人工知能入門
第4回

藤田 悟

黄　潤和

前回の復習：述語論理

- 論理式の要素を述語とパラメータで表現できるように拡張したものを、述語論理と呼ぶ。
  - 鳥(ニワトリ): ニワトリは鳥である
  - 飛ぶ(ニワトリ): ニワトリは空を飛ぶ
  - 鳥(ニワトリ)⇒飛ぶ(ニワトリ): ニワトリが鳥であれば、ニワトリは空を飛ぶ
  - 鳥(カモ): カモは鳥である
- パラメータは変数にしてもよい
  - 鳥(x)⇒飛ぶ(x): xが鳥ならば、xは空を飛ぶ
    - 変数xを用いれば、ニワトリとカモの両者について知識を書く必要がなくなる

前回の復習: 述語論理を用いた推論

- 親(太郎, 一郎)
- 親(太郎, 桜)
- 親(花子, 一郎)
- 親(花子, 桜)
- 男(太郎)
- 男(一郎)
- 女(花子)
- 女(桜)
- 父(x, y) ⇒ 親(x, y) ∧ 男(x)
- 娘(y, x) ⇒ 親(x, y) ∧ 女(y)

- 一郎の父は? 父(x, 一郎)
- 太郎の娘は? 娘(x, 太郎)

今回学ぶこと

- 手続的知識
  - 宣言的知識を操作するための知識
  - プロダクションシステムの考え方を学ぶ
- 人工知能の歴史: エキスパートシステム

手続的知識

手続的知識

- 宣言的知識を、どのように操作して、推論を行うかを記述した知識
  - 手続的知識とは、**宣言的知識を扱うための命令の集合**
  - 宣言的知識と手続的知識の関係は、データとプログラムの関係。

- 親(太郎, 一郎)
- 親(太郎, 桜)
- 親(花子, 一郎)
- 親(花子, 桜)
- 男(太郎)
- 男(一郎)
- 女(花子)
- 女(桜)
- 父(x, y) ⇒ 親(x, y) ∧ 男(x)
- 娘(y, x) ⇒ 親(x, y) ∧ 女(y)

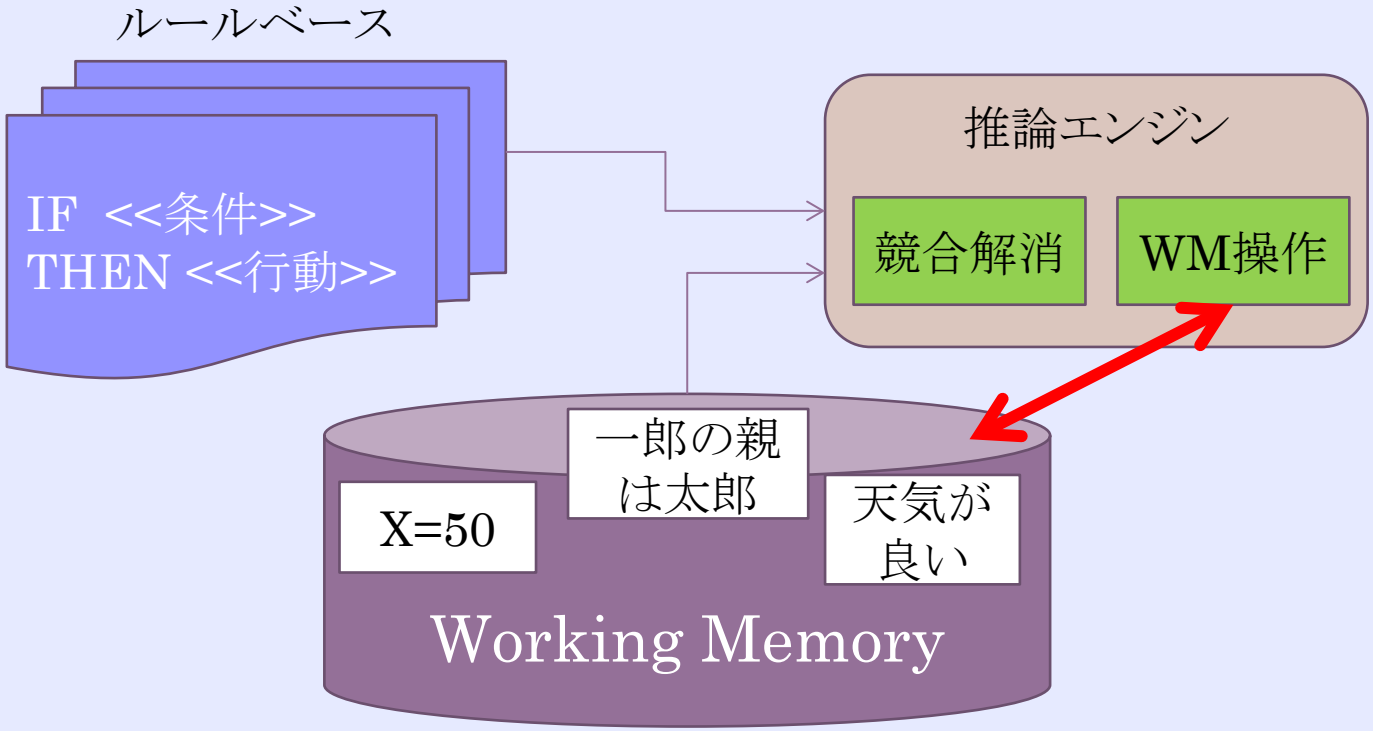
プロダクションシステム

- 手続的知識として、IF-THEN型 のプロダクションルールを記述できる推論システム
  - IF <<条件>>
    THEN <<アクション>>
  - 条件には、宣言的知識の真偽、値の比較、および、それらの組合せ(∧, ∨)を記述する
  - アクションには、宣言的知識の追加/削除、値の読み込み、ユーザへの指示を記述する

