

# 知識工学

第三回 知識表現と問題解決

# コメント

パパパコメント. 合言葉は「知識情報工学citns」



<http://papapac.com/post.php?room=知識情報工学citns>



コメントを投稿したい人

教えてもらった部屋名を入れてね

知識情報工学citns

入室



知識情報工学citnsの部屋

コメント送信

コメントは部外者にも見られる可能性があります。個人情報などは送信しないでください。



# 前回のレビューシート

- アルゴリズムのサンプルソースを見てみたい
  - ネットから拾ってコメントを入れました.
- どのように探索法を使い分けるかが知りたい
  - ツリーの形状を想定し, 一つの階層に多くノードがあれば縦型 (横だと重い)
  - 最短経路を調べたいときには横型 (縦だと深い解を出す恐れ)
- 現実世界を模したアルゴリズムがあれば知りたい
  - 蟻コロニー最適化 <https://www.youtube.com/watch?v=2UxH8-9WJFs>
- スピーカを買ってください
  - すみません. 買いました.



# 縦型探索 (Python)

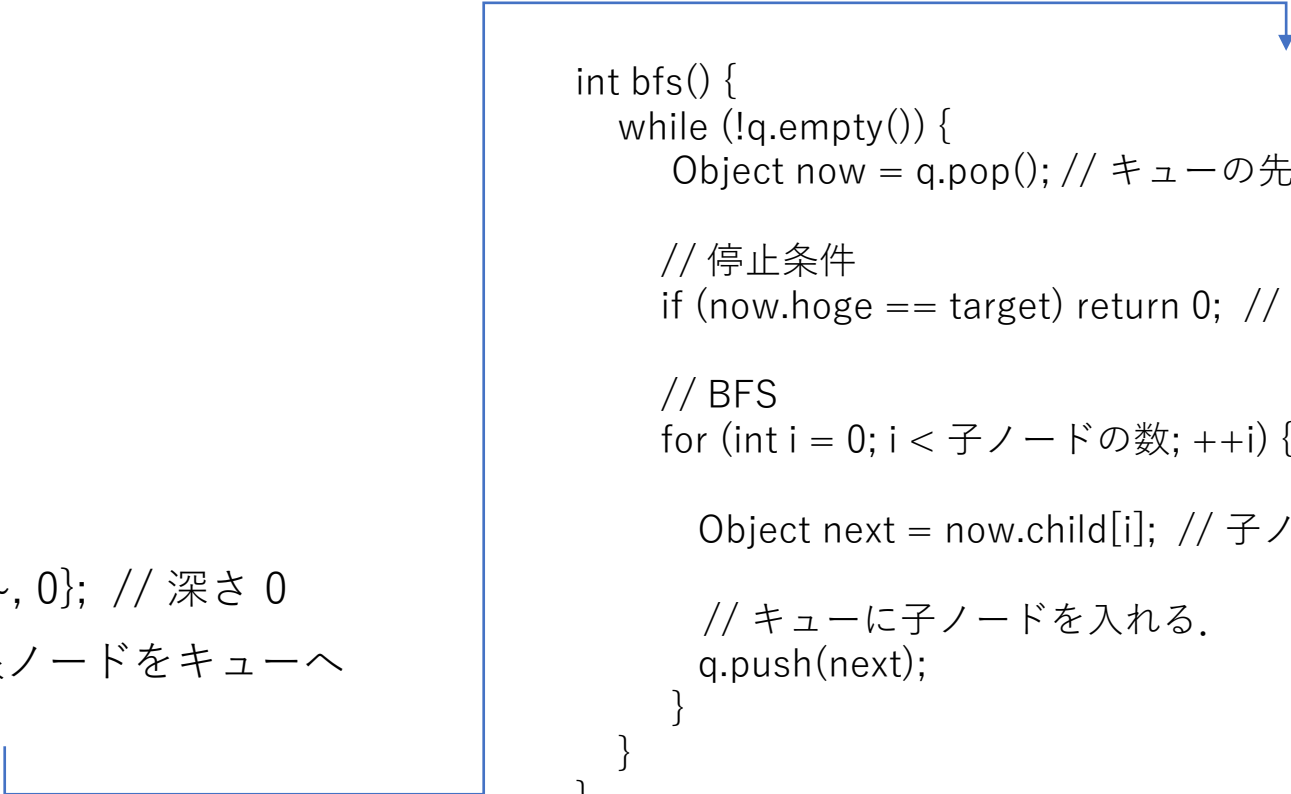
<https://ja.wikipedia.org/wiki/%E6%B7%B1%E3%81%95%E5%84%AA%E5%85%88%E6%8E%A2%E7%B4%A2#:~:text=%E6%B7%B1%E3%81%95%E5%84%AA%E5%85%88%E6%8E%A2%E7%B4%A2%EF%BC%88%E3%81%B5%E3%81%8B%E3%81%95,%E3%81%AA%E9%99%90%E3%82%8A%E6%8E%A2%E7%B4%A2%E3%82%92%E8%A1%8C%E3%81%86%E3%80%82>

```
def depthFirstSearch( start, goal ):
    stack = Stack()           # スタックを作成
    start.setVisited()         # 根ノードを探索候補とする
    stack.push( start )       # 根ノードをスタックへ入れる
    while not stack.empty():   # スタックが空になるまで処理を実行
        node = stack.top()     # スタック内の一番あとに入れたものを処理対象とする
        if node == goal:       # 目標ノードならば終了
            return stack       # stack には2頂点間の経路が入っている
        else:
            child = node.findUnvisitedChild() # 未探索の子ノードを取り出す
            if child == none:
                stack.pop()     # もし子ノードがいなかったらスタックで一番あとに入れたノード（処理対象）を消す
            else:
                child.setVisited() # 子ノードを探索候補にする
                stack.push( child ) # スタックに子ノードを入れる
```

# 横型探索 (C++)

<https://pyteyon.hatenablog.com/entry/2019/03/01/211133%E5%B9%85%E5%84%AA%E5%85%88%E6%8E%A2%E7%B4%A2>

```
struct Object {  
    // 任意の構造体  
    ll depth;  
    ...  
}  
  
queue<int> q;  
Object start = {~~~, 0}; // 深さ 0  
q.push(start); // 根ノードをキューへ
```



```
int bfs() {  
    while (!q.empty()) {  
        Object now = q.pop(); // キューの先頭を取り出す.  
  
        // 停止条件  
        if (now.hoge == target) return 0; // 探索対象ならば終了  
  
        // BFS  
        for (int i = 0; i < 子ノードの数; ++i) {  
            Object next = now.child[i]; // 子ノードを1つずつ取得 (深さを一つ足す)  
  
            // キューに子ノードを入れる.  
            q.push(next);  
        }  
    }  
}
```

面白かった、  
興味がある  
(偶数)





[illegible]

面白かった,  
興味がある  
(奇数)



