Volume 11, Number 2
April 2024

Published by the Association for Computing Machinery
Special Interest Group on Logic and Computation

SIGLOG news

TABLE OF CONTENTS

General Information

1 From the Editor-in-Chief *Daniele Gorla*

2 From the SIGLOG Chair *Catuscia Palamidessi*

Technical Columns

3 Semantics *Prakash Panangaden*

34 Complexity *Andrei Bulatov*

61 Security and Privacy *Steve Kremer*

Regular Features

84 SIGLOG Monthly 248 *Elli Anastasiadi*

Association for Computing Machinery

Advancing Computing as a Science & Profession

SIGLOG NEWS

Published by the ACM Special Interest Group on Logic and Computation

SIGLOG Executive Committee

Chair	Catuscia Palamidessi	Inria Saclay
Vice-Chair	Andrzej Murawski	University of Oxford
Secretary	Sandra Alves	Universidad do Porto
Treasurer	Elaine Pimentel	University College London
	Franck van Breugel	York University
	Liron Cohen	Ben-Gurion University
	Carsten Fuhs	Birkbeck, University of London
	Prakash Panangaden	McGill University
	Simona Ronchi Della Rocca	Univ. of Torino
CADE President	Christoph Weidenbach	Max Planck Inst. Saarbrücken
EATCS President	Artur Czumaj	University of Warwick
EACSL President	Maribel Fernandez	King's College London
ACM ToCL E-in-C	Anuj Dawar	University of Cambridge
ASL Rep	Phokion Kolaitis	UC Santa Cruz

EDITOR IN CHIEF

Daniele Gorla Sapienza University of Rome

TECHNICAL COLUMN EDITORS

Automata	Mahesh Viswanathan	U. of Illinois Urbana-Champaign
Complexity	Andrei Bulatov	Simon Fraser University
Security and Privacy	Steve Kremer	LORIA, Inria Nancy, U. Lorraine
Semantics	Prakash Panangaden	McGill University
Verification	Christel Baier	Technische Universität Dresden

REGULAR FEATURES

Conference Reports	Joost-Pieter Katoen	RWTH Aachen Univ.
SIGLOG Monthly	Elli Anastasiadi	University of Uppsala

Notice to Contributing Authors to SIG Newsletters

As a contributing author, you retain copyright to your article. ACM will refer all requests for republication directly to you. By submitting your article for distribution in any newsletter of the ACM Special Interest Groups, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

— to publish your work online or in print on condition of acceptance by the editor
— to include the article in the ACM Digital Library and in any Digital Library-related services
— to allow users to make a personal copy of the article for noncommercial, educational, or research purposes
— to upload your video and other supplemental material to the ACM Digital Library, the ACM YouTube channel, and the SIG newsletter site

Furthermore, you affirm that:

— if third-party materials were used in your published work, supplemental material, or video, that you have the necessary permissions to use those third-party materials in your work

---

SIGLOG News (ISSN 2372-3491) is an electronic quarterly publication by the Association for Computing Machinery.

From the Editor-in-Chief



Welcome everybody to the second issue of 2024! Spring is usually the time of the year where most of us work to submit (or receive feedback on) papers to the forthcoming summer conferences, where hopefully many SIGLOG members will meet and discuss in person. In the meanwhile, I hope that you will enjoy reading this issue.

In this issue

– The SIGLOG Chair reports a few important things about the status of SIGLOG that I strongly encourage all of you to seriously take into consideration.

– In his first semantics column, Prakash Panangaden honors the memory of Philip Scott, recently passed away after a long illness. Phil was an outstanding researcher, a fantastic mentor, a lovely person, and a great friend of many of us. The article is written by Jean-Simon Pacaud Lemay that presents a brief survey on the categorical semantics for geometry of interaction, a research topic that was substantially advanced by the work of Phil together with Samson Abramsky and Esfandiar Haghverdi.

– In the complexity column, Andrei Bulatov presents a survey on one of the fundamental questions in descriptive complexity: the question of whether there is a logic for P. In this paper, Anuj Dawar and Benedikt Pago describe recent results on how the boundaries have been pushed in several directions and on new and sophisticated methods for proving lower bounds.

– In the security and privacy column, Steve Kremer presents a gentle introduction to Squirrel, an interactive prover for the verification of cryptographic protocols. The paper is jointly written by David Baelde, Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos, and Joseph Lallemand to briefly describe some of the interesting technical challenges on the logic behind Squirrel and on its construction.

– To conclude, we wrap up this number with the latest issue of SIGLOG Monthly, prepared by Elli Anastasiadi.

Enjoy reading!

Daniele Gorla
Sapienza University of Rome
SIGLOG News Editor-in-Chief
editor@siglog.org

Chair's Letter



Dear SIGLOG Members,

This year, SIGLOG's flagship conference, the ACM-IEEE Symposium on Logic in Computer Science (LICS), will be held in Tallin, Estonia, during 8–11 July 2024. It will feature invited talks by Martin Escardo and Alexandra Silva, and invited tutorials by Sam Buss and Alex Simpson. Many thanks to Ugo Dal Lago and Javier Exparza (PC chairs), Pawel Sobocinski (Local Organizer), Sam Staton (Publicity and Proceedings chair), Valentin Blot (Proceedings chair) and all the other people who are contributing to make it a top conference. I am sure it will be an exciting and unmissable event, and I hope to see many of you there.

There are some important news about our relation with the ACM that I need to share with you. Due to the increase of the costs, ACM has decided to raise the fee for the management of the SIGs, which had been constant for several years. For small SIGs like ours, the fee was 10K USD per year (larger SIGs had a higher fee). Starting from 2024, the fee for SIGLOG will increase to 25K USD. In recent years the expenses for SIGLOG-related activities were matching the fee, so this increase in the management fee means that we will go in default, unless we succeed to raise more funds. To this purpose, we will make a campaign to increase the number of SIGLOG memberships, since we have relatively few inscriptions (about 300) compared to the members of the Logic and Computation community. Please stay tuned, we will send an email to the LICS mailing list to explain the situation in more detail, and ask the support of all the (hopefully many) people who want SIGLOG to live.

Catuscia Palamidessi
INRIA Saclay
ACM SIGLOG Chair

SEMANTICS COLUMN

PRAKASH PANANGADEN,
School of Computer Science, McGill University



It is with great sadness that the SIGLOG community learned of the passing away of Philip J. Scott in December 2023. He was a leader of the community and made major contributions to type theory, polymorphic λ-calculus, bounded linear logic, Geometry of Interaction and many other topics. He and Jim Lambek wrote the classic textbook, "Introduction to Higher-order Categorical Logic" which exposed the computer science community to what were then exotic topics for us: type theory and categories, categorical proof theory and toposes.

For me he was a personal friend as well as an occasional collaborator. In the 1990's and early 2000's I often visited Ottawa and stayed with him and his wife Marcia. He taught me things, like hyperdoctrines, that I could not find easily in an accessible form. He had been in Montreal as a post-doctoral fellow of the famous McGill category theory group in the 1980's and retained his friendship with many members of that group, particularly Robert Seely. In his last few months he was struggling with health problems, as was I at the time, so we shared medical experiences. I was amazed at how upbeat he was in the face of serious setbacks. As late as November he wrote to me saying, "perhaps we can meet in Edinburgh next year". He and I shared a special fondness for Edinburgh and spent some time there together in 2011. Alas, the Edinburgh reunion that we were hoping for will never happen now.

I have planned two articles on Phil's contributions to the computer science community. Of course, he made many contributions to the logic and category theory community in the time before contacts between these communities became as tight as it is now. It is impossible to cover all his contributions in one article or indeed even a series of articles. I have chosen to have more in-depth coverage of a couple of selected topics rather than attempt a comprehensive coverage.

The following article by Jean-Simon Pascual-Lemay focusses on Geometry of Interaction. After a personal dedication, the article gives a beautiful account of the Geometry of Interaction from a categorical perspective, unlike the operator algebraic formulation originally presented by Girard. It is written with expository skill which should allow anyone in the semantics community to read it easily. A second article is planned for later in the year.

A Brief Survey on the Categorical Semantics for Geometry of Interaction
In Memory of Phil Scott

Jean-Simon Pacaud Lemay, Macquarie University, Australia



Dedication to Phil Scott (1947 – 2023)

Philip Scott, better known to his friends and colleagues as Phil, was a Professor of Mathematics at the University of Ottawa (Canada). Sadly, Phil passed away[1] after a long battle with cancer – which he bravely fought till the very end – on 18 December 2023 at the age of 76. Phil was born on 27 December 1947 in Leeds (UK). When Phil was still a baby, his family moved to North Carolina (USA), where he spent his childhood. He went on to study mathematics at the University of North Carolina at Chapel Hill (USA), before moving to Canada in the 1970s. Phil did his Ph.D. at the University of Waterloo (Canada), supervised by Denis Higgs, and obtained his doctorate in pure mathematics in 1976. Phil joined the Mathematics Department at the University of Ottawa in 1982, eventually becoming Professor Emeritus, and remained there until his passing.



Phil was an amazing mathematician and had an illustrious career. Phil's most celebrated work is most likely his book with Joachim (Jim) Lambek: "Introduction To Higher-Order Categorical Logic" (1986), which is still to this day highly regarded and is one of the most essential books in category theory. Upon the news of Phil's passing, one of the most recurring comments from mathematicians and computer scientists

[1] Obituary on the Canadian Mathematics Society Website:
https://notes.math.ca/en/article/in-memoriam-phil-scott-1947-2023/

from various research fields was how Phil and Jim's book was their introduction to category theory and categorical logic, shaping careers and research interests. Even as he was fighting cancer, Phil continued to be an active member in his mathematical communities.

On top of his research work, Phil was also an outstanding and supportive mentor to young researchers at all levels, as well as a wonderful teacher. He would always find the time for students, both those who were struggling in classes and those who were interested in pursuing a career in mathematics. Phil was also always happy discussing with young researchers about their research and giving many valuable suggestions. Even till the very end, Phil was supporting the young researchers under his supervision at Ottawa, demonstrating his dedication to being an excellent supervisor.

On a personal level, as well as being my friend, Phil was an important figure in the early stages of my academic studies and had quite an impact on the direction of my research interests. I first met Phil in my first year of undergrad, where he was my teacher for an introduction to group theory course[2]. At the time, I was way over my head and completely lost in my pure math courses: I didn't know how to do a proof, let alone what a proof actually was. It wasn't until Phil helped me that I was finally able to understand what was going on. I still have a clear memory of Phil taking the time after class, going through step by step with me on how to do a basic proof. He explained things so clearly. He genuinely seemed to care that I understood and succeeded. I truly believe that this was a pivotal moment in my academic career: without Phil's help, I would have probably kept on struggling and not learned to love mathematics (let alone go on to make it my career).

Throughout the rest of my undergrad, I had Phil again as a teacher for several other courses, such as advanced linear algebra courses and even history of mathematics courses, which he had lots of fun teaching. Sometime near the end of my undergrad, I was trying to figure out what field of research I should go work in. Seeking advice from Phil, I still remember our discussion where I was first introduced to this foreign concept of "category theory". As a motivating example, Phil used the concept that a vector space was not necessarily isomorphic to its double dual. In hindsight, I now understand that Phil was slowly introducing me to star-autonomous categories and Linear Logic. I credit Phil with opening the door to the path that led me to become a category theorist.

After my time at the University of Ottawa, Phil continued to be a mentor: he was always very supportive and happy to discuss my latest research interests or new results. I would meet up with Phil many times at the Foundational Methods in Computer Science (FMCS) workshop – the unofficial yearly Canadian category theory meetup. Anytime I passed through Ottawa, Phil would always bring us to his favourite restaurant, the Green Door – which was a walk or bike ride down the canal from the University of Ottawa. I also got the chance to spend lots of time with Phil when we both happened to be visiting the University of Edinburgh (UK) at the same time.

Phil's passing is a great loss for many communities, including the Canadian mathematics community and the category theory community. Phil was incredibly kind, a fantastic mentor, and a great friend to many of us. Condolences to his loved ones. May he rest in peace.

Acknowledgements

I would first like to thank Prakash Panangaden for inviting me to write this survey paper in memory of Phil. I'd also like to thank Samson Abramsky, Esfandiar Haghverdi,

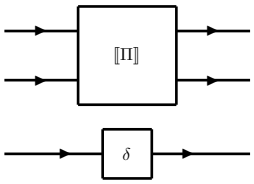
[2]At this point, one should highlight that the University of Ottawa is a bilingual university, teaching courses in both English and French. As such, Phil taught me and many others in French.

and Masahito Hasegawa for looking over the paper and making sure that my history was correct. I'd also like to give a big thank you to Prakash Panangaden, Esfandiar Haghverdi, Rick Blute, Robin Cockett, Chris Heunen, Masahito Hasegawa, and Gordon Plotkin for their encouragement and their insights about Phil and his work.

1. INTRODUCTION

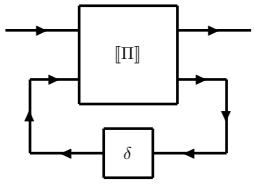
Phil Scott had an illustrious career and made many important contributions[3] in category theory, (linear) logic, and theoretical computer science. Scott is probably best known for his collaborations with J. Lambek on categorical logic and categorical proof theory, and, in particular, their all-important landmark book "Introduction To Higher-Order Categorical Logic" [Lambek and Scott(1988)]. However, another area that Scott was particularly interested in was the theory of *traced monoidal categories* [Joyal et al.(1996)], an area that he and his coauthors significantly contributed to. One aspect of traced monoidal categories in particular that Scott was interested in was using traced monoidal categories to provide a categorical framework for the *Geometry of Interaction*. Scott worked on this mostly with S. Abramsky and E. Haghverdi, the latter of whom worked on this topic for his thesis [Haghverdi(2000)] as a PhD student of Scott at the University of Ottawa. In dedication to Phil Scott, we provide an introductory level brief survey of this story, focusing mainly on the contents of [Abramsky et al.(2002); Haghverdi and Scott(2006)].

Geometry of Interaction was introduced by J.-Y. Girard in [Girard(1989b)], which he followed up on in [Girard(1988); Girard(1989a); Girard(1995)], and is a particular interpretation of Linear Logic [Girard(1987)]. The heart of Geometry of Interaction is to model mathematically the *dynamics* of cut-elimination. Usually, in a categorical model of a logic, formulas are interpreted by objects and proofs by morphisms, where these interpretations are denoted using the brackets $[\![-]\!]$. The soundness of the categorical model means that if a proof Π reduces to another Π' by cut-elimination, then their interpretations must be equal as morphisms in the categorical model, $[\![\Pi]\!] = [\![\Pi']\!]$. Thus, denotational semantics is, in this sense, static, and this kind of interpretation leads to a somewhat bland notion of invariant for cut-elimination. So with the Geometry of Interaction, Girard set out to give a way of modelling the dynamics of cut-elimination with a more meaningful invariant, called the *Execution formula*. Naively, the dynamical interpretation of proofs can be visually represented as follows. For a proof Π of a sequence $\vdash [\Delta], \Gamma$, where Γ is a sequence of formulas and Δ is a sequence of cut formulas that have been applied to the proof of $\vdash \Gamma$, its interpretation $[\![\Pi]\!]$ is graphically represented by an input-output box:



[3]A list of some of Phil Scott's important papers can be found here on his website: https://www.site.uottawa.ca/~phil/papers/

where δ models the cuts in the proof of $\vdash \Gamma$ (i.e. δ models Δ). Then cut-elimination is given by the Execution formula $\mathsf{EX}(\llbracket \Pi \rrbracket, \delta)$, which involves feedback on δ:



Girard's first model for the Geometry of Interaction was based on operator algebras and Hilbert spaces, and thus was expressed using matrices. Later on, Girard also discussed modelling Geometry of Interaction using von Neumann algebras [Girard(2006); Girard(2011)]. For a more in-depth introduction to the Geometry of Interaction, see [Haghverdi(2000), Chapter 5].

Per [Haghverdi and Scott(2010b)], categorial foundations of the Geometry of Interaction were first considered in lectures by Abramsky and by M. Hyland in the early 1990s. The first formal categorical framework for Geometry of Interaction was given by Abramsky and R. Jagadeesan in their LICS1992 paper [Abramsky and Jagadeesan(1992b)], with the journal version appearing soon after [Abramsky and Jagadeesan(1994b)]. Here, Abramsky and Jagadeesan give a "structurally isomorphic" account of the Geometry of Interaction interpretation based on a different underlying model to Girard's operator algebra model. Abramsky and Jagadeesan's model's setting is instead domain theoretic, so the monoidal structure is given by the product rather than the coproduct, and so feedback is interpreted in terms of least fixed points. Using domain equations, Abramsky and Jagadeesan extend this interpretation to the whole of Linear Logic. In [Abramsky and Jagadeesan(1992a); Abramsky and Jagadeesan(1994a)], Abramsky and Jagadeesan introduced game semantics for Multiplicative Linear Logic, where they also demonstrated how the Geometry of Interaction's Execution formula can be interpreted in the category of sets and partial injective functions, where now the monoidal structure is given by the coproduct and the feedback is interpreted by iteration. This showed that there were two different-looking ways of providing what was essentially structurally the same Geometry of Interaction. One is the "wave style" version, where the monoidal product is a product and the feedback is given by a global flow of information through the system. The other is the "particle style" version, where the monoidal product is a coproduct and feedback is thought of as a token following paths in a graph.

Many of the ideas for a categorical interpretation of some sort of feedback operator appear in [Abramsky and Jagadeesan(1992b); Abramsky and Jagadeesan(1994b); Abramsky and Jagadeesan(1992a); Abramsky and Jagadeesan(1994a)]. The precise axiomatization of these ideas can be given in terms of **traced monoidal categories**, which A. Joyal, R. Street, and D. Verity introduced in [Joyal et al.(1996)]. Motivated by knot theory, traced monoidal categories were defined initially in the *braided* monoidal setting. However, it is instead traced *symmetric* monoidal categories that give the desired setting for the feedback used in Geometry of Interaction. Indeed, Abramsky provides the categorical interpretation of the Geometry of Interaction using traced symmetric monoidal categories in [Abramsky(1996)], as well as providing numerous examples. In particular, Abramsky introduced the GOI-construction in [Abramsky(1996)], which is a construction that yields a *compact closed category* whose composition is given by the Execution formula. This formulation of using traced monoidal categories to provide the categorical semantics of Geometry of Interaction is referred to by Scott and Haghverdi as the "Abramsky program". However, the interpretation of the Geom-

etry of Interaction in [Abramsky(1996)] was only done for the multiplicative fragment of Linear Logic. This left open the problem of interpreting a more expressive fragment, in particular one capable of encoding the λ-calculus.

Following the Abramsky program, Scott, Abramsky, and Haghverdi then introduced the concept of *Geometry of Interaction Situations* in [Abramsky et al.(2002)] which are traced symmetric monoidal categories that have the necessary structure in which to interpret an algebraic model of the multiplicative and *exponential* fragments of Linear Logic, called a *linear combinatory algebra*. Scott, Abramsky, and Haghverdi's Geometry of Interaction Situations gave the first fully developed categorical axiomatization for the Geometry of Interaction, covering all known examples as well. Afterwards, Scott and Haghverdi in [Haghverdi and Scott(2006)] used *unique decomposition category* versions of Geometry of Interaction Situations, which give particle style models, to recapture Girard's original operator algebra model of Geometry of Interaction and Execution formula. In follow-up papers [Haghverdi and Scott(2010a); Haghverdi and Scott(2010b)], Scott and Haghverdi continued applying unique decomposition categories to give a deeper analysis of Girard's construction.

Scott and his coauthors' contributions to the categorical semantics of the Geometry of Interaction are important concepts in the theory of traced monoidal categories, influencing and motivating many research topics in the area. We survey the main concepts of the Abramsky program mentioned above. Of course, for a complete story, we invite the interested reader to see the two main papers surveyed [Abramsky et al.(2002); Haghverdi and Scott(2006)], Haghverdi's thesis [Haghverdi(2000)], an introductory paper on this story by Scott and Haghverdi [Haghverdi and Scott(2010a)], as well as some very nicely written notes by M. Shirahata [Shirahata(2003)].

2. TRACED SYMMETRIC MONOIDAL CATEGORIES

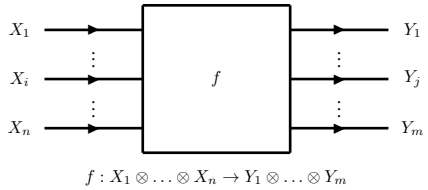
Traced monoidal categories were introduced by Joyal, Street, and Verity in [Joyal et al.(1996)], and were initially defined as *balanced* monoidal categories equipped with a *trace operator*. For the story of Geometry of Interaction, one works in the special case of a traced *symmetric* monoidal category, which is instead a symmetric monoidal category equipped with a trace operator. In this section, we review the definition of traced symmetric monoidal categories and their graphical representation.

We assume that the reader is familiar with the basics of monoidal category theory. For an introduction to monoidal category theory, we invite the reader to see [Blute and Scott(2004); Heunen and Vicary(2019); Selinger(2010)]. For a category $\mathbb{C}$, we will denote the class of objects as $Ob(\mathbb{C})$ and denote objects using capital letters $A, B, X, Y, etc. \in Ob(\mathbb{C})$. Homsets will be denoted as $\mathbb{C}(X, Y)$ and maps will be denoted by minuscule letters $f, g, h, etc. \in \mathbb{C}(X, Y)$. Arbitrary maps will be denoted using an arrow $f : X \to Y$, identity maps as $1_X : X \to X$, and for composition we will use *diagrammatic* notation, that is, the composition of $f : X \to Y$ followed by $g : Y \to Z$ is denoted as $f; g : X \to Z$. Following the conventions used in [Abramsky et al.(2002); Haghverdi and Scott(2006)], for simplicity, we will work with *strict* monoidal categories, meaning that the associativity and unit isomorphisms for the monoidal product are equalities. So for an arbitrary symmetric (strict) monoidal category $\mathbb{C}$, we denote its monoidal product as $\otimes$, the monoidal unit as I, and the natural symmetry isomorphism as $\sigma_{X,Y} : X \otimes Y \xrightarrow{\cong} Y \otimes X$. Strictness allows us to write $A_1 \otimes A_2 \otimes \ldots \otimes A_n$ and $A \otimes I = A = I \otimes A$.

Symmetric monoidal categories enjoy a wonderfully practical graphical calculus, which allows one to write equations of maps in a symmetric monoidal category as string diagrams. This graphical calculus is extremely useful since it provides a more visual representation, which often leads to better intuition for definitions, computa-

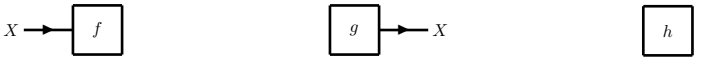
tions, and proofs. For an in-depth introduction to the graphical calculus of monoidal categories, we invite the reader to see [Heunen and Vicary(2019); Selinger(2010)].

In this paper, following the conventions in [Abramsky et al.(2002); Haghverdi and Scott(2006)], our string diagrams are to be read horizontally from left to right. So for an arbitrary map $f : X_1 \otimes \ldots \otimes X_n \to Y_1 \otimes \ldots \otimes Y_m$ will be denoted as a box with inputs on the left, one wire for each X_i, and outputs on the right, with one wire for each Y_j:



$f : X_1 \otimes \ldots \otimes X_n \to Y_1 \otimes \ldots \otimes Y_m$

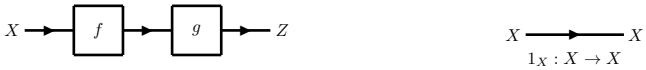
When there is no confusion, we often simply omit the object labels on the wires of our string diagrams. In the special case that one of the inputs or outputs is the monoidal unit I, we do not draw a wire representing it (which lines up with the idea that we are working in the strict case). For example:



$f : X \to I$ $\quad$ $g : I \to X$ $\quad$ $h : I \to I$

Maps of type $I \to I$, so boxes with no input or output wires, are often called "scalars".

Composition is written sequentially, while identity maps are simply drawn as a wire with no box. For example, in the simplest case:



$f ; g : X \to Z$ $\quad$ $1_X : X \to X$

Note that the identity for the monoidal unit $1_I : I \to I$ is drawn as nothing. On the other hand, the monoidal product of maps is written by placing the maps in parallel, while the symmetry isomorphism is drawn by crossing the wires:

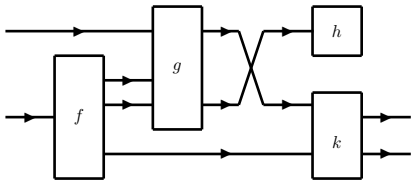


$f \otimes g : X \otimes Z \to Y \otimes W$ $\quad$ $\sigma_{X,Y} : X \otimes Y \to Y \otimes X$

As such, the usual composition is referred to as "sequential composition", while the monoidal product is referred to as "parallel composition". The advantage of a symmetric monoidal category is that symmetry allows us to cross wires without having to worry about which wire passes on top of the other wire. So we may think of the crossing of wires as the wires passing through each other uninterrupted.

Of course, with all this machinery, we can write out more complex maps in our symmetric monoidal category. For example for maps $f : Y \to A \otimes B \otimes C$, $g : X \otimes A \otimes B \to Z \otimes W$, $h : W \to I$, and $k : Z \otimes C \to E \otimes F$, the composite

$(1_X \otimes f); (g \otimes 1_C); (\sigma_{Z,W} \otimes 1_C); (h \otimes k) : X \otimes Y \to E \otimes F$ is drawn out as follows:



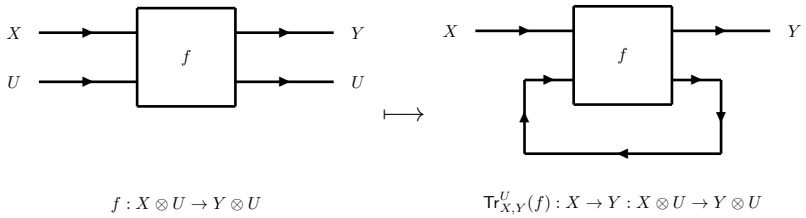
Finally, with all this setup, we can properly add traces to the story. Graphically speaking, traces for symmetric monoidal categories add the possibility of looping back an output wire into an input wire of the same type, which in turn allows loops.

There are many equivalent lists of axioms for a trace operator. Here, we will use the one presented in [Hasegawa and Lemay(2023), Definition 2.2], though we follow the string diagram conventions used in [Abramsky et al.(2002); Haghverdi and Scott(2006)]. For other equivalent axiomatizations of the trace operator, see for example [Haghverdi and Scott(2010a), Definition 4.1] or [Hasegawa(1997), Definition 2.1] or [Hasegawa(2009), Section 3].

Definition 2.1. A **traced symmetric monoidal category** is symmetric monoidal category $\mathbb{C}$ equipped with a **trace operator** Tr, which is a family of operators (indexed by triples of objects $U, X, Y \in \mathbb{C}$):

$$\mathsf{Tr}^U_{X,Y} : \mathbb{C}(X \otimes U, Y \otimes U) \to \mathbb{C}(X, Y) \qquad \frac{f : X \otimes U \to Y \otimes U}{\mathsf{Tr}^U_{X,Y}(f) : X \to Y}$$

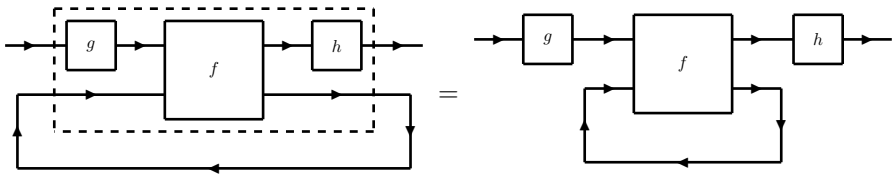
which is drawn in the graphical calculus as follows:



$f : X \otimes U \to Y \otimes U$ $\qquad$ $\mathsf{Tr}^U_{X,Y}(f) : X \to Y : X \otimes U \to Y \otimes U$

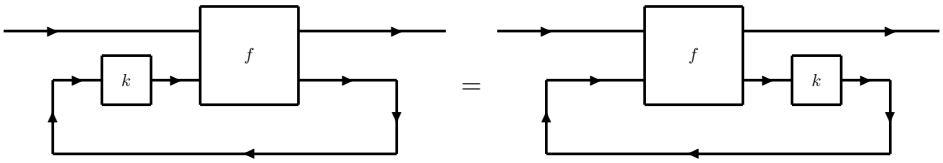
such that the following axioms are satisfied[4] (where the dotted line represents what is being traced when there is possible confusion):

— **[Tightening]**:

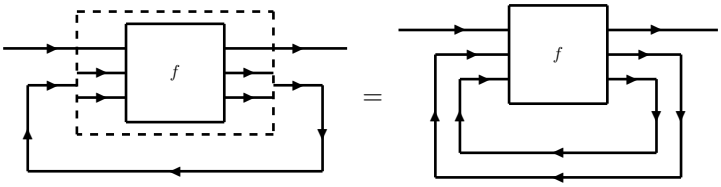


[4]For equational versions of these axioms, see [Hasegawa and Lemay(2023), Definition 2.2]

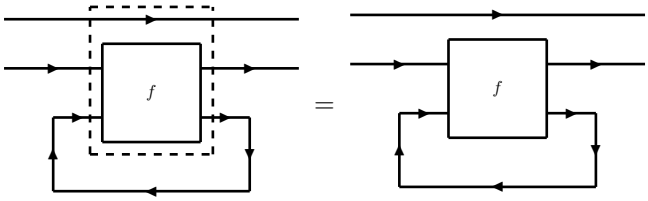
— **[Sliding]**:



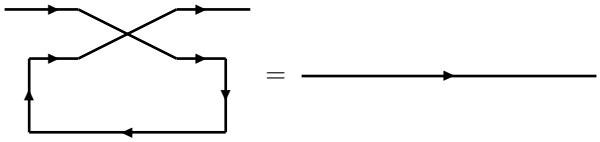
— **[Vanishing]**



— **[Superposing]**



— **[Yanking]**

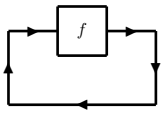


For a map $f : X \otimes U \to Y \otimes U$, the map $\mathsf{Tr}^U_{X,Y}(f) : X \to Y$ is called the **trace** of f.

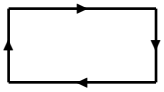
[Tighetening] says that the trace operator is natural in the X and Y arguments, meaning that we can pull maps in and out of the trace operator if they have no interaction with the argument U that's being traced. In other words, anything that happens on the wires that are not being traced does not affect the trace. **[Sliding]** says that the trace operator is instead *dinatural* in the U argument, and this corresponds to the famous cyclic property of the trace from linear algebra (which we will discuss more about below). **[Vanishing]** says that if we trace out $U_1 \otimes U_2$, its the same thing as tracing out U_2 first then tracing out U_1. Keen-eyed readers will note that in other versions of the axioms, such as in [Abramsky et al.(2002); Haghverdi and Scott(2006)], there is a second part to the vanishing axiom which says that for a map $f : X \to Y$, if seen as a map $f : X \otimes I \to Y \otimes I$, then tracing out the monoidal unit does nothing, $\mathsf{Tr}^I_{X,Y}(f) = f$. It turns out that this is, in fact, provable from the other axioms [Hasegawa(2009), Appendix A], and since graphically there is nothing to draw, we have omitted it here in the above definition. **[Superposing]** says that if there is a part that is completely disconnected from what is being traced, then it can be taken out of the trace. Finally, **[Yanking]** tells us what the trace of the symmetry is, and essentially says that there are no knots in this framework.

Before we consider examples, let's first observe that we can also take traces of endomorphisms to give us actual loops. Indeed, note that an endomorphism $f : U \to U$ can

be viewed as a map of type $f : I \otimes U \to I \otimes U$. As such, we may take its trace to obtain the scalar $\mathsf{Tr}^U_{I,I}(f) : I \to I$. Graphically, this is drawn as a loop with a box:

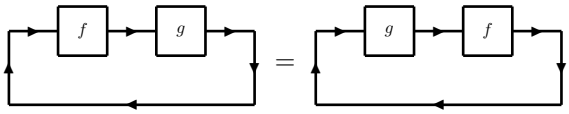


When taking the trace of the identity $1_U : U \to U$, we obtain a scalar which is drawn as an actual loop:



and this is often called the "dimension" of the object U. A simple observation is that we can recapture the famous cyclic property of the trace for matrices in linear algebra. This shows that we can slide endomorphisms around the loop.

LEMMA 2.2. *In a traced symmetric monoidal category, the following equality holds:*



PROOF. This is just a special case of **[Sliding]**. □

Let's now review how the trace operation for matrices gives us an example of a traced symmetric monoidal category.

Example 2.3. Let R be a commutative ring. Let $\mathsf{MAT}(R)$ be the category of matrices over R, that is, the category whose objects are natural numbers $n \in \mathbb{N}$ and where a map $A : m \to n$ is an $m \times n$ matrix A with coefficients in R, with composition given by matrix multiplication and identity maps given by identity matrices. $\mathsf{MAT}(R)$ is a trace symmetric monoidal category whose trace operator is the standard partial trace of matrices from linear algebra. The monoidal product for $\mathsf{MAT}(R)$ is defined on objects by multiplication $m \otimes n = mn$, and on maps as the usual tensor product of matrices. For the trace operator, first recall that for square $n \times n$ matrix A, its trace is equal to the sum of its diagonal coefficients, $\mathsf{Tr}(A) = \sum_{i=1}^{n} A_{i,i}$. Now observe that a map $A : m \otimes k \to n \otimes k$, which is an $mk \times nk$ matrix A, can be expressed in terms of square matrices as follows:

$$A = \begin{bmatrix} A(1,1) & A(1,2) & \dots & A(1,n) \\ A(2,1) & A(2,2) & \dots & A(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ A(m,1) & A(m,2) & \dots & A(m,n) \end{bmatrix}$$

where $A(i,j)$ are square $k \times k$ matrices (with $1 \leq i \leq m$ and $1 \leq j \leq n$). Then the trace $\mathsf{Tr}^k_{m,n}(A) : m \to n$ is the $m \times n$ matrix $\mathsf{Tr}^k_{m,n}(A)$ whose coefficients are the traces of the square matrices:

$$\mathsf{Tr}^k_{m,n}(A) = \begin{bmatrix} \mathsf{Tr}\,(A(1,1)) & \mathsf{Tr}\,(A(1,2)) & \dots & \mathsf{Tr}\,(A(1,n)) \\ \mathsf{Tr}\,(A(2,1)) & \mathsf{Tr}\,(A(2,2)) & \dots & \mathsf{Tr}\,(A(2,n)) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{Tr}\,(A(m,1)) & \mathsf{Tr}\,(A(m,2)) & \dots & \mathsf{Tr}\,(A(m,n)) \end{bmatrix}$$

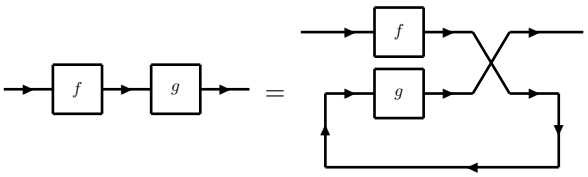
Now note that an endomorphism $A : n \to n$ is precisely a square $n \times n$ matrix A. Therefore, in this case, since the monoidal unit is $I = 1$, we get that the trace operator for endomorphisms gives us back precisely the usual trace for square matrices, that is, $\mathsf{Tr}^n_{1,1}(A) = \mathsf{Tr}(A)$. So Lemma 2.2 corresponds precisely to the all-famous identity that $\mathsf{Tr}(AB) = \mathsf{Tr}(BA)$.

There are many other interesting examples of traced symmetric monoidal categories, see [Haghverdi and Scott(2010a), Examples 4.5 & 4.6] for lists of more examples. In particular, below we will review two important classes of traced symmetric monoidal categories: compact closed categories and traced unique decomposition categories. Other important examples worth mentioning are when the monoidal product is a product, coproduct, or even a biproduct. For products, trace operators capture the notion of feedback via fixpoints, and in fact, to give a trace operator for products is equivalent to providing a special kind of fixpoint operator (called a *Conway operator*) [Hasegawa(1997), Theorem 3.1]. On the other hand, for coproducts, trace operators capture the notion of feedback via iteration, and similarly, to give a trace operator for coproducts is equivalent to giving a special kind of iteration operator [Selinger(2010), Proposition 6.8]. For biproducts, to give a trace operator is equivalent to giving a repetition operator on endomorphisms [Selinger(2010), Proposition 6.11]. Scott and his coauthors often liked to refer to traces for compact closed categories and products as "product or wave style traces" [Haghverdi and Scott(2010a), Examples 4.5], and traces for unique decomposition categories and coproducts as "sum or particle style traces" [Haghverdi and Scott(2010a), Examples 4.5]. Briefly, the intuition is that a product style trace can be understood as passing information in a global information wave, while a sum style trace is interpreted by streams of particles or tokens flowing around a network – see [Abramsky(1996); Abramsky et al.(2002)] for more explanation on these ideas.

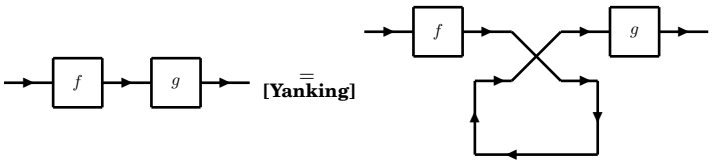
An important identity in a traced symmetric monoidal category is that sequential composition can be reformulated in terms of parallel composition using the trace.

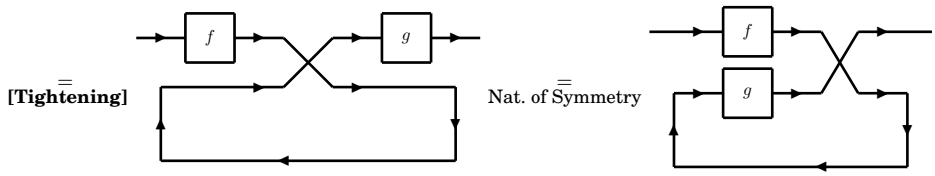
THEOREM 2.4. *[Abramsky et al.(2002), Proposition 2.4] In a traced symmetric monoidal category, the following equality holds:*

—[Generalized Yanking]



PROOF. To demonstrate the usefulness of working with string diagrams, let's prove this using the graphical calculus. So we compute:





We invite the reader to compare these computations with the equational version of this computation found in the proof of [Abramsky et al.(2002), Proposition 2.4]. □

It is also possible to axiomatize trace operators with **[Generalized Yanking]** as an axiom instead, see [Haghverdi(2000), Proposition 2.1.21]. Moreover, using **[Generalized Yanking]**, we get a normal form theorem for maps in traced symmetric monoidal categories.

THEOREM 2.5. *[Abramsky et al.(2002), Theorem 2.5] Let T be a set of maps in a traced symmetric monoidal category $\mathbb{C}$. Then any expression E built from maps in T built from using the monoidal product, composition, and trace can be expressed as $\mathsf{Tr}(\pi; F; \tau)$ where F consists of a monoidal product of maps in T, and π and τ are permutations (i.e. maps constructed from symmetry and identity maps using monoidal product and composition).*

Essentially, from the point of view of the graphical calculus, this normal form theorem says we can always re-draw a map in such a way that we don't have any sequential composition of boxes, and only have parallel composition of boxes.

