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/ai/mi/index.html

金子邦彦

BY NC SA

①人工知能の基礎概念。知識表現の重要性。プロダクションシステム。推論エンジン

② 図解と具体例による説明、理論と実践のバランス、演習、Pythonを用いた実現も紹介

③ 習得できる知識は、属性と値のペアによる知識表現、プロダクションルール、推論プロセス

④ 獲得できるスキルは、知識表現プログラミング、プロダクションルールの設計と実装

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**

例：車の特性を表現

{色: 赤, 製造年: 2022, 種類: セダン}

色、製造年、種類　・・・　属性

赤、2022、セダン　・・・　値

知識表現の方法には様々なアプローチがあります。
以下に、主要な方法とその具体例を示します

知識表現の方法②

- **ルールベース**

例：交通ルールの表現

もし 赤信号 ならば 停止する

- **述語論理**

例：家族関係の表現

親(聖徳太子, 徳川家康)

- **確率モデル**

例：天気予報の表現

P(雨が降る) = 0.7

「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

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

print(m['x'])

m['y'] = 100
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１
属性 x, y についての知識表現

ページ１３～１７

【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 次の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 https://trinket.io/python/f5da5a47cd

```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**）を用いる

もし【条件】ならば【結果】

例：もし雨ならば傘をもつ

　　もし体温が３８度以上ならば病院へ行く

- プロダクションルールにより，特定の**条件が満たされた**場合にとるべき**結果**を明確に定義できる.

演習２
単純なプロダクションシステム

ページ２１～２３

【トピックス】

- **trinket の利用**
- **知識表現**
- **プロダクションシステム**

プロダクションシステムのプログラム

trinket ▶ Run ? Modules

main.py

```python
# 簡単な天気予報システム
# 気温、湿度、気圧の知識ベースと、それに基づく3つの予測ルールを定義し、
# 条件に合致する予報を出力します。

# 知識ベース
temperature = 25 # 気温
humidity = 80 # 湿度
pressure = 1010 # 気圧

# 条件判断関数
def is_hot():
    return temperature > 30

def is_humid():
    return humidity > 70

def may_rain():
    return pressure < 1000

# ルール定義（条件関数と結論のペア）
rules = [
    (is_hot, '暑い日になります'),
    (is_humid, '蒸し暑くなります'),
    (may_rain, '雨が降る可能性があります')
]

# ルールの適用
for condition, conclusion in rules:
    if condition():
        print(conclusion)
```

知識

プロダクション
ルール

Result

Powered by trinket
蒸し暑くなります

① trinket の次のページを開く

https://trinket.io/python/fadcd5d6cd

② 実行する。次のように表示されることを確認

③１１から１８行目の条件判断を確認。その上で、６から９行目の値を「temperature = 33」などに変えて、結果の変化を見よう

プロダクションルール

プロダクションルールは、「**もし～ならば～**」**形式**のルールである

例

もし 体毛がある ならば 哺乳類である

このルールは、「**体毛がある**」という**条件**が満たされた場合、「**哺乳類である**」という結果を導き出す

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]

既存の知識

追加

[体毛, ある]
[種類, **哺乳類**]
[肉食, する]

最新の知識

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]

既存の知識

追加 →

[体毛, ある]
[種類, **哺乳類**]
[肉食, する]

最新の知識

追加 →

[体毛, ある]
[種類, **肉食動物**]
[肉食, する]

さらに最新の知識

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）**を用いる.

特定の**条件が満たされた**場合にとるべき**結果**を明確に定義

例：もし 体毛がある ならば 哺乳類である

・**既存の知識**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

[体毛, ある]
[肉食, する]

既存の知識

追加 →

[体毛, ある]
[種類, **哺乳類**]
[肉食, する]

最新の知識

追加 →

[体毛, ある]
[種類, **肉食動物**]
[肉食, する]

さらに最新の知識

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’]

既存の情報

→

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

最新の情報

→

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

さらに最新の情報

演習３
プロダクションシステム

ページ２９～３１

【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

```python
# データで動物の特徴を表現。「体毛」なら哺乳類、さらに「肉食」なら
# 肉食動物と段階的に分類。ルールを繰り返し適用し、変更がなくなるま
# で続ける。最終結果を出力。
def rule(m):
    changed = False
    if m['syurui'] == '' and m['taimou']:
        m['syurui'] = 'honyurui'
        changed = True
    elif m['syurui'] == 'honyurui' and m['nikusyoku']:
        m['syurui'] = 'nikusyokudoubutsu'
        changed = True
    return changed

print('---start---')
m = {'taimou': True, 'nikusyoku': True, 'syurui': ''}
print(m)

