根拠を示しつつ、
自ら規則を発見してゆく
プログラムの構想

池上 蒔典

2025 年 01 月 10 日

知的なエージェントでは、、、

- データから規則性を発見できる。説明もしてくれる。
- 新しい状況でも、論理的・合目的的に考えて、
  解法を与えようとする。
- 対話をしていたらプログラムが
  合成されてゆく。変化してゆく。

発見と論理性の観点から、、、。

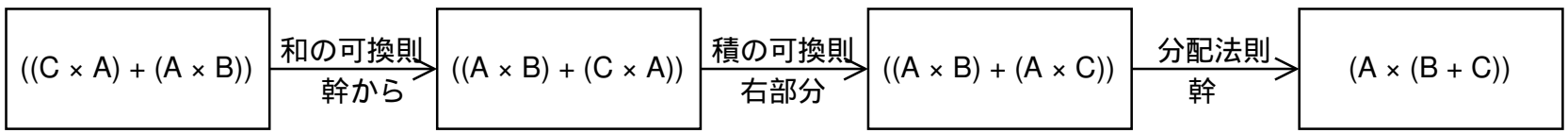
- 深層学習
  理由が解りづらい。
- 手続き型言語
  組み込まれた推論能力が皆無。
- Prolog
  解にたどりつかない。

文字式の変形方法を証明つきで探す

```
$ ros -- call_proof_factorization.lisp
...中略...
開始式を入力して下さい[1] >>
  (d * c * a + c * d * b)
(D * C * A + C * D * B)
目標式を入力して下さい[1] >>
  ((a + b) * c * d)
((A + B) * C * D)
開始式の詳しい中置記法表現には、(((D * C) * A) + ((C * D) * B)) が在る。
目標式の詳しい中置記法表現には、(((A + B) * C) * D) が在る。
  上式 の位置(A D A D)部 に A*B=B*A を適用すると
  (((C * D) * A) + ((C * D) * B)) となる。
  上式 の位置root部 に {C*A}+{C*B}=C*{A+B} を適用すると
  ((C * D) * (A + B)) となる。
  上式 の位置root部 に A*B=B*A を適用すると
  ((A + B) * (C * D)) となる。
  上式 の位置root部 に A*{B*C}={A*B}*C を適用すると
  (((A + B) * C) * D) となる。
かの如く、(D * C * A + C * D * B) から ((A + B) * C * D) への変換は可能。
開始式を入力して下さい[2] >>
```

文字式の変形方法を
証明つきで探すプログラム

公理や正しい定理だけを用いて、
最初の数式を変形させてゆき、
目標とする式となったとき、
最初から目標に至る迄の変形の経路を証明と言う。



$$C \times A + A \times B = A \times (B + C)$$

数式の構造

多項演算は二項演算に変換

$$a + b + c + d + e$$
$$\rightarrow ((((a + b) + c) + d) + e)$$

式は再帰的な木構造

$$\text{式} := (\text{演算子 式 式})$$

内では演算子を前に置く前置記法

$$((((a + b) + c) + d) + e)$$
$$\rightarrow (+\ (+\ (+\ (+\ a\ b)\ c)\ d)\ e)$$

認める公理

$a + b = b + a$	和の 可換則
$a \times b = b \times a$	積の 可換則
$a + (b + c) = (a + b) + c$	和の 結合則
$a * (b \times c) = (a \times b) \times c$	積の 結合則
$a * (b + c) = a * b + a * c$	分配法則 [1]

```
(defparameter *first-axioms*
  (list
    ;; (name                 unknowns form-before          form-after          . proof)
    '(a+b=b+a                (a b)    (+ a b)              (+ b a)               . ())
    '(a*b=b*a                (a b)    (* a b)              (* b a)               . ())
    '(a+{b+c}={a+b}+c        (a b c)  (+ a (+ b c))        (+ (+ a b) c)         . ())
    '(a*{b*c}={a*b}*c        (a b c)  (* a (* b c))        (* (* a b) c)         . ())
    '(c*{a+b}={c*a}+{c*b}    (c a b)  (* c (+ a b))        (+ (* c a) (* c b)) . ())
    '({c*a}+{c*b}=c*{a+b}    (c a b)  (+ (* c a) (* c b)) (* c (+ a b))         . ())
  ))
```

