論理型人工知能入門
～安全安心な人工知能を目指して～

東京理科大学創域情報学部情報理工学科
滝本　宗宏

創域情報学部

目次

人工知能（AI）とは？

祝AIにノーベル賞！

物理学賞

- ジョン・ホップフィールド
  連想記憶の非常に簡単なモデルを提案
  → 物理学者の参入を促進

- ジェフリー・ヒントン
  深層学習（deep learning）を発明

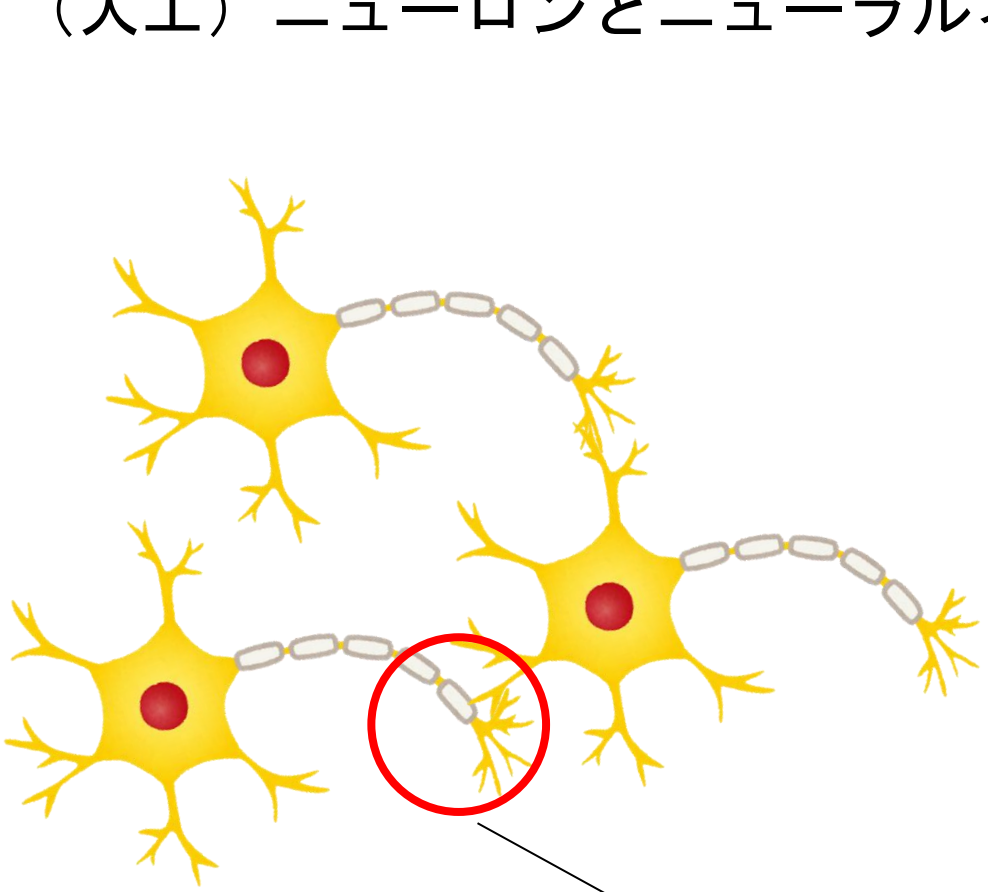
化学賞

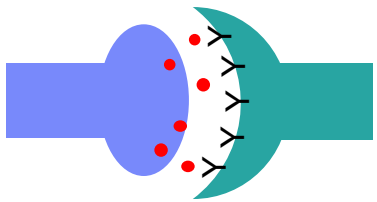
- デイヴィッド・ベイカー
  AIによるたんぱく質の構造予測に成功

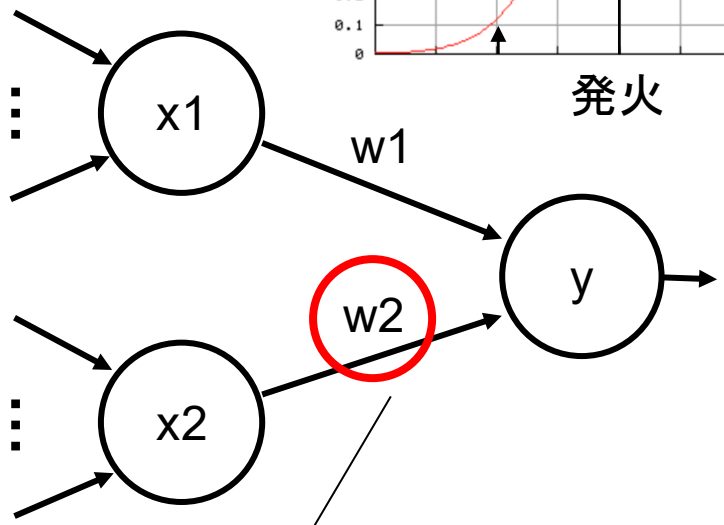
- デミス・ハサビス
- ジョン・ジャンパー
  AI（ AlphaFold ）の実現

