 implies that a FO definable winning strategy for safety player $\mathcal{B}$ (if it exists) can be constructed whenever the SO quantifiers introduced by the choices of the respective players can be eliminated. Theorem 7.5, though, gives no clue on *how* to decide whether or not safety player $\mathcal{B}$ has a winning strategy and if so, whether it can be effectively represented.

7.2 Noninterference for FO Games

In the previous chapters, we have concerned ourselves with checking if a given FO Transition System satisfies Noninterference with Declassification and Agent Model. We now consider the same problem for FO Safety Games, where our goal now is to synthesize FO formulas for dedicated predicates so that altogether a given noninterference property is guaranteed.

Example 7.4.

For the FO Safety Game version of the conference management system in Figure 7.2, no PC member should learn anything about the reports provided for papers for which she has declared conflict of interest. Thus our goal is to devise a strategy for predicate B_1 for the edge between program points 1 and 2 that will be used as the paper assignment which enforces this property.

As for FO Transition System, we again assume that for every predicate R of rank at least 1, agent a *observes* the set of all tuples $\bar{z}$ so that $Ra\bar{z}$ holds. Analogously, we assume that there is a set $\mathcal{R}_{high}$ of input predicates predicates under the control of player $\mathcal{A}$ whose values are meant to be *disclosed* only to privileged agents. These have an attached declassification condition D_{I_h} for relations I_h in $\mathcal{R}_{high}$, which specifies the set of tuples $\bar{y}$ where the value of $O\bar{y}$ may be disclosed to a.

For any agent model $m \in \{s, c, (c,k) \mid k \in \mathbb{N}\}$, we extend the composition transformations $\mathcal{T}_a^{(m)}$ to FO Safety Games. They apply the same structural transformations for games and leave fresh copies of input predicates under the same control as the original predicate: for an input $\mathcal{A}$-predicate A, the fresh copy A' will also be under the control of player $\mathcal{A}$.

The transformations encode the agent models and all declassification conditions, simplifying the safety property φ_a^2 to be verified for $\mathcal{T}_a^{(m)}\mathcal{G}$ to:

$$\bigwedge_{R \in \mathcal{R}_{state}} \forall \bar{z}. Ra\bar{z} \leftrightarrow R'a\bar{z} \tag{7.2}$$

where we assume that the length of the sequence of variables $a\bar{z}$ matches the rank of the corresponding predicate R. We use the same notation and let $\mathcal{T}_a^{(m)}\mathcal{G}$ denote the FO Safety Game obtained from $\mathcal{G}$ in this way.

Example 7.5.

In the FO Safety Game version of the conference management example from Figure 7.2, we are interested in a particular agent a. The predicate A_2 which provides reports for papers, constitutes a predicate whose tuples are only disclosed to agent a if they speak about papers with which a has no conflict. Thus,

$$I_{A_2,a} \;=\; \neg \mathit{Conflict}(a, y_1)$$

The transformed FO Safety Game $\mathcal{T}_a^{(s)}\mathcal{G}$ that encodes declassification conditions for an agent model of only stubborn agents is shown in Figure 7.3.

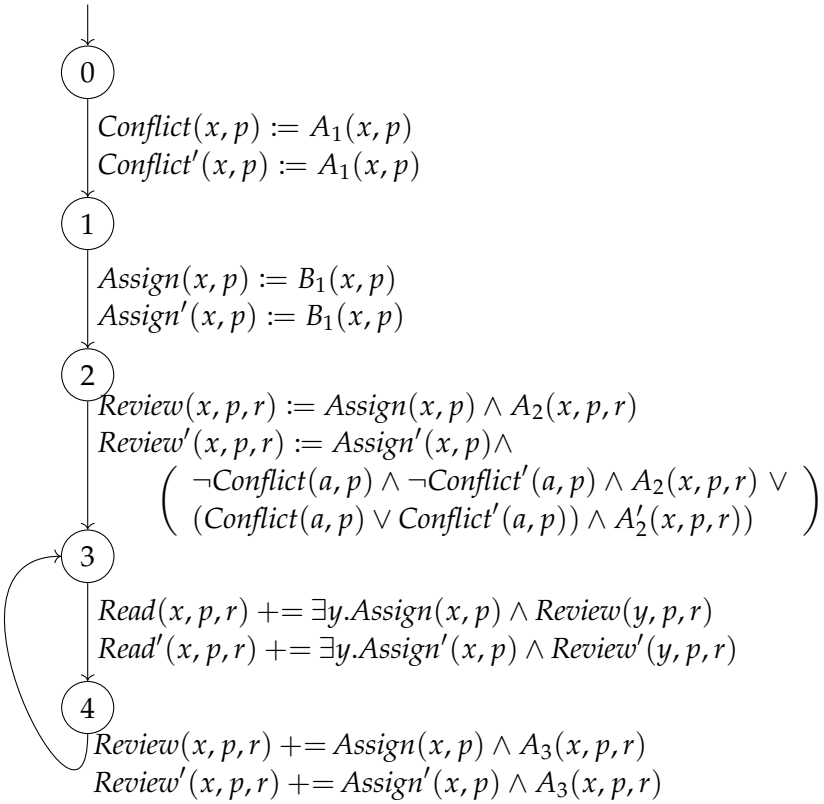


Figure 7.3: Selfcomposition with respect to stubborn agents of the conference management FO Safety Game from Example 7.2

In case that player $\mathcal{B}$ has no choice to make, a slight adaptation of Theorem 6.1 implies that $\mathcal{T}_a^{(m)}\mathcal{G}$ satisfies the invariant (7.2) for all program points u iff φ_a holds for $\mathcal{G}$. This correspondence can be extended to an FO safety game $\mathcal{G}$ where the set of $\mathcal{B}$ predicates $\mathcal{R}_\mathcal{B}$ is non-empty. However, the transformation $\mathcal{T}_a^{(c)}e$ of an edge e using a $\mathcal{B}$-predicate B now uses two $\mathcal{B}$-predicate B and B'. We must ensure that the FO formulas describing the winning strategy σ of the self-composition can be translated back into a meaningful strategy for $\mathcal{G}$. A sufficient condition that allows this is when for each $B \in \mathcal{R}_\mathcal{B}$, $\sigma(B)$ depends only on predicates $R \in \mathcal{R}_{state}$ for which R and R' are equivalent. Then the encoding of the agent model enforces that the same strategy can be used for B and B' as no difference has been observed by anyone yet. More generally, assume that we are given a set $\mathcal{R}_u$ for each program point u where

$$\forall \bar{y}.R\bar{y} \leftrightarrow R'\bar{y} \qquad (R \in \mathcal{R}_u) \tag{7.3}$$

holds whenever program point u is reached. Then the strategy σ for $\mathcal{T}_a^{(m)}\mathcal{G}$ is *admissible* if for each edge (u, θ, v) containing some $B \in \mathcal{R}_\mathcal{B}$, $\sigma(B)$ contains predicates from $R \in \mathcal{R}_u$ only. Due to (7.3), the formulas $\sigma(B)$ and $[\sigma(B)]'$ then are equivalent. Therefore, we obtain:

Theorem 7.6. *Let $\mathcal{G}$ be a FO Safety Game with initial condition* Init *and subset $\mathcal{R}_u \subseteq \mathcal{R}_{state}$ of predicates for each program point u of $\mathcal{G}$. Assume that σ is a strategy so that for each predicate B occurring at some edge (u, θ, v), the FO formula $B\sigma$ only uses predicates from $\mathcal{R}_u$. Then for each agent model $m \in \{s, c\}$ Let $\mathcal{T}_a^{(m)}\mathcal{G}$ denote the corresponding FO Safety Game with respect to agent model m and declassification predicates D_{I_h}, and assume that for each program point u, property* (7.3) *holds whenever u is reached by $\mathcal{T}_a^{(m)}\mathcal{G}\sigma$. Then the following two statements are equivalent:*

1. *$\mathcal{G}\sigma$ satisfies the noninterference property φ_a;*
2. *$\mathcal{T}_a^{(m)}\mathcal{G}\sigma$ satisfies the safety property φ_a^2.*

In particular, each admissible winning strategy for the FO safety game $\mathcal{T}_a^{(m)}\mathcal{G}$ gives rise to a strategy for $\mathcal{G}$ that enforces noninterference.

Finding a strategy that enforces noninterference, thus turns into the synthesis problem for a FO Safety Game— with the extra obligation that potential winning strategies only access subsets of *admissible* predicates only.

Example 7.6.

In the conference management workflow from Figure 7.2 with stubborn agents, a strategy is required at the edge from program point 1 to program point 2. At that point, no secret has yet been encountered. Therefore, *all* predicates are admissible — implying that *any* winning strategy for the FO safety game in Figure 6.1 can be translated back to a strategy which enforces noninterference in $\mathcal{G}$. In particular, we obtain (via $\mathcal{T}_a^{(s)}\mathcal{G}$) that any FO formula ψ_a guarantees noninterference for which $\psi_a \rightarrow \neg Conflict(y_1, y_2)$ holds.

Subsequently, we concentrate on techniques for dealing with FO Safety Games without imposing restrictions onto the construction of strategies (beyond FO definability), and defer the discussion of the extension to the general case to Section 7.7.

7.3 Monadic FO Safety Games

Analogously to the case of FO Transition Systems, assuming that the universe is finite and bounded in size by some $h \geq 0$, FO Safety Games reduce to finite games (of tremendous size, though). Thus, checking invariants as well as the construction of a winning strategy (in case that the game is safe) is effectively possible (and all strategies can be expressed in FO Logic). So here, we again look at systems where all predicates have at most one parameter from an unbounded sort — turning the FO Safety Game into a *monadic* FO Safety Game.

Due to the results of Theorem 7.5 for finite FO Safety Games, we therefore conclude for finite monadic FO Safety Games that safety is decidable. Moreover, in case the game is safe, a positional winning strategy for safety player $\mathcal{B}$ can be effectively computed.

For the general case, the results for monadic FO Transition Systems from Theorem 6.11 carry over, which means that the following cases are undecidable:

Theorem 7.7. *For monadic safety games, safety is undecidable when one of the following conditions is met:*

1. *there are both $\mathcal{A}$-edges as well as $\mathcal{B}$-edges;* [2]
2. *there are $\mathcal{A}$-edges and substitutions with equalities or disequalities;*
3. *there are $\mathcal{B}$-edges and substitutions with equalities or disequalities.*

The proof of statement (2) is the same as for monadic FO Transition System and uses monadic predicates to simulate the counters of a multi-counter machine. Statements (1) and (3) follow from the observation that equality or disequality literals can be replaced by relations under the control of one of the players.

Proof. The proof is very similar to the simulation in the proof of Theorem 6.11. Statement (2) follows directly from the previous proof. In the given simulation, the updates to $f_{l''}$ using equalities in the second step can be replaced by means of a step by player $\mathcal{B}$ together with the substitution:

$$f_{l''} \mapsto f_{l''} \wedge \forall x_1 x_2. \begin{pmatrix} P_i x_1 \vee \neg P' x_1 \vee \\ P_i x_2 \vee \neg P' x_2 \vee \\ B x_1 \vee \neg B x_2 \end{pmatrix}$$

Therefore, statement (1) also follows. A disequality would have served the same purpose if deviation from the correct simulation would have been tracked by means of an error flag. This kind of simulation is exemplified for the proof of statement (3).

For statement (3), we introduce a dedicated error flag *error* and sharpen the invariant to

$$\neg \mathit{error} \wedge \ (\textstyle\bigvee_{j=1}^{n-1} f_j \vee \bigwedge_{j=1}^{n} \neg f_j)$$

The error flag is initially assumed to be *false*, and used to force safety player $\mathcal{B}$ to choose sets B with appropriate properties. Thus, we use

$$\neg \mathit{error} \wedge f_1 \wedge \ \textstyle\bigwedge_{j>1} \neg f_j \wedge \bigwedge_i \forall x. \neg P_i x$$

as initial condition. For the actual simulation, we use a single program point together with edges for each transition of the counter machine. Incrementing counter c_i (combined with state transition from q_l to $q_{l'}$), e.g., is simulated by an edge with the substitution θ':

$$\begin{array}{lcl} P_i y & \mapsto & P_i y \vee B y, \\ f_{l''} & \mapsto & \begin{cases} f_l & \text{if } l'' = l' \\ \mathit{false} & \text{if } l'' \neq l' \end{cases}, \\ \mathit{error} & \mapsto & \mathit{error} \vee (\forall x. \neg B x \vee P_i x) \vee \\ & & (\exists x_1 x_2. \ \neg(B x_1 \wedge \neg P_i x_1) \vee \\ & & \qquad \neg(B x_2 \wedge \neg P_i x_2) \vee x_1 \neq x_2) \end{array}$$

Due to $\neg \mathit{error}$ in the invariant, safety player $\mathcal{B}$ is forced to choose a set B which adds exactly one element to P_i, while the subformula $\bigwedge_j \neg f_j$ forces reachability player $\mathcal{A}$ to choose edges according to the state transitions of the multi-counter machine. □

[2] This statement has been also communicated to us by Igor Walukiewicz

The monadic version of the running example in Figure 7.2 does use $\mathcal{A}$-edges as well as $\mathcal{B}$-edges. [3] However, by substituting any strategy without equalities and disequalities for player $\mathcal{B}$, the given strategy can still be proven *winning* by solving the transformed FO Transition System, which is decidable.

7.3.1 Decidable monadic FO Safety Games

Luckily, not all monadic FO Safety Games are undecidable. In this section we will prove that all safety games that fall outside the fragment mentioned in Theorem 7.7 indeed stay decidable.

For games where no predicate is under the control of either player, the results for FO Transition Systems from Theorem 6.12 carry over.

Theorem 7.8. *Assume that G is a monadic safety game, possibly containing equalities and/or disequalities with $\mathcal{R}_\mathcal{A} = \mathcal{R}_\mathcal{B} = \emptyset$. Then for some $h \geq 0$, $\Psi^{(h)} = \Psi^{(h+1)}$. Therefore, safety of G is decidable.*

Since $\mathcal{R}_\mathcal{B}$ is empty, this proof is the same as the proof for Theorem 6.12 and relies on our variant of the *counting quantifier normal form*. We remark that the given finite upper bound r to the ranks of all formulas $\Psi^{(h)}[v]$ together with the finite model property [20] implies that reachability player $\mathcal{A}$ can win iff $\mathcal{A}$ can win in a a universe of size at most $r'2^{|\mathcal{R}_{state}|}$ where r' is the maximum of r and the rank of Init.

For invariants I that only contain disequalities, the results from Theorem 6.16 apply that retain decidability if no equalities between bound variables are introduced during the weakest precondition computation. This can only be guaranteed if safety player $\mathcal{B}$ does not have control over any predicates. [4]

Theorem 7.9. *Assume that G is a monadic safety game without $\mathcal{B}$-edges (i.e. $\mathcal{R}_\mathcal{B} = \emptyset$) and*

1. *there are no disequalities between bound variables in I,*
2. *in all (positive or negative) equalities $x = y$ or $x \neq y$ in Init and substitutions θ at least one of x, y is from $\mathcal{C}$.*

Then it is decidable whether G is safe.

Again, as $\mathcal{R}_\mathcal{B}$ is empty, this proof is the same as for monadic FO Transition Systems given for Theorem 6.16 and relies on the multiplicity of the encountered formulas.

In a mirrored setting, decidability is retained for invariants that only contain positive equalities if there are no disequalities introduced during the weakest precondition computation. This is only the case when $\mathcal{R}_\mathcal{A} = \emptyset$, i.e., reachability player $\mathcal{A}$ selects universe and control-flow path and safety player $\mathcal{B}$ chooses interpretations of predicates in $\mathcal{R}_\mathcal{B}$. As a consequence, we obtain:

Theorem 7.10. *Assume that G is a monadic safety game without $\mathcal{A}$-edges where*

1. *there are no equalities between bound variables in I,*

[3] Assuming that the number of PC members and papers is bounded, while the number of (versions of) reviews is unbounded.

