Math-for-industry
Education & Research Hub

ISSN2188-1200

IMI Workshop of the Joint Usage Research Projects

Workshop on Logic, Algebra and Category Theory: Theory and Applications

Editors : Guillermo Badia, Daniel Găină and Tomasz Kowalski

九州大学マス・フォア・インダストリ研究所

MI Lecture Note Vol.107 : Kyushu University

IMI Workshop of the Joint Usage Research Projects

Workshop on Logic, Algebra and Category Theory: Theory and Applications

Editors: Guillermo Badia, Daniel Găină and Tomasz Kowalski

About MI Lecture Note Series

The Math-for-Industry (MI) Lecture Note Series is the successor to the COE Lecture Notes, which were published for the 21st COE Program "Development of Dynamic Mathematics with High Functionality," sponsored by Japan's Ministry of Education, Culture, Sports, Science and Technology (MEXT) from 2003 to 2007. The MI Lecture Note Series has published the notes of lectures organized under the following two programs: "Training Program for Ph.D. and New Master's Degree in Mathematics as Required by Industry," adopted as a Support Program for Improving Graduate School Education by MEXT from 2007 to 2009; and "Education-and-Research Hub for Mathematics-for-Industry," adopted as a Global COE Program by MEXT from 2008 to 2012.

In accordance with the establishment of the Institute of Mathematics for Industry (IMI) in April 2011 and the authorization of IMI's Joint Research Center for Advanced and Fundamental Mathematics-for-Industry as a MEXT Joint Usage / Research Center in April 2013, hereafter the MI Lecture Notes Series will publish lecture notes and proceedings by worldwide researchers of MI to contribute to the development of MI.

October 2022
Kenji Kajiwara
Director, Institute of Mathematics for Industry

Workshop on Logic, Algebra and Category Theory: Theory and Applications

MI Lecture Note Vol.107, Institute of Mathematics for Industry, Kyushu University
ISSN 2188-1200
Date of issue: February 10, 2026
Editor: Guillermo Badia, Daniel Găină and Tomasz Kowalski
Institute of Mathematics for Industry, Kyushu University
Graduate School of Mathematics, Kyushu University
Motooka 744, Nishi-ku, Fukuoka, 819-0395, JAPAN
Tel +81-(0)92-802-4402, Fax +81-(0)92-802-4405
URL https://www.imi.kyushu-u.ac.jp/

Preface

Aims and Scope The 2nd Workshop on Logic, Algebra and Category Theory (LAC 2025), held in Fukuoka between September 29 and October 3 in 2025, continued the LAC series' objective of fostering interactions between logic, algebra, and category theory, with a strong focus on their applications to computer science. Building on the success of the first edition, LAC 2025 aimed to provide a forum for researchers interested in foundational mathematical methods and their use in areas such as specification, verification, database theory, and programming language semantics.

As outlined in the workshop abstract, one of the core motivations of LAC 2025 was the observation that many specification languages and formal systems are supported by dedicated logics that are often developed in isolation. This proliferation of logical systems can obscure common structures and relationships between logical properties. The workshop therefore emphasized the use of category theory and universal algebra as unifying tools, enabling abstract and reusable reasoning principles that apply across a wide range of logical systems. By focusing on such general mathematical frameworks, the workshop sought to clarify causal relations between logical properties and to support principled language and tool design.

Participants and format LAC 2025 successfully brought together a diverse community of participants from both academia and industry, underscoring the workshop's mission to bridge foundational theory with real-world applications. The audience included early-career researchers (ECRs), who gained valuable exposure to cutting-edge developments and benefited from meaningful interactions with senior experts experienced in both theoretical advances and practical implementations.

Most of the talks were delivered face-to-face, enabling intensive discussions during sessions and informal exchanges during breaks. At the same time, the workshop successfully incorporated a significant number of online speakers, which broadened the participation and allowed contributors who could not travel to present their work and engage in discussions.

Speakers from industry A distinctive feature of LAC 2025 was the presence of high-profile speakers from the industrial sector, highlighting the relevance of logic- and algebra-based methods in real-world systems.

One such talk was delivered by Hung Q. Ngo (RelationalAI), entitled "Optimization Techniques for a Datalog-Inspired Query Language." His presentation focused on the query optimizer used in RelationalAI's logic engine and surveyed a collection of optimization and evaluation techniques developed in the database theory community over the past 15 years. These included worst-case optimal join algorithms, sum-product queries over semirings, variable elimination, tree decompositions, and tensor decompositions. The talk illustrated how these theoretically grounded techniques can be combined to efficiently evaluate expressive

Datalog-inspired query languages, demonstrating a clear and compelling transfer of foundational research into industrial practice.

Another invited industrial talk was given by Jonathan Lenchner (IBM Research), titled "On Some Relatives of the Ehrenfeucht–Fraïssé Game and Software for Helping with the Analysis of these Games." Lenchner introduced the Multi-Structural (MS) game, a variation of the classical Ehrenfeucht–Fraïssé game that measures the total number of quantifiers required to express a first-order sentence, rather than just quantifier depth. He argued that this measure can be strictly more powerful and informative in certain settings. A key aspect of the talk was the presentation of software tools designed to support the analysis of such games, thereby linking abstract model-theoretic concepts with practical computational assistance.

Speakers from academia Beyond Hung Ngo's contribution, the workshop featured several additional talks in database theory, underscoring the strong presence of this topic at LAC 2025. Guillermo Badia presented "Containment of Conjunctive Queries with Equations and Disequations for Databases over Semiring", addressing query containment problems in enriched algebraic settings. Carles Noguera, who participated online, gave a talk entitled "Fagin's Theorem for Semiring Turing Machines", extending classical results in descriptive complexity to semiring-based computation models.

Another major theme of the workshop was the Maude system and its underlying theory. Two talks were directly dedicated to Maude and its tooling: Santiago Escobar presented "Unification and Narrowing in Maude 3.5", reporting on recent advances in unification and narrowing techniques in the latest version of Maude, while Francisco Durán introduced "NuITP: An Inductive Theorem Prover for Maude", showcasing an inductive theorem prover tightly integrated with the Maude environment.

In addition, the program included talks on transition algebra, which provide a theoretical foundation for the Maude strategy language. These contributions strengthened the conceptual links between algebraic theory and practical strategy-based rewriting highlighting how abstract algebraic frameworks can directly inform and justify concrete language design and tool support.

General remarks Beyond the individual technical contributions, LAC 2025 highlighted how advances in logic, algebra, and category theory can have a ripple effect across multiple areas of computer science. New theoretical insights presented at the workshop influence the design of specification languages, database query optimization, and strengthen the foundations of tools such as Maude. In turn, these developments propagate to practical applications in verification, data management, and industrial systems. By bringing together foundational theory and industrial perspectives, the workshop fostered interactions whose impact shapes future research directions and enables the transfer of ideas across traditionally separate communities.

Details about the workshop, including program abstracts and slides, can be

found at the following link: https://imi.kyushu-u.ac.jp/lac/2025/index.html. This work was supported by the Institute of Mathematics for Industry, Joint Usage/Research Center in Kyushu University. (FY2025 Grant for Project Workshop I Joint Research "Logic, Algebra and Category Theory: Applications in Computer Science" (2025b002))

International Project Research

2025 9/29 -- 10/3 Hybrid

IMI Auditorium (W1-D-413), West Zone 1,
Ito campus, Kyushu University

Logic, Algebra and Category Theory:

Applications in Computer Science

— *Keynote speakers:*

Jonathan Lenchner *(IBM research, USA)*
Hung Q. Ngo *(relationalAI, USA)*

— *Principal Investigator:*

Guillermo Badia

— *Organizing Committee:*

Guillermo Badia *(The University of Queensland, Australia)*
Ronald Fagin *(IBM Research, USA)*
Daniel Gaina *(Kyushu University, Japan)*
Tomasz Kowalski *(Jagiellonian University, Poland)*
Yoshihiro Mizoguchi *(Kyushu University, Japan)*
Adrian Riesco Rodrigues *(Complutense University of Madrid, Spain)*
Ionut Tutu *(Simion Stoilow Institute of Mathematics of the Romanian Academy, Romania)*
Santiago Escobar *(Universitat Politècnica de València, Spain)*

— *Co-organized:*

IMI Kyushu University
Universitat Politècnica de València

— *website:*
https: imi.kyushu-u.ac.jp/lac/

— *email:*
lac2025@imi.kyushu-u.ac.jp

九州大学 KYUSHU UNIVERSITY

Institute of Mathematics for Industry Kyushu University

Joint Research Center for Advanced and Fundamental Mathematics-for-Industry
文部科学大臣認定「産業数学の先進的・基礎的共同研究拠点」
九州大学マス・フォア・インダストリ研究所

E-mail:imikyoten@jimu.kyushu-u.ac.jp
https://joint.imi.kyushu-u.ac.jp/post-18132/

Program

Monday (Chair: Daniel Găină)

10:00 – 10:10 Opening: Daniel Găină

10:10–11:10 Hung Q. Ngo

Optimization techniques for a Datalog-inspired query language

11:10–11:40 Guillermo Badia

Containment of Conjunctive Queries with Equations and Disequations for Databases over Semiring

11:40–14:30 Lunch break

14:30–15:00 Sasha Rubin

Tight inference and real-valued logic

15:00–15:30 Krzysztof Krawczyk

Structural completeness among finitary extensions of R-mingle

Tuesday (Chair: Ionut Tutu)

10:00–10:30 Hiroakira Ono

A glance at extensions of bi-intuitionistic logic

10:30–11:00 Zbyszek Krol

On some forms of logical connections between theories

11:00–12:00 Mark Reynolds, A tree-shaped tableau for Linear Time Temporal Logic

12:00–14:00 Lunch break

14:00–15:00 Kazuhiro Ogata

Formal Specification and Verification of Post-quantum Cryptographic Protocols with Proof Scores

15:00–16:00 Kokichi Futatsugi

Constructing proof scores in CafeOBJ

Wednesday (Chair: Guillermo Badia)

10:00–11:00 Jonathan Lenchner

On Some Relatives of the Ehrenfeucht-Fraïssé Game and Software for Helping with the Analysis of these Games

11:00–12:00 Tomasz Kowalski

Hybrid-Dynamic Ehrenfeucht-Fraïssé Games

12:00–14:30 Lunch break

14:30–15:00 Diamant Pireva

Many-Sorted First-Order Logic with Quantification over Inter-Sort Functions

15:00–15:30 Carles Noguera

Fagin's Theorem for Semiring Turing Machines

Thursday (Chair: Adrian Riesco)

10:00–11:00 Santiago Escobar

Unification and Narrowing in Maude 3.5

11:00–12:00 Francisco Durán

NuITP: An Inductive Theorem Prover for Maude

12:00–14:00 Lunch break

14:00–15:00 Marcel Jackson

Minimal signatures for undecidability of representability for relation algebras

15:00–16:00 Narciso Martí Oliet and Ruben Rubio

Strategies, qualitative and quantitative model checking in Maude

16:00–16:30 Beatriz Alcaide García

MongoDB specification in Maude

Friday (Chair: Tomasz Kowalski)

10:00–11:00 Ionuț Țuțu

Forcing, Transition Algebras, and Calculi

11:00–12:00 Adrián Riesco Rodríguez

Executable Specifications in Transition Algebra

12:00–14:00 Lunch break

14:00–14:30 Go Hashimoto

Model-theoretic Forcing in Transition Algebra

Contents

RelationalAI

Logic, Algebra, & Category Theory 2025

Optimization Techniques for a Datalog-Inspired Query Language

Hung Q. Ngo

Outline

Computational Problems in `Rel`

```
def Δ_exists { exists( (a,b,c) | E(a,b) and E(b,c) and E(c,a)) }
def num_Δ { // 4,159,226
    count[ [a,b,c] : E(a,b) and E(b,c) and E(a,c) and a<b and b<c]
}

def distance[u, v]: {
    min[ [len] :
        len = 0 and u = source and u = v or
        len = min[[w,z] : z = distance[u, w] + 1 and E(w,v)]
    ]
}

// Number of paths of lenght 4, 63,966,233,625,222
def num_4_paths { count[(a,b,c,d,e) : E(a,b) and E(b,c) and E(c,d) and E(d,e)] }
```

There are customer's `Rel` programs with thousands of mutually recursive rules.

Queries in `Rel`

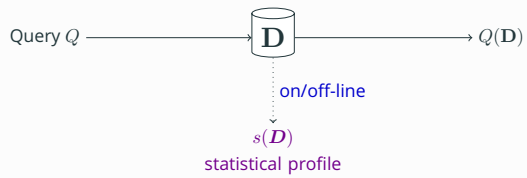
First order logic queries with

- Built-in predicates such as arithmetic ops, hashing, string ops, $\cdots$
- Aggregations (reduce operator such as sum, max, min, $\cdots$)
- Recursion (fixpoint operator) with negation, aggregation

Focus of this talk:

- Conjunctive queries
- Aggregation queries over a conjunctive body

The Main Query Optimization / Evaluation Problem



The Query Optimization Problem

Given Q and $\mathbf{D}$, compute $Q(\mathbf{D})$ in the most efficient (optimal!?) way possible.

Precise Problem Formulation

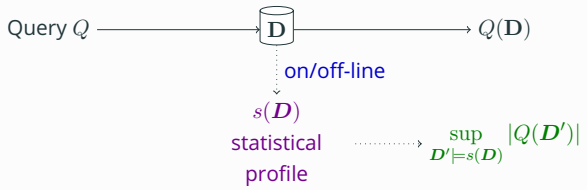
- What do we mean by "query"?
  - Full conjunctive queries
  - Sum-product queries
- What is in the statistical profile $s(\boldsymbol{D})$?
  - Degree constraints
  - $\cdots$ Frequency moment constraints, histograms, samples, ML models
- What do we mean by "optimality"?
  - Worst-Case Optimality
  - Fixed-Parameter Tractability, Fine-Grained Complexity
  - $\cdots$ Instance Optimality

Note: Optimizer designed to work before seeing Q.

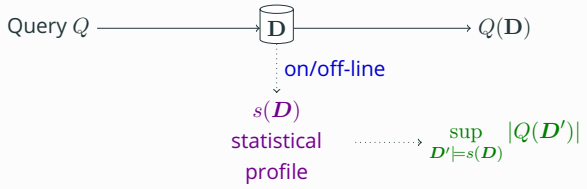
- Optimizer = *Meta-Algorithm* (input: problem, output: algorithm)

Outline

Output Cardinality Bound



Worst-Case Optimal Join (WCOJ) Algorithm



Definition

A "worst-case optimal" join algorithm is an algorithm computing $Q(\boldsymbol{D})$ in time

$$\tilde{O}\left(|\boldsymbol{D}| + \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')|\right)$$

$\tilde{O}$ hides $\log$ and query dependent factors

(For Now) Assume Q is a Full Conjunctive Queries

In a movie database

```
Q(director, actor, movie, actor_age, name) ←
      parent(director, actor)
      ∧ acted_in(actor, movie)
      ∧ director_of(director, movie)
      ∧ age(actor, actor_age) ∧ (20 < actor_age ∨ actor_age != 10)
      ∧ person_name(director, name) ∧ regex_match(".*spiel.*", name)
```

In a graph database with edge relation E,

```
Q(a, b, c) ← E(a, b) ∧ E(a, c) ∧ E(b, c)
```

(For Now) Assume Q is a Full Conjunctive Queries

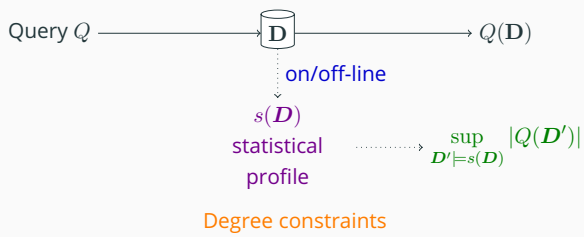
More generally, $\mathcal{H} = (V, \mathcal{E})$ is the hypergraph of a query:

$$Q(\boldsymbol{X}_V) \leftarrow \bigwedge_{S \in \mathcal{E}} R_S(\boldsymbol{X}_S)$$

For example `Q(a, b, c) ← E(a, b) ∧ E(a, c) ∧ E(b, c)`

- $V = \{a, b, c\}$
- $\mathcal{H} = (V, \mathcal{E}) = (V, \{ab, ac, bc\})$
- $R_F = E$ for all $F \in \mathcal{E}$.

What is in the Statistical Profile $s(D)$?



Degree Constraints [ANS 16, ANS 17]

Relation $R(\text{actor}, \text{movie}, \text{role})$, imagine a frequency vector $\boldsymbol{d}_{\text{actor}}$ (billions of entries)

actor	movie	role
alice		
bob		
bob		
bob		
bob		
carol		
carol		

$$d_{\text{actor}}(\text{alice}) = 1 \qquad d_{\text{actor}}(\text{bob}) = 4$$
$$d_{\text{actor}}(\text{carol}) = 2$$
$$d_{\text{actor}}(v) = 0 \qquad v \notin \{\text{alice}, \text{bob}, \text{carol}\}$$

The profile $s(\boldsymbol{D})$ contains degree constraints:

- $\|\boldsymbol{d}_{\text{actor}}\|_\infty = 4$ (degree constraint!)
- $\|\boldsymbol{d}_\emptyset\|_\infty = 7 = |R|$ (cardinality constraint!)
- $\|\boldsymbol{d}_{\text{actor},\text{movie}}\|_\infty = 1$ (functional dependency)

General DC : (X, Y, N) in relation R means $|\pi_Y \sigma_{X=\boldsymbol{x}} R| \leq N, \ \forall \boldsymbol{x}$

Hierarchy of Output Cardinality Bounds Under Degree Constraints

$$\log \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')| = \text{combinatorial-bound}(Q, s) \quad \text{computable (?) but impractical}$$
$$\leq \text{entropic-bound}(Q, s) \quad \text{not (known to be) computable}$$
$$\leq \text{polymatroid-bound}(Q, s) \quad \text{[ANS 17]}$$
$$\leq \text{flow-bound}(Q, s, \sigma) \quad \text{[IMNP 25]}$$
$$\leq \text{chain-bound}(Q, s, \sigma) \quad \text{[N 18]}$$
$$\leq \text{agm-bound}(Q, s) \quad \text{[AGM 08]}$$
$$\leq \text{integral-edge-cover}(Q, s)$$

- We settle for algorithms running in $\tilde{O}\left(|\boldsymbol{D}| + \text{bound}(Q, s)\right)$
- Algorithm is a WCOJ algorithm when the bound is tight

Hierarchy of Join Algorithms

$$\text{(not achievable ?)} \quad \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')|$$
$$\text{(not achievable ?)} \leq \text{entropic-bound}(Q, s)$$
$$\text{(IAAT)} \leq \text{polymatroid-bound}(Q, s) \quad \text{PANDA – [ANS 17]}$$
$$\text{(VAAT)} \leq \text{chain-bound}(Q, s, \sigma) \quad \text{NPRR, LFTJ, GJ}$$
$$\text{(VAAT)} \leq \text{agm-bound}(Q, s) \quad \text{NPRR, LFTJ, GJ}$$
$$\text{(JAAT)} \leq \text{integral-edge-cover}(Q, s) \quad \text{Binary Join Plans}$$

Some bounds are in log-scale, where runtime = $\tilde{O}(|\boldsymbol{D}| + 2^{\text{bound}})$

Hierarchy of Join Algorithms

JAAT	Join at a time
VAAT	Variable at a time
IAAT	Inequality at a time

The Algorithm is in the Pudding

Proof $\Longrightarrow$ Algorithm!

Example: Output Cardinality Bound for the Triangle Query

$$Q_\triangle(A,B,C) \leftarrow R(A,B) \wedge S(B,C) \wedge T(A,C)$$

AGM-bound for Q: (E.g. num triangles in a graph $\leq |E|^{3/2}$)

$$|Q_\triangle| \leq |R|^{\lambda_R} \cdot |S|^{\lambda_S} \cdot |T|^{\lambda_T}$$

whenever $\boldsymbol{\lambda} = (\lambda_R, \lambda_S, \lambda_T)$ is a fractional edge cover for the triangle:

$$\begin{aligned} \lambda_R + \lambda_S &\geq 1 \\ \lambda_R + \lambda_T &\geq 1 \\ \lambda_S + \lambda_T &\geq 1 \end{aligned} \qquad \boldsymbol{\lambda} \geq \mathbf{0}$$

Pick $\boldsymbol{\lambda}$ to minimize the bound.

Example: Proving the AGM Bound for the Triangle Query

Consider a "section" of this query on a given value $a \in \mathrm{Dom}(A)$:

$$Q_\triangle(a, B, C) \leftarrow R(a, B) \wedge S(B,C) \wedge T(a, C)$$

Need (b,c) in the intersection $S \cap (\sigma_{A=a}R \times \sigma_{A=a}T)$, thus

$$\begin{aligned} |\sigma_{A=a}Q_\triangle| &\leq \min\{|S|, |\sigma_{A=a}R| \cdot |\sigma_{A=a}T|\} \\ &\leq |S|^{\lambda_S} \cdot (|\sigma_{A=a}R| \cdot |\sigma_{A=a}T|)^{1-\lambda_S} \\ &\leq |S|^{\lambda_S} \cdot |\sigma_{A=a}R|^{\lambda_R} \cdot |\sigma_{A=a}T|^{\lambda_T} \end{aligned}$$

We used $\lambda_S + \lambda_R \geq 1$ and $\lambda_S + \lambda_T \geq 1$

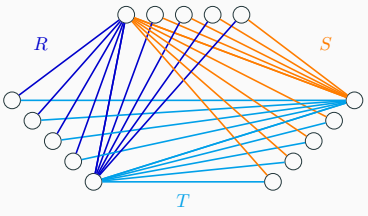
Triangle Query: AGM Bound Based on Hölder Inequality

Iterate over all possible values of a:

$$
\begin{aligned}
|Q_\triangle| &= \sum_a |\sigma_{A=a} Q_\triangle| \\
&\le \sum_a |S|^{\lambda_S} \cdot |\sigma_{A=a} R|^{\lambda_R} \cdot |\sigma_{A=a} T|^{\lambda_T} \\
&= |S|^{\lambda_S} \cdot \sum_a |\sigma_{A=a} R|^{\lambda_R} \cdot |\sigma_{A=a} T|^{\lambda_T} \\
\text{(Hölder)} &\le |S|^{\lambda_S} \cdot \Big(\sum_a |\sigma_{A=a} R|\Big)^{\lambda_R} \cdot \Big(\sum_a |\sigma_{A=a} T|\Big)^{\lambda_T} \qquad \text{recall } \lambda_R + \lambda_S \ge 1 \\
&= |S|^{\lambda_S} \cdot |R|^{\lambda_R} \cdot |T|^{\lambda_T}
\end{aligned}
$$

Example: $|Q_\triangle| \le \sqrt{|R| \cdot |S| \cdot |T|}$

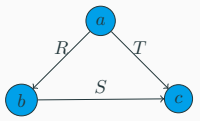
Example: JAAT Query Plans Are Sub-Optimal for Triangle Query

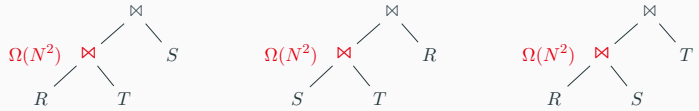


$$|R| = |S| = |T| = 2N - 1$$

$$Q_\triangle(a, b, c) = R(a, b) \wedge S(b, c) \wedge T(a, c)$$

$$\sup_{\mathbf{D}' \models \mathrm{DC}(\boldsymbol{D})} |Q_\triangle(\mathbf{D}')| = O(N^{1.5})$$





Triangle Query: VAAT Algorithm

Inspired by the proof

$$Q_\triangle(A, B, C) \leftarrow R(A, B), S(B, C), T(A, C)$$

Algorithm 1: based on Hölder's inequality proof

for $a \in \pi_A R \cap \pi_A T$ **do**
  for $b \in \pi_B \sigma_{A=a} R \cap \pi_B S$ **do**
    for $c \in \pi_C \sigma_{B=b} S \cap \pi_C \sigma_{A=a} T$ **do**
      Report (a, b, c);

In words, we followed exactly what the proof did:

- For each $a \in \pi_A R \cap \pi_A T$, enumerate $(a, b, c) \in \sigma_{A=a} Q_\triangle$

Triangle Query: VAAT is in the Pudding

Computing this "section" of the query on a given value $a \in \mathrm{Dom}(A)$

$$Q_\triangle(a, B, C) \leftarrow R(a, B), S(B, C), T(a, C)$$

is to compute the intersection $S \cap (\sigma_{A=a}R \times \sigma_{A=a}T)$, which can be done in time

$$\tilde{O}\left(\min\{|S|, |\sigma_{A=a}R| \cdot |\sigma_{A=a}T|\}\right) \leq \tilde{O}\left(|\sigma_{A=a}R|^{\lambda_R} \cdot |\sigma_{A=a}T|^{\lambda_T} \cdot |S|^{\lambda_S}\right)$$

Overall, the algorithm runs in time (Modulo $\tilde{O}(N)$ pre-processing)

$$\tilde{O}\left(\sum_a |\sigma_{A=a}R|^{\lambda_R} \cdot |\sigma_{A=a}T|^{\lambda_T} \cdot |S|^{\lambda_S}\right) = \tilde{O}\big(|S|^{\lambda_S} \cdot |R|^{\lambda_R} \cdot |T|^{\lambda_T}\big)$$

Full Conjunctive Query

$$Q(V) \leftarrow \bigwedge_{S \in \mathcal{E}} R_S(S)$$ Query hypergraph $\mathcal{H} = (V, \mathcal{E})$

AGM-bound for Q: Assuming only cardinality constraints $(\emptyset, S, |R_S|)$

$$|Q| \leq \prod_{S \in \mathcal{E}} |R_S|^{\lambda_S}$$

whenever $\boldsymbol{\lambda} = (\lambda_S)_{S \in \mathcal{E}}$ is a fractional edge cover for $\mathcal{H}$:

$$\forall v \in V \ : \ \sum_{S \in \mathcal{E}, v \in S} \lambda_S \geq 1 \qquad \boldsymbol{\lambda} \geq \mathbf{0}.$$

Pick $\boldsymbol{\lambda}$ to minimize the bound.

Full Conjunctive Query: AGM Bound from Hölder Inequality

Consider a "section" of this query on a given value $a \in \mathrm{Dom}(A)$:

$$\sigma_{A=a}Q(V) \leftarrow \bigwedge_{S \in \mathcal{E}, A \notin S} R_S(S) \wedge \bigwedge_{S \in \mathcal{E}, A \in S} \sigma_{A=a}R_S(S)$$

Now iterate over all possible values of a:

$$|Q| = \sum_a |\sigma_{A=a}Q_\triangle| \leq \sum_a \prod_{S \in \mathcal{E}, A \notin S} |R_S|^{\lambda_S} \cdot \prod_{S \in \mathcal{E}, A \in S} |\sigma_{A=a}R_S|^{\lambda_S}$$

(Hölder's inequality) $$\leq \prod_{S \in \mathcal{E}, A \notin S} |R_S|^{\lambda_S} \cdot \prod_{S \in \mathcal{E}, A \in S} \left(\sum_a |\sigma_{A=a}R_S|\right)^{\lambda_S}$$

$$= \prod_{S \in \mathcal{E}} |R_S|^{\lambda_S}.$$

Full Conjunctive Query: VAAT Algorithm [NPRR 12] [Veldhuizen 14]

$$Q(V) \leftarrow \bigwedge_{S \in \mathcal{E}} R_S(S)$$

Algorithm 2: based on Hölder's inequality proof

for $a \in \bigcap_{S \in \mathcal{E}, A \in S} \pi_A R_S$ **do**

Recursively solve the query section $\sigma_{A=a}Q$ (on variables $V - \{A\}$);
Report $\{a\} \times \sigma_{A=a}Q$

Runtime $\tilde{O}\left(\sum_{S \in \mathcal{E}} |R_S|\right) + \tilde{O}\left(\prod_{S \in \mathcal{E}} |R_S|^{\lambda_S}\right)$. Proof: straightforward.

Reminder: Hierarchy of Join Algorithms

(not achievable ?) $\sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')|$

(not achievable ?) $\leq$ entropic-bound(Q, s)

(IAAT) $\leq$ polymatroid-bound(Q, s) PANDA – [ANS 17]

(VAAT) $\leq$ chain-bound(Q, s, σ) NPRR, LFTJ, GJ

(VAAT) $\leq$ agm-bound(Q, s) NPRR, LFTJ, GJ

(JAAT) $\leq$ integral-edge-cover(Q, s) Binary Join Plans

Some bounds are in log-scale, where runtime = $\tilde{O}(|\boldsymbol{D}| + 2^{\text{bound}})$

Polymatroids

Polymatroid is a function $h : 2^V \to \mathbb{R}_+$ such that $h(\emptyset) = 0$ and

- $h(X) \leq h(Y)$ if $X \subseteq Y$ monotonicity
- $h(X) + h(Y) \geq h(X \cup Y) + h(X \cap Y)$ submodularity

Let Γ_n be the set of all polymatroids on n variables.

The Polymatroid Bound

Theorem (ANS 17)

If $s(\boldsymbol{D})$ contains only degree constraints, then

$$\log \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')| \leq \max_{h \in \Gamma_n \cap DC} h(V)$$

where DC is the set of linear constraints of the form

$$h(X \cup Y) - h(X) \leq \log N, \qquad \textit{for each degree constraint } (X, Y, N),$$

The PANDA algorithm [ANS 17] achieves this bound

Outline

Worst-Case Optimality is Insufficient for Queries with Aggregations

4-cycle detection

- $Q() \leftarrow \exists A, B, C, D\ E(A, B) \wedge E(B, C) \wedge E(C, D) \wedge E(D, A)$
- Worst-case size bound on $|Q|$ is 1, on the conjunction is $|E|^2$
- Answerable in $O(|E|^{\frac{3}{2}})$-time [AYZ 97]

k-cycle detection

- Worst-case size bound on the conjunction is $O(|E|^{\lceil k/2 \rceil})$
- k-cycle detection can be done in $O(|E|^{2-\frac{1}{\lceil k/2 \rceil}})$-time [AYZ 97]

Worst-Case Optimality is Insufficient for Queries with Aggregations

k-walk counting (or any sub-trees in general)

- $Q = \text{count}[X_1, \ldots, X_{k+1} : E_1(X_1, X_2) \wedge \cdots \wedge E(X_k, X_{k+1})]$
- Worst-case size bound on $|Q|$ is 1, on the conjunction is $|E|^{\lceil k/2 \rceil}$
- But, we can answer this in $\tilde{O}(k|E|)$-time.

Conjunctive query with negation

- $Q() \leftarrow R(X, Y) \wedge S(Y, Z) \wedge T(Z, U) \wedge X \neq U$
- Worst-case size bound on $|Q|$ is 1, on the conjunction is $O(N^2)$
- But, we can answer this in $\tilde{O}(N)$-time.

Need Two New Ideas in Our Framework

Idea 1: an abstraction to capture aggregation queries.

- $\max$, $\min$, $\sum$, $\exists$, etc.

Idea 2: a better notion of "optimality"

- Fine-grained complexity, parameterized complexity.

Idea 1: Sum-Product-Queries — Proposed and Studied in AI

Given a semiring $(D, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, a Sum-Product-query is

$$\varphi(\boldsymbol{X}_F) := \bigoplus_{\boldsymbol{X}_{V-F}} \bigotimes_{S \in \mathcal{E}} \psi_S(\boldsymbol{X}_S)$$

Example 1: $(\{\text{true}, \text{false}\}, \vee, \wedge, \text{false}, \text{true})$ is the Boolean semiring.

$$Q() = \bigvee_{A,B,C} E(A, B) \wedge E(A, C) \wedge E(B, C)$$

Example 2: $(\mathbb{R}, +, \times, 0, 1)$ is the sum-product semiring.

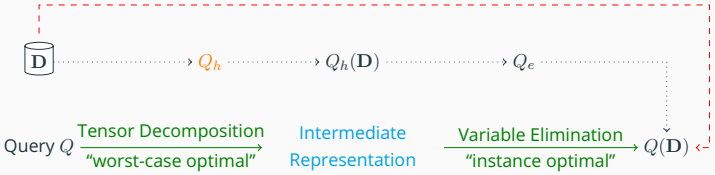
$$Q(A) = \sum_{B,C,D} E(A, B) \times E(B, C) \times E(C, D) \times \mathbf{1}_{A \neq D}$$

Idea 2: Better Notion of Optimality — Fixed Parameter Tractability

$$\tilde{O}\left(|\boldsymbol{D}| + \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')|\right) \Longrightarrow \tilde{O}\left(|\boldsymbol{D}| + \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} f(Q, \boldsymbol{D}') + |Q(\boldsymbol{D})|\right)$$

Accompanied by a meta algorithm

$Q \in$ { database, CSP, PGM, logic, graph, linear algebra } queries



Variable Elimination — $Q = \textbf{count}[A, B, C, D : R(A, B) \wedge S(B, C) \wedge T(C, D)]$

Overloading notation

$$R(a, b) := \mathbf{1}_{(a,b)\in R} \qquad S(b, c) := \mathbf{1}_{(b,c)\in S} \qquad T(c, d) := \mathbf{1}_{(c,d)\in T}$$

Variable elimination [ZP 94] — Works for any semi-ring!

$$\begin{aligned} Q() &= \sum_a \sum_b \sum_c \sum_d R(a, b) \cdot S(b, c) \cdot T(c, d) \\ &= \sum_a \sum_b \sum_c R(a, b) \cdot S(b, c) \cdot \sum_d T(c, d) = \sum_a \sum_b \sum_c R(a, b) \cdot S(b, c) \cdot W(c) \\ &= \sum_a \sum_b R(a, b) \cdot \sum_c S(b, c) \cdot W(c) = \sum_a \sum_b R(a, b) \cdot V(b) \end{aligned}$$

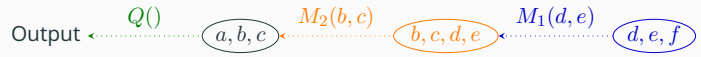
Database jargon: (aggregation | projection | predicate) pushdowns

Indicator Projection, and Fractional Hypertree Width — [GM 06] [ANR 16]

$$\begin{aligned} Q &= \sum_{a,b,c,d,e,f} R(a, b)S(a, c)T(b, c, d, e)W(e, f)V(d, f) \\ &= \sum_{a,b,c,d,e} R(a, b)S(a, c)T(b, c, d, e) \underbrace{\sum_f \pi_{de} T(d, e)\, W(e, f)V(d, f)}_{\tilde{O}(N^{3/2}),\ \text{WCOJ}} \\ &= \sum_{a,b,c,d,e} R(a, b)S(a, c)T(b, c, d, e)M_1(d, e) \\ &= \sum_{a,b,c} R(a, b)S(a, c) \sum_{d,e} \pi_b R(b) \pi_S(c)\, T(b, c, d, e)M_1(d, e) \\ &= \sum_{a,b,c} R(a, b)S(a, c)M_2(b, c) \qquad = \underbrace{\sum_{a,b,c} R(a, b)S(a, c)M_2(b, c)}_{\tilde{O}(N^{3/2}),\ \text{WCOJ}} \end{aligned}$$

Variable Elimination, Message Passing, Belief Propagation, Yannakakis

$$Q = \sum_{a,b,c,d,e,f} R(a,b)S(a,c)T(b,c,d,e)W(e,f)V(d,f)$$



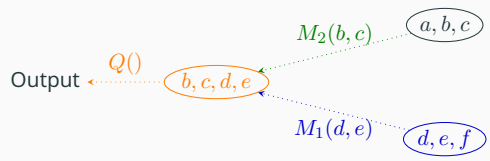
- Yannakakis algorithm [Yannakakis 81]
- Belief propagation [Pearl 82]

A Different Variable Elimination Order for the Same Query

$$\begin{aligned}
Q &= \sum_{a,b,c,d,e,f} R(a,b)S(a,c)T(b,c,d,e)W(e,f)V(d,f) \\
&= \sum_{a,b,c,d,e} R(a,b)S(a,c)T(b,c,d,e) \underbrace{\sum_{f} \pi_{de}T(d,e)\, W(e,f)V(d,f)}_{M_1(d,e) \text{ in } \tilde{O}(N^{3/2}),\ \text{WCOJ}} \\
&= \sum_{b,c,d,e} T(b,c,d,e)M_1(d,e) \underbrace{\sum_{a} \pi_{bc}T(b,c)R(a,b)S(a,c)}_{M_2(b,c) \text{ in } O(N^{3/2}),\ \text{WCOJ}} \\
&= \sum_{b,c,d,e} M_1(d,e)T(b,c,d,e)M_2(b,c)
\end{aligned}$$

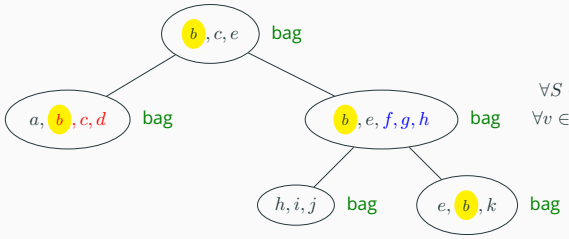
Different Variable Elimination Order ⇒ Different Tree Decomposition

$$Q = \sum_{a,b,c,d,e,f} R(a,b)S(a,c)T(b,c,d,e)W(e,f)V(d,f)$$



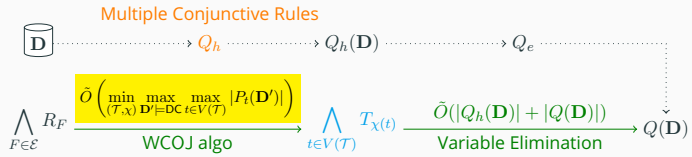
Detour: Tree Decompositions

$$S() \leftarrow R(a,b,d) \wedge c < d \wedge T(c,b,d) \wedge U(b,e) \wedge V(c,e)$$
$$\wedge\, b + e = f \wedge W(b,e,g) \wedge g/f = h \wedge X(i,j,h) \wedge e - b = k.$$



Hypergraph $\mathcal{H} = (V, \mathcal{E})$
Tree $\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}))$
$\forall S \in \mathcal{E}, \exists t \in V(\mathcal{T})$ s.t. $S \subseteq \chi(t)$
$\forall v \in V, \{t \mid v \in \chi(t)\}$ is a subtree

Dynamic Programming over a Single Tree Decomposition — Fix $(\mathcal{T}, \chi)$



$$P_t : T_{\chi(t)} \leftarrow \bigwedge_{F \in \mathcal{E}} R_F$$

Recall the polymatroid bound:

$$\log \min_{(\mathcal{T},\chi)} \max_{\mathbf{D}' \models \mathsf{DC}} \max_{t \in V(\mathcal{T})} |P_t(\mathbf{D}')| \leq \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} \max_{h \in \Gamma_n \cap \mathsf{DC}} h(\chi(t))$$

Realization of Idea 2 — Fractional Hypertree Width

$$\tilde{O}\left(|\boldsymbol{D}| + \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} f(Q, \boldsymbol{D}') + |Q(\boldsymbol{D})|\right)$$

Becomes:

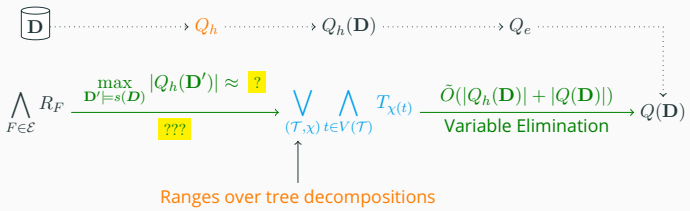
$$\tilde{O}\left(|\boldsymbol{D}| + 2^{\min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} \max_{h \in \Gamma_n \cap \mathsf{DC}} h(\chi(t))} + |Q(\boldsymbol{D})|\right)$$

Corrolaries when the input has only cardinality constraints:

- $\tilde{O}\left(|\boldsymbol{D}| + |Q(\boldsymbol{D})|\right)$ if Q is acyclic [Yannakakis 81]
- $\tilde{O}\left(|\boldsymbol{D}| + |\boldsymbol{D}|^{\mathsf{fhtw}(Q)} + |Q(\boldsymbol{D})|\right)$ fhtw = *fractional hypertree width*, [GM 06]

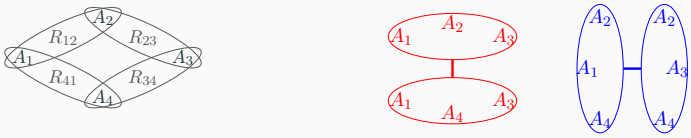
Outline

Multiple Tree Decompositions! — Idea from [Marx 13]



Assuming Boolean conjunctive query for now

Example: $Q : S() \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$



$$(T_{123} \wedge T_{134}) \vee (T_{124} \wedge T_{234}) \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$$

By distributivity, rewrite the head:

$$(T_{123} \vee T_{124}) \wedge (T_{123} \vee T_{234}) \wedge (T_{134} \vee T_{124}) \wedge (T_{134} \vee T_{234}) \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$$

(Each "clause" has one bag per TD)

Example: $Q : S() \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$

$$(T_{123} \vee T_{124}) \wedge (T_{123} \vee T_{234}) \wedge (T_{134} \vee T_{124}) \wedge (T_{134} \vee T_{234}) \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$$

Is the same as evaluating 4 disjunctive datalog rules:

$$T_{123} \vee T_{124} \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$$
$$T_{123} \vee T_{234} \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$$
$$T_{134} \vee T_{124} \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$$
$$T_{134} \vee T_{234} \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$$

Semantics of a Disjunctive Datalog Query

$$P : \qquad T_{123} \vee T_{124} \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}$$

The model contains two tables $T_{123}(A_1, A_2, A_3)$ and $T_{124}(A_1, A_2, A_4)$, such that:

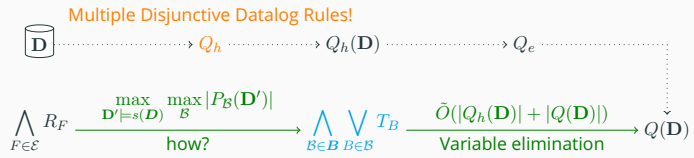
- if $R_{12}(a_1, a_2)$, $R_{23}(a_2, a_3)$, $R_{34}(a_3, a_4)$, and $R_{41}(a_4, a_1)$
- then $T_{123}(a_1, a_2, a_3)$ OR $T_{124}(a_1, a_2, a_4)$.

Output size

- Write $\boldsymbol{T} \models P$ if $\boldsymbol{T}$ is a model
- Define $|\boldsymbol{T}| = \max\{|T_{123}|, |T_{124}|\}$

$\boldsymbol{T} = (T_{123}, T_{124})$.

Multiple Tree Decompositions



$$P_{\mathcal{B}} : \bigvee_{B\in\mathcal{B}} T_B \leftarrow \bigwedge_{F\in\mathcal{E}} R_F \qquad |P_{\mathcal{B}}| \stackrel{\text{def}}{=} \min_{\boldsymbol{T}:\boldsymbol{T}\models P_{\mathcal{B}}} \max_{B\in\mathcal{B}} |T_B|$$

Each $\mathcal{B} \in \boldsymbol{B}$ is a collection of bags, one per TD.

New Problem: Polymatroid Bound for Disjunctive Datalog

$$P_{\mathcal{B}} : \bigvee_{B\in\mathcal{B}} T_B \leftarrow \bigwedge_{F\in\mathcal{E}} R_F \qquad \text{what is } \max_{\mathbf{D}'\models s(\boldsymbol{D})} |P_{\mathcal{B}}(\mathbf{D}')|?$$

Theorem (ANS 17)

Define $DC \stackrel{\text{def}}{=} \{h \mid \quad h(Y|X) \le \log N \qquad \forall (X,Y,N) \in s(\boldsymbol{D})\}$, *then*

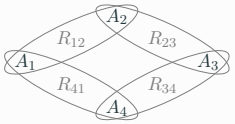
$$\log \sup_{\mathbf{D}'\models s(\boldsymbol{D})} |P_{\mathcal{B}}(\boldsymbol{D}')| \le \max_{h\in\overline{\Gamma}_n^*\cap DC} \min_{B\in\mathcal{B}} h(B) \qquad \textit{Entropic Bound}$$

$$\le \max_{h\in\Gamma_n\cap DC} \min_{B\in\mathcal{B}} h(B) \qquad \textit{Polymatroid Bound}$$

Generalize the entropy argument for conjunctive datalog (fun exercise!)

4-Cycle Example

$$P : \bigvee_{B\in\mathcal{B}} T_B \leftarrow \bigwedge_{F\in\mathcal{E}} R_F \qquad |P(\boldsymbol{D})| \stackrel{\text{def}}{=} \min_{\boldsymbol{T}:\boldsymbol{T}\models P} \max_{B\in\mathcal{B}} |T_B|$$



DC : $|R_{12}| \le N, \quad |R_{23}| \le N, \quad |R_{34}| \le N, \quad |R_{41}| \le N.$

$$P_{123,234} : T_{123} \vee T_{234} \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41} \qquad \mathcal{B} = \{123, 234\}$$

$$\sup_{\mathbf{D}'\models \mathsf{DC}} \log |P_{123,234}(\boldsymbol{D}')| \le \max_{h\in\Gamma_n\cap \mathsf{DC}} \min\{h(A_1A_2A_3), h(A_2A_3A_4)\}$$

$$\le \max_{h\in\Gamma_n\cap \mathsf{DC}} \frac{1}{2}[h(A_1A_2A_3) + h(A_2A_3A_4)]$$

$$\text{(Shannon-inequality)} \le \max_{h\in\Gamma_n\cap \mathsf{DC}} \frac{1}{2}[h(A_1A_2) + h(A_2A_3) + h(A_3A_4)]$$

$$\le \frac{3}{2}\log N.$$

Realization of Idea 2

$$\tilde{O}\left(|\boldsymbol{D}| + \sup_{\boldsymbol{D}'\models s(\boldsymbol{D})} f(Q, \boldsymbol{D}') + |Q(\boldsymbol{D})|\right)$$

Becomes:

$$\tilde{O}\left(|\boldsymbol{D}| + 2^{\max_{\mathcal{B}\in\boldsymbol{B}} \max_{h\in\Gamma_n\cap \mathsf{DC}} \min_{B\in\mathcal{B}} h(B)} + |Q(\boldsymbol{D})|\right)$$

Another problem: WCOJ for disjunctive datalog!

Another Problem : WCOJ for Disjunctive Datalog

$$P_{\mathcal{B}} \ : \ \bigvee_{B\in\mathcal{B}} T_B \leftarrow \bigwedge_{F\in\mathcal{E}} R_F$$ compute a model in worst-case optimal time

Theorem (ANS 17)

Given a collection $s(\boldsymbol{D})$ of degree constraints, and a disjunctive datalog program $P_{\mathcal{B}}$, PANDA computes a model of $P_{\mathcal{B}}$ in time

$$\left(|\boldsymbol{D}| + 2^{\max_{h\in\Gamma_n\cap DC}\min_{B\in\mathcal{B}} h(B)}\right)$$

Realization of Idea 2 — Submodular Width

$$\tilde{O}\left(|\boldsymbol{D}| + \sup_{\boldsymbol{D}'\models s(\boldsymbol{D})} f(Q,\boldsymbol{D}') + |Q(\boldsymbol{D})|\right)$$

Becomes:

$$\tilde{O}\left(|\boldsymbol{D}| + 2^{\max_{\mathcal{B}\in\boldsymbol{B}}\max_{h\in\Gamma_n\cap \mathsf{DC}}\min_{B\in\mathcal{B}} h(B)} + |Q(\boldsymbol{D})|\right)$$

Corrolaries when the input has only cardinality constraints:

- $\tilde{O}\left(|\boldsymbol{D}| + |\boldsymbol{D}|^{\mathsf{subw}(Q)} + |Q(\boldsymbol{D})|\right)$ subw $=$ *submodular-width*, [Marx 13]
- Detecting k-cycle can be done in $O(N^{2-\frac{1}{\lceil k/2\rceil}})$-time [AYZ 97]

Outline

References (and references thereof)

Yannakakis 81 Mihalis Yannakakis: Algorithms for Acyclic Database Schemes. VLDB 1981: 82-94

Pearl 82 Pearl, Judea. "Reverend Bayes on inference engines: A distributed hierarchical approach"). AAAI-82

CGFS 1986 Fan R. K. Chung, Ronald L. Graham, Peter Frankl, James B. Shearer: Some intersection theorems for ordered sets and graphs. J. Comb. Theory, Ser. A 43(1): 23-37 (1986)

ZP 1994 Zhang, N.L., Poole, D.:A Simple Approach to Bayesian Network Computations.In: 7th Canadian Conference on Artificial Intelligence,pp. 171–178

.

AYZ 97 Noga Alon, Raphael Yuster, Uri Zwick: Finding and Counting Given Length Cycles. Algorithmica 17(3): 209-223 (1997)

GM 06 Martin Grohe, Dániel Marx: Constraint solving via fractional edge covers. SODA 2006: 289-298

AGM 08 Albert Atserias, Martin Grohe, Dániel Marx: Size Bounds and Query Plans for Relational Joins. FOCS 2008: 739-748

GLVV 12 Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, Paul Valiant: Size and Treewidth Bounds for Conjunctive Queries. J. ACM 59(3): 16:1-16:35 (2012)

NPRR 12 Hung Q. Ngo, Ely Porat, Christopher Ré, Atri Rudra: Worst-case optimal join algorithms. PODS 2012.

GJ 13 Hung Q. Ngo, Christopher Ré, Atri Rudra: Skew Strikes Back: New Developments in the Theory of Join Algorithms. SIGMOD Records, 2013.

Marx 13 D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. J. ACM, 60(6):Art. 42, 51, 2013.

LFTJ 14 Todd L. Veldhuizen: Triejoin: A Simple, Worst-Case Optimal Join Algorithm. ICDT 2014: 96-106

References (and references thereof)

ANR 16 Mahmoud Abo Khamis, Hung Q. Ngo, Atri Rudra: FAQ: Questions Asked Frequently. PODS 2016: 13-28

ANS 17 Mahmoud Abo Khamis, Hung Q. Ngo, Dan Suciu: What Do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog Have to Do with One Another? PODS 2017: 429-444

ANOS 19 Mahmoud Abo Khamis, Hung Q. Ngo, Dan Olteanu, Dan Suciu: Boolean Tensor Decomposition for Conjunctive Queries with Negation. ICDT 2019: 21:1-21:19

N 18 Hung Q. Ngo: Worst-case Optimal Join Algorithms: Techniques, Results, and Open Problems. PODS 2018.

N 22 Hung Q. Ngo: On an Information Theoretic Approach to Cardinality Estimation (Invited Talk). ICDT 2022: 1:1-1:21

IMNP 25 Sungjin Im, Ben Moseley, Hung Q. Ngo, Kirk Pruhs, and Alireza Samadian: Efficient Algorithms for Cardinality Estimation and Conjunctive Query Evaluation With Simple Degree Constraints. PODS 2025.

Thank You!

Outline

Appendix: PANDA Algorithm

Outline

Appendix: PANDA Algorithm

- Shannon-Flow Inequalities
- IAAT Algorithm
- PANDA for Disjunctive Datalog

The Polymatroid Bound and Its Dual $\max\{h(V) \mid h \in \Gamma_n \cap \mathsf{DC}\}$

More explicitly,

$$
\begin{array}{lrll}
\max & h(V) & & \text{dual vars} \\
\text{s.t.} & h(Y) - h(X) \le \log N, & (X,Y,N) \in \mathsf{DC} & \delta_{Y|X} \\
& h(I \cup J | J) - h(I | I \cap J) \le 0, & I \perp J & \sigma_{I,J} \\
& h(X) - h(Y) \le 0, & \emptyset \neq X \subset Y \subseteq V & \mu_{Y|X} \\
& h(Z) \ge 0, & \emptyset \neq Z \subseteq V. &
\end{array}
$$

$I \perp J$ means $I \not\subseteq J$ and $J \not\subseteq I$.

The Polymatroid Bound and Its Dual

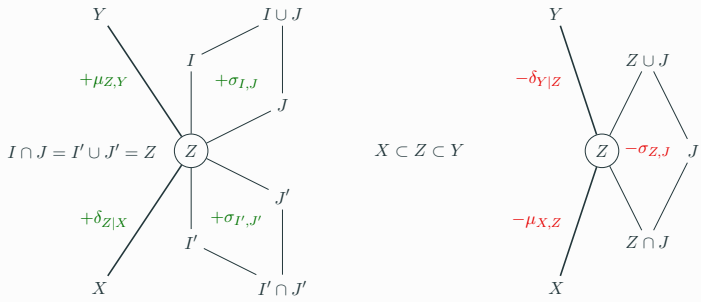
$\max\{h(V) \mid h \in \Gamma_n \cap \mathsf{DC}\}$

$$\begin{aligned} \min \quad & \sum_{(X,Y,N)\in \mathsf{DC}} \log N \cdot \delta_{Y|X} \\ \text{s.t.} \quad & \mathsf{excess}(V) \geq 1, \\ & \mathsf{excess}(Z) \geq 0, \qquad \emptyset \neq Z \subseteq V. \\ & (\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu}) \geq \mathbf{0}. \end{aligned}$$

where, for any $\emptyset \neq Z \in 2^V$, the quantity $\mathsf{excess}(Z)$ is defined by

$$\begin{aligned} \mathsf{excess}(Z) := & \sum_{X:(X,Z)\in \mathsf{DC}} \delta_{Z|X} - \sum_{Y:(Z,Y)\in \mathsf{DC}} \delta_{Y|Z} + \sum_{\substack{I\perp J \\ I\cap J = Z}} \sigma_{I,J} \\ & + \sum_{\substack{I\perp J \\ I\cup J = Z}} \sigma_{I,J} - \sum_{J: J\perp Z} \sigma_{Z,J} - \sum_{X: X\subset Z} \mu_{X,Z} + \sum_{Y: Z\subset Y} \mu_{Z,Y}. \end{aligned}$$

Contributions of coefficients to $\mathsf{excess}(Z)$



Shannon-Flow Inequalities

[ANS 17]

Definition

Given $\boldsymbol{\delta} \geq 0$, the following is a Shannon-flow inequality if it holds for all $h \in \Gamma_n$:

$$h(V) \leq \sum_{(X,Y,N)\in \mathsf{DC}} \delta_{Y|X} \cdot (h(Y) - h(X))$$

- $\boldsymbol{\delta}$ defines a Shannon-flow inequality iff $\exists(\boldsymbol{\sigma}, \boldsymbol{\mu})$ s.t. $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$ is dual-feasible.
- If DC contains only cardinality constraints $(X = \emptyset, Y, N)$, then $\boldsymbol{\delta}$ defines a Shannon-flow inequality iff it is a fractional edge cover of the query hypergraph. Shearer's Lemma!

Example: Shannon-Flow Inequality for Triangle Query

$$h(A,B,C) \le \frac{1}{2}\left(h(A,B) + h(B,C) + h(A,C)\right)$$

A step-by-step proof: [Radhakrishnan 2003]

$$\begin{aligned}
& h(A,B) + h(A,C) + h(B,C) \\
(\text{decomposition}) &= h(A) + h(B|A) + h(B,C) + h(A,C) \\
(\text{sub-modularity}) &\ge (h(A|B,C) + h(B,C)) + (h(B|A) + h(A,C)) \\
(\text{composition}) &= h(A,B,C) + (h(B|A) + h(A,C)) \\
(\text{sub-modularity}) &\ge h(A,B,C) + (h(B|A,C) + h(A,C)) \\
(\text{composition}) &= h(A,B,C) + h(A,B,C)
\end{aligned}$$

Example: Another Shannon-Flow Inequality

$$h(ABCD) \le \frac{1}{2}[h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD)],$$

$$\begin{aligned}
& h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD) \\
(\text{decomposition}) &= h(AB) + h(B) + h(C|B) + h(CD) + h(D|AC) + h(A|BD) \\
(\text{sub-modularity}) &\ge h(AB) + h(B) + h(C|B) + h(CD|B) + h(D|AC) + h(A|BD) \\
(\text{composition}) &= h(AB) + h(C|B) + h(BCD) + h(D|AC) + h(A|BD) \\
(\text{sub-modularity}) &\ge h(AB) + h(C|B) + h(BCD) + h(D|AC) + h(A|BCD) \\
(\text{composition}) &= h(AB) + h(C|B) + h(D|AC) + h(ABCD) \\
(\text{sub-modularity}) &\ge h(AB) + h(C|AB) + h(D|AC) + h(ABCD) \\
(\text{composition}) &= h(ABC) + h(D|AC) + h(ABCD) \\
(\text{sub-modularity}) &\ge h(ABC) + h(D|ABC) + h(ABCD) \\
(\text{composition}) &= h(ABCD) + h(ABCD).
\end{aligned}$$

Shannon-flow Inequalities – Summary [ANS 17]

From LP-duality, there exists $\boldsymbol{\delta} \ge \mathbf{0}$ s.t.

$$\text{polymatroid-bound} := \max\{h(V) \mid h \in C \cap \Gamma_n\} = \sum_{(X,Y,N)\in \mathsf{DC}} \delta_{Y|X} \log N,$$

and for these $\boldsymbol{\delta}$, from Farkas's lemma we have

$$h(V) \le \sum_{(X,Y,N)\in \mathsf{DC}} \delta_{Y|X} \cdot h(Y|X), \quad \forall h \in \Gamma_n$$

Proof Sequence for a Shannon-Flow Inequality

$$h(V) \leq \sum_{(X,Y,N)\in \mathsf{DC}} \delta_{Y|X} \cdot h(Y|X)$$

A proof sequence is a conversion from RHS to LHS using a sequence of steps of the form

(In)equality	Steps ($X \subseteq Y$)
$h(X) + h(Y\|X) = h(Y)$	$h(X) + h(Y\|X) \to h(Y)$
$h(Y) = h(X) + h(Y\|X)$	$h(Y) \to h(X) + h(Y\|X)$
$h(Y) \geq h(X)$	$h(Y) \to h(X)$
$h(Y\|X) \geq h(Y \cup Z\|X \cup Z)$	$h(Y\|X) \to h(Y \cup Z\|X \cup Z)$

Existence of Proof Sequence

Lemma (ANS 2017)

There is a proof sequence for every Shannon-flow inequality. (The length is at most doubly exponential in $|V|$).

The Shannon-flow inequality is a linear combination of dual constraints; the proof sequence is more stringent than that.

Outline

One Inequality At A Time (IAAT)

There is an algorithm (called PANDA) that converts a proof sequence $\rightarrow$ an efficient algorithm to answer the original query

Steps ($X \subseteq Y$)	Relational Operator
$h(X) + h(Y\|X) \rightarrow h(Y)$	(join)
$h(Y) \rightarrow h(X) + h(Y\|X)$	(data partition)
$h(Y) \rightarrow h(X)$	(projection)
$h(Y\|X) \rightarrow h(Y \cup Z\|X \cup Z)$	(NOP)

Theorem

PANDA solves any conjunctive query Q in time $\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\text{polymatroid bound}})$

Example: PANDA for Triangle Query

$$Q(A,B,C) \leftarrow R(A,B), S(B,C), T(A,C)$$
$$R^{\text{heavy}}(A,B) = \{(a,b) \ : \ |\sigma_{A=a}R| > \sqrt{N}\}$$
$$R^{\text{light}}(A,B) = \{(a,b) \ : \ |\sigma_{A=a}R| \leq \sqrt{N}\}$$

Algorithm is in the pudding!

$$\begin{aligned}
h(A,B) + h(A,C) + h(B,C) \quad & R(A,B), S(B,C), T(A,C) \\
= h(A) + h(B|A) + h(B,C) + h(A,C) \quad & R^{\text{heavy}}(A,B), R^{\text{light}}(A,B), S(B,C), T(A,C) \\
\geq (h(A|B,C) + h(B,C)) + (h(B|A) + h(A,C)) \quad & R^{\text{heavy}}(A,B), R^{\text{light}}(A,B), S(B,C), T(A,C) \\
= h(A,B,C) + (h(B|A) + h(A,C)) \quad & I^{\text{heavy}}(A,B,C), R^{\text{light}}(A,B), T(A,C) \\
\geq h(A,B,C) + (h(B|A,C) + h(A,C)) \quad & I^{\text{heavy}}(A,B,C), R^{\text{light}}(A,B), T(A,C) \\
= h(A,B,C) + h(A,B,C) \quad & I^{\text{heavy}}(A,B,C), I^{\text{light}}(A,B,C).
\end{aligned}$$

Example: PANDA for Triangle Query

The real query plan:

$$\begin{aligned}
& R(A,B) \wedge S(B,C) \wedge T(A,C) \\
& = (R^{\text{heavy}}(A,B) \vee R^{\text{light}}(A,B)) \wedge S(B,C) \wedge T(A,C) \\
& = (R^{\text{heavy}}(A,B) \wedge S(B,C) \wedge T(A,C)) \vee (R^{\text{light}}(A,B) \wedge S(B,C) \wedge T(A,C)) \\
& = (R^{\text{heavy}}(A,B) \wedge S(B,C) \wedge T(A,C)) \vee (R^{\text{light}}(A,B) \wedge S(B,C) \wedge T(A,C)) \\
& = I^{\text{heavy}}(A,B,C) \wedge T(A,C) \vee I^{\text{light}}(A,B,C) \wedge S(B,C).
\end{aligned}$$

- Note that $|I^{\text{heavy}}(A,B,C)| \leq N^{3/2}$ and $|I^{\text{light}}(A,B,C)| \leq N^{3/2}$.
- Overall runtime is $\tilde{O}(N^{3/2})$.

Example: PANDA for a More Interesting Example [ANS 17]

$$Q(A,B,C,D) \leftarrow R(A,B) \wedge S(B,C) \wedge T(C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A$$

From the Shannon-flow inequality:

$$h(ABCD) \leq \frac{1}{2}[h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD)],$$

we know

$$\log_2 |Q| \leq \frac{1}{2}[\log_2 |R| + \log_2 |S| + \log_2 |T| + 0 + 0]$$

or

$$|Q| \leq \sqrt{|R||S||T|}$$

Example: Another Shannon-Flow Inequality

$$h(ABCD) \leq \frac{1}{2}[h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD)],$$

$$\begin{aligned}
& h(AB) + h(BC) + h(CD) + h(D|AC) + h(A|BD) \\
\text{(decomposition)} &= h(AB) + h(B) + h(C|B) + h(CD) + h(D|AC) + h(A|BD) \\
\text{(sub-modularity)} &\geq h(AB) + h(B) + h(C|B) + h(CD|B) + h(D|AC) + h(A|BD) \\
\text{(composition)} &= h(AB) + h(C|B) + h(BCD) + h(D|AC) + h(A|BD) \\
\text{(sub-modularity)} &\geq h(AB) + h(C|B) + h(BCD) + h(D|AC) + h(A|BCD) \\
\text{(composition)} &= h(AB) + h(C|B) + h(D|AC) + h(ABCD) \\
\text{(sub-modularity)} &\geq h(AB) + h(C|AB) + h(D|AC) + h(ABCD) \\
\text{(composition)} &= h(ABC) + h(D|AC) + h(ABCD) \\
\text{(sub-modularity)} &\geq h(ABC) + h(D|ABC) + h(ABCD) \\
\text{(composition)} &= h(ABCD) + h(ABCD).
\end{aligned}$$

Example : PANDA for a More Interesting Example

$$\begin{aligned}
& R(A,B) \wedge S(B,C) \wedge T(C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&= R(A,B) \wedge S^{\mathsf{heavy}}(B,C) \wedge T(C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&\vee R(A,B) \wedge S^{\mathsf{light}}(B,C) \wedge T(C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&= R(A,B) \wedge S^{\mathsf{heavy}}(B,C) \wedge T(C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&\vee R(A,B) \wedge S^{\mathsf{light}}(B,C) \wedge T(C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&= R(A,B) \wedge I^{\mathsf{heavy}}(B,C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&\vee I^{\mathsf{light}}(A,B,C) \wedge T(C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&= R(A,B) \wedge I^{\mathsf{heavy}}(B,C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&\vee I^{\mathsf{light}}(A,B,C) \wedge T(C,D) \wedge \mathsf{hash}(A,C) = D \wedge \mathsf{hash}(B,D) = A \\
&= R(A,B) \wedge J^{\mathsf{heavy}}(A,B,C,D) \wedge \mathsf{hash}(A,C) = D \\
&\vee J^{\mathsf{light}}(A,B,C,D) \wedge T(C,D) \wedge \mathsf{hash}(B,D) = A.
\end{aligned}$$

Example : PANDA for a More Interesting Example

Main question

How to define S^{heavy} and S^{light} so that runtime is $\tilde{O}(2^{h^*(A,B,C,D)})$

$$S^{\text{heavy}}(B,C) = \{(b,c) \ : \ |\sigma_{C=c}S| > 2^{h^*(B,C)-h^*(C)}\}$$
$$S^{\text{light}}(B,C) = \{(b,c) \ : \ |\sigma_{C=c}S| \leq 2^{h^*(B,C)-h^*(C)}\}$$

Assuming h^* and $(\boldsymbol{\delta}^*, \boldsymbol{\sigma}^*, \boldsymbol{\mu}^*)$ are primal-dual optimal: $(\delta^*_{CD|\emptyset} > 0$ and $\delta^*_{AB|\emptyset} > 0)$

$$|S^{\text{light}}(B,C) \wedge T(C,D)| \leq 2^{h^*(B,C)-h^*(C)} \cdot 2^{h^*(C,D)} = 2^{h^*(B,C,D)} \leq 2^{h^*(A,B,C,D)}$$
$$|S^{\text{heavy}}(B,C) \wedge R(A,B)| \leq 2^{h^*(C)} \cdot 2^{h^*(A,B)} = 2^{h^*(A,B,C)} \leq 2^{h^*(A,B,C,D)}.$$

$=$ holds because SFI holds with $=$ for h^*.

The Actual PANDA Algorithm (Needs Another Talk)

More complicated because:

- Couldn't prove that every heavy / light copy reaches $h(V)$ eventually.
- Couldn't prove that in the proof sequence we won't ever compose terms which were decomposed in an earlier step

Main ideas to push through:

- A decomposition $h(A,B) \to h(A) + h(B|A)$ corresponds to partitioning R into logarithmically many "uniform" parts.
- Essentially, each each part, both the heavy condition and the light condition are satisfied.
- Induct on logarithmically many subproblems, including constructing a new proof sequence for each of them

The Actual PANDA Algorithm

PANDA runs in Time

$$\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\text{polymatroid bound for } Q}) = \tilde{O}(N + \text{poly}(\log N) \cdot \sup_{\boldsymbol{D}' \models s(\boldsymbol{D})} |Q(\boldsymbol{D}')|)$$

Outline

Appendix: PANDA Algorithm

- Shannon-Flow Inequalities
- IAAT Algorithm
- PANDA for Disjunctive Datalog

Evaluating a Disjunctive Datalog Program within $\max_{h\in\Gamma_n\cap\mathsf{HDC}} \min_{B\in\mathcal{B}} h(B)$

There exists non-negative $\boldsymbol{\lambda} = (\lambda_B)_{B\in\mathcal{B}}$, with $\|\boldsymbol{\lambda}\|_1 = 1$, s.t.

$$\max_{h\in\Gamma_n\cap\mathsf{HDC}} \min_{B\in\mathcal{B}} h(B) = \max_{h\in\Gamma_n\cap\mathsf{HDC}} \sum_{B\in\mathcal{B}} \lambda_B\, h(B)$$

Shannon-flow inequality: There exists $\boldsymbol{\delta} \geq \mathbf{0}$ s.t. (Farkas's lemma)

$$2^{\max_{h\in\Gamma_n\cap\mathsf{HDC}} \min_{B\in\mathcal{B}} h(B)} = \prod_{(X,Y,N)\in s(\boldsymbol{D})} N^{\delta_{Y|X}}$$

$$\sum_{B\in\mathcal{B}} \lambda_B \cdot h(B) \leq \sum_{(X,Y,N)\in s(\boldsymbol{D})} \delta_{Y|X} \cdot h(Y|X), \ \forall h \in \Gamma_n$$

Proof sequence

Shannon-flow inequality: $h(Y|X) \stackrel{\text{def}}{=} h(Y) - h(X),\ X \subseteq Y$

$$\sum_{B\in\mathcal{B}} \lambda_B \cdot h(B) \leq \sum_{(X,Y,N)} \delta_{Y|X} \cdot h(Y|X)$$

Proof sequence, convert RHS to LHS using following steps

(In)equality	Steps ($X \subseteq Y$)
$h(X) + h(Y\|X) = h(Y)$	$h(X) + h(Y\|X) \to h(Y)$
$h(Y) = h(X) + h(Y\|X)$	$h(Y) \to h(X) + h(Y\|X)$
$h(Y) \geq h(X)$	$h(Y) \to h(X)$
$h(Y\|X) \geq h(Y \cup Z\|X \cup Z)$	$h(Y\|X) \to h(Y \cup Z\|X \cup Z)$

Theorem

There is a proof sequence for every Shannon-flow inequality.

Example: $P:\ T_{123} \vee T_{234} \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}.$

$$|R_{12}|, |R_{23}|, |R_{34}|, |R_{41}| \le N \quad \Rightarrow \quad |P| \le N^{3/2}$$

$$\begin{aligned}
\log |P| &\le \min(h(A_1A_2A_3), h(A_2A_3A_4)) && \text{(polymatroid bound)}\\
&\le \frac{1}{2}\big(h(A_1A_2A_3) + h(A_2A_3A_4)\big) && \text{(linearize)}\\
&\le \frac{1}{2}\big(h(A_1A_2) + h(A_2A_3) + h(A_3A_4)\big) && \text{(Shannon-flow)}\\
&\le \frac{3}{2}\log N && \text{(Cardinality constraints)}
\end{aligned}$$

Proof sequence	Proof Step
$h(A_1A_2) + h(A_2A_3) + h(A_3A_4)$	$\big(h(A_3A_4) \to h(A_4\|A_3) + h(A_3)\big)$
$h(A_1A_2) + h(A_2A_3) + h(A_4\|A_3) + h(A_3)$	$\big(h(A_4\|A_3) \to h(A_4\|A_2A_3)\big)$
$h(A_1A_2) + h(A_2A_3) + h(A_4\|A_2A_3) + h(A_3)$	$\big(h(A_2A_3) + h(A_4\|A_2A_3) \to h(A_2A_3A_4)\big)$
$h(A_1A_2) + h(A_2A_3A_4) + h(A_3)$	$\big(h(A_1A_2) \to h(A_1A_2\|A_3)\big)$
$h(A_1A_2\|A_3) + h(A_2A_3A_4) + h(A_3)$	$\big(h(A_1A_2\|A_3) + h(A_3) \to h(A_1A_2A_3)\big)$
$h(A_1A_2A_3) + h(A_2A_3A_4)$	

Example: $P:\ T_{123} \vee T_{234} \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}.$

$$|R_{12}|, |R_{23}|, |R_{34}|, |R_{41}| \le N \quad \Rightarrow \quad |P| \le N^{3/2}$$

$h(A_3A_4) \to h(A_4\|A_3) + h(A_3)$	$R_{34}(A_3, A_4) \to R^{(\ell)}_{34}(A_3, A_4), R^{(h)}_{3}(A_3)$
$h(A_4\|A_3) \to h(A_4\|A_2A_3)$	$R^{(\ell)}_{34}(A_3, A_4) \to R^{(\ell)}_{34}(A_3, A_4)$
$h(A_2A_3) + h(A_4\|A_2A_3) \to h(A_2A_3A_4)$	$R_{23}(A_2, A_3) \bowtie R^{(\ell)}_{34}(A_3, A_4) \to T_{234}(A_2, A_3, A_4)$
$h(A_1A_2) \to h(A_1A_2\|A_3)$	$R_{12}(A_1, A_2) \to R_{12}(A_1, A_2)$
$h(A_1A_2\|A_3) + h(A_3) \to h(A_1A_2A_3)$	$R_{12}(A_1, A_2) \bowtie R^{(h)}_{3}(A_3) \to T_{123}(A_1, A_2, A_3)$

Conjunctive Queries with Equations and Disequations for Databases over Semirings

Guillermo Badia[(a)], Carles Noguera[(b)], Gaia Petreni[(c)], Val Tannen[(d)]

(a): University of Queensland
(b), (c): DIISM, University of Siena
(d): University of Pennsylvania

September 2025

Goal of the talk

- Databases with annotated relations: tuple annotations used to track *provenance*, providing information on how the query results depend on atomic facts.
- *Containment problem* for conjunctive queries *with equations and disequations*: are all the answers to query P also answers to query Q?
- **Are these problems decidable?** For a positive answer: find equivalence with the existence of specific types of mappings between queries (Chandra-Merlin strategy).
- Complexity results for the containment problem.
- Containment for regular CQs over semiring-annotated databases is well-understood since Green (2011).