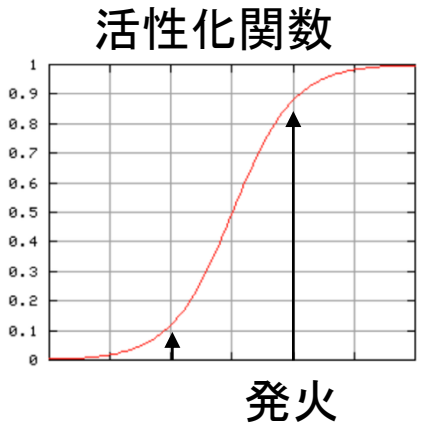
人工知能（AI）とは？

- （人工）ニューロンとニューラルネットワーク





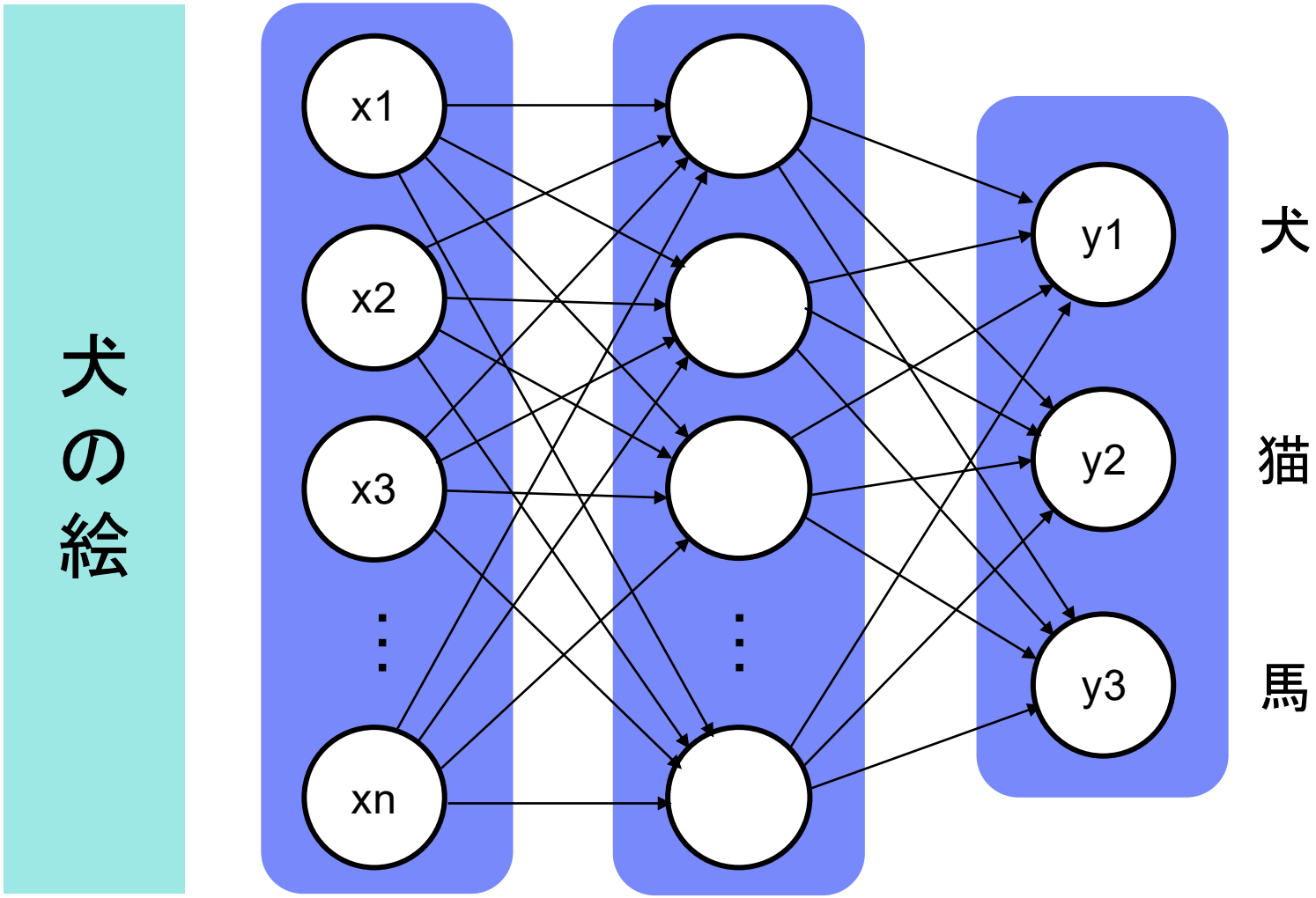




シナプス
（重み）

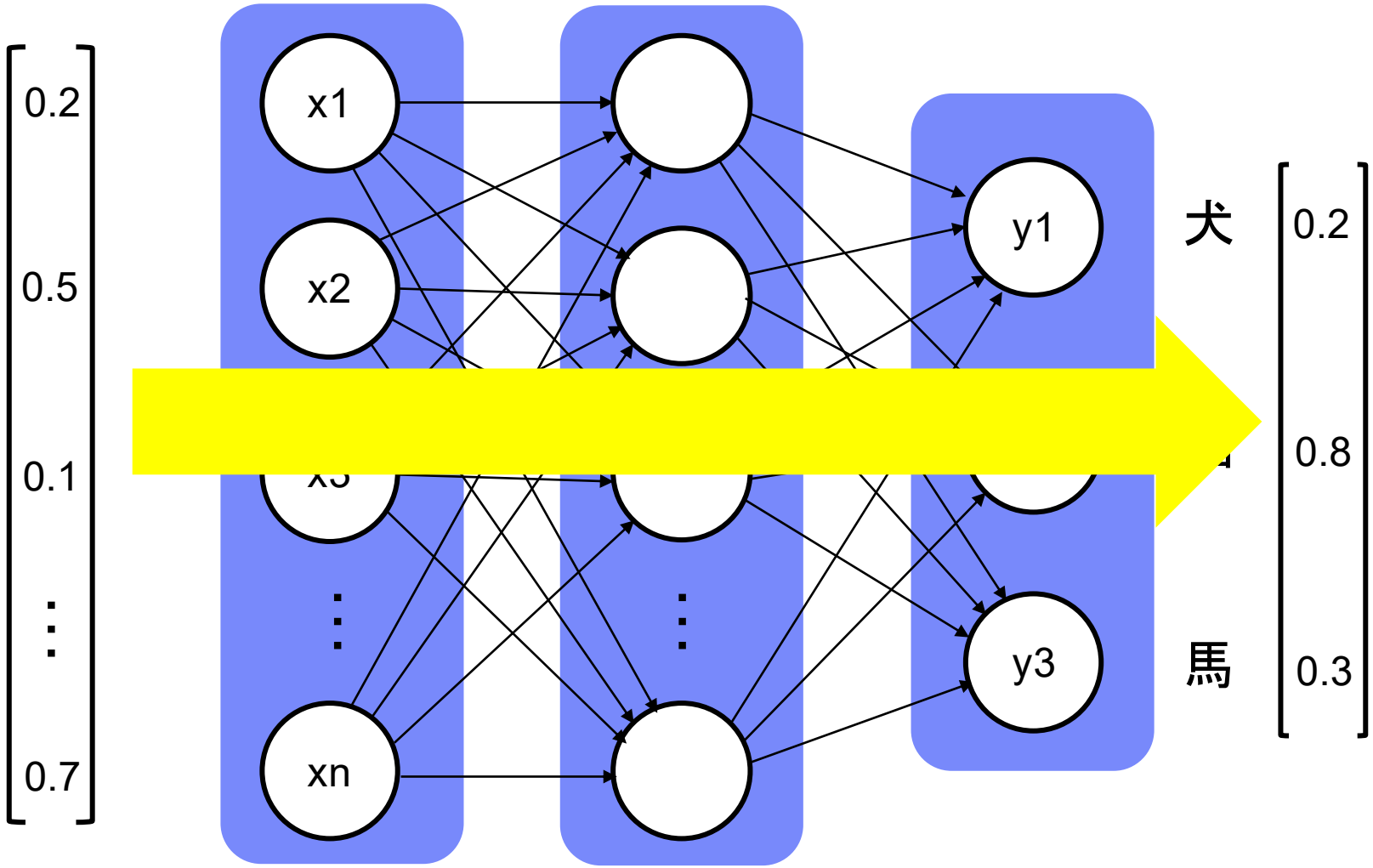
人工知能（AI）とは？

- 誤差逆伝播



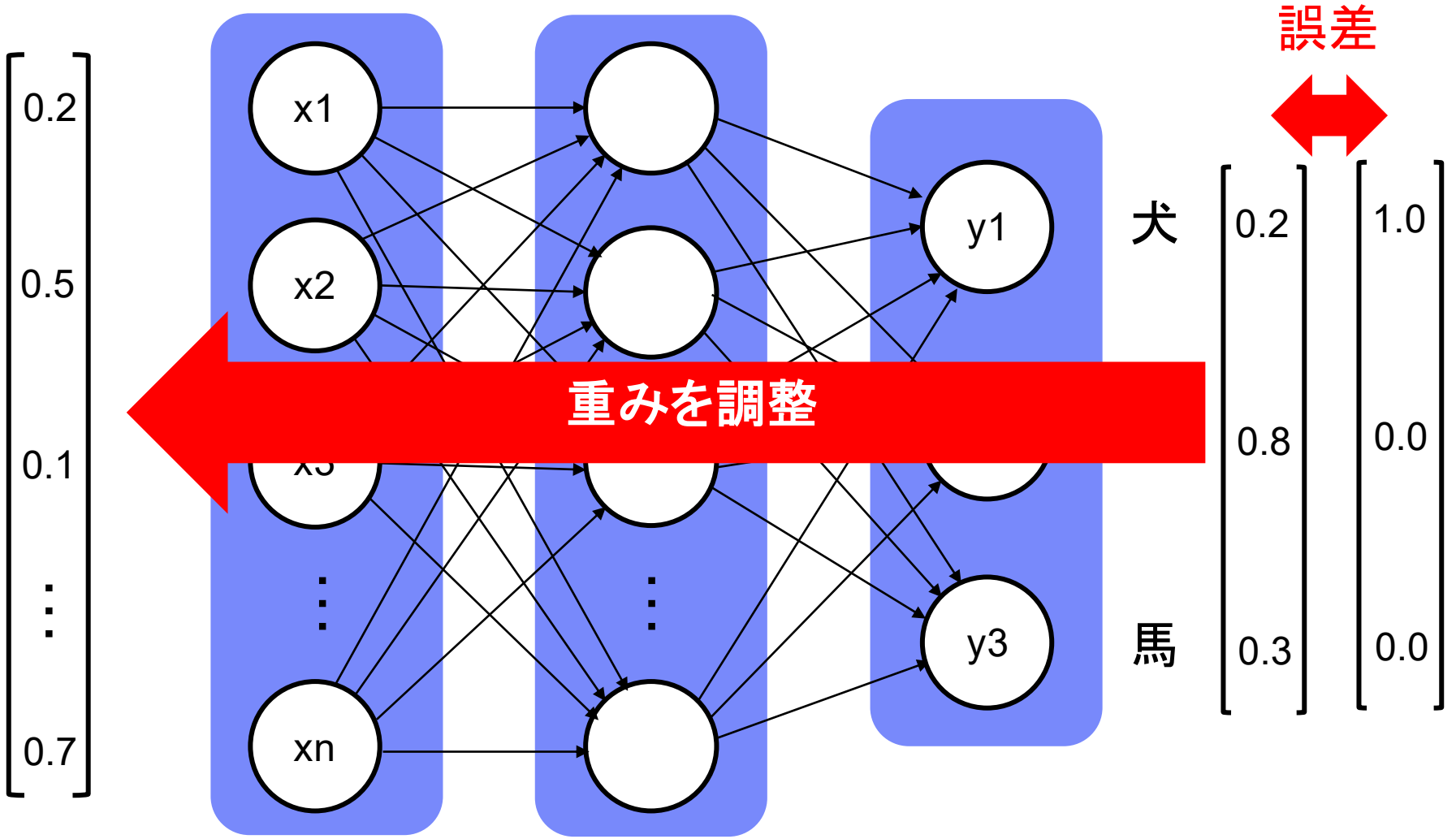
人工知能（AI）とは？

- 誤差逆伝播



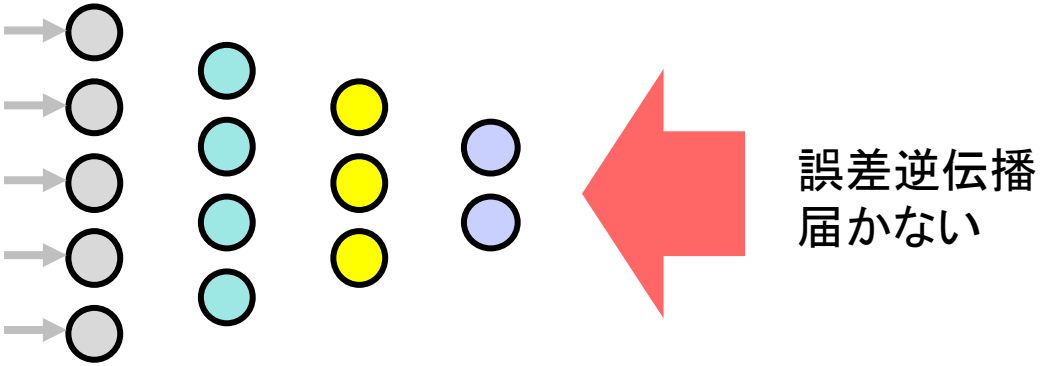
人工知能（AI）とは？

- 誤差逆伝播

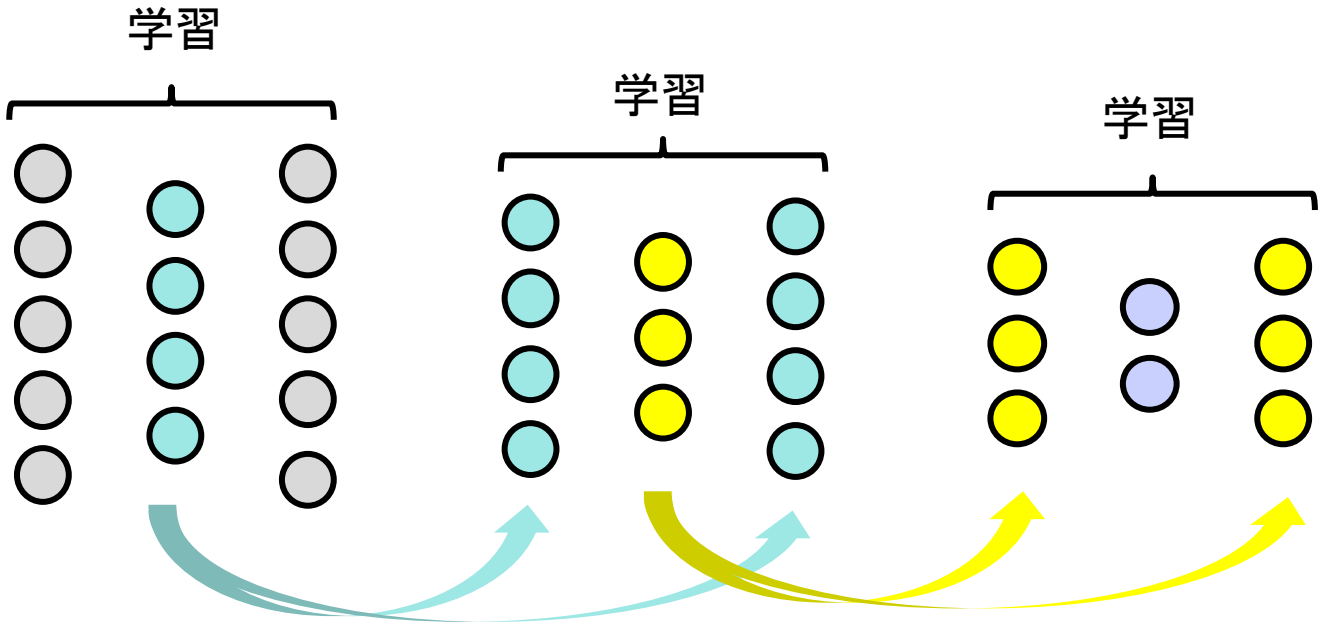


人工知能（AI）とは？

深層学習（Deep Learning）の登場



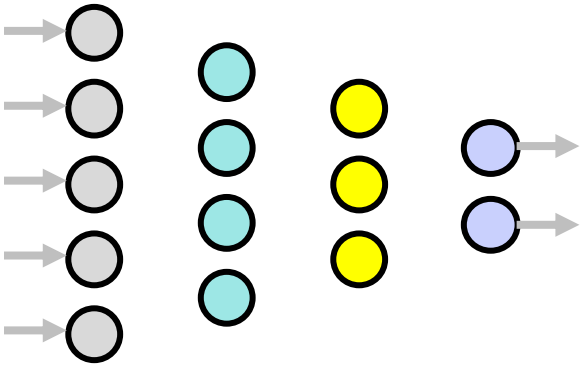
- 自己符号化（Autoencoder）

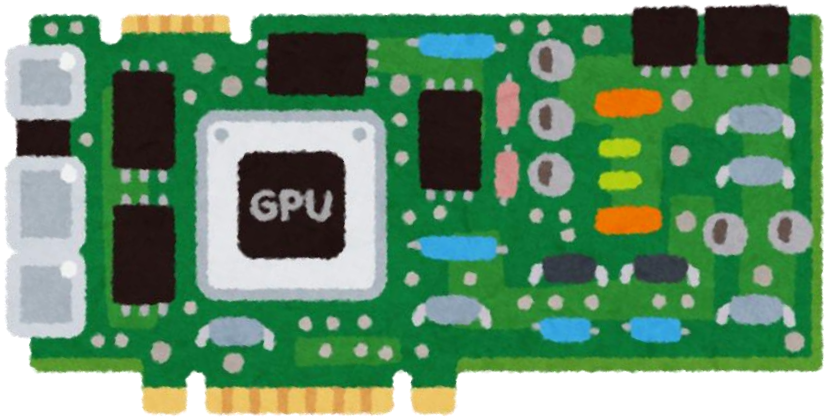


人工知能（AI）とは？

深層学習（Deep Learning）の登場

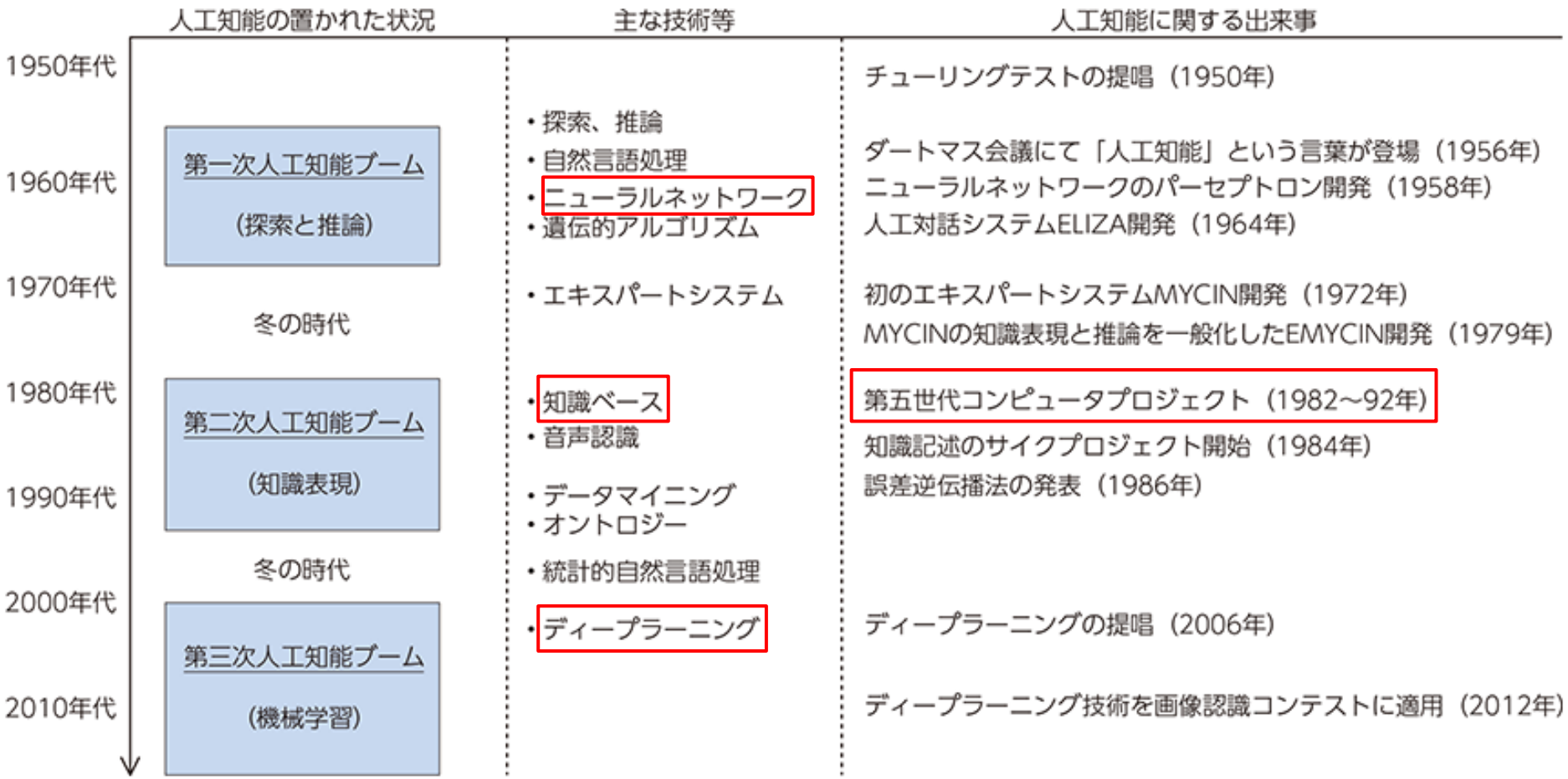
- GPU（並列計算ボード）による高速計算





人工知能（AI）とは？

深層学習（Deep Learning）の登場



論理型AIの重要性と問題点

- 非構造化データ：整理されていない生のデータ
  副次的に生成 ‥ 画像データ，音声データ，テキストなど

- 構造化データ：整理されたデータ
  深層学習（Deep Learning）や機械学習によって，非構造化データから生成 ‥ 表データ，XMLデータなど

- 深層学習や機械学習の流行
  → 多くの構造化データ

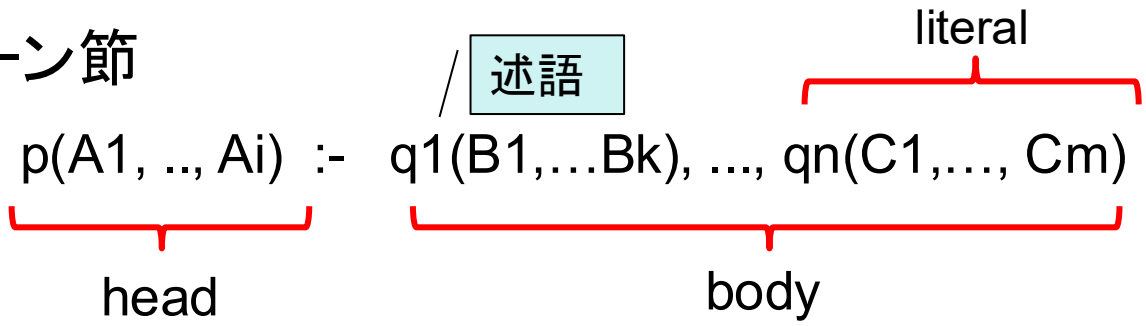
- 構造化データ間の知見（関係）の重要性

帰納論理プログラミング（説明可能AI，XAI）
… 計算コスト高い．

ニューラル
ネットワークによる
高速化

知識の表現と論理型プログラム

- ホーン節



literal1 かつ literal2 かつ … かつ literaln なら head

- ルール

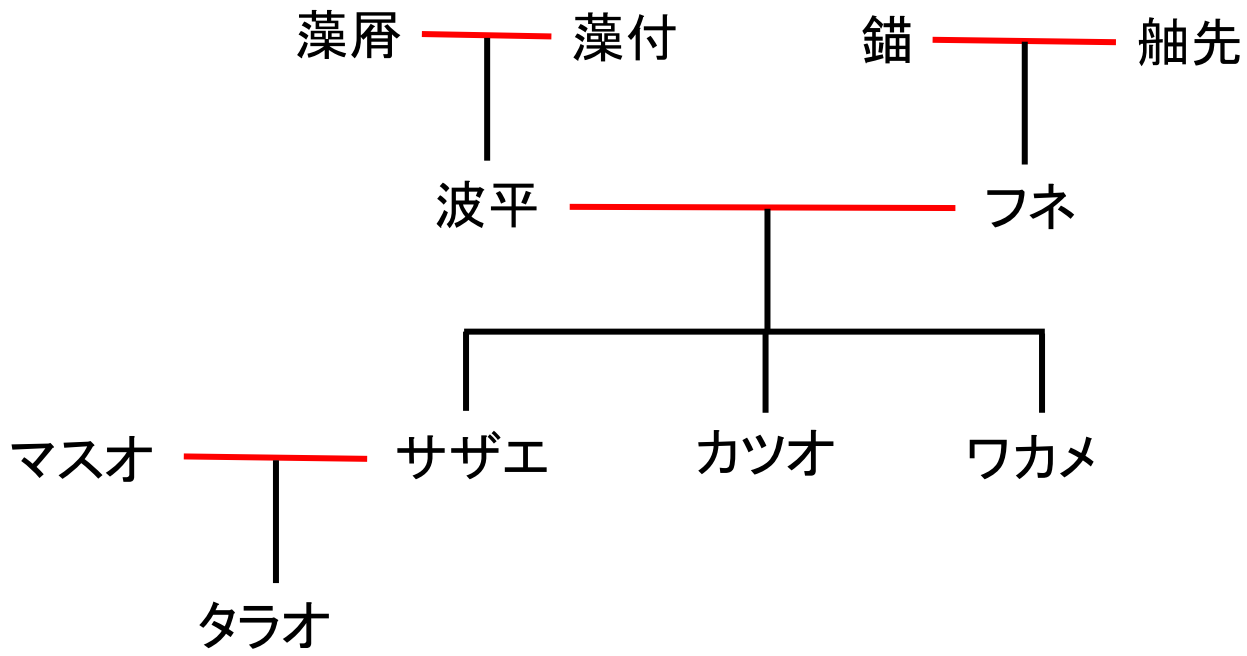
head :- literal1, literal2, … , literaln

- 事実（bodyがない節）

head

知識の表現と論理型プログラム

- 例：磯野家



知識の表現と論理型プログラム

- 事実

