Journal of Universal Computer Science, vol. 27, no. 11 (2021), 1193-1202
submitted: 11/2/2021, accepted: 9/8/2021, appeared: 28/11/2021

On Recurrent Neural Network Based Theorem Prover For First Order Minimal Logic

Ashot Baghdasaryan
(Russian-Armenian University, Yerevan, Armenia
https://orcid.org/0000-0003-1424-0382, baghdasaryana95@gmail.com)

Hovhannes Bolibekyan
(Yerevan State University, Yerevan, Armenia, bolibekhov@ysu.am)

Abstract: There are three main problems for theorem proving with a standard cut-free system for the first order minimal logic. The first problem is the possibility of looping. Secondly, it might generate proofs which are permutations of each other. Finally, during the proof some choice should be made to decide which rules to apply and where to use them. New systems with history mechanisms were introduced for solving the looping problems of automated theorem provers in the first order minimal logic. In order to solve the rule selection problem, recurrent neural networks are deployed and they are used to determine which formula from the context should be used on further steps. As a result, it yields to the reduction of time during theorem proving.

Keywords: Automated theorem prover, Minimal logic, Loop detection, Recurrent neural network
Categories: F4.1, F4.3, I.2.3
DOI: 10.3897/jucs.76563

1 Introduction

Formalized intuitionistic logic was originally developed by [Heyting 1930]. Minimal logic being part of intuitionistic logic, is of a great interest being a constructive logic. Automated theorem proving problems in miminal logic have many applications in a variety of tasks, such as expert systems, software verification and synthesis problems and a number of other areas.

The sequent system GM^- for minimal first order logic was introduced in [Bolibekyan and Chubaryan 2002]. GM^- is a permutation-free sequent system; it avoids the permutation problems of some cut-free sequent systems. GM^- partly addresses the looping problem and hence is advantageous as a system for theorem proving. However, the naive implementation of GM^- will lead to the possibility of looping.

Two types of systems with history mechanism are considered following [Bolibekyan and Baghdasaryan 2019]. $SwMin$ system with history mechanism for propositional fragment of minimal logic was introduced in [Baghdasaryan and Bolibekyan 2020]. In this paper the system is extended to the first order logic where it is known that expressibility is drastically changed in comparison to the propositional fragment. Also a new system for first order minimal logic with another history mechanism ($ScMin$) is introduced. These systems does not contain the problem of looping, but the rule selection problem almost remains unsolved. There is a stoup selection rule, when a formula from the context should be selected to be considered as a stoup. Similar technique originally considered in [Baghdasaryan and Bolibekyan 2020] is developed and applied to the new systems and the comparison between four systems is conducted.

There are different kind of systems where rule selection problem leads to proof search inefficiency issues. For that reason those kind of systems experience some difficulties during automated theorem proving procedures.

In an attempt to resolve the latter issue machine learning methods are leveraged in automated theorem proving systems.

2 Related Work

Some Gentzen-style systems and loop detection mechanisms have been considered earlier in [Herbelin 1994], [Gabbay 1991], [Howe 1997]. This approach was extended further for propositional systems of minimal logic ([Bolibekyan and Baghdasaryan 2018], [Bolibekyan and Baghdasaryan 2019]). However, systems based on those mechanisms are covering only the propositional fragments. In order to have a system for first-order minimal logic, propositional fragment have to be extended with first-order rules.

[Alemi et al. 2016], [Bridge et al. 2014] and [Kaliszyk et al. 2017] show some approaches of applying machine learning methods in the field of automated theorem proving. [Alemi et al. 2016] introduced neural sequence models for premise selection in automated theorem proving, while [Bridge et al. 2014] applied two state of the art machine learning techniques to the problem of selecting a good heuristic in a first-order theorem prover. [Kaliszyk et al. 2017] introduced a new dataset based on Higher-Order Logic (HOL) proofs, for the purpose of developing new machine learning-based theorem-proving strategies. Also some benchmarks for different ML models suited for various tasks were showed. Overall, it is showing the promise of applying machine learning to HOL theorem proving. Our aim is about using state of the art machine learning techniques in first-order logic in order to solve the rule selection problem.

