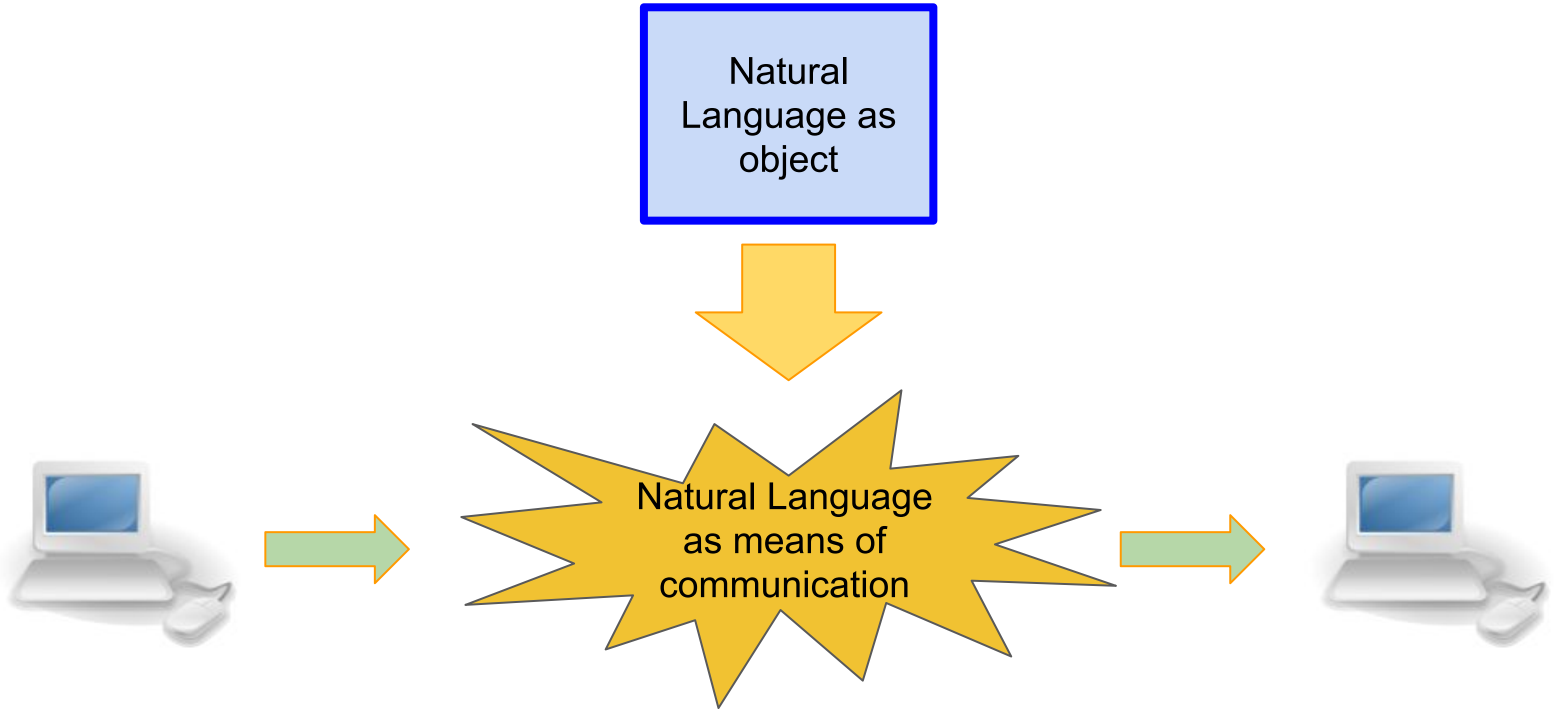
A Promising Path To Autoformalization and General AI

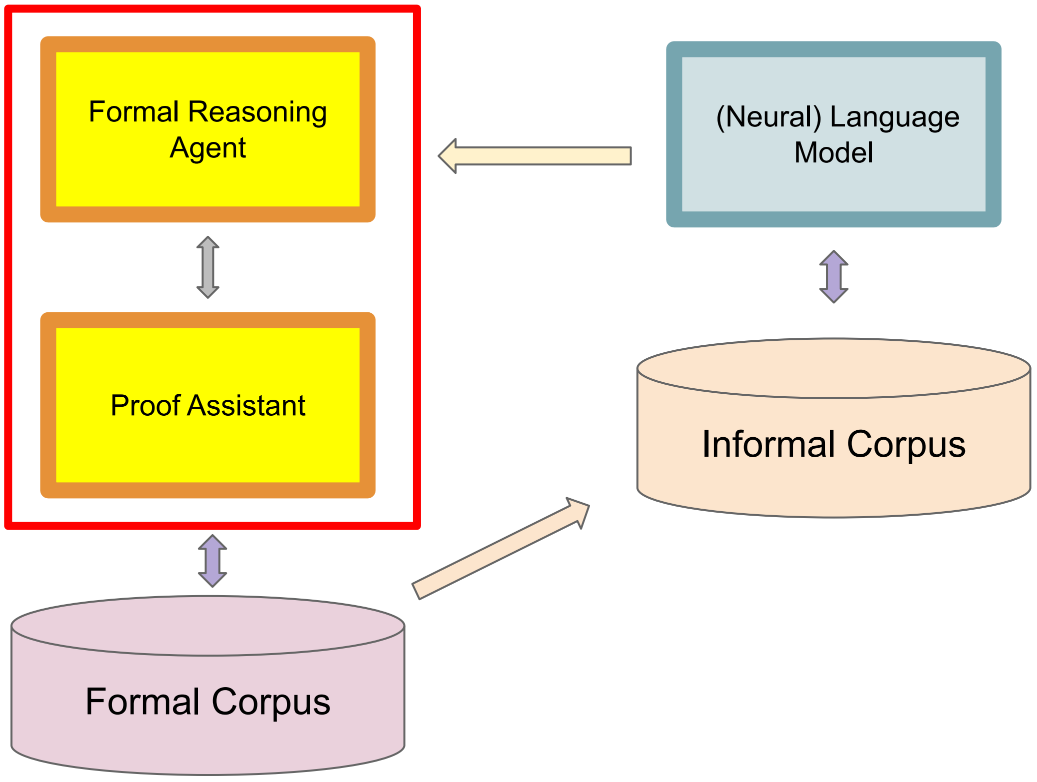
CICM 2020
29th of July 2020

Christian Szegedy
Google Research

Can we demonstrate real language understanding?



