An Object-Oriented Approach to Idempotent Analysis: Integral Equations as Optimal Control Problems

Marco Pedicini (Roma Tre University)

Optimal Control and Inverse Problems in PDE Theory
Paola Loreti's Festschrift

June 9-13, 2025 Riemann International School of Mathematics
Villa Toeplitz, Varese

a contribution of mathematics to computer science: dependent types

Birth of Generic Types

- **Object–Oriented Programming (OOP)**: introduced with Simula 67 (1960s) to model real-world entities.

Birth of Generic Types

- **Object–Oriented Programming (OOP)**: introduced with Simula 67 (1960s) to model real-world entities.
- Soon the need arose to **reuse code** regardless of the element type—the same **length function** for **List<Int>** and **List<String>**.

Birth of Generic Types

- **Object–Oriented Programming (OOP)**: introduced with Simula 67 (1960s) to model real-world entities.
- Soon the need arose to **reuse code** regardless of the element type—the same **length function** for **List<Int>** and **List<String>**.
- Solution: **generic (parameterised) types**.
  Historical milestones: generics in **Ada** (1983), templates in **C++** (1985), generics in **Java 5** (2004).

Automatic Instantiation and Monomorphisation

- The **compiler** automatically produces concrete instances of a generic function for every actual type used (*monomorphisation*).

Automatic Instantiation and Monomorphisation

- The **compiler** automatically produces concrete instances of a generic function for every actual type used (*monomorphisation*).
- Thus we obtain code reuse without sacrificing performance or type safety.

Automatic Instantiation and Monomorphisation

- The **compiler** automatically produces concrete instances of a generic function for every actual type used (*monomorphisation*).
- Thus we obtain code reuse without sacrificing performance or type safety.
- C++ example:

Automatic Instantiation and Monomorphisation

- The **compiler** automatically produces concrete instances of a generic function for every actual type used (*monomorphisation*).
- Thus we obtain code reuse without sacrificing performance or type safety.
- C++ example:

```
template<typename T> size_t length(const vector<T>& v);
```

Automatic Instantiation and Monomorphisation

- The **compiler** automatically produces concrete instances of a generic function for every actual type used (*monomorphisation*).
- Thus we obtain code reuse without sacrificing performance or type safety.
- C++ example:

```
template<typename T> size_t length(const vector<T>& v);
```

becomes length<int>, length<string>, . . . depending on the concrete type.

Generics $\rightarrow$ Dependent Types

- From parametric polymorphism to **dependent types**: types may depend on *values* (e.g. Vector alpha n carrying its length *n* in the type).

Generics $\rightarrow$ Dependent Types

- From parametric polymorphism to **dependent types**: types may depend on *values* (e.g. Vector alpha n carrying its length *n* in the type).
- They **express invariants** at the type level, guaranteeing them at compile time.

Generics $\rightarrow$ Dependent Types

- From parametric polymorphism to **dependent types**: types may depend on *values* (e.g. Vector alpha n carrying its length *n* in the type).
- They **express invariants** at the type level, guaranteeing them at compile time.
- Foundation of modern **proof assistants** (Coq, Agda, Lean), where code and proofs live together.

Dependent Type Theory and Paradoxes

- **Martin–Löf Type Theory** (1972) introduces *Type : Type*, which triggers **Girard's paradox** $\rightarrow$ inconsistency.

Dependent Type Theory and Paradoxes

- **Martin–Löf Type Theory** (1972) introduces *Type : Type*, which triggers **Girard's paradox** $\rightarrow$ inconsistency.
- The **Calculus of Constructions** (Coquand & Huet 1988) replaces *Type : Type* with a hierarchy of universes $\mathcal{U}_0 \in \mathcal{U}_1 \in \mathcal{U}_2 \ldots$.

Dependent Type Theory and Paradoxes

- **Martin–Löf Type Theory** (1972) introduces *Type : Type*, which triggers **Girard's paradox** $\rightarrow$ inconsistency.
- The **Calculus of Constructions** (Coquand & Huet 1988) replaces *Type : Type* with a hierarchy of universes $\mathcal{U}_0 \in \mathcal{U}_1 \in \mathcal{U}_2 \ldots$.
- This **removes the paradox** while preserving high expressiveness for programming and proving.

Dependent Type Theory and Paradoxes

- **Martin–Löf Type Theory** (1972) introduces *Type : Type*, which triggers **Girard's paradox** $\rightarrow$ inconsistency.
- The **Calculus of Constructions** (Coquand & Huet 1988) replaces *Type : Type* with a hierarchy of universes $\mathcal{U}_0 \in \mathcal{U}_1 \in \mathcal{U}_2 \ldots$.
- This **removes the paradox** while preserving high expressiveness for programming and proving.
- It is the **logical "engine"** of Coq and Lean (with universe polymorphism).

Dependent Type Theory and Paradoxes

- **Martin–Löf Type Theory** (1972) introduces *Type : Type*, which triggers **Girard's paradox** $\rightarrow$ inconsistency.
- The **Calculus of Constructions** (Coquand & Huet 1988) replaces *Type : Type* with a hierarchy of universes $\mathcal{U}_0 \in \mathcal{U}_1 \in \mathcal{U}_2 \ldots$.
- This **removes the paradox** while preserving high expressiveness for programming and proving.
- It is the **logical "engine"** of Coq and Lean (with universe polymorphism).

