12. プロダクションシステム，知識表現，推論エンジン

（人工知能）

URL: https://www.kkaneko.jp/cc/mi/index.html

金子邦彦

アウトライン

1. 知識表現
2. プロダクションシステム
3. 知識表現のバリエーション
4. 推論エンジン

12-1 知識表現

知識表現

知識表現は，**コンピュータ**が**理解**し**処理**できる形式に**知識**を整理し，表現する手法

- 知識表現により，**コンピュータ**は知識に基づいた**推論**や**課題解決**を行うことができる

- **知識表現**を行うために，Python言語のような**プログラミング言語**を利用するのがふつう
- プログラミング言語を利用するので，人間も，知識表現の中身を理解，追加，修正可能になる

知識表現の有用性

- **人工知能**：知識表現は，**人工知能での判断**に役だつ．診断，予測など
- **データサイエンス**：大量のデータを活用し，**新たな知識を得たい**とき，知識表現が役に立つ

知識表現を学ぶことは，問題解決能力，プログラミングスキル，AIスキルの向上にもつながる

知識表現の方法①

- 属性と値のペア

属性	値
x	0
y	0

知識表現された知識

知識表現の方法②

- ルールベース

条件	振る舞い
もし雨ならば	傘

知識表現された知識

- 述語論理

述語名	引数
親子	(織田信長, 織田信孝)

知識表現された知識

- 確率モデル

知識	確率
P(雨である)	80 %

知識表現された知識

「属性と値のペア」を Python のプログラムに

属性	値
x	0
y	0

2 つの属性と値のペア

m = {'x': 0, 'y': 0}

Python のプログラム

コンピュータで情報を整理し、検索や操作できるようにするための効果的な方法

Python のプログラム例

```
1  m = {'x': 0, 'y': 0}
2
3  print(m['x'])
4
5  m['y'] = 100
6  print(m['y'])
```

知識表現

x についての知識を表示

y についての知識を 100 に変更し，表示

trinket

- **Trinket** は**オンライン**の **Python**、**HTML** 等の**学習サイト**
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）

- **Python の標準機能**を登載、その他、次のパッケージがインストール済み

math, matplotlib.pyplot, numpy, operator, processing, pygal, random, re, string, time, turtle, urllib.request

trinket でのプログラム実行

- **trinket** は **Python, HTML などのプログラムを書き実行できる**サイト
- https://trinket.io/python/0fd59392c8

のように、違うプログラムには違う URL が割り当てられる

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習
属性 x, y についての知識表現

ページ１３～１６

【トピックス】

- **trinket の利用**
- **知識表現，知識の照会，知識の変更**
- **確認クイズに自主的に挑戦**

① trinket の次のページを開く

https://trinket.io/python/f7b7dc2ca0

② 実行する。次のように表示されることを確認

```
m = {'x': 0, 'y': 0}

print(m['x'])

m['y'] = 100
print(m['y'])
```

Powered by

```
0
100
```

③ 確認クイズに自主的に取り組む

プログラムの1行目を修正して，
a が 100 であるという知識を追加

そして，確かに a が 100 であることを確認表示

④　**体毛がある**，**肉食である**という**知識**の知識表現を次のように考える

属性 **taimou** の値が **True**，属性 **nikusyoku** の値が **True** であることを次のように書くことができる

m = {'taimou': True, 'nikusyoku': True}

属性 **akarui** の値が **True**，属性 **hima** の値が **True** であることを，**同じ書き方でプログラムとして**書きなさい

そして，たしかに **akarui** や **hima** の値が True であることをプログラムで表示しなさい，以上のことを自主的に行う

解答例

③

```python
m = {'x': 0, 'y': 0, 'a': 100}

print(m['a'])

m['y'] = 100
print(m['y'])
```

```
Powered
100
100
```

④

```python
a = {'taimou': True, 'nikusyoku': True}

print(a['taimou'])
print(a['nikusyoku'])

b = {'akarui': True, 'hima': True}

print(b['akarui'])
print(b['hima'])
```

```
Powered by
True
True
True
True
```