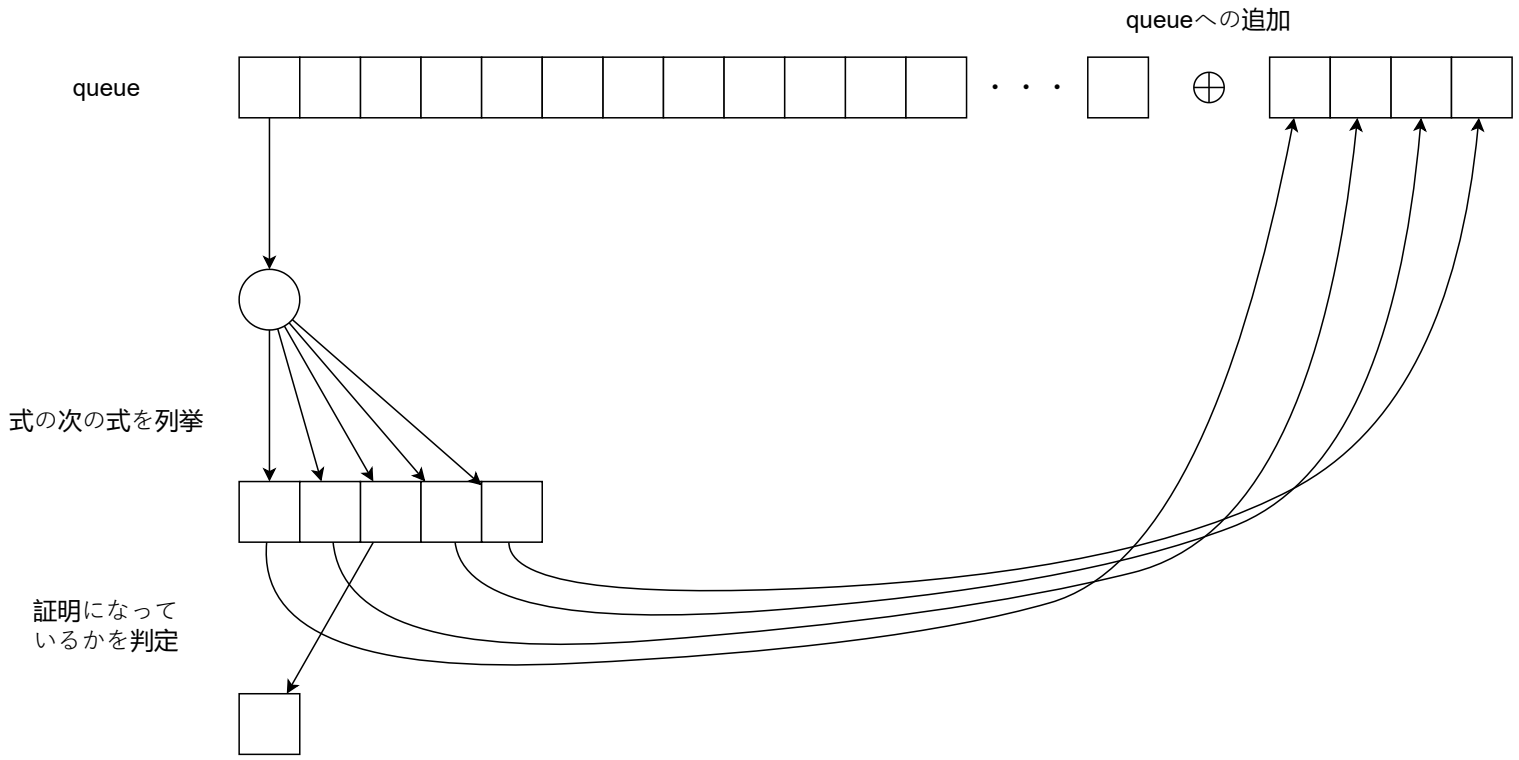
[1] 自然数の実装は今後の ToDo

unification

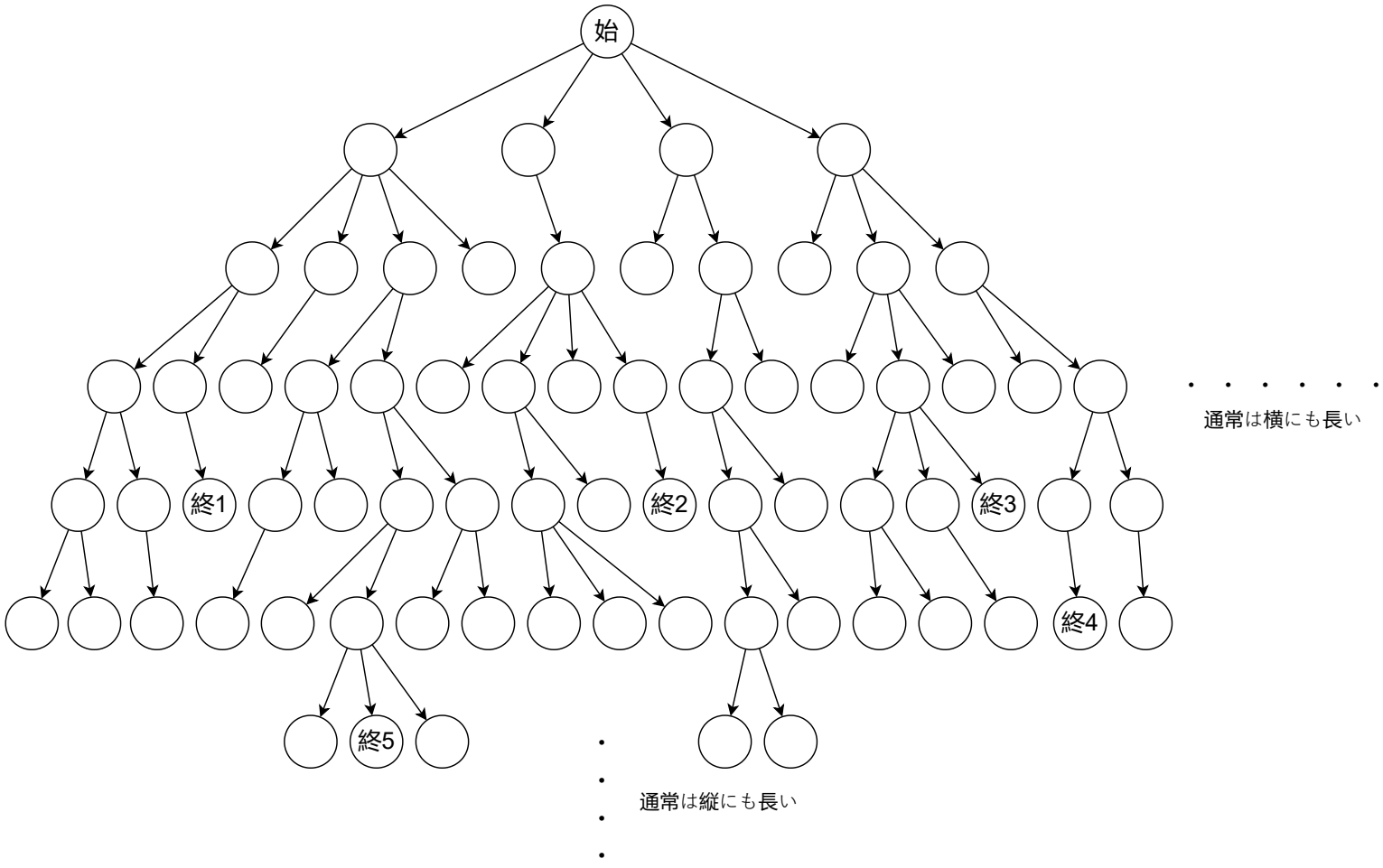
2 つの未決定な式が、同じ式と成り得るか確かめながら、
未知な部分を決定しようとする演算。

```
;; (unification (未知な文字) (未決定な式1) (未決定な式2) (決定されている束縛))
> (unification '(a b) '(+ a b) '(+ 3 4) '())
((B . 4) (A . 3))    ;; b = 3 かつ a = 3 のとき成立。
(A B)
> (unification '(a b) '(+ a b) '(+ b a) '())
((B . A) (A . B))    ;; a = b かつ b = a のとき成立する。
(A B)
> (unification '(a b) '(+ a b) '(+ b a c) '())
+FAILURE+            ;; 要素の数が異なる為 失敗。
> (unification '(a b) '(+ 1 2) '(f 1 2) '())
+FAILURE+            ;; f と + は異なる文字で どちらもも既知なため 失敗。
> (unification '(a) '(+ 1 2) '(a 1 2) '())
((A . +))            ;; 一回表れる右側のaは未知だったが、
                     ;; 左側では + として表われていた為、a = + で成立。
> (unification '(a) '(+ 1 2) '(a 1 a) '())
+FAILURE+            ;; (a . +) と (a . 2) に仮定したが、
                     ;; + も 2 も既知かつ異なる文字であった為、失敗。
```