  Nota Bene: the CoC *definitely is not a* ***civil-engineering statics*** *handbook!*

another contribution of mathematics to computer science

The Lean + Mathlib Project

- **Lean**: a proof assistant based on dependent type theory, developed at Microsoft Research (since 2013).

The Lean + Mathlib Project

- **Lean**: a proof assistant based on dependent type theory, developed at Microsoft Research (since 2013).
- **Mathlib**: community library started in 2017, now $\sim$2.5 million lines, $\approx$60000 formalised theorems, and hundreds of active contributors.

The Lean + Mathlib Project

- **Lean**: a proof assistant based on dependent type theory, developed at Microsoft Research (since 2013).
- **Mathlib**: community library started in 2017, now $\sim$2.5 million lines, $\approx$60000 formalised theorems, and hundreds of active contributors.
- Coverage includes algebra, analysis, geometry, topology, combinatorics, number theory, category theory, and more.

The Lean + Mathlib Project

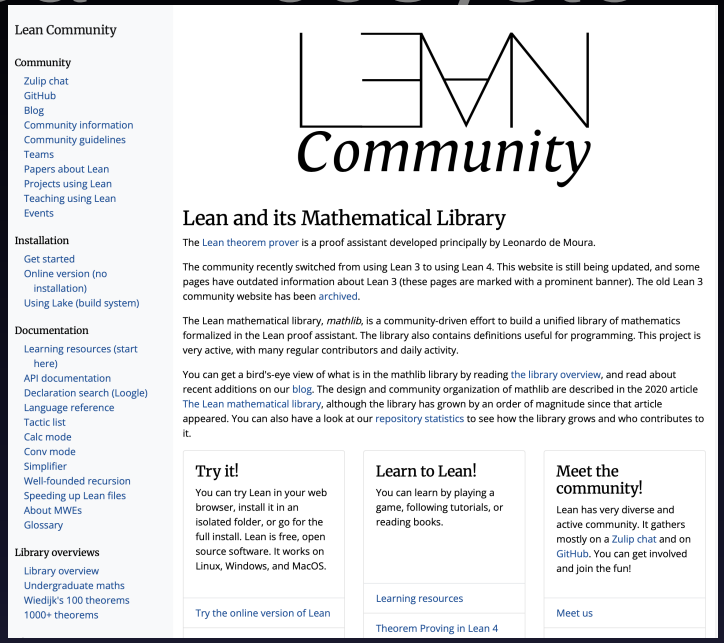
- **Lean**: a proof assistant based on dependent type theory, developed at Microsoft Research (since 2013).
- **Mathlib**: community library started in 2017, now $\sim$2.5 million lines, $\approx$60000 formalised theorems, and hundreds of active contributors.
- Coverage includes algebra, analysis, geometry, topology, combinatorics, number theory, category theory, and more.
- Flagship achievements: formalisation of the Stone–Weierstrass theorem, the *Liquid Tensor* project (Scholze/Buzzard), and parts of analytic number theory.

The Lean + Mathlib Project

- **Lean**: a proof assistant based on dependent type theory, developed at Microsoft Research (since 2013).
- **Mathlib**: community library started in 2017, now $\sim$2.5 million lines, $\approx$60000 formalised theorems, and hundreds of active contributors.
- Coverage includes algebra, analysis, geometry, topology, combinatorics, number theory, category theory, and more.
- Flagship achievements: formalisation of the Stone–Weierstrass theorem, the *Liquid Tensor* project (Scholze/Buzzard), and parts of analytic number theory.
- Fields Medalist **Terence Tao** called Lean/Mathlib "one of the most exciting developments in mathematical practice", encouraging researchers to adopt formalisation.

  T. Tao, on the blog *What's new*, 6 April 2020

Lean Ecosystem



Lean Community

Community
- Zulip chat
- GitHub
- Blog
- Community information
- Community guidelines
- Teams
- Papers about Lean
- Projects using Lean
- Teaching using Lean
- Events

Installation
- Get started
- Online version (no installation)
- Using Lake (build system)

Documentation
- Learning resources (start here)
- API documentation
- Declaration search (Loogle)
- Language reference
- Tactic list
- Calc mode
- Conv mode
- Simplifier
- Well-founded recursion
- Speeding up Lean files
- About MWEs
- Glossary

Library overviews
- Library overview
- Undergraduate maths
- Wiedijk's 100 theorems
- 1000+ theorems

Community

Lean and its Mathematical Library

The Lean theorem prover is a proof assistant developed principally by Leonardo de Moura.

The community recently switched from using Lean 3 to using Lean 4. This website is still being updated, and some pages have outdated information about Lean 3 (these pages are marked with a prominent banner). The old Lean 3 community website has been archived.

The Lean mathematical library, *mathlib*, is a community-driven effort to build a unified library of mathematics formalized in the Lean proof assistant. The library also contains definitions useful for programming. This project is very active, with many regular contributors and daily activity.

You can get a bird's-eye view of what is in the mathlib library by reading the library overview, and read about recent additions on our blog. The design and community organization of mathlib are described in the 2020 article The Lean mathematical library, although the library has grown by an order of magnitude since that article appeared. You can also have a look at our repository statistics to see how the library grows and who contributes to it.

