Playing with Autoformalization over Mizar and ProofWiki

Grzegorz Bancerek Jiří Vyskočil Chad Brown Josef Urban

Czech Technical University in Prague

AITP 2018, Aussois
March 26, 2018

Two Obstacles to Strong Computer Support for Math

1. Low reasoning power of automated reasoning methods, particularly over large complex theories
2. Lack of computer understanding of current human-level (math and exact science) knowledge

- The two are related: human-level math may require nontrivial reasoning to become fully explained. Fully explained math gives us a lot of data for training AITP systems.
- And we want to train AITP on human-level proofs too. Thus getting interesting formalization/ATP/learning feedback loops.
- In 2014 we have decided that the AITP/hammer systems are getting strong enough to try this. And we started to combine them with statistical translation of informal-to-formal math.
- One point was existence of “intermediate” informal corpora like ProofWiki that have a lot of regularity
- 2014: the first 100 proof sentence patterns cover about 50% of ProofWiki

Betting Slide from IHP'14, Paris

- In 25 years, 50% of the toplevel statements in LaTeX-written Msc-level math curriculum textbooks will be parsed automatically and with correct formal semantics
- Hurry up: I will only accept bets up to 10k EUR total (negotiable)
- More at `http://ai4reason.org/aichallenges.html`

Formal, Informal and Semiformal Corpora

- HOL Light and Flyspeck: some 25,000 theorems
- The Mizar Mathematical Library: some 60,000 theorems (most of them rather small lemmas), 10,000 definitions
- Coq: several large projects (Feit-Thompson theorem, ...)
- Isabelle, seL4 and the Archive of Formal Proofs
- Arxiv.org: 1M articles collected over some 20 years (not just math)
- Wikipedia: 25,000 articles in 2010 - collected over 10 years only
- Proofwiki - LaTeX but very semantic, re-invented the Mizar proof style