探索手続きの呼び出し 一回



変形の連鎖と 探索の空間



文字式の変形方法を証明つきで探す

```
$ ros -- call_proof_factorization.lisp
...中略...
開始式を入力して下さい[1] >>
  (d * c * a + c * d * b)
(D * C * A + C * D * B)
目標式を入力して下さい[1] >>
  ((a + b) * c * d)
((A + B) * C * D)
開始式の詳しい中置記法表現には、(((D * C) * A) + ((C * D) * B)) が在る。
目標式の詳しい中置記法表現には、(((A + B) * C) * D) が在る。
  上式 の位置(A D A D)部 に A*B=B*A を適用すると
  (((C * D) * A) + ((C * D) * B)) となる。
  上式 の位置root部 に {C*A}+{C*B}=C*{A+B} を適用すると
  ((C * D) * (A + B)) となる。
  上式 の位置root部 に A*B=B*A を適用すると
  ((A + B) * (C * D)) となる。
  上式 の位置root部 に A*{B*C}={A*B}*C を適用すると
  (((A + B) * C) * D) となる。
かの如く、(D * C * A + C * D * B) から ((A + B) * C * D) への変換は可能。
開始式を入力して下さい[2] >>
```

宇宙の認識について

- 物理現象だけの世界はつまらない。
  エントロピーが上昇するだけで、
  何の有用な情報 (発見) も見いださない。
- 根源の存在。
  天御中主大御神、阿弥陀如来、創造主、太極、、、。
- 物質宇宙が発生する時には、
  物質を生じようという意思が生じていた。
- 泥から何が見だされたのか。泥に何を見い出すのか。
- 言葉により意思を一意に共有でき、
- 文字により強く保存され、再発動される。

