Machine Verification of the No-FTL-Observer Theorem for First-Order General Relativity

Andréka, Higgins, Madarász, Németi, Stannett & Székely

▶Ancient history

Etesi & Németi / Hogarth

Andréka-Németi Group

uncomputable functions might actually be computable

computer

user

suitable version of spacetime

FOL is a really good tool for investigating relativity theories

Isabelle/HOL Theorem Prover

coffee-break chat at HyperNet many aeons ago

Thank you, EPSRC!

Sheffield has many Isabelle enthusiasts

convincing proof needed

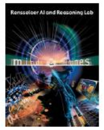
Roughly 10 years ago

Proof Verification and Proof Discovery for Relativity

Naveen Sundar G. • **Selmer Bringsjord** • Joshua Taylor
Department of Computer Science
Department of Cognitive Science
Rensselaer AI & Reasoning (RAIR) Lab
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA
govinn@rpi.edu
Budapest 9/12/12

Rensselaer DEPARTMENT OF COGNITIVE SCIENCE

Rensselaer Computer Science



Sunday, September 23, 12

SpecRel.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\SR--No_FTL_observers\)

```
theorem noFTLObserver:
  assumes iobm:  "IOb m"
  and     iobk:  "IOb k"
  and     mke:   "m sees k at e"
  and     mkf:   "m sees k at f"
  and     enotf: "e ≠ f"
shows          "space2 e f ≤ (c m * c m) * time2 e f"
proof -   (* by reductio *)

(* Step 1: Suppose k is going FTL from m
{
  assume converse: "space2 e f > (c m *

(* Step 2: Consider the m-lightcone at e
  define eCone where "eCone = mkCone e (
  have e_on_econe: "onCone e eCone" by (


(* Step 3: There is a tangent plane for
   defined using some point g on the tan

  have e_is_vertex: "e = vertex eCone" b
  have cm_is_slope: "c m = slope eCone"
  hence outside: "outsideCone f eCone"
    by (metis (lifting) e_is_vertex cm_i
```

arXiv.org > cs > arXiv:1211.6468v2

Computer Science > Logic in Computer Science

[Submitted on 27 Nov 2012 (v1), last revised 18 Jan 2013 (this version, v2)]

Using Isabelle to verify special relativity, with application to hypercom

Mike Stannett, István Németi

Logicians at the Rényi Mathematical Institute in Budapest have spent several years developing versions of relativity theory (special, general, and other variants) based wholly on first order logic, and have argued in favour of the physical decidability, via exploitation of cosmological phenomena, of formally undecidable questions such as the Halting Problem and the consistency of set theory.
The Hungarian theories are very extensive, and their associated proofs are intuitively very satisfying, but this brings its own risks since intuition can sometimes be misleading. As part of a joint project, researchers at Sheffield have recently started generating rigorous machine-verified versions of the Hungarian proofs, so as to demonstrate the soundness of their work. In this paper, we explain the background to the project and demonstrate an Isabelle proof of the theorem "No inertial observer can travel faster than light".
This approach to physical theories and physical computability has several pay-offs: (a) we can be certain our intuition hasn't led us astray (or if it has, we can identify where this has happened); (b) we can identify which axioms are specifically required in the proof of each theorem and to what extent those axioms can be weakened (the fewer assumptions we make up-front, the stronger the results); and (c) we can identify whether new formal proof techniques and tactics are needed when tackling physical as opposed to mathematical theories.

Comments: 14 pages, reformatted with minor corrections
Subjects: **Logic in Computer Science (cs.LO)**; General Relativity and Quantum Cosmology (gr-qc)
ACM classes: F.4.1; J.2
Journal reference: Journal of Automated Reasoning, 52,4 (2014), 361-378
DOI: 10.1007/s10817-013-9292-7

Outline the nature and purpose of your research project including a description of the experimental methods and techniques you will be using (max 4000 characters including spaces)

This proposal concerns the initial stages of the following longer-term strategy for investigating the nature of computation in a relativistic setting.

STRATEGY

[1] Implement the (already developed) many-sorted first-order logic theories of special and general relativity (SpecRel/GenRel) in a mechanised theorem prover. We propose using Isabelle/HOL, one of the best developed and documented systems. A particular advantage is that external automatic theorem provers can be called, thereby enabling parts of the formalisation to be proved fully automatically.

[2] Having implemented the relativistic theories, a basic theory of mobile computation in spacetime should be developed. We propose using membrane system representations, since these include a natural representation of spatial separation and most of the models are Turing complete.

[3] Merge these to generate machine verifiable consistent theories of relativistic computation, and use them to prove the feasibility of hypercomputation in selected (realistic) models of general relativity.

At this initial stage we will

[WP1] Implement axiomatizations of general relativity in first order logic in Isabelle. Our particular focus will be the many-sorted theories SpecRel, AccRel and GenRel [AMNSz], which fully encapsulate the basic theories of special relativity, relativity with accelerated observers, and general relativity, in the sense that the logical models corresponding to these theories are precisely the Lorentzian manifolds posited by theoretical physicists. Suitable background theories - e.g. ordered Euclidean fields (in the algebraic sense) - will also be encoded and published in a repository.

[WP2] Review the very rich literature of membrane systems [PRS] to identify variants of the model that are suitable for our purposes. Mobility appears in various forms, and allows us to model the movement of computational devices. We will define suitable geometric properties for these variants, so as to model the effects of cosmological curvature identified in WP1.