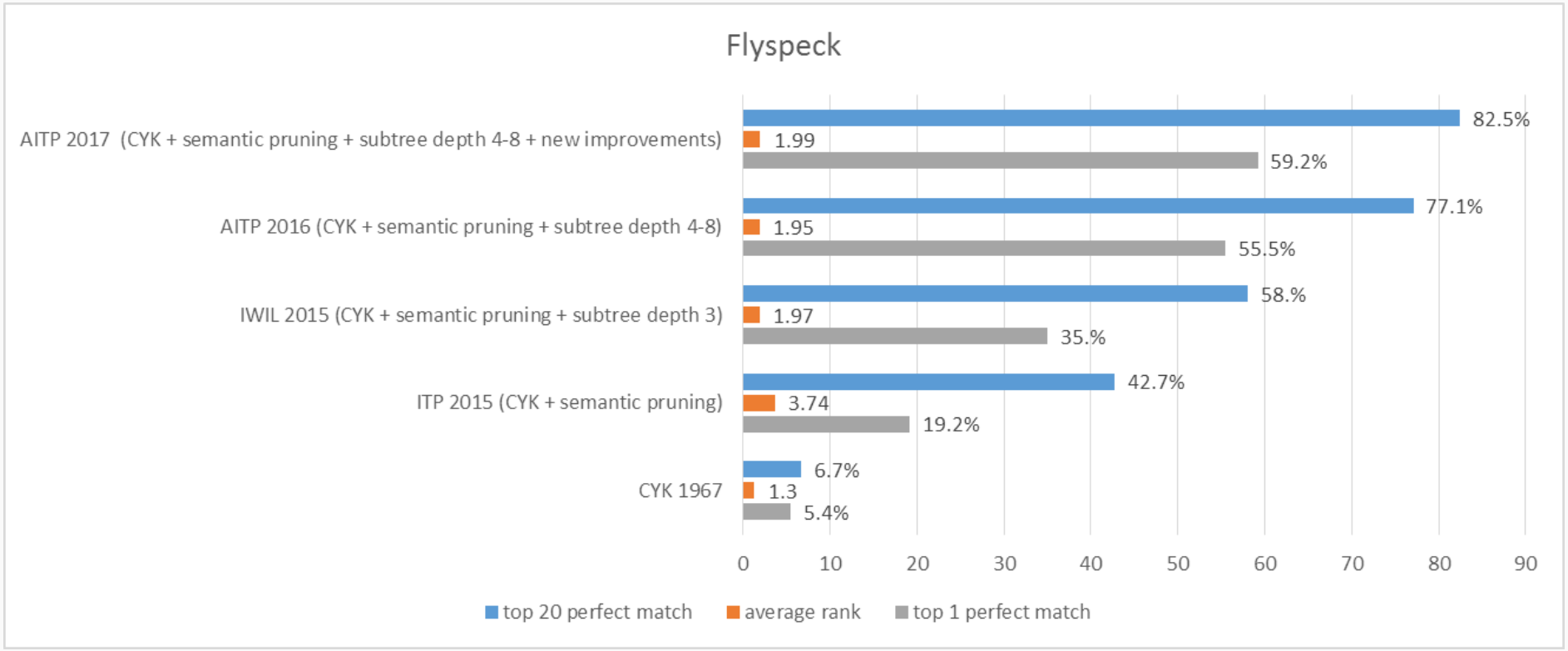
Our Approach/Plan So Far

- There is not yet much aligned informal/formal data
- So try first with “ambiguated” (informalized) formal corpora
- Try first with non black-box architectures such as probabilistic grammars
- Which can be easily enhanced internally by semantic pruning (e.g. type constraints)
- Develop feedback loops between training statistical parsing and theorem proving
- Start employing more sophisticated ML methods
- Progress to more complicated informal corpora/phenomena
- Both directly: ML/ATP with only cruder alignments (theorems, chapters, etc)
- And indirectly: train statistical/precise alignments across informal and formal corpora, use them to enhance our coverage
- Example: word2vec/Glove/neural learning of synonyms over Arxiv