泥という直感が無ければ、その泥を泥だと知覚できない。
泥という言葉が無ければ、泥を主体とした考察ができない。

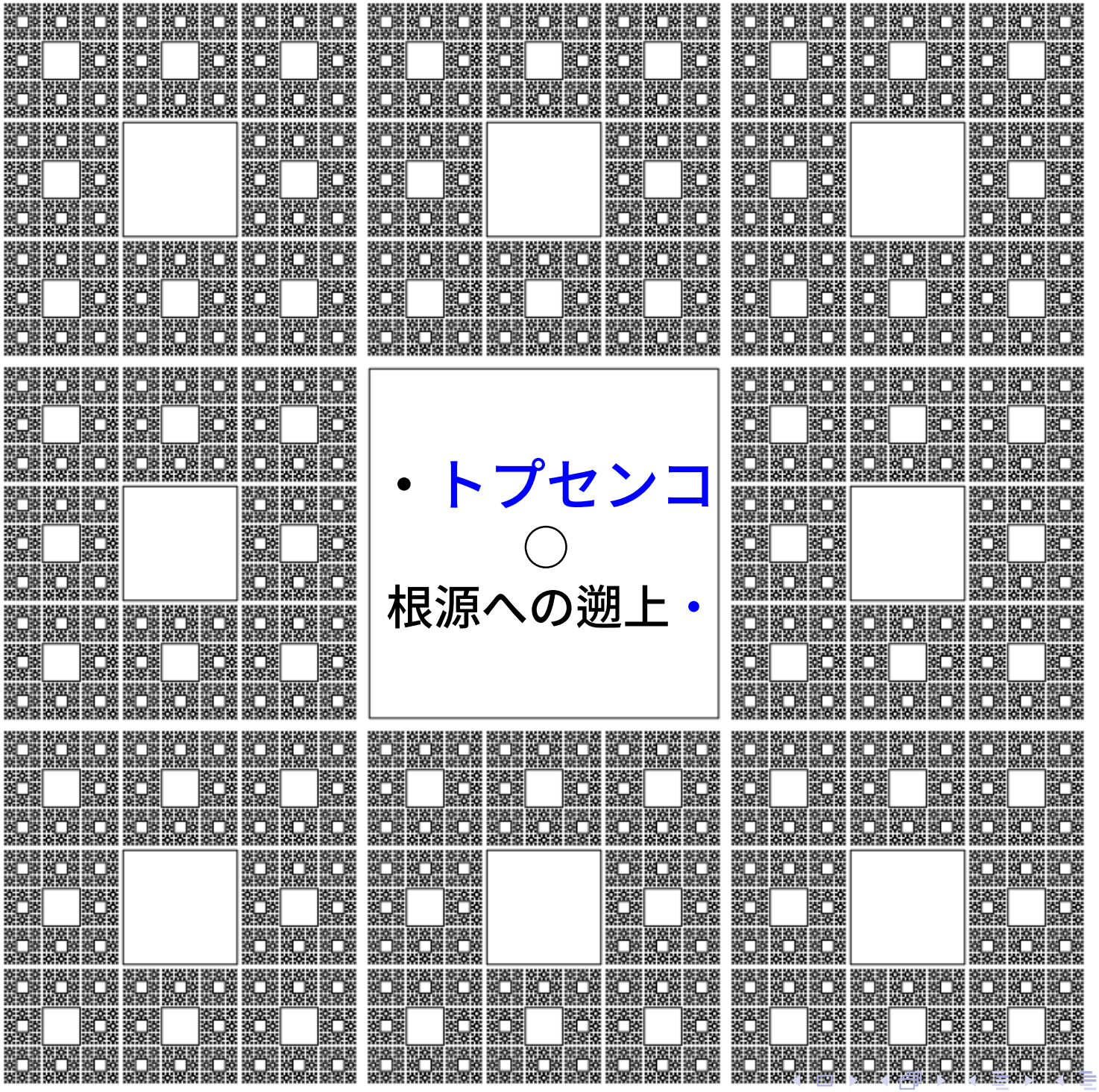
宇宙の認識について

識の相を生ずる根源的原因として文字の形状。かただま。

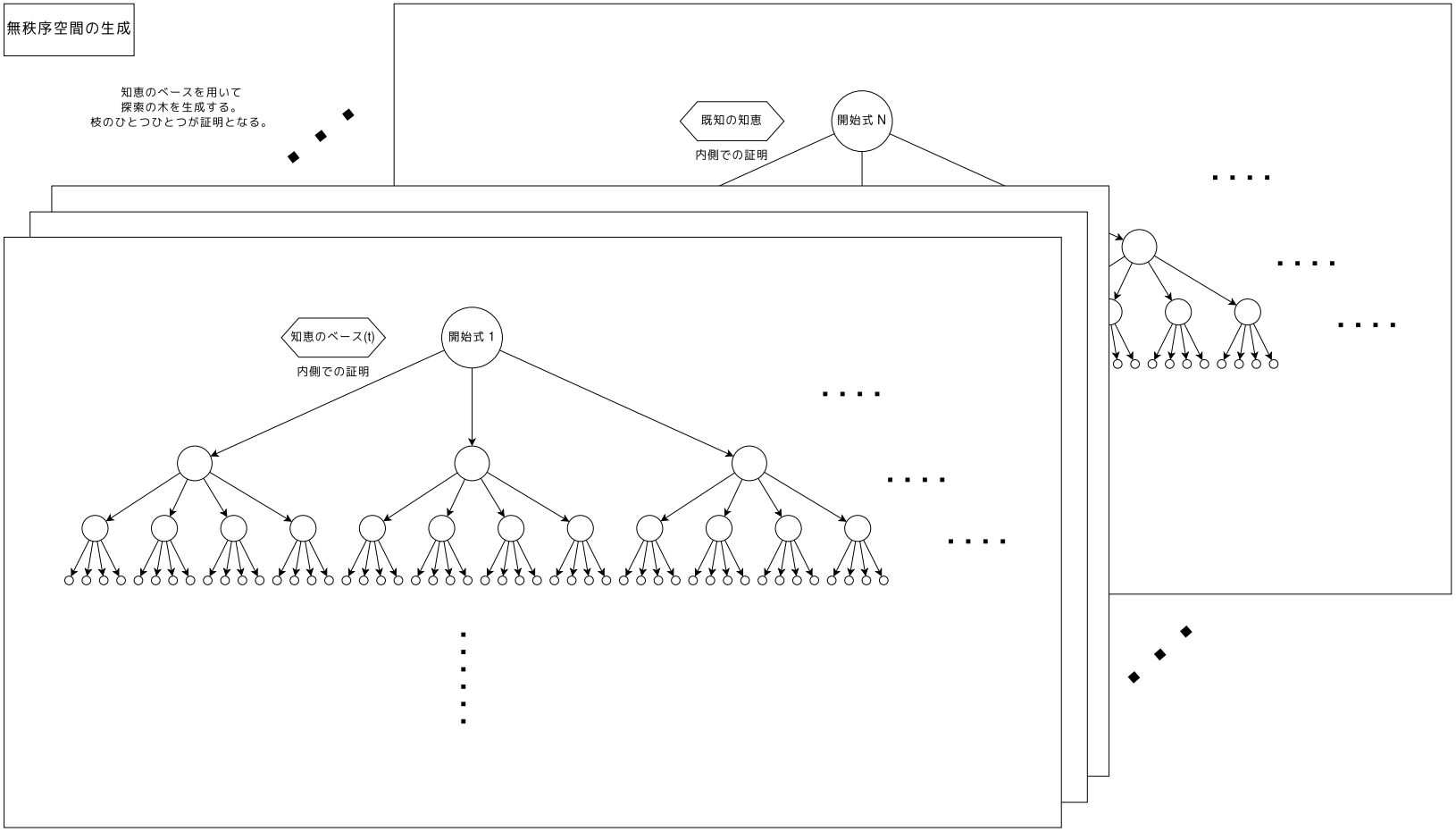
- 固有のスペクトルを生ずる原因を辿ると、固有の形状に (も) なるという立場。
- スリットの形を変えると、スクリーンに映る像 (干渉縞) も変わる、空間内の波形も変わる。
- ○と言えば?、△と言えば? と連想ゲームを続る如く。直線的だけでなく、立体的、胞体的にも広がる。時空にも、ビックバンクランチの前後にも広がる。そして、形状に収束する。