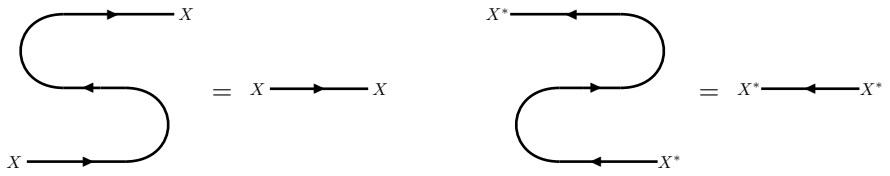
3. COMPACT CLOSED CATEGORIES

An important class of traced symmetric monoidal categories are *compact closed categories*, which are particularly important in categorical quantum foundations [Abramsky and Coecke(2004); Heunen and Vicary(2019)]. Compact closed categories are symmetric monoidal categories where every object has a dual. Every compact closed category comes equipped with a canonical trace operator that captures the classical notion of (partial) trace for matrices, which is a fundamental operation for both classical quantum theory and categorical quantum foundations [Abramsky and Coecke(2005)]. For a more in-depth introduction to compact closed categories, we invite the reader to see [Selinger(2010), Section 4.8] or [Heunen and Vicary(2019), Chapter 3].

Definition 3.1. A **compact closed category** is a symmetric monoidal category such that for every object X, there is an object X^*, called the **dual** of X, and maps $\cup_X : X^* \otimes X \to I$, called the **cup**, and $\cap_X : I \to X \otimes X^*$, called the **cap**, which are drawn in the graphical calculus as follows:



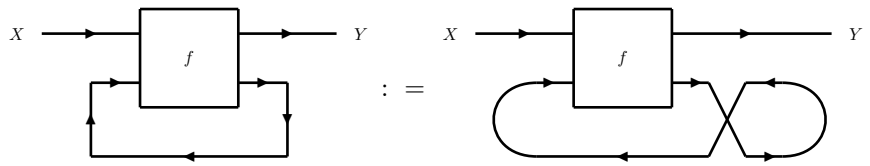
such that the following equality holds:



and these equalities are called the snake equations.

Every compact closed category is a traced symmetric monoidal category, where the trace operator is constructed using the cups and caps, and where the trace operator axioms follow from the snake equations. Furthermore, compact closed categories have a *unique* trace operator.

PROPOSITION 3.2. *[Joyal et al.(1996), Proposition 3.1] A compact closed category is a traced symmetric monoidal category where the trace operator is defined as follows:*



Furthermore, this is the unique *trace operator on a compact closed category [Hasegawa(2009), Section 3.2].*

It is also worth mentioning that, as their name suggests, compact closed categories are also symmetric monoidal *closed* categories. In fact, compact closed categories are also *star-autonomous* categories. It is also interesting to note that the converse is also true, that is, *traced* star-autonomous categories are compact closed [Hajgató and Hasegawa(2013), Theorem 3.6].

There are many interesting examples of compact closed categories. In fact, we have already encountered one, as the category of matrices from Example 2.3 is a compact closed category. However, for a more intuitive example, let's review how finite dimensional vector spaces form a compact closed category.

Example 3.3. Let k be a field, and let FVEC_k be the category of finite dimensional k-vector spaces and k-linear transformations between them. FVEC_k is a compact closed category, where the monoidal product is the standard algebraic tensor product of vector spaces, so in particular $I = k$, and where the dual of a k-vector space U is given by its algebraic dual, that is, $U^* = \{\phi : U \to k |\ \phi \text{ is } k\text{-linear}\}$. The cup $\cup_U : U^* \otimes U \to k$ is defined by evaluation:

$$\cup_U(\phi \otimes x) = \phi(x)$$

To define the cap, let $\{e_1, \ldots, e_n\}$ be a basis of U, which induces a basis $\{e_1^*, \ldots, e_n^*\}$ for U^*. Then the cap $\cap_U : k \to U \otimes U^*$ is defined as:

$$\cap_U(1) = \sum_{i=1}^{n} e_i \otimes e_i^*$$

For a k-linear map $f : V \otimes U \to W \otimes U$, its trace is the k-linear map $\mathsf{Tr}^U_{V,W}(f) : V \to W$ defined as:

$$\mathsf{Tr}^U_{V,W}(f)(m) = \sum_{i=1}^{n} (1_W \otimes e_i^*)\,(f(m \otimes e_i))$$

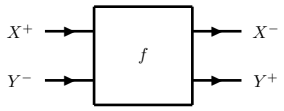
Note that the definitions of the cap and trace are independent of the choice of basis. In particular, when $k = \mathbb{C}$, the field of complex numbers, then $\mathsf{FVEC}_{\mathbb{C}}$ is equivalent to the category of finite dimensional Hilbert spaces, which is also compact closed and the fundamental category of study in categorical quantum foundations.

On the other hand, not every traced symmetric monoidal category is a compact closed category. That said, every traced symmetric monoidal category embeds fully and faithfully into a compact closed category via the INT-construction [Joyal et al.(1996), Section 4]. The name INT-construction comes from the fact that when you apply this construction to the natural numbers $\mathbb{N}$ you get the integers $\mathbb{Z}$ [Haghverdi and Scott(2010a), Example 4.13]. Abramsky introduced another way of constructing a compact closed category from a traced symmetric monoidal category via the so-called "Geometry of Interaction construction" [Abramsky(1996), Section 3], or simply GOI-construction for short. It turns out that the resulting compact closed categories from the INT-construction and the GOI-construction are isomorphic [Haghverdi(2000), Proposition 2.3.6]. However, it is the GOI-construction that better isolates the fundamental properties of Girard's original ideas for Geometry of Interaction, where, in particular, the composition in this compact closed category corresponds to Girard's Execution formula. As such, it is the GOI-construction that plays a more central role in the categorical approaches to Geometry of Interaction in [Abramsky(1996); Abramsky et al.(2002); Haghverdi(2000); Haghverdi and Scott(2010a)].

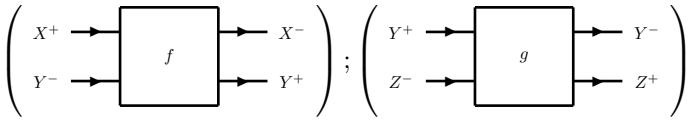
Definition 3.4. For a traced symmetric monoidal category $\mathbb{C}$, define the compact closed category $\mathcal{G}(\mathbb{C})$ as follows:

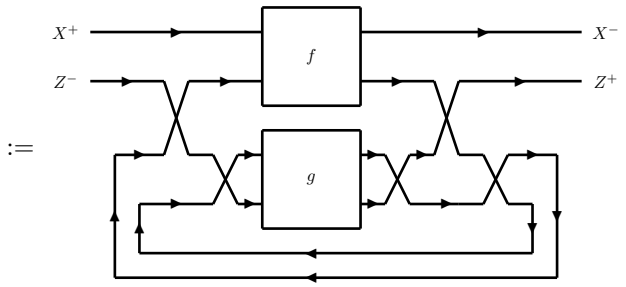
(i) The objects are pairs of objects of $\mathbb{C}$, so $\mathcal{O}\left(\mathcal{G}(\mathbb{C})\right) = \mathcal{O}(\mathbb{C}) \times \mathcal{O}(\mathbb{C})$. We will suggestively denote objects of $\mathcal{G}(\mathbb{C})$ as pairs (X^+, X^-), though this is simply notation and there is no relationship between X^+ and X^-.

(ii) The homsets are $\mathcal{G}(\mathbb{C})\left((X^+, X^-), (Y^+, Y^-)\right) = \mathbb{C}(X^+ \otimes Y^-, X^- \otimes Y^+)$. So a map $f : (X^+, X^-) \to (Y^+, Y^-)$ in $\mathcal{G}(\mathbb{C})$ is a map of type $f : X^+ \otimes Y^- \to X^- \otimes Y^+$ in $\mathbb{C}$.

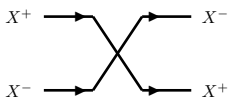


(iii) Composition is defined as follows:



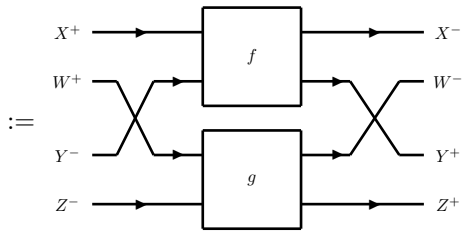


(iv) The identity map $1_{(X^+,X^-)} : (X^+, X^-) \to (X^+, X^-)$ is the symmetry map in $\mathbb{C}$, that is, $1_{(X^+,X^-)} = \sigma_{X^+,X^-}$:



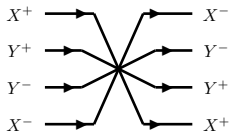
(v) On objects, the monoidal product is defined as the pointwise monoidal product of pairs $(X^+, X^-) \otimes (Y^+, Y^-) = (X^+ \otimes Y^+, X^- \otimes Y^-)$, while on maps, the monoidal product is defined as follows:



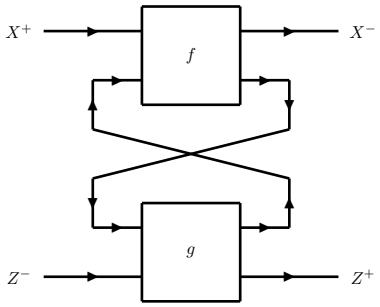


(vi) The monoidal unit is the pair (I, I);

(vii) The symmetry $\sigma_{(X^+,X^-),(Y^+,Y^-)} : (X^+, X^-) \otimes (Y^+, Y^-) \to (Y^+, Y^-) \otimes (X^+, X^-)$ is the permutation $X^+ \otimes Y^+ \otimes Y^- \otimes X^- \to X^- \otimes Y^- \otimes Y^+ \otimes X^+$:



A more intuitive way of understanding the composition in $\mathcal{G}(\mathbb{C})$ is by slightly abusing the flexibility of the wires in the graphical calculus and drawing composition as follows:



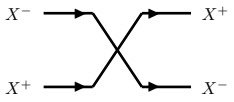
Drawn like this, we see that composition in $\mathcal{G}(\mathbb{C})$ is given by symmetric feedback. Moreover, observe that the composition in $\mathcal{G}(\mathbb{C})$ only depends on the parallel composition of boxes, similar to that of **[Generalized Yanking]**. As mentioned above, $\mathcal{G}(\mathbb{C})$ is a compact closed category, where it turns out that the cups and caps are simply given by the symmetry maps. Furthermore, we can also embed $\mathbb{C}$ into $\mathcal{G}(\mathbb{C})$, and this embedding preserves the traced symmetric monoidal structure strictly.

PROPOSITION 3.5. *[Abramsky et al.(2002), Proposition 2.8] For a traced symmetric monoidal category $\mathbb{C}$, $\mathcal{G}(\mathbb{C})$ is compact closed where:*

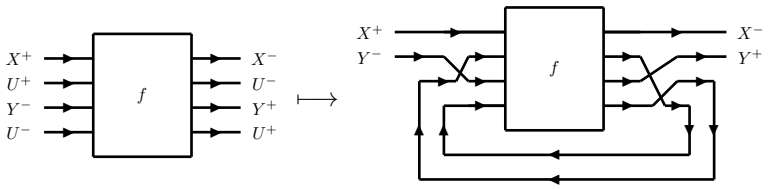
(i) *The dual of (X^+, X^-) is $(X^+, X^-)^* = (X^-, X^+)$*

(ii) *The cup $\cup_{(X^+,X^-)} : (X^-, X^+) \otimes (X^+, X^-) \to (I, I)$ and the cap $\cap_{(X^+,X^-)} : (I, I) \to (X^+, X^-) \otimes (X^-, X^+)$, which are both maps of type $X^- \otimes X^+ \to X^+ \otimes X^-$ in $\mathbb{C}$,*

are given by the symmetry map in $\mathbb{C}$, that is, $\cup_{(X^+,X^-)} = \cap_{(X^+,X^-)} = \sigma_{X^-,X^+}$:



(iii) *The induced traced of a map $f : (X^+, X^-) \otimes (U^+, U^-) \to (Y^+, Y^-) \otimes (U^+, U^-)$, which is a map of type $f : X^+ \otimes U^+ \otimes Y^- \otimes U^- \to X^- \otimes U^- \otimes Y^+ \otimes U^+$ in $\mathbb{C}$, is the double trace of f in $\mathbb{C}$, that is, $\mathsf{Tr}^{(U^+,U^-)}_{(X^+,X^-),(Y^+,Y^-)}(f) : (X^+, X^-) \to (Y^+, Y^-)$, which is drawn on the right as follows:*



Moreover, there is a full and faithful embedding functor $\mathcal{N} : \mathbb{C} \to \mathcal{G}(\mathbb{C})$ which is defined on objects as $\mathcal{N}(X) = (X, I)$ and on maps as $\mathcal{N}(f) = f$. Furthermore, $\mathcal{N}$ preserves the traced symmetric monoidal structure strictly, so in particular we have that $\mathcal{N}(\mathsf{Tr}^U_{X,Y}(f)) = \mathsf{Tr}^{\mathcal{N}(U)}_{\mathcal{N}(X),\mathcal{N}(Y)}(\mathcal{N}(f))$.

Some explicit descriptions of the GOI-construction applied to certain traced symmetric monoidal categories are described in [Abramsky(1996), Section 4]. In particular, when applying the GOI-construction to the category of partial injections or a certain subcategory of Hilbert spaces, it recaptures Girard's original model of Geometry of Interaction. Also, as mentioned above, we have an isomorphism of compact closed categories $\mathcal{G}(\mathbb{C}) \simeq \mathcal{INT}(\mathbb{C})$, where the latter is the INT-construction over $\mathbb{C}$. As such, the GOI-construction is also the free compact closed category over $\mathbb{C}$ in an appropriate bicategorical sense [Haghverdi and Scott(2010a), Proposition 4.14]. For more details on the INT-construction, and its comparision to the GOI-construction, see [Haghverdi(2000), Section 2.2 & 2.3].

4. UNIQUE DECOMPOSITION CATEGORIES

Another important sort of traced symmetric monoidal category that Scott and Haghverdi studied for the categorical interpretations of Geometry of Interaction were *traced unique decomposition categories* [Haghverdi and Scott(2006); Haghverdi and Scott(2010a)]. In particular, in [Haghverdi and Scott(2006)], Scott and Haghverdi explain how traced unique decomposition categories provide the correct categorical framework which properly captures Girard's original Hilbert space model of Geometry of Interaction.

Unique decomposition categories (UDC) were originally introduced by Haghverdi in their PhD thesis [Haghverdi(2000), Chapter 4], and are generalizations of Manes and Arbib's partially additive categories [Manes and Arbib(2012), Section 3.2]. Briefly, a UDC is a symmetric monoidal category with an appropriate notion of countable sums and a matrix representation for maps. This countable sum property is captured by the fact that the homsets of a UDC are **Σ-monoids**, which are generalizations of partially additive monoids [Manes and Arbib(2012), Section 3.1] by dropping the so-called limit axiom. So a **Σ-monoid** is a non-empty set M equipped with a partial operation Σ called the **partial infinitary sum** on countable families of elements of M satisfying a partition-associativity axiom and a unary sum axiom. For a countable family $\{x_i\}_{i \in I}$

we denote its image as $\Sigma_{i\in I}x_i$ and say that $\{x_i\}_{i\in I}$ is **summable** if $\Sigma_{i\in I}x_i$ is defined. Then, the two necessary axioms of a Σ-monoid are:

(i) **Partitition-Associativity Axiom**: If $\{x_i\}_{i\in I}$ is a countable family and if $\{I_j\}_{j\in J}$ is a countable partition of I, then $\{x_i\}_{i\in I}$ is summable if and only if $\{x_i\}_{i\in I_j}$ is summable for every $j \in J$ and $\{\sum_{i\in I_J} x_i\}_{j\in J}$ is summable. Moreover, in this case $\sum_{i\in I} x_i = \sum_{j\in J}\sum_{i\in I_j} x_i$.

(ii) **Unary Sum Axiom:** Every singleton $\{x\}$ is summable and $\sum_{i\in I} x_i = x$ where $\{x\}$ is seen as an $I = \{i\}$ indexed set.

In any Σ-monoid, the empty family is summable, which we denote as 0, and it is an additive unit. A Σ-monoid morphism is a function $f : M \to M'$ between the underlying sets which preserves summability and the partial infinitary sums, that is, if $\{x_i\}_{i\in I}$ is summable in M then $\{f(x_i)\}_{i\in I}$ is summable in M' and $f\left(\sum_{i\in I} x_i\right) = \sum_{i\in I} f(x_i)$.

Now Σ-monoids and Σ-monoid morphisms form a symmetric monoidal closed category ΣMON. Therefore, we may consider categories enriched in ΣMON. Explicitly, a ΣMON-category is a category $\mathbb{C}$ such that each homset is a Σ-monoid where composition preserves the partial infinitary sums, that is, if $\{f_i\}_{i\in I}$ is a summable family of maps in $\mathbb{C}(X, Y)$, then for any suitable maps we also have that $\{g; f_i; h\}_{i\in I}$ is also summable and:

$$g; \left(\sum_{i\in I} f_i\right); h = \sum_{i\in I} g; f_i; h$$

Note that a ΣMON-category has non-empty homsets since the sum of the empty set gives us the map $0_{X,Y} : X \to Y$, which is in fact an actual zero morphism in the categorical sense. Now a symmetric monoidal ΣMON-category is a symmetric monoidal category $\mathbb{C}$ which is also a ΣMON-category such that the monoidal product preserves the partial infinitary sums, that is, if $\{f_i\}_{i\in I}$ and $\{g_i\}_{j\in J}$ are summable family of maps, then $\{f_i \otimes g_j\}_{(i,j)\in I\times J}$ is also a summable family of maps and:

$$\left(\sum_{i\in I} f_i\right) \otimes \left(\sum_{j\in J} g_j\right) = \sum_{i\in I}\sum_{j\in J} f_i \otimes g_j$$

Then a UDC is a symmetric monoidal ΣMON-category which comes equipped with quasi-injections and quasi-projections that satisfy analogues of the axioms for biproducts.

Definition 4.1. A **unique decomposition category (UDC)** is a symmetric monoidal ΣMON-category such that for all finite family of objects $\{X_1, \ldots, X_n\}$ and every $1 \leq j \leq n$ there are maps $\iota_j : X_j \to X_1 \otimes \ldots \otimes X_n$, called the **quasi-injections**, and maps $\rho_j : X_1 \otimes \ldots \otimes X_n \to X_j$, called the **quasi-projections**, such that:

(i) $\iota_j; \rho_j = 1_{X_j}$ and $\iota_j l \rho_i = 0_{X_i,X_j}$ if $i \neq j$

(ii) $\sum_{j=1}^{n} \iota_j; \rho_j = 1_{X_1\otimes\ldots\otimes X_n}$

It is important to note that these quasi-injections and quasi-projections do not make the monoidal product a biproduct. Nevertheless, with these quasi-injections and quasi-projections, we can give a matrix representation for maps in a UDC [Haghverdi(2000), Proposition 4.0.6] similar to the matrix representation we can give for maps in a category with finite biproducts [Heunen and Vicary(2019), Section 2.2.4]. Indeed, for a $f : X_1 \otimes \ldots \otimes X_n \to Y_1 \otimes \ldots \otimes Y_m$, there exists a unique summable finite family of maps

$\{f_{i,j} : X_i \to Y_j\}_{i\in I, j\in J}$ such that:

$$f = \sum_{i\in I, j\in J} \rho_i; f_{i,j}; \iota_j$$

Explicitly, the maps $f_{i,j}$ are given by pre-compositing and post-composing f by the appropriate quasi-injection and quasi-projection respectively, that is, $f_{i,j} := \iota_i; f; \rho_j$. As such, we may represent f as an $n \times m$ matrix with (i, j)-component the map $f_{i,j} : X_i \to Y_j$:

$$f = \begin{bmatrix} f_{1,1} & \cdots & f_{1,m} \\ \vdots & \ddots & \vdots \\ f_{n_1} & \cdots & f_{n,m} \end{bmatrix}$$

This matrix representation is particularly useful since composition in a UDC then corresponds to matrix multiplication.

Now, since maps in a UDC have a matrix representation like in a category with biproducts, it is natural to ask if an analogue of the well-known trace formula for countable biproducts gives a trace for UDC. If the necessary sum is always defined, then the answer is yes! As such, we say that a UDC is **traced** if for every map of type $f : X \otimes U \to Y \otimes U$, which in matrix form is:

$$f := \begin{bmatrix} f_{1,1} : X \to Y & f_{1,2} : X \to U \\ f_{2,1} : U \to Y & f_{2,2} : U \to U \end{bmatrix}$$

the sum $f_{1,1} + \sum_{n\in\mathbb{N}} f_{1,2}; f^n_{2,2}; f_{2,1}$ exists, which is a map of type $X \to Y$. This then gives us the *standard trace formula* for a trace operator, justifying the name traced UDC.

THEOREM 4.2. *[Haghverdi(2000), Proposition 4.0.11] A traced UDC is a traced symmetric monoidal category with trace operator defined on a map $f : X \otimes U \to Y \otimes U$ as follows:*

$$\mathsf{Tr}^U_{X,Y}(f) = f_{1,1} + \sum_{n\in\mathbb{N}} f_{1,2}; f^n_{2,2}; f_{2,1}$$

We note that the matrix representation greatly simplifies the proof of showing that the standard trace formula does indeed give a trace operator. It is also worth mentioning that a (traced) UDC could have another trace operator which is not given by the standard trace formula. However, when talking about a traced UDC, we are referring to the trace operator given above by the standard trace formula.

The two main examples of traced UDC are the category of partial injections and a certain subcategory of Hilbert spaces. These two examples, especially the latter, capture Girard's operator algebra model of Geometry of Interaction.

Example 4.3. Let PINJ be the category of sets and partial injections between them. Then PINJ is a traced UDC [Haghverdi and Scott(2006), Example 8] where:

(i) The monoidal product is given by the disjoint union of sets, $X \otimes Y = X \sqcup Y := (\{1\} \times X) \cup (\{2\} \times Y)$, while the monoidal unit is the empty set, $I = \emptyset$;

(ii) $\mathsf{PINJ}(X, Y)$ is a Σ-monoid where a countable family of partial injections $\{f_i : X \to Y\}_{i\in I}$ is summable if they have pairwise disjoint domains and codomains, and in this case, their sum is given as follows:

$$\left(\sum_{i\in I} f_i\right)(x) = \begin{cases} f_j(x) & \text{if } x \in \mathsf{dom}(f_j) \text{ for some } j \in I \\ \text{undefined} & \text{o.w.} \end{cases}$$

(iii) The quasi-injection $\iota_j : X_j \to X_1 \sqcup \ldots \sqcup X_n$ inserts X_j into the disjoint union, while the quasi-projection $\rho_j : X_1 \sqcup \ldots \sqcup X_n \to X_j$ is only defined on the X_j part:

$$\iota_j(x) = (j, x) \qquad \rho_j(i, x) = \begin{cases} x & \text{if } i = j, \text{ so } x \in X_j \\ \text{undefined} & \text{if } i \neq j \end{cases}$$

(iv) For a partial injection $f : X \sqcup U \to Y \sqcup Y$, its induced trace is worked out to be the partial injection $\mathsf{Tr}^U_{X,Y}(f) : X \to Y$ defined as follows:

$$\mathsf{Tr}^U_{X,Y}(f)(x) = \begin{cases} y & \text{if } f(1,x) = (1,y) \\ y_{n+1} & \text{if } \exists\, n \in \mathbb{N}\ \exists\, u_0, u_1, \ldots, u_n \in U \text{ s.t.} \\ & f(1,x) = (2,u_0) \text{ and } f(2,u_0) = (2,u_1) \text{ and} \\ & \ldots \text{ and } f(2,u_n) = (1,y_{n+1}) \\ \text{undefined} & \text{o.w.} \end{cases}$$

More intuitively, the trace starts by taking an $x \in X$, applying f and if $f(1,x)$ is defined, checking if the second component of $f(1,x)$ is in Y or in U. If it's in Y, then we are done and output the Y component. If it's in U, then we apply f again, and if $f(2, f(1,x))$ is defined, then we check again if its second component is in Y or U. If it's in Y, we are done and output the Y component. Otherwise, if it's in U, we repeat the process again. We do this until if after a finite number of iterations, we land in Y and then output the result. Otherwise, the trace is undefined.

Example 4.4. Let HILB be the category of Hilbert spaces and linear contractions (with norm ≤ 1). Now while HILB is a symmetric monoidal category, where the monoidal product is the usual tensor product of Hilbert spaces, it is not a traced UDC. To obtain a UDC, we must instead consider a subcategory of HILB. So for a set X, let $\ell_2(X)$ be the set of complex-valued functions $a : X \to \mathbb{C}$ such that the sum $\sum_{x \in X} |a(x)|^2$ is finite (or equivalently the set of square summable X-indexed families). Then $\ell_2(X)$ is a Hilbert space with inner product $\langle a, b \rangle = \sum_{x \in X} a(x)\overline{b(x)}$ and induced norm $\|a\| = \left(\sum_{x \in X} |a(x)|^2\right)^{\frac{1}{2}}$. This induces a contravariant faithful functor $\ell_2 : \mathsf{PINJ} \to \mathsf{HILB}$ [Barr(1992), Section 3] which maps a set X to $\ell_2(X)$, and maps a partial injection $f : X \to Y$ to the linear contraction $\ell_2(f) : \ell_2(Y) \to \ell_2(X)$ defined as follows:

$$\ell_2(f)(a)(y) = \begin{cases} a(f(x)) & \text{if } x \in \mathsf{dom}(f) \\ 0 & \text{if } x \notin \mathsf{dom}(f) \end{cases}$$

Note that $\ell_2(f)$ is in fact a partial isometry. In fact, ℓ_2 sends certain special maps in PINJ to important kinds of maps in HILB, such as the fact total bijections get mapped to unitaries in HILB – see [Haghverdi and Scott(2006)] for more details. Heunen also studies the functor ℓ_2 in more detail in [Heunen(2013)]. Now consider the image of ℓ_2, that is, define $\mathsf{HILB}_2 := \ell_2(\mathsf{PINJ})$ to be the category whose objects are Hilbert spaces of the form $\ell_2(X)$ for some set X and whose maps are of the form $\ell_2(f)$ for some partial injection f. Then HILB_2 is naturally equivalent to PINJ, and therefore HILB_2 is a traced UDC [Haghverdi and Scott(2006), Example 9] whose structure is induced from the one of PINJ. Explicitly, the traced UDC structure of HILB_2 is given as follows:

(i) The monoidal product is given by the biproduct $\oplus$ of Hilbert spaces since we have that:

$$\ell_2(X \sqcup Y) \cong \ell_2(X) \oplus \ell_2(Y)$$

However, it is important to note that this is not a biproduct in HILB_2. The monoidal unit is the zero Hilbert space, $\ell_2(\emptyset) = \{0\}$;

(ii) $\mathsf{HILB}_2(\ell_2(Y), \ell_2(X))$ is a Σ-monoid where a countable family $\{\ell_2(f)_i : \ell_2(Y) \to \ell_2(X)\}_{i \in I}$ is summable if and only if $\{f_i : X \to Y\}_{i \in I}$ is summable in PINJ. In this case, their sum is:

$$\ell_2\left(\sum_{i \in I} f_i\right) := \ell_2\left(\sum_{i \in I} f_i\right)$$

(iii) The quasi-injection/projections are the quasi-projections/injections from PINJ, so we have that $\ell_2(\rho_j) : \ell_2(X_j) \to \ell_2(X_1 \sqcup \ldots \sqcup X_n)$ are the quasi-injections and similarly that $\ell_2(\iota_j) : \ell_2(X_1 \sqcup \ldots \sqcup X_n) \to \ell_2(X_j)$ are the quasi-projections. From another perspective, the quasi-injections are the usual injections for the biproduct $\ell_2(X_j) \to \ell_2(X_1) \oplus \ldots \oplus \ell_2(X_n)$, while the quasi-projections are the usual projections for the biproduct $\ell_2(X_1) \oplus \ldots \oplus \ell_2(X_n) \to \ell_2(X_j)$.

(iv) For a map $\ell_2(f) : \ell_2(Y \sqcup U) \to \ell_2(X \sqcup U)$, its trace is the image of the trace of its associated partial injection $f : X \sqcup U \to Y \sqcup U$ in PINJ:

$$\mathsf{Tr}^{\ell_2(U)}_{\ell_2(Y),\ell_2(X)}(\ell_2(f)) := \ell_2\left(\mathsf{Tr}^{U}_{X,Y}(f)\right)$$

Other examples of traced UDC can be found in [Haghverdi(2000), Example 4.0.15] and [Haghverdi and Scott(2010a), Example 4.11], which include partially additive categories and categories with countable biproducts.

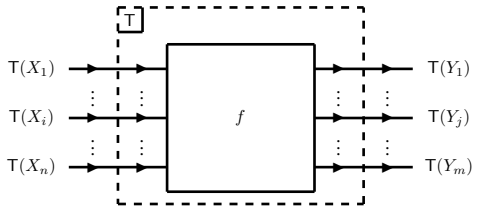
5. GEOMETRY OF INTERACTION SITUATIONS

Traced symmetric monoidal categories and the GOI-construction only account for the Geometry of Interaction interpretation of the *multiplicative* fragment of Linear Logic. One of the main objectives of [Abramsky et al.(2002)] was to provide the categorical semantics for the Geometry of Interaction interpretation of the *exponential* fragment of Linear Logic as well. As such, towards this goal, Scott, Abramsky, and Haghverdi introduced *Geometry of Interaction Situations* [Abramsky et al.(2002), Definition 4.1], which are in particular traced symmetric monoidal categories equipped with a *traced symmetric monoidal functor* and various structural maps, which is used to interpret the exponential modality ! and its rules in the GOI-construction.

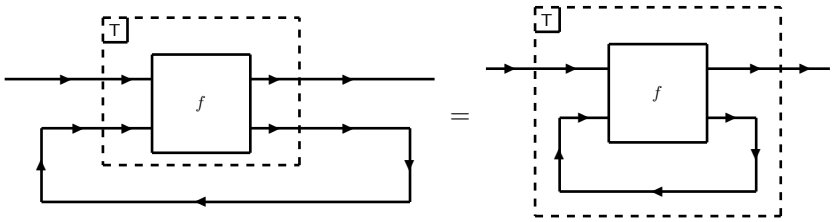
A **traced symmetric monoidal endofunctor** [Abramsky et al.(2002), Section 2] is an endofunctor on a traced symmetric monoidal category which preserves the traced symmetric monoidal structure up to isomorphism. So for a symmetric monoidal category $\mathbb{C}$, recall that an endofunctor $\mathsf{T} : \mathbb{C} \to \mathbb{C}$ is said to be *strong symmetric monoidal* if it has natural isomorphisms $\mathsf{T}(X \otimes Y) \cong \mathsf{T}(X) \otimes \mathsf{T}(Y)$ and $\mathsf{T}(I) \cong I$ which are compatible with the symmetric monoidal structure. In the graphical calculus, we will draw the application of the strong symmetric monoidal functor using a dotted functor box. That T preserves the monoidal structure up to isomorphism allows us to draw wires coming in and out of the box without any issues. So for a map $f : X_1 \otimes \ldots \otimes X_n \to Y_1 \otimes \ldots \otimes Y_m$, the composite:

$$\mathsf{T}(X_1) \otimes \ldots \otimes \mathsf{T}(X_n) \xrightarrow{\cong} \mathsf{T}\left(X_1 \otimes \ldots X_n\right) \xrightarrow{\mathsf{T}(f)} \mathsf{T}\left(Y_1 \otimes \ldots Y_m\right) \xrightarrow{\cong} \mathsf{T}(Y_1) \otimes \ldots \otimes \mathsf{T}(Y_m)$$

is drawn as follows:



Then if $\mathbb{C}$ is a traced symmetric monoidal category, a *traced symmetric monoidal endofunctor* is a strong symmetric monoidal endofunctor $\mathsf{T} : \mathbb{C} \to \mathbb{C}$ which also preserves the trace in the sense that the following equality holds:



Another key part of the definition of a Geometry of Interaction Situation are *retraction* pairs. Using the same notation as in [Abramsky et al.(2002)], a retraction pair will be denoted by $f : X \lhd Y : g$ to mean that X is a retract of Y via f and g, that is, $f; g = 1_X$.

Definition 5.1. A **Geometry of Interaction Situation (GoI Situation)** is a triple $(\mathbb{C}, \mathsf{T}, U)$ where $\mathbb{C}$ is a traced symmetric monoidal category, $\mathsf{T} : \mathbb{C} \to \mathbb{C}$ is a traced symmetric monoidal endofunctor, and $U \in \mathcal{O}(\mathbb{C})$ is an object, called the **reflexive object**, such that T comes equipped with the following retraction pairs of *monoidal* natural transformations:

— **[Digging]**: $\mathsf{e}_X : \mathsf{TT}(X) \lhd \mathsf{T}(X) : \mathsf{e}'_X$
— **[Dereliction]**: $\mathsf{d}_X : X \lhd \mathsf{T}(X) : \mathsf{d}'_X$
— **[Contraction]**: $\mathsf{c}_X : \mathsf{T}(X) \otimes \mathsf{T}(X) \lhd \mathsf{T}(X) : \mathsf{c}'_X$
— **[Weakening]**: $\mathsf{w}_X : I \lhd \mathsf{T}(X) : \mathsf{w}'_X$

and also that U comes equipped with the following retraction pairs:

$$\mathsf{j} : U \otimes U \lhd U : \mathsf{k} \qquad \mathsf{a} : I \lhd U : \mathsf{b} \qquad \mathsf{u} : \mathsf{T}(U) \lhd U : \mathsf{v}$$

As mentioned above, the trace symmetric monoidal functor of a GoI Situation will be used to interpret the exponential modality. So, as their names suggest, the four retraction pairs of monoidal natural transformations correspond to the four main rules of the exponential from Linear Logic. The reflexive object will be used to build an algebraic model of Linear Logic.

The two main examples of a GoI Situation are PINJ and HILB_2, and these can both be seen as capturing the main essence of Girard's original model of Geometry of Interaction.

Example 5.2. PINJ is a GoI Situation where the endofunctor is $\mathsf{T}(-) = \mathbb{N} \times -$ and the reflexive object is $\mathbb{N}$ [Abramsky et al.(2002), Proposition 5.2]. The necessary retraction pairs are given as follows:

(i) For $m, n \in \mathbb{N}$ let $\langle m, n \rangle = \frac{(m+n+1)(m+n)}{2} + n$ be Cantor's pairing function (which recall is a bijection). Then $\mathsf{e}_X : \mathbb{N} \times \mathbb{N} \times X \to \mathbb{N} \times X$ is defined as $\mathsf{e}_X\,(m, n, x) =$

$(\langle m,n\rangle, x)$, and $\mathsf{e}'_X : \mathbb{N}\times X \to \mathbb{N}\times\mathbb{N}\times X$ is defined as $\mathsf{e}'_X(n,x) = (n_1,n_2,x)$ where $\langle n_1,n_2\rangle = n$ (which is well-defined since n_1 and n_2 exists and are unique);

(ii) For some fixed $n_0 \in \mathbb{N}$, $\mathsf{d}_X : X \to \mathbb{N}\times X$ and $\mathsf{d}'_X : X \to \mathbb{N}\times X$ are defined as follows:

$$\mathsf{d}_X(x) = (n_0, x) \qquad\qquad \mathsf{d}_X(n,x) = \begin{cases} x & \text{if } n = n_0 \\ \text{undefined} & \text{o.w.} \end{cases}$$

(iii) $\mathsf{c}_X : (\mathbb{N}\times X)\sqcup(\mathbb{N}\times X) \to \mathbb{N}\times X$ and $\mathsf{c}'_X : \mathbb{N}\times X \to (\mathbb{N}\times X)\sqcup(\mathbb{N}\times X)$ are defined respectively as follows:

$$\mathsf{c}_X(i,n,x) = \begin{cases} (2n,x) & \text{if } i=1 \\ (2n+1,x) & \text{if } i=2 \end{cases} \qquad \mathsf{c}'_X(n,x) = \begin{cases} (1,\frac{n}{2},x) & \text{if } n \text{ is even} \\ (2,\frac{n-1}{2},x) & \text{if } n \text{ is odd} \end{cases}$$

(iv) $\mathsf{w}_X : \emptyset \to \mathbb{N}\times X$ and $\mathsf{w}'_X : \mathbb{N}\times X \to \emptyset$ are the empty partial functions which are nowhere defined.

(v) $\mathsf{j} : \mathbb{N}\sqcup\mathbb{N} \to \mathbb{N}$ and $\mathsf{k} : \mathbb{N} \to \mathbb{N}\sqcup\mathbb{N}$ are defined respectively as follows:

$$\mathsf{j}(i,n) = \begin{cases} 2n & \text{if } i=1 \\ 2n+1 & \text{if } i=2 \end{cases} \qquad \mathsf{k}(n) = \begin{cases} (1,\frac{n}{2}) & \text{if } n \text{ is even} \\ (2,\frac{n-1}{2}) & \text{if } n \text{ is odd} \end{cases}$$

Note that j and k are, up to isomorphism, $\mathsf{c}_{\{*\}}$ and $\mathsf{c}'_{\{*\}}$ for a singleton $\{*\}$.

(vi) $\mathsf{a} : \emptyset \to \mathbb{N}$ and $\mathsf{b} : \mathbb{N} \to \emptyset$ are the empty partial functions which are nowhere defined. Again, note that a and b are, up to isomorphism, $\mathsf{w}_{\{*\}}$ and $\mathsf{w}'_{\{*\}}$ for a singleton $\{*\}$.

(vii) $\mathsf{u} : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ is defined as $\mathsf{u}(m,n) = \langle m,n\rangle$ and $\mathsf{v} : \mathbb{N} \to \mathbb{N}\times\mathbb{N}$ is its inverse $\mathsf{v} = \mathsf{u}^{-1}$, so $\mathsf{v}(n) = (n_1,n_2)$ where $\langle n_1,n_2\rangle = n$. Similarly, note again that u and v are, up to isomorphism, $\mathsf{e}_{\{*\}}$ and $\mathsf{e}'_{\{*\}}$ for a singleton $\{*\}$.

Example 5.3. HILB_2 is a GoI Situation where the endofunctor is defined as $\mathsf{T}(\ell_2(-)) = \ell_2(\mathbb{N}\times -)$ and the reflexive object is $\ell_2(\mathbb{N})$ [Haghverdi and Scott(2006), Proposition 22]. Note that we can also express the endofunctor in terms of the tensor product of Hilbert spaces since $\ell_2(\mathbb{N}\times X) \cong \ell_2(\mathbb{N})\otimes\ell_2(X)$. The necessary retraction pairs are given by applying ℓ_2 to the GoI Situation structure from PINJ. So, up to natural isomorphism, we may write the retraction pairs as follows:

$$\ell_2(\mathsf{e}_X) : \ell_2(\mathbb{N})\otimes\ell_2(\mathbb{N})\otimes\ell_2(X) \triangleleft \ell_2(\mathbb{N})\otimes\ell_2(X) : \ell_2(\mathsf{e}'_X)$$
$$\ell_2(\mathsf{d}_X) : \ell_2(X) \triangleleft \ell_2(\mathbb{N})\otimes\ell_2(X) : \ell_2(\mathsf{d}'_X)$$
$$\ell_2(\mathsf{c}_X) : (\ell_2(\mathbb{N})\otimes\ell_2(X))\oplus(\ell_2(\mathbb{N})\otimes\ell_2(X)) \triangleleft \ell_2(\mathbb{N})\otimes\ell_2(X) : \ell_2(\mathsf{c}'_X)$$
$$\ell_2(\mathsf{w}_X) : \ell_2(\emptyset) \triangleleft \ell_2(\mathbb{N})\otimes\ell_2(X) : \ell_2(\mathsf{w}'_X)$$
$$\ell_2(\mathsf{j}) : \ell_2(\mathbb{N})\oplus\ell_2(\mathbb{N}) \triangleleft \ell_2(\mathbb{N}) : \ell_2(\mathsf{k})$$
$$\ell_2(\mathsf{a}) : \ell_2(\emptyset) \triangleleft \ell_2(\mathbb{N}) : \ell_2(\mathsf{b})$$
$$\ell_2(\mathsf{u}) : \ell_2(\mathbb{N})\otimes\ell_2(\mathbb{N}) \triangleleft \ell_2(\mathbb{N}) : \ell_2(\mathsf{v})$$

In particular, in [Haghverdi and Scott(2006), section 6], Scott and Haghverdi explain how Girard's C^*-algebra model of Geometry of Interaction is perfectly captured using this GoI Situation HILB_2.