Vision of joint proving and auto-formalization



Background And History

John McCarthy: **Computer programs for checking mathematical proofs.** In: A Paper Presented at the Symposium on Recursive Function Theory, New York, April 1961

Donald Lee Simon: **Checking number theory proofs in natural language**. Ph.D thesis (1990)

Claus Zinn: **Understanding informal mathematical discourse**. Ph.D thesis, Institut für Informatik, Universität Erlangen-Nürnberg (2004)

Background And History

Josef Urban: **Translating Mizar for first order theorem provers**. MKM 2003

Josef Urban: **MaLARea: a metasystem for automated reasoning in large theories**. CADE-21 (2007)

Cezary Kaliszyk, Josef Urban, Jiří Vyskočil: **Learning to parse on aligned corpora (Rough Diamond)**. ITP 2015

Cezary Kaliszyk, Josef Urban, Jiří Vyskocil: **System description: statistical parsing of informalized Mizar formulas**. SYNASC 2017

Autformalization vs. Formal Theorem Proving Only

- Most mathematics is given in natural language (this is where the data is)

Autformalization vs. Formal Theorem Proving Only

- Most mathematics is given in natural language (this is where the data is)
- Open-ended exploration? Interestingness is hard to define
    - AlphaZero: Could do self-play. Math cannot be done via self-play unless the interestingness problem is solved (what to explore)
    - Generated mathematics would be alien to us. How to evaluate?
    - How would one communicate with a system that has developed its own notions and theories?

Autformalization vs. Formal Theorem Proving Only

- Most mathematics is given in natural language (this is where the data is)
- Open-ended exploration? Interestingness is hard to define
  - AlphaZero: Could do self-play. Math cannot be done via self-play unless the interestingness problem is solved (what to explore)
  - Generated mathematics would be alien to us. How to evaluate?
  - How would one communicate with a system that has developed its own notions and theories?
- Formalization itself is a hard task
  - Manual formalization requires domain experts
  - Hard to check correctness wrt to natural language
  - Slow

Is (Deep) Reinforcement Learning Useful?

Alemi et al: DeepMath (NIPS 2016): **Deep Neural Networks for Premise Selection**

Whalen: **Holophrasm** (Deep RL for Metamath) (2016)

Loos et al: **Deep Network Guided Proof Search**: LPAR (2017)

Kaliszyk et al: **Reinforcement Learning for Theorem Proving** (2018)

Zombori et al: **Towards Finding Longer Proofs** (2019)

Bansal et al: **HOList/DeepHOL** (Deep RL for HOL Light) ICML (2019)

Machine Translation/Language Modeling (Transformers):

Lample et al: **Unsupervised Machine Translation Using Monolingual Corpora Only** (2017)

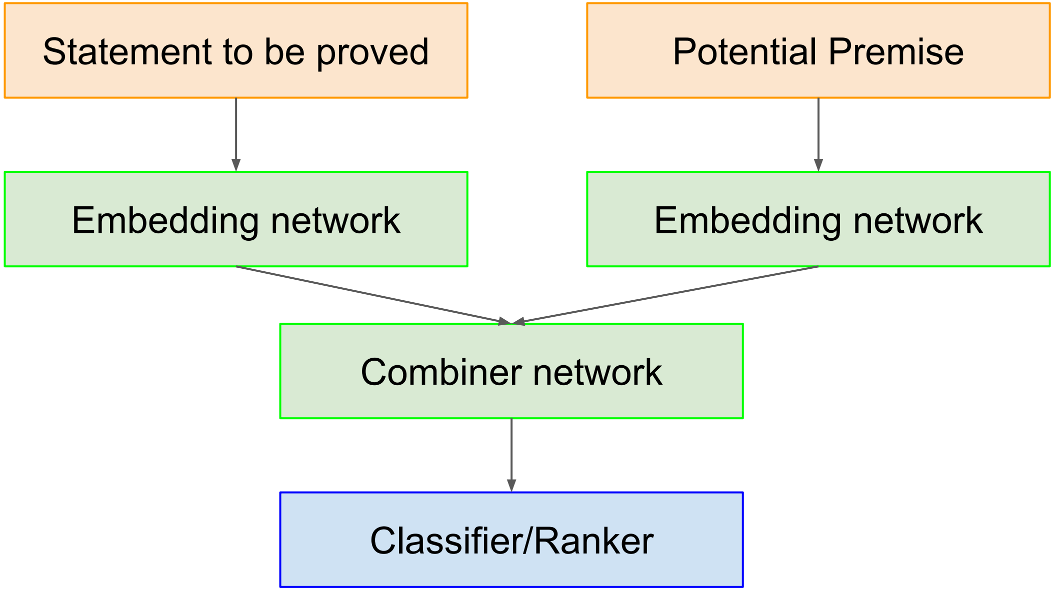
Devlin et al: BERT: **Pre-training of Deep Bidirectional Transformers for Language Understanding**

Lample, Charton: **Deep Learning for Symbolic Mathematics** (ICRL 2019)

Rabe et al: **Language Modeling for Formal Mathematics** (2020)

Brown et al: **Language Models are Few-Shot Learners** (2020) [GPT-3]

Premise Selection Using Deep Learning



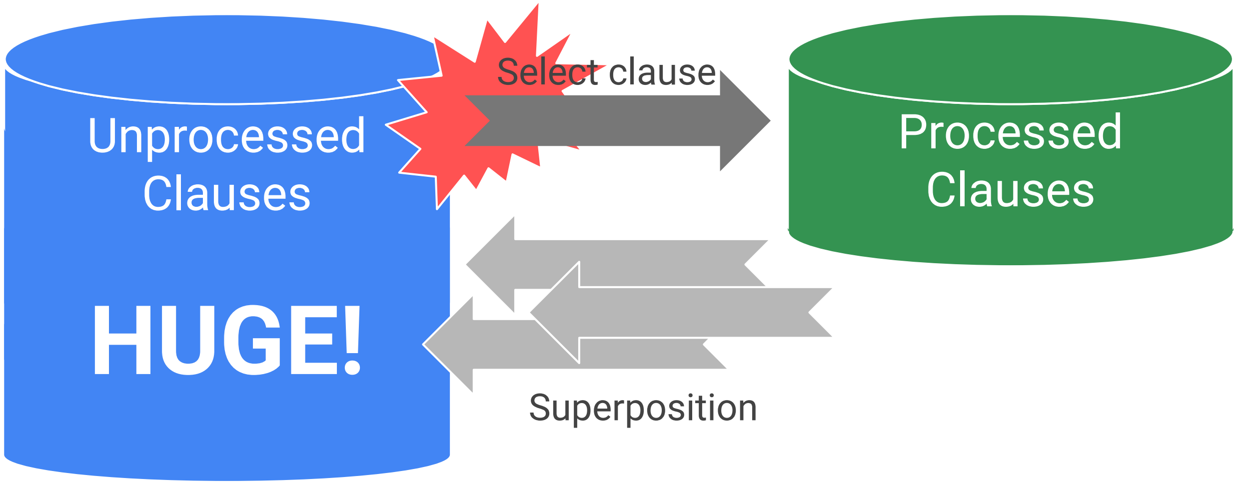
Embedding Network:

- **Convolutional** network
- Recurrent **LSTM** network
- Combined convolutional network with LSTM on top

DeepMath-Deep Sequence Models for Premise Selection

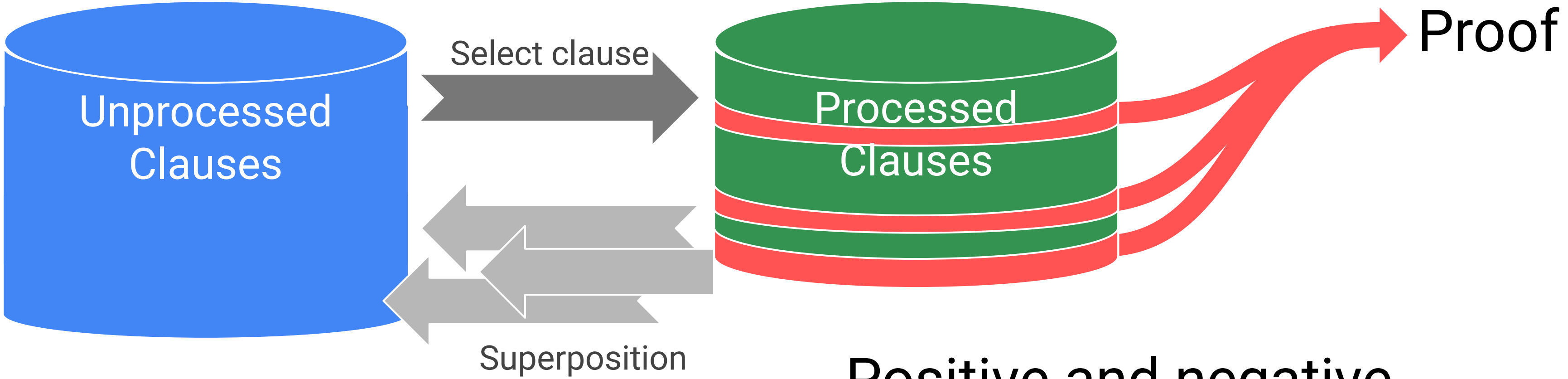
Alemi, A. A., Chollet, F., Een, N., Irving, G., Szegedy, C., & Urban, J, *NIPS 2016*

