https://arxiv.org/pdf/2603.21852v2

All elementary functions from a single operator

Andrzej Odrzywołek
Institute of Theoretical Physics, Jagiellonian University, 30-348 Krakow, Poland
E-mail: andrzej.odrzywolek@uj.edu.pl

April 7, 2026

先日，とある論文が話題になりました．

https://arxiv.org/pdf/2603.21852v2

$$\exp, \ln, \mathrm{inv}, \mathrm{half}, \mathrm{minus}, \sqrt{\ }, \mathrm{sqr}, \sigma,$$
$$\sin, \cos, \tan, \arcsin, \arccos, \arctan,$$
$$\sinh, \cosh, \tanh, \mathrm{arsinh}, \mathrm{arcosh}, \mathrm{artanh}$$

この論文によると，
累乗根や三角関数，指数関数，対数関数などの初等関数や，

https://arxiv.org/pdf/2603.21852v2

$$\pi, e, i, -1, 1, 2, x, y$$
$$+, -, \times, /, \log, \mathrm{pow}, \mathrm{avg}, \mathrm{hypot}$$

円周率などの数学定数，四則演算などは，

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y)$$

このような二項演算子と1を用いて表すことができるというのです.

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y)$$

今回は，このEML演算子について，噛み砕いて解説していきます．

MathAbyss

MathAbyss

EML演算子とは何か

EML演算子とは何か

1.イントロダクション
2.EML論文の概要
3.EML演算子
4.具体的な計算
5.まとめ

この動画は，5つのチャプターによって構成されています．

MathAbyss

1.イントロダクション
2.EML論文の概要
3.EML演算子
4.具体的な計算
5.まとめ

まず，この論文の概要について簡単に説明し，

EML演算子とは何か

1.イントロダクション
2.EML論文の概要
3.EML演算子
4.具体的な計算
5.まとめ

次に，論文で定義されているEML演算子について解説します．

1.イントロダクション
2.EML論文の概要
3.EML演算子
4.具体的な計算
5.まとめ

最後に，主張の証明や，EMLの展望についてお話します．

※注意※

この動画は，EMLについての論文を元に制作していますが，
他の動画と比べて短い期間で制作しているため，
誤った内容を含む可能性があります．

※注意※

誤りだと思われる箇所を見つけられた場合，ご指摘いただけると幸いです．

※注意※

動画の訂正内容等については,
固定コメントや概要欄にて随時更新していきます.

1.イントロダクション
2.EML論文の概要
3.EML演算子
4.具体的な計算
5.まとめ

それでは，本編のスタートです！

2. EML論文の概要

まずは，論文の概要について説明します．

2. EML論文の概要

1.EML論文の主張

2.EML論文の著者

3.NANDゲート

4.EML論文の意義

2. EML論文の概要

1.EML論文とは
2.EML論文の著者
3.NANDゲート
4.EML論文の意義

このチャプターでは，4つの内容をお話します．

EML論文とは

まずは，EML論文について簡単に紹介しておきます．

EML論文とは

https://arxiv.org/abs/2603.21852v2

Cornell University

Learn about arXiv becoming an independent nonprofit.

We gratefully acknowledge support from the Simons Foundation, member institutions, and all contributors. Donate

arXiv > cs > arXiv:2603.21852v2

Search... All fields Search

Help | Advanced Search

Computer Science > Symbolic Computation

[Submitted on 23 Mar 2026 (v1), last revised 4 Apr 2026 (this version, v2)]

All elementary functions from a single binary operator

Andrzej Odrzywołek

A single two-input gate suffices for all of Boolean logic in digital hardware. No comparable primitive has been known for continuous mathematics: computing elementary functions such as sin, cos, sqrt, and log has always required multiple distinct operations. Here I show that a single binary operator, eml(x,y)=exp(x)−ln(y), together with the constant 1, generates the standard repertoire of a scientific calculator. This includes constants such as e, pi, and i; arithmetic operations including addition, subtraction, multiplication, division, and exponentiation as well as the usual transcendental and algebraic functions. For example, exp(x)=eml(x,1), ln(x)=eml(1,eml(eml(1,x),1)), and likewise for all other operations. That such an operator exists was not anticipated; I found it by systematic exhaustive search and established constructively that it suffices for the concrete scientific-calculator basis. In EML (Exp-Minus-Log) form, every such expression becomes a binary tree of identical nodes, yielding a grammar as simple as S -> 1 | eml(S,S). This uniform structure also enables gradient-based symbolic regression: using EML trees as trainable circuits with standard optimizers (Adam), I demonstrate the feasibility of exact recovery of closed-form elementary functions from numerical data at shallow tree depths up to 4. The same architecture can fit arbitrary data, but when the generating law is elementary, it may recover the exact formula.

Comments: 2 figures, Supplementary Information, code available at this https URL
Subjects: Symbolic Computation (cs.SC); Machine Learning (cs.LG)
MSC classes: 26A09 (Primary) 08A40, 68W30 (Secondary)
ACM classes: I.1.1; F.1.1
Cite as: arXiv:2603.21852 [cs.SC]
(or arXiv:2603.21852v2 [cs.SC] for this version)
https://doi.org/10.48550/arXiv.2603.21852

Submission history
From: Andrzej Odrzywolek [view email]
[v1] Mon, 23 Mar 2026 11:40:24 UTC (1,393 KB)
[v2] Sat, 4 Apr 2026 06:31:05 UTC (1,245 KB)

Access Paper:
View PDF
HTML (experimental)
TeX Source
view license

Ancillary files (details):
• SupplementaryInformation.pdf

Current browse context:
cs.SC
< prev | next >
new | recent | 2026-03
Change to browse by:
cs
cs.LG

References & Citations
NASA ADS
Google Scholar
Semantic Scholar
export BibTeX citation

Bookmark

Bibliographic Tools | Code, Data, Media | Demos | Related Papers | About arXivLabs

Bibliographic and Citation Tools

Bibliographic Explorer (What is the Explorer?)

EML論文は，2026年3月23日に提出され，
同年4月4日に更新されている論文です．

https://arxiv.org/abs/2603.21852v2

Computer Science > Symbolic Computation

[Submitted on 23 Mar 2026 (v1), last revised 4 Apr 2026 (this version, v2)]

All elementary functions from a single binary operator

Andrzej Odrzywołek

A single two-input gate suffices for all of Boolean logic in digital hardware. No comparable primitive has been known for continuous mathematics: computing elementary functions such as sin, cos, sqrt, and log has always required multiple distinct operations. Here I show that a single binary operator, eml(x,y)=exp(x)−ln(y), together with the constant 1, generates the standard repertoire of a scientific calculator. This includes constants such as e, pi, and i; arithmetic operations including addition, subtraction, multiplication, division, and exponentiation as well as the usual transcendental and algebraic functions. For example, exp(x)=eml(x,1), ln(x)=eml(1,eml(eml(1,x),1)), and likewise for all other operations. That such an operator exists was not anticipated; I found it by systematic exhaustive search and established constructively that it suffices for the concrete scientific-calculator basis. In EML (Exp-Minus-Log) form, every such expression becomes a binary tree of identical nodes, yielding a grammar as simple as S -> 1 | eml(S,S). This uniform structure also enables gradient-based symbolic regression: using EML trees as trainable circuits with standard optimizers (Adam), I demonstrate the feasibility of exact recovery of closed-form elementary functions from numerical data at shallow tree depths up to 4. The same architecture can fit arbitrary data, but when the generating law is elementary, it may recover the exact formula.