Multiple other examples of GoI Situations can be found in [Abramsky et al.(2002), Section 5].

Usually, a categorical model of the multiplicative and exponential fragments of Linear Logic (MELL), sometimes called a *linear category*, is a symmetric monoidal closed category equipped with a symmetric monoidal comonad, which interprets the exponential modality, that comes equipped with extra natural transformations, which in-

terpret the rules for the exponential. For full details of the categorical semantics of Linear Logic, see [Blute and Scott(2004)]. However, the GOI-construction on a GoI Situation will not necessarily result in a linear category. Instead, Scott, Abramsky, and Haghverdi introduced the notion of a **weak linear category** [Abramsky et al.(2002), Definition 3.1], which is similar to a linear category but with weaker axioms that are nevertheless still sufficient to obtain a Geometry of Interaction interpretation of MELL. Briefly, a weak linear category is a symmetric monoidal closed category $\mathbb{C}$, equipped with a symmetric monoidal functor $! : \mathbb{C} \to \mathbb{C}$, and the following monoidal *pointwise* natural transformations:

$$\mathsf{dig}_X : !(X) \to !!(X) \quad \mathsf{der}_X : X \to !(X) \quad \mathsf{con}_X : !(X) \to !(X) \otimes !(X) \quad \mathsf{weak}_X : !(X) \to I$$

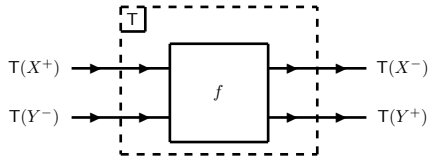
By pointwise natural, we mean natural only with respect to points, i.e., maps of type $I \to X$. See [Abramsky et al.(2002), Appendix II] and [Shirahata(2003), Section 5] for discussions on why pointwise naturality is sufficient in this context. Applying the GOI-construction to a GoI Situation results in a weak linear category.

PROPOSITION 5.4. *[Abramsky et al.(2002), Proposition 4.2.(i)] Let $(\mathbb{C}, \mathsf{T}, U)$ be a GoI Situation. Then $\mathcal{G}(\mathbb{C})$ is a weak linear category where:*

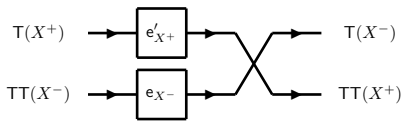
(i) *The endofunctor $! : \mathcal{G}(\mathbb{C}) \to \mathcal{G}(\mathbb{C})$ is defined on objects as:*

$$!(X^+, X^-) := (\mathsf{T}(X^+), \mathsf{T}(X^-))$$

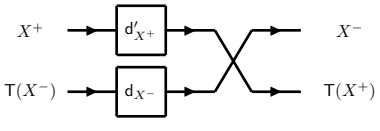
and for a map $f : (X^+, X^-) \to (Y^+, Y^-)$, $!(f) : !(X^+, X^-) \to !(Y^+, Y^-)$, which is a map of type $\mathsf{T}(X^+) \otimes \mathsf{T}(Y^-) \to \mathsf{T}(X^-) \otimes \mathsf{T}(Y^+)$ in $\mathbb{C}$, is defined as follows:

$!(f) :=$ 

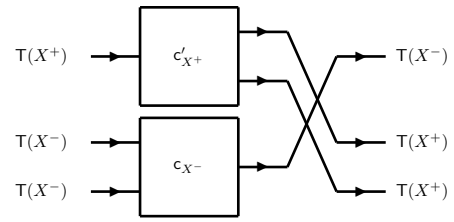
(ii) *The digging $\mathsf{dig}_{(X^+,X^-)} : !(X^+, X^-) \to !!(X^+, X^-)$, which is a map of type $\mathsf{T}(X^+) \otimes \mathsf{TT}(X^-) \to \mathsf{T}(X^-) \otimes \mathsf{TT}(X^+)$ in $\mathbb{C}$, is defined as:*

$\mathsf{dig}_{(X^+,X^-)} :=$ 

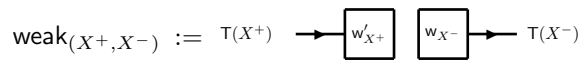
(iii) *The dereliction $\mathsf{der}_{(X^+,X^-)} : (X^+, X^-) \to !(X^+, X^-)$, which is a map of type $X^+ \otimes \mathsf{T}(X^-) \to X^- \otimes \mathsf{T}(X^+)$ in $\mathbb{C}$, is defined as:*

$\mathsf{der}_{(X^+,X^-)} :=$ 

(iv) *The contraction* $\mathsf{con}_{(X^+,X^-)} : !(X^+,X^-) \to !(X^+,X^-) \otimes !(X^+,X^-)$*, which is a map of type* $\mathsf{T}(X^+) \otimes \mathsf{T}(X^-) \otimes \mathsf{T}(X^-) \to \mathsf{T}(X^-) \otimes \mathsf{T}(X^+) \otimes \mathsf{T}(X^+)$ *in* $\mathbb{C}$*, is defined as:*

$\mathsf{con}_{(X^+,X^-)} :=$ 

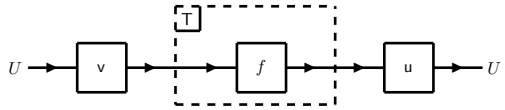
(v) *The weakening* $\mathsf{weak}_{(X^+,X^-)} : !(X^+,X^-) \to (I,I)$*, which is a map of type* $\mathsf{T}(X^+) \to \mathsf{T}(X^-)$ *in* $\mathbb{C}$*, is defined as:*

$\mathsf{weak}_{(X^+,X^-)} :=$ 

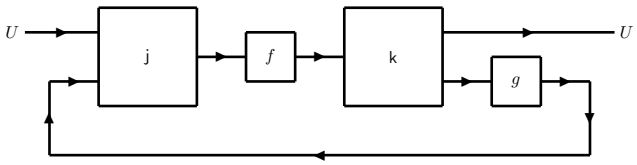
Scott, Abramsky, and Haghverdi also introduced the notion of a **linear combinatory algebra** [Abramsky et al.(2002), Definition 3.4] which briefly is a set A equipped with applicative structure $\cdot$, a unary operator $! : A \to A$, and eight distinguished elements of A which satisfy various axioms. The idea is that a linear combinatory algebra is an algebraic model of MELL. Moreover, in a weak linear category, if V is a special kind of *reflexive object* in the sense of [Abramsky et al.(2002), Definition 3.2], then the homset $\mathbb{C}(I,V)$ can be made into a linear combinatory algebra [Abramsky et al.(2002), Theorem 3.5]. We can apply this to GoI Situations. So for a GoI Situation $(\mathbb{C},\mathsf{T},U)$, (U,U) will be a reflexive object with the necessary properties in the weak linear category $\mathcal{G}(\mathbb{C})$. As such, we get that $\mathcal{G}(\mathbb{C})\,((I,I),(U,U))$ is a linear combinatory algebra. However, note that a map of type $(I,I) \to (U,U)$ in $\mathcal{G}(\mathbb{C})$ corresponds to a map of type $U \to U$ in $\mathbb{C}$. Therefore, we have that $\mathbb{C}(U,U)$ is in fact a linear combinatory algebra.

PROPOSITION 5.5. *[Abramsky et al.(2002), Proposition 4.2.(ii)] Let* $(\mathbb{C},\mathsf{T},U)$ *be a GoI Situation. Then* $\mathbb{C}(U,U)$ *is a linear combinatory algebra where:*

(i) *For a map* $f : U \to U$*,* $!f : U \to U$ *is defined as follows:*

$!f :=$ 

(ii) *For maps* $f : U \to U$ *and* $g : U \to U$*,* $f \cdot g : U \to U$ *is defined as follows:*

$f \cdot g :=$ 

and the eight distinguished elements are defined as in [Abramsky et al.(2002), Section 4].

Moreover, by standardization [Abramsky et al.(2002), Theorem 3.7], every linear combinatory algebra gives rise to a combinatory algebra in the usual sense [Abramsky et al.(2002), Definition 3.6]. So for a GoI Situation $(\mathbb{C},\mathsf{T},U)$, we also get that $\mathbb{C}(U,U)$ is a combinatory algebra.

6. GEOMETRY OF INTERACTION INTERPRETATIONS

In [Haghverdi and Scott(2006)], Scott and Haghverdi pushed this idea of using GoI Situations for the categorical semantics of the Geometry of Interaction interpretation of MELL further by considering *UDC-GoI Situations*. In particular, one of the main objectives of [Haghverdi and Scott(2006)] was to recapture precisely Girard's original model and matrix intuition for Geometry of Interaction, as well as the Execution formula, in the framework of a certain UDC-GoI Situation. For a review on Linear Logic, we refer the reader to see [Haghverdi and Scott(2010a), Section 3] and [Haghverdi(2000), Chapter 2].

By a **UDC-GoI Situation**, we mean a GoI Situation $(\mathbb{C}, \mathsf{T}, U)$ where $\mathbb{C}$ is a traced UDC and T is also an *additive* functor in the sense of preserving the partial infinitary sums. Then one can provide a Geometry of Interaction interpretation for MELL without units in a UDC-GoI Situation $(\mathbb{C}, \mathsf{T}, U)$. Formulas are interpreted as certain subsets of $\mathbb{C}(U, U)$, proofs are interpreted in $\mathbb{C}(U, U)$, and cut-elimination is interpreted by the Execution formula, which uses the trace operator. For the remainder of this section, we assume that we are working in a UDC-GoI Situation $(\mathbb{C}, \mathsf{T}, U)$.

Starting with the interpretation of formulas, we first need to define an orthogonality relation. Now a map $f : U \to U$ is *nilpotent* if there is some $k \in \mathbb{N}$ such that $f^k = 0_{U,U}$. Then we say that maps $f : U \to U$ and $g : U \to U$ are *orthogonal* if their composite $g; f$ (or equivalently $f; g$) is nilpotent. We write $f \perp g$ to say that f and g are orthogonal. Now given a subset $X \subseteq \mathbb{C}(U, U)$, define $X^\perp \subseteq \mathbb{C}(U, U)$ as the subset of all maps $U \to U$ which are orthogonal to those in X, that is:

$$X^\perp = \{f : U \to U |\ \forall g \in X.\ f \perp g\}$$

Then a **type** [Haghverdi and Scott(2006), Definition 14] is a subset $X \subseteq \mathbb{C}(U, U)$ such that $X^{\perp\perp} = X$. Note that types are always inhabited since the zero map is orthogonal to every map, so if X is a type then $0_{U,U} \in X$.

We will then use types to interpret MELL formulas in our UDC-GoI Situation. To do so, we will make use of our GoI Situation retraction pairs. So recall that we have maps $\mathsf{u} : \mathsf{T}(U) \to U$ and $\mathsf{v} : U \to \mathsf{T}(U)$, as well as $\mathsf{j} : U \otimes U \to U$ and $\mathsf{k} : U \to U \otimes U$, and also $\mathsf{u} : \mathsf{T}(U) \to U$ and $\mathsf{v} : U \to \mathsf{T}(U)$. Moreover, since we are also in a UDC, j and k also have matrix representations:

$$\mathsf{j} = \begin{bmatrix} \mathsf{j}_1 : U \to U \\ \mathsf{j}_2 : U \to U \end{bmatrix} \qquad\qquad \mathsf{k} = [\mathsf{k}_1 : U \to U \ \ \mathsf{k}_2 : U \to U]$$

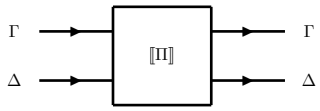
Now let A be a MELL formula. Then the **Geometry of Interaction interpretation** [Haghverdi and Scott(2006), Definition 15] of A, denoted as ΘA, is defined inductively as follows:

(i) If $A \equiv \alpha$ an atom, then $\Theta A = X$ where X is a type;
(ii) If $A \equiv \alpha^\perp$, where α is an atom, then $\Theta A = X^\perp$ where $\Theta\alpha = X$;
(iii) If $A \equiv B \otimes C$, then $\Theta A = Y^{\perp\perp}$ where $Y = \{\mathsf{k}_1; b; \mathsf{j}_1 + \mathsf{k}_2; c; \mathsf{j}_2 |\ b \in \Theta B \text{ and } c \in \Theta C\}$;
(iv) If $A \equiv B \parr C$, then $\Theta A = Y^\perp$ where $Y = \{\mathsf{k}_1; b; \mathsf{j}_1 + \mathsf{k}_2; c; \mathsf{j}_2 |\ b \in (\Theta B)^\perp \text{ and } c \in (\Theta C)^\perp\}$;
(v) If $A \equiv !B$, then $\Theta A = Y^{\perp\perp}$ where $Y = \{\mathsf{v}; \mathsf{T}(b); \mathsf{u} |\ b \in \Theta B \text{ and } c \in \Theta C\}$;
(vi) If $A \equiv ?B$, then $\Theta A = Y^\perp$ where $Y = \{\mathsf{v}; \mathsf{T}(b); \mathsf{u} |\ b \in (\Theta B)^\perp \text{ and } c \in \Theta C\}$.

It follows from the definition that $(\Theta A)^\perp = \Theta A^\perp$.

Now we turn towards understanding interpretations of proofs in our UDC-GoI Situation. For simplicity, let us denote the tensor product of n-copies of U as $U^n = U \otimes \ldots \otimes U$, where by convention $U^0 = I$. Now every MELL (without units) formula will be of the form $\vdash [\Delta], \Gamma$ where Γ is a sequence of formulas and Δ is a sequence of cut formu-

las that have been applied to the proof of $\vdash \Gamma$. Suppose that Γ consists of n formulas and Δ consists of $2m$ formulas. Therefore, a proof Π of $\vdash [\Delta], \Gamma$ will be interpreted by a map of type $[\![\Pi]\!] : U^{n+2m} \to U^{n+2m}$. Since we have that $U \otimes U \triangleleft U$, a map of type $U^{n+2m} \to U^{n+2m}$ corresponds to a map of type $U \to U$. As such, the interpretation of our proof is in $\mathbb{C}(U, U)$. However, following [Haghverdi and Scott(2006)], it will be much more convenient to work with interpreting proofs in $\mathbb{C}(U^{n+2m}, U^{n+2m})$. Also, we slightly abusing notation and denote Γ for U^n and Δ for U^{2m}. As such, we get that the interpretation of our proof is a map of type $[\![\Pi]\!] : \Gamma \otimes \Delta \to \Gamma \otimes \Delta$, which is drawn as on the left below, and since we are in a UDC, also has a matrix representation given by a 2×2-matrix as on the right below:



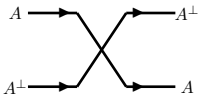
$$[\![\Pi]\!] = \begin{bmatrix} \Pi_{1,1} : \Gamma \to \Gamma & \Pi_{1,2} : \Gamma \to \Delta \\ \Pi_{2,1} : \Delta \to \Gamma & \Pi_{2,2} : \Delta \to \Delta \end{bmatrix}$$

So let Π be a proof of $\vdash [\Delta], \Gamma$. Then the **Geometry of Interaction interpretation** [Haghverdi and Scott(2006), Section 4] of Π, denoted as $[\![\Pi]\!] : \Gamma \otimes \Delta \to \Gamma \otimes \Delta$, is defined inductively on the length of Π as follows:

(i) If Π is an axiom:

$$\frac{}{\vdash A, A^{\perp}}$$

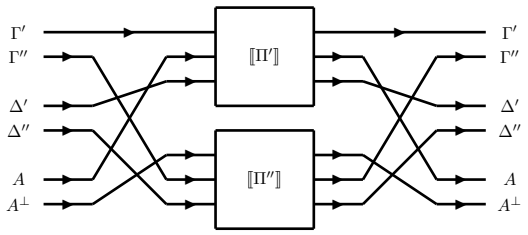
then its interpretation $[\![\Pi]\!]$ is the symmetry map:



(ii) If Π is obtained using the cut rule:

$$\frac{\begin{array}{cc} \Pi' & \Pi'' \\ \vdots & \vdots \\ \vdash [\Delta'], \Gamma', A & \vdash [\Delta''], A^{\perp}, \Gamma'' \end{array}}{\vdash \left[\Delta, \Delta', A, A^{\perp}\right], \Gamma, \Gamma'} \text{ (CUT)}$$

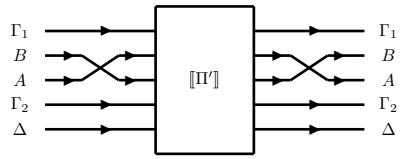
then its interpretation $[\![\Pi]\!]$ is drawn as follows:



(iii) If Π is obtained using the exchange rule:

$$\frac{\begin{array}{c} \Pi' \\ \vdots \\ \vdash [\Delta], \Gamma_1, A, B, \Gamma_2 \end{array}}{\vdash [\Delta], \Gamma_1, B, A, \Gamma_2} \text{ (EXCH.)}$$

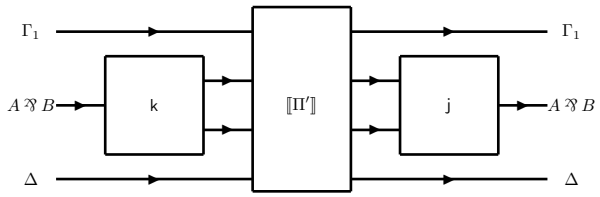
then its interpretation $[\![\Pi]\!]$ is drawn as follows:



(iv) If Π is obtained using the $\parr$ rule:

$$\dfrac{\begin{array}{c}\Pi' \\ \vdots \\ \vdash [\Delta], \Gamma, A, B\end{array}}{\vdash [\Delta], \Gamma, A \parr B}\ (\parr)$$

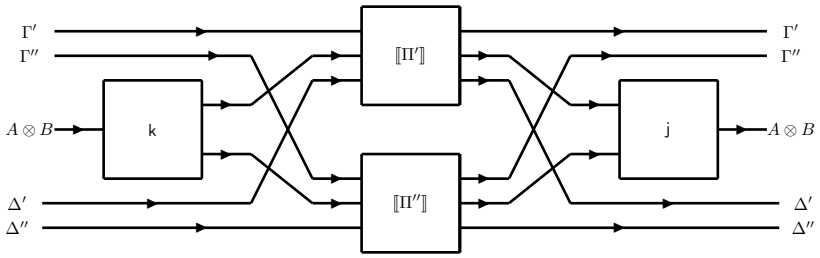
then its interpretation $[\![\Pi]\!]$ is drawn as follows:



(v) If Π is obtained using the $\otimes$ rule:

$$\dfrac{\begin{array}{c}\Pi' \\ \vdots \\ \vdash [\Delta'], \Gamma', A\end{array} \qquad \begin{array}{c}\Pi'' \\ \vdots \\ \vdash [\Delta''], \Gamma'', B\end{array}}{\vdash [\Delta', \Delta''], \Gamma', \Gamma'', A \otimes B}\ (\otimes)$$

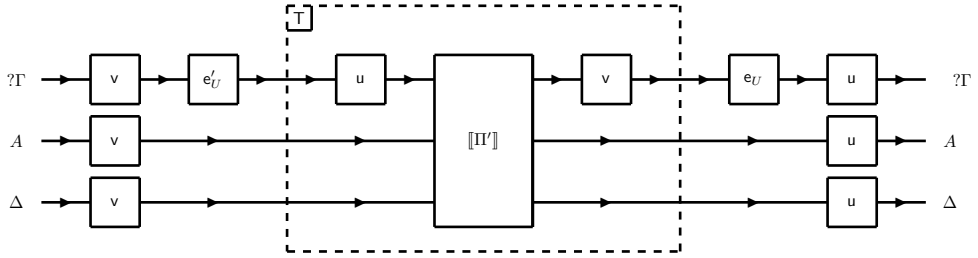
then its interpretation $[\![\Pi]\!]$ is drawn as follows:



(vi) If Π is obtained using the ! rule:

$$\dfrac{\begin{array}{c}\Pi' \\ \vdots \\ \vdash [\Delta], ?\Gamma, A\end{array}}{\vdash [\Delta], ?\Gamma, !A}\ (!)$$

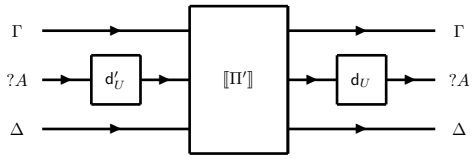
then its interpretation $[\![\Pi]\!]$ is drawn as follows:



(vii) If Π is obtained using the dereliction rule:

$$\dfrac{\begin{array}{c}\Pi'\\ \vdots\\ \vdash [\Delta], \Gamma, A\end{array}}{\vdash [\Delta], \Gamma, ?A}\ \textbf{(DER.)}$$

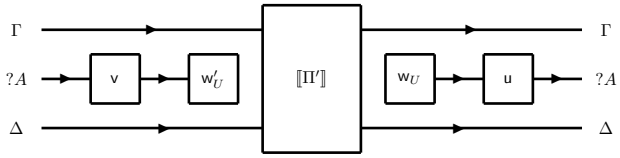
then its interpretation $[\![\Pi]\!]$ is drawn as follows:



(viii) If Π is obtained using the weakening rule:

$$\dfrac{\begin{array}{c}\Pi'\\ \vdots\\ \vdash [\Delta], \Gamma\end{array}}{\vdash [\Delta], \Gamma, ?A}\ \textbf{(WEAK.)}$$

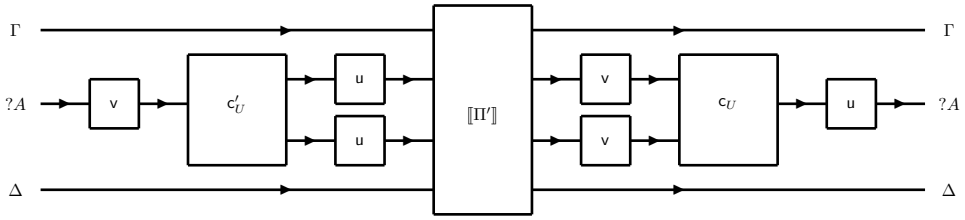
then its interpretation $[\![\Pi]\!]$ is drawn as follows:



(ix) If Π is obtained using the contraction rule:

$$\dfrac{\begin{array}{c}\Pi'\\ \vdots\\ \vdash [\Delta], \Gamma, ?A, ?A\end{array}}{\vdash [\Delta], \Gamma, ?A}\ \textbf{(CON.)}$$

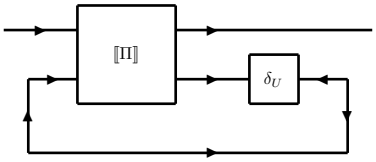
then its interpretation $[\![\Pi]\!]$ is drawn as follows:



Finally, the last thing that remains to be interpreted is cut-elimination. The Geometry of Interaction interpretation of cut-elimination is given by the *Execution formula*, which, as mentioned above, is defined using the trace operator. So consider a proof Π of $\vdash [\Delta], \Gamma$, so with interpretation $[\![\Pi]\!] : \Gamma \otimes \Delta \to \Gamma \otimes \Delta$. While in theory we can already take the trace of $[\![\Pi]\!]$ and trace out Δ, in order to get the correct formula, we will need some extra permutations. So let $\delta_U : U^{2m} \to U^{2m}$ be the tensor product of m-copies of the symmetry map, so $\delta_U = \sigma_{U,U} \otimes \ldots \otimes \sigma_{U,U}$. We may write this permutation as a map of type $\delta_U : \Delta \to \Delta$. Then δ_U represents the cuts in the proof $\vdash \Gamma$, or in other words, δ_U models Δ. This permutation is necessary since it rearranges the cut-formulas so that when we close loops (i.e. take the trace), every cut-formula gets connected to its dual formula.

Then for a proof Π of $\vdash [\Delta], \Gamma$, the **Execution formula** [Haghverdi and Scott(2006), Section 4.1] is defined as follows:

$\mathsf{EX}([\![\Pi]\!], \delta_U) :=$



Since we are in UDC, we can also use the standard trace formula to express the Execution formula as follows:

$$\mathsf{EX}([\![\Pi]\!], \delta_U) = \Pi_{1,1} + \sum_{n \in \mathbb{N}} \Pi_{1,2}; \delta; (\Pi_{2,2}; \delta)^n; \Pi_{2,1}$$

This standard trace formula version of the Execution formula allowed Scott and Haghverdi to show that the Execution formula in HILB_2 is the same as Girard's original execution formula for the operator algebra model [Haghverdi and Scott(2006), Proposition 23].

The main result of [Haghverdi and Scott(2006)] is, of course, the soundness of this Geometry of Interaction interpretation of MELL in a UDC-GoI Situation. In this case, soundness can be described as saying that if a proof is reduced via cut-elimination to its cut-free form, then applying the Execution formula to the starting proof gives a finite sum and results in the interpretation of the cut-free form. In other words, if we run the result of the Execution formula, then it terminates after finitely many steps (so a finite sum) and yields a datum (a cut-free proof).

THEOREM 6.1. *[Haghverdi and Scott(2006), Theorem 21] Let* Π *be a proof of* $\vdash [\Delta], \Gamma$. *Then:*

(i) $\mathsf{EX}([\![\Pi]\!], \delta_U)$ *is a finite sum.*
(ii) *If* Π *reduces to* Π' *by any sequence of cut-elimination steps and* Γ *does not contain any fomrulas of the form* $?A$*, then* $\mathsf{EX}([\![\Pi]\!], \delta_U) = \mathsf{EX}([\![\Pi']\!], \delta'_U)$. *So* $\mathsf{EX}([\![\Pi]\!], \delta_U)$ *is an invariant of reduction.*

(iii) *In particular if* Π' *is any cut-free proof obtained from* Π *by cut-elimination, then we have that* $\mathsf{EX}(\llbracket \Pi \rrbracket, \delta_U) = \llbracket \Pi' \rrbracket$.

In [Haghverdi and Scott(2005)], Scott and Haghverdi also explain how to construct a denotational model for MELL (without units), and thus provide a more direct relation between the Geometry of Interaction semantics to the denotational semantics in the case of MELL. Moreover, they also construct star-autonomous categories from a UDC-GoI Situation via an orthogonality construction [Haghverdi and Scott(2005), Section 5] or via a double glueing construction [Haghverdi and Scott(2005), Section 6.1].

7. FURTHER READING

Scott and his coauthors continued to work on traced monoidal categories and develop other important concepts in the area. In [Haghverdi and Scott(2010b)], Scott and Haghverdi developed a typed version of Geometry of Interaction and its categorical semantics. To do so, they introduced the concept of *partially traced categories*, which are a generalization of traced monoidal categories where the trace operator is now only partially defined. The theory of partially traced categories was developed further by Scott, Malherbe, and Selinger in [Malherbe et al.(2012)]. In [Hamano and Scott(2018)], Scott and Hamano also developed a Geometry of Interaction for *Polarized* Linear Logic, as well as a polarized version of GoI Situations.

REFERENCES

S. Abramsky. 1996. Retracing some paths in process algebra. In *International Conference on Concurrency Theory*. Springer, Berlin, Heidelberg, 1–17.

S. Abramsky and B. Coecke. 2004. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.* IEEE, Turku, Finland, 415–425. DOI:https://doi.org/10.1109/LICS.2004.1319636

S. Abramsky and B. Coecke. 2005. Abstract physical traces. *Theory and Applications of Categories* 14, 6 (2005), 111–124.

S. Abramsky, E. Haghverdi, and P. Scott. 2002. Geometry of Interaction and Linear Combinatory Algebras. *Mathematical Structures in Computer Science* 12, 5 (2002), 625–665. DOI:https://doi.org/10.1017/S0960129502003730

S. Abramsky and R. Jagadeesan. 1992a. Games and full completeness for multiplicative linear logic. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, Berlin, Heidelberg, 291–301.

S. Abramsky and R. Jagadeesan. 1992b. New foundations for the geometry of interaction. In *[1992] Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science*. IEEE, Turku, Finland, 211–222. DOI:https://doi.org/10.1109/LICS.1992.185534

S. Abramsky and R. Jagadeesan. 1994a. Games and full completeness for multiplicative linear logic. *The Journal of Symbolic Logic* 59, 2 (1994), 543–574.

S. Abramsky and R. Jagadeesan. 1994b. New foundations for the geometry of interaction. *Inf. Comput.* 111, 1 (may 1994), 53–119. DOI:https://doi.org/10.1006/inco.1994.1041

M. Barr. 1992. Algebraically compact functors. *Journal of Pure and Applied Algebra* 82, 3 (1992), 211–231.

R. Blute and P. Scott. 2004. Category theory for linear logicians. *Linear logic in computer science* 316 (2004), 3–64.

J.-Y. Girard. 1987. Linear logic. *Theoretical computer science* 50, 1 (1987), 1–101.

J.-Y. Girard. 1988. Geometry of interaction 2: Deadlock-free algorithms. In *International Conference on Computer Logic*. Springer, Berlin, Heidelberg, 76–93.

J.-Y. Girard. 1989a. Geometry of interaction 1: Interpretation of System F. *Studies in Logic and the Foundations of Mathematics* 127 (1989), 221–260.

J.-Y. Girard. 1989b. Towards a geometry of interaction. *Contemp. Math.* 92, 69-108 (1989), 6.

J.-Y. Girard. 1995. Geometry of interaction III: accommodating the additives. *London Mathematical Society Lecture Note Series* , (1995), 329–389.

J.-Y. Girard. 2006. Le point aveugle. *Cours de logique* 1 (2006).

J.-Y. Girard. 2011. Geometry of interaction V: logic in the hyperfinite factor. *Theoretical Computer Science* 412, 20 (2011), 1860–1883.

E. Haghverdi. 2000. *A categorical approach to linear logic, geometry of proofs and full completeness*. Ph. D. Dissertation. University of Ottawa (Canada).

E. Haghverdi and P. Scott. 2005. From geometry of interaction to denotational semantics. *Electronic Notes in Theoretical Computer Science* 122 (2005), 67–87.

E. Haghverdi and P. Scott. 2006. A Categorical Model for the Geometry of Interaction. *Theoretical Computer Science* 350, 2-3 (2006), 252–274.

E. Haghverdi and P. Scott. 2010a. Geometry of Interaction and the Dynamics of Proof Reduction: a tutorial. *New Structures for Physics* 813 (2010), 357–417.

E. Haghverdi and P. Scott. 2010b. Towards a typed geometry of interaction. *Mathematical Structures in Computer Science* 20, 3 (2010), 473–521.

T. Hajgató and M. Hasegawa. 2013. Traced star-autonomous categories are compact closed. *Theory and Applications of Categories* 28, 7 (2013), 206–212.

M. Hamano and P. Scott. 2018. On geometry of interaction for polarized linear logic. *Mathematical Structures in Computer Science* 28, 10 (2018), 1639–1694.

M. Hasegawa. 1997. Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *Typed Lambda Calculi and Applications*. Springer, Berlin, Heidelberg, 196–213. DOI:https://doi.org/10.1007/3-540-62688-3_37

M. Hasegawa. 2009. On traced monoidal closed categories. *Mathematical Structures in Computer Science* 19, 2 (2009), 217–244. DOI:https://doi.org/10.1017/S0960129508007184

M. Hasegawa and J.-S. P. Lemay. 2023. Traced Monads and Hopf Monads. *Compositionality* 5 (Oct. 2023), 1–34. Issue 10. DOI:https://doi.org/10.32408/compositionality-5-10

C. Heunen. 2013. On the functor ℓ^2. In *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky: Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*. Springer, Berlin, Heidelberg, 107–121.

C. Heunen and J. Vicary. 2019. *Categories for Quantum Theory: an introduction*. Oxford University Press, Oxford, UK. DOI:https://doi.org/10.1093/oso/9780198739623.001.0001

A. Joyal, R. Street, and D. Verity. 1996. Traced Monoidal Categories. *Mathematical Proceedings of the Cambridge Philosophical Society* 119, 3 (1996), 447–468. DOI:https://doi.org/10.1017/S0305004100074338

J. Lambek and P. Scott. 1988. *Introduction to higher-order categorical logic*. Vol. 7. Cambridge University Press, Cambridge, UK.

O. Malherbe, P. Scott, and P. Selinger. 2012. Partially traced categories. *Journal of Pure and Applied Algebra* 216, 12 (2012), 2563–2585.

E. G. Manes and M. A. Arbib. 2012. *Algebraic approaches to program semantics*. Springer Science & Business Media, Berlin, Heidelberg.

P. Selinger. 2010. A survey of graphical languages for monoidal categories. In *New Structures for Physics*. Springer, Berlin, Heidelberg, 289–355. DOI:https://doi.org/10.1007/978-3-642-12821-9_4

Ma. Shirahata. 2003. Geometry of Interaction explained. *Surikaisekikenkyusho Kokyuroku* 1318 (2003), 160–187. https://repository.kulib.kyoto-u.ac.jp/dspace/handle/2433/43041 Algebra, Logic and Geometry in Informatics (Kyoto, 2002).

COM

Complexity Column

ANDREI BULATOV, Simon Fraser University
abulatov@cs.sfu.ca



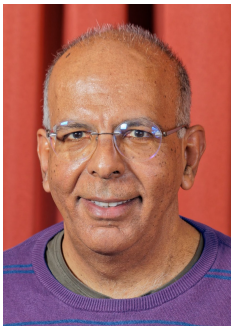
Descriptive complexity theory initiated by celebrated Fagin's theorem aims to characterize complexity classes by the kind of logic necessary to express problems in the class. For many complexity classes such a characterization has been found; this includes the class NP that, according to Fagin's theorem, can be characterized by the monadic second order logic. Some classes, however, eludes such a characterization, and this famously includes the class P. Due to the importance of the class P, finding a logic for P is one of the central open problems in descriptive complexity.

This quarter's *Complexity* column focuses on recent progress in the search for a logic for P. After classic early results of capturing the class P on ordered structures by FPC, the counting logic with a fixed point operator, the majority of effort was directed at expanding FPC in various ways. These candidate logics include extensions of FPC by various linear algebraic operators and several types of symmetry breaking operators. Another approach is to restrict the kind of polynomial time algorithms allowed for solving problems in P to those that do not make any arbitrary choices. This approach is known as choiceless polynomial time and has received a substantial amount of attention recently. In spite of all the developments, finding a logic for the class P remains beyond the reach for now.

This survey reviews the established and new approaches to the problem, and reports on the state of the art that covers many of the recent advances and identifies a number of open problems. The first author of the survey, Anuj Dawar, is one of the leading experts in the area, who obtained many of the results covered in the survey. The second author, Benedikt Pago, is a young researcher in the beginning of his career, who nevertheless has made several strong contributions in the area in the last few years. I am grateful to Anuj and Benedikt for the following survey on a logic for P.

A Logic for P: Are we nearly there yet?

Anuj Dawar, Department of Computer Science and Technology,
University of Cambridge



Benedikt Pago, Department of Computer Science and Technology,
University of Cambridge



The question of whether there is a logic for P has been the driving concern of research in descriptive complexity for close to four decades. While a resolution of the question is not in sight, the search has revealed a great deal of detail about the structure of the complexity class. A number of recent results have pushed the boundaries in several directions and most importantly developed significant new and sophisticated methods for proving lower bounds. This article surveys some of this recent work and maps out the landscape of where we now find ourselves.

1. INTRODUCTION

Whether or not there is a logic for P is the classic problem of descriptive complexity theory. It has been an open problem for over four decades (since its first formulation by Chandra and Harel [1980]) now and much has been written about it, including surveys and textbook presentations (see, for instance [Ebbinghaus and Flum 1999, Chapter 11]. The problem remains a focus of active research and there have been significant recent developments related to it. These may not be well known outside a specialist group of researchers and one aim of the present article is to bring these to a wider audience. To place the new results in their context, we begin with a wider perspective on the question.

The central problem of descriptive complexity has sometimes been described as a *quest* for a logic for P [Grohe 2008]. Yet, far more interesting than the end goal of this quest is the light it casts on the fine structure of the complexity class P, and indeed other complexity classes. When we talk of a logic for P in the context of descriptive complexity, we specifically mean polynomial-time decidable properties of absract finite relational structures, such as graphs. And we are seeking to characterise these by logics that work at the abstract level, not on encodings of the structures as strings. In particular the logics must respect the symmetries that are inherent in abstract structures. For structures that lack such symmetries, such as numbers or strings, we do have suitable logics. For example, there are well-known characterisations of the polynomial-time computable functions on numbers [Cobham 1965].

A logic for P would then reveal a structured way in which algorithms for all polynomial-time decidable properties can be built up from some collection of operations, each of which themselves inherently respect the symmetries of the structures on which they operate. To discover such a logic would be to understand the structure of the complexity class. But, there is another way in which the question of the existence of the logic is tied to the fine structure of the class, which we expand on below.

Gurevich [1988] gave a precise mathematical formulation of the question of what it would mean to have a logic for P. It is precise in the sense that it admits of a clear negative answer. It lays out specific minimal criteria that a logic must satisfy and Gurevich conjectured that there is no such logic. To be precise, a logic would give us, for each finite relational vocabulary σ a decidable set $L[\sigma]$ of sentences, along with a decidable satisfaction relation $\models$, relating finite σ-structures $\mathfrak{A}$ with sentences of $L[\sigma]$, with the property that for any $\phi \in L[\sigma]$, the collection of σ-structures $\text{Mod}(\phi) = \{\mathfrak{A} \mid \mathfrak{A} \models \phi\}$ is closed under isomorphisms. To make this a *logic for* P, we require two further conditions. The first is that for every isomorphism-closed class of finite σ-structures $\mathcal{C}$ that is decidable in polynomial time there is a sentence $\phi \in L[\sigma]$ with $\mathcal{C} = \text{Mod}(\phi)$. The second is what we call the *compiler* condition: there is a computable map taking each $\phi \in L[\sigma]$ to a pair (M, p) where M is a Turing machine and p a polynomial such that if M is run with a clock p on an input coding a σ-structure $\mathfrak{A}$, it accepts if, and only if, $\mathfrak{A} \models \phi$.

What does this have to do with the fine structure of P? It is known [Dawar 1995] that there is a logic for P in the above sense if, and only if, there is a complete problem in P under *first-order reductions* (defined below in Section 2). These are computationally weak and, most importantly, *symmetry-preserving* reductions. Thus, a significant part of the search for a logic for P is about understanding the structure of the class under symmetry-preserving reductions. In particular, many results about the expressive power of logics with algebraic operators can be recast as investigating the reducibilities among algebraic problems inside P under such reductions. It is also worth noting that we know that NP does have complete problems under first-order reductions. Hence, a proof that P does not would show a definitive structural difference between the two complexity classes.

The question of whether there is a logic for P then confronts us with two possible worlds, and we do not know which of these we live in. One possibility is that there is a logic for P, and we can design a high-level language from symmetry-respecting constructs in which we can express all efficiently solvable problems. Moreover, all these problems can be solved by minor variations of a single algorithm for solving a central, complete, problem. The second possibility is that there is no logic for P. In particular this implies that there is no recursive enumeration of the problems in this class. There is therefore an inexhaustible supply of efficient algorithmic techniques to be discovered. Moreover in this world, P is different from NP.

