人工知能概論

•http://www.i.ci.ritsumei.ac.jp/~shirai/index.html

白井 良明
知能情報学科(CC4F)
shirai@ci.ritsumei.ac.jp

人工知能って何？

「人 工知能」とはまるで人間のようにふるまう機械を想像するのではないでしょうか？
これは正しいとも，間違っているともいえます．なぜ なら，人工知能の研究には二つの立場があるからです．

・人間の知能そのものを作ろうとする
・人間が知能を使ってすること を機械にさせようとする

実際の研究のほとんどは後者の立場にたっています．ですので，人工知能の研究といっても，人間のような機械を作っているわけではありません．

初期のAIの歴史

- 1956 Dartmouth 会議（AIの研究会）
  McCarthy主催,
  Minsky, Shannon, Newell, Simonなど

白井先生,
今回は，Dartmouth会議が 1956 年の何月何日に始まったのかを，先生を通じてお尋ねいただけないかと思いメールさせていただきました．
今年はDartmouth会議50周年に当たるため，この日付を
http://www.ai-gakkai.or.jp/jsai/whatsai/AItopics5.html
に載せたりとか，この日にメールでアナウンスするなどしたいと考えています．

1971-72年　MIT　AI Lab 客員研究員

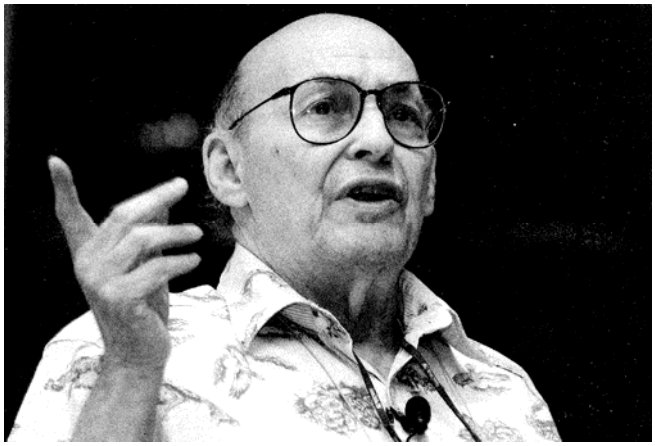
初期のAIの歴史

- 1956 Dartmouth 会議（AIの研究会）
  McCarthy主催,
  Minsky, Shannon, Newell, Simonなど
- 1959 AI Lab. 設立
  Minsky, McCarthy
-

参考文献

- Newell，Shaw＆Simon: Chess-Playing Programs and the Problem of Complexity, IBM J. of R&D, 2 (4) ,1958
- Gelernter: Realization of Geometry Theorem Proving Machine, ICIP (IFIP), 1959
- Serfridge & Nessier: Pattern Recognition by Machine, Scientific American, 203, 1960

2005年米国AI学会での講演



初期のAIの歴史

- 1956 AIの研究会 (Minsky, Shannon, Newell, Simonなど)
- 1959 AI Lab. 設立 (Minsky, McCarthy)
- 1961 AI へのステップ (Minsky)

  探索、パターン認識、学習、問題解決

  GPS (N.S.S)

  LISP (McCarthy)

Newell 電総研訪問(1976年頃)

参考文献

- Green et al: “BASEBALL” An Automatic Question-Answer, WJCC, 1961
- Newell & Simon: Computer Simulation of Human Thinking and Problem-Solving, Computer and Automation, 10, 1961
- Slagle: A Computer Program for Solving Problems in Freshman Calculus (SAINT), Thesis M.I.T., 1961
- Minsky: Steps Toward Artificial Intelligence, Proc,. IRE, 49, 1961
- McCarthy et al: LISP 1.5 Programmer’s Manual, The MIT Press ,1962

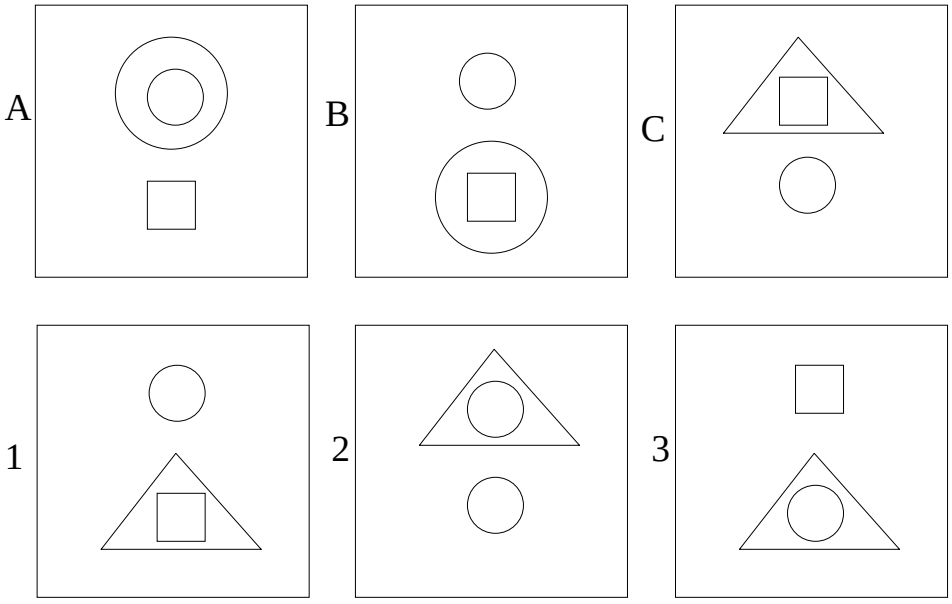
初期のAIの歴史

- 1956 AIの研究会 (Minsky, Shannon, Newell, Simonなど)
- 1959 AI Lab. 設立 (Minsky, McCarthy)
- 1961 AI へのステップ (Minsky)
  探索、パターン認識、学習、問題解決
  GPS (N.S.S)
  LISP (McCarthy)
- 1963 Computer and Thought(Feigenbaumなど)
  STUDENT(Bobrow) , SIR (Raphael)
- 1968 Semantic Information Processing (Minsky)

参考文献

- Feigenbaum & Feldman (ed.): Computer and Thought, Mcgraw-Hill, 1963
- Evans: A Heuristic Program to Solve Geometric Analogy Problems, 1963
- Bobrow: Natural language Input for a Computer Problem-Solving System, Thesis M.I,.T., 1964
- Raphael: SIR: A Computer Program for Systematic Information Retrieval, Thesis M.I,.T., 1964

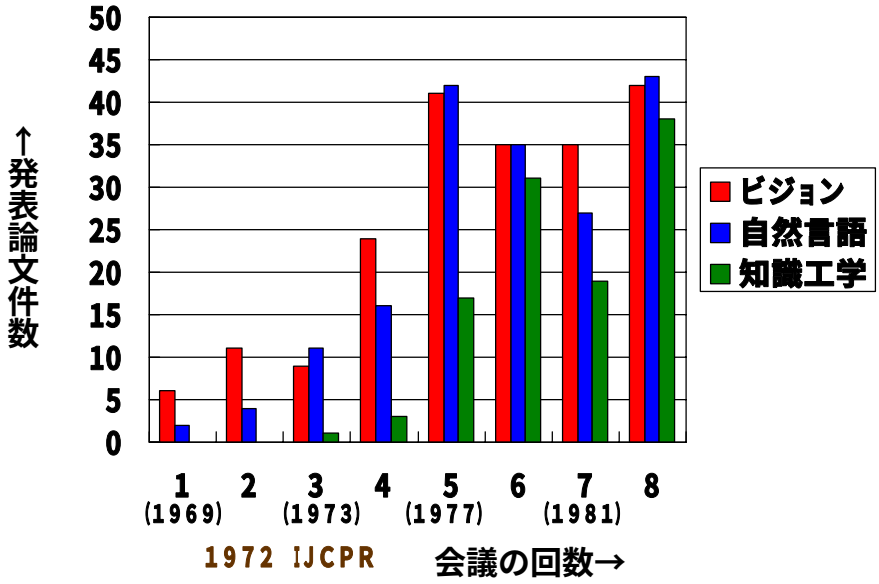
Evans の Analogy



初期のAIの歴史

1956 AIの研究会（Minsky, Shannon, Newell, Simonなど）
1959 AI Lab. 設立（Minsky, McCarthy）
1961 AI へのステップ（Minsky）
探索、パターン認識、学習、問題解決
GPS（N.S.S）
LISP（McCarthy）
1963 Computer and Thought(Feigenbaumなど)
STUDENT(Bobrow)，SIR（Raphael）
1968 Semantic Information Processing（Minsky）
1969 第1回人工知能国際会議

初期のAIの主な分野



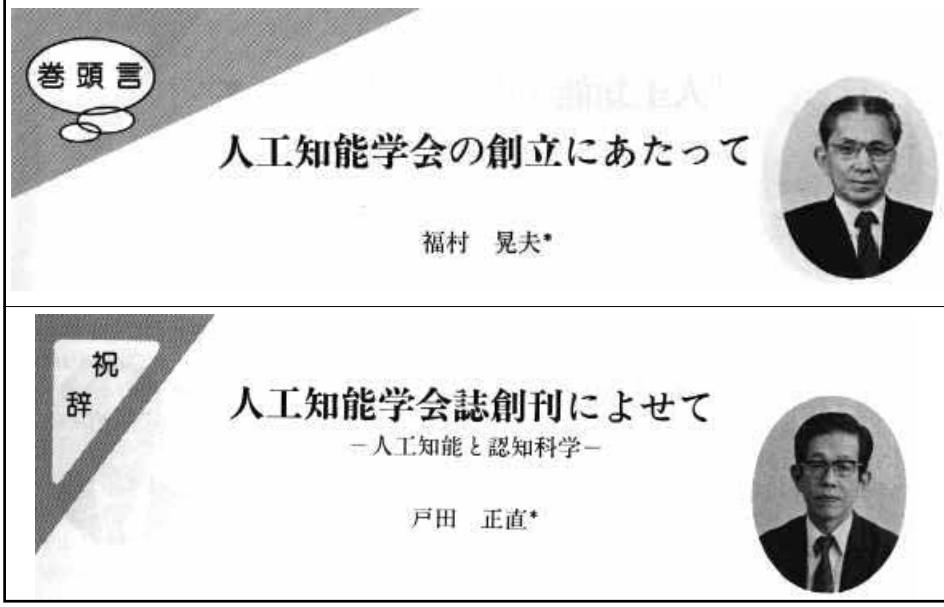
論文数比較

	IJCAI'83	IJCAI'85
知識表現	32	21
知識工学	40	28
知識獲得・学習	22	31
推論(Reasoning)	ー	22
計画と探索	14	20
自然言語処理	45	28
ビジョン (知覚)	42	24
その他	80	89
合計	275	263

80年代初期のIJCAIの論文数

	IJCAI'83	IJCAI'85
知識獲得･学習	22	31
推論(Reasoning)	ー	22
計画と探索	14	20
知識表現	32	21
知識工学	40	28
自然言語処理	45	28
ビジョン(知覚)	42	24
その他	80	89
合計	275	263

人工知能学会誌創刊号(1986年9月)



人工知能学会誌創刊号（1986年9月）

人工知能学会発足によせて
米国人工知能学会(AAAI)会長からのメッセージ

The officers of the American Association for Artificial Intelligence wish the Japanese Society of Artificial Intelligence good fortune in all its efforts. Japan's new society is an encouraging reflection of its members'determination to promote the study of Artificial Intelligence, and the society's activities will surely help to strengthen Japan's already fine worldwide reputation for scientific and engineering accomplishment.

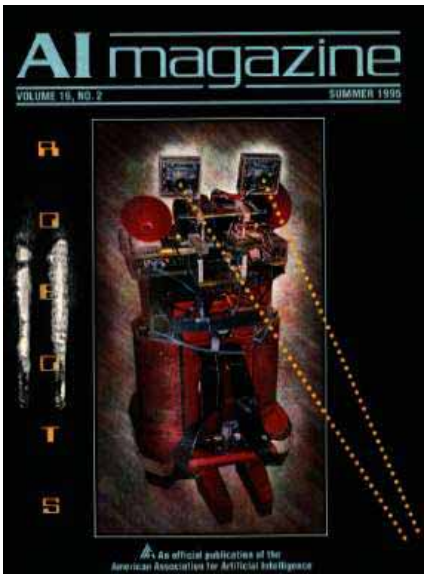
Patrick H. Winston

Professor Patrick H. Winston
AAAI President(1985－1987)

IJCAI'89 セッションと論文数

探索	13
知識表現	47
計画と行動の推論	30
演繹と常識推論	31
学習と知識獲得	44
道具	20
分散処理	11
実時間処理	7
音声と自然言語	16
ビジョンとロボット	16
認知モデル	16

AAAIのAIマガジン

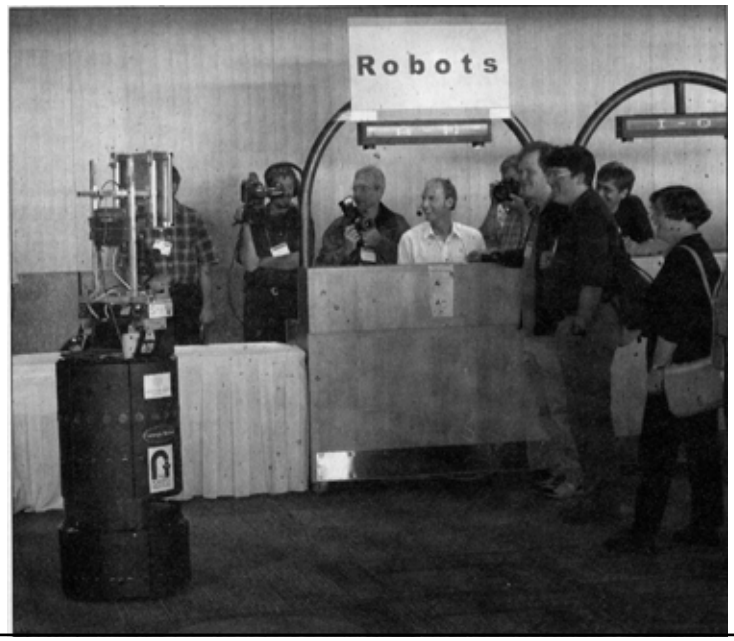


Summer 1995

AAAI Robot Competition ,Exhibition

- 1995 Recycling Trash
- 1996 Office Navigation Nomad
- 1997 Hors d'oeuvres, Anyone?
- 1998 Entertainment robot Lego
  Robocup by wheel and legs
- Challenge: Attend the AAAI Conference

Registration Area of AAAI Meeting



IJCAI'99 セッションと論文数

探索、拘束の充足	20
推論	31
事例ベース推論	10
定性推論，時間推論	15
不確実推論，確率推論	16
計画	13
学習と知識獲得	29
分散AI	12
知識ベース応用	14
自然言語	11
ロボットと知覚	7
認知モデル	8

