What Has Artificial Intelligence Ever Done for Us? (Formalizers)

John Harrison
Intel Corporation

AITP 2017, Obergurgl

26th March 2017 (15:00–15:45)

Contents

- Historical connection of AI, ATP and ITP
- The state of the art in interactive proof
- Case study: the HOL Light Multivariate library
- AI techniques: achievements and potential
  - More automated proofs
  - More elegant or efficient proofs
  - Automatic generalization of proofs
  - Concept/connection discovery?
- Questions / discussions

Historical connection of AI, ATP and ITP

Early research in automated reasoning

Most early theorem provers were fully automatic, and there was a somewhat clear division into:

- Human-oriented AI style approaches (Newell-Simon, Gelerntner)

Early research in automated reasoning

Most early theorem provers were fully automatic, and there was a somewhat clear division into:

- Human-oriented AI style approaches (Newell-Simon, Gelerntner)
- Machine-oriented algorithmic approaches (Davis, Gilmore, Wang, Prawitz)

Early research in automated reasoning

Most early theorem provers were fully automatic, and there was a somewhat clear division into:

- Human-oriented AI style approaches (Newell-Simon, Gelerntner)
- Machine-oriented algorithmic approaches (Davis, Gilmore, Wang, Prawitz)

After a few years the machine-oriented style took over almost completely, with only a few like Bledsoe pursuing AI.

The early victory of machine-oriented methods

A typical comparison of the time of a machine-oriented approach to FOL against the AI approach of Newell, Shore and Simon:

> *[...] the comparison reveals a fundamental inadequacy in their approach. There is no need to kill a chicken with a butcher's knife. Yet the net impression is that Newell-Shore-Simon failed even to kill the chicken with their butcher's knife.*

Wang, "Toward Mechanical Mathematics" (IBM J. Res. Dev 1960)

The early victory of machine-oriented methods

Machine-oriented methods made significant advances with various new algorithms or approaches, e.g.

The early victory of machine-oriented methods

Machine-oriented methods made significant advances with various new algorithms or approaches, e.g.

- Unification-based first-order methods like resolution

The early victory of machine-oriented methods

Machine-oriented methods made significant advances with various new algorithms or approaches, e.g.

- Unification-based first-order methods like resolution
- Completion for equational logic

The early victory of machine-oriented methods

Machine-oriented methods made significant advances with various new algorithms or approaches, e.g.

- Unification-based first-order methods like resolution
- Completion for equational logic
- Wu's algorithm and Gröbner bases for algebra and geometry

The early victory of machine-oriented methods

Machine-oriented methods made significant advances with various new algorithms or approaches, e.g.

- Unification-based first-order methods like resolution
- Completion for equational logic
- Wu's algorithm and Gröbner bases for algebra and geometry
- Cooper's elementary-time algorithm for linear integer (Presburger) arithmetic.

The early victory of machine-oriented methods

Machine-oriented methods made significant advances with various new algorithms or approaches, e.g.

- Unification-based first-order methods like resolution
- Completion for equational logic
- Wu's algorithm and Gröbner bases for algebra and geometry
- Cooper's elementary-time algorithm for linear integer (Presburger) arithmetic. First published in *Machine Intelligence* though...

The early victory of machine-oriented methods

Machine-oriented methods made significant advances with various new algorithms or approaches, e.g.

- Unification-based first-order methods like resolution
- Completion for equational logic
- Wu's algorithm and Gröbner bases for algebra and geometry
- Cooper's elementary-time algorithm for linear integer (Presburger) arithmetic. First published in *Machine Intelligence* though...

Such techniques could often solve some quite large and impressive problems.