father (波平, サザエ)　　father (波平, カツオ)　father (波平, ワカメ)
father (藻屑, 波平)　　mother (サザエ, タラオ)

- ルール

```
parent (A, B)  :-  father (A, B)
parent (A, B)  :-  mother (A, B)
```

- 質問と実行

? parent (サザエ, タラオ)

知識の表現と論理型プログラム

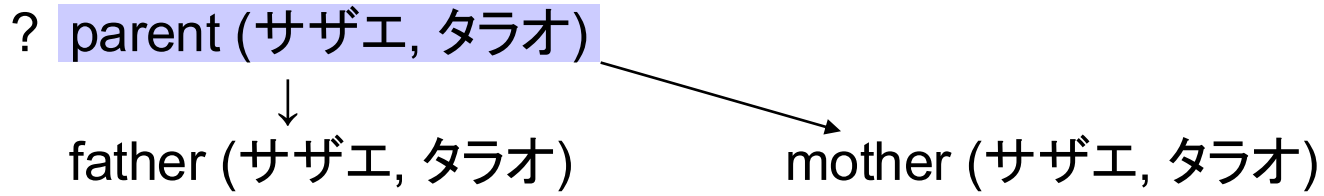
- 事実

father (波平, サザエ)　　father (波平, カツオ)　　father (波平, ワカメ)
father (藻屑, 波平)　　mother (サザエ, タラオ)

- ルール

parent (A, B) :- father (A, B)
parent (A, B) :- mother (A, B)

- 質問と実行



知識の表現と論理型プログラム

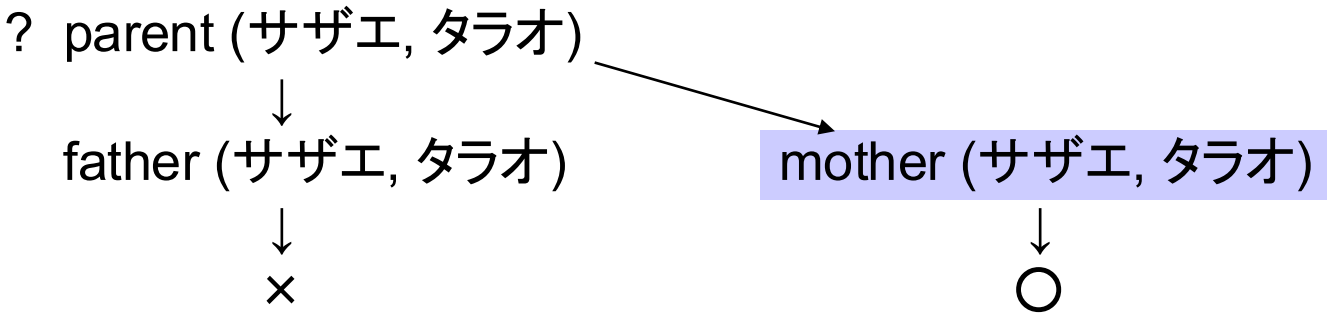
事実

father (波平, サザエ)　father (波平, カツオ)　father (波平, ワカメ)
father (藻屑, 波平)　mother (サザエ, タラオ)

ルール

parent (A, B) :- father (A, B)
parent (A, B) :- mother (A, B)

質問と実行



知識の表現と論理型プログラム

- 事実

father (波平, サザエ)　　father (波平, カツオ)　father (波平, ワカメ)
father (藻屑, 波平)　　mother (サザエ, タラオ)

- ルール

parent (A, B)　:-　father (A, B)
parent (A, B)　:-　mother (A, B)

- 質問と実行

? parent (サザエ, タラオ)
Yes

知識の表現と論理型プログラム

- 事実

father (波平, サザエ)　　father (波平, カツオ)　father (波平, ワカメ)
father (藻屑, 波平)　　mother (サザエ, タラオ)

- ルール

parent (A, B)　:-　father (A, B)
parent (A, B)　:-　mother (A, B)

- 質問と実行（変数を含む場合）

？ parent (波平, B)

知識の表現と論理型プログラム

- 事実

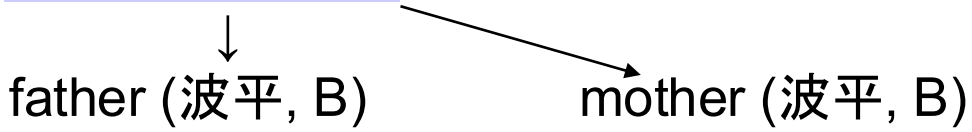
father (波平, サザエ)　　father (波平, カツオ)　　father (波平, ワカメ)
father (藻屑, 波平)　　mother (サザエ, タラオ)