基本的な考え方

•今後の社会の重要な分野の多くは
人工知能研究と関わりを持つ。

高度情報化社会、IMS，

•人工知能研究の扱う内容が変わり
つつある。

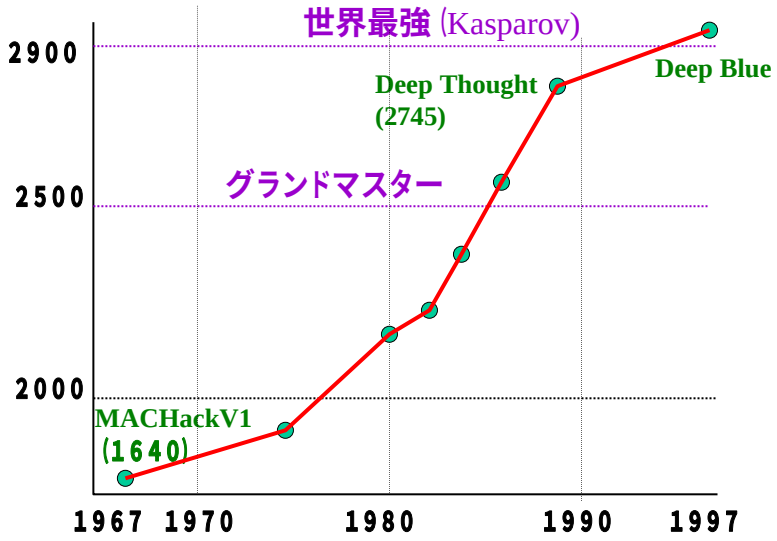
形式重視 → 内容重視

The Hottest Jobs of the Future

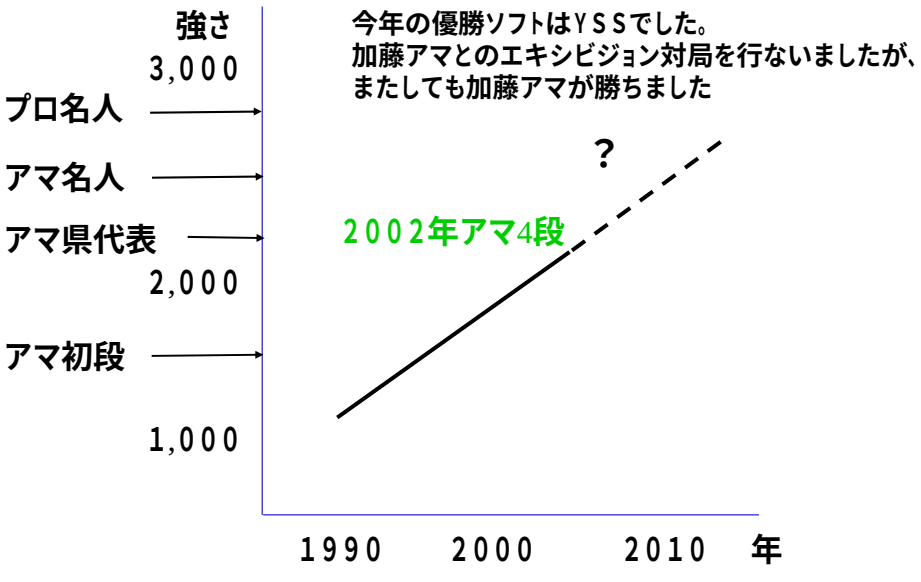
Time May 11, 2000

- Tissue Engineers: 臓器再生、人工皮膚は実用
◯Gene Programmers : DNA解析し，治療法決定
◯Pharmers (Pharmacy + Farmers) : 遺伝子工学による治療効果のある食品生産
◯Frankenfood Monitors : 遺伝子操作による弊害監視
◎Data Miners : 大量データから知識を探す
◎ Hot-line Handymen : 家庭の機器の使い方、修理
◯ Virtual-Reality Actors : pay-per-play, scriptwriter
◯ Narrow Casters : Broadcastから、個人向けへ、CMも
◎ Turing Testers : 人の代わりに人の相手をする
◎ Knowledge Engineers : 顧客の知識→ソフト

チェスマシンの能力変遷



コンピュータの将棋の能力

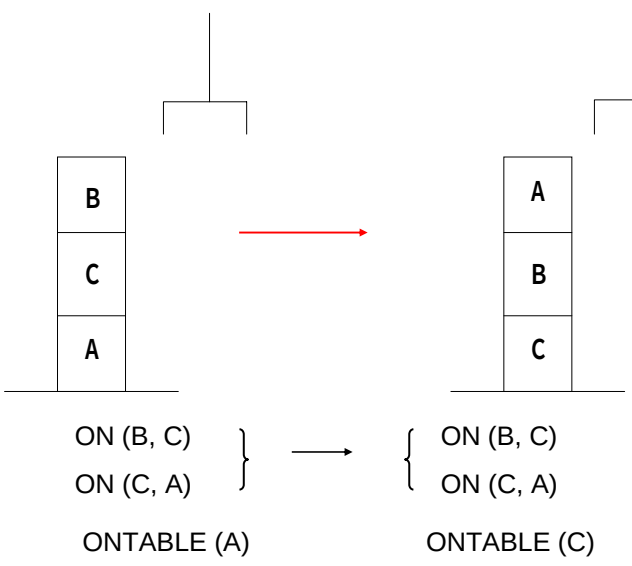


コンピュータ囲碁の歴史

1962年	Remusによる初めてのコンピュータ囲碁の論文
1969年	Zobristによる初めての対局囲碁プログラム（強さは約38級）
1979年	ReitmanとWilcoxのプログラムInterim.2（約15級）
1984年	初めてのコンピュータ囲碁大会（USENIX）
1986年	初めての（Ing Cup, 1985-2000）by 応昌期）
1995年	日本棋院が初めて級位を認定（5級）
2001年	日本棋院が初めて初段に認定

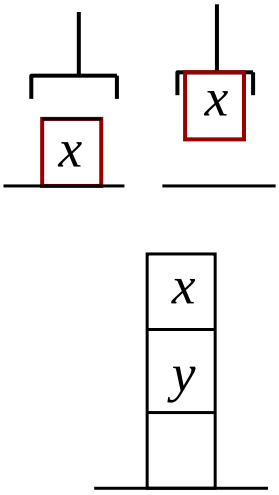
（手談対局4、最高峰3、最強の囲碁2003、銀星囲碁3）

問題の記述：初期状態と目標状態

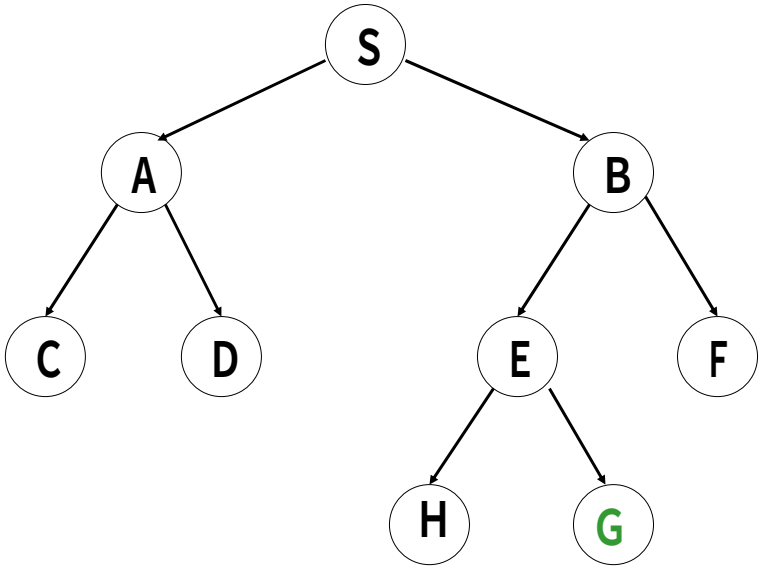


問題の記述：オペレータ

PICKUP (x) ；テーブルの上のxをつかむ
前提条件： ONTABLE (x) ,CLEAR (x) ,EMPTY
削除リスト：ONTABLE (x) ,CLEAR (x) ,EMPTY
追加リスト：HOLD (x) ；手はxをつかんでいる
PUTDOWN (x) ；xをテーブルの上に置く
前提条件： HOLD (x)
削除リスト：HOLD (x)
追加リスト： ONTABLE (x) ,CLEAR (x) ,EMPTY
TAKEOFF (x,y) ；yの上にあるxをつかむ
前提条件：ON (x,y) ,CLEAR (x) ,EMPTY
削除リスト： ON (x,y) ,CLEAR (x) ,EMPTY
追加リスト：CLEAR (y) ,HOLD (x)
PUTON (x,y) ；xをyの上へ置く
前提条件： HOLD (x) ,CLEAR (y)
削除リスト：HOLD (x) ,CLEAR (y)
追加リスト： ON (x,y) ,CLEAR (x) ,EMPTY



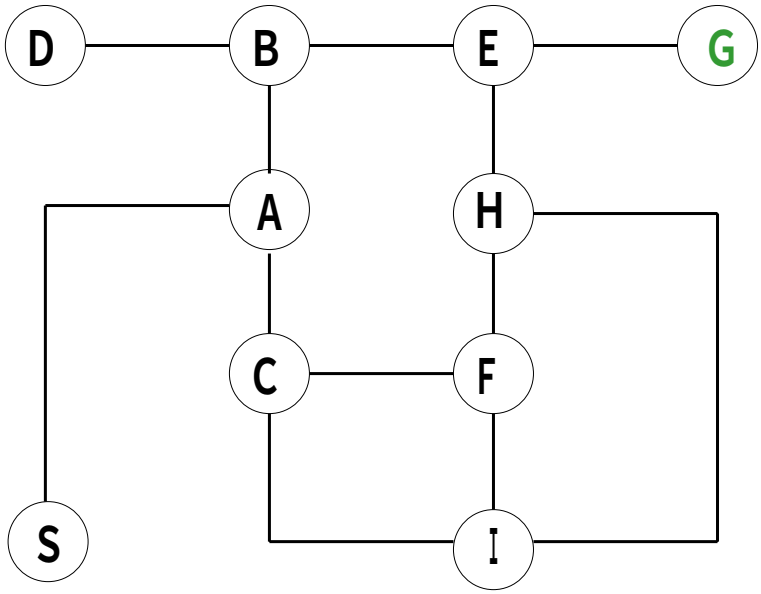
木の探索



procedure width-first tree search

- add (s, **open)**;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**);
- Expand(n);
- Put all child nodes into the last of **open** and attach pointers to n;
- Go to LOOP.

グラフの探索



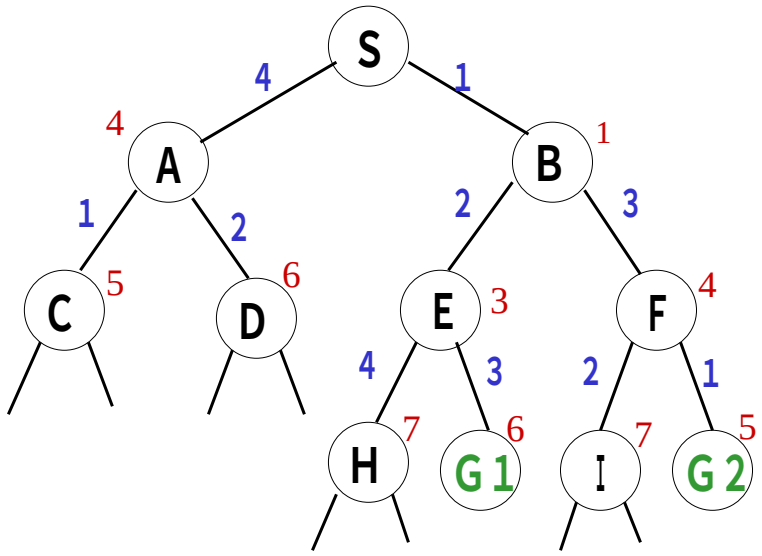
procedure depth-first ~~tree~~ search

- add (s, **open)**;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed**)
- Expand(n);
- Put all child nodes into the top of **open** and attach pointers to n;
- Go to LOOP.

procedure depth-first ~~tree~~ search

- add (s, **open)**;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed)**
- Expand(n);
- Put all child nodes into the top of **open** which are not in **open** nor **closed** and attach pointers to n;
- Go to LOOP.

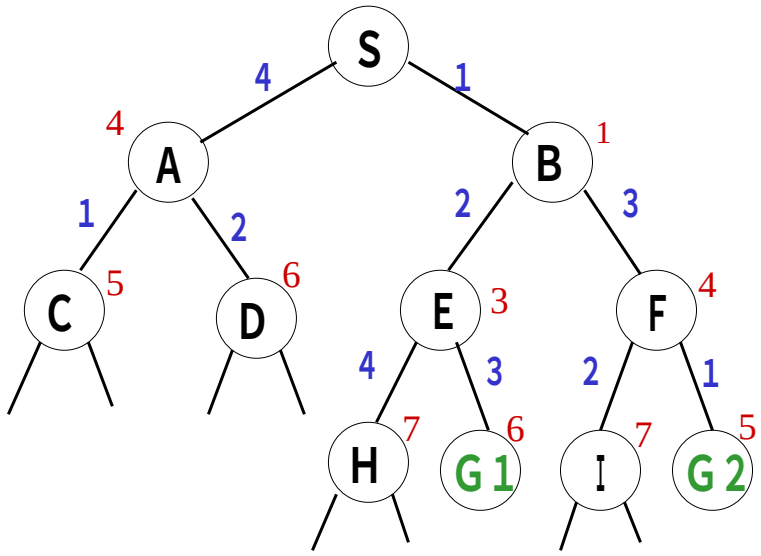
最適経路の探索



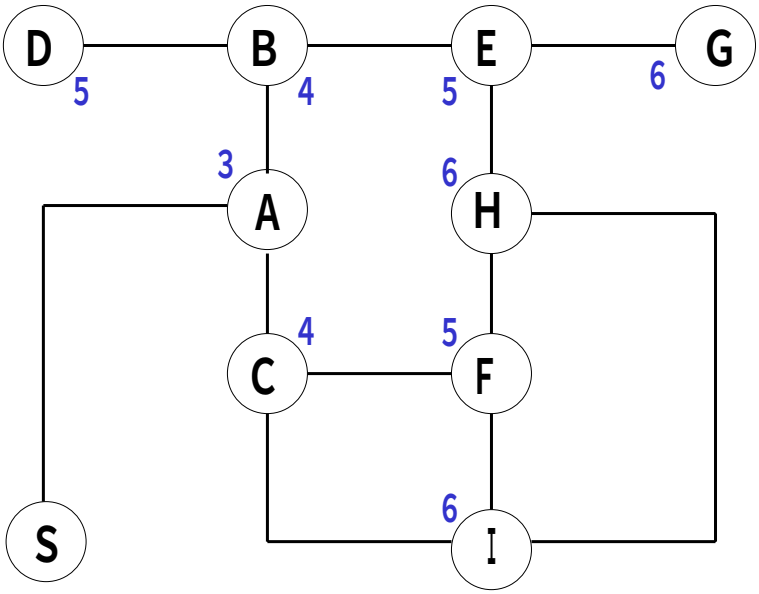
procedure optimal tree search

- add (s, **open)**; g^(s):= 0;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**);
- Expand(n);
- Put all child nodes into **open**
  and attach pointers to n;
- Sort nodes in **open** in increasing order of g^;
- Go to LOOP.

最適経路の探索



グラフ最適経路の探索

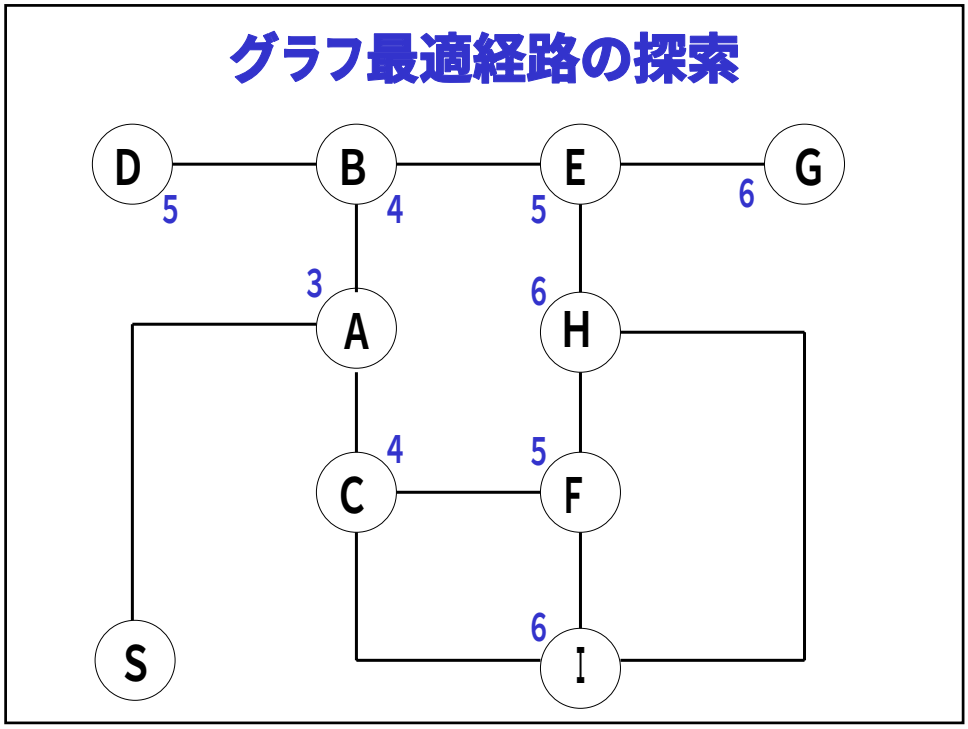


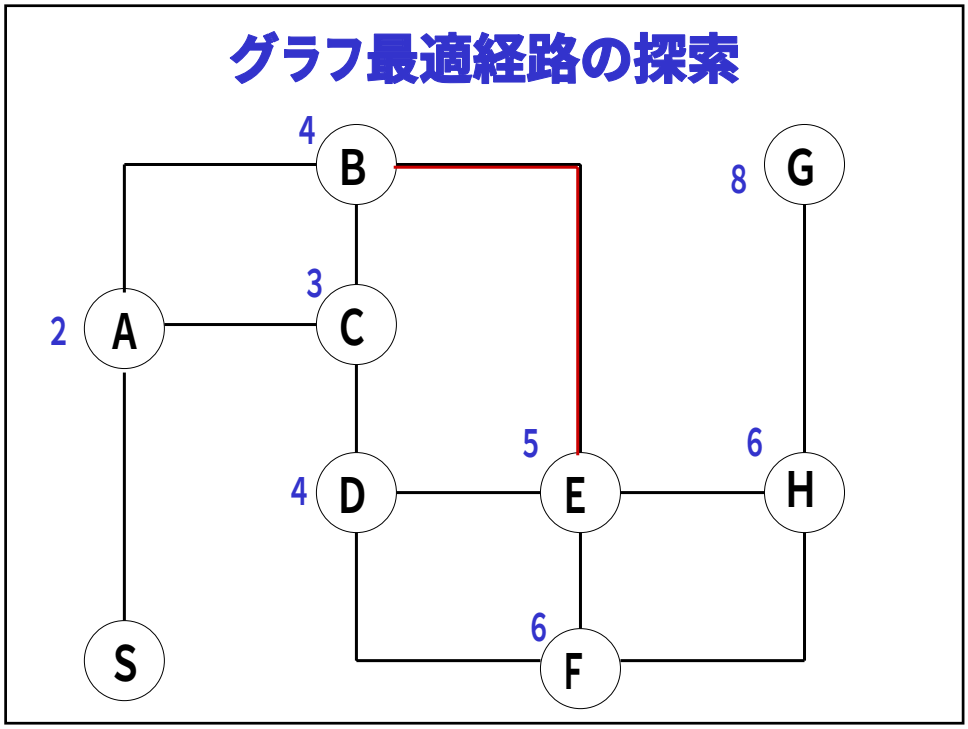
procedure optimal ~~tree~~ search

- add (s, **open)**; g^(s):= 0;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed**)
- Expand(n);
- Put all child nodes into **open**
  and attach pointers to n;
- Sort nodes in **open** in increasing order of g^;
- Go to LOOP.