Comments: 2 figures, Supplementary Information, code available at this https URL
Subjects: **Symbolic Computation (cs.SC)**; Machine Learning (cs.LG)
MSC classes: 26A09 (Primary) 08A40, 68W30 (Secondary)
ACM classes: I.1.1; F.1.1
Cite as: arXiv:2603.21852 [cs.SC]
(or arXiv:2603.21852v2 [cs.SC] for this version)
https://doi.org/10.48550/arXiv.2603.21852

Submission history

From: Andrzej Odrzywolek [view email]
[v1] Mon, 23 Mar 2026 11:40:24 UTC (1,393 KB)
[v2] Sat, 4 Apr 2026 06:31:05 UTC (1,245 KB)

著者については次のセクションで紹介します．

$$\pi, e, i, -1, 1, 2, x, y$$

この論文は，これらの数学定数や変数，

$$\sin, \cos, \tan,$$
$$\arcsin, \arccos, \arctan$$

これらの三角関数,

$$\sinh, \cosh, \tanh,$$
$$\operatorname{arsinh}, \operatorname{arcosh}, \operatorname{artanh}$$

双曲線関数,...

$$\exp, \ln, \sqrt{}$$

指数関数や対数関数といった初等関数に加え，

$+, -, \times, /$

四則演算などの36個の要素が,

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y)$$

この式で定義されるEML演算子と1によって書けることを主張しています.

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y)$$

様々な定数や関数が，
EML演算子と1のみで表すことができるという内容は，
SNS上で話題となり，拡散されました．

https://arxiv.org/pdf/2603.21852v2

All elementary functions from a single operator

Andrzej Odrzywołek

Institute of Theoretical Physics, Jagiellonian University, 30-348 Krakow, Poland

E-mail: andrzej.odrzywolek@uj.edu.pl

April 7, 2026

さて，論文のタイトルを見てみましょう．

https://arxiv.org/pdf/2603.21852v2

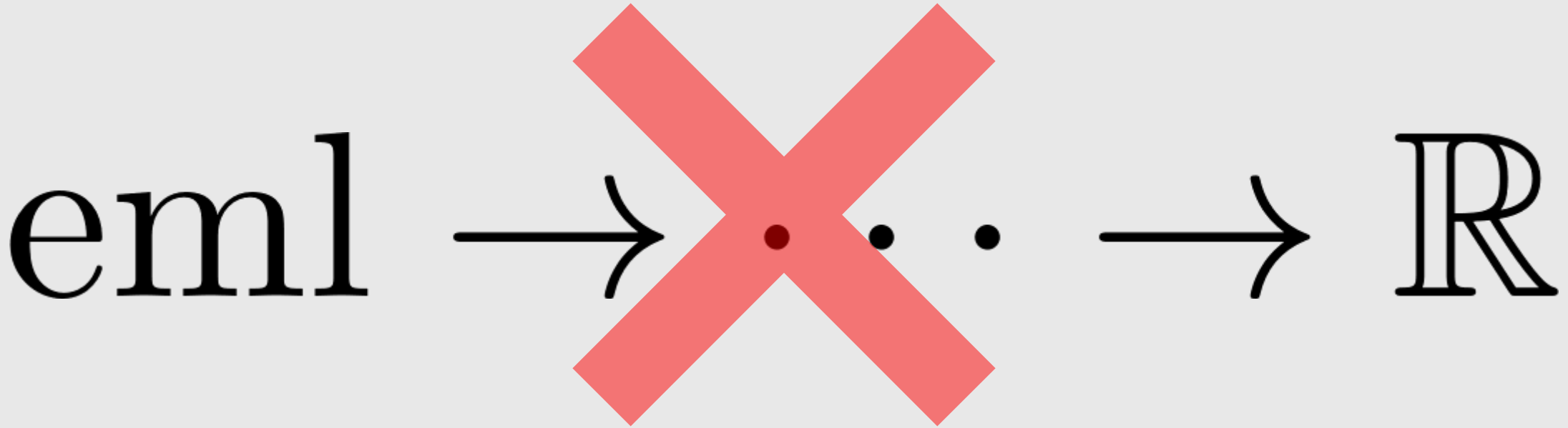
All elementary functions from a single operator

すべての初等関数が1つの演算子によって生成されると書かれています.

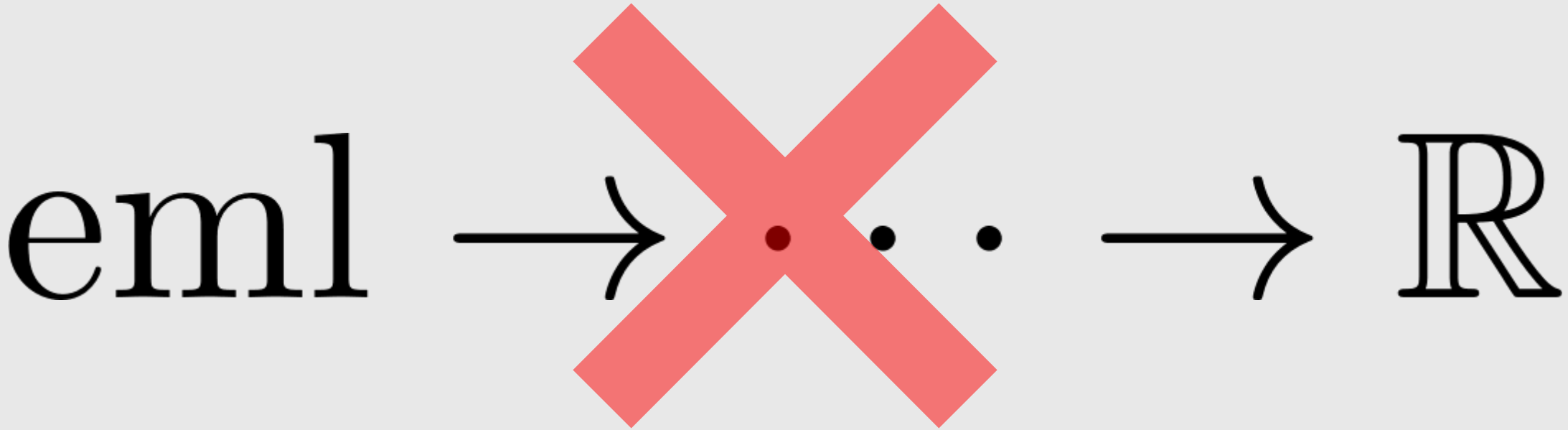
https://arxiv.org/pdf/2603.21852v2

All elementary functions from a single operator

例えば定数関数は，一般的には初等関数に含まれます．



しかし，すべての実数がEMLによって表現できるわけではありません.

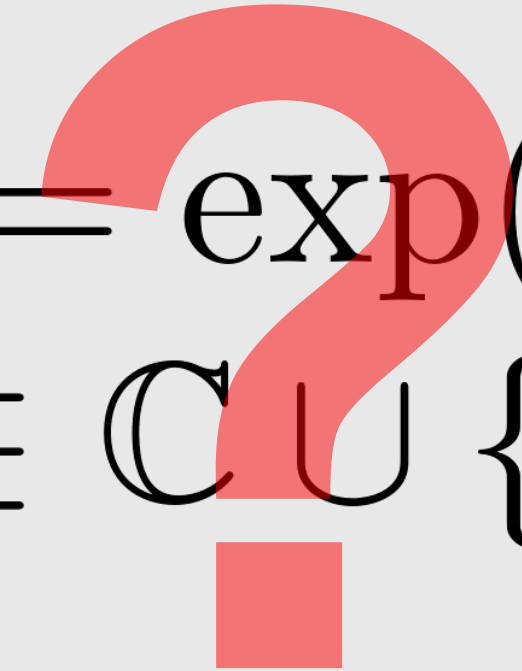


実数は非可算無限個あるわけですから，
それが1つの演算子によって生成されるとは考えにくいです．

$$\mathrm{eml}(x,y) = \exp(x) - \ln(y)$$
$$(x, y \in \mathbb{C} \cup \{\pm\infty\})$$