- ルール

parent (A, B) :- father (A, B)
parent (A, B) :- mother (A, B)

- 質問と実行

? parent (波平, B)



知識の表現と論理型プログラム

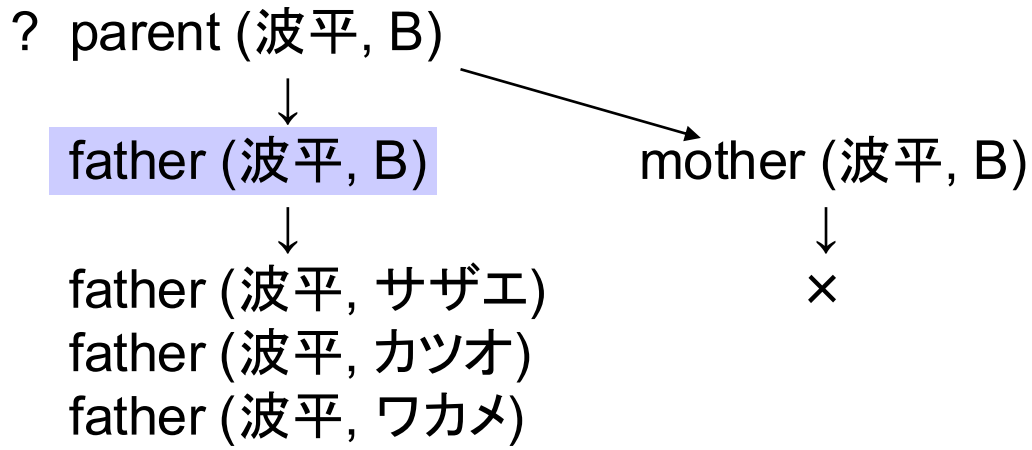
- 事実

father (波平, サザエ)　father (波平, カツオ)　father (波平, ワカメ)
father (藻屑, 波平)　mother (サザエ, タラオ)

- ルール

```
parent (A, B)  :-  father (A, B)
parent (A, B)  :-  mother (A, B)
```

- 質問と実行



知識の表現と論理型プログラム

事実

father (波平, サザエ)　　father (波平, カツオ)　father (波平, ワカメ)
father (藻屑, 波平)　　mother (サザエ, タラオ)

ルール

parent (A, B)　:-　father (A, B)
parent (A, B)　:-　mother (A, B)

質問と実行

? parent (波平, B)
B = サザエ，カツオ，ワカメ

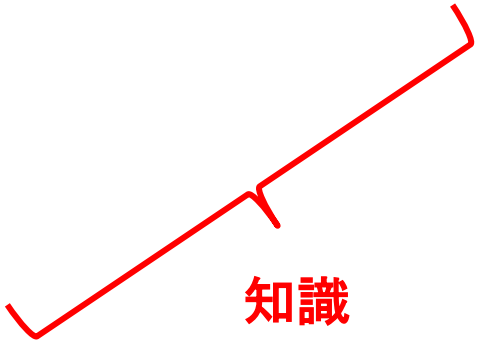
知識の表現と論理型プログラム

- 事実

father (波平, サザエ)　father (波平, カツオ)　father (波平, ワカメ)
father (藻屑, 波平)　mother (サザエ, タラオ)

- ルール

parent (A, B) :- father (A, B)
parent (A, B) :- mother (A, B)



- 質問と実行

? parent (波平, B)
B = サザエ，カツオ，ワカメ

知識の表現と論理型プログラム

演習1

- 準備

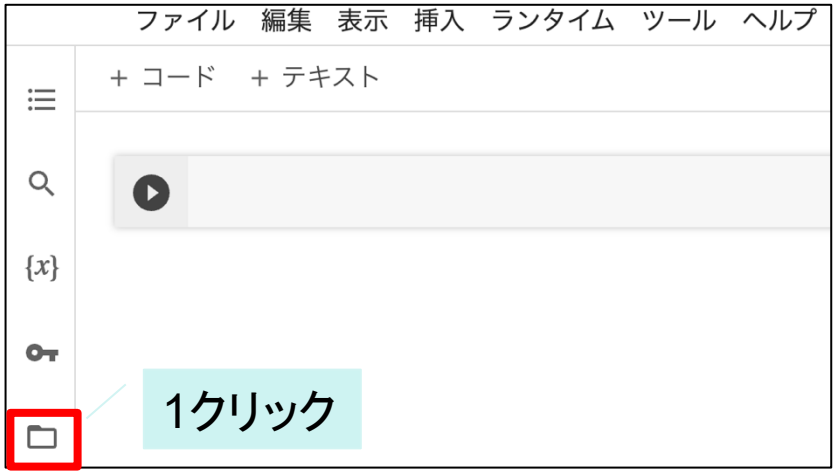
1. Google Colaboratory（以下，Colab）のページを開く.
   https://colab.research.google.com/
2. 「ノートブックを新規作成」する.