人間に直観的な形のルールで手続的知識を定義できる

プロダクションシステムの構成

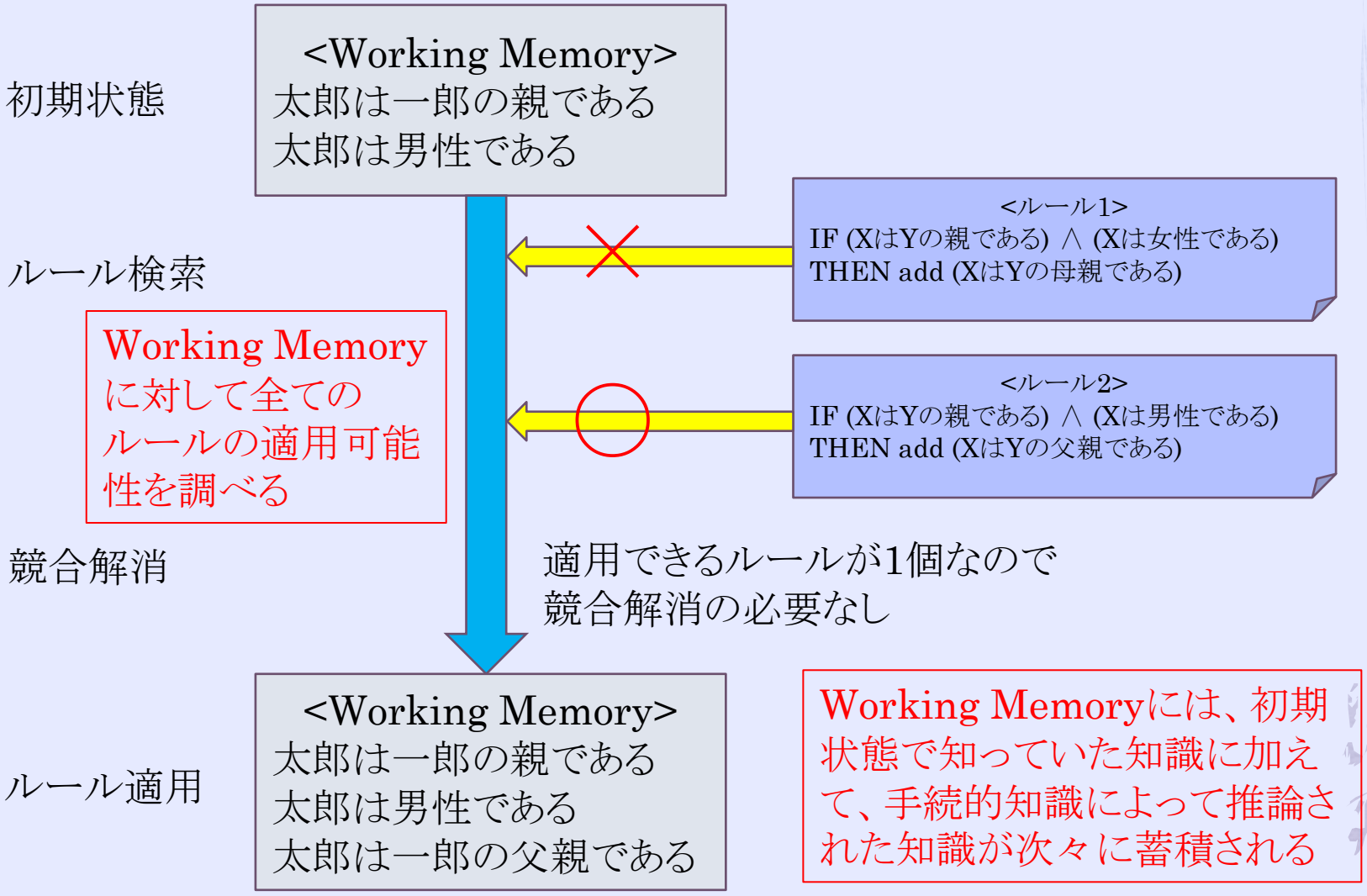
- 宣言的知識を出し入れする**作業メモリ(WM)**
- 手続的知識を保有する**ルールベース**
- WMとマッチするルールを検索し、アクションを決定する**推論エンジン**



プロダクションシステムの例1

- 最初に与えられる宣言的知識
  - 太郎は一郎の親である
  - 太郎は男性である
- 手続的知識(プロダクションルール)
  - ルール1
    - IF (XはYの親である) ∧ (Xは女性である)
      THEN add (XはYの母親である)
  - ルール2
    - IF (XはYの親である) ∧ (Xは男性である)
      THEN add (XはYの父親である)