The E Theorem Prover



System Description: E 1.8,
Stephan Schulz. LPAR (2013)
`www.eprover.org`

The E Theorem Prover - Generating Training Data

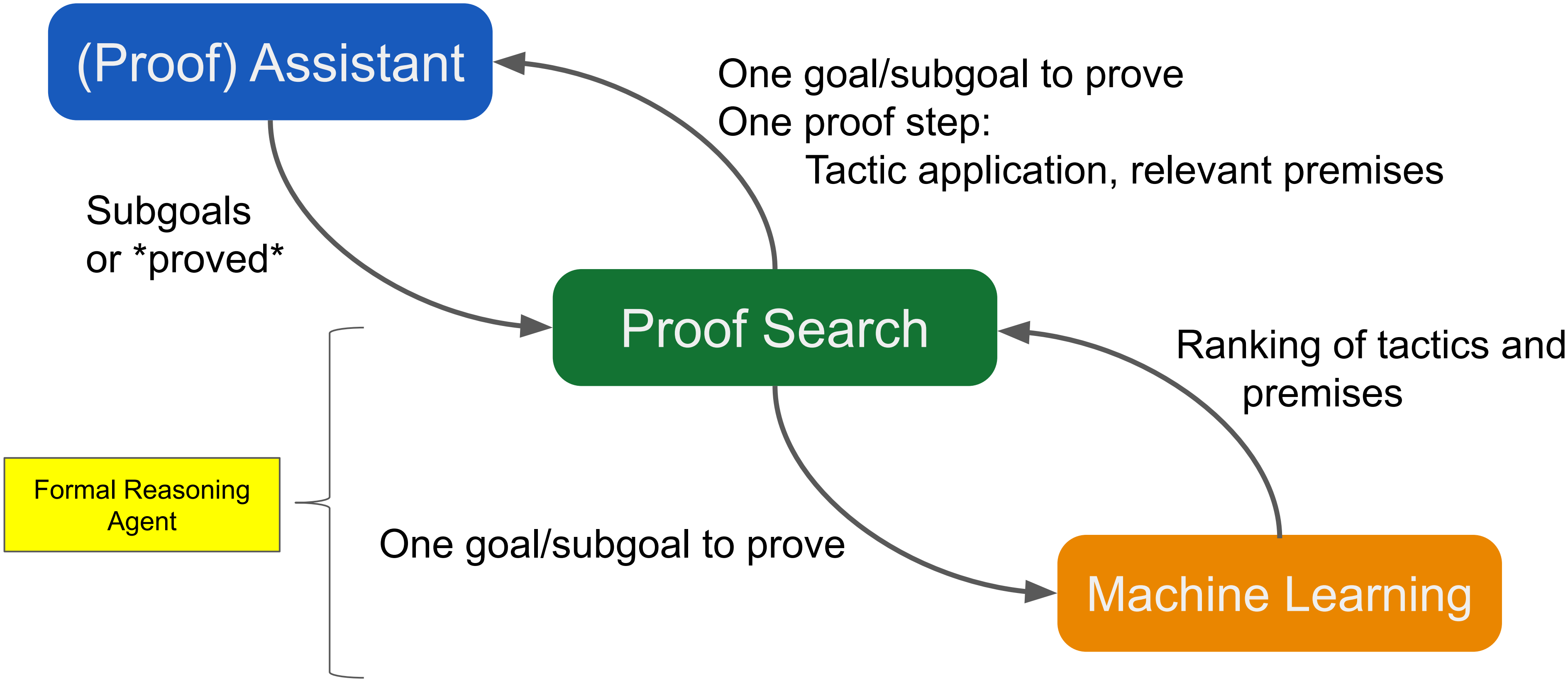


Positive and negative examples for training

Deep Network Guided Proof Search

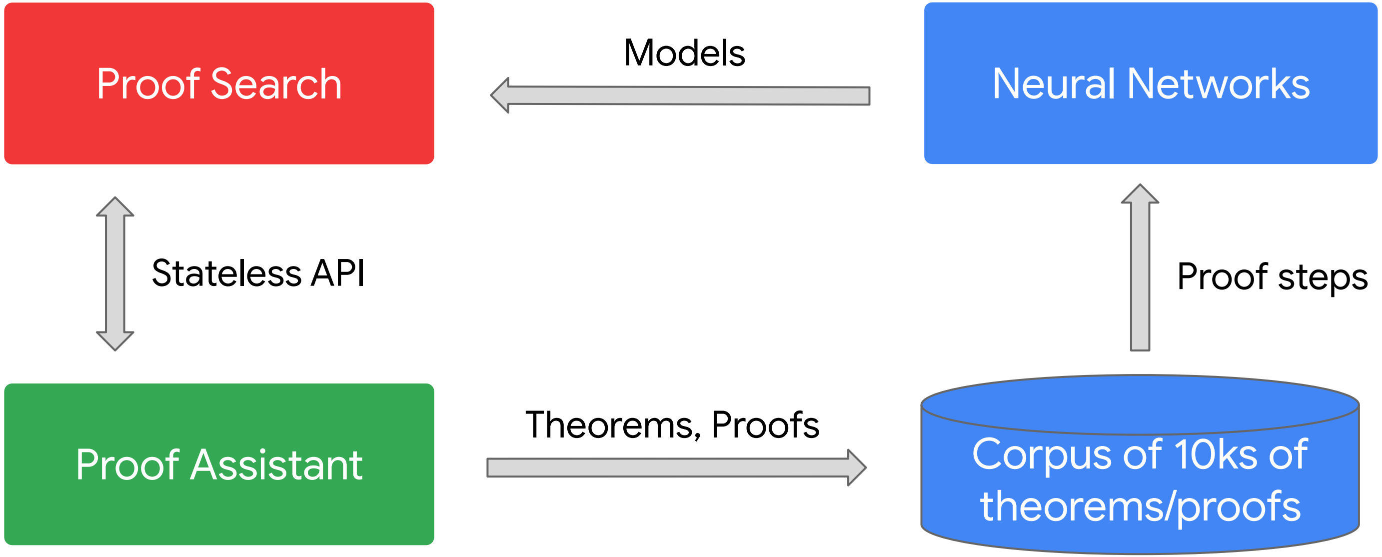
S. Loos, G. Irving, C. Szegedy, and C Kaliszyk. *LPAR* (2017).

APIs for Theorem Prover Developers and ML Researchers



Open Source Release: The HOList Environment

`www.deephol.org`

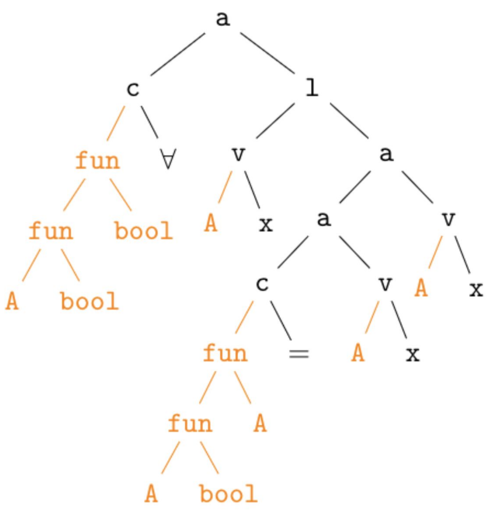


Bansal et al: HOList: An Environment for Machine Learning of Higher-Order Theorem Proving, ICML(1029)

Neural Architectures for **Formulas**

Apply **Graph Neural Networks** to abstract syntax trees. E.g.:

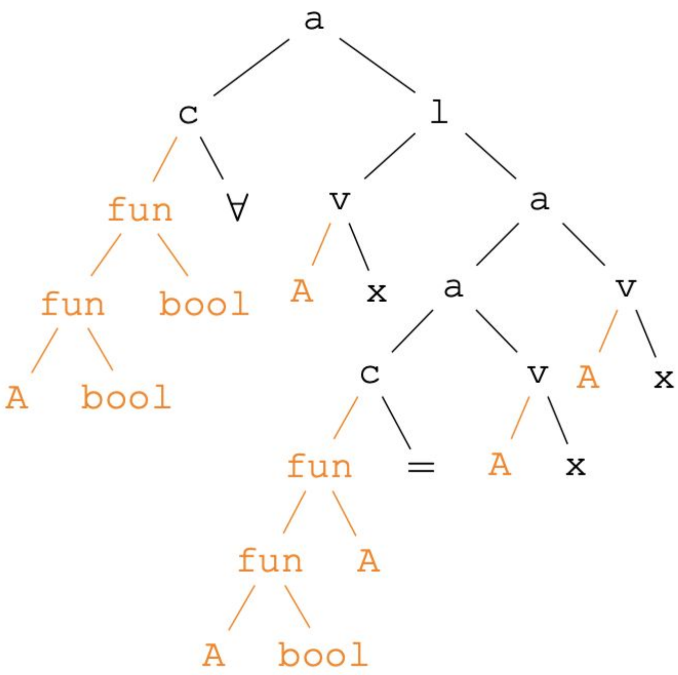
$$\forall x : \ x = x$$

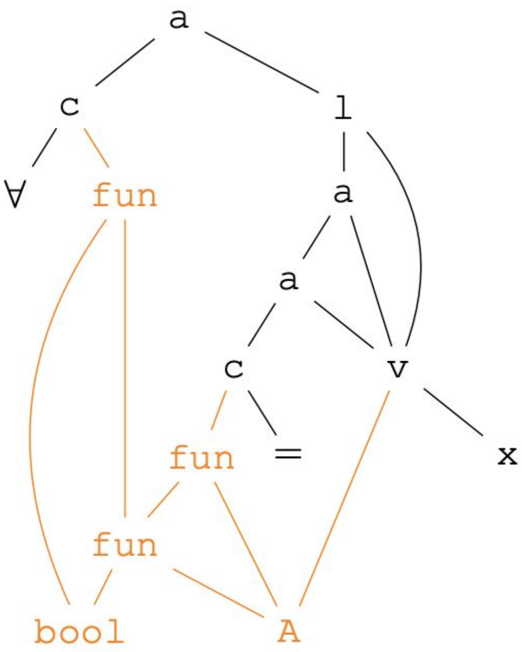


a: function application
c: constant
v: variable
l: lambda/abstraction

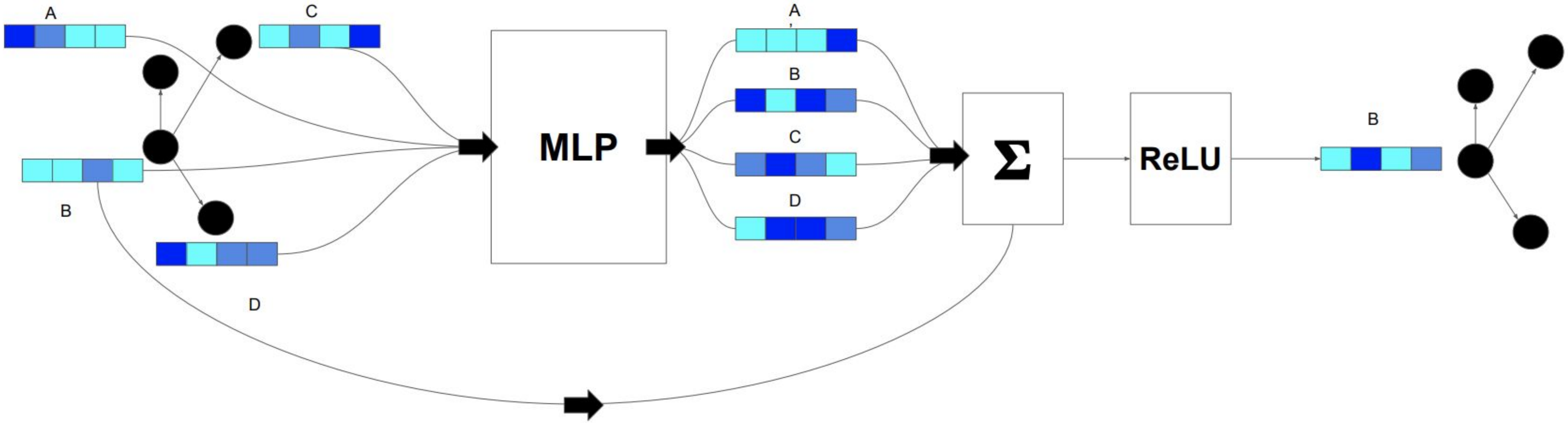
fun, bool, A are type annotations

Compressed Graph Representation



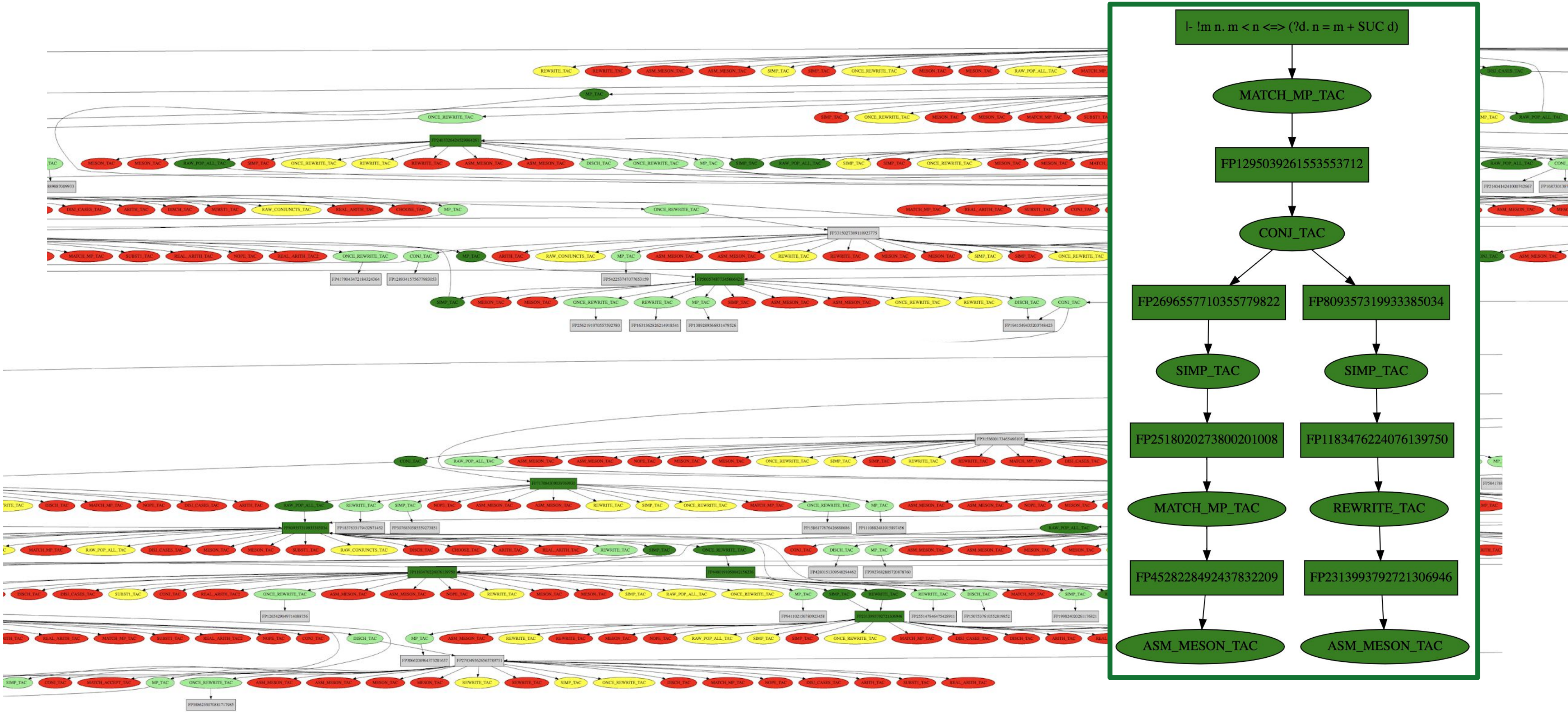


Graph Neural Networks

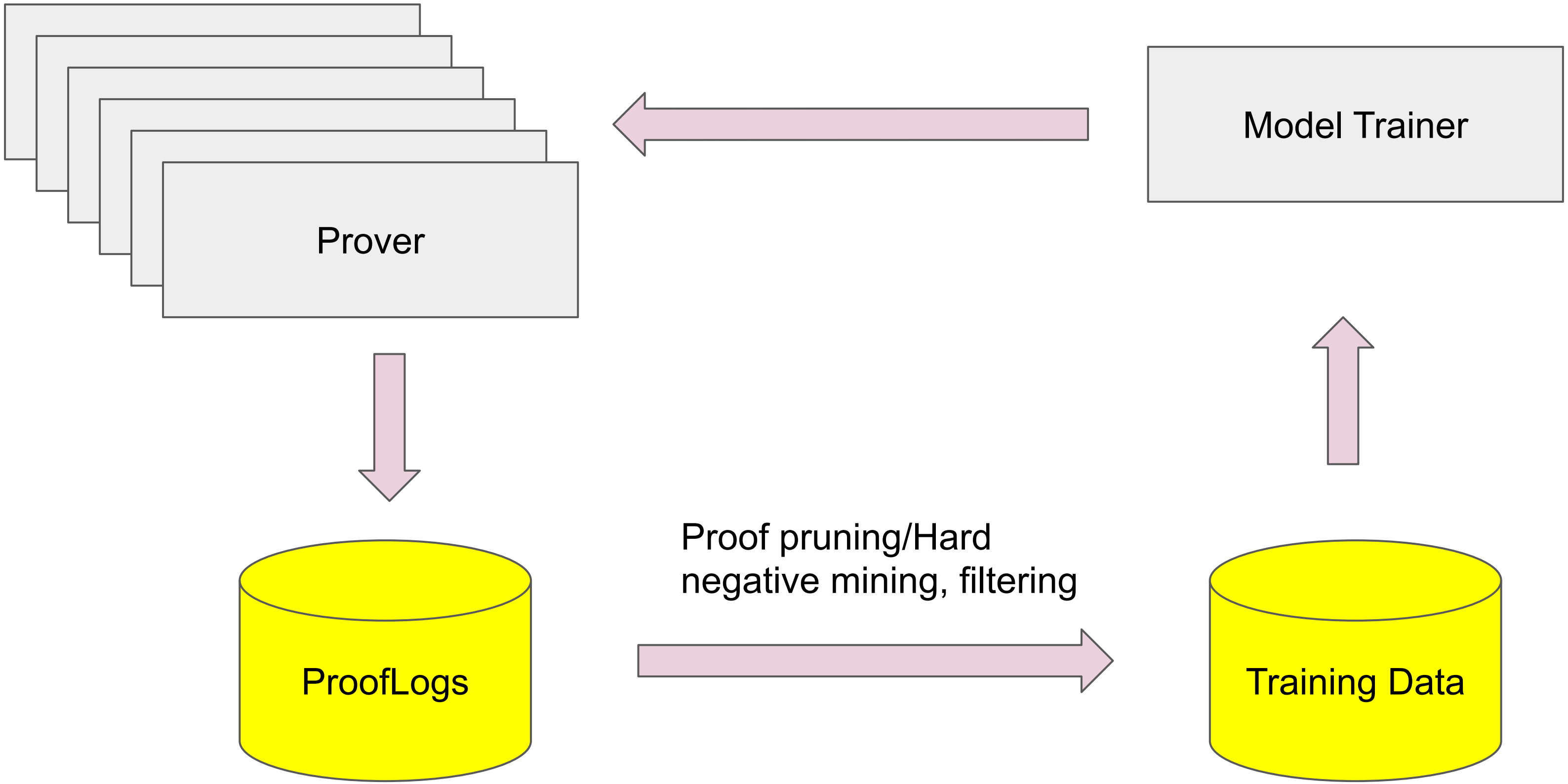


Paliwal et al, Graph Representations for Higher-Order Logic and Theorem Proving AAAI 2020

Proof Search Tree



DeepHOL Loop



Bansal et al.: Learning to Reason in Large Theories without Imitation (2019)

DeepHOL results on HOL-Light Core + Multivariate Analysis

Method	**Proof success rate**
Imitation Learning With Graph Neural Networks	50%
DeepHOL-Zero Reinforcement Learning, Bootstrapping without human proofs	56%
Imitation + Reinforcement Learning	60%
Cumulative over Reinforcement Learning	70%

Cherry picked Example

Human proof

```
let LOCALLY_INTER_OPEN = prove
 (`!P s t u:real^N->bool.
        locally P s /\
        open_in (subtopology euclidean u) s /\
        open_in (subtopology euclidean u) t
        ==> locally P (s INTER t)`,
  REPEAT STRIP_TAC THEN REWRITE_TAC[locally; IN_INTER] THEN
  MAP_EVERY X_GEN_TAC [`v:real^N->bool`; `x:real^N`] THEN STRIP_TAC THEN