自然言語、数学、作法、、、

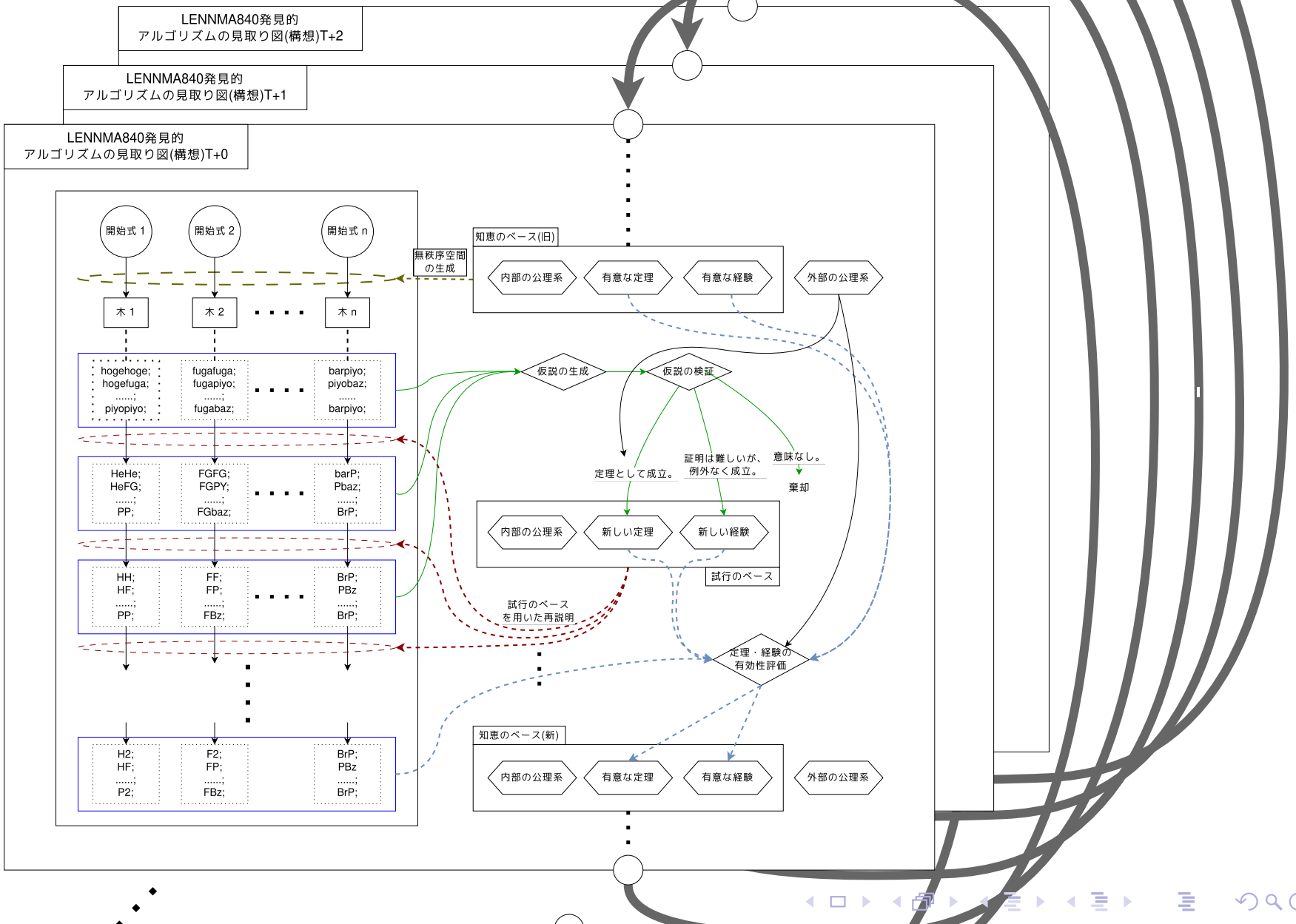
- 抽象化を手に入れた。
  乱雑な星の配列と動きに 秩序が見い出された。
- 機能や構造が 抽象化の適用で構想 開発された。
- 人と人とが対話できること。世代を越えた伝達。


・トプセンコ
○
根源への遡上・

Lennma 定理発見アルゴリズム (構想)



Lennma 定理発見アルゴリズム(構想)



今後の展望

- 定理発見アルゴリズムの実装
  - ◎定理証明 ◎類似した証明の検索
    ◎弱い証明の集合からの一段 強い命題の作成
    ◎ (内部の) 証明への新しい規則に依る書き換え
    ◎エントロピー縮小の為の有意な法則の抽出
- 具体的な問題
  - 論理パズル。
    規則の発見による、深い問題への対処の観察。
  - 中学 高校生 程度の代数の問題。
- ゲーム内のエージェントへの組み込み。
- 外の公理系と内の公理系との一体化。

⋮

根拠を示しつつ、自ら規則を発見し、
自ら発展してゆくソフトウェア。

参考文献、URL

- 猪股 俊光, 益崎 真治、Scheme による記号処理入門、森北出版 (1994).
- 後藤 滋樹, 岩波講座 ソフトウェア科学 8 記号処理プログラミング、 岩波書店 (1988) .
- 佐野 千遥、人工知能と人工生命、 日刊工業新聞社 (1994).
- 佐野 千遥、知的人工生命の学習進化、 森北出版 (1996).
- i-makinori、proof-algebra • GitLab、https://gitlab.com/i-makinori/proof-algebra .
- i-makinori、lennma840 • GitLab、https://gitlab.com/i-makinori/lennma840 .