Try it!
You can try Lean in your web browser, install it in an isolated folder, or go for the full install. Lean is free, open source software. It works on Linux, Windows, and MacOS.

Try the online version of Lean

Learn to Lean!
You can learn by playing a game, following tutorials, or reading books.

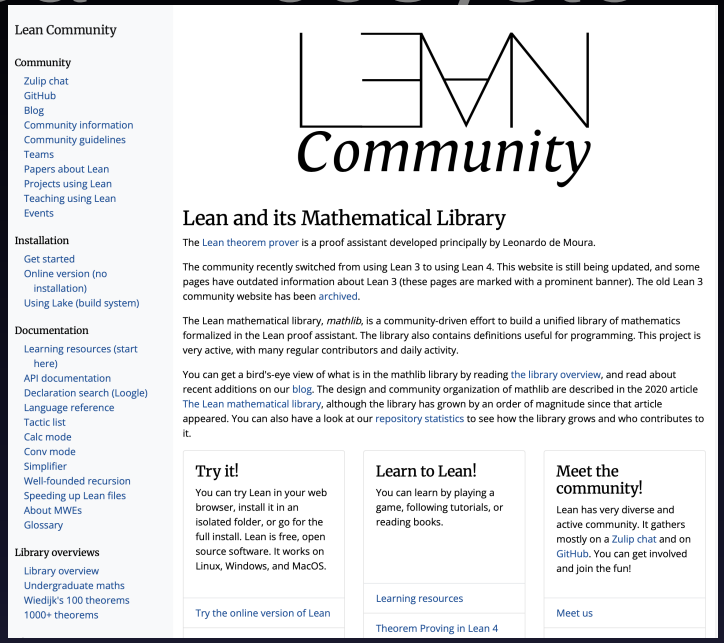
Learning resources

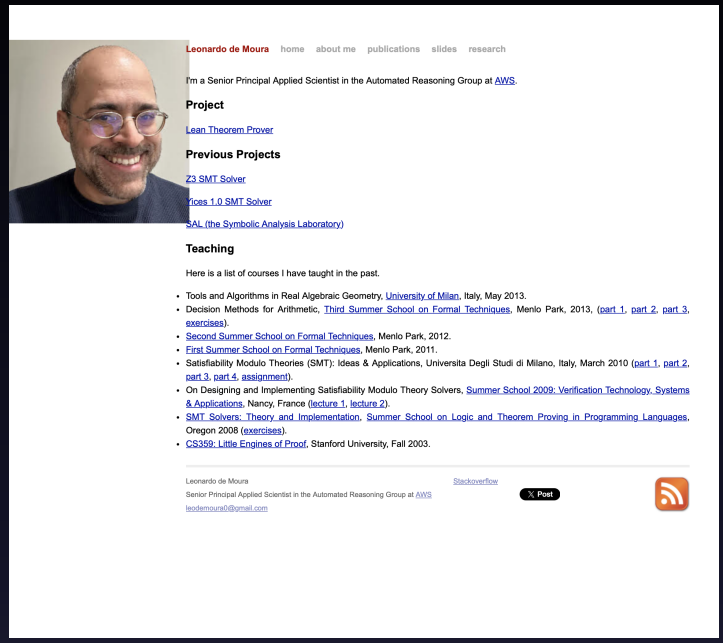
Theorem Proving in Lean 4

Meet the community!
Lean has very diverse and active community. It gathers mostly on a Zulip chat and on GitHub. You can get involved and join the fun!