知識の表現と論理型プログラム

演習１

- ノートブックの操作

1. ファイルをアップロード




```
!wget https://www.rs.tus.ac.jp/mune/ai/code/ape.zip
```

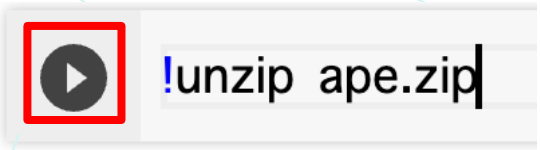
知識の表現と論理型プログラム

演習１

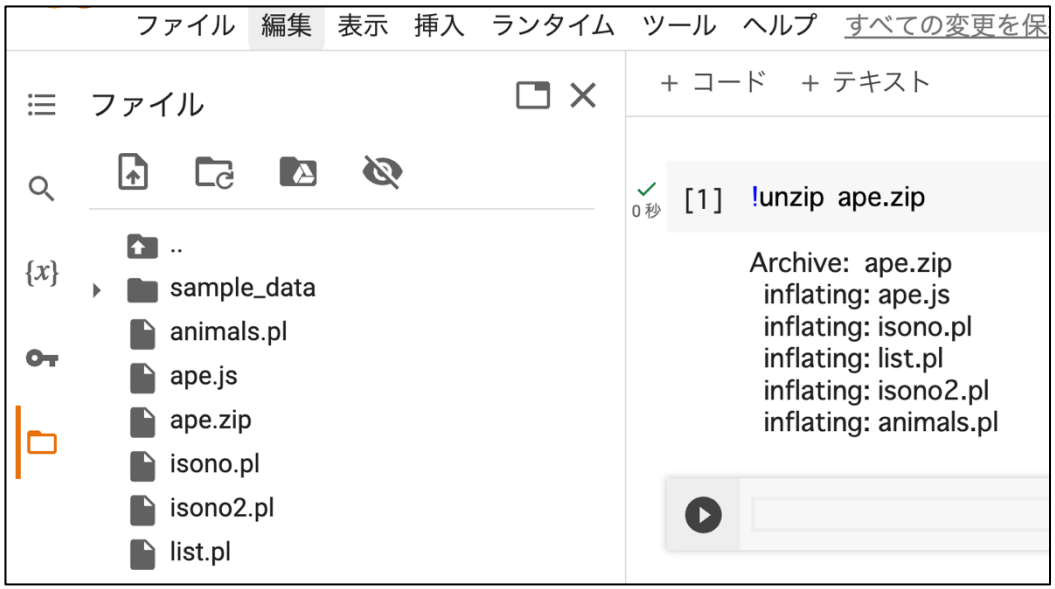
- ノートブックの操作

2. 実行とape.zipの解凍

先頭は「!」　最後は「Shift+Enter」



ここの1クリックでも実行

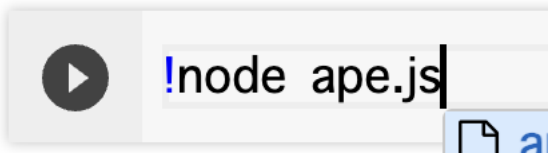


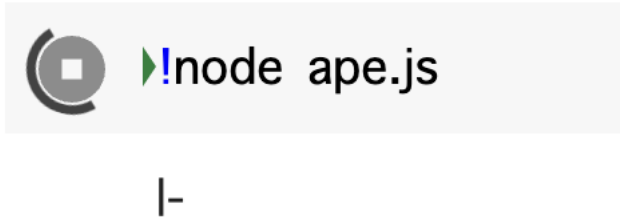
知識の表現と論理型プログラム

演習１

– ノートブックの操作

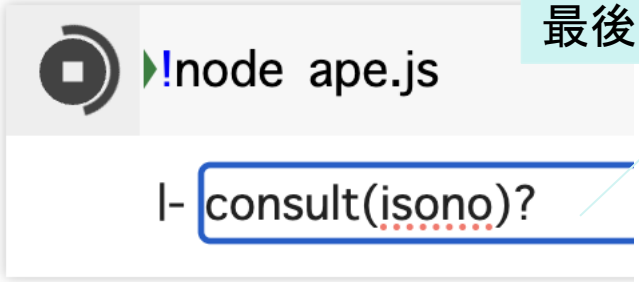
3. XApeの実行



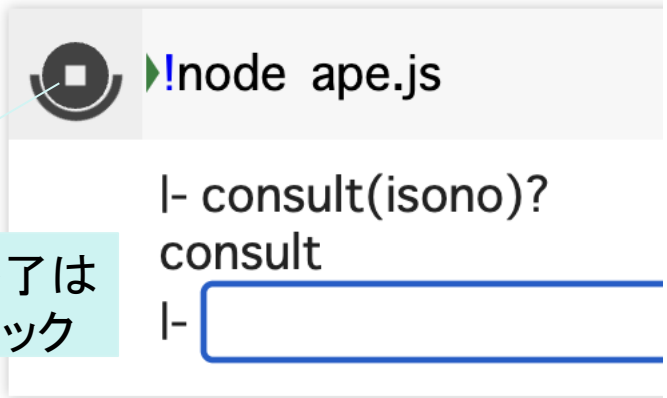


ここをクリックして，
XApeのコマンドを入力

4. XApeのコマンド入力
（ファイルisono.plの読込み）



最後は「Enter」



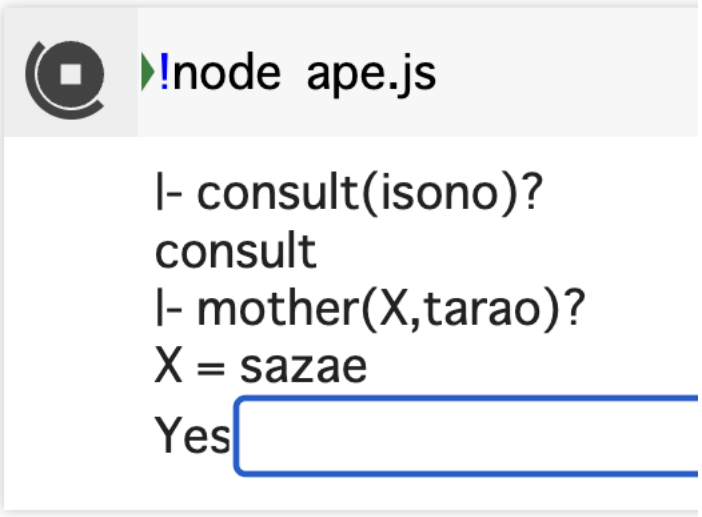
実行の終了は
ここをクリック

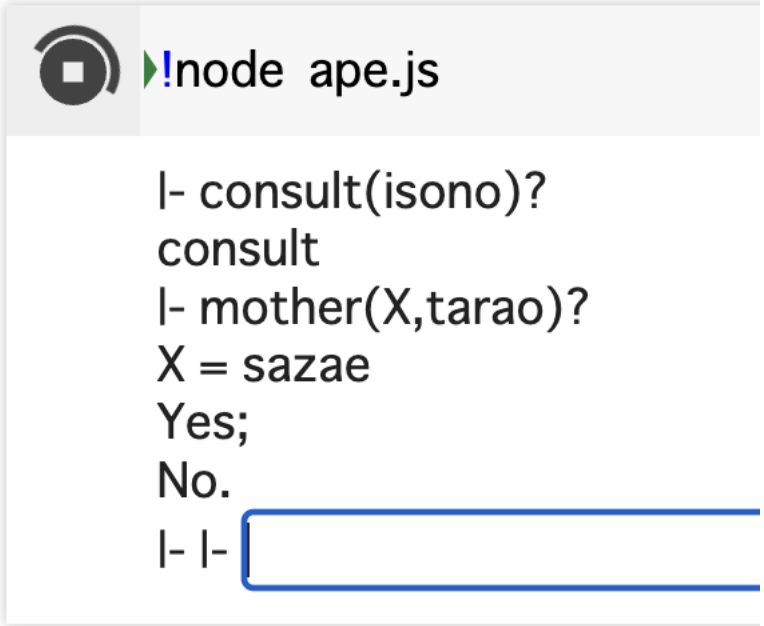
知識の表現と論理型プログラム

演習１

– ノートブックの操作

5. XApeのコマンド入力
（質問の実行）





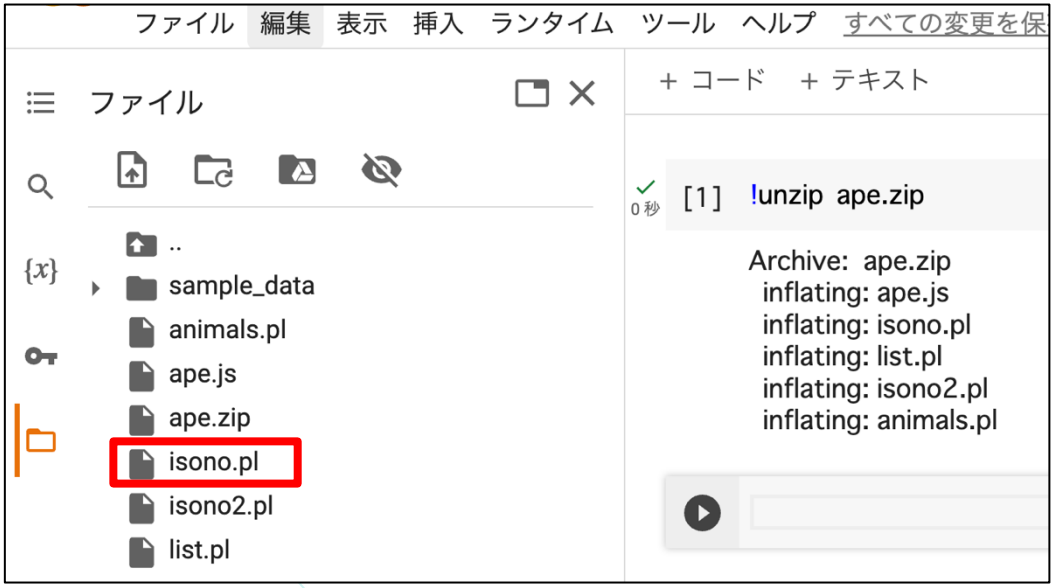
「;」を入力すると次候補を表示.
「.」を入力すると終了.

知識の表現と論理型プログラム

演習１

- ノートブックの操作

6. ファイルを開く/編集する.



ダブルクリック

注：「%」はコメントなので，適宜付けたり消したりしてください.

知識の表現と論理型プログラム

演習１

– ノートブックの操作

7. isono.plに「parent」を加えてみよう.

```
36 mother(fune, sazae).
37 mother(fune, katsuo).
38 mother(fune, wakame).
39
40 parent(X,Y) :- father(X,Y).
41 parent(X,Y) :- mother(X,Y).
```

付加

```
!node ape.js
|- consult(isono)?
consult
|- parent(X, sazae)?
X = fune
Yes;
X = namihei
Yes;
No.
|- |-
```

帰納論理プログラミング

- 背景知識

  father (波平, サザエ)　　father (波平, カツオ)　　father (波平, ワカメ)
  father (藻屑, 波平)　　mother (サザエ, タラオ)

  parent (A, B) :- father (A, B)
  parent (A, B) :- mother (A, B)

- 事例（grandfather)

  [正例]　　[負例]

  grandfather (波平, タラオ)　　grandfather (波平, サザエ)

  grandfather (藻屑, カツオ)　　grandfather (藻屑, タラオ)

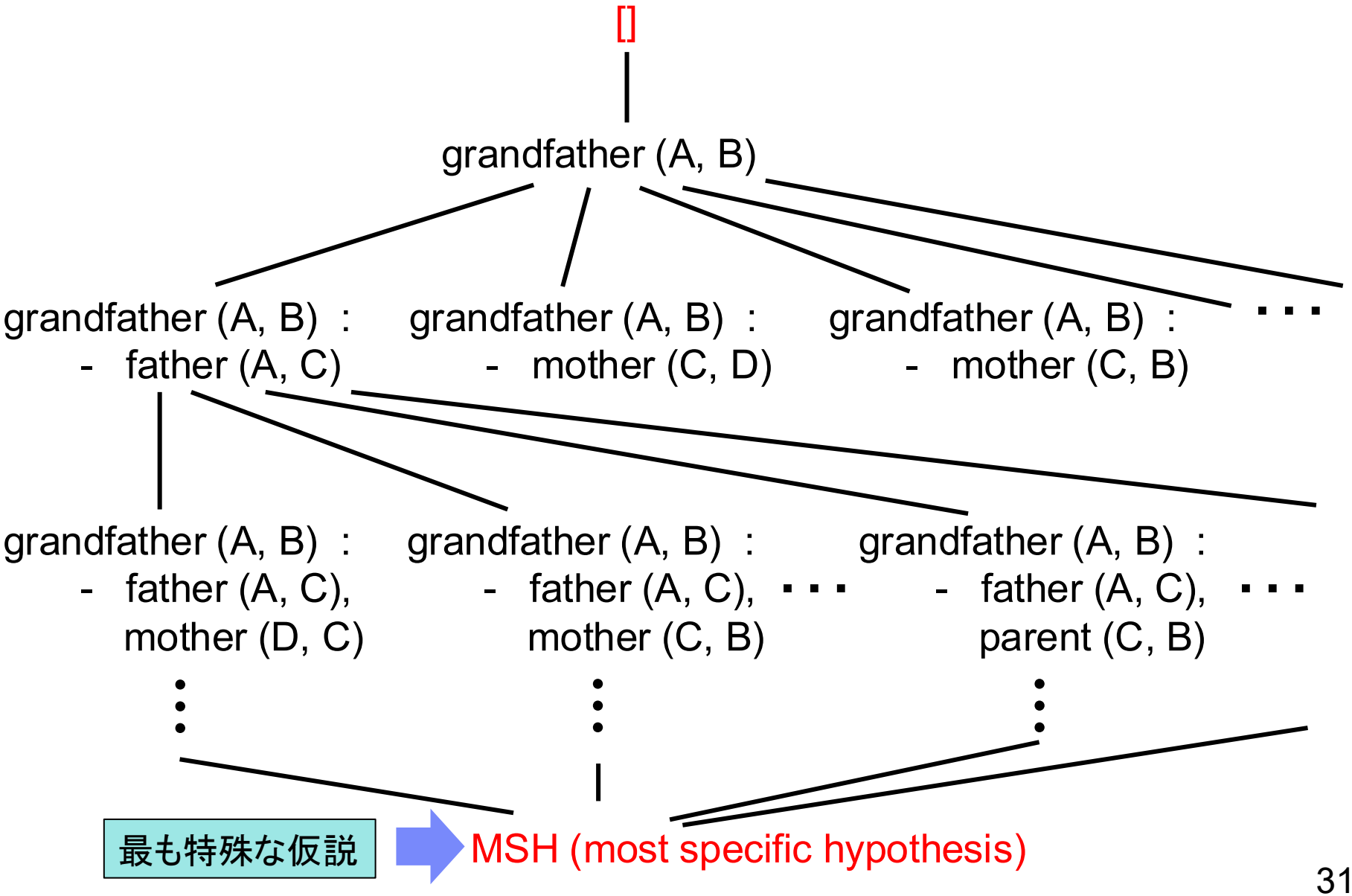
  ⋮　　⋮

- 仮説（新しいルール）の生成

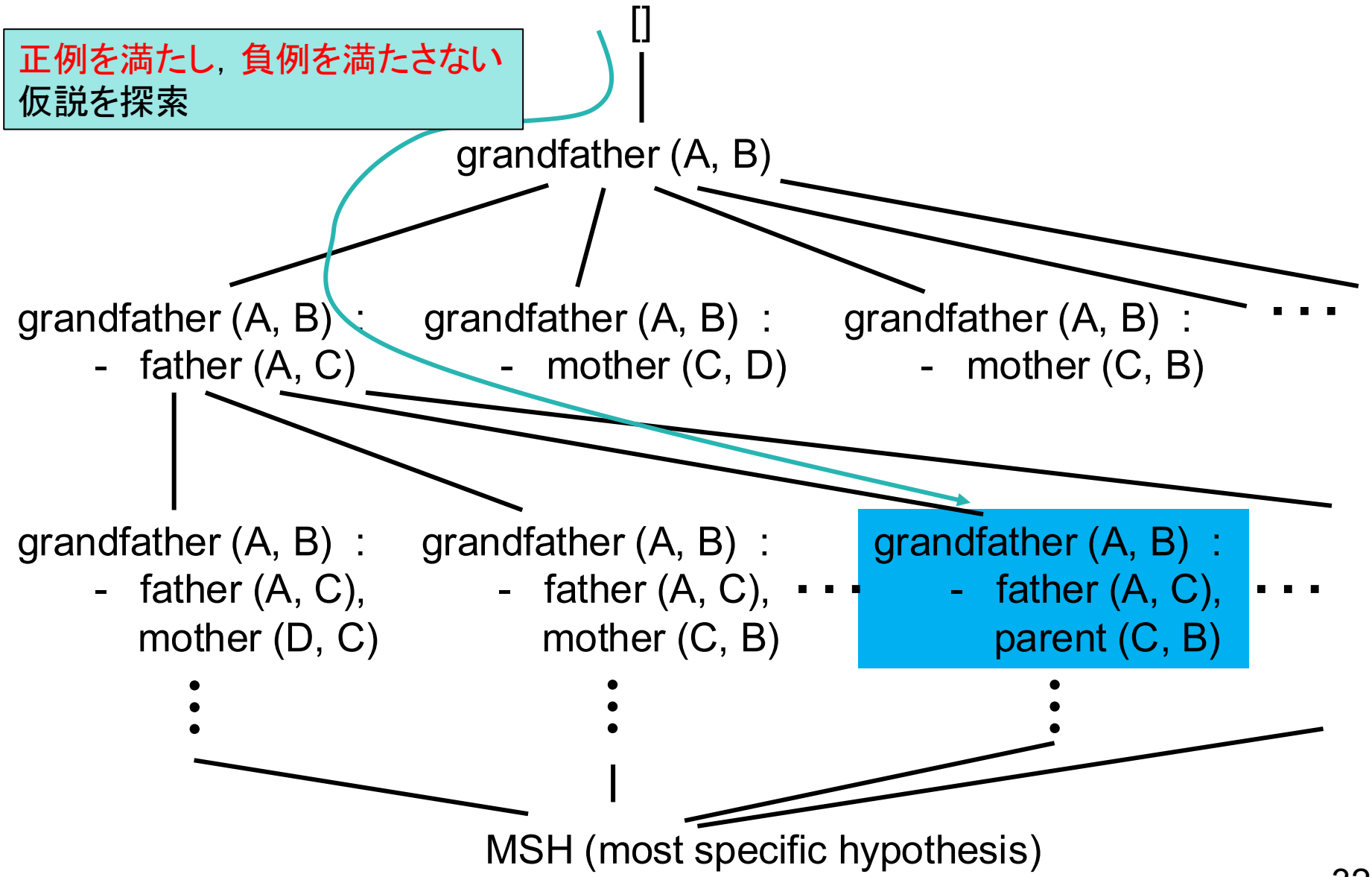
  背景知識と事例のパーツの組合せ → 正例を満たし、負例を満たさない
  ルールを生成

  grandfather (A, B) :- father (A, C), parent (C, B)