[4] The simulation of multi-counter machines in the proofs of Theorem 7.7 shows how predicates under the control of player $\mathcal{B}$ can be used to introduce equalities through SO existential quantifier elimination.

2. *in all (positive or negative) equalities $x = y$ in* Init *and substitutions θ at least one of x, y is from $\mathcal{C}$.*

Then it is decidable whether G is safe.

The proof is analogous to the proof of Theorem 6.16 where the abstraction of equalities now is replaced with an abstraction of disequalities, and Corollary 6.18 is replaced with a similar Corollary 7.12 dealing with disequalities.

In analogy to FO Transition System with invariants containing equalities, we provide a weakest strengthening of monadic FO formulas now containing positive occurrences of disequalities only. Let φ denote a closed monadic formula in negation normal form with no positive occurrences of equalities between bound variables. We define $\varphi^\sharp$ now as the formula obtained from φ by replacing each literal $x \neq y$ (x, y bound variables) with

$$\begin{array}{l}(\bigvee_{R\in\mathcal{R}} Rx \wedge \neg Ry \vee \neg Rx \wedge Ry) \vee \\ (\bigvee_{c\in\mathcal{C}} x = c \wedge y \neq c \vee x \neq c \wedge y = c)\end{array} \tag{7.4}$$

Then, $\varphi^\sharp \to \varphi$ holds, and we claim:

Lemma 7.11. *Let ψ be any monadic FO formula without equalities or disequalities between bound variables such that $\psi \to \varphi$ holds. Then also $\psi \to \varphi^\sharp$ holds.*

Proof. We proceed by induction on the structure of φ. Clearly, the assertion holds whenever φ does not contain disequalities between bound variables. Assume that φ is the literal $x \neq y$ for bound variables x, y. Assume that $\psi \to (x \neq y)$, but ψ does not imply formula (7.4). This means that there is a model M and an assignment ρ such that $M, \rho \models \psi \wedge \bigwedge_{R\in\mathcal{R}} Rx \wedge Ry \vee \neg Rx \wedge \neg Ry \wedge \bigwedge_{c\in\mathcal{C}} (x \neq c \vee y = c) \wedge (x = c \vee y \neq c)$ holds. W.l.o.g., M is minimal, i.e., elements which cannot be distinguished by means of predicates in $\mathcal{R}$ or constant interpretations from $\mathcal{C}$, are equal. But then $M, \rho \not\models (x \neq y)$ — in contradiction to the assumption.

Now assume that $\varphi = \varphi_1 \wedge \varphi_2$. Then $\varphi^\sharp = \varphi_1^\sharp \wedge \varphi_2^\sharp$. Let ψ imply φ. Then $\psi \to \varphi_i$ for each i. Therefore, by induction hypothesis, $\psi \to \psi_i^\sharp$ for all i. As a consequence, $\psi \to \varphi^\sharp$.

Now assume that $\varphi = \varphi_1 \vee \varphi_2$. Then $\varphi^\sharp = \varphi_1^\sharp \vee \varphi_2^\sharp$. If ψ implies φ, then for each model M variable assignment ρ, there is some i so that $M, \rho \models \psi \to \varphi_i$. Assume for a contradiction that $\psi \wedge \neg(\varphi_1^\sharp \vee \varphi_2^\sharp)$ is satisfiable. Then there is some model M, assignment ρ so that $M, \rho \models \psi \wedge \neg\varphi_1^\sharp \wedge \neg\varphi_2^\sharp$. In particular, there is some i so that $M, \rho \models \varphi_i \wedge \neg\varphi_1^\sharp \wedge \neg\varphi_2^\sharp$. By inductive hypothesis, $\varphi_i^\sharp \to \varphi_i$ holds. We conclude that $M, \rho \models \varphi_i \wedge \neg\varphi_1 \wedge \neg\varphi_2$ holds — contradiction. Similar arguments also apply to existential and universal quantification in φ. As a consequence, $\psi \to \varphi^\sharp$ holds. □

Corollary 7.12. *Assume that φ, φ' are monadic FO formulas without positive occurrences of equalities between bound variables. Then*

1. $(\varphi \wedge \varphi')^\sharp = \varphi^\sharp \wedge (\varphi')^\sharp$, *and*
2. $(\exists B.\varphi)^\sharp = (\exists B.\varphi^\sharp)^\sharp$

Proof of Theorem 7.10. This proof now works the same way as the proof of Theorem 6.12. With the different abstraction, we still get the main result proving that there are only finitely many monadic formulas that might appear during the computation of $\Psi_0^{(h)}$, which leads to eventual termination. □

Just as for FO Transition Systems, 2-player monadic FO safety games are undecidable in general. However, for games where one of the players does not choose interpretations for any relation, decidability can be salvaged if the safety condition has acceptable equality/disequality literals only and neither Init, nor the transition relation introduce further equality/disequality literals between bound variables.

7.4 Inductive Invariants for FO Safety Games

Even though the general problem of verification is hard for monadic FO games, invariant inference provided us with powerful incomplete algorithms that are able to prove many general FO Transition Systems safe. We will now lift our methods of inferring inductive invariants to the setting of FO Safety Games. Here, just finding an inductive invariant is not enough to prove a given game safe. Additionally, a winning strategy must be provided or automatically extracted.

We recall that an invariant Ψ is called *inductive* iff for all edges $e = (u, \theta, v)$ of a given FO Safety Game $\mathcal{G}$

$$\Psi[u] \to [\![(u, \theta, v)]\!]^{\top}(\Psi[v])$$

In contrast to FO Safety Games, the weakest precondition may now contain Second Order Quantifiers over predicates in $\mathcal{R}_{\mathcal{A}}$ or $\mathcal{R}_{\mathcal{B}}$. Just as for FO Transition Systems, inductive invariants allow to certify safety of FO Safety Games:

Lemma 7.13. *Assume that Ψ is an inductive invariant for FO Safety Game $\mathcal{G}$, and $\Psi[v] \to I[v]$ for all nodes v. Then $\mathcal{G}$ is safe whenever* Init $\to \Psi[v_0]$ *holds.*

For monadic FO Safety Games, this allows us to check if any given invariant is inductive — and if it is, extract a winning strategy from it:

Theorem 7.14. *Assume that $\mathcal{G}$ is a monadic FO Safety Game with initial condition* Init *and invariant I. Assume further that Ψ is a monadic FO invariant, i.e., maps each program point to a monadic formula. Then the following holds:*

1. *It is decidable whether* Init $\to \Psi[v_0]$ *as well as $\Psi[v] \to I[v]$ holds for each program point v;*
2. *It is decidable whether Ψ is inductive, and if so, an FO definable strategy σ can be constructed which upholds Ψ.*

Thus, a monadic FO Safety Game can be proven safe just by providing an appropriate monadic FO invariant Ψ: the winning strategy itself can be effectively computed.

For a general FO Safety Game $\mathcal{G}$, the resulting Second Order formulas might become undecidable. However, any FO Safety Game $\mathcal{G}$ can be proven safe by the following approach:

1. Come up with a candidate invariant Ψ so that
   - $\Psi[v] \to I[v]$ for all nodes v, and
   - Init $\to \Psi[v_0]$ hold;

2. Come up with a strategy σ which assigns some FO formula to each predicate in $\mathcal{R}_{\mathcal{B}}$;

3. Prove that Ψ is inductive for the FO transition system $\mathcal{G}\sigma$ which is obtained from $\mathcal{G}$ by substituting each occurrence of B with $\sigma(B)$ for all $B \in \mathcal{R}_{\mathcal{B}}$.

To simplify the invariant inference for FO Transition Systems, we turned to the domain of purely universal formulas, where the candidate invariant Ψ consists of universal FO formulas only, while I and Init are in the Bernays-Schönfinkel-Ramsey fragment. In that case, verification conditions from (1) are decidable and allow us to effectively check inductivity of a given universal invariant. For FO Safety Games, a similar restriction proves useful: We find that universal invariants together with a universal strategy can be proven inductive:

Theorem 7.15. *Let $\mathcal{G}$ denote a FO Safety Game where each substitution θ occurring at edges of the control-flow graph uses non-nested FO quantifiers only. Let Ψ denote a universal FO invariant for $\mathcal{G}$, i.e., $\Psi[v]$ is a universal FO formula for each node v. Assume that no $B \in \mathcal{R}_{\mathcal{B}}$ occurs in the scope of an existential FO quantifier, and σ is a strategy which provides a universal FO formula for each $B \in \mathcal{R}_{\mathcal{B}}$. Then it is decidable whether or not Ψ is inductive for $\mathcal{G}\sigma$.*

Theorem 7.15 states that (under mild restrictions on the substitutions occurring at $\mathcal{B}$-edges), the candidate invariant Ψ can be checked for inductiveness — at least when a positional strategy for player $\mathcal{B}$ is provided which is expressed by means of universal FO formulas. This collapses $\mathcal{G}$ into a FO Transition System by replacing all occurrences of predicates under the control of player $\mathcal{B}$. Then Theorem 6.5 is applicable and shows that the verification conditions all fall into the BSR fragment of First-order Logic.

The question remains how for a given invariant I a suitable inductive invariant can be inferred that certifies I. For FO Transition Systems, we solved this by iteratively computing the sequence $\Psi^{(h)}, h \geq 0$ as in (7.1) which, in general, may never reach a fixpoint, but terminates for several classes of FO Transition Systems. The same approach can be applied to FO Safety Games and Theorem 6.9 tells us that fixpoint iteration will always terminate if the fixpoint formula is definable in FO logic. For the case of *monadic* FO Safety Games, this means that the corresponding infinite conjunction is not always FO definable — otherwise decidability would follow.

In general, not every invariant I can be strengthened to an inductive Ψ and universal strategies need not be sufficient to win a universal safety game. Nonetheless, there is a variety of non-trivial cases where existential SO quantifiers can be effectively eliminated together with an explicit construction of the corresponding strategy, e.g., by SO quantifier elimination algorithms SCAN or DLS* (see the overview in [40]). In addition, Theorem 7.15 tells us that for proving inductiveness, it is not necessary to perform *exact* quantifier elimination. Instead, it may suffice to provide an appropriate *strengthening*. Techniques for such *approximate* SO existential quantifier elimination are provided in Section 7.5 and applied in Section 7.6.

7.5 Hilbert's Choice Operator for Second Order Quantifiers

As we have seen in Section 7.4, checking whether a universal FO invariant is inductive can be reduced to SO quantifier elimination. While universal SO quantification can always be removed from formulas with only universal FO quantifiers according to fact 6.5.1, this is not necessarily the case for existential SO quantification. As already observed by Ackermann in [2], the formula

$$\exists B.\, Ba \wedge \neg Bb \wedge \forall x, y. \neg Bx \vee \neg Rxy \vee B\bar{y}$$

expresses that b is not reachable from a via the edge relation R and is not expressible in FO logic. This negative result, though, does not exclude that in a variety of meaningful cases equivalent FO formulas can be constructed. In this section, we recall basic facts about SO existential quantifier elimination together with the notion of a SO *Hilbert choice operator*.

This operator will allow us to extract specific FO definitions for the existentially quantified input predicates. For purely universal FO formulas, we introduce a sequence of candidate substitutions for approximating this operator. We also show that for these formulas, the construction of a SO Hilbert choice operator and thus the construction of a winning strategy for safety player $\mathcal{B}$ from an inductive universal FO invariant, is reducible to SO existential quantifier elimination itself. For an in-depth treatment on SO existential quantifier elimination, we refer to [40].

Let φ denote some universally quantified formula, possibly containing a predicate B of arity r. Let $\bar{y} = y_1 \dots y_r$ and $\bar{y}' = y'_1 \dots y'_r$ be well-sorted sequences of fresh variables. We remark that *any* formula ψ with free variables from $\bar{y}$ can be seen as a definition of B:

$$\varphi[\psi/B] \rightarrow \exists B.\varphi$$

holds for any such ψ. Here, this SO substitution means that every literal Bz and every literal $\neg B\bar{z}'$ is replaced with $\psi[\bar{z}/\bar{y}]$ and $\neg\psi[\bar{z}'/\bar{y}']$, respectively. Let $H_{B,\varphi}$ denote the set of *all* FO formulas ψ such that φ holds when using ψ as a definition for all B literals: This is exactly the set of all formulas ψ s.t. $\exists B.\varphi$ is equivalent to $\varphi[\psi/B]$. Given B and ψ, a general construction of some FO formula $\psi \in H_{B,\varphi}$ is an instance of Hilbert's (Second Order) *choice* operator. If one exists, we write $\psi = \mathcal{H}_{\mathcal{B}}(\varphi)$.

The strongest solution to fix the paper assignment in the conference management example from Example 7.2 is that the PC chair decides not to assign papers to *any* PC member. While guaranteeing safety, this choice is not very useful in practice. The *weakest* choice on the other hand, provides us here with a decent strategy as it allows for more behaviours. In the following we therefore will aim at constructing as *weak* strategies as possible. This follows the intuition that safety player $\mathcal{B}$ wants to ensure safety while still allowing as many possible scenarios as possible.

7.5.1 Ackermannian FO Safety Games

There are several classes of Second Order formulas that allow for Second Order Quantifier Elimination. For example, in purely universal formulas in *simple* normal form as defined in Equation (2.1), existential SO quantifiers can be eliminated:

Lemma 7.16 (Ackermann's lemma [2]). *Assume that φ is in simple normal form, i.e. $\varphi = E \wedge (\forall \bar{y}.F \vee B\bar{y}) \wedge (\forall \bar{y}.G \vee \neg B\bar{y})$. Then we have:*

1. *$\exists B.\varphi = E \wedge (\forall \bar{y}.F \vee G)$;*
2. *For every FO formula ψ, $\exists B.\varphi \leftrightarrow \varphi[\psi/B]$ iff $(E \wedge \neg F) \rightarrow \psi$ and $\psi \rightarrow (\neg E \vee G)$.* □

According to Lemma 7.16, formulas in simple normal form admit a SO Hilbert choice operator, which we define as

$$\mathcal{H}_B(\varphi) = \neg E \vee G \tag{7.5}$$

By this definition, $\mathcal{H}_B(\varphi)$ equals the *weakest* formula ψ for which $\exists B.\varphi$ is equivalent to $\varphi[\psi/B]$.

Example 7.7.

Consider the invariant I from Example 6.3 that specifies the safety property that no PC member can ever read a review for a paper he declared Conflict with. Formally $I[u]$ for all u is

$$I[u] = \forall x, p, r.\neg(\mathit{Conflict}(x, p) \wedge \mathit{Read}(x, p, r)) \tag{7.6}$$

The weakest precondition w.r.t. the second statement amounts to:

$$\exists B_1.\forall x, p.\neg \mathit{Conflict}(x, p) \vee \neg B_1(x, p) = \mathit{true}$$

where we can choose for B_1 any formula ψ (with free variables x, p) which implies $\neg \mathit{Conflict}(x, p)$. Following our earlier definition $\mathcal{H}_{B_1} = \neg \mathit{Conflict}(x, p)$.

Ackermann's Lemma gives rise to a nontrivial class of safety games where existential SO quantifier elimination succeeds. We call $B \in \mathcal{R}_{\mathcal{B}}$ *ackermannian* in the quantifier-free substitution θ iff the for every predicate $R \in \mathcal{R}_{state}$, the CNF of $\theta(R)$ does not contain clauses with both positive and negative B literals.

Theorem 7.17. *Assume we are given a FO Safety Game $\mathcal{G}$ where all substitutions contain non-nested quantifiers only, and a universal inductive invariant Ψ. Assume further that the following holds:*

1. *All predicates B under the control of safety player $\mathcal{B}$ occur in quantifier-free substitutions only and are ackermannian;*
2. *For every $\mathcal{B}$-edge $e = (u, \theta, v)$, every clause of $\Psi[v]$ contains at most one literal with a predicate R where $\theta(R)$ has a predicate from $\mathcal{R}_{\mathcal{B}}$.*

Then the weakest FO strategy for safety player $\mathcal{B}$ can be effectively computed for which Ψ is inductive.