  FIRST_X_ASSUM(MP_TAC o GEN_REWRITE_RULE I [locally]) THEN
  DISCH_THEN(MP_TAC o SPECL [`t INTER v:real^N->bool`; `x:real^N`]) THEN
  ASM_REWRITE_TAC[IN_INTER] THEN ANTS_TAC THENL
   [CONJ_TAC THENL
     [MATCH_MP_TAC OPEN_IN_TRANS THEN EXISTS_TAC `s INTER t:real^N->bool`
THEN
      CONJ_TAC THENL
       [SUBGOAL_THEN `t INTER v:real^N->bool = (s INTER t) INTER v`
         (fun th -> ASM_SIMP_TAC[th; OPEN_IN_INTER; OPEN_IN_REFL]) THEN
        REPEAT(FIRST_X_ASSUM(MP_TAC o MATCH_MP OPEN_IN_IMP_SUBSET)) THEN
        ASM SET_TAC[];
        MATCH_MP_TAC OPEN_IN_SUBTOPOLOGY_INTER_SUBSET THEN
        EXISTS_TAC `u:real^N->bool` THEN
        ASM_SIMP_TAC[OPEN_IN_INTER; OPEN_IN_REFL] THEN
        ASM_MESON_TAC[OPEN_IN_IMP_SUBSET]];
      REPEAT(FIRST_X_ASSUM(MP_TAC o MATCH_MP OPEN_IN_IMP_SUBSET)) THEN
      ASM SET_TAC[]];
    MATCH_MP_TAC MONO_EXISTS THEN X_GEN_TAC `n:real^N->bool` THEN
    MATCH_MP_TAC MONO_EXISTS THEN X_GEN_TAC `l:real^N->bool` THEN
    STRIP_TAC THEN ASM_REWRITE_TAC[] THEN
    CONJ_TAC THENL [ALL_TAC; ASM SET_TAC[]] THEN
    MATCH_MP_TAC OPEN_IN_SUBSET_TRANS THEN
    EXISTS_TAC `s:real^N->bool` THEN ASM_REWRITE_TAC[] THEN
    REPEAT(FIRST_X_ASSUM(MP_TAC o MATCH_MP OPEN_IN_IMP_SUBSET)) THEN
    ASM SET_TAC[]]);;
```