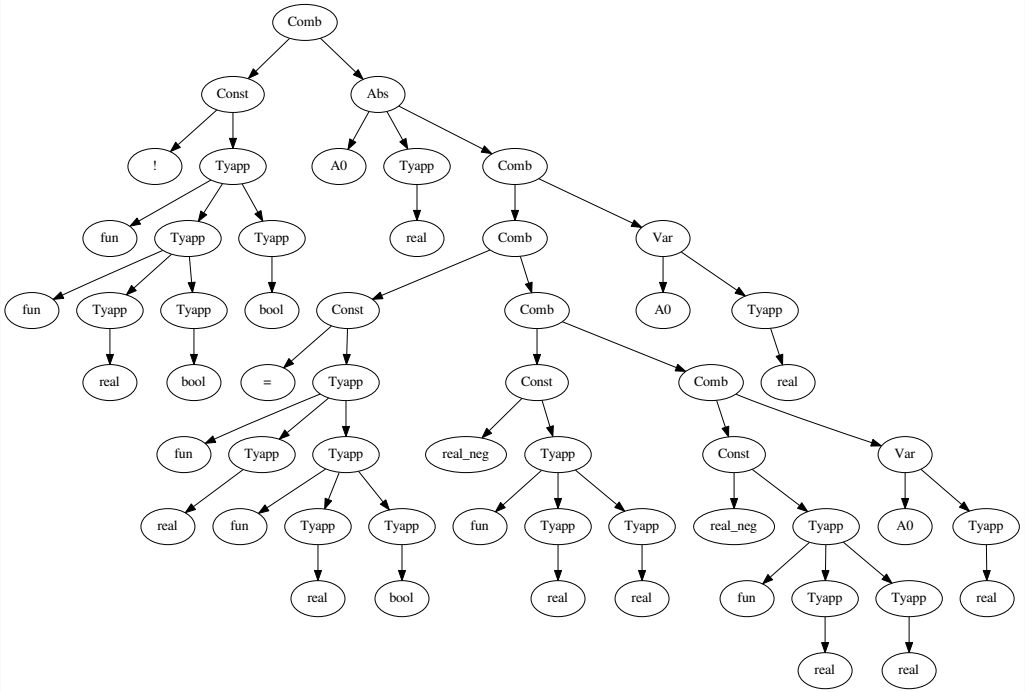
Work Done So Far: Informalized Flyspeck

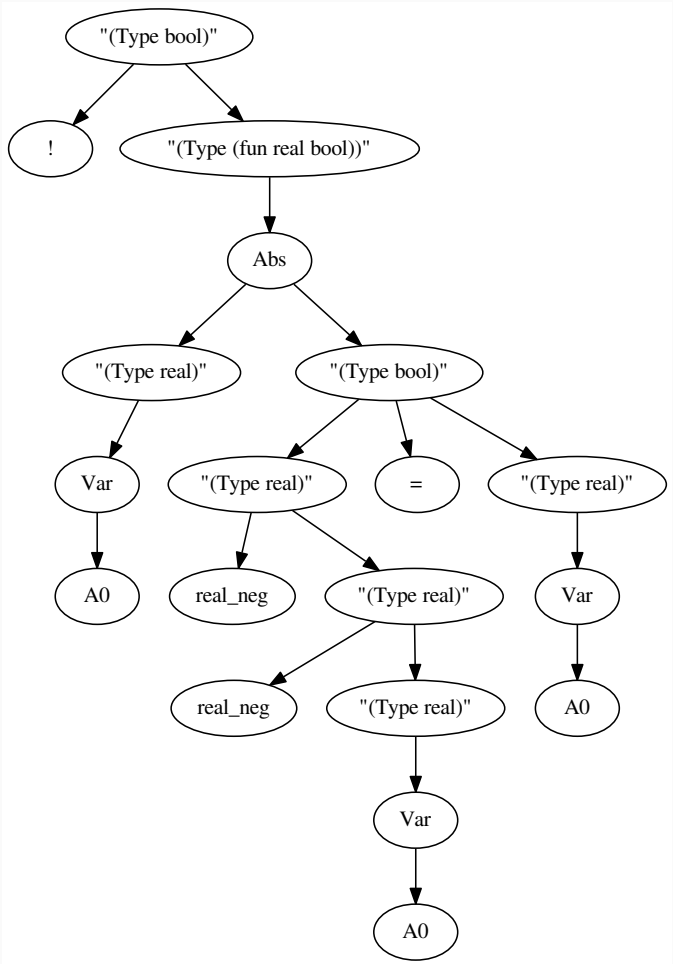
- 22000 Flyspeck theorem statements informalized
  - 72 overloaded instances like “+” for `vector_add`
  - 108 infix operators
  - forget “prefixes” `real_, int_, vector_, matrix_, complex_`, etc.
- Training a probabilistic grammar (context-free, later with deeper context)
- CYK chart parser with semantic pruning (compatible types of variables)
- Using HOL Light and HolyHammer to typecheck and prove the results

Flyspeck Progress



Example grammars





Informalized Mizar

- More natural-language features than HOL (designed by a linguist)
- Pervasisve overloading
- Declarative natural-deduction proof style (re-invented in ProofWiki)
- Adjectives, dependent types, hidden arguments, synonyms
- Addressed by using two layers:
  - user (pattern) layer - resolves overloading, but no hidden arguments completed, etc.
  - semantic (constructor) layer - hidden arguments computed, types resolved, ATP-ready

Examples of Mizar's Linguistic Mechanisms

```
definition
  let P,R be set;
  func P(#)R -> Relation means
  [x,y] in it iff ex z st [x,z] in P & [z,y] in R;
end;
notation synonym P*R for P(#)R; end;
definition
  let X,Y1,Y2,Z be set;
  let P be Relation of X,Y1;
  let R be Relation of Y2,Z;
  redefine func P*R -> Relation of X,Z;
end;
notation
  let f,g be Function;
  synonym g*f for f*g;
end;
```

Old ATP-based Approach

- AITP'17: be lazy and use ATP to connect the layers
- About 13000 Prolog-style formulas encoding the relation between user-level syntax and the semantic (MPTP) encoding
- Also the full set of Mizar typing rules needed for this - ca 30000 background knowledge rules
- Quite bad: Vampire proves about 40% in 60s, E with our mutant strategies about 50%
- Improved to about 60% in May 2017 by JU, however this also showed that our ATP encoding is unsound
- Making it sound would end up even heavier, hence our new approach