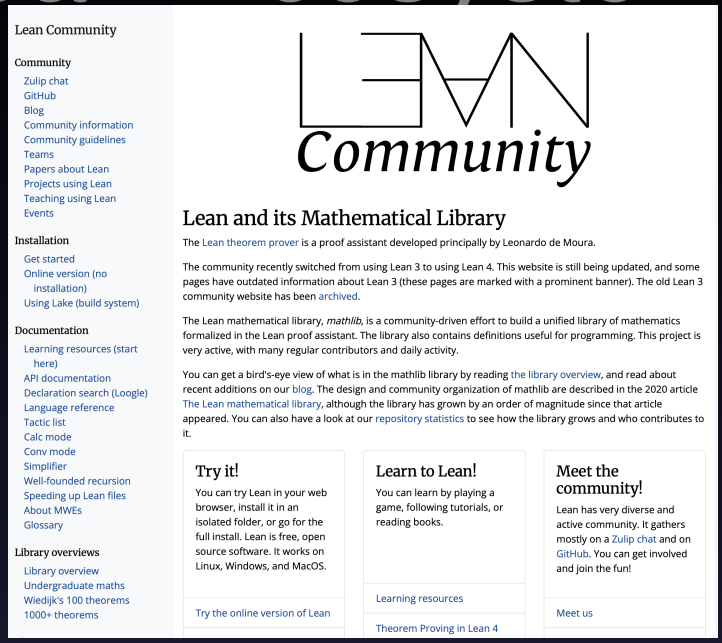
Meet us

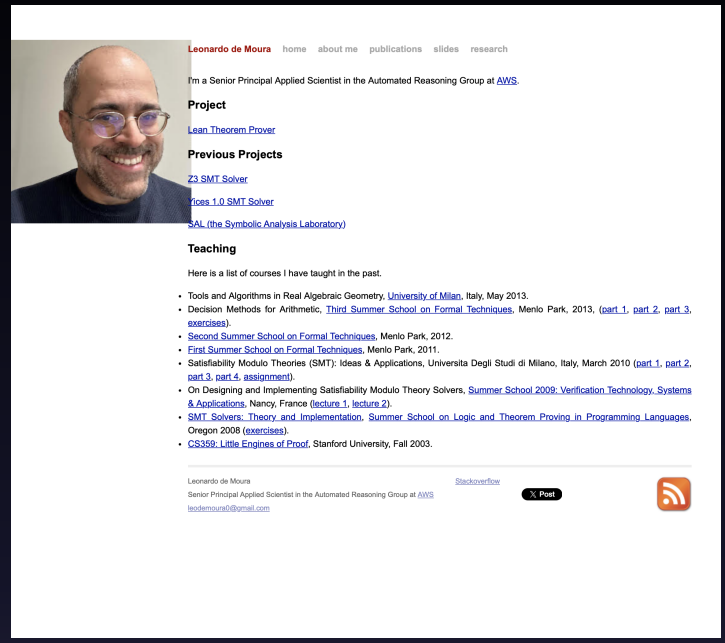
Lean Ecosystem

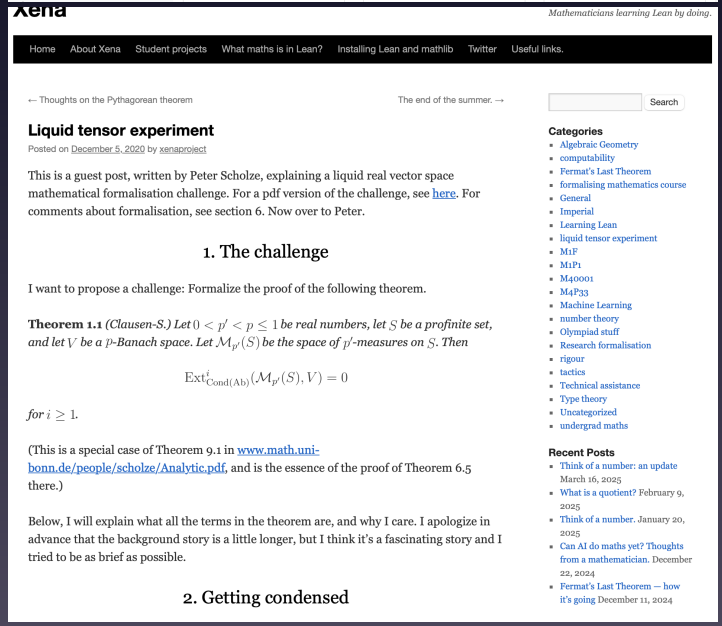




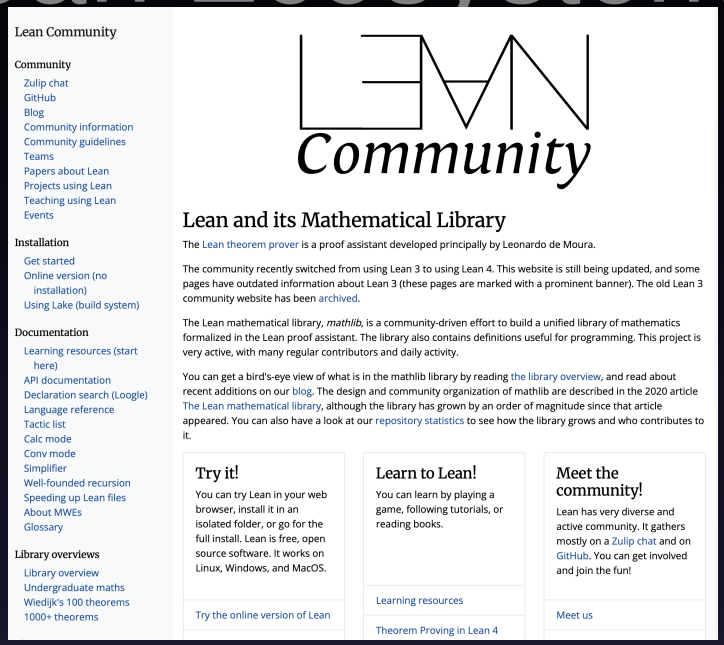
Lean Ecosystem

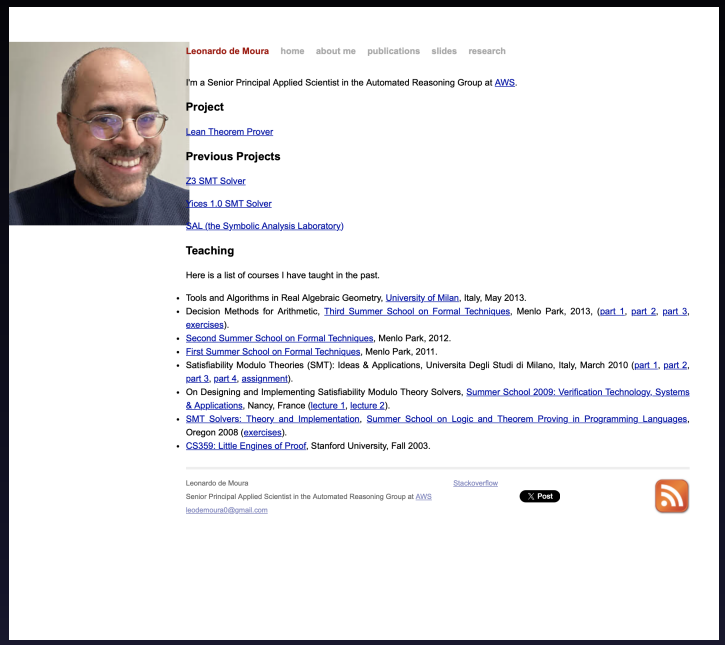


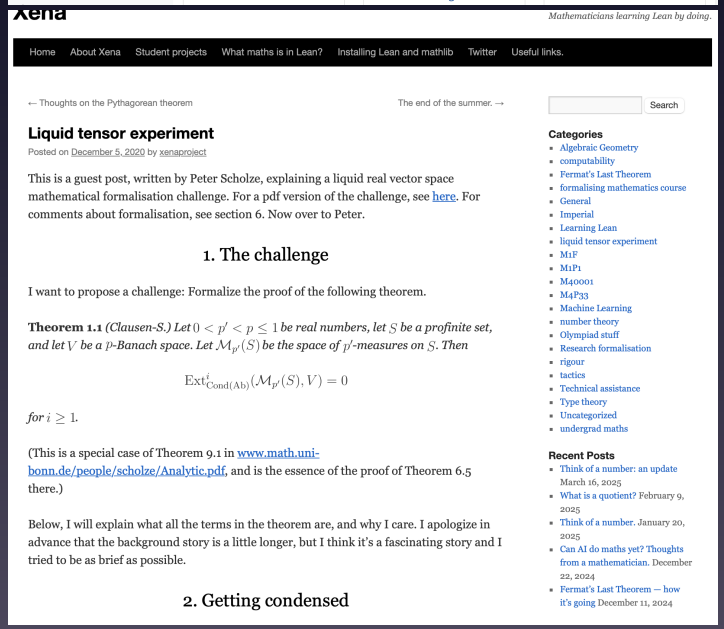


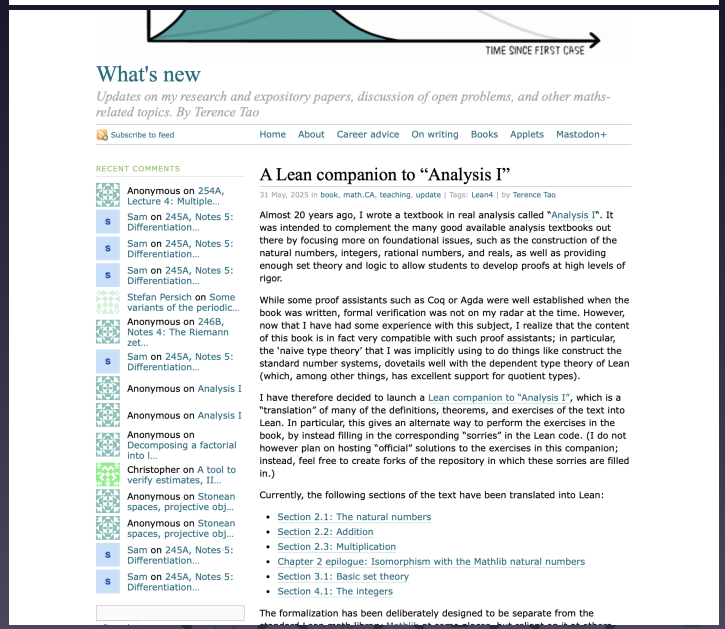


Lean Ecosystem









BACK TO THE NINETIES

Tropicalisation of the Real Numbers

- Classical field: $(\mathbb{R}, +, \times)$ with identities $0, 1$.

Tropicalisation of the Real Numbers

- Classical field: $(\mathbb{R}, +, \times)$ with identities $0, 1$.
- **Tropical semiring**: $(\mathbb{R} \cup \{\infty\}, \min, +)$ with identities $\infty, 0$.

Tropicalisation of the Real Numbers

- Classical field: $(\mathbb{R}, +, \times)$ with identities $0, 1$.
- **Tropical semiring**: $(\mathbb{R} \cup \{\infty\}, \min, +)$ with identities $\infty, 0$.
- Same underlying set, **different algebraic structure**.
  Change of operations yields new mathematics.

Tropicalisation of the Real Numbers

- Classical field: $(\mathbb{R}, +, \times)$ with identities $0, 1$.
- **Tropical semiring**: $(\mathbb{R} \cup \{\infty\}, \min, +)$ with identities $\infty, 0$.
- Same underlying set, **different algebraic structure**. Change of operations yields new mathematics.
- **idempotent addition**: $a \oplus a = a$ leads to piecewise–linear "polynomials" and **tropical geometry**.

Tropicalisation of the Real Numbers

- Classical field: $(\mathbb{R}, +, \times)$ with identities $0, 1$.
- **Tropical semiring**: $(\mathbb{R} \cup \{\infty\}, \min, +)$ with identities $\infty, 0$.
- Same underlying set, **different algebraic structure**. Change of operations yields new mathematics.
- **idempotent addition**: $a \oplus a = a$ leads to piecewise–linear "polynomials" and **tropical geometry**.
- Applications in optimisation, combinatorics, and degeneration of algebraic varieties.