Proof found by DeepHOL

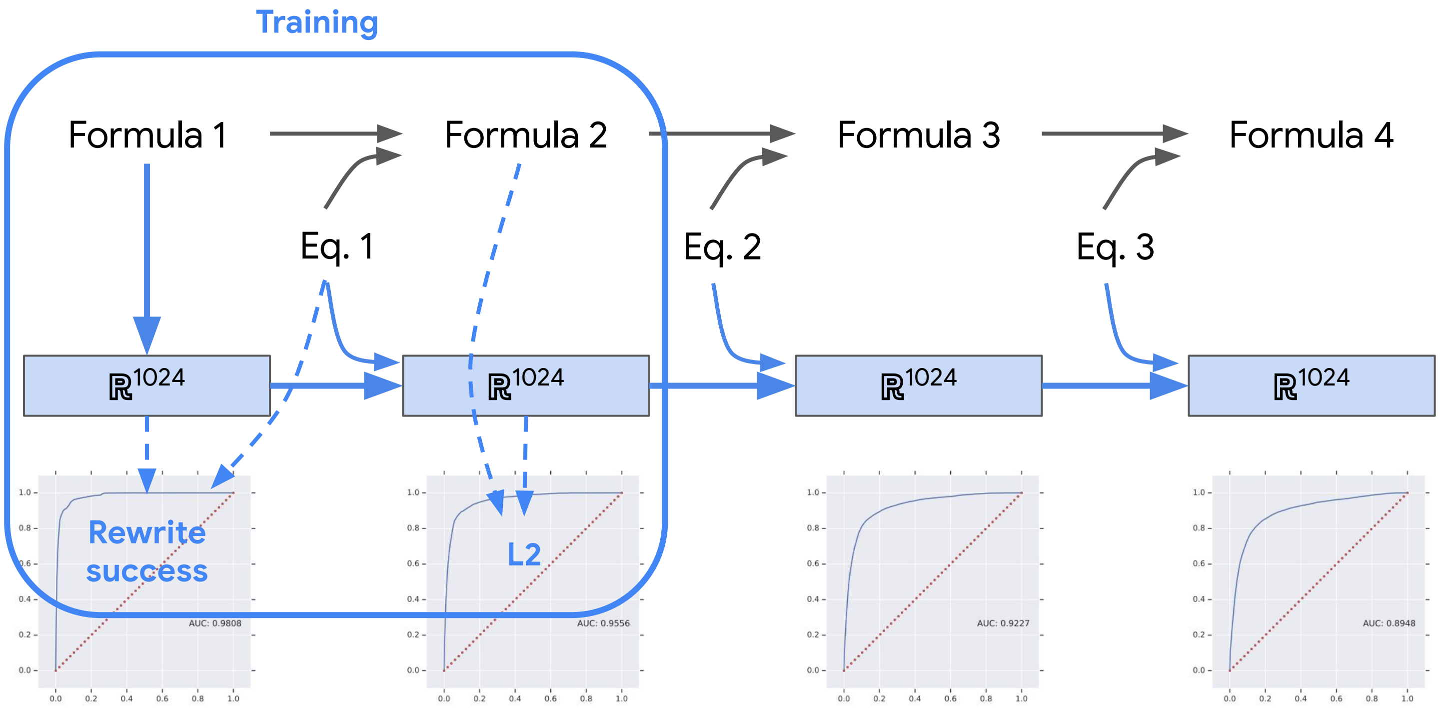
```
g `!P s t u:real^N->bool.
    locally P s /\
    open_in (subtopology euclidean u) s /\
    open_in (subtopology euclidean u) t
    ==> locally P (s INTER t)`;;
```

```
e (MESON_TAC [LOCALLY_OPEN_SUBSET; open_in;
INTER_SUBSET; OPEN_IN_INTER; OPEN_IN_SUBSET_TRANS]);;
val it : goalstack = No subgoals
```

Dennis Lee at al.
Matematical Reasoning in Latent Space, ICLR (2019)

Reasoning in Latent Space

Trivial example: $3^2 = 9$ can be **rewritten** to $3*3 = 9$ by the equality $x^2 = x*x$.



Transformer Networks

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin: **Attention is All You Need**, NIPS 2017

Transformers

Attention: Relaxation of Nearest-Neighbor Lookup in a vector space:

- Key Matrix: K
- Value Matrix: V
- Query Matrix: Q

$$softmax(QK^T)V$$

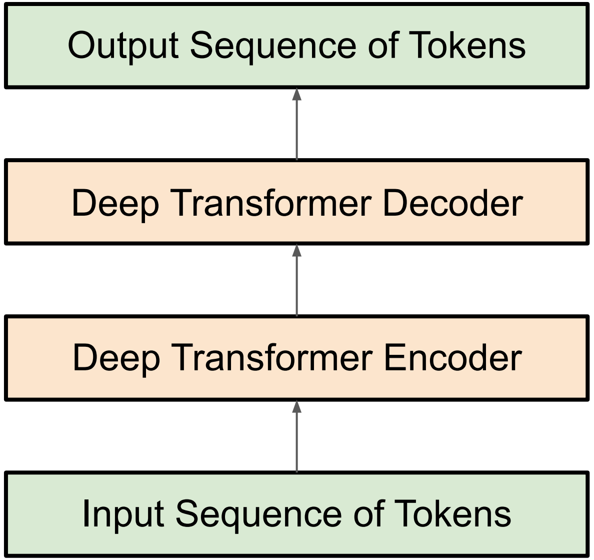
Transformers

Self-Attention: Query and memory comes from the same set of vectors

- A lot of lookups in parallel
  (stacked horizontally: “attention heads”).
- A lot of layers:
  (stacked vertically: “layers”)
- K, V and Q and are computed by matrix products using learned parameters.