The proof is by showing that all Second Order quantifiers introduced by the weakest precondition operator can be eliminated according to Lemma 7.16.

Example 7.8.

Consider the leader election protocol from Example 7.1, together with the inductive invariant from [75]. The complete specification can be found in our suite of examples [67]. Therein, the predicate B is ackermannian, and *msg* appears once in two different clauses of the invariant. Thus by Theorem 7.17, the weakest safe strategy for player $\mathcal{B}$ can be effectively computed. Our solver, described in Chapter 8 finds it to be

$$B(a,i,b) := \neg E \vee \neg next(a,b) \vee \left(\begin{array}{l} \forall n.\ (i \geq n \vee b \neq i) \wedge \\ \forall n.\ (\neg between(b,i,n) \vee i > n) \end{array} \right)$$

where E axiomatizes the ring architecture, i.e., the predicate *between* as the transitive closure of *next* together with the predicate $\leq$. The given strategy is weaker than the intuitive (and also safe) strategy of $(i = a)$, and allows for more behaviors — for example it allows a to send messages that are greater than its own id in case they are not greater than the ids of nodes along the way from b back to a. It also allows any i in case that b is not the next node in the ring, since messages are only sent if this is the case.

7.5.2 Iterative Approximation of Hilbert's Second Order Choice Operator

In general, though, existential SO quantifier elimination must be applied to universally quantified formulas which cannot be brought into simple normal form. To deal with general formulas, we provide a sequence of *candidates* for the Hilbert choice operator which provide the *weakest* Hilbert choice operator — whenever it is FO definable.

Consider a predicate B in a purely universal formula φ in normal form (2.1), i.e. $\varphi =$

$$E \wedge (\forall \bar{y}.F \vee B\bar{y}) \wedge (\forall \bar{y}'.G \vee \neg B\bar{y}') \wedge (\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}')$$

Therein, the sub-formula H can be understood as a binary predicate between the variables $\bar{y}'$ and $\bar{y}$ which may be composed, iterated, post-applied to predicates on $\bar{y}'$ and pre-applied to predicates on $\bar{y}$. The number of compositions of H is governed by the structure of B. We define the formula H^k, $k \geq 0$ with free variables from $\bar{y}, \bar{y}'$ to be an approximation of the last clause of φ after elimination of the predicate B.

$$\begin{array}{rcll} H^0 & = & \bar{y} \neq \bar{y}' & \\ H^k & = & \forall \bar{y}_1.H^{k-1}[\bar{y}_1/\bar{y}'] \vee H[\bar{y}_1/\bar{y}] & \text{for } k > 0 \end{array}$$

We remark that by this definition,

$$H^{k+l} \quad = \quad \forall \bar{y}_1.H^k[\bar{y}_1/\bar{y}'] \vee H^l[\bar{y}_1/\bar{y}]$$

for all $k, l \geq 0$. Furthermore, we define the formulas:

$$\begin{array}{rcl} G \circ H^k & = & \forall y.G[\bar{y}/\bar{y}'] \vee H^k \\ G \circ H^k \circ F & = & \forall \bar{y}'.(G \circ H^k) \vee F[\bar{y}'/\bar{y}] \end{array}$$

By applying k distinct copies of H in this way, we can approximate an equivalent formula to φ with the predicate B eliminated:

Lemma 7.18. *1. For all $k \geq 0$, $\exists B.\varphi$ implies $E \wedge G \circ H^k \circ F$;*

2. If $\exists B.\varphi$ implies some FO formula φ, then for some $k \geq 0$, $E \wedge \bigwedge_{i=0}^{k} G \circ H^i \circ F$ implies $\exists B.\varphi$;

3. If $\exists B.\varphi$ is equivalent to some FO formula, then it is equivalent to $E \wedge \bigwedge_{i=0}^{k} G \circ H^i \circ F$ for some $k \geq 0$. □

Starting from G and iteratively composing with H, provides us with a sequence of candidate SO Hilbert choice operators. Let

$$\gamma_k \quad = \quad \neg E \vee \bigwedge_{i=0}^{k} (G \circ H^i)[\bar{y}/\bar{y}'] \tag{7.7}$$

for $k \geq 0$. The candidate γ_k takes all *i*fold compositions of H with $i \leq k$ into account. Then the following holds:

Lemma 7.19. *For every $k \geq 0$,*

1. $\varphi[\gamma_k/B]$ implies $\exists B.\varphi$;

2. $\gamma_{k+1} \rightarrow \gamma_k$, and if $\gamma_k \rightarrow \gamma k+1$, then $\varphi[\gamma_k/B] = \exists B.\varphi$.

Proof. The first statement follows by definition. For a proof of the second statement, assume that $\gamma_k = \gamma_{k+1}$. This means that $\bigwedge_{i=0}^{k}(\neg E \vee G) \circ H^i = \bigwedge_{k=0}^{k+1}(\neg E \vee G) \circ H^{k+1}$. We therefore conclude that in particular $\bigwedge_{i=0}^{k}(\neg E \vee G) \circ H^i = \bigwedge_{i=0}^{r}(\neg E \vee G) \circ H^i$ for all $k \leq r$ — implying that $\exists B.\varphi = E \wedge \bigwedge_{i=0}^{k} G \circ H^i \circ F$ holds. Furthermore, we calculate:

$$\begin{array}{rcl} \varphi[\gamma_k/B] & = & E \wedge (\bigwedge_{i=0}^{k} \forall \bar{y}.(\neg E \vee G \circ H^i[\bar{y}/\bar{y}'] \vee F) \wedge \\ & & (\forall \bar{y}'.G \vee \neg \bigwedge_{i=0}^{k}(\neg E \vee G) \circ H^i) \wedge \\ & & (\forall \bar{y}\bar{y}'.H \vee \bigwedge_{i=0}^{k}(\neg E \vee G) \circ H^i[\bar{y}/\bar{y}'] \vee \neg(\bigwedge_{i=0}^{k}(\neg E \vee G) \circ H^i)) \end{array}$$

Let us first concentrate on the third clause. Since $(\neg E \vee G) \circ H^0 = \neg E \vee G$, the negated conjunction has $E \wedge \neg G$ as a disjunct. Therefore, the conjunction of that clause with E is equivalent to E. For the fourth clause, we calculate:

$$\begin{array}{rcl} \forall \bar{y}.H \vee \bigwedge_{i=0}^{k}(\neg E \vee G \circ H^i[\bar{y}/\bar{y}'] & = & \bigwedge_{i=0}^{k} G \circ H^{i+1} \\ & = & \bigwedge_{i=1}^{k+1}(\neg E \vee G) \circ H^i \\ & \leftarrow & \bigwedge_{i=0}^{k}(\neg E \vee G) \circ H^i \end{array}$$

Accordingly, the fourth clause also must necessarily be equal to *true*. We conclude that

$$\begin{array}{rcl} \varphi[\gamma_k/B] & = & E \wedge \bigwedge_{i=0}^{k} \forall y.G \circ H^i[\bar{y}/\bar{y}'] \vee F \\ & = & \exists B.\varphi \end{array}$$

which is the statement we wanted to show. □

As a result, the γ_k form a decreasing sequence of candidate strategies for safety player $\mathcal{B}$. We remark, though, that the condition $\bigwedge_{i=1}^{k} G \circ H^i = \bigwedge_{i=1}^{k+1} G \circ H^i$ from statement (2) of Lemma 7.19, is *sufficient* but not necessary for $\exists B.\varphi'$ to be FO definable.

The scenario becomes considerably easier whenever the variables from $\bar{y}$ and $\bar{y}'$ never appear together in any literal. For this case, we prove that the number of times H has to be applied remains bounded:

Lemma 7.20. *Assume that H in the definition 2.1 is of the form*

$$H \quad = \quad \bigwedge_{j=1}^{n}(H_j \vee H'_j \vee G_j) \tag{7.8}$$

for some $n \geq 1$ where all formulas H_j only contain free variables from $\bar{y}$, H'_j only variables from $\bar{y}'$ and G_j none of those. Then

$$\bigwedge_{j=0}^{n} H^j = \bigwedge_{j=0}^{n+r} H^j$$

for all $r \geq 0$.

Proof. For each $k \geq 1$, we have that

$$\begin{array}{rcl} H^k & = & \bigwedge_{j_1 \ldots, j_k}(H_{j_1} \vee H'_{j_k} \vee G_{j_k} \vee \\ & & \bigvee_{i=1}^{k-1} G_{j_i} \vee \forall \bar{y}_i.H'_{j_i}[\bar{y}_i/\bar{y}'] \vee H_{j_{i+1}}[\bar{y}_i/\bar{y}]) \end{array}$$

where for each $i = 1, \ldots, k$, $j_i \in \{1, \ldots, n\}$. Thus, $\bigwedge_{j=0}^{k} H^k$ is the conjunction of all formulas $H_{j_1} \vee H'_{j_r} \vee \bigvee_{i=1}^{r-1}(G_{j_i} \vee \forall \bar{y}_i.H'_{j_i}[\bar{y}_i/\bar{y}'] \vee H_{j_{i+1}}[\bar{y}_i/\bar{y}]) \vee G_{j_r}$ for some $r \leq k$. Now assume for a contradiction, $\bigwedge_{j=0}^{n+2} H^j \neq \bigwedge_{j=0}^{n+1} H^j$ holds. Then there is a clause

$$\begin{array}{rcl} c & = & H_{j_1} \vee H'_{j_{n+2}} \vee G_{j_{n+2}} \vee \\ & & \bigvee_{i=1}^{n+1}(G_{j_i} \vee \forall y_i.H'_{j_i}[\bar{y}_i/\bar{y}'] \vee H_{j_{i+1}}[\bar{y}_i/\bar{y}]) \end{array}$$

which is not implied by $\bigwedge_{j=0}^{n+1} H^j$. On the other hand, there is a sequence $j_1 \ldots j_{n+1}$ positions $r < r'$ exist so that $j_r = j_{r'}$. It follows that we can construct

$$\begin{array}{rcl} c' & = & H_{j_1} \vee H'_{j_{n+2}} \vee G_{j_{n+2}} \vee \\ & & \bigvee_{i=1}^{r-1}(G_{j_i} \vee \forall \bar{y}_i.H'_{j_i}[\bar{y}_i/\bar{y}'] \vee H_{j_{i+1}}[\bar{y}_i/\bar{y}]) \vee \\ & & \bigvee_{i=r'}^{n+1}(G_{j_i} \vee \forall \bar{y}_i.H'_{j_i}[\bar{y}_i/\bar{y}'] \vee H_{j_{i+1}}[\bar{y}_i/\bar{y}]) \end{array}$$

By construction, $c' \Longrightarrow c$. Moreover, c' occurs in the conjunction $\bigwedge_{j=0}^{n+2} H^j$. Accordingly, $\bigwedge_{j=0}^{n+2} H^j \Longrightarrow c$ — in contradiction to our assertion. □

We close this section by noting that there is a SO Hilbert choice operator which can be expressed in SO logic itself. The following theorem is related to Corollary 6.20 of [40], but avoids the explicit use of fixpoint operators in the logic.

Theorem 7.21. *The weakest Hilbert choice operator $\mathcal{H}_B\varphi$ for the universal formula* (2.1) *is definable by the SO formula:*

$$\neg E \vee \exists B.B\bar{y} \wedge (\forall \bar{y}'.G \vee \neg B\bar{y}') \wedge (\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}')$$

Proof. Our goal is to prove that $\exists B.\varphi$ implies the formula $\varphi[\mathcal{H}_B\varphi/B]$. We consider each conjunct of φ in turn.

$$\begin{array}{rcl} \forall \bar{y}.F \vee \mathcal{H}_B\varphi & = & \forall \bar{y}.\exists B.F \vee (B\bar{y} \wedge \\ & & (\forall \bar{y}'.G \vee \neg B\bar{y}') \wedge (\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}')) \\ & \leftarrow & \forall \bar{y}.\exists B.(F \vee B\bar{y}) \wedge \\ & & (\forall y'.G \vee \neg B\bar{y}') \wedge (\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}') \\ & \leftarrow & \exists B.\forall y.(F \vee B\bar{y}) \wedge \\ & & (\forall y'.G \vee \neg B\bar{y}') \wedge (\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}') \\ & = & \exists B.\varphi \end{array}$$

$$\begin{aligned}
\forall \bar{y}'.G \vee \neg \mathcal{H}_B \varphi &= \forall \bar{y}'.\forall B.G \vee \neg B\bar{y}' \vee \\
&\quad (\exists \bar{y}'.B\bar{y}' \wedge \neg G) \vee (\exists \bar{y}\bar{y}'.\neg H \wedge \neg B\bar{y} \wedge B\bar{y}') \\
&= \forall B.\forall \bar{y}'.G \vee \neg B\bar{y}' \vee \\
&\quad (\exists \bar{y}'.\neg G \wedge B\bar{y}') \vee (\exists \bar{y}\bar{y}'.\neg H \wedge \neg B\bar{y} \wedge B\bar{y}') \\
&= \top
\end{aligned}$$

$$\begin{aligned}
&\forall \bar{y}\bar{y}'.H \vee \mathcal{H}_B \varphi \vee \neg \mathcal{H}_B \varphi[\bar{y}'/\bar{y}] \\
=\ & \forall \bar{y}\bar{y}'.H \vee \\
& (\exists B.B\bar{y} \wedge (\forall y'.G \vee \neg B\bar{y}') \wedge (\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}')) \vee \\
& (\forall B.\neg B\bar{y}' \vee \neg(\forall y'.G \vee \neg B\bar{y}') \vee \neg(\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}')) \\
\leftarrow\ & \forall \bar{y}\bar{y}'.\forall B.H \vee \\
& B\bar{y} \wedge (\forall y'.G \vee \neg B\bar{y}') \wedge (\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}') \vee \\
& \neg B\bar{y}' \vee \neg(\forall y'.G \vee \neg B\bar{y}') \vee \neg(\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}') \\
=\ & \forall \bar{y}\bar{y}'.\forall B.H \vee B\bar{y} \vee \neg B\bar{y}' \vee \\
& \neg B\bar{y}' \vee \neg(\forall y'.G \vee \neg B\bar{y}') \vee \neg(\forall \bar{y}\bar{y}'.H \vee B\bar{y} \vee \neg B\bar{y}') \\
=\ & \top
\end{aligned}$$

Altogether therefore, $\varphi[\mathcal{H}_B \varphi] \Longrightarrow \exists B.\varphi$, and the assertion follows. □

The weakest Hilbert choice operator itself can thus be obtained by SO existential quantifier elimination.

In this section we provided a method to extract the weakest universal FO definition for existentially quantified Second Order variables in a given universal formula. For *ackermannian* FO Safety Games, we showed that the weakest universal strategy can always be computed (if it exists). For FO Safety Games outside this class, we gave a fixpoint approach that will construct a FO universal formula if one exists, but might diverge in case it does not. Additionally, we gave a simple syntactic criterion that implies termination of the fixpoint approach.

For a given FO Safety Game $\mathcal{G}$ together with an inductive universal invariant, this approach allows us to automatically extract the weakest universal strategy for safety player $\mathcal{B}$ that upholds the given invariant.

7.6 Approximation and Refinement

While in general safety of FO safety games is undecidable, this does not exclude that in meaningful cases, safety can in fact be proven or dis-proven. In this section, we show how FO safety games can be approximated by simpler FO safety games and, furthermore, how to *refine* a given abstraction when safety of the game can neither be proven nor disproven with the current abstraction. For disproving safety, we rely on *finite* counter-examples, i.e., finite paths in the control-flow graph together with states from finite universes. In comparison to the iteration we established to prove FO Transition Systems safe, we will now generalize the iteration from Chapter 6 to FO Safety Games and provide additional guarantees during the iteration. Namely, we will look at proofs for invariants that are inductive in all universes up to a fixed size.