Take-home message: results of this talk

Klug (1988) and Van der Meyden (1997) show that containment for $\{=\neq\}-$CQs on standard databases is Π_2^p-c.

Cohen, Nutt & Sagiv (2007) give a characterization in terms of mappings between **families of queries**.

Type	Complexity	Known semirings
$\{=\neq\}-$Can. map. (identifications)	Π_2^p-c	$\mathbb{B}$ (**Klug, VdM**), Distr. Latt. (e.g. PosBool[X])
$\{=\neq\}-$Hom. coverage for rel. atoms (identifications)	Π_2^p-c	Lin[X]
$\{=\neq\}-$Injective for rel. atoms (identifications)	Π_2^p-c	Sorp[X]
$\{=\neq\}-$Surjective for rel. atoms (identifications)	Π_2^p-c	Why[X], Trio[X]
$\{=\neq\}-$Bijective for rel. atoms (identifications)	in Π_2^p and NP-hard	$\mathbb{N}[X]$, $\mathbb{B}[X]$
n/a	Undecidable	$\mathbb{N}$ (**Kolaitis et al.**)

Semirings

Definition

A **semiring** is an algebra $\mathbf{K} = (K, +, \cdot, 0, 1)$ where:

- $(K, +, 0)$ is a commutative monoid.
- $(K, \cdot, 1)$ is a monoid.
- $\cdot$ distributes over $+$:

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

$$(y + z) \cdot x = (y \cdot x) + (z \cdot x)$$

- 0 is absorbing: $x \cdot 0 = 0 \cdot x = 0$.

$\mathbf{K}$ is **commutative** if $\cdot$ is commutative.
Easy examples: $\mathbb{N}$, $\mathbb{B}$ (two-element Boolean algebra).

Semirings for Provenance

Definition (Green et al. 2007)

The **provenance polynomials semiring** for X (a countable set of variables) is the semiring of polynomials with variables from X and coefficients from $\mathbb{N}$, with the operations defined as usual: $(\mathbb{N}[X], +, \cdot, 0, 1)$.

Definition (Green,2009)

The **Boolean provenance polynomials** semiring for X is the semiring of polynomials over variables X with Boolean coefficients: $(\mathbb{B}[X], +, \cdot, 0, 1)$.

Semirings for Provenance (cont'd)

Let $f : \mathbb{N}[X] \to \mathbb{N}[X]$ be the mapping that "drops exponents", e.g.,

$$f(2x^2y + 3xy + 2z^3 + 1) = 5xy + 2z + 1.$$

Denote by $\approx_f$ the congruence relation on $\mathbb{N}[X]$ defined by

$$a \approx_f b \iff f(a) = f(b).$$

Definition (Benjelloun et al., 2008)

The **Trio semiring** for X, Trio(X), is the quotient semiring of $\mathbb{N}[X]$ by $\approx_f$.

The why-provenance of a tuple is the set of sets of "contributing" source tuples and it can be captured using the following semiring.

Definition (Buneman et al., 2008)

The **why-provenance** semiring for X is $(\text{Why}(X), \cup, \Cup, \emptyset, \{\emptyset\})$ where $\text{Why}(X) = \mathcal{P}_{\text{fin}}(\mathcal{P}_{\text{fin}}(X))$ and $\Cup$ denotes pairwise union:

$$A \Cup B = \{a \cup b : a \in A, b \in B\}$$

Definition

The **lineage semiring** for X is $(\mathcal{P}_{\text{fin}}(X) \cup \{\bot\}, +, \cdot, \bot, \emptyset)$ where

- X is a set of variables,
- $\bot + S = S + \bot = S$,
- $\bot \cdot S = S \cdot \bot = \bot$,
- $S + T = S \cdot T = S \cup T$ if $S, T \neq \bot$.

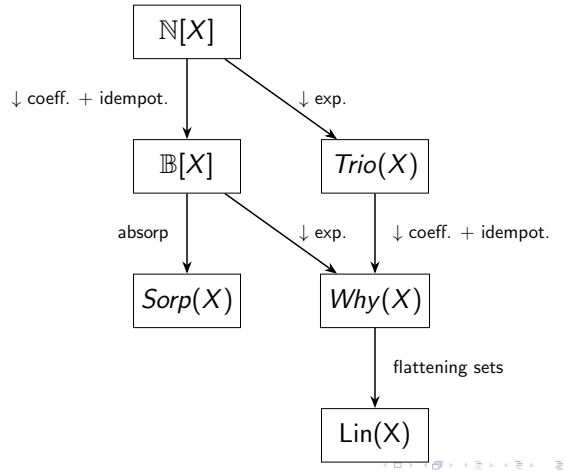
A commutative semiring $\mathbf{K} = (K, +, \cdot, 0, 1)$ is **absorptive** if for every $a, b \in K$

$$a + ab = a.$$

Denote by $\approx$ the smallest congruence on $\mathbb{N}[X]$ that identifies polynomials according to absorption.

Definition

The **absorptive** semiring for X, $\text{Sorp}(X)$, is the quotient semiring of $\mathbb{N}[X]$ by $\approx$.



Fix a countable domain $\mathbb{D}$ of individuals and a semiring $\mathbf{K} = (K, +, \cdot, 0, 1)$.

Definition

An n-ary K-**relation** is a function $R : \mathbb{D}^n \to K$ such that its support, defined by

$$supp(R) = \{t : t \in \mathbb{D}^n, R(t) \neq 0\}$$

is finite.

A $\mathbb{B}$-relation:

Name	City	
James Bond	*Brisbane*	1
James Bond	*Tokyo*	0
Ethan Hunt	*Fukuoka*	1

Set semantics:
2 tuples

A $\mathbb{N}$-relation:

Name	City	
James Bond	*Brisbane*	5
James Bond	*Tokyo*	0
Ethan Hunt	*Fukuoka*	3

Bag semantics:
8 tuples

Definition

If R is an n-ary K-relation and t is an n-tuple, we call the value $R(t) \in K$ the **annotation** of t in R.

Definition

A K-**instance** is a mapping from predicate symbols to K-relations.
If $\mathfrak{A}$ is a K-instance and S is a predicate symbol, we denote by $S^{\mathfrak{A}}$ the value of S in $\mathfrak{A}$.

Example

Where $\mathbb{N}$ is the semiring of natural numbers:

$$R^{\mathfrak{A}} \stackrel{\text{def}}{=} \begin{array}{|ccc|} \hline a & b & 2 \\ d & b & 1 \\ b & c & 1 \\ \hline \end{array} \qquad S^{\mathfrak{A}} \stackrel{\text{def}}{=} \begin{array}{|ccc|} \hline b & g & f & 3 \\ d & a & b & 1 \\ \hline \end{array}$$

Definition

A **conjunctive query** (CQ) is an expression of the form

$$Q(\bar{u}) : -R_1(\bar{u}_1), \ldots, R_n(\bar{u}_n)$$

where

- $Q(\bar{u})$ is the **head** of the query (head(Q)),
- the multiset (bag) of **atoms** $R_1(\bar{u}_1), \ldots, R_n(\bar{u}_n)$ is the **body** of the query (body(Q)),
- $\bar{u}$ is the tuple of distinguished variables and constants,
- $\bar{u}_1, \ldots, \bar{u}_n$ are tuples of variables and constants whose arities are consistent with their associated predicate symbols; each variable appearing in the head also appears somewhere in the body.

Think of CQs as existential formulas where only conjunctions are allowed!

Definition

A **valuation** is a function $\nu : \text{vars}(Q) \to \mathbb{D}$.

Valuations operate component-wise on tuples in the expected way.

Let Q be a CQ

$$Q(\bar{u}) : -R_1(\bar{u}_1), \ldots, R_n(\bar{u}_n)$$

and let $\mathfrak{A}$ be a K-instance of the same schema.

The **result of evaluating** Q **on** $\mathfrak{A}$ is the K-relation defined

$$[\![Q]\!]^{\mathfrak{A}}(t) \stackrel{\text{def}}{=} \sum_{\nu \text{ s.t.} \nu(\bar{u})=t} \prod_{i=1}^{n} R_i^{\mathfrak{A}}(\nu(\bar{u}_i))$$

and the sums and products are in K.

The Natural Order

Let $(K, +, \cdot, 0, 1)$ be a semiring and define

$$a \leq b \iff \exists c : a + c = b.$$

When $\leq$ is a partial order we say that K is **naturally-ordered**.

Example

For $\mathbb{B}[X]$ we have $a \leq b$ iff every monomial in a also appears in b.
For $\mathbb{N}[X]$ we have $a \leq b$ iff every monomial in a also appears in b with an equal or greater coefficient. Thus, $2x^2y \leq 5x^2y + 2z$, but $x + 2y \not\leq 5x + 3y^2$.
For lineage and why-provenance the natural order corresponds to set inclusion.

Definition

Let K be a naturally-ordered semiring and let R_1, R_2 be two K-relations. R_1 is **contained** in R_2 ($R_1 \leq_K R_2$) iff

$$\forall t \in \mathbb{D}^n,\ R_1(t) \leq R_2(t)$$

Definition

Consider two queries P, Q.
P is **contained** in Q ($P \sqsubseteq_K Q$) iff

$$\forall K\text{-instance } \mathfrak{A},\ [\![P]\!]^{\mathfrak{A}} \leq_K [\![Q]\!]^{\mathfrak{A}}$$

CQ with equations and disequations

Definition

A $\{=, \neq\}$-**CQ** is simply a CQ where literals of the form $x = y$ and $x \neq y$ are allowed in the body of the query. (**In evaluating the query in a semiring these literals take only values** 0 **or** 1 **in the usual manner.**)

We focus on queries that are:

- **safe**: the only variables allowed are those in the active domain of the query.
- **consistent**: $x \neq x$ does not follow logically from the body of the query for any variable x.

Completions and identifications

Definition

Given a $\{=, \neq\}$-CQ Q, a **completion** Q' of Q comes from adding either $x = y$ or $x \neq y$ for every couple of variables x, y that appear in a relational atom of Q, as long as the new query is consistent.

Consider the equivalence relation between variables of a completion Q' given by

$x \equiv y$ iff $x = y$ is a logical consequence of the body of Q'.

A **canonical substitution** maps all elements in an equivalence class to a representative.

Definition (Cohen et al., 2007)

An **identification** Q'^{id} of a completion Q' comes by eliminating all equations by applying a canonical substitution to Q'.

Consider the queries

$$q := \exists x, y(R(x,y) \wedge R(y,x))$$

and

$$p := \exists x, y(R(x,y) \wedge x = y).$$

Observe that q has the following two possible identifications:

1. $\exists x, y(R(x,y) \wedge R(y,x) \wedge x \neq y)$
2. $\exists x(R(x,x) \wedge R(x,x))$,

where $\exists x(R(x,x) \wedge R(x,x))$ comes from the completion $\exists x, y(R(x,y) \wedge R(y,x) \wedge x = y)$ and the canonical substitution that sends y to x.
Similarly, p (which is already a completion of itself) has only the following identification:

1. $\exists x R(x,x)$.

Containment mappings

Definition (Cohen et al., 2007)

Given two $\{=,\neq\}$-CQs, q_1 and q_2, a $\{=,\neq\}$-**containment mapping** h from q_1 to q_2 is a function from the variables of q_1 to q_2 that preserves literals in the following sense:

- if the atom $l(\bar{x})$ appears in q_1, the atom $l(h(\bar{x}))$ appears in q_2,
- if the equation (disequation) $l(\bar{x})$ appears in q_1, the atom $l(h(\bar{x}))$ is a **logical consequence** of the body of q_2.

A containment mapping is **one-to-one** or **injective** for relational atoms if the multiset of images of atoms of Q_1 is bag-contained in the multiset of relational atoms of Q_2.

Also, h is **surjective** for relational atoms if the multiset of relational atoms of Q_2 is equal to the multiset of images of atoms of Q_1. (Surjective on relational atoms gives also surjective as a mapping on variables.)

Exact for relational atoms means being both surjective and injective.

Canonical database

Given an identification q^{id} of some $\{=, \neq\}$-CQ q, one can build its **canonical database** $D^{q^{id}}$ as follows.

- For any relation R of the schema of q^{id} we let $R_{D^{q^{id}}}$ contain the tuple of $(x_1, \ldots, x_n)$ iff $R(x_1, \ldots, x_n)$ is an atom in q^{id}.
- By construction, if the identification $\{=, \neq\}$-CQ q^{id} is a formula $\phi(u)$, then u is an answer to the query ϕ in the database $D^{q^{id}}$.

The Boolean case

Theorem (Klug (1988), Kolaitis et al. (1998), Cohen et al. (2007))

For each $\{=, \neq\}$-CQs Q_1, Q_2 with the same tuple of free variables $\overline{u}$, the following are equivalent:

1. $Q_1 \sqsubseteq_{\mathbb{B}} Q_2$.
2. *For every identification Q_1^{id} of Q_1, there is a $\{=, \neq\}$-containment mapping $h_{Q_1^{id}} : Q_2 \longrightarrow Q_1^{id}$.*

The case of distributive lattices

Theorem

Let P and Q be $\{=, \neq\}$-CQs with the same tuple of free variables $\overline{u}$, and K a bounded distributive lattice. Then, the following are equivalent:

1. $P \sqsubseteq_K Q$.
2. *For every identification P^{id} of P, there is a $\{=, \neq\}$-containment mapping $h_{P^{id}} : Q \longrightarrow P^{id}$.*

The case of various provenance semirings

Use the **abstractly tagged version of canonical databases** introduced by Green (2011). (E.g. for $\mathbb{N}[X]$, each tuple of the canonical database gets annotated with a different $p \in X$.)

Theorem

For $\{=, \neq\}$-CQs P, Q with the same tuple of free variables $\overline{u}$, the following are equivalent where $K \in \{\mathbb{B}[X], \mathbb{N}[X]\}$:

1. $P \sqsubseteq_K Q$,
2. *For every identification P^{id} of P, we have that $[\![P^{id}]\!]^{can_K(P^{id})} \leq [\![Q]\!]^{can_K(P^{id})}$.*
3. *For every identification P^{id} of P, there is an $\{=, \neq\}$-containment mapping $h_{P^{id}} : Q \longrightarrow P^{id}$* **exact for relational atoms**.

Theorem

For $\{=, \neq\}$-CQs P, Q with the same tuple of free variables $\overline{u}$, the following are equivalent:

1. $P \sqsubseteq_{Sorp[X]} Q$,
2. *For every identification P^{id} of P, we have that $[\![P^{id}]\!]^{can_{Sorp[X]}(P^{id})} \leq [\![Q]\!]^{can_{Sorp[X]}(P^{id})}$.*
3. *For every identification P^{id} of P, there is an $\{=, \neq\}$-containment mapping $h_{P^{id}} : Q \longrightarrow P^{id}$* **injective for relational atoms**.

Theorem

For $\{=, \neq\}$-CQs P, Q with the same tuple of free variables $\overline{u}$, the following are equivalent where $K \in \{Why[X], Trio[X]\}$:

1. $P \sqsubseteq_K Q$,
2. *For every identification P^{id} of P, we have that $[\![P^{id}]\!]^{can_K(P^{id})} \leq [\![Q]\!]^{can_K(P^{id})}$.*
3. *For every identification P^{id} of P, there is an $\{=, \neq\}$-containment mapping $h_{P^{id}} : Q \longrightarrow P^{id}$* **onto for relational atoms**.

Theorem

For $\{=,\neq\}$-CQs P, Q with the same tuple of free variables $\overline{u}$, the following are equivalent:

1. $P \sqsubseteq_{Lin[X]} Q$,
2. *For every identification P^{id} of P, we have that $[\![P^{id}]\!]^{can_{Lin[X]}(P^{id})} \leq [\![Q]\!]^{can_{Lin[X]}(P^{id})}$.*
3. *For every identification P^{id} of P, and every relational atom $R(\overline{y})$ of P^{id} there is a $\{=,\neq\}$-containment mapping $h_{P^{id}} : Q \longrightarrow P^{id}$ with $R(\overline{y})$ in the image of $h_{P^{id}}$.*

Theorem

The containment problems for $\{=,\neq\}$-CQs over $Lin[X]$, $Trio[X]$, $Why[X]$, $Sorp[X]$, $\mathbb{N}[X]$ and $\mathbb{B}[X]$ are in Π_2^p.

Theorem (Van der Meyden 1997)

The following problem is Π_2^p-hard: Given two safe conjunctive $\{=,\neq\}$-queries Q_1, Q_2, is it true that for every identification Q_1^{id} of Q_1 there is a $\{=,\neq\}$-canonical mapping $h_{Q_1^{id}} : Q_2 \longrightarrow Q_1^{id}$?

Theorem

The containment problem for $\{=,\neq\}$-CQs over $Lin[X]$, $Sorp[X]$, $Why[X]$ and $Trio[X]$, is Π_2^p-complete.

Next steps

- Extend these results to Unions (i.e. disjunctions) of CQs;
- Add negated atoms;
- Go beyond containment and study equivalence.

Tight inference and real-valued logic

Workshop on Logic, Algebra and Category Theory (LAC)

Fukuoka, 2025

This talk is based on ongoing work with Guillermo Badia, Ronald Fagin, Alexander Gray, Carles Noguera, Ryan Riegel, and
Sasha Rubin (University of Sydney)

Motivation: deduction in real-valued logics

Working in logics with truth-values in $[0, 1]$.

Assume we have the following background knowledge:

- value of $\varphi \to \theta$ is in $[0.7, 0.8]$
- value of φ is in $[0.4, 0.5]$

What can we deduce about θ? What is the tightest set of values that θ can take?

Real-valued logic

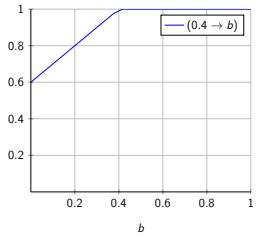
Syntax

- atomic variables: $V = \{p, q, r, \dots\}$
- constants $\overline{0}, \overline{1}$ and connectives $\neg, \to, \&$

Semantics of Łukasiewicz logic

- Assignment $M : V \to [0, 1]$
- $M(\overline{0}) = 0, M(\overline{1}) = 1$
- $M(\neg\varphi) = 1 - M(\varphi)$
- $M(\varphi \,\&\, \theta) = \max\{0, M(\varphi) + M(\theta) - 1\}$
- $M(\varphi \to \theta) = 1 - \max\{0, M(\varphi) - M(\theta)\}$

Write $\to, \&, \neg, \overline{0}, \overline{1}$ for both syntax and semantics.



$a \to b$ equals $1 - \max\{0, a - b\}$

if $a \leq b$ then $a \to b$ has value 1

if $a > b$ then $a \to b$ has value $1 - (a - b)$

Sentences

- Usually 1 is a favored value, e.g., M satisfies φ if $M(\varphi) = 1$.
- Instead, for a set $S \subseteq [0, 1]$ of truth-values, we will express statements

  $$\text{“}M(\varphi) \in S\text{”}$$

  - $(\varphi; S)$ is called a **sentence**
  - S is called the **information set**
  - $M \vDash (\varphi; S)$ if $M(\varphi) \in S$ Say **“M satisfies $(\varphi; S)$”**
  - $M \vDash B$ if M satisfies every sentence in B

Foundations of reasoning with uncertainty via real-valued logics, Fagin, Riegel, Gray (PNAS 2024) introduced a logic of multi-dimensional sentences $(\varphi_1, \cdots, \varphi_d; R)$ with $R \subseteq [0, 1]^d$

Tight-values set

Given:

- a finite set B of background sentences
- a target formula φ

Define the tight-values set T of $\langle B, \varphi \rangle$ as

$T = \bigcup\{M(\varphi) : M \text{ satisfies } B\}$

(all possible values of φ under assignments that satisfy B)

$T = \bigcap\{X : B \vDash (\varphi; X)\}$

(smallest set X of values such that $B \vDash (\varphi; X)$)

We say: from B tightly infer $(\varphi; T)$

Deduction illustrated

From $(\varphi \to \theta; [c, c'])$ and $(\varphi; [d, d'])$

tightly infer

$(\theta; [c \,\&\, d, c' \,\&\, d'])$ if $c' < 1$

$(\theta; [c \,\&\, d, 1])$ otherwise

Recall: $a \,\&\, b = \max\{0, a + b - 1\}$.

- If the value of $\varphi \to \theta$ is in $[0.7, 0.8]$, and
- if the value of φ is in $[0.4, 0.5]$, then
- we tightly infer the value of θ is in $[0.1, 0.3]$

Computational problems

Let T denote the tight-values set of $\langle B, \varphi \rangle$.

1. Decide $s \in T$ where s is a given rational number.
2. Compute (some representation of) T.

Assumption for this talk: Information sets in the background B are closed intervals with rational endpoints.

Use Mixed Integer Linear Programming

MILP

- $P(\bar{x}; \bar{z})$
- real-variables $\bar{x}$, Boolean variables $\bar{z}$
- set of linear constraints $t_1(\bar{x}, \bar{z}) \leq t_2(\bar{x}, \bar{z})$
- deciding if $P(\bar{x}; \bar{z})$ has a solution is NP-complete

For a set X of sentences, build a MILP P_X whose solutions capture the models satisfying X:

- For every subformula ψ, add variable x_ψ and constraint $0 \leq x_\psi \leq 1$
- If $\psi = \overline{0}$, add constraint $x_{\overline{0}} = 0$ (similarly $x_{\overline{1}} = 1$).
- If ψ is $\neg\varphi$, add constraint $x_\psi = 1 - x_\varphi$
- If ψ is $\varphi \to \theta$, add constraints for $x_\psi = 1 - \max\{0, x_\varphi - x_\theta\}$
  we handle this non-linearity by using a Boolean variable to distinguish if the max is 0 or not, and two additional real variables
- If ψ is $\varphi \,\&\, \theta$, is handled similarly.
- If $(\psi; [c, d]) \in X$ then add constraints $c \leq x_\psi \leq d$

Prop. The solutions of P_X, projected onto x_p for atoms p, are exactly the models M that satisfy X.

Let T be the tight-values set of $\langle B, \varphi \rangle$.

Cor. T is a finite union of closed intervals with rational endpoints, computable in EXPTIME.

- Take X to consist of B and $(\varphi; [0, 1])$
- Instantiate the Boolean variables in P_X, to get (exp-many) linear programs.
- LP solution-sets are convex, their projection onto x_φ are closed intervals.

Prop. T may consist of exponentially many disjoint intervals in the worst case.

- Cantor-set like construction.

Computational complexity

Let T be the tight-values set of $\langle B, \varphi \rangle$.

Cor. Deciding if a given rational is in T is NP-complete.

- NP: check if there is a solution to the MILP P_X where $X = B \cup \{ (\varphi; \{s\}) \}$ and s is the given rational
- NP-hard: via a simple reduction from Boolean satisfiability.

Thm. Deciding if a given rational is in the kth disjoint interval of T is in PSPACE.

- Find the ith disjoint interval, $i = 1, 2, \cdots, k$, only storing the last one ("on-the-fly")

Inference in neural networks

- $NN : [0,1]^k \to [0,1]$
- Inference: Given NN and output value s, find input values $\bar{a}$ s.t. $NN(\bar{a}) = s$.
- Some NNs can be represented as Łukasiewicz logic formulas.
  "Neural Networks and Rational McNaughton Functions", Amato, Di Nola, Gerla (MVLSC 2005)
- Inference: Given ψ and truth-value s, find truth-values $\bar{a}$ s.t. $\psi(\bar{a}) = s$.
- The set of such $\bar{a}$ is exactly the tight-values set of $\langle B, \bar{p} \rangle$ where B consists of $(\psi; \{s\})$, and $\bar{p}$ are the atoms in ψ.

Ongoing work

Extend analysis/techniques to handle:

- multi-dimensional sentences
- other real-valued logics, including probabilistic logics

A logic for reasoning about probabilities, Fagin, Halpern, Megiddo (I&C 1990)

Prop. The tight-values set may consist of an exponential number of disjoint intervals.

$$\left((\frac{1}{5} \to x_0) \wedge (x_0 \to \frac{2}{5})\right) \vee \left((\frac{3}{5} \to x_0) \wedge (x_0 \to \frac{4}{5})\right)$$

means

$$x_0 \in [0.2, 0.4] \cup [0.6, 0.8]$$

Adding

$$(x_0 \leftrightarrow (x_1 \oplus x_1)) \vee (x_0 \leftrightarrow (\neg x_1 \oplus \neg x_1))$$

shrinks the intervals into the left-half and into the right-half:

$$x_1 \in [0.1, 0.2] \cup [0.3, 0.4] \cup [0.6, 0.7] \cup [0.8, 0.9]$$

Repeat.

A glance at extensions of bi-intuitionistic logic

Hiroakira Ono (ono@jaist.ac.jp)

LAC 2025 at Kyushu University, September 30th, 2025

Bi-intuitionistic logic

Bi-intuitionistic logic is intuitionistic logic with *co-implication*.

What is co-implication?

- Implication

$$\vdash \gamma \wedge \alpha \Rightarrow \beta \quad \text{iff} \quad \vdash \gamma \Rightarrow \alpha \to \beta.$$

- co-implication = dual of implication.

$$\vdash \beta \Rightarrow \alpha \vee \gamma \quad \text{iff} \quad \vdash \beta \prec \alpha \Rightarrow \gamma.$$

$$\vdash \beta \Rightarrow \alpha \vee \gamma \quad \text{iff} \quad \vdash \beta \prec \alpha \Rightarrow \gamma.$$

- While the usual negation $\neg\alpha$ is an abbreviation of $\alpha \to \bot$, co-negation $\sim\alpha$ is defined to be an abbreviation of $\top \prec \alpha$.
- In classical logic CI,

$$\vdash \beta \Rightarrow \alpha \vee \gamma \quad \text{iff} \quad \vdash \beta \Rightarrow \alpha, \gamma \quad \text{iff} \quad \vdash \beta \wedge \neg\alpha \Rightarrow \gamma.$$

Thus, $\beta \prec \alpha$ means $\beta \wedge \neg\alpha$, which is equal to $\neg(\beta \to \alpha)$. In particular, $\sim\alpha$ is equal to $\neg\alpha$. Thus, nothing new comes out of introducing co-implication in classical logic.

Almost 50 years ago, bi-intuitionistic logic BiInt is introduced and discussed by Cecylia Rauszer. BiInt is a conservative extension of intuitionistic logic Int, whose Kripke frames are the same as those of Int, where

$$w \models \alpha \prec \beta \quad \text{if } u \models \alpha \text{ and } u \not\models \beta, \text{ for some } u \leq w.$$

Can we expect some interesting outcome of introducing co-implication? In the present talk, we will discuss syntactic aspects and symmetric features of logics with co-implication.

Sequent systems for Cl and Int

From syntactic point of view, the best way of making comparisons among Cl, Int and BiInt can be carried out by using sequent formulation. Here, we consider multiple-succedents sequents, thus each sequent is of the following form for some $m, n \geq 0$, where $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n$ are formulas;

$$\alpha_1, \ldots, \alpha_m \Rightarrow \beta_1, \ldots, \beta_n$$

Now, Gentzen's sequent system **LK** for classical logic consists of initial sequents and inference rules. Initial sequents are;

$$(1)\ \alpha \Rightarrow \alpha, \quad (2)\ \bot \Rightarrow\ , \quad (3)\ \ \Rightarrow \top.$$

- Cut

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \alpha, \Sigma \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi}\ (\mathrm{cut})$$

- Structural rules: (exchange, weakening, contraction)

For instance, exchange rules;

$$\frac{\Gamma, \alpha, \beta, \Delta \Rightarrow \Pi}{\Gamma, \beta, \alpha, \Delta \Rightarrow \Pi} \qquad \frac{\Gamma \Rightarrow \Sigma, \alpha, \beta, \Pi}{\Gamma \Rightarrow \Sigma, \beta, \alpha, \Pi}$$

- Cut

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \alpha, \Sigma \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi}\ (\mathrm{cut})$$

- Structural rules: (exchange, weakening, contraction)

For instance, exchange rules;

$$\frac{\Gamma, \alpha, \beta, \Delta \Rightarrow \Pi}{\Gamma, \beta, \alpha, \Delta \Rightarrow \Pi} \qquad \frac{\Gamma \Rightarrow \Sigma, \alpha, \beta, \Pi}{\Gamma \Rightarrow \Sigma, \beta, \alpha, \Pi}$$

Observe the "mirror" symmetry in these rules.

- Rules for disjunction and conjunction

$$\frac{\alpha, \Gamma \Rightarrow \Delta \quad \beta, \Gamma \Rightarrow \Delta}{\alpha \vee \beta, \Gamma \Rightarrow \Delta}\ (\vee \Rightarrow)$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha}{\Gamma \Rightarrow \Delta, \alpha \vee \beta}\ (\Rightarrow \vee 1) \qquad \frac{\Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \vee \beta}\ (\Rightarrow \vee 2)$$

$$\frac{\alpha, \Gamma \Rightarrow \Delta}{\alpha \wedge \beta, \Gamma \Rightarrow \Delta}\ (\wedge 1 \Rightarrow) \qquad \frac{\beta, \Gamma \Rightarrow \Delta}{\alpha \wedge \beta, \Gamma \Rightarrow \Delta}\ (\wedge 2 \Rightarrow)$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \wedge \beta}\ (\Rightarrow \wedge)$$

- Rules for disjunction and conjunction

$$\frac{\alpha, \Gamma \Rightarrow \Delta \quad \beta, \Gamma \Rightarrow \Delta}{\alpha \vee \beta, \Gamma \Rightarrow \Delta}\ (\vee \Rightarrow)$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha}{\Gamma \Rightarrow \Delta, \alpha \vee \beta}\ (\Rightarrow \vee 1) \qquad \frac{\Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \vee \beta}\ (\Rightarrow \vee 2)$$

$$\frac{\alpha, \Gamma \Rightarrow \Delta}{\alpha \wedge \beta, \Gamma \Rightarrow \Delta}\ (\wedge 1 \Rightarrow) \qquad \frac{\beta, \Gamma \Rightarrow \Delta}{\alpha \wedge \beta, \Gamma \Rightarrow \Delta}\ (\wedge 2 \Rightarrow)$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \wedge \beta}\ (\Rightarrow \wedge)$$

Observe the "mirror" symmetry between rules for $\vee$ and $\wedge$.

- Rules for implication and negation (of **LK**)

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \beta, \Sigma \Rightarrow \Pi}{\alpha \to \beta, \Gamma, \Sigma \Rightarrow \Delta, \Pi}\ (\to\Rightarrow) \qquad \frac{\alpha, \Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \to \beta}\ (\Rightarrow\to)$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha}{\neg\alpha, \Gamma \Rightarrow \Delta}\ (\neg\Rightarrow) \qquad \frac{\alpha, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\alpha}\ (\Rightarrow\neg)$$

Next, consider the sequent system **LJ'** for intuitionistic logic, which is obtained from **LK** by restricting Δ to be empty in both $(\Rightarrow\to)$ and $(\Rightarrow\neg)$ (by S. Maehara 1954).

Sequent system **LBJ** for bi-intuitionistic logic BiInt

Lastly, **LBJ** is obtained from **LJ'** by adding rules for co-implication and co-negation.

$$\frac{\alpha \Rightarrow \Delta, \beta}{\alpha \prec \beta \Rightarrow \Delta}\ (\prec\Rightarrow) \qquad \frac{\Gamma \Rightarrow \Delta, \alpha \quad \beta, \Sigma \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi, \alpha \prec \beta}\ (\Rightarrow\prec)$$

$$\frac{\Rightarrow \Delta, \beta}{\sim\beta \Rightarrow \Delta}\ (\sim\Rightarrow) \qquad \frac{\beta, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \sim\beta}\ (\Rightarrow\sim)$$

Now, the "mirror" symmetry is established between rules for $\to$ and $\prec$, and also between $\neg$ and $\sim$.

$$\frac{\Gamma \Rightarrow \Delta, \alpha \quad \beta, \Sigma \Rightarrow \Pi}{\alpha \to \beta, \Gamma, \Sigma \Rightarrow \Delta, \Pi}\ (\to\Rightarrow) \qquad \frac{\alpha, \Gamma \Rightarrow \beta}{\Gamma \Rightarrow \alpha \to \beta}\ (\Rightarrow\to)$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha}{\neg\alpha, \Gamma \Rightarrow \Delta}\ (\neg\Rightarrow) \qquad \frac{\alpha, \Gamma \Rightarrow}{\Gamma \Rightarrow \neg\alpha}\ (\Rightarrow\neg)$$

♣ Cut elimination

Cut elimination for a sequent system **S** says:

- if a sequent is provable in **S** then it is also provable in **S** without any application of cut rule.

1. Cut elimination holds for **LK** and **LJ**. (G. Gentzen)
2. Cut elimination holds for **LJ'**. (S. Maehara)

♣ Cut elimination

Cut elimination for a sequent system **S** says:

- if a sequent is provable in **S** then it is also provable in **S** without any application of cut rule.

1. Cut elimination holds for **LK** and **LJ**. (G. Gentzen)
2. Cut elimination holds for **LJ'**. (S. Maehara)

3. Cut elimination fails for **LBJ**. (Pinto-Uustalu 2009)

Here is a (slightly simplified) counterexample given by Uustalu:

$$q \Rightarrow p, r \rightarrow (q \prec p)$$

Still we can show the following. We say that a sequent system **S** has *analytic cut property*, when

- every sequent which is provable in **S** has a proof P in **S** such that each cut formula is a subformula of the lower sequent in every application of cut rule. (Consequently, the proof P has subformula property.)

4. Analytic cut property holds for **LBJ**. (Kowalski-O 2017)

Some immediate consequences:

- **LBJ** is a conservative extension of **LJ'**.
- BiInt is decidable.

Note that different from Int, disjunction property doesn't hold for BiInt, since the formula $\alpha \vee \sim\alpha$ is always provable.

Moreover,

Theorem

Craig interpolation property holds for BiInt.

- This is shown first in (Kowalski-O 2017) by extending Maehara's method (1960).
- Another proof is given in (O-Sano 2022) by extending Mints' method (Mints 2001) to **LBJ**, originally applied to **LJ'** (in which "imaginary" interpolants is used in addition to "real" ones).
- What is essential in these proofs lies in the fact that any existing analytic cut in a given proof will not disturb the implementation of these methods, as subformula property still holds.

Going further on BiInt

In the calculus **LBJ**, $(\Rightarrow\prec)$ and $(\neg\Rightarrow)$ can be applied without any restriction, while $(\prec\Rightarrow)$ and $(\Rightarrow\neg)$ are not.

- If a sequent $\alpha_1,\ldots,\alpha_m \Rightarrow \beta$ is provable, then $\sim\beta \Rightarrow \sim\alpha_1,\ldots,\sim\alpha_m$ is provable.
- If a sequent $\beta \Rightarrow \alpha_1,\ldots,\alpha_m$ is provable, then $\neg\alpha_1,\ldots,\neg\alpha_m \Rightarrow \neg\beta$ is provable.
- Hence, by applying the above in a consequtive way, we can show that if a sequent $\alpha_1,\ldots,\alpha_m \Rightarrow \beta$ is provable, then $\odot\alpha_1,\ldots,\odot\alpha_m \Rightarrow \odot\beta$ is provable, where $\odot\delta$ means $\neg\sim\delta$.
- In particular, if δ is provable, so is $\odot\delta$.

Duality function

For each formula α, a formula α^d is defined inductively as follows:

- $p^d = p$ for every propositional variable p,
- $\top^d = \bot$ and $\bot^d = \top$,
- $(\alpha\wedge\beta)^d = \alpha^d \vee \beta^d$ and $(\alpha\vee\beta)^d = \alpha^d \wedge \beta^d$,
- $(\alpha\to\beta)^d = \beta^d \prec \alpha^d$ and $(\alpha\prec\beta)^d = \beta^d \to \alpha^d$,
- $(\neg\alpha)^d = \sim\alpha^d$ and $(\sim\alpha)^d = \neg\alpha^d$.

The mapping d defined by $d(\alpha) = \alpha^d$ is called the *duality function*. It is naturally extended to a mapping over finite sequences of formulas. Obviously, $(\alpha^d)^d = \alpha$ holds.

(Discussed also in (Restall 1997) and (Wolter 1998).)

From our observation on "mirror" symmetry of initial sequents and rules of **LBJ**, we can easily show the following by using the induction on the length of a given proof.

Theorem

A sequent $\Gamma \Rightarrow \Delta$ *is provable in* **LBJ** *(without cut) iff* $\Delta^d \Rightarrow \Gamma^d$ *is provable in* **LBJ** *(without cut).*

- In particular, a formula α is provable in **LBJ** iff $\neg(\alpha^d)$ is provable in it.
- From our result it follows that $p, (p \to q) \prec r \Rightarrow q$ is another counterexample of cut eliminability of **LBJ**, as this is the dual to Uustalu's one; $q \Rightarrow p, r \to (q \prec p)$.

Logics over BiInt

Definition

A set L of formulas is a *logic over* BiInt (or, an extension of BiInt) iff

- it is closed under substitution,
- it is closed under the provability in **LBJ**, i.e. if $\alpha_1, \ldots, \alpha_m \Rightarrow \beta$ is provable in **LBJ** for $\alpha_1, \ldots, \alpha_m \in$ L then β is also in L,
- it is closed under $\odot$, i.e. if $\alpha \in$ L then $\odot\alpha \in$ L.

The smallest logic including a set U of formulas is denoted by BiInt[U], and also by BiInt[$\gamma_1, \ldots, \gamma_k$] when $U = \{\gamma_1, \ldots, \gamma_k\}$.

■ **Q.1** Is BiInt[K] always conservative over any K over Int? (Wotler 1998)

When K has the finite model property, the answer is positive, as every finite Heyting algebra can be naturally extended to a bi-Heyting algebra.

■**Q.2** Which logical property of a logic K over Int is preserved by BiInt[K]?

Symmetric lattice structure of logics over BiInt

Let $\mathcal{L}$ be the set of all logics over BiInt, which in fact forms a lattice. For a logic L in $\mathcal{L}$, let S_{L} be the set of formulas $\{\neg(\alpha^d) : \alpha \in \mathsf{L}\}$. We define L^{m} to be the smallest logic in $\mathcal{L}$ which includes S_{L}. L^{m} is called the *mirror image of* L (cf. (Wolter 1998)).

Lemma

A sequent $\Gamma \Rightarrow \Delta$ *is provable in* L *iff* $\Delta^d \Rightarrow \Gamma^d$ *is provable in* L^{m}.

If a logic L is axoimatized by axiom schemes $[\varphi_1, \ldots, \varphi_k]$, then the logic L^{m} can be axoimatized by axiom schemes $[\neg(\varphi_1^d), \ldots, \neg(\varphi_k^d)]$.

Examples:

- Classical logic can be expressed as BiInt$[\mathrm{p} \vee \neg\mathrm{p}]$, which is equal to its mirror image BiInt$[\neg(\mathrm{p} \wedge \sim\mathrm{p})]$.
- The logic BiInt$[(\mathrm{p} \to \mathrm{q}) \vee (\mathrm{q} \to \mathrm{p})]$ is a bi-intuitionistic extension of Gödel logic, Int$[(\mathrm{p} \to \mathrm{q}) \vee (\mathrm{q} \to \mathrm{p})]$. Meanwhile, its mirror image BiInt$[\neg((\mathrm{p} \prec \mathrm{q}) \wedge (\mathrm{q} \prec \mathrm{p}))]$ is a conservative extension of Int (Wolter 1998).

Let m be a mapping on $\mathcal{L}$ defined by $\mathsf{m}(\mathsf{L}) = \mathsf{L}^{\mathsf{m}}$.

Theorem

The mapping m *is a (complete) lattice isomorphism over* $\mathcal{L}$*, which is moreover involutive, i.e.* $\mathsf{m} \circ \mathsf{m} = \mathrm{id}$.

■**Q.3** Which logical property is preserved by the mapping m?

Theorem

The following properties are preserved by the mapping m.

- *decidability,*
- *Kripke completeness,*
- *finite model property,*
- *Craig's interpolation property,*
- *Maksimova's variable separation property.*

Bi-intuitionistic tense logics

What we have shown so far can be extended to bi-intuitionistic tense logics. The basic system BiSKt (Stell, Schmidt and Rydeheard 2016) is an extension of BiInt with two monotone, tense operators $\blacklozenge$ and $\Box$ satisfying that

$$\vdash \blacklozenge\alpha \Rightarrow \beta \quad \text{iff} \quad \vdash \alpha \Rightarrow \Box\beta.$$

Obviously, we can see that the "mirror" symmetry holds between $\blacklozenge$ and $\Box$.

Together with K. Sano, we are preparing a paper in which sufficient conditions are given in terms of Kripke frames, for logics over BiSkt (and also over BiInt) to have Craig and also Lyndon interpolation properties.

A key notion here is bisimulation products, whose idea goes back to (Maksimova 1980), (O 1986) and (Marx 1998).

■**Q.4** How many logics are there over BiInt that have Craig Interpolation property? Finitely many?

On some forms of logical connections between theories

Zbigniew Król

International Center for Formal Ontology

Faculty of Administration and Social Sciences
Warsaw University of Technology
Poland

Logic, Algebra, and Category Theory
Workshop
Kyushu University
Fukuoka. 30.09.2025

Alter-theories

A sentence A is called independent of T iff when both T + A and T + ~ A are consistent.

If T is a consistent theory with non-logical axioms A1, A2, …, then it is always possible to choose a subset of axioms Ak1, Ak2, …, such that these axioms are independent of the others and their set of logical consequences is identical to T.

However, we do not know whether the axioms of every {~Tki} theory i.e. a theory in which one of the axioms, say Aki, is replaced with its negation, i.e. ~Aki, are still independent.

If they are still independent, then we can construct another consistent theory, e.g. ~Ak1, ~Ak2, Ak3, ….

In particular, under certain conditions, it is possible to create a consistent theory, the **so-called alter-theory of T, AT,** whose axioms are the negations of all specific axioms of the initial theory T.

Alter-theories

Example. T- consistent theory, non-logical axioms

A1: $\exists x.\phi(x)$

A2: $\exists x.\sim\phi(x)$.

If these sentences are consistent, then they are also independent, since the theories {~A1, A2}, {A1, ~A2} are consistent. However, their axioms are no longer independent, as they are ”negatively dependent”: $\lfloor_{L}\sim A1 \rightarrow A2\rfloor_{J}$, which follows from the fact that $\sim A1 \equiv \sim[\exists x.\phi(x)] \equiv [\forall x.\sim\phi(x)] \rightarrow \exists x.\sim\phi(x) \equiv A2$. The same applies to $\lfloor_{L}\sim A2 \rightarrow A1\rfloor_{J}$.

Alter-theories cont.

It is always possible to construct a consistent AT for a given consistent T if the specific axioms of T are **negatively independent (negative-independent)**, i.e., if none of the remaining (non-negated) axioms of T can be derived from the conjunction of any finite number of negations of the non-logical axioms of T.
Similarly, we say that the axioms of a consistent theory T are **partially negatively independent** if, from any finite number of negations of these axioms together with a finite number of non-negated axioms, it is impossible to derive any of the remaining axioms of T.
The axioms of a consistent theory T are **absolutely independent** if they are partially negatively independent, and negatively independent.

4

Types of Alter-theories

The concepts introduced above allow us to distinguish between different types of consistent alter-theories:

AT – alter-theories, i.e., theories that arise from a given consistent theory T **by negating all of its non-logical axioms.**

P-AT – partial alter-theories, i.e., theories that arise **by negating only some of the axioms** of the initial theory T, while the remaining axioms are identical to the axioms of T.

Other types can also be distinguished, e.g., **M-AT** (mixed alter-theories), **M-P-AT** (partial mixed alter-theories), **LT** – (lower theories), **UT** – (upper theories), etc.

5

Types of Alter-theories

Usually, in mathematics and logic, only P-AT and M-P-AT are considered, and this is usually limited to the negation of only one axiom, after demonstrating, of course, that it is independent of the others.
Examples. ZF(C)+CH, ZF(C)+~CH, or commutative and non-commutative groups – these are P-AT.
Euclidean geometry, Non-euclidean geometries or Non-well founded set theories – these are typical M-P-ATs, i.e., they do not use a "complete" negation of an axiom (e.g., Euclid's fifth postulate, the axiom of extensionality), but only a "partial" negation of it, i.e., they accept as an axiom a statement or property that results from the "complete" negation of a given axiom.

6

Properties of Alter-theories

Consistent Alter-theories have very interesting properties.

For example, if A is a non-logical thesis of AT, then it cannot be a thesis of T and vice versa, i.e., T and AT do not have any specific (non-logical) theses in common.
This allows (under certain assumptions) for purely syntactic inquiry of sentences independent of a given initial theory T.
Similarly, if T and its P-AT (provided they are consistent) have common non-logical consequences, then these consequences are exclusively the consequences of their common (non-negated) axioms.

7

Properties of Alter-theories

A given theory T may have a consistent AT but may not have consistent P-ATs (or only some of the possible P-ATs are consistent) and *vice versa*.
ATs are a certain idealization, but it turns out that the axioms of a given consistent theory T, even if they are dependent on each other, if they are negatively independent, then there exists a consistent AT (P-ATs must be inconsistent, at least some of them).
The above conclusion allows us to construct AT without the special and laborious task of determining which axioms are dependent on others.

8

Properties of Alter-theories

If we limit ourselves to theories (not necessarily only first-order) based on functional calculus (logic with functions) with identity (=) and with modus ponens as the only rule of inference, then – as is well known – the specific axioms of a given theory, e.g., group theory or Peano arithmetic of natural numbers, PAr, are most often dependent on the axioms of identity and/or vice versa.
Therefore, when constructing an alter-theory of such a theory, the axioms of the theory of identity ID= must also be negated.
This means that - if there is a consistent AT of a certain theory with identity, then it is (often) not itself a theory with identity.

9

Theory of Identity

ID.1. ∀x. x=x;
ID.2. ∀x,y. x=y→y=x;
ID.3. ∀x,y. x=y^y=z→x=z;
ID.4. (P). ∀…∀x,y. (x=y ^ P(x1, …, x, ….)) → P(x1, …, y, ….), for each primitive relation P, where P(x1, …, y, ….) is a formula obtained from the formula P(x1, …, x, ….), in which all occurrences of the free variable x have been replaced by the variable y (provided that y is free for x in P(x1, …, x, ….));
ID.5. (F). ∀…∀x,y. x=y → (F(x1, …, x, ….) = F(x1, …, y, ….)), for each primitive function F, where F(x1, …, y, ….) is a term obtained from the term F(x1, …, x, ….), in which all occurrences of the free variable x have been replaced by the variable y.

10

Properties of Alter-theories

AID= has the following axioms:
AID.1. ∃x. ~I(x,x);
AID.2. ∃x,y. I(x,y) ∧ ~I(y,x);
AID.3. ∃x,y. I(x,y) ∧ I(y,z) ∧ ~I(x,z);
AID.4. (F) ∃x,y. I(x,y) ∧ ~(F(x1, …, x, ….) = F(x1, …, y, ….));
AID.5. (R) ∃x,y. I(x,y) ∧ R(x1, …, x, ….) ∧ ~R(x1, …, y, ….)).

AID= axioms are not all independent, because, for example, in ID=, not-ID.5. is inconsistent with ID.1.; (cf. P(x,y)/x=y). However, these axioms are negatively independent, because the alter-theory **AID=** is consistent, which can be shown by constructing a model of this theory, e.g., a non-transitive model, where we interpret the relation I(x,y) as "ϵ" ("x belongs to y").

11

PAr

PA.a$^{=}$. ∀x. x=x;
PA.b$^{=}$. ∀x,y. x=y → y=x;
PA.c$^{=}$. ∀x,y,z. x=y ^ y=z → x=z.
PA.d$^{=}$. (S,+, •) ∀…∀x,y. x=y → (F(x1, …, x, ….) = F(x1, …, y, ….))
PA.e$^{=}$.. (=) ∀…∀x,y. (x=y ^ P(x1, …, x, ….)) → P(x1, …, y, ….)
Specific (non-logical) axioms:
(PA1) ∀x. 0 ≠ S(x);
(PA2) ∀x,y. S(x)=S(y) → x=y;
(PA3) ∀x. x+0 = x;
(PA4) ∀x,y. x+S(y) = S(x+y);
(PA5) ∀x. x•0 = 0;
(PA6) ∀x,y. x•S(y) = (x•y) + x;
and the induction axiom schema:
(Ind). (ϕ) ∀u_1, u_2,…, u_n{[ϕ(0) ^ ∀x(ϕ(x) → ϕ(S(x))] → ∀x.ϕ(x)}.

12

A-PAr

Note that when constructing A-PAr, we cannot create a consistent negation of the induction axiom schema because, for example, for formulas ϕ without parameters, negating every axiom falling under the **Ind** schema will immediately give us a pair of contradictory statements ϕ and ~ϕ. This means that some axioms falling under the **Ind** schema are negatively dependent.
We can therefore only create partial alter-theories of PAr.

So, the theory
(A-Ind). (ϕ) $\exists u_1, u_2, \ldots, u_n\{[\phi(0) \wedge \forall x(\phi(x) \rightarrow \phi(S(x))] \rightarrow \exists x.\sim\phi(x)\}$
is inconsistent.

13

A-PAr

Ind is not an a priori, unchanging construct that is attached to a theory in an unaltered form but depending on the other axioms of a given theory, we determine which functions, constants, and relations can form the relevant formulas ϕ. **Ind** is also not a concept that is consistent only with PAr or theories in which PAr is constructable, such as ZF(C).

For example, let us consider the theory of the axioms of identity theory, AI and Ind, i.e.
ID=
A.1a. ∀x. S(x)=0
(Ind). (ϕ) $\forall u_1, u_2, \ldots, u_n\{[\phi(0) \wedge \forall x(\phi(x) \rightarrow \phi(S(x))] \rightarrow \forall x.\phi(x)\}$.
In this theory, using Ind, we can obtain the thesis ∀x. S(x)=x.

14

Let us first examine the alter-theory of PAr-(Ind),i.e. the alter-theory of PAr without the **Ind** schema. This theory, let us call it **1P-A-PAr**, has the following axioms:
AID.1. ∃x. ~I(x,x);
AID.2. ∃x,y. I(x,y) ∧ ~I(y,x);
AID.3. ∃x,y. I(x,y) ∧ I(y,z) ∧ ~I(x,z);
AID.4. (S.+, ˣ) ∃x,y. I(x,y) ∧ ~(F(x1, …, x, ….) = F(x1, …, y, ….));
AID.5. (I) ∃x,y. I(x,y) ∧ R(x1, …, x, ….) ∧ ~R(x1, …, y, ….)).

(A-PAr1) ∃x. 0 = S(x); [i.e. ∃x. I(0,S(x))]
(A-PAr2) ∃x,y. S(x)=S(y) ∧ ~(x=y); [i.e. ∃x,y. I(S(x),S(y)) ∧ ~I(x,y)]
(A-PAr3) ∃x. ~(x+0 = x); [as above]
(A-PAr4) ∃x,y. ~(x+S(y) = S(x+y)); [as above]
(A-PAr5) ∃x. ~(xˣ0 = 0); [as above]
(A-PAr6) ∃x,y. ~(xˣS(y) = (xˣy) + x); [as above]

15

It turns out that it is possible to construct a model of this theory (again, it suffices to interpret I(x,y) as a relation "ϵ" ("x belongs to y") and select appropriate elements so that the axioms of our theory are satisfied.

This means that 1P-A-PAr is consistent, and therefore that its axioms are negatively independent.

This also means that we have a proof of the consistency of PAr-(Ind), since AT is consistent iff T is consistent.

Conclusions:

1. The AT construction method revives hopes for the possibility of saving Hilbert's original program for a broader class of theories (/sentences) than previously thought.
2. The AT method allows the independence of (some) statements to be examined in a purely syntactic manner.
3. There is a class of problems concerning the stoping of a Turing machine that are undecidable for a single Turing machine but become decidable for two Turing machines operating "in parallel," i.e., one examines the problem for formula A in T and the other for non-A in the corresponding alter-theory.

Thank you for your attention!

Ideal K-calculus:

Axiom 1 (extensionality): Two sets are identical if and only if they have the same elements.

Axiom (schema) 2 (comprehension): For every property P, there exists a set consisting of those and only those elements that have this property.

E.1. $\forall x\,[(x \in y \equiv x \in z) \rightarrow y = z]$

K.1. $(\varphi)\ \forall x \exists z\,[x \in z \equiv \varphi(x)]$

19

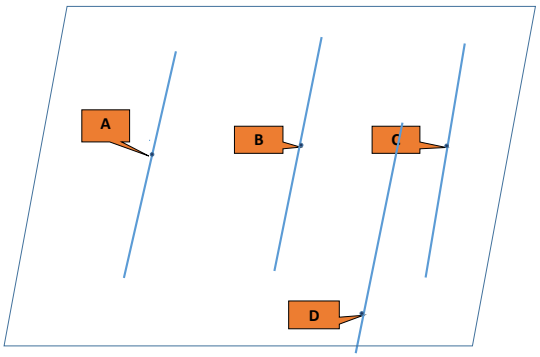
The inconsistency of K.1 formulated in L_c is obvious for intuitive reasons. In L_c, we can formulate the intentional property "x is not an element of any set," i.e., $\varphi^P(x) \equiv \sim(\exists y.\ x \in y)$. No set can correspond to this property, in particular the set postulated by K.1. For if $x \in z$ and at the same time $\sim(\exists y.\ x \in y)$, we obtain a contradiction. Russell's property is a special case of $\varphi^P(x)$ and follows from it. and we obtain a contradiction.

In first-order logic, no two-argument relation (and therefore also the relation "$\in$") can be defined by a certain formula $\varphi(x)$ in this theory. K.1. is false in a homogeneous environment (i.e., with one type of belonging relation) for purely intuitive (intentional) reasons.

20

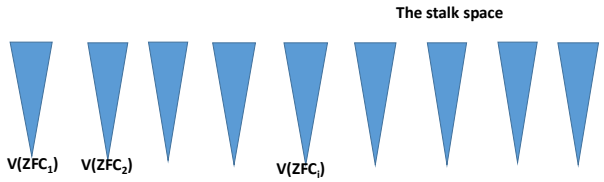
An example of substitution

E^2: point / perpendicular line



21

General description of global substitutions



Bn(*I*) -
boundle of sets over the base space *I*

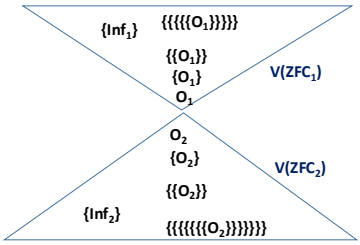
***I* – the set of indexes (R, N, E^2, …)**
= the base space

Bn(*I*)

22

Example: local substitution of ZFC in ZFC from the point of view of models and a theory

- A substitution of ZFC_2 in ZFC_1: substitute a transitive model of ZFC_2 for the empty set **0** from ZFC_1; "local singular self-substitution".
- The two "copies" of ZFC: ZFC_1 and ZFC_2, must be written down in such a way that they have no relational and constant symbols in common: e.g. the symbols "$\in_1$" and "$=_1$" in ZFC_1 correspond to "$\in_2$" and "$=_2$" in ZFC_2. The same concerns the symbols designating constants: "$\mathbf{0}_1$" and "$\mathbf{0}_2$", infinite set, etc.
- It is possible to create the third theory, T(ZFC_1/ ZFC_2), in the following way. Write down every axiom of ZFC_1, and add to them the set of renamed and supplemented axioms of ZFC_2

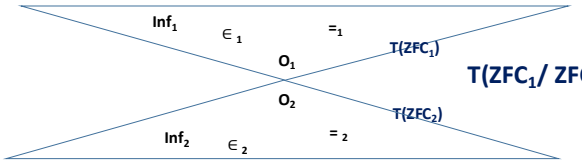


V(ZFC_1/ ZFC_2)

23

Example: local substitutions of ZFC in ZFC - a formal description

- Ax. of extensionality of ZFC_2: $\forall \mathbf{x}, \mathbf{y}, \mathbf{z}.\ [(\mathbf{x} \in_2 \mathbf{0}_1 \wedge \mathbf{y} \in_2 \mathbf{0}_1 \wedge \mathbf{z} \in_2 \mathbf{0}_1) \rightarrow (\mathbf{x} \in_2 \mathbf{y} \equiv \mathbf{x} \in_2 \mathbf{z} \rightarrow \mathbf{y} =_2 \mathbf{z})]$.
- Axiom of pairing: $\forall \mathbf{x} \in_2 \mathbf{0}_1, \forall \mathbf{y} \in_2 \mathbf{0}_1 \exists \mathbf{z} \in_2 \mathbf{0}_1 \forall \mathbf{u} \in_2 \mathbf{z}. (\mathbf{x} \in_2 \mathbf{z} \equiv \mathbf{u} =_2 \mathbf{x} \vee \mathbf{u} =_2 \mathbf{y})$.
- Axiom of infinity: $\exists \mathbf{z} \in_2 \mathbf{0}_1. [(\mathbf{0}_1 \in_2 \mathbf{z}) \wedge (\forall \mathbf{x}. \mathbf{x} \in_2 \mathbf{z} \rightarrow \mathbf{x} \cup \{x\} \in_2 \mathbf{z})]$.
- In a similar way the remaining axioms of ZFC_2 should be adapted.
- One new axiom, the axiom of transitivity: $\forall \mathbf{x}, \mathbf{y}. (\mathbf{x} \in_2 \mathbf{y} \wedge \mathbf{y} \in_2 \mathbf{0}_1) \rightarrow (\mathbf{x} \in_2 \mathbf{0}_1)$, in order to have every set from ZFC_2 "inside" $\mathbf{0}_1$.
- The usual rules concerning the substitutions of identical objects are restricted, e.g. if $(\mathbf{x} =_1 \mathbf{y})$, one can substitute **x/y** only in the formulae built with the use of "$=_1$" or "$\in_1$".
- We refer to the theory T(ZFC_1/ZFC_2) as *substitutional theory* which corresponds to a *substitutional model*.

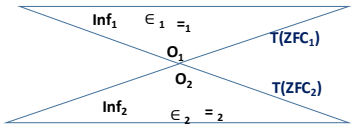


T(ZFC_1/ ZFC_2)

24

Example: local substitutions of ZFC in ZFC

- One has to decide, for instance, if $\mathbf{0}_1 =_1 \mathbf{0}_2$, or $\forall \mathbf{x}.[(\mathbf{x} \in_2 \mathbf{0}_1) \rightarrow (\mathbf{x} =_1 \mathbf{0}_1)]$ or $\forall \mathbf{x}.[\sim (\mathbf{x} \in_2 \mathbf{0}_1) \rightarrow (\mathbf{x} =_1 \mathbf{0}_2)]$, or introduce *global relation of identity*: $\forall \mathbf{x}, \mathbf{y}.(\mathbf{x} = \mathbf{y}) \equiv (\mathbf{x} =_1 \mathbf{y} \wedge \mathbf{x} =_2 \mathbf{y})$.
- In $T(ZFC_1/ZFC_2)$ unrestricted quantifiers in the part corresponding to ZFC_1 are used. The resulting possible mixture of sets belonging to the universes of the component ZFC systems causes that the resulting $V(ZFC_1/ ZFC_2)$ is not "a pure" substitution without some additional axioms.
- In the case when $\sim(\mathbf{0}_1 =_1 \mathbf{0}_2)$, the quantifiers act over the objects of both initial systems of ZFC, however, **the quantifiers cannot detect all the relevant ontological commitment of $T(ZFC_1/ZFC_2)$. The objects of ZFC_1/ZFC_2 "change their nature" and this fact is "unseen" by the quantifiers.**



$T(ZFC_1/ ZFC_2)$

25

- It is now straightforward to define in $T(ZFC_1/ZFC_2)$ the relations "$\in_2$" and "$=_2$", as well as the constant $\mathbf{0}_1$. from ZFC_2. For instance, $\forall \mathbf{x}, \mathbf{y}.(\mathbf{x} \in_2 \mathbf{y})_{ZFC2} \equiv \forall \mathbf{x}, \mathbf{y}.(\mathbf{x} \in_2 \mathbf{y})_{ZFC1/ZFC2}$ or $\mathbf{0}_{1ZFC2} =_2 \mathbf{0}_1$. Therefore ZFC_2 (and ZFC_1) is interpretable in $T(ZFC_1/ZFC_2)$.
- Mutual interpretability of theories, especially of ZFC, is inquired in M. Friedman's *concept calculi*. The obtained results demonstrates the fallibility of **QCE** and **M-QCE**. Moreover, such examples as ZFC_1/ZFC_2, also indicate the inadequacy of the general idea of Quine's ontological commitments of theories because **extensional theories are intensionally undetermined.**
- ZFC can speak about sets or be interpreted as, say, a recipe.
- **Ontology appears independent from quantification; from the point of view of extensional theories which are intensionally undetermined, ontological properties belong to unessential "intensional decoration".**

26

A tree-shaped tableau for Linear Time Temporal Logic

Mark Reynolds, The University of Western Australia

September 2025

A tree-shaped tableau for Linear Time Temporal Logic

Propositional linear time temporal logic (LTL) is the standard temporal logic for computing applications and many reasoning techniques and tools have been developed for it. Tableaux for deciding satisfiability have existed since the 1980s. However, the tableaux for this logic do not look like traditional tree-shaped tableau systems and their processing is often quite complicated. In this talk we describe a novel style of tableau rule which supports a new simple traditional-style tree-shaped tableau for LTL. We outline the proof that it is sound and complete. As well as being simple to understand, to introduce to students and to use, it is also simple to implement and is competitive against state of the art systems. It is particularly suitable for parallel implementations.

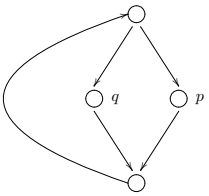
Outline

- The Logic LTL
- A tableau for LTL-Sat
- Soundness
- Completeness

Model of a System:

Thus we suppose that we can model the system as a finite state machine: a set of states and a set of allowed transitions from some states to other states.

In order to allow abstraction of observable basic, or atomic properties, we also suppose that there are a set of atomic properties, and that some properties are true at some states.



Properties:

Examples.
If a process stays in a waiting state then it will eventually be given a resource.
The program will eventually terminate.
The program will never return a null value.
The two processes will never both be in their critical states at the same time.
The program will never be in a logged-in state unless a correct PIN has been entered beforehand.
The system will always dispense a product after the correct money has been inserted.

Structures:

We assume a countable set $\mathcal{L}$ of propositional atoms, or atomic propositions.

A transition structure is a triple (S, R, g) with S a finite set of states, $R \subseteq S \times S$ a binary relation and for each $s \in S$, $g(s) \subseteq \mathcal{L}$.

R is the *transition relation* and *labelling* g tells us which atoms are true at each time.

R is assumed to be total: every state has at least one successor $\forall x \in S. \exists y \in S$ s.t.$(x, y) \in R$

Fullpaths:

Given a structure (S, R, g).
An ω-sequence of states $\langle s_0, s_1, s_2, ...\rangle$ from S is a fullpath (through (S, R, g)) iff for each i, $(s_i, s_{i+1}) \in R$.

If $\sigma = \langle s_0, s_1, s_2, ...\rangle$ is a fullpath then we write $\sigma_i = s_i$, $\sigma_{\geq j} = \langle s_j, s_{j+1}, s_{j+2}, ...\rangle$ (also a fullpath).

LTL Syntax:

Define some strings of symbols as (well formed) formulas, or wffs of LTL.

If $p \in \mathcal{L}$ then p is a wff.

If α and β are wff then so are $\neg\alpha$, $\alpha \wedge \beta$, $X\alpha$, and $\alpha U \beta$.

Read: Not, and(conjunction), tomorrow (or next), and until.

LTL Semantics:

Write $M, \sigma \models \alpha$ iff the formula α is true of the fullpath σ in the structure $M = (S, R, g)$ defined recursively by:

$M, \sigma \models p$	iff	$p \in g(\sigma_0)$, for $p \in \mathcal{L}$
$M, \sigma \models \neg\alpha$	iff	$M, \sigma \not\models \alpha$
$M, \sigma \models \alpha \wedge \beta$	iff	$M, \sigma \models \alpha$ and $M, \sigma \models \beta$
$M, \sigma \models X\alpha$	iff	$M, \sigma_{\geq 1} \models \alpha$
$M, \sigma \models \alpha U \beta$	iff	there is some $i \geq 0$ such that $M, \sigma_{\geq i} \models \beta$ and for each j, if $0 \leq j < i$ then $M, \sigma_{\geq j} \models \alpha$

Abbreviations:

Classical: $\top \equiv p \vee \neg p$, $\bot \equiv \neg\top$, $\alpha \vee \beta \equiv \neg(\neg\alpha \wedge \neg\beta)$, $\alpha \to \beta \equiv \neg\alpha \vee \beta$, $\alpha \leftrightarrow \beta \equiv (\alpha \to \beta) \wedge (\beta \to \alpha)$.
Read: truth, falsity, or(disjunction), implication, iff(equivalence).

Temporal: $F\alpha \equiv (\top U \alpha)$, $G\alpha \equiv \neg F(\neg\alpha)$.
Read: eventually, always.

Example Properties:

$G\neg(p \wedge q)$
$G(p \to (Xq \vee XXq \vee XXXq))$
$G(p \to Fq)$
$G(Gp \to Fq)$
$GFp \wedge GFq$
$G(t \to Gt)$
$Fp \to ((\neg p)U(q \wedge \neg p))$
$FGp \vee GFq$

LTL satisfiability

We start a detailed look at an algorithm to decide the satisfiability of LTL formulas.

We want to invent an algorithm which solves the LTL-SAT problem. Input should be a formula from LTL. Output is "yes" or "no" depending on whether the formula is satisfiable or not.

Satisfiability:

A formula α is satisfiable iff there is some structure (S, R, g) with some fullpath σ through it such that $(S, R, g), \sigma \models \alpha$.

Eg, $\top$, p, Fp, $p \wedge Xp \wedge F\neg p$, Gp are each satisfiable.

Eg, $\bot$, $p \wedge \neg p$, $Fp \wedge G\neg p$, $p \wedge G(p \rightarrow Xp) \wedge F\neg p$ are each not satisfiable.

Can we invent an algorithm for deciding whether an input formula (of LTL) is satisfiable or not?

Build a model:

To test satisfiability of a formula, what about trying to build a model of it?
Eg, suppose that we ask about the satisfiability of $\neg p \wedge X\neg p \wedge (qUp)$

Let's make $\neg p \wedge X\neg p \wedge (qUp)$ true at s_0.

By propositional reasoning we need to make $\neg p$, $X\neg p$ and qUp true at s_0 as well.

Build a model of $\neg p \wedge X\neg p \wedge (qUp)$:

So $\{\neg p \wedge X\neg p \wedge (qUp), \neg p, X\neg p, qUp\}$ are to hold at s_0.

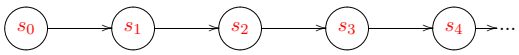
Easy to make $\neg p$ hold there.
Easy to see that we should also make $\neg p$ true at s_1.
But what about qUp?

qUp:

There are two alternative ways to make $\alpha U \beta$ true at a state s.
You can make β true there.
OR
You can make α true there and make $\alpha U \beta$ true at a next state.

In our case, with *qUp* we can not do the former in s_0 so we need to make q true at s_0 and postpone *qUp* until s_1.
And again at s_1 we have to make q true there and postpone *qUp* until s_2.
At s_2 we can use the first case.
Thus we show the formula is satisfiable and we have built a model.

From tableau labels to labelling:



$\{\neg p \wedge X \neg p \wedge (qUp), \neg p, X \neg p, qUp, q\}$ are to hold at s_0.
$\{\neg p, qUp, q\}$ are to hold at s_1.
$\{qUp, p\}$ are to hold at s_2.

Answer: $\{q\}, \{q\}, \{p\}, \{\}, \{\}, ...$

Can this be generalised?:

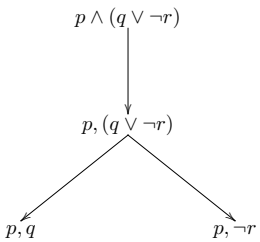
Labelling nodes with formulas is good. Starting from s_0 and working forwards in time is good.

However, some problems:
How to deal with choices (that are not immediately obvious).
What if we need to go on forever building the model?
What if we go on forever making something that is not going to be a model?

The following is my newish tree-shaped tableau for LTL. It builds on work by Sistla and Clarke (1985) and is influenced by LTL tableaux by Wolper (1982) and Schwendimann (1998). There's a GANDALF paper Reynolds (2016) and the idea is also described with an implementation in an IJCAI 2016 paper (BGM 2016).

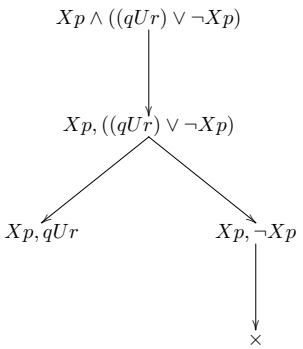
Reminder of Tableau for classical propositional logic:

Possibilities branch into a tree as we work down the page ...



Same rules for LTL:

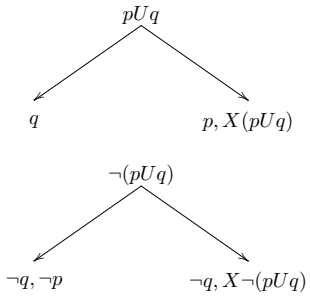
Same things can happen for LTL within a state ...



Rules:

[EMP]: If a node is labelled $\{\}$ then this node can be *ticked*.
[X]: If a node is labelled Γ with some α and $\neg\alpha$ in Γ then this node can be *crossed*.
[DNEG]: If a node is labelled $\Gamma \uplus \{\neg\neg\alpha\}$ then this node can have one child labelled $\Gamma \cup \{\alpha\}$).
[CON]: If a node is labelled $\Gamma \uplus \{\alpha \wedge \beta\}$ in Γ then this node can have one child labelled $\Gamma \cup \{\alpha, \beta\}$.
[DIS]: If a node is labelled $\Gamma \uplus \{\neg(\alpha \wedge \beta)\}$ in Γ then this node can have two children labelled $\Gamma \cup \{\neg\alpha\}$ and $\Gamma \cup \{\neg\beta\}$ respectively.
(plus similar for other abbreviations and their negations)

Until also gives us choices:



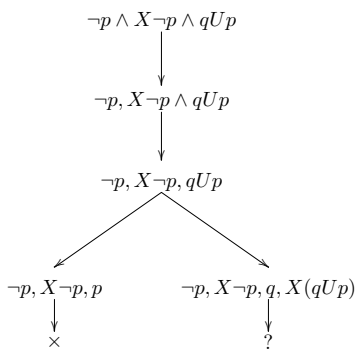
Rules for Until:

[UNT]: If a node is labelled $\Gamma \cup \{\alpha U \beta\}$ in Γ then this node can have two children labelled $\Gamma \cup \{\alpha, X(\alpha U \beta)\}$ and $\Gamma \cup \{\beta\}$.

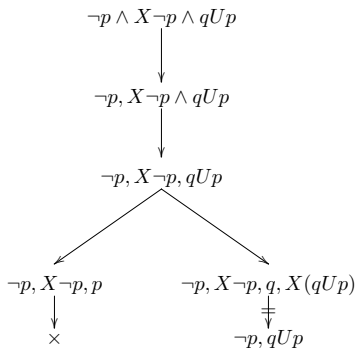
[NUN]: If a node is labelled $\Gamma \cup \{\neg(\alpha U \beta)\}$ in Γ then this node can have two children labelled $\Gamma \cup \{\neg\beta, X\neg(\alpha U \beta)\}$ and $\Gamma \cup \{\neg\alpha, \neg\beta\}$.

(plus similar for F and G.)

But what to do when we want to move forwards in time?:



Introduce a new type of step:



Step Children:

If none of the above (static) rules are applicable to a node then we say that the node is *propositionally complete* and then, and only then, is the following rule applicable.