3 Loop Detection

Further in the text we follow well known definitions of a formula, sequent, proof, context, equivalence of the systems as in [Kleene 1952], [Howe 1997]. Special cases of sequents would be considered, when the sequent would contain formulas not only in the left (context) and the right (conclusion) parts from arrow ($\Gamma \Rightarrow C$) but also above the arrow ($\Gamma \xrightarrow{A} C$). That part of the formula would be called stoup and the main purpose of the stoup is the termination of the proof.

Adding a notion of history to each sequent will assist solving looping problem. Set of all sequents that have appeared in the proof tree is called history. On each step of the inference one needs to go through the checking procedure to make sure if the formula is already in the history for the considered sequence. If it is we have looping and we backtrack. If not the new formula will be added to the history and we proceed with the proof of a new sequent and that happens for each sequent of the inference tree.

Due to a huge number of sequents to be considered for specific checkings on each inference step the approach becomes quite inefficient. To prevent looping much less information can be kept.

The key point to keep the history as small as possible is adding only goal formula to that. The rules of considered systems are introduced in a way that the context cannot decrease; once a formula is in the context it will remain in the context of all sequents above it in the proof tree. Two sequents are the same only when their contexts coincide. Once the context is extended the history can be emptied, since we will never get any of

$$\frac{A,\Gamma \Rightarrow B;\epsilon}{\Gamma \Rightarrow A \supset B;H}\,(\supset R_1) \quad if A \notin \Gamma \qquad \frac{\Gamma \Rightarrow B;H}{\Gamma \Rightarrow A \supset B;H}\,(\supset R_2) \quad if A \in \Gamma$$

$$\frac{A,\Gamma \Rightarrow \bot;\epsilon}{\Gamma \Rightarrow \neg A;H}\,(\neg R_1) \quad if A \notin \Gamma \qquad \frac{\Gamma \Rightarrow \bot;H}{\Gamma \Rightarrow \neg A;H}\,(\neg R_2) \quad if A \in \Gamma$$

$$\frac{\Gamma \Rightarrow A;(C,H) \qquad \Gamma \xrightarrow{B} C;H}{\Gamma \xrightarrow{A \supset B} C;H}\,(\supset L) \quad if C \notin H$$

$$\frac{\Gamma \Rightarrow A;(C,H) \qquad \Gamma \xrightarrow{\bot} C;H}{\Gamma \xrightarrow{\neg A} C;H}\,(\neg L) \quad if C \notin H$$

$$\frac{\Gamma \xrightarrow{A} C;H}{\Gamma \xrightarrow{A \wedge B} C;H}\,(\wedge L_1) \qquad \frac{\Gamma \xrightarrow{B} C;H}{\Gamma \xrightarrow{A \wedge B} C;H}\,(\wedge L_2)$$

$$\frac{\Gamma \Rightarrow A;H \qquad \Gamma \Rightarrow B;H}{\Gamma \Rightarrow A \wedge B;H}\,(\wedge R)$$

$$\frac{A,\Gamma \Rightarrow C;\epsilon \qquad B,\Gamma \Rightarrow C;\epsilon}{\Gamma \xrightarrow{A \vee B} C;H}\,(\vee L) \quad if A,B \notin \Gamma$$

$$\frac{\Gamma \Rightarrow A;H}{\Gamma \Rightarrow A \vee B;H}\,(\vee R_1) \qquad \frac{\Gamma \Rightarrow B;H}{\Gamma \Rightarrow A \vee B;H}\,(\vee R_2)$$

$$\frac{\Gamma \Rightarrow A(b);H}{\Gamma \Rightarrow \forall x A(x);H}\,(\forall R)^{(1)} \qquad \frac{\Gamma \xrightarrow{A(t)} C;H}{\Gamma \xrightarrow{\forall x A(x)} C;H}\,(\forall L)$$