帰納論理プログラミング



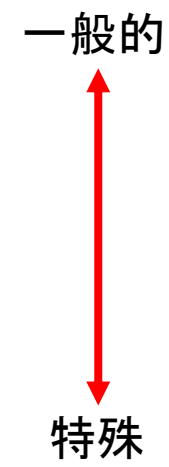
帰納論理プログラミング

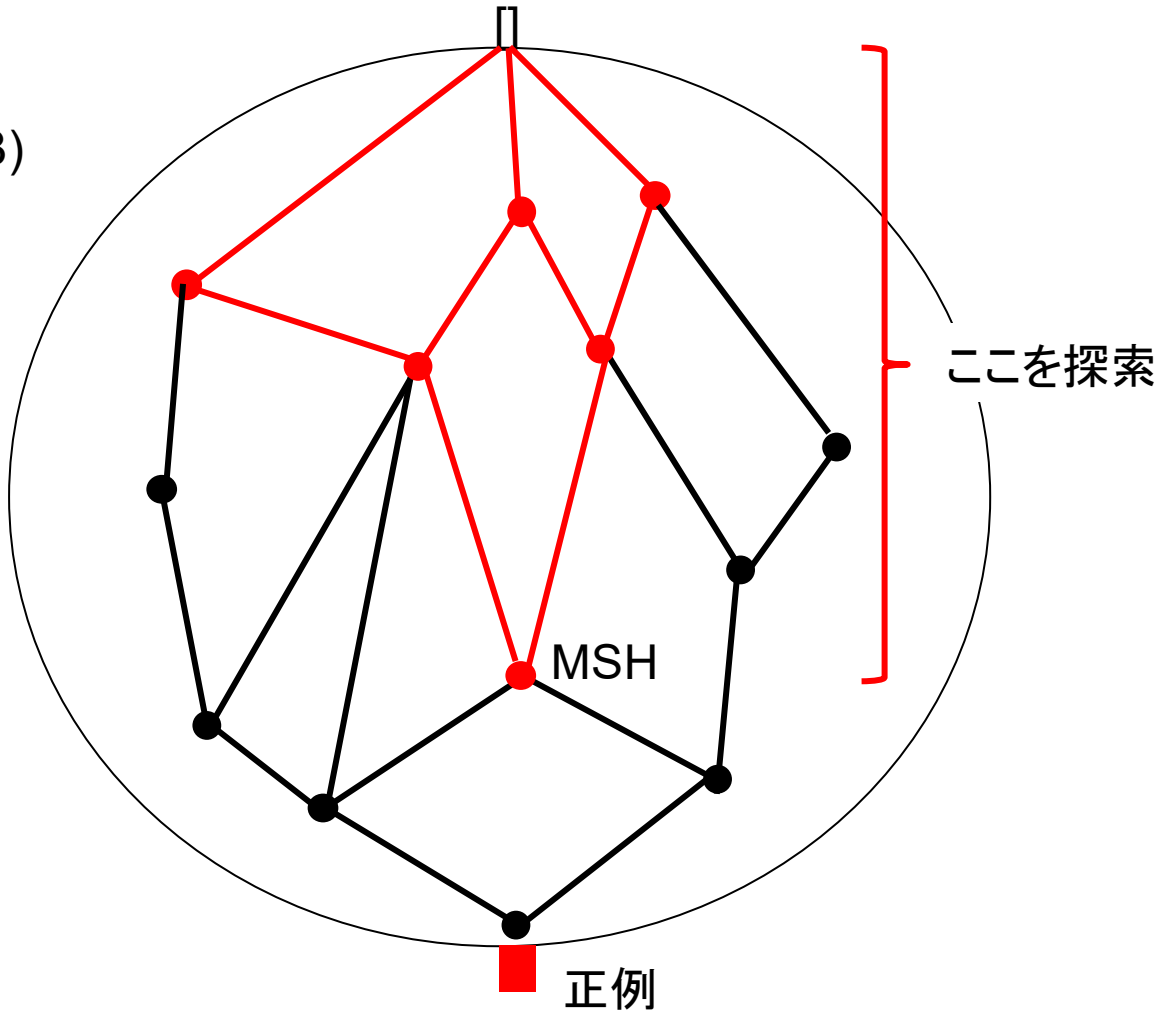


MSHの生成

- MSH（Most Specific Hypothesis）：最弱仮説
  最も特殊な仮説

一般的　　特殊
p(A,B) > p(A,A)
p(A,B) > p(A,B) :- q(A,B)
p(A,B) > p(波平, B)





MSHの生成

- 各正例とモード宣言から生成
  - 例：grandfather(波平, タラオ)
    1. \+ ← 出現した定数
    2. \- ← 背景知識を満たす定数
    3. 各定数を変数で置換.

モード宣言

```
:-modeh(*, grandfather(+person,-person))?
:-modeb(*, father(+person,-person))?
:-modeb(*, mother(+person,-person))?
:-modeb(*, parent(+person,-person))?
```

+father

grandfather(波平, タラオ) :- father(波平, ワカメ),
father(波平, カツオ), father(波平, サザエ),

+mother

grandfather(波平, タラオ) :- father(波平, ワカメ),
father(波平, カツオ), father(波平, サザエ),
~~mother(波平, ？),~~ ~~mother(ワカメ, ？),~~
~~mother(カツオ, ？),~~ mother(サザエ, タラオ)

+parent

grandfather(波平, タラオ) :- father(波平, ワカメ),
father(波平, カツオ), father(波平, サザエ),
mother(サザエ, タラオ),
parent(波平, ワカメ), parent(波平, カツオ),
parent(波平, サザエ), ~~parent(ワカメ, ？),~~
~~parent(カツオ, ？),~~ parent(サザエ, タラオ), ~~parent(タラオ, ？)~~

MSHの生成

- 各正例とモード宣言から生成
  - 例：grandfather(波平, タラオ)

1. \+ ← 出現した定数
2. \- ← 背景知識を満たす定数
3. 各定数を変数で置換.

モード宣言

```
:-modeh(*, grandfather(+person,-person))?
:-modeb(*, father(+person,-person))?
:-modeb(*, mother(+person,-person))?
:-modeb(*, parent(+person,-person))?
```

grandfather(波平, タラオ) :- father(波平, ワカメ),
father(波平, カツオ), father(波平, サザエ),
mother(サザエ, タラオ),
parent(波平, ワカメ), parent(波平, カツオ),
parent(波平, サザエ), parent(サザエ, タラオ)

波平 → A, タラオ → B, ワカメ → C, カツオ → D, サザエ → E

grandfather(A, B) :- father(A, C),
father(A, D), father(A, E),
mother(E, B),
parent(A, C), parent(A, D),
parent(A, E), parent(E, B)

仮説の探索

- A*-likeアルゴリズム
  - 精密化（オープン）：現仮説候補にMSHからリテラルを加える

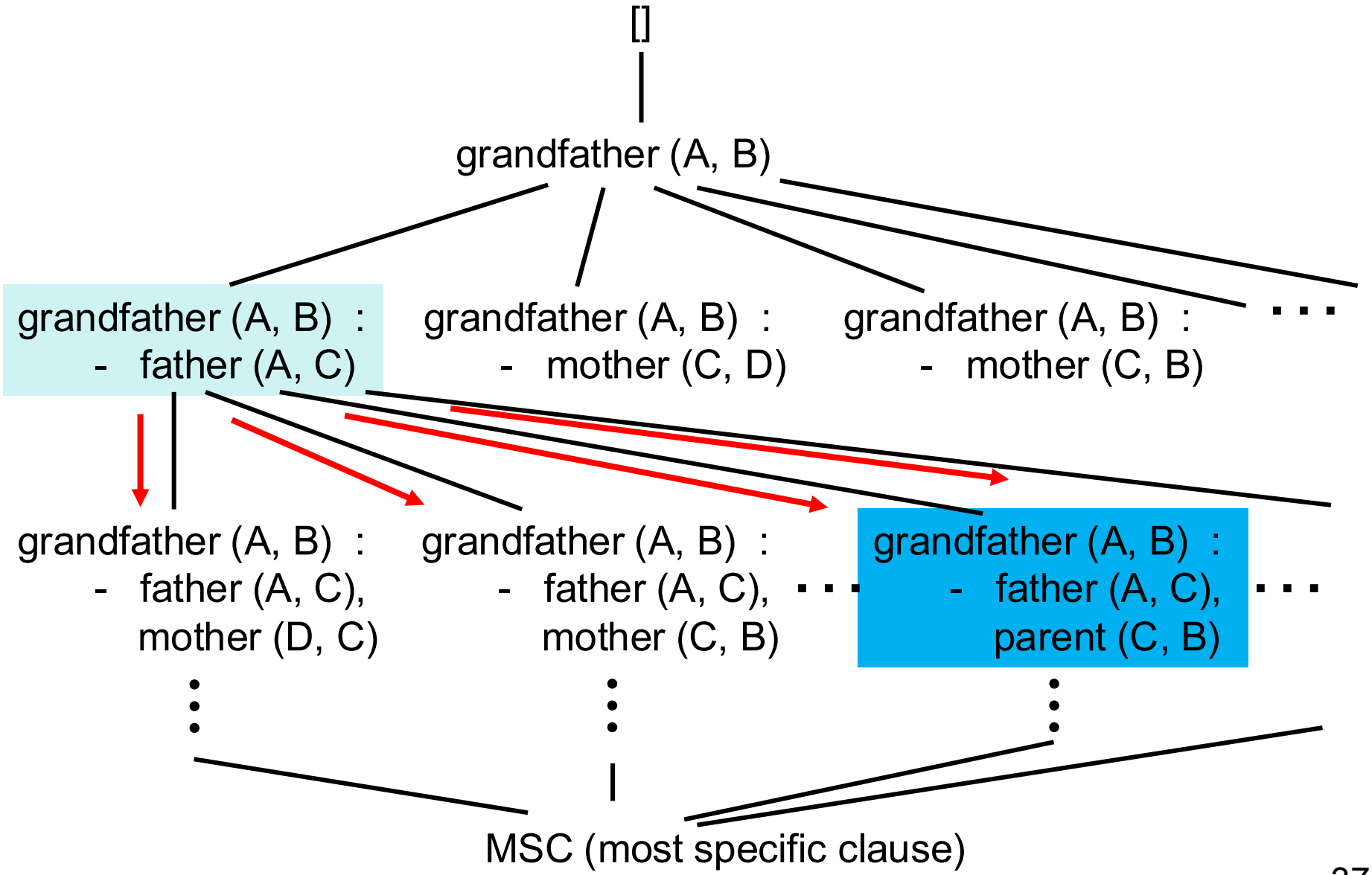
MSH: grandfather(A, B) :-
father(A, C), father(A, D), father(A, E),
parent(A, C), parent(A, D), parent(A, E),
mother(E, B), parent(E, B).

現仮説候補：grandfather(A, B) :- father(A, E).

新仮説候補：grandfather(A, B) :- father(A, E), parent(E,B).