[TRANSITION]: The node, labelled by propositionally complete Γ say, can have one child, called a *step-child*, whose label Δ is defined as follows.
$\Delta = \{\alpha | X\alpha \in \Gamma\} \cup \{\neg\alpha | \neg X\alpha \in \Gamma\}$.

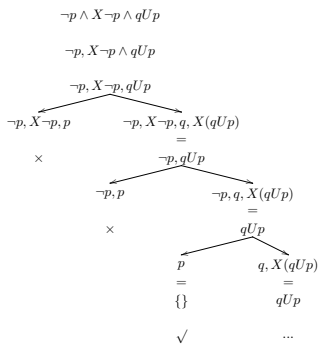
Note that Δ may be empty. After a step rule the try to use the static rules again.

Exercise:

Can now do the whole $\neg p \wedge X\neg p \wedge (qUp)$ example.

Try it.

Exercise answer: $\neg p \wedge X\neg p \wedge (qUp)$ example.



Infinite behaviour:

Still some work to do.

We will try these examples in the next few slides ...
Gp
$G(p \wedge q) \wedge F\neg p$
$p \wedge G(p \leftrightarrow X\neg p) \wedge G(q \rightarrow \neg p) \wedge GF\neg q \wedge GF\neg p$
$p \wedge G(p \rightarrow Xp) \wedge F\neg p$

Example: Gp

Gp gives rise to a very repetitive infinite tableau.

Gp

p, XGp
=
Gp

p, XGp
=
Gp

p, XGp

...

Notice that the infinite fullpath that it suggests is a model for Gp as would a fullpath just consisting of the one state with a self-loop (a transition from itself to itself).

Example: $G(p \wedge q) \wedge F\neg p$

But $G(p \wedge q) \wedge F\neg p$ shows that we can not just accept infinite loops as demonstrating satisfiability....

$$G(p \wedge q) \wedge F\neg p$$

$$G(p \wedge q), F\neg p$$

$$p \wedge q, XG(p \wedge q), F\neg p$$

$$p, q, XG(p \wedge q), F\neg p$$

$$p, q, XG(p \wedge q), \neg p \quad \times \qquad p, q, XG(p \wedge q), XF\neg p$$

$$=$$

$$G(p \wedge q), F\neg p$$

$$\dots$$

$$XG(p \wedge q), XF\neg p$$

$$=$$

$$G(p \wedge q), F\neg p$$

$$\dots$$

$$XG(p \wedge q), XF\neg p$$

$G(p \wedge q) \wedge F\neg p$ continued

Notice that the infinite fullpath that the tableau suggests is this time not a model for $G(p \wedge q) \wedge F\neg p$.

Constant repeating of p, q being made true does not satisfy the conjunct $F\neg p$.

We have postponed the *eventuality* forever.

This is not acceptable.

Eventualities:

An *eventuality* is just a formula of the form $\alpha U\beta$.

(This includes $F\gamma \equiv \top U\gamma$).

If $\alpha U\beta$ appears in the tableau label of a node u then we want β to appear in the label of some later (or equal node) v. In that case we say that the eventuality is *satisfied* by v.

Eventualities are eventually satisfied in any (actual) model of a formula: by the semantics of until.

Loops:

If a label is repeated along a branch and all eventualities are satisfied in between then we can build a model by looping states. In fact, the ancestor can have a superset and it will work.

[LOOP]: If a node n has a proper ancestor (i.e. not itself) m such that $\Gamma(m) \supseteq \Gamma(n)$, m has a STEP-child, and all eventualities in $\Gamma(m)$ are satisfied by labels between m and n (including m itself) then n can be ticked.

Nice example to try:

$p \wedge G(p \leftrightarrow X\neg p) \wedge G(q \rightarrow \neg p) \wedge G(r \rightarrow \neg p) \wedge G(q \rightarrow \neg r) \wedge GFq \wedge GFr$

Are we done yet?

No.

Examples like $G(p \wedge q) \wedge F\neg p$ may have branches that go on forever without a tick. We need to stop and fail branches so that we can answer "no" correctly and terminate and so that we do not get distracted when another branch may be successful. In fact, no infinite branches should be allowed.

Try also: $p \wedge G(p \rightarrow Xp) \wedge F\neg p$

Can't we see that these infinite branches are just getting repetitive without making a model?

Closure set:

As we construct the tableau model we only need to record, in the labels, which formulas we want to be true at that time from a finite set of interesting formulas.
The *closure set* for a formula ϕ is as follows:

$$\{\psi, \neg\psi | \psi \leq \phi\} \cup \{X(\alpha U \beta), \neg X(\alpha U \beta) | \alpha U \beta \leq \phi\}$$

(Where $\psi \leq \phi$ means that ψ is a subformula of ϕ.)

Size of closure set is $\leq 4n$ where n is the length of the initial formula.

This shows that only formulas from a finite set will appear in labels.

...and only $\leq 2^{4n}$ possible labels.

Don't go on further than you need to:

This gives us the idea of *useless* intervals on branches in the tableau.

If a node at the end of a branch (of a partially complete tableau) has a label which has appeared already twice above, and between the second and third appearance there are no new eventualities satisfied then that whole interval of states has been useless!

The **REP**etition rule:

[REP]:
(later called PRUNE rule)
Suppose that $u = u_0, u_1, ..., u_{j-1}, u_j = v, u_{j+1}, ..., u_k = w$ is a sequence of consecutive descendants in order.
Suppose that $\Gamma(u) = \Gamma(v) = \Gamma(w)$, u and v have STEP-children and w is propositionally complete.
Suppose also that for all eventualities $\alpha U \beta \in \Gamma(u)$, if β is satisfied between v and w then β is satisfied between u and v anyway.
Then w can be crossed.

(We assume that there are some unsatisfied eventualities from $\Gamma(u)$ left. Otherwise you should have used the LOOP rule earlier to tick the branch.)

Examples:

Now try $G(p \wedge q) \wedge F\neg p$ and $p \wedge G(p \rightarrow Xp) \wedge F\neg p$.

LTL Tableau Summary:

Is a finite tree of nodes labelled by subsets of the closure set of ϕ such that:

- the root is labelled with $\{\phi\}$
- the labels of children of each node are according to one of the tableau rules

Successful if some leaf is ticked.

Failed if all leaves are crossed.

(Not really a proper description of an algorithm but we will see that the further details of which formula to consider at each step in building the tableau are unimportant).

Proof of Correctness:

This will consist of three parts.

Proof of soundness. If a formula has a successful tableau then it has a model.

Proof of completeness: If a formula has a model then building a tableau will be successful.

Proof of termination. Show that the tableau building algorithm will always terminate.

Part One:, the Proof of Termination:

(Sketch only)

Any reasonable tableau search algorithm will always terminate because there can be no infinitely long branches.

We know this because the REP rule will cross any that go on too long.

Thus there will either be at least one tick or all crosses.

Termination is also why we require that other rules are not used repeatedly in between STEP rules.

Part Two: Proof of Soundness:

(Overview)
Use a successful tableau to make a model of the formula, thus showing that it is satisfiable.

Use a successful branch. Each STEP tells us that we are moving from one state to the next.

Within a particular state we can make all the formulas listed true there (as evaluated along the rest of the fullpath). Atomic propositions listed tell us that they are true at that state.

An induction deals with most of the rest of the formulas.

Eventualities either get satisfied and disappear in a terminating branch or have to be satisfied if the branch is ticked by the LOOP rule.

PART 3: Proof of Completeness:

We have to show that if a formula has a model then it has a successful tableau.

This time we will use the model to find the tableau.

Proof of Completeness:

The basic idea is to use a model (of the satisfiable formula) to show that *in any tableau* there will be a branch (i.e. a leaf) with a tick.

A weaker result is to show that there is some tableau with a leaf with a tick.

Such a weaker result may actually be ok to establish correctness and complexity of the tableau technique.
However, it raises questions about whether a "no" answer from a tableau is correct and it does not give clear guidance for the implementer.

Completeness Proof:

Suppose that ϕ is a satisfiable formula of LTL.
It will have a model. Choose one, say $(S, R, g), \sigma \models \phi$. In what follows we (use standard practice when the model is fixed and) write $\sigma_{\geq i} \models \alpha$ when we mean $(S, R, g), \sigma_{\geq i} \models \alpha$.
Also, build a tableau T for ϕ in any manner as long as the rules are followed. Let $\Gamma(x)$ be the formula set label on the node x in T.
We will show that T has a ticked leaf.
To do this we first construct a sequence $x_0, x_1, x_2, \ldots$ of nodes, with x_0 being the root. This sequence may terminate at a tick (and then we have succeeded) or it may hypothetically go on forever (and more on that later).
In general the sequence will head downwards from a parent to a child node but occasionally it may jump back up to an ancestor.

Invariant

As we go we will also make sure that each node x_i is associated with a state $\sigma_{j(i)}$ in S.
We will ensure that for each i, for each $\alpha \in \Gamma(x_i)$, $\sigma_{\geq j(i)} \models \alpha$.
Start by putting $j(0) = 0$ when x_0 is the tableau root node.
Note that the only formula in $\Gamma(x_0)$ is ϕ and that $\sigma_{\geq 0} \models \phi$.
Good start.

Now suppose that we have identified the x sequence up until x_i.

Consider the rule that is used in T to extend a tableau branch from x_i to some children.

[EMP] If $\Gamma(x_i) = \{\}$ then we are done. T is a successful tableau as required.

[X] Consider if it is possible for the branch to stop at x_i with a cross because of a contradiction. So there is some α with α and $\neg\alpha$ in $\Gamma(x_i)$. But this can not happen as then $\sigma_{\geq j(i)} \models \alpha$ and $\sigma_{\geq j(i)} \models \neg\alpha$.

[DNEG]

So $\neg\neg\alpha$ is in $\Gamma(x_i)$ and there is one child, which we will make $x(i+1)$ and we will put $j(i+1) = j(i)$. Because $\sigma_{\geq j(i)} \models \neg\neg\alpha$ we also have $\sigma_{\geq j(i+1)} \models \alpha$. Also for every other $\beta \in \Gamma(x_{i+1}) = \Gamma(x_i) \cup \{\alpha\}$, we still have $\sigma_{\geq j(i+1)} \models \beta$. So we have the invariant holding.

(CON, DIS etc are similar)

[UNT]

So $\alpha U \beta$ is in $\Gamma(x_i)$ and there are two children. One y is labelled $\Gamma(x_i) \cup \{\beta\}$ and the other, z, is labelled $\Gamma(x_i) \cup \{\alpha, X(\alpha U \beta)\}$.
We know $\sigma_{\geq j(i)} \models \alpha U \beta$. Thus, there is some $k \geq j(i)$ such that $\sigma_{\geq k} \models \beta$ and for all l, if $0 \leq l < k$ then $\sigma_{\geq j(i)+l} \models \alpha$.
If $\sigma_{\geq j(i)} \models \beta$ then we can choose $k = j(i)$ (even if other choices as possible) and otherwise choose any such $k > j(i)$. Again there are two cases, either $k = j(i)$ or $k > j(i)$.
In the first case, when $\sigma_{\geq j(i)} \models \beta$, we put $x_{i+1} = y$ and otherwise we will make $x_{i+1} = z$. In either case put $j(i+1) = j(i)$.
Let us check the invariant. Consider the first case.
We know that we have $\sigma_{\geq j(i+1)} \models \beta$.
In the second case, we know that we have $\sigma_{\geq j(i+1)} \models \alpha$ and $\sigma_{\geq j(i+1)+1} \models \alpha U \beta$. Thus $\sigma_{\geq j(i+1)} \models X(\alpha U \beta)$.
Also, in either case, for every other $\gamma \in \Gamma(x_{i+1})$ we still have $\sigma_{\geq j(i+1)} \models \gamma$.
So we have the invariant holding.

[STEP]

So $\Gamma(x_i)$ is propositionally complete and there is one child, which we will make x_{i+1} and we will put $j(i+1) = j(i) + 1$.
Consider a formula
$\gamma \in \Gamma(x_{i+1}) = \{\alpha | X\alpha \in \Gamma(x_i)\} \cup \{\neg\alpha | \neg X\alpha \in \Gamma(x_i)\}$.
CASE 1: Say that $X\gamma \in \Gamma(x_i)$. Thus, by the invariant, $\sigma_{\geq j(i)} \models X\gamma$. Hence, $\sigma_{\geq j(i)+1} \models \gamma$. But this is just $\sigma_{\geq j(i+1)} \models \gamma$ as required.
CASE 2: Say that $\gamma = \neg\delta$ and $\neg X\delta \in \Gamma(x_i)$. Thus, by the invariant, $\sigma_{\geq j(i)} \models \neg X\delta$. Hence, $\sigma_{\geq j(i)+1} \not\models \delta$. But this is just $\sigma_{\geq j(i+1)} \models \gamma$ as required.
So we have the invariant holding.

[LOOP]

If, in T, the node x_i is a leaf just getting a tick via the LOOP rule then we are done.

T is a successful tableau as required.

[REP]

Now the tricky case.
Suppose that x_i is a node which gets a cross in T via the REP rule.
So there is a sequence
$u = x_h, x_{h+1}, ..., x_{h+a} = v, x_{h+a+1}, ..., x_{h+a+b} = x_i = w$ such that $\Gamma(u) = \Gamma(v) = \Gamma(w)$ and no extra eventualities of u are satisfied between v and w that were not already satisfied between u and v.
What we do now is to choose some such u, v and w, there may be more than one triple, and proceed with the construction as if x_i was v instead of w.
That is we move on to look at the rule (as above, and the rule will not be REP) that is used to get from v to its children.
However, we use $\sigma_{\geq i}$ to make the choice of child x_{i+1} (if there is a choice).
All the reasoning above works because $\Gamma(v) = \Gamma(x_i)$ and so the invariant holds for v instead of x_i as well.
Thus we keep going.

The above construction may end finitely with us finding a ticked leaf and succeeding.
However, at least in theory, it may seem possible that the construction keeps going forever even though the tableau will be finite.
The rest of the proof is to show that this actually can not happen.
The construction can not go on forever. It must stop and the only way that we have shown that that can happen is by finding a tick.

We suppose for contradiction that the construction does go on forever.
Thus, because there are only a finite number of nodes in the tableau, we must meet the REP rule and jump back up the tableau infinitely often.
The proof by contradictions shows that as eventualties are witnessed the path of x_is can never jump back up higher again.
Have a read!

You might like to read up on other approaches to deciding satisfiability in LTL.
Wolper [Wol85]
Schwendiman [Sch98]
Sistla and Clarke [SC85]
Schmitt and Goubault-Larrecq [SGL97]
Automata-based approaches
Resolution-based approaches.
Others, e.g. Small model theorems with axiom systems.
Implementation races, and benchmarking: [GKS10]

What about complexity?

Deciding LTL satisfiability is in PSPACE [SC85].
In fact our tableau approach can be used to show that.
Easiest to use the tableau search to directly show that the problem is in NPSPACE and then Savitch tells us also in PSPACE.
We are allowed to guess the right choices and need to show that "yes" answers can be guessed and checked using memory space bounded by a polynomial in the size of the input. To do this, just guess the right branch and remember at each step: the label here, the previous label (to check you do the STEP rule properly), the label back at an ancestor that you want to LOOP to, and the eventualities that you still have to satisfy from that. (Size of memory usage just linear in size of input even though branch length may be exponential).

Questions

And that's all!

Thank you.

Any questions.

Extended Proof of Completeness:

The following slides are only to be used if there is extra time or questions about the completness proof.

Proof of Completeness:

The basic idea is to use a model (of the satisfiable formula) to show that *in any tableau* there will be a branch (i.e. a leaf) with a tick.

A weaker result is to show that there is some tableau with a leaf with a tick.

Such a weaker result may actually be ok to establish correctness and complexity of the tableau technique.
However, it raises questions about whether a "no" answer from a tableau is correct and it does not give clear guidance for the implementer.

Completeness Proof:

Suppose that ϕ is a satisfiable formula of LTL.
It will have a model. Choose one, say $(S, R, g), \sigma \models \phi$. In what follows we (use standard practice when the model is fixed and) write $\sigma_{\geq i} \models \alpha$ when we mean $(S, R, g), \sigma_{\geq i} \models \alpha$.
Also, build a tableau T for ϕ in any manner as long as the rules are followed. Let $\Gamma(x)$ be the formula set label on the node x in T.
We will show that T has a ticked leaf.
To do this we first construct a sequence $x_0, x_1, x_2, \ldots$ of nodes, with x_0 being the root. This sequence may terminate at a tick (and then we have succeeded) or it may hypothetically go on forever (and more on that later).
In general the sequence will head downwards from a parent to a child node but occasionally it may jump back up to an ancestor.

Invariant

As we go we will also make sure that each node x_i is associated with a state $\sigma_{j(i)}$ in S.
We will ensure that for each i, for each $\alpha \in \Gamma(x_i)$, $\sigma_{\geq j(i)} \models \alpha$.
Start by putting $j(0) = 0$ when x_0 is the tableau root node.
Note that the only formula in $\Gamma(x_0)$ is ϕ and that $\sigma_{\geq 0} \models \phi$.
Good start.

Now suppose that we have identified the x sequence up until x_i.

Consider the rule that is used in T to extend a tableau branch from x_i to some children.

[EMP] If $\Gamma(x_i) = \{\}$ then we are done. T is a successful tableau as required.

[X] Consider if it is possible for the branch to stop at x_i with a cross because of a contradiction. So there is some α with α and $\neg\alpha$ in $\Gamma(x_i)$. But this can not happen as then $\sigma_{\geq j(i)} \models \alpha$ and $\sigma_{\geq j(i)} \models \neg\alpha$.

[DNEG]

So $\neg\neg\alpha$ is in $\Gamma(x_i)$ and there is one child, which we will make $x(i+1)$ and we will put $j(i+1) = j(i)$. Because $\sigma_{\geq j(i)} \models \neg\neg\alpha$ we also have $\sigma_{\geq j(i+1)} \models \alpha$. Also for every other $\beta \in \Gamma(x_{i+1}) = \Gamma(x_i) \cup \{\alpha\}$, we still have $\sigma_{\geq j(i+1)} \models \beta$. So we have the invariant holding.

[CON]

So $\alpha \wedge \beta$ is in $\Gamma(x_i)$ and there is one child, which we will make $x(i+1)$ and we will put $j(i+1) = j(i)$. Because $\sigma_{\geq j(i)} \models \alpha \wedge \beta$ we also have $\sigma_{\geq j(i+1)} \models \alpha$ and $\sigma_{\geq j(i+1)} \models \beta$. Also for every other $\gamma \in \Gamma(x_{i+1}) = \Gamma(x_i) \cup \{\alpha, \beta\}$, we still have $\sigma_{\geq j(i+1)} \models \gamma$.
So we have the invariant holding.

[DIS]

So $\neg(\alpha \wedge \beta)$ is in $\Gamma(x_i)$ and there are two children. One y is labelled $\Gamma(x_i) \cup \{\neg\alpha\}$ and the other, z, is labelled $\Gamma(x_i) \cup \{\neg\beta\}$.
We know $\sigma_{\geq j(i)} \models \neg(\alpha \wedge \beta)$. Thus, $\sigma_{\geq j(i)} \not\models \alpha \wedge \beta$ and it is not the case that both $\sigma_{\geq j(i)} \models \alpha$ and $\sigma_{\geq j(i)} \models \beta$. So either $\sigma_{\geq j(i)} \models \neg\alpha$ or $\sigma_{\geq j(i)} \models \neg\beta$.
If the former, i.e. that $\sigma_{\geq j(i)} \models \neg\alpha$ we will make $x_{i+1} = y$ and otherwise we will make $x_{i+1} = z$. In either case put $j(i+1) = j(i)$.
Let us check the invariant. Consider the first case. The other is exactly analogous.
We already know that we have $\sigma_{\geq j(i+1)} \models \neg\alpha$. Also for every other $\gamma \in \Gamma(x_{i+1}) = \Gamma(y) = \Gamma(x_i) \cup \{\neg\alpha\}$, we still have $\sigma_{\geq j(i+1)} \models \gamma$.
So we have the invariant holding.

[UNT]

So $\alpha U \beta$ is in $\Gamma(x_i)$ and there are two children. One y is labelled $\Gamma(x_i) \cup \{\beta\}$ and the other, z, is labelled $\Gamma(x_i) \cup \{\alpha, X(\alpha U \beta)\}$.
We know $\sigma_{\geq j(i)} \models \alpha U \beta$. Thus, there is some $k \geq j(i)$ such that $\sigma_{\geq k} \models \beta$ and for all l, if $0 \leq l < k$ then $\sigma_{\geq j(i)+l} \models \alpha$.
If $\sigma_{\geq j(i)} \models \beta$ then we can choose $k = j(i)$ (even if other choices as possible) and otherwise choose any such $k > j(i)$. Again there are two cases, either $k = j(i)$ or $k > j(i)$.
In the first case, when $\sigma_{\geq j(i)} \models \beta$, we put $x_{i+1} = y$ and otherwise we will make $x_{i+1} = z$. In either case put $j(i+1) = j(i)$.
Let us check the invariant. Consider the first case.
We know that we have $\sigma_{\geq j(i+1)} \models \beta$.
In the second case, we know that we have $\sigma_{\geq j(i+1)} \models \alpha$ and $\sigma_{\geq j(i+1)+1} \models \alpha U \beta$. Thus $\sigma_{\geq j(i+1)} \models X(\alpha U \beta)$.
Also, in either case, for every other $\gamma \in \Gamma(x_{i+1})$ we still have $\sigma_{\geq j(i+1)} \models \gamma$.
So we have the invariant holding.

[NUN]

So $\neg(\alpha U \beta)$ is in $\Gamma(x_i)$ and there are two children. One y is labelled $\Gamma(x_i) \cup \{\neg\alpha, \neg\beta\}$ and the other, z, is labelled $\Gamma(x_i) \cup \{\neg\beta, X\neg(\alpha U \beta)\}$.
We know $\sigma_{\geq j(i)} \models \neg(\alpha U \beta)$.
So for sure $\sigma_{\geq j(i)} \models \neg\beta$.
Furthermore, possibly $\sigma_{\geq j(i)} \models \neg\alpha$ as well, but otherwise if $\sigma_{\geq j(i)} \models \alpha$ then we can show that we can not have $\sigma_{\geq j(i)+1} \models \alpha U \beta$. Suppose for contradiction that $\sigma_{\geq j(i)} \models \alpha$ and $\sigma_{\geq j(i)+1} \models \alpha U \beta$.

[NUN] continued

We have supposed that $\sigma_{\geq j(i)} \models \alpha$ and $\sigma_{\geq j(i)+1} \models \alpha U \beta$ while also $\sigma_{\geq j(i)} \not\models \alpha U \beta$.
Then there is some $k \geq 0$ such that $\sigma_{\geq j(i)+1+k} \models \beta$ and for all l, if $0 \leq l < k$ then $\sigma_{\geq j(i)+1+l} \models \alpha$.
But then for all l, if $0 \leq l < k+1$ then $\sigma_{\geq j(i)+l} \models \alpha$.
Thus $\sigma_{\geq j(i)} \models \alpha U \beta$.
Contradiction.

[NUN] continued

So we can conclude that there are two cases when the NUN rule is used.
CASE 1: $\sigma_{\geq j(i)} \models \neg\beta$ and $\sigma_{\geq j(i)} \models \neg\alpha$.
CASE 2: $\sigma_{\geq j(i)} \models \neg\beta$ and $\sigma_{\geq j(i)+1} \models \neg(\alpha U \beta)$.
In the first case, when $\sigma_{\geq j(i)} \models \neg\beta$, we put $x_{i+1} = y$ and otherwise we will make $x_{i+1} = z$. In either case put $j(i+1) = j(i)$.
Let us check the invariant. In both cases we know that we have $\sigma_{\geq j(i+1)} \models \neg\beta$.
Now consider the first case. We also have $\sigma_{\geq j(i)} \models \neg\alpha$.
In the second case, we know that we have $\sigma_{\geq j(i)+1} \models \neg(\alpha U \beta)$.
Thus $\sigma_{\geq j(i+1)} \models X\neg(\alpha U \beta)$.
Also, in either case, for every other $\gamma \in \Gamma(x_{i+1})$ we still have $\sigma_{\geq j(i+1)} \models \gamma$.
So we have the invariant holding.

[STEP]

So $\Gamma(x_i)$ is propositionally complete and there is one child, which we will make x_{i+1} and we will put $j(i+1) = j(i)+1$.
Consider a formula
$\gamma \in \Gamma(x_{i+1}) = \{\alpha | X\alpha \in \Gamma(x_i)\} \cup \{\neg\alpha | \neg X\alpha \in \Gamma(x_i)\}$.
CASE 1: Say that $X\gamma \in \Gamma(x_i)$. Thus, by the invariant, $\sigma_{\geq j(i)} \models X\gamma$. Hence, $\sigma_{\geq j(i)+1} \models \gamma$. But this is just $\sigma_{\geq j(i+1)} \models \gamma$ as required.
CASE 2: Say that $\gamma = \neg\delta$ and $\neg X\delta \in \Gamma(x_i)$. Thus, by the invariant, $\sigma_{\geq j(i)} \models \neg X\delta$. Hence, $\sigma_{\geq j(i)+1} \not\models \delta$. But this is just $\sigma_{\geq j(i+1)} \models \gamma$ as required.
So we have the invariant holding.

[LOOP]

If, in T, the node x_i is a leaf just getting a tick via the LOOP rule then we are done.

T is a successful tableau as required.

[REP]

Now the tricky case.
Suppose that x_i is a node which gets a cross in T via the REP rule. So there is a sequence
$u = x_h, x_{h+1}, ..., x_{h+a} = v, x_{h+a+1}, ..., x_{h+a+b} = x_i = w$ such that $\Gamma(u) = \Gamma(v) = \Gamma(w)$ and no extra eventualities of u are satisfied between v and w that were not already satisfied between u and v.
What we do now is to choose some such u, v and w, there may be more than one triple, and proceed with the construction as if x_i was v instead of w.
That is we move on to look at the rule (as above, and the rule will not be REP) that is used to get from v to its children.
However, we use $\sigma_{\geq i}$ to make the choice of child x_{i+1} (if there is a choice).
All the reasoning above works because $\Gamma(v) = \Gamma(x_i)$ and so the invariant holds for v instead of x_i as well.
Thus we keep going.

The above construction may end finitely with us finding a ticked leaf and succeeding.
However, at least in theory, it may seem possible that the construction keeps going forever even though the tableau will be finite.
The rest of the proof is to show that this actually can not happen. The construction can not go on forever. It must stop and the only way that we have shown that that can happen is by finding a tick.

Suppose for contradiction that the construction does go on forever.
Thus, because there are only a finite number of nodes in the tableau, we must meet the REP rule and jump back up the tableau infinitely often.
When we do apply the REP rule with triple (u, v, w) call that a jump triple.
There are only a finite number of jump triples so there must be some that cause us to jump infinitely often.
Say that (u_0, v_0, w_0) is one such.
We can choose u_0 so that for no other infinite jump triple (u_1, v_1, w_1) do we have u_1 being a proper ancestor of u_0.

As we proceed through the construction of $x_0, x_1, ..$ and see a jump every so often, eventually all the jump triples who only cause a jump a finite number of times stop causing jumps.
After that time, (u_0, v_0, w_0) will still cause a jump every so often.
Thus after that time u_0 will never appear again as the x_i that we choose and all the x_is that we choose will be descendants of u_0.
This is because we will never jump up to u_0 or above it (closer to the root).
Say that x_N is the very last x_i that is equal to u_0.

Now consider any $\alpha U \beta$ that appears in $\Gamma(u_0)$. (There must be at least one eventuality in $\Gamma(u_0)$ as it is used to apply rule REP).
A simple induction shows that $\alpha U \beta$ will appear in every $\Gamma(x_i)$ from $i = N$ up until at least when β appears in some $\Gamma(x_i)$ after that (if that ever happens).
This is because if $\alpha U \beta$ is in $\Gamma(x_i)$ and β is not there and does not get put there then $X(\alpha U \beta)$ will also be put in before the next temporal step rule.
Each temporal step rule will thus put $\alpha U \beta$ into the new label.

Now $j(i)$ just increases by 0 or 1 with each increment of i, We also know that $\sigma_{\geq j(i)} \models \alpha U \beta$ from $i = N$ onwards until (and if) β gets put in $\Gamma(x_i)$.
Since σ is a fullpath we will eventually get to some i with $\sigma_{\geq j(i)} \models \beta$.
In that case our construction makes us put β in the label.
Thus we do eventually get to some $i \geq N$ with $\beta \in \Gamma(x_i)$.
Let N_β be the first such $i \geq N$.
Note that all the nodes between u_0 and x_{N_β} in the tableau also appear as x_i for $N < i < N_\beta$ so that they all have $\alpha U \beta$ and not β in their labels $\Gamma(x_i)$.

Now let us consider if we ever jump up above x_{N_β} at any step of our construction (after N_β).
In that case there would be tableau nodes u, v and w arranged according to the jump situation.
Since u is not above u_0 and v is above x_{N_β}, we must have $\Gamma(u) = \Gamma(v)$ with $\alpha U \beta$ in them and not satisfied in between.
But w will be below x_{N_β} at the first such jump, meaning that β is satisfied between v and w.
Contradiction.

The above reasoning applies to all eventualities in $\Gamma(u_0)$.
Thus, after they are all satisfied, the construction x_i does not jump up above any of them.
When the next supposed jump involving u_0 with some v and w happens after that it is clear that all of the eventualities in $\Gamma(u_0)$ are satisfied above v.
This is a contradiction to such a jump ever happening.
Thus we can conclude that there are not an infinite number of jumps after all.
The construction must finish with a tick.
END of completeness proof.

Rajeev Alur and Thomas A. Henzinger.
Real-time logics: Complexity and expressiveness.
Inf. Comput., 104(1):35–77, 1993.

M. Bertello, N. Gigante, A. Montanari and M. Reynolds.
Leviathan: A New LTL Satisfiability Checking Tool Based on a One-Pass Tree-Shaped Tableau.
In *Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, Proceedings*, pages 950–956. IJCAI/AAAI Press, 2016.

Burgess, J. P. Axioms for Tense Logic I: "Since" and "Until", Notre Dame J. Formal Logic, 23(2):367–374, 1982.

J. P. Burgess and Y. Gurevich.
The decision problem for linear temporal logic.
Notre Dame J. Formal Logic, 26(2):115–128, 1985.

E. Emerson and E. C. Clarke.

Using branching time temporal logic to synthesise synchronisation skeletons.
Sci. of Computer Programming, 2, 1982.

E. Emerson and A. Sistla.
Deciding branching time logic.
In *Proc. 16th ACM Symposium on Theory of Computing*, 1984.

M. Fischer and R. Ladner.
Propositional dynamic logic of regular programs.
J. Computer and System Sciences, 18:194–211, 1979.

Oliver Friedmann, Markus Latte, and Martin Lange.
A decision procedure for CTL* based on tableaux and automata.
In *IJCAR'10*, pages 331–345, 2010.

R. Goré.
Tableau methods for modal and temporal logics.

In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, 1999.

H. Kamp.
Tense logic and the theory of linear order.
PhD thesis, University of California, Los Angeles, 1968.

Y. Kesten, Z. Manna, and A. Pnueli.
Temporal verification of simulation and refinement.
In *A decade of concurrency: reflections and perspectives: REX school/symposium, Noordwijkerhout, the Netherlands, June 1–4, 1993*, pages 273–346. Springer–Verlag, 1994.

A. Pnueli.
The temporal logic of programs.
In *Proceedings of the Eighteenth Symposium on Foundations of Computer Science*, pages 46–57, 1977.
Providence, RI.

V. R. Pratt.
Models of program logics.
In *Proc. 20th IEEE. Symposium on Foundations of Computer Science, San Juan*, pages 115–122, 1979.

Mark Reynolds.
A tableau-based decision procedure for CTL*.
Journal of Formal Aspects of Computing, pages 1–41, August 2011.

A. Sistla and E. Clarke.
Complexity of propositional linear temporal logics.
J. ACM, 32:733–749, 1985.

Valentin Goranko, Angelo Kyrilov, and Dmitry Shkatov.
Tableau tool for testing satisfiability in ltl: Implementation and experimental analysis.
Electronic Notes in Theoretical Computer Science, 262(0):113 – 125, 2010.

Proceedings of the 6th Workshop on Methods for Modalities (M4M-6 2009).

M. Reynolds.
A New Rule for LTL Tableaux.
In *Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016, Proceedings*, pages 287–301. EPTCS, 2016.

A. Sistla and E. Clarke.
Complexity of propositional linear temporal logics.
J. ACM, 32:733–749, 1985.

S. Schwendimann.
A new one-pass tableau calculus for PLTL.
In Harrie C. M. de Swart, editor, *Proceedings of International Conference, TABLEAUX 1998, Oisterwijk*, LNAI 1397, pages 277–291. Springer, 1998.

P. Schmitt and J. Goubault-Larrecq.
A tableau system for linear-time temporal logic.
In *TACAS 1997*, pages 130–144, 1997.

M. Vardi and L. Stockmeyer.
Improved upper and lower bounds for modal logics of programs.
In *17th ACM Symp. on Theory of Computing, Proceedings*, pages 240–251. ACM, 1985.

P. Wolper.
The tableau method for temporal logic: an overview.
Logique et Analyse, 28:110–111, June–Sept 1985.

Formal Specification and Verification of Post-quantum Cryptographic Protocols with Proof Scores

Kazuhiro Ogata
Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology（JAIST）

2nd Workshop on Logic, Algebra and Category Theory: LAC 2025
Fukuoka, September 29 – October 3, 2025

Outline

- Formal verification of post-quantum (PQ) hybrid OpenPGP, where both PQ cryptographic primitives and classical cryptographic primitives are used
- Formal verification of PQ hybrid SSH, where both PQ cryptographic primitives and classical cryptographic primitives are used

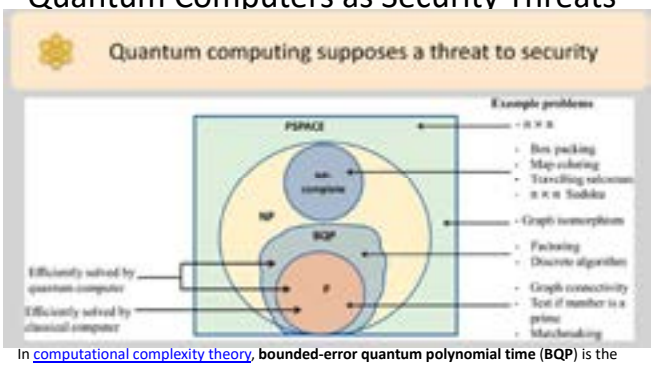
Quantum Computers as Security Threats

- An idea of quantum computers proposed by Faynman, etc. early-80's
- Google, etc. have been spending many resources (money, humans, etc.) toward implementation of large-scale quantum computers
- Shor invented the quantum algorithms that can efficiently solve Integer Factorization and Discrete Logarithm Problem in 1994.

Quantum Computers as Security Threats

- Public-key encryption schemes, such as RSA, currently used will become insecure and unsafe when large-scale quantum computers are available
- Cryptographic primitives, such as KEMs, resistant to quantum computers are actively studied (in the middle of selection of future standard ones by NIST)
- Development of technologies that can be used to guarantee that post-quantum cryptographic protocols are really secure and safe is an urgent research topic

Quantum Computers as Security Threats



In computational complexity theory, **bounded-error quantum polynomial time** (**BQP**) is the class of decision problems solvable by a quantum computer in polynomial time, with an error probability of at most 1/3 for all instances. It is the quantum analogue to the complexity class **BPP**. (from Wikipedia)

Countermeasure

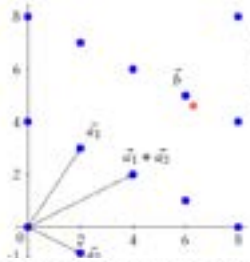


Fig. 2. An illustration of the closest vector problem in 2-dimensional lattice $\mathcal{L}(\vec{a_1}, \vec{a_2})$, where $\vec{a_1} = (2, 1)$ and $\vec{a_2} = (2, -1)$

Definition 1.1. A key encapsulation mechanism (KEM) is a tuple of algorithms (keyGen, Encaps, Decaps) along with a finite key space $\mathcal{K}$:

- $\text{keyGen}() \to (pk, sk)$: A probabilistic key generation algorithm that outputs a public key pk and a secret key sk.
- $\text{Encaps}(pk) \to (c, k)$: A probabilistic encapsulation algorithm that takes as input a public key pk, and outputs an encapsulation (or ciphertext) c and a shared secret $k \in \mathcal{K}$.
- $\text{Decaps}(c, sk) \to k$: A (usually deterministic) decapsulation algorithm that takes as inputs a ciphertext c and a secret key sk, and outputs a shared secret $k \in \mathcal{K}$.

Countermeasure

Key Encapsulation Mechanism (KEM)

- A KEM is a tuple of algorithms (*keygen, encaps, decaps*):

1. *(sk, pk) <- keygen()*: a probabilistic function, outputs a public key *pk* and a secret key *sk*
2. *(K, C) <- encaps(pk)*: a probabilistic function, takes the public *pk*, and outputs a ciphertext *C* and a shared secret key *K*
3. *K <- decaps(sk,C)*: a deterministic function, takes the secret key *sk*, a ciphertext *C*, and outputs the shared secret key *K*

Tools & Techniques Used

- Observational Transition Systems (OTSs) – mathematical model formalizing protocols
- CafeOBJ – proof score-based interactive theorem proving
- CafeInMaude – World's 2nd implementation of CafeOBJ in Maude, equipped with a proof assistant (CiMPA) and a proof generator (CiMPG)
- Invariant Proof Score Generator (IPSG) –automatically generating proof scores for a formal specification nadf a property specification (and lemmas).

Adrian Riesco

Duong Dinh Tran

Formal Specification of PQ OpenPGP

- OpenPGP has been often used to secure emails
- A post-quantum (PQ) version has been proposed, where both classical and post-quantum cryptographic primitives are used, because the implementation of the latter may not be matured enough, while the implementation of the former has been matured.

Formal Specification of PQ OpenPGP

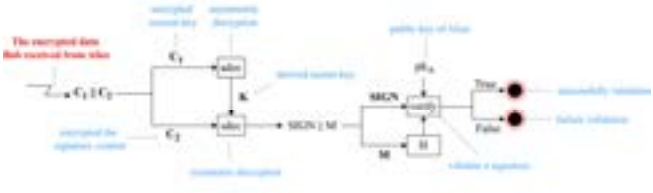
OpenPGP

- OpenPGP is an open standard of PGP (Pretty Good Privacy), the most widely used email/file encryption standard.
- Alice sends the message to Bob:

Formal Specification of PQ OpenPGP

OpenPGP

- When Bob receives the message from Alice:



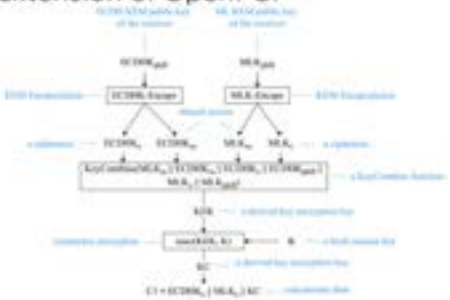
Formal Specification of PQ OpenPGP

- PQ OpenPGP uses:
  1. Elliptic Curve Diffie–Hellman (ECDH) + Module-lattice KEM (ML-KEM) for hybrid key encapsulations
  2. Edwards-curve Digital Signature Algorithm (EdDSA) + Module-lattice DSA (ML-DSA) for hybrid digital signatures

Formal Specification of PQ OpenPGP

Post-quantum extension of OpenPGP

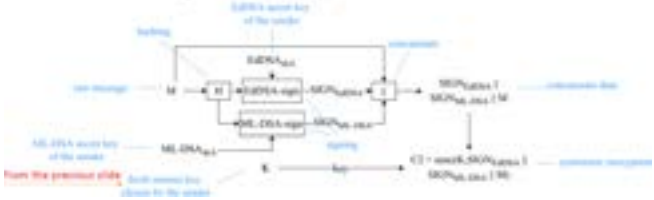
1) Composite KEMs ML-KEM + ECDH KEM for hybrid key encapsulations



Formal Specification of PQ OpenPGP

Post-quantum extension of OpenPGP

2) Composite digital signatures ML-DSA + EdDSA
MUST successfully validate both signatures



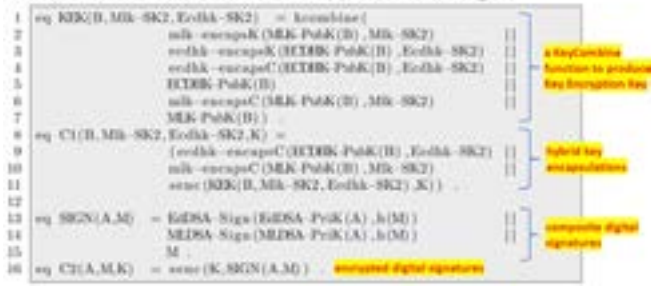
Formal Specification of PQ OpenPGP

Modeling ML-KEM

- To specify a probabilistic function as a deterministic function in CafeOBJ, an argument is added as a random parameter.
- Original: keygen() -> (sk, pk)
- How to represent the mapping between sk and pk?
- CafeOBJ: keygen(sk) -> pk

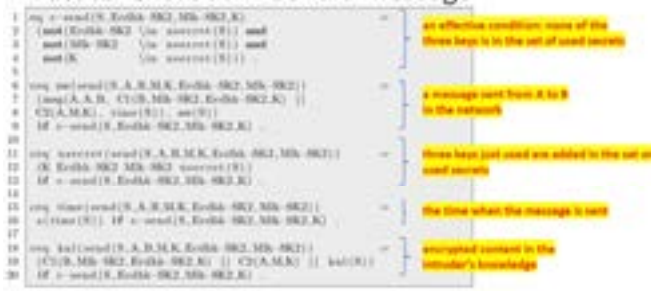
Formal Specification of PQ OpenPGP

Protocol execution: Encrypt a message



Formal Specification of PQ OpenPGP

Protocol execution: Send a message



Formal Specification of PQ OpenPGP

Threat intruder model

- A generic Dolev-Yao intruder who has the following capabilities:
  1. Intercept messages over the network and extract contents
  2. Generate random components, such as the session key, etc.
  3. Use available information to compute KEM ciphertexts, derive shared secrets, etc..
  4. Apply any cryptographic primitive function to obtain useful information
  5. Forge messages using available data
  6. Compromise secrets, such as long-term private keys, etc.
  7. Exploit quantum computing to break the security of traditional cryptography like ECDH and EdDSA

Formal Specification of PQ OpenPGP

Intruder specification

```
op fkMsg   : Sys Prin Prin ECDHK-Cipher MLK-Cipher Data Data
             -> Sys {constr}
op c-fkMsg : Sys Prin Prin ECDHK-Cipher MLK-Cipher Data Data
             -> Bool

eq c-fkMsg(S,A,B,EcdhK-C1,MlK-C1,KC,C2)      =
     (EcdhK-C1 \in kul(S) and
      MlK-C1   \in kul(S) and
      KC       \in kul(S) and
      C2       \in kul(S)) .

ceq nw(fkMsg(S,A,B,EcdhK-C1,MlK-C1,KC,C2)) =
     (msg(intru,A,B, EcdhK-C1 || MlK-C1 || KC || C2,
     time(S)) , nw(S)) if c-fkMsg(S,A,B,EcdhK-C1,MlK-C1,KC,C2) .
eq time(fkMsg(S,A,B,EcdhK-C1,MlK-C1,KC,C2)) = s(time(S)) .
```

encrypted data in the intruder's knowledge

a fake message from the intruder in the network

Formal Specification of PQ OpenPGP

Intruder specification

An intruder has the ECDH public key, then derives the ECDH private key using a large quantum computer.

```
op intruBreakECDH              : Sys Prin -> Sys {constr}
op c-intruBreakECDH            : Sys Prin -> Bool

eq c-intruBreakECDH(S,A)       = (ECDHK-PubK(A) \in kul(S)) .
ceq kul(intruBreakECDH(S,A)) = (ecdhk-getSecret(ECDHK-PubK(A))
                  || kul(S)) if      c-intruBreakECDH(S,A) .
ceq intruBreakECDH(S,A)        = S if not c-intruBreakECDH(S,A) .
```


Formal Verification of PQ OpenPGP

Secrecy of session key

- A message sent from A to B,
- B derives KEK, then decrypts to get K
- Not in leak sources:
  - The session key K
  - ML-KEM shared secret by Decaps
  - ML-KEM long-term private key
- K is not in the intruder's knowledge
- Proofs: Seven lemmas required
- Execution time: 1,8 seconds

Formal Verification of PQ OpenPGP

Forward secrecy of the session key

- Additional conditions:
  - ML-KEM long-term private key in leak sources
  - Only leaked after the message has been sent
- Proofs: Eight lemmas required
- Execution time: 2 seconds

Formal Verification of PQ OpenPGP

Authenticity

- Additional conditions:
  - B decrypts C2
  - B verifies two digital signatures
- A has indeed sent the message to B
- Proofs: Five lemmas required
- Execution time: 2,5 seconds

Formal Verification of PQ OpenPGP

Experimental results

Category	Name	Auxiliaries	Case splitting	Time (ms)
Invariants	keySc (secrecy of session key)	inv{1, 2, 3, 8}	119	1,764
	fwdSc (forward secrecy)	inv{1, 2, 3, 7, 8}	132	2,181
	auth (authenticity)	inv{6}	109	2,497
Lemmas	inv1	inv{8}	72	822
	inv2	inv{4}	87	1,066
	inv3	inv{1, 8, 9}	91	1,616
	inv4	inv{1, 9}	76	892
	inv5	inv{1, 9}	79	944
	inv6	inv{3, 4, 5, 9}	157	8,409
	inv7	no lemma	90	1,069
	inv8	inv{9, 10}	80	944
	inv9	inv{10}	87	1,071
	inv10	no lemma	30	298

Formal Verification of PQ OpenPGP

Challenges

- Understanding the protocol
- Formalizing the protocol
- Validating the specification
- How to discharge a false case?
  - Finding a conjecture lemma*
  - Prove the lemma
  - If all true, then uses to prove

*A non-trivial task

Formal Verification of PQ OpenPGP

```
eq (r4 \in usecret(s)) = false .
eq (r5 \in usecret(s)) = false .
eq (r6 \in usecret(s)) = false .
eq (r1 = intru) = false .
eq (r2 = intru) = false .
eq (r4 \in knt(s)) = true .
eq (r4 \in leakscr(s)) = false .
eq (MLK-PriK(r2) \in' leakscr(s)) = false .
eq ((MLK-PriK(r2) & r6) \in leakscr(s)) = false .
eq (msg(r1,r1,r2,...,n2) \in nw(s)) = false .
```

Conjecture lemma:
r4 in intruder's knowledge <=>
r4 in used secrets

Trong Binh Hoang: Formal specification and analysis of Post-quantum OpenPGP protocol in CafeOBJ, Master's Thesis, JAIST, September, 2025.

Trong Binh Hoang, Duong Dinh Tran, Canh Minh Do and Kazuhiro Ogata: Formal Specification and Analysis of Post-quantum OpenPGP Protocol in CafeOBJ. 37th International Conference on Software Engineering and Knowledge Engineering (SEKE25), KSI Research Inc., pp.159-145, (2025)

Trong Binh Hoang

Duong Dinh Tran

Canh Minh Do

Formal Verification of PQ SSH

version exchange	VERSION_EX	$C \rightarrow S$:	Version_C
	VERSION_EX	$S \rightarrow C$:	Version_S
key exchange algorithms	KEX_ALGR	$C \rightarrow S$:	Suites_C
	KEX_ALGR	$S \rightarrow C$:	Suites_S
key exchange initiation	KEX_HBR_INIT	$C \rightarrow S$:	$\text{ECDH}_{\text{PK}_C}, \text{KEM}_{\text{PK}_C}$
key exchange reply	KEX_HBR_REPLY	$S \rightarrow C$:	$\text{LK}_S, \text{ECDH}_{\text{PK}_S}, \text{KEM}_{C_S}, \text{SIGN}$

Fig. 3. Messages exchanged in the PQ SSH protocol

Formal Verification of PQ SSH

Fig. 4. Exchange hash and signature calculation

Formal Verification of PQ SSH

Step-1	A	$A \rightarrow B$:	$\text{ECDH}_{\text{PK}} \| \text{KEM}_{\text{PK}}$
Step-2	I	learns	$\text{ECDH}_{\text{PK}} \| \text{KEM}_{\text{PK}}$
Step-3	I	$A_2 \rightarrow B$:	$\text{ECDH}_{\text{PK}} \| \text{KEM}_{\text{PK}}$
Step-4	B	$B \rightarrow A_2$:	$\text{LK}_B \| \text{ECDH}_{\text{PK}_2} \| \text{KEM}_C \| \text{SIGN}$
Step-5	I	learns	$\text{LK}_B \| \text{ECDH}_{\text{PK}_2} \| \text{KEM}_C \| \text{SIGN}$
Step-6	I	$B \rightarrow A$:	$\text{LK}_B \| \text{ECDH}_{\text{PK}_2} \| \text{KEM}_C \| \text{SIGN}$

where I denotes the intruder

Fig. 5. Counterexample of auth

Formal Verification of PQ SSH

$$H = \text{hash}(\text{Version}_C \,||\, \text{Version}_S \,||\, \text{Suites}_C \,||\, \text{Suites}_S \,||\, LK_S \,||\, ECDH_{PK_C} \,||\, KEM_{PK_C} \,||\, ECDH_{PK_S} \,||\, KEM_{C_S} \,||\, K \,||\, A \,||\, B)$$

Client and server identifications added

D.D. Tran, K. Ogata, S. Escobar, S. Akleylek, A. Otmani: Formal analysis of Post-Quantum Hybrid Key Exchange SSH Transport Layer Protocol, IEEE Access 12: 1672-1687 (2024)

OGATA Kazuhiro (Japan), Santiago Escobar (Spain), Ayoub Otmani (France), Sedat Akleylek (Turkey): Formal Analysis and Verification of Post-Quantum Cryptographic Protocols (FAVPQC), **ICT for Resilient, Safe and Secure Society, EIG CONCERT-Japan, SICORP, JST**, FY2021 - FY2023

Kazuhiro Ogata (Japan)

Santiago Escobar (Spain)

Sedat Akleylek (Turkey/Estonia)

Ayoub Otmani (France)

Canh Minh Do, Adrian Riesco, Santiago Escobar, Kazuhiro Ogata: Parallel Maude-NPA for Cryptographic Protocol Analysis, IEEE Transactions on Dependable and Secure Computing (IEEE TDSC), Pages 1 – 18, IEEE, 2025. to appear

Canh Minh Do, Tsubasa Takagi, Kazuhiro Ogata: Automated Quantum Protocol Verification Based on Concurrent Dynamic Quantum Logic, ACM Transactions on Software Engineering and Methodology (ACM TOSEM), 34(6): Article No.: 182, 1–36 (2025), ACM, 2025.

D.D. Tran, K. Ogata, S. Escobar, S. Akleylek, A. Otmani: Symbolic verification of Hybrid Post-Quantum TLS 1.2, under review for journal publication

Adrian Riesco, Kazuhiro Ogata, Masaki Nakamura, Daniel Gaina, Duong Dinh Tran, KokichiFutatsug: Proof Scores: A Survey, ACM Computing Surveys (ACM CSUR), 57 (10), Article No.: 251, Pages 1 – 37, ACM, 2025

Some future directions

- A more generic intruder model that can be used for formal verification of other PQ security protocols
- More case studies of PQ protocol formal verification
- Automatic/systematic lemma conjecture
- Making IPSG more scalable

Some future directions

- Formal verification of quantum security protocols in which quantum cryptographic primitives, such as BB84, are used, including an intruder model for it
- To this end, we need to comprehend quantum circuits/protocols/programs better, for which we have been working on formal verification of quantum circuits/protocols/programs

Some future directions

- The following Kaken project has been accepted, where part of the future directions are carried out:

  Logical foundation and formal verification of quantum-resistant security protocols (Fostering Joint International Research)
  2024-09-09 – 2028-03-31





Tsubasa Takagi

Acknowledgement

The staff members of the four funding agencies (JST, CNRS, AEI and TUBITAK) and the four universities (JASIT, Polytechnic University of Valencia, University of Rouen Normandie and Ondokuz Mayis University) have always supported FAVPQC.

We appreciate their efforts and time.

Dr. Yuzuru Tanaka, Program Officer, has always encouraged us to conduct good research.

We are grateful to him for his endless encouragement.

Thank you for listening!







Constructing Proof Scores in CafeOBJ

FUTATSUGI, Kokichi
二木 厚吉
Japan Advanced Institute of Sciecne and Technology
(JAIST)

at LAC2025
on 30 September 2025

Contents of Talk

(1) Specification Verification, CafeOBJ, Proof Scores

(2) Proof Tree Calculus and Well-Founded Induction (PTcalc+WFI)

(3) Constructing Proof Scores in CafeOBJ
based on the appendix A of the following paper:
"Advances of Proof Scores in CafeOBJ"
http://arxiv.org/abs/2112.10373v3
which is a revised and extended version of
the published journal (Sci.Comp.Prog.) paper
with the same title

(4) Concluding Remarks and Future Perspective

What is Specification Verification?

- Specification is a description of models, and **Specification Verification** is to show that the models have desirable properties by deducing the properties from the specification.
- **Specification Verification** in this sense is theorem proving where the specification is a set of axioms and the desirable properties are theorems.

Theorem Proving vs. Specification Verification

- The primary goal of **Theorem Proving** is to show that theorems can be deduced from axioms with deduction rules. Usually the axioms are already established and does not change.
- The primary goal of **Specification Verification** is to check the quality of a specification (a set of axioms) against the desirable properties. The specification is supposed to be improved/refined to have the desirable properties through specification verification.

Characteristics of Specification Verification

- The improvements of a specification (spec for short) include the **addition of necessary axioms**, the **deletion of unnecessary axioms**, and the **improvement of the spec's module structure**.
- For achieving the improvements, a spec and the machinery of the spec verifivation are better to be clear and transparent. That is, the followings are better to be clear and precisely defined, and hopefully simple and transparent.
  (i) **Models of a spec** (models of a system specified),
  (ii) **inference (deduction) rules** used,
  (iii) **rationale for assumptions used** (i.e. why an assumption is justified to be used or not).

All models are wrong

- **"All models are wrong, some are useful."** and specification verification (i.e. modeling, description, and verification of models) is challenging.
- Specification Verification is **a good (may be the best) way for getting a high-quality specification** (i.e. useful description of models).
- Critical flaws continue to exist at the level of domain, requirement, and/or design specification, and specification verification is still **one of the most important challenges in software/system engineering**.

CafeOBJ (`https://cafeobj.org/`)

- CafeOBJ is an **executable algebraic spec language system** based on **equational logic** and **rewriting execution** with transparent and precise definition of the models and the inference/execution rules.
- CafeOBJ's **powerful module system** makes it possible to describe the assumption/conclusion relations in a clear and transparent way.
- **CafeOBJ** can be used for **specification verification** by writing a **proof score** for verifying that any model of a specification (a CafeOBJ module) M satisfies a property of interest p (a Boolean ground term). In symbols $M \models p$.

A little bit of CafeOBJ history

- 1983-1984: Design/implementation of the OBJ2 language system at SRI International (Standord Research Institute)
- 1985-1995: Several CafeOBJ design and implementation attempts inheriting achievements of the OBJ project at SRI
- 1996-1998: An intensive international CafeOBJ project
  - Six Japanese Companies, Five Japanese Universities, Three Foreign Research Group participate
- 1999-2000: Sufficiently reliable and usable CafeOBJ system was available
- 2000-2021: Researches on **Specification Verification in CafeOBJ**
  - Specification verification in an appropriate abstraction level

Proof Scores in CafeOBJ

- A **proof score** is an **executable high level description of an exhaustive symbolic test** (i.e. a complete proof) in reduction rules (i.e. equations) and reduction commands. That is, a proof score is doing a **proof by reduction**.
- A proof score for a specification (a CafeOBJ module) is a piece of code that constructs **proof modules** and does **reductions** with the equations in the specification and the proof modules.
- **Domain/requirement/design engineers** using CafeOBJ can construct proof scores for specification verification.

CafeOBJ Proof Score approach has been applied to the following kinds of problems and found usable.

- Some classical mutual exclusion algorithms
- Some real time algorithms
  e.g. Fischer' s mutual exclusion protocol
- Railway signaling systems
- Authentication protocols
  e.g. NSLPK, Otway-Rees, STS protocols
- Practical sized e-commerce protocol of SET
- UML semantics (class diagram + OCL-assertions)
- Formal Fault Tree Analysis
- Secure workflow models, internal control
- Automatic non-stop software update protocols/systems
- International standard for automotive software
- Protocols for Cloud computing

PTcalc and WFI

- Major high level planing in proof scores includes **lemma**, **case-split**, and **induction**. They are supposed to be found by humans and declared as proof modules.
- Proof modules can be constructed manually with `open-close` constructs, and case-split/induction can be described in a liberal way. It involves, however, **risks of obscuring the rationale of the high level planning**.
- **Proof tree calculus (PTcalc)** and **well-founded induction (WFI)** are incorporated to solve the issue, by realizing a **case-split with exhaustive equations** and an **induction via a well-founded relation**.

PTcalc (Proof Tree Calculus) (1)

- A CafeOBJ's specification (a module) M contains **equations** and defines the set of models $\mathbb{M}od(M)$ each element of that satisfies the equations.
- **PTcalc** (Proof Tree Calculus) is a subsystem of CafeOBJ and helps to prove that a property p holds for any model in $\mathbb{M}od(M)$ (in symbols $M \models p$; M satisfies p). p is expressed as a **Boolean ground term** with fresh constants that correspond to the parameters of the property.

PTcalc (2)

- If the term p is deduced to be equal to `true` with the M's equations (in symbols $M \vdash_{\mathrm{e}} p$), any model of M satisfies p (in symbols $M \models p$) by the soundness of equational deduction (in symbols $M \vdash_{\mathrm{e}} p \Rightarrow M \models p$).
- If the term p is reduced to `true` by using the M's equations as reduction/rewrite rules from left to right (in symbols $M \vdash_{\mathrm{r}} p$), $M \vdash_{\mathrm{e}} p$ holds by the soundness of reduction with respect to equational deduction (in symbols $M \vdash_{\mathrm{r}} p \Rightarrow M \vdash_{\mathrm{e}} p$).
- CafeOBJ's reduction is an implementation of $M \vdash_{\mathrm{r}} p$ and we get $M \vdash_{\mathrm{c}} p \Rightarrow M \vdash_{\mathrm{r}} p$. Because implication ($\Rightarrow$) is transitive the following proof rule is obtained.

  (1) $M \vdash_{\mathrm{c}} p \Rightarrow M \models p$

  (if p reduced to `true` by CafeOBJ at M then M satisfies p)

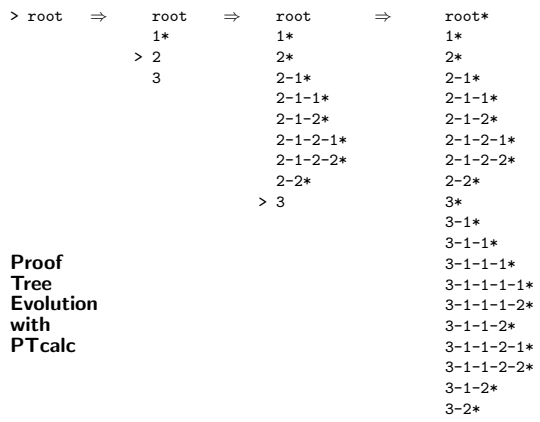
PTcalc (3)

- Usually $M \vdash_{\mathrm{c}} p$ is difficult to prove directly, and we need to find case splitting equations $e_1, \cdots, e_n$ such that at least one of them holds for any model in $\mathbb{M}od(M)$. That is, the equations cover all the possibilities (i.e. are exhaustive).
- Let M_{+e_i} be the module gotten by adding e_i to M, then each model in $\mathbb{M}od(M)$ is a model of M_{+e_j} for some $j \in \{1, 2, \cdots, n\}$, and we get the following proof rule of **case split with exhaustive equations**.

  (2) $(M_{+e_1} \models p \wedge M_{+e_2} \models p \wedge \cdots \wedge M_{+e_n} \models p) \Rightarrow M \models p$

PTcalc (4)

- $M_{+e_i} \vdash_{\mathrm{c}} p$ would be still difficult to prove and **(2)** is applied repeatedly. The repeated applications of **(2)** generate **proof trees** successively.
- Each of the generated proof trees has the **root node** $M \models p$ and each of other **nodes** is of the form $M_{+e_{i_1} \cdots +e_{i_m}} \models p$ ($m \in \{1, 2, \cdots\}$) that is generated as a **child node** of $M_{+e_{i_1} \cdots +e_{i_{m-1}}} \models p$ by applying **(2)**.
- A **leaf node** (i.e. a node without child nodes) $M_{+e_{i_1} \cdots +e_{i_k}} \models p$ ($k \in \{0, 1, \cdots\}$) of a proof tree is called **effective** if $M_{+e_{i_1} \cdots +e_{i_k}} \vdash_{\mathrm{c}} p$ holds.
- A proof tree is called **effective** if all of whose leaf nodes are effective. PTcalc proves $M \models p$ by constructing an effective proof tree whose root node is $M \models p$.



Proof Tree Evolution with PTcalc

An Example of Proof Score about Plus _+_ on Peano Natural Numbers

```
-- PNAT: Peano NATural numbers
mod! PNAT { [Pnat]
op 0 : -> Pnat {constr} .
op s_ : Pnat -> Pnat {constr} .
-- _=_ is built-in
eq (s M:Pnat = s N:Pnat) = (M = N) . }
```

$\texttt{Pnat} \stackrel{\text{def}}{=} \{\texttt{0}, \texttt{s 0}, \texttt{s s 0}, \texttt{s s s 0}, \cdots\}$

```
-- PNAT+ : PNAT with addition _+_
mod! PNAT+ { pr(PNAT)
op _+_ : Pnat Pnat -> Pnat .
vars X Y : Pnat .
eq 0 + Y = Y .
eq s X + Y = s(X + Y) . }
```


A Property of Interest: Associativity of _+_

$\forall x, y, z \in \texttt{Pnat}\ ((x\ \texttt{+}\ y)\ \texttt{+}\ z = x\ \texttt{+}\ (y\ \texttt{+}\ z))$

```
-- module for defining the associativity of _+_
mod! PNAT+assoc { pr(PNAT+)
pred assoc+ : Pnat Pnat Pnat .
vars X Y Z : Pnat .
eq assoc+(X,Y,Z) = ((X + Y) + Z = X + (Y + Z)) . }
```

for fresh constants "`ops x@ y@ z@ : -> Pnat .`"
"`red in PNAT+assoc : assoc+(x@,y@,x@) .`" implies
$(\texttt{PNAT+assoc} \models \texttt{assoc+(x@,y@,z@)})$

Ordinary Induction vs. Well-Founded Induction

- An effective induction scheme varies on problems and an inductive proof score tends to be written in an ad hoc manner.
- Well-founded induction is well known to be powerful enough to subsume a varieties of induction schemes. A method to codify well-founded induction in CafeOBJ proof score has been developed.
- The method makes it possible to codify various induction schemes (including structural induction) in a uniform and transparent style.

Principle of Well-Founded Induction (WFI)

- T : a set.
- p : a predicate on T, i.e. a function from T to the truth values $\{\texttt{true}, \texttt{false}\}$.
- $wf>$: a **well-founded binary relation** on T, i.e. $wf> \subseteq T \times T$ and there is no infinite sequence of $t_i \in T$ with $(t_i, t_{i+1}) \in wf>$ $(i \in \{1, 2, \cdots\})$.

- The **principle of WFI** we use is formulated as follows, where $t\ wf> t'$ means $(t, t') \in wf>$.

$$\boxed{\begin{array}{l}\forall t \in T([\forall t' \in T((t\ wf> t')\ \texttt{implies}\ p(t'))]\ \texttt{implies}\ p(t)) \\ \texttt{implies} \\ \forall t \in T(p(t))\end{array}}$$

  I.e., if $p(t)$ whenever $p(t')$ for all $t' \in T$ such that $t\ wf> t'$, then $p(t)$ for all $t \in T$.

Proof of the WFI Principle:

Assume there exits $t \in T$ such that $\texttt{not}(p(t))$, then there should be $t' \in T$ such that $t\ wf> t'$ and $\texttt{not}(p(t'))$.

Repeating this produces an infinite $wf>$-descending sequence. It conflicts with well-foundedness of $wf>$.

Proof Module for Defining the Induction Hypothesis of WFI

```
mod PNAT+assoc-wfi-hypo {
pr(PNAT+assoc) -- base module on which the proof applies
-- declaring assoc+'s argument tuple,
-- i.e. a sort Ppp of 3 tuples of Pnat and its constructor t
[Ppp] op t : Pnat Pnat Pnat -> Ppp {constr} .
-- a well-founded binary relation _wf>_ on Ppp
vars X Y Z : Pnat .
pred _wf>_ : Ppp Ppp . eq t(s X,Y,Z) wf> t(X,Y,Z) = true .
-- fresh constants for expressing
-- the goal proposition: assoc+(x@,y@,z@)
ops x@ y@ z@ : -> Pnat .
-- induction hypothesis of WFI
ceq[assoc+ih :nonexec]:
    assoc+(X,Y,Z) = true if t(x@,y@,z@) wf> t(X,Y,Z) .
-- fresh constant for refining term x@
op x$1 : -> Pnat . }
```


Executing Proof by Proof Tree Calculus (PTcalc)

```
select PNAT+assoc-wfi-hypo .
:goal{eq assoc+(x@,y@,z@) = true .}
-- exhaustive equations for refining term x@
:def x@ = :csp{eq x@ = 0 . eq x@ = s x$1 .}
:def ih = :init as assoc+ih-1 [assoc+ih] .
:apply(x@ rd- ih rd-)
```

```
PNAT+assoc-wfi-hypo> :show proof
root*
[x@]  1*
[x@]  2*
[ih]  2-1*
```

$$(\texttt{PNAT+assoc-wfi-hypo} \models \texttt{assoc+(x@,y@,z@)}) \Rightarrow (\texttt{PNAT+assoc} \models \texttt{assoc+(x@,y@,z@)})$$

PTcalc+WFI

- PTcalc is more fundamental than WFI and there are many PTcalc proof scores without WFI. However, a **term refinements** (an instance of case-split with exhaustive equations) is necessary to do WFI, and WFI is always PTcalce+WFI.

System Specification in CafeOBJ: ADT or Transition System

- A **static system** is specified as an **Abstract Data Type (ADT)**.
  - ADT is described algebraically with possibly **Conditional Equations**.
- A **dynamic system** is specified as a **Transition System** (i.e., a state transition system).
  - A transition system is described as (i) a special ADT called **OTS (observational transition system)** or (ii) a **Transition Specification**.

A Transition Specification is defined by giving the following three items

- ADT for defining **State Configurations** of the transition system.
- **Transitions** on the state configurations that define the behavior of the transition system. The transitions on the state configurations are defined with possibly conditional **Transition Rules**; a transition rule is a **Special Equation** on state configurations that defines transitions from an instance of the rule's left hand side to the corresponding instance of the rule's right hand side.
- **Initial States** as a subset of the state configurations.

Models of a CafeOBJ Module: Algebras

- For formally verifying specifications, **mathematical models** of a specification (i.e., a CafeOBJ Module) should be defined.
- A mathematical system that consists of a collection of **sets and functions on the sets** is called an **algebra**.
- Each CafeOBJ module denotes a class of algebras each of which interprets the **sorts and operators** declared in the module as **sets and functions on the sets** (i.e., the sorts denote the sets and the operators denote the functions on the sets).

Models of a CafeOBJ Module – Equation, State Transition –

- If **equations** are declared in a module, each model (algebra) A of the module satisfies the equations (i.e., any assignment of A's values to variables makes the left and right hand sides of each equation equal).
- If **a special sort State** and a set TI of **transition rules** on the sort `State` are declared in a module, for each model (algebra) A of the module, TI defines the **transition relation** $TR_{TI} \subseteq A_{\texttt{State}} \times A_{\texttt{State}}$, where $A_{\texttt{State}}$ is the set the sort `State` denotes.

$M \models bt$
"M satisfies bt" or "bt is `true` for every model of M"

- CafeOBJ has a builtin module `BOOL`, with a sort `Bool`, that implements the **propositional calculus** as an executable Boolean algebra. A standard CafeOBJ module includes `BOOL` automatically and every model of the module includes the Boolean algebra.
- For a CafeOBJ module M and a ground term (i.e., a term without variables) bt of the sort `Bool`, let $M \models bt$ mean that every model of M **satisfies** bt (i.e., bt is `true` for every model of M). A ground term can contain **fresh constants** each of which denotes, like a variable, any element of the corresponding sort .

Property Specifications and Proof Modules for ADT

- Let $M \models_{\mathrm{prp}} \pi$ mean that a property π holds for every model of a module M.
- If M specifies an ADT, the property π is supposed to be such that a Boolean ground term bt and a module M^{π}, which satisfies the following implication, can be constructed.

  (pm) $M^{\pi} \models bt \Rightarrow M \models_{\mathrm{prp}} \pi$
- Let M_{bt} be the module defining the operators (functions including predicates) needed to express the term bt.

  $M^{\pi} = M + M_{bt}$

  M^{π}, M_{bt} : **proof modules**

  M_{bt} : **property specification**

Properties for Transition Specifications

- If M is a transition specification and specifies a transition system, π is supposed to be an **invariant property** or a **leads-to property**.
- An **invariant property** is a safety property and is specified with a state predicate p that holds for all reachable states.
- A **leads-to property** is a liveness property and is specified with two state predicates p and q.
  A **(p leads-to q) property** holds for a transition system if the system's entry to the state where p holds implies that the system will surely get into the state where q holds no matter what transition sequence is taken.

Property Specifications and Proof Modules for TSs

- For a transition specification (module) M and its property π, modules M_i^π and Boolean ground terms bt_i $(1 \leq i \leq m)$ that satisfy the following implication can be constructed by making use of **builtin search predicates**.
  (PM) $(M_1^\pi \models bt_1 \wedge M_2^\pi \models bt_2 \wedge \cdots \wedge M_m^\pi \models bt_m) \Rightarrow M \models_{\mathrm{prp}} \pi$
- Let M_{bt_i} be the property specification defining the operators needed to express the term bt_i $(1 \leq i \leq m)$, then $M_i^\pi = M + M_{bt_i}$.
  - Predicates p, q needed to express invariant or leads-to properties are also specified in M_{bt_i} $(1 \leq i \leq m)$.

Model Based Deduction Rule:

(PR1) $M \vdash_{\mathrm{c}} bt \Rightarrow M \models bt$

- CafeOBJ system has a reduction command "`reduce in` M : bt ." for getting a reduced form of bt in the module M by using M's equations as reduction rules from left to right.
- Let $M \vdash_{\mathrm{c}} bt$ means that "`reduce in` M : bt ." returns `true`, then the following primary proof rule says that bt is `true` for every model of M if bt reduces to `true` in M.
  (PR1) $\boxed{M \vdash_{\mathrm{c}} bt \Rightarrow M \models bt}$

Model Based Deduction Rule: (PR2)

- Let M be a module with a signature Σ and a set of equations E ($M = (\Sigma, E)$ in symbols). Equations $e_1, \cdots, e_n$ ($1 \leq n$) are defined to be **exhaustive** for M iff for any model A of M there exists some i ($1 \leq i \leq n$) such that e_i holds in A.
- Let $e_1, \cdots, e_n$ ($1 \leq n$) be exhaustive equations and $M_{+e_i} = (\Sigma \cup Y_i, E \cup \{e_i\})$ ($1 \leq i \leq n$, Y_i is the set of fresh constants in e_i), then the following proof rule holds on which every case-split in CafeOBJ can be based.
  (PR2) $\boxed{(M_{+e_1} \models bt \wedge M_{+e_2} \models bt \wedge \cdots \wedge M_{+e_n} \models bt) \Rightarrow M \models bt}$

Constractors and Constrained Sorts

- A **constractor** of a sort is an operator that constructs elements of the sort. A sort is called **constrained** if there exists a constructor of the sort. Let $bt(y@_1, \cdots, y@_n)$ be a ground term of the sort `Bool` containing fresh constants $y@_i$ of constrained sorts s_i ($1 \leq i \leq n$).