while rule(m):
    pass

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

プロダクション
ルール

知識

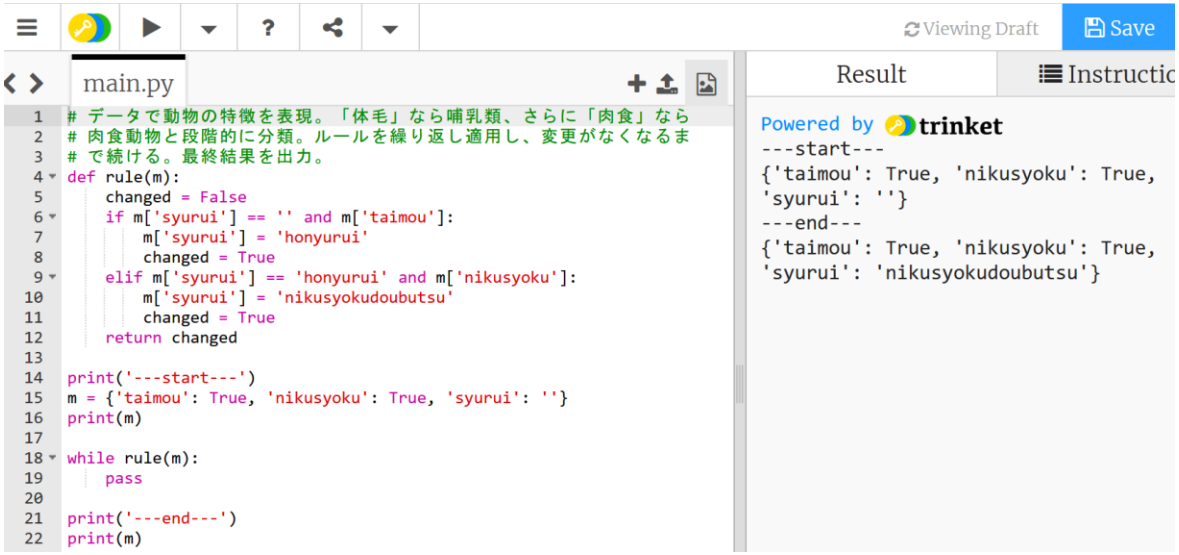
ルールを繰り返し
当てはめ

最新知識の確認表示

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



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

- **属性と値のペア**（今までの説明）

['ichiro_has', 'ball’]

['jiro_has', 'none’]

['saburo_has', 'none’]

- **より複雑な構造**（ここでの説明）

一郎、次郎、三郎ごとに分かれる

{'ichiro': {'has': 'ball'},

'jiro': {'has': 'none'},

'saburo': {'has': 'none'}}

知識表現には様々な方法がある。同じ知識を異なる形で表現できる。状況や目的に応じて適切な表現方法を選択することが重要。

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

「もし～ならば～」形式で表現

もし「ichiro が ball を持っている」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４
知識表現のバリエーションとルールによる知識の変化

ページ３８～４１

【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

③自主的な活動．知識（４～６行目），ルール（１０行目から１３行目）を変更してみる．

失敗を恐れず，失敗からも学ぶ

演習4の発展

- 具体的な変更例

1. 知識の変更：hanakoを追加

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

2. ルールの変更：hanakoへのボールの受け渡しを追加

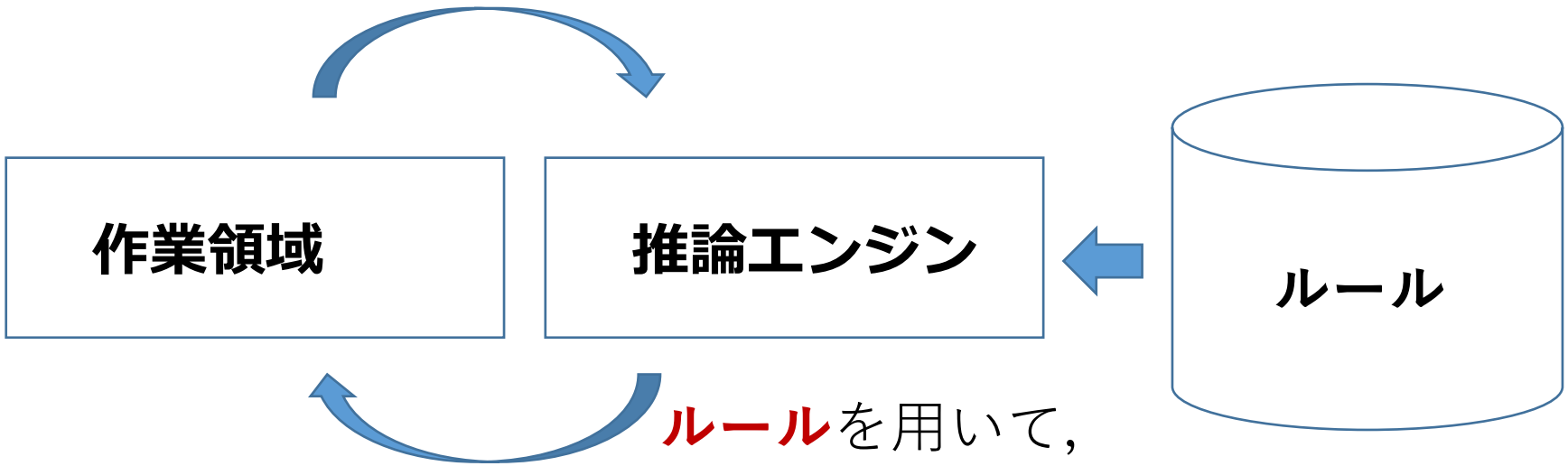
```
elif m['saburo']['has'] == 'ball':
    m['hanako']['has'], m['saburo']['has'] = m['saburo']['has'], 'none'
```