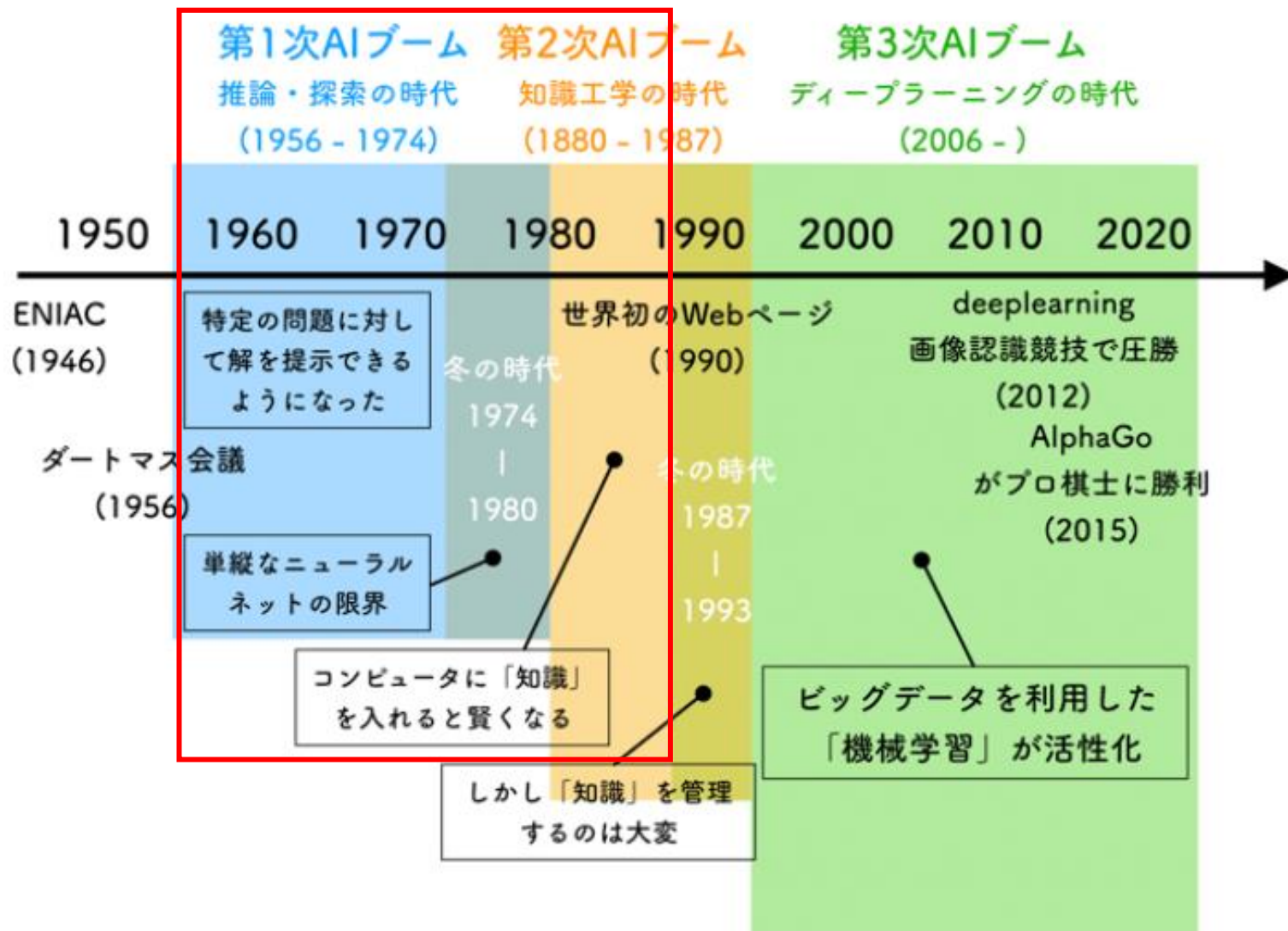


もっと知りたい  
(奇数)



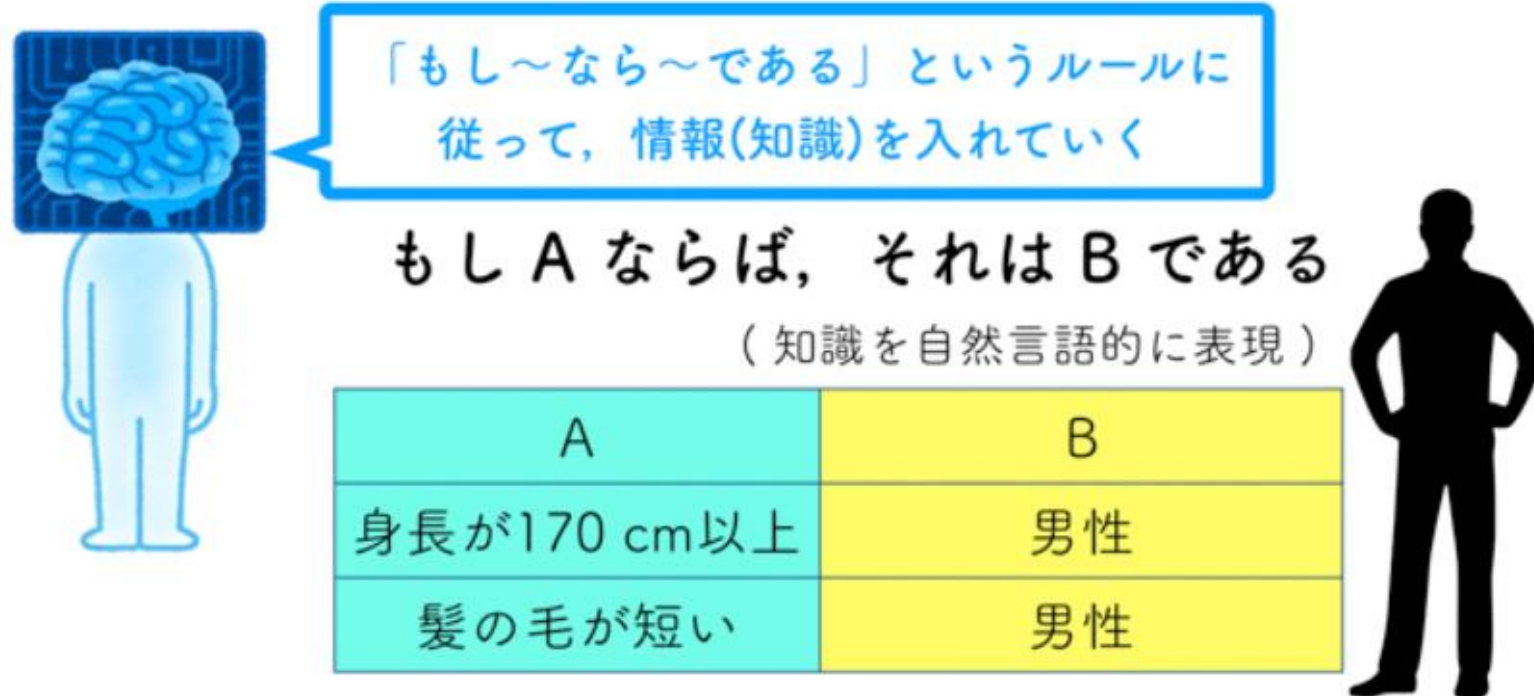
## 今日の内容

- 80年代頃の第二次AIブームと関係する
- トイ問題から方向転換し、現実問題を解くために知識工学が誕生
- 知識をコンピュータ上で表現する方法を学ぶ



# 知識工学の発生

- 前回は、トイ問題という単純化された世界を扱った
- トイ問題には批判も多く、現実世界の問題に挑む必要性が生まれた。
- しかし限界を感じ、知能志向からエキスパートシステムを用いた知識志向へと移っていく。





# 知識の分け方 (1/5)



## 専門知識

強くドメインに依存する知識。他のドメインでは使えないことがほとんど。

(ドメイン = ある専門領域)



## 常識

ドメインに依存しない抽象的な知識。無意識的に我々が使うことも多く、記述が難しい。



# 知識の分け方 (2/5)



**宣言的知識(what)：**物事に対する一般的な知識.

例) 散髪とは、伸びた髪の毛を切って整えることである.



**手続き的知識(how)：**物事に対するやり方についての知識.

例) 散髪では横髪から切り、バランスを見ながら前髪や後ろ髪を切る.  
最後に全体を調整して仕上げをし、シャンプーとブローを行う.