Model Based Deduction Rule: (WFI)

- Let $Y@$ be the set $\{y@_1, \cdots, y@_n\}$, $\overline{y@}$ be the n-tuple $y@_1, \cdots, y@_n$, y_i be variables of sorts $s_i (1 \leq i \leq n)$, and $\overline{y}$ be the n-tuple $y_1, \cdots, y_n$. Let $\_wf>\_$ be a well-founded binary relation on the set of n-tuples $ct_1, \cdots, ct_n$ of ground constructor terms of sorts $s_1, \cdots, s_n$, and $M_{wf>}$ be the module adding the definition of $\_wf>\_$ to M. The following **well-founded induction (WFI)** rule can be used for deducing $M \models bt(y@_1, \cdots, y@_n)$.

  (WFI)

  $\boxed{(M_{wf>} \cup Y@ \cup \{\texttt{cq}\ bt(\overline{y}) = \texttt{true if}\ \overline{y@}\ wf> \overline{y}\ .\}) \models bt(\overline{y@}) \Rightarrow M \models bt(\overline{y@})}$

PTcalc (Proof Tree Calculus): (PR1,PR2)*

- PTcalc is a subsystem of CafeOBJ system to support proving $M \models bt$ with the proof rules **(PR1)** and **(PR2)**.
- $M \vdash_{\mathrm{c}} bt$ does not always holds, and $M \models bt$ is usually difficult to duduce directly using the proof rule **(PR1)**. Exhaustive equations $e_1, \cdots, e_n$ need to be found and the proof rule **(PR2)** should be used.
- $M_{+e_i} \models bt$ would be still difficult to prove with the proof rule **(PR1)**, and **(PR2)** should be applied repeatedly. The repeated applications of **(PR2)** generate **proof trees** successively.

PTcalc (Proof Tree Calculus): Effective Proof Tree

- Each of the generated proof trees has the **root node** $M \models bt$ and each of other **nodes** is of the form $M_{+e_{i_1} \cdots + e_{i_m}} \models bt$ $(1 \leq m)$ that is generated as a **child node** of $M_{+e_{i_1} \cdots + e_{i_{m-1}}} \models bt$ by applying **(PR2)**.
- A **leaf node** (i.e., a node without child nodes) $M_{+e_{l_1} \cdots + e_{l_k}} \models bt$ $(0 \leq k)$ of a proof tree is called **effective** if $M_{+e_{l_1} \cdots + e_{l_k}} \vdash_{\mathrm{c}} bt$ holds.
- A **proof tree** is called **effective** if all of whose leaf nodes are effective. PTcalc proves $M \models bt$ by constructing an effective proof tree whose root node is $M \models bt$.

A Generic Procedure for Constructing PSs: (1)(2)(a)(i)(ii)(b)

A collection of pieces of CafeOBJ code for deducing that a property π holds for every model of a module M ($M \models_{\mathrm{prp}} \pi$ in symbols) is called a proof score. A generic procedure to construct a proof score for deducing $M \models_{\mathrm{prp}} \pi$ is as follows.

(1) Construct proof modules M_i^π and Boolean ground terms bt_i $(1 \leq i \leq m)$ satisfying the following implication by using the rule **(pm)** or **(PM)**.

$$\boxed{(M_1^\pi \models bt_1 \wedge M_2^\pi \models bt_2 \wedge \cdots \wedge M_m^\pi \models bt_m) \Rightarrow M \models_{\mathrm{prp}} \pi}$$

(2) (a) (i)
(ii)
(b)

A Generic Procedure for Constructing PSs: (1)(2)(a)(i)(ii)(b)

(1)

(2) For each $M_i^{\pi} \models bt_i$ $(1 \leq i \leq m)$ do as follows.

(a) Construct a root module M_i^{rt} as follows depending on whether the rule **(WFI)** is used or not.

(i) If bt_i is expressed as $bt_i(\overline{y@})$ with fresh constants $\overline{y@}$ of constrained sorts and the $bt_i(\overline{y@})$'s proof can be achieved with WFI, then define a well-founded relation $\_wf>\_$ appropriately and construct the module M_i^{rt} as follows with variables $\overline{y}$ of the corresponding constrained sorts.

$$M_i^{\mathrm{rt}} = (M_i^{\pi})_{wf>} \cup Y@ \cup \{\texttt{cq } bt_i(\overline{y}) = \texttt{true if } \overline{y@}\ wf> \overline{y}\ .\}$$

(ii) If the condition of (i) is not satisfied, then construct the module M_i^{rt} as follows.

$$M_i^{\mathrm{rt}} = M_i^{\pi}$$

(b)

A Generic Procedure for Constructing PSs: (1)(2)(a)(i)(ii)(b)

(1)

(2) (a) (i)
(ii)

(b) Construct a CafeOBJ code with PTcalc commands that constructs an effective proof tree with the root node $M_i^{\mathrm{rt}} \models bt_i$. If a lemma (a property) λ is found necessary to be proved with respect to a module M^{λ} (i.e., $M^{\lambda} \models_{\mathrm{prp}} \lambda$ needs to be proved), then set $\pi = \lambda$, $M = M^{\lambda}$ and construct a proof score for $M \models_{\mathrm{prp}} \pi$ using the procedure (1)(2)(a)(i)(ii)(b). Note that constructions of proof scores for lemmas may be recursively invoked.

The procedure (1)(2)(a)(i)(ii)(b) does not always terminate. If terminates, the CafeOBJ codes constructed, including the codes for proving necessary lemmas, constitute a proof score for deducing $M \models_{\mathrm{prp}} \pi$.

Characteristics of PTcalc+WFI Proof Scores

Comparing with the most notable theorem proving systems like Coq, Isabelle/HOL, PVS, the spec verification with PTcalc+WFI proof scores in CafeOBJ has the following characteristics.

(i) Systems/services are modeled with **Algebraic Abstract Data Types (ADT)** and/or **Transition Specifications** and an appropriate higher abstraction level can be settled for each specification.

(ii) The only two proof rules **(PR1)**, **(PR2)** are simple and transparent, formalized at the level of **specification satisfaction** $SP \models p$ (i.e. any model of SP satisfies p), and supporting **model based proofs**.

Soundness Conditions for PTcalc+WFI Proof Scores

The conditions for PTclac+WFI proof scores to be correct (sound) are intensively localized into the following three points:

(i) **Exhaustiveness of the equations** declared in a `:csp{...}` command;

(ii) **Validity of the equation** declared inside `(...)` of a `:init` command "`:init ... (...) ...`";

(iii) **Well-foundedness of the binary relation** `_wf>_` used to declare an induction hypothesis.

The well-foundedness of a binary relation `_wf>_` on argument n tuples of a goal predicate can be established, in the majority of cases, via **the proper sub-term relaions on constructor terms**. This contributes to applicability and flexibility of PTcalc+WFI.

Proof Automation and Proof Planning in PTcalc+WFI

- Fully automatic verifications often fail to convey important properties/structures of the systems/problems.
- **Balanced optimal** of **proof automation** and **proof planning** is important; Computers for the tedious formal calculations, and humans for the high level planning. Proof scores intend to meet these requirements.
- In **PTcalc+WFI**, proof automation is achieved by construction of proof trees with effectiveness checks via **fully automated reductions**, and Proof planning is codified into **proof modules** and **reduction commands**.
- This **two layered architecture** is simple, but powerful enough to achieve a nice balance of proof automation and proof planning.

Future Perspective

- There should be many chances of developing significant practical specifications and proof scores in PTcalc+WFI.
- The current CafeOBJ system provides a nice platform for conceiving, experimenting, and/or formalizing new specification verification methods on top of proof scores in PTcalc+WFI.

You can find almost all necessary information on CafeOBJ at:

https://cafeobj.org/

Slides for this talk is at:

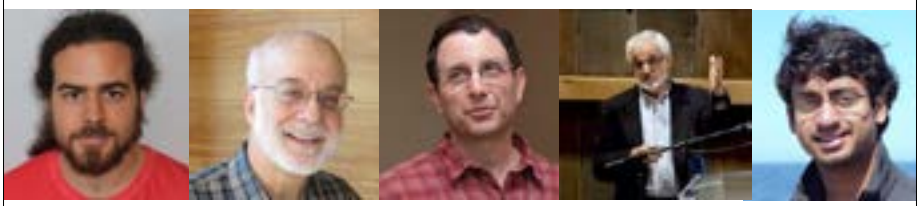
https://cafeobj.org/~futatsugi/talk/lac2025/

A recently published historical and comprehensive survey:

Proof Scores: A Survey
by A Riesco, K Ogata, M Nakamura, D Gaina, D D Tran, K Futatsugi,
ACM Computing Surveys, 57-10, Article 251, pp.1-37, May 2025
https://doi.org/10.1145/3729166

On a Relative of the Ehrenfeucht-Fraïssé Game and Software for Helping with its Analysis

Marco Carmosino (IBM), Ron Fagin (IBM), Neil Immerman (UMass Amherst),
Phokion Kolaitis (IBM and UC Santa Cruz), **Jon Lenchner (IBM)**, and Rik Sengupta (IBM)*



Marco Ron Neil Phokion Rik

*With earlier contributions by Ken Regan, Nikhil Vyas, and Ryan Williams

Outline

- In the Beginning: A paper of Neil Immerman's from 1981 introducing a Game that captures Quantifier # and his Vision for the Game
- Rules of the Game & Fundamental Capture Theorem
- Why Quantifier # is a Potentially more Interesting Complexity Measure than the more usual Quantifier Rank

- **Quantifier Complexity of Boolean Functions**
- Example and First Results
- A Simplifying Concept in Game Analysis: The Notion of **Parallel Play**

Outline

- More Examples
- Culminating Result on the Quantifier Complexity of Boolean Functions
- The Road Ahead to Realize the Vision from Neil's 1981 Paper

- The **WHO-WINS** problem: how hard is it to decide who wins a generic **Ehrenfeucht-Fraïssé (EF) game**?
- How Hard is it to decide **WHO WINS** a **Multi-Structural (MS) game** (the game that captures Quantifier #)?

- Software to help with the Analysis of the Games

Back in 1981….

JOURNAL OF COMPUTER AND SYSTEM SCIENCES 22, 384–406 (1981)

Number of Quantifiers Is Better Than Number of Tape Cells*

NEIL IMMERMAN

Department of Mathematics, Tufts University, Medford, Massachusetts 02155, and Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Received January 5, 1981; revised January 15, 1981

Key Results from the Immerman Paper

1. Introduction of the **Quantifier Number, QN[] measure** – the number of quantifiers needed to express a property via a uniform sequence of FO sentences.

Key Results from the Immerman Paper

1. Introduction of the **Quantifier Number, QN[] measure** – the number of quantifiers needed to express a property via a uniform sequence of FO sentences.

2. Introduction of a combinatorial game that he called the **Separability Game**, that captures quantifier number.

Major focus of this talk!

Key Results from the Immerman Paper

1. Introduction of the **Quantifier Number, QN[] measure** – the number of quantifiers needed to express a property via a uniform sequence of FO sentences.

2. Introduction of a combinatorial game that he called the **Separability Game**, that captures quantifier number.

3. Proof that on *ordered structures*, and for $f(n) \geq \log n$, Major focus of this talk!

$$\mathbf{NSPACE}\lfloor f(n)\rfloor \subseteq \mathbf{QN}\lfloor (f(n))^2/\log n\rfloor \subseteq \mathbf{DSPACE}\lfloor (f(n))^2\rfloor.$$

Where the quantities inside $\lfloor\cdot\rfloor$ are meant in the $U()$ sense.

Key Results from the Immerman Paper

Plugging $f(n) = \log n$, we get:

$$\mathbf{NSPACE}\lfloor f(n)\rfloor \subseteq \mathbf{QN}\lfloor (f(n))^2/\log n\rfloor \subseteq \mathbf{DSPACE}\lfloor (f(n))^2\rfloor$$

$$\text{NSPACE}[\log n] \subseteq \text{QN}[\log n] \subseteq \text{DSPACE}[\log^2 n]$$

Key Results from the Immerman Paper

Plugging $f(n) = \log n$, we get:

$$\mathbf{NSPACE}\lfloor f(n)\rfloor \subseteq \mathbf{QN}\lfloor (f(n))^2/\log n\rfloor \subseteq \mathbf{DSPACE}\lfloor (f(n))^2\rfloor$$

$$\text{NSPACE}[\log n] \subseteq \text{QN}[\log n] \subseteq \text{DSPACE}[\log^2 n]$$

The left-hand inclusion implies that if we can find a single NP property of ordered structures that requires more than O(log n) quantifiers to express, we will have separated NL from NP!!

Key Results from the Immerman Paper

Plugging $f(n) = \log n$, we get:

$$\mathbf{NSPACE}[f(n)] \subseteq \mathbf{QN}[(f(n))^2/\log n] \subseteq \mathbf{DSPACE}[(f(n))^2]$$

$$\text{NSPACE}[\log n] \subseteq \text{QN}[\log n] \subseteq \text{DSPACE}[\log^2 n]$$

The left-hand inclusion implies that if we can find a single NP property of ordered structures that requires more than O(log n) quantifiers to express, we will have separated NL from NP!!

How to prove such a thing?

That's what led Neil to his game!

However…

Number of Quantifiers Is Better
Than Number of Tape Cells*

NEIL IMMERMAN

> Little is known about how to play the separability game. We leave it here as a jumping off point for further research. We urge others to study it, hoping that the separability game may become a viable tool for ascertaining some of the lower bounds which are "well believed" but have so far escaped proof.

Rediscovery of the Game

Back in 2020, a group of us at IBM rediscovered Immerman's Separability Game, but unaware of Neil's paper, we called it the "Multi-Structural Game" – a name we continue to use.

A few years later Neil joined our team.

To the Games….

Before getting to the Multi-Strunctural (MS) game, let's review the more usual Ehrenfeucht-Fraïssé game.

Ehrenfeucht-Fraïssé (EF) Games

An **Ehrenfeucht-Fraïssé (EF) game** is played on two structures **A** and **B** for a fixed set of r rounds.

There are two players, a **Spoiler** and a **Duplicator**. It is useful to think of the players as each having an infinite set of uniquely colored pebbles. Each player has the same set of such pebbles.

Spoiler plays first. He chooses a structure and picks an element from the structure and places a pebble on it.

Duplicator plays next, picking an element from the other structure and placing a pebble of the same color on it.

They keep alternating turns, placing distinctly colored pebbles for the r rounds.

Duplicator wins if the two sets of pebbled elements, under the chosen element by element correspondence, gives rise to a partial isomorphism. **Spoiler** wins otherwise.

7

Multi-Structural (MS) Games

Multi-Structural games are played on two **sets, $\mathcal{A}$ and $\mathcal{B}$, of structures**.

Multi-Structural (MS) Games

Multi-Structural games are played on two **sets, $\mathcal{A}$ and $\mathcal{B}$, of structures**.

Spoiler and Duplicator take turns playing from the two **sets, $\mathcal{A}$ and $\mathcal{B}$** for a fixed number **r** of **rounds**. Spoiler plays first and picks a side to play on, $\mathcal{A}$ or $\mathcal{B}$, and places a pebble on an element from each structure in the set.

Multi-Structural (MS) Games

Multi-Structural games are played on two **sets, $\mathcal{A}$ and $\mathcal{B}$, of structures**.

Spoiler and Duplicator take turns playing from the two **sets, $\mathcal{A}$ and $\mathcal{B}$** for a fixed number **r** of **rounds**. Spoiler plays first and picks a side to play on, $\mathcal{A}$ or $\mathcal{B}$, and places a pebble on an element from each structure in the set.

Duplicator plays on the other side. On Duplicator's turn, she can make copies of any structures on her side, and if she likes, place pebbles on the elements of the structures in all possible ways.

Multi-Structural (MS) Games

Multi-Structural games are played on two **sets, $\mathcal{A}$ and $\mathcal{B}$, of structures**.

Spoiler and Duplicator take turns playing from the two **sets, $\mathcal{A}$ and $\mathcal{B}$** for a fixed number **r** of **rounds**. Spoiler plays first and picks a side to play on, $\mathcal{A}$ or $\mathcal{B}$, and places a pebble on an element from each structure in the set.

Duplicator plays on the other side. On Duplicator's turn, she can make copies of any structures on her side, and if she likes, place pebbles on the elements of the structures in all possible ways.

Play proceeds in this way for r rounds.

To win, Duplicator just needs to demonstrate a partial isomorphism between the pebbled elements picked from one pair of structures, $A \in \mathcal{A}$ and $B \in \mathcal{B}$. Spoiler wins otherwise.

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game.

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game.

Illustrative MS game on Linear Orders

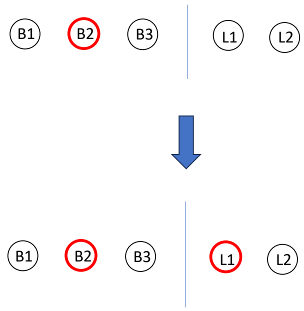
- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game

If Duplicator could not duplicate, she would lose!

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game

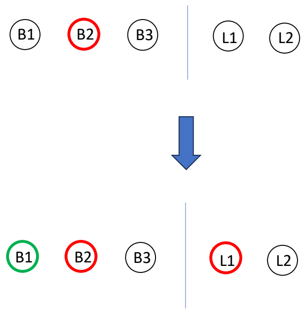
If Duplicator could not duplicate, she would lose!



Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game

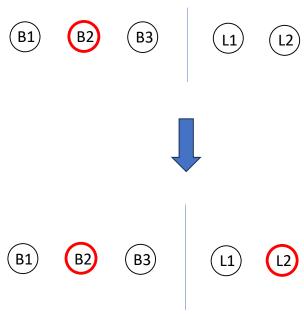
If Duplicator could not duplicate, she would lose!



Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game

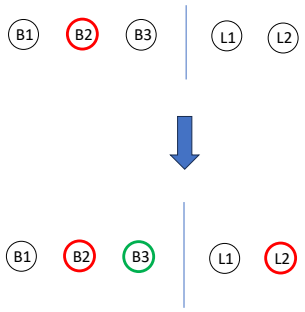
If Duplicator could not duplicate, she would lose!



Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game

If Duplicator could not duplicate, she would lose!

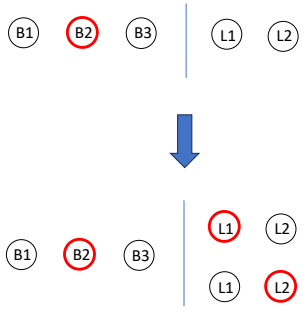


Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game.

Illustrative MS game on Linear Orders

- Pictorially we place smaller elements to the left and bigger elements to the right.
- Two sets of pebbled elements are partially isomorphic if they are in the same left-to-right order.
- Suppose we are playing a 2-round MS game



MS Games and the Oblivious Strategy

Note that in an MS game, it never hurts Duplicator to make as many copies of all structures as she needs and play all possible moves.

Such a strategy is always optimal, and we call it the "Oblivious Strategy".

Equivalence Theorem for EF Games

Equivalence Theorem for Ehrenfeucht-Fraïssé Games: Spoiler wins the r-round EF game on structures A, B iff there is a FO sentence σ with quantifier rank r such that $A \models \sigma$ while $B \models \neg\sigma$.

Equivalence Theorem for Ehrenfeucht-Fraïssé Games (Alternative Formulation): A property P can be described by a FO sentence having quantifier rank r iff Spoiler can win an EF game of r rounds on every pair A, B of structures, with A satisfying P and B not satisfying P.

Equivalence Theorem for MS Games

Equivalence Theorem for Multi-Structural Games: Spoiler wins the r-round MS game on sets $\mathcal{A}$ and $\mathcal{B}$ of structures iff there is a FO sentence σ with r quantifiers such that $A \models \sigma$ for every $A \in \mathcal{A}$, while $B \models \neg\sigma$ for every $B \in \mathcal{B}$.

Equivalence Theorem for Multi-Structural Games (Alternative Formulation): A property P can be described by a FO sentence with r quantifiers iff Spoiler can win the multi-structural game of r rounds on sets $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ of structures, where $\boldsymbol{\mathcal{A}}$ consists of all structures satisfying P and $\boldsymbol{\mathcal{B}}$ consists of all structures that don't satisfy P.

Important Nuance

The proof of the Equivalence Theorem shows that if Spoiler has a winning strategy, then Spoiler playing on the **left** corresponds to there being an **existential quantifier (∃)** in the separating sentence σ, while Spoiler playing on the **right** corresponds to there being a **universal quantifier (∀)** in σ.

Why Use Quantifier Number (QN) rather than Quantifier Rank (QR)?

In his 1981 paper, Neil provided the following construction showing that **every property of n-vertex ordered graphs can be expressed with a sentence having quantifier rank $\boldsymbol{\log n + 3}$**:

Let $\eta_i(x)$ be the formula that says x is the ith element in the n-vertex ordered graph G. $\eta_i(x)$ can be written as a formula, with one free variable, having at most quantifier rank $\log n + 1$. (That this can be done is an easy consequence of a result of Rosenstein's that we shall mention later.)

Let $E^{ij}(x, y)$ be a shorthand way of writing $E(x, y)$ if there is an edge between the ith and jth vertices in the ordered graph G, and $\neg E(x, y)$ if there is no such edge.

Why Use QN rather than QR?

The graph G can be completely described by the following sentence:

$$\sigma_G = \bigwedge_{1 \le i,j \le n} \exists x \exists y (\eta_i(x) \wedge \eta_j(y) \wedge E^{ij}(x, y)).$$

Since η_i, η_j have QR at most $\log n + 1$, σ_G has QR at most $\log n + 3$.

For a property P of n-vertex ordered graphs, let $\{G_i\}$ denote the family of all (n-vertex) graphs satisfying P with associated sentences σ_{G_i}. Then a sentence expressing P is just

$$\Sigma_P = \bigvee_i \sigma_{G_i}$$

which also has QR $\log n + 3$ [though QN may be huge! The above construction uses more than 2^{n^2} quantifiers!].

Why Use QN rather than QR?

In the case, of QN, it is easy to see that ***n* quantifiers** are sufficient to describe every property of n-vertex ordered graphs. Every n-vertex graph can be defined by the sentence

$$\Sigma_G = \exists x_1 \exists x_2 \cdots \exists x_n \underbrace{(x_1 < x_2 < \cdots < x_n \bigwedge_{1 \le i,j \le n} E^{ij}(x_i, x_j))}_{\sigma_G}.$$

So, for any property P of n-vertex ordered graphs, again with instances $\{G_i\}$, we can write:

$$\Sigma_P = \exists x_1 \exists x_2 \cdots \exists x_n \bigvee_i \sigma_{G_i}$$

which also has n quantifiers.

UPSHOT: There is a lot more leeway between $\log n + 1$ and n for QN, than between $\log n + 1$ and $\log n + 3$ for QR!!

Our Problems

In our first paper about the MS game, we showed how many quantifiers were necessary to distinguish **linear orders** of different sizes.

The next interesting ordered structures are the n-bit **binary strings**. We explored questions such as how many quantifiers are needed to distinguish one binary string from all others, or to distinguish one set of binary strings from its complement?

Setup

For the purpose of this talk, an **n-bit string** is a set of n elements equipped with

- a total ordering relation $<(,)$
- $\boldsymbol{min}$ and $\boldsymbol{max}$ constants that identify the 1st and last elements of the string
- a unary $1()$ relation that returns True iff a given element in the string is a 1.

Preliminary Observations

Observation 1: Since a Boolean function on n bits, $f: \{0,1\}^n \rightarrow \{0,1\}$, defines two complementary sets: $f^{-1}(0)$, $f^{-1}(1)$, the number of quantifiers needed to express an arbitrary Boolean function on n bits is the same as the number of quantifiers needed to express an arbitrary set of n-bit strings.

By **expressing a set of n-bit strings** we mean finding a sentence that is true for the given set of n-bit strings and false for the complementary set of n-bit strings.

Preliminary Observations

Observation 2: Analogous to our observation about n-vertex graphs, every set S of n-bit strings can be described via a sentence having n existential quantifiers.

Preliminary Observations

Observation 2: Analogous to our observation about n-vertex graphs, every set S of n-bit strings can be described via a sentence having n existential quantifiers.

The sentence looks like this:

$\exists x_1 \cdots \exists x_n((x_1 < x_2 < \cdots < x_n) \wedge$ "There exists n elements in increasing order"

Preliminary Observations

Observation 2: Analogous to our observation about n-vertex graphs, every set S of n-bit strings can be described via a sentence having n existential quantifiers.

The sentence looks like this:

$$S = \{10\cdots 0, \ldots, 01\cdots 0\} \qquad \exists x_1 \cdots \exists x_n((x_1 < x_2 < \cdots < x_n) \wedge \big((1(x_1) \wedge \neg 1(x_2) \wedge \cdots \wedge \neg 1(x_n)\big) \vee \cdots \vee \big(\neg 1(x_1) \wedge 1(x_2) \wedge \cdots \wedge \neg 1(x_n)\big))$$

"There exists n elements in increasing order"

One line for each of the strings in the set.

Quantifier Complexity of Boolean Functions: Main Results

Note. In everything that follows, $\log(n)$ denotes the base-2 logarithm of n.

► **Theorem A.** *Given an arbitrary $\varepsilon > 0$, every Boolean function on n-bit strings can be defined by a FO sentence having $(1+\varepsilon)\frac{n}{\log(n)} + O_\varepsilon(1)$ quantifiers, where the $O_\varepsilon(1)$ additive term depends only on ε and not n. Moreover, there are Boolean functions on n-bit strings that require $\frac{n}{\log(n)} + O(1)$ quantifiers to define.*

Quantifier Complexity of Boolean Functions: Main Results

Definition. Say that a family, $\{f_n\}_{n=1}^{\infty}$, of Boolean functions on n-bit strings is **sparse** if the cardinality of the set of strings mapping to 1 or to 0 under each f_n is polynomial in n.

Quantifier Complexity of Boolean Functions: Main Results

Definition. Say that a family, $\{f_n\}_{n=1}^{\infty}$, of Boolean functions on n-bit strings is **sparse** if the cardinality of the set of strings mapping to 1 or to 0 under each f_n is polynomial in n.

► **Theorem B.** *Given an arbitrary $\varepsilon > 0$, and a sparse family, $\{f_n\}_{n=1}^{\infty}$, of Boolean functions on n-bit strings, each function f_n can be defined by a FO sentence having $(1+\varepsilon)\log(n)+O_\varepsilon(1)$ quantifiers, where the $O_\varepsilon(1)$ additive term depends only on ε and not n or the choice of sparse family. Moreover, there are sparse families of Boolean functions on n-bit strings, the functions of which require $\log(n)$ quantifiers to define.*

First Results

- Since Rosenstein's work on linear orders in 1981, it has been known that quantifier rank (e.g., quantifier nesting depth) $\log n + 1$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from linear orders of size $> n$.

First Results

- Since Rosenstein's work on linear orders in 1981, it has been known that quantifier rank (e.g., quantifier nesting depth) $\log n + 1$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from linear orders of size $> n$.
- In a paper from LICS 2021 we were able to show that, similarly, for QN, $\log n + O(1)$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from all larger linear orders.

First Results

- Since Rosenstein's work on linear orders in 1981, it has been known that quantifier rank (e.g., quantifier nesting depth) $\log n + 1$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from linear orders of size $> n$.

- In a paper from LICS 2021 we were able to show that, similarly, for QN, $\log n + O(1)$ is both necessary and sufficient to distinguish linear orders of size $\leq n$ from all larger linear orders.

- A key fact that I shall use, but not prove, is that the sentences establishing the minimal QN for linear orders can be chosen to have **strictly alternating quantifier signatures**.

The Concept of Parallel Play

- Suppose that after k rounds of an r round MS game we can partition the pebbled structures as follows:

$$\mathcal{A}_1^k \cong \mathcal{B}_1^k$$

$$\mathcal{A}_2^k \cong \mathcal{B}_2^k$$

The Concept of Parallel Play

- Suppose that after k rounds of an r round MS game we can partition the pebbled structures as follows:

$$\mathcal{A}_1^k \cong \mathcal{B}_1^k$$

$$\mathcal{A}_2^k \cong \mathcal{B}_2^k$$

- Now imagine the remainder of the game being played as two separate **parallel games**.

- If Spoiler can win each of the parallel games by **playing on the same side of the board** in each of the remaining r-k rounds, then he can win the original game.

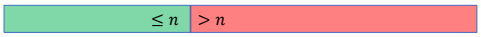
A Result about Linear Orders

In our distinguishing strings (a.k.a. characterizing Boolean functions) work we repeatedly leverage a variant of our LICS 2021 result about linear orders.

We need to be able to say that “between element x and element y in a string there is a linear order of precisely length n”.

A Result about Linear Orders

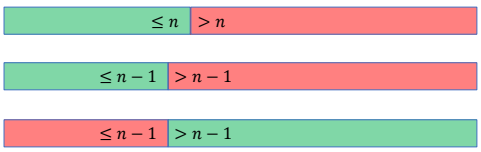
Remember, we know how to distinguish linear orders of sizes $\leq n$ from linear orders of sizes $> n$ with a logarithmic number of quantifiers.



σ_n

A Result about Linear Orders

Remember, we know how to distinguish linear orders of sizes $\leq n$ from linear orders of sizes $> n$ with a logarithmic number of quantifiers.



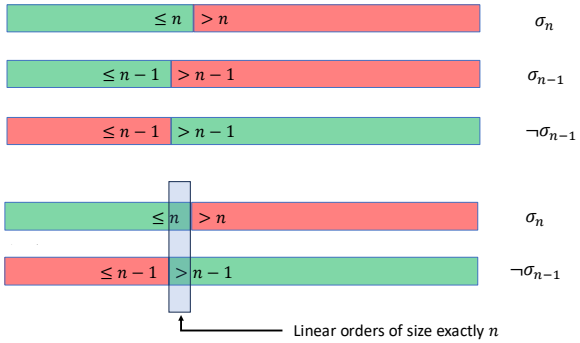
σ_n

σ_{n-1}

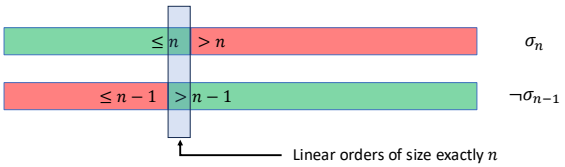
$\neg\sigma_{n-1}$

A Result about Linear Orders

Remember, we know how to distinguish linear orders of sizes $\leq n$ from linear orders of sizes $> n$ with a logarithmic number of quantifiers.



A Result about Linear Orders

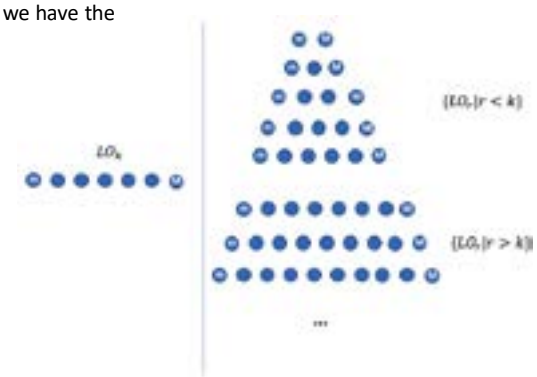


Taking the conjunction, $\sigma_n \wedge \neg\sigma_{n-1}$ we can distinguish the linear order of size n from all other linear orders with **twice this logarithmic number of quantifiers**.

But we can do better! In other words, we can say the same thing with fewer quantifiers.

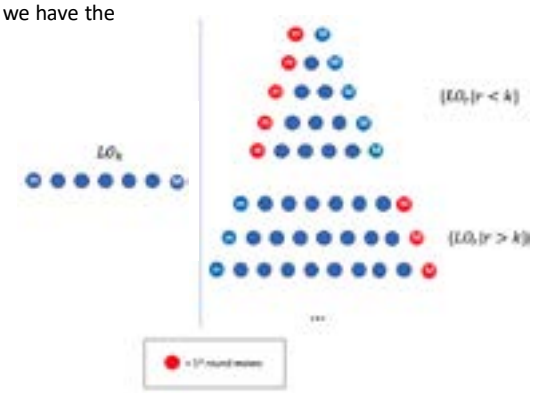
A Result about Linear Orders

From a game point of view, we have the following setup:



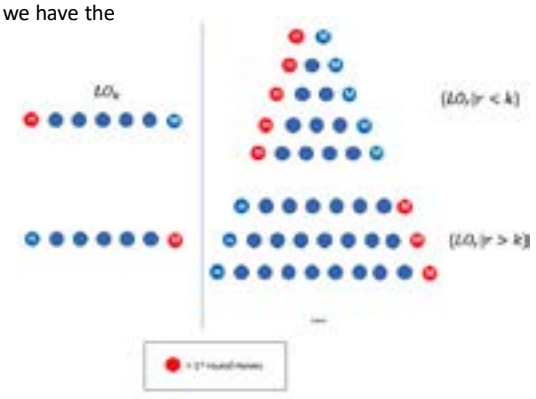
A Result about Linear Orders

From a game point of view, we have the following setup:



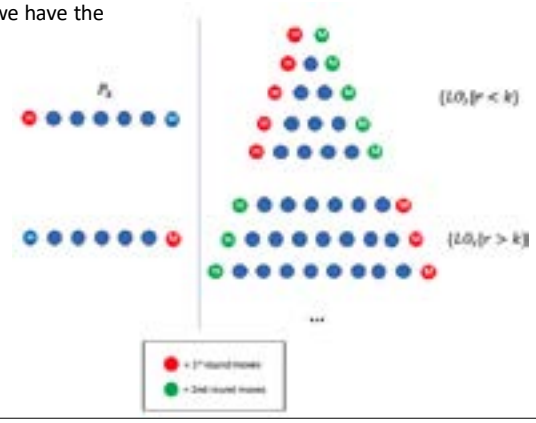
A Result about Linear Orders

From a game point of view, we have the following setup:



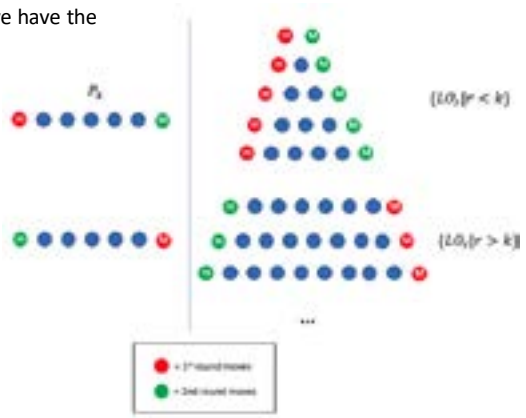
A Result about Linear Orders

From a game point of view, we have the following setup:



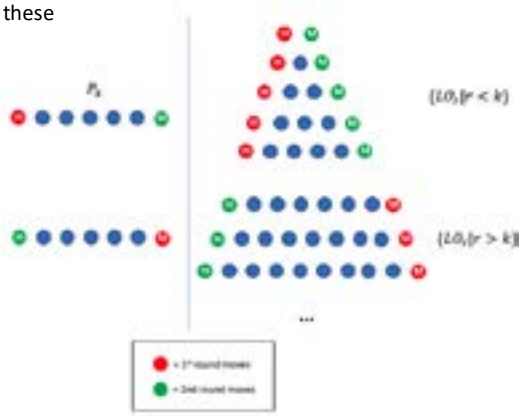
A Result about Linear Orders

From a game point of view, we have the following setup:



A Result about Linear Orders

- Spoiler is almost ready to play these two games in parallel…

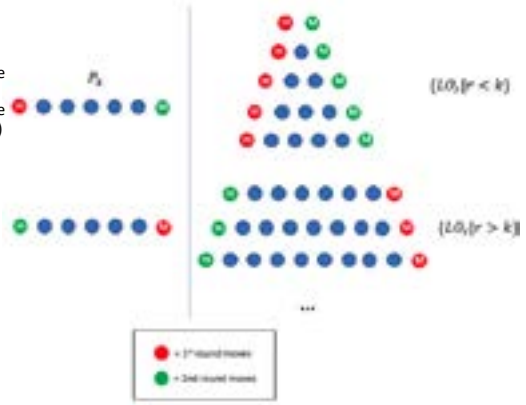


A Result about Linear Orders

- Spoiler uses the following **key fact**:
- The sentences:

  i. Distinguishing the linear order of size n from larger linear orders (σ_n), and
  ii. Distinguishing the linear order of size n from smaller linear orders ($\neg\sigma_{n-1}$)

  both have alternating quantifier signatures, with σ_n possibly having one additional quantifier.



A Result about Linear Orders

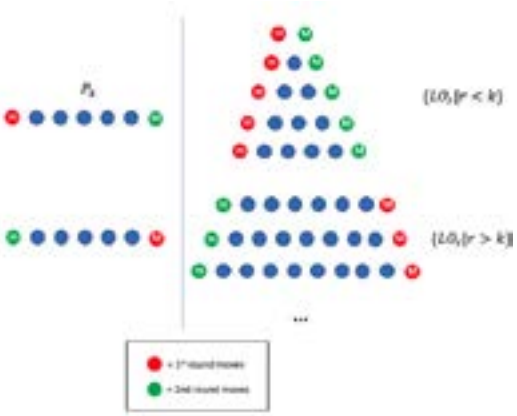
- Spoiler uses the following **key fact**:
- The sentences:

  i. Distinguishing the linear order of size n from larger linear orders (σ_n), and
  ii. Distinguishing the linear order of size n from smaller linear orders ($\neg\sigma_{n-1}$)

  both have alternating quantifier signatures, with σ_n possibly having one additional quantifier.

  Thus, tacking on at most an extra (dummy) quantifier at the beginning or end of each they can be given the same quantifier signature.

- This allows for parallel play, still with log(n) + O(1) quantifiers.

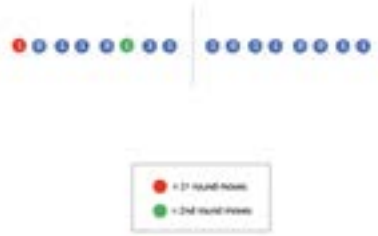


Distinguishing Strings with Few Quantifiers

- How about distinguishing two different **strings** of length n?
- It turns out that we can turn this game into the same parallel play linear order game!

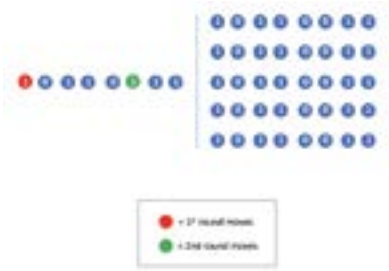
Distinguishing Strings with Few Quantifiers

- How about distinguishing two different **strings** of length n?
- It turns out that we can turn this game into the same parallel play linear order game!
- Spoiler plays his first two moves on the left hand (∃) side. He places his 1st pebble on the 1st element and his second pebble on the first point where the two strings disagree.



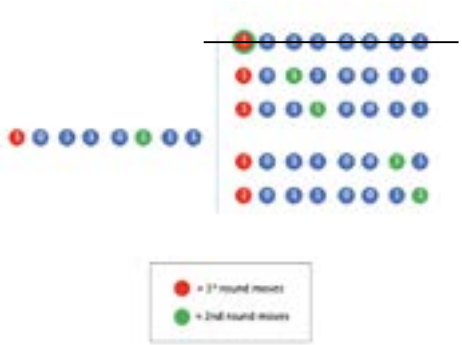
Distinguishing Strings with Few Quantifiers

- How about distinguishing two different **strings** of length n?
- It turns out that we can turn this game into the same parallel play linear order game!
- Spoiler plays his first two moves on the left hand (∃) side. He places his 1st pebble on the 1st element and his second pebble on the first point where the two strings disagree.
- Duplicator must also play her 1st pebble on the 1st element. She will play her 2nd pebble in all possible ways.



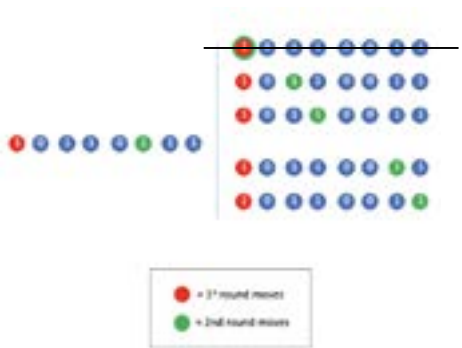
Distinguishing Strings with Few Quantifiers

- How about distinguishing two different **strings** of length n?
- It turns out that we can turn this game into the same parallel play linear order game!
- Spoiler plays his first two moves on the left hand (∃) side. He places his 1st pebble on the 1st element and his second pebble on the first point where the two strings disagree.
- Duplicator must also play her 1st pebble on the 1st element. She will play her 2nd pebble in all possible ways.



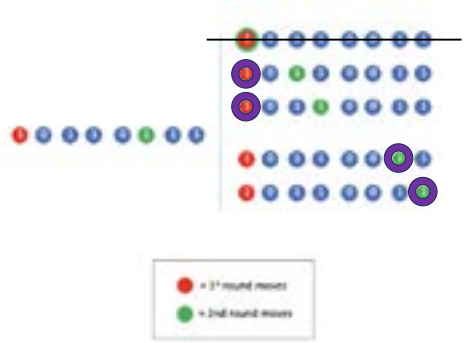
Distinguishing Strings with Few Quantifiers

- **Question**: In one more round, how can Spoiler play in such a way to turn the game into a Parallel Play situation?



Distinguishing Strings with Few Quantifiers

- Spoiler can play his 3rd round moves on top of the red pebbles for the first half of the strings on the right and on top of the green pebbles for the second half of the strings on the right.
- Duplicator just has two meaningful ways to play on the left.
- The game is then reduced to a pair of parallel play linear order games.



Distinguishing Strings with Few Quantifiers

- This argument shows that one can distinguish every two strings of length n from one another with a sentence having $\log n + O(1)$ quantifiers.
- It is also straightforward to show that this many quantifiers may also be necessary.

Distinguishing Strings with Few Quantifiers

- This argument shows that one can distinguish every two strings of length n from one another with a sentence having $\log n + O(1)$ quantifiers.
- It is also straightforward to show that this many quantifiers may also be necessary.
- In fact, it is possible to separate one n-bit string from **all other** n-bit strings with $\log n + O(1)$ quantifiers.
- And recall, we can separate any sparse set of n-bit strings from its complementary set with just $(1 + \epsilon) \log n + O(1)$ quantifiers for arbitrarily small ϵ.

Separating Two Arbitrary Sets of Strings

Recalling our other main result on separating strings:

▸ **Theorem A.** *Given an arbitrary $c > 0$, every Boolean function on n-bit strings can be defined by a FO sentence having $(1+c)\frac{n}{\log(n)} + O_c(1)$ quantifiers, where the $O_c(1)$ additive term depends only on c and not n. Moreover, there are Boolean functions on n-bit strings that require $\frac{n}{\log(n)} + O(1)$ quantifiers to define.*

The Road Ahead to Realize Neil Immerman's Vision

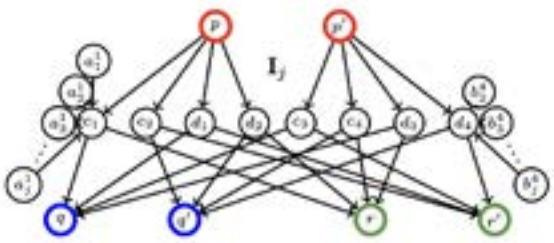
- We've devised a game that characterizes not just QN, but simultaneously QN and number of variables (VN). We are interested in studying the tradeoff between QN and VN for various properties (extending work of Grohe and Schweikardt). Connected to time vs. space tradeoffs.
- Characterize QN for specific string and graph properties.
- **MILLION $$ PROBLEM:** Can we find a concrete property (that is not known to be outside of NL) that requires more than $O(\log n)$ quantifiers? Such a property would necessarily lie outside of NL by Neil's inclusion relation. We have some ideas….

More Recent Results: How Hard is it to decide WHO WINS an MS-Game?

- In 1998 Pazolli showed that, given two **unordered** structures $\{A, B\}$, it is **NP-hard** to decide **WHO WINS** an **Ehrefeucht-Fraïssé game** on these structures.
- In a paper we have just submitted to CSL, we have shown the analogous question for the **MS-game** is **PSPACE-hard**. The reduction is from the PSPACE-hardness of approximating Quantified SAT to within a particular constant c, for $0 < c < 1$.
- The analogous questions for **ordered** structures remain open and appear to be considerably harder (but enticing!).

A key Gadget in these Hardness Arguments

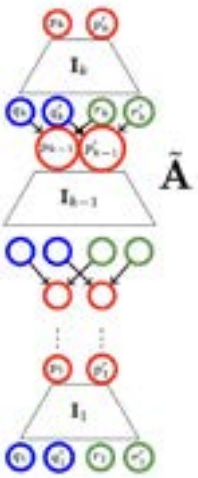
Gadgets by Pazolli and us are closely related to one devised by Cai, Fürer and Immerman to show that Fixed Point Logic + Counting does NOT capture PTIME:



A Stack of Gadgets

Both the Pazolli argument and our arguments (including a refinement of Pazolli's original argument) use a stack of these gadgets.

Can we create software that will verify the game-based arguments we make about such complicated gadgets/stacks of gadgets?

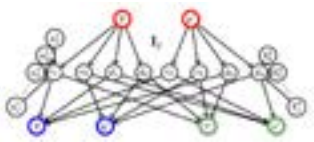


Software to help with the Analysis of the Games – the next frontier?

- We have developed software that can analyze EF-games on reasonably complicated graphs, demonstrating optimal Spoiler and Duplicator play.
- The software works for arbitrary ordered or unordered graphs. Graphs can be directed, undirected, contain loops, constants, and so on.
- Does not yet utilize symmetries (elements with the same k-types) to reduce search trees. Thus, look-ahead analysis only terminates for moderate-sized boards and modest numbers of rounds.

Software to help with the Analysis of the Games – the next frontier?

- We have a notion of an "iconized group", capturing a subgraph that can be replicated throughout a bigger graph, but we cannot yet quite add the CFI-like gadgets.
- These are really iconized groups with specialized input and output "terminals"
- Once the software is a bit more robust, we will extend i to MS-games, though creating an effective user interface, allowing for management of all the bifurcating boards, is challenging.



Software Demo

Thank you!

Hybrid-Dynamic Ehrenfeucht-Fraïssé Games

G. Badia, D. Gaina, A. Knapp, *T. Kowalski*, M. Wirsing

U. of Queensland, Kyushu U., U. of Augsburg,
Jagiellonian U., La Trobe U. Ludwig-Maximilian U.

Logic, Algebra and Category Theory
1 October 2025, KyuDai

Introduction

- Idea: take a hodgepodge of existing results and make a systematic sense of them.
- Ingredients:
  - Some results on bisimulations in hybrid and dynamic logics, for example [Areces et al., 2001, Hodkinson and Tahiri, 2010].
  - Classical first-order stuff: Ehrenfeucht-Fraïssé games, partial isomorphisms, back-and-forth.
  - Institution theory framework.
- Recipe:
  - Mix, shake well, simmer in a pot (see [Badia et al., 2025]) long enough for Daniel to lose patience.
  - Rewrite everything in institution theory framework, get some new results on the way.

Signatures

- All signatures will be of the form $(\Sigma, \mathtt{Prop})$, where $\Sigma = (F, P)$ is a single-sorted first-order signature consisting of
  - a set of constants F (nominals), and
  - a set of binary relation symbols P (accessibility),

  and $\mathtt{Prop}$ is a set of propositional symbols.
- We let Δ range over signatures of the form $(\Sigma, \mathtt{Prop})$. When necessary, we will use indexing in a natural way.
- A signature morphism $\chi : \Delta_1 \to \Delta_2$ consists of a first-order signature morphism $\chi : \Sigma_1 \to \Sigma_2$ and a function $\chi : \mathtt{Prop}_1 \to \mathtt{Prop}_2$.
- The extension of Δ by a fresh nominal k is denoted $\Delta[k]$ yielding the inclusion $\Delta \hookrightarrow \Delta[k]$.
- We denote by $\mathsf{Sig}^{\mathsf{HDPL}}$ the category of HDPL signatures.

Models

- Models over a signature Δ are standard Kripke structures $\mathfrak{M} = (W, M)$, where:
  - W is a first-order structure over Σ, that is, $W = (|\mathfrak{M}|, (\lambda^W)_{\lambda \in P})$. So $|\mathfrak{M}|$ is the set of worlds, and λ^W are accessibility relations.
  - M is a propositional valuation, that is, a map $M : |\mathfrak{M}| \to |\mathsf{Mod}^{\mathsf{PL}}(\mathtt{Prop})|$, where $\mathsf{Mod}^{\mathsf{PL}}$ stands for power-set.
- Convention: $\mathfrak{M}$ and $\mathfrak{N}$ range over Kripke structures of the form (W, M) and (V, N).
- Δ-homomorphism $h : \mathfrak{M} \to \mathfrak{N}$ between two Kripke structures $\mathfrak{M}$ and $\mathfrak{N}$ is a first-order homomorphism $h : W \to V$ such that $h(M(w)) \subseteq N(h(w))$ for all states $w \in |\mathfrak{M}|$, that is, h preserves truth of propositions.
- Δ-homomorphisms form a category $\mathsf{Mod}^{\mathsf{HDPL}}(\Delta)$.

Actions / Programs

The set of actions $\mathcal{A}^{\mathsf{HDPL}}(\Delta)$ over a signature Δ is defined by the following grammar:

$$\mathfrak{a} ::= \lambda \mid \mathfrak{a} \cup \mathfrak{a} \mid \mathfrak{a} \fatsemi \mathfrak{a} \mid \mathfrak{a}^* \ ,$$

where λ is a binary relation on nominals. Actions are interpreted in Kripke structures (W, M) as accessibility relations, as follows:

- $\lambda^{\mathfrak{M}} = \lambda^W$ for all binary relations λ in Δ,
- $(\mathfrak{a}_1 \cup \mathfrak{a}_2)^{\mathfrak{M}} = \mathfrak{a}_1^{\mathfrak{M}} \cup \mathfrak{a}_2^{\mathfrak{M}}$ (union),
- $(\mathfrak{a}_1 \fatsemi \mathfrak{a}_2)^{\mathfrak{M}} = \mathfrak{a}_1^{\mathfrak{M}} \fatsemi \mathfrak{a}_2^{\mathfrak{M}}$ (composition),
- $(\mathfrak{a}^*)^{\mathfrak{M}} = (\mathfrak{a}^{\mathfrak{M}})^*$ (reflexive & transitive closure).

Sentences

The set of sentences $\mathsf{Sen}^{\mathsf{HDPL}}(\Delta)$ over a signature Δ is defined by:

$$\phi ::= p \mid k \mid \bigwedge \Phi \mid \neg\phi \mid \langle \mathfrak{a} \rangle \phi \mid @_k \, \phi \mid \downarrow x \cdot \phi_x \mid \exists x \cdot \phi_x \ ,$$

where (i) p is a propositional symbol, (ii) k is a nominal, (iii) Φ is a finite set of sentences over Δ, (iv) x is a variable for Δ, (v) $\mathfrak{a}$ is an action over Δ, (vi) $\phi_x \in \mathsf{Sen}^{\mathsf{HDPL}}(\Delta[x])$.

- $\langle \mathfrak{a} \rangle \phi$ is read ϕ holds after a run of $\mathfrak{a}$ (possibility),
- $@_k \, \phi$ is read ϕ holds at state k (retrieve),
- $\downarrow x \cdot \phi_x$ is read ϕ_x holds with the current state set to x (store).

Usual abbreviations are in force: $\bigvee \Phi$ for $\neg(\bigwedge_{\phi \in \Phi} \neg\phi)$, $[\mathfrak{a}]\phi$ for $\neg\langle \mathfrak{a} \rangle \neg\phi$, and $\forall x \cdot \phi_x$ for $\neg\exists x \cdot \neg\phi_x$.

Local satisfaction relation

Satisfaction of a sentence ϕ at a world $w \in |\mathfrak{M}|$ in a model $\mathfrak{M} = (W, M)$ over a signature Δ, is defined by induction:

- $(\mathfrak{M}, w) \models p$ if $p \in M(w)$;
- $(\mathfrak{M}, w) \models k$ if $w = k^{\mathfrak{M}}$
- $(\mathfrak{M}, w) \models \bigwedge \Phi$ if $(\mathfrak{M}, w) \models \phi$ for all $\phi \in \Phi$;
- $(\mathfrak{M}, w) \models \neg\phi$ if $(\mathfrak{M}, w) \not\models \phi$;
- $(\mathfrak{M}, w) \models @_k \phi$ if $(\mathfrak{M}, k^{\mathfrak{M}}) \models \phi$;
- $(\mathfrak{M}, w) \models \langle \mathfrak{a} \rangle \phi$ if $(\mathfrak{M}, v) \models \phi$ for some $v \in \mathfrak{a}^{\mathfrak{M}}(w) := \{w' \in |\mathfrak{M}| \mid (w, w') \in \mathfrak{a}^{\mathfrak{M}}\}$;
- $(\mathfrak{M}, w) \models \downarrow x \cdot \phi_x$ if $(\mathfrak{M}^{x \leftarrow w}, w) \models \phi_x$, where $\mathfrak{M}^{x \leftarrow w}$ is the expansion of $\mathfrak{M}$ to $\Delta[x]$ interpreting x as w;
- $(\mathfrak{M}, w) \models \exists x \cdot \phi_x$ if $(\mathfrak{M}^{x \leftarrow v}, w) \models \phi_x$ for some $v \in |\mathfrak{M}|$.

Local satisfaction condition

We call the pair $(\mathfrak{M}, w)$ a pointed model, and w the current state.

Theorem (Local satisfaction condition)

For all signature morphisms $\chi : \Delta_1 \to \Delta_2$, all Δ_2-models $\mathfrak{M}$, all states $w \in |\mathfrak{M}|$, all Δ_1-sentences ϕ, we have

$$(\mathfrak{M}, w) \models \chi(\phi) \text{ iff } (\mathfrak{M}|\chi, w) \models \phi \ .$$

where $\mathfrak{M}|\chi$ is the χ-reduct of $\mathfrak{M}$.

- Technically, this shows that HDPL is a stratified institution (see [Diaconescu, 2017, Găină, 2020]).
- In practice, it means that signature morphisms are very decent translations.
- Folklore in institution theory (see [Diaconescu, 2016]).

Framework for fragments

HDPL is too powerful, so we want to consider fragments. Let $\mathcal{L} = (\mathsf{Sig}^{\mathsf{HDPL}}, \mathsf{Sen}^{\mathcal{L}}, \mathsf{Mod}^{\mathsf{HDPL}}, \models)$ of HDPL be an arbitrary fragment of HDPL closed under Boolean connectives. Boolean closure in not needed for the games, but it is for game sentences.

In practice, any $\mathcal{L}$ is obtained from HDPL by discarding:

- some constructors for actions from the grammar which defines actions in HDPL, and/or
- some constructors for sentences from the grammar which defines sentences in HDPL.

Two extreme cases:

- drop the action part, get hybrid propositional logic (HPL),
- drop the hybrid part, get propositional dynamic logic (PDL).

Notational conventions

- Let $\mathcal{O} \subseteq \{\Diamond, @, \downarrow, \exists\}$ be the subset of the sentence constructors which belong to $\mathcal{L}$.
  - $\Diamond \in \mathcal{O}$ means that $\mathcal{L}$ is closed under possibility over actions, but no assumption is made concerning the existence of action constructors. One or more constructors for actions can be discarded from $\mathcal{L}$.
- Let $\mathsf{Mod}_p(\Delta) = \{(\mathfrak{M}, w) \mid \mathfrak{M} \in |\mathsf{Mod}(\Delta)| \text{ and } w \in |\mathfrak{M}|\}$ be the class of pointed Δ-models.
- Let $\mathsf{Mod}_p(\Delta, \phi) = \{(\mathfrak{M}, w) \in \mathsf{Mod}_p(\Delta) \mid (\mathfrak{M}, w) \models \phi\}$ be the class of pointed Δ-models which satisfy a sentence ϕ.
- Let $\mathsf{Sen}_b : \mathsf{Sig} \to \mathbb{S}\mathrm{et}$ be the subfunctor of $\mathsf{Sen} : \mathsf{Sig} \to \mathbb{S}\mathrm{et}$ which maps each signature $\Delta = (\Sigma, \mathtt{Prop})$ to the set of basic (or atomic) sentences $F \cup \mathtt{Prop}$.

Elementary equivalence

Definition

Let Δ be a signature.

- Two pointed Δ-models $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ are $\mathcal{L}$-elementarily equivalent, in symbols, $(\mathfrak{M}, w) \equiv (\mathfrak{N}, v)$, when $(\mathfrak{M}, w) \models \phi$ iff $(\mathfrak{N}, v) \models \phi$, for all Δ-sentences $\phi \in \mathsf{Sen}^{\mathcal{L}}(\Delta)$.
- Two Δ-sentences ϕ_1 and ϕ_2 are semantically equivalent if they are satisfied by the same pointed models, that is, $\mathsf{Mod}_p(\Delta, \phi_1) = \mathsf{Mod}_p(\Delta, \phi_2)$.

- $\mathcal{L}$-elementary equivalence is an equivalence relation for any $\mathcal{L}$.
- Strength of $\equiv$ depends on $\mathcal{L}$.

Ehrenfeucht-Fraïssé games in our framework

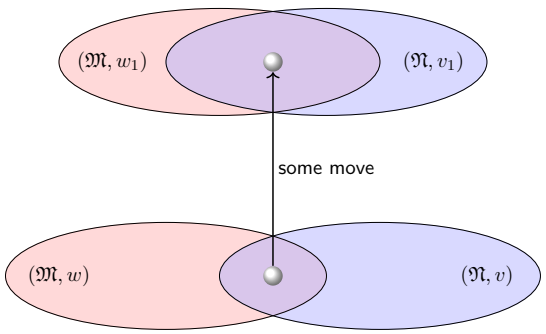
Idea based on [Găină and Kowalski, 2020].

- A game is a collection of all its plays over a set of gameboard trees.
- A play of the game starts with two pointed models $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ defined over the same signature Δ, and a gameboard tree tr with $root(tr) = \Delta$.
- $\exists$loise loses if the following game property is not satisfied:

$$(\mathfrak{M}, w) \models \phi \text{ iff } (\mathfrak{N}, v) \models \phi \quad \text{for all } \phi \in \mathsf{Sen}_b(\Delta).$$

- Otherwise, the play can continue and $\forall$belard can move one of the pointed models along an edge of the gameboard tree. Wlog, $\forall$belard picks up the first pointed model $(\mathfrak{M}, w)$.
- A winning strategy for a player is a sequence of moves that ensure that the player wins any play of the game.

Idea illustrated

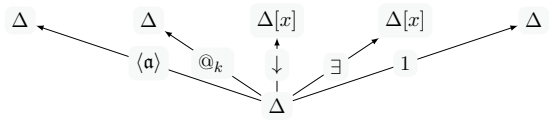


Gameboard trees

A gameboard tree describes the possible moves in the play of a game. The edges are labelled to account for different kinds of "quantifiers": $\langle \mathfrak{a} \rangle$, $@_k$, $\downarrow$, $\exists$.

- Nodes are labelled by signatures
- Edges are labelled by sentence operators; they are uniquely identified by source and label.

Here are all possible labels for HDPL. For a particular $\mathcal{L}$ there may be fewer possible labels. But the idle label 1 always occurs for technical reasons.



Labels

$\Diamond \in \mathcal{O}$: A pair consisting of the identity signature morphism $1_\Delta : \Delta \to \Delta$ and the modal operator $\langle \mathfrak{a} \rangle$.

$@ \in \mathcal{O}$: A pair consisting of the identity signature morphism $1_\Delta : \Delta \to \Delta$ and the operator retrieve $@_k$.

$\downarrow \in \mathcal{O}$: A pair consisting of a signature inclusion $\Delta \longrightarrow \Delta[x]$ and the operator store $\downarrow x$.

$\exists \in \mathcal{O}$: A pair consisting of a signature inclusion $\Delta \longrightarrow \Delta[x]$ and the first-order quantifier $\exists x$.

idle: The identity signature morphism $1_\Delta : \Delta \to \Delta$. The idle edge serves to construct complex gameboard trees from simpler components and to define game sentences that represent conjunctions.

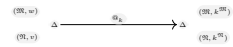
Moves

$\Diamond \in \mathcal{O}$:

A move along $\Delta \xrightarrow{\langle \mathfrak{a} \rangle} \Delta$. The next state w_1 chosen by ∀belard is accessible from w via $\mathfrak{a}^{\mathfrak{M}}$. ∃loise needs to find a state v_1 accessible from v via $\mathfrak{a}^{\mathfrak{N}}$ such that for her new pointed model $(\mathfrak{N}, v_1)$ the game property still holds.

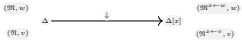
$@ \in \mathcal{O}$:

A move along an edge $\Delta \xrightarrow{@_k} \Delta$. ∀belard changes the current state w to $k^{\mathfrak{M}}$. The only possible choice for ∃loise is $(\mathfrak{N}, k^{\mathfrak{N}})$. If the game property still holds, the play continues with $(\mathfrak{M}, k^{\mathfrak{M}})$ and $(\mathfrak{N}, k^{\mathfrak{N}})$.



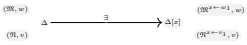
$\downarrow \in \mathcal{O}$:

A move along $\Delta \xrightarrow{\downarrow} \Delta[x]$. ∀belard gives name x to his current state. ∃loise can only respond by giving name x to her current state.



$\exists \in \mathcal{O}$:

A move along $\Delta \xrightarrow{\exists} \Delta[x]$. ∀belard gives name x to a new arbitrary state $w_1 \in |\mathfrak{M}|$ without changing his current state. ∃loise needs to match ∀belard's choice by giving name x to a state $v_1 \in |\mathfrak{N}|$.

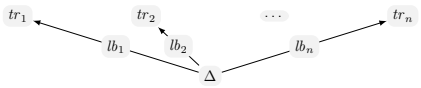


idle: $\Delta \xrightarrow{!} \Delta$, no change in models.

Game sentences

The set of game sentences Θ_{tr} over a gameboard tree tr with root labelled by a finite signature Δ, is defined by structural induction on gameboard trees:

- Base case ($tr = \Delta$). Put $\Theta_\Delta = \{\bigwedge_{\rho \in \mathsf{Sen}_b(\Delta)} \rho^{f(\rho)} \mid f : \mathsf{Sen}_b(\Delta) \to \{0,1\}\}$, where $\rho^0 = \rho$ and $\rho^1 = \neg\rho$. (All atomic "state descriptions")
- Inductive step ($tr = \Delta(\xrightarrow{lb_1} tr_1 \ldots \xrightarrow{lb_n} tr_n)$), as below:



- Fix an index $i \in \{1, \ldots, n\}$. Then define
  1. a subset $S_i \subseteq \mathcal{P}(\Theta_{tr_i})$ of the powerset of Θ_{tr_i}, and
  2. a Δ-sentence φ_Γ for each set of game sentences $\Gamma \in S_i$.

  Depending on the label lb_i, there are five cases:

Game sentences

which we will not go into. Except two:

- Possibility move ($\Delta \xrightarrow{\langle \mathfrak{a} \rangle} \Delta$): This case assumes $\Diamond \in \mathcal{O}$.
  1. $S_i \coloneqq \mathcal{P}(\Theta_{tr_i})$, and
  2. $\varphi_\Gamma \coloneqq (\bigwedge_{\gamma \in \Gamma} \langle \mathfrak{a} \rangle \gamma) \wedge ([\mathfrak{a}] \bigvee \Gamma)$ for all $\Gamma \in S_i$.
     (For any $\gamma \in \Gamma$ one can go somewhere where γ holds, and wherever one goes some γ will hold.)
- Existential q. move: ($\Delta \xrightarrow{\exists} \Delta[x]$): This case assumes $\exists \in \mathcal{O}$.
  1. $S_i \coloneqq \mathcal{P}(\Theta_{tr_i})$, and
  2. $\varphi_\Gamma \coloneqq (\bigwedge_{\gamma \in \Gamma} \exists x \cdot \gamma) \wedge (\forall x \cdot \bigvee \Gamma)$, for all $\Gamma \in S_i$.

Definition

The set of game sentences over tr is $\Theta_{tr} = \{\varphi_{\Gamma_1} \wedge \cdots \wedge \varphi_{\Gamma_n} \mid \Gamma_1 \in S_1, \ldots, \Gamma_n \in S_n\}$.

Fraïssé-Hintikka theorem

Theorem

Let Δ be a finite signature.

1. *For all pointed models $(\mathfrak{M}, w)$ defined over Δ, and all gameboard trees tr with $root(tr) = \Delta$, there exists a unique game sentence $\varphi \in \Theta_{tr}$ such that $(\mathfrak{M}, w) \models \varphi$.*
2. *For all pointed models $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ defined over Δ and all gameboard trees tr with $root(tr) = \Delta$, the following are equivalent:*
   - 2.1 $(\mathfrak{M}, w) \approx_{tr} (\mathfrak{N}, v)$
   - 2.2 *There exists a unique game sentence $\varphi \in \Theta_{tr}$ such that $(\mathfrak{M}, w) \models \varphi$ and $(\mathfrak{N}, v) \models \varphi$.*
3. *For each sentence ϕ defined over Δ, there exists a gameboard tree tr with $root(tr) = \Delta$ and a set of game sentences $\Psi_\phi \subseteq \Theta_{tr}$ such that $\phi \leftrightarrow \bigvee \Psi_\phi$ is a theorem of $\mathcal{L}$.*

Finite EF games: exactly what you would expect

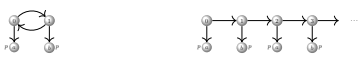
Corollary

Let $\mathfrak{M}$ and $\mathfrak{N}$ be two Kripke structures defined over a finite signature Δ. The following are equivalent:

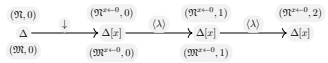
1. *$(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ are $\mathcal{L}$-elementarily equivalent. In symbols $(\mathfrak{M}, w) \equiv (\mathfrak{N}, v)$.*
2. *$\exists$loise has a winning strategy for the EF game starting with $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$, that is, $(\mathfrak{M}, w) \approx_{tr} (\mathfrak{N}, v)$ for all finite gameboard trees tr.*

Example 1

Let Δ be a signature with no nominals, one binary relation λ, and one propositional symbol p. Let $\mathfrak{M}$ and $\mathfrak{N}$ be the Δ-models shown below:



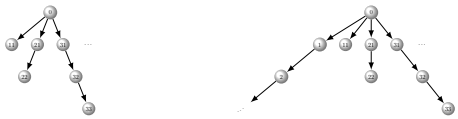
1. If $\mathcal{L}$ is PDL, then $(\mathfrak{M}, 0)$ and $(\mathfrak{N}, 0)$ are $\mathcal{L}$-elementarily equivalent.
2. If $\mathcal{L}$ is HPL, then $(\mathfrak{M}, 0) \not\approx_{tr} (\mathfrak{N}, 0)$ for any complete gameboard tree tr of height greater or equal than 3, as shown below:



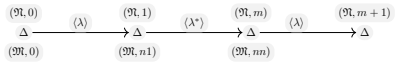
In the third round, $\exists$loise loses.

Example 2

Let Δ be the signature with one relation λ. Let $\mathfrak{M}$ be the model on the left, and $\mathfrak{N}$ the model on the right, with one infinite path.

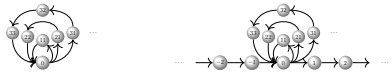


1. In all fragments of HPL, ∃loise has a winning strategy over finite gameboard trees starting with $(\mathfrak{M}, 0)$ and $(\mathfrak{N}, 0)$.
2. In any fragment of HDPL closed under the Booleans and actions, ∃loise loses the game played over $\Delta \xrightarrow{\langle\lambda\rangle} \Delta \xrightarrow{\langle\lambda^*\rangle} \Delta \xrightarrow{\langle\lambda\rangle} \Delta$ starting with $(\mathfrak{M}, 0)$ and $(\mathfrak{N}, 0)$:

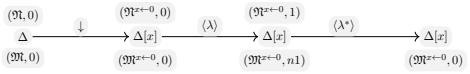


Example 3

Let Δ be a signature with one binary relation λ, and let $\mathfrak{M}$ and $\mathfrak{N}$ be model on the left and right, respectively.



1. $(\mathfrak{M}, 0) \equiv (\mathfrak{N}, 0)$ in any fragment $\mathcal{L}$ of HPL closed under the Booleans.
2. $(\mathfrak{M}, 0) \not\equiv (\mathfrak{N}, 0)$ in any fragment of HDPL closed under the Booleans, actions and store. For ∀belard can win in three steps:

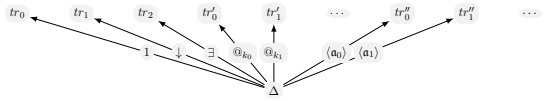


Interlude: a glossary of equivalences

	already occurred
$\cong$	isomorphism
$\equiv$	elementary equivalence
$\approx_{tr}$	∃loise has a winning strategy in a finite EF game
	to come shortly
$\approx_{\omega}$	∃loise has a winning strategy in a countable EF game
$\equiv_{\mathcal{I}}$	back-and-forth equivalence
$\equiv_{B}$	bisimulation equivalence

Gameboard trees

A countable gameboard tree tr is of countable height and countable width, and is defined recursively as follows. Start with:



where

- $F = \{k_i \mid i < \alpha\}$ is an enumeration of all Δ-nominals and α is the cardinal of $\mathsf{Sen}(\Delta)$,
- $\mathcal{A}(\Delta) = \{\mathfrak{a}_i \mid i < \alpha\}$ is an enumeration of all actions defined over Δ.

Extend so that each of tr_0, tr_1, tr_2, tr'_i and tr''_i are trees of the same form as tr.

ω-bisimulations

Definition

Let $\mathfrak{M}$ and $\mathfrak{N}$ be Kripke structures. A relation $B_\ell \subseteq (|\mathfrak{M}|^\ell \times |\mathfrak{M}|) \times (|\mathfrak{N}|^\ell \times |\mathfrak{N}|)$ is an ℓ-bisimulation from $\mathfrak{M}$ to $\mathfrak{N}$ if for all $(\overline{w}, w)\ B_\ell\ (\overline{v}, v)$ the following hold:

(prop) $p \in M(w)$ iff $p \in N(v)$ for all propositional symbols $p \in \mathtt{Prop}$;

(nom) $w = k^{\mathfrak{M}}$ iff $v = k^{\mathfrak{N}}$ for all nominals $k \in F$;

(wvar) $\overline{w}(j) = w$ iff $\overline{v}(j) = v$ for all $1 \leq j \leq \ell$;

(forth) if $\Diamond \in \mathcal{O}$ then for all actions $\mathfrak{a} \in \mathcal{A}(\Delta)$ and all states $w' \in \mathfrak{a}^{\mathfrak{M}}(w)$ there exists $v' \in \mathfrak{a}^{\mathfrak{N}}(v)$ such that $(\overline{w}, w')\ B_\ell\ (\overline{v}, v')$;

(back) if $\Diamond \in \mathcal{O}$ then for all actions $\mathfrak{a} \in \mathcal{A}(\Delta)$ and all states $v' \in \mathfrak{a}^{\mathfrak{N}}(v)$ there exists $w' \in \mathfrak{a}^{\mathfrak{M}}(w)$ such that $(\overline{w}, w')\ B_\ell\ (\overline{v}, v')$;

(atv) if $@ \in \mathcal{O}$ then $(\overline{w}, \overline{w}(j))\ B_\ell\ (\overline{v}, \overline{v}(j))$ for all $1 \leq j \leq \ell$;

(atn) if $@ \in \mathcal{O}$ then $(\overline{w}, k^{\mathfrak{M}})\ B_\ell\ (\overline{v}, k^{\mathfrak{N}})$ for all nominals $k \in F$;

An ω-bisimulation from $\mathfrak{M}$ to $\mathfrak{N}$ is a family of ℓ-bisimulations $B = (B_\ell)_{\ell \in \omega}$ from $\mathfrak{M}$ to $\mathfrak{N}$ such that for all natural numbers $\ell \in \omega$ and all tuples $(\overline{w}, w) \in |\mathfrak{M}|^\ell \times |\mathfrak{M}|$ and $(\overline{v}, v) \in |\mathfrak{N}|^\ell \times |\mathfrak{N}|$ the following conditions are satisfied:

(st) if $\downarrow \in \mathcal{O}$ and $(\overline{w}, w)\ B_\ell\ (\overline{v}, v)$ then $(\overline{w}\ w, w)\ B_{\ell+1}\ (\overline{v}\ v, v)$, where the juxtaposition stands for the concatenation of sequences; and

(ex) if $\exists \in \mathcal{O}$ and $(\overline{w}, w)\ B_\ell\ (\overline{v}, v)$ then:

(ex-f) for all $w' \in |\mathfrak{M}|$ there is $v' \in |\mathfrak{N}|$ such that $(\overline{w}\ w', w)\ B_{\ell+1}\ (\overline{v}\ v', v)$,

(ex-b) for all $v' \in |\mathfrak{N}|$ there is $w' \in |\mathfrak{M}|$ such that $(\overline{w}\ w', w)\ B_{\ell+1}\ (\overline{v}\ v', v)$.