As we already stated for positional determinacy, a FO safety game G played in a fixed finite universe U is effectively propositional. Accordingly, it is decidable whether or not

safety player $\mathcal{B}$ has a winning strategy for U. In particular:

Lemma 7.22. *For every $t \geq 0$, it is decidable whether safety player $\mathcal{B}$ wins all plays in universes of size at most t.*

Proof. It is possible to enumerate all universes up to size t and solve the game for each one individually. However, instead of enumerating all universes up to size t, we can solve the game *symbolically*. In order to do so, we consider the *domain closure* axiom D_t (considered for example in [88]) for a finite set $\mathcal{C}$:

$$\begin{aligned} D_t \;=\; & \forall x_1 \dots x_{t+1}. \bigvee_{i=1}^{t} \bigvee_{j=i+1}^{t+1} x_i = x_j \wedge \\ & \bigwedge_{c \in \mathcal{C}} \bigvee_{i=1}^{t-1} \bigvee_{j=i+1}^{t} x_i = x_j \vee \bigvee_{j=1}^{t} x_j = c \end{aligned} \tag{7.9}$$

which expresses that the universe has at most t elements. It allows to transform each SO formula φ into a quantifier-free formula $\mathcal{B}_t[\varphi]$ with free FO variables from $\mathcal{C}$ and $\bar{y} = y_1 \dots y_t$, such that

$$\varphi \wedge D_t \leftrightarrow (\exists \bar{y}. \mathcal{B}_t[\varphi] \wedge \bar{D}_t) \tag{7.10}$$

where the formula

$$\bar{D}_t = \;\; \bigwedge_{x \in \mathcal{C}} \bigvee_{j=1}^{t} x = y_j \tag{7.11}$$

states that each constant from $\mathcal{C}$ may only take one of the values provided by $\bar{y}$. The transformation $\mathcal{B}_t$ replaces each universal FO quantification in φ by a conjunction and each existential FO quantification by a disjunction where the fresh variables $y_1, \dots, y_t$ represent the possibly distinct elements of the universe.[5] As a consequence, all SO quantifiers can be eliminated as well.

By means of the transformation $\mathcal{B}_t$, we obtain from the iteration in Section 7.1 a fixpoint computation where all encountered formulas are quantifier-free and use the FO constants from G as well as the FO variables from y. Up to equivalence, the number of such formulas is finite. Therefore, the obtained iteration effectively terminates with some assignment $\overline{\Psi}_t^{(h)}$. It remains to verify whether

$$\forall y. \neg \bar{D}_t \vee \neg \mathcal{B}_t[\mathsf{Init}] \vee \overline{\Psi}_t^{(h)}$$

holds. □

Equation (7.10) inspires us to a general definition. We call closed formulas φ, φ' *equivalent* up to universe size $t \geq 0$ iff

$$\varphi \wedge D_t \leftrightarrow \varphi' \wedge D_t$$

In this case, we also write $\varphi \;\equiv_t\; \varphi'$. The following three statements are equivalent:

1. $\varphi \;\equiv_t\; \varphi'$;
2. $\mathcal{B}_t[\varphi] \wedge \bar{D}_t \;\leftrightarrow\; \mathcal{B}_t[\varphi'] \wedge \bar{D}_t$;
3. For each universe U of cardinality at most t, every model s over universe U for the predicates in $\mathcal{R}_{state}$ and every variable assignment ν for the constant interpretations in $\mathcal{C}$, $s, \nu \models \varphi$ iff $s, \nu \models \varphi'$.

[5] In the presence of sorts $s_1 \dots s_n$, for the sake of the proof we encode them into predicates $S_1 \dots S_n$ beforehand.

For universes of bounded size, the sequence of candidates for Hilberts Choice Operator given in Section 7.5 is *precise*:

Lemma 7.23. *Let φ denote some universal FO formula in normal form* (2.1)*, let B be a predicate symbol of arity r, and let γ_k be defined as in Section 7.5.2. Then*

$$\varphi[\gamma_k/B] \;\equiv_t\; \exists B.\varphi$$

whenever $k \geq t^r$ holds.

Proof. Consider the predicate H with free variables from y, y'. Let U be a universe of cardinality at most t. Let s be a model over U and ν an assignment of the free FO variables of φ to U. Then w.r.t. s and ν, H can be considered as a relation $H_U \subseteq U^r \times U^r$. Therefore, $s, \nu \models \forall yy'.H^k \leftrightarrow H^{k+1}$ holds. If φ is in normal form (2.1), then

$$\begin{array}{lcl} s,\nu & \models & \varphi[\gamma_k/B] \\ & = & E \wedge \gamma_k \circ F \wedge (\forall y'.G \vee \neg\gamma_k) \wedge \\ & & (\forall y'.(\bigwedge_{i=0}^{k} G \circ H^i \circ H) \vee \bigvee \neg(\bigwedge_{i=0}^{k} G \circ H^i)) \\ & = & E \wedge \gamma_k \circ F \wedge \\ & & (\forall y'.(\bigwedge_{i=1}^{k} G \circ H^i) \vee \bigvee \neg(\bigwedge_{i=0}^{k} G \circ H^i)) \\ & = & E \wedge \gamma_k \circ F \\ & \leftarrow & \exists B.\varphi \end{array}$$

and the assertion follows. □

Let us re-consider the case where we are given a candidate invariant Ψ which we want to prove inductive and, if so, construct a winning strategy for safety player $\mathcal{B}$. Assume further that all substitutions at $\mathcal{A}$-edges have non-nested FO quantifiers only, and all substitutions at $\mathcal{B}$-edges are quantifier-free. According to Theorem 7.15, it can effectively be decided for each $\mathcal{A}$-edge $e = (v, \theta, v')$ that $\Psi[v] \rightarrow \forall A.(\Psi[v']\theta)$ holds.

It remains to verify the corresponding conditions for each $\mathcal{B}$-edge $e = (v, \theta, v')$. Lemma 7.22 together with Lemma 7.23 allow us to search for a winning strategy by *counter-example guided abstraction refinement* (CEGAR) (see, e.g., [27]). For that, we establish the following iteration.

(0) Set $t := t_0$ for some initial threshold $t_0 \geq 0$.

(1) Check whether $\Psi[v] \rightarrow \exists B.(\Psi[v']\theta)$ for all universes of cardinality at most t. If not, output *not inductive* and stop.

(2) Construct γ_k for $\Psi[v']\theta$ as in Section 7.5.2 for $k = t^r$. By construction, γ_k is a universal FO formula. Therefore, it can be effectively checked whether $\Psi[v] \rightarrow (\Psi[v']\theta)[\gamma_k/B]$ holds.

If so, output *inductive* and also return γ_k as strategy for player $\mathcal{B}$. If not, $\Psi[v] \wedge \neg(\Psi[v']\theta)[\gamma_k/B]$ is satisfiable where, by Lemma 7.23, the cardinality t' of the universe required by the counter-example, necessarily exceeds t. Then set $t := t'$ and proceed with step (1).

Thus, we have obtained a practical means to verify or refute inductivity of a given FO universal candidate invariant.

In general, it is more involved to infer the inductive invariant itself. In order to come up with a practical approach that works for *general* FO safety games $\mathcal{G}$, we again rely on *abstraction*, i.e., strengthening of formulas. In contrast to FO Transition Systems, here we need to abstract both FO as well as SO existential quantifiers.

Theorem 7.24. *Let $t \geq 0$ and $\bar{y} = y_1 \dots y_t$. For every SO formula φ, a universal FO formula $\mathcal{T}_t[\varphi]$ can be constructed with additional free FO variables from $\bar{y}$ such that*

1. $\forall \bar{y}.(\mathcal{T}_t[\varphi] \rightarrow \varphi)$; *and*
2. $\varphi \equiv_t \exists \bar{y}.\mathcal{T}_t[\varphi]$.

Proof. W.l.o.g., assume that φ is in prenex as well as in negation normal form. The fresh variables from y are meant to represent the elements of a universe U of size at most h.

We remark that the transformation $\mathcal{B}_t$ cannot be used as it does not satisfy requirement (1). The best universal abstraction $\varphi^\sharp$ from Section 6.7.2 can be used and yields a correct algorithm. It does not come with guarantees up to a specific universe size t, and does not satisfy requirement (2). Here, we will thus use a slightly different approximation for existential quantifiers: Instead of replacing them with a disjunction over all universally quantified variables in scope, we replace them by a disjunction over the domain elements $y_1 \dots y_t$.

For SO quantification, we inductively assume that the body has already been transformed into a universal FO formula. Universal SO quantification therefore can be exactly eliminated, while for existential SO quantification we substitute the appropriate approximation of the corresponding Hilbert choice operator. We define:

$$\begin{array}{lcllcl}
\mathcal{T}_t[Px] & = & Px & \mathcal{T}_t[\neg Px] & = & \neg Px \\
\mathcal{T}_t[x = y] & = & x = y & \mathcal{T}_t[x \neq y] & = & x \neq y \\
\mathcal{T}_t[\varphi_1 \vee \varphi_2] & = & \mathcal{T}_t[\varphi_1] \vee \mathcal{T}_t[\varphi_2] & & & \\
\mathcal{T}_t[\varphi_1 \wedge \varphi_2] & = & \mathcal{T}_t[\varphi_1] \wedge \mathcal{T}_t[\varphi_2] & & & \\
\mathcal{T}_t[\exists x.\varphi_1] & = & \exists^\sharp x.\mathcal{T}_t[\varphi_1] & \mathcal{T}_t[\forall x.\varphi_1] & = & \forall x.\mathcal{T}_t[\varphi_1] \\
\mathcal{T}_t[\exists B.\varphi_1] & = & \exists^\sharp B.\mathcal{T}_t[\varphi_1] & & &
\end{array}$$

where for universal FO formulas φ'

$$\exists^\sharp x.\varphi' = \bigvee_{j=1}^{t} \varphi'[y_j/x]$$

This strengthening can be further weakened by considering in the disjunction not only the y_j, but also all constants as well as all universally quantified variables x' in whose scope the current subformula occurs, just as for the universal abstraction we used in Section 6.7.2.

For $\varphi = \forall A.\varphi_1$, we first bring $\mathcal{T}_t[\varphi_1]$ into prenex normal form and thereafter, the quantifier-free part into conjunctive normal form. Since the resulting formula has no existential quantifiers, precise quantifier elimination according to fact 6.5.1 can be applied to compute $\mathcal{T}_t[\varphi]$. For a SO variable B of arity r and universal FO formula φ' in normal form (2.1), we use the corresponding γ_k:

$$\exists^\sharp B.\varphi' = \mathcal{T}_t[\varphi'[\gamma_k/B]]$$

where for φ', γ_k is defined as in Section 7.5.2 with $k = t^r$. This completes the construction.

Since φ was in negation normal form, we verify by induction on the structure of φ that $\forall y.\mathcal{T}_t[\varphi] \rightarrow \varphi$, i.e., statement (1) holds. Now let U denote a universe of cardinality $t' \leq t$, and let $\eta : \{y_1, \ldots, y_t\} \to U$ be an interpretation of the y_i such that the image of η has cardinality t'. Assume that s is a structure that satisfies φ (i.e. $s, \nu \models \varphi$ for the empty valuation ν). Due to statement (1) it suffices for a proof of statement (2) to verify that then $s, \nu \oplus \eta \models \mathcal{T}_t[\varphi]$ holds as well. Again, the proof is by induction on the structure of φ where for SO existential quantifiers, we rely on Lemma 7.23. □

While the transformation $\mathcal{B}_t$ affected both existential and universal quantifiers, the transformation $\mathcal{T}_t$ now only affects existential quantification and is precise on universal quantifiers. Thereby, the first-order part of the transformation $\mathcal{T}_t$ is a variation of the approximation of FO existential quantifiers for FO Transition Systems given in Theorem 6.26. Compared to the version for FO Transition Systems, $\mathcal{T}_t$ is an extension to SO quantification and comes with a precision guarantee for universes up to cardinality t.

Based on the abstraction of existential quantifiers, we now define the strengthened iteration $\Psi_t^{(h)}$ by:

$$\begin{array}{rcl} \Psi_t^{(0)}[v] & = & \mathcal{T}_t(I[v]) \\ \Psi_t^{(h)}[v] & = & \Psi_t^{(h-1)}[v] \\ & & \wedge \bigwedge_{e=(v,\theta,v') \in E_{\mathcal{A}}} \forall A_e.\mathcal{T}_t[\Psi_t^{(h-1)}[v']\theta] \\ & & \wedge \bigwedge_{e=(v,\theta,v') \in E_{\mathcal{B}}} \exists^{\natural} B_e.\Psi_t^{(h-1)}[v']\theta \text{ for } h > 0 \end{array} \tag{7.12}$$

For all $h \geq 0$ and all program points v, $\Psi_t^{(h)}$ is a universal FO formula. Moreover, $\Psi_t^{(h)}$ can be used to automatically infer strategies and invariants that prove safety up to a given bound for any general FO Safety Game $\mathcal{G}$.

Lemma 7.25. *For all $h \geq 0$ and all program points v,*

1. *$\Psi_t^{(h)}[v]$ can effectively be computed;*
2. *Whether or not $\Psi_t^{(h)}[v] \rightarrow \Psi_t^{(h+1)}[v]$ is decidable;*
3. *$\mathcal{T}_t[\Psi^{(h)}[v]] = \Psi_t^{(h)}[v]$;*
4. *$\forall y.\Psi_t^{(h)}[v] \rightarrow \Psi^{(h)}[v]$.*

When the strengthened iteration $\Psi_t^{(h)}$ with universal FO formulas stabilizes, we obtain a sufficient condition for safety player $\mathcal{B}$ to win the game, and moreover, obtain a positional strategy how to do so. More precisely, we have:

Theorem 7.26. *Assume that the initial condition* Init *is an FO formula in the BSR fragment. Assume that for some $h \geq 0$, $\Psi_t^{(h)} = \Psi_t^{(h+1)}$.*

1. *If $\forall y.\mathsf{Init} \rightarrow \Psi_t^{(h)}[v_0]$ then the game is safe, and a winning FO strategy for safety player $\mathcal{B}$ can effectively be computed.*
2. *Assume that $\mathsf{Init} \rightarrow \Psi^{(h)}[v_0]$ holds in all universes up to size t, but $\forall y.\mathsf{Init} \rightarrow \Psi_t^{(h)}[v_0]$ does not hold. Let U be any universe for which there is a counter example s, ν with $s, \nu \models \mathsf{Init} \wedge \exists y.\neg\Psi_t^{(h)}[v_0]$. Then the cardinality of U exceeds t.*

Here, the assumption $\Psi_t^{(h)} = \Psi_t^{(h+1)}$ for some $h \geq 0$ is met whenever the infinite conjunction $\bigwedge_{h \geq 0} \Psi_t^{(h)}[v]$ is FO-definable for all program points v, as shown in Theorem 6.9.

Based on Theorem 7.26, we provide a counter-example guided abstraction refinement loop for proving an arbitrary FO safety game $\mathcal{G}$ safe, as well as computing a winning strategy for safety player $\mathcal{B}$ (in case $\mathcal{G}$ is safe):

(0) Start with some threshold $t := t_0$.

(1) Check whether whether the game is safe for universes up to t. If not, output *unsafe* and stop.

(2) Perform the abstract fixpoint iteration (7.12). Assume that for some $h \geq 0$, $\Psi_t^{(h)} = \Psi_t^{(h+1)}$. Then check whether $\forall y. \neg \mathsf{Init} \vee \Psi_t^{(h)}[v_0]$ holds. If so, output *safe*, extract strategy according to Theorem 7.26, and stop.

Otherwise, construct a model which satisfies $\exists y. \mathsf{Init} \wedge \neg \Psi_t^{(h)}[v_0]$. Assume that the universe of that counter example has cardinality t'. Due to Theorem 7.26, $t' > t$ holds. Therefore, set $t := t'$ and proceed to step (1).