Perhaps the most important aspect of the search for a logic for P is that we have developed methods for showing that certain "candidate logics" do not, in fact, capture all of P. This requires proving unconditional lower bounds against ever larger natural fragments of P. Seen in this light, a result showing that some such candidate logic cannot express some problem in P should not be seen as a failure of the logic but as a triumph of our ability to prove impossibility results. In this survey, we will look at some such recent results.

Fixed-point Logic with Counting. The first "candidate logic for P" and still by far the most studied one is FPC, *fixed-point logic with counting*. It did not hold the status of candidate logic for very long, having been first suggested in an article by Immerman [1986] and shown not to capture P a few years later in the work of Cai et al. [1992].

Still, having been killed off as a candidate for P, FPC has had a long and productive after-life. It continues to be studied as a benchmark for natural, symmetric, fragments of P. It was the subject of its own survey in this column [Dawar 2015]. We give a full definition of this logic in Section 3. For now it is worth noting that most straightforward and many not so straightforward polynomial-time algorithms can be translated into FPC formulas. Thus, showing that a problem is not FPC definable does rule out a slew of natural algorithmic methods for solving it.

The construction by Cai et al. [1992] that eliminated FPC as a logic for P has itself had a strong and continuing relevance in the over three decades since. Variations of the construction, commonly referred to as the CFI construction, have been used to show lower bounds for stronger logics than FPC, but it has also been used as a technique for showing lower bounds in many other areas, from the hardness of approximation [Atserias and Dawar 2019] to arithmetic circuits [Dawar and Wilsenach 2020] to pick just two. We review the CFI construction in the form we use in this survey in Section 3.

Besides separating FPC from P, the CFI construction also demonstrates the limits of the Weisfeiler-Leman method for deciding graph isomorphism. The Weisfeiler-Leman algorithms are a family of algorithms, one for each natural number k, which define an equivalence relation on graphs approxmating isomorphism, which we call k-WL equivalence. Until the work of [Cai et al. 1992] it was an open question whether there was a constant k for which the k-WL equivalence is exactly the relation of isomorphism. The connection with FPC stems from the fact that any property definable in this logic is invariant under the k-WL equivalence for some constant k. Indeed, the problems of finding a logic for P and finding efficient algorithms for graph isomorphism are closely intertwined. There are many classes of structures $\mathcal{C}$ defined by structural restriction on the class of finite structures, for which it has been shown that there is a logic (often FPC itself) which captures P on $\mathcal{C}$. In particular this is known for any class of graphs which excludes some graph minor [Grohe 2017]. Proofs of such capturing results proceed by showing that a canonical labelling algorithm can be implemented for structures in $\mathcal{C}$ and so, in particular, that the isomorphism problem for structures in $\mathcal{C}$ is in P. Indeed, we know of no other method of showing that P can be captured on a class $\mathcal{C}$.

Another aspect of the relationship between logics for P and isomorphism problems is that methods for proving lower bounds for candidate logics for P usually proceed by constructing hard examples for ever more powerful approximations of isomorphism. A particularly instructive example is the invertible-map equivalences which we discuss further in Section 4.

One way of understanding FPC is as the maximal natural fragment of P that can be built up from fully symmetric operations. This follows from the characterisation of FPC as those properties decided by polytime-uniform symmetric circuits with fully symmetric gates (see [Dawar and Wilsenach 2021] for the full definitions). Then the search for logics beyond FPC that stay within P can be seen as searching for ways to introduce symmetry-breaking operations in a controlled and limited way into the logic, that remains safe in the sense that it still does not allow us to define any properties that are not isomorphism invariant. This search leads us in three different directions that we review in this paper.

Linear Algebraic Logics. The first direction is to extend FPC explicitly with operators for solving certain linear-algebraic problems. The starting point for this is the insight that the CFI construction reduces to the problem of solving systems of linear equations over the two-element field and can therefore be solved by a Gaussian elimination algorithm [Atserias et al. 2009]. This algorithm crucially involves a symmetry-breaking operation, in the form of choosing a pivot. One solution is to package the algorithm into

a black box whose output is guaranteed to be isomorphism invariant. One such proposal leads us to fixed-point logic with rank FPR. This extends FPC with operators that yield the rank of a matrix (over a suitable finite field) as atomic operations. In its more powerful variation [Grädel and Pakusa 2019] these operators allow the characteristic of the field as a variable parameter. It is this stronger form that was shown by Lichter to be strictly weaker than P in his breakthrough result [Lichter 2023].

A number of other extensions of FPC with algebraic operations have been studied. They can be seen as including black-box operations for a variety of polynomial-time algebraic problems, particularly solvability problems over finite fields, rings or groups. Much of this line of work is best understood as investigating what problems of this form are inter-reducible to each other using symmetry-preserving reductions, i.e. reductions definable in FPC [Dawar et al. 2013].

It is also instructive to look at an extension of FPC simultaneously with all *linear-algebraic operators*. Such a logic LA^ω is defined in [Dawar et al. 2019], including all possible operators defined over finite fields. The logic is not confined to P but it admits of powerful methods for proving inexpressibility. These are based on *invertible map games*, which provide approximations of isomorphism which are still polynomial-time decidable but more discriminating than the Weisfeiler-Leman equivalences. These equivalences are crucial to Lichter's result on the expressive power of FPR. The result can be further extended to show that the infinitary linear-algebraic logic LA^ω cannot express all properties in P. Indeed, the proof shows something interesting about the mutual reducibility of various solvability problems. The problem of solving systems of linear equations $(\mathrm{mod}\ p)$ for a prime number p cannot be reduced by a first-order (or indeed FPC) reduction to the problem of solving such equations $(\mathrm{mod}\ q)$ for any prime $q \neq p$. Moreover, the problem of solving systems of linear equations $(\mathrm{mod}\ 2^m)$ for arbitrary integers $m \geq 2$ cannot be reduced to solving such equations $(\mathrm{mod}\ p)$ for prime numbers p.

We take a closer look at the line of work on linear-algebraic logics in Section 4.

Witnessed Symmetric Choice. A second direction that has been investigated is to explicitly include in the logic a symmetry-breaking or *choice* operator. The study of choice operations has a long history in logic, going back to Hilbert (see [Leisenring 1969]), the central idea being that when we have a formula $\phi(x)$, we can define a term $\varepsilon x\phi$ which is interpreted as some, arbitrarily chosen, element from among the set of elements that satisfy ϕ, provided this set is non-empty. Having an unrestricted choice operator in the logic allows one to define formulas whose semantics are not invariant under isomorphisms. In the context of finite model theory, these choice operators were first considered by Arvind and Biswas [1987] who defined a fixed-point logic with choice and showed, essentially, that this is very much like having a built-in order. A clear exposition of variations on such choice operators is given by Blass and Gurevich [2000].

To recover a meaningful isomorphism-invariant semantics for the logic, we need to constrain choice operators in some way. One suggestion is that we only allow the choice operator to yield a well-defined element of a structure $\mathfrak{A}$ when the collection of elements satisfying $\phi(x)$ forms an *orbit* of the automorphism group of $\mathfrak{A}$. This is what is called *symmetric choice* and it does give a logic whose semantics is isomorphism-invariant. However, adding such an operator to fixed-point logic does not necessarily give a logic contained in P as it is not clear how to evaluate a formula in polynomial time. The semantics includes an isomorphism test and we do not know of polynomial time algorithms for testing isomorphism. Gire and Hoang [1998] suggested a way of adding symmetric choice while guaranteeing a polynomial-time semantics, and that is to put the onus on the formula to specify the automorphisms that witness the fact that $\phi(x)$ defines an orbit. Then, in evaluating the formula we only have to check the fact

that they are automorphisms and this can be done in polynomial time. Defining the semantics of such *witnessed symmetric choice* operators is subtle and we review it in some detail in Section 5. In recent work, Lichter and Schweitzer [2022] have developed significant new techniques for the analysis of these operators.

Choiceless Polynomial Time. A third direction, orthogonal to the above is the study of choiceless polynomial time, initiated by Blass et al. [1999]. Rather than seeking to add symmetry-breaking in a safe way to logic, they ask the question of what is the maximal fragment of P that can be computed by algorithms that do not use any kind of choice. For this purpose they define a machine model which works directly on relational structures. They give it rich expressive power by allowing the machines to construct higher-order data structures on top of the relations and a high degree of parallelism. The resulting logic still cannot express simple counting properties so, as with FPC, we add an explicit ability to count. This formalism, when restricted to polynomially bounded resources is called *choiceless polynomial-time with counting* and is what we denote CPT. A formal definition is given in Section 6.

The expressive power of CPT includes that of FPC and is easily shown to properly include it. This is because CPT is able to take advantage of padding. In short, if a formula of FPC cannot distinguish two structures $\mathfrak{A}$ and $\mathfrak{B}$, it also cannot distinguish structures which are obtained by taking the disjoint union of $\mathfrak{A}$ and $\mathfrak{B}$ respectively with some other large structure $\mathfrak{C}$. On the other hand, since CPT is limited by polynomial resources in the size of its input, if $\mathfrak{C}$ is large enough, it can exploit these additional resources to distinguish $\mathfrak{A} + \mathfrak{C}$ from $\mathfrak{B} + \mathfrak{C}$. Thus, a suitably padded version of the CFI construction can be used to separate FPC from CPT.

It remains an open question whether every problem in P is in CPT. Blass et al. [2002] list four candidate problems for separating CPT from P. They were: an unpadded version of the CFI problem; the isomorphism problem for multipedes; deciding whether a graph contains a perfect matching; and computing the determinant of a matrix over a finite field. In subsequent work, all of these have been shown to be in fact definable in CPT. The first in [Dawar et al. 2008], the second in [Abu Zaid et al. 2014]l the third in [Anderson et al. 2015] and the fourth in a construction due to Rossman (see [Blass and Gurevich 2005]). Interestingly, Blass [2016] views the fact that the solution for each of these requires fundamentally different techniques as evidence that CPT does not, in fact, capture all polynomial-time computation. Perhaps the most interesting of these is the CPT algorithm for deciding the CFI problem. This is an algorithm that makes essential use of higher-order structures that are highly symmetric in order to get around the limitations of FPC. This algorithm has been the object of much further investigation to determine the limits of its power, and we survey some of this work in Section 6. It does show that the use of symmetric higher-order objects can compensate to some extent for the lack of symmetry-breaking operations but to what extent remains unclear. In particular, we do not know if in CPT we can compute the rank of a matrix over a finite field, and hence whether CPT subsumes the power of FPR. Similarly, we do not know if CPT can be used to simulate a witnessed symmetric choice operator. Thus, it remains possible that we can obtain more powerful logics by combining CPT with the approaches outlined above. In Section 6.2 we look at some recent work that seeks to combine CPT with a witnessed symmetric choice operation.

2. PRELIMINARIES

We assume the reader is familiar with standard notions from first-order logic and its semantics on finite relational structures. For an exposition, one may consult [Ebbinghaus and Flum 1999]. We are mainly concerned with extensions of first-order logic. We deal with finite relational signatures σ (also sometimes called vocabularies) and finite

σ-structures (denoted by Fraktur letters, $\mathfrak{A}$, $\mathfrak{B}$, etc.), where a structure $\mathfrak{A}$ interprets the symbols of σ as relations over a finite universe A. For each relation symbol R in σ we write $\mathrm{ar}(R)$ for its arity. We do not work with a general definition of what constitutes a logic, but for each of the logics we consider, there is a syntactic notion of formulas with free and bound variables. A formula $\phi(\vec{x})$ of a logic L in a signature σ, with free variables among $\vec{x}$ defines, in the usual way, a mapping taking a σ-structure $\mathfrak{A}$ to a relation $\phi^{\mathfrak{A}} \subseteq A^{\vec{x}}$. We call this the *query* defined by ϕ. When ϕ is a *sentence*, i.e. it has no free variables, it defines a class of structures also referred to as a *Boolean query*. For any logic L, we write L both to mean the collection of formulas that make up the logic and the collection of decision problems on finite structures that are defined by sentences of the logic. A key property of queries is that they are *isomorphism-invariant*. Recall that an *isomorphism* between two σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ is a bijection $\eta : A \to B$ such that for each $R \in \sigma$ and tuple $\vec{a}$ of elements of A of the right length, $R^{\mathfrak{A}}(\vec{a})$ if, and only if, $R^{\mathfrak{B}}(\eta(\vec{b}))$. A formula ϕ defines an isomorphism-invariant query in the sense that for any isomorphism $\eta : \mathfrak{A} \to \mathfrak{B}$, we have $\vec{a} \in \phi^{\mathfrak{A}}$ if, and only if, $\eta(\vec{a}) \in \phi^{\mathfrak{B}}$.

The basic extension of first-order logic that is the starting point of our study is *fixed-point logic* FP. This can be defined variously as the extension of first-order logic with an operator for *least fixed points* or *inflationary fixed points*. While these are different logics, they are equivalent in terms of the classes of structures they can define (see [Ebbinghaus and Flum 1999]). The starting point for the search for logics for P is the theorem of Immerman [1986] and Vardi [1982] to the effect that FP captures P on structures which include a relation that linearly orders the universe.

THEOREM 2.1 (IMMERMAN-VARDI). *If σ is a signature including a binary relation $<$, then on any class of structures in which $<$ is interpreted as a linear order,* FP *captures* P.

The result easily extends from classes of structures in which an order is present to ones in which an order is definable. Indeed, it can be extended to results where a *canonisation* is definable. A canonisation function on a class $\mathcal{C}$ of σ structures is a function Γ that takes a structure $\mathfrak{A} \in \mathcal{C}$ to a $\sigma \cup \{<\}$-structure $\Gamma(\mathfrak{A})$ in which $<$ is a linear order; the reduct of $\Gamma(\mathfrak{A})$ to σ is isomorphic to $\mathfrak{A}$; and such that for isomorphic structures $\mathfrak{A}$ and $\mathfrak{B}$ in $\mathcal{C}$, $\Gamma(\mathfrak{A})$ and $\Gamma(\mathfrak{B})$ are isomorphic as ordered structures. In other words, Γ produces a canonical ordered version of the structure. In any logic L which extends FP, if there is a canonical definable function in L on a class $\mathcal{C}$, then every polynomial-time decidable property in $\mathcal{C}$ is definable in L.

In order to make precise sense of the above, we need to know what it means to define a function between structures in a logic L. This is given by the notion of interpretations. Fix vocabularies σ and τ with $\tau = (R_1, \ldots, R_m)$, and $\mathrm{ar}(R_i) = r_i$ for each $i \in [m]$ and let L be a logic. A *d-ary L-interpretation of τ in σ* is a sequence of L-formulas in vocabulary σ consisting of: (i) a formula $\delta(\vec{x})$; (ii) a formula $\varepsilon(\vec{x}, \vec{y})$; and (iii) for each relation symbol $R_i \in \tau$ a formula $\phi_{R_i}(\vec{x}_1, \ldots, \vec{x}_{r_i})$, where each $\vec{x}$, $\vec{y}$ or $\vec{x}_j$ is a d-tuple of variables. We call d the *dimension* of the interpretation. We say that an interpretation $\mathcal{I}$ associates a τ-structure $\mathfrak{B}$ to a σ-structure $\mathfrak{A}$ if there is a map h from $\{\vec{a} \in A^d \mid \mathfrak{A} \models \delta[\vec{a}]\}$ to the universe B of $\mathfrak{B}$ such that: (i) h is surjective onto B; (ii) $h(\vec{a}_1) = h(\vec{a}_2)$ if, and only if, $\mathfrak{A} \models \varepsilon[\vec{a}_1, \vec{a}_2]$; and (iii) $R_i^{\mathfrak{B}}(h(\vec{a}_1), \ldots, h(\vec{a}_{r_i}))$ if, and only if, $\mathfrak{A} \models \phi_{R_i}[\vec{a}_1, \ldots, \vec{a}_{r_i}]$. Note that an interpretation $\mathcal{I}$ associates a τ-structure with $\mathfrak{A}$ only if ε defines an equivalence relation on A^d that is a congruence with respect to the relations defined by the formulas ϕ_R. In such cases, however, $\mathfrak{B}$ is uniquely defined up to isomorphism and we write $\mathcal{I}(\mathfrak{A}) = \mathfrak{B}$. It is also worth noting that the size of $\mathfrak{B}$ is at most n^d, if $\mathfrak{A}$ is of size n.

If $\mathcal{C}$ and $\mathcal{D}$ are two classes of structures and $\mathcal{I}$ is an L-interpretation such that for any structure $\mathfrak{A}$ we have $\mathfrak{A} \in \mathcal{C}$ if, and only if, $\mathcal{I}(\mathfrak{A}) \in \mathcal{D}$ then we say that $\mathcal{I}$ is an L-reduction of $\mathcal{C}$ to $\mathcal{D}$. If there is an L-reduction from $\mathcal{C}$ to $\mathcal{D}$, we say that $\mathcal{C}$ is L-reducible to $\mathcal{D}$ and write this $\mathcal{C} \leq_L \mathcal{D}$.

An automorphism of a structure $\mathfrak{A}$ is an isomorphism of $\mathfrak{A}$ with itself. The collection of automorphisms of $\mathfrak{A}$ form a group which we denote $\mathbf{Aut}(\mathfrak{A})$. This induces an equivalence relation on the universe of $\mathfrak{A}$ where two elements a and b are equivalent if there is a $\pi \in \mathbf{Aut}(\mathfrak{A})$ such that $\pi(a) = b$. The equivalence classes of this relation are the *orbits* of $\mathbf{Aut}(\mathfrak{A})$. More generally, we define the orbits of $\mathbf{Aut}(\mathfrak{A})$ on k-tuples of elements of $\mathfrak{A}$ analogously.

We write $\mathbb{N}$ for the natural numbers and $\mathbb{P}$ for the collection of prime numbers. For $n \in \mathbb{N}$, we write $[n]$ for the set $\{1, \ldots, n\}$. We also write $\mathbb{Z}_m$ for the Abelian group (or ring, as context demands) of integers $(\mathrm{mod} m)$. For $p \in \mathbb{P}$, we write $\mathbb{F}_p$ to denote the field $\mathbb{Z}_p$.

We use the term *preorder* to mean a *linear preorder*. That is, a binary relation $\preceq$ which is reflexive and transitive and such that for every a and b in its domain either $a \preceq b$ or $b \preceq a$. Such a preorder induces an equivalence relation $\sim$ defined by $a \sim b$ if, and only if, $a \preceq b$ and $b \preceq a$. The preorder $\preceq$ can be seen as a linear order of the corresponding equivalence classes, which we refer to as *colour classes*.

3. FPC AND THE CFI-CONSTRUCTION

Immerman [1986] first suggested that one might extend the power of FP with a mechanism for counting, and this might suffice to express all properties in P. A more rigorous definition of the logic was given by Grädel and Otto [1993] and a full treatment can be found in Otto's monograph [1997]. The logic has two sorts of variables: element variables $x_1, x_2, \ldots$ ranging over the domain of the structure, and number variables $\nu_1, \nu_2, \ldots$ ranging over the set of integers $\{0, \ldots, n\}$, where n is the size of the structure. In addition, we also have second order variables $X_1, X_2, \ldots$, each of which has a type which is a finite string in $\{\mathrm{element}, \mathrm{number}\}^*$. Thus, if X is a variable of type $(\mathrm{element}, \mathrm{number})$, it is to be interpreted by a binary relation relating elements to numbers. The logic allows us to build up *numerical terms* according to the following rule:

> If ϕ is a formula and x is a variable, then $\#x\phi$ is a term.

The intended semantics is that $\#x\phi$ denotes the number (i.e. the non-negative integer) of elements that satisfy the formula ϕ. The formulas of FPC are now described by the following set of rules:

— all atomic formulas of first-order logic are formulas of FPC;
— if τ_1 and τ_2 are numerical terms (that is each one is either a number variable or a term of the form $\#x\phi$) then each of $\tau_1 < \tau_2$ and $\tau_1 = \tau_2$ is a formula;
— if ϕ and ψ are formulas then so are $\phi \wedge \psi$, $\phi \vee \psi$ and $\neg\psi$;
— if ϕ is a formula, x is an element variable and ν is a number variable, then $\exists x\, \phi$ and $\exists \nu \leq \eta\, \phi$ are formulas; and
— if X is a relation symbol of type α, $\vec{z}$ is a tuple of variables whose sorts match the type α and $\vec{t}$ is a tuple of terms of type α, then $[\mathbf{fp}_{X,\vec{z}}\phi](\vec{t})$ is a formula.
The intended semantics here is that the tuple of elements denoted by $\mathbf{t}$ is in the inflationary fixed-point of the operator defined by ϕ, binding the variables in $X, \mathbf{z}$.

A key tool in analysing the expressive power of FPC is to look at equivalence in a weaker logic—first-order logic with counting quantifiers and a bounded number of variables. For each natural number i, we have a quantifier $\exists^i$ where $\mathfrak{A} \models \exists^i x\, \phi$ if, and only if, there are at least i distinct elements $a \in A$ such that $\mathfrak{A} \models \phi[a/x]$. While the

extension of first-order logic with counting quantifiers is no more expressive than FO itself (in contrast to the situation with counting terms), the presence of these quantifiers does affect the number of variables that are necessary to express a query. Let C^k denote the k-variable fragment of first-order logic with counting quantifiers. That is, C^k consists of those formulas in which no more than k variables appear, free or bound. For two structures $\mathfrak{A}$ and $\mathfrak{B}$, we write $\mathfrak{A} \equiv^{C^k} \mathfrak{B}$ to denote that the two structures are not distinguished by any sentence of C^k. The link between this and FPC is the following fact, established by Immerman and Lander [1990]:

THEOREM 3.1. *For every sentence ϕ of* FPC*, there is a k such that if $\mathfrak{A} \equiv^{C^k} \mathfrak{B}$, then $\mathfrak{A} \models \phi$ if, and only if, $\mathfrak{B} \models \phi$.*

Indeed, this theorem follows from the fact that for any formula ϕ of FPC, there is a k so that on structures with at most n elements, ϕ is equivalent to a formula θ_n of C^k Additionally, it can be shown that the quantifier depth of θ_n is bounded by a polynomial function of n, but the important bound for us is that the number of variables k is bounded by a constant that only depends on ϕ.

The equivalence relations $\equiv^{C^k}$ have many different characterisations, some of which arose in contexts removed from the connection with logic, such as the Weisfeiler-Leman family of equivalences arising in the study of graph isomorphism (see the discussion in [Cai et al. 1992] for the connection). They give, for each k, an equivalence relation decidable in time $n^{O(k)}$, which serves as an approximation of the isomorphism relation. The relations get finer with increasing k, approaching isomorphism in the limit. The relations $\equiv^{C^k}$ have very useful characterisations in terms of two-player games, including the counting game of Immerman and Lander [1990] and the bijection game of Hella [1996]. These provide a means of arguing that two structures $\mathfrak{A}$ and $\mathfrak{B}$ are $\equiv^{C^k}$-equivalent. By Theorem 3.1, this can then be used to show that some property is not definable in FPC, by showing that it is not closed under $\equiv^{C^k}$ for any fixed k.

Cai et al. [1992] construct, for each k, a pair of graphs G_k and H_k that are not isomorphic but for which $G_k \equiv^{C^k} H_k$. The construction has been generalised to obtain a number of further inexpressibility results. Here, we give one very general form of the construction (from [Dawar et al. 2023]) which will serve several purposes. The structures we construct are not graphs, but structures over a vocabulary $\chi = (N, C, I, \preceq)$, where N and C are relations of arity 4 and I and $\preceq$ are binary. Fix an ordered graph $G = (V, E, \leq)$ that is, a simple undirected graph (V, E) together with a linear order $\leq$ on V; a positive integer m; and a function $\lambda : E \to \mathbb{Z}_m$. This gives rise to a χ-structure $\mathsf{CFI}[G, m, \lambda]$ defined as follows. For each vertex $v \in V$, we write $N_G(v)$ to denote the set of its neighbours in G.

— Let $B_v = \{\vec{a} \in \mathbb{Z}_m^{N_G(v)} \mid \sum \vec{a} = 0\}$. Then, the universe of $\mathsf{CFI}[G, m, \lambda]$ is $B = \biguplus_{v \in V} B_v$.
— For $\vec{a} \in B_u$ and $\vec{b} \in B_v$, we have $\vec{a} \preceq \vec{b}$ if, and only if, $u \leq v$.
— For $u \in V$ and $v \in N_G(u)$, let $N_{u,v} = \{(\vec{a}, \vec{b}) \in B_u^2 \mid \vec{a}(v) = \vec{b}(v)\}$. Then the relation N in $\mathsf{CFI}[G, m, \lambda]$ is defined as the set

$$\{(\vec{a}, \vec{b}, \vec{c}, \vec{d}) \mid \{(u, v) \mid (\vec{a}, \vec{b}) \in N_{u,v}\} \leq \{(u', v') \mid (\vec{c}, \vec{d}) \in N_{u',v'}\}\},$$

where the order $\leq$ is extended, lexicographically to sets of pairs of vertices from V.
— For $u \in V$ and $v \in N_G(u)$, let $C_{u,v} = \{(\vec{a}, \vec{b}) \in B_u^2 \mid \vec{a}(v) + 1 = \vec{b}(v)\}$. Then the relation C in $\mathsf{CFI}[G, m, \lambda]$ is defined as the set

$$\{(\vec{a}, \vec{b}, \vec{c}, \vec{d}) \mid \{(u, v) \mid (\vec{a}, \vec{b}) \in C_{u,v}\} \leq \{(u', v') \mid (\vec{c}, \vec{d}) \in C_{u',v'}\}\}.$$

— $I = \{(\vec{a}, \vec{b}) \in B_u \times B_v \mid (u,v) \in E, \vec{a}(v) + \vec{b}(u) = \lambda(u,v)\}$.

Thus, the structure replaces each vertex v in the ordered graph G with a collection of vectors. The relations N and C encode into these vectors the structure of the cyclic group $\mathbb{Z}_m$. Specifically, we wish to have available in our structure the binary relations $N_{u,v}$ and $C_{u,v}$. We cannot do this explicitly in a fixed vocabulary as it is an unbounded collection of relations, but we encode them into relations of arity 4, using the underlying order on the vertices of G. Finally, I introduces a twist along each edge given by the function λ. We call G the *base graph* of $\mathsf{CFI}[G, m, \lambda]$.

The key property is that the isomorphism class of the structure $\mathsf{CFI}[G, m, \lambda]$ is determined by *global* properties of λ and cannot be determined by its local structure.

LEMMA 3.2. *For a connected ordered graph $G = (V, E, \leq)$ and functions $\lambda, \mu : E \to \mathbb{Z}_m$ we have $\mathsf{CFI}[G, m, \lambda] \cong \mathsf{CFI}[G, m, \mu]$ if, and only if, $\sum_{e \in E} \lambda(e) = \sum_{e \in E} \mu(e)$.*

Then, if G is *sufficiently* connected, we can show that the structures $\mathsf{CFI}[G, m, \lambda]$ and $\mathsf{CFI}[G, m, \mu]$ cannot be distinguished in suitable logics. The proof of Cai et al. showed (essentially) that $\mathsf{CFI}[G, 2, \lambda] \equiv^{C^k} \mathsf{CFI}[G, 2, \mu]$ for any λ and μ as long as G does not have any separators with fewer than k vertices. This can be strengthened to work for all values of $m \geq 2$ (and, indeed, for all Abelian groups rather than just the cyclic groups $\mathbb{Z}_m$ [Atserias et al. 2009]) and to just requiring that the treewidth of G is larger than k [Dawar and Richerby 2007]. We refer to a structure of the form $\mathsf{CFI}[G, m, \lambda]$ as an *m-CFI structure* and a structure is a *CFI structure* if it is an m-CFI structure for some m.

Fix now a set S of positive integers. We define the *S-CFI query* be the class of structures $\{\mathsf{CFI}[G, m, \lambda] \mid m \in S \text{ and } \sum_{e \in E(G)} \lambda(e) = 0\}$. When $S = \{m\}$ is a singleton, we just write m-CFI query rather than $\{m\}$-CFI query. Then, the original result of [Cai et al. 1992] can be restated as follows

THEOREM 3.3. *The 2-CFI query is not definable in* FPC.

In the version of CFI structures that we have defined above, the base graph is an *ordered* graph. In fact, we can define a similar construction starting from any graph, without an order or, more generally, starting with a graph $G = (V, E, \ll)$ with a preorder $\ll$ on the vertices. The coding into relations N and C is more intricate as a result, but the details are not needed here. We refer to the CFI structures with unordered base graphs in Section 6 below.

We now turn to the expressibility of CFI queries in extended logics.

4. LINEAR ALGEBRAIC EXTENSIONS OF FPC

The 2-CFI query can be reduced to the problem of solving systems of linear equations over the field $\mathbb{F}_2$, and this was the insight that led to the definition of *rank logic* by [Dawar et al. 2009]. More generally, we can now see that for any m, the m-CFI query can be reduced to the problem of solving systems of equations over the cyclic group $\mathbb{Z}_m$. To see this, suppose $\mathfrak{B} = (B, N, C, I, \preceq)$ is a structure of the form $\mathsf{CFI}[G, m, \lambda]$. Then, we can easily recover the graph G from $\mathfrak{B}$. In particular, consider the equivalence relation on B given by $a \preceq b \wedge b \preceq a$. We can identify the equivalence classes of this relation with the vertices of G, and we write B_v for the equivalence class corresonding to a vertex v. Moreover, we can then recover from N the binary relations $N_{u.v}$ for each edge $\{u, v\}$ of G and hence also the set of edges. Similarly, the binary relations $C_{u,v}$ are definable given parameters in B_u and B_v. Our system of equations then has, for each $a \in B_u$ and

each $v \in N_G(u)$ a variable a_v, and the following equations.

$$\begin{aligned} a_v &= b_v && \text{for all } (a,b) \in N_{u,v} \\ a_v &= b_v + 1 && \text{for all } (b,a) \in C_{u,v} \\ a_v + b_u &= 0 && \text{for all } (a,b) \in I \\ \textstyle\sum_{v \in N_G(u)} a_v &= 0 && \text{for all } a \in B_u \end{aligned}$$

Note that the structure $\mathsf{CFI}[G, m, \lambda]$ is in the m-CFI query if, and only if, $\sum_{e \in E(G)} \lambda(e) = 0$ and by Lemma 3.2, this is the case if, and only if, it is isomorphic to the structure $\mathsf{CFI}[G, m, \zeta]$ where ζ is the constant zero function. The system of equations constructed from $\mathfrak{B}$ above is indeed satisfiable in $\mathbb{Z}_m$ if the binary relation I codes the constant zero function and not otherwise.

This observation is what leads to the suggestion that we can obtain logics more powerful than FPC by coding into them the ability to solve systems of equations over finite fields. The first proposal along these lines was the logic introduced in [Dawar et al. 2009] that incorporates an operator for computing the rank of a matrix over a finite field. The reasons for considering an operator specifically for *rank* are two-fold. First, being able to compute the rank of a matrix allows us easily to determine the solvability of a system of equations, since a system $A\vec{x} = \vec{b}$ is solvable if, and only if, the rank of the matrix $A|\vec{b}$ is the same as that of A. Secondly, rank seems a natural generalisation of the notion of counting: rather than just allowing us to determine the cardinality of a definable set, it allows us to determine the dimension of a definable vector space.

The rank logic FPR^- is obtained by closing the logic FPC as defined above under the following rule for forming new numerical terms:

— Let t be a positive integer and p a prime number. If η is a numerical term and $\vec{x}$ and $\vec{y}$ are t-tuples of variables (possibly of mixed sort), then $\mathbf{rk}_p^t \vec{x}\vec{y}\eta$ is a numerical term.

To define the semantics, let $D_{\vec{x}}$ denote the collection of t-tuples from $A \uplus \{0, \ldots, |A|\}$ that are possible interpretations of the tuple of variables $\vec{x}$ and similarly let $D_{\vec{y}}$ be the possible interpretations of $\vec{y}$. Then, the term $\eta(\vec{x}, \vec{y})$ defines a function $\eta : D_{\vec{x}} \times D_{\vec{y}} \to \mathbb{N}$ which gives for each possible interpretation of $\vec{x}, \vec{y}$ the value of the term. Then, $\mathbf{rk}_p^t \vec{x}\vec{y}\eta$ denotes the rank of the matrix $M \in \mathbb{F}_p^{D_{\vec{x}} \times D_{\vec{y}}}$, where $M(a,b) = \eta(a,b) \pmod p$. It should be noted that, while this logic only allows us to define the rank of matrices over finite prime fields, it is shown in [Dawar et al. 2009] that these operators do in fact allow us to define the rank of matrices over arbitrary finite fields.

The logic FPR defined by [Grädel and Pakusa 2019] is a varation in which, rather than having an operator $\mathbf{rk}_p^t$ for each prime number, p is a parameter. In other words, p can be a numerical term whose value may vary. We can let the term $\mathbf{rk}_p^t \vec{x}\vec{y}\eta$ take the value 0 when the term p is not a prime number. In introducing this variation, Grädel and Pakusa showed that FPR is strictly more powerful that FPR^-.

THEOREM 4.1 ([GRÄDEL AND PAKUSA 2019]). *There is a property definable in* FPR *that is not definable in* FPR^-.

Indeed, their proof shows that the $\mathbb{P}$-CFI query is not definable in FPR^-, where $\mathbb{P}$ is the set of all prime numbers.

The reason is that in any formula of FPR^- there is a finite set $\Omega \subseteq \mathbb{P}$ of prime numbers that appear as characteristics in rank operators. What Grädel and Pakusa show is that, over the collection of p-CFI structures for some fixed prime $p \notin \Omega$, any such formula is equivalent to a formula of FPC. That is to say, on these structures we can eliminate the use of rank operators of fixed characteristic other than p. But, since

no sentence of FPC can distinguish $\mathsf{CFI}[G, p, \lambda]$ from $\mathsf{CFI}[G, p, \mu]$ for any p and any λ and μ, we conclude that for any formula ϕ of FPR^- there are structures in the $\mathbb{P}$-CFI query and structures outside it that ϕ does not distinguish.

This rather ingenious proof combines two elements: the known limitations of FPC on CFI queries, which are established by game arguments, with a rank operator elimination argument. It does not directly use a game characterisation of FPR^-. Indeed, a game had been proposed by [Dawar and Holm 2012] for studying the expressive power of FPR^-, known as the invertible map game. The idea behind it is that just as Hella's bijection game characterises counting logic because bijections preserve all cardinalities, a *Spoiler-Duplicator* game could be built in which Duplicator plays *invertible linear maps* in order to capture properties of definable vector spaces. The game is defined as follows.

Invertible Map Game. The game is played between *Spoiler* and *Duplicator* on a pair of structures $\mathfrak{A}$ and $\mathfrak{B}$, with universes A and B respectively. It is parametrised by a positive integer k and a set Ω of prime numbers. We have, as usual, k pairs of pebbles $(s_i, t_i)_{i\in[k]}$. The pebbles s_i may be placed on elements of A and the pebbles t_i on elements of B. Thus, the position of the game at any point has at most k pebbles in A and at most k pebbles in B and this defines a partial map $f : A \rightharpoonup B$ taking the element pebbled by s_i to the element pebbled by t_i (we are assuming that distinct pebbles are always placed on distinct elements, which we can do without loss of generality). Play proceeds as follows.

(1) *Spoiler* announces a set $j_1, \ldots, j_{2m} \in [k]$ of indices of pebbles that are going to move in this round;
(2) *Spoiler* chooses a prime $p \in \Omega$;
(3) *Duplicator* gives a partition of A^{2m} into parts $P_1, \ldots, P_t$ and a partition of B^{2m} into parts $Q_1, \ldots, Q_t$ along with an invertible linear map $S : \mathbb{F}_p^{A^m} \to \mathbb{F}_p^{B^m}$ so that

$$SP_iS^{-1} = Q_i \quad \text{for } 1 \leq i \leq t,$$

where we think of the relations $P_i \subseteq A^{2m}$ and $Q_i \subseteq B^{2m}$ as matrices in $\{0,1\}^{A^m\times A^m}$ and $\{0,1\}^{B^m\times B^m}$, respectively;
(4) *Spoiler* chooses some $i \in \{1, \ldots, t\}$ and an $\vec{a} \in P_i$ and $\vec{b} \in Q_i$; and
(5) the pebbles $s_{j_1}, \ldots, s_{j_{2m}}$ are placed on $\vec{a}$ and $t_{j_1}, \ldots, t_{j_{2m}}$ on $\vec{b}$.

If, at the end of any move, the partial map $f : A \rightharpoonup B$ given by the pebbled positions is not a partial isomorphism from $\mathfrak{A}$ to $\mathfrak{B}$, then *Spoiler* has won the game, otherwise it can continue. One can then show that for any formula ϕ of FPR^-, there is a constant k and a finite set Ω such that if *Duplicator* has a winning strategy to play forever on $\mathfrak{A}$ and $\mathfrak{B}$, then ϕ cannot distinguish these structures. Thus, the games could be used, in principle, to show inexpressibility in FPR^- or, by taking $\Omega = \mathbb{P}$, also in FPR. We write $\mathfrak{A} \equiv^{\mathrm{IM}}_{k,\Omega} \mathfrak{B}$ to indicate that Duplicator has a winning strategy to play forever on $\mathfrak{A}$ and $\mathfrak{B}$ in the k, Ω-invertible-map game.

Indeed, the games show inexpressibility not just in logics with rank operators, but really logics with any operators whose semantics is invariant under invertible linear maps. This idea was formalised by Dawar et al. [2019] who defined, for each set $\Omega \subseteq \mathbb{P}$ a logic $\mathrm{LA}^\omega(\Omega)$, which is an infinitary logic including all possible linear-algebraic operators with prime characteristic in Ω. We refer to [Dawar et al. 2019] for the formal definition but note here that this is a very rich logic, which can define undecidable properties and so is not a candidate logic for P. Its purpose is rather to delimit what can be done in principle with linear-algebraic operators and give a logical meaning to the invertible map game.

The main result established in [Dawar et al. 2019] is the following.

THEOREM 4.2 ([DAWAR ET AL. 2019]). *If $\Omega \subseteq \mathbb{P}$ and p is a prime not in Ω, then the p-CFI query is not definable in* $\mathrm{LA}^\omega(\Omega)$.

Thus, if any logic contained with $\mathrm{LA}^\omega(\Omega)$ is to capture P, then Ω must contain all prime numbers. The proof makes intricate use of new algebraic machinery, in particular Maschke's theorem from representation theory, exploiting key properties of the p-CFI structures. The first is that these structures are *homogeneous* in that the orbits of the automorphism group acting on k-tuples can be defined in counting logic with l variables, where l depends only on k and not on the size of the structures. The second is the property that the automorphism group itself is an Abelian p-group. Together these imply that the partition into orbits, seen as an $\mathbb{F}_q$-algebra for $q \neq p$ is semi-simple and its structure can be determined by counting the types of its simple algebras. This enables us to give, within FPC a definition of the equivalence relation $\equiv^{\mathsf{IM}}_{k,\mathbb{P}\setminus\{p\}}$, now seen as an equivalence relation on k-tuples of elements in the structures. With this, we are able to show that all linear-algebraic operators with characteristics in $\mathbb{P} \setminus \{p\}$ can be eliminated in favour of counting terms, and Theorem 4.2 follows.

While Theorem 4.2 is a major limiting result on the power of the invertible map equivalences, the proof does not use the characterisation of these equivalences in terms of games. The first use of the games to prove a strong result is due to [Lichter 2023], who established the following.