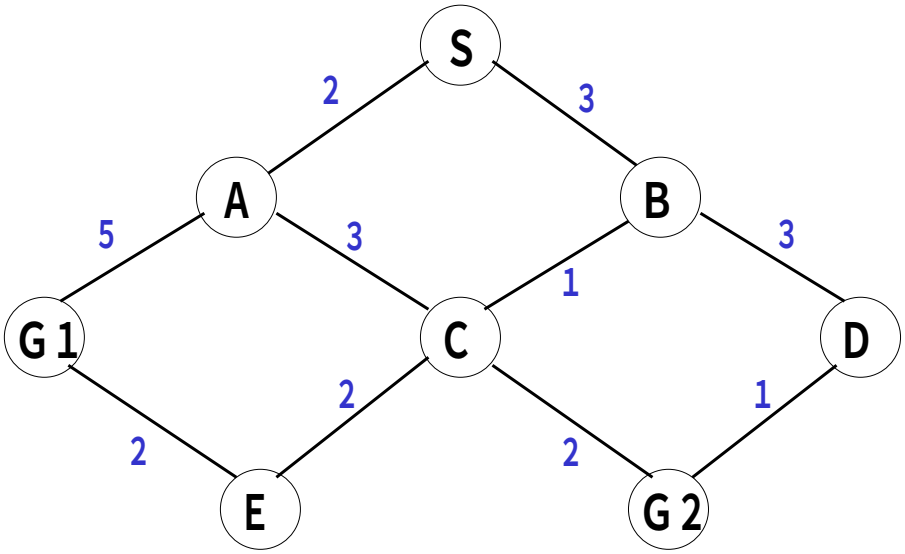
procedure optimal ~~tree~~ search

- add (s, **open)**; g^(s):= 0;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed)**
- Expand(n);
- Put all child nodes into **open**, and attach pointers to n;
  which are not in **open** nor **closed.**
  If child node j was in **open** and g^(n, j) < g^(j),
  then attach pointers to n;
- Sort nodes in **open** in increasing order of g^;
- Go to LOOP.

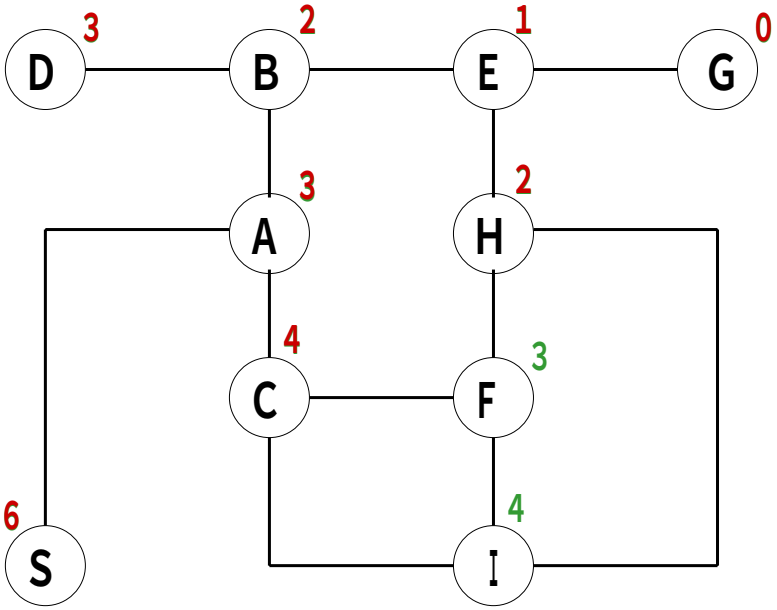

グラフ最適経路の探索
D
B
E
G
5
4
5
6
3
6
A
H
4
5
C
F
S
6
I


グラフ最適経路の探索
4
B
8
G
3
2
A
C
5
6
4
D
E
H
6
S
F

ゴールをopenに入れる例



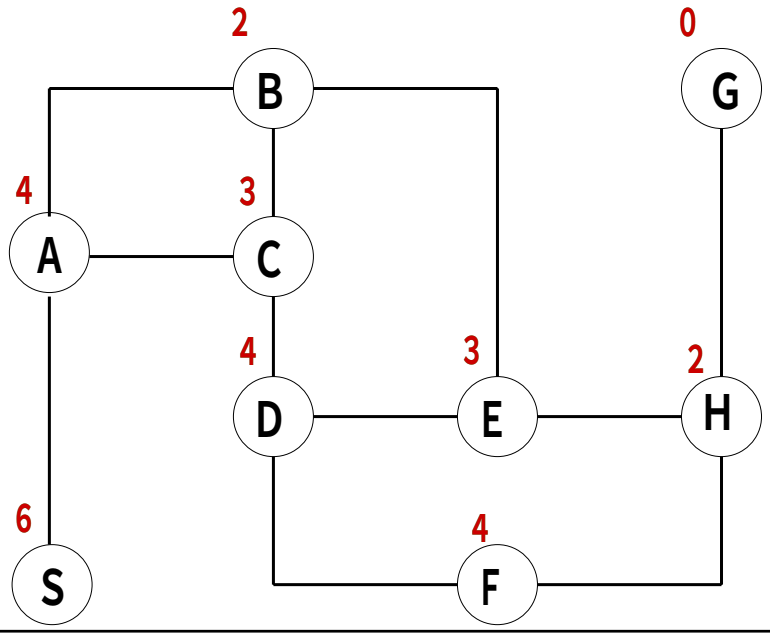
グラフの最良優先探索



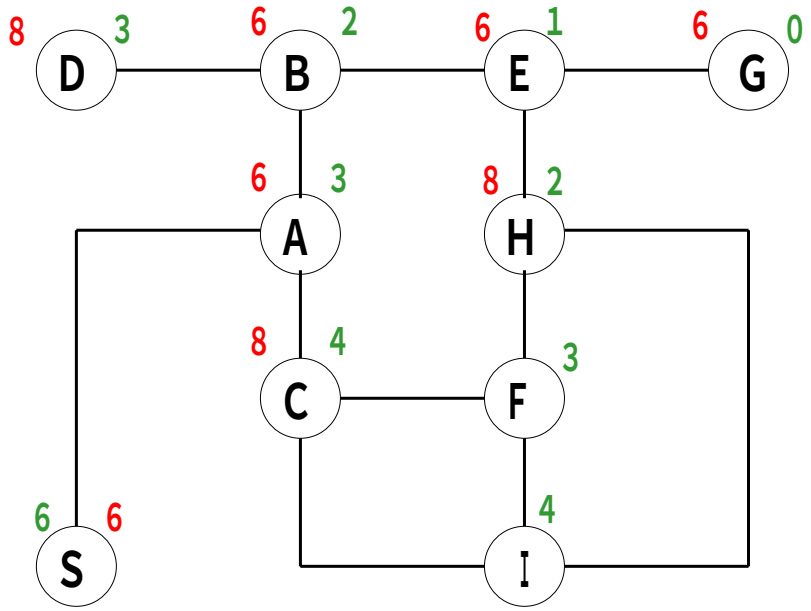
procedure best-first search

- add (s, **open)**;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed)**
- Expand(n);
- Put all child nodes except in **open** or **closed** into **open**,
  Sort nodes in **open** in increasing order of h^ ;
- Go to LOOP;

グラフの最良優先探索



A*アルゴリズム



procedure optimal ~~tree~~ search

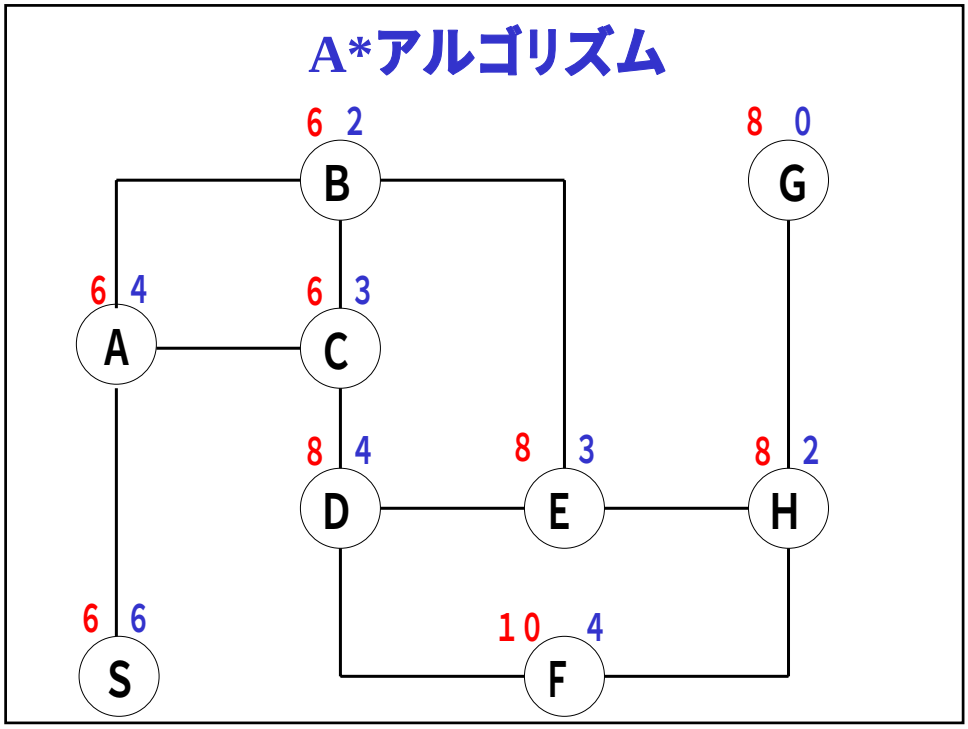
- add (s, **open)**; g^(s):= 0;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed)**
- Expand(n);
- Put all child nodes into **open**,
  which are not in **open** nor **closed.**
  If child node j was in **open**, select min{g^(j), g^(n, j)},
  and attach pointers to n;
- Sort nodes in **open** in increasing order of g^;
- Go to LOOP.

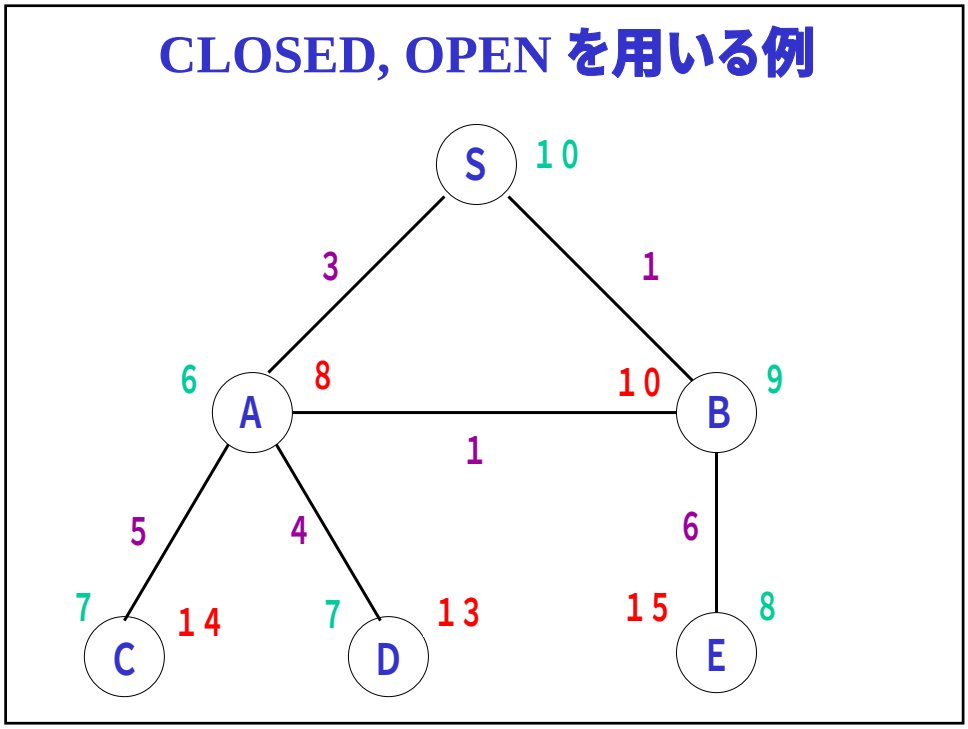
procedure optimal search

- add (s, **open)**; g^(s):= 0;
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed)**
- Expand(n);
- Put all child nodes except in **open** or **closed** into **open**, compute g, and attach pointers to n;
  If child node j was in **open**, select min{g^(j), g^(n, j)}, and attach pointers to n;
- Sort nodes in **open** in increasing order of g^;
- Go to LOOP.

procedure A-algorithm search

- add (s, **open)**; f^(s):=h^(s);
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed)**
- Expand(n);
- Put all child nodes except in **open** or **closed** into **open**, compute g;
  f^(n, j):= g^(n, j) + h^(n); if node j is in **open** or **closed**, select min{f^(j), f^(n, j)}; and put into **open**;
- Sort nodes in **open** in increasing order of f^;
- Go to LOOP.