$$\frac{\Gamma \Rightarrow A(t);H}{\Gamma \Rightarrow \exists x A(x);H}\,(\exists R) \qquad \frac{\Gamma \xrightarrow{A(b)} C;H}{\Gamma \xrightarrow{\exists x A(x)} C;H}\,(\exists L)^{(1)}$$

$$\frac{A,\Gamma \xrightarrow{A} B;H}{A,\Gamma \Rightarrow B;H}\,(C)^{*} \qquad \frac{\Gamma \Rightarrow A;(A,H)}{\Gamma \xrightarrow{\bot} A;H}\,(\bot) \qquad \frac{}{\Gamma \xrightarrow{A} A;H}\,(ax)^{(2)}$$

* B is a first order formula, which doesn't contain implication, conjunction and negation. $A(x)$ is a formula; t is a term free for x in $A(x)$.
b is a variable free for x in $A(x)$ and (unless b is x) not occuring free in $A(x)$.
(1) The variable b of the postulate shall not occur free in its conclusion (when the $A(x)$ does not contain the x free, then $A(b)$ is $A(x)$ no matter what variable b is)
(2) A can be $\bot$.

Figure 1: The propositional system SwMin

$$\frac{A,\Gamma \Rightarrow B;\{B\}}{\Gamma \Rightarrow A \supset B;H}\,(\supset R_1)\quad A\notin\Gamma \qquad \frac{\Gamma \Rightarrow B;(B,H)}{\Gamma \Rightarrow A \supset B;H}\,(\supset R_2)\quad A\in\Gamma, B\notin H$$

$$\frac{A,\Gamma \Rightarrow \bot;\{\bot\}}{\Gamma \Rightarrow \neg A;H}\,(\neg R_1)\quad A\notin\Gamma \qquad \frac{\Gamma \Rightarrow \bot;(\bot,H)}{\Gamma \Rightarrow \neg A;H}\,(\neg R_2)\quad A\in\Gamma, \bot\notin H$$

$$\frac{\Gamma \Rightarrow A;(A,H) \qquad \Gamma \xrightarrow{B} C;H}{\Gamma \xrightarrow{A\supset B} C;H}\,(\supset L)\quad if A\notin H$$

$$\frac{\Gamma \Rightarrow A;(A,H) \qquad \Gamma \xrightarrow{\bot} C;H}{\Gamma \xrightarrow{\neg A} C;H}\,(\neg L)\quad if A\notin H$$

$$\frac{\Gamma \xrightarrow{A} C;H}{\Gamma \xrightarrow{A\wedge B} C;H}\,(\wedge L_1) \qquad \frac{\Gamma \xrightarrow{B} C;H}{\Gamma \xrightarrow{A\wedge B} C;H}\,(\wedge L_2)$$

$$\frac{\Gamma \Rightarrow A;(A,H) \qquad \Gamma \Rightarrow B;(B,H)}{\Gamma \Rightarrow A\wedge B;H}\,(\wedge R)\quad if A,B\notin H$$

$$\frac{A,\Gamma \Rightarrow C;\{C\} \qquad B,\Gamma \Rightarrow C;\{C\}}{\Gamma \xrightarrow{A\vee B} C;H}\,(\vee L)\quad if A,B\notin\Gamma$$

$$\frac{\Gamma \Rightarrow A;(A,H)}{\Gamma \Rightarrow A\vee B;H}\,(\vee R_1)\quad if A\notin H \qquad \frac{\Gamma \Rightarrow B;(B,H)}{\Gamma \Rightarrow A\vee B;H}\,(\vee R_2)\quad if B\notin H$$

$$\frac{\Gamma \Rightarrow A(b);H}{\Gamma \Rightarrow \forall x A(x);H}\,(\forall R)^{(1)} \qquad \frac{\Gamma \xrightarrow{A(t)} C;H}{\Gamma \xrightarrow{\forall x A(x)} C;H}\,(\forall L)$$