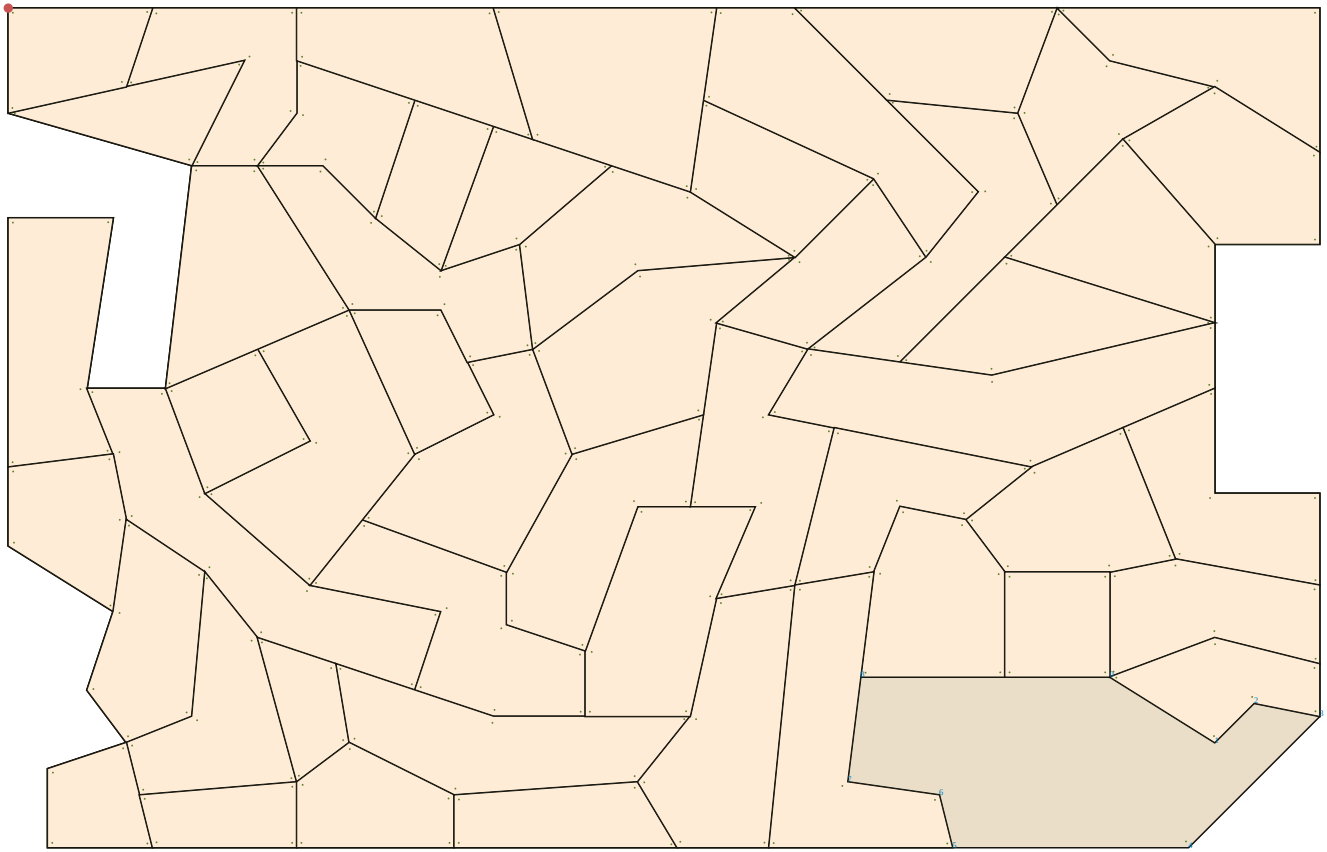
質疑応答

根拠を示しつつ、自ら規則を発見してゆく
プログラムの構想

- 動機。
- コンセプト。
- 文字式の変形方法を証明つきで探す。
- Lennma840 定理発見アルゴリズム (構想)。
- 実現された場合の応用先について。
- エントロピーを低下させるかも知れないプログラムのスケッチ (おまけ)。
- Lennma840(構想) の生命性 (おまけ)。

パズルの解探索プログラム

枠の形状、ピース１の形状、ピース２の形状、、、ピースｎの形状が与えられる。
ピース１からピースｎまでを用いて、枠の形をつくれ。



とあるプログラミングコンテストでの問題

Lennma840(構想) の生命性

宇宙は生命であると考ええている

- 肉体の構築。
- 意識の発生。
- 進化の意思。
- 発見。合目的性。
- 自己複製。