[WP3] Study the computational power and complexity aspects of the model defined in WP2. Various topologies and geometries will be considered.

[WP4] Develop a detailed case study. We will select an uncomputable problem P - for example, the Halting Problem, or the consistency of set theory - and attempt to prove and machine-verify the following claim: in simpler relativistic settings, P remains uncomputable, but when more complicated (and more relativistic) spacetimes are considered, P can be solved. This will confirm that the computational power of a device depends on the physical setting in which it finds itself.

THE TEAM

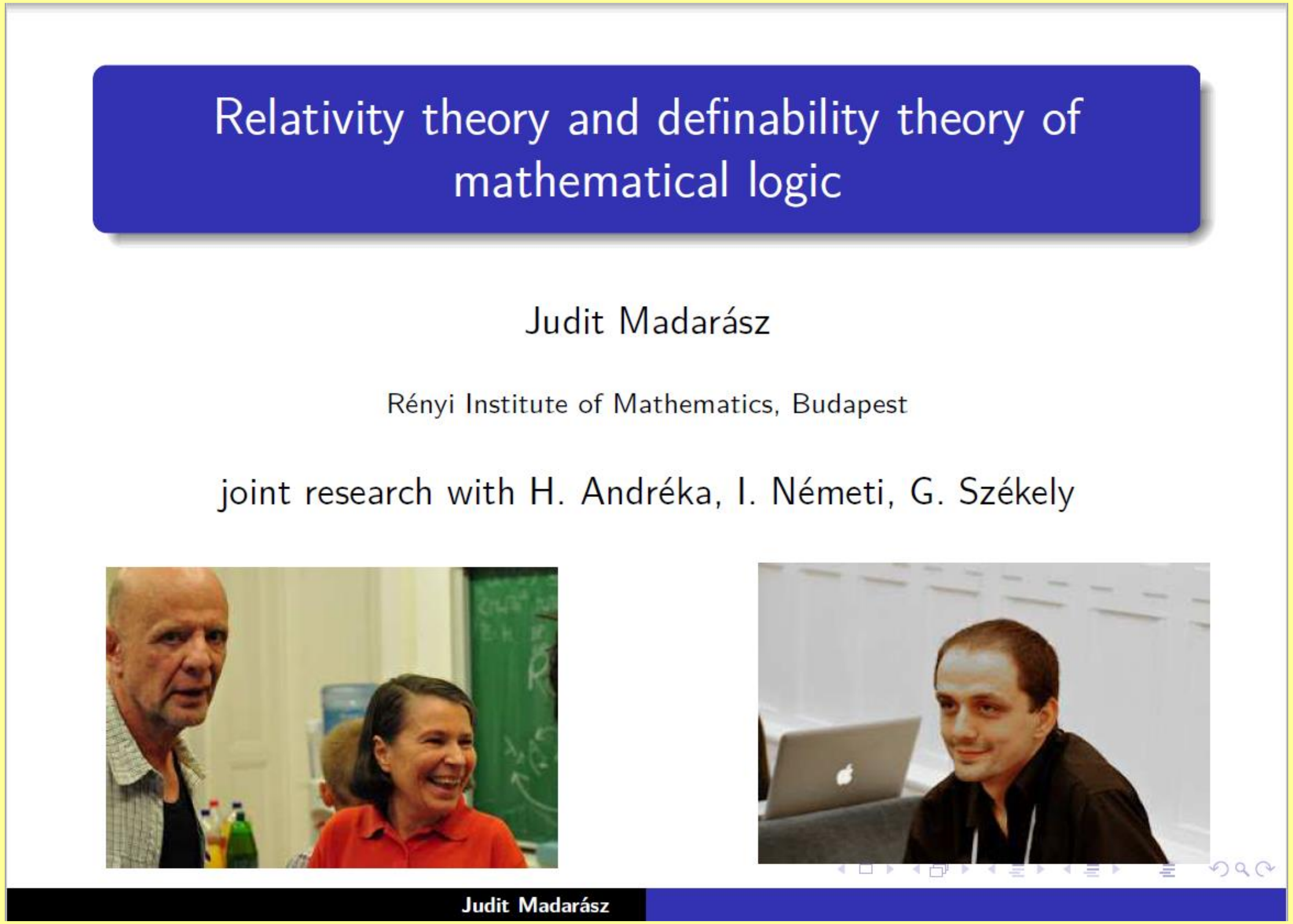
Andréka, Madarász, Németi and Székely are experts in the logics involved, and developed the underlying theories on which the project is based. Stannett has worked with the Budapest group, and is well-versed in their logics; he is, moreover, an expert on hypercomputation theory. Struth and Foster have expertise in interactive and automatic theorem provers, and Struth has a background in theoretical physics. Gheorghe is an expert in membrane computing and other relevant computational models.

The nominated PhD students all work in topics directly related to the project, and will receive appropriate further training.

Because of their complementary skills, the participants are ideally placed to pursue this project.