また，EMLに複素数や無限大を代入するというもどかしさもあります．

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y)$$
$$(x, y \in \mathbb{C} \cup \{\pm\infty\})$$



これについては後で解説しますが，複素数や拡張実数に
EMLを適用することの正当性を考えなくてはなりません．

$$\mathrm{eml} \to \times \cdot \cdot \to \mathbb{R}$$

$$\mathrm{eml}(x,y) = \exp(x) - \ln(y)$$

$$(x, y \in \mathbb{C} \cup \{\pm\infty\})$$

この問題を念頭に置いていただければと思います．

EML論文の著者

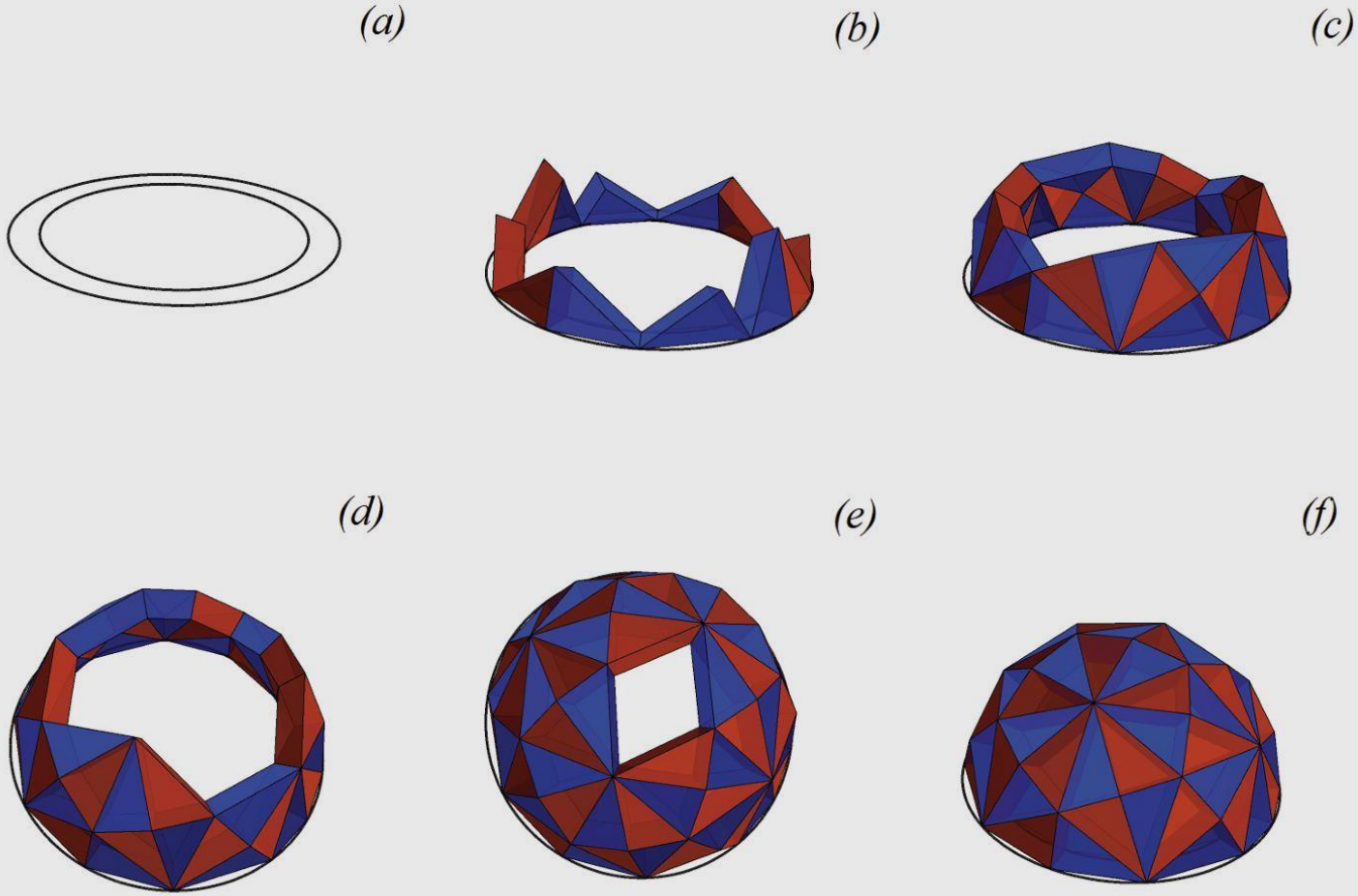
では，この論文の著者について述べておきましょう．

Andrzej Odrzywołek

論文の著者は，ポーランドにある
ヤギェウォ大学の理論物理学研究所に所属しており，

EML論文の著者

https://th.if.uj.edu.pl/~odrzywolek/homepage/publications/PDF/igloos%20final.pdf

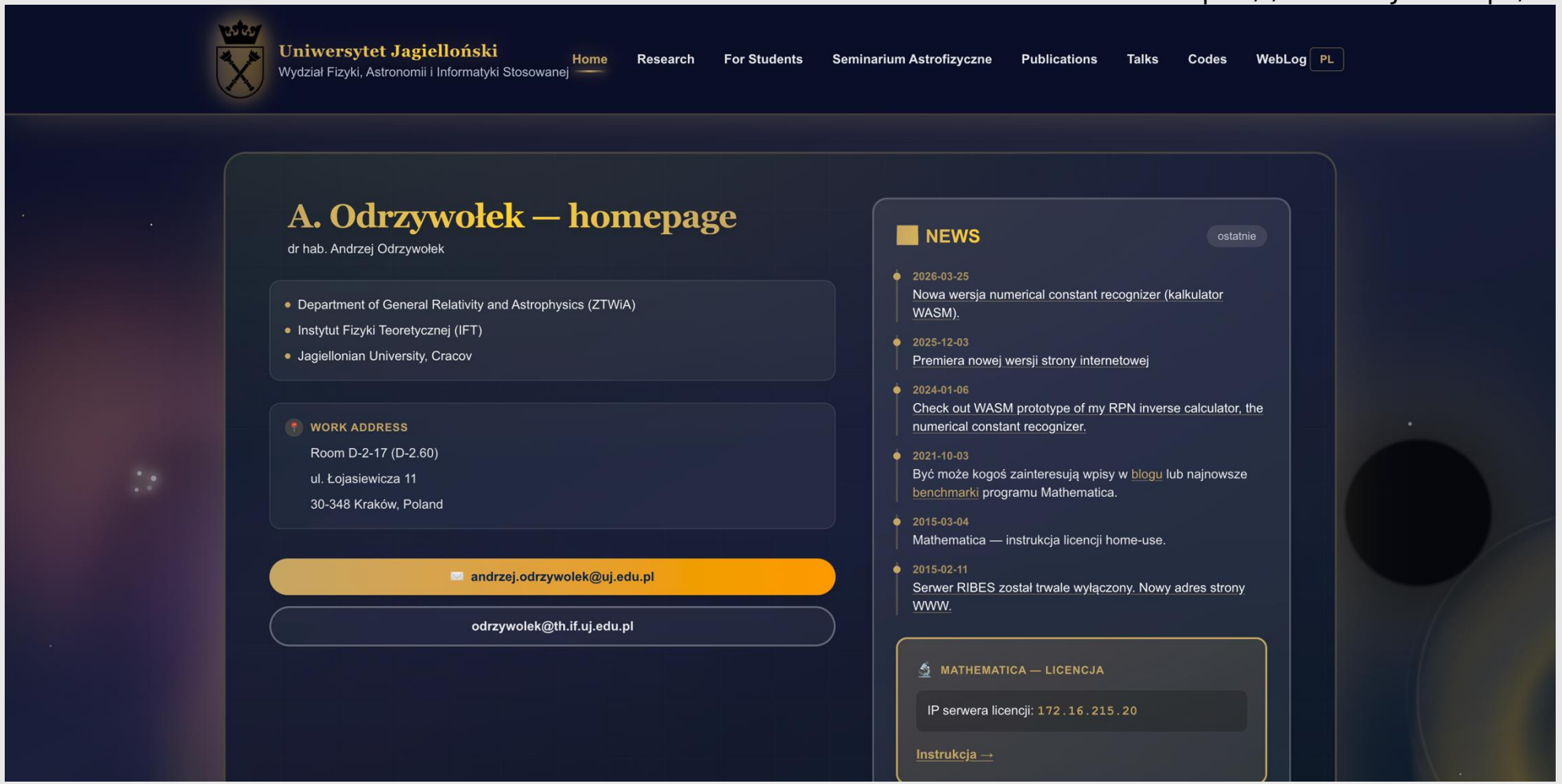


理論物理学者でありながら，

過去には離散幾何学に関する論文を発表しています．

EML論文の著者

https://th.if.uj.edu.pl/~odrzywolek/

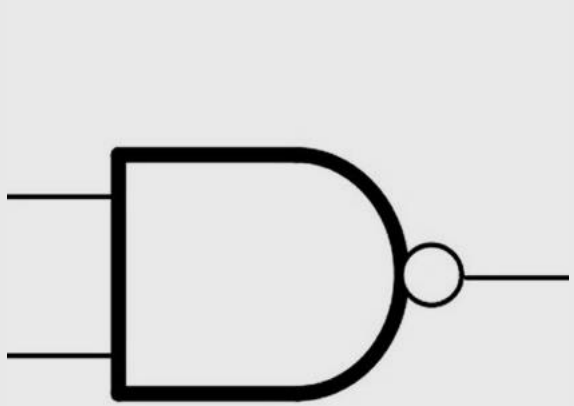


詳しくは，著者本人のWebページをご覧ください．

NANDゲート

次に，NANDゲートについて解説します．

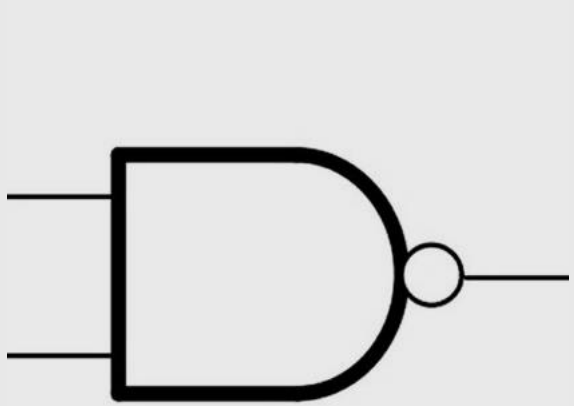
NANDゲート



A	B	NAND(A,B)
0	0	1
0	1	1
1	0	1
1	1	0

NANDゲートとは，両方が真であるときにのみ
偽となるような論理回路のことです．

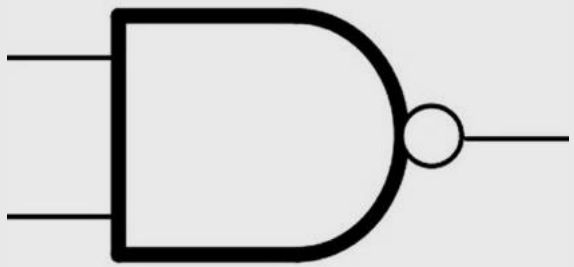
NANDゲート



A	B	NAND(A,B)
0	0	1
0	1	1
1	0	1
1	1	0

