Informal2Formal: Automating Formalization by Statistical and Semantic Parsing of Mathematics

Josef Urban   Cezary Kaliszyk   Jiri Vyskocil
Qingxiang Wang   Chad Brown

Czech Technical University in Prague

University of Innsbruck

Bonn, February 2nd, 2020

European Research Council
Established by the European Commission

Outline

Autoformalization

Demos

PCFG-based Parsing

Neural Parsing

Chad's Remarks

My Remarks and Reactions

Autoformalization

- Goal: Learn understanding of informal math formulas and reasoning
- Experiments with the CYK chart parser linked to semantic methods
- Experiments with neural methods
- Combined with semantic methods: Type checking, theorem proving
- Feedback loops between the learning and the semantic methods
- Math is a much nicer area than unrestricted NLP:
- We (believe we) can express informal math formally, prove things, etc.
- If we achieve grounding math, we might ground scientific texts, law, etc.
- Corpora: Flyspeck, Mizar, Proofwiki, Stacks, Arxiv, etc.
- Isabelle/AFP?, Coq/Feit-Thompson?, Lean/Mathlib?, Naproche/SAD?
- Some aligned corpora - Flyspeck, Feit-Thompson, Compendium of Cont. Lattices, Rewriting and All That; but most not aligned (requires unsupervised MT methods)

Demos

- Inf2formal over HOL Light: `http://grid01.ciirc.cvut.cz/~mptp/demo.ogv`
- Inf2formal over Mizar: `http://grid01.ciirc.cvut.cz/~mptp/t2m/`
- Nearest neighbor search for similar sentences in Arxiv: `http://grid01.ciirc.cvut.cz/~mptp/arxsim.html`
- ForSet – Chad's set theory backend: `https://github.com/JUrban/ForSet/`

Outline

Autoformalization

Demos

PCFG-based Parsing

Neural Parsing

Chad's Remarks

My Remarks and Reactions

Statistical/Semantic Parsing of Informalized HOL

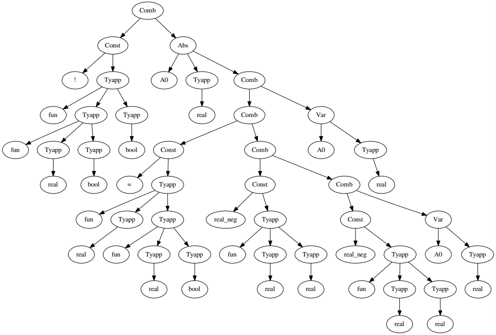
- Training and testing examples exported form Flyspeck formulas
  - Along with their informalized versions
- Grammar parse trees
  - Annotate each (nonterminal) symbol with its HOL type
  - Also “semantic (formal)” nonterminals annotate overloaded terminals
  - guiding analogy: word-sense disambiguation using CYK is common
- Terminals exactly compose the textual form, for example:
- REAL_NEGNEG: $\forall x. - -x = x$

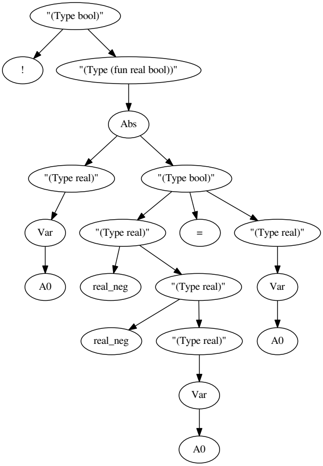
```
(Comb (Const "!" (Tyapp "fun" (Tyapp "fun" (Tyapp "real") (Tyapp "bool"))
(Tyapp "bool"))) (Abs "A0" (Tyapp "real") (Comb (Comb (Const "=" (Tyapp "fun"
(Tyapp "real") (Tyapp "fun" (Tyapp "real") (Tyapp "bool")))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Comb (Const
"real_neg" (Tyapp "fun" (Tyapp "real") (Tyapp "real"))) (Var "A0" (Tyapp
"real"))))) (Var "A0" (Tyapp "real")))))
```