Many thanks to the Royal Society for the funding

Main sources for current work



Presented at
Workshop: The Formal Semantics of Theories: Conceptual and Historical Foundations, University of Salzburg, 7-8 June 2018

Current position

New GenRel Proof

- 16 files
- 6000 lines so far

Tidied-Up SpecRel Proof

- 4 files
- 1500 lines

SpecRel Proof Files	Lines
SpaceTime	839
SomeFunc	26
Axioms	267
SpecRel	363

GenRel Proof Files	Lines
Sorts	487
Points	639
Functions	262
Norms	238
Vectors	164
Matrices	51
WorldView	40
Affine	1345
Sublemma4	132
WorldLine	429
GenRel	132
Sublemma3	435
MainLemma	709
PresentationLemma	270
Cones	416
GenRelNoFTL (incomplete)	296

mike's laptop
2 cores, 16GB

What an Isabelle proof looks like (roughly)

SpecRel.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\SR--No_FTL_observers\)

```
theorem noFTLObserver:
  assumes iobm:  "IOb m"
  and     iobk:  "IOb k"
  and     mke:   "m sees k at e"
  and     mkf:   "m sees k at f"
  and     enotf: "e ≠ f"
shows     "space2 e f ≤ (c m * c m) * time2 e f"
proof -   (* by reductio *)

(* Step 1: Suppose k is going FTL from m's viewpoint. *)
{
  assume converse: "space2 e f > (c m * c m) * time2 e f"


(* Step 2: Consider the m-lightcone at e *)
  define eCone where "eCone = mkCone e (c m)"
  have e_on_econe: "onCone e eCone" by (simp add: eCone_def)



(* Step 3: There is a tangent plane for eCone containing both e and f,
   defined using some point g on the tangent line  *)

  have e_is_vertex: "e = vertex eCone" by (simp add: eCone_def)
  have cm_is_slope: "c m = slope eCone" by (simp add: eCone_def)
  hence outside: "outsideCone f eCone"
    by (metis (lifting) e_is_vertex cm_is_slope converse outsideCone.simps)
```

1. Define all of your terms (takes ages)
2. Prove basic mathematical statements as necessary
3. Name the result
4. State the assumptions
5. State the result
6. Write out the proof (help is available)
7. Remember to include comments for humans

Some stuff can be inherited from existing theories

Sorts.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\)

```


(* A linordered_field is a field with a linear (total) order
  INHERITED SYNTAX:
    linorder: <, ≤, ≥, >
    field: a * b, a / b, inverse a
           a + b, a - b, -a
           0, 1
*)


(*
  The set of quantities is assumed to be an ordered field. We may
  sometimes need to assume that the field is also Euclidean, ie
  square roots exists, but this is not a general requirement so it
  will be added as a separate axiom class later if it is needed.
*)
class Quantities = linordered_field
begin

```

No need to define what an ordered field is, as lots of stuff has already been proven about them

And other stuff you may need to define yourself

```
abbreviation affine :: "('a Point ⇒ 'a Point) ⇒ bool"
  where "affine A ≡ ∃ L T . (linear L) ∧ (translation T) ∧ (A = T ∘ L)"


abbreviation isLinearPart :: "('a Point ⇒ 'a Point) ⇒ ('a Point ⇒ 'a Point) ⇒ bool"
  where "isLinearPart A L ≡ (affine A) ∧ (linear L) ∧
          (∃ T. (translation T ∧ A = T ∘ L))"

abbreviation isTranslationPart  :: "('a Point ⇒ 'a Point) ⇒ ('a Point ⇒ 'a Point) ⇒ bool"
  where "isTranslationPart A T ≡ (affine A) ∧ (translation T) ∧
          (∃ L. (linear L ∧ A = T ∘ L))"

abbreviation affInvertible :: "('a Point ⇒ 'a Point) ⇒ bool"
  where "affInvertible A ≡ ∀ q . (∃ p . (A p = q) ∧ (∀x. A x = q ⟶ x = p))"

(* affine approximation *)
abbreviation affineApprox :: "('a Point ⇒ 'a Point) ⇒
                             ('a Point ⇒ 'a Point => bool) ⇒
                              'a Point ⇒ bool"
  where "affineApprox A f x ≡ (isFunction f) ∧
```

Some proofs are very simple

Sorts.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\)

```
abbreviation sqr :: "'a ⇒ 'a"
  where "sqr x ≡  x*x"

abbreviation hasRoot :: "'a ⇒ bool"
  where "hasRoot x ≡ ∃ r . x = sqr r"

abbreviation isNonNegRoot :: "'a ⇒ 'a ⇒ bool"
  where "isNonNegRoot x r ≡ (r ≥ 0) ∧ (x = sqr r)"

abbreviation hasUniqueRoot :: "'a ⇒ bool"
  where "hasUniqueRoot x ≡ ∃! r . isNonNegRoot x r"

lemma lemAbsIsRootOfSquare: "isNonNegRoot (sqr x) (abs x)"
  by simp



lemma lemSqrt:
  assumes "hasRoot x"
  shows   "hasUniqueRoot x"
proof -
  obtain r where "x = sqr r" using assms(1) by auto
  define rt where "rt = (if (r ≥ 0) then r else (-r))"
  hence rt: "rt ≥ 0 ∧ sqr rt = x" using rt_def ‹x = sqr r› by auto
  hence rtroot: "isNonNegRoot x rt" by auto

  { fix y
```

In general a proof has many steps, and you have to prove every single step in full, no matter how trivial.

Isabelle has some basic proof methods built in; you have to decide which one to use at each stage, e.g. “simp” and “auto” use basic rewrite and inference rules to check that the claimed result holds.