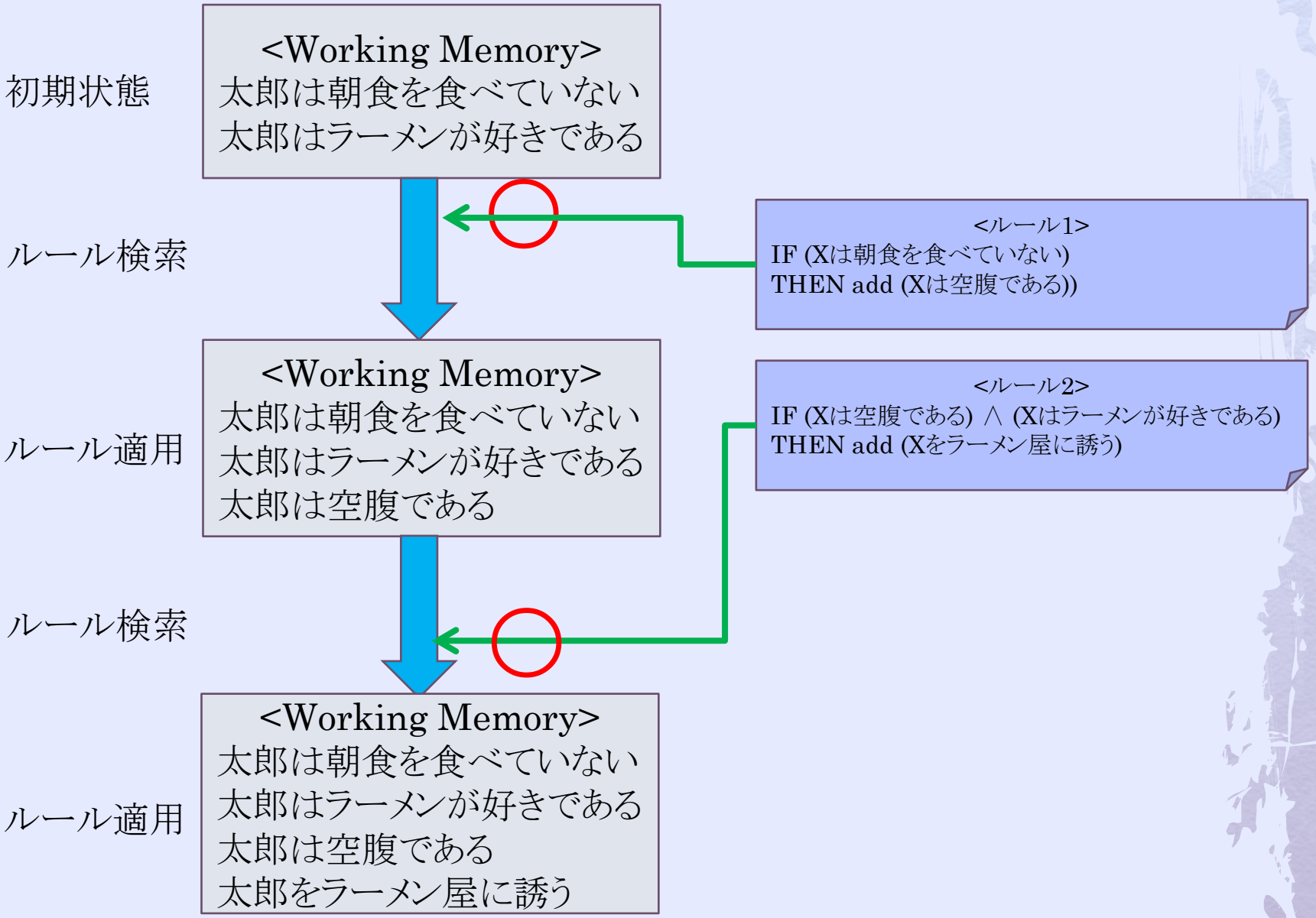
プロダクションシステムの動作例1



プロダクションシステムの例2

- 最初に与えられる宣言的知識
  - 太郎は朝食を食べていない
  - 太郎はラーメンが好きである
- 手続的知識
  - ルール1
    - IF (Xは朝食を食べていない)
      THEN add (Xは空腹である)
  - ルール2
    - IF (Xは空腹である) ∧ (Xはラーメンが好きである)
      THEN add (Xをラーメン屋に誘う)