- becomes

```
("(Type bool)" ! ("(Type (fun real bool))" (Abs ("(Type real)"
(Var A0)) ("(Type bool)" ("(Type real)" real_neg ("(Type real)"
real_neg ("(Type real)" (Var A0)))) = ("(Type real)" (Var A0))))))
```

Example grammars





CYK Learning and Parsing (KUV, ITP 17)

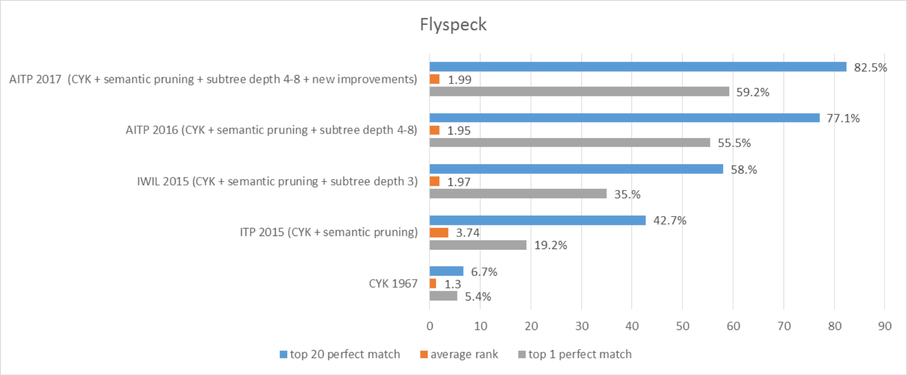
- Induce PCFG (probabilistic context-free grammar) from the trees
  - Grammar rules obtained from the inner nodes of each grammar tree
  - Probabilities are computed from the frequencies
- The PCFG grammar is binarized for efficiency
  - New nonterminals as shortcuts for multiple nonterminals
- CYK: dynamic-programming algorithm for parsing ambiguous sentences
  - input: sentence – a sequence of words and a binarized PCFG
  - output: N most probable parse trees
- Additional semantic pruning
  - Compatible types for free variables in subtrees
- Allow small probability for each symbol to be a variable
- Top parse trees are de-binarized to the original CFG
  - Transformed to HOL parse trees (preterms, Hindley-Milner)
  - typed checked in HOL and then given to an ATP (hammer)

Autoformalization based on PCFG and semantics

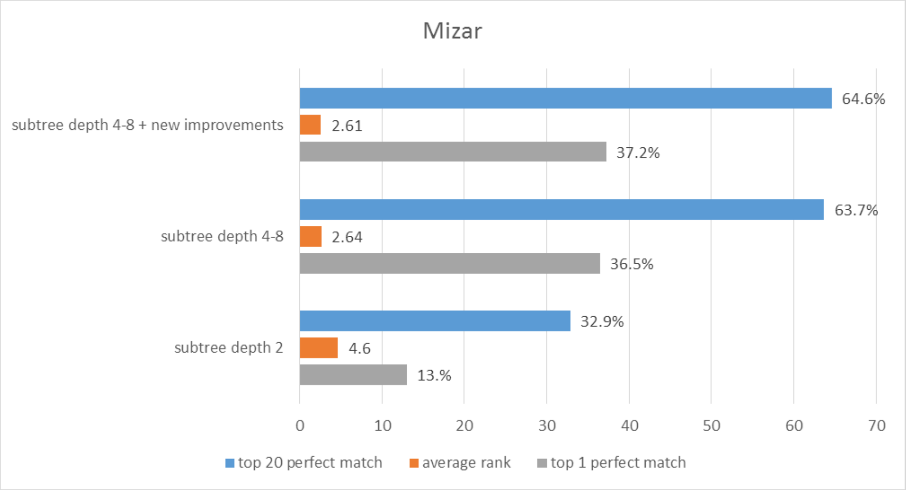
- "`sin ( 0 * x ) = cos pi / 2`"
- produces 16 parses
- of which 11 get type-checked by HOL Light as follows
- with all but three being proved by HOL(y)Hammer
- demo: `http://grid01.ciirc.cvut.cz/~mptp/demo.ogv`

```
sin (&0 * A0) = cos (pi / &2) where A0:real
sin (&0 * A0) = cos pi / &2 where A0:real
sin (&0 * &A0) = cos (pi / &2) where A0:num
sin (&0 * &A0) = cos pi / &2 where A0:num
sin (&(0 * A0)) = cos (pi / &2) where A0:num
sin (&(0 * A0)) = cos pi / &2 where A0:num
csin (Cx (&0 * A0)) = ccos (Cx (pi / &2)) where A0:real
csin (Cx (&0) * A0) = ccos (Cx (pi / &2)) where A0:real^2
Cx (sin (&0 * A0)) = ccos (Cx (pi / &2)) where A0:real
csin (Cx (&0 * A0)) = Cx (cos (pi / &2)) where A0:real
csin (Cx (&0) * A0) = Cx (cos (pi / &2)) where A0:real^2
```