評価値の計算

仮説の探索


[]
grandfather (A, B)
grandfather (A, B) :
- father (A, C)
grandfather (A, B) :
- mother (C, D)
grandfather (A, B) :
- mother (C, B)
grandfather (A, B) :
- father (A, C),
mother (D, C)
grandfather (A, B) :
- father (A, C),
mother (C, B)
grandfather (A, B) :
- father (A, C),
parent (C, B)
MSC (most specific clause)

仮説の探索

- A*-likeアルゴリズム
  - 被覆: 正例や負例が仮説を満たすとき，その仮説はその正例や負例を被覆する.
  - 評価関数 f: 評価値を計算（スコアが大きいほうがよい）

$$f = p - (n + c + h)$$

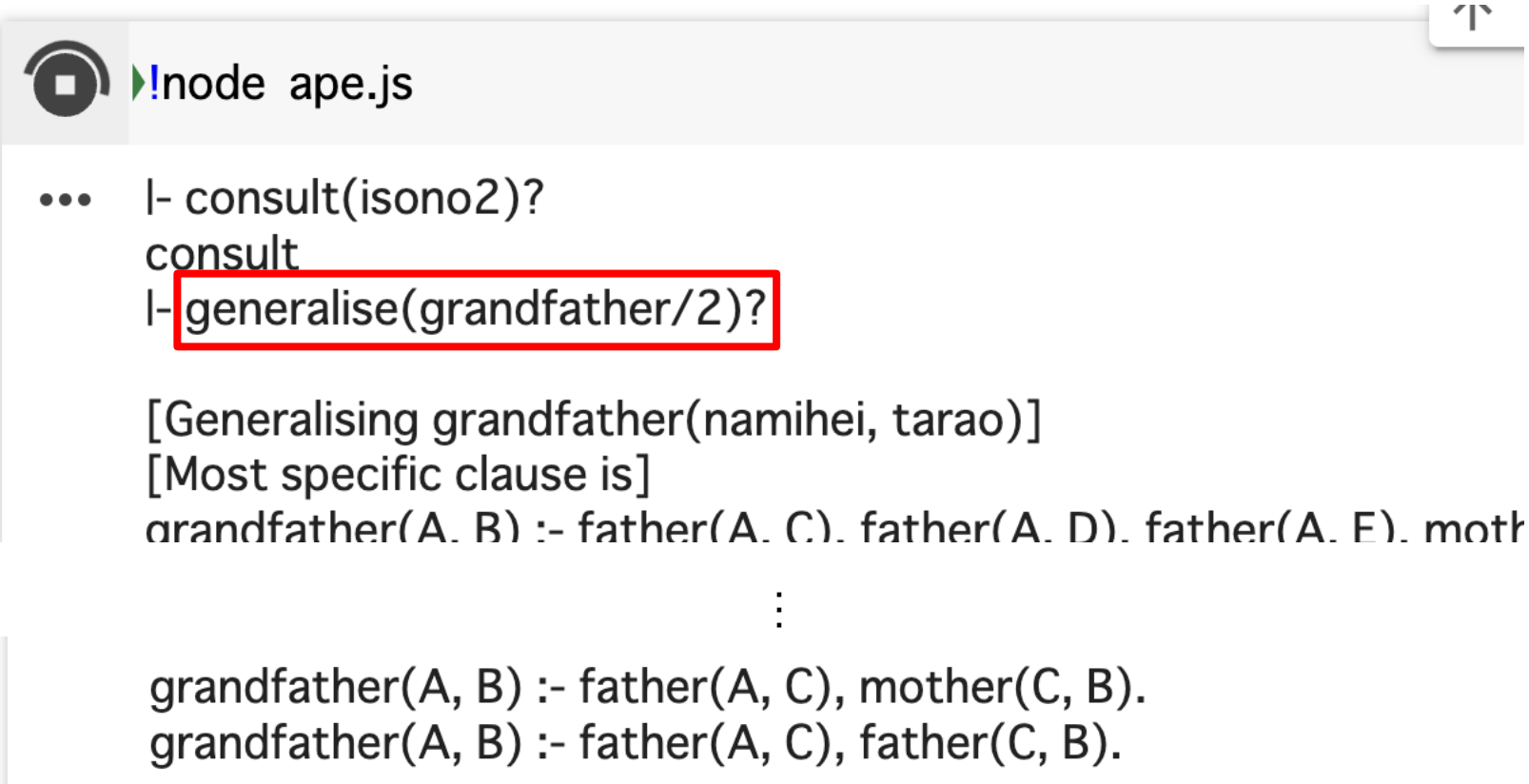
p	n	c	h
被覆する**正例の数**	被覆する**負例の数**	仮説のリテラル数	残りの予測リテラル数

1. 被覆する正例が多く，
2. 被覆する負例が少なく，
3. リテラル数が少ない（単純）

良い仮説

帰納論理プログラミング

- 演習２
  - 「grandfather」の学習
    1. 「isono2.pl」を読み込む.
    2. 「generalise(grandfather/2)?」を実行.



帰納論理プログラミング

演習２

- 背景知識とmodeの付加
  1. 「isono2.pl」に次のmode宣言を付加：
     :-modeb(*, parent(+person,-person))?
  2. 「isono2.pl」に次の背景知識を付加：
     parent(X,Y) :- father(X,Y).
     parent(X,Y) :- mother(X,Y).
  3. 「isono2.pl」の読込み.
  4. 「generalise(grandfather/2)?」の実行.

- 「animals」の学習
  1. 「animals.pl」の読込み.
  2. 「generalise(class/2)?」の実行.

正例からの学習

- 確率論理プログラミング（SLP, stochastic logic programming）
  - 事例空間全体の確率分布 ＝ 正例の確率分布

  - SLPによって，確率的に正例の生成を試みる.

ほとんどが負例

  - 例：

モード宣言：	:- modeh(*,p(+list,-elem))?
背景知識：	elem(a). elem(b). elem(c).
	list([]).
	list([H|T]) :- elem(H), list(T).
正例：	p([b,c],c).
	p([a,b,c],b).
	p([a,a],a).

  - modehから節を生成：‘*p’(A,B) :- list(A), elem(B).

正例からの学習

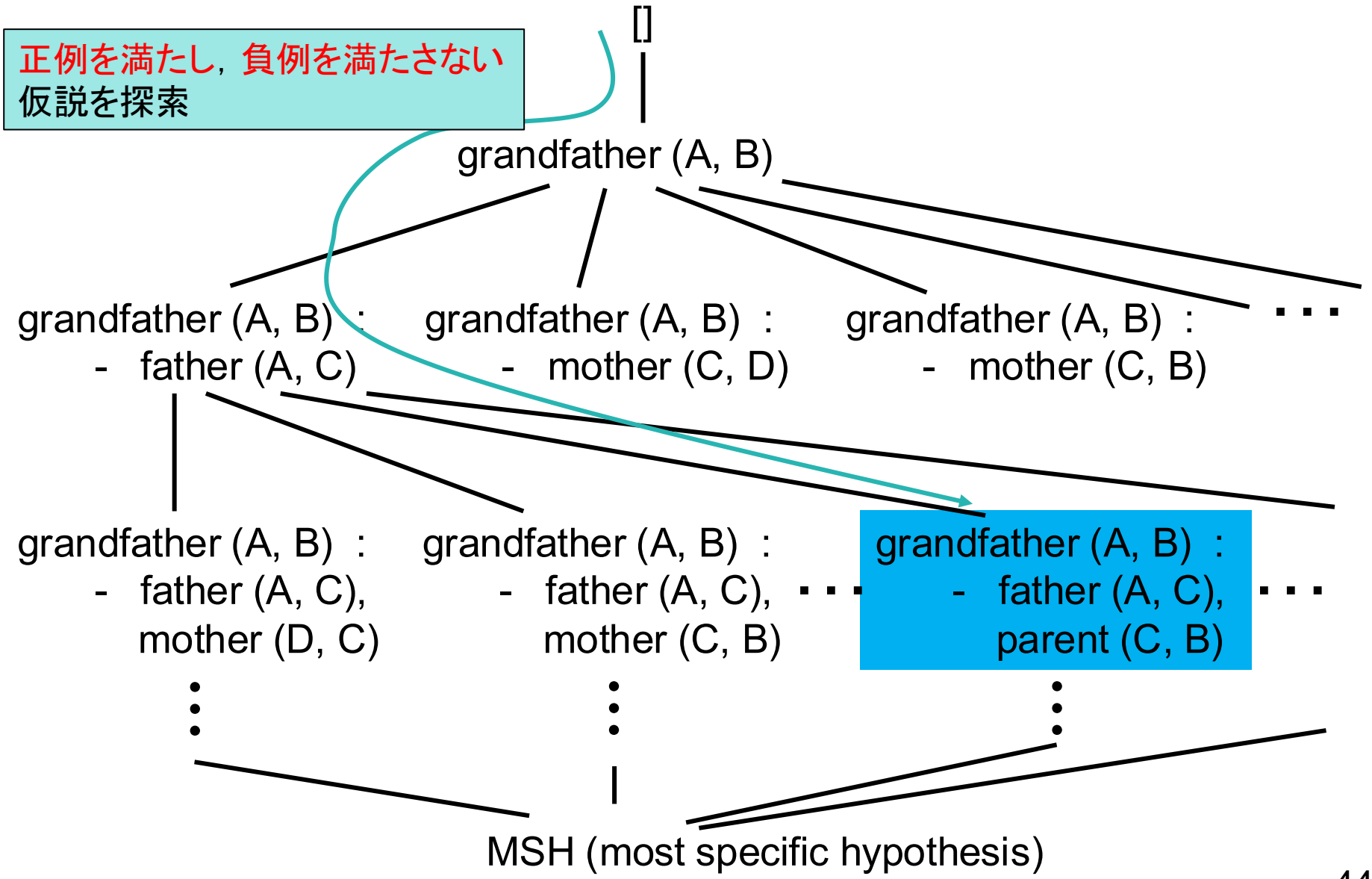
- 確率論理プログラミング（SLP, stochastic logic programming）
  - 例：　モード宣言：　:- modeh(*,p(+list,-elem))?
    背景知識：　elem(a). elem(b). elem(c).
    list([]).
    list([H|T]) :- elem(H), list(T).
    正例：　p([b,c],c).
    p([a,b,c],b).
    p([a,a],a).

1. modehから節を生成：‘*p’(A,B) :- list(A), elem(B).
2. 確率の付与：すべての正例に対して，prologのゴール節として実行（融合操作数をカウント）．　4:elem(a).　3:elem(b).　3:elem(c).
   3:list([]).　7:list([H|T]) :- elem(H), list(T).
3. サンプリング：’*p’(X,Y). を実行
   listの選択：3/10の確率：list([]).　7/10の確率:list([H|T] :- elem(H), list(T).
   elemの選択：
   4/10の確率：elem(a). 3/10の確率:elem(b). 3/10の確率：elem(c).

帰納論理プログラミング

- 演習3
  - 「isono2.pl」の先頭にある「set(posonly)?」のコメントをはずして実行.
  - 「animals.pl」の先頭にある「set(posonly)?」のコメントをはずして実行.

ニューロ記号処理による事例検査



ニューロ記号処理による事例検査

1. 多くの正例を満たす.
2. 少ない負例を満たす.
3. 単純.

仮説候補1:
grandfather(A,B) :- parent(A,C), parent(C,B).
（正例:4，負例:1）

正例:
grandfather (波平, タラオ)
grandfather (藻屑, サザエ)
grandfather (藻屑, カツオ)
grandfather (藻屑, ワカメ)

負例:
grandfather (波平, サザエ)
grandfather (波平, カツオ)
grandfather (波平, ワカメ)
grandfather (藻屑, 波平)
grandfather (フネ, タラオ)

仮説候補2:
grandfather(A,B) :- father(A,C), parent(C,B).
（正例:4，負例:0）

ニューロ記号処理による事例検査

1. 多くの正例を満たす.
2. 少ない負例を満たす.
3. 単純.