プロダクションシステムの動作例2

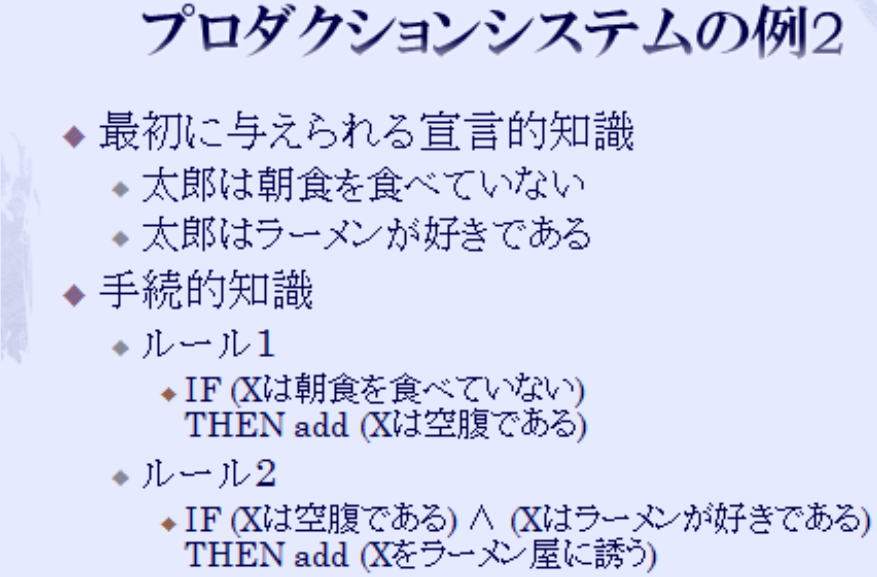


(演習1) 息子を見つける

- 例題1の初期の宣言的知識と、手続的知識に追加して、一郎が太郎の息子であることを推論できるように、例題を改造せよ。
- 一郎が太郎の息子であることの推論過程を説明せよ。

(演習2) 息子を見つける

- 例2の初期の宣言的知識に、太郎がうどんが好きなことを追加し、手続的知識に、うどんが好きである人が空腹なら、うどん屋に誘うと言うルールを追加せよ。
- この場合、太郎をどこに誘うのか推論過程を示して、説明せよ。



プロダクションシステムの長所・短所

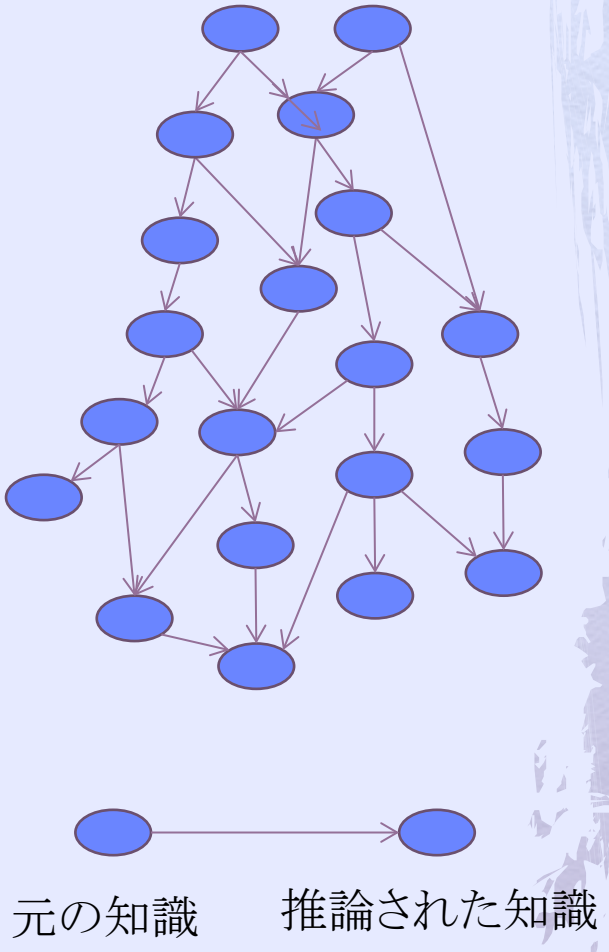
- 長所
  - IF-THEN のルールは思いつきやすく、書きやすい。
  - WM の読み書きだけなので、直観的にわかりやすい。
  - …
- 短所
  - ルール全体で、どういう流れの推論ができるのか、見通しがわかりにくく複雑。
  - ルールが増えてくると、競合するルールも書けてしまい、矛盾が生じる。
  - …

知識の単調性・非単調性

- 一般に、プロダクションシステムを用いると、初期の宣言的知識に加え、手続的知識から推論された宣言的知識が追加されていく
  - 時間と共に、WM 内の知識量は**単調に増加**する
- しかし、否定する知識が生まれると、これまで妥当だった推論が棄却される場合がある。
  - 「遊園地に行く」「おにぎりを用意する」と推論している途中で、「明日は遊園地は休園」という情報が加わると、一気に推論結果が無駄になる。
  - これを、**知識の非単調性**と呼ぶ