Interpretation and Logical Appeal

- In **model theory**, an **interpretation** assigns meaning to the symbols $\{+, \times, 0, 1\}$, turning the set $\mathbb{R}$ into a specific structure.

Interpretation and Logical Appeal

- In **model theory**, an **interpretation** assigns meaning to the symbols $\{+, \times, 0, 1\}$, turning the set $\mathbb{R}$ into a specific structure.
- A logician is drawn to the power of **switching interpretations** while preserving the underlying domain: the same first-order language yields *non-isomorphic* models with radically different theorems.

Interpretation and Logical Appeal

- In **model theory**, an **interpretation** assigns meaning to the symbols $\{+, \times, 0, 1\}$, turning the set $\mathbb{R}$ into a specific structure.
- A logician is drawn to the power of **switching interpretations** while preserving the underlying domain: the same first-order language yields *non-isomorphic* models with radically different theorems.
- This flexibility underlies many dualities (e.g. Boolean vs. Heyting algebras) and motivates exploring tropical methods in logic and computer science.

generics in control theory

Application of generic programming in control theory

- HJB is central in optimal control theory.

Application of generic programming in control theory

- HJB is central in optimal control theory.
- Tropical algebra: $(\min, +)$ instead of $(+, \times)$.

Application of generic programming in control theory

- HJB is central in optimal control theory.
- Tropical algebra: $(\min, +)$ instead of $(+, \times)$.
- Useful when data is irregular or systems are discrete.

Application of generic programming in control theory

- HJB is central in optimal control theory.
- Tropical algebra: $(\min, +)$ instead of $(+, \times)$.
- Useful when data is irregular or systems are discrete.
- Give and test **generic implementation** of the same solution against classical or tropical problems on reals.

Application of generic programming in control theory

- HJB is central in optimal control theory.
- Tropical algebra: $(\min, +)$ instead of $(+, \times)$.
- Useful when data is irregular or systems are discrete.
- Give and test **generic implementation** of the same solution against classical or tropical problems on reals.

P. Loreti and M. Pedicini, **Idempotent analogue of resolvent kernels for a deterministic optimal control problem**, Mat. Zametki 69 (2001), no. 2, 235–244. MR 2002d:49038

Maslov's Insight

- V. P. Maslov noticed that **viscosity solutions** of certain HJB equations behave *linearly* in suitable idempotent semirings (Maslov 1986; McEneaney 1994)

Maslov's Insight

- V. P. Maslov noticed that **viscosity solutions** of certain HJB equations behave *linearly* in suitable idempotent semirings (Maslov 1986; McEneaney 1994)
- Bellman's dynamic programming principle becomes **linear** over $\left(\mathbb{R} \cup \{+\infty\}, \min, +\right)$.

Maslov's Insight

- V. P. Maslov noticed that **viscosity solutions** of certain HJB equations behave *linearly* in suitable idempotent semirings (Maslov 1986; McEneaney 1994)
- Bellman's dynamic programming principle becomes **linear** over $\big(\mathbb{R} \cup \{+\infty\}, \min, +\big)$.
- This opens the door to algebraic and numerical methods based on idempotent analysis.

Continuous HJB Problem

We studied the optimal-control HJB equation

$$\max\big(u(x)-\psi(x),\ \lambda u(x)+\max_{a\in A}\{-b(x,a)\cdot Du(x)-f(x,a)\}\big)=0.$$

Continuous HJB Problem

We studied the optimal-control HJB equation

$$\max\big(u(x)-\psi(x),\ \lambda u(x)+\max_{a\in A}\{-b(x,a)\cdot Du(x)-f(x,a)\}\big)=0.$$

Terminal cost ψ, discount $\lambda>0$.

Continuous HJB Problem

We studied the optimal-control HJB equation

$$\max\big(u(x)-\psi(x),\ \lambda u(x)+\max_{a\in A}\{-b(x,a)\cdot Du(x)-f(x,a)\}\big)=0.$$

Terminal cost ψ, discount $\lambda>0$.

Control set A,

Continuous HJB Problem

We studied the optimal-control HJB equation

$$\max\big(u(x)-\psi(x),\ \lambda u(x)+\max_{a\in A}\{-b(x,a)\cdot Du(x)-f(x,a)\}\big)=0.$$

Terminal cost ψ, discount $\lambda>0$.

Control set A, **dynamics** $b(x,a)$,

Continuous HJB Problem

We studied the optimal-control HJB equation

$$\max\big(u(x)-\psi(x),\ \lambda u(x)+\max_{a\in A}\{-b(x,a)\cdot Du(x)-f(x,a)\}\big)=0.$$

Terminal cost ψ, discount $\lambda>0$.

Control set A, **dynamics** $b(x,a)$, **running cost** $f(x,a)$.

Continuous HJB Problem

We studied the optimal-control HJB equation

$$\max\big(u(x)-\psi(x),\ \lambda u(x)+\max_{a\in A}\{-b(x,a)\cdot Du(x)-f(x,a)\}\big)=0.$$

Terminal cost ψ, discount $\lambda>0$.

Control set A, **dynamics** $b(x,a)$, **running cost** $f(x,a)$.