まとめ

- **知識表現**は，**コンピュータ**が**理解**し**処理**できる形式に**知識**を整理し，表現する手法
- 知識表現により，**コンピュータ**は知識に基づいた**推論**や**課題解決**を行うことができる
- **<u>属性と値のペアを用いた知識表現</u>**

  事実や条件についての知識を扱うための有用な方法

「**体毛がある**」「**肉食である**」といった**知識**は、「**taimou‘: True, ’nikusyoku‘: True**」のように**Pythonプログラミング言語で扱うことが可能**

- 知識表現を学ぶことは，プログラミングスキルやAIスキルの向上にもつながる

12-2 プロダクションシステム

プロダクションシステム

- **プロダクションシステム**は，「**もし～ならば～**」**形式**の**ルール**（**プロダクションルール**）を用いて**知識を表現**し，**問題解決**を行う．

条件	振る舞い
もし雨ならば	傘

- 特定の**条件が満たされた**場合，ルールに従って，**明確な結果を導き出す**ことができる．
- 複雑な問題でも，ルールを１つずつ適用することで論理的に解決可能．

プロダクションルール

もし、**ある生物が体毛を持つならば**、**その生物は哺乳類**である。

→　**もし**，属性「**体毛**」の値が**True ならば**，属性「**種類**」の値が「**哺乳類**」

体毛の有無を確認することで，その生物が哺乳類であるかを判断できる可能性あり

プロダクションシステムの特徴

- **既存の情報（事実や条件）**と**プロダクションルール**を**組み合わせる**ことで、**新たな事実や知識を導き出す**ことが可能である

- 新たな情報の発見や、知識の深化につながる

プロダクションルールの適用による知識の増加①

もし **‘体毛’ = ‘ある’** ならば [**‘種類’**, **‘哺乳類’**]

プロダクションルール

[体毛, ある]
[肉食, する]

既存の情報

追加

[体毛, ある]
[種類, **哺乳類**]
[肉食, する]

最新の情報

プロダクションルールの適用による知識の増加②

もし **‘体毛’ = ‘ある’** ならば [**‘種類’, ‘哺乳類’**]
もし **‘種類’ = ‘哺乳類’ and ‘肉食’ = ‘する’** ならば [**‘種類’, ‘肉食動物’**]

２つのプロダクションルール

[体毛, ある]
[肉食, する]

既存の情報

追加 →

[体毛, ある]
[種類, **哺乳類**]
[肉食, する]

最新の情報

追加 →

[体毛, ある]
[種類, **肉食動物**]
[肉食, する]

さらに最新の情報

ルールが複数あり，**知識の追加や変化が**，**連続する**

ここまでのまとめ

- **プロダクションシステム**は，「**もし～ならば～**」**形式**の**ルール**（**プロダクションルール**）を用いて**知識を表現**し，**問題解決**を行う．
- 特定の**条件が満たされた**場合，ルールに従って，**明確な結果を導き出す**ことができる．

（例）「**もし体毛があるなら哺乳類**」という**ルール**で、**生物の種類を特定**。

- **既存の情報（事実や条件）**と**プロダクションルール**を**組み合わせる**ことで、**新たな事実や知識を導き出す**ことが可能である

演習

２つのプロダクションルール

もし **‘明るさ’ = ‘明るい’** ならば **[‘昼か夜か’, ‘昼’]**

もし **‘昼か夜か’ = ‘昼’** and **‘暇か’ = ‘はい’**

ならば **[‘行動’, ‘外出’]**

次を確認しなさい

スタート

[‘明るさ’, ‘明るい’]
[‘暇か’, ‘はい’]

→

[‘明るさ’, ‘明るい’]
[‘暇か’, ‘はい’]
[‘**昼か夜か’, ‘昼**’]

→

[‘明るさ’, ‘明るい’]
[‘暇か’, ‘はい’]
[‘**昼か夜か’, ‘昼**’]
[‘**行動’, ‘外出**’]