In comparison to the fixpoint iteration we used in Chapter 6, this is an improved variant which features *progress guarantees* up to a given universe size. In addition, it is able to deal with both FO Transition Systems and FO Safety Games and produces counter examples in case the invariant can be proven to not be inductive.

7.7 Restricting Strategies

Not in all cases is it possible to use all state relations to compute the strategy for safety player $\mathcal{B}$. In Section 4.3, we indicated that Noninterference for a FO Safety Game $\mathcal{G}$ can be reduced to safety, provided that the winning strategy for the selfcomposition of $\mathcal{G}$ can be translated back to $\mathcal{G}$ itself. Intuitively, *self-composition* for the verification of noninterference introduces primed versions R' for each predicate R where the difference between R' and R originates from accesses to different versions of *secret* input predicates. Accordingly, noninterference in the resulting system amounts to proving that R and R' always coincide. Winning strategies for the resulting safety game are only useful, however, if they can be realized by means of corresponding strategies of the original system $\mathcal{T}$ (before duplication of predicates). We suggested in Section 4.3 to restrict strategies to *admissible* predicates only, i.e., those predicates whose values are guaranteed not to differ from their primed counterparts. For the running conference management example, all predicates were in fact admissible, allowing us to circumvent the problem entirely. However, this does not always have to be the case.

Assume that we are given a FO Safety Game $\mathcal{G}$, an inductive invariant Ψ, and a subset $\mathcal{R}_u$ of admissible predicates for each program point u. (For the case of Noninterference R_u will contain the predicates R where both copies are guaranteed to have the same contents.)

For all edges (u, θ, v) not containing predicates from $\mathcal{R}_{\mathcal{B}}$, $\Psi[u] \rightarrow \Psi[v]\theta$ already holds (as Ψ is inductive). Our task is now to come up with a FO definable strategy $\sigma(B_e)$ for

each program points u where player $\mathcal{B}$ has a choice at an outgoing edge e, where only predicates from $\mathcal{R}_u$ are allowed for the formula $\sigma(B_e)$.

In case that a strategy exists that only uses admissible predicates, for each $\mathcal{B}$-edge (u, θ, v) with occurrence of some predicate $B \in \mathcal{R}_\mathcal{B}$

$$\exists B. \forall R \notin \mathcal{R}_u. \Psi[u] \rightarrow (\Psi[v]\theta) \tag{7.13}$$

is universally true. We will use this fact to compute such a strategy σ by first removing all occurrences of non-admissible predicates from the implication and then applying the algorithm for existential SO quantifier quantifier elimination to obtain a strategy $B\sigma$ as a FO formula speaking only about admissible predicates.

Formula (7.13) implies that $\exists B. \forall R' \notin \mathcal{R}_u. \Psi[u] \rightarrow (\Psi[v]\theta)$ is equivalent to $\forall R' \notin \mathcal{R}_u. \Psi[u] \rightarrow (\Psi[v]\theta)[B\sigma/B])$. Therefore, if $\Psi[u] \rightarrow (\Psi[v]\theta)[B\sigma/B]$ is universally true for each $\mathcal{B}$-edge (u, θ, v), σ is a strategy guaranteeing that Ψ is inductive. Otherwise, the invariant Ψ can iteratively be strengthened as in (7.1).

Given the candidate invariant Ψ, the construction of an admissible FO strategy now performs two kinds of SO quantifier elimination in a row:

- First, the non-admissible predicates are removed by means of universal SO quantifier elimination;
- Second, the single existential SO quantifier is eliminated in order to construct the strategy.

This modification can be readily plugged into the fixpoint iteration from (7.1) and thus into the CEGAR loop from Section 7.6.

7.8 Alternative synthesis approaches

For synthesizing controllers for systems with an infinite state space, several approaches have been introduced that automatically construct, from a symbolic description of a given concrete game, a finite-state abstract game [46, 4, 14, 34, 86]. The main method to obtain the abstract state space is predicate abstraction, which partitions the states according to the truth values of a set of predicates. States that satisfy the same predicates are indistinguishable in the abstract game. The abstraction is iteratively refined by introducing new predicates. Applications include the control of real-time systems [34] and the synthesis of drivers for I/O devices [86]. In comparison, our approach provides a general modeling framework of First-order Safety Games to unify different applications of synthesis for infinite-state systems.

None of these works, however, give similar decidability results or provide a CEGAR loop as in Section 7.6, which not only computes winning strategies for safety player $\mathcal{B}$, but also provides progress guarantees for a general class of FO Safety Game.

7.9 Introduced Concepts

$\mathcal{G}$	Variable for a FO Safety Game
$\mathcal{A}$	Reachability Player of a FO Safety Game trying to break the invariant
$\mathcal{B}$	Safety Player of a FO Safety Game trying to uphold the invariant
$\mathcal{R}_\mathcal{A}, \mathcal{R}_\mathcal{B}$	Input predicates under the control of $\mathcal{A}$, $\mathcal{B}$
$E_\mathcal{A}$,$E_\mathcal{B}$	Edges using input predicates from $\mathcal{R}_\mathcal{A}$, $\mathcal{R}_\mathcal{B}$
τ	Variable for plays, FO Safety Game equivalent of traces
σ	Variable for strategies, usually associated to safety player $\mathcal{B}$
σ is winning	Safety Player $\mathcal{B}$ wins all plays conforming to σ
$\mathcal{G}$ is safe	There exists a winning strategy σ for safety player $\mathcal{B}$ that always upholds the invariant I in FO Safety Game $\mathcal{G}$
$\mathcal{H}_B\varphi$	Second Order Hilbert Choice operator, giving a witness for $\exists B.\varphi$

7.10 Conclusion

We have introduced *First-order Safety Games* as an extension of FO Transition System that allow for input predicates that are chosen with the intention to uphold the safety objective rather than violate it. This allows us to tackle interesting synthesis questions for FO systems. For example, this allows us to automatically infer a safe paper assignment for the running conference management example.

Just as their propositional counterparts, FO Safety Games are positionally determined and we showed that already acyclic FO Safety Games are just as powerful as Second Order Logic. We then examined the case where all occurring predicates are monadic or nullary and provided a complete classification into decidable and undecidable cases. For the non-monadic case, we again concentrated on *universal* FO safety properties. We provided techniques for certifying safety and designed candidates for synthesizing First-order definitions of predicates as strategies that enforce the given safety objective. We showed that for the class of ackermannian FO Safety Games, strategies can always be automatically synthesized. We moved on to the general case, where we showed in which cases universal invariants can be checked for inductivity, as well as in which cases universal invariants can be automatically inferred — making the game effectively decidable. We then introduced an abstraction refinement technique that can prove a given safety game *safe*, while constructing a strategy to ensure this safety as well as provide meaningful counterexamples. During the computation, it provides guarantees up to a given universe size, while certifying overall safety whenever it terminates.

To tailor the synthesis solutions to the case of noninterference and other scenarios where only parts of the state space can be used to construct the strategy, we extended our approach so that only strategies were considered that refer to *admissible* predicates.

Chapter 8

NIWO,
FO Transition System solver

Contents

8 NIWO, FO Transition System solver

We have implemented the proposed techniques into the tool NIWO. NIWO is a fully automated, complete implementation of the symbolic model checking approach mentioned in Chapter 5 and the invariant inference approach in Chapter 6 for general FO Transition Systems. It also features initial support for FO Safety Games as provided in Chapter 7.

It is written entirely in Scala and uses Z3 as a backend for satisfiability of formulas in FO Logic. It offers command line interfaces for both pipelines, and either produces FOLTL/LTL formulas for symbolic model checking or infers inductive invariants for FO Transition Systems. NIWO is open source, licensed under the MIT license and can be found at [67].

8.1 Architecture

Formulas Formula manipulation is at the core of NIWO, since many of our abstraction techniques rely on involved algorithms transforming formulas. For FO Logic this includes conversions to conjunctive or disjunctive normal form, prenex normal form or negation normal form as well as structural substitutions of literals by formulas, checking if particular subformulas exist, instantiating quantifiers for a fixed universe, abstracting existential quantifiers and others. This is captured by the *Formula* class of NIWO which (together with the algorithms in the *FOTransformers* and the *FormulaFunctions* object) easily allow for simplification and manipulation. NIWO includes its own bottom-up rewriting simplification engine, but it can also use the Z3 backend to simplify a given formula. For checking satisfiability of a formula, NIWO passes it to the Z3 backend. For formulas in the Bernays-Schönfinkel-Ramsey fragment we use the built-in support of Z3, while for temporal formulas we either use an encoding of timestamps into the natural numbers and ask Z3 to solve the resulting formula using the theory of linear integer arithmetic or (as detailed during the symbolic model checking description) pass it to the dedicated LTL satisfiability solver Aalta.

Input formats NIWO is able to parse and handle both a textual representation of a FO Transition Systems together with an invariant as well as a workflow as presented in Section 3.1 together with a specification of Noninterference. An example of an input file specifying a workflow is shown in Figure 8.1, closely resembling the syntax for multi-agent workflows introduced in Chapter 3. Most UTF-8 symbols used in the examples like the boolean operators "$\neg$", "$\wedge$", "$\vee$" or "$\rightarrow$" can either be written with their corresponding symbols or in their more common Ascii forms "!", "&&", "||" and "->". The additional target block is optional and specifies that for this example, we are only interested in Noninterference violations based on observations of agents in the relation *Read*. To parameterize the declassification conditions with the free agent a of the Noninterference property, use the first component of the target block (if it exists).

An example of a FO Safety Game as input is shown in Figure 8.2. While for workflows,

```
1   Workflow
2
3   forallmay x:A,p:P
4       True → Conf += (x,p)
5   forallmay x:A,p:P
6       ¬Conf(x,p) → Assign += (x,p)
7   forall x:A,p:P,r:R
8       (Assign(x,p) ∧ Oracle(x,p,r)) → Review += (x,p,r)
9   loop {
10      forallmay xa:A,xb:A,p:P,r:R (Assign(xa,p) ∧ Review(xb,p,r)) → Read +=
(xa,xb,p,r)
11      forallmay x:A,p:P,r:R (Assign(x,p)) → Review += (x,p,r)
12  }
13
14  Declassify
15
16  Oracle(x:A,p:P,r:R): ¬Conf(xat:A,p:P)
17
18  Target
19
20  Read(xat:A, xbt:A, pt:P, rt:R)
```

Figure 8.1: Example input file: easychair_stubborn.wfspec

the signature is computed implicitly and relations are classified as state or input relations based on their name, the input format for FO Transition Systems makes this notion explicit and expects an explicit signature as the first block. The invariant is then given explicitly and is applied to all nodes of the given system. Additionally, the user can give the initial condition Init for the predicates *AxiomPredicates* explicitly, while the other state predicates are assumed to be empty. This is useful to describe *constant* input relations that do not change over time and have certain properties. For the leader election proof examples, for example, these contain the assumptions about the ring topology.

These are represented by the case classes *InvariantSpec* and *NISpec* respectively. An *InvariantSpec* then consists of the FO Transition System as a *TransitionSystem* object, the initial conditions as well as the invariant. A *NISpec* consists of the corresponding *Workflow*, the version of the agent model and the declassification conditions.

Both the *TransitionSystem* as well as the *Workflow* have a similar structure. They consist of a *Signature* and a list of control structures (blocks) representing the control flow graph of the system. In Figure 8.3, we show which classes and traits exist to represent the different blocks. Both the loop construct as well as the nondeterministic choice construct are aggregates of their respective block type and may be nested. Both concrete variants of a *SimpleBlock* are made up of statements. The *SimpleWFBlock* represents one of the two possible workflow blocks *ForallWFBlock* and *ForallMayWFBlock* corresponding to the same concepts from the definition of workflows. Both are made up of statements that either add tuples to a specific relation, subtract tuples from it or set the relation to a

```
1   Signature
2       EmptyPredicates: Conf(x:A, p:P), Assign(x:A,p:P), Review(x:A,p:P,r:R),
3                        Read(x:A,p:P,r:R)
4       AxiomPredicates: —
5       As: A1(x:A,p:P), A2(x:A,p:P,r:R), A3(x:A,p:P,r:R)
6       Bs: B1(x:A,p:P)
7       Constants: —
8
9   Transition System
10    // player A predicate for Conf
11    Conf(x,p) := A1(x,p)
12    // player B predicate for Assign
13    Assign(x,p) := B1(x,p)
14    Review(x,p,r) := (Assign(x,p) ∧ ¬Conf(x,p) ∧ A2(x,p,r))
15    loop {
16        Read(x,p,r) := ∃y:A. (Assign(x,p) ∧ Review(y,p,r))
17        Review(x,p,r) := (Assign(x,p) ∧ ¬Conf(x,p) ∧ A3(x,p,r))
18    }
19
20  Invariant
21    ∀x:A,p:P,r:R. ¬(Conf(x,p) ∧ Read(x,p,r))
```

Figure 8.2: Example input file: easychair_singletrace.tsspec

specific set of tuples. These match the semantics of the workflow blocks:

---

forall $\bar{x}: \bar{s}$ [**may**]. $\varphi \rightarrow R \pm= \bar{y}$

A *SimpleTSBlock* represents an edge in a FO Transition System and consists of updates setting a relation to a specific set of tuples, characterized by a formula. These are exactly the several formulas $\theta(R\bar{y})$ in an edge of the FO Transition System.

The traits *Block*,*Loop*,*NondetChoice* and *SimpleBlock* are unifying traits that allow us to write algorithms that deal with both concrete types of blocks. This allows us to use destructuring pattern matching to distinguish if a given object is f.e. a loop — which will be true for both *WFLoop* and *TSLoop*. To ensure that this works seamlessly all mentioned block types use self-recursive type parameters to enable pattern matching on both the concrete type of block as well as the different abstract types.

Symbolic Model Checking Pipeline In symbolic model checking mode, NIWO implements the approach described in Chapter 5. The pipeline is shown in Figure 8.4. To do so, it can be called using the *LTLCLI* command line interface with the parameter of the **.wfspec* file to be checked. It should contain a workflow specification. The file is then parsed by the *WorkflowParser* into a *NISpec*. The encoding in LTL is then orchestrated by the *MainLTLWorkflows* object, which encodes the *NISpec* into FO HyperLTL via the *Properties* object. It then further compiles it to the satisfiability of an LTL query as described in Chapter 5 using the *FOTransformers* object. The resulting FOLTL and LTL formulas are then written to disk, where they can be fed into any LTL satisfiability solver — for example into Aalta by the *measure.sh* script. Additionally, NIWO outputs the quantified FOLTL used to create the LTL file, some metrics about the formulas as well as a pretty-printed version of the LTL formula in the form of a **.ppltl* file.

Invariant Inference Pipeline The inference pipeline is slightly more involved. It is shown in Figure 8.5. Here, both workflow specification as well as transition system specification files can be read and will be fed into the respective parser, resulting in either a *NISpec* (in the case of a workflow) or a more general *InvariantSpec* (in case the input is already a transition system). In case the input is a *NISpec*, we apply the conversion shown in Section 6.1 to encode the workflow, together with the agent model and declassification conditions into a FO Transition System, and encapsulate it (together with its invariant) into a *InvariantSpec*.

In both cases, we now deal with an *InvariantSpec*, which is passed to the *GraphBuilder* object to be parsed into a *TSGraph*, a *scala-graph* graph with edges labeled with *SimpleTSBlock*. From here on, the *InvariantChecker* executes the inference algorithm described in Chapter 6. It checks for edges where the labeling of graph nodes to their invariant formula is not inductive using Z3. It then strengthens the respective edge using the abstractions discussed earlier and then repeats the process until either

1. the candidate invariant becomes inductive;

2. or the label formula at the initial node is no longer implied by Init.

In both cases, it produces *dot* files detailing the process as well as the resulting invariant and metrics about the input in separate files.