THEOREM 4.3 ([LICHTER 2023]). *There is a problem in* P *that is not definable in* FPR.

Indeed, the problem is the T-CFI query, where $T = \{2^i \mid i \in \mathbb{N}\}$. Lichter's proof builds on the proof from [Grädel and Pakusa 2019] that shows that over the class of 2-CFI structures, all rank operators with characteristic other than 2 can be eliminated and extends it to 2^i-CFI structures. He then shows that there are, for all k and i, graphs G such that $\mathsf{CFI}[G, 2^i, \lambda] \equiv^{\mathsf{IM}}_{k,\{2\}} \mathsf{CFI}[G, 2^i, \mu]$ for any λ and μ. It is this last result that builds an intricate game-winning strategy in the invertible map game.

Lichter's use of the invertible map game actually shows that the T-CFI query cannot be defined using *any* linear algebraic operators of characteristic 2, and is not confined just to the rank operators. It turns out that this can be combined with the strong results from the proof of Theorem 4.2 to the effect that linear operators of characteristic p can be eliminated on the class of m-CFI structures when m is co-prime with p. In particular, this means that when m is a power of 2, all linear operators of characteristic other than 2 can be replaced by suitable formulas of FPC. Together this establishes that the T-CFI query is not definable in LA^ω at all.

THEOREM 4.4 ([DAWAR ET AL. 2023]). *There is a problem in* P *that is not definable in* LA^ω.

In one sense, this is a conclusion to the line of work which sought a logic for P by extending FPC with algebraic operations over finite fields. In another sense, this is the most successful line in extending the strong lower bound methods developed in the CFI construction to logics significantly stronger than FPC. Also, the CFI construction not only showed the limits of FPC, it also showed the limits of the combinatorial Weisfeiler-Leman algorithms to solve the graph isomorphism problem. The invertible map game introduced an entirely new algebraic paradigm to polynomial-time decidable approximations of graph isomorphism. Theorem 4.4 finally settles the status of this approximation.

5. WITNESSED SYMMETRIC CHOICE CONSTRUCTS

Another approach is to extend fixed-point logic with a non-deterministic *choice operator*. This is inspired by the observation that in many natural computations, choices are made but do not influence the result. For instance, suppose one wants to determine whether there exists a path from s to t in a given graph via depth-first search. This involves making a choice as to which vertex to visit next at each step. If there is an s-t-path, the algorithm will always output "yes", regardless of which choices it made, and if there is none, it will always output "no". Defining a logic based on this idea requires to have a general condition on the choices which ensures that the outcome of the computation is indeed invariant under them. It can be shown that this is the case if the *choice set* from which an element or tuple is non-deterministically selected is always an *orbit* of the input structure, i.e. all elements of the choice set are related via an automorphism. Non-deterministic choices from orbits therefore lead to a deterministic computation result.

Hence, choosing from orbits is a valid operation that one can incorporate into fixed-point logic without breaking the symmetry-invariance of the computation model. This of course also requires the orbits to be definable by a formula of the logic. But there are some subtleties involved: Suppose we describe a fixed-point operator that allows one to choose in each iteration a tuple from some set defined by a formula $\phi(\bar{x})$. This is not a logic because in general, the set defined by ϕ may be a *union* of several orbits – and then, the result of the computation does depend on the choices. Therefore, during the evaluation it must be ensured in every choice step that ϕ indeed defines only a single orbit. This has the same complexity as the graph isomorphism problem and is hence not known to be possible in polynomial time. A way to still ensure polynomial time evaluation complexity is to require that the formula defining the choice set also has to provide *witnesses* for the choice set being an orbit. These witnesses are suitable automorphisms of the structure. An algorithm evaluating the formula then just needs to verify that the witnesses are indeed automorphisms and relate every element of the choice set with each other.

A fixed-point logic with witnessed symmetric choice (but without counting) was first suggested by Gire and Hoang in [1998]. As they show, this new operator allows to linearly order (and hence distinguish) the CFI graphs that were used to separate FPC from P. This established witnessed symmetric choice as a proper extension of FPC.

Dawar and Richerby [2003] study a variation of this logic with a few modifications in the details of the semantics. Their logic allows parameters in the fixed-point formulas, which provably increases the expressive power. Also, they introduce "reduct semantics": The choice set is not required to be an orbit of the input structure itself, but only of its reduct to the relations appearing in the formula. Without this condition, the presence of relations that are not mentioned in the formula could influence its semantics. The investigation in [Dawar and Richerby 2003] focuses on a symmetric choice logic without the witnessing aspect, called $\mathrm{SC\text{-}IFP}$. This simplifies the analysis but the logic is likely not contained in P. Indeed, $\mathrm{SC\text{-}IFP}$ is powerful enough to express the graph isomorphism problem. Conversely, it can be evaluated by a polynomial time Turing machine with access to a graph isomorphism oracle (this complexity class is denoted $\mathrm{P}^{\mathbf{GI}}$). On linearly ordered structures, the closure of $\mathrm{SC\text{-}IFP}$ under FO-interpretations actually captures $\mathrm{P}^{\mathbf{GI}}$.

Logics with symmetric choice are indeed not a priori interpretation-closed, and to obtain the full expressive power, it is necessary to explicitly add a certain interpretation operator to the syntax. The relationship between the expressive power and the nesting depth of choice and interpretation operators is the subject of a recent article by Lichter [2022]. The central result is that fixed-point logic with counting and witnessed

symmetric choice fails to capture P because it is strictly less expressive than the same logic enriched by an interpretation operator. An analogous case was already made for the same logic *without counting* in [Gire and Hoang 1998]. The techniques needed for the separation result in the presence of counting are much more sophisticated and we present them in some detail below. But first, let us illustrate less formally why an interpretation operator adds additional power to choice logics.

Imagine a graph G that is rigid, i.e. has no non-trivial automorphisms. Every admissible choice set on such a graph is a singleton and hence, a symmetric choice operator does not yield any extra power. But there may still exist an FO-interpretation Θ such that $\Theta(G)$ does have non-trivial automorphisms. Then on $\Theta(G)$, a fixed-point logic with symmetric choice could indeed be strictly more powerful than the fixed-point logic itself. Thus, it is not obvious that choice logics should be interpretation-closed like fixed-point logics are.

For this reason, [Gire and Hoang 1998] suggested enhancing symmetric choice logics with an operator that takes an interpretation Θ and a formula ψ, and evaluates ψ in $\Theta(G)$. They argue that this operator allows one to count the cardinality of the subset of the input structure G defined by a formula $\phi(x)$: Use an interpretation Θ that maps G to a structure with just one unary relation $R = \{a \in V(G) \mid G \models \phi(a)\}$. Then it is possible to define a linear order on R using the symmetric choice operator in $\Theta(G)$; the order allows to define $|R|$, even if there is no explicit counting operator in the logic.

In [Lichter 2022], the base logic is FPC, so counting does not separate FPC with witnessed symmetric choice from its extension with the interpretation operator. Instead, Lichter constructs graphs G and H which are not distinguishable in FPC and have no non-trivial automorphisms. Thus, the symmetric choice operator cannot be used to separate them, either. However, it is possible to interpret in G and H, respectively, highly symmetric graphs that can be distinguished using symmetric choice.

THEOREM 5.1 ([LICHTER 2022]).

$$\mathrm{FPC{+}WSC} < \mathrm{FPC{+}WSC{+}I} \leq \mathrm{P}.$$

FPC+WSC stands for inflationary FPC with witnessed symmetric choices, and FPC+WSC+I is its extension with an interpretation operator. We present the definition of these two logics as in [Lichter 2022]. The logic FPC+WSC is essentially based on the semantics from [Dawar and Richerby 2003] and contains a witnessing mechanics similar to [Gire and Hoang 1998]; the main difference to these previous logics is that counting is part of the syntax and the choices are witnessed "in hindsight" rather than during the computation. Formally, the WSC-fixed-point operator with free variables/parameters $\bar{p}\bar{\nu}$ (element and numeric variables) looks like this:

$$\Phi(\bar{p}\bar{\nu}) = \mathsf{ifp\text{-}wsc}_{R,\bar{x};R^*;S,\bar{y},\bar{y}';z_1z_2}\cdot\Big(\Phi_{\text{step}}(\bar{p},\bar{\nu},\bar{x}),\Phi_{\text{choice}}(\bar{p},\bar{\nu},\bar{y}),\Phi_{\text{wit}}(\bar{p}\bar{\nu}\bar{y}\bar{y}'z_1z_2),\Phi_{\text{out}}(\bar{p}\bar{\nu})\Big).$$

Here, R, R^* and S are new relation symbols, and the four FPC+WSC-formulas in the scope of the operator have the following roles:

— $\Phi_{\text{step}}(\bar{p}\bar{\nu}\bar{x})$ defines in a structure $\mathfrak{A}$ with universe A the tuples $\bar{a} \in A^{|\bar{x}|}$ to be added to the fixed-point in each step, using the relation S that holds a previously chosen tuple.
— $\Phi_{\text{choice}}(\bar{p}\bar{\nu}\bar{y})$ defines the set of tuples $\bar{a} \in A^{|\bar{y}|}$ from which the choice used by the step-formula is made (the *choice set*).
— $\Phi_{\text{wit}}(\bar{p}\bar{\nu}\bar{y}\bar{y}'z_1z_2)$ defines a set of automorphisms that witness the choice set to be an orbit (for details, see below).
— $\Phi_{\text{out}}(\bar{p}\bar{\nu})$ is evaluated when the fixed-point is reached, and expresses the desired property of the fixed-point relation.

Given a structure $\mathfrak{A}$ and a tuple $\bar{u}$ that is of sorts matched to $\bar{p}\bar{\nu}$, then $\mathfrak{A} \models \Phi(\bar{u})$ if and only if $(\mathfrak{A}, R^*) \models \Phi_{\text{out}}(\bar{u})$ for one/every fixed-point relation R^* which is obtained as a result of the following inflationary fixed-point process. We describe for simplicity one fixed-point computation but it is to be understood as a set (or a tree) of several computations, one for every way to resolve the choices.

Let $R_0 := \emptyset$. Given R_i, the next stage R_{i+1} of the fixed-point is obtained as follows. First, let $T_{i+1} := \{\bar{v} \mid \mathfrak{A} \models \Phi_{\text{choice}}(\bar{u}, \bar{v})\}$ be the choice set for this step. Then S_{i+1} is set to be the singleton relation $\{\bar{v}\}$ for an arbitrarily chosen $\bar{v} \in T_{i+1}$. Finally, $R_{i+1} := R_i \cup \{\bar{a} \in A^{|\bar{x}|} \mid (\mathfrak{A}, R_i, S_{i+1}) \models \Phi_{\text{step}}(\bar{u}, \bar{a})\}$. Note that S is part of the vocabulary of Φ_{step}, so the computation of the next stage indeed depends on the chosen tuple. Since the structure is finite, a fixed-point is always reached. Let R^* denote this fixed-point relation. Before we can evaluate the output formula, it needs to be checked that every choice set in the computation is indeed correctly witnessed as an orbit. The witnessing formula has access to the fixed-point R^*. For each step $i+1$ in the computation, it is required that T_{i+1} is an orbit of the structure $(\mathfrak{A} \restriction \tau(\Phi), \bar{u}, R_1, ..., R_i)$, where $\tau(\Phi)$ denotes the vocabulary of Φ (this is the reduct semantics mentioned before). The formula Φ_{wit} defines in $(\mathfrak{A}, \bar{u}, R_i, R^*)$ for every two tuples $\bar{v}, \bar{v}' \in T_{i+1}$ a set of pairs $(w, w') \in A^2$ that satisfy $\Phi_{\text{wit}}(\bar{u}, \bar{v}\bar{v}', w, w')$. This set of pairs is intended to code a permutation of A (if it does not, then the orbit is not correctly witnessed). If for every pair $\bar{v}, \bar{v}' \in T_{i+1}$, Φ_{wit} defines an automorphism of $(\mathfrak{A} \restriction \tau(\Phi), \bar{u}, R_1, ..., R_i)$ that takes $\bar{v}$ to $\bar{v}'$, then T_{i+1} is witnessed to be an orbit. If all choice sets in the fixed-point computation are witnessed in this way, then the tuple $\bar{u}$ satisfies the whole formula Φ in $\mathfrak{A}$ if $(\mathfrak{A}, R^*) \models \Phi_{\text{out}}(\bar{u})$. If not all choices were witnessed correctly, then $\bar{u}$ is not a satisfying assignment for the free variables $\bar{p}\bar{\nu}$ of Φ.

Since every choice set in the computation is an orbit, it can be shown that $(\mathfrak{A}, R^*) \models \Phi_{\text{out}}(\bar{u})$ either holds for every possible fixed-point R^* or for none of them. It is also provable that either every possible choice set is witnessed or none of them is, so for the evaluation of this operator, it is indeed sufficient to choose an arbitrary tuple from T_{i+1} every time – we do not need to explore the space of all possible computations.

The logic FPC+WSC is FPC augmented with the described witnessed choice fixed-point operator. According to the explanations above, the operator can be evaluated in polynomial time and is isomorphism-invariant.

Now FPC+WSC+I extends this logic by the following interpretation operator. Again, let $\bar{p}\bar{\nu}$ be the free variables of the operator. Let $\Theta(\bar{p}\bar{\nu})$ be an FPC+WSC+I-interpretation with parameters that takes τ- to σ-structures. Let Φ be an FPC+WSC+I$[\sigma]$-sentence. Then

$$\Psi(\bar{p}\bar{\nu}) = \mathsf{I}(\Theta(\bar{p}\bar{\nu}); \Phi)$$

is an FPC+WSC+I$[\tau]$-formula. A τ-structure $\mathfrak{A}$ and an appropriately typed assignment $\bar{u}$ to the variables satisfy $\Psi(\bar{u})$ if the structure $\Theta(\mathfrak{A}, \bar{u})$ satisfies Φ. Thus, FPC+WSC+I is interpretation-closed by definition. It is also contained in P because the structure $\Theta(\mathfrak{A}, \bar{u})$ can be efficiently computed.

Theorem 5.1 shows that FPC+WSC is indeed not by itself interpretation-closed. The counterexample used in the proof is a combination of CFI graphs (over $\mathbb{Z}_2$) with *multipedes* [Gurevich and Shelah 1996]. A multipede is a CFI-like construction applied to a bipartite base graph. It has pairs of "feet" (hence the name), such that the two feet within each pair are indistinguishable in bounded-variable counting logic. Unlike CFI graphs, multipedes have no non-trivial automorphisms. The CFI graphs that are used here differ from the ones we have presented in Section 3 in so far as they also have edge gadgets, i.e. one pair of vertices per edge that represent the two elements of $\mathbb{Z}_2$; this is closer to the original construction in [Cai et al. 1992]. In the multipede-CFI combination, each vertex-pair in an edge gadget of the CFI graph is identified with a

pair of feet of a multipede. This yields a rigid graph. It is then shown that, just like CFI graphs and multipedes on their own, these CFI-multipede-combinations yield pairs of non-isomorphic structures that are indistinguishable in bounded-variable counting logic and hence FPC. The lack of automorphisms means that FPC+WSC collapses to FPC. But it is possible to remove the multipede from such a structure with the interpretation operator. The remaining CFI graph does possess non-trivial symmetries, and these can be exploited by the choice operator to define a linear order and distinguish the non-isomorphic structures. Since a polynomial time algorithm can also extract the CFI graphs from the combined structures and tell them apart, the isomorphism problem on these structures is in P. This is how Theorem 5.1 is proved.

It remains open whether FPC+WSC+I captures P or not. A first step towards a negative answer is also achieved in [Lichter 2022]. The suggested approach towards separating FPC+WSC+I from P is to show a strict operator hierarchy for FPC+WSC+I in the sense that an increase in the alternating nesting depth of choice and interpretation operators strictly increases the expressive power. From the proof of Theorem 5.1 it follows that FPC+WSC+I-formulas with one interpretation operator are strictly more expressive than FPC+WSC-formulas, so this already separates the first levels of the operator hierarchy. In [Lichter 2022], another, more systematic construction for showing this and potentially also the separations between higher levels of the hierarchy is presented: The general idea is to nest the CFI construction and consider CFI graphs whose base graphs are again CFI graphs. It is shown that FPC+WSC+I can canonise a CFI graph if it can define (and order) the orbits of pairs in the base graph. The method used for this is based on Gurevich's canonisation algorithm [Gurevich 1997], which can be implemented with the witnessed choice operator – more details will be given later in Section 6.2. Canonising the CFI graph with this approach requires one to put one more choice and interpretation operator "on top" of the formula for defining the orbits of the base graph. So in order to canonise, say, a CFI graph whose base graph is again a CFI graph, one first canonises the "base" CFI graph (provided its base graph has definable 2-orbits). This allows to define its 2-orbits, and then in turn to canonise the "topmost" CFI graph. The operator nesting depth increases accordingly with the CFI nesting depth. The challenging part is to prove that this can indeed not be achieved with a lower operator nesting depth. In [Lichter 2022], a lower bound to this effect is proved at least for the lowest level of the nesting hierarchy, i.e. FPC+WSC without nested choice operators: That is, a certain two-fold CFI-construction (involving a number of disjoint copies of the base CFI graph) is presented, which provably cannot be canonised using choice operators without nesting and interpretations. The same graphs can however be canonised by an FPC+WSC-formula that nests a choice operator, an interpretation operator and another choice operator. The problem that remains open is to lift this to FPC+WSC+I-formulas of higher operator nesting depth. In particular, it would be desirable to prove that for every constant operator nesting depth, there exist structures that cannot be distinguished at that level of the nesting hierarchy. One of the challenges here is to understand how the orbit structure of a structure can be modified by the application of an interpretation. This is crucial in order to limit the expressive power of the choice operator on interpreted structures as well.

6. EXTENSION WITH HIGHER-ORDER OBJECTS

6.1. Choiceless Polynomial Time

The third direction extending fixed-point logic that we consider is to allow fixed-points over more expressive objects. In FPC, the fixed-point computed by any given formula is a relation of constant arity over the input structure. In Choiceless Polynomial Time (CPT), it is a *hereditarily finite set* with atoms from the input structure. Given a set A,

the set of hereditarily finite objects over A, $\mathrm{HF}(A)$, is defined as $\bigcup_{i\in\mathbb{N}} \mathrm{HF}_i(A)$, where $\mathrm{HF}_0(A) := A \cup \{\emptyset\}, \mathrm{HF}_{i+1}(A) := \mathrm{HF}_i(A) \cup 2^{\mathrm{HF}_i(A)}$. In what follows, we write *h.f. sets* for hereditarily finite sets. Relations of constant arity can be encoded as h.f. sets but not every h.f. set in $\mathrm{HF}(A)$ is representable as a fixed-arity relation $R \subseteq A^k$. An important difference between fixed-point computations over relations and over h.f. sets is also that in the latter case, the universe is growing during the computation: In FPC, each computation step just adds tuples in A^k to the constructed fixed-point; in CPT, each step can create new h.f. sets in $\mathrm{HF}(A)$ which were previously not available. For this reason, CPT computations are not automatically guaranteed to terminate. To enforce containment in P, CPT sentences are equipped with an explicit polynomial bound, and computations exceeding this bound in time or space are aborted. In terms of expressive power, CPT is known to be strictly more powerful than FPC but it is still open whether it captures P or not.

The original definition of CPT is from 1999 and due to Blass, Gurevich and Shelah [1999]. They devised it as a symmetry-invariant machine model rather than a logic. We describe a later formulation of CPT due to Rossman [2010] that is known as *BGS-logic*. The presentation follows the one in [Schalthöfer 2020].

Definition 6.1 (*Syntax of BGS-logic*). Let σ be a relational vocabulary. The set $\mathrm{BGS}[\sigma]$ of BGS-terms and -formulas is defined inductively as follows:

— Every variable x is a term.
— $\emptyset$ and Atoms are terms.
— If r, s are terms, then $\mathsf{Unique}(s), \mathsf{Pair}(r, s), \mathsf{Union}(s)$ and $\mathsf{Card}(s)$ are terms.
— If $s_1, ..., s_k$ are terms and $R \in \sigma$ is a k-ary relation symbol, then $Rs_1...s_k$ is a formula.
— If ϕ, ψ are formulas, then so are $\phi \wedge \psi$ and $\phi \vee \psi$.
— If s, t are terms, then $s = t$ and $s \in t$ are formulas.
— If r, s are terms, x is a variable that is not free in r, and ϕ is a formula, then $\{s : x \in r : \phi\}$ is a *comprehension term*.
— If s is a term with only one free variable, then s^* is an *iteration term*.

In BGS-logic, iteration terms are unbounded, so they cannot generally be evaluated in polynomial time. The logic CPT is obtained from BGS by putting an explicit polynomial bound on every BGS-term and -formula. CPT is the set of all pairs (t, p) where t is a BGS-term/-formula, and $p : \mathbb{N} \longrightarrow \mathbb{N}$ is a polynomial. This bound is only relevant for the evaluation of iteration terms. BGS-terms evaluate in a given structure $\mathfrak{A}$ to h.f. sets over its universe. Here we only present the semantics of terms which are not standard FO expressions:

Definition 6.2 (*Semantics of BGS*). Let $\mathfrak{A}$ be a finite σ-structure with universe A and β be an assignment that maps the free variables of the following terms and formulas to values in $\mathrm{HF}(A)$. The evaluation of terms and formulas in $(\mathfrak{A}, \beta)$ is defined as follows:

— $[\![\emptyset]\!]^{\mathfrak{A}} = \emptyset$ and $[\![\mathsf{Atoms}]\!]^{\mathfrak{A}} = A$.
— $[\![\mathsf{Unique}(s)]\!]^{\mathfrak{A},\beta} = \begin{cases} a, \text{if } [\![s]\!]^{\mathfrak{A},\beta} = \{a\}, \\ \emptyset, \text{otherwise.} \end{cases}$
— $[\![\mathsf{Pair}(r, s)]\!]^{\mathfrak{A},\beta} = \{[\![r]\!]^{\mathfrak{A},\beta}, [\![s]\!]^{\mathfrak{A},\beta}\}$.
— $[\![\mathsf{Union}(s)]\!]^{\mathfrak{A},\beta} = \bigcup_{b\in[\![s]\!]^{\mathfrak{A},\beta}} b$.
— $[\![\mathsf{Card}(s)]\!]^{\mathfrak{A},\beta} = |[\![s]\!]^{\mathfrak{A},\beta}|$, represented as a von-Neumann ordinal in $\mathrm{HF}(A)$.
— $[\![\{s : x \in r : \phi\}]\!]^{\mathfrak{A},\beta} = \{[\![s]\!]^{\mathfrak{A},\beta[x\mapsto b]} \mid b \in [\![r]\!]^{\mathfrak{A},\beta} \text{ with } \mathfrak{A}, \beta[x \mapsto b] \models \phi\}$.

— For the evaluation of a polynomially bounded iteration term (t^*, p) in $\mathfrak{A}$, define the sequence $(t^i)_{i\in\mathbb{N}}$ as $t^0 = \emptyset$ and $t^{i+1} = t[x/t^i]$, where x is the free variable of t. Then

$$[\![(t^*,p)]\!]^{\mathfrak{A}} = \begin{cases} [\![t^\ell]\!]^{\mathfrak{A}}, & \text{for the least } \ell \leq p(|A|) \text{ with } t^\ell = t^{\ell+1} \text{ and} \\ & \text{size}([\![t^i]\!]^{\mathfrak{A}}) \leq p(|A|) \text{ for all } i \leq \ell, \text{ if such an } \ell \text{ exists,} \\ \emptyset, & \text{otherwise.} \end{cases}$$

By size$([\![t^i]\!]^{\mathfrak{A}})$, we mean the total number of h.f. sets appearing in the *transitive closure* of $[\![t^i]\!]^{\mathfrak{A}}$. The transitive closure of a set $a \in \mathrm{HF}(A)$ is the least set $\mathrm{tc}(a) \subseteq \mathrm{HF}(A)$ containing a and satisfying that if $b \in \mathrm{tc}(a)$, then $b \subseteq \mathrm{tc}(a)$. In other words, the size of a h.f. set is measured as the number of all h.f. sets appearing in it at any nesting depth.

The power of CPT. The first reason why CPT is more powerful than FPC is simply because it benefits from padding in the instances. For example, if one augments CFI graphs, which are indistinguishable in FPC, with a sufficient (exponential) number of isolated vertices, then they are still indistinguishable in FPC while CPT suddenly has exponential resources available. This enables CPT to distinguish the padded non-isomorphic CFI graphs [Blass et al. 2002]. But as it turns out, CPT can even distinguish CFI graphs *without padding*, and this is where the expressive power of deeply nested sets really comes into play. In 2008, Dawar, Richerby and Rossman presented a sophisticated construction of a highly symmetric h.f. set which can be used by a CPT sentence to characterise the parity of a given CFI graph [Dawar et al. 2008]. The CFI graphs considered here are again the classical ones from [Cai et al. 1992], which have edge and vertex gadgets. The h.f. set construction from [Dawar et al. 2008] works as long as the base graph of the CFI graph is equipped with a linear order on its vertices, like it is the case with the CFI structures in Section 3. The construction was later shown [Pakusa et al. 2018] to be adaptable to two other classes of base graphs, one of which is a strict generalisation of the linearly ordered ones:

THEOREM 6.3 ([DAWAR ET AL. 2008], [PAKUSA ET AL. 2018]).
CPT defines the CFI query over base graphs $G = (V, E)$ which are

— *linearly ordered, or*
— *preordered with colour class size $\mathcal{O}(\log |V|)$, or*
— *that contain a vertex of degree $\Omega(|V|)$.*

The preordered base graphs here are not equipped with a *linear* order but with an "approximation" thereof, i.e. a preorder with small colour classes (see Section 2). The CFI structures over these are also preordered, in such a way that the gadgets for vertices which share a colour class in the base graph end up in the same colour class of the CFI graph. The CFI structures we presented in Section 3 require a linear order on the base graph but the construction from [Cai et al. 1992] also works with preorders or no order at all. The crucial h.f. set construction that underlies all results from Theorem 6.3 represents the parity of the given CFI graph in such a way that it can be easily extracted from it in CPT. For these "parity-tracking" h.f. sets to be CPT-definable in the CFI graph, they have to be highly symmetric and at the same time of polynomial size, which is not easy to ensure. Therefore, pushing this idea beyond the base graphs mentioned in the above theorem, in particular to completely unordered ones, seems challenging. The question of how far exactly it can be pushed seems vital to understanding the limitations of CPT.

A related h.f. set construction has been used to not only decide the CFI query in CPT but also for canonisation and hence capturing P on certain "CFI-like" structures. As in the case of the CFI query, the CPT-definability of the relevant h.f. sets depends highly on the symmetries of the structures in question: The first canonisation result of this

kind, due to Abu Zaid, Grädel, Grohe and Pakusa [2014], requires the structures to be preordered in such a way that the size of the colour classes is bounded by a constant, and the automorphism group of each colour class is Abelian. These specific properties are satisfied for example by CFI structures over linearly ordered base graphs but also by multipedes. A more recent result due to Lichter and Schweitzer [2021] shows that canonisation is also possible if the symmetries of each colour class are dihedral groups; these are still very close to being Abelian.

THEOREM 6.4 ([ABU ZAID ET AL. 2014], [LICHTER AND SCHWEITZER 2021]). *CPT captures* P *on preordered structures with constant-size colour classes whose automorphism groups are*

— *Abelian, or*
— *dihedral.*

Since the CFI structures which separate rank logic from P have Abelian colour classes, the canonisation result also entails a separation between rank logic and CPT (to be precise, the colour class size is not bounded there, but the canonisation algorithm also works without the size limitation if the automorphism groups can be ordered in CPT [Pakusa 2015]). It is, however, not known whether CPT strictly contains rank logic or if the two logics are incomparable to each other. Showing the former would (at least implicitly) require to prove that CPT defines the CFI query also over unordered base graphs. This seems presently out of reach and may well not be true. We therefore focus now on the opposite, namely approaching the limitations of CPT.

Limitations of CPT. Unlike for FPC or linear-algebraic logic, a game characterisation for CPT is not known. The difficulty is that the expressive power of a given CPT sentence is not linked to any fixed number of pebbles but depends on the properties of the h.f. sets constructed throughout the evaluation of the sentence. The following approximate view has proven useful: Given a CPT sentence Π (involving an iteration term) and a structure $\mathfrak{A}$, the evaluation of Π on $\mathfrak{A}$ gives rise to a sequence $a_0, a_1, \ldots$ of h.f. sets over A, where each a_i is the i-th iteration stage. In general it is hard to say what the a_i may look like exactly, but we do know that

(1) for each a_i, $\mathrm{size}(a_i) \leq \mathrm{poly}(|A|)$, and
(2) each a_i is invariant under the automorphisms of $\mathfrak{A}$.

The second condition is due to CPT being a symmetric computation model. These two limitations of the constructed h.f. sets are enough to prove interesting inexpressibility results. Already in [Blass et al. 1999], an argument along these lines is given to prove that CPT without the Card-operator cannot count: without it, it is impossible to define whether the universe of the input structure has even or odd cardinality. To prove this, one considers sufficiently large structures without any relations. They are maximally symmetric, i.e. the automorphism group of such a structure with an n-element universe is $\mathbf{Sym}_n$. A combinatorial or group-theoretic analysis shows that $\mathbf{Sym}_n$-symmetric h.f. sets of size $\leq \mathrm{poly}(n)$ have a special structure and can in a certain precise sense be described with a constant number of pebbles. Therefore, on $\mathbf{Sym}_n$-symmetric structures, it is actually possible to reason about the power of CPT using standard pebble games and to prove inexpressibility results in this way. A similar argument was later used by Rossman to show that CPT cannot define the dual (i.e. the set of all hyperplanes) of a given finite vector space [2010]. This constitutes a *functional* polynomial-time problem that is provably not CPT-definable. To separate CPT from P, we are looking for a *decision* problem, however.

Again, a promising candidate seems to be the CFI query. As mentioned before, this is known to be CPT-definable on ordered or preordered base graphs with small colour

classes. The presence of such an order/preorder drastically reduces the symmetries of the structure and thus, in these cases, the collection of h.f. sets satisfying (1) and (2) is much bigger than on highly symmetric structures – rich enough to contain the rather specific and complex h.f. sets used to define the CFI query. Towards separating CPT from P, one can study the h.f. sets satisfying (1) and (2) on *unordered* base graphs with many symmetries and attempt to show that they are in a sense not powerful enough for defining the CFI query. Concretely, one would have to lift the approach outlined above to these unordered CFI graphs and prove that all h.f. sets satisfying (1) and (2) admit a certain description with only few pebbles – then the Duplicator winning strategy for the bijective pebble game on CFI graphs can be lifted to CPT. The proof of the undefinability of counting in CPT without Card exploited the specific properties of $\mathbf{Sym}_n$-symmetric h.f. sets of polynomial size; thus, we can expect that the base graphs most suitable for lower bounds should be $\mathbf{Sym}_n$-symmetric. The most obvious example would be complete graphs – but by the third item of Theorem 6.3, the CFI query over complete graphs is actually CPT-definable.

Another class of graphs whose automorphism group contains $\mathbf{Sym}_n$ (although acting on the powerset $2^{[n]}$) are the n-dimensional hypercubes. A first account of the hardness of the CFI query over unordered hypercubes was given in [Pago 2021]. There it is shown that CPT cannot define a preorder with logarithmically sized colour classes in the n-dimensional hypercubes for all $n \in \mathbb{N}$. Consequently, it is impossible to use the CPT algorithm from the second item in Theorem 6.3 for the CFI query over unordered hypercubes: There is no CPT-definable preprocessing that could be applied to these instances to make them suitable for the preorder-based CPT-algorithm. The reason is that any ordered partition of the hypercube into colour classes of logarithmic size breaks its symmetries so much that the automorphism-closure of any h.f. set representing the preorder has super-polynomial size. In other words, there is no h.f. set satisfying conditions (1) and (2) which encodes a preorder with log-size colour classes (or rather the smallest symmetry-closed set of such preorders) in the hypercube.

The CPT-algorithm for CFI that requires such a preorder is the most general one that is known. Thus, this result establishes the CFI query over unordered hypercubes as an instance that beats the current techniques. Whether it beats CPT as a whole is unknown so far. In order to prove that no CPT sentence at all defines the CFI query on hypercubes, we would have to show that *every* h.f. set over the hypercube that is "powerful enough" (i.e. admits no description with few pebbles) to distinguish CFI graphs cannot satisfy (1) and (2); in [Pago 2021] this is established for *particular* h.f. sets that encode said preorders.

A language in which this problem can arguably be approached more easily is that of circuits. It is possible to translate h.f. sets over CFI graphs into circuits with Boolean XOR gates. The conditions (1) and (2) then state that the corresponding circuit has a polynomial number of gates and is symmetric under the automorphisms of the base graph. The condition that a h.f. set is "powerful enough" to distinguish CFI graphs means in the circuit setting that there is a sufficient number of input gates whose value has an influence on the output of the XOR circuit (ultimately this represents the number of pebbles needed to describe the h.f. set). With this circuit translation, showing that CPT cannot define the CFI query reduces to proving lower bounds against these symmetric XOR circuits [Pago 2023]. To be precise, the circuit translation makes an assumption on the CPT sentences, namely that the h.f. set they construct to define the CFI query encodes the parity of the CFI graph in a certain specific and natural way (this is a technical condition on the automorphism structure of the h.f. set). This is a design pattern that is shared by all the CPT algorithms from Theorem 6.3.

Thus, a strong enough lower bound against symmetric XOR circuits with the properties mentioned above would entail that CPT cannot define the CFI query on base graphs with the same symmetries for which the circuit lower bound holds – at least not using a "parity-tracking" h.f. set, which is the only technique known so far. In [Pago 2023], a nearly sufficient lower bound against XOR circuits with the hypercube symmetries is also established. It remains for future work to strengthen this to the extent necessary for concluding the CPT lower bound. As explained, this would only rule out CPT sentences using h.f. sets with a specific structure that is particularly useful for representing parities. A further step to finally separate all of CPT from P would then be to either generalise the connection between h.f. sets and XOR circuits to *all* h.f. sets (over CFI graphs) or to show that every CPT sentence defining the CFI query can be translated into one that uses only h.f. sets with the nice specific structure.

In any case, advances towards separating CPT from P along whatever route will most likely require us to obtain a more detailed understanding of the connection between the *size* of a circuit (or circuit-like object such as a h.f. set), its *symmetries*, and its *expressive power*. The goal is to show that there are highly symmetric structures on which every polynomial-size symmetric h.f. set is "weak" in terms of the expressiveness it grants any CPT sentence constructing it.

6.2. Choiceless Polynomial Time with witnessed symmetric choice

There is a way to combine the two paradigms of witnessed symmetric choice (see Section 5) and higher order objects to obtain a logic called Choiceless Polynomial Time with witnessed symmetric choice, $\mathrm{CPT{+}WSC}$. This was defined by Lichter and Schweitzer in [2022]. Even though the limitations of CPT are still not known and thus it cannot be said whether $\mathrm{CPT{+}WSC}$ is a strict extension of CPT or not, there is a very good reason for introducing a formalism like $\mathrm{CPT{+}WSC}$: For this logic to capture P on a class of structures, it is sufficient if the logic defines the *isomorphism problem* on this class. This is striking because in general, we only know of the connection between *canonisation* and capturing P, via the Immerman-Vardi theorem. Determining whether two structures are isomorphic is a priori an easier problem than computing a canon of a given structure: If we can do the latter, we can also do the former, but the other direction is in general not known to hold. The attractive feature of $\mathrm{CPT{+}WSC}$ is that here, definability of isomorphism does in fact entail definability of canonisation.

THEOREM 6.5 ([LICHTER AND SCHWEITZER 2022]). *If* $\mathrm{CPT{+}WSC}$ *defines isomorphism on a class of structures* $\mathcal{C}$*, then* $\mathrm{CPT{+}WSC}$ *defines a canonisation of* $\mathcal{C}$*-structures and captures* P *on* $\mathcal{C}$*.*

To be precise, $\mathcal{C}$ is subject to a certain closure condition saying that the expansion of any structure in $\mathcal{C}$ with the initial segment of a linear order is still in $\mathcal{C}$. We refrain from presenting the definition of $\mathrm{CPT{+}WSC}$ here, since this is similar to $\mathrm{FPC{+}WSC}$ from Section 5, just with CPT as the base logic. The proof of the above theorem is technically quite involved and so we only mention the ingredients.

To achieve canonisation, an algorithm by Gurevich [1997] is used. This algorithm presupposes the existence of a so-called *complete invariant*: A complete invariant is a (in this case $\mathrm{CPT{+}WSC}$-definable) function I that maps structures to natural numbers in such a way that $I(\mathfrak{A}) = I(\mathfrak{B})$ if and only if $\mathfrak{A} \cong \mathfrak{B}$. Given a complete invariant, a structure $\mathfrak{A}$ can be canonised by constructing a linear order on its universe A one by one: Assuming the order constructed so far is $a_1 < a_2 < ... < a_k$, then the next element a_{k+1} to be inserted after a_k is the one that minimises $I(\mathfrak{A}, (a_1 < ... < a_{k+1}))$, i.e. the invariant of $\mathfrak{A}$ expanded with this initial segment of a linear order. In $\mathrm{CPT{+}WSC}$, this algorithm can be implemented as follows: Assuming a $\mathrm{CPT{+}WSC}$-definable complete invariant is available, firstly the set of orbits of a structure is ordered as in Gurevich's

algorithm. To also order the elements within each orbit, the choice operator is used to select in every step an arbitrary next element from the orbit. Thus the elements within each orbit are ordered arbitrarily, which is fine since they are symmetric to each other and so regardless of how the choices are resolved, one always arrives at the same canon for any two isomorphic structures.

It remains to show that if $\mathrm{CPT{+}WSC}$ defines the isomorphism problem on $\mathcal{K}$, then there also exists a $\mathrm{CPT{+}WSC}$-definable complete invariant. This is the technically involved part of the proof. It relies on an alternative formulation of CPT that was introduced by Grohe, Schweitzer and Wiebking [2021] a year earlier. They define a computation model called *Deep Weisfeiler Leman* (DWL), which is a Turing machine that receives as input a finite structure and only has isomorphism-invariant access to it (via the algebraic information associated with its stable 2-dimensional Weisfeiler Leman colouring). It was shown by [Grohe et al. 2021] that the computation model DWL is equivalent to CPT and that in a sense, DWL "computation traces" define complete invariants. The heart of this proof is to show that a DWL algorithm that decides isomorphism given a pair of structures $(\mathfrak{A}, \mathfrak{B})$ can be "decoupled" so as to operate on $\mathfrak{A}$ and $\mathfrak{B}$ separately – this is crucial because a complete invariant is a property of *one* structure, not of a pair of structures. To sum up, this yields that whenever CPT defines the isomorphism problem, then there exists also a CPT-definable complete invariant. In [Lichter and Schweitzer 2022], this result was then lifted to $\mathrm{CPT{+}WSC}$, again via a Deep Weisfeiler Leman formulation of $\mathrm{CPT{+}WSC}$.

7. OPEN PROBLEMS AND FUTURE DIRECTIONS

As we have mentioned, it is still open to separate any of the logics $\mathrm{FPC{+}WSC{+}I}$, CPT or $\mathrm{CPT{+}WSC}$ from P (or to prove that one of them captures it). For $\mathrm{FPC{+}WSC{+}I}$ and CPT there exist at least suggested approaches that can be followed. To prove lower bounds for $\mathrm{FPC{+}WSC{+}I}$, the main challenge one has to deal with seems to be understanding in how far (potentially multi-dimensional) interpretations can modify the symmetries of structures, in particular CFI graphs. For CPT, progress on size lower bounds for certain symmetric circuits (or circuit-like structures), particularly over the automorphism group of hypercubes, would be highly useful towards a separation from P. A further strengthening of the connection between the power of CPT and the complexity of such symmetric circuits may also be needed.