Flyspeck Progress



First Mizar Results (100-fold Cross-validation)



Outline

Neural Autoformalization (Wang et al., 2018,2020)

- generate about 1M Latex - Mizar pairs based on Bancerek's work
- train neural seq-to-seq translation models (Luong – NMT)
- evaluate on about 100k examples
- many architectures tested, some work much better than others
- very important latest invention: *attention* in the seq-to-seq models
- more data very important for neural training – our biggest bottleneck (you can help!)
- Recent addition: unsupervised methods (Lample et all 2018) – no need for aligned data!

Neural Autoformalization data

Rendered LaTeX	If $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.
Mizar	`X c= Y & Y c= Z implies X c= Z;`
Tokenized Mizar	`X c= Y & Y c= Z implies X c= Z ;`
LaTeX	`If $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.`
Tokenized LaTeX	`If $ X \subseteq Y \subseteq Z $ , then $ X \subseteq Z $ .`

Neural Autoformalization results

Parameter	Final Test Perplexity	Final Test BLEU	Identical Statements (%)	Identical No-overlap (%)
128 Units	3.06	41.1	40121 (38.12%)	6458 (13.43%)
256 Units	1.59	64.2	63433 (60.27%)	19685 (40.92%)
512 Units	1.6	**67.9**	66361 (63.05%)	21506 (44.71%)
1024 Units	**1.51**	61.6	**69179 (65.73%)**	**22978 (47.77%)**
2048 Units	2.02	60	59637 (56.66%)	16284 (33.85%)

Neural Fun – Performance after Some Training

Rendered LaTeX	Suppose s_8 is convergent and s_7 is convergent . Then $\lim(s_8+s_7) = \lim s_8 + \lim s_7$
Input LaTeX	`Suppose $ { s _ { 8 } } $ is convergent and $ { s _ { 7 } } $ is convergent . Then $ \mathop { \rm lim } ( { s _ { 8 } } { + } { s _ { 7 } } } ) \mathrel { = } \mathop { \rm lim } { s _ { 8 } } { + } \mathop { \rm lim } { s _ { 7 } } $ .`
Correct	`seq1 is convergent & seq2 is convergent implies lim ( seq1 + seq2 ) = ( lim seq1 ) + ( lim seq2 ) ;`
Snapshot-1000	`x in dom f implies ( x * y ) * ( f | ( x | ( y | ( y | y ) ) ) ) = ( x | ( y | ( y | ( y | y ) ) ) ) ) ;`
Snapshot-2000	`seq is summable implies seq is summable ;`
Snapshot-3000	`seq is convergent & lim seq = 0c implies seq = seq ;`
Snapshot-4000	`seq is convergent & lim seq = lim seq implies seq1 + seq2 is convergent ;`
Snapshot-5000	`seq1 is convergent & lim seq2 = lim seq2 implies lim_inf seq1 = lim_inf seq2 ;`
Snapshot-6000	`seq is convergent & lim seq = lim seq implies seq1 + seq2 is convergent ;`
Snapshot-7000	`seq is convergent & seq9 is convergent implies lim ( seq + seq9 ) = ( lim seq ) + ( lim seq9 ) ;`