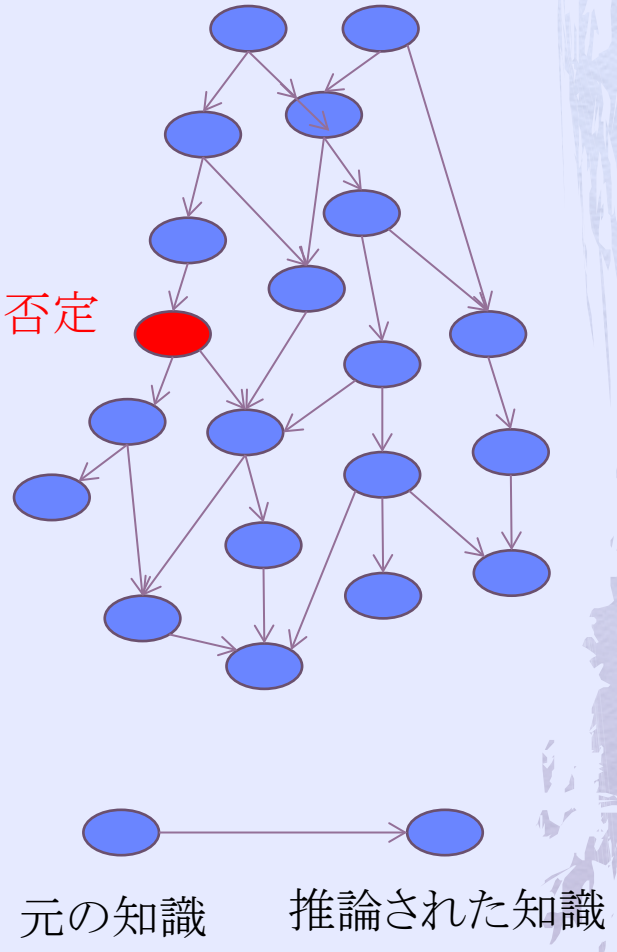
Truth Maintenance System

- 知識をWMに追加するに当たって、推論の根拠となる知識へのリンクを保持する。
- 根拠となる知識が否定されたら、その知識から推論された結果を削除し、矛盾を解消する



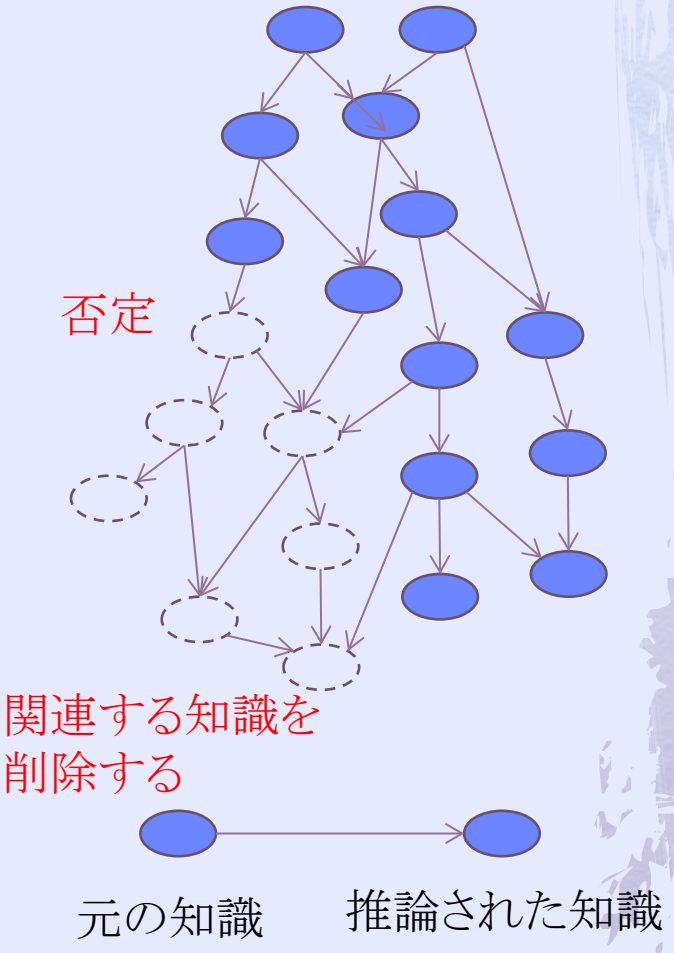
Truth Maintenance System

- 知識をWMに追加するに当たって、推論の根拠となる知識へのリンクを保持する。
- 根拠となる知識が否定されたら、その知識から推論された結果を削除し、矛盾を解消する



Truth Maintenance System

- 知識をWMに追加するに当たって、推論の根拠となる知識へのリンクを保持する。
- 根拠となる知識が否定されたら、その知識から推論された結果を削除し、矛盾を解消する



An example:

Rules defined in the rule base file: CarShop

```
rule  "CarRule1"
if    "?x is inexpensive"
then "?x is made in Japan"
rule  "CarRule2"
if    "?x is small"
then "?x is made in Japan"
rule  "CarRule3"
if    "?x is expensive"
then "?x is a foreign car"
rule  "CarRule4"
if     "?x is big"
       "?x needs a lot of gas"
then    "?x is a foreign car"
rule  "CarRule5"
if    "?x is made in Japan"

       "?x has Toyota's logo"
then "?x is a Toyota"
rule  "CarRule6"
if    "?x is made in Japan"
       "?x is a popular car"
then "?x is a Toyota"
```

```
rule  "CarRule7"
if    "?x is made in Japan"
       "?x has Honda's logo"
then "?x is a Honda"
rule  "CarRule8"
if    "?x is made in Japan"
       "?x has a VTEC engine"
then "?x is a Honda"
rule  "CarRule9“
if    "?x is a Toyota"
       "?x has several seats"
       "?x is a wagon"
then "?x is a Carolla Wagon"
rule  "CarRule10"
if    "?x is a Toyota"
       "?x has several seats"
       "?x is a hybrid car"
then "?x is a Prius"
rule  "CarRule11"
if    "?x is a Honda"
       "?x is stylish"
       "?x has several color models"
       "?x has several seats"
       "?x is a wagon"
then "?x is an Accord Wagon"
```

```
rule  "CarRule12"
if    "?x is a Honda"
       "?x has an aluminium body"
       "?x has only 2 seats"
then "?x is a NSX"
rule  "CarRule13"
if    "?x is a foreign car"
       "?x is a sports car"
       "?x is stylish"
       "?x has several color models"
       "?x has a big engine"
then "?x is a Lamborghini Countach"
rule  "CarRule14"
if    "?x is a foreign car"
       "?x is a sports car"
       "?x is red"
       "?x has a big engine"
then "?x is a Ferrari F50"
rule  "CarRule15"
if    "?x is a foreign car"
       "?x is a good face"
then "?x is a Jaguar XJ8"
```