急に話が変わったように思われるかもしれませんが，
NANDゲートは，EML演算子と非常によく似た性質があります．

NANDゲート



A	B	NAND(A,B)
0	0	1
0	1	1
1	0	1
1	1	0

実は，任意の論理回路は，
NANDゲートによって表現することができるのです．

$$\mathrm{NOT}(\mathrm{A}) = \mathrm{NAND}(\mathrm{A}, \mathrm{A})$$

例えば，NOTゲートはこのようになり，

$$\mathrm{AND}(\mathrm{A},\mathrm{B}) = \mathrm{NAND}(\mathrm{NAND}(\mathrm{A},\mathrm{B}),\mathrm{NAND}(\mathrm{A},\mathrm{B}))$$

ANDゲートはこのようになり，

$$\mathrm{OR}(\mathrm{A},\mathrm{B}) = \mathrm{NAND}(\mathrm{NAND}(\mathrm{A},\mathrm{A}),\mathrm{NAND}(\mathrm{B},\mathrm{B}))$$

ORゲートはこのようになり，

$$\mathrm{XOR}(\mathrm{A},\mathrm{B}) = \mathrm{NAND}(\mathrm{NAND}(\mathrm{A},\mathrm{NAND}(\mathrm{A},\mathrm{B})),\mathrm{NAND}(\mathrm{B},\mathrm{NAND}(\mathrm{A},\mathrm{B})))$$

XORゲートはこのようになります.

eml

↕

NAND

EML演算子は，論理回路でのNANDゲートに相当するというのが，
今回の論文の主張です．

EML論文の意義

最後に，この論文の意義についてお話します．

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y)$$

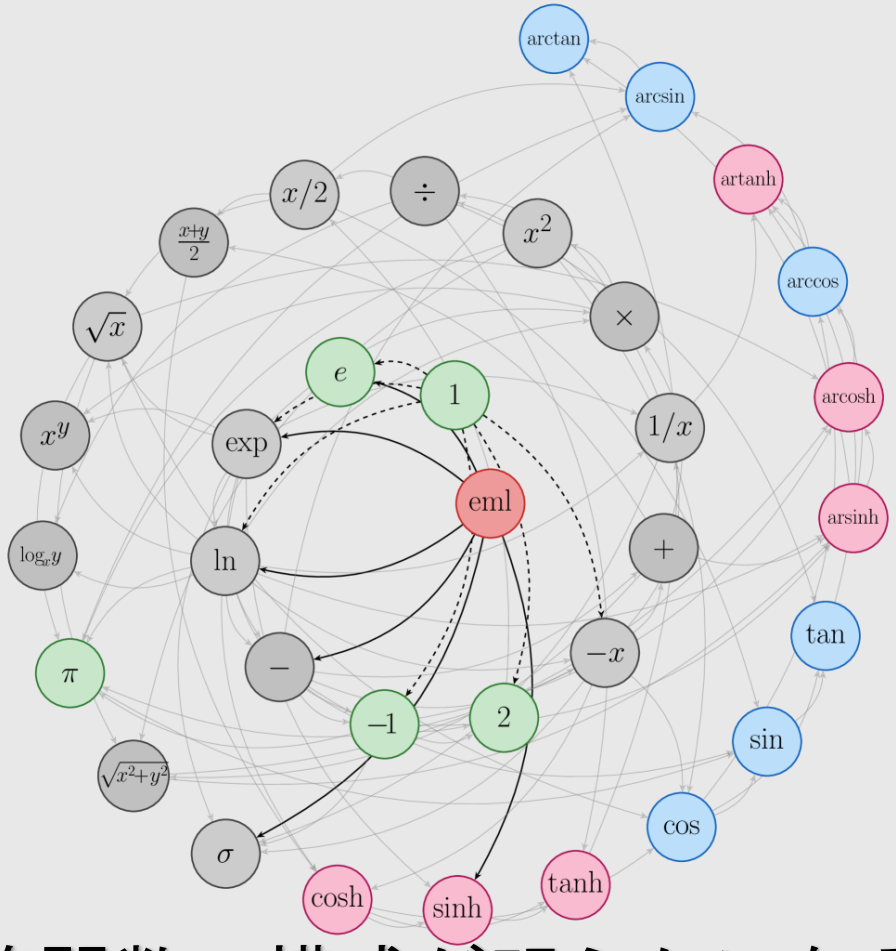
論文によると，EML演算子と1によって，
様々な定数や初等関数が表現できます．

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y)$$

これの何が嬉しいのでしょうか.