# 知識の分け方 (3/5)



**経験的知識**：完全に正しいという保証はないが，対象となる問題の大半で成り立つ知識．表面的，個人的，非明示的，非論理的．

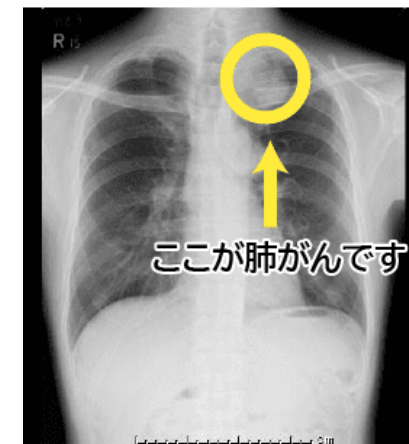
例) 単位を落としそうなとき，あの先生なら頼み込めばなんとかなる．

**理論的知識**：その問題領域にある科学的・数学的知識．

例) 飛行機が故障した際の，背景にある構造力学や流体力学等の知識．  
または，バードストライクにおいて，判明している鳥の行動習性など．



# 知識の分け方 (4/5)

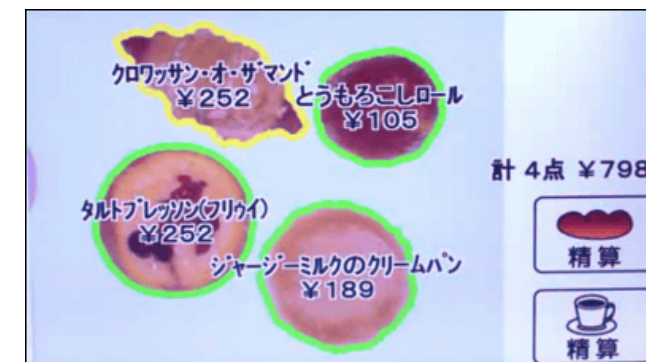


**ドメイン知識**：問題領域固有の知識.

例) 経営, 医療, 旅行などの分野ごとの膨大な知識.

**タスク知識**：問題解決に対する振る舞いの知識

例) 計画, 判定, 判断などに関する知識.

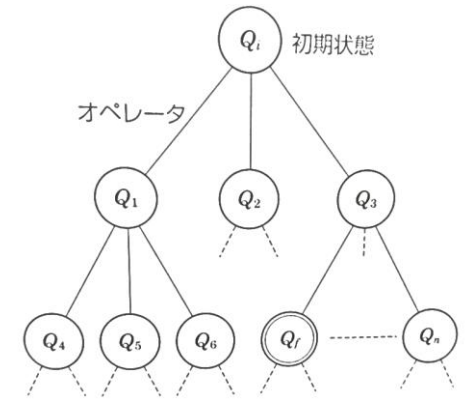


自動ガン画像診断システムでは, ドメインが医療, タスクが判断 (診断)

ドメイン知識) ガンならば, 肺のとある部分に濃い影がある

タスク知識) 判断するアルゴリズムの仕組み

# 知識の分け方 (5/5)



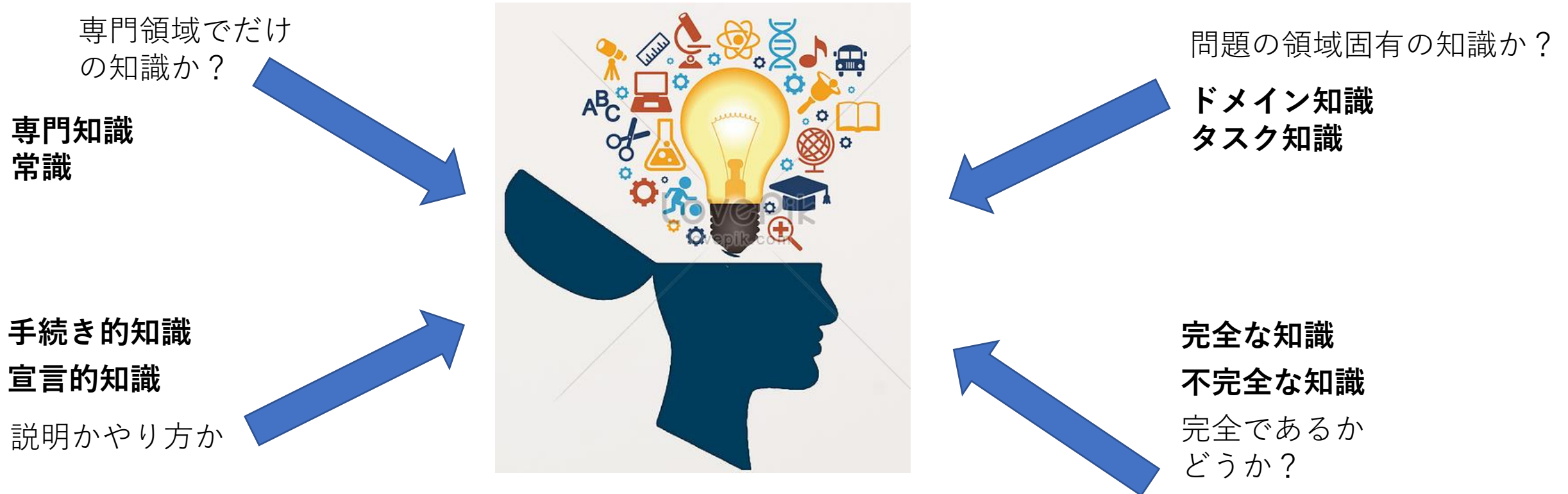
- **完全な知識**：常に正しい知識. または対象に関する全ての記述を持つ知識.
  - 例) オセロや将棋は, 全ての盤面を漏れなく表現できる.
- **不完全な知識**：常に正しいとは限らない知識. 不完全, 仮説的.
  - 例) ガンになる原因は複数あるが, 未だ完全に解明されていない.





# 知識の分け方

何種類もの知識があるわけではなく，人が持つ総合的な知識をある角度から見て，その見方から分割していることに注意．





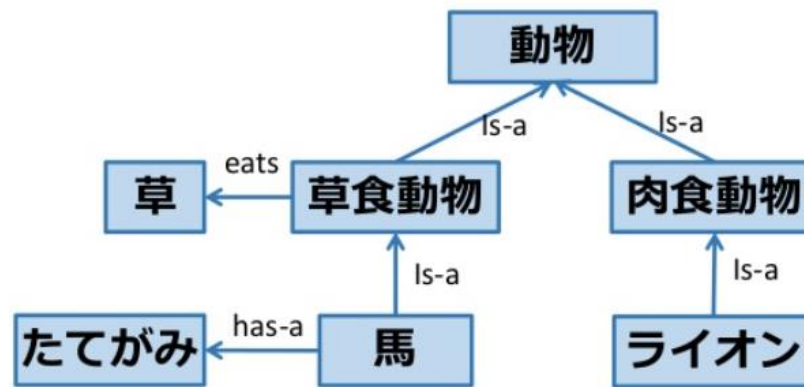
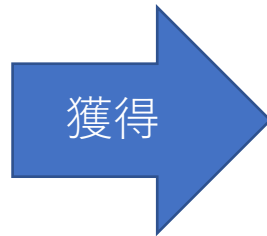
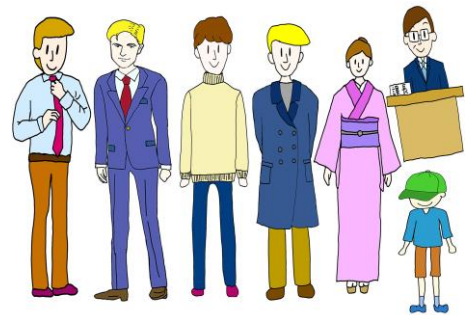
# 知識の分け方（ワーク）

---

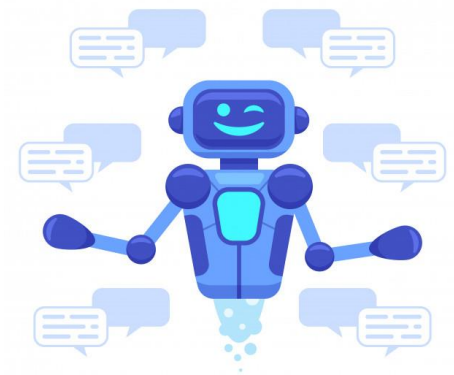
- この他には、どのような分け方があると思いますか？
  - 専門領域だけの知識か？
  - 問題の領域固有の知識か？
  - 説明かやり方か？
  - 完全であるかどうか？
- 良くない例
  - 黒板の知識とそれ以外の知識
  - 理由：チョークの知識，プロジェクターの知識など対象物は無数にあるのに，なぜ黒板だけを選択したのかが説明できない。

# 知識処理の3フェーズ

1. 知識獲得：必要とする知識の取得.
2. 知識表現：形式的表現. セマンティックネットワーク, 形式論理など
3. 知識利用：表現された知識を用いて問題を解決. 推論.