The seventies: From automated to interactive proving

However, during the 1970s there was increasing interest in more 'interactive' theorem proving

The seventies: From automated to interactive proving

However, during the 1970s there was increasing interest in more 'interactive' theorem proving

- Abilities of ATP systems had grown fast but were now starting to plateau

The seventies: From automated to interactive proving

However, during the 1970s there was increasing interest in more 'interactive' theorem proving

- Abilities of ATP systems had grown fast but were now starting to plateau
- More interactive computing environments made it natural/convenient

The seventies: From automated to interactive proving

However, during the 1970s there was increasing interest in more 'interactive' theorem proving

- Abilities of ATP systems had grown fast but were now starting to plateau
- More interactive computing environments made it natural/convenient

It seems paradoxical that 'difficult' full automation was pursued seriously long before 'easy' partial automation.

Still no resurgence of AI

However, the rise of interactive theorem proving, if anything, led to even less interest in AI:

Still no resurgence of AI

However, the rise of interactive theorem proving, if anything, led to even less interest in AI:

> *I wrote an automatic theorem prover in Swansea for myself and became shattered with the difficulty of doing anything interesting in that direction and I still am. I greatly admired Robinson's resolution principle, a wonderful breakthrough; but in fact the amount of stuff you can prove with fully automatic theorem proving is still very small. So I was always more interested in amplifying human intelligence than I am in artificial intelligence.*

Robin Milner, interviewed by Martin Berger, 2003.

Early interactive provers (1960s–1970s)

A non-exhaustive list of early work in the field:

- Paul Abrahams's `Proofchecker`
- Bledsoe and Gilbert's checker for Morse's set theory
- The SAM family
- AUTOMATH
- Mizar
- LCF

Early interactive provers (1960s–1970s)

A non-exhaustive list of early work in the field:

- Paul Abrahams's `Proofchecker`
- Bledsoe and Gilbert's checker for Morse's set theory
- The SAM family
- AUTOMATH
- Mizar
- LCF

The last three have been quite influential on the current state of the field. Also important ideas from program verification systems. . .

The state of the art in interactive proof

Progress in interactive theorem proving

Work since the early proof checkers has focused on

- Exploring various foundations, particularly type-theoretic
- Efficient and convenient proof input languages
- Methods for ensuring provers are reliable
- Developing mathematical libraries
- Incorporating automated decision procedures for subproblems

Progress in interactive theorem proving

Work since the early proof checkers has focused on

- Exploring various foundations, particularly type-theoretic
- Efficient and convenient proof input languages
- Methods for ensuring provers are reliable
- Developing mathematical libraries
- Incorporating automated decision procedures for subproblems

Many of the interesting problems arise from conflicts and incompatibilities between these

- How to support programmability in a proof language without making it unreadable? (Combining 'procedural' and 'declarative' proof constructs, ...)

Progress in interactive theorem proving

Work since the early proof checkers has focused on

- Exploring various foundations, particularly type-theoretic
- Efficient and convenient proof input languages
- Methods for ensuring provers are reliable
- Developing mathematical libraries
- Incorporating automated decision procedures for subproblems

Many of the interesting problems arise from conflicts and incompatibilities between these

- How to support programmability in a proof language without making it unreadable? (Combining 'procedural' and 'declarative' proof constructs, ...)
- How to incorporate decision procedures without sacrificing reliability? (Proof/certificate reconstruction/checking, reflection, ...)

Some automation available in leading ITPs

- Conditional rewriting and related simplification
- Pure logic proof search (SAT, FOL, HOL)
- Decision procedures for numerical theories (linear arithmetic and algebra, SMT).
- Quantifier elimination procedures for arithmetical theories
- Derived procedures for inductive and recursive definitions
- More specialized decision procedures for particular contexts

Typical ‘efficient style of proof’ in interactive systems

The typical ‘efficient’ style is to use a few high-level steps to break the proof down or establish useful lemmas or intermediate assertions, then use some automated procedure.