My car is inexpensive
My car has VTEC engine
My is stylish

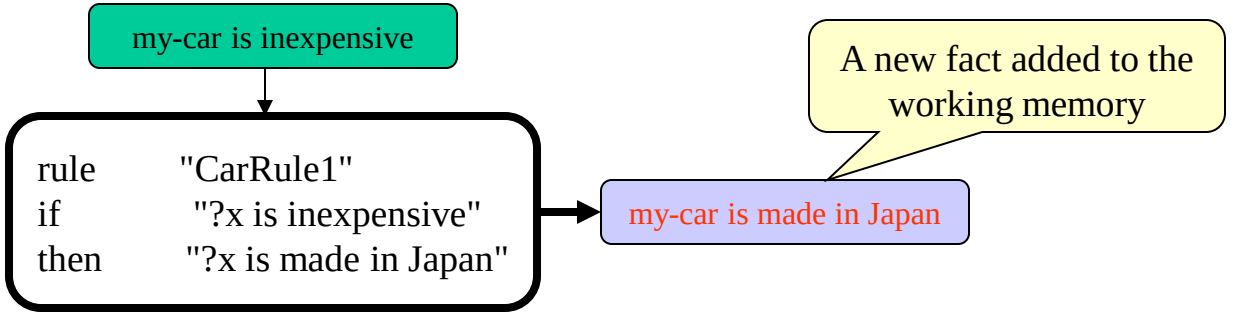
Initial Working Memory:

My car is inexpensive
My car has VTEC engine
My car is stylish
My car has several color models
My car has several seats
My car is a wagon

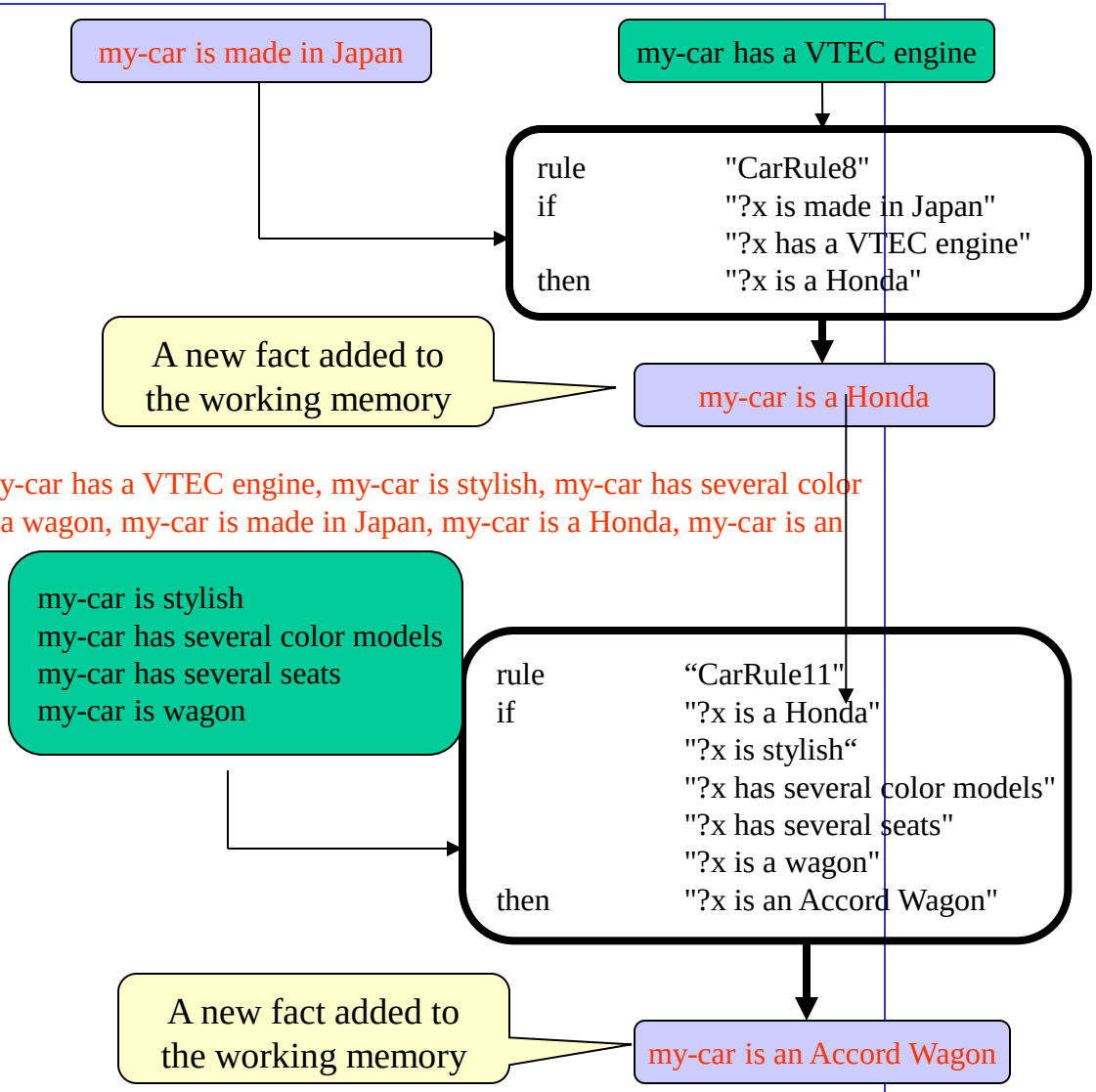
Output:

```
% java RuleBaseSystem
% java RuleBaseSystem
ADD:my-car is inexpensive
ADD:my-car has a VTEC engine
ADD:my-car is stylish
ADD:my-car has several color models
ADD:my-car has several seats
ADD:my-car is a wagon
CarRule1 [?x is inexpensive]->?x is made in Japan
CarRule2 [?x is small]->?x is made in Japan
CarRule3 [?x is expensive]->?x is a foreign car
CarRule4 [?x is big, ?x needs a lot of gas]->?x is a foreign car
CarRule5 [?x is made in Japan, ?x has Toyota's logo]->?x is a Toyota
CarRule6 [?x is made in Japan, ?x is a popular car]->?x is a Toyota
CarRule7 [?x is made in Japan, ?x has Honda's logo]->?x is a Honda
CarRule8 [?x is made in Japan, ?x has a VTEC engine]->?x is a Honda
CarRule9 [?x is a Toyota, ?x has several seats, ?x is a wagon]->?x is a Carolla Wagon
CarRule10 [?x is a Toyota, ?x has several seats, ?x is a hybrid car]->?x is a Prius
CarRule11 [?x is a Honda, ?x is stylish, ?x has several color models, ?x has several seats, ?x is a wagon]->?x is an Accord Wagon
CarRule12 [?x is a Honda, ?x has an aluminium body, ?x has only 2 seats]->?x is a NSX
CarRule13 [?x is a foreign car, ?x is a sports car, ?x is stylish, ?x has several color models, ?x has a big engine]->?x is a Lamborghini
Countach
CarRule14 [?x is a foreign car, ?x is a sports car, ?x is red, ?x has a big engine]->?x is a Ferrari F50
CarRule15 [?x is a foreign car, ?x is a good face]->?x is a Jaguar XJ8
apply rule:CarRule1
Success: my-car is made in Japan
ADD:my-car is made in Japan
apply rule:CarRule2
apply rule:CarRule3
apply rule:CarRule4
apply rule:CarRule5
apply rule:CarRule6
apply rule:CarRule7
```