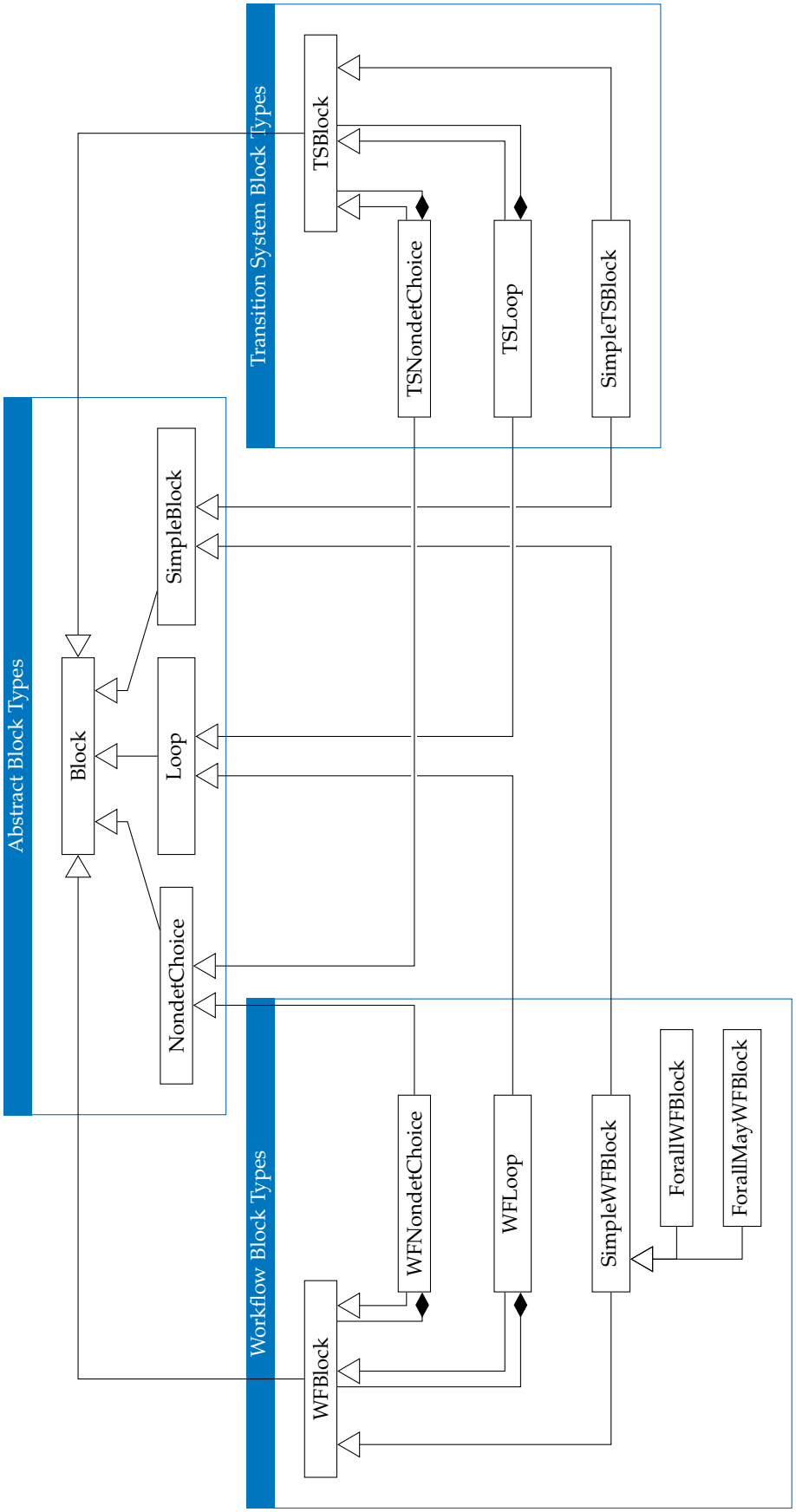


Figure 8.3: Classes and traits representing blocks

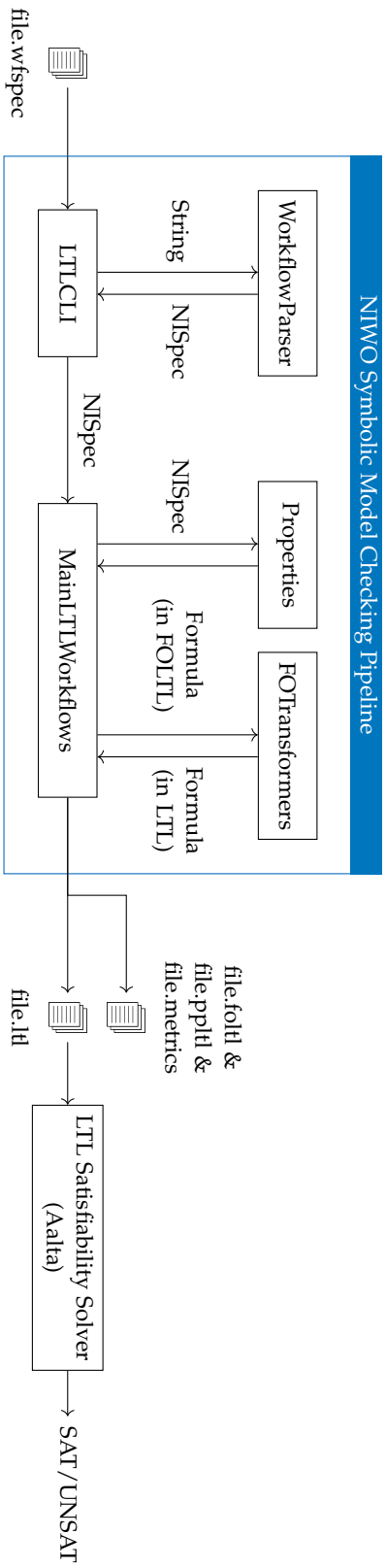


Figure 8.4: Symbolic Model Checking Pipeline

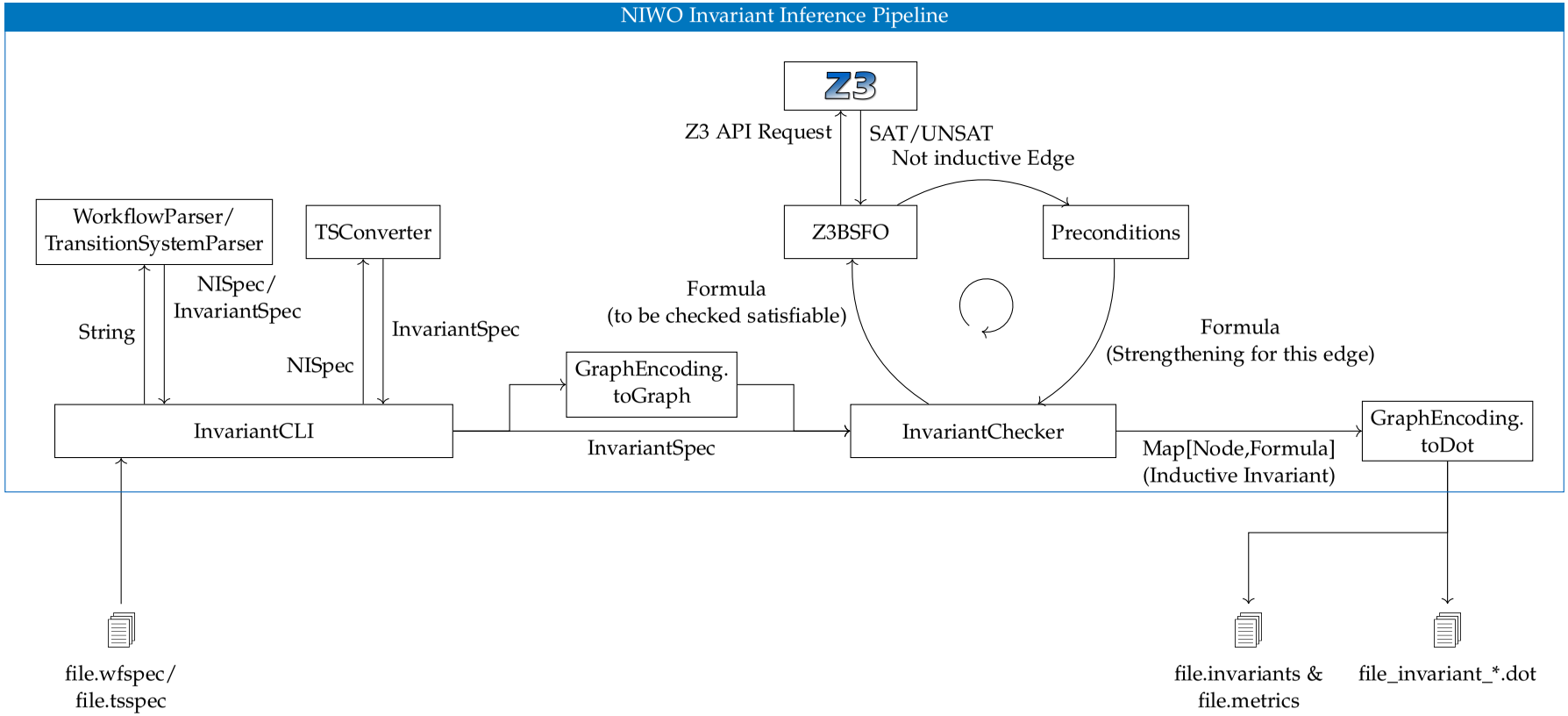


Figure 8.5: Invariant Inference Pipeline

Synthesis Pipeline Based upon the invariant inference pipeline, NIWO implements a basic implementation of FO Safety Games and is able to extract strategies for ackermannian predicates as described in Chapter 7.

Metrics NIWO is made up of 4200 *class lines of code (cloc)* in mostly functional style Scala code spread over 39 files (documented by roughly 1100 lines of comments) and bundled with a test suite of 2000 cloc in 29 files that test all functionalities NIWO offers. It comes with a suite of 80 example and test input files.

8.2 Experimental Evaluation

We evaluated NIWO on examples of multi-agent systems, among these the examples mentioned throughout this thesis, as well as synthetic examples to evaluate scalability issues.

We used several examples including the following: *Notebook* is an event-based model of a notebook-like data structure where several people can write messages, but everyone can only read his own data. It is proven safe by our implementation, even in the presence of causal agents. *Leader Election, Safety* is the leader election example shown in Chapter 7. The safety property states that at no point there are two different nodes declaring themselves leader. *Conference, Safety* uses the safety property that no author should be able to read reviews to his own paper, while all other conference examples use the Noninterference property developed in the previous chapters. *Conference* is the example conference management from Example 3.3, where our implementation finds the counterexample described in Section 5.2.3. *Conference, acceptance* is a slight variation that forgoes reviews and replaces it by an acceptance relation. Since it is very similar to the initial conference example, we use it to showcase the impact of small changes to the workflow to the verification problem. *Conference, acyclic* is an acyclic version of the conference management example which exhibits a very similar attack.

Conference, safe as well as *Conference, acyclic, safe* are versions of the conference management example that fix the paper assignment so that the counterexample is no longer possible. The resulting FO Transition System is safe and is proven safe by NIWO.

Conference, acyclic (game), *Conference (game)* and *Leader Election (game)* are FO Safety Game variants of the transition systems where the paper assignment step (in the conference case) or the initial message to be sent (in the leader election case) are replaced by a $\mathcal{B}$ predicate. They are the FO Safety Games shown in Chapter 7.

All experiments were carried out on a desktop machine using an Intel i7-3820 clocked at 3.60 GHz with 15.7 GiB of RAM and running Debian with a timeout of 20 minutes. A test suite bundled with a prebuilt version of NIWO can be found at [81].

8.2.1 Symbolic Model Checking

The results of the experiments for NIWO using the symbolic model checking pipeline are shown in Table 8.1.

In addition to the realistic examples we used synthetic examples to illustrate the scalability of the approach in several dimensions. The *Fixed-Arity-X* examples show the

Table 8.1: Symbolic Model Checking Evaluation

Name	Agent model	Size	Result	Universe size	FOLTL size	LTL size	Time
Notebook	stubborn	2	safe	(1,1)	266	240	0.18 s
Notebook	causal (1)	2	safe	(3,2)	309	993	2.92 s
Conference	stubborn	5	safe	(2,1,1)	628	1089	2.50 s
Conference	causal (1)	5	unsafe	(4,2,2)	700	8771	91.86 s
Conference, acceptance	stubborn	5	safe	(2,1)	628	1089	2.47 s
Conference, acceptance	causal (1)	5	unsafe	(4,2)	700	5187	45.63 s
Conference, acyclic	stubborn	4	safe	(2,1)	469	698	0.75 s
Conference, acyclic	causal (1)	4	unsafe	(4,2,1)	541	4116	4.91 s
University	stubborn	3	safe	(1,1)	305	202	0.01 s
University	causal (2)	3	unsafe	(4,3,3,2)	408	2727	1.28 s
Fixed-Arity-10	stubborn	10	unsafe	(2)	1928	5299	0.89 s
Fixed-Arity-15	stubborn	15	unsafe	(2)	3963	11114	3.09 s
Fixed-Arity-20	stubborn	20	unsafe	(2)	6723	19054	16.85 s
Fixed-Arity-10-safe	stubborn	10	safe	(2)	1924	5283	33.40 s
Fixed-Arity-15-safe	stubborn	15	safe	(2)	3959	11098	158.83 s
Fixed-Arity-20-safe	stubborn	20	safe	(2)	6719	19038	740.91 s
Increasing-Arity-2	stubborn	2	safe	(2)	180	335	0.08 s
Increasing-Arity-3	stubborn	3	safe	(3)	301	2206	6.20 s
Increasing-Arity-4	stubborn	4	safe	(4)	451	21894	- s
Sorted-Increasing-Arity-2	stubborn	2	safe	(1,1)	180	163	0.03 s
Sorted-Increasing-Arity-3	stubborn	3	safe	(1,1,1)	301	270	0.09 s
Sorted-Increasing-Arity-5	stubborn	5	safe	(1,1,1,1,1)	630	559	0.40 s
Sorted-Increasing-Arity-10	stubborn	10	safe	(1,...,1)	1960	1719	8.97 s
Causal-1	stubborn	4	safe	(2)	654	1129	8.23 s
Causal-1	causal (1)	4	unsafe	(4)	747	2805	6.05 s
Causal-2	stubborn	6	safe	(2)	778	1353	26.73 s
Causal-2	causal (2)	6	unsafe	(6)	965	6338	195.31 s
Sorted-Causal-2	causal (2)	3	unsafe	(4,3,1)	378	1184	1.47 s
Sorted-Causal-3	causal (3)	4	unsafe	(5,4,4,1)	598	3169	1.96 s
Sorted-Causal-5	causal (5)	5	unsafe	(7,6,6,6,6,1)	1197	14510	17.05 s

```
1   Workflow
2
3   forallmay i:A True → R += (i)
4   forallmay i:A,j:B (R(i)) → S += (i,j)
5   forallmay i:A,j:B,k:C (S(i,j)) → T += (i,j,k)
6   forallmay i:A,j:B,k:C,l:D (T(i,j,k)) → U += (i,j,k,l)
7   forallmay i:A,j:B,k:C,l:D,m:E (U(i,j,k,l)) → V += (i,j,k,l,m)
8
9   Target
10
11  V(u:A,v:B,w:C,x:D,y:E)
```

Figure 8.6: Sorted-Increasing-Arity-5.wfspec

behavior when increasing the number of relations. These cases contain X relations that are successive copies of each other starting from some secret input. The *Fixed-Arity-X-safe* examples are similar, but are devoid of counterexamples. The *Sorted-Increasing-Arity-X* and *Increasing-Arity-X* examples show the impact of using sorts. To exemplify, *Sorted-Increasing-Arity-5* is shown in Figure 8.6. They contain X relations of arities $1, \ldots, X$, respectively. For every relation of arity n, a tuple containing the first $n-1$ variables has to be present in the relation with arity $n-1$. For the *Increasing-Arity* cases, all variables refer to the same sort, whereas in the *Sorted-Increasing-Arity* variant, every relation of arity n uses n different sorts. *causal-X* and *Sorted-causal-X* cases showcase the scalability with the number of causal agents that are part of the attack. These cases are set up in a way that a successful attack needs to consist of at least X causal agents.