Unsupervised NMT Fun on Short Formulas

```
len <* a *> = 1 ;                    len <* a *> = 1 ;
assume i < len q ;                   i < len q ;
len <* q *> = 1 ;                    len <* q *> = 1 ;
s = apply ( v2 , v1 ast t ) ;        s = apply ( v2 , v1 ) . t ;
s . ( i + 1 ) = tt . ( i + 1 )       s . ( i + 1 ) = tau1 . ( i + 1 )
1 + j <= len v2 ;                    1 + j <= len v2 ;
1 + j + 0 <= len v2 + 1 ;            1 + j + 0 <= len v2 + 1 ;
let i be Nat ;                       i is_at_least_length_of p ;
assume v is_applicable_to t ;        not v is applicable ;
let t be type of T ;                 t is_orientedpath_of v1 , v2 , T ;
a ast t in downarrow t ;             a *' in downarrow t ;
t9 in types a ;                      t `2 in types a ;
a ast t <= t ;                       a *' <= t ;
A is_applicable_to t ;               A is applicable ;
Carrier ( f ) c= B                   support ppf n c= B
u in B or u in { v } ;               u in B or u in { v } ;
F . w in w & F . w in I ;            F . w in F & F . w in I ;
GG . y in rng HH ;                   G0 . y in rng ( H1 ./. y ) ;
a * L = Z_ZeroLC ( V ) ;             a * L = ZeroLC ( V ) ;
not u in { v } ;                     u >> v ;
u <> v ;                             u <> v ;
v - w = v1 - w1 ;                    vw = v1 - w1 ;
v + w = v1 + w1 ;                    v + w = v1 + w1 ;
x in A & y in A ;                    assume [ x , y ] in A ;
```

Outline

Chad's remarks

- Naproche/Forthel encoding of notions
- Set vs Type Theory encodings of groups and structures
- `http://grid01.ciirc.cvut.cz/~chad/setsslides.pdf`

Outline

Mizar and Andrzej Trybulec – Unsung Hero of Mathematician-Friendly Formalization

- His motivation: proof checker for refactoring his topology PhD thesis
- Started (1970's) by analyzing a topology paper by H. Patkowska.
- Mizar's proof style and language: Jaskowski (1934) style natural deduction proofs – *On the rules of suppositions*.
- The Discourse Representation Theory motivating Naproche might be its derivative via Montague and Tarski. Jeff Pelletier (Montague's PhD student, funny stories about Montague) wrote a comparison of Gentzen and Jaskowski style ND (good to learn about eigenvariables).
- Obvious inferences: work on the right human-friendly granularity of the proof steps (started by M. Davis, continued by Rudnicki and Trybulec).
- Fast internal proof checker *critical* for library refactoring (not hammers).

Mizar and Andrzej Trybulec

- ENOD: Experience Not Only Doctrine! – build a large math library (in the 90s!) Move away from just the formula/proof language tweaking.
- Math is full of soft types and overloading (200 meanings of + in MML). Mizar's advanced soft types (adjectives and registration) precede Haskell type classes and derived typeclass mechanisms in other ITPs.
- Freek Wiedijk (repeating Andrzej): mathematicians don't use type theory. They use set theory and soft types.
- Jeremy Avigad showing me Gonthier's groups encoding in 2009 - my lack of faith in type theory.
- Freek again: COBOL's too much natural language was a failure – beware.
- Peter Koepke – translating Mizar to German in 2000s (connected to Solovay's Mizar visit in 2002?)