既存の情報

最新の情報

さらに最新の情報

演習
プロダクションシステムを動かす

ページ２７～２８

【トピックス】

- **trinket の利用**
- **知識表現**
- **プロダクションシステム**
- **プロダクションルールを適用することによる知識の増加**

プロダクションシステムのプログラム

```
m = {'taimou': True, 'nikusyoku': True}

def rule(m):
    changed = False
    if m['taimou'] == True:
        m['syurui'] = 'honyurui'
        changed = True
    if m['syurui'] == 'honyurui' and m['nikusyoku'] == True:
        m['syurui'] = 'nikusyokudoubutsu'
        changed = True

print('---start---')
print(m)

while(True):
    if rule(m) != True:
        break

print('---end---')
print(m)
```

知識

プロダクションルール

知識の確認表示

ルールを繰り返し当てはめ

知識の確認表示

① trinket の次のページを開く

https://trinket.io/python/a08b56265f

② 実行する。次のように表示されることを確認

main.py

```
1  m = {'taimou': True, 'nikusyoku': True}
2
3  def rule(m):
4      changed = False
5      if m['taimou'] == True:
6          m['syurui'] = 'honyurui'
7          changed = True
8      if m['syurui'] == 'honyurui' and m['nikusyoku'] == True:
9          m['syurui'] = 'nikusyokudoubutsu'
10         changed = True
11
12 print('---start---')
13 print(m)
14
15 while(True):
16     if rule(m) != True:
17         break
18
19 print('---end---')
20 print(m)
```

Result　Instructions

Powered by trinket

```
---start---
{'taimou': True, 'nikusyoku': True}
---end---
{'taimou': True, 'nikusyoku': True, 'syurui': 'nikusyokudoubutsu'}
```

③自主的な活動．知識（１行目），ルール（５行目から１０行目）を変更したり，増やしてみる．

失敗を恐れず，失敗からも学ぶ

12-3 知識表現のバリエーション

知識表現のバリエーション

ichiro, jiro, saburo の 3人について,

属性 **has** を扱う

	属性	値
ichiro	**has**	**ball**
jiro	**has**	**none**
saburo	**has**	**none**

3 つの属性と値のペア

それぞれに,
ichiro, jiro, saburo
という付加情報を
付けている

知識を表現する Python のプログラム

	属性	値
ichiro	has	ball
jiro	has	none
saburo	has	none

```
m = {'ichiro': {'has': 'ball'},
     'jiro':   {'has': 'none'},
     'saburo': {'has': 'none'}}
```

Python のプログラム

コンピュータで情報を整理し、検索や操作できるようにするための効果的な方法

知識の変化

変化

```
m = {'ichiro': {'has': 'ball'},
     'jiro':   {'has': 'none'},
     'saburo': {'has': 'none'}}
```

```
m = {'ichiro': {'has': 'none'},
     'jiro':   {'has': 'ball'},
     'saburo': {'has': 'none'}}
```

ichiro が ball を持っている.
他の人はもっていない
（何も持っていないことを示す「**none**」を使用）

jiro が ball を持っている.
他の人はもっていない
（何も持っていないことを示す「**none**」を使用）

変化前の知識

変化後の知識

知識の変化を定めるルール

既存の知識をもとに，知識を変化させるルール

もし「ichiro が ball を持っている」のならば

→ ichiro は jiro に ball を渡す

```
if m['ichiro']['has'] == 'ball':
    m['ichiro']['has'] = 'none'
    m['jiro']['has'] = 'ball'
```

Python 言語で書いたルール

演習
知識表現のバリエーションとルールによる知識のへkな

ページ３５，３６

【トピックス】

- **trinket の利用**
- **知識表現，知識の照会，知識の変更**
- **確認クイズに自主的に挑戦**

知識表現とルールのプログラム

```
m = {'ichiro': {'has': 'ball'},
     'jiro':   {'has': 'none'},
     'saburo': {'has': 'none'}}

def rule(m):
    changed = False
    if m['ichiro']['has'] == 'ball':
        m['ichiro']['has'] = 'none'
        m['jiro']['has'] = 'ball'
        changed = True
    return changed

print('----start----')
print(m)

while(True):
    if rule(m) != True:
        break

print('----finished----')
print(m)
```

