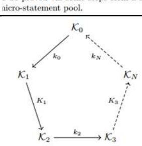
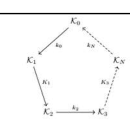
Changelog from revision 0 to revision 1…

Zijian Wang felix_wzj@yahoo.com 8/11/2022

<table>
<tr><th>No.</th><th>Modification</th><th>In the past</th><th>Now</th></tr>
<tr><td>1</td><td>Insert date of compiling</td><td>/</td><td>August 11, 2022</td></tr>
<tr><td>2</td><td>Change the style of abstract</td><td>Abstract
In this article I present a new type of automated statement proof algorithm based on new data structures, i.e. brackets and map graphs and new algorithms. The brackets provide an elegant low-knowledge representation of mathematical concepts. The map graphs offer an efficient machine-learning method which let the computer learn knowledge while proving. Additionally, the new finding is totally built on the category theory. Furthermore, a prototype of the program is given.</td><td>Abstract
In this article a new type of automated statement proof algorithm based on new data structures, i.e. brackets and map graphs and new algorithms is presented. The brackets provide an elegant low-knowledge representation of mathematical concepts. The map graphs offer an efficient machine-learning method which let the computer learn knowledge while proving. Additionally, the new finding is totally built on the category theory. Furthermore, a prototype of the program is given.</td></tr>
<tr><td>3</td><td>Number sections</td><td>Introduction</td><td>1 Introduction</td></tr>
<tr><td>4</td><td>Add references to section 1</td><td>During the long period of development of the research on automated theorem provers, there have already existed a great number of research and implementation on proofs focusing on a specific mathematical subject, such as algebra equations (for example, Wolfram Mathematica's FindEquationProof function [1]) or geometrical theorems (for example, the Geometry Expert or GEX [2] of Key Laboratory of Mathematics Mechanization, Chinese Academy of Sciences). Some relatively new works in this aspect includes Microsoft's Lean prover and Z3 prover.</td><td>During the long period of development of the research on automated theorem provers, there have already existed a great number of research and implementation on proofs focusing on a specific mathematical subject, such as algebra equations (for example, Wolfram Mathematica's FindEquationProof function [9]) or geometrical theorems (for example, the Geometry Expert or GEX [4] of Key Laboratory of Mathematics Mechanization, Chinese Academy of Sciences). Some relatively new works in this aspect includes Microsoft's Lean prover [6] and Z3 [1] algorithm.</td></tr>
<tr><td>5</td><td>Add structures of paper</td><td>Structure of the paper. To be finished later.</td><td>Structure of the paper.
The paper mainly consists of seven sections. The Introduction section demonstrates the background of the research and fundamental opinions around it. The Preliminaries section, made up of two subsections, claims to minimized pre-knowledge required to understand the paper and the symbols I select to use. The third section defines a set of mathematical structures that will be used by the algorithm. The forth section explains the detailed workflow procedures of the algorithm. The next section claims the conclusions of the research and the to-do improvements in the future. Finally the Acknowledgments section expresses my gratitude to the help and the References section claims the referential resources used while conducting the project.</td></tr>
<tr><td>6</td><td>Fix formatting</td><td>where A and B are sets and$x$and y are elements in A and B.</td><td>$$f : A \to B$$ $$x \mapsto y$$ where A and B are sets and$x$and y are elements in A and B.</td></tr>
<tr><td>7</td><td>Remove proof of T1 & number theorems</td><td>Theorem. The Zermelo's Well-Ordering Theorem. [3]
Every set S can be well-ordering as long as it has a choice function.