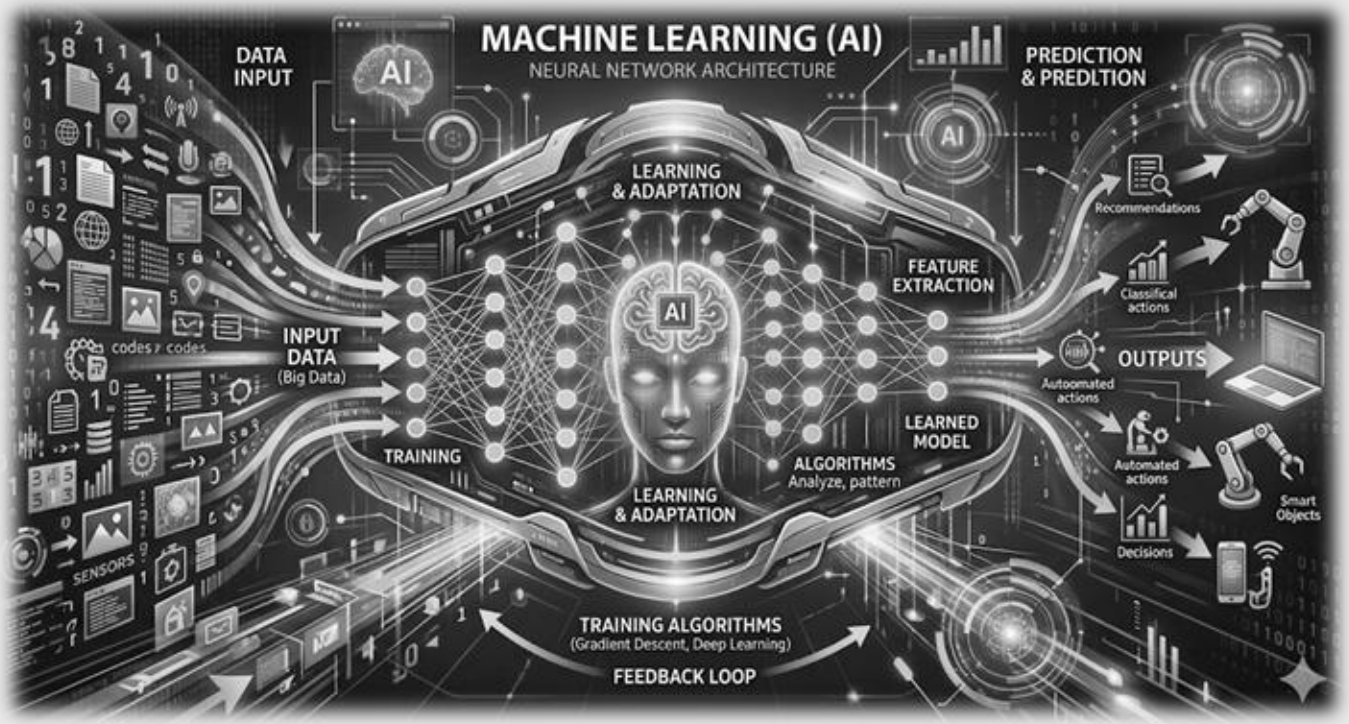
EML論文の意義

https://arxiv.org/pdf/2603.21852v2

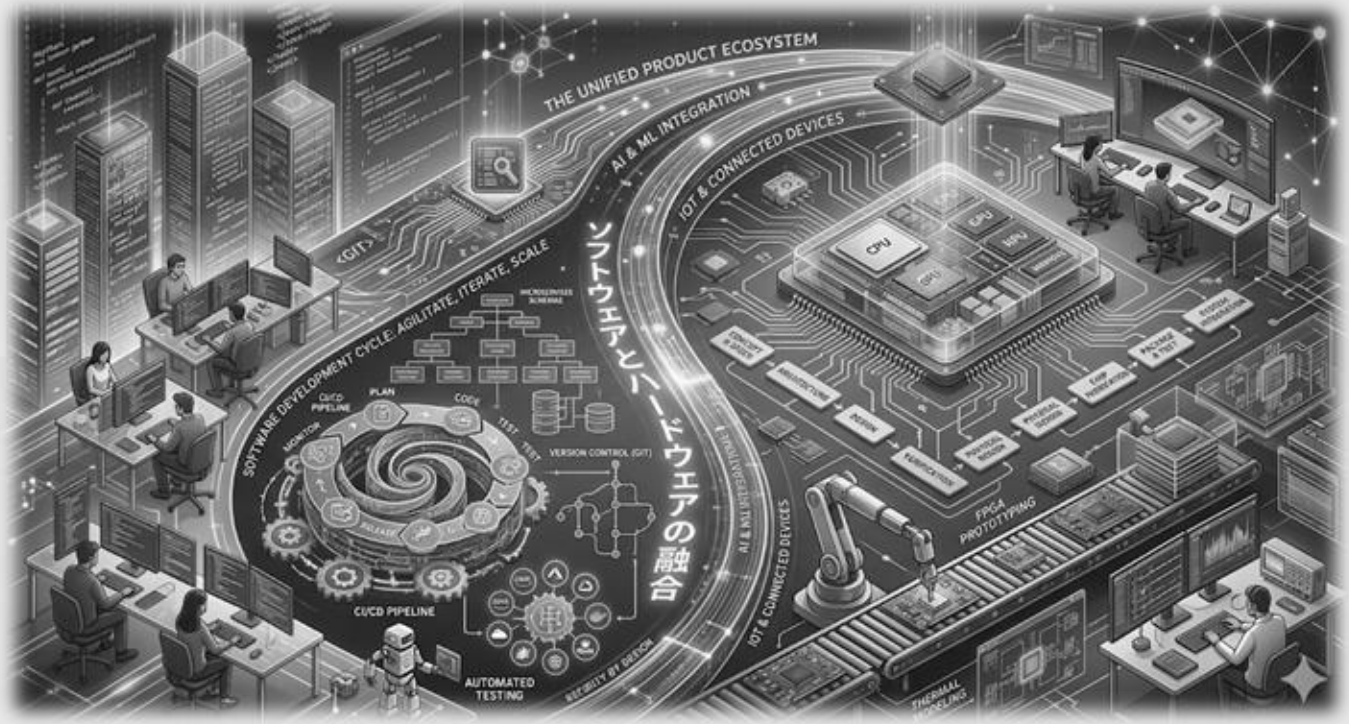


定数や初等関数の構成が明らかになることで，

よりシンプルな処理が可能となります．



例えば，AIの機械学習や，...