知識

ルール

知識の確認表示

ルールを繰り返し当てはめ

知識の確認表示

① trinket の次のページを開く

https://trinket.io/python/c47a9c3b2e

② 実行する。次のように表示されることを確認

```
1  m = {'ichiro': {'has': 'ball'},
2       'jiro':   {'has': 'none'},
3       'saburo': {'has': 'none'}}
4
5  def rule(m):
6      changed = False
7      if m['ichiro']['has'] == 'ball':
8          m['ichiro']['has'] = 'none'
9          m['jiro']['has'] = 'ball'
10         changed = True
11     return changed
12
13 print('----start----')
14 print(m)
15
16 while(True):
17     if rule(m) != True:
18         break
19
20 print('----finished----')
21 print(m)
```

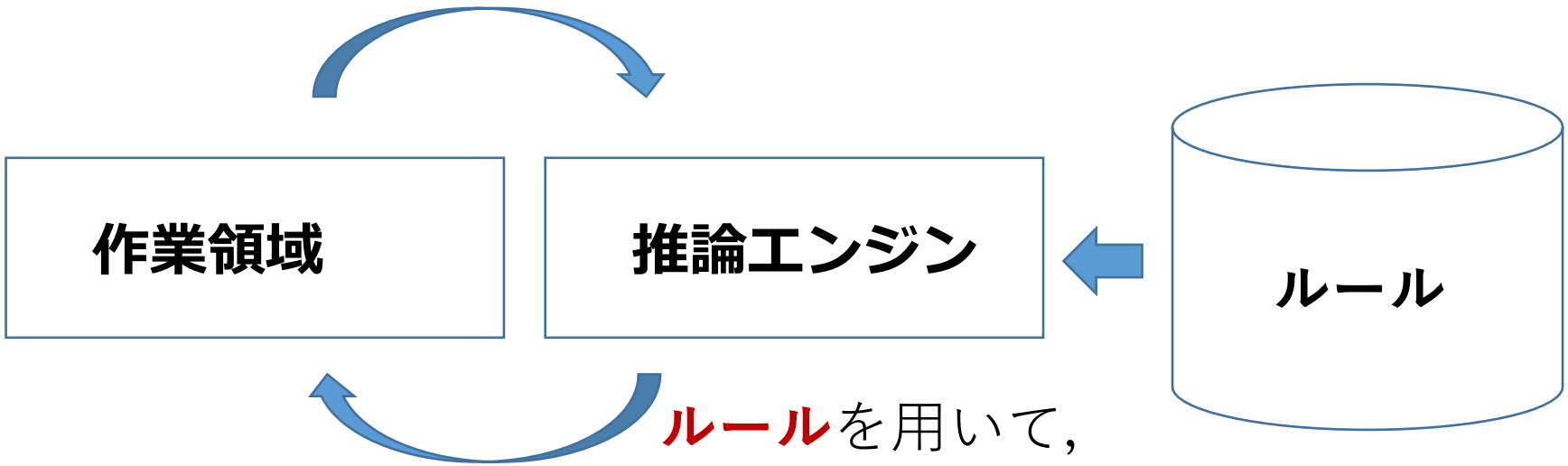
```
Powered by trinket
----start----
{'ichiro': {'has': 'ball'}, 'jiro': {'has': 'none'}, 'saburo': {'has': 'none'}}
----finished----
{'ichiro': {'has': 'none'}, 'jiro': {'has': 'ball'}, 'saburo': {'has': 'none'}}
```