Typical ‘efficient style of proof’ in interactive systems

The typical ‘efficient’ style is to use a few high-level steps to break the proof down or establish useful lemmas or intermediate assertions, then use some automated procedure.

```
let SUM_OF_NUMBERS = prove
 (`!n. nsum(1..n) (\i. i) = (n * (n + 1)) DIV 2`,
  INDUCT_TAC THEN
  SIMP_TAC[NSUM_CLAUSES_NUMSEG] THEN
  ASM_ARITH_TAC);;
```

Typical ‘efficient style of proof’ in interactive systems

The typical ‘efficient’ style is to use a few high-level steps to break the proof down or establish useful lemmas or intermediate assertions, then use some automated procedure.

```
let SUM_OF_NUMBERS = prove
 (`!n. nsum(1..n) (\i. i) = (n * (n + 1)) DIV 2`,
  INDUCT_TAC THEN
  SIMP_TAC[NSUM_CLAUSES_NUMSEG] THEN
  ASM_ARITH_TAC);;
```

However, it is often a lengthy process to break a less trivial proof down so as to harness automation effectively without having it spin out of control.

The rise of learning

Finally, the use of modern learning-based techniques began to invade theorem proving:

The rise of learning

Finally, the use of modern learning-based techniques began to invade theorem proving:

- Large theorem databases processed into a form suitable for AI exploration
  - The Mizar Mathematical Library (MML)
  - The HOL Light Multivariate libraries and Flyspeck proof
  - The Isabelle Archive of Formal Proofs (AFP)

The rise of learning

Finally, the use of modern learning-based techniques began to invade theorem proving:

- Large theorem databases processed into a form suitable for AI exploration
  - The Mizar Mathematical Library (MML)
  - The HOL Light Multivariate libraries and Flyspeck proof
  - The Isabelle Archive of Formal Proofs (AFP)
- Learning-based techniques applied to these datasets, notably premiss selection for automated subsystems.

The rise of learning

Finally, the use of modern learning-based techniques began to invade theorem proving:

- Large theorem databases processed into a form suitable for AI exploration
  - The Mizar Mathematical Library (MML)
  - The HOL Light Multivariate libraries and Flyspeck proof
  - The Isabelle Archive of Formal Proofs (AFP)
- Learning-based techniques applied to these datasets, notably premiss selection for automated subsystems.

Slightly paradoxical that it is in the world of *interactive* rather than *automated* theorem proving that we have the large datasets needed to train the AI techniques!

Case study: The HOL Light Multivariate library

The HOL Light Multivariate library

- Developed over the course of more than 10 years, originally to support the Flyspeck proof.

The HOL Light Multivariate library

- Developed over the course of more than 10 years, originally to support the Flyspeck proof.
- Covers topology, analysis, geometry, integration and measure in Euclidean spaces $\mathbb{R}^n$

The HOL Light Multivariate library

- Developed over the course of more than 10 years, originally to support the Flyspeck proof.
- Covers topology, analysis, geometry, integration and measure in Euclidean spaces $\mathbb{R}^n$
- Doubly interesting from the point of view of AI/learning:
  - Large enough (especially in combination with Flyspeck) to be valuable training material

The HOL Light Multivariate library

- Developed over the course of more than 10 years, originally to support the Flyspeck proof.
- Covers topology, analysis, geometry, integration and measure in Euclidean spaces $\mathbb{R}^n$
- Doubly interesting from the point of view of AI/learning:
  - Large enough (especially in combination with Flyspeck) to be valuable training material
  - Plenty of room for improvement in terms of quality of proofs and generality of results.

The HOL Light Multivariate library

- Developed over the course of more than 10 years, originally to support the Flyspeck proof.
- Covers topology, analysis, geometry, integration and measure in Euclidean spaces $\mathbb{R}^n$
- Doubly interesting from the point of view of AI/learning:
  - Large enough (especially in combination with Flyspeck) to be valuable training material
  - Plenty of room for improvement in terms of quality of proofs and generality of results.
- Kaliszyk and Urban's HOL(y)Hammer is already making an impact here and we believe much more may be possible in the future

The core Multivariate library

Covers general properties of $\mathbb{R}^n$ and sometimes more general spaces:

The core Multivariate library

Covers general properties of $\mathbb{R}^n$ and sometimes more general spaces:

File	Lines	Contents
misc.ml	2361	Background stuff
metric .ml	8528	Metric spaces and general topology
vectors.ml	10766	Basic vectors, linear algebra
determinants.ml	4733	Determinant and trace
topology.ml	35288	Topology of euclidean space
convex.ml	17826	Convex sets and functions
paths.ml	27867	Paths, simple connectedness etc.
polytope.ml	8952	Faces, polytopes, polyhedra etc.
degree.ml	11934	Degree theory, retracts etc.
derivatives.ml	5763	Derivatives
clifford.ml	979	Geometric (Clifford) algebra
integration.ml	26107	Integration
measure.ml	29806	Lebesgue measure
TOTAL	190910	

Multivariate theories continued

Complex analysis and real analysis as special cases and more:

File	Lines	Contents
complexes.ml	2237	Complex numbers
canal.ml	3907	Complex analysis
transcendentals.ml	7926	Real & complex transcendentals
realanalysis.ml	16258	Some analytical stuff on $\mathbb{R}$
moretop.ml	8339	Further topological results
cauchy.ml	23773	Complex line integrals
geom.ml	1249	Geometric concepts (angles etc.)
cross.ml	279	Cross products in $\mathbb{R}^3$
gamma.ml	3778	Real and complex Γ function
lpspaces	1311	L_p spaces
flyspeck.ml	7091	Some concepts and lemmas for Flyspeck
TOTAL	76148	

Multivariate theories continued

Complex analysis and real analysis as special cases and more:

File	Lines	Contents
complexes.ml	2237	Complex numbers
canal.ml	3907	Complex analysis
transcendentals.ml	7926	Real & complex transcendentals
realanalysis.ml	16258	Some analytical stuff on $\mathbb{R}$
moretop.ml	8339	Further topological results
cauchy.ml	23773	Complex line integrals
geom.ml	1249	Geometric concepts (angles etc.)
cross.ml	279	Cross products in $\mathbb{R}^3$
gamma.ml	3778	Real and complex Γ function
lpspaces	1311	L_p spaces
flyspeck.ml	7091	Some concepts and lemmas for Flyspeck
TOTAL	76148	

In total, over 16000 theorems, some trivial, some quite interesting.
Credits: JRH, Marco Maggesi, Valentina Bruno, Graziano Gentili, Gianni Ciolli, Lars Schewe, . . .

A few examples (1)

Brouwer's fixed-point theorem

```
|- ∀f:real^N->real^N s.
      compact s ∧ convex s ∧ ¬(s = {}) ∧
      f continuous_on s ∧ IMAGE f s SUBSET s
      ⇒ ∃x. x IN s ∧ f x = x
```

A few examples (1)

Brouwer's fixed-point theorem

```
|- ∀f:real^N->real^N s.
      compact s ∧ convex s ∧ ¬(s = {}) ∧
      f continuous_on s ∧ IMAGE f s SUBSET s
      ⇒ ∃x. x IN s ∧ f x = x
```

Invariance of domain:

```
|- ∀f:real^N->real^N s.
      f continuous_on s ∧ open s ∧
      (∀x y. x IN s ∧ y IN s ∧ f x = f y ⇒ x = y)
      ⇒ open(IMAGE f s)
```

A few examples (1)

Brouwer's fixed-point theorem

```
|- ∀f:real^N->real^N s.
       compact s ∧ convex s ∧ ¬(s = {}) ∧
       f continuous_on s ∧ IMAGE f s SUBSET s
       ⇒ ∃x. x IN s ∧ f x = x
```

Invariance of domain:

```
|- ∀f:real^N->real^N s.
       f continuous_on s ∧ open s ∧
       (∀x y. x IN s ∧ y IN s ∧ f x = f y ⇒ x = y)
       ⇒ open(IMAGE f s)
```

The fundamental theorem of calculus:

```
|- ∀f:real->real f' s a b.
       COUNTABLE s ∧
       a <= b ∧ f real_continuous_on real_interval[a,b] ∧
       (∀x. x IN real_interval(a,b) DIFF s
            ⇒ (f has_real_derivative f'(x)) (atreal x))
       ⇒ (f' has_real_integral (f(b) - f(a))) (real_interval[a,b])
```