The size column gives the number of edges of the graph of the resulting FO Transition System. The agent model column gives the considered agent model, i.e. "stubborn" means $agent\_model^{(s)}$ while "causal (k)" means $agent\_model^{(c,k)}$ (there are at most k agents that act causally with all others behaving stubbornly). The FO Transition System is *safe* iff the LTL formula was proven *unsatisfiable* by Aalta and *unsafe* otherwise. The next column gives the sizes of the considered universes. For example, to show that *Conference* is safe with respect to one causal agent, it is enough to consider universes containing 4 reviewers, 2 papers and 2 reviews (one per paper), respectively. The universes' sizes are given as a tuple, for instance $(4, 2, 2)$. The size of both the FOLTL and LTL formulas is the number of nodes in the formulas abstract syntax tree. The last column is the time (in seconds) that it takes Aalta to check the satisfiability of the LTL formula (averaged over 10 runs).

The implementation is able to handle all examples based on real applications in less than 100 seconds. Even though the size of the resulting formula is exponential in the number of agents in the universe, Aalta was still able to check the satisfiability of formulas consisting of thousands of LTL operators in reasonable time. As expected of a satisfiability solver, giving a counterexample for a formula is almost always faster than proving it unsatisfiable for formulas of comparable complexities.

The *Fixed-Arity* cases show that workflows handle an increasing number of relations with the same arity quite well. Here, adding another relation increases the size of the

formula by a small factor, since the size of the needed universe stays the same - only the universally quantified encoding of the control flow graph grows. Increasing the necessary arity of the relations increases the minimum size of the universe - as shown by the *Increasing-Arity* cases. In case that all necessary agents are of the same sort, the formula grows exponentially, whereas it grows a lot slower in case that increasing the arity introduces a new sort. Since in those cases the size of the needed universe is exactly one agent per sort, the resulting LTL formula is even smaller than the FOLTL specification. The biggest factor in increasing the state space of the workflow, however, is the number of necessary causal agents as shown by the two variants of the *Causal-X* cases. Since every causal agent that we consider adds another copy of all of the agents needed to verify the workflow for only stubborn agents, adding the first causal agent doubles the minimum amount of agents in the universe. Since the size of the LTL formula is exponential in the number of agents, adding more causal agents causes the size of the resulting LTL formula to grow rapidly.

8.2.2 Invariant Inference

The results of the experiments using the invariant inference pipeline are shown in Table 8.2. NIWO is able to prove safe all our examples that do not exhibit attacks. Interestingly, even though termination is not guaranteed for causal agents or for general FO Transition Systems, our tool still terminated on all examples we considered.

The first columns give the size of the encoded FO Transition System (the number of edges the transition system consists of), and the considered agent model. Here, "causal" means $agent\_model^{(c)}$, specifying that *all* agents may act causally. The result is marked as *proven safe* iff our tool could iteratively find a strengthened universal inductive invariant that implies NDA. It is marked *proven inductive* if NIWO could prove the FO Transition System safe when starting from an initial invariant. It is marked as *proven counterexample* in case NIWO has proven the system unsafe for a specific universe and it is marked *proven not inductive* if NIWO terminated but had to strengthen the invariant until it was no longer implied by the initial condition. The number of times the invariant was strengthened is recorded in the column "Inference Steps". The size of the largest invariant (column "Largest Inv.") is the number of nodes in the abstract syntax tree of the largest formula assigned to any node. The last column reports the time (in milliseconds) for checking validity (averaged over 10 runs).

As expected of a modern satisfiability solver, Z3 was able to check the satisfiability of all formulas easily even though the size of the resulting boolean formula is exponential in the maximum universal quantifiers per block and the number of inference steps needed. For stubborn agents, all examples terminated after at most 2.5 seconds. For causal agents, the number of inference steps increased and invariants became significantly larger. Still, all examples terminated within at most 10 seconds.

To infer an inductive invariant and a safe strategy for causal agents, multiple iterates of the fixpoint iteration from Section 6.3 must be computed. Each iteration requires formulas to be brought into conjunctive normal form — possibly increasing formula size drastically. To cope with that increase, formula simplification turns out not to be sufficient. We try two different approaches to overcome this challenge: First, we provide the solver with parts of the inductive invariant, so fewer strengthening steps were needed. Given the initial direction, inference terminates much faster and provides

us with a useful strategy. For the second approach, we do not supply a strengthened invariant, but accelerate fixpoint iteration by further strengthening of formulas. In that way, we enforce termination while still verifying safety.

Compared to the symbolic model checking pipeline, the inference pipeline it is much more focused on invariants and in general is not guaranteed to terminate. However, not only did it terminate on all our examples, it also did so a lot faster than the symbolic model checking pipeline did, even for the more general agent model where *all* agents act causally, which could not even be handled by the symbolic model checking pipeline.

8.2.3 Synthesis

As an extension to the inference pipeline, we also tested NIWO on FO Safety Game versions of our benchmarks to see which strategies would be extracted. The results are shown in Table 8.3. For *ackermannian* predicates like the leader election example from Chapter 7 NIWO finds the weakest universal strategy upholding the invariant, while for non-ackermannian predicates, it resorts to further abstraction.

For *Conference, Safety*, the extracted strategy is exactly what we expect — not assigning authors to their own papers. The strategy for the leader election example is shown in Example 7.8 and is the weakest possible universal strategy.

For FO Safety Games with a complex inferred invariant like NDA for the full conference management example, NIWO is still able to prove the game safe and extract a strategy even without any given initial invariant. In comparison, though, the extracted strategy becomes a lot stronger than needed, due to the abstraction happening during the inference algorithm. For *Conference (game)* without any initial invariant to guide the approximative approach, NIWO is only able to extract the empty assignment as a safe strategy. For benchmarks where an initial (almost) inductive invariant is given, i.e. *Leader Election (game), initial invariant* as well as *Conference (game), initial invariant*, NIWO terminates quicker with fewer inference steps and the extracted strategy is more useful in practice. *Leader Election (game), initial invariant* is the example mentioned in Example 7.8, where the extracted safe strategy is in fact a superset of the safe strategy used in the non-game benchmark *Leader Election*.

8.3 Conclusion

In this chapter, we have presented NIWO, a fully automated solver for FO Transition Systems and FO Safety Games. It implements the symbolic model checking pipeline for FO Transition Systems developed in Chapter 5 by creating the necessary LTL formulas which are then solved by the LTL satisfiability solver Aalta. It also implements the invariant inference pipeline for FO Transition Systems shown in Chapter 6 and extended to FO Safety Games in Chapter 7, which infers universal inductive invariants to prove that a given safety property holds.

We have evaluated NIWO on all examples shown in this thesis. Our experiments indicate that we are able to prove real-world examples safe. We are also able to extract meaningful strategies that assure safety (given an initial invariant, which is not necessarily inductive).

Table 8.2: Invariant Inference Evaluation

Name	Model	Size	Result	Inference Steps	Largest Inv.	Time
Conference, Safety	-	6	proven safe	4	50	705 ms
Leader Election	-	4	proven inductive	0	42	351 ms
Conference, acyclic	stubborn	4	proven safe	3	179	338 ms
Conference, acyclic	causal	8	proven not inductive	3	1670	1537 ms
Conference, acyclic, safe	stubborn	5	proven safe	4	247	322 ms
Conference, acyclic, safe	causal	10	proven safe	4	4285	2352 ms
Conference	stubborn	6	proven safe	5	220	347 ms
Conference	causal	11	proven not inductive	7	2615	2924 ms
Conference, fixed universe	causal	11	proven counterexample	7	-	2114 ms
Conference, safe	stubborn	7	proven safe	6	709	1059 ms
Conference, safe	causal	13	proven inductive	2	102	2460 ms

Table 8.3: Synthesis Evaluation

Name	Model	Size	Result	Inference Steps	Largest Inv.	Time
Conference (game), Safety	-	6	found safe strategy	4	50	736 ms
Leader Election (game), initial invariant	-	4	found safe strategy	0	42	346 ms
Conference, acyclic (game)	causal	8	found safe strategy	4	137	1985 ms
Conference (game)	stubborn	6	found safe strategy	4	850	6817 ms
Conference (game), initial invariant	causal	11	found safe strategy	2	102	2460 ms
Conference (game), approximation	causal	11	found strong safe strategy	8	5090	3359 ms

Chapter 9

Conclusion

Contents

9 Conclusion

In this work, we provided techniques to model and analyze multi-agent systems with unbounded numbers of possible participants. These techniques can be applied to prove safety or noninterference of real-world distributed systems like web-based conference management systems, online banking tools, ad-hoc networks, network protocols, etc..

To formally reason about these systems, we introduced the formal model of *First-order Transition Systems* and showed how to translate several existing modeling languages into this model. We then showed how to formalize properties and hyper-properties of FO Transition Systems. In particular, we extended *HyperLTL* to *First-order HyperLTL*, a powerful temporal logic we use to specify temporal Hyperproperties of FO Transition Systems like Noninterference. For this, we formalized the behavior of the participating agents into the two specific agent models of *stubborn* and *causal* agents and discussed how these agent models affect the overall system.

We then developed approaches to verify a given FO Transition System. For temporal Hyperproperties, we gave both a bounded symbolic model checking approach as well as a symbolic model checking approach by coding both the transition system and the property into FO HyperLTL. Both these approaches can be applied to prove (bounded) safety for *quantifier-free* FO Transition Systems. For FO Transition Systems that are not quantifier-free, we have proven undecidability even for the specific property of Noninterference.

We then provided more specialized methods for restricted forms of Noninterference based on the encoding of declassification and agent models into the transition system itself, thus effectively reducing Noninterference to a safety property of the resulting system. For the latter, we then applied techniques based on inductive invariants. We exemplified our methods on various versions of parts of conference management systems. We also identified classes of FO Transition Systems where this approach yields a decision procedure. In particular, we discussed the decidable cases of *monadic* and *guarded* FO Transition Systems. For examples outside these classes, we provided a fixpoint iteration approach that can prove safety of a given system in case the algorithm terminates. We also showed that the iteration always terminates for *stratified guarded* FO Transition Systems.

Afterwards, we extended our analysis approaches to *synthesis* questions on *First-order Safety Games*. We have shown in which cases we are not only able to prove a given system safe, but also which concrete *strategy* the actors under our control can use to ensure safety. This strategy can always be precisely synthesized for *ackermannian* FO Safety Games. For general FO Safety Games, our methods are able to synthesize a strategy in case that we are given an inductive invariant, and a FO definable strategy exists. We also improved the fixpoint iteration to a CEGAR loop for FO Safety Games that provides guarantees up to a specific universe size and also generates counter-examples in case the system does not satisfy the property.

In addition to the theoretical foundations of FO Transition Systems and FO Safety Games, we reported on NIWO, a fully automated analysis tool for FO Transition Systems and FO Safety Games and showed that it performs well on the examples discussed throughout this thesis.

In summary, we developed the theory of FO Transition Systems and FO Safety Games to model information flow questions for multi-agent systems, proved under which assumptions the problem becomes decidable and applied practical and effective algorithms to solve bounded symbolic model checking, symbolic model checking and synthesis questions.

9.1 Future Work

In this work, we have provided a general framework to verify safety as well as Noninterference properties of multi-agent systems. In future work, we would like to extend our methods to more complex properties that can not be encoded by purely universal First-order formulas. We would like to tackle more complex temporal properties that include liveness and/or fairness constraints. Additionally, it might be possible to augment our analysis by background theories like integers and cardinality constraints. This would, for example, allow us to extend our methods to questions of *quantitative* information flow [52, 91] or planning [35].

As we detailed in Chapter 7, a counter-example guided approach can be used to solve both FO transition systems and games. The version described in this thesis provides progress guarantees on the size of the considered universe. In comparison, it would also be interesting to use abstraction and/or interpolation to generalize the counterexamples to exclude structurally similar counterexamples regardless of universe size. This would not improve the guarantees of the algorithm but improve practical runtimes.

There also remain many open questions about FO Safety Games. In general, it would be interesting to further explore FO Safety Games and their connection to both synthesis in propositional transition systems as well as Second Order Logic. This could lead to techniques for *bounded* synthesis [80].

We provided the decidable fragment of *ackermannian* FO Safety Games, but there might be more general fragments for which (approximative) Second Order Quantifier Elimination succeeds, even if the computed strategy might not be the weakest possible one. In addition, even if Quantifier Elimination is not guaranteed to succeed, we would be interested in variants of the known semi-algorithms for Quantifier Elimination like DLS* [40] that could provide us with candidate strategies.

For FO Transition Systems, we have shown that decidability can be retained for the fragment of *stratified guarded* FO Transition Systems. We would like to know whether a similar fragment can be found for FO Safety Games and whether the known decidability results can be extended.

The implementation of our automated analysis and synthesis tool NIWO is rather mature. Still, it would be interesting to experiment with different fixpoint algorithms in order to fight the bottleneck of exploding formula sizes. It would also be useful to add more input formats that can be expressed by FO Transition Systems. This includes the ability to specify guards as well as first-order variables, as well as the ability to parse additional specification languages like RML [73].

List of Figures

List of Tables

Bibliography

[1] Aharon Abadi, Alexander Rabinovich, and Mooly Sagiv. "Decidable fragments of many-sorted logic". In: *Journal of Symbolic Computation* 45.2 (2010), pp. 153–172.

[2] Wilhelm Ackermann. "Untersuchungen über das Eliminationsproblem der mathematischen Logik". In: *Mathematische Annalen* 110 (1935), pp. 390–413.

[3] Sheldon B. Akers. "Binary decision diagrams". In: *IEEE Transactions on computers* 6 (1978), pp. 509–516.

[4] Luca de Alfaro and Pritam Roy. "Solving Games Via Three-Valued Abstraction Refinement". In: *Proc. CONCUR*. Vol. 4703. Springer-Verlag, 2007, pp. 74–89.

[5] Torben Amtoft and Anindya Banerjee. "Information Flow Analysis in Logical Form". In: *Proc. SAS 2004*. Ed. by Roberto Giacobazzi. Springer, 2004, pp. 100–115.

[6] Henrik Reif Andersen. *A Polyadic Modal μ-Calculus*. Tech. rep. Danmarks Tekniske Universitet, 1994.

[7] Myrto Arapinis, Sergiu Bursuc, and Mark Ryan. "Privacy Supporting Cloud Computing: ConfiChair, a Case Study". In: *Proc. POST 2012*. Springer Verlag, 2012, pp. 89–108.

[8] James Aspnes and Eric Ruppert. "An introduction to population protocols". In: *Middleware for Network Eccentric and Mobile Applications*. Springer, 2009, pp. 97–120.

[9] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[10] Thomas Ball and Sriram K Rajamani. "The SLAM toolkit". In: *International Conference on Computer Aided Verification*. Springer. 2001, pp. 260–264.

[11] Michael Barnett, Robert DeLine, Manuel Fähndrich, K Rustan M Leino, and Wolfram Schulte. "Verification of Object-Oriented Programs with Invariants." In: *Journal of Object Technology* 3.6 (2004), pp. 27–56.

[12] Gilles Barthe, Pedro R D'Argenio, and Tamara Rezk. "Secure information flow by self-composition". In: *Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004.* IEEE. 2004, pp. 100–114.

[13] Thomas Bauereiß and Dieter Hutter. "Information flow control for workflow management systems". In: *it - Information Technology* 56.6 (2014), pp. 294–299.

[14] Thomas Bauereiß, Armando Pesenti Gritti, Andrei Popescu, and Franco Raimondi. *CoSMeDis: A Distributed Social Media Platform with Formally Verified Confidentiality Guarantees*. to appear in Security and Privacy 2017. 2017.

[15] Heinrich Behmann. "Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem". In: *Mathematische Annalen* 86.3-4 (1922), pp. 163–229.