$$\frac{\Gamma \Rightarrow A(t);H}{\Gamma \Rightarrow \exists x A(x);H}\,(\exists R) \qquad \frac{\Gamma \xrightarrow{A(b)} C;H}{\Gamma \xrightarrow{\exists x A(x)} C;H}\,(\exists L)^{(1)}$$

$$\frac{A,\Gamma \xrightarrow{A} B;H}{A,\Gamma \Rightarrow B;H}\,(C)^{*} \qquad \frac{\Gamma \Rightarrow A;(A,H)}{\Gamma \xrightarrow{\bot} A;H}\,(\bot) \qquad \frac{}{\Gamma \xrightarrow{A} A;H}\,(ax)^{(2)}$$

* B is a first order formula, which doesn't contain implication, conjunction and negation. $A(x)$ is a formula; t is a term free for x in $A(x)$.
b is a variable free for x in $A(x)$ and (unless b is x) not occuring free in $A(x)$.
(1) The variable b of the postulate shall not occur free in its conclusion (when the $A(x)$ does not contain the x free, then $A(b)$ is $A(x)$ no matter what variable b is)
(2) A can be $\bot$.

Figure 2: The propositional system ScMin

the sequents below the extended one again. Goal formulas are the only ones to be stored in the history. If we come across a goal already in the history we have the same goal and the same context as another sequent, that is, a loop.

There are two slightly different approaches to doing this. There is the straightforward extension and modification of the system which we shall call a $SwMin$, and there is an approach which involves storing more formulas in the history, but that detects loops more quickly. This we will call as $ScMin$, and the implementation is in some cases more efficient than the $SwMin$.

In scope of considered systems A, B, C are formulas, Γ , H are finite sequences of zero or more formulas, B, Γ is shorthand for $\{B\} \cup \Gamma$, sequent $\Gamma \Rightarrow C; H$ has context Γ, goal C, history H and no stoup, and sequent $\Gamma \xrightarrow{A} C; H$ has context Γ, goal C, history H and stoup A. When the history has been extended we have parenthesised (C, H) for emphasis, while by ϵ we denote empty history. The $SwMin$ system is displayed in Figure 1, and the $ScMin$ system in Figure 2.

Theorem 1. *The systems $SwMin/ScMin$ and GM^- are equivalent.*

The idea in proof is focused on constructing proof trees based on a given pattern in a first system and considering inference rule under the focus as in [Bolibekyan and Baghdasaryan 2019] for propositional fragment.

4 Rule Selection

In $SwMin/ScMin$ systems the rule selection problem remains unsolved. There is a stoup selection rule, when a formula from the context should be selected to be considered as a stoup. $SwProv$ and $ScProv$ automated theorem provers are developed based on $SwMin$ and $ScMin$ systems respectively. In order to remove the rule selection problem $SwProv$ and $ScProv$ automated theorem provers were powered by neural networks, which will be called $SwNNProv$ and $ScNNProv$ respectively. At every point of the proof tree, where stoup selection rule occurred, the context formula which will be selected as a stoup would be determined based on neural networks prediction.

4.1 Sequent To Vector Transformation

Skolem normal form representations are used in order to have more structured formulas. Numerical representations for the sequents are introduced by assigning a specific number to each symbol in the formula. As a result, similar sequents get identical representation vectors (the respresentations of formulas $(A \supset B) \supset A$ and $(C \supset D) \supset C$ are the same).

For assigning fixed length encoding to each sequent in the proof tree autoencoders [Baldi 2012] are applied. Two slightly different approaches were considered. The first one (AE) is based on the concepts of undercomplete autoencoders and is using L2 loss function:

$$Loss_{AE} = \frac{1}{2N} \sum_{i=1}^{N} (x^{(i)} - \hat{x}^{(i)})^2, \tag{1}$$

where N is the number of examples, x is the numerical representation of the sequent, while $\hat{x}$ is the output of the autoencoder.