Grzegorz Bancerek, CCL, $\TeX$, ProofWiki

- Grzegorz Bancerek (and team): Compendium of Continuous Lattices (CCL) - Mizar aligned with the book
- I got the tex sources of CCL in 2004 planning to use them for statistical translation to Mizar
- But it took 10 years to declare and start the autoformalization project (CICM 2014)
- Grzegorz's other achievements: translation of Mizar to $\TeX$ and PDF
- If our goal is good controlled language for READING, Mizar + this translation might be a good approach – see the latest Mizar PDFs.
- Also translation of Mizar to ProofWiki (ca 500 ProofWiki pages)
- (Co-)author of 124 Mizar articles (10% of the library)

Some Replies on Automation and Machine Learning

- 2014 AI/TP challenges: `http://ai4reason.org/aichallenges.html`. Unlike the recent PR efforts by the poor Google/Facebook/Microsoft companies and AI teams, I have put my money where my mouth is - you can still bet me.
- Hammers: yes, use strong AI/TP systems to find the proofs, but then refactor the proofs into readable Mizar-style proofs.
- And YES, combinations of ML and AR/TP are very useful and a very cool AI topic.
- And NO, deep learning (even if interesting) has NOT been the critical missing piece for developing AI/TP so far. The largest 40-70% ATP improvements in real time (rlCoP, ENIGMA) are so far done with gradient boosted trees.
- Example Mizar proof (Knaster-Tarski) found by ENIGMA: `http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/knaster#T21`
- Its E-ENIGMA proof: `http://grid01.ciirc.cvut.cz/~mptp/t21_knaster`.

Some References

- C. Kaliszyk, J. Urban, J. Vyskocil: Automating Formalization by Statistical and Semantic Parsing of Mathematics. ITP 2017: 12-27
- Q. Wang, C. Kaliszyk, J. Urban: First Experiments with Neural Translation of Informal to Formal Mathematics. CICM 2018: 255-270
- Qingxiang Wang, Chad E. Brown, Cezary Kaliszyk, Josef Urban: Exploration of neural machine translation in autoformalization of mathematics in Mizar. CPP 2020: 85-98
- Cezary Kaliszyk, Josef Urban, Jirí Vyskocil: Learning to Parse on Aligned Corpora (Rough Diamond). ITP 2015: 227-233
- Cezary Kaliszyk, Josef Urban, Jirí Vyskocil, Herman Geuvers: Developing Corpus-Based Translation Methods between Informal and Formal Mathematics: Project Description. CICM 2014: 435-439
- C. Kaliszyk, J. Urban, H. Michalewski, M. Olsak: Reinforcement Learning of Theorem Proving. NeurIPS 2018: 8836-8847
- T. Gauthier, C. Kaliszyk, J. Urban, R. Kumar, M. Norrish: Learning to Prove with Tactics. CoRR abs/1804.00596 (2018).
- J. Jakubuv, J. Urban: ENIGMA: Efficient Learning-Based Inference Guiding Machine. CICM 2017: 292-302
- Karel Chvalovský, Jan Jakubuv, Martin Suda, Josef Urban: ENIGMA-NG: Efficient Neural and Gradient-Boosted Inference Guidance for E. CADE 2019: 197-215
- Jan Jakubuv, Josef Urban: Hammering Mizar by Learning Clause Guidance. ITP 2019: 34:1-34:8
- L. Czajka, C. Kaliszyk: Hammer for Coq: Automation for Dependent Type Theory. J. Autom. Reasoning 61(1-4): 423-453 (2018)
- J. C. Blanchette, C. Kaliszyk, L. C. Paulson, J. Urban: Hammering towards QED. J. Formalized Reasoning 9(1): 101-148 (2016)
- ARG ML&R course: `http://arg.ciirc.cvut.cz/teaching/mlr19/index.html`
- C. Kaliszyk: `http://cl-informatik.uibk.ac.at/teaching/ss18/mltp/content.php`

Thanks and Advertisement

- Thanks for your attention!
- AITP – Artificial Intelligence and Theorem Proving
- March 22–27, 2020, Aussois, France, `aitp-conference.org`
- ATP/ITP/Math vs AI/Machine-Learning people, Computational linguists
- Discussion-oriented and experimental - submit a talk abstract!
- Grown to 80 people in 2019