[16] Robert Berger. *The undecidability of the domino problem*. 66. American Mathematical Soc., 1966.

[17] C. Bhardwaj and S. Prasad. "Parametric information flow control in ehealth". In: *Proceedings HealthCom 2015*. Oct. 2015, pp. 102–107. DOI: 10.1109/HealthCom.2015.7454481.

[18] Armin Biere, Alessandro Cimatti, Edmund M Clarke, Ofer Strichman, Yunshan Zhu, et al. "Bounded model checking." In: *Advances in computers* 58.11 (2003), pp. 117–148.

[19] Michael Blondin, Javier Esparza, Stefan Jaax, and Antonín Kucera. "Black Ninjas in the Dark: Formal Analysis of Population Protocols." In: *LICS*. Ed. by Anuj Dawar and Erich Grädel. ACM, 2018, pp. 1–10. DOI: 10.1145/3209108.

[20] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

[21] Egon Börger and Robert Stärk. "History and Survey of ASM Research". In: *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer, 2003, pp. 343–367. ISBN: 978-3-642-18216-7.

[22] Egon Börger and Robert Stärk. "Tool Support for ASMs". In: *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer, 2003, pp. 313–342. ISBN: 978-3-642-18216-7.

[23] Ronald J Brachman, Hector J Levesque, and Raymond Reiter. *Knowledge representation*. MIT press, 1992.

[24] Aaron R Bradley. "SAT-based model checking without unrolling". In: *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer. 2011, pp. 70–87.

[25] Adam Chlipala. *Certified programming with dependent types: a pragmatic introduction to the Coq proof assistant*. MIT Press, 2013.

[26] Alessandro Cimatti. "Industrial applications of model checking". In: *Summer School on Modeling and Verification of Parallel Processes*. Springer. 2000, pp. 153–168.

[27] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. "Counterexample-guided abstraction refinement". In: *International Conference on Computer Aided Verification*. Springer. 2000, pp. 154–169.

[28] Edmund M Clarke, Thomas A Henzinger, Helmut Veith, and Roderick Bloem. *Handbook of model checking*. Vol. 10. Springer, 2018.

[29] Michael Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. "Temporal Logics for Hyperproperties". In: *Proc. of POST*. 2014.

[30] Michael R. Clarkson and Fred B. Schneider. "Hyperproperties". In: *Journal of Computer Security* 18.6 (2010), pp. 1157–1210.

[31] Patrick Cousot, Radhia Cousot, and Laurent Mauborgne. "Logical Abstract Domains and Interpretations". In: *The Future of Software Engineering*. Springer, 2011, pp. 48–71. DOI: 10.1007/978-3-642-15187-3_3.

[32] Remco M Dijkman, Marlon Dumas, and Chun Ouyang. "Semantics and analysis of business process models in BPMN". In: *Information and Software technology* 50.12 (2008), pp. 1281–1294.

[33] R. Dimitrova, B. Finkbeiner, M. Kovács, M. N. Rabe, and H. Seidl. "Model Checking Information Flow in Reactive Systems". In: *Proc. VMCAI'12*. 2012, pp. 169–185.

[34] Rayna Dimitrova and Bernd Finkbeiner. "Counterexample-Guided Synthesis of Observation Predicates". In: *Formal Modeling and Analysis of Timed Systems*. Ed. by Marcin Jurdziński and Dejan Ničković. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 107–122. ISBN: 978-3-642-33365-1.

[35] Kutluhan Erol, Dana S Nau, and Venkatramana S Subrahmanian. "Complexity, decidability and undecidability results for domain-independent planning". In: *Artificial intelligence* 76.1-2 (1995), pp. 75–88.

[36] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[37] Bernd Finkbeiner, Christian Müller, Helmut Seidl, and Eugen Zalinescu. "Verifying Security Policies in Multi-agent Workflows with Loops". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM, 2017, pp. 633–645. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134080.

[38] Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. "Algorithms for Model Checking HyperLTL and HyperCTL*". In: *Computer Aided Verification*. Springer. 2015, pp. 30–48.

[39] Bernd Finkbeiner, Helmut Seidl, and Christian Müller. "Specifying and Verifying Secrecy in Workflows with Arbitrarily Many Agents". In: *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*. Ed. by Cyrille Artho, Axel Legay, and Doron Peled. Vol. 9938. Lecture Notes in Computer Science. 2016, pp. 157–173. ISBN: 978-3-319-46519-7. DOI: 10.1007/978-3-319-46520-3_11.

[40] Dov M Gabbay, R Schmidt, and Andrzej Szalas. *Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, 2008.

[41] Pranav Garg, Christof Löding, P Madhusudan, and Daniel Neider. "ICE: A robust framework for learning invariants". In: *International Conference on Computer Aided Verification*. Springer. 2014, pp. 69–87.

[42] J. A. Goguen and J. Meseguer. "Security Policies and Security Models". In: *IEEE Symp. on Security and Privacy*. 1982, pp. 11–20.

[43] Erich Gradel and Wolfgang Thomas. *Automata, logics, and infinite games: a guide to current research*. Vol. 2500. Springer Science & Business Media, 2002.

[44] Yuri Gurevich. "Evolving algebras 1993: Lipari guide". In: *arXiv preprint arXiv:1808.06255* (2018).

[45] Joseph Y. Halpern and Kevin R. O'Neill. "Secrecy in Multiagent Systems". In: *ACM Trans. Inf. Syst. Secur.* 12.1 (Oct. 2008), 5:1–5:47. ISSN: 1094-9224.

[46] T.A. Henzinger, R. Jhala, and R. Majumdar. "Counterexample-guided control". In: *Proc. ICALP'03*. Vol. 2719. LNCS. Springer, 2003, pp. 886–902.

[47] Ian Hodkinson and Mark Reynolds. "11 Temporal logic". In: *Studies in logic and practical reasoning*. Vol. 3. Elsevier, 2007, pp. 655–720.

[48] Ian M. Hodkinson, Frank Wolter, and Michael Zakharyaschev. "Decidable fragment of first order temporal logics". In: *Ann. Pure Appl. Logic* 106.1-3 (2000), pp. 85–134.

[49] Markus Holzer, Martin Kutrib, and Andreas Malcher. "Complexity of multi-head finite automata: Origins and directions". In: *Theoretical Computer Science* 412.1-2 (2011), pp. 83–96.

[50] Gerard J Holzmann. *The SPIN model checker: Primer and reference manual*. Vol. 1003. Addison-Wesley Reading, 2004.

[51] Sebastian Hunt and David Sands. "On flow-sensitive security types". In: *Proc. POPL 2006*. Ed. by J. Gregory Morrisett and Simon L. Peyton Jones. 2006, pp. 79–90.

[52] J. W. Gray III. "Toward a mathematical foundation for information flow security". In: *Proceedings IEEE Symp. on Security and Privacy*. May 1991, pp. 210–34.

[53] Neil Immerman, Alex Rabinovich, Tom Reps, Mooly Sagiv, and Greta Yorsh. "The boundary between decidability and undecidability for transitive-closure logics". In: *International Workshop on Computer Science Logic*. Springer. 2004, pp. 160–174.

[54] Emmanuel Jeandel. "The periodic domino problem revisited". In: *Theoretical Computer Science* 411.44-46 (2010), pp. 4010–4016.

[55] Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, and Rahul Mangharam. "Modeling and verification of a dual chamber implantable pacemaker". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2012, pp. 188–203.

[56] Sudeep Kanav, Peter Lammich, and Andrei Popescu. "A Conference Management System with Verified Document Confidentiality". In: *Computer Aided Verification*. Ed. by Armin Biere and Roderick Bloem. Cham: Springer International Publishing, 2014, pp. 167–183. ISBN: 978-3-319-08867-9.

[57] Sudeep Kanav, Peter Lammich, and Andrei Popescu. "A Conference Management System with Verified Document Confidentiality". In: *CAV 2014*. Springer Verlag, 2014, pp. 167–183. ISBN: 978-3-319-08867-9.

[58] Aleksandr Karbyshev, Nikolaj Bjørner, Shachar Itzhaky, Noam Rinetzky, and Sharon Shoham. "Property-directed inference of universal invariants or proving their absence". In: *Journal of the ACM (JACM)* 64.1 (2017), p. 7.

[59] Denis Kuperberg, Julien Brunel, and David Chemouil. "On Finite Domains in First-Order Linear Temporal Logic". In: *Int. Symposium on Automated Technology for Verification and Analysis*. Springer. 2016, pp. 211–226.

[60] Marta Kwiatkowska, Gethin Norman, and David Parker. "PRISM: Probabilistic symbolic model checker". In: *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer. 2002, pp. 200–204.

[61] Daniel Leivant. "Higher order logic". In: *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume2, Deduction Methodologies*. Ed. by Dov M. Gabbay, Christopher J. Hogger, J. A. Robinson, and Jörg H. Siekmann. Oxford University Press, 1994, pp. 229–322.

[62] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. "GOLOG: A logic programming language for dynamic domains". In: *The Journal of Logic Programming* 31.1 (1997), pp. 59–83. ISSN: 0743-1066. DOI: https://doi.org/10.1016/S0743-1066(96)00121-5.

[63] Zohar Manna and Amir Pnueli. "Verification of Concurrent Programs: The Temporal Framework". In: *The Correctness Problem in Computer Science*. Ed. by Robert S. Boyer and J Strother Moore. Academic Press, London, 1981, pp. 215–273.

[64] Heiko Mantel. "Possibilistic Definitions of Security – An Assembly Kit". In: *Proc. of the 13th IEEE Computer Security Foundations Workshop (CSFW)*. IEEE Computer Society, July 2000, pp. 185–199.

[65] Heiko Mantel and Christian Probst. "On the Meaning and Purpose of Attack Trees". In: *Proceedings of the 32nd IEEE Computer Security Foundations Symposium (CSF)*. 2019, pp. 184–199.

[66] René Mazala. "Infinite Games". In: *Automata, Logics, and Infinite Games*. Ed. by Erich Grädel, Wolfgang Thomas, and Thomas Wilke. LNCS 2500, Springer, Heidelberg, pp. 23–38.

[67] Christian Müller. *NIWO - First Order Transition System Solver*. https://versioncontrolseidl.in.tum.de/mueller/loopingworkflows. 2017.

[68] Christian Müller and Helmut Seidl. "Stratified Guarded First Order Transition Systems". In: *Static Analysis - 27th International Symposium, SAS 2020, November 18-20, 2020, Proceedings*. to appear. 2020.

[69] Christian Müller, Helmut Seidl, and Eugen Zalinescu. "Inductive Invariants for Noninterference in Multi-agent Workflows". In: *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, 2018, pp. 247–261. ISBN: 978-1-5386-6680-7. DOI: 10.1109/CSF.2018.00025.

[70] Timothy Nelson, Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. "Toward a More Complete Alloy". In: *Proc. of the 3rd Int. Conf. on Abstract State Machines, Alloy, B, VDM, and Z (ABZ 2012)*. Vol. 7316. Lecture Notes in Computer Science. Springer, 2012, pp. 136–149.

[71] Hans Jürgen Ohlbach. "SCAN - Elimination of Predicate Quantifiers". In: *Proc. of the 13th Int. Conf. on Automated Deduction - CADE-13*. 1996, pp. 161–165. DOI: 10.1007/3-540-61511-3_77.

[72] Oded Padon, Neil Immerman, Aleksandr Karbyshev, Ori Lahav, Mooly Sagiv, and Sharon Shoham. "Decentralizing SDN policies". In: *ACM SIGPLAN Notices*. Vol. 50. 1. ACM. 2015, pp. 663–676.

[73] Oded Padon, Neil Immerman, Sharon Shoham, Aleksandr Karbyshev, and Mooly Sagiv. "Decidability of inferring inductive invariants". In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM. 2016, pp. 217–231.

[74] Oded Padon, Giuliano Losa, Mooly Sagiv, and Sharon Shoham. "Paxos made EPR: decidable reasoning about distributed protocols". In: *PACMPL* 1.OOPSLA (2017), 108:1–108:31. DOI: 10.1145/3140568.

[75] Oded Padon, Kenneth L. McMillan, Aurojit Panda, Mooly Sagiv, and Sharon Shoham. "Ivy: safety verification by interactive generalization". In: *Proc. of the 37th ACM SIGPLAN Conf. on Programming Language Design and Implementation, PLDI 2016*. 2016, pp. 614–630. DOI: 10.1145/2908080.2908118.

[76] Lawrence C Paulson. *Isabelle: A generic theorem prover*. Vol. 828. Springer Science & Business Media, 1994.

[77] Frank P Ramsey. "On a problem of formal logic". In: *Classic Papers in Combinatorics*. Springer, 2009, pp. 1–24.

[78] Arnold L Rosenberg. "On multi-head finite automata". In: *IBM Journal of Research and Development* 10.5 (1966), pp. 388–394.

[79] A. Sabelfeld and D. Sands. "Dimensions and Principles of Declassification". In: *Proceedings CSFW'05*. IEEE Computer Society, 2005, pp. 255–269. ISBN: 0-7695-2340-4.

[80] Sven Schewe and Bernd Finkbeiner. "Bounded synthesis". In: *International Symposium on Automated Technology for Verification and Analysis*. Springer. 2007, pp. 474–488.

[81] Helmut Seidl, Christian Müller, and Bernd Finkbeiner. *How to Win First Order Safety Games - Software Artifact*. Oct. 2019. DOI: 10.5281/zenodo.3514277.

[82] Helmut Seidl, Christian Müller, and Bernd Finkbeiner. "How to Win First-Order Safety Games". In: *Verification, Model Checking, and Abstract Interpretation - 21st International Conference, VMCAI 2020, New Orleans, LA, USA, January 16-21, 2020, Proceedings*. Ed. by Dirk Beyer and Damien Zufferey. Vol. 11990. Lecture Notes in Computer Science. Springer, 2020, pp. 426–448. ISBN: 978-3-030-39321-2. DOI: 10.1007/978-3-030-39322-9_20.

[83] A. P. Sistla and E. M. Clarke. "The Complexity of Propositional Linear Temporal Logics". In: *J. ACM* 32.3 (July 1985), pp. 733–749.

[84] Marc Spielmann. "Abstract state machines: verification problems and complexity". PhD thesis. RWTH Aachen University, Germany, 2000.

[85] David Sutherland. "A model of information". In: *Proc. 9th National Computer Security Conference*. DTIC Document. 1986, pp. 175–183.

[86] A. Walker and L. Ryzhyk. "Predicate abstraction for reactive synthesis". In: *2014 Formal Methods in Computer-Aided Design (FMCAD)*. Oct. 2014, pp. 219–226. DOI: 10.1109/FMCAD.2014.6987617.

[87] Hao Wang. "Dominoes and the AEA case of the decision problem". In: *Computation, Logic, Philosophy*. Springer, 1990, pp. 218–245.

[88] Christoph Wernhard. "Approximating Resultants of Existential Second-Order Quantifier Elimination upon Universal Relational First-Order Formulas". In: *Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics (SOQE 2017), Dresden, Germany, December 6-8, 2017.* Ed. by Patrick Koopmann, Sebastian Rudolph, Renate A. Schmidt, and Christoph Wernhard. Vol. 2013. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 82–98.

[89] Christoph Wernhard. "Heinrich Behmann's Contributions to Second-Order Quantifier Elimination from the View of Computational Logic". In: *arXiv preprint arXiv:1712.06868* (2017).

[90] Christoph Wernhard. "Second-order quantifier elimination on relational monadic formulas–A basic method and some less expected applications". In: *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. Springer. 2015, pp. 253–269.

[91] H. Yasuoka and T. Terauchi. "On bounding problems of quantitative information flow". In: *Proceedings ESORICS'10*. Springer, 2010, pp. 357–372. ISBN: 3-642-15496-4.

[92] S. Zdancewic and A. C. Myers. "Observational Determinism for Concurrent Program Security". In: *Proceedings CSFW'03*. 2003.