Method	Retention Loss
AE	0.015
CAE	0.014

Fig. 3: Autoencoders retention quality.

Method	Precision	Recall	F1	Accuracy
AE + GRU	0.7	0.74	0.72	72%
CAE + GRU	0.7	0.75	0.72	73%
AE + LSTM	0.71	0.75	0.73	73%
CAE + LSTM	0.72	0.78	0.75	75%

Fig. 4: Quality metrics for SwNNProv prover.

In the second approach (CAE) the loss function is modified using the idea of contractive autoencoders [Rifai et al. 2011]. As the representations of the similar sequents are close to each other, the impact of the changes in the input to the hidden layer has to be minimized.

$$Loss_{CAE} = \frac{1}{2N}\sum_{i=1}^{N}(x^{(i)} - \hat{x}^{(i)})^2 + \lambda\frac{1}{N}\sum_{i=1}^{N}\left\|\nabla_{x^{(i)}}h(x^{(i)})\right\|_F^2 \tag{2}$$

where

$$\left\|\nabla_{x^{(i)}}h(x^{(i)})\right\|_F^2 = \sum_{j=1}^{M}\sum_{k=1}^{L}\left(\frac{\partial h(x^{(i)})_j}{\partial x_k^{(i)}}\right)^2 \tag{3}$$

h is the hidden layer representation, N is the number of examples, M is the hidden layer size, L is the input layer size, x is the numerical representation of the sequent, $\hat{x}$ is the output of the autoencoder, λ is the regularization hyper-parameter.

So, $\nabla_{x^{(i)}}h(x^{(i)})$ is the Jacobian of the encoder and the minimisation of the Frobenius norm of the Jacobian matrix makes the encoder to become less sensitive to the input changes. Figure 3 is showing the comparison between two approaches in terms of retention quality.

4.2 Recurrent Neural Networks in Proof Search

A standard library of propositional and first-order logic problems is used as a training dataset. More than 100000 formulas are generated with the help of 5000 predefined formulas of minimal logic. Dataset was extended using some formulas from the TPTP problem library [Sutcliffe 2017]. Training examples are generated by passing each element of training set to $SwProv/ScProv$ prover. Every point of proof tree, where a stoup formula has to be selected from the context, all sequents in that branch of tree are considered and sequence of vectors is generated by the procedure described in "Sequent to Vector transformation" chapter. To differentiate successful outcomes while training the neural network one needs to take numerical representation for each stoup candidate formula and corresponding ground truth label (whether this is the right selection).

Two types of recurrent neural network are deployed, where the first one consists of gated recurrent unit (GRU) [Cho et al. 2014] as recurrent layer and 2-dense layers with skip connections [Fischer et al. 2015], while the second one consists of LSTM [Sundermeyer et al. 2012] layer and 2-dense layers with skip connections. The output of recurrent layer (feature vector extractor module) is concatenated with numerical representation of stoup candidate formula and then is mapped into 2-length one-hot encoded vector via dense layer with softmax activation function. As a final step cross

entropy is used as a loss function:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N}(y_i * \log(p(y_i)) + (1 - y_i) * \log(1 - p(y_i)), \quad (4)$$

where y is the label and $p(y)$ is the predicted probability of the candidate formula being right for all N examples.

As it is more important not to omit the right subtrees (misclassify as 0) rather than to keep wrong subtrees (misclassy as 1), loss function is modified by adding penalization term for recall. Also, in order to have a higher value for recall metric prediction threshold is changed. Figure 4 shows the comparison of different approaches for $SwNNProv$ prover in terms of precision, recall, F1 score and accuracy metrics calculated on the testing dataset. It becomes clear, that the combination of contractive autoencoder compression and network with LSTM cells shows the best result (75% accuracy, 0.78 recall).

As a result, the bad branches are removed from the proof tree in almost 75% of cases, which reduces automated theorem proving duration.