Sledgehammer

If you want some help, try invoking "sledgehammer"

Points.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\)

```
lemma lemScaleAssoc: "(α ⊗ (β ⊗ p)) = ((α * β) ⊗ p)"
  sledgehammer
```

Isabelle will ask various theorem provers to try finding a proof for you

Provers: cvc4 z3 spass e remote_vampire ☑ Isar proofs ☑ Try methods

```
Proof found...
"cvc4": Try this: by (simp add: mult_assoc) (0.0 ms)
"z3": Try this: by (simp add: local.mult.semigroup_axioms semigroup.assoc) (0.0 ms)
```

they can use proven results you don't know about

```
Isar proof (15 ms):
proof -
  have "∀f a aa ab. ¬ semigroup f ∨ f (f (a::'a) aa) ab = f a (f aa ab)"
    by (meson semigroup.assoc)
  then show ?thesis
    using local.mult.semigroup_axioms by auto
qed
"e": Try this: by (simp add: mult_assoc) (0.0 ms)
(No Isar proof available.)
"remote_vampire": The prover gave up
"spass": Try this: by (simp add: mult_assoc) (0.0 ms)
(No Isar proof available.)
```

the proofs different systems generate might be the same or different (and they may not be able to find one at all)

You may still need to provide detailed guidance

Sorts.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\)

```
abbreviation sqr :: "'a ⇒ 'a"
  where "sqr x ≡  x*x"

abbreviation hasRoot :: "'a ⇒ bool"
  where "hasRoot x ≡ ∃ r . x = sqr r"

abbreviation isNonNegRoot :: "'a ⇒ 'a ⇒ bool"
  where "isNonNegRoot x r ≡ (r ≥ 0) ∧ (x = sqr r)"

abbreviation hasUniqueRoot :: "'a ⇒ bool"
  where "hasUniqueRoot x ≡ ∃! r . isNonNegRoot x r"

lemma lemAbsIsRootOfSquare: "isNonNegRoot (sqr x) (abs x)"
  by simp



lemma lemSqrt:
  assumes "hasRoot x"
  shows   "hasUniqueRoot x"
proof -
  obtain r where "x = sqr r" using assms(1) by auto
  define rt where "rt = (if (r ≥ 0) then r else (-r))"
  hence rt: "rt ≥ 0 ∧ sqr rt = x" using rt_def ‹x = sqr r› by auto
  hence rtroot: "isNonNegRoot x rt" by auto

  { fix y
```

"hasRoot" says a root exists

I use "obtain" to generate a witness

I manipulate the witness to obtain a value I think will have some required properties…

… and then guide Isabelle through a proof that I'm right.

Proof development process for No-FTL-GR

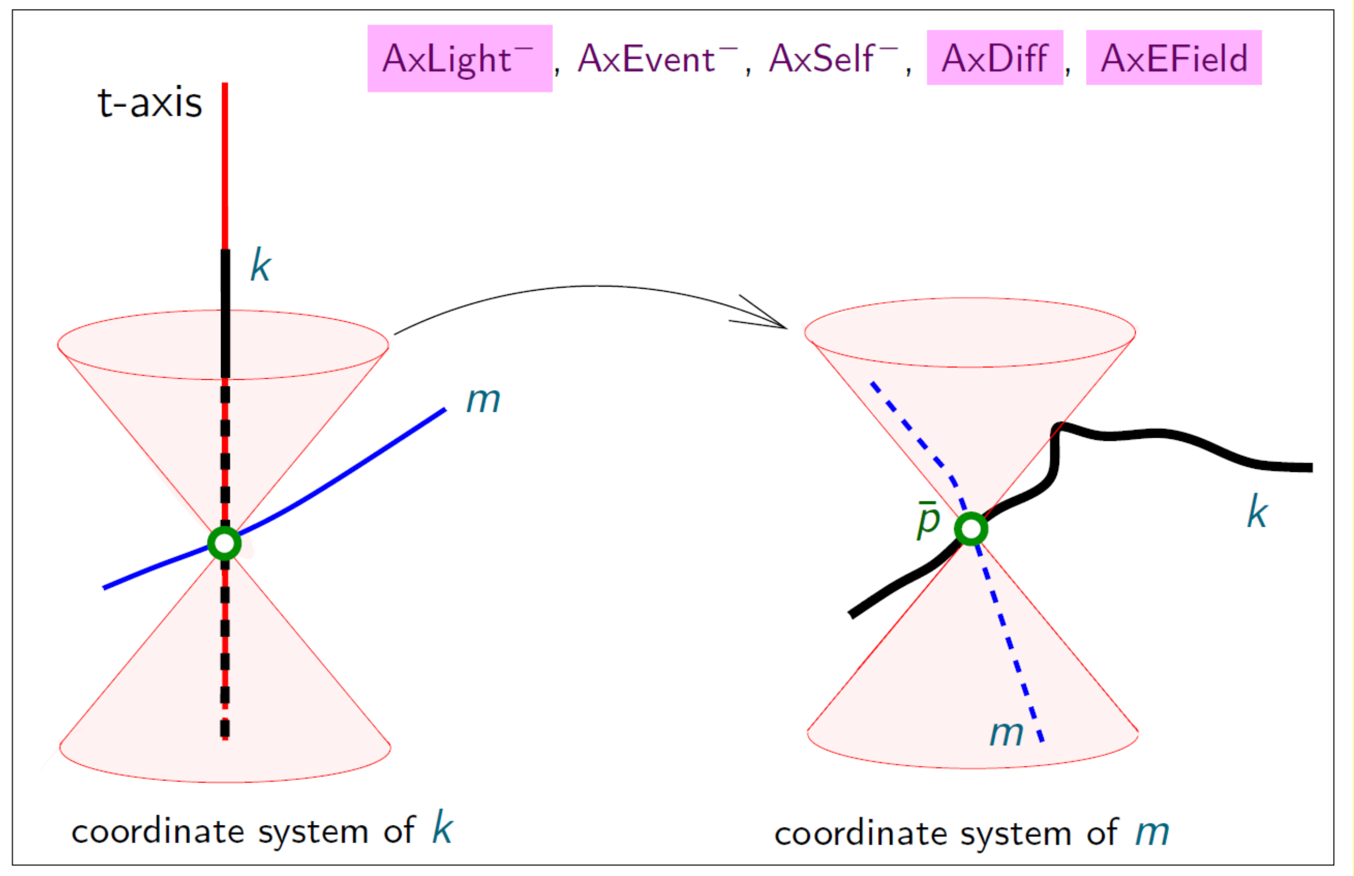
- Background definitions and general ideas taken from earlier Andréka-Németi group presentations
- Additional hand-written proofs specially provided by Judit (thank you!)
- Conversion to Isabelle mostly done by Mike (some by Edward, thank you!)
- Gaps in proofs mostly dealt with by Mike (liaising sometimes with Budapest)