Initial facts in the working memory

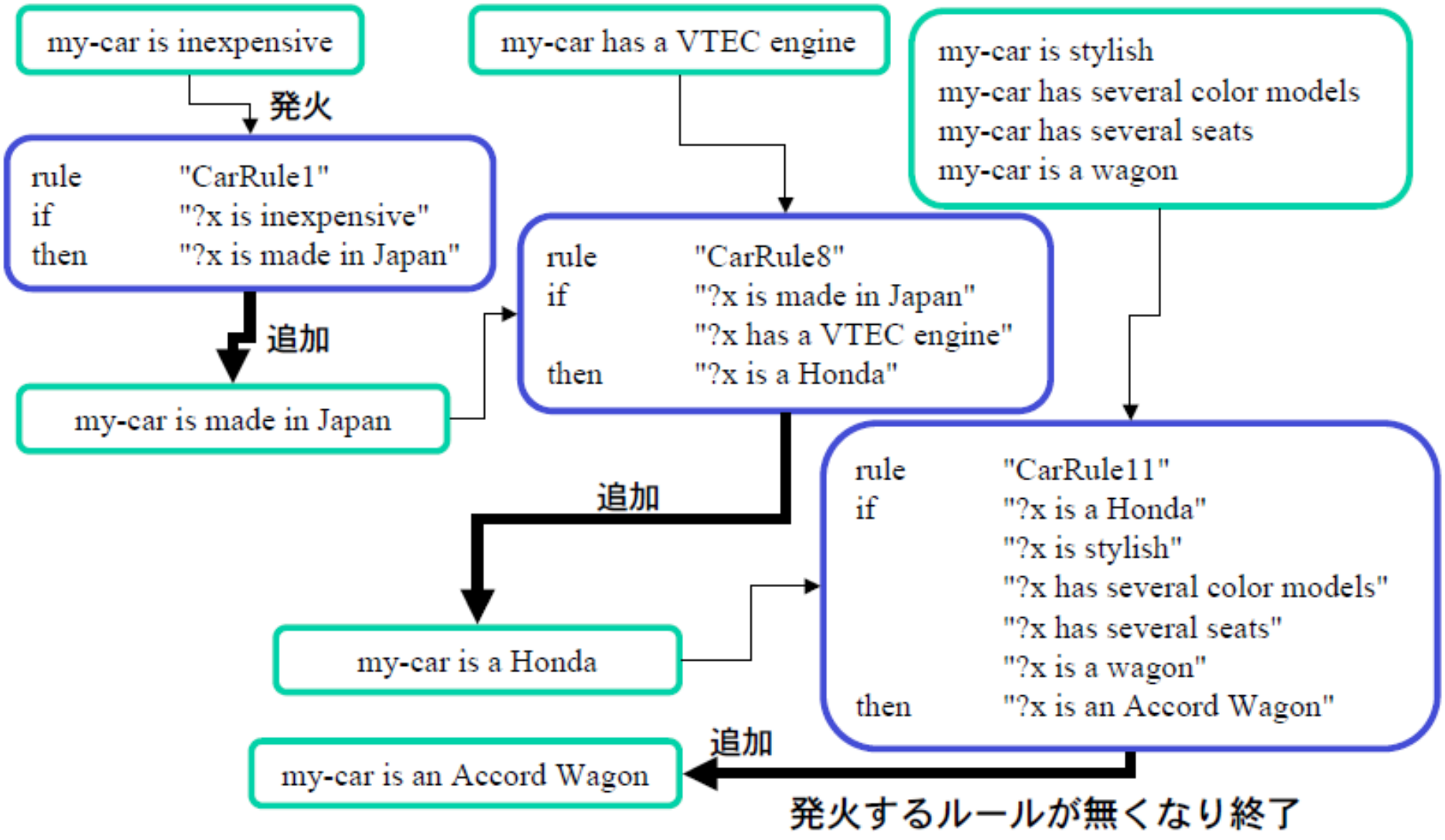


apply rule:CarRule8
Success: my-car is a Honda
ADD:my-car is a Honda
apply rule:CarRule9
apply rule:CarRule10
apply rule:CarRule11
Success: my-car is an Accord Wagon
ADD:my-car is an Accord Wagon
apply rule:CarRule12
apply rule:CarRule13
apply rule:CarRule14
apply rule:CarRule15
Working Memory[my-car is inexpensive, my-car has a VTEC engine, my-car is stylish, my-car has several color models, my-car has several seats, my-car is a wagon, my-car is made in Japan, my-car is a Honda, my-car is an Accord Wagon]
apply rule:CarRule1
apply rule:CarRule2
apply rule:CarRule3
apply rule:CarRule4
apply rule:CarRule5
apply rule:CarRule6
apply rule:CarRule7
apply rule:CarRule8
apply rule:CarRule9
apply rule:CarRule10
apply rule:CarRule11
apply rule:CarRule12
apply rule:CarRule13
apply rule:CarRule14
apply rule:CarRule15
Working Memory[my-car is inexpensive, my-car has a VTEC engine, my-car is stylish, my-car has several color models, my-car has several seats, my-car is a wagon, my-car is made in Japan, my-car is a Honda, my-car is an Accord Wagon]
No rule produces a new assertion



Forward Chaining-他の例

rules



エキスパートシステム

歴史: エキスパートシステム

- 1980年代後半に、プロダクションシステムの考え方が広がり、産業界に「人工知能ブーム」が起こった。
  - 様々な専門家(エキスパート)の知識を、IF-THENルールとして聞きだし、専門家と同じ判断をするシステムが多く開発された。
- しかし、以下の理由などで、そのブームは去った。
  - 専門家の知識は簡単には記述できない。