From the point of view of idempotent analysis and generic calculus we studied it:

Continuous HJB Problem

We studied the optimal-control HJB equation

$$\max\big(u(x)-\psi(x),\ \lambda u(x)+\max_{a\in A}\{-b(x,a)\cdot Du(x)-f(x,a)\}\big)=0.$$

Terminal cost ψ, discount $\lambda>0$.

Control set A, **dynamics** $b(x,a)$, **running cost** $f(x,a)$.

From the point of view of idempotent analysis and generic calculus we studied it:

In P. Loreti and M. Pedicini. **An Object-Oriented Approach to Idempotent Analysis: Integral Equations as Optimal Control Problems**. Contemporary Mathematics, vol. 377, AMS. Proceedings of the Conference on Idempotent Mathematics and Mathematical Physics. Editors G.L. Litvinov and V.P. Maslov (2005)

Discretised Scheme

The explicit up-wind discretisation reads

$$u_h(x) = \min\big(\psi(x),\ \inf_{a\in A}\big((1-\lambda h)\,u_h(x+hb(x,a)) + hf(x,a)\big)\big).$$

Discretised Scheme

The explicit up-wind discretisation reads

$$u_h(x) = \min\Big(\psi(x),\ \inf_{a\in A}\big((1-\lambda h)\,u_h(x+hb(x,a)) + hf(x,a)\big)\Big).$$

Under standard assumptions (Bardi–Capuzzo Dolcetta 1997) the scheme is **monotone** and **convergent**.

Linear Form over $\mathbb{R}_{\min,+}$

- Define $S(x) = u_h(x)$, $F(x) = \psi(x)$, and the (min–plus) linear operator

$$(HS)(x) = \inf_{a \in A} \big((1 - \lambda h)\, S(x + hb(x,a)) + hf(x,a) \big).$$

- Then **Bellman equation**: $S = HS \oplus F$ with $\oplus = \min$ and $\odot = +$.
- Hence HJB Equation is **linear** in the idempotent semiring $\mathbb{R}_{\min,+}$.

Idempotent Integral Equation

Using the **idempotent integral** $\int^{\oplus}$, we rewrite

$$u_h(x) = \psi(x) \oplus \tilde{\lambda} \int_{B(x,A)}^{\oplus} u_h(\xi) \odot G_h(x,\xi)\, d\xi,$$

Idempotent Integral Equation

Using the **idempotent integral** $\int^{\oplus}$, we rewrite

$$u_h(x) = \psi(x) \oplus \tilde{\lambda} \int^{\oplus}_{B(x,A)} u_h(\xi) \odot G_h(x,\xi)\, d\xi,$$

where $\tilde{\lambda} = 1 - \lambda h$, $B(x,a) = x + hb(x,a)$ and the kernel

$$G_h(x,\xi) = \frac{h}{1-\lambda h} \min\{f(x,a) : a \in A,\ \xi = x + hb(x,a)\}.$$

Idempotent Integral Equation

Using the **idempotent integral** $\int^{\oplus}$, we rewrite

$$u_h(x) = \psi(x) \oplus \tilde{\lambda} \int_{B(x,A)}^{\oplus} u_h(\xi) \odot G_h(x,\xi)\, d\xi,$$

where $\tilde{\lambda} = 1 - \lambda h$, $B(x,a) = x + hb(x,a)$ and the kernel

$$G_h(x,\xi) = \frac{h}{1-\lambda h} \min\{f(x,a) : a \in A,\ \xi = x + hb(x,a)\}.$$

This is a **Volterra-type** equation in the tropical semiring.

Resolvent Kernels and p-Step Approximation

- **Iterated kernels** $K^{(n)}$ defined by tropical convolution yield the series

$$u_h(x) = \psi(x) \oplus \bigoplus_{n=1}^{\infty} \tilde{\lambda}^n \int_{B^n(x,A)}^{\oplus} \psi(t) \odot K^{(n)}(t,x)\, dt.$$

Resolvent Kernels and p-Step Approximation

- **Iterated kernels** $K^{(n)}$ defined by tropical convolution yield the series

$$u_h(x) = \psi(x) \oplus \bigoplus_{n=1}^{\infty} \tilde{\lambda}^n \int_{B^n(x,A)}^{\oplus} \psi(t) \odot K^{(n)}(t,x)\, dt.$$

- Truncating at $n = p$ gives the **p-step dynamic programming approximation**:

$$u_h(x) = \psi(x) \oplus \bigoplus_{n=1}^{p} \tilde{\lambda}^n \int_{B^n(x,A)}^{\oplus} \psi(t) \odot K^{(n)}(t,x) dt \oplus$$
$$\oplus \tilde{\lambda}^{p+1} \int_{B^{p+1}(x,A)}^{\oplus} u_h(t) \odot K^{(p+1)}(t,x) dt.$$

Resolvent Kernels and p-Step Approximation

- **Iterated kernels** $K^{(n)}$ defined by tropical convolution yield the series

$$u_h(x) = \psi(x) \oplus \bigoplus_{n=1}^{\infty} \tilde{\lambda}^n \int_{B^n(x,A)}^{\oplus} \psi(t) \odot K^{(n)}(t,x)\, dt.$$

- Truncating at $n = p$ gives the **p-step dynamic programming approximation**:

$$u_h(x) = \psi(x) \oplus \bigoplus_{n=1}^{p} \tilde{\lambda}^n \int_{B^n(x,A)}^{\oplus} \psi(t) \odot K^{(n)}(t,x) dt \oplus \\ \oplus \tilde{\lambda}^{p+1} \int_{B^{p+1}(x,A)}^{\oplus} u_h(t) \odot K^{(p+1)}(t,x) dt.$$

- Matches the classical value-iteration algorithm.