Proof: First, we select element $\mho \notin S$. As the ZFC system's selection axiom enables to select element $g(S')$ from each subset S' of S. Let a_0:=$g(S)$. Using transfinite recursive theorem to each ordinal α we define:
$$a_\alpha := \begin{cases} g(S\backslash\{a_\beta \mid \beta < \alpha\}), & S \neq \{a_\beta \mid \beta < \alpha\} \\ \mho, & S = \{a_\beta \mid \beta < \alpha\} \end{cases}$$</td><td>Theorem 2.1 The Zermelo's Well-Ordering Theorem. [5]
Every set S can be well-ordering as long as it has a choice function.
The proof can be found in citation [5].</td></tr>
<tr><td>8</td><td>Format tables</td><td><table><tr><td>$\mathcal{N}$</td><td>integer</td><td>number</td><td>infinity</td><td>$\sum$</td><td>=</td><td>/</td><td>−</td><td>abs</td><td><</td><td>for</td></tr><tr><td>$p(\mathcal{N})$</td><td>0</td><td>0</td><td>0</td><td>3</td><td>2</td><td>2</td><td>2</td><td>1</td><td>2</td><td>2</td></tr></table></td><td><table><tr><th>$\mathcal{N}$</th><th>integer</th><th>number</th><th>infinity</th><th>$\sum$</th><th>=</th><th>/</th><th>-</th><th>abs</th><th><</th><th>for</th></tr><tr><td>$p(\mathcal{N})$</td><td>0</td><td>0</td><td>0</td><td>3</td><td>2</td><td>2</td><td>2</td><td>1</td><td>2</td><td>2</td></tr></table>Table 1: The table for $p(\mathcal{N})$</td></tr>
<tr><td>9</td><td>Fix formatting of equations</td><td>$, (1, \boldsymbol{n}), (2, \mathbf{1})\}), (2, \infty),$</td><td>$(1, \boldsymbol{n}), (2, 1)\}), (2, \infty), (3, \{(0,$</td></tr>
<tr><td>10</td><td>Format definitions</td><td>Definition. A derivation is a set of ordered pairs of ordinals and categories in the form below.
$$\mathcal{D} := \{(i, \mathcal{M}_i) \mid i \in \mathrm{On}_{\geq 1}\}$$ Where $\mathcal{M}_i$ is a micro-statement pool and for each element in $\mathcal{D}$, its left pair element is a unique ordinal. The $\mathcal{D}$ set is well-ordered following the sort of i. On the beginning, the the initial pool equals to the condition, that is the micro-statements given by the problem. On the other hand, the conclusions of the proof problem, to be proved via some steps form a set $\mathcal{R}$, called the requirement set are to be inferred to exist inside the micro-statement pool.</td><td>Definition 4.1 A derivation is a set of ordered pairs of ordinals and categories in the form below.
$$\mathcal{D} := \{(i, \mathcal{M}_i) \mid i \in On_{\geq 1}\}$$ where $\mathcal{M}_i$ is a micro-statement pool and for each element in $\mathcal{D}$, its left pair element is a unique ordinal. The $\mathcal{D}$ set is well-ordered following the sort of i. On the beginning, the the initial pool equals to the condition, that is the micro-statements given by the problem. On the other hand, the conclusions of the proof problem,</td></tr>
<tr><td>11</td><td>Center and name figures</td><td></td><td>
Figure 1: The initial map.</td></tr>
<tr><td>12</td><td>Add pseudo code algorithms</td><td>/</td><td>The more detailed explanation of the recursive functor can be found in the algorithm below.
Algorithm 1 Pseudo code for recursive functor $\mathcal{F}_t$
Require: Pool $\mathcal{M}_t$, Knowledge base $\mathcal{Q}$, random r, Just used reflection functor index i
Ensure: Next pool $\mathcal{M}_{t+1}$
function $\mathcal{F}_t(\mathcal{M}_t, \mathcal{Q}, r, i)$ ▷ This is the sub-process
$\mathcal{K} \leftarrow$ SelectReflectionFunctor($\mathcal{Q}$, r, i)
return $\mathcal{K}\mathcal{M}_t$ ▷ Apply functor $\mathcal{K}$ to pool
end function
function SelectReflectionFunctor($\mathcal{Q}$, r, i) ▷ Just used: $\mathcal{K}_i \in \mathrm{Ob}(\mathcal{Q})$
$M \leftarrow Mor(\mathcal{Q})$
for all $m \in M$ do ▷ Get morphisms starting at $\mathcal{K}_i$
if $s(m)! = \mathcal{K}_i$ then
$M \leftarrow M\backslash\{m\}$
end if
end for
if $|M| == 1$ then ▷ The easiest condition
return $t(m \in M)$
else
$n \leftarrow r\%|M|$
return $t(n)$
end if
end function</td></tr>
</table>