A few examples (2)

The Lebesgue differentiation theorem

```
|- ∀f:real^1->real^N s.
        is_interval s ∧ f has_bounded_variation_on s
        ⇒ negligible {x | x IN s ∧ ¬(f differentiable at x)}
```

A few examples (2)

The Lebesgue differentiation theorem

```
|- ∀f:real^1->real^N s.
      is_interval s ∧ f has_bounded_variation_on s
      ⇒ negligible {x | x IN s ∧ ¬(f differentiable at x)}
```

Rademacher's theorem on differentiability of Lipschitz function

```
|- ∀f:real^M->real^N s.
     open s ∧
     (∃B. ∀x y. x IN s ∧ y IN s ⇒ norm(f x - f y) <= B * norm(x - y))
     ⇒ negligible {x | x IN s ∧ ¬(f differentiable (at x))}
```

A few examples (2)

The Lebesgue differentiation theorem

```
|- ∀f:real^1->real^N s.
      is_interval s ∧ f has_bounded_variation_on s
      ⇒ negligible {x | x IN s ∧ ¬(f differentiable at x)}
```

Rademacher's theorem on differentiability of Lipschitz function

```
|- ∀f:real^M->real^N s.
     open s ∧
     (∃B. ∀x y. x IN s ∧ y IN s ⇒ norm(f x - f y) <= B * norm(x - y))
     ⇒ negligible {x | x IN s ∧ ¬(f differentiable (at x))}
```

The Little Picard theorem:

```
|- ∀f:complex->complex a b.
     f holomorphic_on (:complex) ∧
     ¬(a = b) ∧ IMAGE f (:complex) INTER {a,b} = {}
     ⇒ ∃c. f = λx. c
```

AI Techniques: achievements and potential

More automated proofs

HOL(y)Hammer's combination of learning and ATP linkup is often able to automate the proof of simple theorems, e.g. union of nowhere dense sets is nowhere dense:

```
|- ∀s t.
     interior(closure s) = {} ∧ interior(closure t) = {}
     ⇒ interior(closure(s UNION t)) = {}
```

More automated proofs

HOL(y)Hammer's combination of learning and ATP linkup is often able to automate the proof of simple theorems, e.g. union of nowhere dense sets is nowhere dense:

```
|- ∀s t.
      interior(closure s) = {} ∧ interior(closure t) = {}
      ⇒ interior(closure(s UNION t)) = {}
```

There seems much more potential here:

- Kaliszyk and Urban reported in 2014 that 39% of the toplevel Flyspeck theorems could be proved automatically.
- There has been steady progress since then and more can be expected.

More elegant or efficient proofs

There are quite a few relatively clumsy or lengthy proofs in the Multivariate library that HOL(y)Hammer can do better:

More elegant or efficient proofs

There are quite a few relatively clumsy or lengthy proofs in the Multivariate library that HOL(y)Hammer can do better:

```
let FACE_OF_POLYHEDRON_POLYHEDRON = prove
 (`!s:real^N->bool c. polyhedron s /\ c face_of s ==> polyhedron c`,
  REPEAT STRIP_TAC THEN FIRST_ASSUM
   (MP_TAC o GEN_REWRITE_RULE I [POLYHEDRON_INTER_AFFINE_MINIMAL]) THEN
  REWRITE_TAC[RIGHT_IMP_EXISTS_THM; SKOLEM_THM] THEN
  SIMP_TAC[LEFT_IMP_EXISTS_THM; RIGHT_AND_EXISTS_THM; LEFT_AND_EXISTS_THM] THEN
  MAP_EVERY X_GEN_TAC
   [`f:(real^N->bool)->bool`; `a:(real^N->bool)->real^N`;
    `b:(real^N->bool)->real`] THEN
  STRIP_TAC THEN
  MP_TAC(ISPECL [`s:real^N->bool`; `f:(real^N->bool)->bool`;
                `a:(real^N->bool)->real^N`; `b:(real^N->bool)->real`]
         FACE_OF_POLYHEDRON_EXPLICIT) THEN
  ANTS_TAC THENL [ASM_REWRITE_TAC[] THEN ASM_MESON_TAC[]; ALL_TAC] THEN
  DISCH_THEN(MP_TAC o SPEC `c:real^N->bool`) THEN ASM_REWRITE_TAC[] THEN
  ASM_CASES_TAC `c:real^N->bool = {}` THEN
  ASM_REWRITE_TAC[POLYHEDRON_EMPTY] THEN
  ASM_CASES_TAC `c:real^N->bool = s` THEN ASM_REWRITE_TAC[] THEN
  DISCH_THEN SUBST1_TAC THEN MATCH_MP_TAC POLYHEDRON_INTERS THEN
  REWRITE_TAC[FORALL_IN_GSPEC] THEN
  ONCE_REWRITE_TAC[SIMPLE_IMAGE_GEN] THEN
  ASM_SIMP_TAC[FINITE_IMAGE; FINITE_RESTRICT] THEN
  REPEAT STRIP_TAC THEN REWRITE_TAC[IMAGE_ID] THEN
  MATCH_MP_TAC POLYHEDRON_INTER THEN
  ASM_REWRITE_TAC[POLYHEDRON_HYPERPLANE]);;
```