仮説候補1:
grandfather(A,B) :- parent(A,C), parent(C,B).
（正:20%, 否:80%）

判断

正例:
grandfather (波平, タラオ)
grandfather (藻屑, サザエ)
grandfather (藻屑, カツオ)
grandfather (藻屑, ワカメ)

○

学習

負例:
grandfather (波平, サザエ)
grandfather (波平, カツオ)
grandfather (波平, ワカメ)
grandfather (藻屑, 波平)
grandfather (フネ, タラオ)

✗

判断

仮説候補2:
grandfather(A,B) :- father(A,C), parent(C,B).
（正:90%, 否:10%）

ニューロ記号処理による事例検査

事例の学習

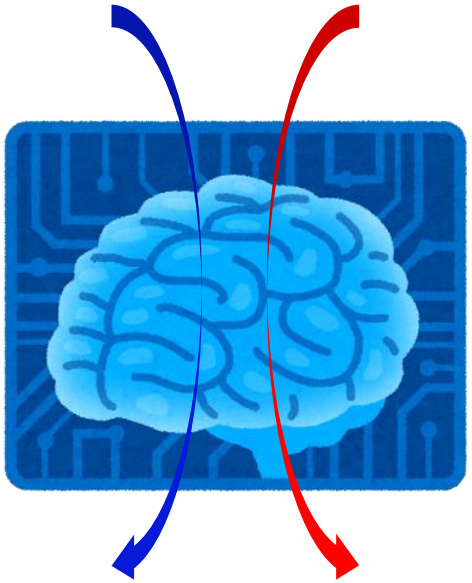
対応するリテラル：1.0，他：0.0

入力：
father(A,C), mother(A,C), parent(A,C), parent(C,B), …

正例：
- grandfather (波平, タラオ)
- grandfather (藻屑, サザエ)
- grandfather (藻屑, カツオ)
- grandfather (藻屑, ワカメ)

負例：
- grandfather (波平, サザエ)
- grandfather (波平, カツオ)
- grandfather (波平, ワカメ)
- grandfather (藻屑, 波平)
- grandfather (フネ, タラオ)

MSH
背景知識で
言い替える



MSH
背景知識で
言い替える

出力：　**(正：100%, 否：0%)**　(正：0%, 否：100%)

ニューロ記号処理による事例検査

仮説が妥当かニューラルネットワークに尋ねながら仮説を探索

[]

grandfather (A, B)

grandfather (A, B) :
- father (A, C)

grandfather (A, B) :
- mother (C, D)

grandfather (A, B) :
- mother (C, B)

grandfather (A, B) :
- father (A, C),
mother (D, C)

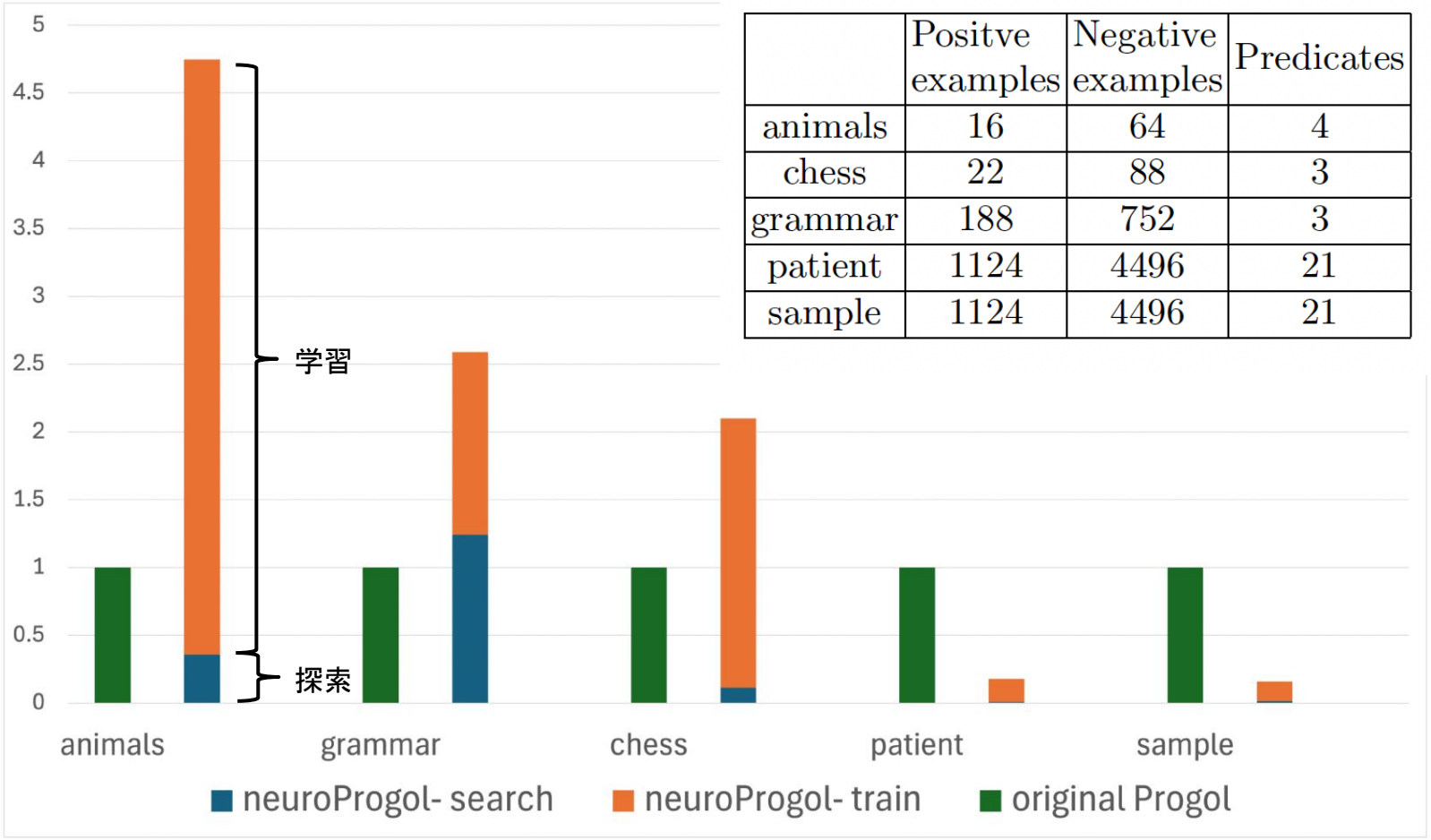
grandfather (A, B) :
- father (A, C),
mother (C, B)

grandfather (A, B) :
- father (A, C),
parent (C, B)

MSH (most specific hypothesis)

ニューロ記号処理による事例検査

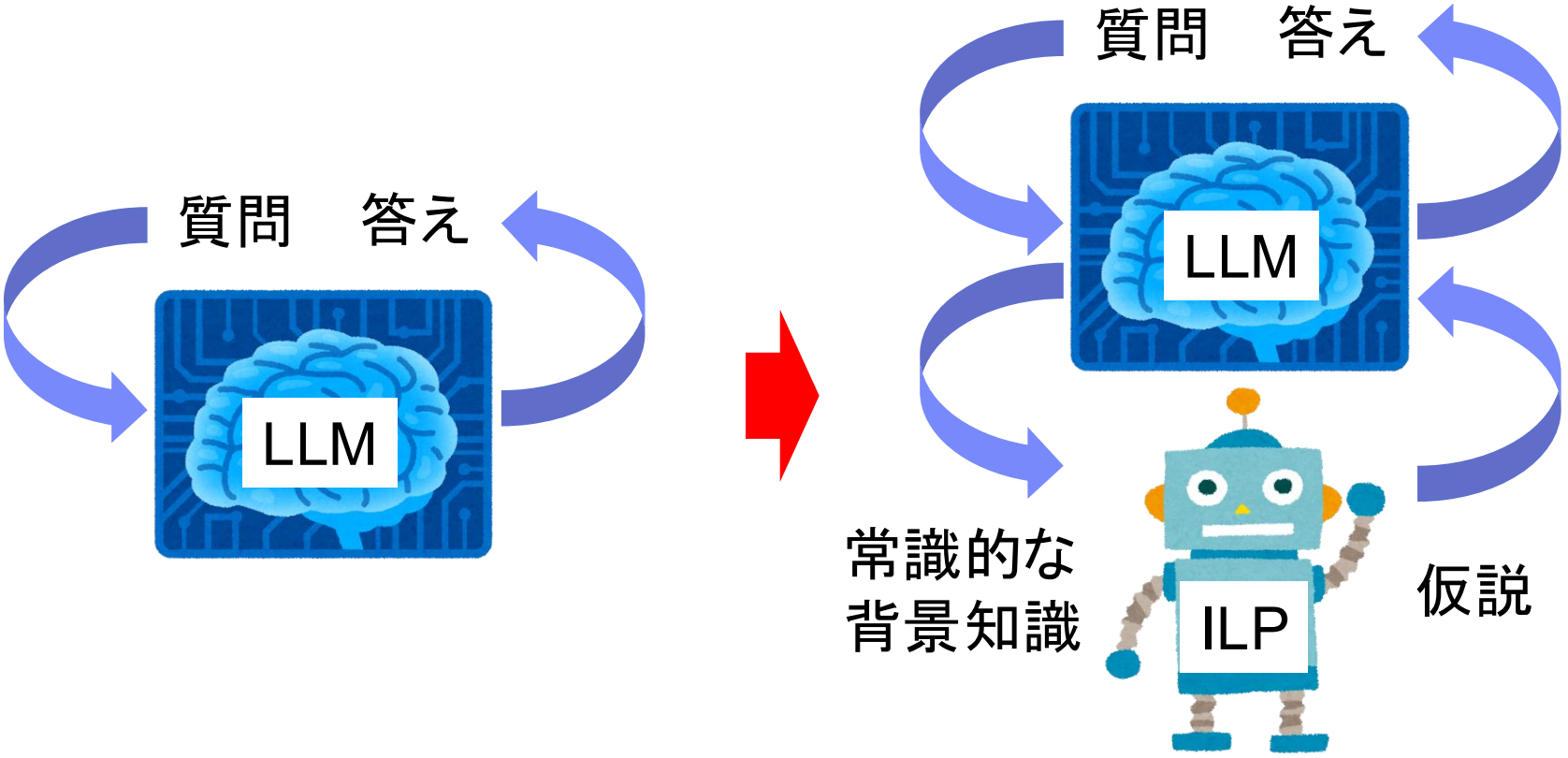
実験結果



	Positve examples	Negative examples	Predicates
animals	16	64	4
chess	22	88	3
grammar	188	752	3
patient	1124	4496	21
sample	1124	4496	21

ニューロ記号処理 ⇒ 真に考える機械へ

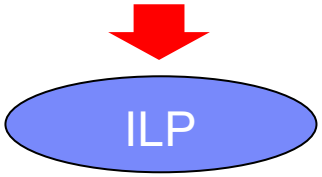
- 大規模言語モデル（LLM）＋帰納論理プログラミング（ILP）



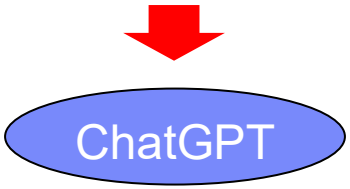
ニューロ記号処理 ⇒ 真に考える機械へ

- 大規模言語モデル（LLM）+ 帰納論理プログラミング（ILP）

乳がん患者の臨床情報と予後の情報




```
class(A, 0:Not_Recurred) :- osm(A, l0), oss(A, 0:LIVING), threegene(A, ER+/HER2-_High_Prolif).
class(A, 1:Recurred) :- hist(A, Mixed), hormone(A, YES), vital(A, Died_of_Disease).
class(A, 1:Recurred) :- claudin(A, LumB), vital(A, Died_of_Disease), subtype(A, Breast_Invasive_Ductal_Carcinoma).
class(A, 0:Not_Recurred) :- surgery(A, BREAST_CONSERVING), vital(A, Died_of_Other_Causes), pr(A, Negative).
class(A, 1:Recurred) :- radio(A, NO), er(A, Negative), vital(A, Died_of_Disease).
class(A, 1:Recurred) :- diagnosis(A, l2), threegene(A, ER+/HER2-_Low_Prolif), vital(A, Died_of_Disease).
class(A, 0:Not_Recurred) :- oss(A, 0:LIVING), claudin(A, Normal), laterality(A, Right).
class(A, 0:Not_Recurred) :- oss(A, 0:LIVING), rfsm(A, l1), intclust(A, lv2).
class(A, 1:Recurred) :- cellularity(A, Low), cohort(A, 1), vital(A, Died_of_Disease).
class(A, 1:Recurred) :- osm(A, l1), rfsm(A, l0), vital(A, Living).
class(A, 1:Recurred) :- hist(A, Lobular), threegene(A, ER+/HER2-_High_Prolif), vital(A, Died_of_Disease).
 …
```



1. 閉経前（Pre-menopausal）患者での再発リスク

解釈

閉経前患者では,

- 腫瘍分子型によって予後が大きく分岐
- 特に claudin-low 型では再発リスク増大

が示唆されています.

生活習慣との関連推測

閉経前乳癌は,

- 肥満
- 睡眠リズム
- ストレス
- エストロゲン代謝
- 出産歴

など生活背景と関連することが多く，ILPが「閉経状態」を強く使っている点は興味深いです.

医学的仮説

> 「閉経前患者では，生活習慣由来のホルモン環境変動が，claudin-low 系腫瘍の再発を促進する可能性」

は十分検討価値があります.

まとめ

- 論理型A.I. は安心安全なA.I. を実現
- 東京理科大学の論理型A.I.は多くの成果を出している.
- 分散型探索と被覆並列化で高速な帰納論理プログラミングが実現できる.
- 論理型A.I.で，A.I.の新たなブレイクスルーを！
- 補完し合う深層学習と論理型A.I.へ.