Obvious with Hindsight

Every stage in this process requires considerable invention and intuition

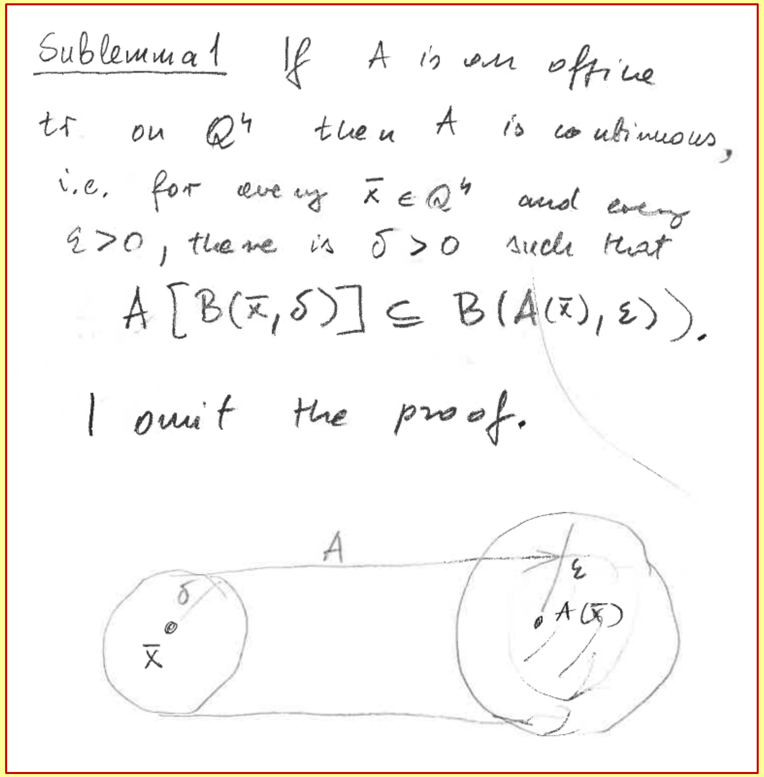
▶Re-using earlier slides…

- Reliance on images
- Translation into written mathematics not obvious



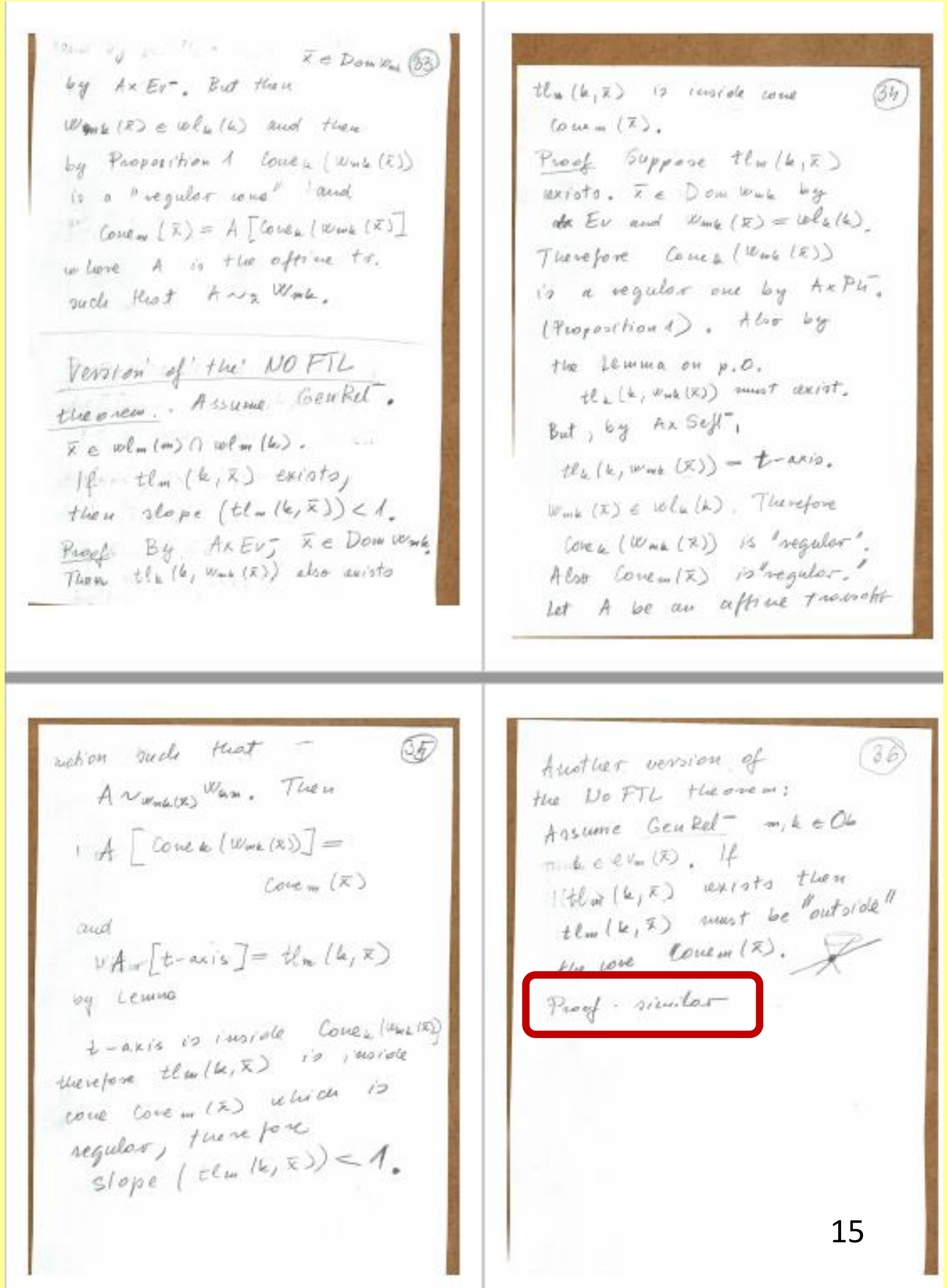
Hand-written proofs…

- Much easier to convert
- Still requires intuition to fill gaps


Sublemma 1 If A is an affine tr. on Q^4 then A is continuous, i.e. for every $\bar{x} \in Q^4$ and every $\varepsilon > 0$, there is $\delta > 0$ such that

$$A[B(\bar{x},\delta)] \subseteq B(A(\bar{x}),\varepsilon).$$

I omit the proof.


33

… $\bar{x} \in Dom\, w_{mk}$ by AxEv⁻. But then $w_{mk}(\bar{x}) \in wl_k(k)$ and then by Proposition 1 $Cone_k(w_{mk}(\bar{x}))$ is a "regular cone" and $Cone_m(\bar{x}) = A[Cone_k(w_{mk}(\bar{x}))]$ where A is the affine tr. such that $A \sim_{\bar{x}} w_{mk}$.

Version of the NO FTL theorem. Assume GenRel⁻. $\bar{x} \in wl_m(m) \cap wl_m(k)$. If $tl_m(k,\bar{x})$ exists, then $slope(tl_m(k,\bar{x})) < 1$.

Proof By AxEv⁻ $\bar{x} \in Dom\, w_{mk}$. Then $tl_k(k, w_{mk}(\bar{x}))$ also exists

34

$tl_m(k,\bar{x})$ is inside cone $Cone_m(\bar{x})$.

Proof Suppose $tl_m(k,\bar{x})$ exists. $\bar{x} \in Dom\, w_{mk}$ by AxEv and $w_{mk}(\bar{x}) = wl_k(k)$. Therefore $Cone_k(w_{mk}(\bar{x}))$ is a regular one by AxPh⁻, (Proposition 1). Also by the Lemma on p.0. $tl_k(k, w_{mk}(\bar{x}))$ must exist. But, by AxSelf⁻, $tl_k(k, w_{mk}(\bar{x}))$ = t-axis. $w_{mk}(\bar{x}) \in wl_k(k)$. Therefore $Cone_k(w_{mk}(\bar{x}))$ is 'regular'. Also $Cone_m(\bar{x})$ is 'regular.' Let A be an affine transf-