ω-bisimilar models

Definition

Two pointed models $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ are ω-bisimilar if there exists an ω-bisimulation B from $\mathfrak{M}$ to $\mathfrak{N}$ such that $w\ B_0\ v$.

- Notation for ω-bisimilarity: $(\mathfrak{M}, w) \equiv_B (\mathfrak{N}, v)$.
- An ω-bisimulation is a relation between $|\mathfrak{M}|^*$ and $|\mathfrak{N}|^*$.
- Very roughly: ω-bisimulations are bisimulations with memory.

Lemma

Let B be an ω-bisimulation between $(\mathfrak{M}, \mu)$ and $(\mathfrak{N}, \nu)$ such that $(\mu, \mu')\ B_1\ (\nu, \nu')$ for some $(\mu, \mu') \in |\mathfrak{M}| \times |\mathfrak{M}|$ and $(\nu, \nu') \in |\mathfrak{N}| \times |\mathfrak{N}|$. Let $\chi : \Delta \to \Delta[x]$ be a signature extension with a variable x. Then $B^x = (B^x_\ell)_{\ell \in \omega}$ defined by

$$(\overline{w}, w)\ B^x_\ell\ (\overline{v}, v) \text{ iff } (\mu\, \overline{w}, w)\ B_{\ell+1}\ (\nu\, \overline{v}, v),$$

for all $\ell \in \omega$, all $(\overline{w}, w) \in |\mathfrak{M}|^\ell \times |\mathfrak{M}|$ and all $(\overline{v}, v) \in |\mathfrak{N}|^\ell \times |\mathfrak{N}|$,

is a bisimulation between $(\mathfrak{M}^{x \leftarrow \mu}, \mu')$ and $(\mathfrak{N}^{x \leftarrow \nu}, \nu')$.

Back-and-forth systems

A partial isomorphism $h : \mathfrak{M} \rightarrow\!\!\!\!\!\rightarrow \mathfrak{N}$ is a bijection between a subset of $|\mathfrak{M}|$ and a subset of $|\mathfrak{N}|$, preserving and reflecting all accessibility relations λ.

Definition

A back-and-forth system between two Kripke structures $\mathfrak{M}$ and $\mathfrak{N}$ over a signature $\Delta = ((F, P), \mathtt{Prop})$ is a non-empty family $\mathcal{I}$ of basic partial isomorphisms between $\mathfrak{M}$ and $\mathfrak{N}$ satisfying the following:

- ($@$-extension) If $\mathcal{L}$ is closed under retrieve, then for all $h \in \mathcal{I}$ and all $k \in F$, there exists a $g \in \mathcal{I}$ such that $h \subseteq g$ and $k^{\mathfrak{M}} \in \mathrm{dom}(g)$.
- ($\Diamond$-extension) If $\mathcal{L}$ is closed under possibility over an action $\mathfrak{a}$, then:
  - (forth) for all $h \in \mathcal{I}$, all $w_1 \in \mathrm{dom}(h)$ and all $w_2 \in |\mathfrak{M}|$ such that $w_1 \ \mathfrak{a}^{\mathfrak{M}} \ w_2$, there exists a $g \in \mathcal{I}$ such that $h \subseteq g$, $w_2 \in \mathrm{dom}(g)$, and $g(w_1) \ \mathfrak{a}^{\mathfrak{N}} \ g(w_2)$;
  - (back) for all $h \in \mathcal{I}$, all $v_1 \in \mathrm{rng}(h)$ and all $v_2 \in |\mathfrak{N}|$ such that $v_1 \ \mathfrak{a}^{\mathfrak{N}} \ v_2$, there exists a $g \in \mathcal{I}$ such that $h \subseteq g$, $v_2 \in \mathrm{rng}(g)$, and $g^{-1}(v_1) \ \mathfrak{a}^{\mathfrak{M}} \ g^{-1}(v_2)$.
- ($\exists$-extension) If $\mathcal{L}$ is closed under existential quantifiers, then:
  - (forth) for all $h \in \mathcal{I}$ and all $w \in |\mathfrak{M}|$, there exists a $g \in \mathcal{I}$ such that $h \subseteq g$ and $w \in \mathrm{dom}(g)$;
  - (back) for all $h \in \mathcal{I}$ and all $v \in |\mathfrak{N}|$, there exists a $g \in \mathcal{I}$ such that $h \subseteq g$ and $v \in \mathrm{rng}(g)$.

Back-and-forth systems

Definition

- Two Kripke structures $\mathfrak{M}$ and $\mathfrak{N}$ are back-and-forth equivalent, if there is a back-and-forth system $\mathcal{I}$ between $\mathfrak{M}$ and $\mathfrak{N}$, in symbols, $\mathfrak{M} \equiv_{\mathcal{I}} \mathfrak{N}$.
- Two pointed models $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ are back-and-forth equivalent, if there is a back-and-forth system $\mathcal{I}$ between $\mathfrak{M}$ and $\mathfrak{N}$ such that $h(w) = v$ for some $h \in \mathcal{I}$, in symbols, $(\mathfrak{M}, w) \equiv_{\mathcal{I}} (\mathfrak{N}, v)$.

Bisimulations, back-and-forth systems, ω-EF games

Theorem

Let $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ be two pointed models over a signature Δ. Then:

$$(\mathfrak{M}, w) \equiv_B (\mathfrak{N}, v) \quad \textit{iff} \quad (\mathfrak{M}, w) \approx_\omega (\mathfrak{N}, v).$$

Theorem

Let $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ be two pointed models over a signature Δ.

1. *If $(\mathfrak{M}, w) \equiv_{\mathcal{I}} (\mathfrak{N}, v)$ for some back-and-forth system $\mathcal{I}$, then $(\mathfrak{M}, w) \approx_\omega (\mathfrak{N}, v)$.*
2. *Assume that (i) $\downarrow \in \mathcal{O}$, and (ii) $@ \in \mathcal{O}$ whenever $\Diamond \in \mathcal{O}$ or $\exists \in \mathcal{O}$. Then the converse holds as well, that is:*

   If $(\mathfrak{M}, w) \approx_\omega (\mathfrak{N}, v)$, then $(\mathfrak{M}, w) \equiv_{\mathcal{I}} (\mathfrak{N}, v)$ for some back-and-forth system $\mathcal{I}$.

Image-finite models

A model $\mathfrak{M}$ is image-finite if each state has a finite number of direct successors.

Theorem

Let $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ be two image-finite pointed models defined over a signature Δ. Then:

$$(\mathfrak{M}, w) \equiv (\mathfrak{N}, v) \quad \textit{iff} \quad (\mathfrak{M}, w) \approx_{\omega} (\mathfrak{N}, v) \ .$$

Corollary (Hennessy-Milner theorem)

Let $(\mathfrak{M}, w)$ and $(\mathfrak{N}, v)$ be two image-finite pointed models. Then:

$$(\mathfrak{M}, w) \equiv (\mathfrak{N}, v) \quad \textit{iff} \quad (\mathfrak{M}, w) \equiv_{B} (\mathfrak{N}, v) \ .$$

Countable rooted models

Theorem

Assume that $\{\Diamond, @, \downarrow\} \subseteq \mathcal{O}$. Let $(\mathfrak{M}, w_0)$ and $(\mathfrak{N}, v_0)$ be two rooted pointed models that are countable. Then:

$$(\mathfrak{M}, w_0) \cong (\mathfrak{N}, v_0) \quad \textit{iff} \quad (\mathfrak{M}, w_0) \equiv_{B} (\mathfrak{N}, v_0) \ .$$

Corollary

Assume that $\{\Diamond, @, \downarrow\} \subseteq \mathcal{O}$. Let $(\mathfrak{M}, w_0)$ and $(\mathfrak{N}, v_0)$ be two rooted image-finite pointed models. Then:

$$(\mathfrak{M}, w_0) \cong (\mathfrak{N}, v_0) \quad \textit{iff} \quad (\mathfrak{M}, w_0) \equiv (\mathfrak{N}, v_0) \ .$$

Bibliography

Areces, C., Blackburn, P., and Marx, M. (2001).
Hybrid Logics: Characterization, Interpolation and Complexity.
J. Symbolic Logic, 66(3):977–1010.

Badia, G., Gaina, D., Knapp, A., Kowalski, T., and Wirsing, M. (2025).
A modular bisimulation characterisation for fragments of hybrid logic.
The Bulletin of Symbolic Logic, page 1–24.

Diaconescu, R. (2016).
Quasi-varieties and Initial Semantics for Hybridized Institutions.
J. Logic Comput., 26(3):855–891.

Diaconescu, R. (2017).
Implicit Kripke Semantics and Ultraproducts in Stratified Institutions.
J. Log. Comput., 27(5):1577–1606.

Găină, D. (2020).
Forcing and Calculi for Hybrid Logics.
J. ACM, 67(4):25:1–25:55.

Găină, D. and Kowalski, T. (2020).
Fraïssé-Hintikka Theorem in Institutions.
J. Log. Comput., 30(7):1377–1399.

Hodkinson, I. and Tahiri, H. (2010).
A bisimulation characterization theorem for hybrid logic with the current-state binder.
Rev. Symbolic Logic, 3(2):247–261.

Fagin's Theorem for Semiring Turing Machines

Carles Noguera

Joint work with Guillermo Badia, Thomas Eiter, Manfred Droste, Rafael Kiesel, and Erik Paul

Fukuoka, Japan, LAC 2025

Finite model theory

The second half of the twentieth century has seen a growing interest in the connections between mathematical logic and computer science, which, among many other crucial developments, has justified an increased interest in *finite* mathematical structures.

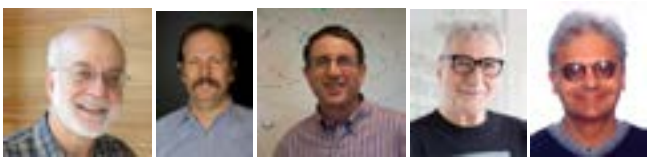
- 1950: Boris Trakhtenbrot, *The Impossibility of an Algorithm for the Decidability Problem on Finite Classes*. First-order tautologies over finite models are not recursively enumerable.
- 1973: Ronald Fagin writes a PhD thesis that inaugurates the systematic study of first-order logic based on finite models.

Descriptive complexity: just some famous results

Ron Fagin (1973): ESO *captures* NP. That is, for each ESO-formula φ, the class of finite structures axiomatized by φ form an NP set, and conversely, for each NP set of finite structures can be described by an ESO-formula.

Moshe Vardi (1981) and Neil Immerman (1986): LFP *captures* P over ordered structures.

Moshe Vardi (1982) and Serge Abiteboul and Victor Vianu (1989): PFP *captures* PSPACE over ordered structures.



Automata

- 1943: Warren Sturgis McCulloch and Walter Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, introduce neural networks and use them to compute Boolean functions.
- 1951: Stephen Cole Kleene defines regular languages in order to describe neural networks.
- 1956: Kleene describes the relations between finite automata and neural networks.
- 1957: John Myhill and Anil Nerode characterize regular languages as those with a finite number of equivalence classes (which coincide with the number of states of their minimal accepting finite automaton).

Weighted automata

- Marcel-Paul Schützenberger (1961) introduced weighted automata, nondeterministic finite automata augmented with values from a semiring as weights on the transitions, to model e.g. the cost involved when executing a transition, the amount of resources or time needed for this, or the probability or reliability of its successful execution.
- The theory of weighted automata and weighted context-free grammars was essential for the solution of such classical automata-theoretic problems as the decidability of the equivalence of unambiguous context-free languages and regular languages.

Weighted automata and weighted Logics

Manfred Droste, Werner Kuich, and Heiko Vogler. 2009. *Handbook of Weighted Automata* (1st. ed.). Springer Publishing Company, Incorporated.

Manfred Droste and Paul Gastin. Weighted automata and weighted logics, *Theoretical Computer Science* 380:69–86, 2007.

They give a weighted Büchi theorem (a result from the prehistory of descriptive complexity stating that regular languages are precisely those definable in MSO).

Weighted Turing machines and complexity classes – 1

Carsten Damm, Markus Holzer, and Pierre McKenzie. The complexity of tensor calculus, *Computational Complexity* 11:54–89, 2002.


They introduce what they call algebraic Turing machines (more general than weighted automata with weights also coming from semirings), as well as obvious notions of complexity classes that are counting classes from the unweighted perspective.

Weighted Turing machines and complexity classes – 2

- Peter Kostolányi. Weighted automata and logics meet computational complexity, Inf. Comput. 301, 105213, 2024.
- G. Badia, M. Droste, C. Noguera, and E. Paul. Logical Characterizations of Weighted Complexity Classes. In *49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024)*. Leibniz International Proceedings in Informatics (LIPIcs), Volume 306, pp. 14:1-14:16, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

Another perspective: Semiring Turing machines

Thomas Eiter and Rafael Kiesel. Semiring reasoning frameworks in AI and their computational complexity, *Journal of Artificial Intelligence Research* 77:207–293, 2023.


They differ from the more common weighted Turing machines in that semiring values can appear explicitly on the tape of the machine and there are infinitely many possible transitions.

The development of a Fagin theorem is posed as an open problem.

Semirings

Definition 1 (Semirings)

$\mathcal{R} = \langle R, \oplus, \otimes, 0, 1\rangle$, where addition $+$ and multiplication $\cdot$ are binary operations on R, and $0, 1 \in R$ such that

- $\langle R, \oplus, 0\rangle$ is a commutative monoid, $\langle R, \otimes, 1\rangle$ is a monoid, and $0 \neq 1$;
- for each $a, b, c \in R$, $a \otimes (b \oplus c) = a \otimes b \oplus a \otimes c$ and $(b \oplus c) \otimes a = b \otimes a \oplus c \otimes a$;
- for each $a \in R$, we have that $a \otimes 0 = 0 \otimes a = 0$.

$\mathcal{R}$ is commutative if the monoid $\langle R, \otimes, 1\rangle$ is commutative.

Semirings: examples

- the Boolean semiring $\mathbb{B} = \langle\{0, 1\}, \min, \max, 0, 1\rangle$,
- any bounded distributive lattice $\langle L, \vee, \wedge, 0, 1\rangle$,
- the semiring of natural numbers $\langle\mathbb{N}, +, \cdot, 0, 1\rangle$,
- the semiring of extended natural numbers $\langle\mathbb{N} \cup \{+\infty\}, +, \cdot, 0, 1\rangle$ where $0 \cdot (+\infty) = 0$,
- the max-plus or arctic semiring $\text{Arct} = \langle(\text{R})_+ \cup \{-\infty\}, \max, +, -\infty, 0\rangle$, where $(\text{R})_+$ denotes the set of non-negative real numbers,
- the restriction of the arctic semiring to the natural numbers $\mathbb{N}_{\max} = \langle\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0\rangle$,
- the min-plus or tropical semiring $\text{Trop} = \langle(\text{R})_+ \cup \{+\infty\}, \min, +, +\infty, 0\rangle$,
- the restriction of the tropical semiring to the natural numbers $\mathbb{N}_{\min} = \langle\mathbb{N} \cup \{+\infty\}, \min, +, +\infty, 0\rangle$,
- the semiring $\mathcal{F}_* = \langle[0, 1], \max, *, 0, 1\rangle$ given by a t-norm $*$,

Semiring Turing machine – 1

Given a commutative semiring $\mathcal{R}$, a Semiring Turing Machine over $\mathcal{R}$ is an 8-tuple $M = \langle R', Q, \Sigma_I, \Sigma, \iota, \mathcal{X}, \sqcup, \delta\rangle$, where

- $R' \subseteq R$ is a finite set of semiring values, (intuitively, this is a set of fixed values that are "known" to M)
- Q is a finite set of states,
- Σ_I is a finite set of symbols (the input alphabet),
- Σ is a finite set of symbols (the tape alphabet) such that $\Sigma_I \subseteq \Sigma$,
- $\iota \in Q$ is the initial state,
- $\sqcup \in \Sigma \setminus \Sigma_I$ is the blank symbol,
- $\mathcal{X} \in \Sigma \setminus \Sigma_I$ is the semiring placeholder,
- $\delta \subseteq (Q \times \Sigma) \times (Q \times \Sigma) \times \{-1, 1\} \times (R' \cup \{\mathcal{X}\})$ is a weighted transition relation, where the last entry of the tuple is the weight.

We call an SRTM M terminating, if there exists a function $\ell\colon (\Sigma_I \cup R)^* \longrightarrow \mathbb{N}$ such that for every input $s \in (\Sigma_I \cup R)^*$ every computation path (defined as usual) of M on s has at most length $\ell(s)$.

Semiring Turing machine – 2

Given $x \in (\Sigma \times R)^{\mathbb{N}}$, such that $x = (\sigma_i, r_i)_{i \in \mathbb{N}}$, we use $x_n^{\Sigma} = \sigma_n$ and $x_n^{R} = r_n$.

The value $val(c)$ of a terminating SRTM M on a configuration $c = \langle q, x, n \rangle$, where $q \in Q$ is a state, $x \in (\Sigma \times R)^*$ is the string on the tape, and $n \in \mathbb{N}$ is the head position, is recursively defined by

$$val(c) = \sum_{\tau \in \delta(q, x_n^{\Sigma})} wt(\tau, c) \otimes val(\tau(c)),$$

where $\tau = \langle (q', \sigma'), d, v \rangle \in \delta(q, x_n^{\Sigma})$, if $\langle (q, x_n^{\Sigma}), (q', \sigma'), d, v \rangle \in \delta$, $wt(\tau, c)$ denotes the weight of the transition and $\tau(c)$ denotes the new configuration after the transition; the empty sum has value 1.

Semiring Turing machine – 3

- The output $\|M\|(s)$ of a terminating SRTM M on an input string $s \in (\Sigma_I \cup R)^*$ is defined as $val(\iota, x(s), 0)$.
- For a commutative semiring $\mathcal{R}$, $\mathrm{NP}_{\infty}(\mathcal{R})$ is defined as the complexity class containing those functions $f \colon (\Sigma_I \cup R)^* \longrightarrow R$ that can be expressed as $f(s) = \|M\|(s)$ for some SRTM M where $\ell(s)$ is polynomially bounded in $|s|$.

SRTMs and weighted Turing machines are equivalent, when no weights are allowed as inputs.

Weighted logics – 1

A vocabulary τ is a pair $\langle \mathrm{Rel}_\tau \cup \mathrm{Rel}_\tau^{\mathrm{w}}, \mathrm{ar}_\tau \rangle$ where Rel_τ and $\mathrm{Rel}_\tau^{\mathrm{w}}$ are disjoint sets of relation symbols and $\mathrm{ar}_\tau \colon \mathrm{Rel}_\tau \cup \mathrm{Rel}_\tau^{\mathrm{w}} \longrightarrow \mathbb{N}_+$ is the arity function.

A τ-structure $\mathfrak{A}$ is a pair $\langle A, \mathcal{I}_{\mathfrak{A}} \rangle$ where A is a set, called the universe of $\mathfrak{A}$, and $\mathcal{I}_{\mathfrak{A}}$ is an interpretation, which maps every symbol $R \in \mathrm{Rel}_\tau$ to a set $R^{\mathfrak{A}} \subseteq A^{\mathrm{ar}_\tau(R)}$ and every symbol $W \in \mathrm{Rel}_\tau^{\mathrm{w}}$ to an $\mathcal{R}$-relation, i.e., a mapping $W^{\mathfrak{A}} \colon A^{\mathrm{ar}_\tau(W)} \longrightarrow \mathcal{R}$.

We assume that each structure is finite, that is, its universe is a finite set.

A structure is called ordered if it is given for a vocabulary $\tau \cup \{<\}$ where $<$ is interpreted as a linear ordering with endpoints. By $\mathrm{Str}(\tau)_<$ we denote the class of all finite ordered τ-structures.

Weighted logics – 2

We define first-order formulas β over a signature τ and weighted first-order formulas φ over τ and a semiring $\mathcal{S}$, respectively, by the grammars

$$\beta ::= x \leqslant y \mid R(x_1, \ldots, x_n) \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta$$
$$\varphi ::= \beta \mid r \mid W(x_1, \ldots, x_m) \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus x.\varphi \mid \bigotimes x.\varphi,$$

where $R \in \mathrm{Rel}_\tau$, and $n = \mathrm{ar}_\tau(R)$; $W \in \mathrm{Rel}^{\mathrm{w}}_\tau$, and $m = \mathrm{ar}_\tau(W)$; $x, x_1, \ldots, x_n \in \mathcal{V}$ and $x, x_1, \ldots, x_m \in \mathcal{V}$ are first-order variables; and $r \in \mathcal{R}$.

Likewise, we define second-order formulas β over τ and weighted second-order formulas φ over τ and $\mathcal{S}$ through

$$\beta ::= x \leqslant y \mid R(x_1, \ldots, x_n) \mid X(x_1, \ldots, x_n) \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$
$$\varphi ::= \beta \mid r \mid W(x_1, \ldots, x_n) \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus x.\varphi \mid \bigotimes x.\varphi \mid \bigoplus X.\varphi \mid \bigotimes X.\varphi,$$

with $R \in \mathrm{Rel}_\tau$ and $n = \mathrm{ar}_\tau(R) = \mathrm{ar}(X)$; $W \in \mathrm{Rel}^{\mathrm{w}}_\tau$ and $m = \mathrm{ar}_\tau(W)$; $x, x_1, \ldots, x_n \in \mathcal{V}$ and $x, x_1, \ldots, x_m \in \mathcal{V}$ are first-order variables; $X \in \mathcal{V}$ is a second-order variable; and $r \in \mathcal{R}$.

Weighted logics – 3

For $\varrho \in \mathfrak{A}_\mathcal{V}$ and a formula $\beta \in \mathrm{SO}(\tau)$ the relation "$\langle \mathfrak{A}, \varrho \rangle$ satisfies β", denoted by $\langle \mathfrak{A}, \varrho \rangle \models \beta$, is defined as

$$\begin{aligned}
\langle \mathfrak{A}, \varrho \rangle \models R(x_1, \ldots, x_n) &\iff x_1, \ldots, x_n \in \mathrm{dom}(\varrho) \text{ and } (\varrho(x_1), \ldots, \varrho(x_n)) \in R^{\mathfrak{A}} \\
\langle \mathfrak{A}, \varrho \rangle \models X(x_1, ..., x_n) &\iff x_1, ..., x_n, X \in \mathrm{dom}(\varrho), \langle \varrho(x_1), \ldots, \varrho(x_n) \rangle \in \varrho(X) \\
\langle \mathfrak{A}, \varrho \rangle \models \neg\beta &\iff \langle \mathfrak{A}, \varrho \rangle \models \beta \text{ does not hold} \\
\langle \mathfrak{A}, \varrho \rangle \models \beta_1 \vee \beta_2 &\iff \langle \mathfrak{A}, \varrho \rangle \models \beta_1 \text{ or } \langle \mathfrak{A}, \varrho \rangle \models \beta_2 \\
\langle \mathfrak{A}, \varrho \rangle \models \exists x.\beta &\iff \langle \mathfrak{A}, \varrho[x \to a] \rangle \models \beta \text{ for some } a \in A \\
\langle \mathfrak{A}, \varrho \rangle \models \exists X.\beta &\iff \langle \mathfrak{A}, \varrho[X \to I] \rangle \models \beta \text{ for some } I \subseteq A.
\end{aligned}$$

Weighted logics – 4

Let $\varphi \in \mathrm{wSO}(\tau, S)$ and $\mathfrak{A} \in \mathrm{Str}(\tau)_<$, assume the subsets of A^n ordered by the lexicographic ordering induced by $<$.

$$\begin{aligned}
[\![\beta]\!](\mathfrak{A}, \varrho) &= \begin{cases} 1 & \text{if } \langle \mathfrak{A}, \varrho \rangle \models \beta \\ 0 & \text{otherwise} \end{cases} \\
[\![W(x_1, \ldots, x_n)]\!](\mathfrak{A}, \varrho) &= W^{\mathfrak{A}}(\varrho(x_1), \ldots, \varrho(x_n)), \text{ where } W \in \mathrm{Rel}^{\mathrm{w}}_\tau \\
[\![r]\!](\mathfrak{A}, \varrho) &= r \\
[\![\varphi_1 \oplus \varphi_2]\!](\mathfrak{A}, \varrho) &= [\![\varphi_1]\!](\mathfrak{A}, \varrho) \oplus [\![\varphi_2]\!](\mathfrak{A}, \varrho) \\
[\![\varphi_1 \otimes \varphi_2]\!](\mathfrak{A}, \varrho) &= [\![\varphi_1]\!](\mathfrak{A}, \varrho) \otimes [\![\varphi_2]\!](\mathfrak{A}, \varrho) \\
[\![\bigoplus x.\varphi]\!](\mathfrak{A}, \varrho) &= \sum_{a \in A} [\![\varphi]\!](\mathfrak{A}, \varrho[x \to a]) \\
[\![\bigotimes x.\varphi]\!](\mathfrak{A}, \varrho) &= \prod_{1 \leqslant i \leqslant k} [\![\varphi]\!](\mathfrak{A}, \varrho[x \to a_i]) \\
[\![\bigoplus X.\varphi]\!](\mathfrak{A}, \varrho) &= \sum_{I \subseteq A^{\mathrm{ar}(X)}} [\![\varphi]\!](\mathfrak{A}, \varrho[X \to I]) \\
[\![\bigotimes X.\varphi]\!](\mathfrak{A}, \varrho) &= \prod_{1 \leqslant i \leqslant l_{\mathrm{ar}(X)}} [\![\varphi]\!](\mathfrak{A}, \varrho[X \to I_i^{\mathrm{ar}(X)}]).
\end{aligned}$$

Weighted existential second-order logic

Weighted existential second-order logic over the semiring $\mathcal{R}$ wESO[$\mathcal{R}$] is the fragment of wSO[$\mathcal{R}$] which includes all formulas of wFO[$\mathcal{R}$] and is closed under the quantifiers $\exists X$ and $\bigoplus X$.

Capturing a complexity class by means of a logic

Consider a weighted logic L[$\mathcal{R}$] (with weights in a semiring $\mathcal{R}$). We say that L[$\mathcal{R}$] captures $\mathrm{NP}_\infty(\mathcal{R})$ over ordered structures in the vocabulary $\tau = \{R_1, \ldots, R_j\}$ if the following holds:

1. For every L[$\mathcal{R}$]-formula φ, there exists a $P \in \mathrm{NP}_\infty(\mathcal{R})$ such that $P(\mathrm{enc}(\mathfrak{A})) = \|\varphi\|(\mathfrak{A})$ for every finite ordered τ-structure $\mathfrak{A}$, and
2. For every $P \in \mathrm{NP}_\infty(\mathcal{R})$, there exists an L[$\mathcal{R}$]-formula φ such that $P(\mathrm{enc}(\mathfrak{A})) = \|\varphi\|(\mathfrak{A})$ for every finite ordered τ-structure $\mathfrak{A}$.

Main result: Fagin's theorem

Theorem 2

The logic wESO[$\mathcal{R}$] *captures the quantitative complexity class* $\mathrm{NP}_\infty(\mathcal{R})$.

Final remarks – 1

- The proof of the theorem uses the fact that our semirings are commutative, because we use multiplicative quantifiers without concern for the order of the product.
- The theorem also holds for finite structures without an ordering $<$ if the semiring $\mathcal{R}$ is idempotent.
- Our proposed variation of Droste–Gastin weighted logic allows atomic formulas to appear in the semiring layer. So it can be seen as a logic to speak about databases over semirings.
- SRTMs allow users to perform computations over a semiring, using any values without need of bringing in complex encoding. Weighted Turing machines had access only to finitely many values, so were only good for finitely generated semirings. (e.g. the semiring of rationals was excluded)

Final remarks – 2

- The quantitative complexity class $\mathrm{NP}_\infty(\mathcal{R})$ we have studied here is different in general from the one we studied by Badia et al (2024) for weighted Turing machines. But they coincide when we restrict to SRTMs with only finitely many transitions.
- The Eiter-Kiesel original paper on SRTMs included several results linking $\mathrm{NP}(\mathcal{R})$ to classical complexity classes which carry over to our $\mathrm{NP}_\infty(\mathcal{R})$.

Unification and Narrowing in Maude 3.5

Santiago Escobar
Valencian Research Institute for Artificial Intelligence (VRAIN)
Universitat Politècnica de València Spain

September 30, 2025

Outline

Outline

Why rewriting logic?

1. Models and formal specification are easily written in Maude (simplicity, expressiveness, and performance)
2. Rewriting modulo associativity, commutativity and identity
3. Differentiation between concurrent and functional fragments of a model
4. Order-sorted and parameterized specifications
5. Infrastructure for formal analysis and verification (including search command, LTL model checker, theorem prover, etc.)
6. Reflection (meta-modeling, symbolic execution, building tools)
7. Application areas:
   - Models of computation (λ-calculi, π-calculus, petri nets, CCS),
   - Programming languages (C, Java, Haskell, Prolog),
   - Distributed algorithms and systems (security protocols, real-time, probabilistic),
   - Biological systems

Why adding Symbolic capabilities to Maude?

1. Logical features were included in preliminary designs of the language (80's) but never implemented in Maude
2. Automated reasoning capabilities by adding logical variables
3. Differentiation between concurrent and functional fragments of a model are lifted to differentiation between symbolic models and equational reasoning.
4. Equational unification and narrowing modulo combinations of A,C,U and variant equations
5. Infrastructure for formal analysis and verification lifted:
   - from equational reduction to equational unification,
   - from search to symbolic reachability,
   - from LTL model checker to logical LTL model checker,
   - from theorem proving to narrowing-based theorem proving,
   - from SMT solving to variant-based SMT solving.

   ⋮

What have we done!!

- Maude 2.4 (2009) Built-in AC Unification and Narrowing-based search (rule & axioms)
- Maude 2.6 (2011) Built-in ACU Unification. Variant Unification. Narrowing search (eqs)
- Maude 2.7 (2015) A+C+U Unification. Built-in Variant unification. Narrowing search
- Maude 2.7.1 (2016) Built-in Bounded Associativity
- Maude 3.0 (2019) Built-in Narrowing-based search
- Maude 3.2 (2022) Minimal Unification for axioms, for variants, for narrowing
- Maude 3.3 (2023) Improved Folding Narrowing-based search
- Maude 3.4 (2024) Folding Narrowing-based search, Meta-interpreters, Object notation, External Processes.
- Maude 3.5 (2025) Improved performance, disjunctive patterns

Valencia: narrowing with constraints, anti-unification, homeomorphic embedding, security

Outline

Rewriting logic in a nutshell

A rewrite theory is

$\mathcal{R} = (\Sigma, Ax \uplus E, R)$, with:

1. (Σ, R) a set of rewrite rules of the form $t \to s$
   (i.e., system transitions)
2. $(\Sigma, Ax \uplus E)$ a set of equational properties of the form $t = s$
   (i.e., E are equations and Ax are axioms such as ACU)

Intuitively, $\mathcal{R}$ specifies a concurrent system, whose states are elements of the initial algebra $T_{\Sigma/(Ax \uplus E)}$ specified by $(\Sigma, Ax \uplus E)$, and whose concurrent transitions are specified by the rules R.

Rewriting logic in a nutshell

```
mod VENDING-MACHINE is
  sorts Coin Item Marking Money State .
  subsort Coin < Money .
  op empty :  -> Money .
  op __ :  Money Money -> Money [assoc comm id:  empty] .
  subsort Money Item < Marking .
  op __ :  Marking Marking -> Marking [assoc comm id:  empty] .
  op <_> :  Marking -> State .
  ops $ q :  -> Coin .
  ops cookie cap :  -> Item .
  var M : Marking .
  rl [add-$] :  < M > => < M $ > [narrowing] .
  rl [add-q] :  < M > => < M q > [narrowing] .
  rl [buy-c] :  < M $ > => < M cap > [narrowing] .
  rl [buy-a] :  < M $ > => < M cookie q > [narrowing] .
  eq [change]:  q q q q = $ [variant] .
endm
```

Rewriting logic in a nutshell

```
Maude> search <$ q q q> =>! <cookie cap St:State> .
Solution 1 (state 3)
states:  6 rewrites:  5 in 0ms cpu (0ms real)
St:State --> null
No more solutions.
states:  6 rewrites:  5 in 0ms cpu (1ms real)
Maude> show path 3 .
state 0, State: < $ q q q >
===[ rl St $ => St cookie q .  ]===>
state 2, State: < $ cookie >
===[ rl St $ => St cap .  ]===>
state 3, State: < cap cookie >
```


Rewriting modulo

Rewriting is

Given $(\Sigma, Ax \uplus E, R)$, $t \to_{R,(Ax \uplus E)} s$ if there is

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in R;
- a matching σ (E-normalized and modulo Ax) such that $t|_p =_{(Ax \uplus E)} \sigma(l)$, and $s = t[\sigma(r)]_p$.

Ex: `< $ q q q >` $\to$ `< $ cookie >`
using "`rl < M $ > => < M cookie q > .`"
modulo AC of symbol "`__`"
Ex: `< q q q q >` $\to$ `< cap >`
using "`rl < M $ > => < M cap > .`"
modulo simplification with `q q q q = $` and AC of symbol "`__`"

Narrowing modulo

Narrowing is

Given $(\Sigma, Ax \uplus E, R)$, $t \leadsto_{\sigma,R,(Ax \uplus E)} s$ if there is

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in R;
- a unifier σ (E-normalized and modulo Ax) such that $\sigma(t|_p) =_{(Ax \uplus E)} \sigma(l)$, and $s = \sigma(t[r]_p)$.

Ex: `< X q q >` $\leadsto$ `< $ cookie >`
using "`rl < M $ > => < M cookie q > .`"
using substitution $\{X \mapsto$ `$ q` $\}$ modulo AC of symbol "`__`"
Ex: `< X q q >` $\leadsto$ `< cap >`
using "`rl < M $ > => < M cap > .`"
using substitution $\{X \mapsto$ `q q` $\}$
modulo simplification with `q q q q = $` and AC of symbol "`__`"

Outline

Unification modulo axioms

Definition

Given equational theory (Σ, Ax), an Ax-unification problem is

$$t \stackrel{?}{=} t'$$

An Ax-unifier is an order-sorted substitution σ s.t.

$$\sigma(t) =_{Ax} \sigma(t')$$

Decidability

- at most one mgu (syntactic unification, i.e., empty theory)
- a finite number (associativity–commutativity)
- an infinite number (associativity)

Unification Command in Maude

Maude provides a Ax-unification command of the form:

```
unify [ n ] in ⟨ModId⟩ :
  ⟨Term-1⟩ =? ⟨Term'-1⟩ /\ ... /\ ⟨Term-k⟩ =? ⟨Term'-k⟩ .
irredundant unify [ n ] in ⟨ModId⟩ :
  ⟨Term-1⟩ =? ⟨Term'-1⟩ /\ ... /\ ⟨Term-k⟩ =? ⟨Term'-k⟩ .
```

- ModId is the name of the module
- n is a bound on the number of unifiers
- new variables are created as `#n:Sort`
- Implemented at the core level of Maude (C++)

AC-Unification in Maude

```
Maude> unify [100] in NAT :
         X:Nat + X:Nat + Y:Nat =? A:Nat + B:Nat + C:Nat .

Solution 1
X:Nat --> #1:Nat + #2:Nat + #3:Nat + #5:Nat + #6:Nat + #8:Nat
Y:Nat --> #4:Nat + #7:Nat + #9:Nat
A:Nat --> #1:Nat + #1:Nat + #2:Nat + #3:Nat + #4:Nat
B:Nat --> #2:Nat + #5:Nat + #5:Nat + #6:Nat + #7:Nat
C:Nat --> #3:Nat + #6:Nat + #8:Nat + #8:Nat + #9:Nat
...

Solution 100
X:Nat --> #1:Nat + #2:Nat + #3:Nat + #4:Nat
Y:Nat --> #5:Nat
A:Nat --> #1:Nat + #1:Nat + #2:Nat
B:Nat --> #2:Nat + #3:Nat
C:Nat --> #3:Nat + #4:Nat + #4:Nat + #5:Nat
```


ACU-Unification in Maude

```
Maude> unify [100] in QID-SET : X:QidSet , X:QidSet , Y:QidSet =? A:QidSet , B:QidSet , C:QidSet .
unify [100] in QID-SET : X:QidSet, X:QidSet, Y:QidSet =? A:QidSet, B:QidSet, C:QidSet .
Decision time: 0ms cpu (1ms real)

Solution 1
X:QidSet --> empty
Y:QidSet --> empty
A:QidSet --> empty
B:QidSet --> empty
C:QidSet --> empty

Solution 2
X:QidSet --> #1:QidSet
Y:QidSet --> empty
A:QidSet --> #1:QidSet, #1:QidSet
B:QidSet --> empty
C:QidSet --> empty
```


Irredundant Unification in Maude

```
Maude> unify in UNIF-VENDING-MACHINE :
        < q q X:Marking > =? < $ Y:Marking > .

Unifier 1
X:Marking --> $
Y:Marking --> q q

Unifier 2
X:Marking --> $ #1:Marking
Y:Marking --> q q #1:Marking


Maude> irredundant unify in UNIF-VENDING-MACHINE :
        < q q X:Marking > =? < $ Y:Marking > .

Unifier 1
X:Marking --> $ #1:Marking
Y:Marking --> q q #1:Marking
```

Identity Unification in Maude

```
mod LEFTID-UNIFICATION-EX is
    sorts Magma Elem . subsorts Elem < Magma .
    op _ : Magma Magma -> Magma [left id: e] .
    ops a b c d e : -> Elem .
endm
```

```
Maude> unify in LEFTID-UNIFICATION-EX : X:Magma a =? (Y:Magma a) a .
Solution 1              Solution 2
X:Magma --> a           X:Magma --> #1:Magma a
Y:Magma --> e           Y:Magma --> #1:Magma
```

```
Maude> unify in LEFTID-UNIFICATION-EX : a X:Magma =? (a a) Y:Magma .
No unifier.
```

```
mod COMM-ID-UNIFICATION-EX is
    sorts Magma Elem . subsorts Elem < Magma .
    op _ : Magma Magma -> Magma [comm id: e] .
    ops a b c d e : -> Elem .
endm
```

```
Maude> unify in COMM-ID-UNIFICATION-EX : X:Magma a =? (Y:Magma a) a .
Solution 1      Solution 2              Solution 3
X:Magma --> a   X:Magma --> a #1:Magma  X:Magma --> a
Y:Magma --> e   Y:Magma --> #1:Magma    Y:Magma --> e
```


A-Unification in Maude

```
Maude> unify in UNIFICATION-EX4 : X:NList : Y:NList : Z:NList =? P:NList : Q:NList .

Solution 1
X:NList --> #1:NList : #2:NList
Y:NList --> #3:NList
Z:NList --> #4:NList
P:NList --> #1:NList
Q:NList --> #2:NList : #3:NList : #4:NList

Solution 2
X:NList --> #1:NList
Y:NList --> #2:NList : #3:NList
Z:NList --> #4:NList
P:NList --> #1:NList : #2:NList
Q:NList --> #3:NList : #4:NList

Solution 3
X:NList --> #1:NList
Y:NList --> #2:NList
Z:NList --> #3:NList : #4:NList
P:NList --> #1:NList : #2:NList : #3:NList
Q:NList --> #4:NList

Unifier 4
X:NList --> #1:NList
Y:NList --> #2:NList
Z:NList --> #3:NList
P:NList --> #1:NList : #2:NList
Q:NList --> #3:NList

Unifier 5
X:NList --> #1:NList
Y:NList --> #2:NList
Z:NList --> #3:NList
P:NList --> #1:NList
Q:NList --> #2:NList : #3:NList
```


Incomplete A-Unification in Maude

Possible warnings and situations:

- Associative unification using cycle detection.
- Associative unification algorithm detected an infinite family of unifiers.
- Associative unification using depth bound of 5.
- Associative unification algorithm hit depth bound.

Example:

```
Maude> unify in UNIFICATION-EX4 : 0 : X:NList =? X:NList : 0 .
Warning: Unification modulo the theory of operator _:_ has encountered
an instance for which it may not be complete.

Solution 1
X:NList --> 0
Warning: Some unifiers may have been missed due to incomplete
unification algorithm(s).
```

AU-Unification in Maude

```
Maude> irredundant unify in UNIFICATION-EX5 :
         X:NList : Y:NList : Z:NList =? P:NList : Q:NList .
Decision time: 2ms cpu (2ms real)

Unifier 1
X:NList --> #3:NList : #4:NList
Y:NList --> #1:NList
Z:NList --> #2:NList
P:NList --> #3:NList
Q:NList --> #4:NList : #1:NList : #2:NList

Unifier 2
X:NList --> #1:NList
Y:NList --> #3:NList : #4:NList
Z:NList --> #2:NList
P:NList --> #1:NList : #3:NList
Q:NList --> #4:NList : #2:NList

Unifier 3
X:NList --> #1:NList
Y:NList --> #2:NList
Z:NList --> #4:NList : #3:NList
P:NList --> #1:NList : #2:NList : #4:NList
Q:NList --> #3:NList
```

AU fewer unifiers than A (5 vs 3) & unify returns many more than irredundant unify (32 vs 3)

Axiomatization of Booleans in Maude using axioms and variant equations

```
fmod BOOL-FVP is protecting TRUTH-VALUE .
   op _and_ : Bool Bool -> Bool [assoc comm] .
   op _xor_ : Bool Bool -> Bool [assoc comm] .
   op not_ : Bool -> Bool .
   op _or_ : Bool Bool -> Bool .
   op _<=>_ : Bool Bool -> Bool .
   vars X Y Z W : Bool .

   eq X and true = X [variant] .
   eq X and false = false [variant] .
   eq X and X = X  [variant] .
   eq X and X and Y = X and Y [variant] .    *** AC extension
   eq X xor false = X [variant] .
   eq X xor X = false [variant] .
   eq X xor X xor Y = Y [variant] .          *** AC extension
   eq not X = X xor true [variant] .
   eq X or Y = (X and Y) xor X xor Y [variant] .
   eq X <=> Y = true xor X xor Y [variant] .
endfm
```


Unification modulo axioms w/o minimality

```
==========================================
unify in BOOL-FVP : X and not Y and not Z =? W and Y and not X .
Decision time: 0ms cpu (0ms real)

Unifier 1
X --> #2:Bool and not #1:Bool
Z --> #1:Bool
Y --> #2:Bool and not #1:Bool
W --> not #1:Bool

.....

Unifier 5
X --> not #1:Bool
Z --> #1:Bool
Y --> not #1:Bool
W --> not #1:Bool
==========================================
irredundant unify in BOOL-FVP : X and not Y and not Z =? W and Y and not X .
Decision time: 0ms cpu (0ms real)

Unifier 1
X --> #1:Bool and #2:Bool
Z --> #1:Bool and #2:Bool
Y --> #1:Bool
W --> #2:Bool and not #1:Bool

Unifier 2
X --> #2:Bool
Z --> #1:Bool
Y --> #2:Bool
W --> not #1:Bool
```

Outline

From equational reduction to variants (1/4)

E,Ax-variant

Given a term t and an equational theory $Ax \uplus E$, (t', θ) is an E,Ax-variant of t if $\theta(t)\downarrow_{E,Ax} =_{Ax} t'$ [Comon-Delaune-RTA05]

Exclusive Or

$$X \oplus 0 \to X \qquad X \oplus (Y \oplus Z) = (X \oplus Y) \oplus Z$$
$$X \oplus X \to 0 \qquad X \oplus Y = Y \oplus X$$
$$X \oplus X \oplus Y \to Y \qquad \text{(axioms: } Ax\text{)}$$

Computed Variants

For $X \oplus X$: $(0, id), (0, \{X \mapsto a\}), (0, \{X \mapsto a \oplus b\}), \ldots$

From equational reduction to variants (2/4)

Finite and complete set of E,Ax-variants

A preorder relation of generalization between variants provides a notion of most general variant.

Computed Variants

For $X \oplus Y$ there are 7 most general E,Ax-variants

1. $(X \oplus Y, id)$
2. $(0, \{X \mapsto U, Y \mapsto U\})$
3. $(Z, \{X \mapsto 0, Y \mapsto Z\})$
4. $(Z, \{X \mapsto Z \oplus U, Y \mapsto U\})$
5. $(Z, \{X \mapsto Z, Y \mapsto 0\})$
6. $(Z, \{X \mapsto U, Y \mapsto Z \oplus U\})$

From equational reduction to variants (3/4)

Finite Variant Property

Theory has FVP if finite number of most general variants for every term.

Common

- Cryptographic Security Protocols: Public or shared encryption, Exclusive Or, Abelian groups, Diffie-Hellman
- Satisfiability Modulo Theories Natural Presburger Arithmetic, Integer Presburger Arithmetic, Lists, Sets

Used in application areas

Equational Unification, Logical Model Checking, Cyber-Physical systems, Partial evaluation, Confluence tools, Termination tools, Theorem provers

From equational reduction to variants (4/4)

Test for FVP

Whether a theory has FVP is undecidable in general, though there are approximations techniques.

Computing most general variants

Given a theory that has FVP, it is possible to compute all the most general variants by using the Folding Variant Narrowing Strategy (Escobar et al. 2012)

E,Ax-variants - Example

$$\begin{array}{c} X \oplus 0 \to X \\ X \oplus X \to 0 \\ X \oplus X \oplus Y \to Y \\ \text{(cancellation rules: } E\text{)} \end{array} \qquad \begin{array}{c} X \oplus (Y \oplus Z) = (X \oplus Y) \oplus Z \\ X \oplus Y = Y \oplus X \\ \text{(axioms: } Ax\text{)} \end{array}$$

- For $X \oplus X$ only E,Ax-variant is: $(0, id)$
- For $X \oplus Y$ there are 7 most general E,Ax-variants
  1. $(X \oplus Y, id)$
  2. $(0, \{X \mapsto U, Y \mapsto U\})$
  3. $(Z, \{X \mapsto 0, Y \mapsto Z\})$
  4. $(Z, \{X \mapsto Z \oplus U, Y \mapsto U\})$
  5. $(Z, \{X \mapsto Z, Y \mapsto 0\})$
  6. $(Z, \{X \mapsto U, Y \mapsto Z \oplus U\})$
  7. $(Z_1 \oplus Z_2, \{X \mapsto U \oplus Z_1, Y \mapsto U \oplus Z_2\})$

Variant Command in Maude

Maude provides variant generation:

`get variants [` n `] in` $\langle ModId \rangle$ `:` $\langle Term \rangle$ `.`
`get irredundant variants [` n `] in` $\langle ModId \rangle$ `:` $\langle Term \rangle$ `.`

- ModId is the name of the module
- n is a bound on the number of variants
- new variables are created as `#n:Sort` and `%n:Sort`
- Implemented at the core level of Maude (C++)
- Folding variant narrowing strategy is used internally
- Terminating if Finite Variant Property
- Incremental output if not Finite Variant Property
- Irredundant version only if Finite Variant Property

Exclusive-or Variants

```
fmod EXCLUSIVE-OR is
  sorts Nat NatSet .  subsort Nat < NatSet .
  op 0 : -> Nat .
  op s : Nat -> Nat .
  op mt : -> NatSet .
  op _*_ : NatSet NatSet -> NatSet [assoc comm] .
  vars X Z : [NatSet] .
  eq [idem] :     X * X = mt    [variant] .
  eq [idem-Coh] : X * X * Z = Z [variant] .
  eq [id] :       X * mt = X    [variant] .
endfm
```

```
Maude> get variants in EXCLUSIVE-OR : X * Y .
Variant 1                                        Variant 7
[NatSet]: #1:[NatSet] * #2:[NatSet]   .........  [NatSet]: %1:[NatSet]
X --> #1:[NatSet]                                X -->  %1:[NatSet]
Y --> #2:[NatSet]                                Y --> mt
```


Abelian Group Variants

```
fmod ABELIAN-GROUP is
  sorts Elem .
  op _+_ : Elem Elem -> Elem [comm assoc] .
  op -_ : Elem -> Elem .
  op 0 : ->  Elem .
  vars X Y Z : Elem  .
  eq X + 0 = X [variant] .
  eq X + (- X) = 0 [variant] .
  eq X + (- X) + Y = Y [variant] .
  eq - (- X) = X [variant] .
  eq - 0 = 0 [variant] .
  eq (- X) + (- Y) = -(X + Y) [variant] .
  eq -(X + Y) + Y = - X [variant] .
  eq -(- X + Y) = X + (- Y) [variant] .
  eq (- X) + (- Y) + Z  = -(X + Y) + Z [variant] .
  eq -(X + Y) + Y + Z = (- X) + Z [variant] .
endfm
```

```
Maude> get variants in ABELIAN-GROUP : X + Y .
Variant 1                                    Variant 47
Elem: #1:Elem + #2:Elem    ................  Elem: - (%2:Elem + %3:Elem)
X --> #1:Elem                                X --> %4:Elem + - (%1:Elem + %2:Elem)
Y --> #2:Elem                                Y --> %1:Elem + - (%3:Elem + %4:Elem)
```

Incremental Variant Generation

```
fmod NAT-VARIANT is
  sort Nat .
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor] .
  op _+_ : Nat Nat -> Nat .
  vars X Y : Nat .
  eq [base] : 0 + Y = Y [variant] .
  eq [ind] :  s(X) + Y = s(X + Y) [variant] .
endfm
```

```
Maude> get variants in NAT-VARIANT : s(0) + X .
Variant 1
Nat: s(#1:Nat)
X --> #1:Nat
```

```
Maude> get variants [10] in NAT-VARIANT : X + s(0) .
Variant 1                                                    Variant 10
Nat: #1:Nat + s(0)    ....................................   Nat: s(s(s(s(s(0)))))
X --> #1:Nat                                                 X --> s(s(s(s(0))))
```

Infinite!!!

Variant Generation in Incomplete Theories (due to assoc)

```
fmod VARIANT-UNIFICATION-ASSOC is
  protecting NAT .
  sort NList .
  subsort Nat < NList .

  op _:_ : NList NList -> NList [assoc ctor] .

  var E : Nat .
  var L : NList .

  ops tail prefix : NList ~> NList .
  ops head last : NList ~> Nat .
  eq head(E : L) = E [variant] .
  eq tail(E : L) = L [variant] .
  eq prefix(L : E) = L [variant] .
  eq last(L : E) = E [variant] .

  op duplicate : NList ~> Bool .
  eq duplicate(L : L) = true [variant] .
endfm
```

```
Maude> get variants in VARIANT-UNIFICATION-ASSOC :
         duplicate(prefix(L) : tail(L)) .

Variant 1
[Bool]: duplicate(prefix(#1:NList) : tail(#1:NList))
L --> #1:NList

Variant 2
[Bool]: duplicate(%1:NList : tail(%1:NList : %2:Nat))
L --> %1:NList : %2:Nat

Variant 3
[Bool]: duplicate(prefix(%1:Nat : %2:NList) : %2:NList)
L --> %1:Nat : %2:NList

Variant 4
[Bool]: duplicate(#1:Nat : #2:NList : #2:NList : #3:Nat)
L --> #1:Nat : #2:NList : #3:Nat

Variant 5
[Bool]: duplicate(#1:Nat : #2:Nat)
L --> #1:Nat : #2:Nat

Warning: Unification modulo the theory of operator _:_ has encountered
an instance for which it may not be complete.
Variant 6
Bool: true
L --> %1:Nat : %1:Nat : %1:Nat

Variant 7
Bool: true
L --> %1:Nat : %1:Nat

No more variants.
Warning: Some variants may have been missed due to incomplete unification algorithm(s).
```


Finite Variant Property

- Theory has FVP if there is a finite number of most general E,Ax-variants for every term.
- If finite number of unifiers from t, E,Ax-narrowing must compute them, though infinite redundant E,Ax-narrowing sequences may exist
- [Comon-Delaune-RTA05] An equational theory has the finite variant property if there is a bound n in the number of steps for each term

$$\forall t, \exists n, \forall \sigma \text{ s.t. } (\sigma\downarrow_{E,Ax})(t) \xrightarrow{\leq n}_{E,Ax} \sigma(t)\downarrow_{E,Ax}$$

Finite Variant Property

1. [Comon-Delaune-RTA05]
   Exclusive Or (max. bound 1)
2. [Comon-Delaune-RTA05]
   Abelian group (max. bound 2)
3. [Comon-Delaune-RTA05]
   Diffie-Hellman (max. bound 4)
4. [Comon-Delaune-RTA05]
   Homomorphism (NOT)
5. [Escobar-Meseguer-Sasse-RTA08]
   Sufficient & necessary conditions for FVP

Folding Variant-Narrowing

1. Complete narrowing strategy modulo axioms Ax with smaller search space than unrestricted Ax-narrowing.
2. Terminating when FVP (i.e., decidable narrowing-based $As \uplus E$-unification procedure)
3. Optimally terminating (no other possible narrowing strategy terminates for more equational theories)

- Based on *E,Ax-variant*, folding variant narrowing strategy:
  1. it only uses substitutions in normal form
  2. if a variant is an instance of a more general variant computed previously, stop narrowing path
  3. complete under very general assumptions on Ax and E

Outline

Admissible Theories

Maude provides order-sorted $Ax \uplus E$-unification algorithm for all order-sorted theories $(\Sigma, Ax, \vec{E})$ s.t.

1. Maude has an Ax-unification algorithm,
2. E equations specified with the `eq` and `variant` keywords.
3. E is unconditional, convergent, sort-decreasing and coherent modulo Ax.
4. The `owise` feature is not allowed.

Equational Unification Command in Maude

Maude provides a $(Ax \uplus E)$-unification command of the form:

`variant unify [` n `] in` $\langle ModId \rangle$ `:`
$\langle Term\text{-}1 \rangle$ `=?` $\langle Term'\text{-}1 \rangle$ `/\ ... /\` $\langle Term\text{-}k \rangle$ `=?` $\langle Term'\text{-}k \rangle$ `.`
`filtered variant unify [` n `] in` $\langle ModId \rangle$ `:`
$\langle Term\text{-}1 \rangle$ `=?` $\langle Term'\text{-}1 \rangle$ `/\ ... /\` $\langle Term\text{-}k \rangle$ `=?` $\langle Term'\text{-}k \rangle$ `.`

- ModId is the name of the module
- n is a bound on the number of unifiers
- new variables are created as `#n:Sort` and `%n:Sort`
- Implemented at the core level of Maude (C++)
- Terminating if Finite Variant Property
- Incremental output if not Finite Variant Property

Variant Unification modulo axioms w/o minimality

```
==========================================
variant unify in BOOL-FVP : (X or Y) <=> Z =? true .

Unifier 1
rewrites: 489 in 1828ms cpu (2110ms real) (267 rewrites/second)
X --> true
Y --> #1:Bool
Z --> true

...

Unifier 12
rewrites: 2934 in 9927ms cpu (11571ms real) (295 rewrites/second)
X --> %1:Bool and %2:Bool
Y --> %1:Bool and %3:Bool
Z --> (%1:Bool and %2:Bool) xor (%1:Bool and %3:Bool) xor %1:Bool and %2:Bool and %3:Bool

No more unifiers.
rewrites: 3006 in 9998ms cpu (11657ms real) (300 rewrites/second)
==========================================
filtered variant unify in BOOL-FVP : (X or Y) <=> Z =? true .
rewrites: 3224 in 10161ms cpu (11957ms real) (317 rewrites/second)

Unifier 1
X --> #1:Bool xor #2:Bool
Y --> #1:Bool
Z --> #2:Bool xor #1:Bool and #1:Bool xor #2:Bool

No more unifiers.
Advisory: Filtering was complete.
```

Outline

Narrowing-based Symbolic Reachability Analysis

- Model checking techniques effective in verification of concurrent systems
- However, standard techniques only work for:
  - specific initial state (or finite set of initial states)
  - the set of states reachable from the initial state is finite
  - abstraction techniques
- Various model checking techniques for infinite-state systems exist, but they are less developed
  - Stronger limitations on the kind of systems and/or the properties that can be model checked

Narrowing Search Command in Maude

Narrowing-based search command of the form:

`vu-narrow [` n`,` m `] in` $\langle ModId \rangle$ `:` $\langle Term\text{-}1 \rangle$ $\langle SearchArrow \rangle$ $\langle Term\text{-}2 \rangle$.

- n is the bound on the desired reachability solutions
- m is the maximum depth of the narrowing tree
- `Term-1` is not a variable but may contain variables
- `Term-2` is a pattern to be reached
- `SearchArrow` is either `=>1`, `=>+`, `=>*`, `=>!`
- `=>!` denotes strongly irreducible terms or rigid normal forms.
- Implemented at the core level of Maude (C++)
- "`{fold} vu-narrow {filter,delay}`" is the most general version (new things to come)

Outline

Ex1 - Constrained Horn Clauses for Program Verification TPLP 2022

```
int sum_upto(int x) {
 int r = 0 ;
 while (x > 0) {
 r = r + x; x = x - 1; }
 return r;
}
```

This imperative program is translated into a logic program and the Hoare triple

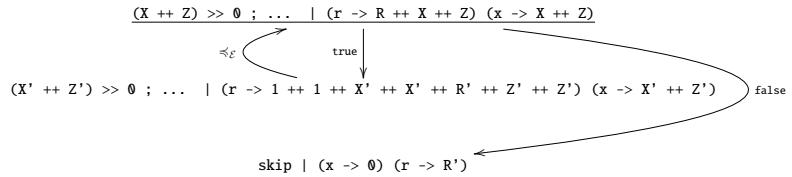
$$\{m \geq 0\} sum = sum\_upto(m) \{sum \geq m\}$$

is satisfied only if the corresponding logic program is satisfiable.

Ex1 - Constrained Horn Clauses for Program Verification TPLP 2022

```
mod CHC is protecting NAT-FVP * (op _+_ to _++_, op _>_ to _>>_) .
  ...
  eq (nat V) ; P | M = P | (M (V -> 0))  .            --- New Variable
  eq (V = E) ; P | M = E ; (V = {}) ; P | M  .        --- Assignment
  eq N ; (V = {}) ; P | M  = P | (M  (V -> N)) .      --- Cont'd
  eq V ; P | (M  (V -> N)) = N ; P | (M  (V -> N)) .  --- Variable
  eq (E1 > E2) ; P | M = E1 ; E2 ; > ; P | M .        --- Comparison
  eq N ; E2 ; > ; P | M = E2 ; N ; > ; P | M .        --- Cont'd
  eq N2 ; N1 ; > ; P | M = (N1 >> N2) ; P | M .       --- Cont'd
  eq (E1 + E2) ; P | M  = E1 ; E2 ; + ; P | M .       --- Addition
  eq N ; E2 ; + ; P | M = E2 ; N ; + ; P | M .        --- Cont'd
  eq N2 ; N1 ; + ; P | M = (N1 ++ N2) ; P | M .       --- Cont'd
  eq E - 1 ; P | M  = E ; - ; P | M  .                --- Predecessor
  eq N ; - ; P | M   = pred(N) ; P | M .              --- Cont'd
  eq while E {B} ; P | M = E ; while E {B} ; P | M .  --- While
  rl true ; while E {B} ; P | M => B ; while E {B} ; P | M [narrowing] .
  rl false ; while E {B} ; P | M => P | M [narrowing] .
endm
}
```

Ex1 - Constrained Horn Clauses for Program Verification TPLP 2022




```
Maude> {fold} vu-narrow {delay, filter}
  while (x > 0) {r = r + x ; x = x - 1} | (x -> X ++ Z) (r -> R)
  =>*
  skip | (x -> W) (r -> X) .

No solution.
rewrites: 79 in 16ms cpu (19ms real) (4725 rewrites/second)
```


Ex2 - Constrained Horn Clauses for Program Verification TPLP 2022

```
type tree = Leaf | Node of int * tree * tree ;;
let min x y = if x < y then x else y ;;
let rec min-leafdepth t = match t with
     | Leaf -> 0
     | Node(x,l,r) -> 1+min(min-leafdepth(l),min-leafdepth(r)) ;;
let rec left-drop n t = match t with
    | Leaf -> Leaf
    | Node(x,l,r) -> if n <= 0 then Node(x,l,r) else left-drop (n-1) l ;;
}
```

the Tree-Processing program, written in OCaml syntax is translated into a logic program and the property

$$\forall n, t : n \geq 0 \implies \mathit{min\text{-}leafdepth}(\mathit{left\text{-}drop}(n,t)) + n) \geq \mathit{min\text{-}leafdepth}(t) \qquad (1)$$

is satisfied only if the corresponding logic program is satisfiable.

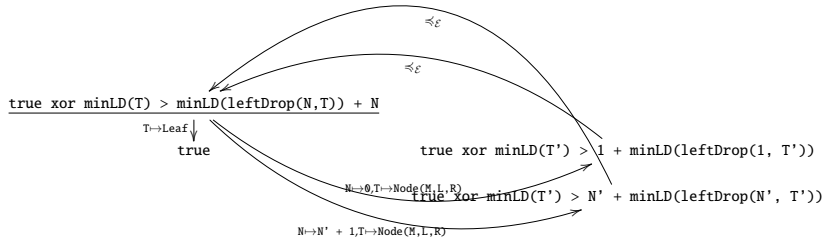
Ex2 - Constrained Horn Clauses for Program Verification TPLP 2022

```
mod TREE is protecting NAT-FVP .
  sort Tree .
  op Leaf : -> Tree .
  op Node : Nat Tree Tree -> Tree .
  vars N M : Nat . vars T L R : Tree .

  op minLD : Tree -> Nat .
  eq minLD(Leaf) = 0 .
  eq minLD(Node(N,L,R)) = 1 + min(minLD(L),minLD(R)) .
  rl minLD(Leaf) => 0 [narrowing] .
  rl minLD(Node(N,L,R)) => 1 + min(minLD(L),minLD(R)) [narrowing] .

  op leftDrop : Nat Tree -> Tree .
  eq leftDrop(N,Leaf) = Leaf .
  eq leftDrop(0,Node(M,L,R)) = Node(M,L,R) .
  eq leftDrop(N + 1,Node(M,L,R)) = leftDrop(N,L) .
  rl leftDrop(N,Leaf) => Leaf [narrowing] .
  rl leftDrop(0,Node(M,L,R)) => Node(M,L,R) [narrowing] .
  rl leftDrop(N + 1,Node(M,L,R)) => leftDrop(N,L) [narrowing] .
endm
}
```

Ex2 - Constrained Horn Clauses for Program Verification TPLP 2022




```
Maude> {fold} vu-narrow {delay, filter}
  not (minLeafDepth(T) > (minLD(leftDrop(N,T)) + N))  =>* false .

No solution.
rewrites: 19 in 1ms cpu (1ms real) (12541 rewrites/second)
```


Outline

Applications

- Variant-based unification itself
- Formal reasoning tools :
  - Relying on unification capabilities:
    - termination proofs
    - proofs of local confluence and coherence
  - Relying on narrowing capabilities:
    - narrowing-based theorem proving
    - testing
- Logical model checking (model checking with logical variables)
- Cryptographic protocol analysis:
  - the Maude-NPA tool (narrowing + unification in Maude)
  - the Tamarin and AKISS protocol analyzers also use Maude capabilities
- Program transformation: partial evaluation, slicing
- SMT based on narrowing or by variant generation.
- Narrowing-based Theorem Prover NuITP
- Deductive Model Checking DMCheck

Thank you!

More information in the Maude webpage.

NuITP: An Inductive Theorem Prover for Maude

F. Durán[1], S. Escobar[3], J. Meseguer[2], J. Sapiña[3]
[1]Universidad de Málaga
[2]University of Illinois at Urbana-Champaign
[3]Universitat Politècnica de València

2nd Workshop on Logic, Algebra and Category Theory: LAC 2025
Fukuoka, September 29 – October 3, 2025

Outline

1. Introduction
2. Examples
3. Gilbreath's card trick
4. Inference rules
5. Conclusions

Introduction

- Inductive theorem proving for equational programs has two problems:
  - Expressiveness. Types and subtypes, conditional equations, and rewriting modulo associativity and/or commutativity and/or identity axioms.
  - Scalability. The theorem prover should scale up to large proofs. Tactics, auxiliary lemmas, automatic reasoning.
- Maude has been endowed with new symbolic equational reasoning techniques during the last 15 years that tackle expressiveness but also scalability.
  - Equality predicates, order-sorted conditional narrowing, variant narrowing, variant unification, variant satisfiability, and order-sorted congruence closure.
- NuITP is a next-generation inductive theorem prover based on inductive order-sorted first-order logic. Theoretical foundations in [Meseguer-JLAMP2025].

Features

- Equational Theories in Maude $(\Omega, B_\Omega, \emptyset) \subseteq (\Sigma_1, B_1, E_1) \subseteq (\Sigma, B, E)$
- B any combination of associativity (A), commutativity (C) and identity (U)
- E a set of convergent conditional equations
- Constructor subtheory $(\Omega, B_\Omega, \emptyset)$ and Finite Variant subtheory (Σ_1, B_1, E_1)
- Formulas $(w_1 = w_1' \wedge \ldots \wedge w_n = w_n') \rightarrow$
  $(u_1^1 = v_1^1 \vee \ldots \vee u_{m_1}^1 = v_{m_1}^1) \wedge \ldots \wedge (u_1^k = v_1^k \vee \ldots \vee u_{m_k}^k = v_{m_k}^k)$
- Reduction path ordering (RPO) given by user via annotations
- Generator sets over constructors
- Proof tactics given by user
- Internalization of previously proved auxiliary lemmas
- Automatic Equality Predicate Simplification

Peano

```
fmod PEANO+ADD-NO-ORDER is
  sorts Nat NzNat .
  subsorts NzNat < Nat .

  op 0 : -> Nat .
  op s_ : Nat -> NzNat .

  op _+_ : Nat Nat -> Nat [ assoc comm ] .
  eq N:Nat + 0 = N:Nat .
  eq N:Nat + s M:Nat = s(N:Nat + M:Nat) .
endfm
```

Peano

```
fmod PEANO+ADD-NO-ORDER is
  sorts Nat NzNat .
  subsorts NzNat < Nat .

  op 0 : -> Nat [ ctor ] .
  op s_ : Nat -> NzNat [ ctor ] .

  op _+_ : Nat Nat -> Nat [ assoc comm ] .
  eq N:Nat + 0 = N:Nat .
  eq N:Nat + s M:Nat = s(N:Nat + M:Nat) .
endfm
```

Peano

```
fmod PEANO+ADD-WITH-ORDER is
  sorts Nat NzNat .
  subsorts NzNat < Nat .

  op 0 : -> Nat [ ctor metadata "1" ] .
  op s_ : Nat -> NzNat [ ctor metadata "2" ] .

  op _+_ : Nat Nat -> Nat [ assoc comm metadata "3" ] .
  eq N:Nat + 0 = N:Nat .
  eq N:Nat + s M:Nat = s(N:Nat + M:Nat) .
endfm
```

Peano: associativity of addition

```
NuITP> set goal X:Nat + (Y:Nat + Z:Nat) = (X:Nat + Y:Nat) + Z:Nat .

  Initial goal set.

  Goal Id: 0
  Skolem Ops:
    None
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    None
  Goal:
    ($1:Nat + ($2:Nat + $3:Nat)) = (($1:Nat + $2:Nat) + $3:Nat)
```

Peano: associativity of addition

```
NuITP> apply gsi to 0 on $3 with 0 ;; s(K:Nat) .

  Generator Set Induction (GSI) applied to goal 0.

  Goal Id: 0.1
  Skolem Ops:
    None
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    None
  Goal:
    ($1:Nat + ($2:Nat + 0)) = (($1:Nat + $2:Nat) + 0)

  Goal Id: 0.2
  Skolem Ops:
    $4.Nat
  Executable Hypotheses:
    (($1:Nat + $2:Nat) + $4) => ($1:Nat +($2:Nat + $4))
  Non-Executable Hypotheses:
    None
  Goal:
    ($1:Nat + ($2:Nat + s $4)) = (($1:Nat + $2:Nat) + s $4)
```

Peano: associativity of addition

```
BuiTP> apply eps to 0.1 .

  Equality Predicate Simplification (EPS) applied to goal 0.1.

  Goal 0.1.1 has been proved.

  Unproved goals:

  Goal Id: 0.2
  Skolem Ops:
    $4.Nat
  Executable Hypotheses:
    (($1:Nat + $2:Nat) + $4) => ($1:Nat + ($2:Nat + $4))
  Non-Executable Hypotheses:
    None
  Goal:
    ($1:Nat + ($2:Nat + s $4)) = (($1:Nat + $2:Nat) + s $4)

  Total unproved goals: 1
```

Peano: associativity of addition

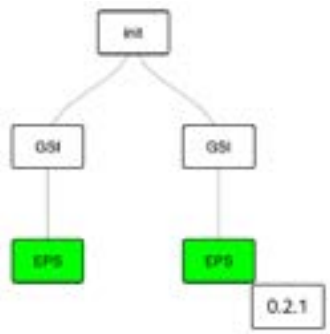
```
BuiTP> apply eps to 0.2 .

  Equality Predicate Simplification (EPS) applied to goal 0.2.

  Goal 0.2.1 has been proved.

  qed
```

Peano: associativity of addition



Peano: commutativity of addition, a new proof

```
NuITP> apply gsi! to 0 on $3 with 0 ;; s(K:Nat) .

  Generator Set Induction with Equality Predicate Simplification (GSI!)
  applied to goal 0.

  Goals 0.1.1 and 0.2.1 have been proved.

  qed
```

Peano: commutativity of addition

```
NuITP> set goal (X:Nat + Y:Nat = Y:Nat + X:Nat) .

  Initial goal set.

  Goal Id: 0
  Skolem Ops:
    None
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    None
  Goal:
    ($1:Nat + $2:Nat) =($2:Nat + $1:Nat)
```

Peano: commutativity of addition

```
NuITP> apply gsi! to 0 on $1 with 0 ;; s K:Nat .

  Generator Set Induction with Equality Predicate Simplification (GSI!)
  applied to goal 0.

  Goal Id: 0.1.1
  Skolem Ops:
    None
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    None
  Goal:
    $2:Nat =(0 + $2:Nat)

  Goal Id: 0.2.1
  Skolem Ops:
    $3.Nat
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    ($3 + $2:Nat) = ($2:Nat + $3)
  Goal:
    s($2:Nat + $3) = (s $3 + $2:Nat)
```

Peano: commutativity of addition

```
NuITP> apply gsi! to 0.1.1 on $2 with 0 ;; s K:Nat .

  Generator Set Induction with Equality Predicate Simplification (GSI!)
  applied to goal 0.1.1.

  Goals 0.1.1.1.1 and 0.1.1.2.1 have been proved.

  Unproved goals:

  Goal Id: 0.2.1
  Skolem Ops:
    $3.Nat
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    ($3 + $2:Nat) =($2:Nat + $3)
  Goal:
    s($2:Nat + $3) =(s $3 + $2:Nat)

  Total unproved goals: 1
```

Peano: commutativity of addition

```
NuITP> apply gsi! to 0.2.1 on $2 with 0 ;; s K:Nat .

  Generator Set Induction with Equality Predicate Simplification (GSI!)
  applied to goal 0.2.1.

  Goals 0.2.1.1.1 and 0.2.1.2.1 have been proved.

  qed
```

Commutativity and associativity of addition in one shot

```
NuITP> genset GEN-NAT for Nat is 0 ;; s N:Nat .

  Generator set GEN-NAT for sort Nat added.

  GEN-NAT (default):
    0
    s N:Nat
```

Commutativity and associativity of addition in one shot

```
NuITP> set goal (X:Nat + Y:Nat = Y:Nat + X:Nat) /\ X1:Nat + (Y1:Nat + Z:Nat) = (X1:Nat + Y1:Nat) + Z:Nat .

  Initial goal set.

  Goal Id: 0
  Generated By: init
  Skolem Ops:
    None
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    None
  Goal:
    (($2:Nat + $4:Nat) = ($4:Nat + $2:Nat)) /\
    ($1:Nat + ($3:Nat + $5:Nat)) = (($1:Nat + $3:Nat) + $5:Nat)
```

Commutativity and associativity of addition in one shot

```
NuITP> apply gsi! to 0 .

  Generator Set Induction (GSI) with Equality Predicate Simplification applied to goal 0.

  Goals 0.1.1, 0.10.1, 0.11.1, 0.12.1, 0.13.1, 0.14.1, 0.15.1, 0.16.1, 0.17.1, 0.18.1,
        0.19.1, 0.2.1, 0.20.1, 0.21.1, 0.22.1, 0.23.1, 0.24.1, 0.27.1, 0.28.1, 0.29.1,
        0.3.1, 0.30.1, 0.31.1, 0.32.1, 0.5.1, 0.6.1, 0.7.1 and 0.8.1 have been
    proved.

  qed
```

Lists: associativity of concatenation

```
fmod LIST-APPEND is
  sorts Nat List .

  op 0 : -> Nat [ ctor metadata "1" ] .
  op s : Nat -> Nat [ ctor metadata "2" ] .

  op nil : -> List [ ctor metadata "3" ] .
  op _;_ : Nat List -> List [ ctor metadata "4" ] .

  op _@_ : List List -> List [ metadata "5" ] .
  eq nil @ L:List = L:List .
  eq (N:Nat ; L:List) @ Q:List = N:Nat ; (L:List @ Q:List) .
endfm
```

```
NuITP> set goal (L:List @ P:List) @ Q:List = L:List @ (P:List @ Q:List) .
```

```
NuITP> apply gsi! to 0 on $1 with nil ;; (m:Nat ; R:List) .

  Generator Set Induction with Equality Predicate Simplification (GSI!) applied to
  goal 0.

  Goals 0.1.1 and 0.2.1 have been proved.

  qed
```

Lists: reverse of non-empty lists

```
NuITP> gmaset GLIST for List is X:Elt L:List .

  Generator set GLIST for sort List added.

  GLIST (default):
    X:Elt L:List

NuITP> set goal rev(Q:List Y:Elt) = Y:Elt rev(Q:List) .

  Initial goal set.

  Goal Id: 0
  Generated By: init
  Skolem Ops:
    None
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    None
  Goal:
    rev($1:List $2:Elt) = $2:Elt rev($1:List)

NuITP> apply ni! to 0 on $1 .

  Generator Set Induction (SSI) with Equality Predicate Simplification applied to goal 0.

  Goal 0.1.1 has been proved.

  qed
```

```
fmod REVERSING-LISTS is
  sorts Elt List .
  subsort Elt < List .

  op __ : List List -> List [ ctor assoc metadata "1" ] .

  op rev : List -> List [ metadata "2" ] .
  eq rev(X:Elt) = X:Elt .
  eq rev(X:Elt L:List) = rev(L:List) X:Elt .
endfm
```

Lists: reverse of non-empty lists with ni

```
fmod REVERSING-LISTS is
  sorts Elt List .
  subsort Elt < List .

  op __ : List List -> List [ ctor assoc metadata "1" ] .

  op rev : List -> List [ metadata "2" ] .
  eq rev(X:Elt) = X:Elt .
  eq rev(X:Elt L:List) = rev(L:List) X:Elt .
endfm
```

```
NuITP> set goal rev(Q:List Y:Elt) = Y:Elt rev(Q:List) .

NuITP> apply ni! to 0 on rev($1:List $2:Elt) .

  Narrowing Induction (NI) with Equality Predicate Simplification applied to goal 0.

  Goals 0.1.1 and 0.2.1 have been proved.

  qed
```

Lists: reverse of non-empty lists with ni

```
fmod REVERSING-LISTS is
  sorts Elt List .
  subsort Elt < List .

  op __ : List List -> List [ ctor assoc metadata "1" ] .

  op rev : List -> List [ metadata "2" ] .
  eq rev(X:Elt) = X:Elt .
  eq rev(X:Elt L:List) = rev(L:List) X:Elt .
endfm
```

```
NuITP> show goal 0.1 .

  Goal Id: 0.1
  Generated By: NI
  Skolem Ops:
    $3:Elt
    $4:Elt
    $5:List
  Executable Hypotheses:
    rev($5 $4) => $4 rev($5)
  Non-Executable Hypotheses:
    None
  Goal:
    ($4 rev($3 $5)) = rev($5 $4) $3

NuITP> show goal 0.2 .

  Goal Id: 0.2
  Generated By: NI
  Skolem Ops:
    None
  Executable Hypotheses:
    None
  Non-Executable Hypotheses:
    None
  Goal:
    ($4:Elt rev($3:Elt)) = rev($4:Elt) $3:Elt
```

Gilbreath's card trick (1/5)

- Norman L. Gilbreath's principle
- Given an initial deck of cards with some adequate properties, after a random shuffle the resulting deck will preserve some of those properties.
- In the Gilbreath's card trick, given an initial deck of cards with alternating colors (e.g., red and black), after shuffling it once, if we deal the resulting deck in pairs, each pair will always contain one card of each color.