Apart from these lower bound challenges, there are also further potential directions towards capturing P that are worth exploring. Specifically the algebraic approach à la rank logic/linear algebraic logic could be taken further by considering not only operators over fields but over rings or even groups instead. Since the examples that separate rank logic from P are encodings of equation systems over rings, upgrading the logic in this direction would be the natural next step.

A concrete idea that seems promising is to incorporate operators over the *ring of integers* into a logic. This is motivated by an enhancement of the k-dimensional Weisfeiler Leman/k-consistency algorithms for graph isomorphism and CSP, respectively, that has recently been suggested by Ó Conghaile [2022]. These new algorithms are called *cohomological* k-WL/k-consistency and operate with formal $\mathbb{Z}$-linear combinations of local isomorphisms/homomorphisms. Despite being surprisingly simple, cohomological k-consistency is powerful enough to solve all CSPs with affine linear equations over any ring, and the cohomological k-WL algorithm can in fact distinguish the CFI structures on which rank logic and linear-algebraic logic fail. An interesting question is thus whether there is a natural version of rank logic/linear-algebraic logic over $\mathbb{Z}$ which expresses these algorithms. Since such a logic would be able to tell apart all known CFI

constructions, this would also inspire the search for new counterexamples. One possible such direction is CFI-like construction over non-Abelian groups.

Taking this further, one might also think about group-theoretic operators. All major algorithms for the graph isomorphism problem which are not based on the Weisfeiler Leman method employ group-theoretic techniques such as the Schreier-Sims algorithm. A classical example is the polynomial time algorithm for isomorphism on bounded-degree graphs due to Babai and Luks [1983]. Such algorithms are inherently non-symmetric because they require to make choices of group generators. But on the other hand, Gaussian elimination also makes symmetry-breaking choices and yet there is rank logic, which naturally imports the power of Gaussian elimination into a symmetry-invariant descriptive formalism. Is there a natural way to define a logic that allows to express, say, the Babai-Luks graph isomorphism test and related algorithms?

The linear algebraic logics $\mathrm{LA}^{\omega}(\Omega)$ are defined by allowing all operators (or generalized quantifiers) having certain strong closure conditions—in this case being closed under linear isomorphisms. This has led to other formulations of classes of quantifiers defined by their closure conditions. One recent line of research is based on quantifiers corresponding to constraint satisfaction problems and closure under polymorphisms [Hella 2023; Dawar and Hella 2024], and this merits further investigation.

REFERENCES

F. Abu Zaid, E. Grädel, M. Grohe, and W. Pakusa. 2014. Choiceless Polynomial Time on structures with small Abelian colour classes. In *Mathematical Foundations of Computer Science 2014*. Lecture Notes in Computer Science, Vol. 8634. Springer, 50–62. http://logic.rwth-aachen.de/pub/pakusa/cptcan.pdf

M. Anderson, A. Dawar, and B. Holm. 2015. Solving Linear Programs without Breaking Abstractions. *J. ACM* 62 (2015), 48:1–48:26.

V. Arvind and S. Biswas. 1987. Expressibility of First Order Logic with a Nondeterministic Inductive Operator. In *4th Annual Symp. on Theoretical Aspects of Computer Science (LNCS)*, Vol. 247. Springer, 323–335.

A. Atserias, A. Bulatov, and A. Dawar. 2009. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science* 410, 18 (2009), 1666–1683.

A. Atserias and A. Dawar. 2019. Definable inapproximability: new challenges for duplicator. *Journal of Logic and Computation* 29, 8 (2019), 1185–1210.

L. Babai and E. Luks. 1983. Canonical labeling of graphs. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. 171–183.

A. Blass. 2016. Symbioses between mathematical logic and computer science. *Annals of Pure and Applied Logic* 167 (2016), 868–878. DOI:http://dx.doi.org/10.1016/J.APAL.2014.04.018

A. Blass and Y. Gurevich. 2000. The Logic of Choice. *Journal of Symbolic Logic* 65 (2000), 1264–1310.

A. Blass and Y. Gurevich. 2005. A quick update on the open problems in Blass-Gurevich-Shelah's article "On polynomial time computations over unordered structures". (2005). https://www.microsoft.com/en-us/research/wp-content/uploads/2017/01/150a.pdf

A. Blass, Y. Gurevich, and S. Shelah. 1999. Choiceless polynomial time. *Annals of Pure and Applied Logic* 100, 1-3 (1999), 141–187.

A. Blass, Y. Gurevich, and S. Shelah. 2002. On polynomial time computation over unordered structures. *The Journal of Symbolic Logic* 67, 3 (2002), 1093–1125.

J. Cai, M. Fürer, and N. Immerman. 1992. An optimal lower bound on the number of variables for graph identification. *Combinatorica* 12 (1992), 389–410.

A. Chandra and D. Harel. 1980. Structure and complexity of relational queries. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*. IEEE, 333–347.

DOI:http://dx.doi.org/10.1109/SFCS.1980.41
A. Cobham. 1965. The Intrinsic Difficulty of Functions. In *Logic, Methodology and Philosophy of Science II*. North-Holland, 24–30.
A. Ó Conghaile. 2022. Cohomology in Constraint Satisfaction and Structure Isomorphism. In *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria (LIPIcs)*, Vol. 241. 75:1–75:16. DOI:http://dx.doi.org/10.4230/LIPICS.MFCS.2022.75
A. Dawar. 1995. Generalized Quantifiers and Logical Reducibilities. *Journal of Logic and Computation* 5, 2 (1995), 213–226.
A. Dawar. 2015. The nature and power of fixed-point logic with counting. *ACM SIGLOG News* 2, 1 (2015), 8–21.
A. Dawar, E. Grädel, B. Holm, E. Kopczynski, and W. Pakusa. 2013. Definability of linear equation systems over groups and rings. *Logical Methods in Computer Science* 9 (2013). DOI:http://dx.doi.org/10.2168/LMCS-9(4:12)2013
A. Dawar, E. Grädel, and W. Pakusa. 2019. Approximations of Isomorphism and Logics with Linear-Algebraic Operators. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 132. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 112:1–112:14. DOI:http://dx.doi.org/10.4230/LIPIcs.ICALP.2019.112
A. Dawar, M. Grohe, B. Holm, and B. Laubner. 2009. Logics with rank operators. In *2009 24th Annual IEEE Symposium on Logic In Computer Science*. IEEE, 113–122.
A. Dawar, E. Grädel, and M. Lichter. 2023. Limitations of the invertible-map equivalences. *Journal of Logic and Computation* 33 (2023), 961–969. DOI:http://dx.doi.org/10.1093/logcom/exac058
A. Dawar and L. Hella. 2024. Quantifiers Closed Under Partial Polymorphisms. In *32nd EACSL Annual Conference on Computer Science Logic (LIPIcs)*, Vol. 288. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 23:1–23:19. DOI:http://dx.doi.org/10.4230/LIPICS.CSL.2024.23
A. Dawar and B. Holm. 2012. Pebble games with algebraic rules. In *International Colloquium on Automata, Languages, and Programming*. Springer, 251–262.
A. Dawar and D. Richerby. 2003. A fixed-point logic with symmetric choice. In *International Workshop on Computer Science Logic*. Springer, 169–182.
A. Dawar and D. Richerby. 2007. The Power of Counting Logics on Restricted Classes of Finite Structures. In *CSL 2007:Computer Science Logic (LNCS)*, Vol. 4646. Springer, 84–98.
A. Dawar, D. Richerby, and B. Rossman. 2008. Choiceless Polynomial Time, Counting and the Cai–Fürer–Immerman graphs. *Annals of Pure and Applied Logic* 152, 1-3 (2008), 31–50.
A. Dawar and G. Wilsenach. 2020. Symmetric Arithmetic Circuits. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 168. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 36:1–36:18. DOI:http://dx.doi.org/10.4230/LIPIcs.ICALP.2020.36
A. Dawar and G. Wilsenach. 2021. Symmetric circuits for rank logic. *ACM Transactions on Computational Logic (TOCL)* 23, 1 (2021), 1–35.
H. Ebbinghaus and J. Flum. 1999. *Finite Model Theory*. Springer.
F. Gire and H.K. Hoang. 1998. An Extension of Fixpoint Logic with a Symmetry-Based Choice Construct. *Information and Computation* 144, 1 (1998), 40–65. DOI:http://dx.doi.org/https://doi.org/10.1006/inco.1998.2712
E. Grädel and M. Otto. 1993. Inductive definability with counting on finite structures. In *Computer Science Logic: 6th Workshop, CSL'92 San Miniato, Italy, September*

28–October 2, 1992 Selected Papers 6. Springer, 231–247.
E. Grädel and W. Pakusa. 2019. Rank logic is dead, long live rank logic! *The Journal of Symbolic Logic* 84, 1 (March 2019). https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/rank-logic-is-dead-long-live-rank-logic/F161E336281F2B67E8B3E4CDA5614933
M. Grohe. 2008. The quest for a logic capturing PTIME. In *2008 23rd Annual IEEE Symposium on Logic in Computer Science*. IEEE, 267–271. DOI: http://dx.doi.org/10.1109/LICS.2008.11
M. Grohe. 2017. *Descriptive complexity, canonisation, and definable graph structure theory.* Cambridge University Press.
M. Grohe, P. Schweitzer, and D. Wiebking. 2021. Deep Weisfeiler Leman. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2600–2614.
Y. Gurevich. 1988. Logic and the Challenge of Computer Science. In *Current Trends in Theoretical Computer Science*. Computer Science Press.
Y. Gurevich. 1997. From invariants to canonization. *Bulletin of the European Association of Theoretical Computer Science (BEATCS)* 63 (1997), 115–119.
Y. Gurevich and S. Shelah. 1996. On finite rigid structures. *The Journal of Symbolic Logic* 61, 2 (1996), 549–562.
L. Hella. 1996. Logical hierarchies in PTIME. *Information and Computation* 129, 1 (1996), 1–19.
L. Hella. 2023. The Expressive Power of CSP-Quantifiers. In *31st EACSL Annual Conference on Computer Science Logic (LIPIcs)*, Vol. 252. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 25:1–25:19. DOI: http://dx.doi.org/10.4230/LIPICS.CSL.2023.25
N. Immerman. 1986. Relational Queries computable in Polynomial Time. *Information and Control* 68 (1986), 86–104.
N. Immerman and E. Lander. 1990. Describing graphs: A first-order approach to graph canonization. In *Complexity theory retrospective*. Springer, 59–81.
A. C. Leisenring. 1969. *Mathematical Logic and Hilbert's ε-Symbol*. MacDonald Technical and Scientific.
M. Lichter. 2022. Witnessed Symmetric Choice and Interpretations in Fixed-Point Logic with Counting. (2022). https://arxiv.org/abs/2210.07869
M. Lichter. 2023. Separating Rank Logic from Polynomial Time. *J. ACM* (2023), 14:1–14:53. https://doi.org/10.1145/3572918
M. Lichter and P. Schweitzer. 2021. Canonization for Bounded and Dihedral Color Classes in Choiceless Polynomial Time. In *29th EACSL Annual Conference on Computer Science Logic (CSL 2021) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 183. 31:1–31:18. DOI: http://dx.doi.org/10.4230/LIPIcs.CSL.2021.31
M. Lichter and P. Schweitzer. 2022. Choiceless Polynomial Time with Witnessed Symmetric Choice *(LICS '22)*. Association for Computing Machinery, Article 30, 13 pages. https://doi.org/10.1145/3531130.3533348
M. Otto. 1997. *Bounded Variable Logics and Counting*. Lecture Notes in Logic, Vol. 9. Springer.
B. Pago. 2021. Choiceless Computation and Symmetry: Limitations of Definability. In *29th EACSL Annual Conference on Computer Science Logic (CSL 2021) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 183. 33:1–33:21. DOI: http://dx.doi.org/10.4230/LIPIcs.CSL.2021.33
B. Pago. 2023. Lower Bounds for Choiceless Polynomial Time via Symmetric XOR-Circuits. In *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 272. 73:1–73:15. DOI: http://dx.doi.org/10.4230/LIPIcs.MFCS.2023.73
W. Pakusa. 2015. *Linear Equation Systems and the Search for a Logical Characterisation*

of Polynomial Time. Ph.D. Dissertation. RWTH Aachen.
W. Pakusa, S. Schalthöfer, and E. Selman. 2018. Definability of Cai-Fürer-Immerman problems in Choiceless Polynomial Time. *ACM Transactions on Computational Logic (TOCL)* 19, 2 (2018), 1–27. DOI: http://dx.doi.org/10.1145/3154456
B. Rossman. 2010. Choiceless Computation and Symmetry. In *Fields of Logic and Computation*. Springer, 565–580.
S. Schalthöfer. 2020. *Choiceless Computation and Logic*. Ph.D. Dissertation. RWTH Aachen.
M. Vardi. 1982. The complexity of relational query languages. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. 137–146.

S&P

SECURITY AND PRIVACY COLUMN

STEVE KREMER,
LORIA, Inria Nancy-Grand Est, Université de Lorraine
Steve.Kremer@inria.fr



Cryptographic protocols are really hard to design: they mix difficulties stemming from distributed systems, rely on cryptographic primitives and need to provide complex guarantees in the presence of an arbitrary attacker running in parallel. Rigorous security proofs are basically the only hope to avoid design flaws as witnessed by numerous examples. This has been recognized since the early 80's and two main approaches have been actively developed since. The *computational model* is rooted in complexity theory and relies on a reduction of the security of the protocol to a security assumption of a cryptographic primitive which itself may rely on the hardness of a computational problem. Attackers are any probabilistic polynomial time Turing Machines (PPTMs). A security proof in this model provides strong guarantees. However, getting proofs in such a detailed model correct is challenging. The *symbolic model* takes a different approach: messages are modeled as first-order logic terms and cryptographic primitives such as encryption schemes are idealized. Typically, only the intended functionality of an encryption is modeled by an equation such as $\mathsf{dec}(\mathsf{enc}(\mathsf{m},\mathsf{k}),\mathsf{k}) = \mathsf{m}$ to state that decryption and encryption cancel out when the same key k is used. Crucially, only the properties that are explicitly stated hold and the adversary actions are restricted to those. At the cost of being more abstract this approach is amenable to automation and tools nowadays succeed in proving industrial protocols, such as TLS or Signal.

In 2012, Bana and Comon introduced a new idea that aims at getting the advantages of both approaches: the computationally complete symbolic adversary (CCSA). The idea is to design a first-order logic where instead of modeling the actions that are allowed by the attacker, they only state the axioms that an attacker may *not* violate. By showing that these axioms are sound in a computational interpretation of the first-order terms one recovers the strong guarantees of the computational model while protocol proofs are carried out in first-order logic. This approach has been pushed further in recent years and implemented in an interactive theorem prover dubbed the Squirrel prover. In this edition, several of the designers of Squirrel describe the most recent, higher-order version of the underlying logic and the challenges in its design.

The Squirrel Prover and its Logic

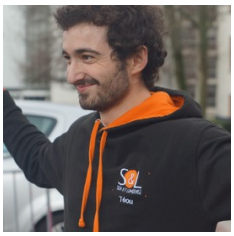
David Baelde, Univ Rennes, CNRS, IRISA



Stéphanie Delaune, CNRS, Univ Rennes, IRISA



Charlie Jacomme, Université de Lorraine, LORIA, Inria Nancy Grand-Est



Adrien Koutsos, Inria Paris



Joseph Lallemand, CNRS, Univ Rennes, IRISA



The Squirrel system [Baelde et al. 2021] is an interactive prover for the verification of cryptographic protocols. It relies on a dedicated higher-order logic and provides security guarantees akin to those classically obtained by cryptographers, against attackers modeled as arbitrary probabilistic polynomial-time Turing machines. In this paper, we provide a high-level introduction to the logic behind Squirrel, and briefly describe some of the interesting technical challenges encountered during its construction.

1. INTRODUCTION

Security protocols are widely used today to secure transactions that take place over public channels like the Internet. Common uses include the secure transfer of sensitive information such as credit card numbers, or user authentication on a system. Because of their presence in many widely used applications (*e.g.* electronic commerce, government-issued ID), developing methods and tools to verify security protocols has become an important research challenge. Such tools help increase our trust in protocols, and hence on the applications that rely on them.

Formal methods have brought various approaches to prove that cryptographic protocols indeed guarantee the expected security properties. An effective approach in this area of research consists in modeling cryptographic messages as first-order terms, together with an equational theory that represents attacker capabilities. This idea, originally proposed in [Dolev and Yao 1981], has been refined over the years, resulting in a variety of so-called *symbolic* models. These models encompass broad categories of attackers and facilitate the automated verification of protocols. They have led to the development of successful tools such as ProVerif [Blanchet 2001] and Tamarin [Meier et al. 2013]. However, it is important to note that security in a symbolic model does not necessarily imply security in the cryptographers' standard model, called the *computational* model. In that model, attackers are represented by probabilistic polynomial-time Turing machines (PPTMs), and one proves that a protocol is indistinguishable from an idealized, obviously secure version of it. Verification techniques for the computational model, though crucially needed, often exhibit less flexibility or automation compared to ones for symbolic models. As an illustration, secret keys are faithfully modeled in the computational model as long bitstrings that are drawn uniformly at random, whereas they are modeled using abstract names in symbolic models. In symbolic models, two distinct secret keys are represented by different names, which cannot be equal. However, in the computational model, as in reality, it is possible (although unlikely) that the sampled bitstrings are equal. In this column, we present a recent logic-based method for verifying cryptographic protocols in the computational model, and some practical aspects of its implementation in the Squirrel tool [Baelde et al. 2021; Baelde et al. 2023]. This system is built on the Computationally Complete Symbolic Attacker (CCSA) approach of [Bana and Comon-Lundh 2012; Bana and Comon-Lundh 2014], which relies on the symbolic setting of logic, but avoids the limitations of the symbolic models mentioned above. Instead of modeling attacker capabilities by rules stating what the adversary can do, the CCSA method relies on the specification of what the attacker *cannot* do. Starting from the security properties of cryptographic primitives, one derives rules that articulate which pairs of message sequences are indistinguishable. These rules are proved sound w.r.t. the interpretation of terms as PPTMs. Therefore, a proof of a security property using these rules implies security in the computational model under the initial cryptographic assumptions. The CCSA logic was later extended into a meta-logic, which served as basis for the first version of Squirrel [Baelde et al. 2021], before being generalized to a fully-fledged higher-order logic in [Baelde et al. 2023]. The Squirrel tool is a proof assistant developed in a collaborative effort, which has been successfully used on a variety of case studies [Baelde et al. 2022; Comon et al. 2020; Cremers et al. 2022]. Instructions for installing the system are available on its website, together with a user manual, tutorials, and an in-browser interface for playing with the tool without installing it:

https://squirrel-prover.github.io



We first provide, in Section 2, an introduction to the computational model and the CCSA approach, showing in particular how cryptographic assumptions translate into logical rules. We elaborate on this in Section 3 to show how protocols can be modeled in CCSA logic, discussing a subtle issue w.r.t. the intended notion of security. Finally, Section 4 formally defines the higher-order CCSA logic, on which Squirrel is based, and discusses some of its interesting technical features: the distinction between local and global formulas, the subtleties tied to reasoning about probabilistic objects, and the key notion of bi-deduction for reasoning about computational indistinguishability.

Related work. Squirrel is only a recent addition to the list of available systems for verifying cryptographic protocols. We have mentioned above some tools that provide guarantees in symbolic models; we now briefly present the tools providing guarantees in the computational model. Several such systems exist, based on different approaches. The earliest one is CryptoVerif [Blanchet 2008], which mechanizes proofs based on high-level game transformations, following the style of pen-and-paper proofs. The most prominent system today is probably EasyCrypt [Barthe et al. 2011], which is a proof assistant also featuring higher-order logic. It notably embeds a domain-specific *probabilistic relational Hoare logic* [Barthe et al. 2009], which can capture cryptographic game transformations. This design makes it possible to carry out most cryptographic arguments within EasyCrypt. Other systems are currently being developed, with various goals: notably CryptHOL [Basin et al. 2020], which explores an alternative modeling technique in Isabelle/HOL, and F$\star$ [Swamy et al. 2016], which is a general-purpose program verification framework based on refinement types that can be used (via external arguments) to provide computational security guarantees. Those approaches can be compared on several criteria [Barbosa et al. 2021].

Here, we only compare with the closest two tools, and only with respect to three criteria, to highlight differences: modeling, automation, and proof methodology. Regarding modeling, CryptoVerif and Squirrel have a similar level of detail and expressivity, although CryptoVerif supports a larger range of cryptographic assumptions. EasyCrypt is more expressive, with a higher level of detail, at the cost of modeling overhead, and thus better suited for proving properties of primitives. Protocol specifications in CryptoVerif and Squirrel are given in a process algebra, whereas protocols are encoded in EasyCrypt as APIs, which is inconvenient for interactive protocols. Finally, CryptoVerif does not support stateful protocols, while Squirrel and EasyCrypt do.

Overall, the current level of automation of Squirrel sits somewhere between CryptoVerif, which can apply cryptographic arguments automatically, although it often requires hints about which game transformations to use, and EasyCrypt, which does not automatically apply cryptographic arguments.

Finally, the most important difference between Squirrel and the other tools is the associated proof methodology: CryptoVerif relies on game transformations and EasyCrypt performs Hoare-style proofs of programs, while Squirrel reasons over execution traces of protocols.

2. A BASIC INDISTINGUISHABILITY LOGIC

We introduce in this section the CCSA approach, which is designed to reason on cryptographic protocols, *i.e.* concurrent programs relying on cryptographic primitives to achieve some functionality while running in a malicious environment. To simplify our exposition, we delay the treatment of general protocols to Section 3, and restrict our presentation to simple sequential cryptographic games for the time being. These games are used by cryptographers to express security properties of cryptographic primitives against attackers modeled as arbitrary polynomial-time computations. A key concept about games is *computational indistinguishability*: roughly, two games are

Game $\mathcal{G}$:

```
1 : sk ←$ {0,1}^η;
2 : m ← A();
3 : t ← enc(m, sk);
4 : x ← A(t);
5 : return x
```

A secret key sk of length η is randomly sampled. The interactive attacker $\mathcal{A}$ is asked to provide a message m which is then encrypted (using sk) and sent back to $\mathcal{A}$. Afterwards, $\mathcal{A}$ returns a second message x which is the final output of the game.

Fig. 1. An example cryptographic game.

indistinguishable if any polynomial-time adversary has at-best a negligible probability in distinguishing them. In the CCSA approach, we use first-order logic to establish computational indistinguishability: we first model the messages derived during a game's execution as first-order terms, and then reason about computational indistinguishability in first-order logic, using inference rules derived from cryptographic assumptions.

A key aspect of the logic is that it abstracts cryptographic arguments as much as possible. Notably, its syntax does not mention the length of keys or the value of probabilities. Further, reduction-based arguments are implicit: they justify the soundness of the proof rules, but the code underlying reductions is never explicitly stated.

2.1. The core of the logic

Security in the real world is often conditional, as there is usually a small but non-zero probability that the adversary manages to break the security of a system. Typically, one cannot rule out that a lucky attacker guesses a secret key. To discard such unlikely cases, security is considered up to a *negligible* probability of attack. A function $f : \mathbb{N} \to [0, 1]$ is *negligible* if it is asymptotically lower than the inverse of any polynomial, *i.e.* if $\forall\, k \in \mathbb{N}.\ \exists n_0.\ \forall n \geq n_0.\ f(n) \leq \frac{1}{n^k}$. Conversely, f is *overwhelming* if $1 - f$ is negligible. In what follows, we will thus use a *security parameter* $\eta \in \mathbb{N}$ which can be, *e.g.*, the length of the secret keys, and security must hold with *overwhelming* probability w.r.t. η.

In the cryptographic literature, security properties are commonly specified using so-called *games*, in which an adversary tries to mount an attack: the game might represent an adversary attempting to guess a secret value, to forge a signature, ... Games are usually written in pseudo-code using an imperative style, as shown in Figure 1. The statement $x \xleftarrow{\$} D$ stores the result of a random sampling following distribution D into variable x: *e.g.* $\mathsf{sk} \xleftarrow{\$} \{0,1\}^\eta$ uniformly samples a secret key sk of η bits. Games are parameterized by an abstract *interactive attacker*, denoted by $\mathcal{A}$. The statement $o \leftarrow \mathcal{A}(i)$ calls the adversary $\mathcal{A}$ on input i, and stores its answer in variable o. The adversary is stateful, retaining information across its invocations. Standard assignments are denoted in the same way: $x \leftarrow e$ stores the result of evaluating e into variable x.

Describing cryptographic games as probabilistic imperative programs lets cryptographers rely on the reader's intuitive understanding of their semantics, and avoids introducing convoluted execution models. However, this comes at a cost: formally reasoning over such programs can be difficult (one needs to deal with statefulness, loop invariants, probabilistic dependencies, ...); moreover, encoding complex protocols as imperative programs is possible but not natural, and complicates security proofs.

Cryptographic games in CCSA. In the CCSA approach, we do not explicitly represent games. Instead, we shall only represent the messages computed at different points in a game, using first-order terms. Those are pure, unlike the stateful games, and thus easier to reason about. Our terms are built using:

— *honest function symbols* which represent the various primitives used to compute messages (*e.g.* pairing, encryption);
— *attacker function symbols*, noted att_i for $i \in \mathbb{N}$, modeling arbitrary computations performed by adversaries;
— *name symbols*, representing the sources of randomness: essentially, a name is a pointer to a memory cell holding a value sampled at random before the game starts.

Example 2.1. Let us illustrate this approach by modeling the final result of the game of Figure 1 as a single term. The sampling of the secret key is modeled by a name symbol sk. The first call to the attacker $\mathcal{A}()$ is modeled by the term $\mathsf{att}_0()$. In the imperative game modeling style, internal functions (*e.g.* the encryption enc) can be probabilistic, but this is not so in our approach, where we precisely track probabilistic dependencies. Thus, we explicitly specify the randomness used by the encryption function enc, by introducing a name symbol r and using the term $\mathsf{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})$ to model t in the game. Then, the second call to the adversary can be modeled using a different adversarial function symbol, as: $\mathsf{att}_1(\mathsf{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk}))$. Importantly, we have modeled two successive calls to the stateful attacker $\mathcal{A}$ in the game by two pure functions (att_0 and att_1). This is without loss of generality, as any state computed during the first call to $\mathcal{A}$ and used in the second call can be recomputed when modeling the second call as att_1.

Models of the logic. Defining the logic's semantics in order to have a faithful translation from games to terms requires a bit of care. The interpretation of a term is parameterized by the security parameter $\eta \in \mathbb{N}$, as well as two explicit sources of randomness provided by a pair $\rho = (\rho_h, \rho_a)$ of tapes filled with random bits. A model $\mathbb{M}$ of our logic must:

— provide for each honest function symbol f its interpretation as a PPTM $\mathcal{M}_{\mathsf{f}}$;
— associate to each name n a unique sub-sequence of η bits in ρ_h using a PPTM $\mathcal{M}_{\mathsf{n}}$ with access to ρ_h, such that distinct names use disjoint parts of ρ_h;
— interpret any adversarial function symbol att as a PPTM $\mathcal{M}_{\mathsf{att}}$ which can only access the random tape ρ_a (but not ρ_h).

For the sake of simplicity, we forced names to be interpreted as uniform and independent random samplings in $\{0,1\}^\eta$.

Our models are first-order models with a tailored interpretation domain. Thus, terms can be interpreted in $\mathbb{M}$ using the standard semantics of first-order logic. In our specific setting, a term t is interpreted as a function associating, to each value of the security parameter η and random tapes ρ, the interpretation $[\![t]\!]_{\mathbb{M}}^{\eta,\rho}$. More precisely, the function $\eta, \rho \mapsto [\![t]\!]_{\mathbb{M}}^{\eta,\rho}$ is computed by a PPTM.

For example, the interpretation $[\![\mathsf{n}]\!]_{\mathbb{M}}^{\eta,\rho}$ of a name symbol n is computed by $\mathcal{M}_{\mathsf{n}}(\eta, \rho_h)$, and an adversarial computation is computed by $[\![\mathsf{att}_i(t)]\!]_{\mathbb{M}}^{\eta,\rho} = \mathcal{M}_{\mathsf{att}_i}(\eta, [\![t]\!]_{\mathbb{M}}^{\eta,\rho}, \rho_a)$. Importantly, this interpretation is compositional and makes all probability dependencies explicit, which will facilitate reasoning over cryptographic messages.

Example 2.2. Coming back to $\mathcal{G}$ from Figure 1, the value in the variable x at the end of $\mathcal{G}$'s execution follows the same probability distribution as

$$[\![\mathsf{att}_1(\mathsf{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk}))]\!]_{\mathbb{M}}^{\eta,\rho}$$

where the tapes in ρ are sampled at random, and where $\mathbb{M}$ interprets the encryption and attacker functions as does $\mathcal{G}$.

Using Turing machines to interpret terms gives us a high level of expressivity. For instance, conditional branching in games can be internalized in the first-order terms using an $(\mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ e)$ function symbol whose interpretation is forced to be the natural one in all models.

Example 2.3. Consider the following game:

Game $\mathcal{G}_{\mathsf{sk}}$

```
1 : sk ←$ {0,1}^η;
2 : m ← A();
3 : if m = sk then
4 :     return 0
5 : else
6 :     return 1
```

This game samples a fresh secret key and queries the attacker to obtain a guess m for the key. The game outputs 0 if the attacker correctly guessed the key, and 1 otherwise. The message returned at the end of the game can be represented by the following term:

$$\phi_{\mathsf{sk}} \stackrel{\text{def}}{=} \mathsf{if}\ (\mathsf{att}_0() = \mathsf{sk})\ \mathsf{then}\ 0\ \mathsf{else}\ 1.$$

2.2. The computational indistinguishability predicate $\sim$

One of the most commonly used notions to define cryptographic properties is *computational indistinguishability*. Roughly, indistinguishability is expressed as a guessing game in which an adversary must figure out which one of two scenarios $\mathcal{G}_I$ or $\mathcal{G}_R$ it is interacting with. More precisely, two games $\mathcal{G}_I$ and $\mathcal{G}_R$ are indistinguishable, which we write $\mathcal{G}_I \approx \mathcal{G}_R$, when any PPTM adversary has at-best a negligible probability of guessing whether it is interacting with $\mathcal{G}_I$ or $\mathcal{G}_R$. Formally, we define the *advantage* of the attacker $\mathcal{A}$ as the probability that it makes the two games behave differently. We then require this advantage to be negligible in the security parameter η:

$$\forall \mathcal{A} \in \mathsf{PPTM}.\ \ \eta \mapsto |\Pr(\mathcal{G}_I(\mathcal{A}) = 1) - \Pr(\mathcal{G}_R(\mathcal{A}) = 1)| \text{ is negligible.}$$

Typically, the game $\mathcal{G}_R$ will correspond to an attack against the real primitives, while $\mathcal{G}_I$ will represent an attack against an idealized implementation of the primitives for which security is obvious.

Example 2.4. Coming back to Example 2.3, the indistinguishability $\mathcal{G}_{\mathsf{sk}} \approx (\mathbf{return}\ 1)$ states that the probability that the adversary $\mathcal{A}$ guesses the secret key sk is negligible. In that case, a simple probabilistic independence argument can be used to show that the probability that $\mathcal{A}$ computes sk is $\frac{1}{2^\eta}$. Thus, the indistinguishability holds.

Squirrel's logic relies on a predicate $\sim$ to represent computational indistinguishability. Formally, for two terms u, v, the predicate $u \sim v$ holds in a model $\mathbb{M}$ if $\mathcal{G}_u^{\mathbb{M}} \approx \mathcal{G}_v^{\mathbb{M}}$, where $\mathcal{G}_t^{\mathbb{M}}$ is the game where the adversary is provided with $[\![t]\!]_{\mathbb{M}}^{\eta,\rho}$ and must produce a bit b, which the game returns. In other words, $u \sim v$ in $\mathbb{M}$ when:

$$\forall \mathcal{A} \in \mathsf{PPTM}.\ \ \eta \mapsto |\Pr_\rho(\mathcal{A}([\![u]\!]_{\mathbb{M}}^{\eta,\rho}, \rho_a) = 1) - \Pr_\rho(\mathcal{A}([\![v]\!]_{\mathbb{M}}^{\eta,\rho}, \rho_a) = 1)| \text{ is negligible.}$$

Example 2.5. Reusing $\mathcal{G}_{\mathsf{sk}}$ and ϕ_{sk} from Example 2.3, checking $\phi_{\mathsf{sk}} \sim 1$ amounts to verifying that $\mathcal{G}_{\mathsf{sk}} \approx (\mathbf{return}\ 1)$.

In the example above, we have used the indistinguishability predicate on booleans. In that case, the semantics of $\sim$ can be restated in a simpler way, without a quantification over all distinguishers $\mathcal{A}$: it simply means that the two booleans have the same probability of being 1. The general semantics of $\sim$ becomes useful, though, to reason on it by decomposing terms: for instance, it allows us to derive $f(u) \sim f(v)$ from $u \sim v$ for any u and v of arbitrary types. Indeed, the existence of a PPTM distinguishing between $f(u)$ and $f(v)$ implies the existence of a PPTM distinguishing between u and v: the latter distinguisher is obtained by composing the former with the PPTM computing f.

Logical rules. We now present some rules that can be used to reason over $\sim$. We intuitively describe some of those rules here, starting with the simpler rules that do not rely on any security assumptions over the primitives.

Example 2.6. The following formulas are valid:

$$\textsc{Refl}\ \frac{}{u \sim u} \qquad \textsc{Rewrite}\ \frac{(u = v) \sim \mathsf{true} \qquad C[v] \sim w}{C[u] \sim w} \qquad \textsc{Fresh}\ \frac{}{(t \neq \mathsf{n}) \sim \mathsf{true}}$$

when t is a ground term in which n does not occur

Fig. 2. Basic inference rules over $\sim$.

(1) $t \sim t$ for any term t; indistinguishability is reflexive.
(2) $\mathsf{n} \sim \mathsf{n}'$ for any names n, n'; two uniform random samplings are indistinguishable.
(3) $(\mathsf{n}, \mathsf{n}) \not\sim (\mathsf{n}, \mathsf{n}')$; the attacker sees, in one case, the same value twice, and in the other, two distinct values (with overwhelming probability). Notice that this implies that $\sim$ does not lift to an arbitrary context: we have seen that $\mathsf{n} \sim \mathsf{n}'$, but this example shows that we do not have for all context C that $C[\mathsf{n}] \sim C[\mathsf{n}']$.
(4) $(\mathsf{if}\ \mathsf{true}\ \mathsf{then}\ u\ \mathsf{else}\ v = u) \sim \mathsf{true}$ for any terms u, v; more generally, any term occurring in an indistinguishability can be rewritten into an equal term.
(5) $(\mathsf{n} \neq \mathsf{n}') \sim \mathsf{true}$; names, *i.e.* random samplings, collide with negligible probability.

We provide a first set of inference rules in Figure 2. REFL corresponds to the reflexivity of $\sim$, and REWRITE allows replacing two terms that are equal with overwhelming probability in any context. Finally, FRESH exploits the fact that a term syntactically contains all its probabilistic dependencies: if a name n does not occur in a term t, then n is a uniform random sampling independent of t, and thus $t = \mathsf{n}$ can only be true with negligible probability.

Example 2.7. Let us go back to Example 2.5 and the formula $\phi_{\mathsf{sk}} \sim 1$. We assume an additional proof rule $\textsc{Simpl}_{\equiv}$ that allows to replace a term with another one equal to it with probability one, which we note $\equiv$, and where for instance, $v \equiv (\mathsf{if}\ \mathsf{false}\ \mathsf{then}\ u\ \mathsf{else}\ v)$. We can prove our goal with the following derivation:

$$\textsc{Rewrite}\ \frac{\textsc{Simpl}_{\equiv}\ \dfrac{\textsc{Fresh}\ \dfrac{}{(\mathsf{att}_0() \neq \mathsf{sk}) \sim \mathsf{true}}}{((\mathsf{att}_0() = \mathsf{sk}) = \mathsf{false}) \sim \mathsf{true}} \qquad \textsc{Simpl}_{\equiv}\ \dfrac{\textsc{Refl}\ \dfrac{}{1 \sim 1}}{(\mathsf{if}\ \mathsf{false}\ \mathsf{then}\ 0\ \mathsf{else}\ 1) \sim 1}}{(\mathsf{if}\ \mathsf{att}_0() = \mathsf{sk}\ \mathsf{then}\ 0\ \mathsf{else}\ 1) \sim 1}$$

Here, we use twice $\textsc{Simpl}_{\equiv}$: once with the previously mentioned equality over a false conditional, and once using the fact that $((\mathsf{att}_0() = \mathsf{sk}) = \mathsf{false}) \equiv (\mathsf{att}_0() \neq \mathsf{sk})$.

2.3. Unforgeability of hash functions: EUF-CMA

To reason over cryptographic games, a final piece of the puzzle is missing: assumptions over cryptographic primitives. For each primitive (a cipher, a signature, a hash, ...), there is a usual set of cryptographic assumptions made over it. Such assumptions are expressed as an indistinguishability between two games, and cryptographic proofs take the form of reduction-based arguments. To prove that the security of the primitive, for instance expressed as $P_R \approx P_I$, implies the security of some game $\mathcal{G}_R \approx \mathcal{G}_I$, we prove the contrapositive: assuming that there is a distinguisher against $\mathcal{G}_R \approx \mathcal{G}_I$, we build a distinguisher against $P_R \approx P_I$. In our logic, we hide those reduction-based arguments behind dedicated rules, one for each cryptographic assumption.

Consider the example of a keyed hash function $\mathsf{h}(x, \mathsf{sk})$ and of the EUF-CMA assumption. Intuitively, a keyed hash function is a function that takes a message and a key, and produces a short unpredictable value. More precisely, if a key sk is secret and randomly sampled, then an attacker has a negligible probability of guessing the hash $\mathsf{h}(m, \mathsf{sk})$ of any m, unless the attacker was directly given this value. Such a hash function is called *unforgeable*, and this is formalized in the $\text{EUF-CMA}_{\mathsf{h}}$ cryptographic game:

EUF-CMA$_{\mathsf{h}}$	Hashing Oracle $O(x)$
1 : $\mathsf{sk} \xleftarrow{\$} \{0,1\}^{\eta}$;	1 : $\mathcal{L} \leftarrow \{x\} \cup \mathcal{L}$;
2 : $\mathcal{L} \leftarrow \emptyset$;	2 : **return** $\mathsf{h}(x, \mathsf{sk})$
3 : $t, m \leftarrow \mathcal{A}^{O}()$;	
4 : **return** $(t = \mathsf{h}(m, \mathsf{sk}) \wedge m \notin \mathcal{L})$	

This game asks the attacker to return a pair of values t, m such that $t = \mathsf{h}(m, \mathsf{sk})$. The attacker is allowed to interact with the hashing oracle O, but it must return a m that was never queried to O. This is enforced by storing in a set $\mathcal{L}$ all oracle inputs.

We can express the security of the hash function by assuming that for all attackers $\mathcal{A}$, EUF-CMA$_{\mathsf{h}} \approx (\mathbf{return}\ \mathsf{false})$. Our goal is now to derive an inference rule which is sound under this assumption.