  - 膨大な「常識」について知識ベースを作れない。
  - 書かれない知識については動かない。(例外に弱い)
  - 結果の性能が保証できない

実際のエキスパートシステム

- 論理回路合成エキスパートシステム
  - 論理回路部品を最小化するために試行錯誤するエキスパートシステム
- ソフトウェア自動合成システム
  - プログラムの一部を excel 表から自動合成するシステム
- 薬品系工場の稼働スケジューリングシステム
  - どの装置で、その薬品の化合を行うかをスケジューリングするエキスパートシステム
- ネットワーク設計エキスパートシステム
  - 職場環境に適切なネットワーク設計をアドバイスするシステム

(演習3) 遊園地？モール？

- 休日に遊園地に出かけるか、ショッピングモールに出かけるかを決定するプロダクションシステムを作成せよ。
  - 初期状態の宣言的知識
    - 天気が良い、遊園地が遠い、など
  - 手続的知識
    - IF (天気が良い) ∧ (暇である) THEN …
    - THEN の部分には、add(xxx), delete(xxx) というようなWMへの追加、削除の命令を書く
- Working Memory の変化の様子を、適用するルールと共に示せ。

(課題) PCの故障診断

- PC の故障診断のプロダクションシステムを設計せよ。
- 初期の宣言的知識として、「電源ランプがつかない」、「HDDの動作音がしない」などの観察できる状況を与えて、そこから、どのようにPCが故障しているかを判断する過程をWMの状態を示して、説明せよ。
  - THEN 部には、WMへの宣言的知識の追加(add)と削除(delete)だけが行えるものとする

(オプション課題)

- 教科書 p36 演習2に回答せよ

http://www.amzi.com/ExpertSystemsInProlog/xsipfrtop.htm

2.1 The Bird Identification System

```
bird(laysan_albatross):- family(albatross), color(white).

bird(black_footed_albatross):- family(albatross), color(dark).

bird(whistling_swan) :- family(swan), voice(muffled_musical_whistle).

bird(trumpeter_swan) :- family(swan), voice(loud_trumpeting).

family(albatross).

color(dark).
```

Enter your query:

bird(X).

Run query ☐ Show parse results

```
Parsing rulesets.

Attaching builtins to database.
Attachments done.

Parsing query.
X = black_footed_albatross
```

http://ioctl.org/logic/prolog-latest