F. Durán, S. Escobar, J. Meseguer, J. Sapiña: NuITP: An Inductive Theorem Prover for Equational Program Verification. PPDP 2024: 6:1-11

Gilbreath's card trick (2/5)

1. Begin with an even deck of cards so that they are sorted alternating colors (red and black);
2. Split the deck in two, not necessarily equal piles;
3. If the bottom cards of each pile are equal, take one of the cards and move (rotate) it to the top of that pile;
4. Riffle both piles (the shuffle does not need to be perfect); and
5. Deal the resulting deck in pairs. All pairs will always have a card of each color (e.g., either red-black or black-red cards).

Gilbreath's card trick (3/5)

```
op paired : Card Card -> Boolean [metadata "6"] .
eq paired(red, black) = True [variant] .
eq paired(black, red) = True [variant] .
eq paired(C, C) = False [variant] .

op opposite : List List -> Boolean [metadata "7"] .
eq opposite(nil, L) = False [variant] .
eq opposite(L, nil) = False [variant] .
eq opposite(C1 L1, C2 L2) = paired(C1, C2) [variant] .

op alter : List -> Boolean [metadata "8"] .
eq alter(nil) = True .
eq alter(C) = True .
ceq alter(C1 C2 L) = alter(C2 L) if paired(C1, C2) = True .
ceq alter(C1 C2 L) = False if paired(C1, C2) = False .

op pairedList : List -> Boolean [metadata "9"] .
eq pairedList(nil) = True .
eq pairedList(C) = False .
eq pairedList(C C L) = False .
ceq pairedList(C1 C2 L) = pairedList(L) if paired(C1, C2) = True .

op even : List -> Boolean [metadata "11"] .
eq even(nil) = True .
eq even(C) = False .
eq even(L1 C1 L2 C2 L3) = even(L1 L2 L3) .

op rotate : List -> List [metadata "13"] .
eq rotate(nil) = nil [variant] .
eq rotate(C L) = L C [variant] .
```

Gilbreath's card trick (4/5)

```
eq shuffle(nil, nil, nil) = True .
eq shuffle(nil, nil, C3 L3) = False .
eq shuffle(C1 L1, L2, nil) = False .
ceq shuffle(C1 L1, nil, C3 L3) = False if paired(C1, C3) = True .
ceq shuffle(C1 L1, nil, C3 L3) = shuffle(L1, nil, L3)
  if paired(C1, C3) = False .
eq shuffle(L1, C2 L2, nil) = False .
ceq shuffle(nil, C2 L2, C3 L3) = False if paired(C2, C3) = True .
ceq shuffle(nil, C2 L2, C3 L3) = shuffle(nil, L2, L3)
  if paired(C2, C3) = False .
```

```
ceq shuffle(C1 L1, C2 L2, C3 L3) = True
  if paired(C1, C3) = False
  /\ shuffle(L1, C2 L2, L3) = True .
ceq shuffle(C1 L1, C2 L2, C3 L3) = True
  if paired(C2, C3) = False
  /\ shuffle(C1 L1, L2, L3) = True .
ceq shuffle(C1 L1, C2 L2, C3 L3) = False
  if paired(C1, C3) = True
  /\ paired(C2, C3) = True .
ceq shuffle(C1 L1, C2 L2, C3 L3) = False
  if shuffle(L1, C2 L2, L3) = False
  /\ shuffle(C1 L1, L2, L3) = False .
ceq shuffle(C1 L1, C2 L2, C3 L3) = False
  if paired(C1, C3) = True
  /\ shuffle(C1 L1, L2, L3) = False .
ceq shuffle(C1 L1, C2 L2, C3 L3) = False
  if paired(C2, C3) = True
  /\ shuffle(L1, C2 L2, L3) = False .
```

Gilbreath's card trick (5/5)

```
> set goal ((alter(L1:List L2:List) = True)
            /\ (even((L1:List L2:List)) = True)
            /\ (opposite(L1:List, L2:List) = True)
            /\ (shuffle(L1:List, L2:List, L3:List) = True))
           -> (pairedList(L3:List) = True) .
```

```
> set goal ((alter(L1:List L2:List) = True)
            /\ (even(L1:List L2:List) = True)
            /\ (opposite(L1:List, L2:List) = False)
            /\ (shuffle(L1:List, L2:List, L3:List) = True))
           -> (pairedList(rotate(L3:List)) = True) .
```

	User Cmnds.		R. Applied		G. Generated	
Rule	Goal 1	Goal 2	Goal 1	Goal 2	Goal 1	Goal 2
CAS	100	119	118	145	472	580
CS			70	81	70	81
CUT	9	14	9	14	18	28
CVUL			13	23	32	48
EPS			773	958	773	958
GND			24	23	48	46
LSB			112	104	112	104
NI	12	15	12	15	84	120
NS	18	23	18	23	256	288
RST			6	5	6	5
UFREE	1	1	1	1	8	8
Total	140	172	1156	1392	1879	2266

NuITP and its Inference Rules

- Simplification rules
  - Equality Predicate Simplification (EPS),
  - Constructor Variant Unification Left (CVUL),
  - Constructor Variant Unification Failure Right (CVUFR),
  - Substitution Left and Right (SUBL, SUBR),
  - Narrowing Simplification (NS),
  - Clause Subsumption (CS),
  - Equation Rewriting (EQ),
  - Inductive Congruence Closure (ICC), and
  - Variant Satisfiability (VARSAT).
- Inductive rules
  - Generator Set Induction (GSI),
  - Narrowing Induction (NI),
  - Lemma Enrichment (LE),
  - Split (SP),
  - Case (CAS),
  - Variable Abstraction (VA), and
  - Cut.

The inference rules of this system transform inductive goals of the form

$$[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \rightarrow \Lambda$$

Generator Set Induction (GSI)

The **GSI** rule generalizes standard structural induction on constructors.

$$\frac{\{[\overline{X} \uplus \overline{Y}_u, \mathcal{E}, H \& H_u] \Vdash (\Gamma \to \bigwedge_{j\in J} \Delta_j)\{z \mapsto \overrightarrow{u^\bullet}\}\}_{u\in G}}{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \to \bigwedge_{j\in J} \Delta_j}$$

where $z \in vars(\Gamma \to \bigwedge_{j\in J}\Delta_j)$ has sort s, $\{u_1,\ldots,u_n\}$ is a B_0-generator set for s, with B_0 the non-unit axioms of the theory $\vec{\mathcal{E}}$,
Y_i are the variables of the u_i, $Y_i = vars(u_i)$, for $1 \leqslant i \leqslant n$, and
the induction hypotheses H_i are $H_i = \{(\Gamma \to \Delta_j)\{z \to v\} \mid v \in PST_{B_0,s}(u_i) \wedge j \in J\}$
where $PST_{B_0,s}(u)$ are the proper B_0-subterms of u

Generator Set Induction (GSI)

$$\frac{\{[\overline{X} \uplus \overline{Y}_{\vec{u}}, \mathcal{E}, H \& H_{\vec{u}}] \Vdash (\Gamma \to \bigwedge_{j\in J} \Delta_j)\{\vec{z} \mapsto \overrightarrow{\vec{u}}\}\}_{\vec{u}\in G_1\times\ldots\times G_n}}{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \to \bigwedge_{j\in J} \Delta_j}$$

where, (i) $\vec{z}$ (resp. $\vec{s}$) denotes the tuple $(z_1,\ldots,z_n)$ (resp. $(s_1,\ldots,s_n)$),
(ii) for $\vec{u} = (u_1,\ldots,u_n)$, $\{\vec{z} \mapsto \overrightarrow{\vec{u}}\}$ denotes the substitution $\{z_1 \mapsto \overrightarrow{u_1}, \ldots, z_n \mapsto \overrightarrow{u_n}\}$,
(iii) $G_1 \times \ldots \times G_n$ is the cartesian product of B_0-generator sets G_i for sorts s_i, $1 \leq i \leq n$, all having *fresh* variables, and such that for each i, j, $1 \leq i < j \leq n$, $vars(G_i) \cap vars(G_j) = \emptyset$, and
(iv) $Y_{\vec{u}} = vars(\vec{u})$.

Case (CAS)

$$\frac{\left\{[\overline{X}, \mathcal{E}, H] \Vdash (\Gamma \to \Lambda)\{z \mapsto u_i\}\right\}_{1\leq i\leq n}}{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \to \Lambda}$$

Narrowing simplification (NS)

The **NS** rule performs one step of symbolic evaluation on a term $f(\vec{v})$ of an equality of the form $f(\vec{v}) = u$ appearing anywhere in a goal $\Gamma \to \Lambda$.

(1) f is a non-constructor function symbol, so that its defining equations are sufficiently complete,
(2) the argument subterms $\vec{v}$ are constructor terms,
(3) term u belongs to the FVP subtheory $\mathcal{E}_1$ of the module's theory $\mathcal{E}$, and
(4) the equations defining f, oriented as rewrite rules, have the form $\{[i]: f(\vec{u_i}) \to r_i \text{ if } \Gamma_i\}_{i \in I}$, where the $\vec{u_i}$ are construtor terms and the r_i are terms in $\mathcal{E}_1$.

$$\frac{\{[\overline{X} \uplus \overline{Y}_{i,j}, \mathcal{E}, H \& \widehat{H'}] \Vdash (\Gamma_i, (\Gamma \to \Lambda)[r_i = u]_p)\overline{\alpha}_{i,j}\}_{i \in I_0}^{j \in J_i}}{[\overline{X}, \mathcal{E}, H] \Vdash (\Gamma \to \Lambda)[f(\vec{v}) = u]_p}$$

where $I_0 \subseteq I$ is the subset of rule labels of the rules defining f such that there is at least one B-unifier of $f(\vec{v})$ and $f(\vec{u_i})$, and
$\{\alpha_{i,j}\}_{j \in J_i}$ denotes a complete set of B-unifiers of the equation $f(\vec{v}) = f(\vec{u_i})$.

Narrowing induction (NI)

$$\frac{\{[\overline{X} \uplus \overline{Y}_{i,j}, \mathcal{E},\ H \& H_{i,j}] \Vdash (\Gamma_i, (\Gamma \to \bigwedge_{l \in L} \Delta_l)[r_i]_p)\overline{\alpha}_{i,j}\}_{i \in I_0}^{j \in J_i}}{[\overline{X}, \mathcal{E}, H] \Vdash (\Gamma \to \bigwedge_{l \in L} \Delta_l)[f(\vec{v})]_p}$$

Constructor Variant Unification Left (CVUL)

The **CVUL** rule unifies those equations in the condition that belong to $\mathcal{E}_1$.

$$\frac{\{[\overline{X} \uplus \overline{Y}_{\alpha},\ \mathcal{E},\ H \& \widehat{H'}] \Vdash (\Gamma' \to \Lambda)\overline{\alpha}\}_{\alpha \in Unif_{\mathcal{E}_1}^{\Omega}(\Gamma^{\circ})}}{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma, \Gamma' \to \Lambda}$$

where Γ is a conjunction of $\mathcal{E}_1$-equalities,
Γ' does not contain any $\mathcal{E}_1$-equalities, and
$Unif_{\mathcal{E}_1}^{\Omega}(\Gamma)$ denotes the set of constructor $\mathcal{E}_1$-unifiers of Γ

Constructor Variant Unification Failure Right (CVUFR)

The **CVUFR** rule may be seen as a restricted version of the more general **CVUL** rule.

$$\frac{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \to \Lambda \wedge \Delta}{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \to \Lambda \wedge (u = v, \Delta)}$$

where $u = v$ is a $\mathcal{E}_{|_{\mathbb{X}}}$-equality and $Unif^{\Omega}_{\mathcal{E}_1}((u = v)^{\omega}) = \varnothing$.

Equality Simplification (EPS) and Proof Search (ES)

$$\frac{\{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma'_i \to \Lambda'_i\}_{i \in I}}{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \to \Lambda}$$

$$\frac{[\overline{X}, \mathcal{E}, H] \Vdash (\Gamma \to \Lambda)[\top]_{\vec{p}}}{[\overline{X}, \mathcal{E}, H] \Vdash (\Gamma \to \Lambda)[u = v]^{B}_{\vec{p}}} \text{ if } E \cup eq(H) \cup B \vdash_{[L_1, L_2], \phi, mod, k} u = v$$

Lemma Enrichment (LE) and Split (SP)

$$\frac{[\overline{X}_0, \mathcal{E}, H_0] \Vdash \Gamma' \to \bigwedge_{j \in J} \Delta'_j \qquad [\overline{X}, \mathcal{E}, H] \Vdash H_0 \qquad [\overline{X}, \mathcal{E}, H \& \{\Gamma' \to \Delta'_j\}_{j \in J}] \Vdash \Gamma \to \Lambda}{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \to \Lambda}$$

$$\frac{\{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma_i \theta, \Gamma \to \Lambda\}_{i \in I} \qquad [\overline{X}, \mathcal{E}, H] \Vdash H_0 \qquad [\overline{X}_0, \mathcal{E}, H_0] \Vdash cnf(\bigvee_{i \in I} \Gamma_i)}{[\overline{X}, \mathcal{E}, H] \Vdash \Gamma \to \Lambda}$$

Cut and Variable Abstraction

$$\frac{[\overline{X},\mathcal{E},H] \Vdash \Gamma \to \Gamma' \qquad [\overline{X},\mathcal{E},H] \Vdash \Gamma,\Gamma' \to \Lambda}{[\overline{X},\mathcal{E},H] \Vdash \Gamma \to \Lambda}$$

$$\frac{[\overline{X},\mathcal{E},H] \Vdash z = u,\Gamma' \to \Lambda'}{[\overline{X},\mathcal{E},H] \Vdash (\Gamma \to \Lambda)[u]_p}$$

Induction Congruence Closure

The **ICC** rule is a *modus ponens* type of rule that tries to discharge a goal $[\overline{X},\mathcal{E},H] \Vdash \Gamma \to \Lambda$ by assuming its condition Γ to prove its conclusion Λ.

$$\frac{\{[\overline{X},\mathcal{E},H] \Vdash \Gamma_i^\sharp \to \Lambda_i^\sharp\}_{i\in I}}{[\overline{X},\mathcal{E},H] \Vdash \Gamma \to \Lambda}$$

where: (i) by definition, $\overline{\Lambda}_i^\sharp \in \overline{\Lambda}|_{\mathcal{E}_{\overline{X}\cup\overline{Y}_U} \cup \Pi_{\Gamma_U}^+ \cup \Pi_{\Gamma_i}^{\sharp-}}$ and we always pick $\overline{\Lambda}_i^\sharp = \top$ if $\top \in \overline{\Lambda}|_{\mathcal{E}_{\overline{X}\cup\overline{Y}_U} \cup \Pi_{\Gamma_U}^+ \cup \Pi_{\Gamma_i}^{\sharp-}}$;

(ii) $\Gamma_i^\sharp \to \Lambda_i^\sharp$ is obtained from $\overline{\Gamma}_i^\sharp \to \overline{\Lambda}_i^\sharp$ by converting back the Skolem constants associated to the variables of $\Gamma \to \Lambda$ into those same variables, and

(iii) the case $\overline{\Gamma}^\sharp = \bot$ is the case when there are no conjunctions in the disjunctive normal form $\overline{\Gamma}^\sharp$

Conclusions

- We have introduced the NuITP, explained its most commonly used inference rules, and illustrated their use in proving the card trick benchmark
- Main objective expressiveness & scalability

Future work

- Improve strategy language
- More expressiveness
- Improve user interface
- Parallelization
- Backend for other tools
- Proof certification

Minimally undecidable reducts of Tarski's relation algebras

Marcel Jackson (joint with Robin Hirsch and Jaš Šemrl)

La Trobe University, Melbourne, Australia

Relation Algebras and results

Tarski relation algebras

Binary relations on a set X

The subsets $\wp(X \times X)$ of $X \times X$ forms:

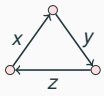
- a Boolean algebra wrt $\cap, \cup, -, \varnothing, X \times X$
- a involuted monoid with respect to $\circ$, converse $^{-1}$ and $=_X$ (identity relation)
- some properties relating the "action" part to the "logic" part

Tarski relation algebras

Binary relations on a set X

The subsets $\wp(X \times X)$ of $X \times X$ forms:

- a Boolean algebra wrt $\cap, \cup, -, \varnothing, X \times X$
- a involuted monoid with respect to $\circ$, converse $^{-1}$ and $=_X$ (identity relation)
- some properties relating the "action" part to the "logic" part
  - Distributivity of $\circ$ over $\cup$
  - Peircean law: $x \circ y \subseteq -z^{-1} \Leftrightarrow y \subseteq z \leq -y^{-1}$



x ; y can't sit over z^{-1}
$\Leftrightarrow$
y ; z can't sit over x^{-1}

Tarski relation algebras

Binary relations on a set X

The subsets $\wp(X \times X)$ of $X \times X$ forms:

- a Boolean algebra wrt $\cap, \cup, -, \varnothing, X \times X$
- a involuted monoid with respect to $\circ$, converse $^{-1}$ and $=_X$ (identity relation)
- some properties relating the "action" part to the "logic" part

History

- 1800s: de Morgan, Peirce, Schröder (logic of relatives)
- 1940s: Tarski (relation algebras)

Tarski relation algebras

Binary relations on a set X

The subsets $\wp(X \times X)$ of $X \times X$ forms:

- a Boolean algebra wrt $\cap, \cup, -, \varnothing, X \times X$
- a involuted monoid with respect to $\circ$, converse $^{-1}$ and $=_X$ (identity relation)
- some properties relating the "action" part to the "logic" part

(Tarski 1941) "*Is it the case that every sentence of the calculus of relations which is true in every domain of individuals is derivable from the axioms adopted under the second method? This problem presents some difficulties and still remains open. I can only say that I am practically sure that I can prove with the help of the second method, all of the hundreds of theorems to be found in Schröder's Algebra und Logic der Relative*"

Tarski relation algebras

Binary relations on a set X

The subsets $\wp(X \times X)$ of $X \times X$ forms:

- a Boolean algebra wrt $\cap, \cup, -, \varnothing, X \times X$
- a involuted monoid with respect to $\circ$, converse $^{-1}$ and $=_X$ (identity relation)
- some properties relating the "action" part to the "logic" part

The next problem is the so-called representation problem. Is every model of the axiom system of the calculus of relations isomorphic with a class of binary relations which contains the relations $1, 0, 1', 0'$ *and is closed under all the operations considered in this calculus?*

Tarski relation algebras

Binary relations on a set X

The subsets $\wp(X \times X)$ of $X \times X$ forms:

- a Boolean algebra wrt $\cap, \cup, -, \varnothing, X \times X$
- a involuted monoid with respect to $\circ$, converse $^{-1}$ and $=_X$ (identity relation)
- some properties relating the "action" part to the "logic" part

Fundamental representability question

When is an abstract algebra $\langle A, +, \cdot, -, ;, \smile, 0, 1, 1' \rangle$ isomorphic to an algebra of binary relations?

Tarski relation algebras

Binary relations on a set X

The subsets $\wp(X \times X)$ of $X \times X$ forms:

- a Boolean algebra wrt $\cap, \cup, -, \varnothing, X \times X$
- a involuted monoid with respect to $\circ$, converse $^{-1}$ and $=_X$ (identity relation)
- some properties relating the "action" part to the "logic" part

. . . temporarily skipping 60 years of history. . .

Robin Hirsch and Ian Hodkinson 2001

This problem is undecidable for finite algebras

The question of representability of finite algebras of finite sets remains open

Examples

Point algebra

The 8-element relation algebra with atoms $<, >, =$

;	$<$	$>$	$=$
$<$	$<$	1	$<$
$>$	1	$>$	$>$
$=$	$<$	$>$	$=$

This is representable by giving these symbols their usual interpretation on a dense linear order such as $\langle \mathbb{Q}, < \rangle$, but not on any finite set

Examples

Lyndon algebras

Boolean atoms are $1', c_1, \ldots, c_n$ where $c_i \,;\, c_i = 1' + c_i$ and $c_i \,;\, c_j = -(c_i + c_j)$. Representable only over affine plane of order $n - 1$

The question of which orders an affine plane exists remains open

Examples

ZFC can be expressed equationally within the equational theory of relation algebras

Examples

ZFC can be expressed equationally within the equational theory of relation algebras

Axiom of Extensionality:

$$\forall x \forall y \big(\forall z (z \in x \leftrightarrow z \in y) \rightarrow x = y\big)$$

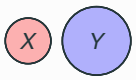
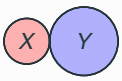
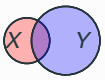
Relation Algebra:

$$(\in^{\smile} ;(-\in)) + ((-\in)^{\smile} ; \in) + 1' \approx 1$$

(In general it is known that the language of relation algebras captures the 3-variable fragment of the first order predicate calculus of binary relations)

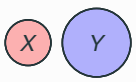
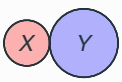
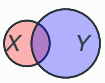
5

Region Connection Calculus RCC8

Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^{\smile}$ X
	X po Y		$X = Y$

6

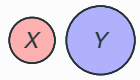
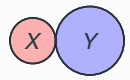
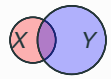
Region Connection Calculus RCC8

Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^{\smile}$ X
	X po Y		$X = Y$

6

Example

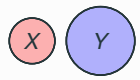
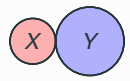
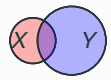
Region Connection Calculus RCC8

Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^\smile$ X
	X po Y		$X = Y$

Example
et

6

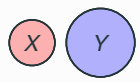
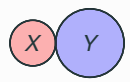
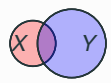
Region Connection Calculus RCC8

Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^\smile$ X
	X po Y		$X = Y$

Example
et ∘ *in*

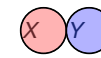
6

Region Connection Calculus RCC8

Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^\smile$ X
	X po Y		$X = Y$

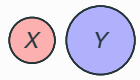
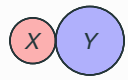
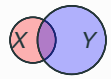
Example
et ∘ *in*

X et Y in Z



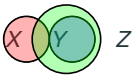
6

Region Connection Calculus RCC8

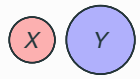
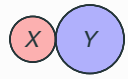
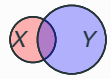
Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^\smile$ X
	X po Y	X, Y	$X = Y$

Example
et ∘ *in*

X et Y in Z

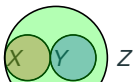


Region Connection Calculus RCC8

Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^\smile$ X
	X po Y	X, Y	$X = Y$

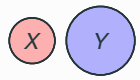
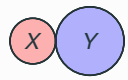
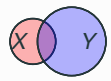
Example
et ∘ *in*

X et Y in Z



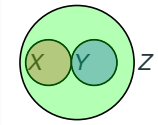
6

Region Connection Calculus RCC8

Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^\smile$ X
	X po Y	X, Y	$X = Y$

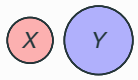
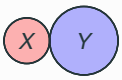
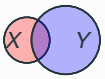
Example
et ∘ *in*

X et Y in Z

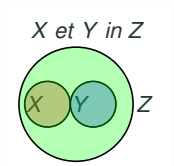


6

Region Connection Calculus RCC8

Example	Relation	Example	Relation
	X dis Y		X in Y Y over X
	X et Y		X it Y Y it$^{\smile}$ X
	X po Y		$X = Y$

Example
$et \circ in$
$=$
$\{po, it, in\}$



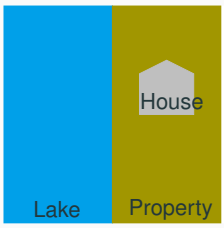
Qualitative reasoning example: the algebraic propagation algorithm

There is a house H

Qualitative reasoning example: the algebraic propagation algorithm

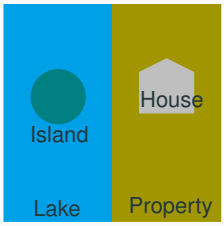
There is a house H
Properly within a property P

Qualitative reasoning example: the algebraic propagation algorithm



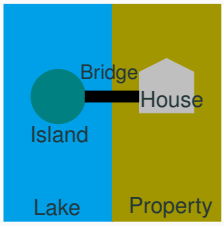
There is a house *H*
Properly within a property *P*
Bordering lake *L*

Qualitative reasoning example: the algebraic propagation algorithm



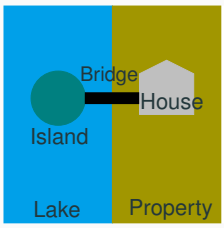
There is a house *H*
Properly within a property *P*
Bordering lake *L*
Containing island *I*

Qualitative reasoning example: the algebraic propagation algorithm



There is a house *H*
Properly within a property *P*
Bordering lake *L*
Containing island *I*
A bridge *B* connects *I* to *H*

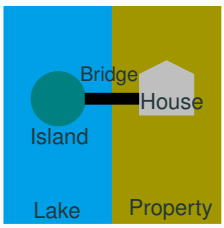
Qualitative reasoning example: the algebraic propagation algorithm



There is a house H
Properly within a property P
Bordering lake L
Containing island I
A bridge B connects I to H

How does B relate to L?

Qualitative reasoning example: the algebraic propagation algorithm

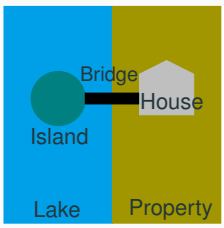


There is a house H
Properly within a property P
Bordering lake L
Containing island I
A bridge B connects I to H

How does B relate to L?

1. $B\ \{et, po\}\ I\ \{in\}\ L$

Qualitative reasoning example: the algebraic propagation algorithm

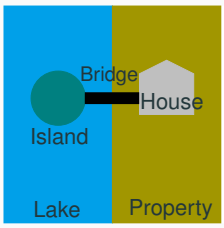


There is a house H
Properly within a property P
Bordering lake L
Containing island I
A bridge B connects I to H

How does B relate to L?

1. $B\ \{et, po\}\ I\ \{in\}\ L$
2. $\Rightarrow B\ \{po, it, in\}\ L$ (composing relations)

Qualitative reasoning example: the algebraic propagation algorithm

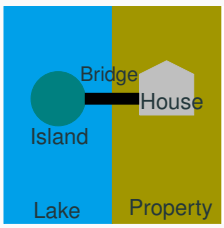


There is a house H
Properly within a property P
Bordering lake L
Containing island I
A bridge B connects I to H

How does B relate to L?

1. $B\ \{et, po\}\ I\ \{in\}\ L$
2. $\Rightarrow B\ \{po, it, in\}\ L$ (composing relations)
3. $B\ \{et\}\ H\ \{in\}\ P\ \{et\}\ L$

Qualitative reasoning example: the algebraic propagation algorithm

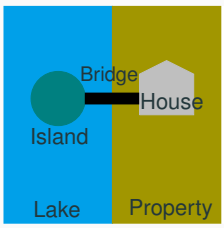


There is a house H
Properly within a property P
Bordering lake L
Containing island I
A bridge B connects I to H

How does B relate to L?

1. $B\ \{et, po\}\ I\ \{in\}\ L$
2. $\Rightarrow B\ \{po, it, in\}\ L$ (composing relations)
3. $B\ \{et\}\ H\ \{dis\}\ L$

Qualitative reasoning example: the algebraic propagation algorithm

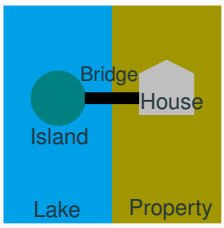


There is a house H
Properly within a property P
Bordering lake L
Containing island I
A bridge B connects I to H

How does B relate to L?

1. $B\ \{et, po\}\ I\ \{in\}\ L$
2. $\Rightarrow B\ \{po, it, in\}\ L$ (composing relations)
3. $B\ \{et\}\ H\ \{dis\}\ L$
4. $\Rightarrow B\ \{dis, et, po, over, it^{\smile}\}\ L$ (composing relations)

Qualitative reasoning example: the algebraic propagation algorithm

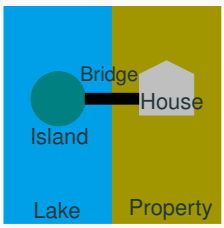


There is a house H
Properly within a property P
Bordering lake L
Containing island I
A bridge B connects I to H

How does B relate to L?

1. $B\ \{et, po\}\ I\ \{in\}\ L$
2. $\Rightarrow B\ \{po, it, in\}\ L$ (composing relations)
3. $B\ \{et\}\ H\ \{dis\}\ L$
4. $\Rightarrow B\ \{dis, et, po, over, it^{\smile}\}\ L$ (composing relations)
5. (2) and (4) give $B\ \{po, it, in\} \cap \{dis, et, po, over, it^{\smile}\}\ L$

Qualitative reasoning example: the algebraic propagation algorithm



There is a house H
Properly within a property P
Bordering lake L
Containing island I
A bridge B connects I to H

How does B relate to L?

1. $B\ \{et, po\}\ I\ \{in\}\ L$
2. $\Rightarrow B\ \{po, it, in\}\ L$ (composing relations)
3. $B\ \{et\}\ H\ \{dis\}\ L$
4. $\Rightarrow B\ \{dis, et, po, over, it^{\smile}\}\ L$ (composing relations)
5. (2) and (4) give $B\ \{po, it, in\} \cap \{dis, et, po, over, it^{\smile}\}\ L$
6. $\Rightarrow B\ \{po\}\ L$

Hirsch Hodkinson proof

Tiling algebras

Algebras encoding the tiling problem for square tiles

Hirsch Hodkinson proof

Tiling algebras

Algebras encoding the tiling problem for square tiles

Jónsson signature (also *allegories* in the sense of Peter Freyd)

The argument can be carried through using only the operations $\cdot, ;, {}^\smile$

How far down does undecidability of representability (UR) pervade?

Undecidability of representability for subreducts

Hirsch and J 2012: undecidability of representability for reducts

$\{+, \cdot, ;, 1'\}, \{\backslash, ;, 1'\}, \{\Rightarrow, ;, 1'\}, \{\leq, -, 1'\}$

Here, as usual, $x \backslash y := x \cdot (-y)$, $x \Rightarrow y := -x \vee y$ and $x \leq y$ means $x \cdot y = x$

Undecidability of representability for subreducts

Hirsch and J 2012: undecidability of representability for reducts

$\{+, \cdot, ;, 1'\}, \{\backslash, ;, 1'\}, \{\Rightarrow, ;, 1'\}, \{\leq, -, 1'\}$

Neuzerling 2016

$\{+, \cdot, ;\}, \{;, \leq, -\}$

Undecidability of representability for subreducts

Hirsch and J 2012: undecidability of representability for reducts

$\{+, \cdot, ;, 1'\}, \{\backslash, ;, 1'\}, \{\Rightarrow, ;, 1'\}, \{\leq, -, 1'\}$

Neuzerling 2016

$\{+, \cdot, ;\}, \{;, \leq, -\}$

Very small cases

$\{;, \Rightarrow\}$ (Lewis-Smith, Semrl 2023), $\{;, -\}$ (Hirsch, J, Šemrl 2022)

Undecidability of representability for subreducts

Hirsch and J 2012: undecidability of representability for reducts

$\{+, \cdot, ;, 1'\}, \{\backslash, ;, 1'\}, \{\Rightarrow, ;, 1'\}, \{\leq, -, 1'\}$

Neuzerling 2016

$\{+, \cdot, ;\}, \{;, \leq, -\}$

Main result (Hirsch-J-Šemrl, Semigroup Forum 111 (2025) 469–489)

1. The signature $\{;, -\}$ is a minimal subset of Tarski's having UR

Undecidability of representability for subreducts

Hirsch and J 2012: undecidability of representability for reducts

$\{+, \cdot, ;, 1'\}, \{\backslash, ;, 1'\}, \{\Rightarrow, ;, 1'\}, \{\leq, -, 1'\}$

Neuzerling 2016

$\{+, \cdot, ;\}, \{;, \leq, -\}$

Main result (Hirsch-J-Šemrl, Semigroup Forum 111 (2025) 469–489)

1. The signature $\{;, -\}$ is a minimal subset of Tarski's having UR
2. But there is an infinite chain of increasingly weak term reduct signatures, each with UR, but whose limit is $\{;\}$ with DR

Undecidability of representability for subreducts

Hirsch and J 2012: undecidability of representability for reducts

$\{+, \cdot, ;, 1'\}, \{\backslash, ;, 1'\}, \{\Rightarrow, ;, 1'\}, \{\leq, -, 1'\}$

Neuzerling 2016

$\{+, \cdot, ;\}, \{;, \leq, -\}$

Main result (Hirsch-J-Šemrl, Semigroup Forum 111 (2025) 469–489)

1. The signature $\{;, -\}$ is a minimal subset of Tarski's having UR
2. But there is an infinite chain of increasingly weak term reduct signatures, each with UR, but whose limit is $\{;\}$ with DR
3. Moreover, UR holds for a term reduct of a term reduct with DR

Some example open problems on decidability of representability

- The finite representability problem for $\{;, \cdot\}$ (Bredikhin and Schein 1978)
- $\{;, +\}$ (Andreka 1990s)
- $\{;, \smile\}$ (Schein 1974)
- $\{;, \leq, 1'\}$ (Hirsch, 2005)

Some example open problems on decidability of representability

- The finite representability problem for $\{;, \cdot\}$ (Bredikhin and Schein 1978)
- $\{;, +\}$ (Andreka 1990s)
- $\{;, \smile\}$ (Schein 1974)
- $\{;, \leq, 1'\}$ (Hirsch, 2005)

Theorem from Schein 1974

The free involuted semigroup is representable as binary relations

Some example open problems on decidability of representability

- The finite representability problem for $\{;, \cdot\}$ (Bredikhin and Schein 1978)
- $\{;, +\}$ (Andreka 1990s)
- $\{;, \smile\}$ (Schein 1974)
- $\{;, \leq, 1'\}$ (Hirsch, 2005)

Theorem from Schein 1974

The free involuted semigroup is representable as binary relations

These either involve $\smile$ but don't seem amenable to the tiling method, or avoid $\smile$ but seem incapable of fully encoding the the partial group embedding problem. All are nontrivial

Methods

Partial group embedding problem

Trevor Evans 1953

The uniform word problem in a class is Turing equivalent to the problem of deciding if partial algebras complete to full algebras in the class

Partial group embedding problem

Trevor Evans 1953

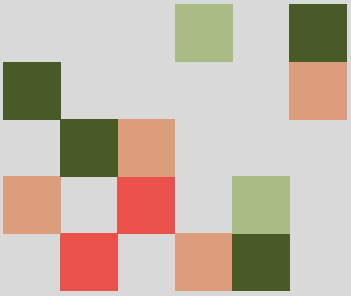
The uniform word problem in a class is Turing equivalent to the problem of deciding if partial algebras complete to full algebras in the class

Groups

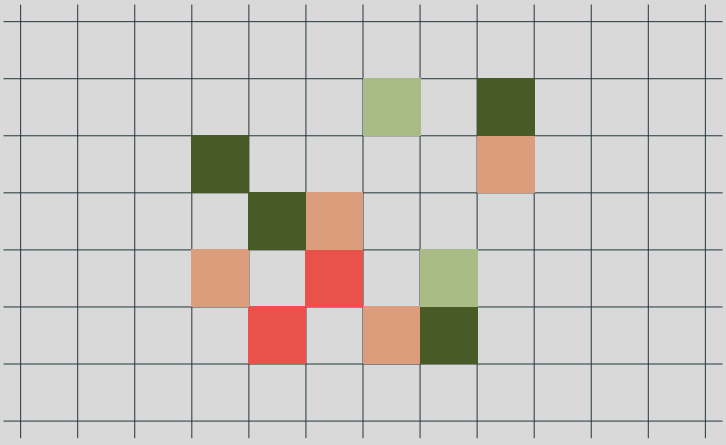
The uniform word problem for groups is undecidable (Novikov 1955, Boone 1958). The uniform word problem for finite groups is undecidable (Slobodskoĭ 1982)

In the particular case of groups, we may interpret “complete to full algebras” in a very flexible way

12



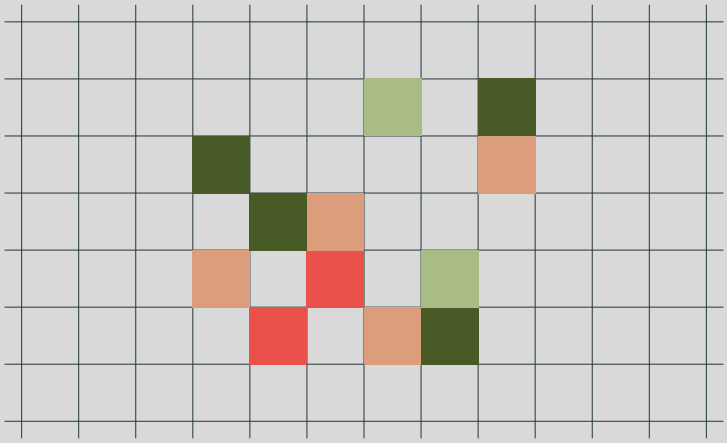
13



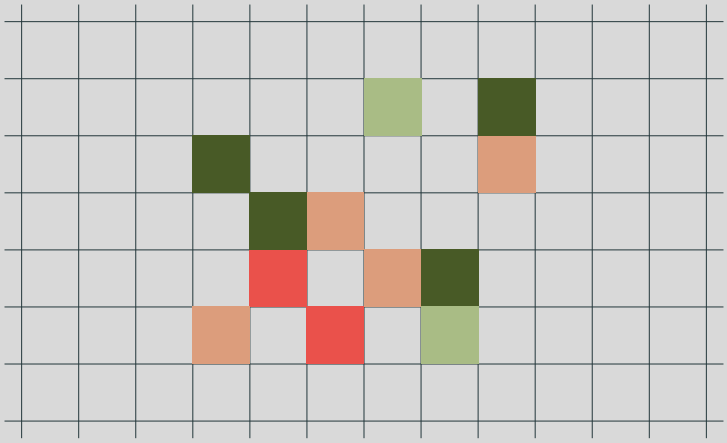
13

That 7-element example with 12 coloured squares is from Dietrich and Wanless 2018, after a 10-element example with 26 coloured squares in Hirsch and J (2012)

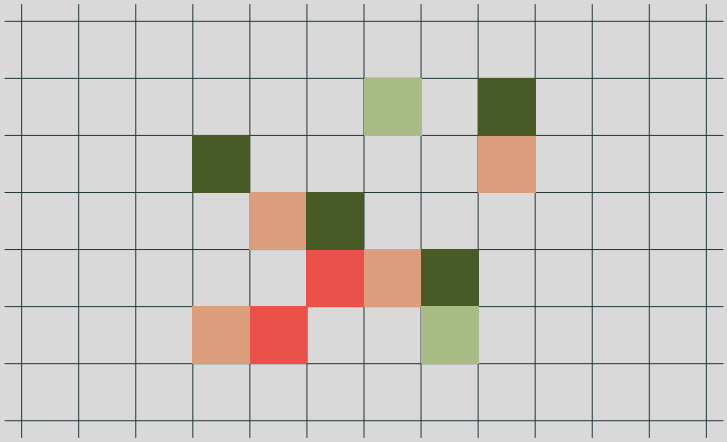
14



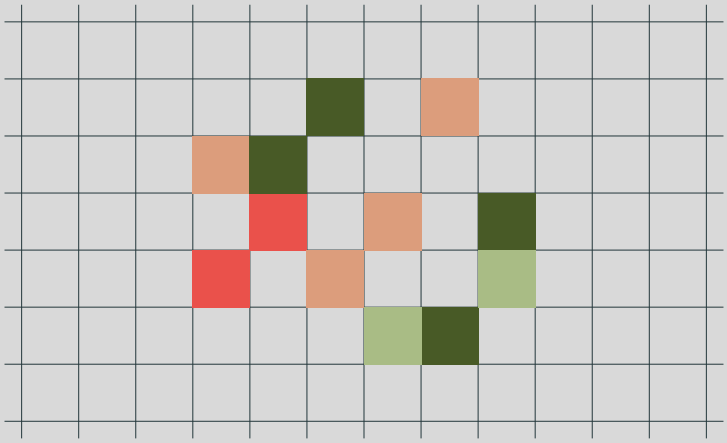
15



15



15



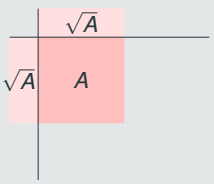
15

Square partial groups

Square partial group A

There is $e \in A$ and a subset $\sqrt{A} \subseteq A$ with

1. $e \cdot x = x \cdot e = x$ for $x \in \sqrt{A}$
2. $x \cdot y$ if and only if $x, y \in \sqrt{A}$ or $e \in \{x, y\}$
3. for each $x \in \sqrt{A}$ there is x' with $xx' = e = x'x$
4. $\sqrt{A} \cdot \sqrt{A} = A$



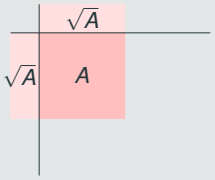
16

Square partial groups

Square partial group A

There is $e \in A$ and a subset $\sqrt{A} \subseteq A$ with

1. $e \cdot x = x \cdot e = x$ for $x \in \sqrt{A}$
2. $x \cdot y$ if and only if $x, y \in \sqrt{A}$ or $e \in \{x, y\}$
3. for each $x \in \sqrt{A}$ there is x' with $xx' = e = x'x$
4. $\sqrt{A} \cdot \sqrt{A} = A$



Theorem: the following are undecidable

Input: a square partial group A

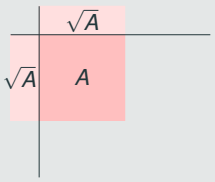
- does A embed into a group?
- does A embed into a finite group?

Square partial groups

Square partial group A

There is $e \in A$ and a subset $\sqrt{A} \subseteq A$ with

1. $e \cdot x = x \cdot e = x$ for $x \in \sqrt{A}$
2. $x \cdot y$ if and only if $x, y \in \sqrt{A}$ or $e \in \{x, y\}$
3. for each $x \in \sqrt{A}$ there is x' with $xx' = e = x'x$
4. $\sqrt{A} \cdot \sqrt{A} = A$



Theorem: the following are recursively inseparable

- finite square partial groups A that *do not* embed into a group
- finite square partial groups A that embed into finite groups

Green's relations

Definition: $\mathcal{L}$ (with $\mathcal{R}$ defined dually)

$a \leq_{\mathcal{L}} b$ if $\exists x\ xb = a$. Define the binary relation $\mathcal{L}$ by

$$a \mathcal{L} b \iff a \leq_{\mathcal{L}} b \text{ and } b \leq_{\mathcal{L}} a$$

Definition: $\mathcal{H}$

$\mathcal{H} = \mathcal{L} \cap \mathcal{R}$

Split systems

Given a square partial group **A**

Split system $\mathcal{A}$

$$\{a_{12} \mid a \in \sqrt{A}\} \cup \{a_{23} \mid a \in \sqrt{A}\} \cup \{a_{13} \mid a \in A\} \cup \{a_{ii} \mid a = e\}$$

with multiplication $a_{ij} \cdot a_{jk} = a_{ik}$

Theorem (Sapir, 1997)

It is undecidable to determine, given a split system $\mathcal{A}$, if there is a semigroup embedding $\mathcal{A}$ in which $\{a_{ij} \mid a \in A\}$ lie within an $\mathcal{H}$ class for each $i, j \in \{1, 2, 3\}$

18

Split system as a semigroup

Split system $\mathcal{A}$

$$\{a_{12} \mid a \in \sqrt{A}\} \cup \{a_{23} \mid a \in \sqrt{A}\} \cup \{a_{13} \mid a \in A\} \cup \{a_{ii} \mid a = e\}$$

with multiplication $a_{ij} \cdot a_{jk} = a_{ik}$

Note that if A is an actual group, then $\sqrt{A} = A$ and we could define a_{21}, a_{32}, a_{31} as well and obtain a Brandt groupoid $B_3(A)$ with inverses: $(a_{ij})^{\smile} = a_{ji}^{-1}$

19

Split system as a semigroup

Split system $\mathcal{A}$

$$\{a_{12} \mid a \in \sqrt{A}\} \cup \{a_{23} \mid a \in \sqrt{A}\} \cup \{a_{13} \mid a \in A\} \cup \{a_{ii} \mid a = e\}$$

with multiplication $a_{ij} \cdot a_{jk} = a_{ik}$

Note that if A is an actual group, then $\sqrt{A} = A$ and we could define a_{21}, a_{32}, a_{31} as well and obtain a Brandt groupoid $B_3(A)$ with inverses: $(a_{ij})^{\smile} = a_{ji}^{-1}$

As a semigroup $S(\mathcal{A})$

Add a 0 and let all undefined products be 0. Note that $a_{ij} \leq_{\mathcal{L}} e_{jj}$ and $a_{ij} \leq_{\mathcal{R}} e_{ii}$

19

Example

$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

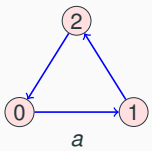
Example

$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes

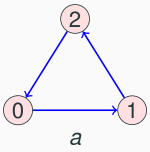
$\cdot$	e_1	a_1	b_1	e_2	a_2	b_2	e_3	a_3	b_3
e_1	0	0	0	e_3	a_3	b_3	0	0	0
a_1	0	0	0	a_3	b_3	e_3	0	0	0
b_1	0	0	0	b_3	e_3	c_3	0	0	0

As binary relations



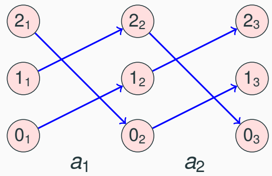
$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

As binary relations



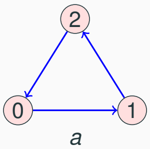
·	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



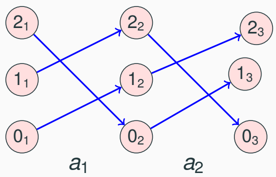
21

As binary relations



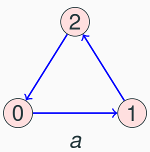
·	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



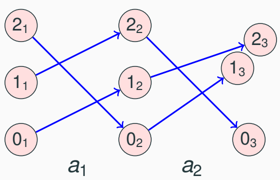
21

As binary relations



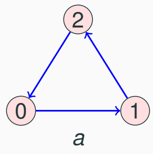
·	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



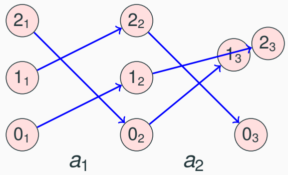
21

As binary relations



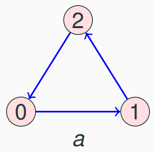
$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



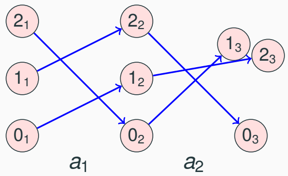
21

As binary relations



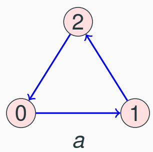
$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



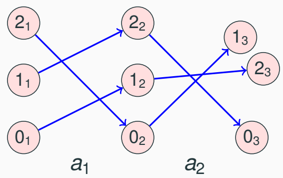
21

As binary relations



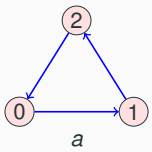
$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



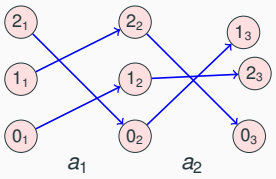
21

As binary relations

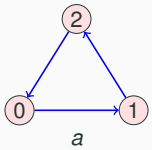


·	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes

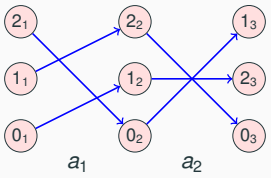


As binary relations

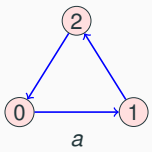


·	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes

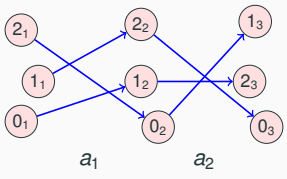


As binary relations

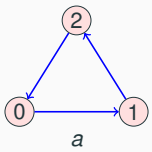


·	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



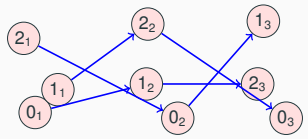
As binary relations



a

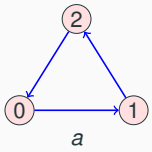
$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



a_1 a_2

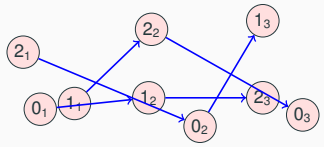
As binary relations



a

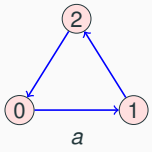
$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



a_1 a_2

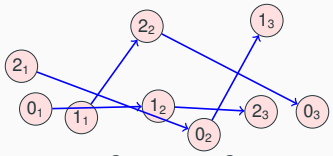
As binary relations



a

$\cdot$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

becomes



a_1 a_2

Critical observation for some elements e, a

If $e; a = e$ then $e \leq_{\mathcal{L}} a$.
If $e; 1 = a; 1$ and e, a are known to represent as injective partial functions, then $e \mathcal{L} a$

Critical observation for some elements e, a

If $e; a = e$ then $e \leq_{\mathcal{L}} a$.
If $e; 1 = a; 1$ and e, a are known to represent as injective partial functions, then $e \mathcal{L} a$

Only “injective partial functions” is not abstract. This focusses on methods to force certain elements to be representable as injective partial functions

Critical observation for some elements e, a

If $e; a = e$ then $e \leq_{\mathcal{L}} a$.
If $e; 1 = a; 1$ and e, a are known to represent as injective partial functions, then $e \mathcal{L} a$

Only “injective partial functions” is not abstract. This focusses on methods to force certain elements to be representable as injective partial functions

Tricks for defining a as an “injective partial functions”

- $a; a^{\smile} \leq 1'$ and $a^{\smile}; a \leq 1'$ (too obvious to be a trick!)

Critical observation for some elements e, a

If $e; a = e$ then $e \leq_{\mathcal{L}} a$.
If $e; 1 = a; 1$ and e, a are known to represent as injective partial functions, then $e \mathrel{\mathcal{L}} a$

Only "injective partial functions" is not abstract. This focusses on methods to force certain elements to be representable as injective partial functions

Tricks for defining a as an "injective partial functions"

- $a; a^\smile \leq 1'$ and $a^\smile; a \leq 1'$ (too obvious to be a trick!)
- $((a\,;0') \cdot a = 0 \;\;\&\;\; (0'\,;a) \cdot a = 0)$ (Hirsch and Jackson, 2011)

Critical observation for some elements e, a

If $e; a = e$ then $e \leq_{\mathcal{L}} a$.
If $e; 1 = a; 1$ and e, a are known to represent as injective partial functions, then $e \mathrel{\mathcal{L}} a$

Only "injective partial functions" is not abstract. This focusses on methods to force certain elements to be representable as injective partial functions

Tricks for defining a as an "injective partial functions"

- $a; a^\smile \leq 1'$ and $a^\smile; a \leq 1'$ (too obvious to be a trick!)
- $((a\,;0') \cdot a = 0 \;\;\&\;\; (0'\,;a) \cdot a = 0)$ (Hirsch and Jackson, 2011)
- $((-(a\,;-1'))\,;1\,;(-(-1'\,;a)) = 1) \;\;\&\;\; 1\,;1 = 1 \;\;\&\;\; -1\,;-1 = -1 \;\;\&\;\; \ldots$
  (new trick for Hirsch, J and Šemrl 2025)

Generalised kernels

Consider the relation algebraic terms $K_{L,n}(x) := (x\,;x^\smile)^n$ and $K_{R,n} := (x^\smile\,;x)^n$

Generalised kernels

Consider the relation algebraic terms $K_{L,n}(x) := (x \,;\, x^\smile)^n$ and $K_{R,n} := (x^\smile \,;\, x)^n$

Observation

The operations $K_{L,n}$ and $K_{R,n}$ do have an obvious definition in $S(\mathcal{A})$, even if $^\smile$ itself does not:

$$K_{L,n}(a_{ij}) = e_{ii} \text{ and } K_{R,n}(a_{ij}) = e_{jj}$$

Generalised kernels

Consider the relation algebraic terms $K_{L,n}(x) := (x \,;\, x^\smile)^n$ and $K_{R,n} := (x^\smile \,;\, x)^n$

Observation

The operations $K_{L,n}$ and $K_{R,n}$ do have an obvious definition in $S(\mathcal{A})$, even if $^\smile$ itself does not:

$$K_{L,n}(a_{ij}) = e_{ii} \text{ and } K_{R,n}(a_{ij}) = e_{jj}$$

Idea: if $(a_{ij})^\smile$ *were to equal* $(a^{-1})_{ji}$, *then* $a_{ij} \,;\, (a_{ij})^\smile = a_{ij} \,;\, (a^{-1})_{ji} = (aa^{-1})_{ii} = e_{ii}$

Generalised kernels

Consider the relation algebraic terms $K_{L,n}(x) := (x \,;\, x^\smile)^n$ and $K_{R,n} := (x^\smile \,;\, x)^n$

Observation

The operations $K_{L,n}$ and $K_{R,n}$ do have an obvious definition in $S(\mathcal{A})$, even if $^\smile$ itself does not:

$$K_{L,n}(a_{ij}) = e_{ii} \text{ and } K_{R,n}(a_{ij}) = e_{jj}$$

An $\mathfrak{H}$-embedding

If $S(\mathcal{A})$ is isomorphic to a system of binary relations respecting $K_{L,n}$ and $K_{R,n}$, then $\mathcal{A}$ embeds into a group (an undecidable problem)

Proof: we know that $e_{ii} \geq_{\mathfrak{R}} a_{ij}$ from before.

Generalised kernels

Consider the relation algebraic terms $K_{L,n}(x) := (x \,;\, x^\smile)^n$ and $K_{R,n} := (x^\smile \,;\, x)^n$

Observation

The operations $K_{L,n}$ and $K_{R,n}$ do have an obvious definition in $S(\mathcal{A})$, even if $^\smile$ itself does not:

$$K_{L,n}(a_{ij}) = e_{ii} \text{ and } K_{R,n}(a_{ij}) = e_{jj}$$

An $\mathcal{H}$-embedding

If $S(\mathcal{A})$ is isomorphic to a system of binary relations respecting $K_{L,n}$ and $K_{R,n}$, then $\mathcal{A}$ embeds into a group (an undecidable problem)

Proof: we know that $e_{ii} \geq_{\mathcal{R}} a_{ij}$ from before. But $e_{ii} = (a_{ij} \,;\, (a_{ij})^\smile)^n$ shows that $a_{ij} \geq_{\mathcal{R}} e_{ii}$ also.

Generalised kernels

Consider the relation algebraic terms $K_{L,n}(x) := (x \,;\, x^\smile)^n$ and $K_{R,n} := (x^\smile \,;\, x)^n$

Observation

The operations $K_{L,n}$ and $K_{R,n}$ do have an obvious definition in $S(\mathcal{A})$, even if $^\smile$ itself does not:

$$K_{L,n}(a_{ij}) = e_{ii} \text{ and } K_{R,n}(a_{ij}) = e_{jj}$$

An $\mathcal{H}$-embedding

If $S(\mathcal{A})$ is isomorphic to a system of binary relations respecting $K_{L,n}$ and $K_{R,n}$, then $\mathcal{A}$ embeds into a group (an undecidable problem)

Proof: we know that $e_{ii} \geq_{\mathcal{R}} a_{ij}$ from before. But $e_{ii} = (a_{ij} \,;\, (a_{ij})^\smile)^n$ shows that $a_{ij} \geq_{\mathcal{R}} e_{ii}$ also. Similarly for $\mathcal{L}$ wrt e_{jj} and therefore $a_{ij}\ \mathcal{H}\ b_{ij}$ (for all a, b)

If and only if

An $\mathcal{H}$-embedding (from previous slide)

If $S(\mathcal{A})$ is isomorphic to a system of binary relations respecting $K_{L,n}$ and $K_{R,n}$, then $\mathcal{A}$ embeds into a group

If and only if

An $\mathcal{H}$-embedding (from previous slide)

If $S(\mathcal{A})$ is isomorphic to a system of binary relations respecting $K_{L,n}$ and $K_{R,n}$, then $\mathcal{A}$ embeds into a group

Converse direction

$S(\mathcal{A})$ is isomorphic to a system of binary relations respecting $K_{L,n}$ and $K_{R,n}$, if $\mathcal{A}$ embeds into a group

Proof: this is just because if A completes to G, then $S(\mathcal{A})$ embeds in the Brandt semigroup $B_3(G)$ which is representable, even as injective partial functions

24

Weaker and weaker signatures

Obviously $\{K_{L,2^{n+1}}, K_{R,2^{n+1}}, ;\}$ are term functions in $\{K_{L,2^n}, K_{R,2^n}, ;\}$, so we have an infinite descending chain of weaker (?) and weaker signatures having undecidability of representability

25

Weaker and weaker signatures

Theorem

The only terms expressible in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ for all n are those that are expressible in $\{;\}$ (that is, semigroups)

25

Weaker and weaker signatures

Theorem

The only terms expressible in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ for all n are those that are expressible in $\{;\}$ (that is, semigroups)

Proof. Every term $t(x_1, \ldots, x_n)$ in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ is a term in $\{;, {}^\smile\}$ so can be expressed as a semigroup word in the alphabet $\{x_1, \ldots, x_n, x_1^\smile, \ldots, x_n^\smile\}$.

Weaker and weaker signatures

Theorem

The only terms expressible in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ for all n are those that are expressible in $\{;\}$ (that is, semigroups)

Proof. Every term $t(x_1, \ldots, x_n)$ in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ is a term in $\{;, {}^\smile\}$ so can be expressed as a semigroup word in the alphabet $\{x_1, \ldots, x_n, x_1^\smile, \ldots, x_n^\smile\}$.
By Schein's Theorem (free involuted semigroup is representable), such a representation is unique.

Weaker and weaker signatures

Theorem

The only terms expressible in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ for all n are those that are expressible in $\{;\}$ (that is, semigroups)

Proof. Every term $t(x_1, \ldots, x_n)$ in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ is a term in $\{;, {}^\smile\}$ so can be expressed as a semigroup word in the alphabet $\{x_1, \ldots, x_n, x_1^\smile, \ldots, x_n^\smile\}$.
By Schein's Theorem (free involuted semigroup is representable), such a representation is unique.
The length of any term involving $K_{L,m}$ or $K_{R,m}$ is at least $2m$ under this "norm".

Weaker and weaker signatures

Theorem

The only terms expressible in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ for all n are those that are expressible in $\{;\}$ (that is, semigroups)

Proof. Every term $t(x_1, \ldots, x_n)$ in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ is a term in $\{;, \breve{}\}$ so can be expressed as a semigroup word in the alphabet $\{x_1, \ldots, x_n, x_1\breve{}, \ldots, x_n\breve{}\}$.
By Schein's Theorem (free involuted semigroup is representable), such a representation is unique.
The length of any term involving $K_{L,m}$ or $K_{R,m}$ is at least $2m$ under this "norm".
So a term expressible in $\{K_{L,2^n}, K_{R,2^n}, ;\}$ for all n must involve ; only

25

Strategies, qualitative and quantitative model checking in Maude

Narciso Martí-Oliet, Rubén Rubio

Universidad Complutense de Madrid

LAC 2025 — October 2, 2025, Fukuoka, Japan

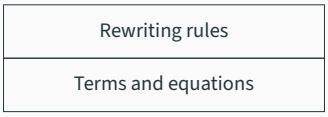
Maude

```
         \||||||||||||||||||/
       --- Welcome to Maude ---
         /||||||||||||||||||\
```

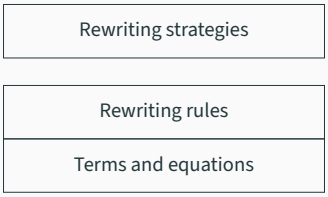
https://maude.cs.illinois.edu

- Maude is a high-level language and high-performance system.
- It supports both equational and rewriting logic computation.
- It is a flexible and general semantic framework for giving semantics to a wide range of languages and models of concurrency.
- It is also a good logical framework, i.e., a metalogic in which many other logics can be naturally represented and implemented.
- Moreover, it is reflective allowing many advanced metaprogramming and metalanguage applications.

Anatomy of a Maude specification



Anatomy of a Maude specification



Anatomy of a Maude specification

- Order-sorted signature $\Omega = (K, \Sigma, S)$.
- Equations and membership axioms

$$(\forall X) \quad \begin{matrix} t = t' \\ t : s \end{matrix} \quad \textbf{if} \quad \bigwedge_i u_i = v_i \wedge \bigwedge_j u_j : s_j$$

- Operator axioms, like commutativity, associativity, and identity.

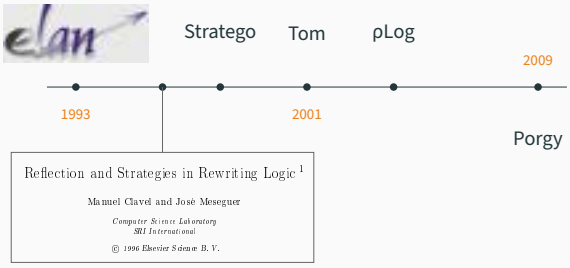
Anatomy of a Maude specification

- A rewrite theory $\mathscr{R} = (\Sigma, E \cup A, R)$ adds rewrite rules R on top of the equational theory.
- Rules do not need to be either confluent or terminating.

$$(\forall X) \quad t \Rightarrow t' \quad \textbf{if} \quad \bigwedge_i u_i = v_i \wedge \bigwedge_j u_j : s_j \wedge \bigwedge_k u_k \Rightarrow v_k$$

Earlier strategy languages and reflection 1993-

Earlier strategy languages and reflection 1993-



Reflection and Strategies in Rewriting Logic [1]

Manuel Clavel and José Meseguer

Computer Science Laboratory
SRI International

Design and first prototype 2004-2006

Towards a Strategy Language for Maude*

Narciso Martí-Oliet[a], José Meseguer[b], and Alberto Verdejo[a]

[a] *Facultad de Informática, UCM, Madrid, Spain*
[b] *Department of Computer Science, UIUC, Urbana-Champaign, USA*

2004 2006

Deduction, Strategies, and Rewriting

Steven Eker[a], Narciso Martí-Oliet[b], José Meseguer[c], and Alberto Verdejo[b]

[a] *Computer Science Laboratory, SRI International, Menlo Park, CA, USA*
[b] *Facultad de Informática, Universidad Complutense de Madrid, Spain*
[c] *Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA*

The strategy language in Core Maude 2020

MaudE3

maude.cs.illinois.edu

Journal of Logical and Algebraic Methods in Programming 110 (2020) 100497

Programming and symbolic computation in Maude

Francisco Durán [a], Steven Eker [b], Santiago Escobar [c], Narciso Martí-Oliet [d,*], José Meseguer [e], Rubén Rubio [d], Carolyn Talcott [b]

[a] Universidad de Málaga, Spain
[b] SRI International, CA, USA
[c] Universitat Politècnica de València, Spain
[d] Universidad Complutense de Madrid, Spain
[e] University of Illinois at Urbana-Champaign, IL, USA

Examples with the Maude strategy language

maude.ucm.es/strategies

Language semantics

- λ-calculus
- Prolog (negation and cut)
- Eden
- CCS

Algorithms & deduction

- Equational completion
- SAT solving
- Simplex algorithm
- Sudoku solver

Concurrency problems

- Lamport's bakery
- Dining philosophers

Computational models

- Membrane systems
- Population protocols

Plan of the talk

1. The language
2. Examples
3. Model checking
4. Probabilistic extension
5. Probabilistic and statistical model checking

The Maude strategy language

Introductory example — Crossing the river

Example — Crossing the river

```
fmod RIVER is
  sort River Side Group .
  subsort Side < Group .

  ops left right : -> Side [ctor] .
  ops shepherd wolf goat cabbage : -> Group [ctor] .

  ops __  : Group Group -> Group [ctor assoc comm prec 40] .
  op  _|_ : Group Group -> River [ctor comm] .

  op initial : -> River .
  eq initial = left shepherd wolf goat cabbage | right .
endfm
```

Example — Crossing the river

```
fmod RIVER is
  sort River Side Group .
  subsort Side < Group .

  ops left right : -> Side [ctor] .
  ops shepherd wolf goat cabbage : -> Group [ctor] .

  ops __  : Group Group -> Group [ctor assoc comm prec 40] .
  op  _|_ : Group Group -> River [ctor comm] .

  op initial : -> River .
  eq initial = left shepherd wolf goat cabbage | right .
endfm
```

Example — Crossing the river

```
fmod RIVER is
  sort River Side Group .
  subsort Side < Group .

  ops left right : -> Side [ctor] .
  ops shepherd wolf goat cabbage : -> Group [ctor] .

  ops __  : Group Group -> Group [ctor assoc comm prec 40] .
  op  _|_ : Group Group -> River [ctor comm] .

  op initial : -> River .
  eq initial = left shepherd wolf goat cabbage | right .
endfm
```

Example — Crossing the river

```
fmod RIVER is
  sort River Side Group .
  subsort Side < Group .

  ops left right : -> Side [ctor] .
  ops shepherd wolf goat cabbage : -> Group [ctor] .

  ops __  : Group Group -> Group [ctor assoc comm prec 40] .
  op  _|_ : Group Group -> River [ctor comm] .

  op initial : -> River .
  eq initial = left shepherd wolf goat cabbage | right .
endfm
```

Example — Crossing the river

```
mod RIVER-CROSSING is
  protecting RIVER .

  vars G G' : Group .

  rl [wolf-eats] :    goat wolf G | shepherd G' =>
                           wolf G | shepherd G' .
  rl [goat-eats] : cabbage goat G | shepherd G' =>
                           goat G | shepherd G' .

  rl [alone]   :         shepherd G | G' => G | shepherd G' .
  rl [wolf]    :    shepherd wolf G | G' => G | shepherd wolf G' .
  rl [goat]    :    shepherd goat G | G' => G | shepherd goat G' .
  rl [cabbage] : shepherd cabbage G | G'
                                  => G | shepherd cabbage G' .
endm
```


Example — Crossing the river

```
mod RIVER-CROSSING is
  protecting RIVER .

  vars G G' : Group .

  rl [wolf-eats] :    goat wolf G | shepherd G' =>
                           wolf G | shepherd G' .
  rl [goat-eats] : cabbage goat G | shepherd G' =>
                           goat G | shepherd G' .

  rl [alone]   :         shepherd G | G' => G | shepherd G' .
  rl [wolf]    :    shepherd wolf G | G' => G | shepherd wolf G' .
  rl [goat]    :    shepherd goat G | G' => G | shepherd goat G' .
  rl [cabbage] : shepherd cabbage G | G'
                                  => G | shepherd cabbage G' .
endm
```

Example — Crossing the river

```
mod RIVER-CROSSING is
  protecting RIVER .

  vars G G' : Group .

  rl [wolf-eats] :    goat wolf G | shepherd G' =>
                           wolf G | shepherd G' .
  rl [goat-eats] : cabbage goat G | shepherd G' =>
                           goat G | shepherd G' .

  rl [alone]   :         shepherd G | G' => G | shepherd G' .
  rl [wolf]    :    shepherd wolf G | G' => G | shepherd wolf G' .
  rl [goat]    :    shepherd goat G | G' => G | shepherd goat G' .
  rl [cabbage] : shepherd cabbage G | G'
                                  => G | shepherd cabbage G' .
endm
```

Rule application

label

Rule application

label

```
Maude> srewrite shepherd left | shepherd right using alone .

Solution 1
result River: shepherd shepherd left | right

Solution 2
result River: left | shepherd shepherd right

No more solutions.
```

Rule application

label

```
Maude> srew shepherd left | shepherd right using alone .
```

- shepherd shepherd left | right
- left | shepherd shepherd right

Rule application

all

```
Maude> srew shepherd wolf left | right using all .
```

- wolf left | shepherd right
- left | shepherd wolf right

Rule application

all

```
Maude> srew shepherd wolf left | right using all .
```

- wolf left | shepherd right — **rl** [alone]
- left | shepherd wolf right — **rl** [wolf]

Rule application

label $[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$

```
Maude> srew shepherd left | shepherd right
       using alone[G <- left] .
```

- left | shepherd shepherd right — **rl** shepherd left | ...