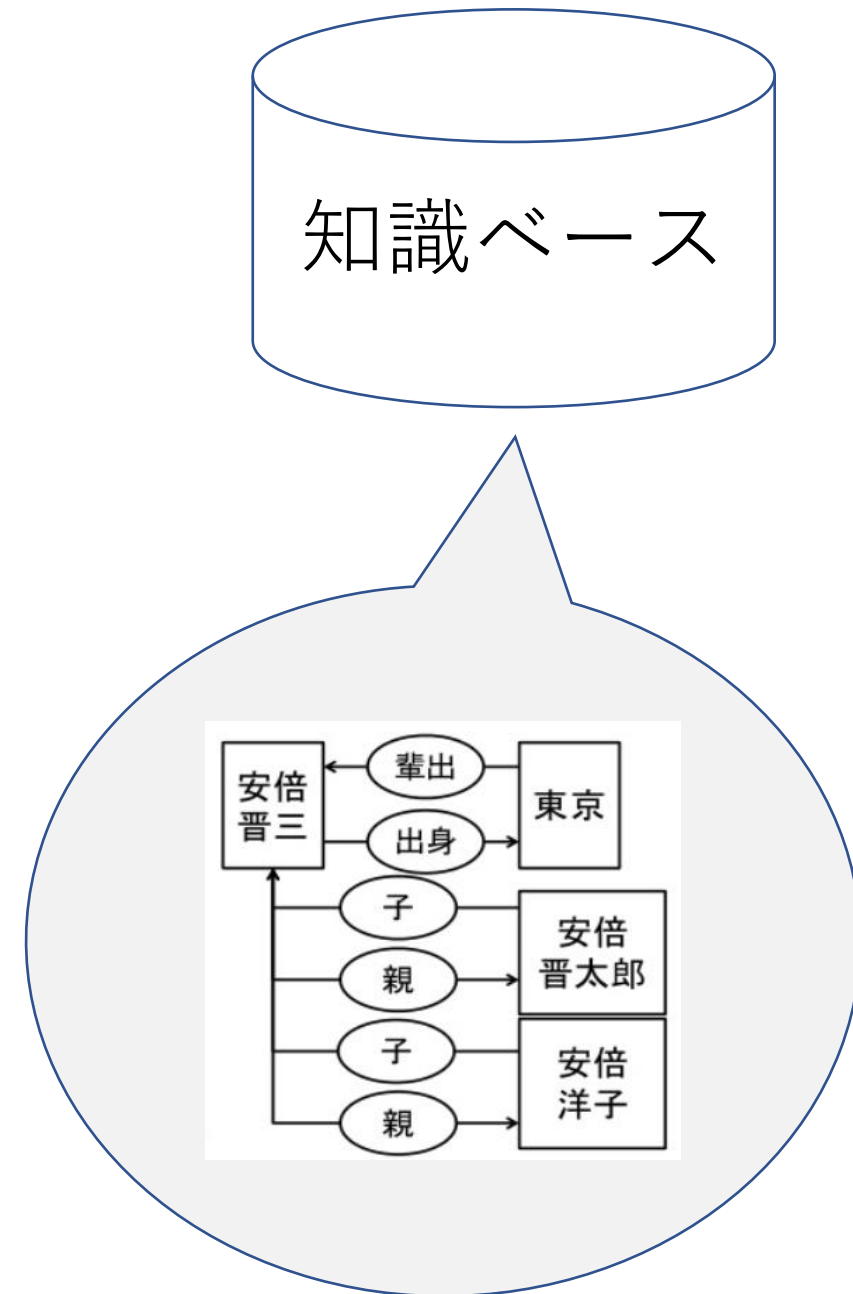
知識表現



# 知識ベースとは？

- 知識を溜め込んだ集合（ベース＝基地）
- 知識に特化したデータベース

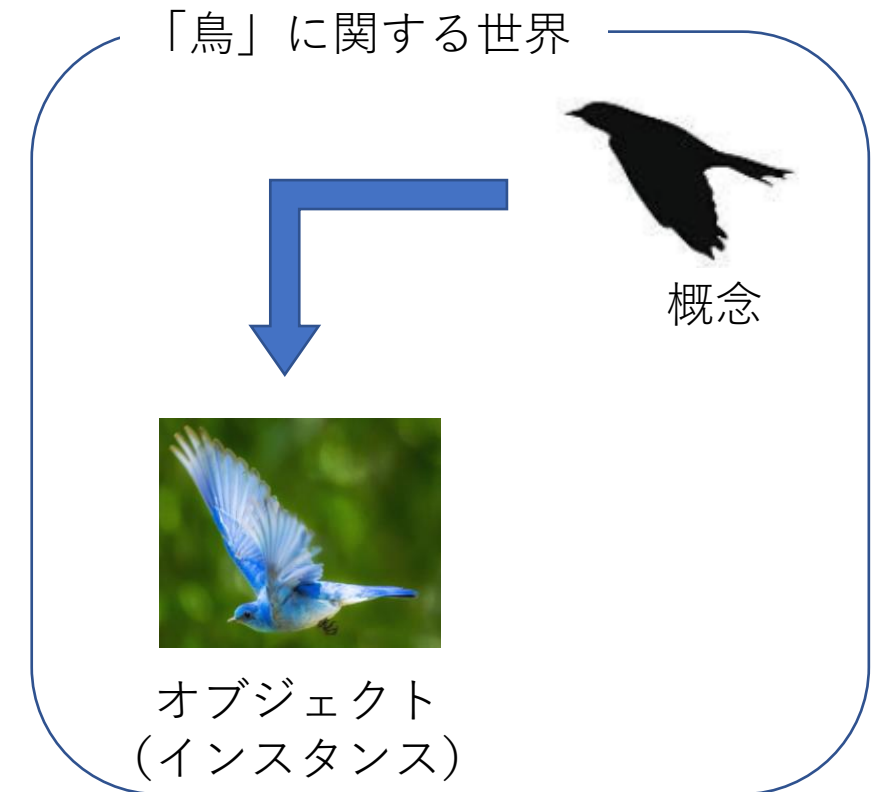
例）言語的タスクで、右の図に示される知識がインプットされていた場合、  
（安倍晋三，出身，？）＝東京  
のような問い合わせができる





# 知識ベースに必要なもの

- 世界に関連する概念の記述（プログラミングのクラスに近い）
  - 鳥は羽を持つ
- 世界に存在するオブジェクトとその属性
  - ピーちゃんは青い羽を持つ
- オブジェクト間の関係
  - ピーちゃんの子供はピッピである
- 概念間の関係
  - カラスは鳥に含まれる



# 知識ベースの長所と短所

## 長所

- 一貫性：知識ベースが不変なら同じ解が返される.
- 永続性：コピーができるため，知識ベースの複製が可能.

## 短所

- そのドメイン(部分領域)でのタスクにしか対応できない.
- 融通性や適応性に欠け，汎用ではない.