More elegant or efficient proofs

There are quite a few relatively clumsy or lengthy proofs in the Multivariate library that HOL(y)Hammer can do better:

```
let FACE_OF_POLYHEDRON_POLYHEDRON = prove
 (`!s:real^N->bool c. polyhedron s /\ c face_of s ==> polyhedron c`,
  REPEAT STRIP_TAC THEN FIRST_ASSUM
   (MP_TAC o GEN_REWRITE_RULE I [POLYHEDRON_INTER_AFFINE_MINIMAL]) THEN
  REWRITE_TAC[RIGHT_IMP_EXISTS_THM; SKOLEM_THM] THEN
  SIMP_TAC[LEFT_IMP_EXISTS_THM; RIGHT_AND_EXISTS_THM; LEFT_AND_EXISTS_THM] THEN
  MAP_EVERY X_GEN_TAC
   [`f:(real^N->bool)->bool`; `a:(real^N->bool)->real^N`;
    `b:(real^N->bool)->real`] THEN
  STRIP_TAC THEN
  MP_TAC(ISPECL [`s:real^N->bool`; `f:(real^N->bool)->bool`;
                 `a:(real^N->bool)->real^N`; `b:(real^N->bool)->real`]
         FACE_OF_POLYHEDRON_EXPLICIT) THEN
  ANTS_TAC THENL [ASM_REWRITE_TAC[] THEN ASM_MESON_TAC[]; ALL_TAC] THEN
  DISCH_THEN(MP_TAC o SPEC `c:real^N->bool`) THEN ASM_REWRITE_TAC[] THEN
  ASM_CASES_TAC `c:real^N->bool = {}` THEN
  ASM_REWRITE_TAC[POLYHEDRON_EMPTY] THEN
  ASM_CASES_TAC `c:real^N->bool = s` THEN ASM_REWRITE_TAC[] THEN
  DISCH_THEN SUBST1_TAC THEN MATCH_MP_TAC POLYHEDRON_INTERS THEN
  REWRITE_TAC[FORALL_IN_GSPEC] THEN
  ONCE_REWRITE_TAC[SIMPLE_IMAGE_GEN] THEN
  ASM_SIMP_TAC[FINITE_IMAGE; FINITE_RESTRICT] THEN
  REPEAT STRIP_TAC THEN REWRITE_TAC[IMAGE_ID] THEN
  MATCH_MP_TAC POLYHEDRON_INTER THEN
  ASM_REWRITE_TAC[POLYHEDRON_HYPERPLANE]);;
```

Could also consider reordering of lemmas and dependencies.

Automatic generalization of proofs

Can AI actually discover more general versions of theorems? This is getting more into the realm of science fiction.

Automatic generalization of proofs

Can AI actually discover more general versions of theorems? This is getting more into the realm of science fiction.