4.3 Inference

In order to reduce processing time during automated theorem proving, recurrent neural networks are deployed. During theorem proving, at each point of proving tree, where rule selection problem appears, all the candidate stoup formulas with the respective proving subtrees are passed through neural network. For all the candidate formulas probabilities were obtained via RNN. Then the candidate formula with the highest probability is selected as stoup and proof tree is extended in that direction. If it brings to unsuccessful proof, it continues with selecting the other candidate formula as stoup with the higher probability and so on. For non-provable sequents theorem prover is exploring the whole space of candidate formulas and brings up with non-provable conclusion.

As the recurrent neural network inference is computationally very expensive and the main goal was to reduce time consumption during proof search, some inference time reductions and benchmarkings are done. NVIDIA TensorRT is applied to the model for getting the benefits like layer/tensor fusion and automatic precision calibration during the inference. It turns out, that using float16 instead of float32 during inference does not affect significantly to the accuracy of the model, while it drastically speeds up the inference. Figure 5 shows the comparison between several GPUs with different architectures as well as the models with and without TesnorRT and with float32 or float16 arithmetics in terms of how many inferences can be done in a second.

GPU	Architecture	without TensorRT (inf/sec)	TensorRT float32 (inf/sec)	TensorRT float16 (inf/sec)
Nvidia P100	Pascal	870	1200	1400
Nvidia K80	Kepler	250	350	400
Nvidia V100	Volta	1100	1600	3100
Nvidia T4	Turing	740	1000	1900
Nvidia Jetson Nano	Maxwell	50	65	70

Fig. 5: Inference benchmark for $SwNNProv$.

5 Results

In order to demonstrate the power of the RNN based provers the following formula was passed to the SwProv and SwNNProv:

$\Rightarrow (A \supset B) \supset ((A \supset \neg B) \supset \neg A)$

The following part is the same for the both provers:

$$\dfrac{\dfrac{\dfrac{\dots}{(A \supset B), (A \supset \neg B), A \Rightarrow \bot}}{(A \supset B), (A \supset \neg B) \Rightarrow \neg A}}{\Rightarrow (A \supset B) \supset ((A \supset \neg B) \supset \neg A)}$$

At this point, rule selection problem has been detected. After passing the proof subtree and candidate formulas to the RNN, SwNNProv prover predicts that the "right" choice for the stoup is $(A \supset \neg B)$:

$$\dfrac{\dfrac{(A \supset B), (A \supset \neg B), A \Rightarrow A \qquad \dfrac{\dfrac{\dots}{(A \supset B), (A \supset \neg B), A \Rightarrow B}}{(A \supset B), (A \supset \neg B), A \xrightarrow{\neg B} \bot}}{(A \supset B), (A \supset \neg B), A \xrightarrow{A \supset \neg B} \bot}}{(A \supset B), (A \supset \neg B), A \Rightarrow \bot}$$

Rule selection problem occurs one more time. At this time SwNNprov recommends to continue with $(A \supset B)$ as a stoup:

$$\dfrac{\dfrac{(A \supset B), (A \supset \neg B), A \Rightarrow A \qquad (A \supset B), (A \supset \neg B), A \xrightarrow{B} B}{(A \supset B), (A \supset \neg B), A \xrightarrow{A \supset B} B}}{(A \supset B), (A \supset \neg B), A \Rightarrow B}$$

In contrast, SwProv will make many unnecessary computations while trying to enrich the proof tree using all the candidate stoup formulas.

In result of constructing new proof systems for the minimal first-order logic and deploying concept of neural network in the prover experiments revealed proof search space reduction and the level of accuracy up to 75% training 150 epochs.

Compared to the prover without neural network time spent for the proof is reduced for almost twice. Figure 6 shows the comparison between $SwProv/ScProv$ (based on $SwMin/ScMin$ system, with the rule selection problem) and $SwNNProv/ScNNProv$ (LSTM network powered prover) automatic theorem provers. Experiments are done on NVIDIA 1080Ti GPU with 8 core processors. In complex formulas RNN powered provers obviously are performing much efficient and faster.