休憩

# 知識の活用

世界に関する知識をどのように活用すればよいかは、別の知識が必要になる。

- **メタ知識**：知識に関する知識。知識をどう使えばよいかなど



鳥の種類を特定するには、  
○○, △△, ××  
が分かればいいな。



# 知識表現の方法：プロダクシヨンルール

ある事実から導出される結論というような推論知識や，ある条件が成立したときに行われる行動に関する知識を記述できる．

一般に「If A then B」や「 $A \rightarrow B$ 」といった形式で記述される．

例) if ( <X> is a bird) and ( <X> cannot fly)  
then (<X> is a penguin)

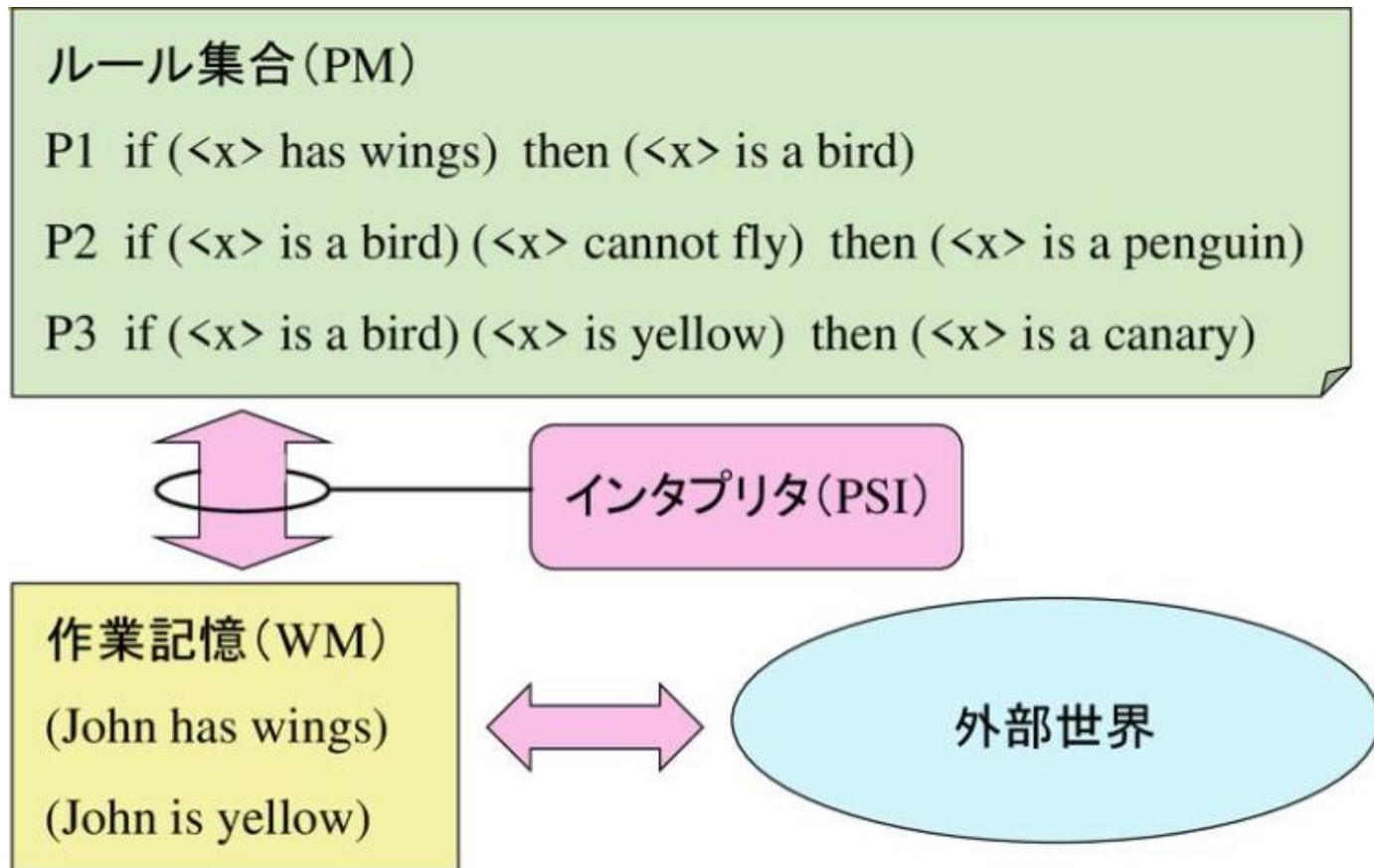
**前方推論**：Aを用いてBを推論する

**後方推論**：BからAを推論する

# プロダクション ルールの構成

---

- ルール集合  
(知識ベース)
- ワーキングメモリ  
(データベース)
- インタプリタ  
(推論エンジン)