A*アルゴリズム
6 2
B
8 0
G
6 4
A
6 3
C
8 4
D
8 3
E
8 2
H
6 6
S
10 4
F


CLOSED, OPEN を用いる例
S
10
3
1
6
A
8
10
B
9
1
5
4
6
7
C
14
7
D
13
15
E
8

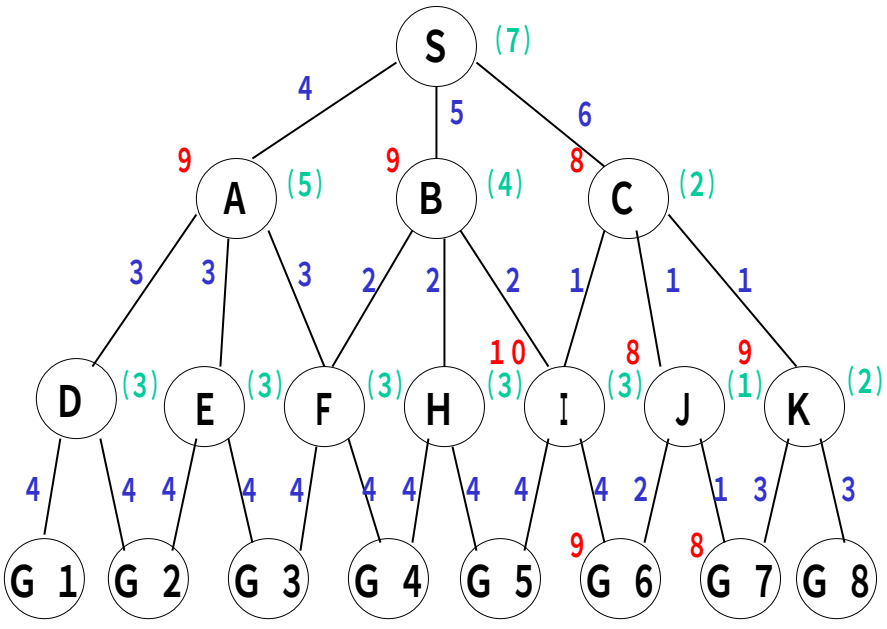
procedure A-algorithm search

- add (s, **open)**; f^(s):= h^(s);
- LOOP: if **open** = null then exit(fail);
- n:= first(**open**);
- If goal(n) then exit (success);
- Remove(n, **open**); add(n, **closed)**
- Expand(n);
- Put all child nodes except in **open** or **closed** into **open**, compute g;
  f^(n, j):= g^(n, j) + h^(n); if node j is in **open** or **closed**, select min{f^(j), f^(n, j)} and put into **open**;

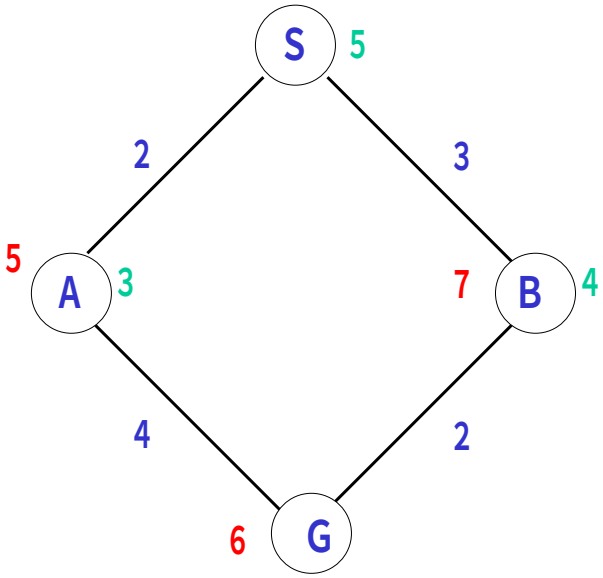
Sort nodes in **open** in increasing order of f^;

- Go to LOOP;

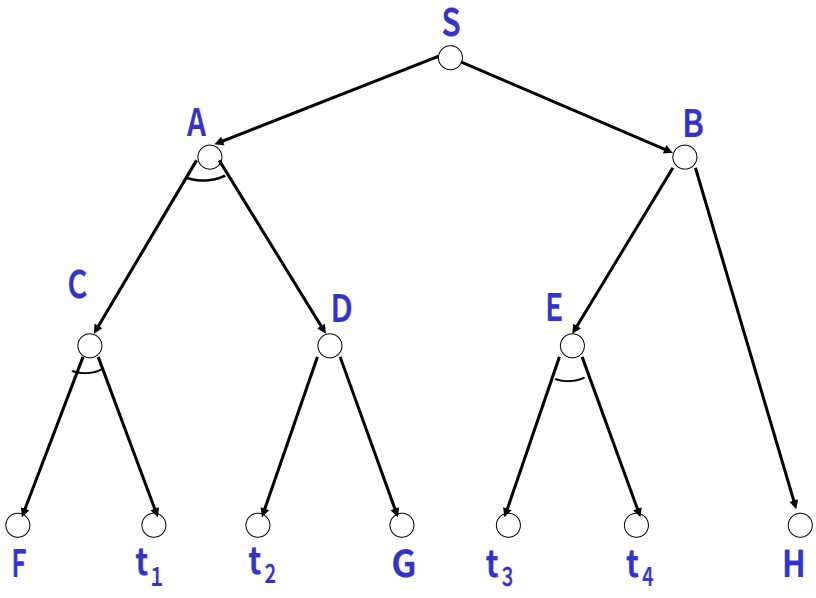
A*アルゴリズムが有効な例

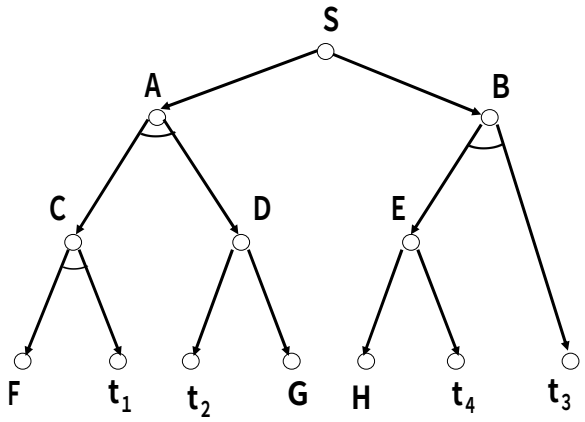


A*アルゴリズムとならない例



AND-OR木の探索



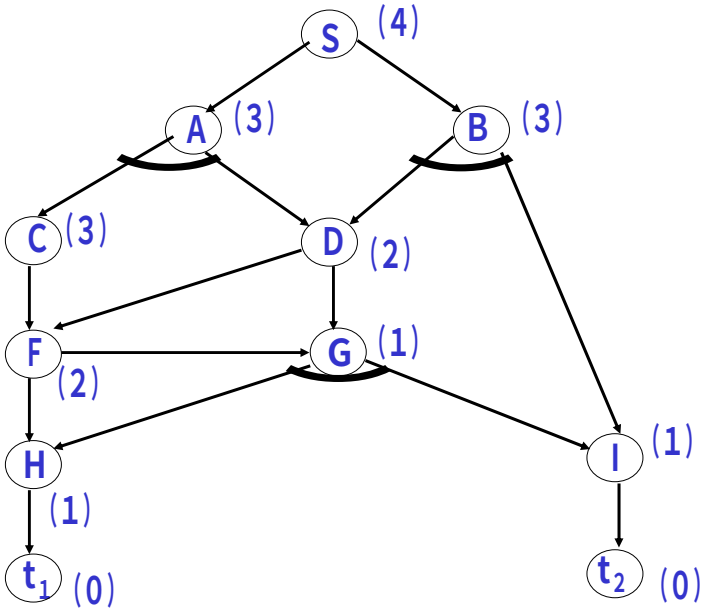


procedure depth-first and-or

- add (s, **open)**;
- LOOP: if **open** = null then exit(fail);
- p:= first(**open**);
- If solved(p) then exit (success);
- Remove(p, **open**);
- Expand(p);
- Put all partial graphs except unsolved one into the top of **open**;
- Go to LOOP.

procedure optimal depth-first and-or

- add (s, **open)**; f(s):=0
- LOOP: if **open** = null then exit(fail);
- p:= first(**open**);
- If solved(p) then exit (success);
- Remove(p, **open**);
- Expand(p);
- Put all partial graphs except unsolved one into the top of **open**;
- Sort nodes in **open** in increasing order of f(p);
- Go to LOOP.



教科書の修正（HPに掲載）

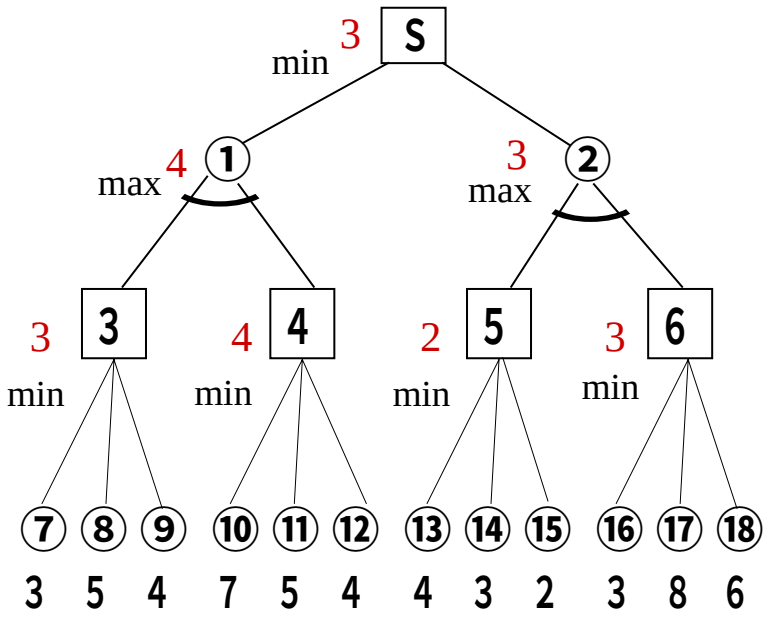
教科書の修正（2つ）
1) p.33 の例3.1を2箇所

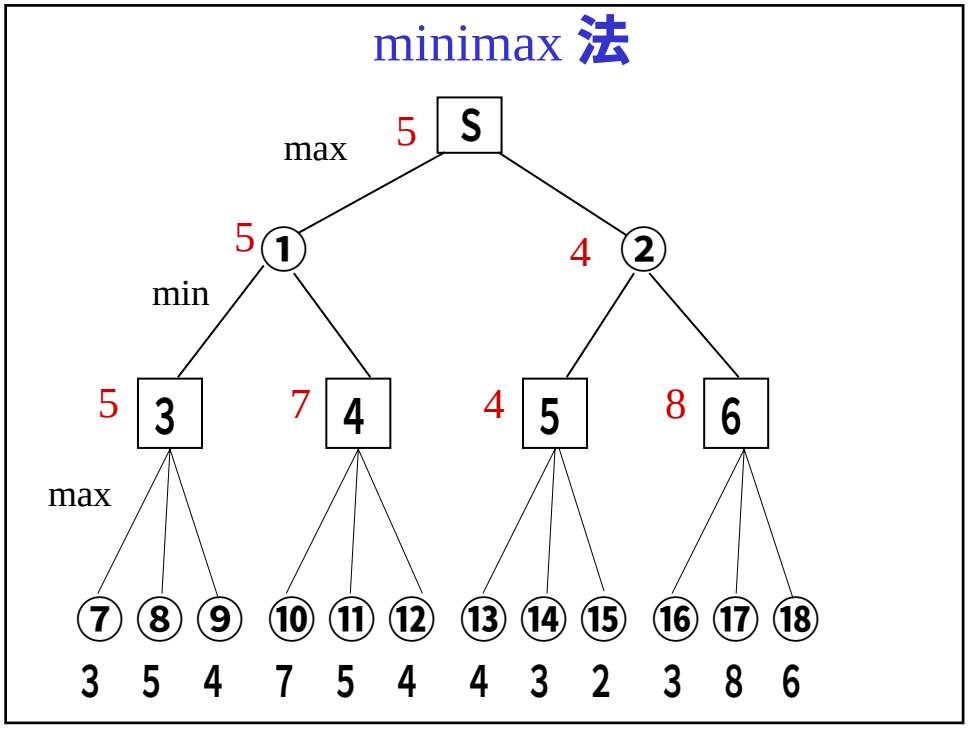
例3.1 この手続きを図3.3AND/ORグラフに適用して時のopenリスト内の部分解グラフを表3.1に示す。3回目の繰返しのときに、...のところ、3回目を4回目に