Rule application

label $[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$

```
Maude> srew shepherd left | shepherd right
       using alone[G ← right] .
```

- `shepherd shepherd left | right` `rl shepherd right | ...`

Rule application

label $[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$

crl $l(\overline{x}) \Rightarrow r(\overline{x}, \overline{y})$ **if** $C(\overline{x}, \overline{y})$ [nonexec] .

Rule application

label $[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$

crl $l(\overline{x}) \Rightarrow r(\overline{x}, \overline{y})$ **if** $C(\overline{x}, \overline{y})$ [nonexec] .

rl [replace] : G ⇒ G' [nonexec] .

Rule application

label $[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$

crl $l(\overline{x})$ ⇒ $r(\overline{x}, \overline{y})$ **if** $C(\overline{x}, \overline{y})$ [nonexec] .

rl [replace] : G ⇒ G' [nonexec] .

```
Maude> srew goat wolf using replace[G' ← cabbage] .
```

- cabbage
- goat cabbage
- cabbage wolf

Rule application

top(β)

```
Maude> srew goat wolf using top(replace[G' ← cabbage]) .
```

- cabbage

Rule application

label $[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]\{\alpha_1, \ldots, \alpha_m\}$

crl l ⇒ r **if** $C \wedge l_1$ ⇒ $r_1 \wedge C'$.

Tests

match P s.t. C

Tests

match P s.t. C

```
Maude> srew left | right using match G | G' s.t. G =/= G' .

  • left | right
```

Tests

match P s.t. C

```
Maude> srew goat | goat using match G | G' s.t. G =/= G' .

No solution.
```

Tests

xmatch P **s.t.** C

```
Maude> srew wolf goat cabbage using xmatch wolf cabbage .
```

- wolf goat cabbage

Tests

amatch P **s.t.** C

```
Maude> srew left wolf | right using amatch wolf .
```

- left wolf | right

Concatenation

α ; β

Concatenation

α ; β

```
Maude> srew shepherd left | wolf right using alone .
```

- left | shepherd wolf right

Concatenation

α ; β

```
Maude> srew shepherd left | wolf right using alone ; wolf .
```

- shepherd wolf left | right

One solution

one(α)

```
Maude> srew shepherd left | shepherd right using one(alone)
.
```

- left | shepherd shepherd right

Disjunction

$\alpha \mid \beta$

```
Maude> srew shepherd wolf goat left | right
       using wolf | goat .
```

- goat left | shepherd wolf right
- wolf left | shepherd goat right

Regular expressions

$\alpha *$ **idle** **fail**

```
Maude> srew wolf goat goat | shepherd using wolf-eats * .
```

- wolf goat goat | shepherd
- wolf goat | shepherd
- wolf | shepherd

Regular expressions

$\alpha *$ **idle** **fail**

```
Maude> srew wolf goat goat | shepherd using wolf-eats + .
```

- ~~wolf goat goat | shepherd~~
- wolf goat | shepherd
- wolf | shepherd

Regular expressions

α * idle fail

```
Maude> srew wolf goat goat | shepherd using wolf-eats ! .
```

- ~~wolf goat goat | shepherd~~
- ~~wolf goat | shepherd~~
- wolf | shepherd

Regular expressions

α * idle fail

```
Maude> srew shepherd using idle .
```

- shepherd

Regular expressions

α * idle fail

```
Maude> srew shepherd using fail .

No solution.
```

Conditionals

α ? β : γ

```
Maude> srew wolf goat left | shepherd right
       using wolf-eats ? idle : alone .
```

- wolf left | shepherd right

Conditionals

α ? β : γ

```
Maude> srew wolf goat shepherd left | right
       using wolf-eats ? idle : alone .
```

- wolf goat left | shepherd right

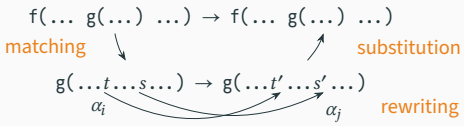
Conditionals

α ? β : γ

$$\begin{aligned} \alpha \texttt{ or-else } \beta &\equiv \alpha \texttt{ ? idle : } \beta \\ \texttt{try}(\alpha) &\equiv \alpha \texttt{ ? idle : idle} \\ \texttt{not}(\alpha) &\equiv \alpha \texttt{ ? fail : idle} \\ \texttt{test}(\alpha) &\equiv \texttt{not}(\texttt{not}(\alpha)) \end{aligned}$$

Rewriting of subterms

matchrew P s.t. C by x_1 using α_1 , … , α_n using x_n



Rewriting of subterms

matchrew P s.t. C by x_1 using α_1 , … , α_n using x_n

```
Maude> srew wolf left | goat using matchrew G left | G' by G
       using replace[G' ← cabbage] .
```

- `cabbage left | goat`

Rewriting of subterms

matchrew P s.t. C by x_1 using α_1 , … , α_n using x_n

```
Maude> srew wolf left | goat using matchrew G left | G' by G
       using replace[G' ← G'] .
```

- `goat left | goat`

Overview of the strategy language

label$[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]\{\alpha_1, \ldots, \alpha_m\}$

match P **s.t.** C **idle** **fail**

$\alpha\ ;\ \beta$ $\alpha\ |\ \beta$ $\alpha*$ $\alpha\ ?\ \beta : \gamma$

matchrew P **s.t.** C **by** x_1 **using** α_1, **...**, x_n **using** α_n

Strategy calls

slabel $(t_1, \ldots, t_n)$

```
Maude> srew wolf goat goat goat goat | shepherd
       using repeat(2, 3) .
```

- wolf goat goat
- wolf goat

Strategy modules

Rewrite theory — System module

Equational theory — Functional module

Strategy modules

Rewrite strategy — Strategy module

Rewrite theory — System module

Equational theory — Functional module

Strategy modules

Declarations

strat *slabel* : $s_1 \cdots s_n$ @ s .

Definitions

sd *slabel* $(t_1, \ldots, t_n)$:= α .

csd *slabel* $(t_1, \ldots, t_n)$:= α **if** C .

A recursive strategy

```
strat repeat : Nat Nat @ Word .

vars N M : Nat .

sd repeat(0, N) := idle .
sd repeat(0, s N) := wolf-eats ; repeat(0, N) .
sd repeat(s M, s N) := wolf-eats ; repeat(M, N) .
```

A recursive strategy

```
smod RIVER-REPEAT is
  protecting RIVER-CROSSING .

  strat repeat : Nat Nat @ Word .

  vars N M : Nat .

  sd repeat(s M, s N) := wolf-eats ; repeat(M, N) .
  sd repeat(0, s N) := idle | wolf-eats ; repeat(0, N) .
endsm
```

Reflection of the strategy language

```
op _[_]{_} : Qid Substitution StrategyList -> RuleApplication .
op match_s.t._ : Term EqCondition -> Strategy .

op sd_:=_[_]. : CallStrategy Strategy AttrSet -> StratDefinition .
op smod_is_sorts_.______endsm : ··· -> StratModule .

op metaSrewrite : Module Term Strategy ··· ~> ResultPair? .
```

R. Rubio, N. Martí-Oliet, I. Pita, and A. Verdejo. **Metalevel transformation of strategies.** *J. Log. Algebr. Methods Program.*, 124, 2022.

Search controlled by a strategy

Crossing the river

```
Maude> search initial =>* left | right shepherd wolf goat cabbage .

Solution 1 (state 32)
empty substitution

No more solutions.
```

Crossing the river

```
Maude> search initial =>* left | right shepherd wolf goat cabbage .
Maude> show path 32 .
state 0, River: left shepherd wolf goat cabbage | right
==[ wolf ]==>
state 2, River: left goat cabbage | right shepherd wolf
==[ alone ]==>
state 8, River: left shepherd goat cabbage | right wolf
==[ goat ]==>
state 16, River: left cabbage | right shepherd wolf goat
==[ alone ]==>
state 24, River: left shepherd cabbage | right wolf goat
==[ cabbage ]==>
state 32, River: left | right shepherd wolf goat cabbage
```

Crossing the river

```
Maude> search initial =>* left | right shepherd wolf goat cabbage .
Maude> show path 32 .
state 0, River: left shepherd wolf goat cabbage | right
==[ wolf ]==>
state 2, River: left goat cabbage | right shepherd wolf
==[ alone ]==>
state 8, River: left shepherd goat cabbage | right wolf
==[ goat ]==>
state 16, River: left cabbage | right shepherd wolf goat
==[ alone ]==>
state 24, River: left shepherd cabbage | right wolf goat
==[ cabbage ]==>
state 32, River: left | right shepherd wolf goat cabbage
```

Crossing the river

```
smod RIVER-CROSSING-STRAT is
  protecting RIVER-CROSSING .

  strats eagerEating oneCrossing eating @ River .

  var G : Group .

  sd eagerEating := match left | G cabbage goat ? idle
                 : ((eating or-else oneCrossing) ; eagerEating) .

  sd oneCrossing := alone | wolf | goat | cabbage .
  sd eating      := wolf-eats | goat-eats .
endsm
```


Crossing the river

```
smod RIVER-CROSSING-STRAT is
  protecting RIVER-CROSSING .

  strats eagerEating oneCrossing eating @ River .

  var G : Group .

  sd eagerEating := match left | G cabbage goat ? idle
                 : ((eating or-else oneCrossing) ; eagerEating) .

  sd oneCrossing := alone | wolf | goat | cabbage .
  sd eating      := wolf-eats | goat-eats .
endsm
```

Crossing the river

```
smod RIVER-CROSSING-STRAT is
  protecting RIVER-CROSSING .

  strats eagerEating oneCrossing eating @ River .

  var G : Group .

  sd eagerEating := match left | G cabbage goat ? idle
                 : ((eating or-else oneCrossing) ; eagerEating) .

  sd oneCrossing := alone | wolf | goat | cabbage .
  sd eating      := wolf-eats | goat-eats .
endsm
```

Crossing the river

```
smod RIVER-CROSSING-STRAT is
  protecting RIVER-CROSSING .

  strats eagerEating oneCrossing eating @ River .

  var G : Group .

  sd eagerEating := match left | G cabbage goat ? idle
                  : ((eating or-else oneCrossing) ; eagerEating) .

  sd oneCrossing := alone | wolf | goat | cabbage .
  sd eating      := wolf-eats | goat-eats .
endsm
```

Crossing the river

```
smod RIVER-CROSSING-STRAT is
  protecting RIVER-CROSSING .

  strats eagerEating oneCrossing eating @ River .

  var G : Group .

  sd eagerEating := match left | G cabbage goat ? idle
                  : ((eating or-else oneCrossing) ; eagerEating) .

  sd oneCrossing := alone | wolf | goat | cabbage .
  sd eating      := wolf-eats | goat-eats .
endsm
```

Crossing the river with `eagerEating`

```
Maude> search initial =>* left | right shepherd wolf goat cabbage
              using eagerEating .

Solution 1 (state 30)
empty substitution

No more solutions.
```

Crossing the river with `eagerEating`

```
Maude> show path 30 .
state 0, River: left shepherd wolf goat cabbage | right
==[ goat ]==>
state 23, River: left wolf cabbage | right shepherd goat
==[ alone ]==>
state 24, River: left shepherd wolf cabbage | right goat
==[ wolf ]==>
state 25, River: left cabbage | right shepherd wolf goat
==[ goat ]==>
state 27, River: left shepherd goat cabbage | right wolf
==[ cabbage ]==>
state 28, River: left goat | right shepherd wolf cabbage
==[ alone ]==>
state 29, River: left shepherd goat | right wolf cabbage
==[ goat ]==>
state 30, River: left | right shepherd wolf goat cabbage
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; not(eating) ; safe) .
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; (eating ? fail : idle) ; safe) .
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; not(eating) ; safe) .
```

```
Maude> search initial =>* R s.t. risky(R) using safe .

No solution.
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; not(eating) ; safe) .
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; not(eating) ; safe) .
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; not(eating) ; safe) .
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; not(eating) ; safe) .
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; not(eating) ; safe) .
```

Strategic failure

```
sd safe := (match left | G)
             ? idle
             : (oneCrossing ; not(eating) ; safe) .
```

Model checking with strategies

Model checking

Model (Kripke structure)

$\mathscr{K} = (S, \rightarrow, I, AP, \ell)$
$\ell : S \rightarrow \mathscr{P}(AP)$

Temporal property

φ_{AP}

Model checking $\mathscr{K} \models \varphi$

Model checking

Model (Kripke structure)

$\mathscr{K} = (S, \to, I, AP, \ell)$
$\ell : S \to \mathscr{P}(AP)$

Temporal property

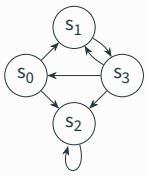
φ_{AP}

Model checking $\quad \mathscr{K} \models \varphi$

$(T_{\Sigma/E}, \to_R^1, I, AP, \ell)$ for a rewriting theory $\mathscr{R} = (\Sigma, E, R)$

Abstract strategies

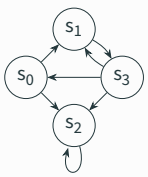
$\mathscr{A} = (S, \to)$



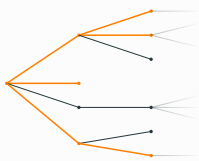
Γ_A

Abstract strategies

$\mathscr{A} = (S, \to)$



$E \subseteq \Gamma_A$



Model checking with strategies — linear-time case

Standard $\mathcal{K} \models \varphi$ iff $\ell(\pi) \models \varphi$ for every execution π in $\mathcal{K}$

Model checking with strategies — linear-time case

Standard $\mathcal{K} \models \varphi$ iff $\ell(\pi) \models \varphi$ for every execution π in $\mathcal{K}$

Strategy $(\mathcal{K}, E) \models \varphi$ iff $\ell(\pi) \models \varphi$ for every execution $\pi \in E$

Model checking with strategies — linear-time case

Standard $\mathcal{K} \models \varphi$ iff $\ell(\pi) \models \varphi$ for every execution π in $\mathcal{K}$

Strategy $(\mathcal{K}, E) \models \varphi$ iff $\ell(\pi) \models \varphi$ for every execution $\pi \in E$

In practice: to reuse standard algorithms, find an ARS whose traces *coincide* with E.

Model checking with Maude strategies — linear-time case

Small-step non-deterministic operational semantics

- Execution states $\mathcal{XS}$: term + strategy progress.
- Control $\rightarrow_c$ and system $\rightarrow_s$ transitions (rule rewrites).

$$\twoheadrightarrow = \rightarrow_c^* \circ \rightarrow_s$$

$$E(\alpha, I) = \{\, \mathsf{term}(x) \,:\, t \mathbin{@} \alpha \twoheadrightarrow x_2 \twoheadrightarrow \cdots \twoheadrightarrow x_k \twoheadrightarrow \cdots, t \in I \,\}$$

Model checking with Maude strategies — linear-time case

Small-step non-deterministic operational semantics

- Execution states $\mathcal{XS}$: term + strategy progress.
- Control $\rightarrow_c$ and system $\rightarrow_s$ transitions (rule rewrites).

$$\twoheadrightarrow = \rightarrow_c^* \circ \rightarrow_s$$

$$E(\alpha, I) = \{\, \mathsf{term}(x) \,:\, t \mathbin{@} \alpha \twoheadrightarrow x_2 \twoheadrightarrow \cdots \twoheadrightarrow x_k \twoheadrightarrow \cdots, t \in I \,\}$$

$$(\mathscr{K}, E(\alpha, I)) \vDash \varphi \iff (\mathcal{XS}, \twoheadrightarrow, \{t \mathbin{@} \alpha\}_{t \in I}, \ell \circ \mathsf{term}) \vDash \varphi$$

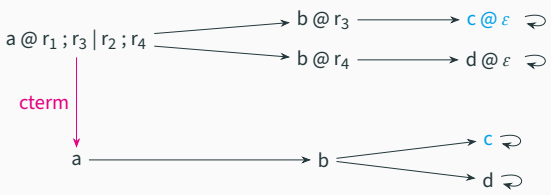
Model checking with strategies — branching-time case

And for branching time?

Initial idea: use the *equivalent* ARS of the linear-time case.

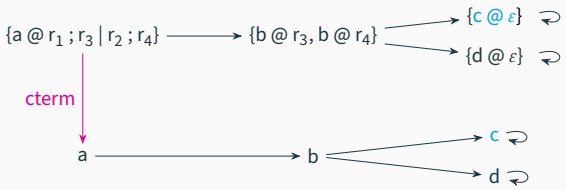
Model checking with strategies (branching-time case)

The branching structure of the executions is not preserved.



Model checking with strategies — branching-time case

However, this can be solved by merging states.



Model checking CTL* properties

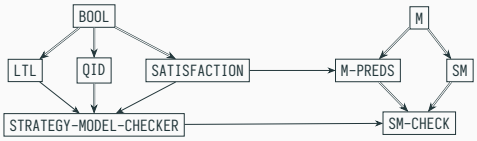
$$
\begin{aligned}
E \vDash p &\iff \forall\, \pi \in E \quad p \in \ell(\pi_0) \\
E \vDash \mathbf{A}\,\phi &\iff \forall\, \pi \in E \quad E_{\pi_0}, \pi \vDash \phi \\
E \vDash \mathbf{E}\,\phi &\iff \exists\, \pi \in E \quad E_{\pi_0}, \pi \vDash \phi
\end{aligned}
$$

$$
\begin{aligned}
E, \pi \vDash \Phi &\iff E \vDash \Phi \\
E, \pi \vDash \bigcirc \phi &\iff E_{\pi_0\pi_1}, \pi[1..] \vDash \phi \\
E, \pi \vDash \Diamond \phi &\iff \exists\, n \geq 0 \quad E_{\pi[..n]}, \pi[n..] \vDash \phi \\
E, \pi \vDash \Box \phi &\iff \forall\, n \geq 0 \quad E_{\pi[..n]}, \pi[n..] \vDash \phi \\
E, \pi \vDash \phi_1 \,\mathcal{U}\, \phi_2 &\iff \exists\, n \geq 0 \quad \begin{array}{l} E_{\pi[..n]}, \pi[n..] \vDash \phi_2 \wedge \\ \forall\, 0 \leq k < n \quad E_{\pi[..k]}, \pi[k..] \vDash \phi_1 \end{array}
\end{aligned}
$$

$$
E_{\pi_1 \cdots \pi_n} = \{\pi_n \pi_+ \,:\, \pi_1 \cdots \pi_n \pi_+ \in E\}
$$

Model checking in Maude

The Maude model checker for strategy-controlled systems



R. Rubio, N. Martí-Oliet, I. Pita, and A. Verdejo. **Model checking strategy-controlled systems in rewriting logic.** *Automat. Softw. Eng.*, 29(1), 2022.

The Maude model checker for strategy-controlled systems

The Maude model checker for strategy-controlled systems


```
fmod SATISFACTION is
  sort State Prop .
  op _|=_ : State Prop -> Bool .
endfm
```

The Maude model checker for strategy-controlled systems


```
fmod SATISFACTION is
  sort State Prop .
  op _|=_ : State Prop -> Bool .
endfm
```

The Maude model checker for strategy-controlled systems


```
op modelCheck : State Formula Qid QidList
                -> ModelCheckResult .
```

Model checking — Crossing the river

```
Maude> reduce modelCheck(initial,
                         [] (risky -> <> death),
                         'eagerEating) .
rewrites: 129
result Bool: true
```

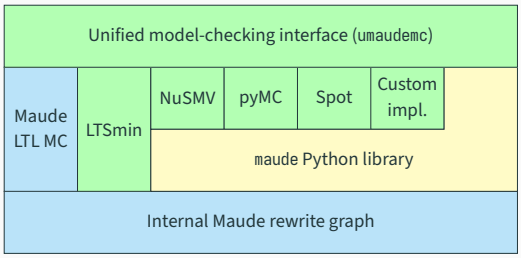
Model checking — Crossing the river

```
Maude> reduce modelCheck(initial,
                         [] (risky -> <> death),
                         'eagerEating) .
rewrites: 44
result ModelCheckerResult: counterexample(..., ...)
```

Model checking — Crossing the river

```
Maude> reduce modelCheck(initial,
                         <> goal,
                         'eagerEating) .
rewrites: 24
result ModelCheckerResult: counterexample(..., ...)
```

External model checkers for branching-time logics



`github.com/fadoss/umaudemc` · `pip install umaudemc`

Supported logics

	LTL	CTL	CTL*	μ-calculus
Extended Maude	✓			
LTSmin	✓	✓	✓	✓
pyModelChecking	✓	✓	✓	
NuSMV	✓	✓		
Spot	✓			
Builtin		✓		✓

R. Rubio, N. Martí-Oliet, I. Pita, and A. Verdejo. **Strategies, model checking and branching-time properties in Maude.** *J. Log. Algebr. Methods Program.*, 123, 2021.

Model checking — Crossing the river

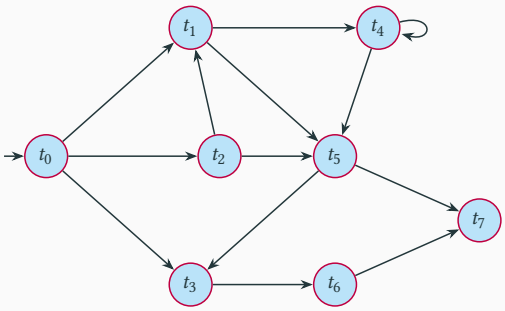
```
$ umaudemc check river.maude initial 'A [] E <> goal' safe
The property is satisfied in the initial state
(11 system states, 44 rewrites, holds in 11/11 states)
```

Model checking — Crossing the river

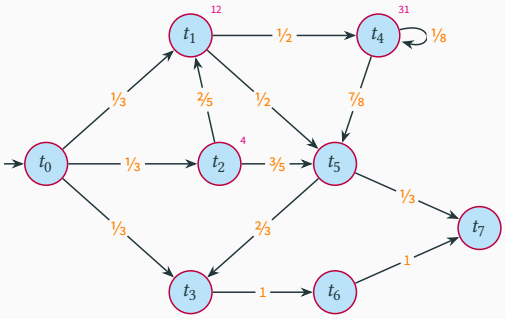
```
$ umaudemc check river.maude initial 'A [] E <> goal' eagerEating
The property is not satisfied in the initial state
(35 system states, 106 rewrites)
| left shepherd wolf goat cabbage | right
v  rl G' | shepherd G => G | shepherd G' [label alone] .
0 left wolf goat cabbage | right shepherd
```

Quantitative specification and verification

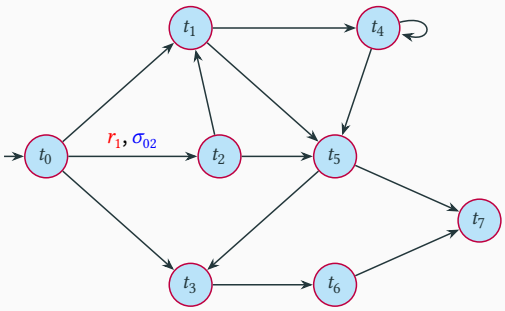
Probability assignment



Probability assignment



Probability assignment



Probability assignment methods

- `uniform`
- `uaction(`r_1`=`w_1`, ..., `r_m`=`w_m`)`
- `term(`e`)`
- `metadata`
- `pmaude` (scheck only)
- `strategy`

Probabilistic extension of the strategy language

$\alpha_1 \mid \cdots \mid \alpha_n$

`matchrew` P `s.t.` C `by` $\cdots$

Probabilistic extension of the strategy language

`choice`$(w_1 : \alpha_1, \ \ldots \ , w_n : \alpha_n)$

`matchrew` P `s.t.` C `with weight` w `by` $\cdots$

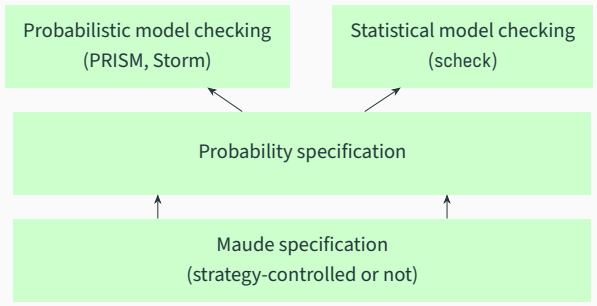
Probabilistic extension of the strategy language

`choice`$(w_1 : \alpha_1, \ \ldots \ , w_n : \alpha_n)$

`matchrew` P `s.t.` C `with weight` w `by` $\cdots$

`sample X :=` $\pi(t_1, \ \ldots, \ t_n)$ `in` α

Quantitative specification and verification



R. Rubio, N. Martí-Oliet, I. Pita, and A. Verdejo. **QMaude: quantitative specification and verification in rewriting logic.** In *FM 2023*, volume 14000 of *LNCS*. Springer, 2023.

Quantitative verification

- Probabilistic model checking
  - Probability of LTL or PCTL formulas
  - Steady-state analysis
  - Transient-state analysis
  - Expected values of rewards
  - With any assignment method except continuous distributions
  - For models without unquantified nondeterminism (DTMC) or even some well-ordered nondeterminism (MDP)
- Statistical model checking
  - Monte Carlo estimation of QuaTEx formulas
  - Parameterized queries
  - Multithreaded or distributed

Quantitative model checking — Example

```
$ umaudemc pcheck river initial '<> goal' --assign uniform
                  eagerEating

Result: 0.006211 (relative error 6.366e-06)
```

Quantitative model checking — Example

```
$ umaudemc pcheck river initial '<> goal' --assign uniform
                safe
Result: 1.0
```

Quantitative model checking — Example

```
$ umaudemc pcheck river initial '<> goal' --assign uniform
                safe --steps
Result: 54.9975 (relative error 9.778e-06)

$ umaudemc scheck river initial eaten.quatex --assign step
        'choice(1 : alone, 2 : wolf, 1 : goat, 1 : cabbage)'
Number of simulations = 7470
  μ = 2.6088353413   σ = 4.40767926235   r = 0.0999696468
```

Statistical model checking — QuaTEx formulas

```
Eaten(n) = if s.rval("S:State |= bad") then
               n
           else
               # Eaten(n + 1)     // next operator
           fi;

eval E[Eaten(0)];
```

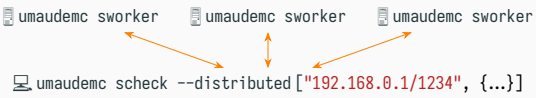
Statistical model checking — Parallelism

Monte Carlo simulations can be easily parallelized

Process-level parallelism

- scheck supports it with `--jobs n` or `-j n`

Machine-level parallelism



References

maude.ucm.es/strategies

- M. Clavel, F. Durán, S. Eker, S. Escobar, P. Lincoln, N. Martí-Oliet, J. Meseguer, R. Rubio, and C. Talcott. ***Maude Manual v3.5.1.*** July 2025. URL: maude.cs.illinois.edu.
- F. Durán, S. Eker, S. Escobar, N. Martí-Oliet, J. Meseguer, R. Rubio, and C. Talcott. **Programming and symbolic computation in Maude.** *J. Log. Algebraic Methods Program.*, 110, 2020.
- R. Rubio. ***Model checking of strategy-controlled systems in rewriting logic.*** PhD thesis, Universidad Complutense de Madrid, 2022.
- R. Rubio, N. Martí-Oliet, I. Pita, and A. Verdejo. **QMaude: quantitative specification and verification in rewriting logic.** In *FM 2023*, volume 14000 of *LNCS*. Springer, 2023.

Thank you

Strategies, qualitative and quantitative model checking in Maude

Narciso Martí-Oliet, Rubén Rubio
Universidad Complutense de Madrid

LAC 2025

MongoDB Specification in Maude

LAC 2025

BEATRIZ ALCAIDE

MongoDB

- DBMS
- NoSQL
- Distributed
  - Sharding
  - Replication
- Document-based
  - JSON-like documents

Why specify MongoDB?

- It dominates the No-SQL space:
  - EA
  - Toyota
  - Google
  - Uber
  - Adobe

Documents

Example:

```
{
  name:
  {
    first: "Jane",
    last: "Doe"
  }
  age: 28,
  groups: ["A", "B", "C]
}
```

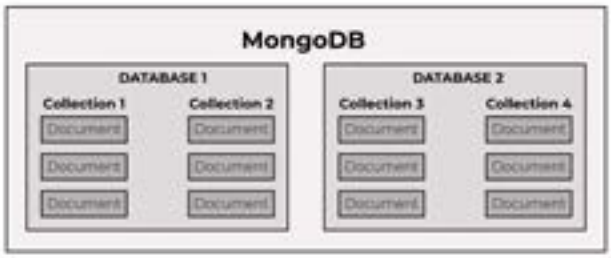
4

Quering Documents

MONGODB	SQL equivalence
db.collection('**inventory**').find({ **status: 'D'** });	SELECT * FROM inventory WHERE **status = "D"**
db.collection('**inventory**').find({ **status: 'A',** **qty: { $lt: 30 }** });	SELECT * FROM **inventory** WHERE **status = "A"** AND **qty < 30**

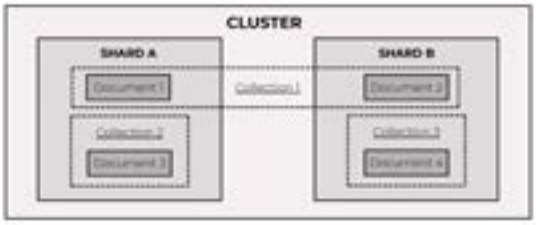
5

MongoDB's Data Model



6

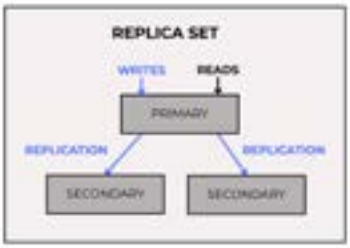
Sharding



7

Replication

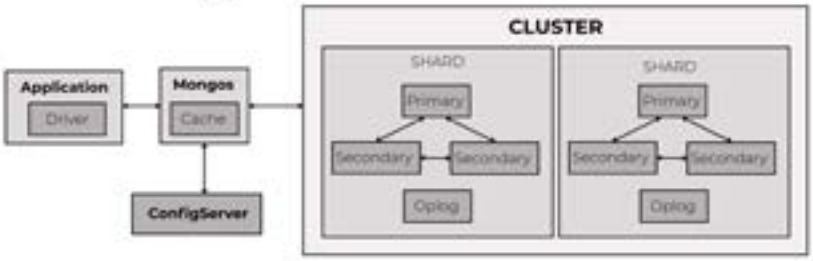
The **primary node** receives all the write operations



Secondary nodes read the write operations from an operation log (**Oplog**)

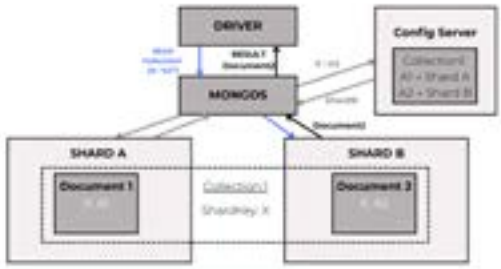
8

MongoDB's Architecture



9

Routing



MongoDB uses **Shard Keys** to route sharded queries, which can be:

- Hashed-Based
- Range-Based

10

Problem

An overload on the config servers can create a **bottleneck**.

Solution

Mongos uses a **cache** memory to locally store cluster metadata.

11

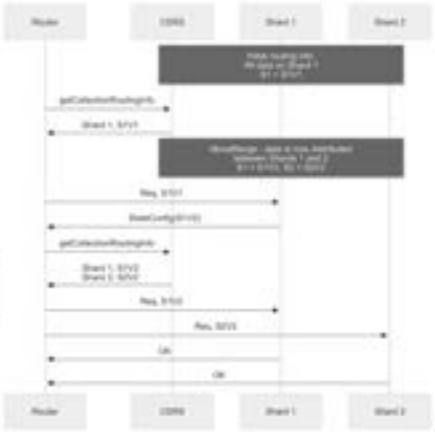
How do we know that the metadata in the **cache** memory is **up-to-date**?

→ **Versioning Protocol**

12

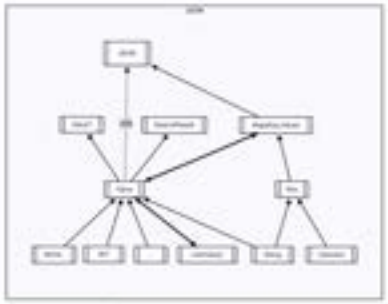
Versioning Protocol

The **routing info** tells the router the set of shards that should be contacted.



13

JSONs



14

Collection

```
sort UnshardedCollection ShardedCollection Collection .
subsorts UnshardedCollection ShardedCollection < Collection .

op UC : Map{Nat, JSON} Nat Nat -> UnshardedCollection [ctor] .
      *** nat1 = objectID counter, nat2 = version

op SC : Map{Nat, JSON} Nat -> ShardedCollection [ctor] .
```

Database

```
fmod DATABASE is
  pr MAP{String, Collection} * (sort Map{String, Collection} to Database) .
endfm

fmod DATABASEMAP is
  pr MAP{String, Database} * (sort Map{String, Database} to DatabaseMap) .
endfm
```

15

Node

ops primary secondary : -> NodeType [ctor] .

op node : NodeType DatabaseMap Nat -> Node [ctor] .
***nat = counter

ReplicaSet

op RS : Node NodeList UnshardedCollection -> ReplicaSet [ctor].

fmod SHARDMAP is
pr MAP(Nat, ReplicaSet) * (sort Map(Nat, ReplicaSet) to ShardMap) .
endfm

Cluster

op CL : Nat ShardMap -> Cluster [ctor] . ***Nat = idCounter

ShardKey

op hashBased rangeBased : -> TypeShardKey [ctor] .
op SK : TypeShardKey ShardValues -> ShardKey [ctor] .

16

Update Replica Set

Asynchronous message-passing communication model

17

Cache

Namespace:

Database name and Collection name

fmod METADATA-CACHE is

pr MAP(NameSpace, Routing-Info) * (sort Map(NameSpace, Routing-Info) to Cache) .

endfm

Mongos

op MGS : Cache ReplicaSet -> Mongos [ctor] . ***replicaset = configServer

18

Queries

The find function receives a **JSON-like** object that **defines** the **conditions** documents must meet to be included in the query result.

Future work

- Conformance testing.
- Consensus algorithm.
- Expand query operators.
- Indexes.
- Model check.

Thank you!

Forcing, Transition Algebras, and Calculi

Go Hashimoto and Daniel Găină (IMI, Japan)
Ionuț Țuțu (IMAR, Romania)

2nd Workshop on Logic, Algebra and Category Theory
Fukuoka, 2025

In this talk

1. Short presentation of transition algebra
2. Applicability to process calculi
3. Proof system, soundness, completeness via forcing
4. Tool support and introduction to SpeX

Transition algebra (TA)

* at a glance, yet another logic used to reason about labelled transition systems
* deserves further examination thanks to a blend of special features…

Transition algebra (TA)

* at a glance, yet another logic used to reason about labelled transition systems
* deserves further examination thanks to a blend of special features:
  - provides support both for the static, structural aspects of systems, via equations, and for the dynamic aspects of systems, via transitions
  - uniform treatment of states and transition labels (in particular, quantification over labels)
  - unrestricted use of equations and transitions
  - increased expressivity by employing actions similar to those found in dynamic logics
  - operational semantics (more to follow)

TA signatures

* ordinary algebraic signatures
* pairs (S,F), where:
  - S is a set of so-called sorts
  - F is an $S^* \times S$-indexed family of sets $F_{w \to s}$ of operation symbols of arity w and sort s
* as usual, we also write

$$\sigma : w \to s \in F$$

  in place of $\sigma \in F_{w \to s}$

Models

* (S,F)-algebras A interpreting:
  - every sort $s \in S$ as a set A_s
  - every operation symbol $\sigma : w \to s \in F$ as a function $\sigma^A : A_w \to A_s$
  - for any sort $t \in S$, every element $e \in A_t$ as a binary S-sorted relation $(e_s \subseteq A_s \times A_s)_{s \in S}$

Sentences

* built using standard Boolean connectives and quantifiers from two kinds of atoms:
  - equations $t = t'$
  - transitions $t\ a\ t'$
  - where t and t' are terms having the same sort and a is an action
* actions are built from terms using
  - sequential composition $a \fatsemi b$
  - choice $a \cup b$
  - iteration a^*

Semantics

* $A \models t = t'$ when $t^A = t'^A$
* $A \models t\ a\ t'$ when $(t^A, t'^A) \in a^A$

and so on, where

* $(a \fatsemi b)^A = a^A \fatsemi b^A$
* $(a \cup b)^A = a^A \cup b^A$
* $(a^*)^A = (a^A)^* = \bigcup \{(a^A)^n \mid n \in \mathbb{N}\}$

Quiz time

Which of the following models satisfies $\forall y \cdot \exists! z \cdot y \to z$?

Quiz time

Which of the following models satisfies $\forall y \cdot \exists! z \cdot y \to z$?

a) 

b) 

c) 

d) 

e) 

Quiz time

How do the models of $\forall y \cdot \exists! z \cdot y \to z \wedge \exists! x \cdot x \to y$ look like?

Quiz time

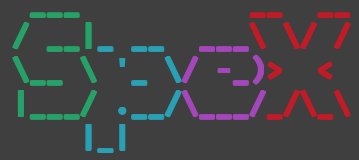
How do the models of $\forall y \cdot \exists! z \cdot y \to z \wedge \exists! x \cdot x \to y$ look like?

Quiz time

What if we add one of the following constraints?

* $\forall x, x' \cdot x \to^* x'$
* $\forall x, x' \cdot x \to^* x' \vee x' \to^* x$

Application to process calculi

Demo in 

Syntactic entailment

* get to $\Gamma \models \varphi$ via $\Gamma \vdash \varphi$
* where $\vdash$ is defined by proof rules of the following form:

$$\frac{\Gamma \vdash t = t'}{\Gamma \vdash t' = t} \qquad \frac{\Gamma \vdash t\ a\ t',\ \Gamma \vdash t'\ b\ t''}{\Gamma \vdash t\ (a \fatsemi b)\ t''} \qquad \frac{\Gamma \vdash t\ a\ t'}{\Gamma \vdash t\ (a \cup b)\ t'}$$

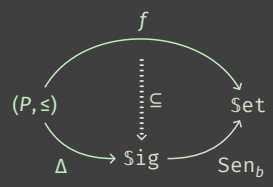
$$\frac{\Gamma \vdash t\ (a \fatsemi b)\ t'',\ \Gamma \cup \{t\ a\ x, x\ b\ t'\} \vdash \varphi}{\Gamma \vdash \varphi}$$

$$\frac{\Gamma \vdash \neg\neg\varphi}{\Gamma \vdash \varphi} \qquad \frac{\Gamma \cup \{\varphi\} \vdash \bot}{\Gamma \vdash \neg\varphi} \qquad \dots$$

Nota bene

* $\vdash$ is ω_1-compact whereas $\models$ is not so for uncountable signatures
* therefore, we cannot hope for a general completeness result for TA
* for at most countable signatures, neither $\vdash$ nor $\models$ is compact
* so we cannot tackle completeness using the classical Henkin method
* which leads us to forcing…

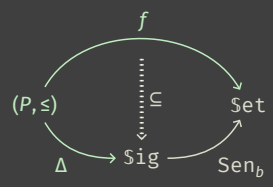
Forcing properties



where

* $(P,\leq)$ is a poset of so-called conditions
* Δ maps $p \leq q$ to $\Delta_p \subseteq \Delta_q$, and similarly for f
* $f(p) \subseteq \mathrm{Sen}_b(\Delta_p)$
* $f(p) \models \varphi$ implies $\varphi \in f(q)$ for some $q \geq p$

Syntactic forcing



where

* $p = (\Delta_p, \Gamma_p)$ where $\Delta_p = \Sigma \cup C_p$ and $\Gamma_p \subseteq \mathrm{Sen}(\Delta_p)$ consistent
* $p \leq q$ iff $\Delta_p \subseteq \Delta_q$ and $\Gamma_p \subseteq \Gamma_q$
* $\Delta(p) = \Delta_p$
* $f(p) = \Gamma_p \cap \mathrm{Sen}_b(\Delta_p)$

Forcing relation

* $p \Vdash \varphi$ when $\varphi \in f(p)$ for atomic sentences
* $p \Vdash t\,(a \fatsemi b)\,t'$ when $p \Vdash t\,a\,\tau$ and $p \Vdash \tau\,b\,t'$ for some Δ_p-term τ
* $p \Vdash t\,(a \cup b)\,t'$ when $p \Vdash t\,a\,t'$ or $p \Vdash t\,b\,t'$
* $p \Vdash t\,a^*\,t'$ when $p \Vdash t\,a^n\,t'$ for some $n \in \mathbb{N}$
* $p \Vdash \neg\varphi$ when $q \nVdash \varphi$ for all $q \geq p$
* $p \Vdash \bigvee \Phi$ when $p \Vdash \varphi$ for some $\varphi \in \Phi$
* $p \Vdash \exists x \cdot \varphi$ when $p \Vdash \theta(\varphi)$ for some substitution θ

Forcing properties

* $p \Vdash \neg\neg\varphi$ iff for all $q \geq p$ there is $r \geq q$ such that $r \Vdash \varphi$
* if $p \leq q$ and $p \Vdash \varphi$ then $q \Vdash \varphi$
* if $p \Vdash \varphi$ then $p \Vdash \neg\neg\varphi$
* we cannot have both $p \Vdash \varphi$ and $p \Vdash \neg\varphi$
* $\Gamma_p \vdash \varphi$ iff $p \Vdash \neg\neg\varphi$

Generic sets and models

Theorem
1. Every $p \in P$ belongs to a generic set $G \subseteq P$.

* G is an ideal
* for all $q \in G$ and sentences φ there is $r \in G$ such that $r \geq q$ and either $r \Vdash \varphi$ or $r \Vdash \neg\varphi$

Generic sets and models

Theorem

1. Every $p \in P$ belongs to a generic set $G \subseteq P$.

* G is an ideal
* for all $q \in G$ and sentences φ there is $r \in G$ such that $r \geq q$ and either $r \Vdash \varphi$ or $r \Vdash \neg\varphi$

2. If $p \in G$ and G is generic, then $\bigcup\{\Gamma_q \mid q \in G\}$ is a maximally consistent set that includes Γ_p.

Generic sets and models

Theorem

1. Every $p \in P$ belongs to a generic set $G \subseteq P$.

* G is an ideal
* for all $q \in G$ and sentences φ there is $r \in G$ such that $r \geq q$ and either $r \Vdash \varphi$ or $r \Vdash \neg\varphi$

2. If $p \in G$ and G is generic, then $\bigcup\{\Gamma_q \mid q \in G\}$ is a maximally consistent set that includes Γ_p.

3. G admits a countable and reachable generic model A.

* $A \models \varphi$ iff $q \Vdash \varphi$ for some $q \in G$

Completeness

Theorem

1. Every consistent set of sentences has a countable model.

Completeness

Theorem
1. Every consistent set of sentences has a countable model.

2. $\Gamma \vdash \varphi$ iff $\Gamma \models \varphi$.

Completeness

Theorem
1. Every consistent set of sentences has a countable model.

2. $\Gamma \vdash \varphi$ iff $\Gamma \models \varphi$.

* suppose ad absurdum $\Gamma \nvdash \varphi$

Completeness

Theorem
1. Every consistent set of sentences has a countable model.

2. $\Gamma \vdash \varphi$ iff $\Gamma \models \varphi$.

* suppose ad absurdum $\Gamma \nvdash \varphi$
* we get $\Gamma \nvdash \neg\neg\varphi$, hence $\Gamma \cup \{\neg\varphi\} \nvdash \bot$

Completeness

Theorem
1. Every consistent set of sentences has a countable model.

2. $\Gamma \vdash \varphi$ iff $\Gamma \models \varphi$.

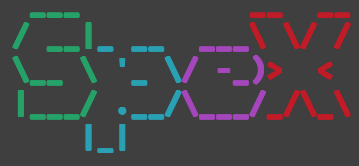
* suppose ad absurdum $\Gamma \nvdash \varphi$
* we get $\Gamma \nvdash \neg\neg\varphi$, hence $\Gamma \cup \{\neg\varphi\} \nvdash \bot$
* it follows that $\Gamma \cup \{\neg\varphi\}$ is consistent, so it has a model

Completeness

Theorem
1. Every consistent set of sentences has a countable model.

2. $\Gamma \vdash \varphi$ iff $\Gamma \models \varphi$.

* suppose ad absurdum $\Gamma \nvdash \varphi$
* we get $\Gamma \nvdash \neg\neg\varphi$, hence $\Gamma \cup \{\neg\varphi\} \nvdash \bot$
* it follows that $\Gamma \cup \{\neg\varphi\}$ is consistent, so it has a model
* thus contradicting $\Gamma \models \varphi$

Back to 

Term rewriting as a substructure for specification-language interpreters

* solid mathematical foundation
* algebraic, close to standard notation used by working theoretical computer scientists
* great for rapid prototyping

but

* still somewhat rigid as a meta-language
* limited support for modularization

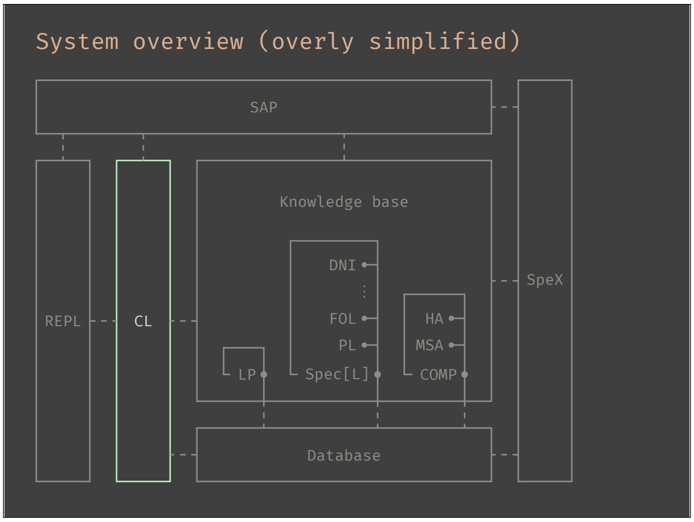
Object-based programming

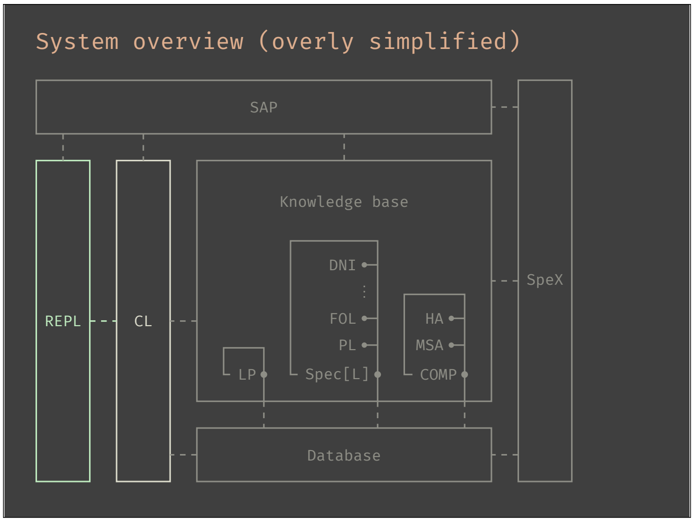
* we rewrite configurations
  multisets of ↲
  1. objects `< Id : Class | Attributes >`
  2. messages `message(To, From, Arguments)`
* using rules of the form

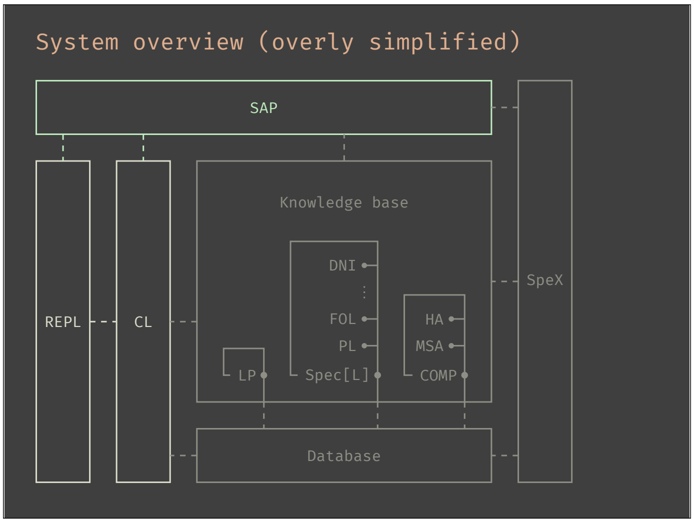
```
rl < Id : Class | ... >
   message(Id, ...)
 ⇒ ...
```

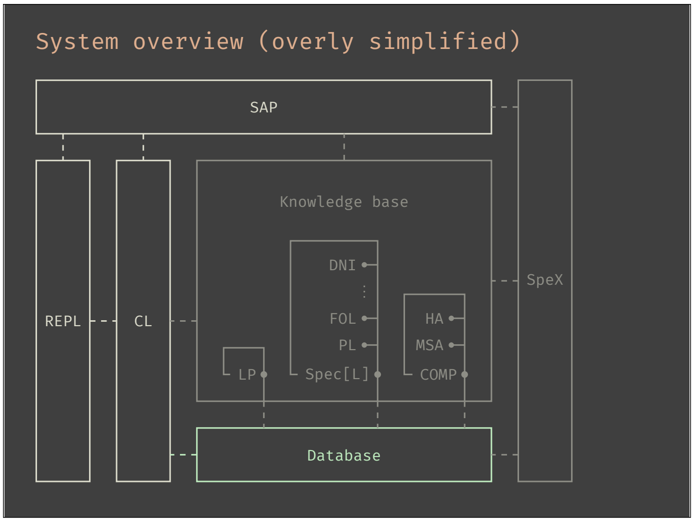
SpeX

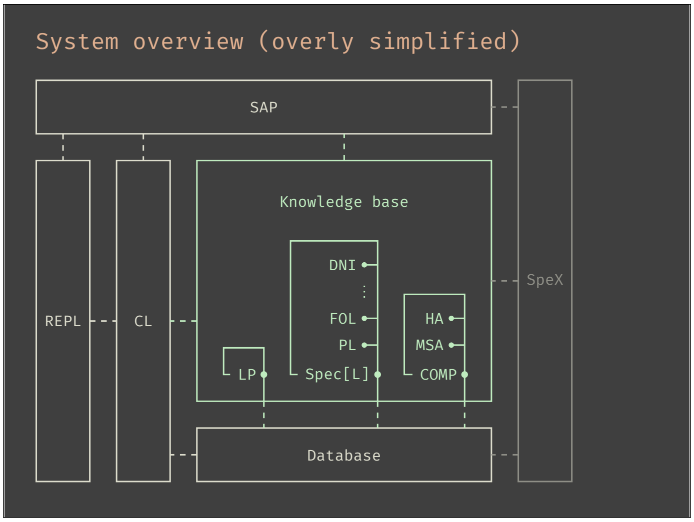
* not a plain interpreter, but an ‘environment’
* integrates specification-language processors
* language agnostic
* offers a basic system UI ‘for free’
* based on Maude 3 (OBP with external rewrites)

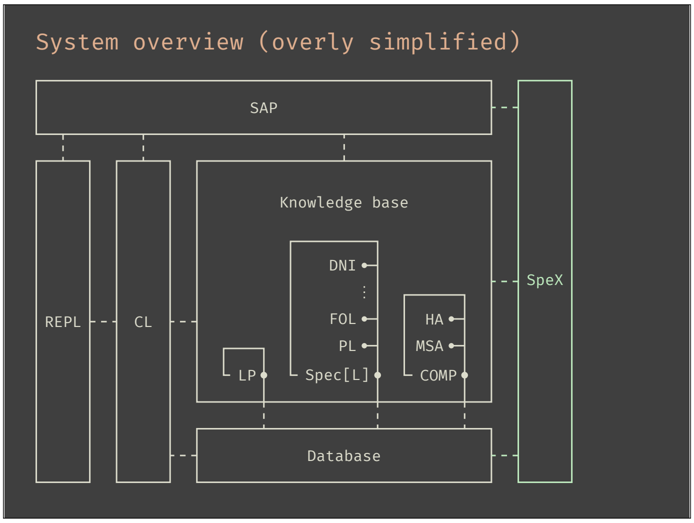

System overview (overly simplified)
SAP
Knowledge base
DNI
FOL
PL
Spec[L]
HA
MSA
COMP
LP
REPL
CL
SpeX
Database


System overview (overly simplified)
SAP
Knowledge base
DNI
FOL
PL
Spec[L]
HA
MSA
COMP
LP
REPL
CL
SpeX
Database


System overview (overly simplified)
SAP
Knowledge base
DNI
FOL
PL
Spec[L]
HA
MSA
COMP
LP
REPL
CL
SpeX
Database


System overview (overly simplified)
SAP
REPL
CL
Knowledge base
DNI
FOL
PL
Spec[L]
LP
HA
MSA
COMP
Database
SpeX


System overview (overly simplified)
SAP
REPL
CL
Knowledge base
DNI
FOL
PL
Spec[L]
LP
HA
MSA
COMP
Database
SpeX


System overview (overly simplified)
SAP
REPL
CL
Knowledge base
DNI
FOL
PL
Spec[L]
LP
HA
MSA
COMP
Database
SpeX

Integrating new languages into SpeX

* by means of processors
  - objects of class PROC
  - interact with SpeX (the object)

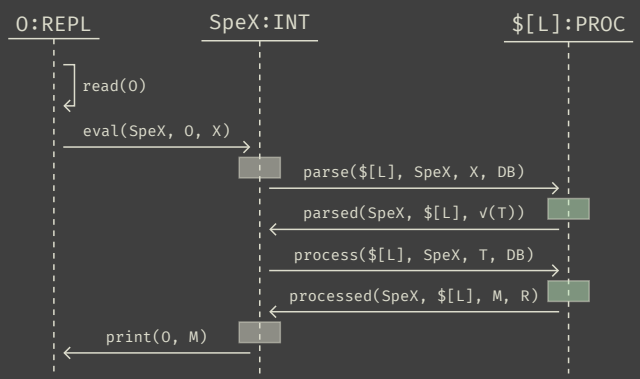
* receive messages of the form

```
parse($[L], SpeX, Input, DB)
process($[L], SpeX, AnnotatedTerm, DB)
```

* reply with messages of the form

```
parsed(SpeX, $[L], ParsingOutcome)
processed(SpeX, $[L], Text, Record)
```

A basic execution scenario



Example: Spec[DNI] (codev'd J. Fiadeiro and C. Chiriță)

```
spec Bind is
  including Base .
  mod __bind_ : Protein Organelle >< Coat .
  ...
  ax store k:Nominal
     forall-local {p:Protein, o:Organelle}
     [ p o bind z:Coat ]
     (forall-local {o':Organelle}
       brane(o') = @(k) brane(o))
     and
     (forall-local {c':Coat}
       c' = z implies brane(c') = @(k) brane(o))
     [label: bind-effect] .
endspec
```

```
Example: Spec[DNI] (codev'd J. Fiadeiro
                    and C. Chiriță)

spec Bind is
  including Base .
  mod __bind_ : Protein Organelle × Coat .
  ...
  ax store k:Nominal
     forall-local {p:Protein, o:Organelle}
     [ p o bind z:Coat ]
     (forall-local {o':Organelle}
       brane(o') = @(k) brane(o))
     and
     (forall-local {c':Coat}
       c' = z implies brane(c') = @(k) brane(o))
     [label: bind-effect] .
endspec
```

```
Example: Spec[DNI] (codev'd J. Fiadeiro
                    and C. Chiriță)

spec Bind is
  including Base .
  mod __bind_ : Protein Organelle × Coat .
  ...
  ax store k:Nominal
     forall-local {p:Protein, o:Organelle}
     [ p o bind z:Coat ]
     (forall-local {o':Organelle}
       brane(o') = @(k) brane(o))
     and
     (forall-local {c':Coat}
       c' = z implies brane(c') = @(k) brane(o))
     [label: bind-effect] .
endspec
```

```
Example: Spec[DNI] (codev'd J. Fiadeiro
                    and C. Chiriță)

spec Bind is
  including Base .
  mod __bind_ : Protein Organelle × Coat .
  ...
  ax store k:Nominal
     forall-local {p:Protein, o:Organelle}
     [ p o bind z:Coat ]
     (forall-local {o':Organelle}
       brane(o') = @(k) brane(o))
     and
     (forall-local {c':Coat}
       c' = z implies brane(c') = @(k) brane(o))
     [label: bind-effect] .
endspec
```

```
Example: COMP (codev'd R.Diaconescu)

bobj WATCH is
  syncing (UP-TO-24-COUNTER as HOUR)
      and (UP-TO-60-COUNTER as MINUTE)
      and (UP-TO-60-COUNTER as SECOND) .
  op _:_:_ : Nat Nat Nat → State .
  act tick_ : State → State .
  act inc-min_ : State → State .
  ...
endbo

open WATCH
  check tick inc-min (H:Nat : M:Nat : S:Nat)
      ~ inc-min tick (H:Nat : M:Nat : S:Nat)
  forall M:Nat < 60 = true
     and S:Nat < 60 = true .
close
```


```
Example: COMP (codev'd R.Diaconescu)

bobj WATCH is
  syncing (UP-TO-24-COUNTER as HOUR)
      and (UP-TO-60-COUNTER as MINUTE)
      and (UP-TO-60-COUNTER as SECOND) .
  op _:_:_ : Nat Nat Nat → State .
  act tick_ : State → State .
  act inc-min_ : State → State .
  ...
endbo

open WATCH
  check tick inc-min (H:Nat : M:Nat : S:Nat)
      ~ inc-min tick (H:Nat : M:Nat : S:Nat)
  forall M:Nat < 60 = true
     and S:Nat < 60 = true .
close
```


```
Example: IPDL (dev'd K.Sojakova,
                    M.Codescu,
                and J.Gancher)

protocol real =
  newfamily SendInShare[bound N + 2 bound N + 2
                        dependentBound I]
            indices: m, n, i ... : bool in
  newfamily OTMsg-0[bound N + 2 bound N + 2
                    bound K ]
            indices: n, m, k ... : bool in
  ...
  parties || 1OutOf4OTReal
  where parties = ...
    and 1OutOf4OTReal = ...

See https://arxiv.org/abs/2507.22705
```

Example: Spec[TA] (codev'd D. Găină and A. Riesco)

```
spec Institute is
  including CCS .

  ops theorem, coffee, coin ...
  ops Institute, Mathematician, CoffeeVM ...

  ax Institute
     = (Mathematician | CoffeeVM) \ coin \ coffee .
  ax Mathematician
     =(tau)> (snd(coin) . rcv(coffee) .
              snd(theorem) . Mathematician) .
  ax CoffeeVM
     =(tau)> (rcv(coin) . snd(coffee) . CoffeeVM) .
endspec
```

Obtaining SpeX and TATP

* from the git repositories:

  https://gitlab.com/ittutu/spex

  https://github.com/Transition-Algebra/TATP

* then, provided Maude 3(>.2) is installed:

```
./configure
make
[sudo] make install
```

Happy hacking!

Executable Specifications in Transition Algebra

A. Riesco (joint work with Daniel Găină and Ionuț Țuțu)

Universidad Complutense de Madrid, Madrid, Spain

Workshop on Logic, Algebra and Category Theory, LAC 2025
Fukuoka, Japan
October 3, 2025

Introduction

- Novel denotational semantics called Transition Algebra (TA), introduced for algebraic specification languages that are executable through rewriting.
- TA supports actions, which are composed from binary relations using union ($\cup$), composition ($\fatsemi$), and iteration ($*$).
- TA can provide a rigorous foundation, in logical terms, for a core fragment of the Maude strategy language.
- Maude is a specification language that implements Rewriting Logic.
- The strategy language significantly enhances its expressive power by enabling fine-grained control over the application of transition rules.

Introduction

- TA follows the principles of institution theory, a framework that formalizes the notion of a logical system using four ingredients: signatures, sentences, models, and a satisfaction relation between models and sentences.
- In TA, each signature is equipped with a separate set of plain, unstructured *transition labels*, which are interpreted in models as *atomic transition relations*.
- Here, we adopt a more minimalistic approach.
- We use terms both to denote elements in models and to form atomic transition labels.

Introduction

- It allows us to quantify over atomic transitions using ordinary first-order variables.
- This mechanism serves as a method of encoding second-order quantification (over transition relations) using first-order quantifiers.
- We focus on defining a fragment of TA with good computational properties, making specifications written in this logical framework executable via rewriting or narrowing.
- We introduce a notion of Horn clause in two distinct forms: equational and transitional.
- The framework allows the conditions to include actions that are arbitrarily complex and not necessarily atomic.

Introduction

- We define a system of proof rules for reasoning about clauses in TA.
- The resulting entailment system is both sound and complete.
- Our completeness proof does not rely on the use of a *cut* rule.
- This means that properties of systems specified by clauses in TA can be proved without requiring the discovery of intermediate results.

Introduction - rewriting

- Our rewrite rules are derived from general Horn clauses by imposing restrictions on the occurrences of variables in order to ensure executability.
- Conditions, while not themselves rewrite rules, may include complex compositions of actions and thus require their own operational semantics.
- We proceed in two steps.

Introduction - rewriting

- First, we define a rewriting relation on sets of conditions.
- We interpret these sets of conditions as goals G that are meant to be proved by rewriting.
- Proving a goal amounts to rewriting it to the empty set.
- To prove G we may need to apply other rewrite rules, which may be conditional.
- Unlike term rewriting, for goal rewriting we do not check the conditions of rules straight away; instead, we add them to G so that they are checked at some point during rewriting.
- The purpose is to develop a framework for rewriting that is as general as possible, independent of any particular computational strategy.

Introduction - rewriting

- Second, we define term rewriting using equational rewrite rules and, separately, term transitions using transitional rewrite rules.
- The latter captures the idea of rewriting along a path described by an action, hence giving an operational interpretation to transitions in the system.
- This separation clarifies how terms transition in response to rules and how side conditions are validated through structured rewriting.

Introduction - rewriting

- We establish a completeness result for rewriting, thus ensuring that a goal G follows (semantically) from a set Γ of clauses only if G can be operationally reduced to the empty set through a sequence of rewriting steps.
- This result bridges the gap between the model-theoretic semantics of TA and its operational semantics defined via rewriting.
- To achieve it, the set of rules needs to satisfy two key assumptions: *confluence* of equational rewriting rules and *coherence* of transitional rewrite rules.

Introduction - narrowing

- To define narrowing, we use the same rewrite rules as those used for rewriting.
- The definition of narrowing can be obtained directly from rewriting by replacing matching with unification.
- We consider queries defined as existentially quantified goals, revisit the classical notion of solution to a query, and propose a suitable narrowing relation.
- A query $\exists X \cdot Q$ is satisfiable with respect to a set Γ of clauses, meaning that there exists a substitution θ such that $Q\theta$ follows from Γ, if and only if $\exists X \cdot Q$ can be reduced to a trivial query of the form $\exists X' \cdot \emptyset$ through a sequence of narrowing steps that 'compute' θ.
- The completeness of narrowing holds under confluence, coherence, and also an additional termination requirement for goal rewriting.

Introduction - implementation

- We have implemented a tool prototype for writing TA specifications, reducing terms, checking transitions, and solving queries.
- This was made possible by SpeX, which is both a heterogeneous logical environment and a collection of Maude libraries that facilitate the development of new specification languages.
- Thanks to Maude meta-level, it has been possible to define a dedicated syntax for TA specifications, to parse specifications, transform them into internal Maude representations, and then use those representations to reduce terms and solve queries.

Introduction - implementation

- The close relationship between Rewriting Logic and Transition Algebra makes Maude the most appropriate language for these tasks, as it already provides mechanisms for parsing, matching, and unification.
- Both the sentences and the execution mechanisms of Rewriting Logic and Transition Algebra are different, hence no direct translation is possible.
- For that reason, the tool builds on the features provided by Maude to obtain specially designed resolution procedures for Transition Algebra.

Horn clause in Transition Algebra

- Remember we have $t_1 = t_2$ and $\mathfrak{a}(t_1, t_2)$.
- A Horn clause is a sentence of the form $\forall X \cdot \varphi$ if H where
  1. X is a finite block of variables;
  2. H is a finite set of equations and transition rules; and
  3. φ is an atomic sentence.

Horn clause in Transition Algebra

We start with atomic sentences.

$$(R)\ \frac{}{\Gamma \vdash t = t} \qquad (S)\ \frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash t_2 = t_1} \qquad (T)\ \frac{\Gamma \vdash t_1 = t_2 \quad \Gamma \vdash t_2 = t_3}{\Gamma \vdash t_1 = t_3}$$

$$(F)\ \frac{\Gamma \vdash t_1 = t_1' \ \ldots\ \Gamma \vdash t_n = t_n'}{\Gamma \vdash \sigma(t_1, \ldots, t_n) = \sigma(t_1', \ldots, t_n')} \qquad (L)\ \frac{\Gamma \vdash t_i = t_i' \text{ for both } i \in \{1,2\} \quad \Gamma \vdash \tau = \tau' \quad \Gamma \vdash \tau(t_1, t_2)}{\Gamma \vdash \tau'(t_1', t_2')}$$

These rules are sound.

Horn clause in Transition Algebra

We add transition relations and quantifier-free clauses.

$$(Comp)\ \frac{\Gamma \vdash \mathfrak{a}_1(t_1, t_2) \quad \Gamma \vdash \mathfrak{a}_2(t_2, t_3)}{\Gamma \vdash (\mathfrak{a}_1 \fatsemi \mathfrak{a}_2)(t_1, t_3)} \qquad (Union)\ \frac{\Gamma \vdash \mathfrak{a}_i(t_1, t_2)}{\Gamma \vdash (\mathfrak{a}_1 \cup \mathfrak{a}_2)(t_1, t_2)}$$

$$(Eq)\ \frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash \mathfrak{a}^0(t_1, t_2)} \qquad (Star)\ \frac{\Gamma \vdash \mathfrak{a}^n(t_1, t_2)}{\Gamma \vdash \mathfrak{a}^*(t_1, t_2)}$$

$$(MP)\ \frac{\Gamma \vdash \varphi \text{ if } H \quad \Gamma \vdash \phi \text{ for all } \phi \in H}{\Gamma \vdash \varphi} \qquad (Subst)\ \frac{\Gamma \vdash \forall X \cdot \gamma}{\Gamma \vdash \gamma\theta} \text{ for all } \theta : X \to \emptyset$$

These* rules are sound.

Horn clause in Transition Algebra

We add transition rules and quantified clauses.

$$(Comp_l)\ \frac{\Gamma \cup \{\mathfrak{a}_1(t_1, x), \mathfrak{a}_2(x, t_2)\} \vdash_{\Sigma[z]} \gamma}{\Gamma \cup \{(\mathfrak{a}_1 \fatsemi \mathfrak{a}_2)(t_1, t_2)\} \vdash \gamma} \qquad (Union_l)\ \frac{\Gamma \cup \{\mathfrak{a}_i(t_1, t_2)\} \vdash \gamma \text{ for all } i \in \{1, 2\}}{\Gamma \cup \{(\mathfrak{a}_1 \cup \mathfrak{a}_2)(t_1, t_2)\} \vdash \gamma}$$

$$(Star_l)\ \frac{\Gamma \cup \{\mathfrak{a}^n(t_1, t_2)\} \vdash \gamma \text{ for all } n \in \mathbb{N}}{\Gamma \cup \{\mathfrak{a}^*(t_1, t_2)\} \vdash \gamma} \qquad (Imp)\ \frac{\Gamma \cup H \vdash \varphi}{\Gamma \vdash \varphi \text{ if } H}$$

$$(UQ_E)\ \frac{\Gamma \vdash_{\Sigma} \forall X \cdot \gamma}{\Gamma \vdash_{\Sigma[X]} \gamma} \qquad (UQ_I)\ \frac{\Gamma \vdash_{\Sigma[X]} \gamma}{\Gamma \vdash_{\Sigma} \forall X \cdot \gamma}$$

Horn clause in Transition Algebra

Lemma (Soundness)

The proof rules defined in the tables above are sound. That is, the least entailment system closed under the proof rules defined in the tables above is sound.

Theorem (Completeness)

Let $\vdash$ be any sound entailment system closed under the proof rules defined in the tables above. For all sets of clauses Γ and all clauses γ defined over the same signature, we have $\Gamma \models \gamma$ iff $\Gamma \vdash \gamma$.

Operational semantics - rewriting

- We consider two kinds of rewrite rules to 'execute' specifications.
- An *equational rewrite rule* is a Horn clause of the form $\forall X \cdot (\ell = r)$ if H such that ℓ is not a variable and $X = \mathrm{var}(\ell) \cup \mathrm{var}(H)$.
- A *transitional rewrite rule* is a Horn clause of the form $\forall X \cdot \tau(\ell, r)$ if H such that ℓ is not a variable and $X = \mathrm{var}(\tau) \cup \mathrm{var}(\ell) \cup \mathrm{var}(H)$.
- Whenever we refer to a *term rewriting system* in Transition Algebra we mean a pair (Σ, Γ) consisting of a signature Σ and a set $\Gamma \subseteq \mathrm{Sen}(\Sigma)$ of equational and transitional rewrite rules.

Operational semantics - rewriting

Definition (Rewriting step)

Let (Σ, Γ) be a term rewriting system. The *one-step-rewriting* relation on ground equations and transitions is defined as follows:

- EQ: $C[t] \overset{\text{EQ}}{\rightsquigarrow} C[r\theta] \cup H\theta$ whenever $t = \ell\theta$ for some rule $\forall Y \cdot (\ell = r)$ if $H \in \Gamma$ and substitution $\theta : Y \to \emptyset$;
- $=$: $\{t = t\} \cup G \overset{=}{\rightsquigarrow} G$;
- TR: $\{\tau(t_1, t_2)\} \cup G \overset{\text{TR}}{\rightsquigarrow} \{r\theta = t_2\} \cup H\theta \cup G$ whenever $\tau = \kappa\theta$ and $t_1 = \ell\theta$ for some $\forall Y \cdot \kappa(\ell, r)$ if $H \in \Gamma$ and $\theta : Y \to \emptyset$;
- $\fatsemi$: $\{(\mathfrak{a}_1 \fatsemi \mathfrak{a}_2)(t_1, t_2)\} \cup G \overset{;}{\rightsquigarrow} \{\mathfrak{a}_1(t_1, t), \mathfrak{a}_2(t, t_2)\} \cup G$, for any Σ-term t;
- $\cup$: $\{(\mathfrak{a}_1 \cup \mathfrak{a}_2)(t_1, t_2)\} \cup G \overset{\cup}{\rightsquigarrow} \{\mathfrak{a}_i(t_1, t_2)\} \cup G$, for any $i \in \{1, 2\}$;
- $\circlearrowleft$: $\{\mathfrak{a}^*(t_1, t_2)\} \cup G \overset{\circlearrowleft}{\rightsquigarrow} \{\mathfrak{a}^n(t_1, t_2)\} \cup G$, for any $n \in \mathbb{N}$.