As a warm-up, we restrict ourselves to models where the interpretation of h satisfies the EUF-CMA assumption, and analyze the validity of a few statements:

— $(\mathsf{h}(0, \mathsf{sk}) = \mathsf{att}_0()) \sim \mathsf{false}$ is valid. Otherwise, we would have a model (and thus an attacker) able to output with non-negligible probability the value of $\mathsf{h}(0, \mathsf{sk})$ without having access to any other hash values. This attacker would trivially break EUF-CMA$_{\mathsf{h}}$, by directly outputting $0, \mathsf{h}(0, \mathsf{sk})$.
— $(\mathsf{h}(0, \mathsf{sk}) = \mathsf{att}_1(\mathsf{sk})) \sim \mathsf{false}$ is not valid, because the secret key is leaked to the attacker. As such, the computation of $\mathsf{att}_1(\mathsf{sk})$ cannot be seen as a computation made by an adversary $\mathcal{A}^O$ in the EUF-CMA$_{\mathsf{h}}$ game, and the assumption does not apply.
— $(\mathsf{h}(0, \mathsf{sk}) = \mathsf{att}_1(\mathsf{h}(1, \mathsf{sk}))) \sim \mathsf{false}$ is valid. Otherwise, we would have a model and thus an attacker breaking EUF-CMA$_{\mathsf{h}}$: the attacker would compute $\mathsf{h}(1, \mathsf{sk})$ with a call to oracle $O(1)$, after which $\mathcal{L} = \{1\}$, and return $\mathsf{att}_1(\mathsf{h}(1, \mathsf{sk}))$, which is the hash of $0 \notin \mathcal{L}$.

To generalize this, we need to answer the following question: under which conditions over t and m is $(\mathsf{h}(m, \mathsf{sk}) = t) \sim \mathsf{false}$ a valid formula? We answer this by providing sufficient conditions under which the terms t and m can be produced by some attacker interacting with the EUF-CMA$_{\mathsf{h}}$ game in such a way that m is never queried to O.

The first main condition is that all syntactic occurrences of sk in t and m must be as a key to h (*i.e.* all occurrences are of the form $\mathsf{h}(u, \mathsf{sk})$ for some u). Otherwise, the terms cannot be simulated, as the adversary $\mathcal{A}$ against EUF-CMA$_{\mathsf{h}}$ does not know the key.

Once we know that sk is only used in key position, it is easy to syntactically track in t and m what will be computed using queries to O: it is precisely the set of subterms of the form $\mathsf{h}(x, \mathsf{sk})$, *i.e.* $S^{\mathsf{h}}_{\mathsf{sk}}(t, m) = \{\mathsf{h}(u, \mathsf{sk}) \mid \mathsf{h}(u, \mathsf{sk})$ subterm of t or $m\}$. Using this set, we can then define a formula, inside our logic, expressing that all previously seen hash values are distinct from m. Those two conditions are sufficient to obtain a rule enabling us to use EUF-CMA to establish indistinguishabilities in the logic:

$$\textsc{Euf-Cma}\ \frac{\bigwedge_{\mathsf{h}(u,\mathsf{sk}) \in S^{\mathsf{h}}_{\mathsf{sk}}(t,m)} \big((u = m) \sim \mathsf{false}\big)}{(\mathsf{h}(m, \mathsf{sk}) = t) \sim \mathsf{false}}\ \mathsf{sk}\ \textit{only occurs as}\ \mathsf{h}(x, \mathsf{sk})\ \textit{in}\ t\ \textit{and}\ m$$

This rule has a logical premise, which becomes a proof obligation when the rule is applied to a proof goal, and a syntactic side condition automatically checked by Squirrel.

3. MODELING INTERACTIVE PROTOCOLS WITH RECURSIVE FUNCTIONS

Now that we have introduced the core CCSA framework, let us turn to how cryptographic protocols are modeled. Compared to the simple games of the previous section, protocols are concurrent systems, involving several communicating agents. An agent may for instance be a key server, a bank, a user's terminal... When analyzing the security of a protocol, it is desirable to make worst-case assumptions. A typical one is that

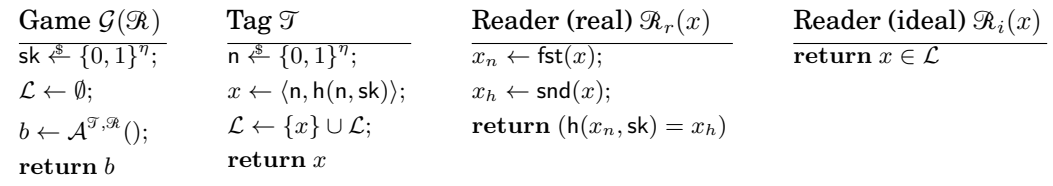


Fig. 3. Authentication game for the BASIC HASH protocol.

the adversary has full control over the network: it receives all messages output by protocol agents, and is tasked with feeding inputs to them, with messages resulting from arbitrary computations. We also assume that the adversary schedules the actions of the protocol's agents: it decides when to spawn a new session, when to advance in a session, ... This can actually be modeled using cryptographic games, representing the agents as oracles to which the adversary has access. Adversarial computations then induce a sequence of interactions with the protocol's agents, *i.e.* an execution trace of the protocol. Following this style, quantifying over all adversaries implies quantifying over all execution traces. We present this approach in more details in what follows, illustrating it on a simplified version of the BASIC HASH protocol [Brusò et al. 2010].

3.1. An example: BASIC HASH protocol

The BASIC HASH protocol is a simple RFID protocol in which a tag $\mathcal{T}$ tries to authenticate itself to a reader $\mathcal{R}$. This is an access control protocol: *e.g.* the tag may be embedded in an RFID fob, and the reader could guard access to some building. The tag and the reader both share a secret key sk. Whenever the tag wants to authenticate itself to the reader, it samples a random value n, hashes it with sk using an unforgeable keyed hash function as presented in Section 2.3, and sends the pair $\langle \mathsf{n}, \mathsf{h}(\mathsf{n}, \mathsf{sk}) \rangle$ of the name and its hash digest to the reader. The reader can check that a message x it received from the network was generated by the tag by extracting the first component $\mathsf{fst}(x)$ of x, re-computing its hash using the secret key sk and checking that this yields the second component $\mathsf{snd}(x)$ of its input.

Authentication game. We can express the fact that this protocol provides some form of authentication using the cryptographic game presented in Figure 3. The definitions of $\mathcal{T}$ and $\mathcal{R}_r$ correspond to the tag and reader agents described above, with a single addition that will be useful to express the security property: before sending its output $x = \langle \mathsf{n}, \mathsf{h}(\mathsf{n}, \mathsf{sk}) \rangle$ to the reader, the tag logs x into a set $\mathcal{L}$. Further, $\mathcal{R}_i$ corresponds to an idealized reader, which checks that its input x originates from $\mathcal{T}$ by directly verifying that x appears in the log $\mathcal{L}$. Intuitively, the idealized reader looks, across space and time, into the past internal memories of the tag to check whether it indeed generated x. Obviously, $\mathcal{R}_i$ is only useful as a trick to model the expected authentication property, and cannot be physically implemented.

The cryptographic game $\mathcal{G}(\mathcal{R})$, parameterized by the reader $\mathcal{R} \in \{\mathcal{R}_r, \mathcal{R}_i\}$, samples the secret key sk, initializes the log $\mathcal{L}$, and runs the adversary $\mathcal{A}^{\mathcal{T},\mathcal{R}}$ by letting it interact with the tag and reader oracles. The adversary must try to guess a bit b indicating whether it is interacting with the real or ideal reader $\mathcal{R}$: its guess b is the final result of the game. The protocol is secure if $\mathcal{A}$ has a negligible advantage in guessing the correct bit b, *i.e.* if the computational indistinguishability $\mathcal{G}(\mathcal{R}_r) \approx \mathcal{G}(\mathcal{R}_i)$ holds. Indeed, if an attacker can make $\mathcal{R}_r$ accept while breaking authentication (*i.e.* no corresponding tag produced the message), trying to do the same execution with $\mathcal{R}_i$ would fail by construction, making it trivial to distinguish the two worlds.

3.2. Modeling protocol executions

During its execution, $\mathcal{A}$ can call the tag and reader oracles any number of times and according to any interleaving. We are going to model the interaction of $\mathcal{A}$ with the protocol agents $\mathcal{T}, \mathcal{R}$ along an execution trace tr representing a fixed but arbitrary interleaving. Thus, an *execution trace* tr is a finite sequence $\tau_1, \ldots, \tau_n$ of *timestamps*, where the j-th timestamp τ_j in the trace represents the agent the adversary interacted with at this step. To distinguish multiple interactions with the same agents, actions are indexed by a unique replication index i in some set of indices $\mathcal{I}$. For BASIC HASH, we use $\tau = \mathsf{T}(i)$ and $\tau = \mathsf{R}(i)$ for an interaction with respectively the tag and the reader. For convenience, we also force τ_1 to a special value init.

$$\mathcal{T} ::= \tau_1, \ldots, \tau_n \qquad \tau_i ::= \mathsf{init} \mid \mathsf{T}(i) \mid \mathsf{R}(i) \qquad (n \in \mathbb{N}, i \in \mathcal{I})$$

Modeling execution traces. We assume a typed logic, to enable reasoning over different kinds of objects. Models of the logic provide an interpretation domain for each type, where for instance $\mathsf{message}$ is interpreted as the set of bitstrings and bool as $\{0,1\}$. Intuitively, the type of all terms used in Section 2 is $\mathsf{message}$ or bool, with *e.g.* the equality function symbol $=$ typed as $\mathsf{message} \to \mathsf{message} \to \mathsf{bool}$. The types index and $\mathsf{timestamp}$ respectively represent the sets of indices $\mathcal{I}$ and timestamps $\mathcal{T}$, and we use function symbols T and R, both of type $\mathsf{index} \to \mathsf{timestamp}$, to build timestamps from indices, and $\mathsf{init} : \mathsf{timestamp}$ for the initial timestamp. The execution trace tr is implicitly encoded using two function symbols:

$$\mathsf{happens} : \mathsf{timestamp} \to \mathsf{bool} \qquad \cdot < \cdot : \mathsf{timestamp} \to \mathsf{timestamp} \to \mathsf{bool}$$

where $\mathsf{happens}(\tau)$ states that τ has been scheduled in the execution trace, and $\tau < \tau'$ that τ occurred before τ' in the trace. Each model $\mathbb{M}$ of the logic fixes the execution trace by interpreting the types and function symbols above. We require, through an ad hoc restriction on models of the logic, that the interpretation of index in any model is finite and independent of the security parameter η. Further, we assume that any scheduled timestamp uniquely corresponds to either init, some $\mathsf{T}(i)$ or some $\mathsf{R}(i)$.

Modeling the game's execution. In the logic, the interaction of the adversary $\mathcal{A}$ with the oracles $\mathcal{T}$ and $\mathcal{R}$ along an execution trace is modeled using several mutually recursive functions called *macros*. A macro is identified by its name m and is evaluated at a particular timestamp τ: informally, $\mathsf{m}@\tau$ refers to the value of m at point τ in the execution. We will use the following macros: $\mathsf{input}@\tau$ and $\mathsf{output}@\tau$ denote the messages input and output by the oracle scheduled at time τ in the execution trace; and $\mathsf{frame}@\tau$ is roughly the sequence of all messages output by any agent during the execution up to timestamp τ. The frame is crucial to express security properties, as it represents the full list of messages seen by the attacker. We assume a function symbol $\mathsf{pred} : \mathsf{timestamp} \to \mathsf{timestamp}$ that maps any scheduled timestamp different from init to its predecessor in the execution trace, and all other timestamps to a fixed constant undef, distinct from all scheduled timestamps. The macros can be defined as follows for the game $\mathcal{G}(\mathcal{R}_r)$ using the real reader:

$$\begin{array}{l}
\mathsf{input_r}@\tau = \mathsf{match}\ \tau\ \mathsf{with} \\
\quad \mid \mathsf{init} \to \mathsf{empty} \\
\quad \mid _ \to \mathsf{att}_\tau(\mathsf{frame_r}@\mathsf{pred}(\tau))
\end{array}
\qquad
\begin{array}{l}
\mathsf{frame_r}@\tau = \mathsf{match}\ \tau\ \mathsf{with} \\
\quad \mid \mathsf{init} \to \mathsf{empty} \\
\quad \mid _ \to \langle \mathsf{frame_r}@\mathsf{pred}(\tau), \mathsf{output_r}@\tau \rangle
\end{array}$$

$$\begin{array}{l}
\mathsf{output_r}@\tau = \mathsf{match}\ \tau\ \mathsf{with} \\
\quad \mid \mathsf{init} \to \mathsf{empty} \\
\quad \mid \mathsf{T}(i) \to \langle \mathsf{n}(i), \mathsf{h}(\mathsf{n}(i), \mathsf{sk}) \rangle \\
\quad \mid \mathsf{R}(i) \to \mathsf{h}(\mathsf{fst}(\mathsf{input_r}@\tau), \mathsf{sk}) = \mathsf{snd}(\mathsf{input_r}@\tau)
\end{array}$$

where empty is a symbol of type message representing the empty bitstring, and where we generalized names to be of type index → message, so that each $\mathsf{n}(i)$ is an independent fresh name. The r subscript indicates that these macros are for the game using the real reader. The macros definitions for the game with the ideal reader $\mathcal{G}(\mathcal{R}_i)$, subscripted by i, are identical except for the output macro: we only need to introduce a macro $\mathcal{L}@\tau$ for the value of the log $\mathcal{L}$ *after* the execution of the oracle at τ, and to modify the $\mathsf{R}(i)$ case of output:

$$\begin{array}{l} \mathsf{output_i}@\tau \;=\; \mathsf{match}\ \tau\ \mathsf{with} \\ \qquad\qquad \mid \mathsf{R}(i) \to \mathsf{input_i}@\tau \in \mathcal{L}@\tau \\ \qquad\qquad \mid \ldots \to \ldots \text{ (as before)} \end{array} \qquad \begin{array}{l} \mathcal{L}@\tau \;=\; \mathsf{match}\ \tau\ \mathsf{with} \\ \qquad\qquad \mid \mathsf{init} \to \emptyset \\ \qquad\qquad \mid \mathsf{T}(i) \to \{\mathsf{output_i}@\tau\} \cup \mathcal{L}@\mathsf{pred}(\tau) \\ \qquad\qquad \mid \mathsf{R}(i) \to \mathcal{L}@\mathsf{pred}(\tau) \end{array}$$

Then, the formula $\mathsf{frame_r}@\tau \sim \mathsf{frame_i}@\tau$ expresses the fact that, for any model $\mathbb{M}$ defining an execution trace tr and an adversary $\mathcal{A}$, the adversary has a negligible probability of distinguishing the real and ideal versions of the protocol, *i.e.* $\mathcal{G}(\mathcal{R}_r) \approx \mathcal{G}(\mathcal{R}_i)$ holds *along this particular execution trace* (see Section 3.3 below for a discussion of the security guarantees provided by this formula).

Cryptographic reasoning with recursive functions. The EUF-CMA reasoning rule presented in Section 2.3 does not directly apply to terms containing recursive definitions: we need to be able to reason over the occurrences of the hash function inside terms, and thus possibly inside an arbitrary recursive unfolding. A first way to fix the EUF-CMA rule is to forbid any occurrence of a hash in all the recursive functions occurring inside a term, but it is of course too limiting: in BASIC HASH, we have for instance a set of hash occurrences of the form $\mathsf{h}(\mathsf{n}(i), \mathsf{sk})$ in $\mathsf{output_r}$. We need to be able to reason over such a set, for instance using some universal quantification over indices or timestamps. We formally introduce such constructions in Section 4.2, and extend the cryptographic rules in Section 4.4.

Implementation in Squirrel. The encoding of protocols using macros as described above can be generalized to support a large class of security protocols. Given a protocol $\mathcal{P}$, we can use the same definitions for frame and input, and we only need to adapt the definition of output according to the protocol specification, and, if necessary, introduce additional macros for the protocol's mutable variables (such as the macro $\mathcal{L}$ for the log $\mathcal{L}$). This method is expressive enough to encode complex protocol behaviors, and allows all relevant information to be expressed as messages that our logic can handle. However, it is arguably not very intuitive to manipulate for users. To help with that, Squirrel allows users to describe protocols as processes in a process algebra (a variant of the applied π-calculus), and automatically translates them as macros (the actual definition of Squirrel macros is slightly more involved than the one presented here, we refer readers to [Baelde et al. 2022] for details).

3.3. The issue of asymptotic security

With the modeling style described here for protocols, we always reason on an arbitrary but fixed trace. When proving a property, we show that it holds (up to negligible probability) in all models of the logic, and the model includes the interpretation of the order on timestamps, *i.e.* the interleaving of actions, as well as the interpretation of all attacker function symbols. In particular, this means that in any given model, the trace and the interpretation of adversarial functions is fixed, and only then do the security parameter η and random tape ρ vary. In other words, when proving a property ϕ, what we establish can informally be described by the formula:

$$\forall \text{ trace } \mathsf{tr}.\ \forall \text{ adversary } \mathcal{A}.\ \Pr_\rho(\phi \text{ holds in } \mathsf{tr} \text{ against } \mathcal{A}) \text{ is overwhelming in } \eta.$$

While such a statement gives good security guarantees, it is weaker than what is commonly expected. In cryptographic games, the adversary is indeed given access to the protocol through oracles it may call at will any (polynomial) number of times. That is, the adversary can *choose* the trace, rather than having it imposed on him. In particular, the length of the trace may be chosen by the adversary and thus depend on η, while it is fixed in our model. That would correspond to the stronger (informal) formula:

$$\forall \text{ adversary } \mathcal{A}.\ \Pr_\rho(\phi \text{ holds in } \mathsf{tr} \text{ chosen by } \mathcal{A}) \text{ is overwhelming in } \eta.$$

It is possible to overcome this limitation by changing the way we model protocols, to let the adversary adaptively choose the trace during the protocol execution. However, such alternatives have not yet been explored in practice.

4. THE GENERAL LOGIC & PROOF SYSTEM

We have considered so far a typed language with first-order terms and a single indistinguishability predicate, with inference rules that can be expressed as axioms in first-order logic. Except for the addition of recursively defined functions, all this fits within the original CCSA logic of [Bana and Comon-Lundh 2014], which is plain first-order logic. Thus, proving statements in that logic can be done using any proof system for classical first-order logic; only the CCSA axiom schemes would reflect the intended probabilistic semantics of terms and the cryptographic assumptions on primitives.

We have seen, however, that syntactic side conditions, which are needed for cryptographic rules, cannot be easily lifted to terms containing recursive definitions. In addition, the logic appears ill-suited for some security properties. Recall the BASIC HASH example, whose security we defined by using a real and an ideal system. Yet, the intuitive security property should only be about a high-level fact verified by all executions of the real system: whenever a reader accepts, the value it received was produced by an honest tag.

To improve this aspect of the logic, a core idea is to introduce a more specific logic, where special status is given to terms of type bool. We have seen axioms involving indistinguishability, of the form $t \sim \mathsf{true}$ or $t \sim \mathsf{false}$. For such formulas, using computational indistinguishability for all PPTM distinguishers is overkill, and we can take a much simpler approach: $t \sim \mathsf{true}$ simply means that t is true with overwhelming probability. Further, it is convenient to view t itself as a specific kind of formula: to this effect, we can use function symbols representing propositional connectives, as we did for the if-then-else construction; less obviously, the same can be done for quantifiers. In that context, t will be called a *local* formula, while the real first-order formulas are called *global*. It turns out that CCSA proofs tend to intensively reason on local formulas. This calls for a proof system that conveniently handles these like proper formulas. Squirrel comes with one such proof system, which we present in this section.

4.1. Higher-order terms with a relaxed semantics

Before properly introducing local and global formulas and their proof systems, we must come back to the semantics of terms. Indeed, since [Baelde et al. 2023], the logic behind Squirrel features higher-order terms. There have been several motivations for this significant generalization of the original CCSA logic. First, Squirrel developments predominantly consist of proofs of local formulas, and higher-order features at this level help structure these proofs in a modular way, and re-use them. Second, the constraints of the original CCSA logic, namely that all terms are interpreted as PPTMs, make sense for modeling cryptographic protocols and adversaries but are sometimes too restrictive when writing proofs. For instance, it can be useful to talk about the discrete logarithm over finite groups, even though this operation is (hopefully) not PTIME. It can also be useful to quantify over infinite types in local formulas, *e.g.* to specify properties of

primitives over messages: anticipating on what follows, we will be able to write local formulas such as $\forall x, y : \mathsf{message}.\ \mathsf{fst}(\langle x, y\rangle) = x$.

The terms of our logic are thus simply-typed λ-calculus terms, additionally featuring recursive definitions. In the tool, some limited forms of polymorphism are even available, but we leave them unaccounted for in the theory. Each type τ is interpreted in a model $\mathbb{M}$ as a family of sets $[\![\tau]\!]^\eta_{\mathbb{M}}$ indexed by the security parameter η, with $[\![\tau_1 \to \tau_2]\!]^\eta_{\mathbb{M}} = [\![\tau_1]\!]^\eta_{\mathbb{M}} \to [\![\tau_2]\!]^\eta_{\mathbb{M}}$. We require that $[\![\mathsf{bool}]\!]^\eta_{\mathbb{M}} = \{0,1\}$ and $[\![\mathsf{message}]\!]^\eta_{\mathbb{M}} = \{0,1\}^*$ for all $\mathbb{M}$ and η. The potential dependency in η is however crucial *e.g.* to faithfully model the finite groups used in Diffie-Hellman key exchanges.

Terms are no longer interpreted as PPTMs, but more generally as discrete random variables, *i.e.* functions from random tapes to the desired domain. More precisely, a model fixes a set of *finite tapes* $\mathbb{T}_{\mathbb{M},\eta}$ for each η, and we let $\mathbb{RV}_{\mathbb{M}}(\tau)$ be the set of all η-indexed sequences of functions from $\mathbb{T}_{\mathbb{M},\eta}$ to $[\![\tau]\!]^\eta_{\mathbb{M}}$. Then, given $\mathbb{M}$, η and ρ, any term t of type τ is interpreted as $(\rho \in \mathbb{T}_{\mathbb{M},\eta} \mapsto [\![t]\!]^{\eta,\rho}_{\mathbb{M}})_{\eta\in\mathbb{N}} \in \mathbb{RV}_{\mathbb{M}}(\tau)$ as follows:

$$\begin{aligned}
[\![x]\!]^{\eta,\rho}_{\mathbb{M}} &= \mathbb{M}(x)(\eta,\rho)\\
[\![t\ t']\!]^{\eta,\rho}_{\mathbb{M}} &= [\![t]\!]^{\eta,\rho}_{\mathbb{M}}([\![t']\!]^{\eta,\rho}_{\mathbb{M}})\\
[\![\lambda(x:\tau').t]\!]^{\eta,\rho}_{\mathbb{M}} &= (a \in [\![\tau']\!]^\eta_{\mathbb{M}} \mapsto [\![t]\!]^{\eta,\rho}_{\mathbb{M}[x/a]})
\end{aligned}$$

where $\mathbb{M}(x)$ is the interpretation of x in $\mathbb{M}$, and $\mathbb{M}[x/a]$ is the model modified to interpret x as a random variable X such that $X(\eta,\rho) = a$. Note that the interpretation of a term only depends on the interpretation of its subterms for the same values of η and ρ, and thus the value of $\mathbb{M}[x/a](x)$ is only relevant on η, ρ. We refer the reader to [Baelde et al. 2023] for the full details, including the interpretation of recursive definitions when they satisfy a suitable well-foundedness condition, as well as how to check that some terms do correspond to PPTM computations in order to apply cryptographic assumptions.

While it could seem more natural to have infinite tapes, rather than finite but arbitrarily long ones, this has technical motivations. It ensures that all functions computed on tapes actually correspond to random variables, *i.e.* that they are measurable. This restriction of our logic (and of the original CCSA logic) is not limiting when it comes to modeling cryptographic protocols and attackers, which all run in PTIME. It is however crucial that the length of tapes can grow with η. The restriction to finite tapes is standard in formal cryptographic reasoning: it is present, *e.g.*, in approaches based on Hoare logics [Barthe et al. 2011; Petcher and Morrisett 2015; Basin et al. 2020].

4.2. Local and global formulas

Squirrel's logic is structured around two kinds of formulas: we call *global* formulas the formulas of first-order logic built around the $\sim$ predicate, and *local* formulas the terms of type bool. To clearly distinguish the two, we add tildes to logical connectives and quantifiers at the global level. We also note formulas with different letters: ϕ and ψ for local formulas and F and G for global ones.

$$\begin{aligned}
F &:= F\ \tilde\wedge\ F' \mid F\ \tilde\vee\ F' \mid F \mathrel{\tilde\Rightarrow} F' \mid \tilde\neg F \mid \tilde\forall x.F \mid \tilde\exists x.F \mid \vec u \sim \vec v \mid [\phi] \mid \dots\\
\phi &:= \phi \wedge \phi' \mid \phi \vee \phi' \mid \phi \Rightarrow \phi' \mid \neg\phi \mid \forall x.\phi \mid \exists x.\phi \mid u = v \mid \dots
\end{aligned}$$

Atomic global formulas include indistinguishabilities, but also atoms of the form $[\phi]$, which can be understood as $\phi \sim \mathsf{true}$. The syntax above is open-ended, as more predicates will be used: in addition to custom predicates introduced by the user, *e.g.* to model verifications on messages, we will make use of several extra predicates later in this section. Note, however, that we will not use equality at the global level.

Indistinguishability atoms $\vec u \sim \vec v$ can only be formed when u_i and v_i have the same type for all i. Moreover, this type must be of order at most 1, *i.e.* of the form $\tau_1 \to \dots \to \tau_n$ where all τ_k are base atomic types. Further, we assume that elements of base types can

be encoded as bitstrings, which allows interpreting these atoms as in Section 2.2, with a generalization: we now consider a distinguisher that can access the (semantics of) terms as oracles, which can be called on arbitrary inputs to obtain new data.

As said before, local formulas are just boolean terms. As such, the syntax above should be understood as syntactic sugar: for instance, $\phi \wedge \psi$ stands for $(\wedge)\ \phi\ \psi$ where $(\wedge)$ is a constant function of type $\mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$. We also view local quantifiers in this way: $\forall(x : \tau).\ \phi$ is a notation for $\forall_\tau\ (\lambda(x : \tau).\ \phi)$ where $\forall_\tau$ is a constant of type $(\tau \to \mathsf{bool}) \to \mathsf{bool}$. We require that all models interpret these logical constants as expected. In particular, $[\![\forall x.\ \phi]\!]^{\eta,\rho}_{\mathbb{M}} = 1$ iff $[\![\phi]\!]^{\eta,\rho}_{\mathbb{M}[x/a]} = 1$ for all $a \in [\![\tau]\!]^{\eta}_{\mathbb{M}}$.

Example 4.1. Consider once again the BASIC HASH protocol of Figure 3. Instead of expressing its security by using an equivalence, we can also state it with a local formula over the previously defined macros $\mathsf{input}_\mathsf{r}$ and $\mathsf{output}_\mathsf{r}$:

$$\begin{aligned}\forall i : \mathsf{index}.\ \ &\mathsf{happens}(\mathsf{R}(i)) \wedge \mathsf{output}_\mathsf{r}@\mathsf{R}(i) = \mathsf{true} \\ &\Rightarrow \exists j : \mathsf{index}.\ \mathsf{T}(j) < \mathsf{R}(i) \wedge\ \mathsf{input}_\mathsf{r}@\mathsf{R}(i) = \mathsf{output}_\mathsf{r}@\mathsf{T}(j)\end{aligned}$$

Asking that this formula holds with overwhelming probability properly expresses authentication for any trace: whenever some tag $\mathsf{R}(i)$ accepts, its input value must have been produced in the past by an honest reader $\mathsf{T}(j)$.

As usual in first-order logic, a global formula can be satisfied (or not) in a model: we write $\mathbb{M} \models F$ when this is the case. A global formula is valid when it is satisfied in all models. We write $F \models G$ if, for all $\mathbb{M}$, $\mathbb{M} \models F$ implies $\mathbb{M} \models G$. A local formula, however, is valid if it is overwhelmingly true in all models: ϕ is valid iff $[\phi]$ is valid, *i.e.* $\eta \mapsto \Pr_{\rho \in \mathbb{T}_{\mathbb{M},\eta}}([\![\phi]\!]^{\eta,\rho}_{\mathbb{M}} = 1)$ is overwhelming.

Example 4.2. We have $[\phi \Rightarrow \psi] \models [\phi] \tilde{\Rightarrow} [\psi]$: if both $\phi \Rightarrow \psi$ and ϕ are overwhelmingly true, then so must be ψ. However, $[\phi] \tilde{\Rightarrow} [\psi] \not\models [\phi \Rightarrow \psi]$: indeed, $[\phi] \tilde{\Rightarrow} [\psi]$ might be satisfied in a model where ϕ is not overwhelmingly true, but true for half of the tapes; hence ψ could be always false, and $[\phi \Rightarrow \psi]$ does not hold.

Example 4.3. We have $[\phi]\ \tilde{\vee}\ [\psi] \models [\phi \vee \psi]$ but not the converse: there might be models where only ϕ holds for half of the random tapes and only ψ holds for the other half, hence $\phi \vee \psi$ is overwhelmingly true (even exactly true) while neither ϕ nor ψ is.

In each model, global formulas are either fixed to true or false, but their quantifiers range over random variables, *i.e.* elements of $\mathbb{RV}_{\mathbb{M}}(\tau)$ for some τ. In contrast, local formulas are probabilistic and interpreted in $\mathbb{RV}_{\mathbb{M}}(\mathsf{bool})$, but their quantifiers range over individuals, *i.e.* elements $a \in [\![\tau]\!]^{\eta,\rho}_{\mathbb{M}}$. Despite this essential difference, we can relate quantifiers at the two levels, as shown in [Baelde et al. 2023, Proposition 2].

PROPOSITION 4.4. *For all $\mathbb{M}$, we have $\mathbb{M} \models [\forall(x : \tau).\phi]$ iff. $\mathbb{M} \models \tilde{\forall}(x : \tau).\ [\phi]$, and similarly for the existential quantifiers.*

4.3. Proof system for global and local reasoning

Squirrel's proof system is given in a natural deduction style (though the tool's tactics are sometimes closer to sequent calculus rules) and it deals with two kinds of sequents. Global sequents are of the form $\mathcal{E}; \Theta \vdash F$ where F is a global formula, Θ is a set of global formulas, and $\mathcal{E}$ is a list of typed variable declarations containing at least all free variables of Θ, F. Such a sequent can be read as the global formula $\tilde{\forall}\mathcal{E}.(\tilde{\wedge}\Theta \tilde{\Rightarrow} F)$, and it is valid when the associated formula is valid. Local sequents are of the form $\mathcal{E}; \Theta; \Gamma \vdash \phi$ where ϕ is a local formula, Γ is a set of local formulas, and the other components are as before, with $\mathcal{E}$ declaring all free variables of Θ, Γ and ϕ. The formula associated to a local sequent is $\tilde{\forall}\mathcal{E}.(\tilde{\wedge}\Theta \tilde{\Rightarrow} [\wedge\Gamma \Rightarrow \phi])$.

It is important to note, in the meaning of local sequents, that the second implication happens at the local level. As a result, the *global* hypotheses of Θ and the *local* hypotheses of Γ take a different meaning, illustrated in the following selected rules:

$$\frac{}{\mathcal{E};\Theta,F\vdash F}\qquad\frac{}{\mathcal{E};\Theta;\Gamma,\phi\vdash\phi}\qquad\frac{}{\mathcal{E};\Theta,[\phi];\Gamma\vdash\phi}\qquad\frac{\mathcal{E};\Theta;\emptyset\vdash\phi}{\mathcal{E};\Theta\vdash[\phi]}$$

The first rule is the usual axiom rule for the global logic. The second and third rules are local and global versions of the axiom rule for the local logic. The last rule allows proving a global sequent whose conclusion is $[\phi]$ from the corresponding local sequent.

The local equality predicate gives rise to the ability to rewrite at the local level, which is in fact a generalization of the REWRITE rule from Figure 2:

$$\frac{\mathcal{E};\Theta;\Gamma\vdash u=v\quad\mathcal{E};\Theta;\Gamma\vdash\phi\{x\mapsto u\}}{\mathcal{E};\Theta;\Gamma\vdash\phi\{x\mapsto v\}}$$

In the tool, a powerful `rewrite` tactic is provided, which builds on the previous rule as well as more basic rules, to perform various kinds of rewriting. The tactic can be invoked on a goal $\mathcal{E};\Theta;\Gamma\vdash\phi$ to selectively replace occurrences of u into v inside Θ,Γ and ϕ when $u=v$ is part of the hypotheses (or can be simply derived from them). If $u=v$ is a local hypothesis, however, rewriting will only be possible in Γ and ϕ; rewriting in Θ requires a global hypothesis $[u=v]$, and is only possible at specific occurrences in Θ.

The previous result relating global and local quantifiers justifies that our sequents only feature a top-level environment $\mathcal{E}$ corresponding to a global quantification. The rules for universal quantifiers in our proof system are as follows, assuming that x does not occur in $\mathcal{E}$, and that t is a well-typed term of type τ in environment $\mathcal{E}$:

$$\frac{\mathcal{E},x:\tau;\Theta\vdash F}{\mathcal{E};\Theta\vdash\tilde{\forall}(x:\tau).F}\qquad\frac{\mathcal{E};\Theta\vdash\tilde{\forall}(x:\tau).F}{\mathcal{E};\Theta\vdash F\{x\mapsto t\}}\qquad\frac{\mathcal{E},x:\tau;\Theta;\Gamma\vdash\phi}{\mathcal{E};\Theta;\Gamma\vdash\forall(x:\tau).\phi}\qquad\frac{\mathcal{E};\Theta;\Gamma\vdash\forall(x:\tau).\phi}{\mathcal{E};\Theta;\Gamma\vdash\phi\{x\mapsto t\}}$$

Let us mention one last articulation between the global and local logics:

$$\frac{\mathcal{E};\Theta;\Gamma_1\vdash\phi_1\quad\mathcal{E};\Theta\vdash(\wedge\Gamma_0\Rightarrow\phi_0)\sim(\wedge\Gamma_1\Rightarrow\phi_1)}{\mathcal{E};\Theta;\Gamma_0\vdash\phi_0}$$

In the tool, this rule is made available through a convenient `rewrite equiv` tactic, typically used to establish local formulas when they essentially derive from indistinguishability assumptions.

Example 4.5. A common assumption over hash functions, stronger than EUF-CMA, states that they can be seen as pseudo-random functions (PRF): for a fresh secret key sk, an attacker cannot distinguish between $x\mapsto\mathsf{h}(x,\mathsf{sk})$ and a fully random function. It notably yields in our logic that $\mathsf{h}(0,\mathsf{sk}),\mathsf{h}(1,\mathsf{sk})\sim\mathsf{n},\mathsf{m}$. Applying on both sides of the equivalence the function $\lambda x,y:\mathsf{message}.\ x\neq y$, we can obtain the validity of the indistinguishability formula $\mathsf{h}(0,\mathsf{sk})\neq\mathsf{h}(1,\mathsf{sk})\sim\mathsf{n}\neq\mathsf{m}$. With this global formula and the `rewrite equiv` tactic, we can reduce the proof of the local sequent $\mathcal{E};\Theta;\cdot\vdash\mathsf{h}(0,\mathsf{sk})\neq\mathsf{h}(1,\mathsf{sk})$ to the proof of $\mathcal{E};\Theta;\cdot\vdash\mathsf{n}\neq\mathsf{m}$.

A more complete description of our proof system is given in [Baelde et al. 2023]. However, it is important to point out that the system presented in that paper (or any earlier one) is only a collection of correct rules: it does not come with any completeness or cut-elimination result. Ongoing investigations are starting to provide such proof-theoretical results, though in a propositional fragment that allows to view our two-level logic in the richer framework of modal logic.

4.4. Cryptographic reasoning with recursive functions

It is natural to lift cryptographic reasoning to either local or global reasoning depending on the nature of the cryptographic assumption. In the case of EUF-CMA from Section 2.3, it could become a local rule of the form:

$$\textsc{Euf-Cma}\ \frac{\mathcal{E};\Theta;\Gamma \vdash \bigwedge_{\mathsf{h}(u,\mathsf{sk})\in S^{\mathsf{h}}_{\mathsf{sk}}(t,m)}(u \neq m)}{\mathcal{E};\Theta;\Gamma \vdash \mathsf{h}(m,\mathsf{sk}) \neq t}\ \mathsf{sk}\ \textit{only occurs as}\ \mathsf{h}(x,\mathsf{sk})\ \textit{in}\ t\ \textit{and}\ m$$

Yet, now that terms contain recursive functions, the meaning of forbidding occurrences of sk is unclear: forbidden computations (*e.g.* a leak of a secret key) might occur deep inside the recursive functions. Further, sk may be an indexed key $\mathsf{k}(i)$, and occurrences with distinct indices $\mathsf{k}(j)$ for $j \neq i$ should be ignored. Checking for such indirect occurrences requires a deeper analysis of the terms involved.

Example 4.6. Let $\mathsf{k}(i)$ be a name symbol indexed by an integer i and representing a key, and consider the following recursive function:

$$\mathsf{keys} = \lambda(i : \mathsf{index}).\ \mathsf{if}\ (i = 0)\ \mathsf{then}\ \mathsf{empty}\ \mathsf{else}\ \langle \mathsf{k}(i), \mathsf{keys}(i-1)\rangle$$

where we assume to have functions symbols for the standard operations over integers. Basically, $\mathsf{keys}(i)$ computes the list $[\mathsf{k}(i), \mathsf{k}(i-1), \ldots, \mathsf{k}(1)]$ (encoded as nested pairs). Consider some index terms i_0, j_0 and assume that $j_0 < i_0$. To prove $\mathsf{h}(\mathsf{att}(\mathsf{keys}(j_0)), \mathsf{k}(i_0)) \neq t$ with EUF-CMA, we need to check, among other things, that $\mathsf{k}(i_0)$ does not occur in $\mathsf{keys}(j_0)$. The term $\mathsf{keys}(j_0)$ cannot be evaluated without providing a concrete value for j_0. Moreover, if we attempted to directly check the condition on $\mathsf{keys}(j_0)$'s body, we would find that it contains $\mathsf{k}(i)$ (for a bound i), which can *a priori* be equal to $\mathsf{k}(i_0)$. To conclude here, we need to exploit the hypothesis that $j_0 < i_0$ and details on how the recursive function keys is defined. Thus, we need a more involved check exploiting additional information that cannot be directly obtained from a superficial scrutiny of the terms.