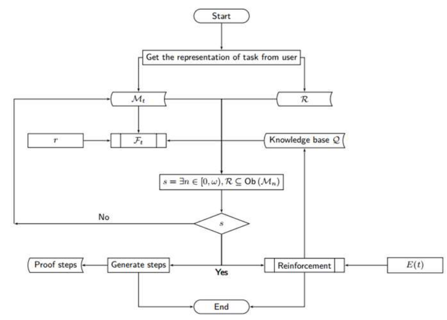
<table>
<tr><td>13</td><td>Add a flowchart about the entire workflow</td><td>/</td><td>

Figure 3: The flowchart of the entire algorithm.</td></tr>
<tr><td>14</td><td>Fix English name of NEYC</td><td>I express my sincerest greatest gratitude to my tutor Prof. Shao Xinhui, the Yingcai Project of P.R.Chi which enables senior high school students to conduct science research, my senior high academy North Eastern Yucai School (NEYC) of P.R.China and the corresponding university Northern Eastern Univers (NEU) of P.R.China. I also thank my parents for encouraging me to conduct this research project.</td><td>I express my sincerest gratitude to my tutor Prof. Shao Xinhui, the Yingcai Project of P.R.China, w enables senior high school students to conduct science research, my senior high academy Northeast cai School (NEYC) of P.R.China and the corresponding university Northern Eastern University (NEU P.R.China. I also thank my parents for encouraging me to conduct this research project.</td></tr>
<tr><td>15</td><td>Expand acknowledgements</td><td>/</td><td>Additionally, I had few knowledge about LaTeXbefore writing this paper. The lshort documentation and the TeXStack Exchange helped me a lot. Here I express my genuine thankfulness to the online resources and documentation of the used LaTeXpackages and software components.</td></tr>
<tr><td>16</td><td>Change reference style</td><td>References
[1] S. Wolfram, Wolfram Language & System Documentation Center. Wolfram Research Inc., 2019.
[2] X.-S. Gao, “Geoexpert,” 1998.
[3] W.-W. Li, Daishuxue Fangfa. High Education Press, 1 ed., 2019.
[4] H. Ebbinghaus, Memory. Teachers College, Columbia University, 1913.
[5] K. Entacher, “A collection of selected pseudorandom number generators with linear structures,” 1999.
[6] A. S. Tanenbaum, Operating Systems: Design and Implementatiion. Publishing House of Electronics Industry, 3 ed., 2015.
[7] Z. Wang, “The defqed repository,” 2022.</td><td>References
[1] de Moura, L., and Bjørner, N. Z3: an efficient smt solver. In Tools and Algorithms for the Construction and Analysis of Systems (Berlin, Heidelberg, 04 2008), vol. 4963, Springer Berlin Heidelberg, pp. 337–340.
[2] Ebbinghaus, H. Memory: A contribution to experimental psychology. Teachers College, Columbia University, New York City, 1913.
[3] Entacher, K. A collection of selected pseudorandom number generators with linear structures.
[4] Gao, X.-S. Geoexpert. Beijing, 1998.
[5] Li, W.-W. Daishuxue Fangfa, 1 ed., vol. 1. High Education Press, Beijing, 2019.
[6] Moura, L. d., and Ullrich, S. The lean 4 theorem prover and programming language. In Automated Deduction – CADE 28 (Cham, 2021), Springer International Publishing, pp. 625–635.
[7] Tanenbaum, A. S. Operating Systems: Design and Implementatiion, 3 ed. Publishing House of Electronics Industry, 2015.
[8] Wang, Z. The defqed repository. online, 2022.
[9] Wolfram, S. Wolfram Language & System Documentation Center. Wolfram Research Inc., 2019.</td></tr>
<tr><td>17</td><td>Add 2 references</td><td>[1] [2] [3] [4] [5] [6] [7]</td><td>[1] [2] [3] [4] [5] [6] [7] [8] [9]</td></tr>
<tr><td>18</td><td>Change Preliminary section text</td><td>Readers should have basic understandings of the category theory [3] and axiomatic set theory. At least, they should know the basic concepts as well as these conclusions and theorems:</td><td>The definitions and theories presented in the paper utilizes the concepts of the category theory [5] and axiomatic set theory [5]. Therefore, readers are recommended to review understandings and conclusions about the two concepts before going over the paper.</td></tr>
</table>

The document is currently using Git for version control. The remote repo is: [ZijianFelixWang/ANT-APCT-Paper: Source for paper A new type of automated prover based on category theory. (github.com)]

The current revisions can be found under rev1 branch. The draft before revision can be found under rev0 branch.