Operational semantics - rewriting

Definition (Rewriting)

The general *rewriting* relation $\overset{*}{\rightsquigarrow}$ on Transition Algebra goals is the reflexive and transitive closure of the one-step-rewriting relation. We define it by $\overset{*}{\rightsquigarrow} = \bigcup\{\overset{n}{\rightsquigarrow} \mid n \in \mathbb{N}\}$, where:
$G \overset{0}{\rightsquigarrow} G$, and
if $G_1 \overset{lb}{\rightsquigarrow} G_2$ and $G_2 \overset{n}{\rightsquigarrow} G_3$, then $G_1 \overset{n+1}{\rightsquigarrow} G_3$, for each label $lb \in \{\text{EQ}, =, \text{TR}, \fatsemi, \cup, \circlearrowleft\}$.

Operational semantics - rewriting

- So far, we have defined rewriting on sets of ground equations and transitions.
- To extend it to equations and transitions with variables from some given block X, we treat the variables in X as new constants that we add to the underlying signature.
- Concretely, for any term rewriting system (Σ, Γ) and block X of variables, rewriting sets of equations and transitions with variables from X amounts to applying rewriting over $(\Sigma[X], \Gamma)$.

Operational semantics - rewriting

This notion of goal rewriting is sound in the following sense: in the presence of the rewriting rules (Horn clauses) provided by Γ, the outcome of a rewrite is always stronger than the original goal.

Theorem (Soundness of rewriting)

Let (Σ, Γ) *be a term rewriting system. For any goals* $G, G' \subseteq \mathrm{Sen}(\Sigma)$,

$$G \overset{*}{\rightsquigarrow} G' \quad \textit{implies} \quad \Gamma \cup G' \models G.$$

Corollary

For any term rewriting system (Σ, Γ) *and goal* $G \subseteq \mathrm{Sen}(\Sigma)$, $G \overset{*}{\rightsquigarrow} \emptyset$ *implies* $\Gamma \models G$.

Operational semantics - narrowing

- Next, we look for solutions to *queries*, which are defined as existentially quantified goals and are usually denoted $\exists X \cdot Q$, where X is a block of variables and Q is a goal with variables from X.
- As in equational logic programming, a *solution* to $\exists X \cdot Q$ is a substitution $\theta \colon X \to X'$ such that $\Gamma \models \forall X' \cdot Q\theta$.
- We solve queries by narrowing, which is similar to rewriting, but employs most general unifiers instead of matching substitutions in order to compute 'values' for the variables in X such that the equalities and transitions in Q hold true.

Operational semantics - narrowing

Definition (Narrowing step)

Let (Σ, Γ) be a term rewriting system. The *one-step-narrowing* relation on queries, indexed by so-called *computed substitutions*, is defined as follows:

EQ $\exists X \cdot C[t] \overset{\mathrm{EQ}}{\twoheadrightarrow}_{\psi_X} \exists X' \cdot (C[r] \cup H)\psi$ when t is not a variable, $\psi = \mathrm{mgu}\{t \doteq \ell\} \colon X \uplus Y \to X'$ for some rule $\forall Y \cdot (\ell = r)$ if $H \in \Gamma$, and $\psi_X \colon X \to X'$ is the restriction of ψ to X;

$=$ $\exists X \cdot \{t_1 = t_2\} \cup Q \overset{=}{\twoheadrightarrow}_{\psi} \exists X' \cdot Q\psi$ when $\psi = \mathrm{mgu}\{t_1 \doteq t_2\} \colon X \to X'$;

TR $\exists X \cdot \{\tau(t_1, t_2)\} \cup Q \overset{\mathrm{TR}}{\twoheadrightarrow}_{\psi_X} \exists X' \cdot (\{r = t_2\} \cup H \cup Q)\psi$ when $\psi = \mathrm{mgu}\{\tau \doteq \kappa, t_1 \doteq \ell\} \colon X \uplus Y \to X'$ for some rule $\forall Y \cdot \kappa(\ell, r)$ if $H \in \Gamma$ and $\psi_X \colon X \to X'$ is the restriction of ψ to X;

$\fatsemi$ $\exists X \cdot (a_1 \fatsemi a_2)(t_1, t_2)\} \cup Q \overset{;}{\twoheadrightarrow}_{\iota_x} \exists X \cup \{x\} \cdot \{a_1(t_1, x), a_2(x, t_2)\} \cup Q$, for any new variable x, distinct from those in X, and for the obvious inclusion $\iota_x \colon X \hookrightarrow X \cup \{x\}$;

Operational semantics - narrowing

Definition (Narrowing step)

Let (Σ, Γ) be a term rewriting system. The *one-step-narrowing* relation on queries, indexed by so-called *computed substitutions*, is defined as follows:

- $\cup$ $\exists X \cdot \{(\mathfrak{a}_1 \cup \mathfrak{a}_2)(t_1, t_2)\} \cup Q \xrightarrow{\cup} \!\!\!\!\rightarrow_{1_X} \exists X \cdot \{\mathfrak{a}_i(t_1, t_2)\} \cup Q$, for any $i \in \{1, 2\}$, where $1_X \colon X \to X$ is the identity; and
- $\circlearrowleft$ $\exists X \cdot \{\mathfrak{a}^*(t_1, t_2)\} \cup Q \xrightarrow{\circlearrowleft} \!\!\!\!\rightarrow_{1_X} \exists X \cdot \{\mathfrak{a}^n(t_1, t_2)\} \cup Q$, for any $n \in \mathbb{N}$, where $1_X \colon X \to X$ is the identity.

Operational semantics - narrowing

The narrowing relation corresponds to the reflexive and transitive closure of the one-step-narrowing relation.

Definition (Narrowing)

The general *narrowing* relation is defined by $\xrightarrow{*} \!\!\!\!\rightarrow_\psi = \bigcup\{\xrightarrow{n} \!\!\!\!\rightarrow_\psi \mid n \in \mathbb{N}\}$, where:

1. $\exists X \cdot Q \xrightarrow{0} \!\!\!\!\rightarrow_{1_X} \exists X \cdot Q$, and
2. if $\exists X_1 \cdot Q_1 \xrightarrow{lb} \!\!\!\!\rightarrow_{\psi_1} \exists X_2 \cdot Q_2$ and $\exists X_2 \cdot Q_2 \xrightarrow{n} \!\!\!\!\rightarrow_{\psi_2} \exists X_2 \cdot Q_3$, then $\exists X_1 \cdot Q_1 \xrightarrow{n+1} \!\!\!\!\rightarrow_{\psi_1 ; \psi_2} \exists X_3 \cdot Q_3$, for each label $lb \in \{\mathtt{EQ}, =, \mathtt{TR}, ;, \cup, \circlearrowleft\}$.

Operational semantics - narrowing

The following result establishes our first link between narrowing and rewriting.

Lemma (Reduction Lemma)

If $\exists X \cdot Q \xrightarrow{n} \!\!\!\!\rightarrow_\psi \exists X' \cdot Q'$, *then* $Q\psi \overset{n}{\rightsquigarrow} Q'$.

One important consequence of Reduction Lemma is the soundness of narrowing.

Theorem (Soundness of narrowing)

Let (Σ, Γ) *be a term rewriting system. If* $\exists X \cdot Q \xrightarrow{*} \!\!\!\!\rightarrow_\psi \exists X' \cdot Q'$, *then for any solution* $\theta \colon X' \to X''$ *to* $\exists X' \cdot Q'$, *the substitution* $\psi ; \theta$ *is a solution to* $\exists X \cdot Q$.

Operational semantics - term rewriting

Definition (Term rewriting)

Let (Σ, Γ) be a term rewriting system. We overload the notation used for rewriting goals and define a *one-step-rewriting* relation on Σ-terms as follows: for any context c and term t, $c[t] \xrightarrow{\text{EQ}} c[r\theta]$ if there exist an equational rule $\forall Y \cdot (\ell = r)$ if $H \in \Gamma$ and a substitution $\theta\colon Y \to \emptyset$ such that $t = \ell\theta$ and $H\theta \overset{*}{\rightsquigarrow} \emptyset$.
The family of *n-step rewriting* relations $\{\xrightarrow{n} \subseteq T_\Sigma \times T_\Sigma \mid n \in \mathbb{N}\}$ is defined inductively as follows:

1. $t \xrightarrow{0} t$;
2. if $t_1 \xrightarrow{\text{EQ}} t_2$ and $t_2 \xrightarrow{n} t_3$, then $t_1 \xrightarrow{n+1} t_3$.

We write $t \xrightarrow{*} t'$ whenever $t \xrightarrow{n} t'$ for some $n \in \mathbb{N}$.

Operational semantics - term rewriting

Term rewriting extends canonically from terms to actions:

- $\fatsemi$: $\mathfrak{a}_1 \fatsemi \mathfrak{a}_2 \xrightarrow{*} \mathfrak{a}_1' \fatsemi \mathfrak{a}_2'$ when $\mathfrak{a}_1 \xrightarrow{*} \mathfrak{a}_1'$ and $\mathfrak{a}_2 \xrightarrow{*} \mathfrak{a}_2'$;
- $\cup$: $\mathfrak{a}_1 \cup \mathfrak{a}_2 \xrightarrow{*} \mathfrak{a}_1' \cup \mathfrak{a}_2'$ when $\mathfrak{a}_1 \xrightarrow{*} \mathfrak{a}_1'$ and $\mathfrak{a}_2 \xrightarrow{*} \mathfrak{a}_2'$; and
- $\circlearrowleft$: $\mathfrak{a}^* \xrightarrow{*} \mathfrak{b}^*$ when $\mathfrak{a} \xrightarrow{*} \mathfrak{b}$.

Operational semantics - term rewriting

Term rewriting together with action rewriting allows us to define a transition relation on terms.

Definition (Term transition)

Let (Σ, Γ) be a term rewriting system. The relation $\xrightarrow{\mathfrak{a}}$ on Σ-terms is defined by induction on the structure of $\mathfrak{a}$:

- TR: $t_1 \xrightarrow{\tau} t_2$ if there exist a rewrite rule $\forall Y \cdot \kappa(\ell, r)$ if $H \in \Gamma$ and a substitution $\theta : Y \to \emptyset$ such that
  1. $t_1 \xrightarrow{*} \ell\theta$,
  2. $\tau \xrightarrow{*} \kappa\theta$,
  3. $r\theta \xrightarrow{*} t_2$, and
  4. $H\theta \overset{*}{\rightsquigarrow} \emptyset$;
- $\fatsemi$: $t_1 \xrightarrow{\mathfrak{a}_1 \fatsemi \mathfrak{a}_2} t_2$ if $t_1 \xrightarrow{\mathfrak{a}_1} t$ and $t \xrightarrow{\mathfrak{a}_2} t_2$ for some term $t \in T_\Sigma$;
- $\cup$: $t_1 \xrightarrow{\mathfrak{a}_1 \cup \mathfrak{a}_2} t_2$ if $t_1 \xrightarrow{\mathfrak{a}_i} t$ for some $i \in \{1, 2\}$; and
- $\circlearrowleft$: $t_1 \xrightarrow{\mathfrak{a}^*} t_2$ if $t_1 \xrightarrow{\mathfrak{a}^n} t_2$ for some $n \in \mathbb{N}$.

Operational semantics - term rewriting

Lemma

Let (Σ, Γ) be a term rewriting system. For all Σ-actions $\mathfrak{a}, \mathfrak{b}$ and all Σ-terms t_1, t_2,

$$\text{if } \mathfrak{a} \xrightarrow{*} \mathfrak{b} \text{ and } t_1 \xrightarrow{\mathfrak{b}} t_2, \text{ then } t_1 \xrightarrow{\mathfrak{a}} t_2.$$

Lemma

Let (Σ, Γ) be a term rewriting system.

1. *If $t_1 \xrightarrow{*} u \xleftarrow{*} t_2$, then $\{t_1 = t_2\} \rightsquigarrow^{*} \emptyset$.*
2. *If $t_1 \xrightarrow{\mathfrak{a}} u \xleftarrow{*} t_2$, then $\{\mathfrak{a}(t_1, t_2)\} \rightsquigarrow^{*} \emptyset$.*

Operational semantics - term rewriting

Corollary

Let (Σ, Γ) be a term rewriting system.

- *Term rewriting is sound, meaning that $t_1 \xrightarrow{*} t_2$ implies $\Gamma \models t_1 = t_2$.*
- *Term transition is sound, meaning that $t_1 \xrightarrow{\mathfrak{a}} t_2$ implies $\Gamma \models \mathfrak{a}(t_1, t_2)$.*

Operational semantics - term rewriting

Proposition

Let (Σ, Γ) be a ground confluent term rewriting system.

1. *If $\{t_1 = t_2\} \rightsquigarrow^{*} \emptyset$, then $t_1 \xrightarrow{*} u \xleftarrow{*} t_2$ for some Σ-term u.*
2. *Moreover, if (Σ, Γ) is coherent and $\{\mathfrak{a}(t_1, t_2)\} \rightsquigarrow^{*} \emptyset$, then $t_1 \xrightarrow{\mathfrak{a}} u \xleftarrow{*} t_2$ for some Σ-term u.*

Completeness of operational semantics - rewriting

All completeness results are based on confluence and coherence assumptions.

Theorem (Completeness of rewriting)

Let (Σ, Γ) *be a ground confluent and coherent term rewriting system. For all goals* $G \subseteq \mathrm{Sen}(\Sigma)$,

$$G \overset{*}{\rightsquigarrow} \emptyset \quad \textit{if} \quad \Gamma \models G.$$

Completeness of operational semantics - narrowing

Definition (Strict goal rewriting)

We say that a goal G *rewrites strictly* to G', and write $G \overset{*!}{\rightsquigarrow} G'$, when there exists a sequence of rewriting steps $G \rightsquigarrow G_1 \rightsquigarrow G_2 \rightsquigarrow \cdots \rightsquigarrow G'$ where all the occurrences of the '$\fatsemi$' step are instances of the following stricter decomposition of actions:

$\boxed{\overset{\fatsemi!}{\rightsquigarrow}}$ $\{(\mathfrak{a}_1 \fatsemi \mathfrak{a}_2)(t_1, t_2)\} \cup G \overset{\mathfrak{i}}{\rightsquigarrow} \{\mathfrak{a}_1(t_1, t), \mathfrak{a}_2(t, t_2)\} \cup G$, for any *normal* Σ-term t.

That is, $\overset{*!}{\rightsquigarrow} = \bigcup\{\overset{n!}{\rightsquigarrow} \mid n \in \mathbb{N}\}$, where:

1. $G \overset{0!}{\rightsquigarrow} G$, and
2. if $G_1 \overset{lb}{\rightsquigarrow} G_2$ and $G_2 \overset{n!}{\rightsquigarrow} G_3$, then $G_1 \overset{n+1!}{\rightsquigarrow} G_3$, for each label $lb \in \{\mathtt{EQ}, =, \mathtt{TR}, \fatsemi!, \cup, \circlearrowleft\}$.

Completeness of operational semantics - narrowing

Proposition

Let (Σ, Γ) *be a canonical and coherent term rewriting system, and let* $\mathrm{nf}(t_1)$ *and* $\mathrm{nf}(t_2)$ *be the (unique) normal forms of terms* t_1 *and* t_2*, respectively. For any action* $\mathfrak{a}$*, if* $\{\mathfrak{a}(t_1, t_2)\} \overset{*}{\rightsquigarrow} \emptyset$*, then*

1. $\{\mathfrak{a}(t_1, \mathrm{nf}(t_2))\} \overset{*}{\rightsquigarrow} \emptyset$*, and*
2. $\{\mathfrak{a}(\mathrm{nf}(t_1), t_2)\} \overset{*}{\rightsquigarrow} \emptyset$.

Completeness of operational semantics - narrowing

Lemma (Lifting Lemma)

Let (Σ, Γ) *be a term rewriting system such that:*

1. *for each equational rule* $\forall Y \cdot (\ell = r)$ *if* $H \in \Gamma$, $Y = \mathrm{var}(l)$, *and*
2. *for each transition rule* $\forall Y \cdot \kappa(\ell, r)$ *if* $H \in \Gamma$, $Y = \mathrm{var}(\kappa) \cup \mathrm{var}(\ell)$.

For every query $\exists X \cdot Q$, *normal substitution* $\theta \colon X \to Z$, *and goal* G *over* $\Sigma[Z]$ *such that* $Q\theta \overset{n!}{\rightsquigarrow} G$, *there exist a narrowing derivation* $\exists X \cdot Q \overset{n}{\twoheadrightarrow}_{\psi} \exists X' \cdot Q'$ *and a normal substitution* $\delta \colon X' \to Z$ *such that* $Q'\delta = G$, *and* $\psi \fatsemi \delta = \theta$.

$$\begin{array}{ccc} \exists X \cdot Q & \overset{n}{\dashrightarrow}_{\psi} & \exists X' \cdot Q' \\ {\scriptstyle\theta}\downarrow & & \downarrow{\scriptstyle\delta} \\ Q\theta & \overset{n!}{\rightsquigarrow} & G \end{array}$$

Completeness of operational semantics - narrowing

Theorem

Let (Σ, Γ) *be a canonical and coherent term rewriting system such that:*

1. *for each equational rule* $\forall Y \cdot (\ell = r)$ *if* $H \in \Gamma$, $Y = \mathrm{var}(\ell)$, *and*
2. *for each transition rule* $\forall Y \cdot \kappa(\ell, r)$ *if* $H \in \Gamma$, $Y = \mathrm{var}(\kappa) \cup \mathrm{var}(\ell)$.

For every normal solution $\theta \colon X \to Z$ *to a query* $\exists X \cdot Q$ *there exists a narrowing derivation* $\exists X \cdot Q \overset{*}{\twoheadrightarrow}_{\psi} \exists X' \cdot \emptyset$ *and a substitution* $\delta \colon X' \to Z$ *such that* $\theta = \psi \fatsemi \delta$.

Corollary (Completeness of narrowing)

Under the conditions above,

$$\exists X \cdot Q \overset{*}{\twoheadrightarrow}_{\psi} \exists X' \cdot \emptyset \quad \text{if} \quad \Gamma \models \exists X \cdot Q.$$

TATP

DEMO

`https://github.com/Transition-Algebra/TATP`

TATP - Implementation notes (rewriting)

- We can take advantage of the meta-level capabilities of Maude for designing a resolution procedure for reductions.
- Given a term t, we look for an equation $\ell = r$ if H, such that ℓ matches t, obtaining a substitution θ and a context c.
- Then, a new substitution θ' extending θ is required for satisfying the condition H; θ' will be used for instantiating r, replacing ℓ in the context c.
- The process is repeated until no equations can be applied to the current term.
- The condition H may contain both equalities and transitions.
- For equalities, the current substitution is applied to both sides of the equation and they are reduced independently as described.
- Once they cannot be further reduced, equality modulo axioms is checked.

TATP - Implementation notes (rewriting)

- For transitions, we also apply the current substitution, but the process is slightly different to the one before.
- First, the matching is twofold: we need the term to match the left-hand side and the actions to match the labels.
- Then, the implementation includes built-in transitions implementing the rules for composition, choice, and iteration, which are used when required.
- Finally, the matching with the right-hand side might extend the substitution.

TATP - Implementation notes (rewriting)

- Then, θ' is computed by using backtracking: for each transition in the condition, as many child nodes can be generated as there are possible substitutions.
- It is important to note that, because we work modulo equational axioms such as associativity and commutativity, several different substitutions can be generated when matching a term and a pattern.
- The number of substitutions generated by a transition might be infinite.
- For this reason, the traversal of the tree follows a depth-first strategy and each substitution is computed lazily, only if required.
- Also note that equalities must be recomputed in general when the substitution changes.

TATP - Implementation notes (rewriting)

- Termination is ensured by supporting two bounds: in the number of steps (including both reductions and transitions) and in the number of conditions.
- The bounds are considered differently.
- The number of steps is computed globally, so each application of an equation or a transition reduces the bound
- The number of conditions is used locally for each reduction step and restarted again for the next one.
- Our implementation stores the terms already tried to reduce in the conditions, avoiding loops that often occur when dealing with some operators, in particular iteration.

TATP - Implementation notes (narrowing)

- For solving queries, we adapt a general logic-programming procedure that is already integrated into SpeX and matches closely our definition of narrowing.
- This procedure relies on the Maude strategy language and involves the rewriting of three kinds of 'configurations': *initial*, *open*, and *solved*.
- An initial configuration is a pair consisting of a specification (set of Horn clauses) and a query that is meant to be checked.
- Optionally, for `echeck`, an initial configuration may also provide a list of (sets of) clauses (identified by labels) to be used in subsequent narrowing steps.

TATP - Implementation notes (narrowing)

- An open configuration includes, besides the specification and a query, information about the partial solution computed up to that point.
- This means that, for any chain of narrowing steps $\exists X_0 \cdot Q_0 \twoheadrightarrow_{\psi_1} \exists X_1 \cdot \emptyset_1 \twoheadrightarrow_{\psi_2} \exists X_2 \cdot \emptyset_2 \twoheadrightarrow_{\psi_3} \cdots$ we have a corresponding chain of configuration rewrites such that, for each index n, the nth open configuration includes an encoding of the substitution $\psi_1 \fatsemi \psi_2 \fatsemi \cdots \fatsemi \psi_n$.
- Open configurations may also carry lists of (sets of) clauses for `echeck` and information about the variables in use.
- The latter information is passed to built-in Maude functions for computing unifiers.
- Finally, a solved configuration is a configuration whose query is $\top$.

TATP - Implementation notes (narrowing)

- The rewriting of configurations is controlled using Maude strategies. Every change of configurations consists of several sub-steps:
  1. first, we non-deterministically select a context or formula within the current query where narrowing is applied;
  2. then we select (also non-deterministically) a suitable clause,
  3. we compute all possible unifiers (which may not be unique if the computations are performed modulo associativity and commutativity),
  4. apply substitutions,
  5. compile new partial solutions, and
  6. update the information on the variable names that are currently in use and on the list of (sets of) clauses meant to guide the following narrowing steps.

TATP - Implementation notes (narrowing)

- Similarly to term rewriting, termination can be enforced by specifying a bound on the number of narrowing steps that can be applied.
- Still, the search space for solved configurations can easily become infeasibly large even for low bounds.
- Because of this, the explicit search (implemented for `echeck`) is often more convenient as it drastically limits the extent of the search space by specifying which clauses are admissible at each step.
- Its downside is that the search procedure is no longer fully automatic, and completeness cannot be guaranteed.
- To counterbalance this limitation, if no solved configuration is found (given a sequence `HCL` of Horn-clause labels), then the explicit search returns a final list of open configurations that may be rewritten to a solved one (by modifying or extending `HCL`).

Conclusions

- We have examined Transition Algebra, an extension of many-sorted first-order logic that includes features from dynamic logic, where actions are used as labels of transitions, thus allowing us to capture and analyse complex behaviours.
- This logic is useful for modelling and studying systems where the actions are as important as the system states upon which they produce an effect.
- Unlike previous definitions, and for greater flexibility in the use of actions, we have extended (and simplified) the framework in order to support arbitrary terms as actions.
- We have devised a sound and complete set of proof rules for reasoning about Horn clauses.
- We have developed an operational semantics for Transition Algebra based on term rewriting, transition checking, and narrowing.
- We have shown that rewriting and narrowing are both sound and complete, thus advocating for Transition Algebra as an appropriate framework for system verification.

Conclusions

- The features above provide advantages from both theoretical and practical points of view.
- On the one hand, systems can be more easily and accurately described than with conventional equational logics.
- Furthermore, Transition Algebra allows for integrating specification and verification organically within a single framework, which is useful especially when considering properties that cannot be easily expressed equationally.
- On the other hand, its simple but powerful operational semantics is very well suited for axiomatization.
- We have implemented a Maude prototype that supports specifications in Transition Algebra and solving queries on them.

Ongoing work

- On the theoretical side, we want to explore additional action operators, such as tests and parallel composition.
- We aim to continue the study of narrowing.
- Though sound and complete, in its current form, narrowing is still rather inefficient from a computational perspective.
- We intend to look further into different forms of narrowing, as can be found in the equational-logic literature.
- Moreover, we aim to explore the use of explicit narrowing as a useful tool in interactive theorem proving.
- On the practical side, we are interested in continuing the development of the tool.
- We also plan to specify and analyse a variety of systems to asses the approach and to design strategies to ease the verification process.

Model-theoretic forcing in transition algebra

Go Hashimoto and Daniel Găină

Kyushu University

- **Expressive power of TA (Categoricity)**

$$\mathfrak{B} \models T_{\mathfrak{A}} \iff \mathfrak{B} \simeq \mathfrak{A}$$

- **Forcing method**
- **Omitting Types Theorem (OTT) and its applications**

Cardinality assumptions

In general, results on infinitary logics (α-compactness, etc.) depend on cardinals. So to simplify the situation, we add some assumptions about cardinals.

Assumption 1

Note that there are two kinds of infinite cardinals:"successor cardinal α^+", and "limit cardinal".

1. *General Continuum Hypothesis. (GCH)*

$$\alpha = 2^{\beta} \quad \text{for all infinite successor cardinal } \alpha = \beta^+$$

2. *Non-existence of inaccessible cardinals. ($\neg IC$)*

$$\neg IC \Leftrightarrow cf(\alpha) < \alpha \text{ for all uncountable limit cardinals } \alpha$$

where $cf(\alpha) = \min\{\beta \in On \mid \exists_{\gamma:\beta\to\alpha}\, \alpha = \cup_{i\in\beta}\gamma_i\}$.

These are useful for induction on infinite cardinals. This is because they allow us to rewrite any uncountable cardinal using smaller cardinals.

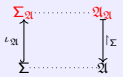
Categoricity

- A set of sentences is called a categorical theory if it has a unique model up to isomorphism.
- Let $\mathrm{Th}(\mathfrak{A}) := \{\phi \in \mathrm{Sen}(\Sigma) \mid \mathfrak{A} \models \phi\}$ for a Σ-model $\mathfrak{A}$.
  If $\mathrm{Th}(\mathfrak{A})$ is categorical, it means that $\mathfrak{A}$ can be represented by sentences.
- In general this is not true, but the following theorem says that it is possible if we allow signature expansions.

Theorem 1 (Categoricity)

Let $\mathfrak{A}$ be a Σ-model. There exist:

1. $\iota_{\mathfrak{A}} : \Sigma \hookrightarrow \Sigma_{\mathfrak{A}}$ *– sort-preserving inclusion,*
2. $\mathfrak{A}_{\mathfrak{A}}$ *– reachable*[1] *$\iota_{\mathfrak{A}}$-expansion of $\mathfrak{A}$,*



such that $\mathrm{Th}(\mathfrak{A}_{\mathfrak{A}})$ is categorical (i.e. any $\Sigma_{\mathfrak{A}}$-model of $\mathrm{Th}(\mathfrak{A}_{\mathfrak{A}})$ is isomorphic to $\mathfrak{A}_{\mathfrak{A}}$).

[1] That is, all elements of $\mathfrak{A}_{\mathfrak{A}}$ can be expressed using the symbols in $\Sigma_{\mathfrak{A}}$

Proof.(sketch) For simplicity, let α be cardinal, $\Sigma = (\{s\}, \alpha, <)$, and $\mathfrak{A} = (\alpha, <_\alpha)$.

- Intuitively, we can do this by adding all the elements, functions, and relations on $\mathfrak{A}$ as symbols to Σ. [2]
- Generally, when two reachable models satisfy the same atomic sentences, they are isomorphic. Since $\mathrm{Th}(\mathfrak{A}_{\mathfrak{A}})$ includes atomic sentences, It is sufficient that:

$$\mathfrak{B} \models \mathrm{Th}(\mathfrak{A}_{\mathfrak{A}}) \text{ implies "}\mathfrak{B} \text{ is reachable" for all model } \mathfrak{B}.$$

[$\alpha < \omega$] Let $\mathfrak{A}_{\mathfrak{A}} := \mathfrak{A}$. $\mathfrak{B} \models \mathrm{Th}(\mathfrak{A}_{\mathfrak{A}}) \ni \neg\exists x \cdot \wedge \{x \neq a \mid a \in \mathfrak{A}\}$ implies "$\mathfrak{B}$ is reachable".

[$\alpha = \omega$] Here we use "$*$". We use the successor relation $suc^{\mathfrak{A}_{\mathfrak{A}}} := \{(i, i+1) \mid i \in \omega\}$.
$\mathfrak{B} \models \mathrm{Th}(\mathfrak{A}_{\mathfrak{A}}) \ni \forall x \cdot\ 0 =(suc^*)\Rightarrow x$ implies "$\mathfrak{B}$ is reachable".

[$\alpha > \omega$ is successor cardinal] From (GCH), we can write $\alpha = 2^\beta$.
By the induction hypothesis, β can already be expressed. Therefore, we can limit the number of elements by using the function $\cdot(\cdot) : \alpha \times \beta \simeq 2^\beta \times \beta \ni (f, x) \mapsto f(x) \in 2$.
$\mathfrak{B} \models \mathrm{Th}(\mathfrak{A}_{\mathfrak{A}}) \ni \forall f, g < \alpha\, (\forall x < \beta\, f(x) = g(x)) \to f = g$ implies "$\mathfrak{B}$ is reachable".

[$\alpha > \omega$ is limit cardinal] From $(\neg IC)$ we can wright $\alpha = \cup_{i\in\beta}\gamma_i\, (\beta < \alpha, \gamma : \beta \to \alpha)$.
Since α is a limit of smaller cardinals, which can be expressed by the induction hypothesis, we can do this by using the symbol corresponding to γ.
$\mathfrak{B} \models \mathrm{Th}(\mathfrak{A}_{\mathfrak{A}}) \ni \forall n < \alpha\, \exists i < \beta \cdot n \leq \gamma_i$ implies "$\mathfrak{B}$ is reachable". □

[2] In reality, we don't need that many symbols; it is enough to add as many symbols as the cardinality of $\mathfrak{A}$.

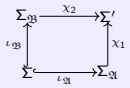
Applications of categoricity

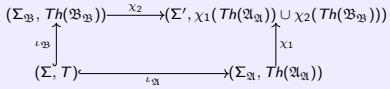
Corollary 2

In TA, (α-)Compactness, and (α-)Upward Löwenheim-Skolem Property fails.

Corollary 3 (Robinson Consistency)

Let (Σ, T) be a maximally consistent theory with at least two non-isomorphic[3] *models $\mathfrak{A}$ and $\mathfrak{B}$. Construct a pushout as illustrated on the left side of the diagram below.*





Then $\chi_1(Th(\mathfrak{A}_{\mathfrak{A}})) \cup \chi_2(Th(\mathfrak{B}_{\mathfrak{B}}))$ is not consistent.

- Since $\mathfrak{A} \models T$ and $\mathfrak{A}_{\mathfrak{A}} \restriction_\Sigma = \mathfrak{A}$, we have $T \subseteq Th(\mathfrak{A}_{\mathfrak{A}})$.
- Since $\mathfrak{B} \models T$ and $\mathfrak{B}_{\mathfrak{B}} \restriction_\Sigma = \mathfrak{B}$, we have $T \subseteq Th(\mathfrak{B}_{\mathfrak{B}})$.
- However, $\chi_1(Th(\mathfrak{A}_{\mathfrak{A}})) \cup \chi_2(Th(\mathfrak{B}_{\mathfrak{B}}))$ is not consistent, because any model of $\chi_1(Th(\mathfrak{A}_{\mathfrak{A}})) \cup \chi_2(Th(\mathfrak{B}_{\mathfrak{B}}))$, by Theorem 1, would imply that $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$.

[3] For example, if Σ has only one label $\leq$ and "$\leq$ is order" $\in T$, then such models exist because the effect of "$*$" disappears.

In the case of ZFC

Categoricity and its results use GCH and $\neg IC$ in addition to ZFC. However, the story also has applications when using only ZFC.

Fact 4

$ZFC + GCH + \neg IC \nvdash \bot \iff ZFC \nvdash \bot$

Proposition 5

Let ZFC be consistent ($ZFC \nvdash \bot$). $ZFC + GCH + \neg IC \vdash \varphi \implies ZFC \nvdash \neg\varphi$

Proof. Suppose $ZFC \vdash \neg\varphi$, towards a contradiction.

- Since $ZFC \subseteq ZFC + GCH + \neg IC$, $ZFC + GCH + \neg IC \vdash \neg\varphi$.
- From the assumption, $ZFC + GCH + \neg IC \vdash \bot$.
- By the fact $ZFC \vdash \bot$.

But this contradicts $ZFC \nvdash \bot$. □

Therefore, the results shown so far cannot be disproven from ZFC as long as ZFC is consistent (i.e., as long as modern mathematics is not broken).

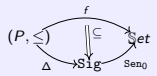
From here on, GCH and $\neg IC$ will not be used.

- **Expressive power of TA (Categoricity)**
  - $T \Leftarrow$representation$= \mathfrak{A}$
  - Negative results.
- **Forcing method**
  - $T =$implementation$\Rightarrow \mathfrak{A}$
  - Positive results.
- **Omitting Types Theorem (OTT) and its applications**

Forcing property

Definition 6

A forcing property is a tuple $\mathbb{P} = (P, \leq, \Delta, f)$, where:



1. $(P, \leq)$ is a partially ordered set with a least element 0.
2. $\Delta : (P, \leq) \to \mathtt{Sig}$ is a functor, which maps each $(p \leq q) \in (P, \leq)$ to $\Delta(p) \subseteq \Delta(q)$.
3. $f : (P, \leq) \to \mathbb{S}et$ is a functor such that $\subseteq : f \implies \Delta; \mathtt{Sen}_0$ is a natural transformation.
4. If $f(p) \models \varphi$ then $\varphi \in f(q)$ for some $q \geq p$, for all atomic sentences $\varphi \in \mathtt{Sen}_0(\Delta(p))$.

Forcing relation

Definition 7 (Forcing relation)

The forcing relation $\Vdash$ between conditions and sentences is defined by:

- $p \Vdash \varphi$ if $\varphi \in f(p)$, for all atomic sentences $\varphi \in \mathrm{Sen}_0(\Delta(p))$.
- $p \Vdash (\mathfrak{a}_1 \mathbin{;} \mathfrak{a}_2)(t_1, t_2)$ if $p \Vdash \mathfrak{a}_1(t_1, t)$ and $p \Vdash \mathfrak{a}_2(t, t_2)$ for some term $t \in T_{\Delta(p)}$.
- $p \Vdash (\mathfrak{a}_1 \cup \mathfrak{a}_2)(t_1, t_2)$ if $p \Vdash \mathfrak{a}_1(t_1, t_2)$ or $p \Vdash \mathfrak{a}_2(t_1, t_2)$.
- $p \Vdash \mathfrak{a}^*(t_1, t_2)$ if $p \Vdash \mathfrak{a}^n(t_1, t_2)$ for some natural number $n < \omega$.
- $p \Vdash \neg\phi$ if there is no $q \geq p$ such that $q \Vdash \phi$.
- $p \Vdash \vee\Phi$ if $p \Vdash \phi$ for some $\phi \in \Phi$.
- $p \Vdash \exists x \cdot \phi$ if $p \Vdash \phi[x \leftarrow t]$ for some ground term t.

Generic set and generic model

Definition 8 (Generic set)

A subset of conditions $G \subseteq P$ is generic if

1. G is an ideal, i.e.,
   - $G \neq \emptyset$,
   - for all $p \in G$ and all $q \leq p$ we have $q \in G$, and
   - for all $p, q \in G$ there exists $r \in G$ such that $p \leq r$ and $q \leq r$ (directedness);
2. G determines the truth of sentences, i.e.,
   - for all conditions $p \in G$ and all sentences $\phi \in \mathrm{Sen}(\Delta(p))$ there exists a condition $q \in G$ such that $q \geq p$ and either $q \Vdash \phi$ or $q \Vdash \neg\phi$ holds.

Theorem 9 (Generic model theorem)

Let Δ_G be signature of G, i.e., the "union" of all signatures of the conditions in a generic set G
There is a reachable Δ_G-model $\mathfrak{A}_G$ which is generic for G i.e.,

$$\mathfrak{A}_G \models \phi \iff r \Vdash \phi \text{ for some } r \in G$$

for all Δ_G-sentences ϕ.

Semantic forcing

Example 10

- Depending on the logical property being studied, a condition may take on different forms.
- For OTT and DLS, a condition is a triple $(\Sigma[C], \Phi, \mathcal{M})$: [a]
  - Φ - set of $\Sigma[C]$-sentences
  - $\mathcal{M}$ - class of $\Sigma[C]$-models

[a] Strictly speaking, this is not a tuple, since $\mathcal{M}$ can be a proper class.

- Syntactic forcing uses $p = (\Sigma_p, \Phi_p)$.

$$p \Vdash \neg\neg\phi \iff \Phi_p \vdash_{\Sigma_p} \phi$$

- Semantic forcing uses $p = (\Sigma_p, \Phi_p, \mathcal{M}_p)$.

$$p \Vdash \neg\neg\phi \iff \Phi_p \models_{\Sigma_p, \mathcal{M}_p} \phi$$

where $\Phi \models_{\Sigma, \mathcal{M}} \phi :\iff \mathfrak{A} \models \Phi$ implies $\mathfrak{A} \models \phi$ for all $\mathfrak{A} \in \mathcal{M}$.
Note: $\Phi \models_{\Sigma} \phi :\iff \mathfrak{A} \models \Phi$ implies $\mathfrak{A} \models \phi$ for all $\mathfrak{A}$.

- **Expressive power of TA (Categoricity)**
- **Forcing method**
- **Omitting Types Theorem (OTT) and its applications**

$$(\mathfrak{A} \models \Phi) + \begin{cases} \mathfrak{A} \models T(x) \text{ for some } x \in |\mathfrak{A}| & (\textit{realize}) \\ \mathfrak{A} \not\models T(x) \text{ for each } x \in |\mathfrak{A}| & (\textit{omit}) \quad \leftarrow \text{OTT} \end{cases}$$

Examples

Example 11 (1-type)

- $\Sigma = (\{Nat\}, \{0 :\rightarrow Nat, s : Nat \rightarrow Nat, \_ + \_ : Nat\ Nat \rightarrow Nat\})$
- $\Phi = \{\forall x \cdot x + 0 = 0, \forall x, y \cdot x + s(y) = s(x + y)\}$
- $T(x) = \{x \neq 0, x \neq s(0), x \neq s^2(0), \dots\}$
- If $\mathfrak{A}$ omits [a] $T \iff \mathfrak{A}$ is reachable by $0 :\rightarrow Nat$ and $s : Nat \rightarrow Nat$
- Lemma: Φ locally omits [b] T

[a] $\mathfrak{A}$ omits $T :\iff$ there is no element (solution) $x \in |\mathfrak{A}|$ that satisfies $T(x)$
[b] Φ locally omits $T :\iff \Phi \cup \Gamma \not\vdash_{\Sigma[x]} T$ for all $\Gamma \in \mathcal{P}(\mathrm{Sen}(\Sigma[x]))$, such that $\mathrm{card}(\Gamma) < \mathrm{card}(\mathrm{Sen}(\Sigma[x]))$.

Example 12 (ω-type)

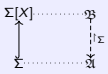
- $\Sigma = (\{s\}, \{r :\rightarrow s \mid r \in \mathbb{R}\})$
- $X = \{x_i \mid i \in \mathbb{N}\}$ a set of variables for Σ (Note that $\mathbb{R}^{\mathbb{N}} \approx 2^{\mathbb{N} \times \mathbb{N}} \approx 2^{\mathbb{N}} \approx \mathbb{R}$)
- $T(X) = \cup_{i \in \mathbb{N}} T(x_i) \cup \{x_i \neq x_j$ [a] $\mid i \neq j\ i, j \in \mathbb{N}\}$ where $T(x) = \{x \neq r \mid r :\rightarrow s \in F\}$:
- $\mathfrak{A}$ omits $T(X) \iff$ There are at most finite solutions to $T(x)$ (though they may exist).
- Lemma: Any countable and satisfiable set of sentences Φ locally omits $T(X)$.

[a] If we replace "$\neq$" with another relation, we can allow finite chains of solutions while eliminating infinite chains.

Omitting Types Theorem

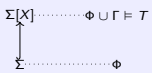
Definition 13 (Type)

- $T \subseteq \mathrm{Sen}(\Sigma[X])$ is a type, where $\alpha = \mathrm{card}(\mathrm{Sen}(\Sigma))$ and $\alpha^{\mathrm{card}(X)} \leq \alpha$.
- A Σ-model $\mathfrak{A}$ realizes T if $\mathfrak{B} \models T$ for some expansion $\mathfrak{B}$ of $\mathfrak{A}$ to $\Sigma[X]$.
- $\mathfrak{A}$ *omits* T if $\mathfrak{A}$ does not realize T.



Definition 14 (Isolated type)

- Let Σ be a signature of power $\mathrm{card}(\mathrm{Sen}(\Sigma)) = \alpha$.
- $\Phi \subseteq \mathrm{Sen}(\Sigma)$ isolates $T \subseteq \mathrm{Sen}(\Sigma[X])$ iff $\Phi \cup \Gamma \vDash_{\Sigma[X]} T$ for some $\Gamma \in \mathcal{P}_\alpha(\mathrm{Sen}(\Sigma[X]))$.
- Φ *locally omits* T if Φ does not isolate T.



Theorem 15 (Omitting Types Theorem)

- Σ - *signature of power* α.
- $\mathcal{L}$ *is a (syntactic) fragment of* TA.
- *If* $\alpha > \omega$ *then we assume that fragment* $\mathcal{L}$ *is compact.*

If Φ has a model, and Φ locally omits $T_i \subseteq \mathrm{Sen}(\Sigma[X_i])$ for all $i < \alpha$, then Φ has a model which omits T_i for all $i < \alpha$.

Proof

- The proof of OTT is based on forcing.
- The key is to construct a generic set G that forces all negations of formulas in the type T.

$\Sigma \setminus T$	$\phi_0(x)$	$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	$\phi_4(x)$	$\phi_5(x)$	$\cdots$	add counterexamples
c_0	true	true	true	true	not yet	true	$\cdots$	$\Phi_0 \leftarrow \neg\phi_4(c_0)$
c_1	true	not yet	false	true	false	true	$\cdots$	$\Phi_1 \leftarrow \neg\phi_1(c_1)$
c_2	false	false	false	true	true	true	$\cdots$	$\Phi_2 \leftarrow \neg\phi_0(c_2)$
c_3	true	true	true	not yet	false	true	$\cdots$	$\Phi_3 \leftarrow \neg\phi_3(c_3)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$

- Then Generic Model Theorem delivers a model $\mathfrak{A}$ such that

$$\mathfrak{A} \models \phi \iff G \Vdash \phi, \text{ for all sentences } \phi.$$

Hence, $\mathfrak{A}$ omits T.

Below is an example of a type that cannot be omitted if the Henkin constants are added all at once. Note that the Kleene star "$*$" is not used in this example. In the case of the proof of OTT, the classical method would run into problems if there were just an infinite number of sorts.

Example 16 (0-type)

- $\Sigma = (\{s_n \mid n \in \mathbb{N}\}, F, P)$
- $T = \{\exists x_0 : s_0, \ldots, x_n : s_0 \cdot \wedge_{i \neq j} x_i \neq x_j \mid n \in \mathbb{N}\}$: there infinitely many elements of sort s_0
- $\mathfrak{A}$ omits $T \iff \mathfrak{A}_{s_0}$ is finite
- $\phi_n = \exists z_n : s_n \cdot \top \implies \exists x_1 : s_0, \ldots, x_n : s_0 \cdot \wedge_{i \neq j} x_i \neq x_j$ for all $n > 0$: if there exists an element of sort s_n then there exist at least n elements of sort s_0
- Lemma: $\Phi = \{\phi_n \mid n > 0\}$ locally omits T.

Applications of OTT

- OTT + compactness $\implies$ Löwenheim-Skolem Properties
- OTT + compactness $\implies$ Joint Robinson Consistency $\overset{\text{negation}}{\Longleftrightarrow}$ Interpolation
- OTT for uncountable languages of regular cardinality $\implies$ inf-compactness [4]
- Completeness of $\mathcal{L} \overset{OTT}{\Longrightarrow}$ completeness of the fragment obtained from $\mathcal{L}$ by restricting the semantics to models which omit a certain type.
  - type = "infinity" $\implies$ completeness of finite model theory
  - type is based on constructors $\implies$ completeness of constructor-based logics

[4] Each set of sentences Φ has an infinite model whenever each finite subset $\Phi_{fin} \subseteq \Phi$ has an infinite model.

Example 17 (Natural numbers)

(Σ, Φ):

- $\Sigma = (\{Nat\}, \{0 :\rightarrow Nat, s : Nat \rightarrow Nat, \_ + \_ : Nat\ Nat \rightarrow Nat\})$
- $F^c = \{0 :\rightarrow Nat, s : Nat \rightarrow Nat\}$ – constructors
- Nat is a sort interpreted as finite
- $\Phi = \{\forall x \cdot x + 0 = 0, \forall x, y \cdot x + s(y) = s(x + y)\}$

$$(CB)\ \frac{\Phi \vdash \psi[x \leftarrow s^n(0)] \text{ for all } n \in \mathbb{N}}{\Phi \vdash \forall x \cdot \psi} \qquad (FN)\ \frac{\Phi \cup \{\forall x_0, \ldots, x_n \cdot \vee_{i \neq j} x_i = x_j\} \vdash \psi \text{ for all } n \in \mathbb{N}}{\Phi \vdash \psi}$$

Completeness of TA

$\Phi \nvdash \bot \implies$ there is a model $\mathfrak{A}$ s.t. $\mathfrak{A} \models \Phi$

Completeness of TA + CB + FN

$\Phi \nvdash_{CB,FN} \bot \implies$ there is a model $\mathfrak{A}$ s.t. $\mathfrak{A} \models \Phi$ and omits infiniteness and inreachability.

Example 18 (Lists)

(Σ, Φ):

1. $\Sigma = (\{Elt, List\}, \{empty :\rightarrow List, \_;\_ : List\ Elt \rightarrow List, add : List\ List \rightarrow List\})$
2. $F^c = \{empty :\rightarrow List, \_;\_ : List\ Elt \rightarrow List\}$ – constructors
3. $\Phi = \{\forall x \cdot add(x, empty) = x, \forall x, y, e \cdot add(x, y; e) = add(x, y); e\}$

$$(CB)\ \frac{\Phi \vdash \forall e_1, \ldots, e_n \cdot \psi[x \leftarrow e_1; \ldots; e_n; empty] \text{ for all } n \in \mathbb{N}}{\Phi \vdash \forall x \cdot \psi}$$

Conclusion and current work

- **Expressive power of TA (Categoricity)**
  - TA is strong enough to uniquely determine models, by using sentences.
  - Categoricity can be used to show negative results.
- **Forcing method**
  - Forcing makes it possible to construct models even for infinitary logic.
  - Forcing can be used to show positive results.
- **Omitting Types Theorem (OTT) and its applications**
  - Omitting Types is a way to avoid adding unnecessary elements to models.
  - OTT has applications such as proving the completeness of constructor-based proofs.

Monoidal categories + TA = MTA (Monoidal Transition Algebra)

1. Monoidal categories can be used to discuss quantum systems.

   Quantum system + Transition = Dynamic quantum system [5]

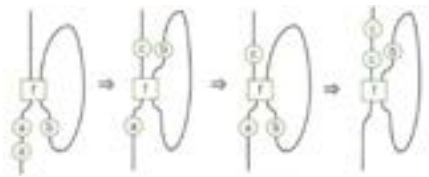
2. Since second-order logical expressions are possible, differential equations can be handled grammatically.

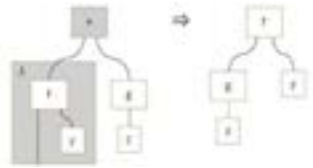
   Differential equation + Transition = Hybrid system

Since monoidal categories generate new objects (sorts) through tensor products, the forcing approach that adapts to the infinite number of sorts is effective.

$$\{s,\ s \otimes s,\ s \otimes s \otimes s,\ s \otimes s \otimes s \otimes s,\ s \otimes s \otimes s \otimes s \otimes s,\ \dots\}$$

[5] e.g., a system that changes its next action depending on the observation result





$$(\lambda x.f(x,y))g(z) =(\beta)\Rightarrow f(g(z),y)$$

MI レクチャーノートシリーズ刊行にあたり

本レクチャーノートシリーズは、文部科学省 21 世紀 COE プログラム「機能数理学の構築と展開」（H15-19 年度）において作成した COE Lecture Notes の続刊であり、文部科学省大学院教育改革支援プログラム「産業界が求める数学博士と新修士養成」（H19-21 年度）および、同グローバル COE プログラム「マス・フォア・インダストリ教育研究拠点」（H20-24 年度）において行われた講義の講義録として出版されてきた。平成 23 年 4 月のマス・フォア・インダストリ研究所（IMI）設立と平成 25 年 4 月の IMI の文部科学省共同利用・共同研究拠点として「産業数学の先進的･基礎的共同研究拠点」の認定を受け、今後、レクチャーノートは、マス・フォア・インダストリに関わる国内外の研究者による講義の講義録、会議録等として出版し、マス・フォア・インダストリの本格的な展開に資するものとする。

2022 年 10 月
マス・フォア・インダストリ研究所
所長　梶原 健司

IMI Workshop of the Joint Usage Research Projects

Workshop on Logic, Algebra and Category Theory: Theory and Applications

発　行　2026年 2月10日
編　集　Guillermo Badia, Daniel Găină and Tomasz Kowalski

発　行　九州大学マス・フォア・インダストリ研究所
九州大学大学院数理学府
〒819-0395 福岡市西区元岡744
九州大学数理・IMI 事務室
TEL 092-802-4402　FAX 092-802-4405
URL https://www.imi.kyushu-u.ac.jp/

印　刷　城島印刷株式会社
〒810-0012 福岡市中央区白金 2 丁目 9 番 6 号
TEL 092-531-7102　FAX 092-524-4411

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note	Mitsuhiro T. NAKAO Kazuhiro YOKOYAMA	Computer Assisted Proofs - Numeric and Symbolic Approaches - 199pages	August 22, 2006
COE Lecture Note	M.J.Shai HARAN	Arithmetical Investigations - Representation theory, Orthogonal polynomials and Quantum interpolations- 174pages	August 22, 2006
COE Lecture Note Vol.3	Michal BENES Masato KIMURA Tatsuyuki NAKAKI	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2005 155pages	October 13, 2006
COE Lecture Note Vol.4	宮田　健治	辺要素有限要素法による磁界解析 - 機能数理学特別講義 21pages	May 15, 2007
COE Lecture Note Vol.5	Francois APERY	Univariate Elimination Subresultants - Bezout formula, Laurent series and vanishing conditions - 89pages	September 25, 2007
COE Lecture Note Vol.6	Michal BENES Masato KIMURA Tatsuyuki NAKAKI	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2006 209pages	October 12, 2007
COE Lecture Note Vol.7	若山　正人 中尾　充宏	九州大学産業技術数理研究センター キックオフミーティング 138pages	October 15, 2007
COE Lecture Note Vol.8	Alberto PARMEGGIANI	Introduction to the Spectral Theory of Non-Commutative Harmonic Oscillators 233pages	January 31, 2008
COE Lecture Note Vol.9	Michael I.TRIBELSKY	Introduction to Mathematical modeling 23pages	February 15, 2008
COE Lecture Note Vol.10	Jacques FARAUT	Infinite Dimensional Spherical Analysis 74pages	March 14, 2008
COE Lecture Note Vol.11	Gerrit van DIJK	Gelfand Pairs And Beyond 60pages	August 25, 2008
COE Lecture Note Vol.12	Faculty of Mathematics, Kyushu University	Consortium “MATH for INDUSTRY” First Forum 87pages	September 16, 2008
COE Lecture Note Vol.13	九州大学大学院 数理学研究院	プロシーディング「損保数理に現れる確率モデル」 ― 日新火災・九州大学 共同研究2008年11月 研究会 ― 82pages	February 6, 2009

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.14	Michal Beneš, Tohru Tsujikawa Shigetoshi Yazaki	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2008 77pages	February 12, 2009
COE Lecture Note Vol.15	Faculty of Mathematics, Kyushu University	International Workshop on Verified Computations and Related Topics 129pages	February 23, 2009
COE Lecture Note Vol.16	Alexander Samokhin	Volume Integral Equation Method in Problems of Mathematical Physics 50pages	February 24, 2009
COE Lecture Note Vol.17	矢嶋　　徹 及川　正行 梶原　健司 辻　　英一 福本　康秀	非線形波動の数理と物理 66pages	February 27, 2009
COE Lecture Note Vol.18	Tim Hoffmann	Discrete Differential Geometry of Curves and Surfaces 75pages	April 21, 2009
COE Lecture Note Vol.19	Ichiro Suzuki	The Pattern Formation Problem for Autonomous Mobile Robots —Special Lecture in Functional Mathematics— 23pages	April 30, 2009
COE Lecture Note Vol.20	Yasuhide Fukumoto Yasunori Maekawa	Math-for-Industry Tutorial: Spectral theories of non-Hermitian operators and their application 184pages	June 19, 2009
COE Lecture Note Vol.21	Faculty of Mathematics, Kyushu University	Forum “Math-for-Industry” Casimir Force, Casimir Operators and the Riemann Hypothesis 95pages	November 9, 2009
COE Lecture Note Vol.22	Masakazu Suzuki Hoon Hong Hirokazu Anai Chee Yap Yousuke Sato Hiroshi Yoshida	The Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium on Computer Mathematics Mathematical Aspects of Computer and Information Sciences 436pages	December 14, 2009
COE Lecture Note Vol.23	荒川　恒男 金子　昌信	多重ゼータ値入門 111pages	February 15, 2010
COE Lecture Note Vol.24	Fulton B.Gonzalez	Notes on Integral Geometry and Harmonic Analysis 125pages	March 12, 2010
COE Lecture Note Vol.25	Wayne Rossman	Discrete Constant Mean Curvature Surfaces via Conserved Quantities 130pages	May 31, 2010
COE Lecture Note Vol.26	Mihai Ciucu	Perfect Matchings and Applications 66pages	July 2, 2010

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.27	九州大学大学院 数理学研究院	Forum “Math-for-Industry” and Study Group Workshop Information security, visualization, and inverse problems, on the basis of optimization techniques　100pages	October 21, 2010
COE Lecture Note Vol.28	ANDREAS LANGER	MODULAR FORMS, ELLIPTIC AND MODULAR CURVES LECTURES AT KYUSHU UNIVERSITY 2010　62pages	November 26, 2010
COE Lecture Note Vol.29	木田　雅成 原田　昌晃 横山　俊一	Magma で広がる数学の世界　157pages	December 27, 2010
COE Lecture Note Vol.30	原　　隆 松井　　卓 廣島　文生	Mathematical Quantum Field Theory and Renormalization Theory 201pages	January 31, 2011
COE Lecture Note Vol.31	若山　正人 福本　康秀 高木　　剛 山本　昌宏	Study Group Workshop 2010 Lecture & Report　128pages	February 8, 2011
COE Lecture Note Vol.32	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2011 “TSUNAMI-Mathematical Modelling” Using Mathematics for Natural Disaster Prediction, Recovery and Provision for the Future　90pages	September 30, 2011
COE Lecture Note Vol.33	若山　正人 福本　康秀 高木　　剛 山本　昌宏	Study Group Workshop 2011 Lecture & Report　140pages	October 27, 2011
COE Lecture Note Vol.34	Adrian Muntean Vladimír Chalupecký	Homogenization Method and Multiscale Modeling　72pages	October 28, 2011
COE Lecture Note Vol.35	横山　俊一 夫　　紀恵 林　　卓也	計算機代数システムの進展　210pages	November 30, 2011
COE Lecture Note Vol.36	Michal Beneš Masato Kimura Shigetoshi Yazaki	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2010 107pages	January 27, 2012
COE Lecture Note Vol.37	若山　正人 高木　　剛 Kirill Morozov 平岡　裕章 木村　正人 白井　朋之 西井　龍映 栄　伸一郎 穴井　宏和 福本　康秀	平成23年度 数学・数理科学と諸科学・産業との連携研究ワークショップ　拡がっていく数学　～期待される “見えない力” ～ 154pages	February 20, 2012

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.38	Fumio Hiroshima Itaru Sasaki Herbert Spohn Akito Suzuki	Enhanced Binding in Quantum Field Theory 204pages	March 12, 2012
COE Lecture Note Vol.39	Institute of Mathematics for Industry, Kyushu University	Multiscale Mathematics: Hierarchy of collective phenomena and interrelations between hierarchical structures 180pages	March 13, 2012
COE Lecture Note Vol.40	井ノ口順一 太田　泰広 筧　三郎 梶原　健司 松浦　望	離散可積分系・離散微分幾何チュートリアル2012 152pages	March 15, 2012
COE Lecture Note Vol.41	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2012 “Information Recovery and Discovery” 91pages	October 22, 2012
COE Lecture Note Vol.42	佐伯　修 若山　正人 山本　昌宏	Study Group Workshop 2012 Abstract, Lecture & Report 178pages	November 19, 2012
COE Lecture Note Vol.43	Institute of Mathematics for Industry, Kyushu University	Combinatorics and Numerical Analysis Joint Workshop 103pages	December 27, 2012
COE Lecture Note Vol.44	萩原　学	モダン符号理論からポストモダン符号理論への展望 107pages	January 30, 2013
COE Lecture Note Vol.45	金山　寛	Joint Research Workshop of Institute of Mathematics for Industry (IMI), Kyushu University “Propagation of Ultra-large-scale Computation by the Domain-decomposition-method for Industrial Problems (PUCDIP 2012)” 121pages	February 19, 2013
COE Lecture Note Vol.46	西井　龍映 栄　伸一郎 岡田　勘三 落合　啓之 小磯　深幸 斎藤　新悟 白井　朋之	科学・技術の研究課題への数学アプローチ ―数学モデリングの基礎と展開― 325pages	February 28, 2013
COE Lecture Note Vol.47	SOO TECK LEE	BRANCHING RULES AND BRANCHING ALGEBRAS FOR THE COMPLEX CLASSICAL GROUPS 40pages	March 8, 2013
COE Lecture Note Vol.48	溝口　佳寛 脇　隼人 平坂　貢 谷口　哲至 島袋　修	博多ワークショップ「組み合わせとその応用」 124pages	March 28, 2013

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.49	照井　章 小原　功任 濱田　龍義 横山　俊一 穴井　宏和 横田　博史	マス・フォア・インダストリ研究所　共同利用研究集会 II 数式処理研究と産学連携の新たな発展　137pages	August 9, 2013
MI Lecture Note Vol.50	Ken Anjyo Hiroyuki Ochiai Yoshinori Dobashi Yoshihiro Mizoguchi Shizuo Kaji	Symposium MEIS2013: Mathematical Progress in Expressive Image Synthesis　154pages	October 21, 2013
MI Lecture Note Vol.51	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2013 “The Impact of Applications on Mathematics”　97pages	October 30, 2013
MI Lecture Note Vol.52	佐伯　修 岡田　勘三 髙木　剛 若山　正人 山本　昌宏	Study Group Workshop 2013 Abstract, Lecture & Report　142pages	November 15, 2013
MI Lecture Note Vol.53	四方　義啓 櫻井　幸一 安田　貴徳 Xavier Dahan	平成25年度　九州大学マス・フォア・インダストリ研究所 共同利用研究集会　安全・安心社会基盤構築のための代数構造 ～サイバー社会の信頼性確保のための数理学～　158pages	December 26, 2013
MI Lecture Note Vol.54	Takashi Takiguchi Hiroshi Fujiwara	Inverse problems for practice, the present and the future　93pages	January 30, 2014
MI Lecture Note Vol.55	栄　伸一郎 溝口　佳寛 脇　隼人 渋田　敬史	Study Group Workshop 2013 数学協働プログラム Lecture & Report 98pages	February 10, 2014
MI Lecture Note Vol.56	Yoshihiro Mizoguchi Hayato Waki Takafumi Shibuta Tetsuji Taniguchi Osamu Shimabukuro Makoto Tagami Hirotake Kurihara Shuya Chiba	Hakata Workshop 2014 ~ Discrete Mathematics and its Applications ~　141pages	March 28, 2014
MI Lecture Note Vol.57	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2014: “Applications + Practical Conceptualization + Mathematics = fruitful Innovation”　93pages	October 23, 2014
MI Lecture Note Vol.58	安生健一 落合啓之	Symposium MEIS2014: Mathematical Progress in Expressive Image Synthesis　135pages	November 12, 2014

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.59	西井　龍映 岡田　勘三 梶原　健司 髙木　　剛 若山　正人 脇　　隼人 山本　昌宏	Study Group Workshop 2014 数学協働プログラム Abstract, Lecture & Report　196pages	November 14, 2014
MI Lecture Note Vol.60	西浦　　博	平成26年度九州大学 IMI 共同利用研究・研究集会（I） 感染症数理モデルの実用化と産業及び政策での活用のための新たな展開　120pages	November 28, 2014
MI Lecture Note Vol.61	溝口　佳寛 Jacques Garrigue 萩原　　学 Reynald Affeldt	研究集会 高信頼な理論と実装のための定理証明および定理証明器 Theorem proving and provers for reliable theory and implementations (TPP2014)　138pages	February 26, 2015
MI Lecture Note Vol.62	白井　朋之	Workshop on “β-transformation and related topics”　59pages	March 10, 2015
MI Lecture Note Vol.63	白井　朋之	Workshop on “Probabilistic models with determinantal structure” 107pages	August 20, 2015
MI Lecture Note Vol.64	落合　啓之 土橋　宜典	Symposium MEIS2015: Mathematical Progress in Expressive Image Synthesis　124pages	September 18, 2015
MI Lecture Note Vol.65	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2015 “The Role and Importance of Mathematics in Innovation”　74pages	October 23, 2015
MI Lecture Note Vol.66	岡田　勘三 藤澤　克己 白井　朋之 若山　正人 脇　　隼人 Philip Broadbridge 山本　昌宏	Study Group Workshop 2015 Abstract, Lecture & Report 156pages	November 5, 2015
MI Lecture Note Vol.67	Institute of Mathematics for Industry, Kyushu University	IMI-La Trobe Joint Conference “Mathematics for Materials Science and Processing” 66pages	February 5, 2016
MI Lecture Note Vol.68	古庄　英和 小谷　久寿 新甫　洋史	結び目と Grothendieck-Teichmüller 群 116pages	February 22, 2016
MI Lecture Note Vol.69	土橋　宜典 鍛治　静雄	Symposium MEIS2016: Mathematical Progress in Expressive Image Synthesis　82pages	October 24, 2016
MI Lecture Note Vol.70	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2016 “Agriculture as a metaphor for creativity in all human endeavors” 98pages	November 2, 2016
MI Lecture Note Vol.71	小磯　深幸 二宮　嘉行 山本　昌宏	Study Group Workshop 2016 Abstract, Lecture & Report　143pages	November 21, 2016

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.72	新井　朝雄 小嶋　　泉 廣島　文生	Mathematical quantum field theory and related topics　133pages	January 27, 2017
MI Lecture Note Vol.73	穴田　啓晃 Kirill Morozov 須賀　祐治 奥村　伸也 櫻井　幸一	Secret Sharing for Dependability, Usability and Security of Network Storage and Its Mathematical Modeling　211pages	March 15, 2017
MI Lecture Note Vol.74	QUISPEL, G. Reinout W. BADER, Philipp MCLAREN, David I. TAGAMI, Daisuke	IMI-La Trobe Joint Conference Geometric Numerical Integration and its Applications　71pages	March 31, 2017
MI Lecture Note Vol.75	手塚　　集 田上　大助 山本　昌宏	Study Group Workshop 2017 Abstract, Lecture & Report　118pages	October 20, 2017
MI Lecture Note Vol.76	宇田川誠一	Tzitzéica 方程式の有限間隙解に付随した極小曲面の構成理論 ―Tzitzéica 方程式の楕円関数解を出発点として―　68pages	August 4, 2017
MI Lecture Note Vol.77	松谷　茂樹 佐伯　　修 中川　淳一 田上　大助 上坂　正晃 Pierluigi Cesana 濵田　裕康	平成29年度　九州大学マス・フォア・インダストリ研究所 共同利用研究集会（I） 結晶の界面，転位，構造の数理　148pages	December 20, 2017
MI Lecture Note Vol.78	瀧澤　重志 小林　和博 佐藤憲一郎 斎藤　　努 清水　正明 間瀬　正啓 藤澤　克樹 神山　直之	平成29年度　九州大学マス・フォア・インダストリ研究所 プロジェクト研究　研究集会（I） 防災・避難計画の数理モデルの高度化と社会実装へ向けて 136pages	February 26, 2018
MI Lecture Note Vol.79	神山　直之 畔上　秀幸	平成29年度　AIMaP チュートリアル 最適化理論の基礎と応用　96pages	February 28, 2018
MI Lecture Note Vol.80	Kirill Morozov Hiroaki Anada Yuji Suga	IMI Workshop of the Joint Research Projects Cryptographic Technologies for Securing Network Storage and Their Mathematical Modeling　116pages	March 30, 2018
MI Lecture Note Vol.81	Tsuyoshi Takagi Masato Wakayama Keisuke Tanaka Noboru Kunihiro Kazufumi Kimoto Yasuhiko Ikematsu	IMI Workshop of the Joint Research Projects International Symposium on Mathematics, Quantum Theory, and Cryptography　246pages	September 25, 2019
MI Lecture Note Vol.82	池森　俊文	令和2年度　AIMaP チュートリアル 新型コロナウイルス感染症にかかわる諸問題の数理 145pages	March 22, 2021

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.83	早川健太郎 軸丸　芳揮 横須賀洋平 可香谷　隆 林　　和希 堺　　雄亮	シェル理論・膜理論への微分幾何学からのアプローチと その建築曲面設計への応用　49pages	July 28, 2021
MI Lecture Note Vol.84	Taketoshi Kawabe Yoshihiro Mizoguchi Junichi Kako Masakazu Mukai Yuji Yasui	SICE-JSAE-AIMaP Tutorial Advanced Automotive Control and Mathematics　110pages	December 27, 2021
MI Lecture Note Vol.85	Hiroaki Anada Yasuhiko Ikematsu Koji Nuida Satsuya Ohata Yuntao Wang	IMI Workshop of the Joint Usage Research Projects Exploring Mathematical and Practical Principles of Secure Computation and Secret Sharing　114pages	February 9, 2022
MI Lecture Note Vol.86	濱田　直希 穴井　宏和 梅田　裕平 千葉　一永 佐藤　寛之 能島　裕介 加葉田雄太朗 一木　俊助 早野　健太 佐伯　　修	2020年度採択分　九州大学マス・フォア・インダストリ研究所 共同利用研究集会 進化計算の数理　135pages	February 22, 2022
MI Lecture Note Vol.87	Osamu Saeki, Ho Tu Bao, Shizuo Kaji, Kenji Kajiwara, Nguyen Ha Nam, Ta Hai Tung, Melanie Roberts, Masato Wakayama, Le Minh Ha, Philip Broadbridge	Proceedings of Forum “Math-for-Industry” 2021 -Mathematics for Digital Economy-　122pages	March 28, 2022
MI Lecture Note Vol.88	Daniel PACKWOOD Pierluigi CESANA, Shigenori FUJIKAWA, Yasuhide FUKUMOTO, Petros SOFRONIS, Alex STAYKOV	Perspectives on Artificial Intelligence and Machine Learning in Materials Science, February 4-6, 2022　74pages	November 8, 2022

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.89	松谷　茂樹 落合　啓之 井上　和俊 小磯　深幸 佐伯　　修 白井　朋之 垂水　竜一 内藤　久資 中川　淳一 濵田　裕康 松江　　要 加葉田雄太朗	2022年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 材料科学における幾何と代数 III　356pages	December 7, 2022
MI Lecture Note Vol.90	中山　尚子 谷川　拓司 品野　勇治 近藤　正章 石原　　亨 鍛冶　静雄 藤澤　克樹	2022年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 データ格付けサービス実現のための数理基盤の構築　58pages	December 12, 2022
MI Lecture Note Vol.91	Katsuki Fujisawa Shizuo Kaji Toru Ishihara Masaaki Kondo Yuji Shinano Takuji Tanigawa Naoko Nakayama	IMI Workshop of the Joint Usage Research Projects Construction of Mathematical Basis for Realizing Data Rating Service 610pages	December 27, 2022
MI Lecture Note Vol.92	丹田　　聡 三宮　　俊 廣島　文生	2022年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 時間・量子測定・準古典近似の理論と実験 ～古典論と量子論の境界～　150pages	Janualy 6, 2023
MI Lecture Note Vol.93	Philip Broadbridge Luke Bennetts Melanie Roberts Kenji Kajiwara	Proceedings of Forum “Math-for-Industry” 2022 -Mathematics of Public Health and Sustainability-　170pages	June 19, 2023
MI Lecture Note Vol.94	國廣　　昇 池松　泰彦 伊豆　哲也 穴田　啓晃 縫田　光司	2023年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 現代暗号に対する安全性解析・攻撃の数理　260pages	Janualy 11, 2024
MI Lecture Note Vol.96	澤田　茉伊	2023年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 デジタル化時代に求められる斜面防災の思考法　70pages	March 18, 2024

シリーズ既刊

Issue	Author／Editor	Title	Published
MI Lecture Note Vol.97	Shariffah Suhaila Syed Jamaludin Zaiton Mat Isa Nur Arina Bazilah Aziz Taufiq Khairi Ahmad Khairuddin Shaymaa M.H.Darwish Ahmad Razin Zainal Abidin Norhaiza Ahmad Zainal Abdul Aziz Hang See Pheng Mohd Ali Khameini Ahmad	International Project Research-Workshop (I) Proceedings of 4th Malaysia Mathematics in Industry Study Group (MMISG2023)　172pages	March 28, 2024
MI Lecture Note Vol.98	中澤　　嵩	2024 年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 自動車性能の飛躍的向上を目指す Data-Driven 設計　92pages	January 30, 2025
MI Lecture Note Vol.99	Jacques Garrigue	2024 年度採択分 九州大学マス・フォア・インダストリ研究所 共同利用研究集会 コンピュータによる定理証明支援とその応用　308pages	March 17, 2025
MI Lecture Note Vol.100	Yutaka Jitsumatsu Masayoshi Ohashi Akio Hasegawa Katsutoshi Shinohara Shintaro Mori	IMI Workshop of the Joint Usage Research Projects Mathematics for Innovation in Information and Communication Technology 274pages	March 19, 2025
MI Lecture Note Vol.101	Makoto Ohsaki Yoshiki Jikumaru	IMI Workshop of the Joint Usage Research Projects Evolving Design and Discrete Differential Geometry:towards Mathematics Aided Geometric Design 528pages	October 1st, 2025
MI Lecture Note Vol.102	Keunsu Kim	Young Researchers and Students-Workshop (I) Topological Data Analysis and Industrial Mathematics 198 pages	December 22, 2025
MI Lecture Note Vol.103	Kulbir Ghuman Pierluigi Cesana, Kenji Kajiwara, Yu Kaneko Linh Thi Hoai Nguyen Daniel Packwood, Yasser Salah Eddine Bouchareb	International Project Research-Workshop (I) Advancing Materials Data, Design and Discovery 102 pages	December 26, 2025
MI Lecture Note Vol.104	Soon-Sun Kwon Minjung Gim Jae-Hun Jung	International Project Research-Workshop (I) orum “Math for Industry” 2025 - Challenge of Mathematics for Industry in the AI era – 444pages	January 5, 2026

MI Lecture Note Vol.105	Zaiton Mat Isa Arifah Bahar Shariffah Suhaila Syed Jamaludin Zaitul Marlizawati Zainuddin Sharidan Shafie Ahmad Fadillah Embong Shaymaa Mustafa Nur Arina Bazilah Aziz Nik Zetti Amani Nik Faudzi Mohamad Shahiir Saidin Mohd Rashid Admon	International Project Research-Workshop (I) Malaysia – Japan Symposium on Mathematical and Statistical Modelling 122pages	Junualy 8, 2026
MI Lecture Note Vol.106	ハザリカ・ヘマンタ 村井政徳 太田史朗 窪田上太郎 道勇治 藤白隆司 石澤友浩 田中剛 サハレ・アヌラグ 犬飼隆義 廣瀬慧 福本康秀	2025年度採択分 九州大学マス・フォア・インダストリ研究所共同利用研究集会 干拓地における液状化ハザードマップの改善に向けた新たな手法 216pages	January 30, 2026

九州大学マス・フォア・インダストリ研究所
九州大学大学院 数理学府
〒819-0395 福岡市西区元岡744　TEL 092-802-4402　FAX 092-802-4405
URL https://www.imi.kyushu-u.ac.jp/