③自主的な活動．知識（１～３行目），ルール（７行目から１０行目）を変更してみる．

失敗を恐れず，失敗からも学ぶ

12-4 プロダクションシステムの仕組み

プロダクションシステムの仕組み



ルールを用いて,
作業領域内の知識
を**変化**させる**推論**
を行う

推論エンジンの仕組み

(1) 作業領域と**ルール**を調べる

ルールの**条件**により使える**ルール**を探す

(2) (1) で探した**条件**について，そこで定められた

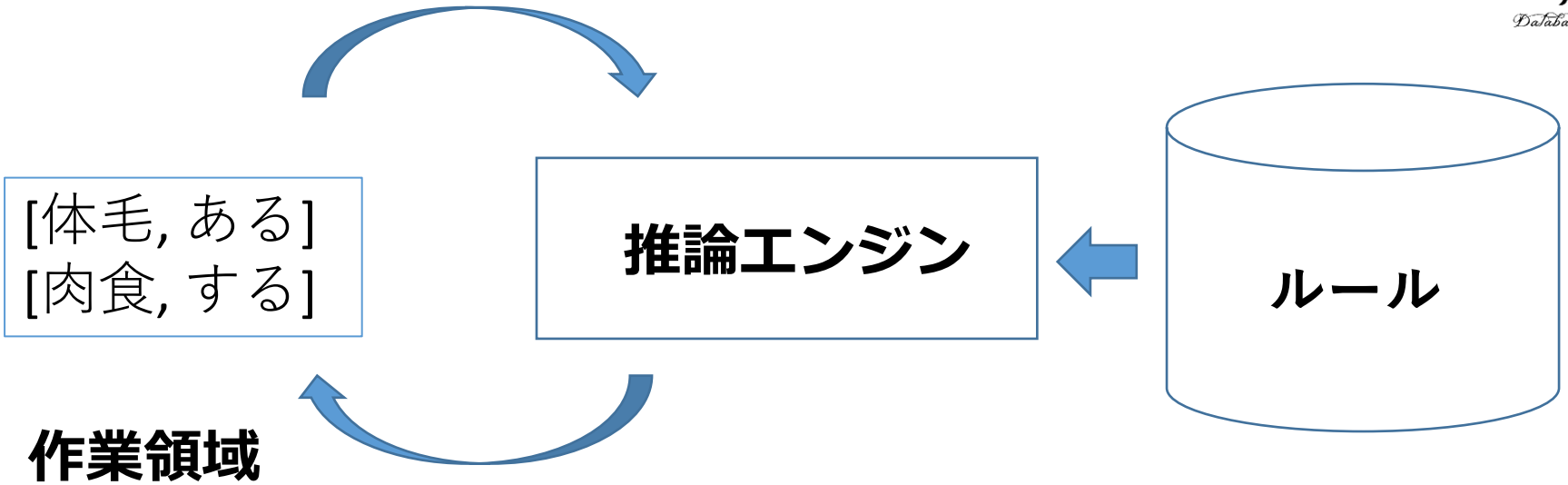
ルールの通りに，**作業領域**を書き換える

＝**推論**

(3) 以上を繰り返し，**使えるルール**が**見つからなくなったら**，終了.