6 Conclusion and Future Work

In this paper, three main problems of theorem proving with a Gentzen-style cut-free system of minimal logic are considered. The main contribution of this work is the extension of propositional fragments of minimal logic systems with first-order logic rules as well as the approach to solve the rule selection problem, in the way of expressing it

as a machine learning problem and proposing methods for solving it based on recurrent neural networks. A discussion on different representations of sequents has been provided. In particular, representations using two types of autoencoders have been proposed.

Example	SwProv	SwNNProv	ScProv	ScNNProv	Provable
$(A \wedge \neg A) \supset B$	1.7	1.5	1.7	1.6	No
$(A \supset B) \supset (A \supset C) \supset (A \supset (B \supset C))$	2.6	1.2	2.9	1.3	Yes
$((\neg\neg A \supset A) \supset A) \vee (\neg A \supset \neg A) \vee (\neg\neg A \supset \neg\neg A) \vee (\neg\neg A \supset A)$	4.4	4.6	4.3	4.6	Yes
$\neg\neg((\neg A \supset B) \supset (\neg A \supset \neg B) \supset A)$	42	25	33	19	Yes
$((((A \supset B) \supset ((B \supset C) \supset (A \supset C)))) \supset C) \supset ((B \supset C) \supset (((((A \supset B) \supset ((B \supset C) \supset (A \supset C)))) \vee B) \supset C))$	37	16	45	21	Yes
$(((((A \supset B) \supset ((C \supset B) \supset ((A \vee C) \supset B)))) \supset C) \supset (((((A \supset B) \supset ((C \supset B) \supset ((A \vee C) \supset B)))) \supset \neg C) \supset \neg(((A \supset B) \supset ((C \supset B) \supset ((A \vee C) \supset B))))))$	129	15	183	27	Yes
$(((G \supset A) \supset J) \supset ((P \vee (Q \wedge P)) \supset P) \supset E) \supset (((H \supset B) \supset I) \supset C \supset J \supset (A \supset H) \supset F \supset G \supset (((C \supset C) \supset I) \supset ((P \vee (Q \wedge P)) \supset P)) \supset (A \supset C) \supset (((F \supset A) \supset B) \supset I) \supset E)$	869	174	950	213	Yes
$((((((G \supset A) \supset J) \supset D \supset E) \supset (((H \supset B) \supset I) \supset C \supset J \supset (A \supset H) \supset F \supset G \supset (((C \supset B) \supset I) \supset D) \supset (A \supset C) \supset (((F \supset A) \supset B) \supset I) \supset E)) \wedge B) \supset ((((G \supset A) \supset J) \supset D \supset E) \supset (((H \supset B) \supset I) \supset C \supset J \supset (A \supset H) \supset F \supset G \supset (((C \supset B) \supset I) \supset D) \supset (A \supset C) \supset (((F \supset A) \supset B) \supset I) \supset E))$	1359	96	1743	121	Yes
$\neg\exists x\forall y(q(y) \supset r(x,y)) \wedge \exists x\forall y(s(y) \supset r(x,y)) \supset \neg\forall x(q(x) \supset s(x))$	152	32	173	36	Yes
$(\forall x((f(x) \wedge (g(x) \vee h(x))) \supset q(x)) \wedge (\forall y((q(y) \wedge h(y)) \supset p(y)) \wedge \forall z(k(z) \supset h(z)))) \supset \forall t((f(t) \wedge k(t)) \supset p(z))$	72	12	79	14	Yes

Fig. 6: Proving time comparison (in miliseconds).

In order to solve the rule selection problem a new approach with recurrent neural networks was introduced. Our contribution in this part was to adapt these algorithms to the automated theorem provers for reducing the proof tree, which to the best of our knowledge have never been addressed before. Also, some optimizations and benchmarkings are done in order to have a faster models during inference of neural networks.