ソフトウェアやハードウェアの開発への応用が期待されます.

$$\mathrm{eml} \not\to \cdot \to \mathbb{R}$$

$$\mathrm{eml}(x,y) = \exp(x) - \ln(y)$$

$$(x, y \in \mathbb{C} \cup \{\pm\infty\})$$

とはいえ，先ほども述べた通り，

このEMLにはある種のもどかしさが残っているため，注意が必要です．

3. EML演算子

次に，この論文の主役である，EML演算子について解説します．

3. EML演算子

1. EML演算子
2. 36種類の演算子

3. EML演算子

1.EML演算子
2.36種類のリスト

このチャプターでは，2つの内容をお話します．

EML演算子

まずは，EML演算子について，詳しく解説します．

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y)$$

EML演算子は，このように定義されます．

$$\mathrm{eml}(x, y) \neq \mathrm{eml}(y, x)$$

すぐに分かることとして，EMLは非可換，
つまり交換法則が成り立ちません．

EMLでは，無限大を扱うことがあります．

$$\ln(0) = -\infty$$

$$\exp(-\infty) = 0$$

$\ln(0) = -\infty$とし，$\exp(-\infty) = 0$とします．

$\mathbb{C}$

また，複素数を扱うこともあります．

$$\exp(x+iy) = \exp(x)(\cos(y) + i\sin(y))$$
$$(x, y \in \mathbb{R})$$

複素指数関数は，通常通りこのように定義し，

$$\ln(z) = \log_e(|z|) + i\,\mathrm{Arg}\,z$$
$$(-\pi < \mathrm{Arg}\,z \leq \pi)$$

複素対数関数は，主値を考えることにします．

36種類のリスト

論文では，EMLと1によって，
36種類の定数・関数・演算が生成できることが示されています．

36種類のリスト

具体的な表示については次のチャプターで解説することにして,
ここでは36個全てを紹介していきます.

$$\pi, e, i, -1, 1, 2, x, y$$

まず，数学定数・変数として，この8個があります．

$$\pi, e, i, -1, 1, 2, x, y$$

πは円周率，eはネイピア数，iは虚数単位です．

$$\exp(x) = e^x$$

$$\ln(x) = \log_e(x)$$

また，関数として，指数関数と対数関数，

$$\mathrm{inv}(x) = 1/x$$

$$\mathrm{half}(x) = x/2$$

$$\mathrm{minus}(x) = -x$$

逆数，2ぶんの1倍，マイナス1倍，

$$\sqrt{x} = x^{1/2}$$

$$\mathrm{sqr}(x) = x^2$$

平方根，2乗，

$$\sigma(x) = 1/(1+\exp(-x))$$

$$= \frac{1}{1+e^{-x}}$$

そしてシグモイド関数の8個に加え，

$$\sin(x) = (\exp(ix) - \exp(-ix))/2i$$

$$\cos(x) = (\exp(ix) + \exp(-ix))/2$$

$$\tan(x) = (\sin(x))/(\cos(x))$$

三角関数,...

$$\arcsin(x) = -i\ln(ix + \sqrt{1-x^2})$$

$$\arccos(x) = -i\ln(x + i\sqrt{1-x^2})$$

$$\arctan(x) = (i/2)\ln((i+x)/(i-x))$$

逆三角関数,...

$$\sinh(x) = (e^x - e^{-x})/2 = -i\sin(ix)$$

$$\cosh(x) = (e^x + e^{-x})/2 = \cos(ix)$$

$$\tanh(x) = (\sinh(x))/(\cosh(x)) = -i\tan(ix)$$

双曲線関数,...

$$\operatorname{arsinh}(x) = \ln(x + \sqrt{x^2 + 1})$$

$$\operatorname{arcosh}(x) = \ln(x + \sqrt{x^2 - 1})$$

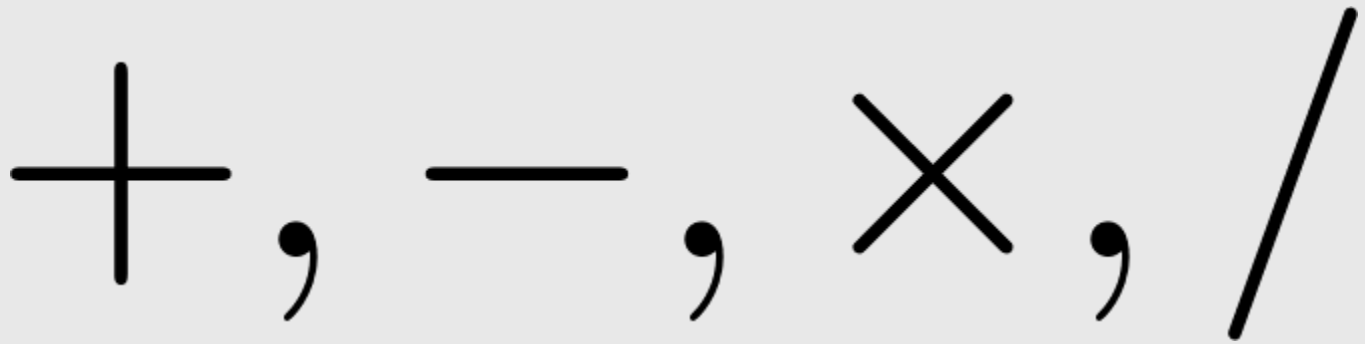
$$\operatorname{artanh}(x) = (1/2)\ln((1 + x)/(1 - x))$$

逆双曲線関数の,...

https://arxiv.org/pdf/2603.21852v2

$$\exp, \ln, \mathrm{inv}, \mathrm{half}, \mathrm{minus}, \sqrt{\ }, \mathrm{sqr}, \sigma,$$
$$\sin, \cos, \tan, \arcsin, \arccos, \arctan,$$
$$\sinh, \cosh, \tanh, \mathrm{arsinh}, \mathrm{arcosh}, \mathrm{artanh}$$

合計20個があります.



さらに，演算として，四則演算，

$$\log(x, y) = \log_x y = (\ln y)/(\ln x)$$

$$\mathrm{pow}(x, y) = x^y = \exp(y \ln x)$$

任意の底の対数や冪乗，

$$\mathrm{avg}(x, y) = (x + y)/2$$

算術平均,

$$\mathrm{hypot}(x, y) = \sqrt{x^2 + y^2}$$

2乗和の平方根の8個があります.

3. EML演算子　36種類のリスト

https://arxiv.org/pdf/2603.21852v2

Type	Elements	Count
Constants	$\pi, e, i, -1, 1, 2, x, y$	8
Functions	$\exp, \ln, \mathrm{inv}, \mathrm{half}, \mathrm{minus}, \sqrt{\ }, \mathrm{sqr}, \sigma,$ $\sin, \cos, \tan, \arcsin, \arccos, \arctan,$ $\sinh, \cosh, \tanh, \mathrm{arsinh}, \mathrm{arcosh}, \mathrm{artanh}$	20
Operations	$+, -, \times, /, \log, \mathrm{pow}, \mathrm{avg}, \mathrm{hypot}$	8
Total		36

これらの計36個の式が，EMLと1によって表現できるとしているのが，
今回の論文なのです．

4. 具体的な計算

最後に，EMLを用いた具体的な計算を見ていきます．

4. 具体的な計算

https://arxiv.org/pdf/2603.21852v2

Some simple examples of tree/circuit representations are shown in Fig. 2. The examples shown are natural logarithm, identity, negation $\mathrm{minus}(x) = -x$, reciprocal $\mathrm{inv}(x) = 1/x$, and multiplication. Ability to compute the identity function using an EML tree of depth 4 allows some input variables to be moved down the tree (see next Subsection). Other elementary functions, e.g. trigonometric ones, have trees too large to be shown in print, cf. Table 4.

論文では，36個のリストすべてに対して，
その具体的な表示が与えられているわけではありません．

4. 具体的な計算

https://arxiv.org/pdf/2603.21852v2

Some simple examples of tree/circuit representations are shown in Fig. 2. The examples shown are natural logarithm, identity, negation $\mathrm{minus}(x) = -x$, reciprocal $\mathrm{inv}(x) = 1/x$, and multiplication. Ability to compute the identity function using an EML tree of depth 4 allows some input variables to be moved down the tree (see next Subsection). Other elementary functions, e.g. trigonometric ones, have trees too large to be shown in print, cf. Table 4.

これは，実際に書き下すと，膨大な量になってしまうからです．

4. 具体的な計算

https://arxiv.org/pdf/2603.21852v2

Table 4: Complexity of various functions in EML tree representation. EML Compiler column gives RPN code length K for expressions generated from EML compiler. The value of K of EML formula can be computed using e.g. Mathematica `LeafCount`. For the identity function x, the compiler returns x directly (leaf count 1); the shortest non-trivial EML expression have leaf count 9. Last column show results of direct exhaustive search for shortest expressions. Numbers in parentheses show length of formulas which do not use the extended reals ($\pm$inf in floating-point). If search timed out, reached lower limit for K is given.