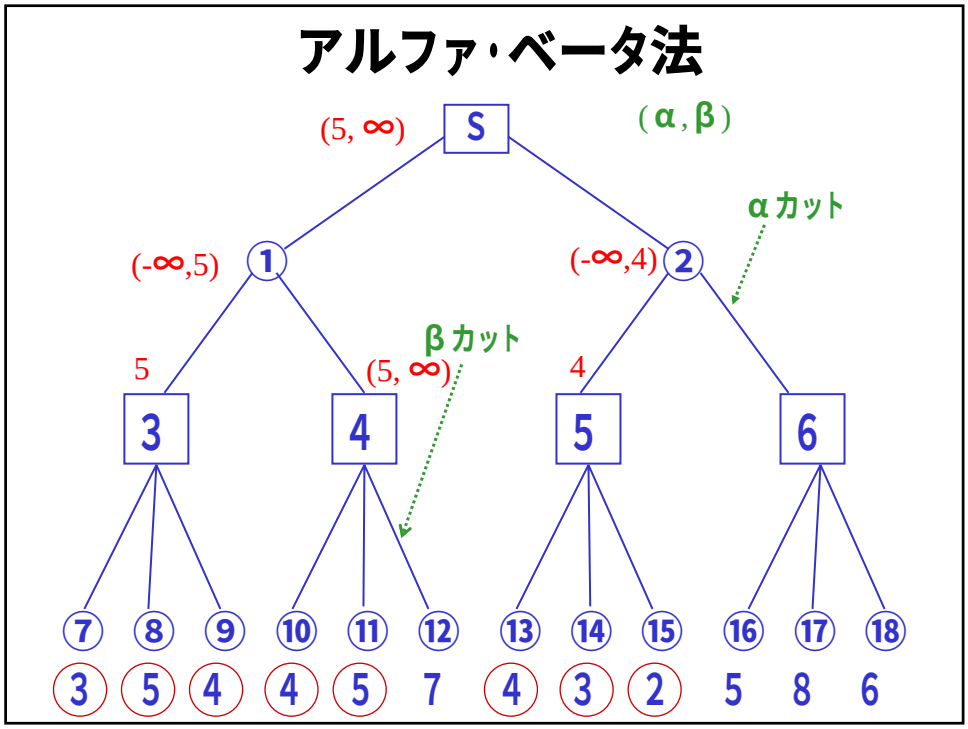
・・・
フはopenに入れられない。6回目の...の6回目を7回目に

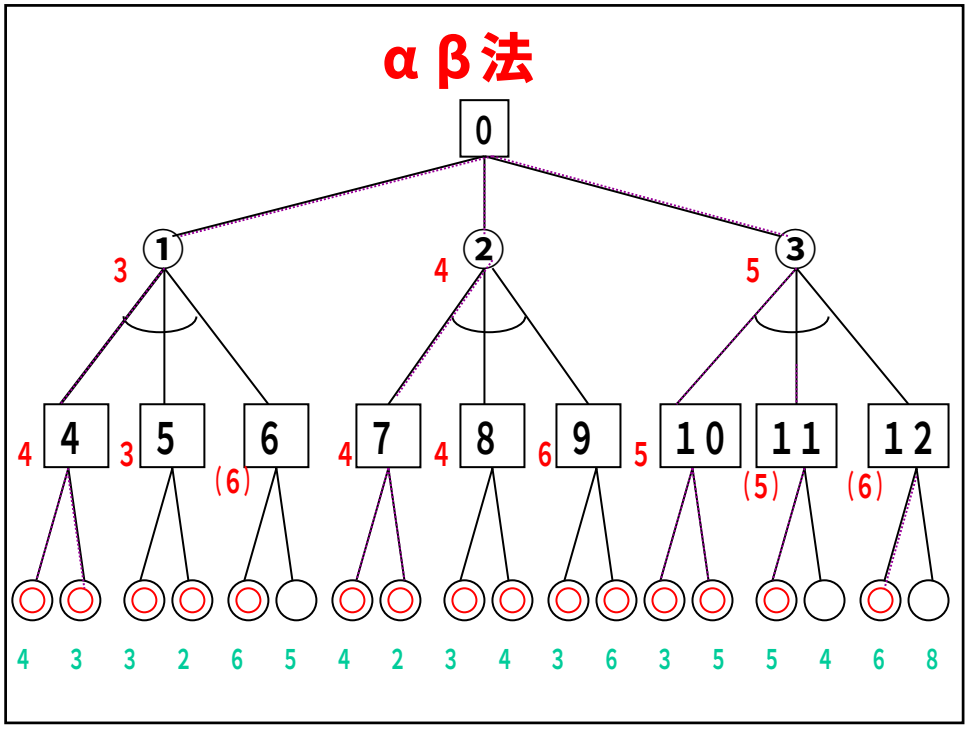
2) 表3.1は、correction に修正。

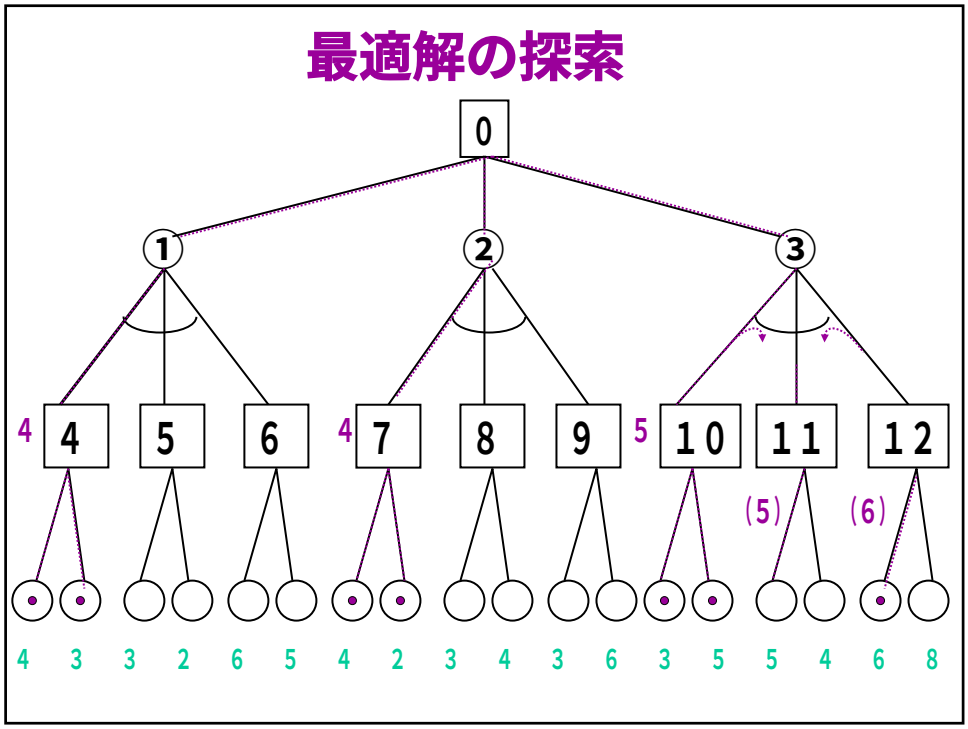
道のコストがない場合

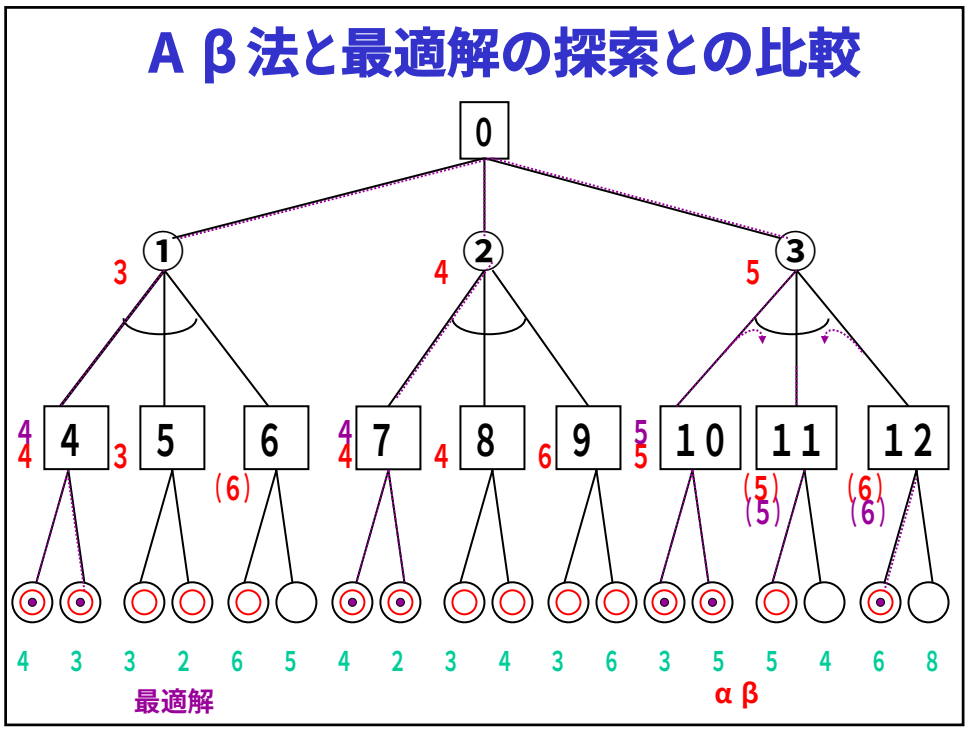


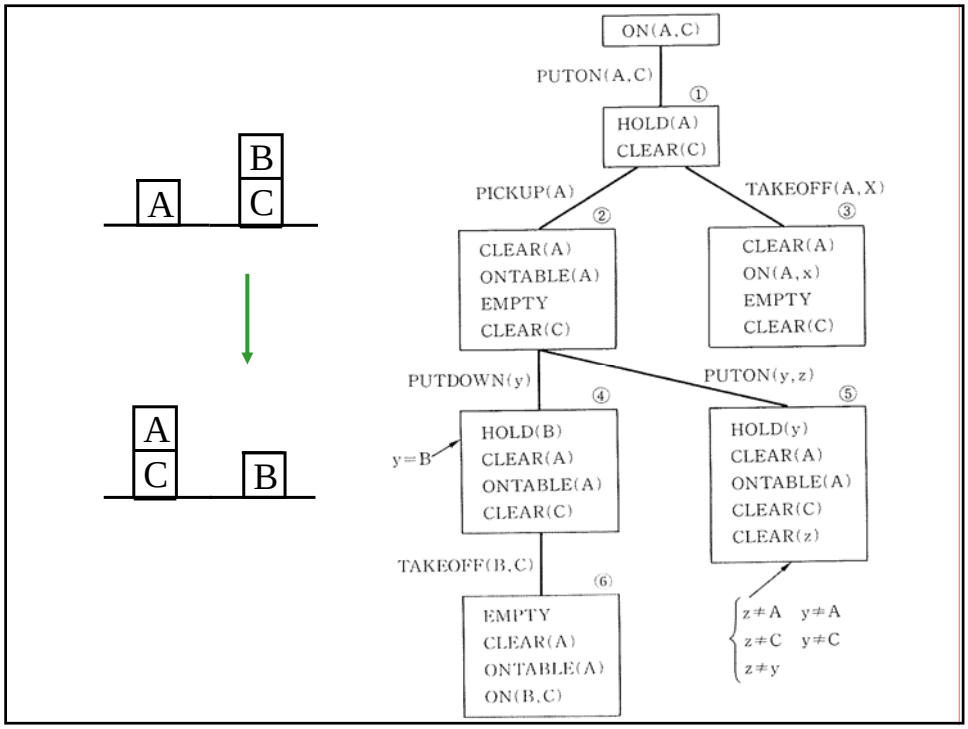

minimax 法
max
5
S
5
1
4
2
min
5
3
7
4
4
5
8
6
max
7 8 9 10 11 12 13 14 15 16 17 18
3 5 4 7 5 4 4 3 2 3 8 6


アルファ・ベータ法
(5, ∞)
S
(α, β)
α カット
(-∞,5)
1
(-∞,4)
2
β カット
5
(5, ∞)
4
3
4
5
6
7 8 9 10 11 12 13 14 15 16 17 18
3 5 4 4 5 7 4 3 2 5 8 6


αβ法
0
1
2
3
3
4
5
4
5
6
7
8
9
10
11
12
4
3
(6)
4
4
6
5
(5)
(6)
4 3 3 2 6 5 4 2 3 4 3 6 3 5 5 4 6 8


最適解の探索
0
1
2
3
4
5
6
7
8
9
10
11
12
4
4
5
(5)
(6)
4 3 3 2 6 5 4 2 3 4 3 6 3 5 5 4 6 8


Αβ法と最適解の探索との比較
0
1
2
3
4
5
6
7
8
9
10
11
12
4 3 3 2 6 5 4 2 3 4 3 6 3 5 5 4 6 8
最適解
αβ


A
B
C
A
C
B
ON(A,C)
PUTON(A,C)
①
HOLD(A)
CLEAR(C)
PICKUP(A)
②
TAKEOFF(A,X)
③
CLEAR(A)
ONTABLE(A)
EMPTY
CLEAR(C)
CLEAR(A)
ON(A,x)
EMPTY
CLEAR(C)
PUTDOWN(y)
④
PUTON(y,z)
⑤
y=B
HOLD(B)
CLEAR(A)
ONTABLE(A)
CLEAR(C)
HOLD(y)
CLEAR(A)
ONTABLE(A)
CLEAR(C)
CLEAR(z)
TAKEOFF(B,C)
⑥
EMPTY
CLEAR(A)
ONTABLE(A)
ON(B,C)
z≠A y≠A
z≠C y≠C
z≠y

D_1：さるの位置の差
D_2：箱　〃　〃
D_3：バナナ〃　〃

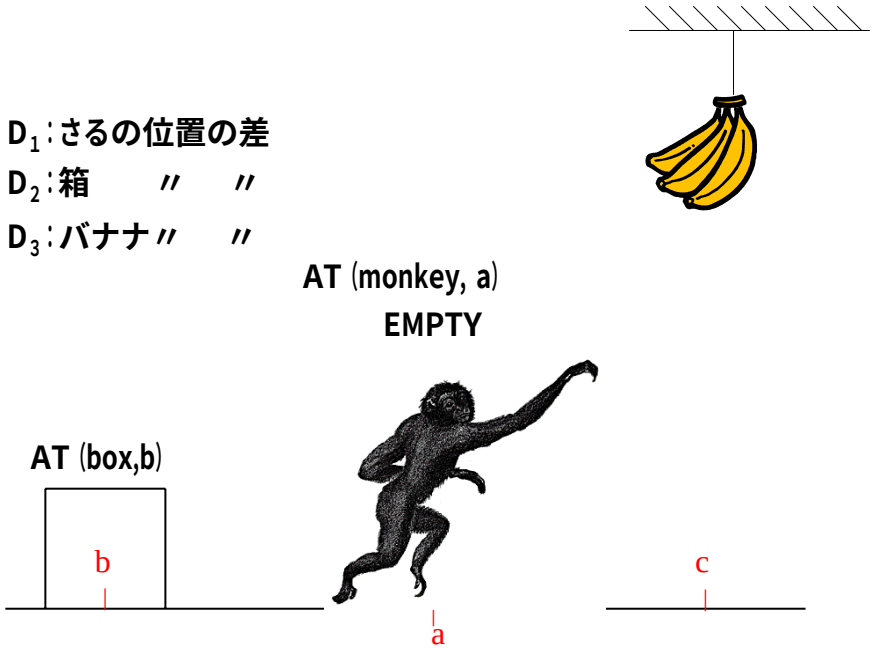
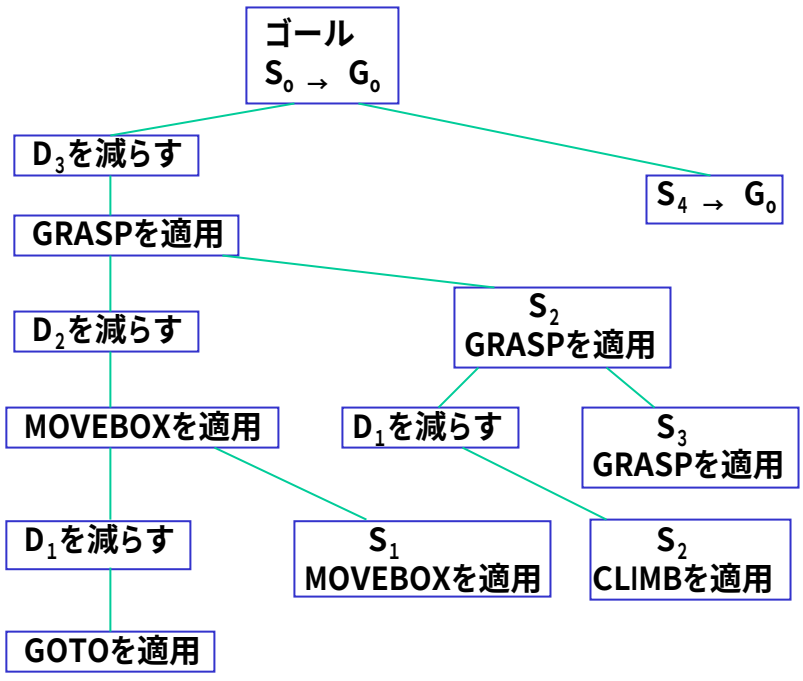


表4.1　差異と差異を縮めるオペレータ

差異＼オペレータ	GOTO	MOVE BOX	CLIMB	GRASP
D_1 (猿の位置)	○	○	○	
D_2 (箱の位置)		○		
D_3 (猿の手の状態)				○

オペレータ

- GOTO(u)
- MOVEBOX(v)
  - **前提条件**： AT(monkey, x), AT(box, x)
- CLIMB
  - **前提条件**： AT(monkey, x), AT(box, x)
- GRASP
  - **前提条件**： AT(box, c), ON(monkey, box)



recursive **procedure** GPS(S,G)

1 LOOP1: **if S satisfy G,** then exit(S)
2 G と S の差異をすべて求め、最も重要な差異をdとする
3 d を減らすオペレータをすべてoplistに入れる
3 LOOP2: **if** empty(oplist) **then** return(fail)
5 operator:= first(oplist), remove(operator, oplist)
pc:= operator's precondition
if pc does not decrease the difference, **then goto** LOOP2
6 **if** pc= null **then goto** APPLY
7 S1:= GPS(S, pc)
8 **if** S1= fail **then goto** LOOP2
9 APPLY: S:= operator(S1)
10 **goto** LOOP1

階層的計画のオペレータ

- GOTO(u)
- MOVEBOX(v)
  - **前提条件**: (1)AT(monkey, x)
  - (3)AT(box, x)
- CLIMB
  - **前提条件**: (1)AT(monkey, x)
  - (3)AT(box, x)
- GRASP
  - **前提条件**: (3)AT(box, c)
  - (2)ON(monkey, box)

階層的計画のオペレータ

- GOTO(u)
- MOVEBOX(v)
  - **前提条件**： (3)AT(box, x)
- CLIMB
  - **前提条件**： (3)AT(box, x)
- GRASP
  - **前提条件**： (3)AT(box, c)

計画

MOVEBOX(c)

GRASP

階層的計画のオペレータ

- GOTO(u)
- MOVEBOX(v)
  - **前提条件**： (3)AT(box, x)
- CLIMB
  - **前提条件**： (3)AT(box, x)
- GRASP
  - **前提条件**： (3)AT(box, c))
  - (2)ON(monkey, box)

計画

MOVEBOX(c)

CLIMB

GRASP

階層的計画のオペレータ

- GOTO(u)
- MOVEBOX(v)
  - **前提条件**: (3)AT(box, x)
  - (1)AT(monkey, x)
- CLIMB
  - **前提条件**: (3)AT(box, x)
  - (1)AT(monkey, x)
- GRASP
  - **前提条件**: (3)AT(box, c)
  - (2)ON(monkey, box)

計画

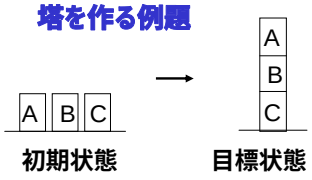
GOTO(b)

MOVEBOX(c)