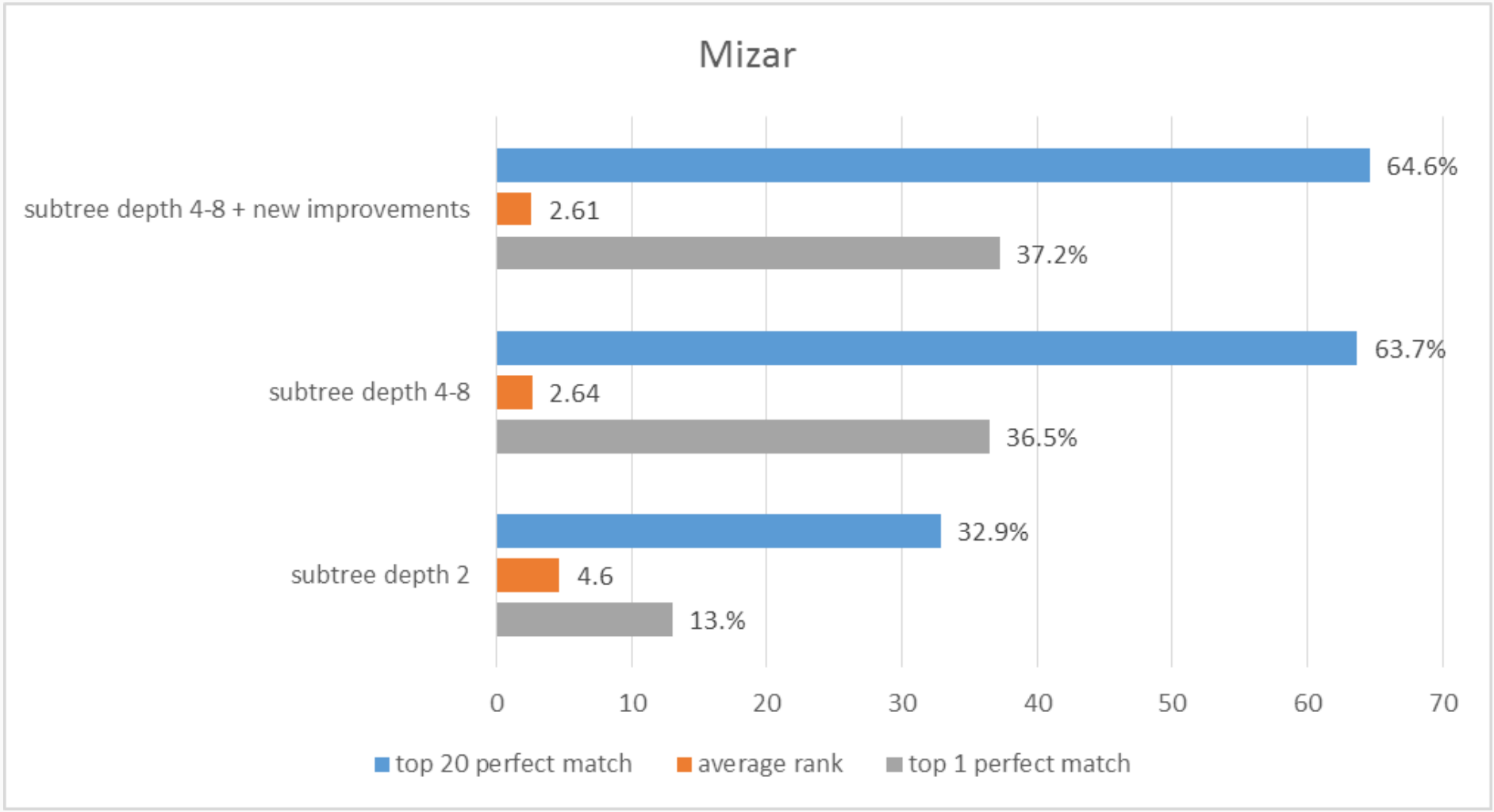
Enhancing our Parser with Mizar-style Algorithms

- Chad Brown: Mizar-style typing and elaboration inside the chart parser
- The typing and elaboration (of patterns to constructors) proceed in a mutual recursion
- Sometimes (in an incomplete parse) type guards need to be assumed
- They can be discharged or the parse may be pruned out at a later stage (when a bigger part of the formula is parse)

Elaboration results for toplevel statements

- Input: About 60K Mizar Theorems in Pattern Representation
- Output: Constructor Based Version or Failure or Timeout (20s)
- About 95% elaborated with no assumed pattern guards
- Another 2% elaborated with some assumed pattern guards
- Roughly 2% fail to fill in some implicit arguments
- Roughly 1% time out
- Things are more tricky when elaborating incomplete parses
- However we finally have a reasonable toolchain to go from ambiguated Mizar to ATP
- Not ATP-evaluated yet

First Mizar Results (100-fold Cross-validation)



ProofWiki vs Mizar – our CICM'14 Example

Example: ProofWiki vs Mizar vs Mizar-style automated proof

== Theorem ==
Let $(S, \circ)$ be an [[Definition:Algebraic Structure|algebraic structure]] that has a [[Definition:Zero Element|zero element]] $z \in S$. Then z is unique.
== Proof ==
Suppose z_1 and z_2 are both zeroes of $(S, \circ)$.
Then by the definition of [[Definition:Zero Element|zero element]]:
$z_2 \circ z_1 = z_1$ by dint of z_1 being a zero;
$z_2 \circ z_1 = z_2$ by dint of z_2 being a zero.
So $z_1 = z_2 \circ z_1 = z_2$.
So $z_1 = z_2$ and there is only one zero after all.
{{qed}}
// *NB: Informal proofs are buggy!*

```
Th9:  e1 is_a_left_unity_wrt o &
e2 is_a_right_unity_wrt o implies e1 = e2
proof
assume that A1:  e1 is_a_left_unity_wrt o and
A2:  e2 is_a_right_unity_wrt o;
thus e1 = o.(e1,e2) by A2,Def6 .= e2 by A1,Def5;
end;

z1 is_a_unity_wrt o & z2 is_a_unity_wrt o
implies z1 = z2 proof
assume that A1:  z1 is_a_unity_wrt o and
A2:  z2 is_a_unity_wrt o;
A3:  o.(z2,z1) = z1 by Th3,A2; ::[ATP]
A4:  o.(z2,z1) = z2 by Def 6,Def 7,A1,A3; ::[ATP]
hence z1 = z2 by Th9,A1,Def 7,A2; ::[ATP]
end;
```

Can We Align Proofwiki with Mizar and Parse It?