Constant	EML Compiler	Direct search	Function	EML Compiler	Direct search	Operator	EML Compiler	Direct search
1	1	1 (1)	x	1	9	$x-y$	83	11 (11)
0	7	7 (7)	e^x	3	3	$x+y$	27	19 (19)
-1	17	15 (17)	$\ln x$	7	7	$x\times y$	41	17 (17)
2	27	19 (19)	$-x$	57	15	x/y	105	17 (17)
-2	43	27 (27)	$\frac{1}{x}$	65	15	x^y	49	25
$1/2$	91	29 (35)	$x-1$	43	11	$\log_x y$	117	29
$-1/2$	107	31 (37)	$x+1$	27	19	$(x+y)/2$	287	>27
$2/3$	143	39 (39)	$x/2$	131	27	x^2+y^2	175	>27
$-2/3$	159	45 (47)	$2x$	131	19			
$\sqrt{2}$	165	>47	$\sqrt{x}$	139	43 $\geq$? >35			
i	131	>55	x^2	75	17			
e	3	3						
π	193	>53						

論文には，EMLの表示を探索するプログラムによって見つかった表示の長さと，総当りによって見つかった表示の長さが，表でまとめられています．

4. 具体的な計算

https://arxiv.org/pdf/2603.21852v2

Table 4: Complexity of various functions in EML tree representation. EML Compiler column gives RPN code length K for expressions generated from EML compiler. The value of K of EML formula can be computed using e.g. Mathematica `LeafCount`. For the identity function x, the compiler returns x directly (leaf count 1); the shortest non-trivial EML expression have leaf count 9. Last column show results of direct exhaustive search for shortest expressions. Numbers in parentheses show length of formulas which do not use the extended reals ($\pm$inf in floating-point). If search timed out, reached lower limit for K is given.

Constant	EML Compiler	Direct search	Function	EML Compiler	Direct search	Operator	EML Compiler	Direct search
1	1	1 (1)	x	1	9	$x-y$	83	11 (11)
0	7	7 (7)	e^x	3	3	$x+y$	27	19 (19)
-1	17	15 (17)	$\ln x$	7	7	$x\times y$	41	17 (17)
2	27	19 (19)	$-x$	57	15	x/y	105	17 (17)
-2	43	27 (27)	$\frac{1}{x}$	65	15	x^y	49	25
$1/2$	91	29 (35)	$x-1$	43	11	$\log_x y$	117	29
$-1/2$	107	31 (37)	$x+1$	27	19	$(x+y)/2$	287	>27
$2/3$	143	39 (39)	$x/2$	131	27	x^2+y^2	175	>27
$-2/3$	159	45 (47)	$2x$	131	19			
$\sqrt{2}$	165	>47	$\sqrt{x}$	139	43 $\geq$? >35			
i	131	>55	x^2	75	17			
e	3	3						
π	193	>53						

短いものについては，実際に手を動かして計算できるので，

4. 具体的な計算

https://arxiv.org/pdf/2603.21852v2

Type	Elements	Count
Constants	$\pi, e, i, -1, 1, 2, x, y$	8
Functions	$\exp, \ln, \mathrm{inv}, \mathrm{half}, \mathrm{minus}, \sqrt{\ }, \mathrm{sqr}, \sigma,$ $\sin, \cos, \tan, \arcsin, \arccos, \arctan,$ $\sinh, \cosh, \tanh, \mathrm{arsinh}, \mathrm{arcosh}, \mathrm{artanh}$	20
Operations	$+, -, \times, /, \log, \mathrm{pow}, \mathrm{avg}, \mathrm{hypot}$	8
Total		36

このチャプターでは，36個のリストにあるものを，
1つずつ生成してみます．

$$e = \mathrm{eml}(1, 1)$$

ネイピア数eは，このように生成できます．

e

$$\exp(x) = \mathrm{eml}(x, 1)$$

指数関数は，このように生成できます．

e exp

$$\ln(x) = \mathrm{eml}(1, \mathrm{eml}(\mathrm{eml}(1, x), 1))$$

対数関数は，これまでより少々長いですが，このように生成できます．

$e \exp \ln$

$$x - y = \mathrm{eml}(\ln(x), \exp(y))$$

これを用いると，引き算を生成することができます．

$$e\ \exp\ \ln\ -$$

$$1 - x = \mathrm{eml}(\ln(1), \exp(x))$$

よって，$1 - x$が生成できるので，

$$e\ \exp\ \ln\ -$$

$$\mathrm{minus}(x) = -x = (1 - x) - 1$$

これを用いると，$-x$を生成することができます．

$e\ \exp\ \ln\ -\ \mathrm{minus}$

$$x + y = x - \mathrm{minus}(y)$$

また，yを$-y$に置き換えることで，足し算を生成することができます．

$$e \exp \ln - \text{minus} +$$

$$x \times y = \exp(\ln x + \ln y)$$

掛け算は，このように生成することができ，

$$e \exp \ln - \text{minus} + \times$$

$$x/y = \exp(\ln x - \ln y)$$

割り算は，このように生成することができるので，

e exp ln　　minus

四則演算がすべて生成できることが分かりました.

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /$$

$$\mathrm{inv}(x) = 1/x$$

$$\mathrm{half}(x) = x/2$$

もちろん，逆数や1/2倍も生成できます．

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}$$

$$-1 = \mathrm{minus}(1)$$

$$2 = 1 - (-1)$$

$$i = \exp(\mathrm{half}(\ln(-1)))$$

$$\pi = -i \times \ln(-1)$$

ネイピア数以外の定数についても，このように生成することができ，

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$

$$x == \exp(\ln(x))$$

このような恒等関数も生成できるので，

$$\exp \ln - \text{minus} + \times / \text{inv half}$$

$$\pi \quad e \quad i \quad -1 \quad 1 \quad 2 \quad x \quad y$$

8個の定数と変数すべてが生成できることが分かりました.

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1$$

$$\mathrm{sqr}(x) = x^2 = \exp(2 \times \ln(x))$$

$$\sqrt{x} = \exp(\mathrm{half}(1) \times \ln(x))$$

2乗，平方根については，このように生成でき，

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }$$

$$\sigma(x) = \frac{1}{1+e^{-x}} = \mathrm{inv}(1+\exp(-x))$$

シグモイド関数は，このように生成できます．

$$e\,\exp\,\ln\,-\,\mathrm{minus}\,+\,\times\,/\,\mathrm{inv}\,\mathrm{half}\,-1\,2\,i\,\pi$$
$$x\,y\,1\,\mathrm{sqr}\,\sqrt{\ }\,\sigma$$

$$\log(x,y) = \log_x y = (\ln y)/(\ln x)$$

$$\mathrm{pow}(x,y) = x^y = \exp(y \times \ln(x))$$

さらに，一般の対数や冪乗についても，このように生成でき，

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }\ \sigma\ \log\ \mathrm{pow}$$

$$\mathrm{avg}(x, y) = (x + y)/2$$
$$\mathrm{hypot}(x, y) = \sqrt{x^2 + y^2} = \sqrt{\mathrm{pow}(x, 2) + \mathrm{pow}(y, 2)}$$

算術平均や，平方和の平方根も生成できるので，

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }\ \sigma\ \log\ \mathrm{pow}\ \mathrm{avg}\ \mathrm{hypot}$$

$$\sin, \cos, \tan, \arcsin, \arccos, \arctan,$$
$$\sinh, \cosh, \tanh, \mathrm{arsinh}, \mathrm{arcosh}, \mathrm{artanh}$$

残すは三角関数と双曲線関数になりました.