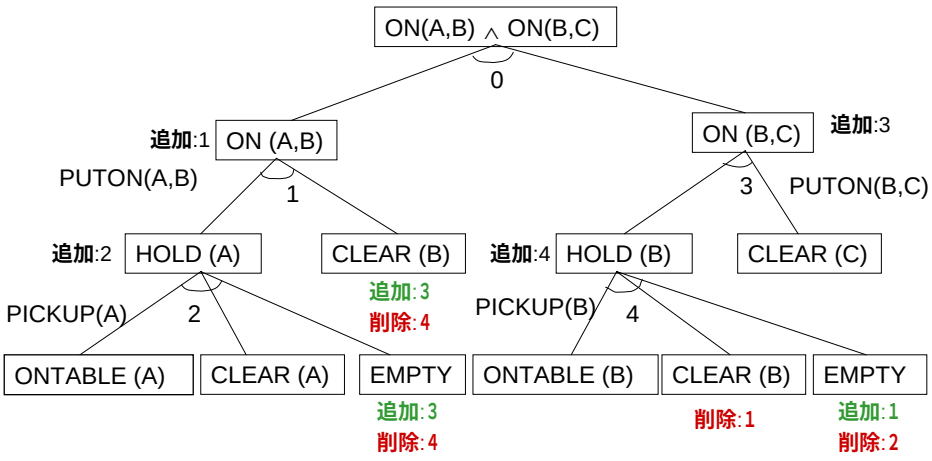
CLIMB

GRASP

塔を作る例題



複数の目標のAND/OR木

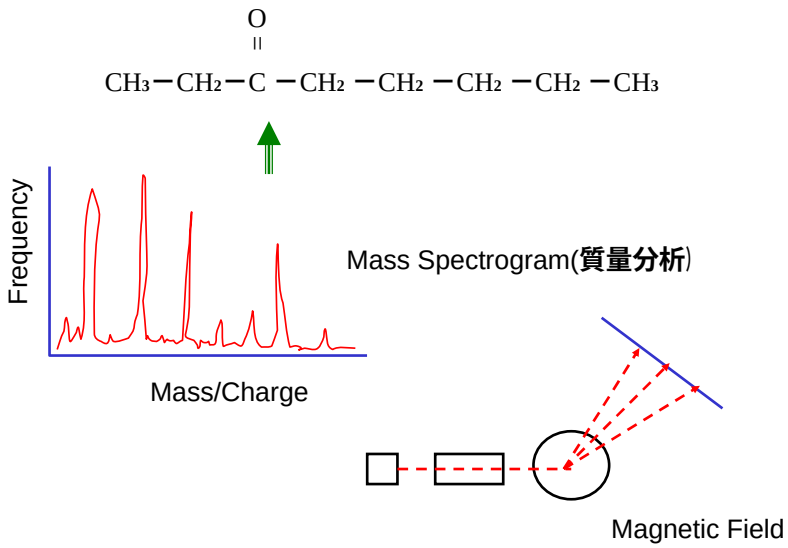


初期のAIの歴史

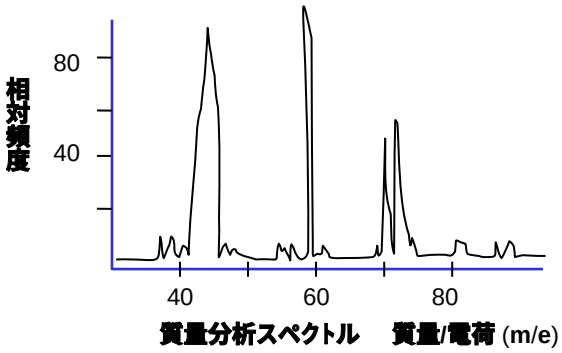
- 1956 AIの研究会（Minsky, Shannon, Newell, Simonなど）
- 1959 AI Lab. 設立（Minsky, McCarthy）
- 1961 AI へのステップ（Minsky）
- 1963 Computer and Thought(Feigenbaumなど)

リンダーバーグ(遺伝学)

知識工学の誕生（1970-1979）



DENDRAL



R1：もし43に高いピークがあり、71に高いピークがあり、58になんらかのピークがあるならば、 Nプロピルケトン3を含む。

R2：もし43に高いピークがあり、71に高いピークがあり、58にピークがなければ、 イソプロピルケトン3を含む。

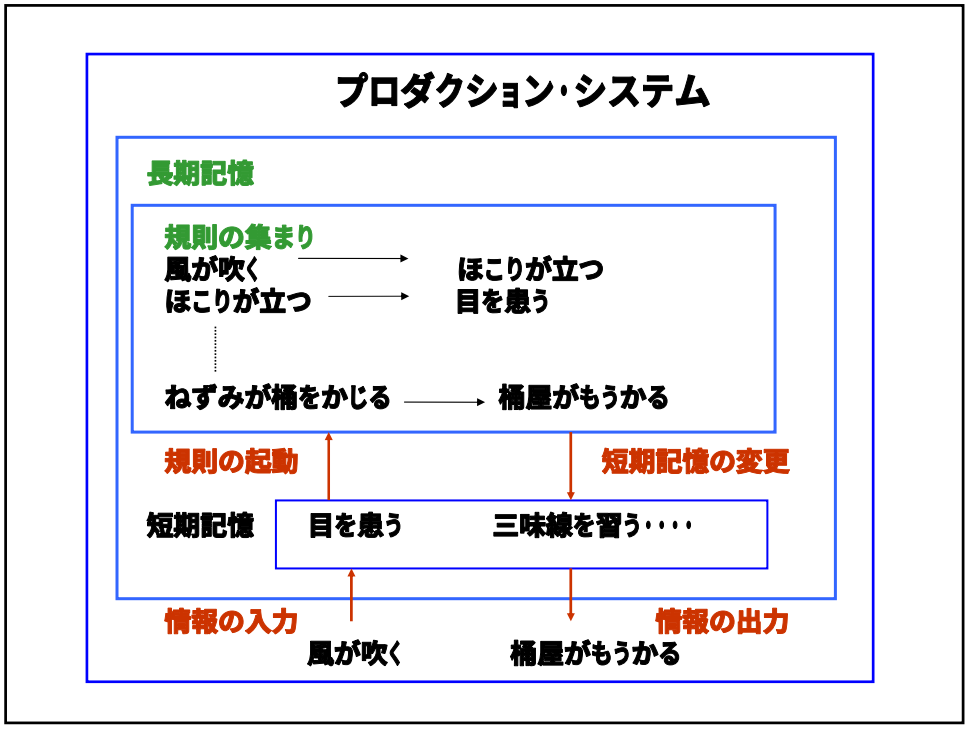
1975 ACM Turing Award Lecture

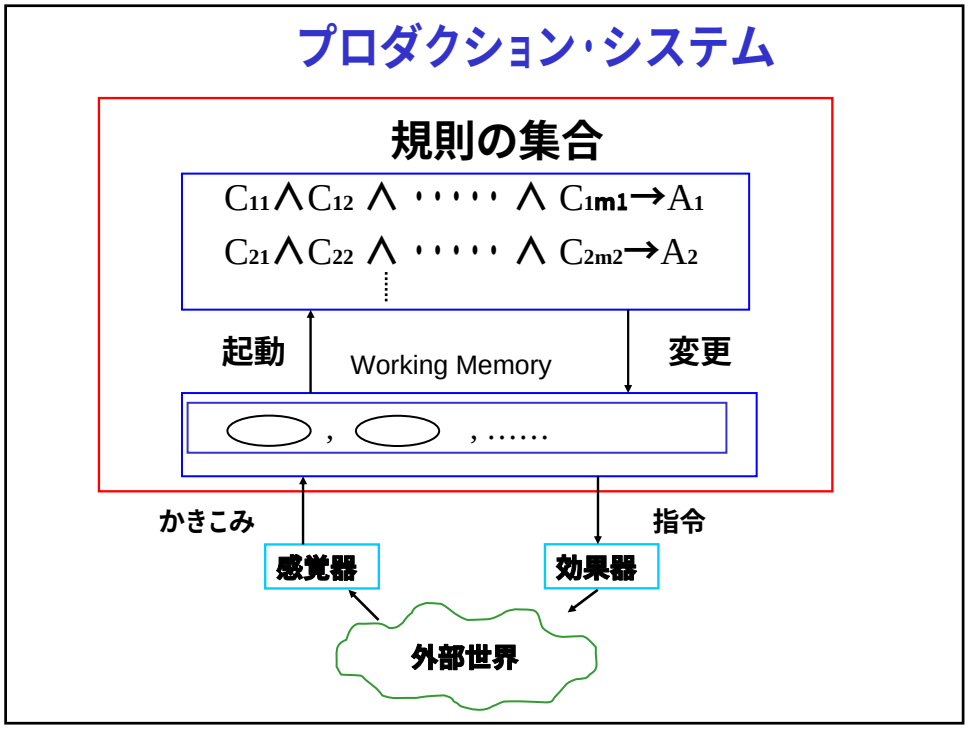
Newell

Simon

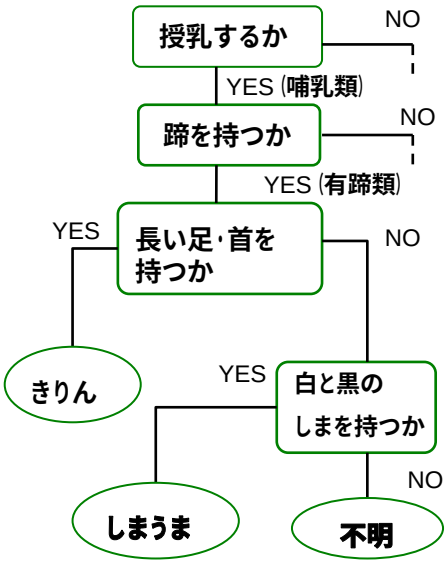
Computer Science as Empirical Inquiry: Symbols and Search

Principal instigators of the idea that human cognition can be described in terms of a symbol system, and they have developed detailed theories for human problem solving verbal learning and inductive behavior using computer programs.


プロダクション・システム
長期記憶
規則の集まり
風が吹く
ほこりが立つ
ほこりが立つ
目を患う
ねずみが桶をかじる
桶屋がもうかる
規則の起動
短期記憶の変更
短期記憶
目を患う
三味線を習う・・・・
情報の入力
情報の出力
風が吹く
桶屋がもうかる

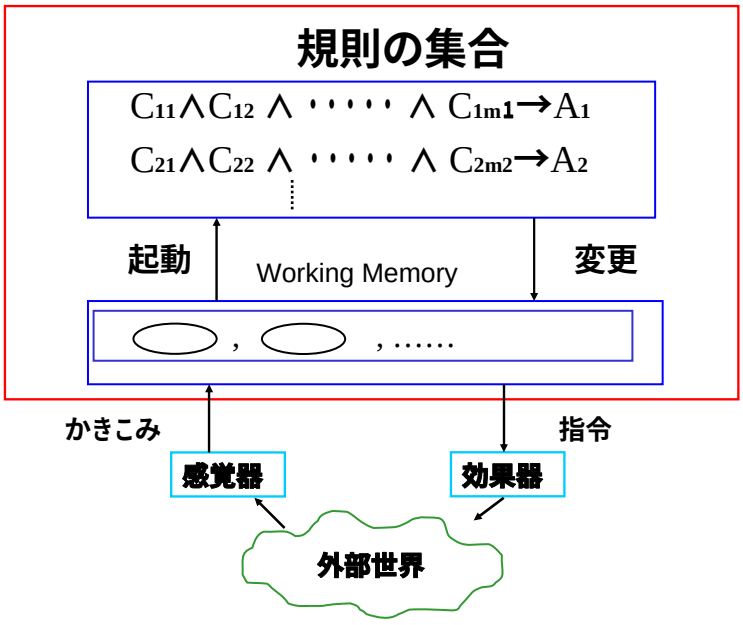

プロダクション・システム
規則の集合
C11∧C12 ∧ ・・・・・ ∧ C1m1→A1
C21∧C22 ∧ ・・・・・ ∧ C2m2→A2
起動
Working Memory
変更
, , ……
かきこみ
指令
感覚器
効果器
外部世界

プロダクションシステムと従来プログラムとの比較



・もし動物が授乳するならば、
　ほ乳類である。
・もし動物がほ乳類で、蹄を持てば
　有蹄類である。
・もし動物が有蹄類で、
　長い足と長い首を持てば
　きりんである。
・もし動物が有蹄類で、
　白と黒のしまを持てば
　しまうまである。
・・・

プロダクション・システム



Conflict Resolution

- 1. **最初に照合した規則**
- 2. **最も厳しい前提条件**
- 3. **最近**
- 4. **最近使われた変数をもつ**
- 5. **実行時に評価**

車の故障診断の例

```
((スタート1
    ((キー、音あり)(エンジン　×))((エンジンまわり)))
(　スタート2
    ((キー、音なし)(レバー　×))((レバー位置直せ)(終)))
(　スタート3
    ((キー、音なし)))((電気系統)))
(　電気1
    ((電気系統)(機器　×))((ヒューズ)(終)))
(エンジン1
    ((エンジンまわり)(燃料計　低))((燃料入れよ)(終)))
(エンジン2
    ((エンジンまわり)))((修理店に)(終)))
```

OPS5 by McDemott 1982

表現は三つ組(triplet)形式

もの-属性-値 (object-attribute-value)

(block ^color red ^size 5)

プロダクション規則

(P **規則の名前**

(**条件** 1) … (**条件** n) → (**実行** 1) … (**実行** m))

競合の解消

WMの項目にはタイムタグがつく

同一の項目に照合する同一の規則を2度以上実行できない

最新の項目に照合する規則を優先

複数の項目の場合、タグの新しい順に並べ、順に比較し、
新しい方を優先。項目がつきた規則は捨てる。

OPS5の例

プロダクション規則

```
(P start-count-red
   (subtask ^name count-red-block) ~(count ^value <X>)
 → (make count ^value 0) )
(P count-red
   (subtask ^name count-red-block) (block ^color red)
 (count ^value <X>)
 → (modify 3  ^value (compute <X> +1) )
(P end-count-red
   (subtask ^name count-red-block) (count ^value <X>)
 → (remove 1) )
```

MYCIN　(by Shortliffe)

1973年頃
Harvard大数学科からStanford 大医学部へ
そこからFeigenbaumのところへ来て研究

Mycin の使いすぎを防ぐための
早期診断

後ろ向き推論(MYCIN)

もしC_{11}で, C_{12}でC_{1n} ならば　　A_1 という確からしさは CF_1である

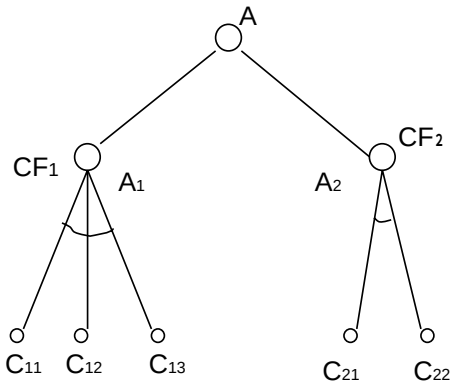
規則の確信度

R_1：$C_{11} \wedge C_{12} \wedge \cdots \wedge C_{1n} \rightarrow A_1(CF_1)$

事実の確信度

Cij　(CFij)

AND/OR木



確からしさ
Certainty Factor
確信度
信頼性係数

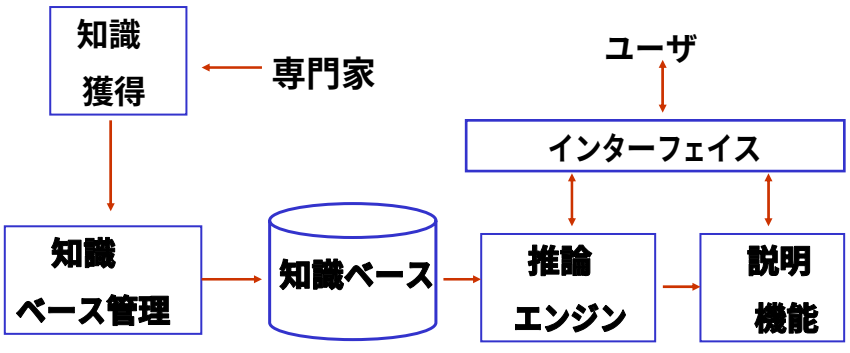
e-Learning の教材 (上野@情報学研究所)

- WebELSにコンテンツが5件アップロードされている。
- 視聴方法は以下の通り:

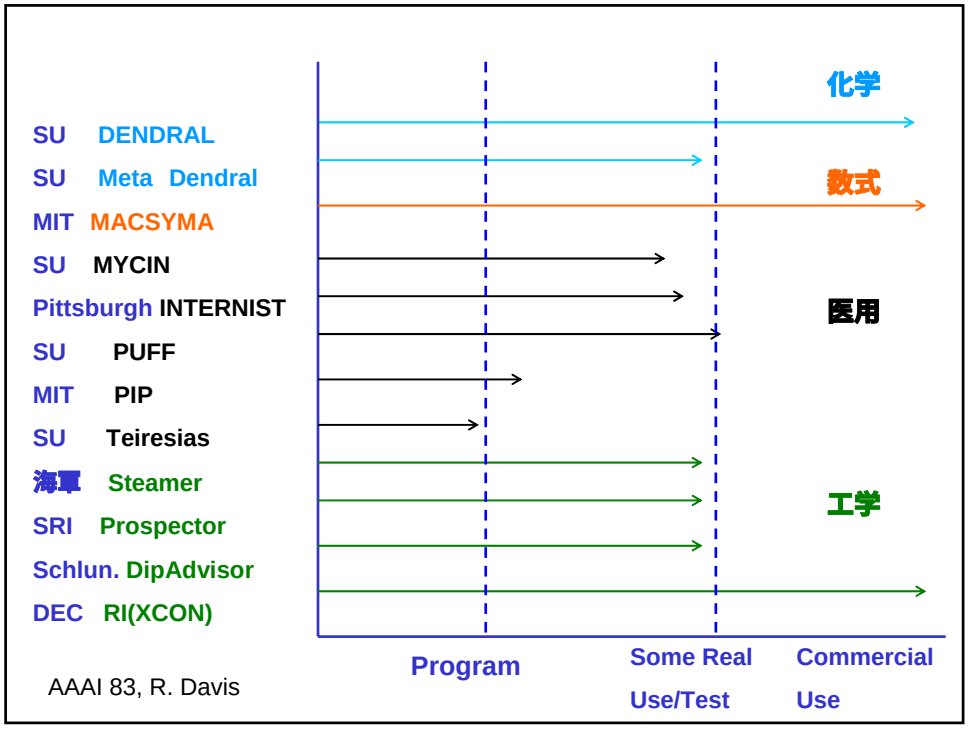
1) Internet Explorerで次ぎのURLにアクセスする。
   http://weblsx.ex.nii.ac.jp/index_jp.html
   WebELSトップページが表示されるので、中央にある「WebELSx」をクリック。
2) WebELSx トップページが表示されるので、「ゲスト・ログイン」をクリック 。
3) 公開コンテンツリストが表示されるので、見たい科目の「見る」をクリック。

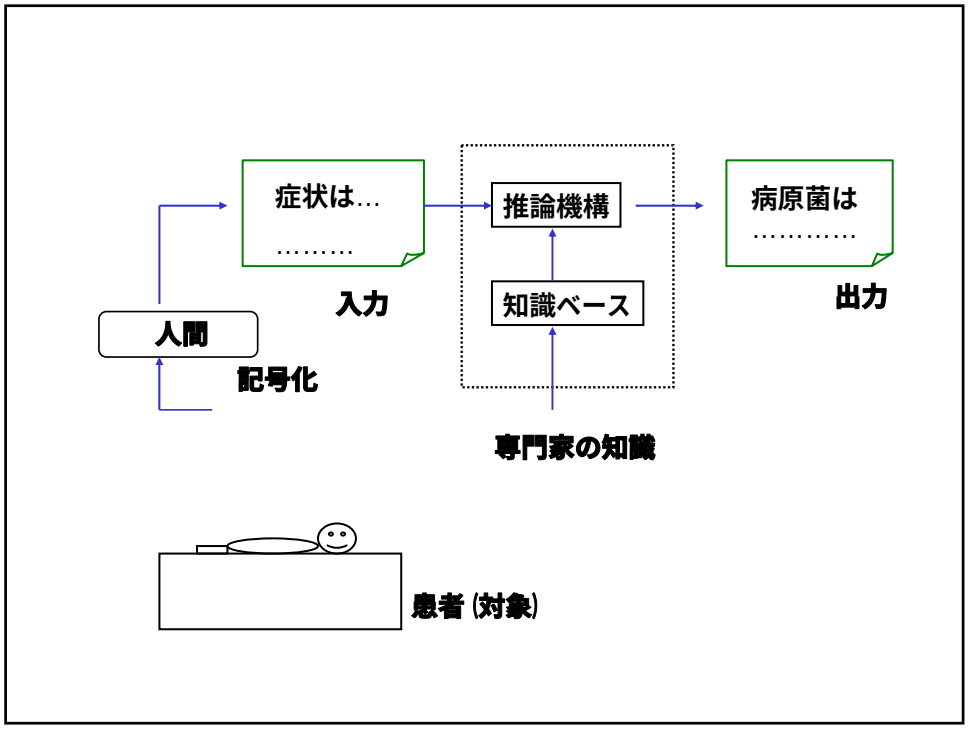
Knowledge Modeling and Reasoning I (情報学専攻 上野)
(AIとは)
Knowledge Modeling and Reasoning II (情報学専攻 上野)
(知識表現と推論)

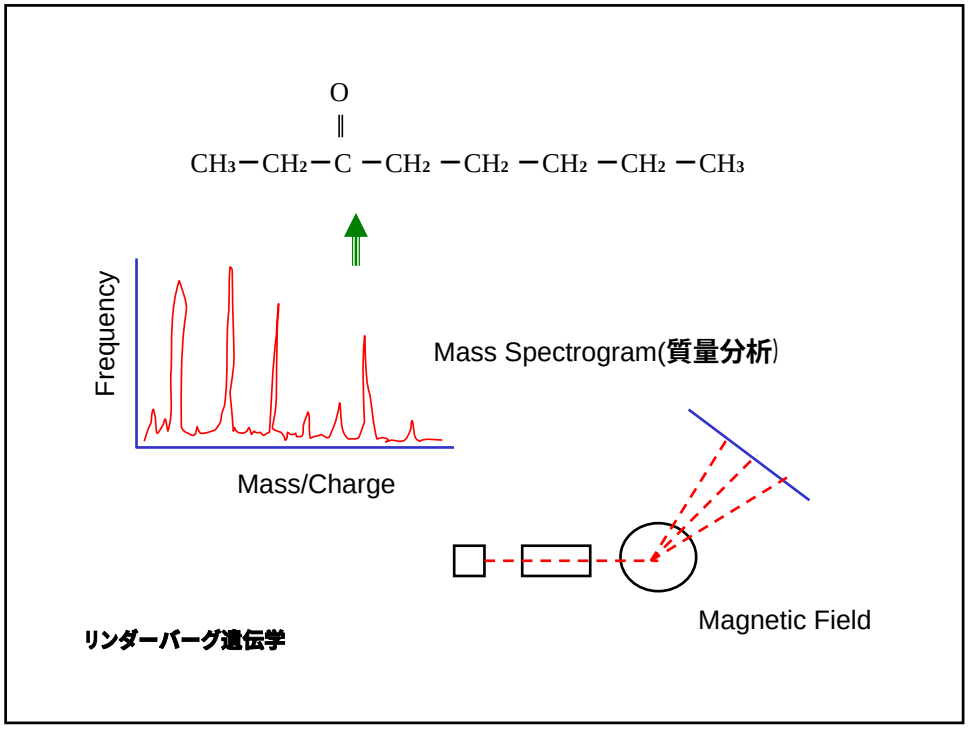
エキスパート・システムの構成

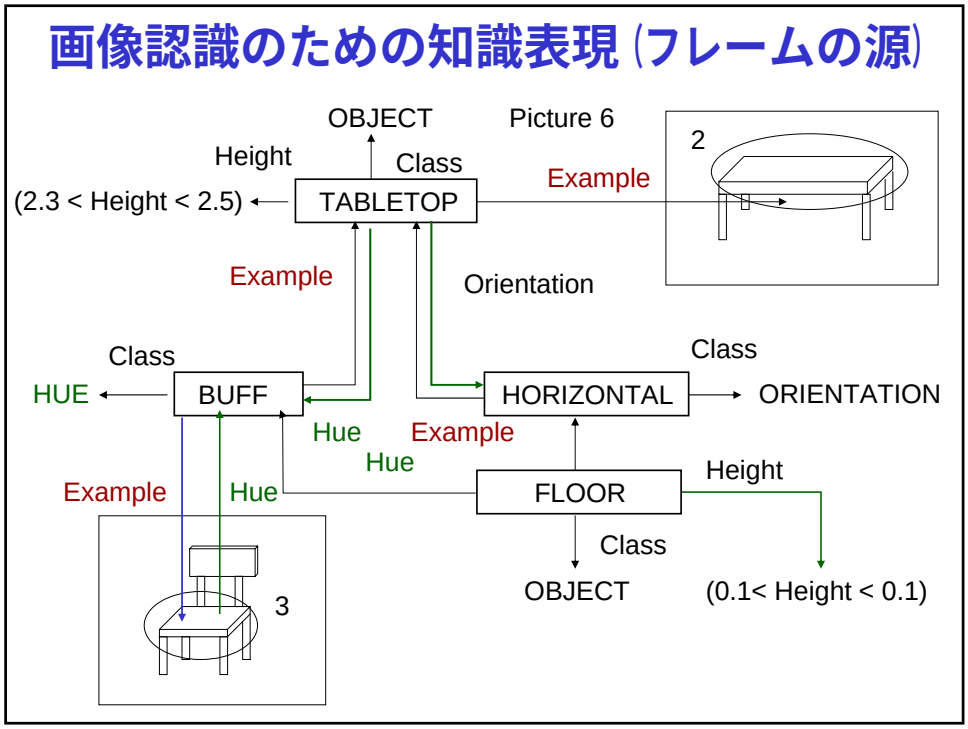


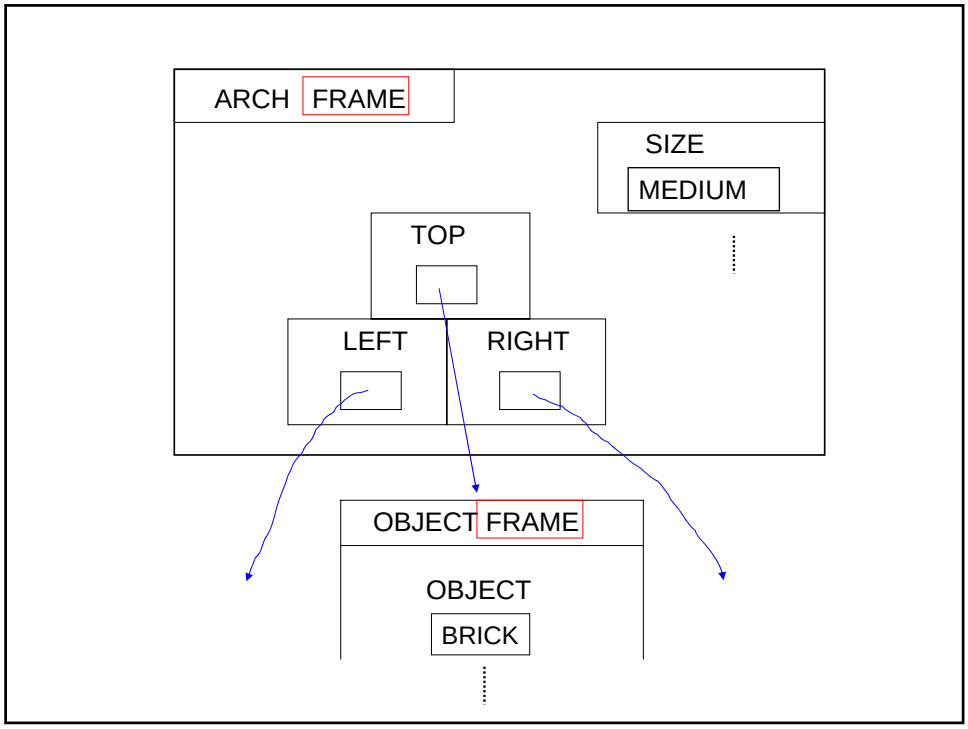
EMYCIN へ


SU DENDRAL
SU Meta Dendral
MIT MACSYMA
SU MYCIN
Pittsburgh INTERNIST
SU PUFF
MIT PIP
SU Teiresias
海軍 Steamer
SRI Prospector
Schlun. DipAdvisor
DEC RI(XCON)
化学
数式
医用
工学
Program
Some Real Use/Test
Commercial Use
AAAI 83, R. Davis


症状は…
……
入力
推論機構
知識ベース
病原菌は
……
出力
人間
記号化
専門家の知識
患者 (対象)


O
CH₃−CH₂−C −CH₂ −CH₂ −CH₂ −CH₂ −CH₃
Frequency
Mass/Charge
Mass Spectrogram(質量分析)
Magnetic Field
リンダーバーグ遺伝学


画像認識のための知識表現（フレームの源）
OBJECT
Picture 6
Height
Class
(2.3 < Height < 2.5)
TABLETOP
Example
2
Example
Orientation
Class
Class
HUE
BUFF
HORIZONTAL
ORIENTATION
Hue
Example
Hue
Example
Hue
Height
FLOOR
Class
OBJECT
(0.1< Height < 0.1)
3



フレーム (Frame)の例

鳥フレーム
動物の一種
足の数 ： 2
羽の数 ： 2
移動法 ： 歩く、飛ぶ

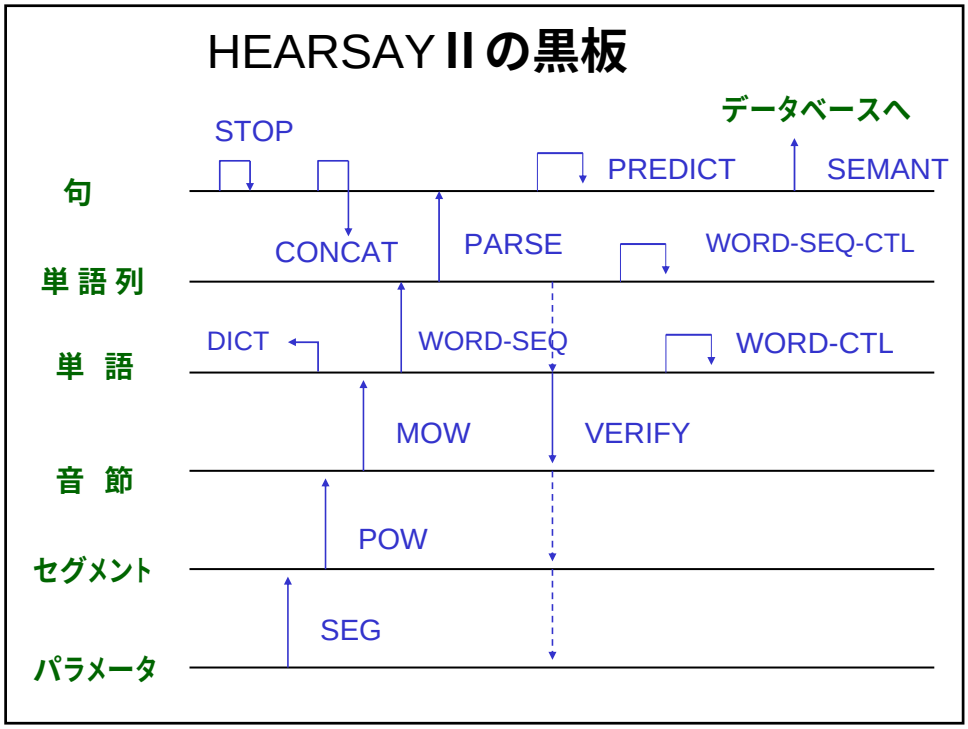
スズメフレーム
鳥の一種
食物 ： 昆虫、穀物
大きさ ： 5～10cm

ペンギンフレーム
鳥の一種
食物 ： 魚
移動法 ： 歩く、泳ぐ

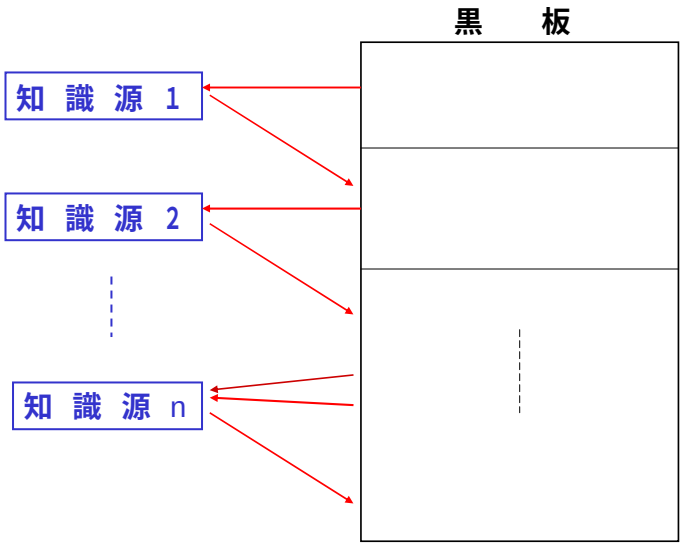
CHAIR Frame
- Specialization-of : FURNITURE
- USE : for sitting
- Number-of-legs : (DEFAULT = 4)
- Number-of-arms: 0, 1, or 2
- Seat-height : [20, 60]
- Back-height : from Seat-position to [70, 120]
- Color-of-seat : (DEFAULT = GRAY)
- Color-of-seat : (DEFAULT = GRAY)