35

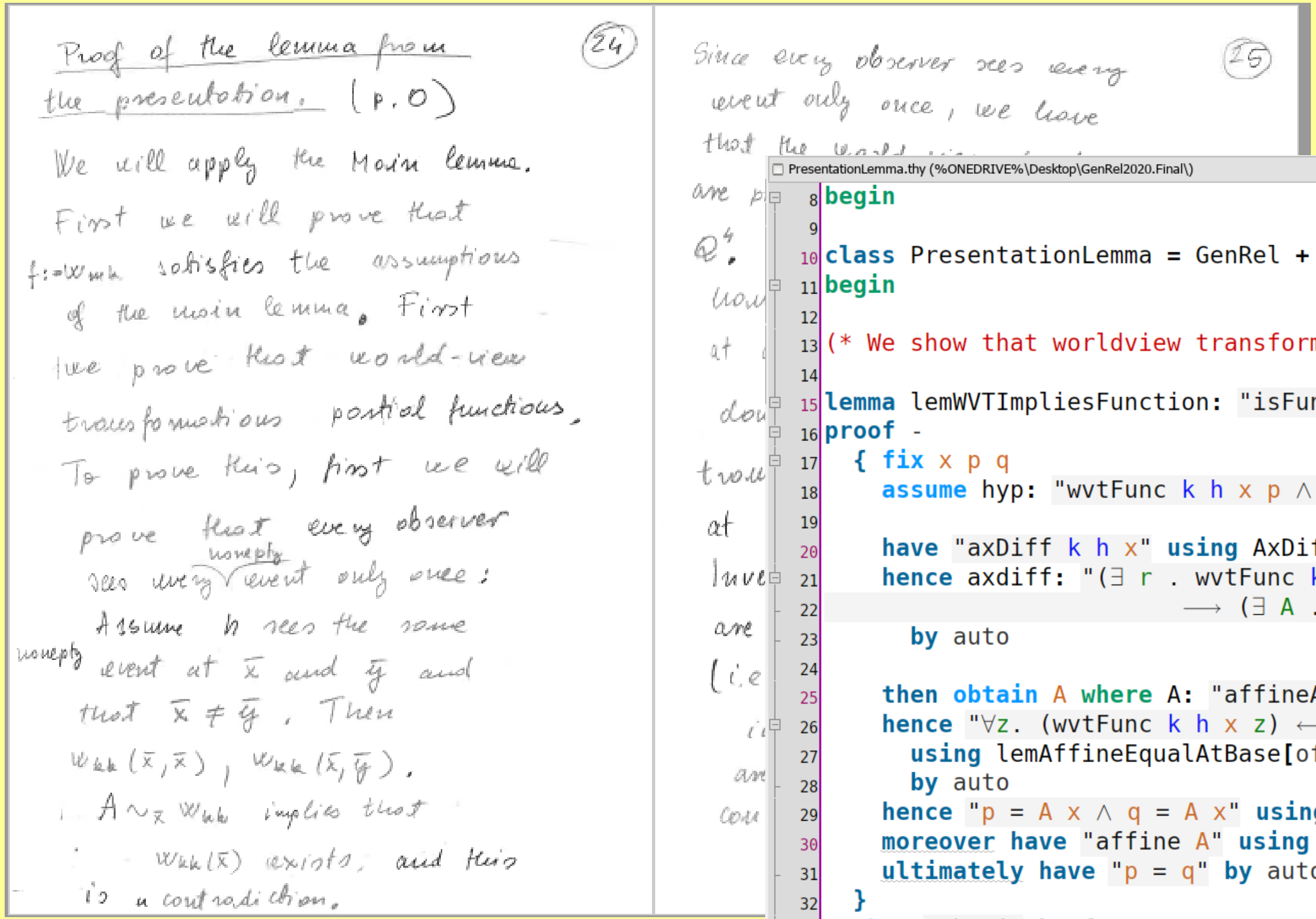
-ation such that $A \sim_{w_{mk}(\bar{x})} w_{km}$. Then $A[Cone_k(w_{mk}(\bar{x}))] = Cone_m(\bar{x})$ and $A[t\text{-axis}] = tl_m(k,\bar{x})$ by Lemma. t-axis is inside $Cone_k(w_{mk}(\bar{x}))$ therefore $tl_m(k,\bar{x})$ is inside cone $Cone_m(\bar{x})$ which is regular, therefore $slope(tl_m(k,\bar{x})) < 1$.

36

Another version of the No FTL theorem: Assume GenRel⁻ $m, k \in Ob$ … If $tl_m(k,\bar{x})$ exists then $tl_m(k,\bar{x})$ must be "outside" the cone $Cone_m(\bar{x})$.