Generalized subterms. We therefore introduce a generalized notion of subterm, to compute an over-approximation of all the subterms that may occur during the evaluation of a recursive function, for any model and random tapes. In addition, to allow exploiting semantic information (*e.g.* the fact that for any $\mathsf{k}(i)$ in $\mathsf{keys}(j)$, we have $i \neq 0$), we gather together with each subterm a local formula expressing the condition under which that subterm may be computed. A generalized subterm, or *occurrence*, σ in a term t is a tuple $(\mathcal{V}, \phi, s)$: s is, roughly, a subterm of t (expanding recursive definitions), $\mathcal{V}$ is a set of typed variables bound above s in t, and ϕ is the condition under which s is evaluated in t, gathered from all if conditions. We use this notion, denoted $\mathsf{ST}(t)$, in cryptographic rules to replace the previous conditions on syntactic subterms. For instance, a condition expressing that $\mathsf{k}(i_0)$ does not occur in t now instead requires that for each $(\mathcal{V}, \phi, \mathsf{k}(i)) \in \mathsf{ST}(t)$, the formula $\forall \mathcal{V}.\ \phi \Rightarrow i_0 \neq i$ holds. Taking all this into account, the EUF-CMA rule finally becomes:

$$\textsc{Euf-Cma}\ \frac{\mathcal{E};\Theta;\Gamma \vdash \left(\bigwedge_{(\mathcal{V},\phi,\mathsf{k}(i))\in\mathsf{ST}_{\mathsf{h}}(t,m)} \forall \mathcal{V}.\ \phi \Rightarrow i_0 \neq i\right) \wedge \left(\bigwedge_{(\mathcal{V},\phi,\mathsf{h}(u,\mathsf{k}(i)))\in\mathsf{ST}(t,m)} \forall \mathcal{V}.\ (\phi \wedge i_0 = i) \Rightarrow u \neq m\right)}{\mathcal{E};\Theta;\Gamma \vdash \mathsf{h}(m,\mathsf{k}(i_0)) \neq t}$$

where $\mathsf{ST}_{\mathsf{h}}(t,m)$ is $\mathsf{ST}(t,m)$, where we excluded occurrences of k in subterms of the form $\mathsf{h}(\cdot,\mathsf{k}(\cdot))$. An additional premise (omitted here) checks that t and m are computable in polynomial time.

Making the verification effective. As presented here, the set of occurrences in a term may be infinite: thus, we cannot generate one premise for each occurrence. To alleviate

this issue, we use heuristics that help Squirrel construct a proof obligation for the user that over-approximates this infinite set in a way that makes it expressible as a single formula. We illustrate how this can be done on an example.

Example 4.7. Continuing Example 4.6, consider arbitrary indices i_0, j_0 such that $j_0 < i_0$. Recall that $\mathsf{keys}(j_0)$ computes the list of keys $[\mathsf{k}(j_0), \ldots, \mathsf{k}(1)]$. In the style described above, a condition for a rule may require us to consider all elements $(\mathcal{V}, \phi, \mathsf{k}(i))$ of $\mathsf{ST}(\mathsf{att}(\mathsf{keys}(j_0)))$ (for any i) to show that i cannot be equal to i_0. There are countably many such elements, all of the form $(\emptyset, j_0 \neq 0 \wedge j_0 - 1 \neq 0 \wedge \cdots \wedge j_0 - \ell \neq 0, \mathsf{k}(j_0 - \ell))$ for $\ell \in \mathbb{N}$. They can all be subsumed by using instead the occurrence $(i, 0 < i \wedge i \leq j_0, \mathsf{k}(i))$. The corresponding proof obligation is $\forall i : \mathsf{index}.\ 0 < i \wedge i \leq j_0 \Rightarrow i \neq i_0$, which can indeed be proved under the hypothesis that $j_0 < i_0$.

4.5. Reasoning over traces: induction and case analysis

We now clarify the interpretation of traces in this logic, and what kind of reasoning we can perform over them. Following the modeling approach outlined in Section 3, recall that we may use $\mathsf{A} : \mathsf{index} \to \mathsf{timestamp}$ to represent an indexed action. Assuming for simplicity that we only have this action A, we restrict our attention to models $\mathbb{M}$ where

$$[\![\mathsf{timestamp}]\!]_{\mathbb{M}} = \{\mathsf{init}, \mathsf{undef}\} \cup \{\mathsf{A}(n) \mid n \in N\}$$

where N is a finite subset of $[\![\mathsf{index}]\!]_{\mathbb{M}}$ and undef is a special value used to interpret all timestamps that do not happen. Then, the constants init and A are interpreted naturally, independently of η, ρ:

$$[\![\mathsf{init}]\!]^{\eta,\rho}_{\mathbb{M}} = \mathsf{init} \qquad [\![\mathsf{A}]\!]^{\eta,\rho}_{\mathbb{M}}(n) = \mathsf{A}(n) \text{ for } n \in N \qquad [\![\mathsf{A}]\!]^{\eta,\rho}_{\mathbb{M}}(n) = \mathsf{undef} \text{ otherwise}$$

The following axiom is sound in this class of models:

$$\left[\forall x : \mathsf{timestamp}.\ \mathsf{happens}(x) \Rightarrow \left(x = \mathsf{init} \vee \exists i : \mathsf{index}.\ x = \mathsf{A}(i)\right)\right]$$

It can effectively be used to reason by case analysis on timestamps, but only at the level of local formulas. At the global level, it is tempting to postulate a stronger axiom:

$$\tilde{\forall} x : \mathsf{timestamp}.\ [\mathsf{happens}(x)] \mathrel{\tilde{\Rightarrow}} \left([x = \mathsf{init}] \mathrel{\tilde{\vee}} \tilde{\exists} i : \mathsf{index}.\ [x = \mathsf{A}(i)]\right)$$

However, this is too strong, and unsound in our class of models: consider for instance a random variable over timestamps that is init for some random tapes and some $\mathsf{A}(n)$ for other tapes. To avoid this problem, we crucially need to quantify over random variables that are actually constant, *i.e.* which depend neither on η nor on ρ. Assuming a global predicate $\mathsf{const}(\cdot)$ with exactly this meaning, we can now write the following, which is sound in our class of models:

$$\forall x : \mathsf{timestamp}.\ \mathsf{const}(x) \mathrel{\tilde{\wedge}} [\mathsf{happens}(x)] \mathrel{\tilde{\Rightarrow}} \left([x = \mathsf{init}] \mathrel{\tilde{\vee}} \tilde{\exists} i : \mathsf{index}.\ [x = \mathsf{A}(i)]\right)$$

In practice, such axioms are used in Squirrel proofs through the `case` tactic. This tactic, when given a timestamp, will perform a case analysis for local or global goals. However, when the goal is a global sequent, it will generate an additional subgoal to verify that the timestamp is constant, corresponding to the hypothesis in our axiom above. In practice, the constancy assumptions come from the target security property. For instance, the equivalence of protocols modeled in Section 3.2 would more precisely be stated as $\tilde{\forall}\tau.\ \mathsf{const}(\tau) \mathrel{\tilde{\Rightarrow}} \mathsf{frame}_{\mathsf{r}}@\tau \sim \mathsf{frame}_{\mathsf{i}}@\tau$.

Case analysis over timestamps is not always sufficient to prove a goal, and induction is sometimes needed. At the local level, induction is supported by the following axiom, which is obviously valid in our intended class of models:

$$[\forall (p : \mathsf{timestamp} \to \mathsf{bool}).\ (\forall x.\ (\forall y.\ y < x \Rightarrow p\ y) \Rightarrow p\ x) \Rightarrow (\forall x.\ p\ x)]$$

At the global level, the lack of higher-order quantification forces us to resort to an axiom scheme and, more importantly, we need $\mathsf{const}(\cdot)$ assumptions. For any global formula F, the following axiom is valid:

$$\big(\tilde{\forall} x.\ \mathsf{const}(x) \mathrel{\tilde{\Rightarrow}} (\tilde{\forall} y.\ \mathsf{const}(y) \mathrel{\tilde{\wedge}} [y < x] \mathrel{\tilde{\Rightarrow}} F\{x \mapsto y\}) \mathrel{\tilde{\Rightarrow}} F\big) \mathrel{\tilde{\Rightarrow}} \big(\tilde{\forall} x.\ \mathsf{const}(x) \mathrel{\tilde{\Rightarrow}} F\big)$$

Without the $\mathsf{const}(\cdot)$ restrictions, the axiom would be unsound: in a nutshell, the problem is that the well-founded relation $<$ over $[\![\mathsf{timestamp}]\!]_{\mathbb{M}}$ does not lift to a well-founded relation $[\cdot < \cdot]$ over random variables. This can be generalized to arbitrary sets of actions, and to arbitrary well-founded types: one may for instance reason by induction over natural numbers or trees, along the same lines.

4.6. Bi-deduction & cryptographic reductions

Given the scope of the logic, the question of automating some proof steps in Squirrel was quickly raised. A key property of computational indistinguishability is that it is preserved by public computations: if $\vec{u} \sim \vec{v}$ holds then for any adversarially computable function f, we have $f(\vec{u}) \sim f(\vec{v})$. This property, introduced in [Baelde et al. 2022], is of particular interest for two reasons:

— it is the theoretical justification behind many cryptographic proof steps;
— it is simple, notably because it is compositional, as opposed to indistinguishability $\sim$ which is not (see point 3 of Example 2.6).

For these reasons, this property turned-out to be a nice target for automation, reducing the proof-burden put on the user and allowing more high-level reasoning steps.

Bi-Deduction. Let $\sharp(\vec{v}_0; \vec{v}_1)$ be a pair of *sequences of terms*, representing the left and right scenarios of an equivalence. We say that it is *bi-deducible* from another pair $\sharp(\vec{u}_0; \vec{u}_1)$, which we write $\sharp(\vec{u}_0; \vec{u}_1) \rhd \sharp(\vec{v}_0; \vec{v}_1)$, if there exists a *simulator* function $\mathcal{S}$ that can be efficiently computed by the adversary and such that $\mathcal{S}([\![\vec{u}_i]\!]) = [\![\vec{v}_i]\!]$ for any $i \in \{0; 1\}$. Crucially, the same simulator $\mathcal{S}$ is used for both $i = 0$ and 1: $\mathcal{S}$'s behavior cannot depend on i as it does not know which side it is interacting with. The link between bi-deduction and indistinguishability is captured by the following rule:

$$\textsc{BiDeduce}\ \frac{\sharp(\vec{u}_0; \vec{u}_1) \rhd \sharp(\vec{v}_0; \vec{v}_1) \qquad \vec{u}_0 \sim \vec{u}_1}{\vec{v}_0 \sim \vec{v}_1}$$

Indeed, any efficient adversary $\mathcal{A}$ against the conclusion of this rule can be turned into an adversary $\mathcal{B}$ against the premise by simply composing $\mathcal{A}$ with the simulator $\mathcal{S}$ witnessing the bi-deduction premises. $\mathcal{B}$ is an efficient adversary as both $\mathcal{A}$ and $\mathcal{S}$ are, and it has the same advantage as $\mathcal{A}$'s.

The BiDeduce rule transforms the $\forall$ quantification over adversary of an equivalence into an $\exists$ quantification: instead of proving a property against any possible adversaries, of which there are infinitely many, it suffices to establish the existence of a program witnessing bi-deduction. Of course, this observation is not new, as this is the key idea behind all reduction-based arguments, whether it be in cryptography, computability or computational complexity. The main novelty is how we use bi-deduction as an automated entailment checker integrated in Squirrel tactics.

Rules. As already mentioned, bi-deduction is compositional, which allows decomposing bi-deductions as follows:

$$\textsc{BD.Tuple}\ \frac{\sharp(\vec{u}_0; \vec{u}_1) \rhd \sharp(v_1; w_1) \qquad \dots \qquad \sharp(\vec{u}_0; \vec{u}_1) \rhd \sharp(v_n; w_n)}{\sharp(\vec{u}_0; \vec{u}_1) \rhd \sharp(v_1, \dots, v_n; w_1, \dots, w_n)}$$

Here are two selected (simple) bi-deduction rules:

$$\textsc{BD.FA}\ \frac{\sharp(\vec{u}_0;\vec{u}_1) \rhd \sharp(\vec{v}_0;\vec{v}_1)}{\sharp(\vec{u}_0;\vec{u}_1) \rhd f(\sharp(\vec{v}_0;\vec{v}_1))} \qquad \textsc{BD.Trans}\ \frac{\sharp(\vec{u}_0;\vec{u}_1) \rhd \sharp(\vec{w}_0;\vec{w}_1) \qquad \sharp(\vec{u}_0,\vec{w}_0;\vec{u}_1,\vec{w}_1) \rhd \sharp(\vec{v}_0;\vec{v}_1)}{\sharp(\vec{u}_0;\vec{u}_1) \rhd \sharp(\vec{v}_0;\vec{v}_1)}$$

where common behaviors between the left and right scenarios are factorized when possible: *e.g.* $f(\sharp(\vec{v}_0;\vec{v}_1))$ denotes $\sharp(f(\vec{v}_0);f(\vec{v}_1))$. The BD.FA rule lets the adversary simulate the computation f: indeed, if $\mathcal{S}$ computes $\sharp(\vec{v}_0;\vec{v}_1)$ from $\sharp(\vec{u}_0;\vec{u}_1)$, then $f \circ \mathcal{S}$ computes $\sharp(f(\vec{v}_0);f(\vec{v}_1))$ from the same inputs. The transitivity rule BD.Trans allows extending the input $\sharp(\vec{u}_0;\vec{u}_1)$ of a simulator trying to compute $\sharp(\vec{v}_0;\vec{v}_1)$ by any sequence $\sharp(\vec{w}_0;\vec{w}_1)$ that is bi-deducible from $\sharp(\vec{u}_0;\vec{u}_1)$.

The more interesting bi-deduction rules deal with functions (seen as oracles) and conditioned terms, for which we now present two simplified rules. Conditioned terms are written $(t \mid \phi)$, which denotes the pair of terms $(\phi, \mathsf{if}\ \phi\ \mathsf{then}\ t\ \mathsf{else}\ \mathsf{witness})$, where witness is a function symbol of the same type as t.

$$\textsc{BD.Oracle}\ \frac{\rhd \sharp(t_0;t_1) \quad \vdash \begin{bmatrix} w_0\{y \mapsto t_0\} = v_0 \wedge \\ w_1\{y \mapsto t_1\} = v_1 \end{bmatrix}}{\lambda y.\, \sharp(w_0;w_1) \rhd \sharp(v_0;v_1)} \qquad \textsc{BD.Cond}\ \frac{\rhd \sharp(\psi_0;\psi_1) \quad \vdash \begin{bmatrix} \psi_0 \Rightarrow (\phi_0 \wedge v_0 = w_0) \wedge \\ \psi_1 \Rightarrow (\phi_1 \wedge v_1 = w_1) \end{bmatrix}}{\sharp((w_0 \mid \phi_0);(w_1 \mid \phi_1)) \rhd \sharp((v_0 \mid \psi_0);(v_1 \mid \psi_1))}$$

The BD.Oracle allows the simulator to call an oracle $\lambda y.\, \sharp(w_0;w_1)$ it received as input on any arguments $\sharp(t_0;t_1)$ that it can compute; the obtained terms $\sharp(w_0\{y \mapsto t_0\};w_1\{y \mapsto t_1\})$ must be established, in the second proof obligation, to be equal to the wanted terms $\sharp(v_0;v_1)$. In BD.Cond, the simulator retrieves from its inputs a conditioned term: the bi-deduction premise $\rhd \sharp(\psi_0;\psi_1)$ guarantees that the simulator knows when to extract the terms from its input; and the logical premise ensures that when it does (*i.e.* when ψ_0 or ψ_1 holds), the extracted value $\sharp(w_0;w_1)$ is given as inputs (*i.e.* when ϕ_0 or ϕ_1 holds) and is equal to the wanted value $\sharp(v_0;v_1)$.

Implementation. We implemented a fully automated heuristic procedure for bi-deduction and integrated it in the `apply` tactic of Squirrel: concretely, it is used to automatically check the entailment between equivalences, *e.g.* $\vec{u}_0 \sim \vec{u}_1 \Rrightarrow \vec{v}_0 \sim \vec{v}_1$. While this procedure is heuristic, and thus incomplete, it remains a useful tool in practice.

Advanced bi-deduction. Bi-deduction gives a formal framework to establish the existence of a simulator $\mathcal{S}$ reducing an equivalence $\vec{u}_0 \sim \vec{u}_1$ to another equivalence $\vec{v}_0 \sim \vec{v}_1$. This looks very similar to the problem of applying a cryptographic assumption expressed as a pair of games $\mathcal{G}_0 \approx \mathcal{G}_1$ (in the style of Section 2): for example, given a target equivalence $\vec{u}_0 \sim \vec{u}_1$, does there exist $\mathcal{S}$ such that $\mathcal{S}^{\mathcal{G}_i}$ computes $\vec{u}_i$ for all $i \in \{0;1\}$? Extending bi-deduction to be able to handle cryptographic games would allow supporting new cryptographic assumptions in Squirrel's logic and implementation without having to manually design rules as we did in Section 2.3 and Section 4.4. This would drastically reduce the theoretical and programming work needed to support a new rule while limiting the risk of mistakes. Furthermore, it would allow non-expert users to add new cryptographic assumptions in Squirrel simply by writing, in user-space, the pair of games $\mathcal{G}_0 \approx \mathcal{G}_1$ associated to the assumptions.

Unfortunately, the simulators covered by our previous notion of bi-deduction are too restricted for this. First, they can only interact with *stateless* oracles that are provided as first-order terms of the logic. Second, they are deterministic programs: while they may receive randomly sampled values from their inputs, they cannot do random samplings themselves. This is too limiting: oracles of cryptographic games are

Protocol name	LoC	Assumptions	Security Properties
Basic Hash ⋆	100	PRF, EUF	auth. & unlinkability
Hash Lock ⋆	130	PRF, EUF	auth. & unlinkability
LAK (with pairs) ⋆	250	PRF, EUF	auth. & unlinkability
MW ⋆	300	PRF, EUF, XOR	auth. & unlinkability
Feldhofer ⋆	270	ENC-KP, INT-CTXT	auth. & unlinkability
Private authentication ⋆	100	CCA_1, ENC-KP	anonymity
Private authentication	100	CCA$	anonymity
Signed DDH [ISO 9798-3]	240	EUF, DDH	auth. & strong secrecy
$\text{IKEV1}_{\text{PSK}}$ ⋆	850	PRF	auth. & strong secrecy
$\text{IKEV2}^{\text{SIGN}}{}_{\text{PSK}}$ ⋆	300	PRF, EUF	auth. & strong secrecy
BCGNP ⋆	355	PRF, CCA_1, XOR	strong secrecy
FSXY ⋆	620	PRF, CCA_1, XOR	strong secrecy
SC-AKE ⋆	745	PRF, EUF, CCA_1, XOR	auth. & strong secrecy
CANAuth	450	EUF	auth.
SLK06	80	EUF	auth.
YPLRK05	160	EUF	auth.
OSK (without reader)	320	ROM	strong secrecy
OSK	370	EUF	auth.
YubiKey	270	INT-CTXT	auth.
YubiHSM	720	CCA_1, INT-CTXT	auth.

Fig. 4. Some results obtained with the Squirrel prover on various protocols.

often stateful (*e.g.* EUF-CMA relies on a stateful log $\mathcal{L}$); and the simulators justifying Squirrel's cryptographic rules must sample all the randomness that does not come from the cryptographic game. Recently, we designed and implemented in Squirrel an extended notion of bi-deduction supporting stateful oracles and probabilistic simulators [Baelde et al. 2024]. This introduces a number of interesting aspects: we equipped the bi-deduction predicate with Hoare-style pre- and post-conditions to handle game statefulness; and random sampling usage is tracked through *randomness constraints* that ensure that the constructed simulator does not directly access the game's random samplings. This approach, which showed good initial results, may become the standard way of handling cryptographic assumptions in Squirrel.

5. CONCLUSION

We have developed a logic based on the CCSA logic described in [Bana and Comon-Lundh 2014], along with proof systems for verifying reachability and equivalence properties within that framework. Our work demonstrates that this approach offers a straightforward, high-level strategy for conducting computer-assisted proofs of cryptographic protocols, offering asymptotic security guarantees within the computational model. This is reinforced by the development of the interactive prover Squirrel and its application across different case studies [Baelde et al. 2021; Baelde et al. 2022; Cremers et al. 2022; Baelde et al. 2024], summarized in Figure 4. The number of LoC mentioned includes both the model and the proof script. The cryptographic assumptions on which the proof relies are also indicated, as well as the security properties under study. We analyze reachability properties, namely various forms of authentication, and some equivalence-based properties, *e.g.* unlinkability on several RFID protocols, an anonymity property, as well as a strong form of secrecy. The case studies are divided into two blocks, with non-stateful protocols first and then stateful ones. The Squirrel prover was extended to post-quantum security proofs in [Cremers et al. 2022], the corresponding protocols with security guarantees against a quantum attacker are indicated with a ⋆.

Ongoing and Future Work. As explained in Section 3, the Squirrel tool allows cryptographic protocols to be described in a process algebra, whereas the translation from protocols to the logic relies on the definition of mutually recursive functions modeling the protocol observables depending on the execution trace. To ease formal proofs, this translation does not follow the granularity of elementary execution steps in the process algebra. Instead, it groups together elementary steps in blocks following specific patterns. Whereas the soundness of the translation is easy to obtain for a simple class of protocols, some works remains to be done for more complex protocols.

We also aim to improve the automation of the prover. One option is to discharge parts of the proof that do not rely on any cryptographic assumption to SMT solvers. Another option is to rely on type systems: techniques based on typing have mainly been used in symbolic models, but we are currently working to develop such a system to obtain computational guarantees in the CCSA framework.

We aim to extend the scope of the post-quantum work initiated in [Cremers et al. 2022] and to analyze hybrid post-quantum protocols that rely on a combination of asymmetric post-quantum algorithms with well-known and well-studied pre-quantum asymmetric cryptography, *e.g.* based on the discrete logarithm problem.

Lastly, as mentioned in Section 3, the current notion of security in Squirrel differs from the standard one in the computational model, as the number of sessions we consider is arbitrary but fixed, in the sense that it does not depend on the security parameter. We plan to overcome this limitation, notably by leveraging the composition result outlined in [Comon et al. 2020]. Furthermore, we intend to expand on our methodology to go beyond asymptotic guarantees and provide concrete security bounds.

ACKNOWLEDGMENTS

This work received funding from the France 2030 program managed by the French National Research Agency under grant agreement No. ANR-22-PECY-0006.

REFERENCES

David Baelde, Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos, and Solène Moreau. 2021. An Interactive Prover for Protocol Verification in the Computational Model. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 537–554.

David Baelde, Stéphanie Delaune, Adrien Koutsos, and Solène Moreau. 2022. Cracking the Stateful Nut: Computational Proofs of Stateful Security Protocols using the Squirrel Proof Assistant. In *35th IEEE Computer Security Foundations Symposium, CSF 2022, Haifa, Israel, August 7-10, 2022*. IEEE, 289–304.

David Baelde, Adrien Koutsos, and Joseph Lallemand. 2023. A Higher-Order Indistinguishability Logic for Cryptographic Reasoning. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023*. 1–13.

David Baelde, Adrien Koutsos, and Justine Sauvage. 2024. Cryptographic Reductions By Bi-Deduction. (March 2024). https://hal.science/hal-04511718

Gergei Bana and Hubert Comon-Lundh. 2012. Towards Unconditional Soundness: Computationally Complete Symbolic Attacker. In *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings (Lecture Notes in Computer Science)*, Pierpaolo Degano and Joshua D. Guttman (Eds.), Vol. 7215. Springer, 189–208.

Gergei Bana and Hubert Comon-Lundh. 2014. A Computationally Complete Symbolic Attacker for Equivalence Properties. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. ACM, 609–620.

Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, and Bryan Parno. 2021. SoK: Computer-Aided Cryptography. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 777–795.

Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. 2009. Formal certification of code-based cryptographic proofs. In *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles*

of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009, Zhong Shao and Benjamin C. Pierce (Eds.). ACM, 90–101.

Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. 2011. Computer-Aided Security Proofs for the Working Cryptographer. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings (Lecture Notes in Computer Science)*, Phillip Rogaway (Ed.), Vol. 6841. Springer, 71–90.

David A. Basin, Andreas Lochbihler, and S. Reza Sefidgar. 2020. CryptHOL: Game-Based Proofs in Higher-Order Logic. *J. Cryptol.* 33, 2 (2020), 494–566.

Bruno Blanchet. 2001. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada*. IEEE Computer Society, 82–96.

Bruno Blanchet. 2008. A Computationally Sound Mechanized Prover for Security Protocols. *IEEE Trans. Dependable Secur. Comput.* 5, 4 (2008), 193–207.

Mayla Brusò, Konstantinos Chatzikokolakis, and Jerry den Hartog. 2010. Formal Verification of Privacy for RFID Systems. In *23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*. IEEE Computer Society, 75–88.

Hubert Comon, Charlie Jacomme, and Guillaume Scerri. 2020. Oracle Simulation: A Technique for Protocol Composition with Long Term Shared Secrets. In *2020 ACM SIGSAC Conference on Computer and Communications Security, CCS 2020*. ACM, 1427–1444.

Cas Cremers, Caroline Fontaine, and Charlie Jacomme. 2022. A Logic and an Interactive Prover for the Computational Post-Quantum Security of Protocols. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 125–141.

Danny Dolev and Andrew Chi-Chih Yao. 1981. On the Security of Public Key Protocols (Extended Abstract). In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*. IEEE Computer Society, 350–357.

Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings (Lecture Notes in Computer Science)*, Natasha Sharygina and Helmut Veith (Eds.), Vol. 8044. Springer, 696–701.

Adam Petcher and Greg Morrisett. 2015. The Foundational Cryptography Framework. In *Principles of Security and Trust - 4th International Conference, POST 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings (Lecture Notes in Computer Science)*, Riccardo Focardi and Andrew C. Myers (Eds.), Vol. 9036. Springer, 53–72.

Nikhil Swamy, Catalin Hritcu, Chantal Keller, Aseem Rastogi, Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cédric Fournet, Pierre-Yves Strub, Markulf Kohlweiss, Jean Karim Zinzindohoue, and Santiago Zanella Béguelin. 2016. Dependent types and multi-monadic effects in F. In *43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016*, Rastislav Bodík and Rupak Majumdar (Eds.). ACM, 256–270.

CFP

SIGLOG MONTHLY 248

April 2024

ELLI ANASTASIADI, UPPSALA UNIVERSITY, SE



April 2024 edition of SIGLOG Monthly, featuring deadlines, calls and community announcements.

Past Issues - How to submit an announcement

1. TABLE OF CONTENT

— DEADLINES (Section 2)

— CALLS

  — GandALF 2024 (CALL FOR PAPERS) (Section 3)

  — SLSS 2024 (CALL FOR ABSTRACTS) (Section 4)

  — LearnAut 2024 (CALL FOR PAPERS) (Section 5)

  — HYPER 2024 (CALL FOR PRESENTATIONS AND PARTICIPATION) (Section 6)

  — SEFM'24 (CALL FOR PAPERS) (Section 7)

  — NLS 2024 (CALL FOR PARTICIPATION) (Section 8)

  — ILO 2024 (CALL FOR PARTICIPATION) (Section 9)

— JOB ANNOUNCEMENTS

  — POSTDOC/RESEARCH ASSOCIATE position in modal type theory and secure compilation (Section 10)

2. DEADLINES

GandALF 2024:	Apr 07, 2024 (Abstract), Apr 10, 2024 (Paper)
SLSS 2024:	Apr 07, 2024 (Submission deadline)
POSTDOC/RESEARCH ASSOCIATE position in modal type theory and secure compilation:	Apr 07, 2024 (Application deadline)
GÖDEL PRIZE 2024:	Apr 12, 2024 (Nominations)
HIGHLIGHTS 2024:	Apr 15, 2024 (Early), Jun 17, 2024 (Regular)
FM 2024:	Apr 15, 2024 (Abstracts), Apr 19, 2024 (Full papers)
WADT 2024:	Apr 15, 2024 (Abstracts), Sep 16, 2024 (Full papers)
ILO 2024:	Apr 15, 2024 (Registration)
LearnAut 2024:	Apr 18, 2024 (Submission deadline), Jul 07, 2024 (Workshop)
HYPER 2024:	Apr 25, 2024 (Submission deadline)
THIRTEENTH SUMMER SCHOOL ON FORMAL TECHNIQUES 2024:	Apr 30, 2024 (Application deadline)
CiE 2024:	May 15, 2024 (Informal presentations)
PHD AND POSTDOC POSITIONS AT UNIVERSITY OF WARSAW:	May 31, 2024 (Applications)
SEFM'24:	Jun 07, 2024 (Abstract), Jun 14, 2024 (Paper)

3. GANDALF 2024: FIFTEENTH INTERNATIONAL SYMPOSIUM ON GAMES, AUTOMATA, LOGICS, AND FORMAL VERIFICATION

Reykjavik University, Iceland
19–21 June 2024
https://scool24.github.io/GandALF/
CALL FOR PAPERS

— The Fifteenth International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 24) will be held in Reykjavik (Iceland) on June 19-21, 2024. This year, GandALF is part of the Reykjavik Summer of Cool Logic 2024 (SCooL 2024) and is co-located with the Twelfth Scandinavian Logic Symposium (SLSS 2024) and the Fifth Nordic Logic Summer School (NLS 2024).

— The aim of GandALF 2024 is to bring together researchers from academia and industry who are actively working in the fields of Games, Automata, Logics, and Formal Verification. The idea is to cover an ample spectrum of themes, ranging from theory to applications, and stimulate cross-fertilization. Papers focused on formal methods are especially welcome. Authors are invited to submit original research or tool papers on all relevant topics in these areas. Papers discussing new ideas that are at an early stage of development are also welcome.

— TOPICS: see https://scool24.github.io/GandALF/ and submission instructions see: https://scool24.github.io/SLSS/

— The proceedings will be published by Electronic Proceedings in Theoretical Computer Science. Authors of selected papers will be invited to submit a revised version of their work to a special issue of Logical Methods in Computer Science.

— The previous editions of GandALF already led to special issues of the International Journal of Foundations of Computer Science (GandALF 2010), Theoretical Computer Science (GandALF 2011 and 2012), Information and Computation (GandALF 2013, 2014, 2016, 2017, 2018, 2019, and 2020), Acta Informatica (GandALF 2015) and Logical Methods in Computer Science (GandALF 2021, 2022, and 2023).

— SUBMISSION

Submitted papers should not exceed 14 pages (excluding references and clearly marked appendices) typeset using EPTCS format (please use the LaTeX style provided at https://style.eptcs.org/), be unpublished, and contain original research. For papers reporting experimental results, authors are encouraged to make their data available with their submission. Submissions must be in PDF format and will be handled via easychair at the following address: https://easychair.org/conferences/?conf=gandalf23

— IMPORTANT DATES

Abstract submission:	Apr 07, 2024
Paper submission:	Apr 10, 2024
Acceptance notification:	May 10, 2024
Camera-ready deadline:	Jun 10, 2024
Conference dates:	19-21 Jun 2024

— Co-chairs

— Antonis Achilleos (Reykjavik University)

— Andrian Francalanza (University of Malta)

4. SLSS 2024: TWELFTH SCANDINAVIAN LOGIC SYMPOSIUM

Reykjavik University, Iceland
14-16 June 2024
https://scool24.github.io/SLSS/
CALL FOR ABSTRACTS

— The twelfth Scandinavian Logic Symposium (SLSS 2024) will be held at Reykjavik University, Iceland, during 14-16 June, 2024, under the auspices of the Scandinavian Logic Society. The previous four meetings of the SLSS were held in Bergen in Norway (2022), Gothenburg in Sweden (2018), Tampere in Finland (2014), and Roskilde in Denmark (2012).

— The primary aim of the Symposium is to promote research in the field of logic (broadly conceived) carried out in research communities in Scandinavia. Moreover, it warmly invites the participation of logicians from all over the world. The meeting will include invited lectures and a forum for participants to present contributed talks.

— For topics and submission instructions see: https://scool24.github.io/SLSS/

— Co-chairs

— Antonios Achilleos (Reykjavik University)

— Dag Westerståhl (Stockholm University, Tsinghua University)

— Invited Speakers

— Fausto Barbero (University of Helsinki)

— Sara Negri (University of Genoa)

— Aybüke Özgün (ILLC, University of Amsterdam)

— IMPORTANT DATES

Submission deadline:	Apr 07, 2024
Notification:	Apr 30, 2024
Final programme:	TBA
Conference:	Jun 14-16 2024

5. LEARNAUT 2024: LEARNING AND AUTOMATA

Tallinn University, Tallinn, Estonia, July 7th
ICALP 2024 workshop
Website: https://learnaut24.github.io/
CALL FOR PAPERS

— ABOUT

Learning models defining recursive computations, like automata and formal grammars, are the core of the field called Grammatical Inference (GI). The expressive power of these models and the complexity of the associated computational problems are major research topics within mathematical logic and computer science. Historically, there has been little interaction between the GI and ICALP communities, though recently some important results started to bridge the gap between both worlds, including applications of learning to formal verification and model checking, and (co-)algebraic formulations of automata and grammar learning algorithms.

— The aim of this workshop is to bring together experts on logic who could benefit from grammatical inference tools, and researchers in grammatical inference who could find in logic and verification new fruitful applications for their methods.

— We invite submissions of recent work, including preliminary research, related to the theme of the workshop. The Program Committee will select a subset of the abstracts for oral presentation. At least one author of each accepted abstract is expected to represent it at the workshop.

— SUBMISSION INSTRUCTIONS & TOPICS

For topics and submission instructions see: https://learnaut24.github.io/calls.html .

Note that accepted papers will be made available on the workshop website but will not be part of formal proceedings (i.e., LearnAut is a non-archival workshop). We do accept submissions of work recently published, currently under review or work-in-progress.

— IMPORTANT DATES

Submission deadline:	Apr 18, 2024
Notification of acceptance:	May 13, 2024
Early registration:	May 17 (ICALP)
Workshop:	Jul 07, 2024

— ORGANIZERS

— Sophie Fortz (King's College London, UK)

— Franz Mayr (Universidad ORT Uruguay, UY)

— Joshua Moerman (Open Universiteit, Heerlen, NL)

— Matteo Sammartino (Royal Holloway, University of London, UK)

6. HYPER 2024: 3RD WORKSHOP ON HYPERPROPERTIES: ADVANCES IN THEORY AND APPLICATIONS

Montreal, Canada, co-located with CAV 2024
July 23, 2024
https://hyperworkshop24.cispa.io/
https://easychair.org/my/conference?conf=hyper24
CALL FOR PRESENTATIONS AND PARTICIPATION

— The HYPER workshop aims to bring together researchers interested in the broad area of hyperproperties and working in the areas of formal methods and control, cybersecurity, and machine learning. Presentation proposals shall be submitted in the form of an extended abstract of up to three pages in LNCS format (not including references) via easychair. Submissions can overlap with previously published work and will be judged based on their relevance to the topic of the workshop. The workshop will have no formal proceedings.

— Submission link: https://easychair.org/my/conference?conf=hyper24

— IMPORTANT DATES

Submission deadline:	Apr 25, 2024
Notification date:	May 13, 2024
Workshop:	Jul 23, 2024

— Invited Speakers:

— Fred Schneider (Cornell University)

— Hagit Attiya (Technion)

— Jana Hofmann (Azure Research)

— Xiang Yin (Shanghai Jiao Tong University)

7. SEFM'24: 22ND INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND FORMAL METHODS

University of Aveiro, Portugal
4-8 November 2024
https://sefm-conference.github.io/2024/
CALL FOR PAPERS

— ABOUT

The conference aims to bring together researchers and practitioners from academia, industry and government, to advance the state of the art in formal methods, to facilitate their uptake in the software industry, and to encourage their integration within practical software engineering methods and tools.

— SUBMISSION INSTRUCTIONS & TOPICS

For topics and submission instructions see: https://sefm-conference.github.io/2024/callforpapers/ .

— IMPORTANT DATES

Abstract submission:	7 June 2024 (AoE)
Paper submission:	14 June 2024 (AoE)
Author notification:	Aug 15, 2024
Workshops:	Nov 4-5 2024
Conference:	Nov 6-8 2024

8. NLS 2024: FIFTH NORDIC LOGIC SUMMER SCHOOL

Reykjavik University, Iceland
10-13 June 2024
https://scool24.github.io/NLS/
CALL FOR PARTICIPATION

— The fifth Nordic Logic Summer School is arranged under the auspices of the Scandinavian Logic Society. The four previous schools were organised in Bergen in Norway (2022), Stockholm in Sweden (2017), Helsinki in Finland (2015), and Nordfjordeid in Norway (2013). The intended audience is advanced master students, PhD-students, postdocs and experienced researchers wishing to learn the state of the art in a particular subject.

— Programme Committee Co-chairs:

  — Nina Gierasimczuk (DTU Compute)

  — Lauri Hella (Tampere University)

— Members:

  — Torben Braüner (Roskilde University)

  — Fredrik Engström (University of Gothenburg)

  — Åsa Hirvonen (University of Helsinki)

  — Ana Ozaki (University of Bergen & University of Oslo)

— Invited Speakers:

  — Miika Hannula (University of Helsinki)

  — Sandra Kiefer (University of Oxford)

  — Greg Restall (University of St Andrews)

  — Jandson Ribeiro (Cardiff University)

  — Rineke Verbrugge (University of Groningen)

— Registration info at: https://scool24.github.io/fees/ and https://fienta.com/twelfth-scandinavian-logic-symposium-slss-2024

9. ILO 2024: INTERNATIONAL LOGIC OLYMPIAD

https://www.logicolympiad.org/
CALL FOR PARTICIPATION

— The International Logic Olympiad 2024 (ILO2024) – a world-wide contest on Logic for high school students.

— Key Benefits For Students:

  — Academic Achievements: Win cash prize awards and certificates that enhance your academic portfolio

— Cultural Exchange: Engage in global learning experiences and connect with peers worldwide.

— Mentorship Opportunities: Receive guidance from experts in logic and related fields

— Exclusive opportunities for Finalists: Contest finalists are invited to compete on the Stanford University campus. (Includes free tours and activities between final in-person rounds.)

— Key Benefits For For schools:

— Global Recognition: Elevate your school's reputation with international acclaim

— Networking Opportunities: Connect with top educational institutions

— Collaborative Projects: Get invited to cross-border collaborations and initiatives

— IMPORTANT DATES

Registration:	Apr 15, 2024
Final Round at Stanford:	Jul 2024

10. POSTDOC/RESEARCH ASSOCIATE POSITION IN MODAL TYPE THEORY AND SECURE COMPILATION

University of Kent
JOB ANNOUNCEMENT

— We seek a talented and motivated postdoc/Research associate to join the School of Computing at the University of Kent, Canterbury, UK. The Research Associate will participate in cutting edge research on type-based enforcement and compilation techniques for enforcement of security properties of higher-order programs. The position is on an EPSRC funded project titled "TYPDSEC: Type-based information declassification and its secure compilation". The post is based in Canterbury and will be directly supervised by Dr. Vineet Rajani (https://vineetrajani.github.io/).

— The project will be in the intersection of modal type theory, information flow control and secure compilation. It will also involve mechanisation in the HOL4 theorem prover and implementation in the CakeML ecosystem. Applicants should have a strong background in formal methods evidenced by high-quality research publications or artifacts in top-tier venues of programming languages, formal verification or security research. The project will involve close collaboration with Prof. Magnus Myreen (Chalmers).

— As a Research Associate you will:

— build novel type theories, proof techniques and compilation methods to reason about hyperproperties relevant for security of higher-order programs.

— work on mechanisation of the above in HOL4.

— integrate them in the CakeML framework, a real world compiler for the ML like language.

— To be successful in this role you must:

— have a PhD or equivalent experience in Computing or in a related discipline.

— have a track record of peer-reviewed publications at scientific workshops, conferences or journals.

— have excellent mathematical skills relevant for analysis of computer programs.

— The university of Kent is walking distance from the charming city of Canterbury. It has a high speed connection to London, and travel to Europe is convenient by rail or car. Please apply through any of the following URLs:

— https://www.jobs.ac.uk/job/DGO984/research-associate

— https://jobs.kent.ac.uk/Vacancy.aspx?id=7355&forced=2

— We are looking to have the successful candidate start as soon as possible. For any queries formal or informal, please do not hesitate to get in touch Dr. Vineet Rajani (v.rajani@kent.ac.uk).

Application deadline: Apr 07, 2024

Links: SIGLOG website, LICS website, SIGLOG Monthly