Yet even here, Urban and Kaliszyk's automated probabilistic parser and prover made a discovery. . .

Automatic generalization of proofs

Can AI actually discover more general versions of theorems? This is getting more into the realm of science fiction.

Yet even here, Urban and Kaliszyk's automated probabilistic parser and prover made a discovery. . .

One Multivariate theorem `MATRIX_NEG_NEG`, originally stated for square matrices, generalizes to arbitrary rectangular ones.

Automatic generalization of proofs

Can AI actually discover more general versions of theorems? This is getting more into the realm of science fiction.

Yet even here, Urban and Kaliszyk's automated probabilistic parser and prover made a discovery…

One Multivariate theorem `MATRIX_NEG_NEG`, originally stated for square matrices, generalizes to arbitrary rectangular ones.

This is a real example, but is admittedly relatively trivial. There is scope for *much* more, and the Multivariate library makes the perfect target.

Generalizing from $\mathbb{R}^n$

Many theorems are developed for the concrete setting of Euclidean spaces $\mathbb{R}^n$ (for convenience, simplicity or immediate applicability), but they often hold in some more general structure, e.g.

Generalizing from $\mathbb{R}^n$

Many theorems are developed for the concrete setting of Euclidean spaces $\mathbb{R}^n$ (for convenience, simplicity or immediate applicability), but they often hold in some more general structure, e.g.

- In any vector space, normed vector space, or Hilbert space.
- In any normed space (vector space with a 'norm')
- In any inner product space (vector space with an 'inner product')
- In any metric space
- In any topological space

There are also intermediate possibilities like 'any Hausdorff space' or 'any separable metric space'.

Metric spaces

A metric space is a set X together with a 'distance' function $d : X \times X \to \mathbb{R}$ satisying these properties:

- $\forall x, y \in X.\, 0 \leq d(x, y)$
- $\forall x, y \in X.\, d(x, y) = 0 \Leftrightarrow x = y$
- $\forall x, y \in X.\, d(x, y) = d(y, x)$
- $\forall x, y, z \in X.\, d(x, z) \leq d(x, y) + d(y, z)$ ('the triangle law')

Metric spaces

A metric space is a set X together with a ‘distance’ function $d : X \times X \to \mathbb{R}$ satisying these properties:

- $\forall x, y \in X.\, 0 \leq d(x, y)$
- $\forall x, y \in X.\, d(x, y) = 0 \Leftrightarrow x = y$
- $\forall x, y \in X.\, d(x, y) = d(y, x)$
- $\forall x, y, z \in X.\, d(x, z) \leq d(x, y) + d(y, z)$ (‘the triangle law’)

The classic example is the Euclidean distance in $\mathbb{R}^n$:
$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_n - y_n)^2}$
and in particular $d(x, y) = |x - y|$ over $\mathbb{R}$. Many analytical theorems originally stated for these special metrics are valid for any metric.

Metric spaces

A metric space is a set X together with a 'distance' function $d : X \times X \to \mathbb{R}$ satisying these properties:

- $\forall x, y \in X.\ 0 \leq d(x, y)$
- $\forall x, y \in X.\ d(x, y) = 0 \Leftrightarrow x = y$
- $\forall x, y \in X.\ d(x, y) = d(y, x)$
- $\forall x, y, z \in X.\ d(x, z) \leq d(x, y) + d(y, z)$ ('the triangle law')

The classic example is the Euclidean distance in $\mathbb{R}^n$:
$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_n - y_n)^2}$
and in particular $d(x, y) = |x - y|$ over $\mathbb{R}$. Many analytical theorems originally stated for these special metrics are valid for any metric.

HOL Light's Multivariate library has quite a bit of infrastructure and some theorems already proved for metric spaces (mainly by Marco Maggesi), but it would be nice to *automatically* generalize more Euclidean theorems.

Metrics in HOL Light (Maggesi)

A metric on an arbitrary subset of a type α is conveniently encapsulated as a type α metric, where

Metrics in HOL Light (Maggesi)

A metric on an arbitrary subset of a type α is conveniently encapsulated as a type α metric, where

- mdist $M\ (x, y)$ gives the distance between x and y in metric M.
- mspace M gives the subset of α on which the metric M is defined.