# インタプリタの役割

## 1. マッチング

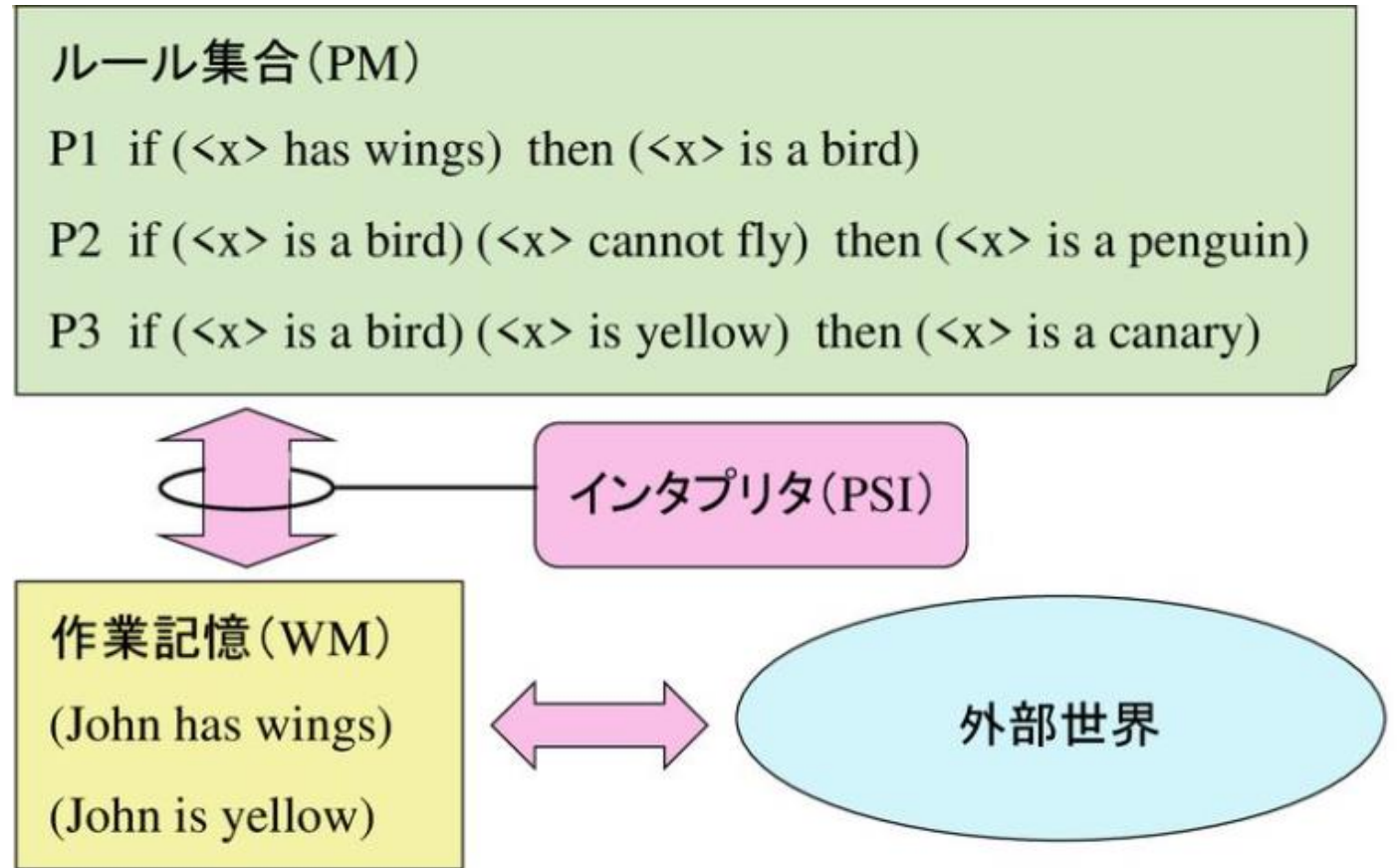
PMとWMを照らし合わせ、PSIが適用可能なルールの集合を作る

## 2. 競合解消

優先度の高いルールから採用する

## 3. 実行

ルールに従って、WMを書き換える



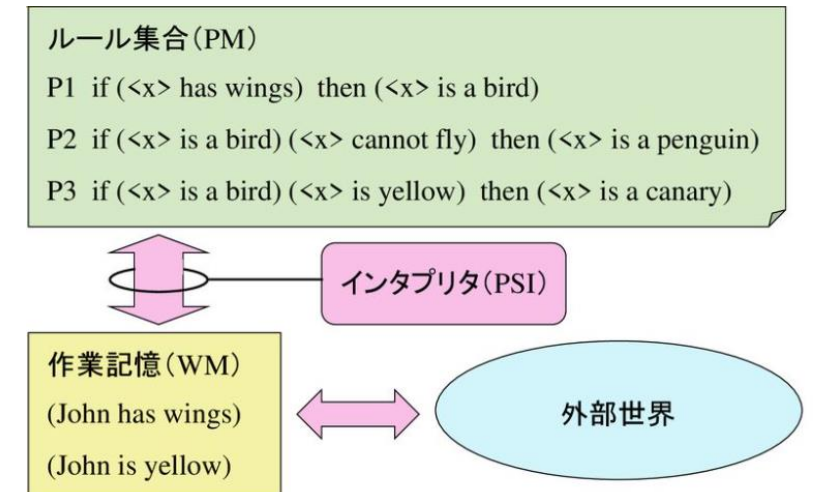
# プロダクションシステムの長所と短所

## 長所

- システム構造が明確
- 可読性，拡張性が高い
- 結果が出てくる過程を説明しやすい

## 短所

- 一貫性が低く，知識の使い回しは難しい
- ルールの相互作用が不明確



## エキスパートシステムの解説動画（おまけ前まで）

---

マイシンは69%の確率で正しい処方をすることができました。これは感染症の専門医が正しい処方をする確率80%よりも低い水準ですが、専門医でない医師よりはよい結果でした。



一般の医師



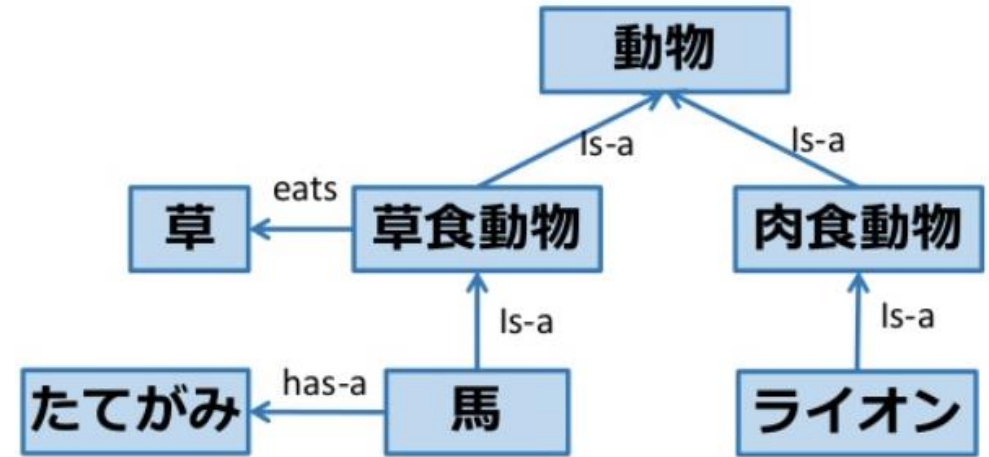
< MYCIN <



感染症専門医

# 知識表現の方法： セマンティック（意味） ネットワーク

- 機械と自然言語の中間のための言語として開発された.
- 概念の上下関係を示す事が可能.
- 例えば, 「馬は動物か?」を問い合わせると, 馬の上方を探索して動物に達するため, 「Yes」と返す.