- 変更後の結果

ichiro → jiro → saburo → hanako の順でボールが移動します。

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

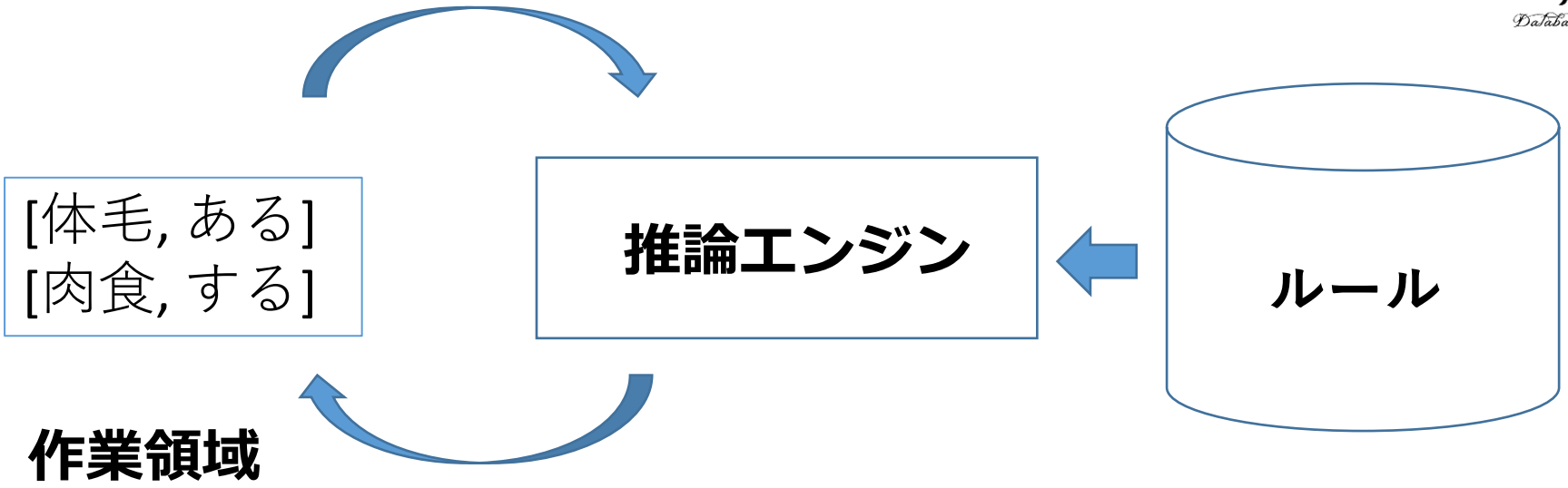
ルールの通りに，**作業領域**を書き換える

＝**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

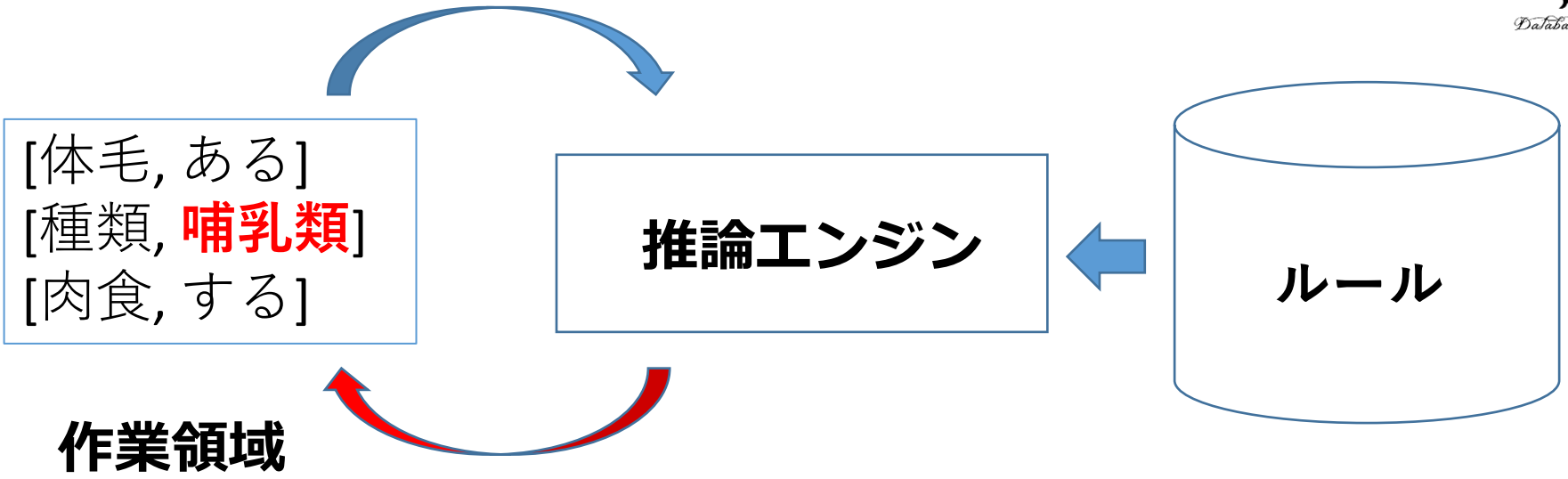
推論エンジンの仕組み



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

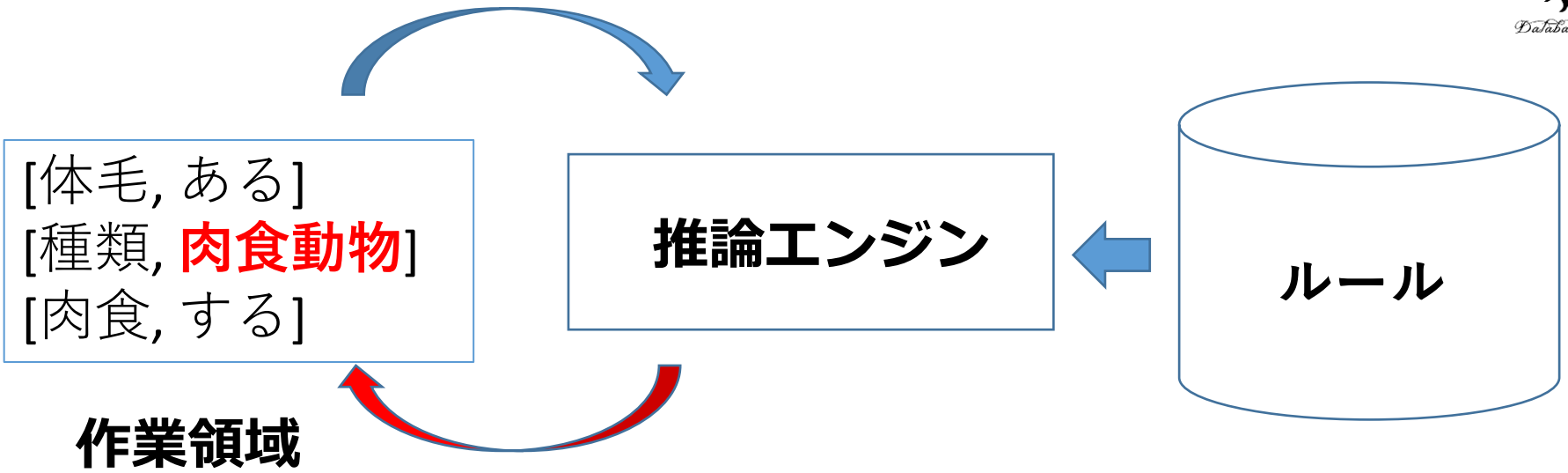


ルール１をあてはめ

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']

全体まとめ

知識表現の重要性

•知識表現は、**コンピュータが理解・処理できる形式で知識を整理**する手法

•目的：AIによる推論や問題解決を可能にする

知識表現の主要な方法

•**属性と値のペア**：例 {'x': 0, 'y': 0}

•**ルールベース**：例 もし雨ならば傘を持つ

プロダクションシステム

•「**もし～ならば～**」形式の**ルール**を用いた問題解決システム

•構成要素：作業領域、ルール、推論エンジン

•動作：**推論エンジンがルールを適用し、作業領域の知識を更新**

•AI・エキスパートシステムの基盤技術 、データ分析や工学的問題解決に活用可能

授業の学ぶ意義と満足感

①**基礎スキルの習得。**Pythonを使った実践的演習。プロダクションルールの理解。知識をコンピュータで扱う

② **AI の基本原理の理解。知識表現。プロダクションシステム。ルールベース**の問題解決手法。**推論エンジン**

③ 問題解決能力の向上

④ 未来への準備。AIの知識とスキルの向上