Proof: similar

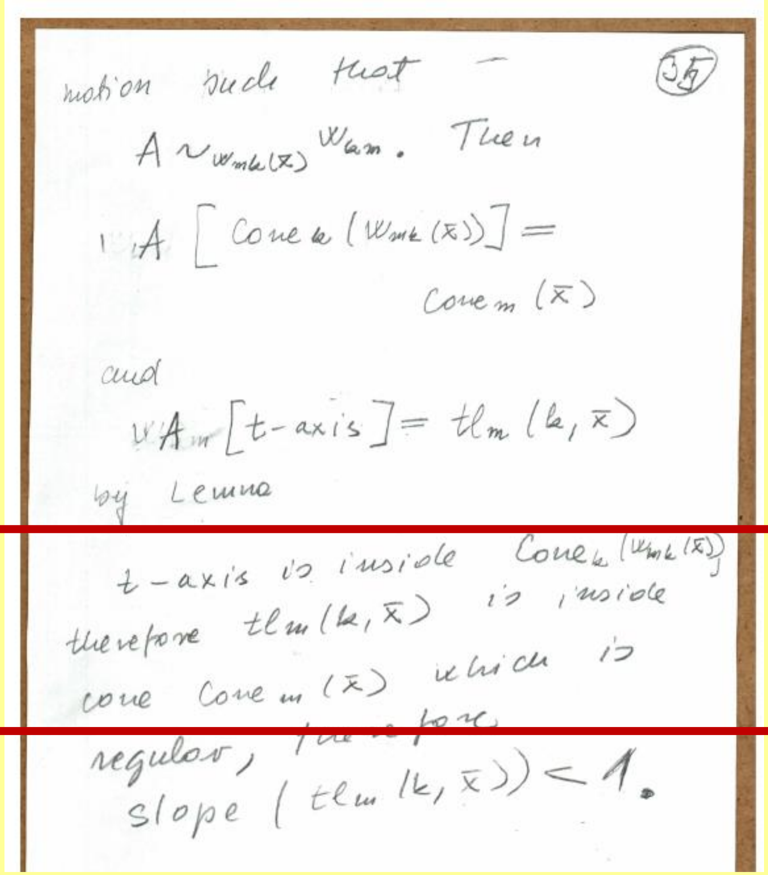
▶Line-by-line conversion…



PresentationLemma.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\)

```
begin

class PresentationLemma = GenRel + MainLemma
begin

(* We show that worldview transformations satisfy the requirements for lemMainLemma *)

lemma lemWVTImpliesFunction: "isFunction (wvtFunc k h)"
proof -
  { fix x p q
    assume hyp: "wvtFunc k h x p ∧ wvtFunc k h x q"

    have "axDiff k h x" using AxDiff by blast
    hence axdiff: "(∃ r . wvtFunc k h x r)
                        ⟶ (∃ A . (affineApprox A (wvtFunc k h) x ))"
      by auto

    then obtain A where A: "affineApprox A (wvtFunc k h) x" using hyp by auto
    hence "∀z. (wvtFunc k h x z) ⟷ (z = A x)"
      using lemAffineEqualAtBase[of "wvtFunc k h" "A" "x"]
      by auto
    hence "p = A x ∧ q = A x" using hyp by blast
    moreover have "affine A" using A by auto
    ultimately have "p = q" by auto
  }
  thus ?thesis by force
qed

```

Some gaps might or might not need filling



notion such that — (35)

$A \sim_{w_{mk}(\bar{x})} w_{km}$. Then

$A[\mathrm{Cone}_k(w_{mk}(\bar{x}))] = \mathrm{Cone}_m(\bar{x})$

and

$A[t\text{-axis}] = tl_m(k, \bar{x})$

by Lemma

t-axis is inside $\mathrm{Cone}_k(w_{mk}(\bar{x}))$

therefore $tl_m(k, \bar{x})$ is inside

cone $\mathrm{Cone}_m(\bar{x})$ which is

regular, therefore

$\mathrm{slope}(tl_m(k, \bar{x})) < 1$.

Cones.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\)

```
(*
(* A point is inside (regularConeSet m x) *)
abbreviation insideCone :: "'a Point ⇒ 'a Point ⇒ bool"
  where "insideCone x p ≡ (∃ v ∈ lineVelocity (lineJoining x p) . sNorm2 v ≤ 1)"

(* Two points are on the same half of (regularConeSet m x) *)
abbreviation sameHalf :: "'a Point ⇒ 'a Point ⇒ 'a Point ⇒ bool"
  where "sameHalf x u v ≡
    (∀ α . ((v⊖x) = (α ⊗ (u⊖x)) ⟶ α > 0)) ∧
    (∀ α β . ((α ≤ 1) ∧ (β ≤ 1) ∧ (α + β = 1)) ⟶ (insideCone x ((α⊗u) ⊕ (β⊗v))))"
*)

lemma lemSameHalfUnderInvertibleAffine:
  assumes "affine A ∧ affInvertible A"
and       "y = A x" (* can be deduced from other assumptions *)
and       "applyToSet (asFunc A) (regularConeSet x) = regularConeSet y"
and       "sameHalf x u v"
shows     "sameHalf y (A u) (A v)"
proof -
  define u' where u': "u' = A u"
  define v' where v': "v' = A v"
  obtain T L where TL: "translation T ∧ linear L ∧ A = T∘L" using assms(1) by auto

  obtain A' where A': "(affine A') ∧ (∀ p q . A p = q ⟷ A' q = p)"
    using assms(1) lemInverseAffine[of "A"] by blast
  then obtain T' L' where TL': "translation T' ∧ linear L' ∧ A' = T'∘L'"
    by blast
```

Last sentence of hand-written proof ⟶ *Launchpad for entire secondary proof?*

▶What do the results mean physically?

PresentationLemma.thy (%ONEDRIVE%\Desktop\GenRel2020.Final\)

```
begin

class PresentationLemma = GenRel + MainLemma
begin

(* We show that worldview transformations satisfy the requirements for lemMainLemma *)

lemma lemWVTImpliesFunction: "isFunction (wvtFunc k h)"
proof -
  { fix x p q
    assume hyp: "wvtFunc k h x p ∧ wvtFunc k h x q"

    have "axDiff k h x" using AxDiff by blast
    hence axdiff: "(∃ r . wvtFunc k h x r)
                        ⟶ (∃ A . (affineApprox A (wvtFunc k h) x ))"
      by auto

    then obtain A where A: "affineApprox A (wvtFunc k h) x" using hyp by auto
    hence "∀z. (wvtFunc k h x z) ⟷ (z = A x)"
      using lemAffineEqualAtBase[of "wvtFunc k h" "A" "x"]
      by auto
    hence "p = A x ∧ q = A x" using hyp by blast
    moreover have "affine A" using A by auto
    ultimately have "p = q" by auto
  }
  thus ?thesis by force
qed

```

- Is it reasonable that worldview relations should be functions?

Lessons learned

- Doing this stuff is challenging (= fun)
- Started slowly, but speeding up as we learn to “think like the machine”
- Proof conversion requires the programmer to have an intuitive grasp of the subject matter
  - should aim to help the author prove things directly in the system rather than need help from a translator
- There are lots of basic mathematical assumptions built into proofs
  - need to equip the theorem prover to degree-level standard

And finally…

- Original goal still seems achievable
  - unfunded, will take many years to complete)
- New added focus
  - examine the difficulties involves in converting “mathematical physics” proofs into machine-verifiable format
  - develop software support to make this easier
  - generate new automated proof systems targeted at physicists (link up with Bringsjord et al.)