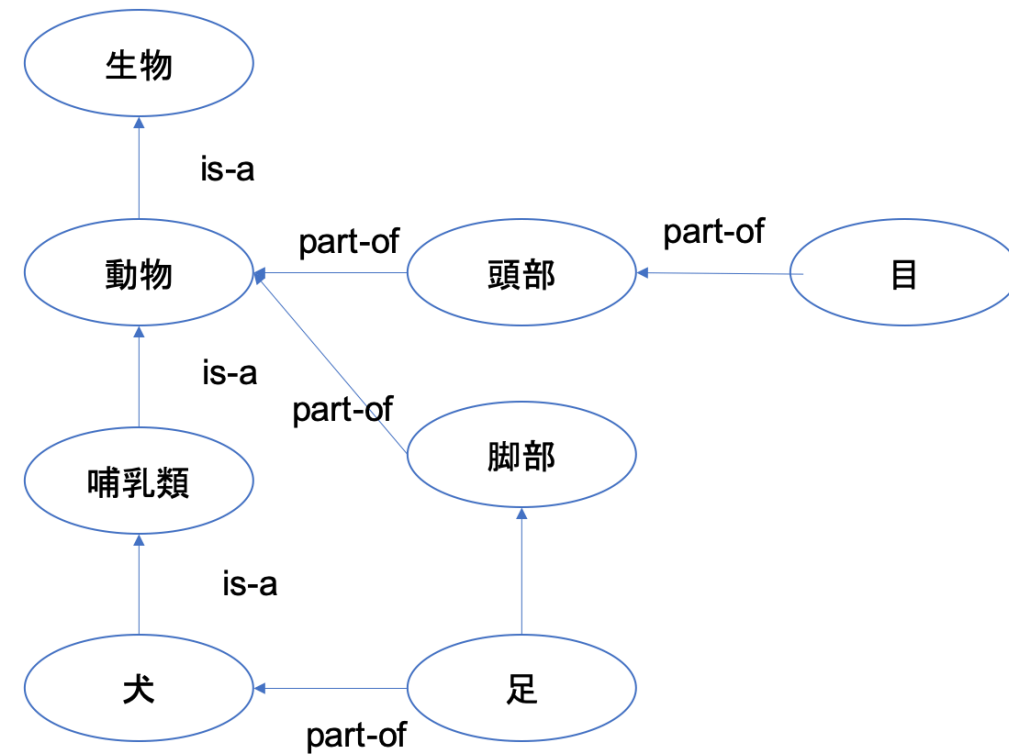
# 関係性の種類

**Is-a**：概念の包含関係(～の一種)

**Has-a/Part-of**：具体的な部分と全体

上位の方に向けてリンクを貼る

その他にも，目的に合わせて多様な種類のリンクを貼れる



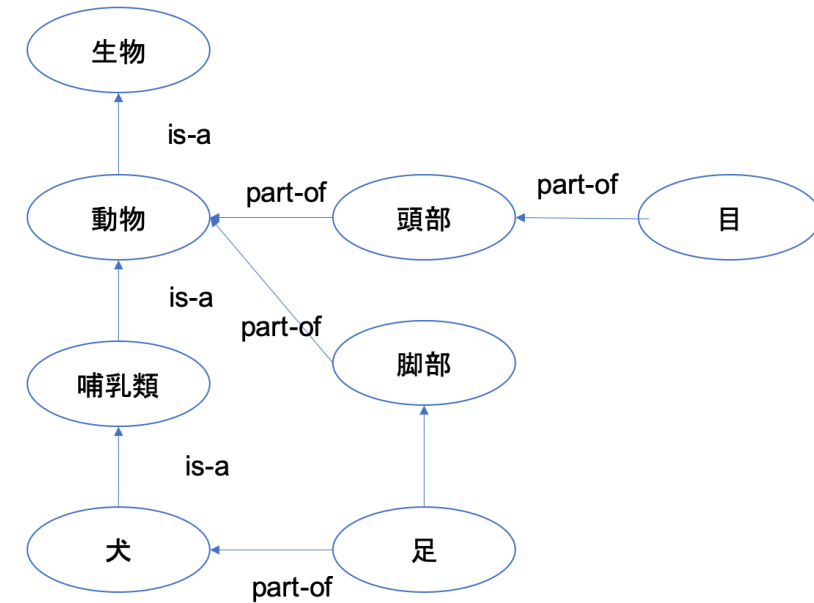
# セマンティックネットワークの長所と短所

## 長所

- 表現がシンプルで可読性が高い
- 連想がある程度可能である
- 属性継承が可能である

## 短所

- インタプリタを個別に作成しなくてはならない
- 推論の妥当性が保証されない
- 大規模になると処理効率が低下する



# 知識表現の方法：フレームシステム

- ある事柄を表す枠組み（フレーム）を，「事柄の名称」，「事柄の属性」，「事柄の属性の値」で表現したもの。
- ぴーちゃんの足の数を問い合わせると，親クラスの情報を見に行く
- **デーモン**：体重が必要になったら計算を行うなど，トリガーの役割

鳥	クラス	
IS-A	value	動物
Has-A	default	羽
足の数	default	2
体重	If-Needed	calc-weight



ぴーちゃん	インスタンス	
IS-A	value	鳥
Has-A	value	くちばし
子供	value	ピッピ
誕生日	value	2020/1/4
	If-added	calc-age

# フレームシステムの長所と短所

## 長所

- 知識を構造化できる
- 手続きを付加することで柔軟な振る舞いができる
- 階層表現や継承表現ができる

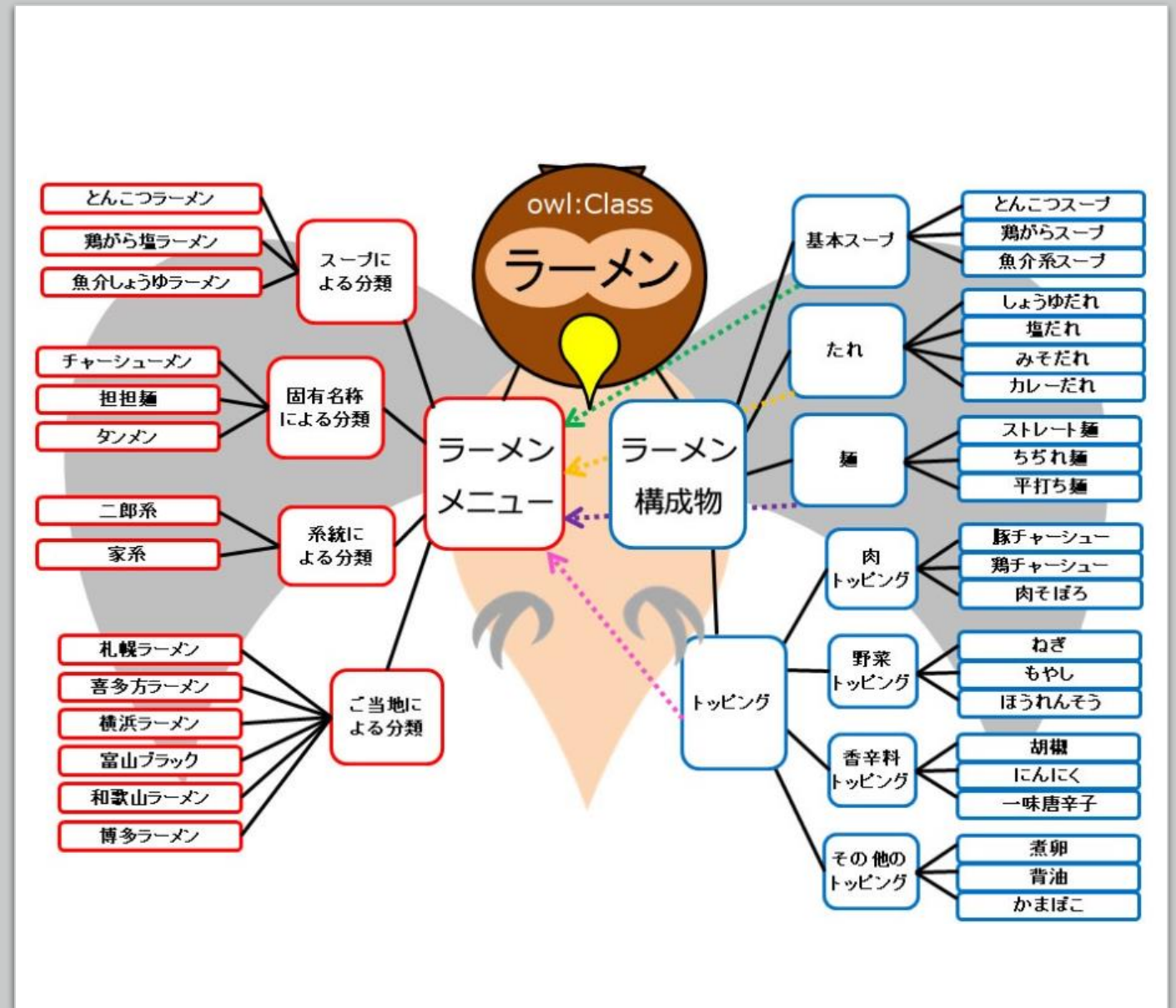
## 短所

- インタプリタの構築が大変
- 推論の妥当性が保証されない
- 知識間の整合性を取りにくい

鳥	クラス	
IS-A	value	動物
Has-A	default	羽
足の数	default	2
体重	If-Needed	calc-weight

# オントロジー

- もともとは哲学用語で「存在論」を意味する
- 物事の共通概念をまとめていったもの
- セマンティックネットと似ているが記法や表現方法に制約はなく、知識の「内容」や「役割」を自由に記述できる。