- Since 2015: Grzegorz Bancerek aligning Mizar and Proofwiki
- Over 500 ProofWiki pages
- Example: `https://proofwiki.org/wiki/Arithmetic_iff_Way_Below_Relation_is_Multiplicative_in_Algebraic_Lattice`
- Not just automated translation, but made to fit the math already developed in ProofWiki
- How do we use it?

ProofWiki vs Mizar Again

PW code

```
Let $\left(S, \preceq\right)$ be an ordered set.  Let $x \in S$.  Then
$\left\ x\right\$ is a chain of $\left(S, \preceq\right)$.
```

PW display

Let $(S, \preceq)$ be an ordered set. Let $x \in S$. Then $\{x\}$ is a chain of $(S, \preceq)$.

Mizar

```
for A being non empty reflexive RelStr for a being Element of A holds {a}
is Chain of A
```

Mizar parse

```
(Bool for (Varlist (Set (Var A))) being (Type (@ListOfAdjectives
(Adjective ($#~nv2_struct_0 non (Attribute ($#nv2_struct_0 empty))))
(Adjective (Attribute ($#nv3_orders_2 reflexive)))) ($#nl1_-
orders_2 RelStr)) (Bool for (Varlist (Set (Var a))) being (Type
(@ListOfAdjectives) ($#nm1_struct_0 Element) of (Set (Var A))) holds
(Bool (Set ($#nk6_domain_1 {) (Set (Var a)) ($#nk6_domain_1 })) is (Type
(@ListOfAdjectives) ($#nm2_orders_2 Chain) of (Set (Var A))))))
```

ProofWiki vs Mizar Issues

1. The Pr∞fWiki `chain` can map directly to the Mizar-style subtree `($#nm2_orders_2 chain)`, possibly additionally aligning `chain` with `Chain` as synonyms.
2. The Pr∞fWiki TEX text `"\left\{ {x}\right\}"` needs to be mapped to the Mizar-style subtree `(Set ($#nk6_domain_1 {) (Set (Var x)) ($#nk6_domain_1_part_1 }))`.
3. `"ordered set"` needs to be mapped to Mizar `"non empty reflexive RelStr"`.
4. `"Let...Then..."` needs to be mapped to Mizar as `"for...holds..."`. Etc.

ProofWiki vs Mizar Issues - Proposed Solutions

- (1) is just a new grammar rule that can be learned from the treebank.
- The other examples however require more complex tree transformations
- So we added grammar extension that allows evaluation of arbitrary Lisp-like programs at nonterminal positions
- Following is an example of a subtree (and code) that performs the mapping (2):

```
(Set  ("PW_TeX_Singleton@@@
      (lambda (L LSB LB X RB R RSB)
            (list (gtree '$#nk6_domain_1 '{) X (gtree
'$#nk6_domain_1_part_1 '})))"
     "\left" "\{" "{" (Set (Var "x")) "}" "\right" "\}" ))
```

Learning Lisp Programs

- We plan to learn Lisp-like programs by the following bootstrapping procedure:
  1. The parser run on the corpus of Pr∞fWiki texts will identify the parts of input that cannot be parsed yet.
  2. This can be done by using a special low-probability nonterminal `"UNKNOWN"` that propagates through most of the grammar rules, marking the failed fragments.
  3. The failed fragments will be aligned with the corresponding Mizar subtrees.
  4. This yields a corpus of Pr∞fWiki - Mizar pairs where the parsing fails so far.
  5. This corpus can be mined for common frequent patterns.
  6. Use symbolic learning methods (ILP, Genetic Programming, etc.) to gradually create a corpus of more and more advanced Lisp-like functions that build on each other.
  7. Sometimes we'll add a difficult Lisp function manually.
  8. As usual the most probable parses will be subjected to typechecking and large-theory ATP, using the whole Mizar library as a background knowledge and the internal Pr∞fWiki steps as lemmas

Thanks for listening!

- Questions?