結論を得る

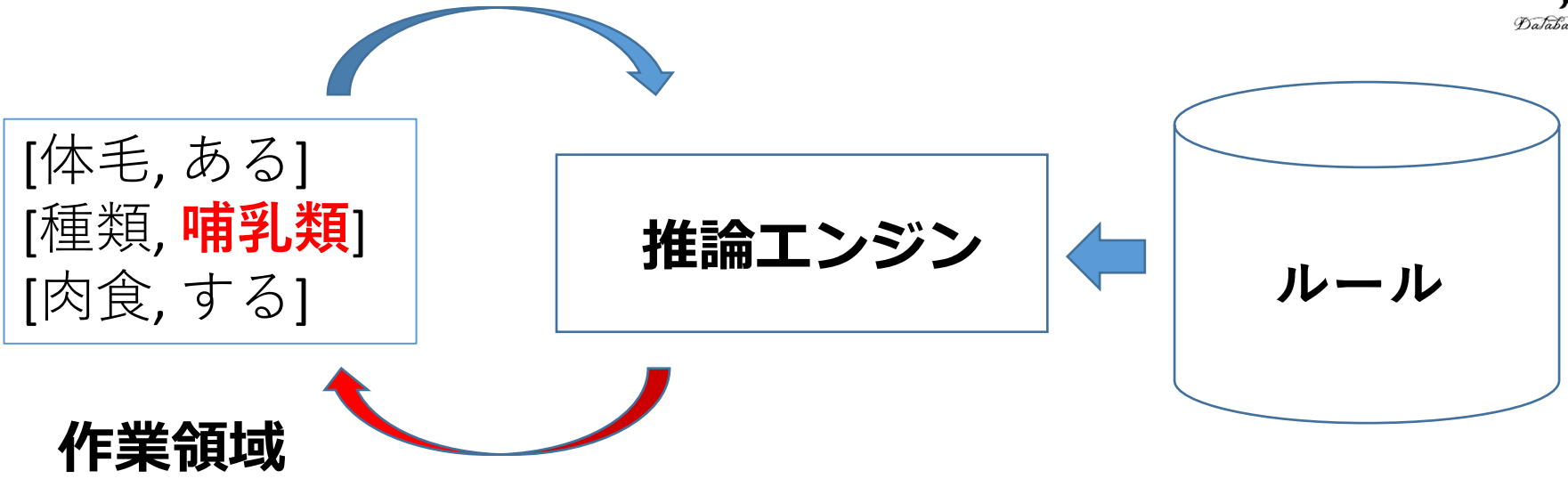
推論エンジンの仕組み



1. もし**‘体毛’ = ‘ある’** ならば→ [‘種類’, **‘哺乳類’**]
2. もし**‘種類’ = ‘哺乳類’ and ‘肉食’ = ‘する’** ならば
   → [‘種類’, **‘肉食動物’**]

ルール

推論エンジンの仕組み

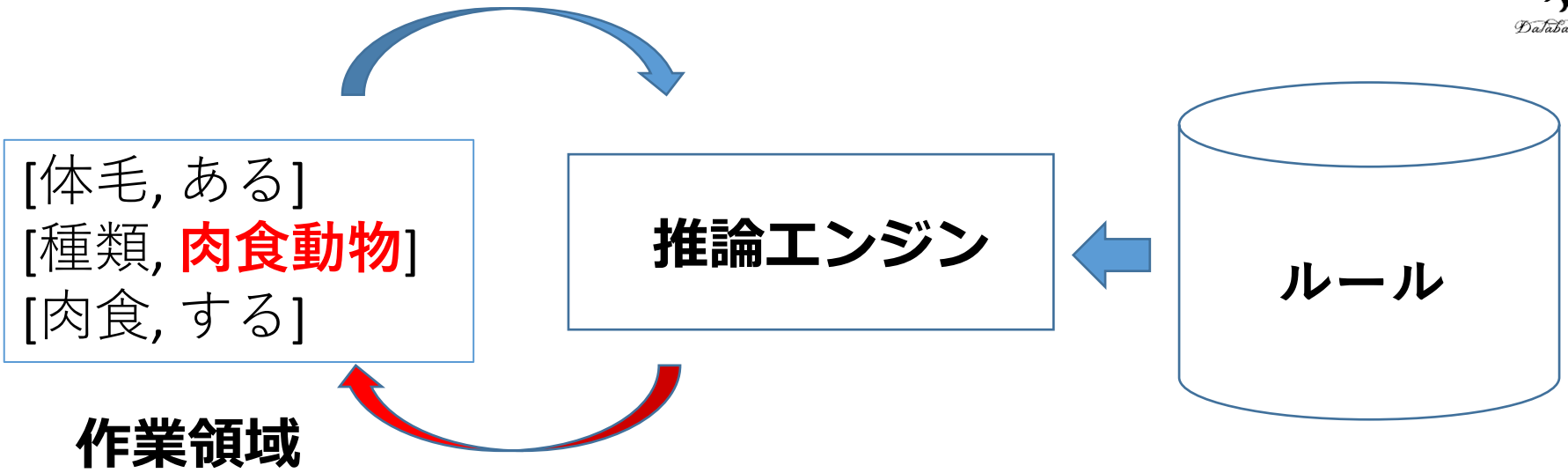


ルール１をあてはめ

1. もし**‘体毛’ = ‘ある’** ならば→ [‘種類’, **‘哺乳類’**]
2. もし**‘種類’ = ‘哺乳類’ and ‘肉食’ = ‘する’** ならば
   → [‘種類’, **‘肉食動物’**]