## セマンティックネットとオントロジーの解説動画（おまけ前まで）

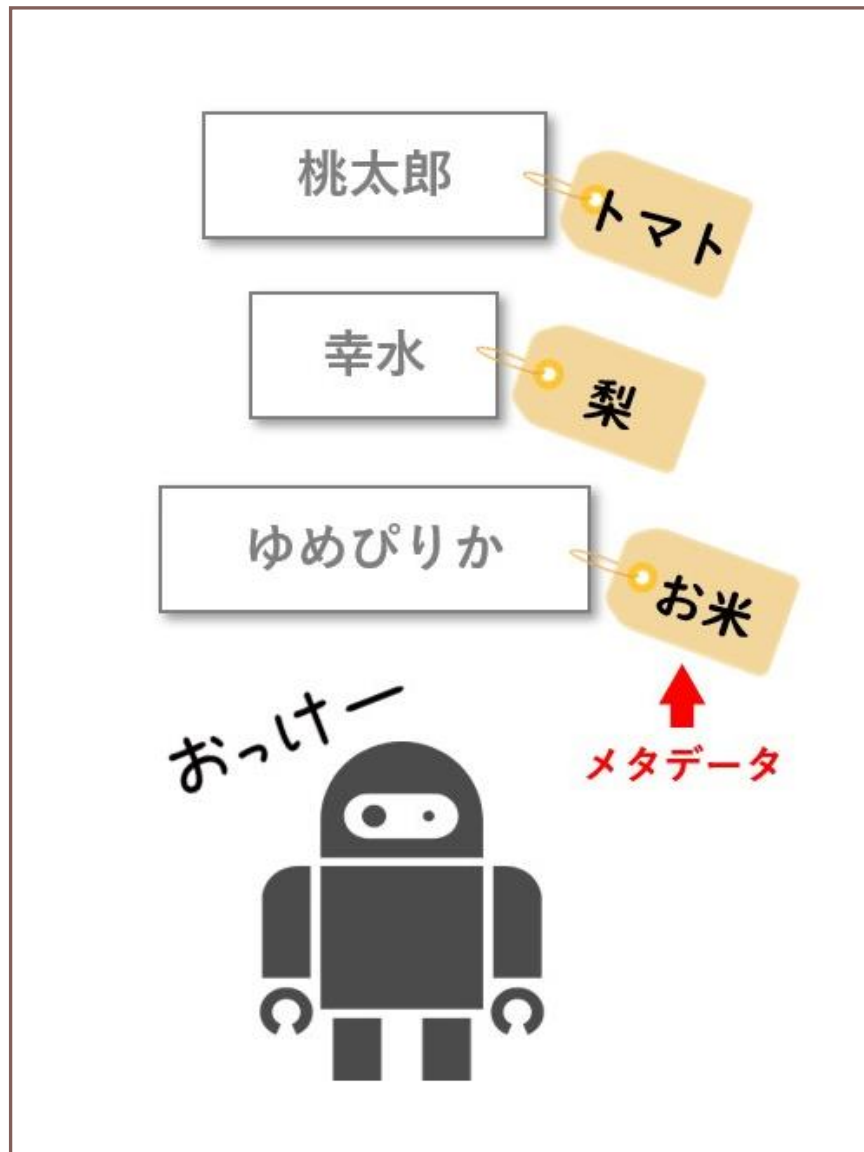




# セマンティックWeb

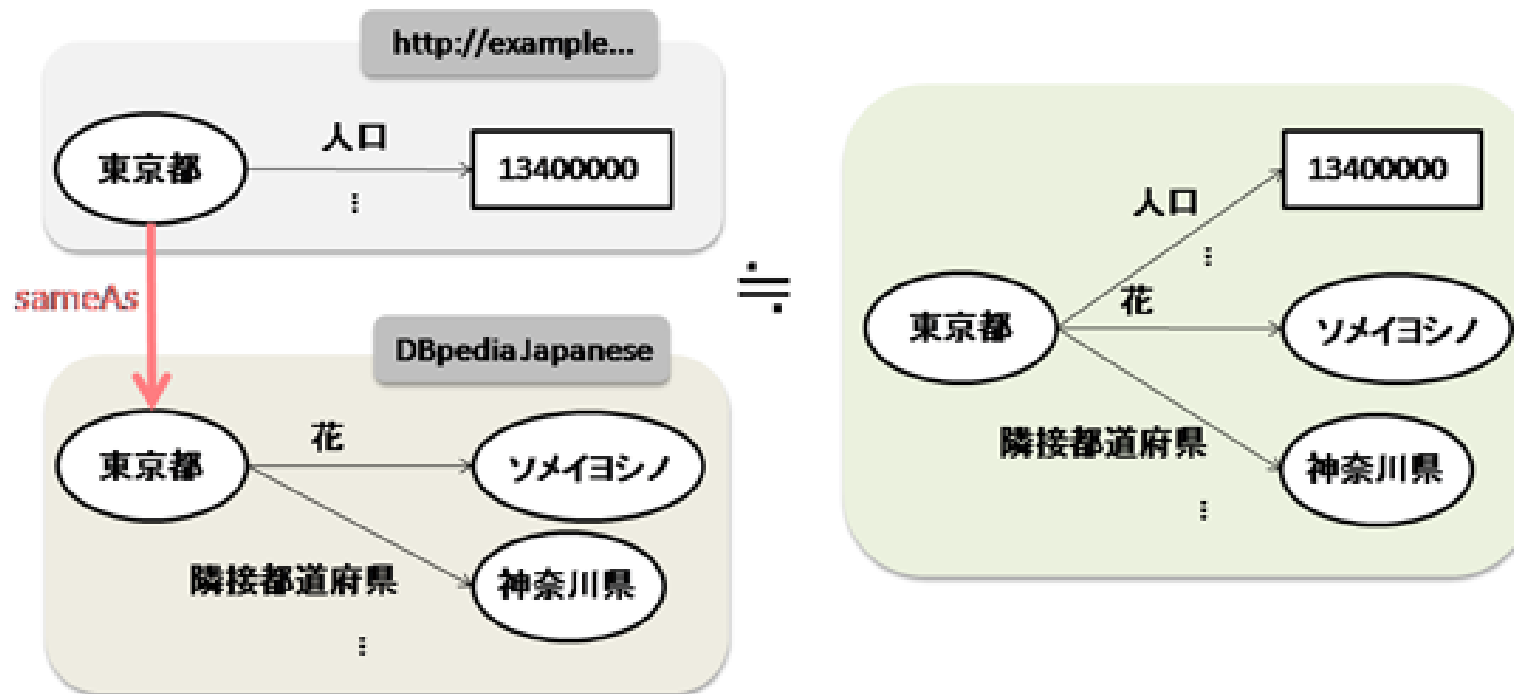
- 今まで人間が判断していた情報の意味を，Webページのメタデータとして付与する
- 情報をコンピュータが自動的に処理
- メタデータは，＜主語，述語，目的語＞の3要素から成るので，RDFトリプルという(Resource Description Framework)

例) 多胡は知識工学の担当者である。  
＜多胡，担当する，知識工学＞



# Linked Open Data

- セマンティックWebの応用
- 現状, URLやhtmlから関係したページはリンクなしには辿れない
  - 人間が中身を見てリンクを貼るか判断している
- URLのリンクでデータをつなぐのではなく, メタデータでつなぐ



# LODの例

- 気象庁防災情報XML履歴データベース

<http://agora.ex.nii.ac.jp/cps/weather/report/>



1. どこかの気象台を開き、個別の事例（日時のリンク）を開く。

[http://agora.ex.nii.ac.jp/cgi-bin/cps/report\\_each.pl?id=20210425192608\\_0\\_VPWW53\\_015000](http://agora.ex.nii.ac.jp/cgi-bin/cps/report_each.pl?id=20210425192608_0_VPWW53_015000)

防災情報電文検索（府県気象情報／宮古島地方気象台）	
1	<div>2021-04-21 16:28:37+09</div> <div>府県気象情報</div> <div>宮古島地方気象台</div>
2	<div>2021-01-08 10:15:29+09</div> <div>府県気象情報</div> <div>宮古島地方気象台</div>

2. 個別事例のページの下の方に、XML電文があり、そこでメタデータが見られる

```
<Title>気象特別警報・警報・注意報</Title>
<DateTime>2021-04-21T07:28:10Z</DateTime>
<Status>通常</Status>
<EditorialOffice>稚内地方気象台</EditorialOffice>
<PublishingOffice>稚内地方気象台</PublishingOffice>
</Control>
<Head xmlns="http://xml.kishou.go.jp/jmaxml1/information">
<Title>宗谷地方気象警報・注意報</Title>
```

3. 自動でメタデータからまとめられた情報も見る事ができる

網走・北見・紋別 地方	網走地方
着雪注意報	着雪注意報
2021-04-21 15:57:00+09	2021-04-21 15:57:00+09
波浪注意報	波浪注意報
2021-04-20 16:53:00+09	2021-04-20 16:53:00+09
強風注意報	強風注意報
2021-04-20 04:20:00+09	2021-04-20 04:20:00+09
<a href="#">もっと見る &gt;</a>	<a href="#">もっと見る &gt;</a>
	長野県

# 今日のまとめ

- 知識の種類について学んだ
- 知識の表現方法について学んだ
- 知識の繋がりを活用したWebリンクを学んだ
- 次回は5/11. 論理表現についてやります.

# レビューシートの提出

- 今日の授業に関するレビューシートを，manabaから提出すること.

後日不明点があれば，多胡まで.

7号館5階 第9実験室内 第9研究室

tago@net.it-chiba.ac.jp