Metrics in HOL Light (Maggesi)

A metric on an arbitrary subset of a type α is conveniently encapsulated as a type α metric, where

- mdist $M\ (x, y)$ gives the distance between x and y in metric M.
- mspace M gives the subset of α on which the metric M is defined.

The special case `euclidean_metric` gives the usual distance function dist so

$$\text{mdist}\ \texttt{euclidean\_metric}\ (x, y) = \text{dist}\ (x, y)$$

and similarly for `real_euclidean_metric` which gives the usual metric on $\mathbb{R}$ (which is not identical with $\mathbb{R}^1$ in our formulation).

Generalizing Euclidean theorems to metric spaces

One can often generalize Euclidean theorems to the metric setting as follows:

Generalizing Euclidean theorems to metric spaces

One can often generalize Euclidean theorems to the metric setting as follows:

- (Always) replace each $\text{dist} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ with a general metric $\text{mdist}\ \mathsf{M}_n$.

Generalizing Euclidean theorems to metric spaces

One can often generalize Euclidean theorems to the metric setting as follows:

- (Always) replace each $\mathsf{dist} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ with a general metric $\mathsf{mdist}\ \mathsf{M}_n$.
- (Usually) add membership constraints $x \in \texttt{mspace}\ \mathsf{M}_n$ for points originally in $\mathbb{R}^n$, since in general a metric is not defined on the whole type.

Generalizing Euclidean theorems to metric spaces

One can often generalize Euclidean theorems to the metric setting as follows:

- (Always) replace each $\text{dist} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ with a general metric $\text{mdist}\ \mathsf{M}_n$.
- (Usually) add membership constraints $x \in \texttt{mspace}\ \mathsf{M}_n$ for points originally in $\mathbb{R}^n$, since in general a metric is not defined on the whole type.
- (Sometimes) eliminate use of the special point 0, perhaps replacing it with an arbitrary point (after a case split for the empty metric?)

Generalizing Euclidean theorems to metric spaces

One can often generalize Euclidean theorems to the metric setting as follows:

- (Always) replace each dist : $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ with a general metric mdist M_n.
- (Usually) add membership constraints $x \in$ `mspace` M_n for points originally in $\mathbb{R}^n$, since in general a metric is not defined on the whole type.
- (Sometimes) eliminate use of the special point 0, perhaps replacing it with an arbitrary point (after a case split for the empty metric?)

There are already quite a number of general metric theorems where the Euclidean forms are derived as special cases, so the relationship may well be learnable!

Concept/connection discovery?

So it may well be feasible to generalize theorems automatically to a more general setting.

Concept/connection discovery?

So it may well be feasible to generalize theorems automatically to a more general setting.

But could AI actually discover the more general setting, e.g. *invent* the concept of a metric space?

Concept/connection discovery?

So it may well be feasible to generalize theorems automatically to a more general setting.

But could AI actually discover the more general setting, e.g. *invent* the concept of a metric space?

- Metric spaces were first introduced by Fréchet after careful experimentation with different axioms
- There are structures that drop one or other (pseudometric space, quasimetric space) or strengthen them (ultrametric spaces)
- AI methods might actually be able to do a more efficient job than people on assessing which axioms seem to be needed where

Concept/connection discovery?

So it may well be feasible to generalize theorems automatically to a more general setting.

But could AI actually discover the more general setting, e.g. *invent* the concept of a metric space?

- Metric spaces were first introduced by Fréchet after careful experimentation with different axioms
- There are structures that drop one or other (pseudometric space, quasimetric space) or strengthen them (ultrametric spaces)
- AI methods might actually be able to do a more efficient job than people on assessing which axioms seem to be needed where

Discovering less 'obviously similar' concepts like topological spaces seems to be a tall order as one needs to replace metric reasoning with just reasoning about open sets, and so reshape proofs significantly.

Questions?