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }\ \sigma\ \log\ \mathrm{pow}\ \mathrm{avg}\ \mathrm{hypot}$$

$$\mathbb{C}$$

ここからは，複素数を使っていきます．

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }\ \sigma\ \log\ \mathrm{pow}\ \mathrm{avg}\ \mathrm{hypot}$$

$$\sin(x) = (\exp(i \times x) - \exp(-i \times x))/(2 \times i)$$
$$\cos(x) = (\exp(i \times x) + \exp(-i \times x))/2$$
$$\tan(x) = (\sin(x))/(\cos(x))$$

三角関数は，このように生成でき，

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }\ \sigma\ \log\ \mathrm{pow}\ \mathrm{avg}\ \mathrm{hypot}$$

$$\arcsin(x) = \mathrm{minus}(i) \times \ln(i \times x + \exp(\mathrm{half}(\ln(1 - \mathrm{pow}(x, 2)))))$$
$$\arccos(x) = \pi/2 - \arcsin(x)$$
$$\arctan(x) = (i/2) \times \ln((i + x)/(i - x))$$

逆三角関数は，このように生成でき，

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }\ \sigma\ \log\ \mathrm{pow}\ \mathrm{avg}\ \mathrm{hypot}$$

$$\sinh(x) = \mathrm{half}(\exp(x) - \exp(-x))$$

$$\cosh(x) = \mathrm{half}(\exp(x) + \exp(-x))$$

$$\tanh(x) = (\sinh(x))/(\cosh(x))$$

双曲線関数は，このように生成でき，

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }\ \sigma\ \log\ \mathrm{pow}\ \mathrm{avg}\ \mathrm{hypot}$$

$$\mathrm{arsinh}(x) = \ln(x + \exp(\mathrm{half}(\ln(\mathrm{pow}(x, 2) + 1))))$$
$$\mathrm{arcosh}(x) = \ln(x + \exp(\mathrm{half}(\ln(\mathrm{pow}(x, 2) - 1))))$$
$$\mathrm{artanh}(x) = \mathrm{half}(\ln((1 + x)/(1 - x)))$$

逆双曲線関数は，このように生成できるので，

4. 具体的な計算

$$e\ \exp\ \ln\ -\ \mathrm{minus}\ +\ \times\ /\ \mathrm{inv}\ \mathrm{half}\ -1\ 2\ i\ \pi$$
$$x\ y\ 1\ \mathrm{sqr}\ \sqrt{\ }\ \sigma\ \log\ \mathrm{pow}\ \mathrm{avg}\ \mathrm{hypot}$$

sin	arcsin	sinh	arsinh
cos	arccos	cosh	arcosh
tan	arctan	tanh	artanh

36個のリストすべてがEMLによって生成できることが分かりました.

5. まとめ

いかがでしたか?

https://arxiv.org/pdf/2603.21852v2

All elementary functions from a single operator

Andrzej Odrzywołek

Institute of Theoretical Physics, Jagiellonian University, 30-348 Krakow, Poland

E-mail: andrzej.odrzywolek@uj.edu.pl

April 7, 2026

今回は，話題になったEML論文について解説しました．

5. まとめ

https://arxiv.org/pdf/2603.21852v2

4.1 EML compiler

The output of the `VerifyBaseSet` procedure provides the data (see Fig. 1) required to reconstruct any primitive or composite elementary expression in terms of EML Sheffer, (4a). I provide a prototype EML compiler, coded in Python, that converts formulas into pure EML form. An EML expression can be evaluated symbolically in Mathematica, or numerically in any IEEE754-compliant language. Pure EML form could also be executed on hardware (or an emulated machine) that has only a single instruction: the EML itself. In particular, the EML code can be executed on a single instruction stack machine, closely resembling a single-button RPN calculator. Pure-EML form could possibly be implemented efficiently in FPGA or analog circuits.

著者は，EMLによる表示を与える
Pythonによるコンパイラを制作しており，

5. まとめ

https://arxiv.org/pdf/2603.21852v2

Since standard activation functions are themselves elementary, any conventional neural network is a special case of an EML tree architecture. Current networks can learn symbolic algebra [49] and digit-level arithmetic [50], but their internal mechanisms remain opaque [11], and efficient exact evaluation of elementary functions as continuous real-valued operations is still beyond their reach. EML representations go further: as demonstrated in Subsect. 4.3, trained weights can snap to exact binary values, recovering closed-form elementary subexpressions alongside approximations. When this succeeds, the discovered circuits are legible as elementary function expressions — a form of interpretability unavailable to conventional architectures.

EMLの，深層学習などへの応用が期待されています．

$$\mathrm{edl}(x,y) = (\exp(x))/(\ln(y))$$

$$-\mathrm{eml}(y,x) = \ln(x) - \exp(y)$$

また，EML以外の演算子として，
EDLや，引数を入れ替えたEMLなども取り上げていますが，

5. まとめ

https://arxiv.org/pdf/2603.21852v2

One might complain that the EML representation of elementary functions requires complex arithmetic for real math, at least internally. Just as quantum computing uses complex amplitudes to compute real probabilities, EML uses complex intermediates to compute real elementary functions. This seems inevitable. We must somehow compute the imaginary unit i, π, and all trigonometric/hyperbolic functions *via* Euler's formula, (2). For that, we use $\ln x$ for $x < 0$. A continuous Sheffer working purely in the real domain seems impossible. My search for alternatives, e.g., using pairs of trigonometric/hyperbolic functions and their inverses instead of $\exp / \ln$, found nothing. Quite surprisingly, the requirement to use complex numbers internally causes only minor problems in practice of using (3) in Computer Algebra Systems or numerical simulations.

複素数の演算を避けることは難しいと言及しており，

5. まとめ

https://arxiv.org/html/2603.21852v2

The operator EML, (3), provides a single sufficient primitive from which real elementary functions can be constructed and evaluated. Consequently, a wide class of computations built from such functions can also be cast in EML form. It is not unique; several close variants are likely to exhibit similar properties, including EDL, (4b), and the swapped-argument form $-\mathrm{eml}(y,x)$, (4c). More operators of this kind exist. Perhaps an entire continuous family of them awaits discovery, with properties more convenient than (3). For example, the requirement for one of the constants: $-\infty, 1, e, \ldots$ to be always present among terminal symbols makes its use less elegant and more complicated (cf. Subsect. 4.3) in comparison with, e.g., standard neural nets or the NAND gate. The latter is able to generate[3] 0s and 1s out of 'anything'. The EML operator does not have this useful property. Whether an EML-type binary Sheffer working without pairing with a distinguished constant exists is an open question. Proving such impossibility for any given candidate is non-trivial: one might expect $f(x,x)$ being constant to suffice, but consider $B(x,y)=x-\frac{y}{2}$, for which $B(x,x)=\frac{x}{2}$ yet $B(B(x,x),x)=0$. Such traps illustrate why systematic search is essential in this work. A *ternary* operator, $T(x,y,z)=e^x \,/\, \ln x \times \ln z \,/\, e^y$, for which $T(x,x,x)=1$ is next candidate for further analysis [34].

このような演算子が他にないか，変数の数を変えるとどうなるか，
定数を必要としないものはないかといった，様々な課題が残っています．

5. まとめ

今後の展開にも，注目していきたいです．

EML演算子とは何か

MathAbyssでは，数学に関する記事を公開しているWebサイト
「MathAbyss」を運営しております．

この動画に対する高評価，YouTubeのチャンネル登録を
よろしくお願いします!
メンバーシップでは，動画の先行視聴や限定動画等の
特典をご用意しております!

各種SNS等のフォローもしていただけると励みになります！

最後までご視聴いただき，ありがとうございました！！！

MathAbyss

MathAbyss

ご視聴ありがとうございました！
チャンネル登録よろしくお願いします！