From an experimental point of view, our contribution lies in the comparison of models developed for solving the rule selection problem, which are using two methods of compression and two types of recurrent neural networks with LSTM and GRU cells. Results are showing that our approach obtains a good outcome and it reduces the proof time for almost twice. Future research should be devoted to the development of new types of machine learning models (bidirectional RNNs, attention mechanisms) and to the training of new models based on enriched dataset of first order minimal logic formulas. Future research could continue to explore the modal fragments (S4, S5) of minimal logic.

References

[Alemi et al. 2016] Alexander A. Alemi, François Chollet, Niklas Eén, Geoffrey Irving, Christian Szegedy, Josef Urban: "DeepMath - deep sequence models for premise selection". NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2243–2251 (2016)

[Baghdasaryan and Bolibekyan 2020] A.R. Baghdasaryan, H.R. Bolibekyan: "On Machine Learning Powered Theorem Prover for Propositional Fragment of Minimal Logic", Collaborative Technologies and Data Science in Artificial Intelligence Applications, Logos Verlag Berlin, pp. 135-142 (2020)

[Baldi 2012] P. Baldi: Autoencoders, "Unsupervised Learning and Deep Architectures", (2012)

[Bolibekyan and Baghdasaryan 2018] Bolibekyan H., Baghdasaryan A.: "On some systems of minimal predicate logic with history mechanism", The Bulletin of Symbolic Logic, Volume 24, Number 2, pp. 232-233 (2018)

[Bolibekyan and Baghdasaryan 2019] H.R. Bolibekyan, A.R. Baghdasaryan: "On some systems of propositional minimal logic with loop detection", Reports of NAS RA, Volume 119, Number 2, pp. 110-115 (2019)

[Bolibekyan and Chubaryan 2002] Bolibekyan H.R., Chubaryan A.A.: "On the sequent systems of weak arithmetics", Doklady National'noy Akademii Nauk RA, vol. 102, N3, pp. 214-218 (in Russian), (2002)

[Bridge et al. 2014] Bridge, J.P., Holden, S.B., Paulson, L.C.: "Machine Learning for First-Order Theorem Proving". J Autom Reasoning 53, 141–172 (2014)

[Cho et al. 2014] K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio: "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches", (2014)

[Fischer et al. 2015] P. Fischer, O. Ronneberger, T. Brox: U-Net: "Convolutional Networks for Biomedical Image Segmentation", (2015)

[Gabbay 1991] Gabbay D .: "Algorithmic proof with diminishing resources.", Springer Lecture Notes in Computer Science, vol 1, pp.156-173. (1991)

[Herbelin 1994] Herbelin, H. A λ-calculus Structure Isomorphic to Gentzen-style Sequent Calculus Structure. Springer Computer Science Logic LNCS, v. 933, pp. 61-75. (1994).

[Heyting 1930] A. Heyting, Sur la logique intuitionniste, Bull. Acad. Royale Belgique 16 (1930), 957–963.

[Howe 1997] Howe J.M.: "Two Loop Detection Mechanisms: a Comparison.", Springer Lecture Notes in Artificial Intelligence, Volume 1227, pp. 188–200. (1997)

[Kaliszyk et al. 2017] C. Kaliszyk, F. Chollet, C. Szegedy: HolStep: "A machine learning dataset for higher-order logic theorem proving". ICLR (2017)

[Kleene 1952] Kleene S.C.: "Introduction to metamathematics.", D.Van Nostrand Comp., Inc., New York- Toronto, (1952)

[Rifai et al. 2011] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio: "Contractive Auto-Encoders: Explicit Invariance During Feature Extraction", ICML'11: Proceedings of the 28th International Conference on International Conference on Machine Learning, pp. 833–840, (2011)

[Sundermeyer et al. 2012] M. Sundermeyer, R. Schluter, H. Ney: "LSTM neural networks for language modeling", INTERSPEECH, pp. 194–197, (2012)

[Sutcliffe 2017] G. Sutcliffe: "The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0", Journal of Automated Reasoning, Volume 59, Number 4, pp. 483-502, (2017)