Skip-Tree Model Pretraining via Transformers

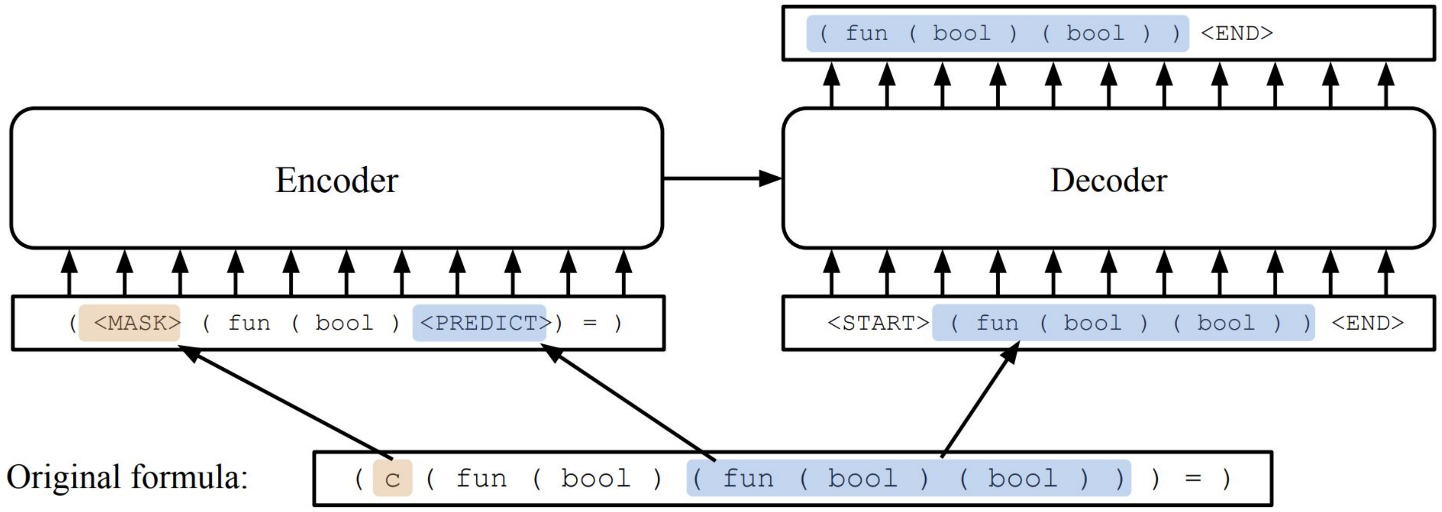
Machine Translation:



Rabe et al: **Language Modeling for Formal Mathematics** (2020)

Skip-Tree Model Pretraining via Transformers

Task:



Skip Tree Inference Tasks

- $\texttt{<PREDICT>} \Rightarrow (x \Leftrightarrow (\ b \vee x1) \wedge (b \vee x0))$
- $\texttt{<PREDICT>} \Rightarrow (g \setminus \{s\}) = g$
- $\texttt{<PREDICT>} \Rightarrow (x1/y1 = x2/y2 \ \Leftrightarrow \ x1 * y2 = x2 * y1)$

Skip Tree Inference Tasks

- $\texttt{<PREDICT>} \Rightarrow (x \Leftrightarrow (\, b \vee x1) \wedge (b \vee x0))$
- $\texttt{<PREDICT>} \Rightarrow (g \setminus \{s\}) = g$
- $\texttt{<PREDICT>} \Rightarrow (x1/y1 = x2/y2 \;\Leftrightarrow\; x1 * y2 = x2 * y1)$

The ground truth answers are as follows:

- $((b \Leftrightarrow \texttt{False}) \Rightarrow (x \Leftrightarrow x0)) \wedge (b \Leftrightarrow \texttt{True}) \Rightarrow (x \Leftrightarrow x1)$
- $\neg(s \in g)$
- $0 < y1 \wedge 0 < y2$, note that $0 \neq y1 \wedge 0 \neq y2$ would be a more general assumption.

Skip-Tree Inference Tasks

- $\forall x, n \in \mathbb{N}: \ (x^n = 1) = \texttt{<PREDICT>}$
- $\forall m, n: \ n \leq m \Rightarrow m - n + n = \texttt{<PREDICT>}$
- $\forall l, m: \ \texttt{<PREDICT>} = \texttt{APPEND}(\texttt{REVERSE}(m), \ \texttt{REVERSE}(l))$

Skip-Tree Inference Tasks

- $\forall x, n \in \mathbb{N}: \ (x^n = 1) = \texttt{<PREDICT>}$
- $\forall m, n: \ n \leq m \Rightarrow m - n + n = \texttt{<PREDICT>}$
- $\forall l, m: \ \texttt{<PREDICT>} = \texttt{APPEND}(\texttt{REVERSE}(m), \ \texttt{REVERSE}(l))$

The ground truth for the tasks is:

- $x = 1 \vee n = 0$
- m
- $\texttt{REVERSE}(\texttt{APPEND}(l, m))$

Success Rate of Transformer Based Inference

Dataset	Type Inference	Hard Type Inference	Assumptions	Equalities
Skip-tree (uniform)	96.21%	**94.40%**	40.85%	**46.57%**
Skip-tree (weighted)	**96.23%**	93.32%	**40.86%**	42.89%
Skip-tree (small)	95.89%	90.42%	39.23%	40.91%
Skip-tree (no <MASK>)	96.07%	32.50%	38.38%	41.60%
Skip-sequence (long)	9.44%	0.06%	0.53%	0.56%
Skip-sequence (medium)	48.94%	5.97%	3.32%	3.55%
Skip-sequence (short)	77.25%%	3.21%	0.68%	2.06%

Neural Network Model Parameters over the years

Name	Year	Million Parameters
QuocNet (Google)	2012	600
AlexNet (University of Toronto)	2012	60
Inception-v1 (Google)	2014	12
ResNet-152 (Microsoft)	2015	60
Bert-Large (Google)	2018	340
GPT-2 Large (OpenAI)	2018	1500
Meena (Google)	2019	2600
GPT-3 (OpenAI)	2020	175000
GShard (Google)	2020	600000

Rough Estimated Compute Cost of Training Models

Name	**Year**	**PetaFLOPS-days (310^{22} floating point operations)**
AlexNet (University of Toronto)	2012	10^{-2}
ResNet-152 (Microsoft)	2015	10^{-1}
Bert-Large (Google)	2018	10^{2}
GPT-2 Large (OpenAI)	2018	10^{2}
Meena (Google)	2019	10^{4}
GPT-3 (OpenAI)	2020	10^{4}
GShard (Google)	2020	10^{4}

Forward Looking Ideas

- Pretrain large models in unsupervised manner (generatively) for all available formal/informal mathematics
- Use cycle-consistency to train translation models without parallel corpora
- Use hindsight experience replay to do guided exploration.
- Use images as inputs for mathematical text

Forward Looking Ideas

- Pretrain large models in unsupervised manner (generatively) for all available formal/informal mathematics
-

Forward Looking Ideas

- Pretrain large models in unsupervised manner (generatively) for all available formal/informal mathematics
- Use cycle-consistency to train translation models without parallel corpora

Forward Looking Ideas

- Pretrain large models in unsupervised manner (generatively) for all available formal/informal mathematics
- Use cycle-consistency to train translation models without parallel corpora
- Use hindsight experience replay to do guided exploration.

Forward Looking Ideas

- Pretrain large models in unsupervised manner (generatively) for all available formal/informal mathematics
- Use cycle-consistency to train translation models without parallel corpora
- Use hindsight experience replay to do guided exploration.
- Use images as inputs for mathematical text

Vision of joint proving and auto-formalization