ルール

推論エンジンの仕組み



ルール２をあてはめ

1. もし**‘体毛’ = ‘ある’** ならば→ [‘種類’, **‘哺乳類’**]
2. もし**‘種類’ = ‘哺乳類’ and ‘肉食’ = ‘する’** ならば
   → [‘種類’, **‘肉食動物’**]

ルール

プロダクションシステムの主な機能

プロダクションシステムの主な機能

1.知識表現: 規則や事実などの**知識を適切な形式で表現**する。

2.ルールを用いた推論: **事前に設定されたルール**を使用して**知識を変化させたり増やしたりする推論を行う。**

- **ルール**は、「**もし～ならば～**」形式で表され、条件（**もし～**）が満たされた場合に実行されるアクション（**ならば～**）が定義される
- **推論エンジン**は、該当するルールに基づいてアクションを実行。

プロダクションシステムは人工知能（AI）やエキスパートシステムなどの分野で広く使用される手法の一つ

プロダクションシステム

- **作業領域の知識**

知識は、次のように書くことができる。

['体毛', 'ある']

['肉食', 'する']

- **ルール**

ルールは次のように書くことができる。

もし'体毛' = ‘ある’ならば→ ['種類', '哺乳類']

もし'種類' = '哺乳類' and '肉食' = ‘する’ならば

→ ['種類', '肉食動物']

プロダクションシステムの特質

- **ルールは複数可能**: プロダクションシステムでは、**複数のルールが存在する**ことが一般的。

- **ルールの適用順序による結果の変化**: 実は，ルールを適用する順序によって、結果が異なることがありえる。

ルール1: もし 雨が降っている ならば 傘を持つ，すぐに出かける

ルール2: もし 雨が降っている かつ 風が強い ならば コートを着る，傘は置いていく，すぐに出かける

プロダクションシステムの仕組み　まとめ

① **作業領域：**作業領域には知識を配置する。作業領域内の知識は変化する可能性がある。 知識は、次のような形式で表現できる。

['体毛', 'ある’]

['肉食', 'する']

② **推論エンジン：** 推論エンジンは、**ルール**を使用して**作業領域内の知識を変更**する**推論**を行います。

③ **ルール**：ルールは、**既存の知識**から**新しい知識を生成**したり、**知識を変更するための規則**です。 ルールは次のような形式で表現することができる。

もし'体毛' = ‘ある’**ならば**→ ['種類', '哺乳類']

もし'種類' = '哺乳類' and '肉食' = 'する'**ならば**→ ['種類', '肉食動物']

①知識表現というAI技術の理解

　知識を**コンピュータに理解させる手法**である。通常、知識表現にはプログラミング言語（例：Python）が用いられ、**人間もその内容を理解し、追加や修正が可能**である。知識表現は、**高度なAIシステムの設計や開発の基礎**となる。

②ルールベースの知識表現

　プロダクションシステムの基礎となる**ルールベースの知識表現**は、工学の分野での**問題解決**や**最適化**の手法として有用である。**データ分析**や**AIシステム設計**でも役立つ概念である。

全体まとめ

- 知識表現は、**コンピュータが理解し、処理できる形式で知識を整理**し、**表現**する手法。
- **知識表現**により**コンピュータ**は**知識に基づいた推論**や**問題解決**を行うことが可能となる。
- **知識表現**の**方法**には、「**属性と値のペア**」や「**ルールベース**」などが存在する。

  属性と値のペア　**{'x': 0, 'y': 0}**

  ルールベース」 **もし雨ならば傘**・・・条件と振る舞いの組み合わせ

- **プロダクションシステム**は、**ルール**を「**もし～ならば～**」の**形式で表現**し、**問題解決**を行うシステム。
- **推論エンジン**は、**使えるルール**を探し、**ルールに指定された振る舞い**の通りに**作業領域を書き換える。**使えるルールが見つからなくなったら**推論を終了**。