Generic Implementation

We provide **two generic templates** required for the implementation of integral kernels. These templates are automatically adapted when used with usual real numbers or within min-plus algebra.

Generic Implementation

We provide **two generic templates** required for the implementation of integral kernels. These templates are automatically adapted when used with usual real numbers or within min-plus algebra.

Then, we provide two modules defining operations and constants on the real numbers structure, and functions used for the discretization of intervals. This part is introduced to realize the proper instatiation of the arithmetical structure.

Generic Implementation

We provide **two generic templates** required for the implementation of integral kernels. These templates are automatically adapted when used with usual real numbers or within min-plus algebra.

Then, we provide two modules defining operations and constants on the real numbers structure, and functions used for the discretization of intervals. This part is introduced to realize the proper instatiation of the arithmetical structure.

The second part of the implementation consists just in the definition of the particular problem we aim to solve. So, we specify the integral equation of second type by the **kernel** and its **constant term function** (in the analog of the optimal control problem, they can be derived from dynamics, the cost function and the stopping time cost function).

Solution

At the very end of the program, the approximate **solution of the integral equation** is computed (and in the analog of the optimal control problem, this corresponds to an approximation of the value function).

Solution

At the very end of the program, the approximate **solution of the integral equation** is computed (and in the analog of the optimal control problem, this corresponds to an approximation of the value function).

We also compared the approximate solution with viscosity solutions (the classic approximation). In some problems they coincide.

Idempotent Equations and Applications

Reformulated analogues of integral/differential equations:

- HJB has an idempotent formulation for optimal trajectory computation.

Applications:

- Optimization (e.g. scheduling, timed networks)
- Control theory (Bellman–Hamilton–Jacobi)
- Game theory and decision
- Tropical mathematics (degenerations of classical structures)
- Automata theory (quantitative semantics)
- Classical limits of quantum mechanics (Maslov's idempotent mechanics)

Concluding Remarks

- Idempotent analysis offers a combinatorial view of HJB
- Tropical solutions often match viscosity solutions
- Min-plus convolutions provide a practical computational method
- Useful in discrete models, numerical schemes, graph optimization

Thanks!

Martino Bardi and Italo Capuzzo-Dolcetta, **Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations**, Systems & Control: Foundations & Applications, Birkhäuser Boston Inc., Boston, MA, 1997, With appendices by Maurizio Falcone and Pierpaolo Soravia. MR 99e:49001

I. Capuzzo-Dolcetta and H. Ishii, **Approximate solutions of the Bellman equation of deterministic control theory**, Appl. Math. Optim. **11** (1984), no. 2, 161–181. MR 85f:49046

M. Falcone, **A numerical approach to the infinite horizon problem of deterministic control theory**, Appl. Math. Optim. **15** (1987), no. 1, 1–13. MR 88c:49025

V.N. Kolokoltsov and V.P. Maslov, **Idempotent analysis and its applications. Appendix by Pierre Del Moral. Transl. from the Russian by V. E. Nazaikinskij.**, Mathematics and its Applications (Dordrecht). 401. Dordrecht: Kluwer Academic Publishers., 1997 (English).

Grigori L. Litvinov and Victor P. Maslov, **The correspondence principle for idempotent calculus and some computer applications**, Idempotency (Bristol, 1994), Publ. Newton Inst., vol. 11, Cambridge Univ. Press, Cambridge, 1998, E-print math. GM/0101021 (http://arXiv.org), pp. 420–443. MR 99c:16050

G. L. Litvinov and E. V. Maslova, **Universal numerical algorithms and their software implementation**, Programmirovanie (2000), no. 5, 53–62, translation in Program. Comput. Software 26 (2000), no. 5, 275–280, E-print math. SC/0102114 (http://arXiv.org). MR 1 820 305

P. Loreti and M. Pedicini, **Idempotent analogue of resolvent kernels for a deterministic optimal control problem**, Mat. Zametki **69** (2001), no. 2, 235–244. MR 2002d:49038

V. P. Maslov, **On a new principle of superposition for optimization problems**, Uspekhi Mat. Nauk **42** (1987), no. 3(255), 39–48, 255. MR 88h:35103

V. P. Maslov and S. N. Samborskiĭ, **Stationary Hamilton-Jacobi and Bellman equations (existence and uniqueness of solutions)**, Idempotent analysis, Adv. Soviet Math., vol. 13, Amer. Math. Soc., Providence, RI, 1992, pp. 119–133. MR 94d:49048

A. A. Stepanov and M. Lee, **The Standard Template Library**, Tech. Report HPL-94-34, Hewlett Packard Company, Technical Report, Palo Alto CA, 1994.

V. Volterra and J. Pérès, **Théorie générale des fonctionelles**, Collection de monographies sur la théorie des fonctions, vol. 1, Gauthier-Villars, Paris, 1936.