FOLDING-CHAIR Frame
- Specialization-of : CHAIR
- Number-of-legs : 4
- Number-of-legs : 0
- Seat-height : [20, 50]
- Back-height : from Seat-position to [70, 100]

FOLDING-CHAIR-1 Frame
- Specialization-of : FOLDING-CHAIR
- Color-of-seat : BROWN
- Color-of-seat : BROWN
- Seat-height : 45
- Back-height : from Seat-position to 80



黒板モデル

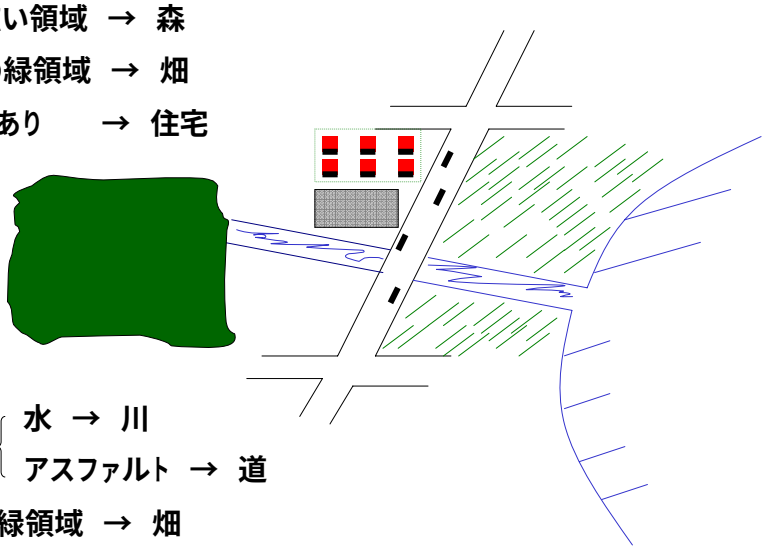


フレームを用いた画像認識

緑濃い広い領域 → 森

長方形の緑領域 → 畑

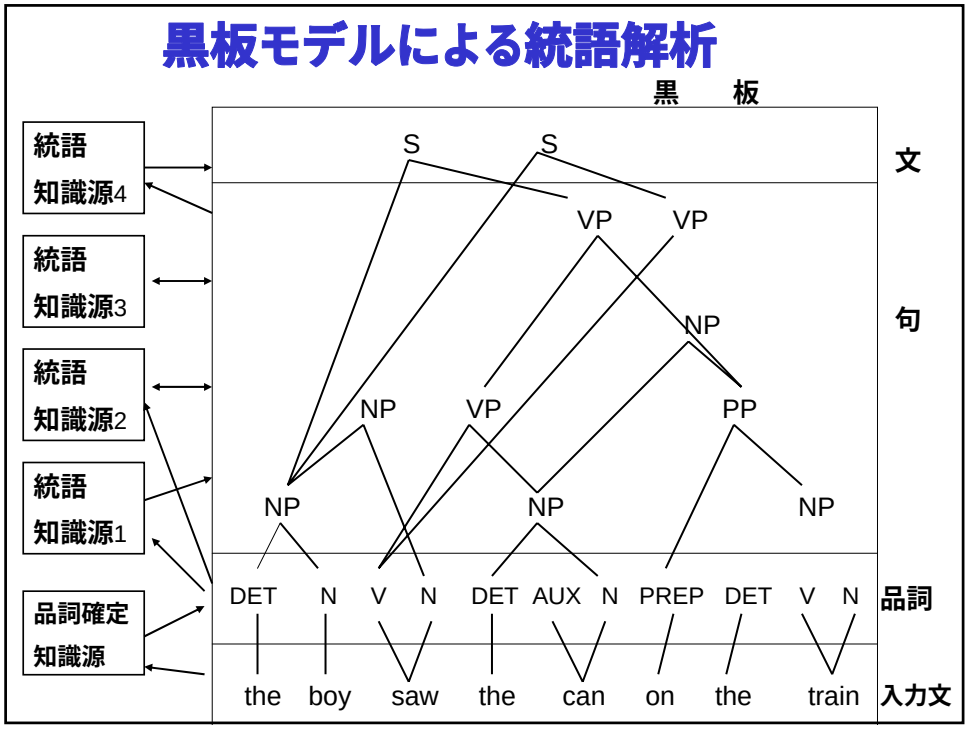
混在、影あり → 住宅

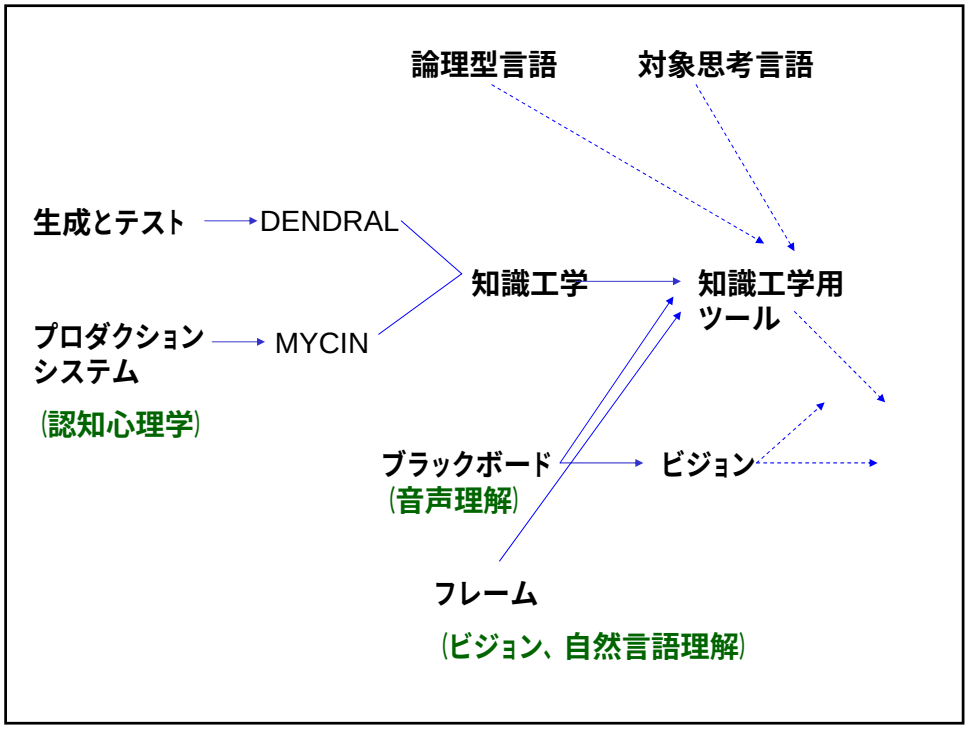


細長い { 水 → 川
アスファルト → 道

長方形の緑領域 → 畑

道と川の交点 → 橋


黒板モデルによる統語解析
黒　板
統語
知識源4
統語
知識源3
統語
知識源2
統語
知識源1
品詞確定
知識源
S
S
VP
VP
NP
NP
VP
PP
NP
NP
NP
文
句
品詞
入力文
DET N V N DET AUX N PREP DET V N
the boy saw the can on the train


論理型言語
対象思考言語
生成とテスト
DENDRAL
プロダクション
システム
MYCIN
(認知心理学)
知識工学
知識工学用
ツール
ブラックボード
(音声理解)
ビジョン
フレーム
(ビジョン、自然言語理解)

(suzuki-ken (date-of birth (1967 8 9))
(age (26))
(sex (male))
(institution (Naniwa University))
(address (1-1, Naniwaku Osaka))
(fields ((electronics) (artificial-intelligence)))

フレームの構造

(suzuki-ken (date-of-birth (VALUE (1967 8 9))
(IF-ADDED (**年齢を求める手続き**)))
(age (VALUE (26)))
(sex (DEFAULT (male))
(institution (VALUE (Naniwa University)))
(address (IF-NEEDED(**所属の住所を求める手続き**)))
(fields (VALUE (electronics) (artificial-intelligence))))

```
(suzuki-ken (IS-A (VALUE (member)))
            (date-of-birth (VALUE ((1967 8 9))))
            (age (VALUE (26)))
            (sex)
            (institution (VALUE (Naniwa University)))
            (address)
            (fields (VALUE (electronics) (artificial-intelligence))))
(member (date-of-birth (IF-ADDED (年齢を求める手続き)))
        (sex (DEFAULT (male))
        (address (IF-NEEDED (所属の住所を求める手続き))))
```

```
(room (HAS-AS-PART (DEFAULT (wall floor ceiling)))
      ' ' ' ' ' ' ' ' '
      ' ' ' ' ' ' ' ' ')
(floor (PART-OF (VALUE (room)))
      ' ' ' ' ' ' ' ' '
      ' ' ' ' ' ' ' ' ')
```

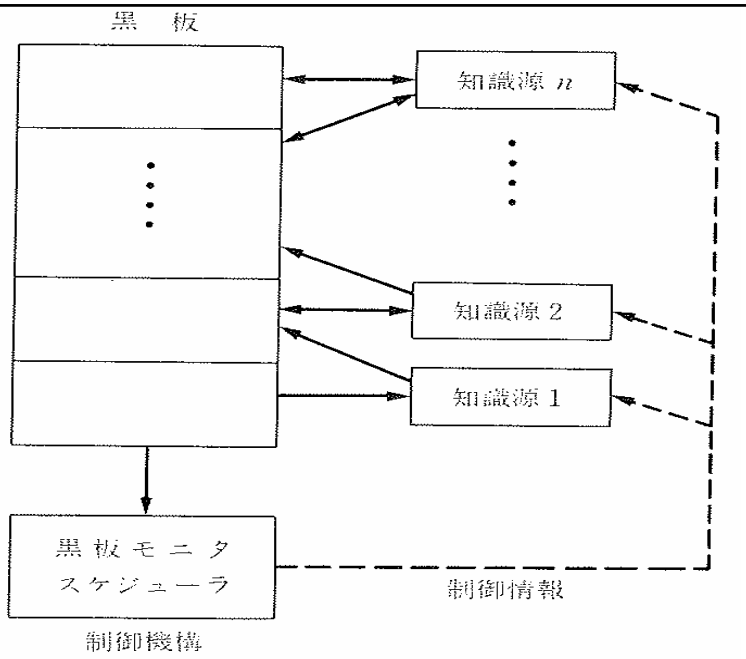
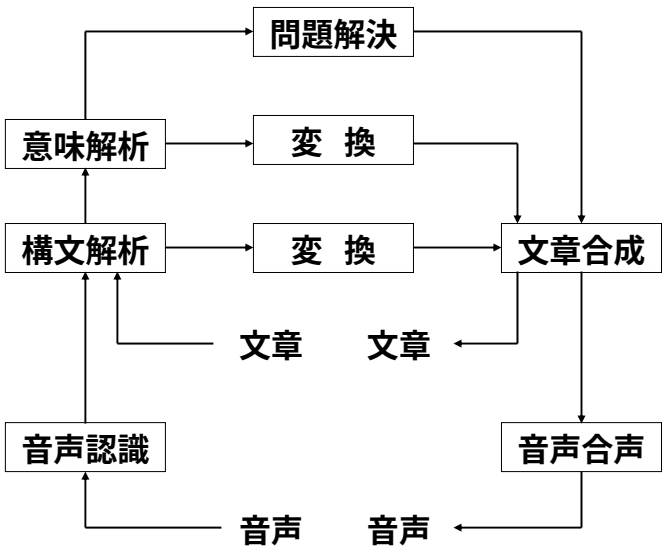


図 8.2　黒板モデルの構成



They buy a car with four doors.

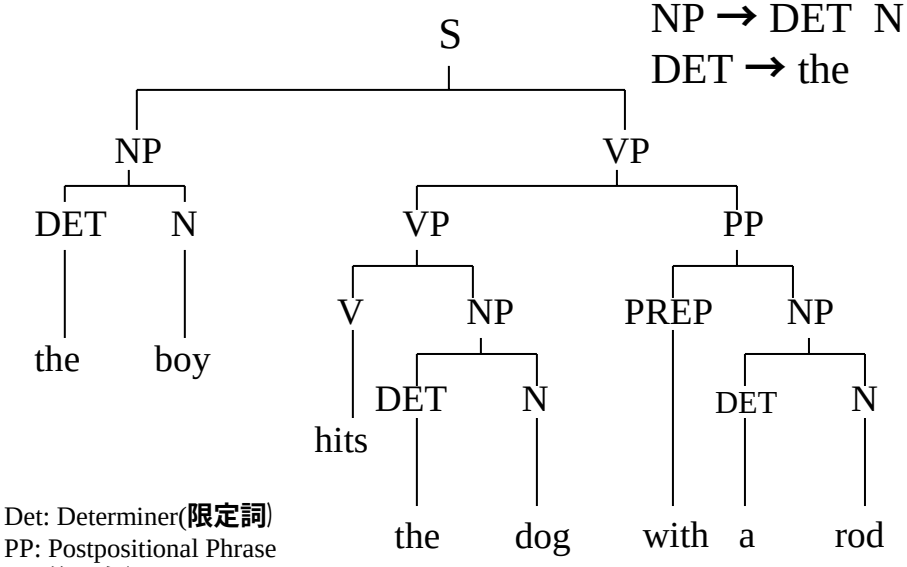
dollars.

文法の種類

- **定義**
  - **終端記号**(terminal symbol, category) a, b, …
    非終端記号(nonterminal …) A, B, ...
    記号列(string) **α**, **β** …
- **正規文法**(regular grammar)
  A → a, A → aB
- **文脈自由文法**(context-free grammar)
  A → **β**
- **文脈依存文法**(context-sensitive grammar)
  α → **β**
  - **1型**: $|\alpha| \leq |\beta|$
  - 0**型**: **制限なし**

文脈自由文法

S → NP VP
NP → DET N
DET → the



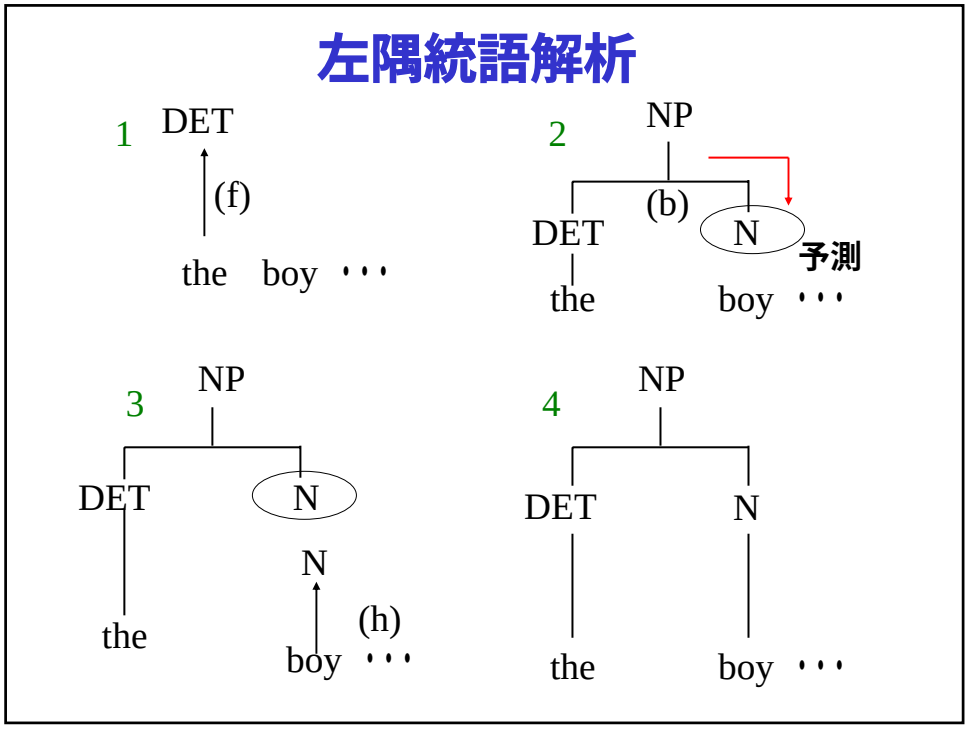
Det: Determiner(**限定詞**)
PP: Postpositional Phrase
(**後置句**)

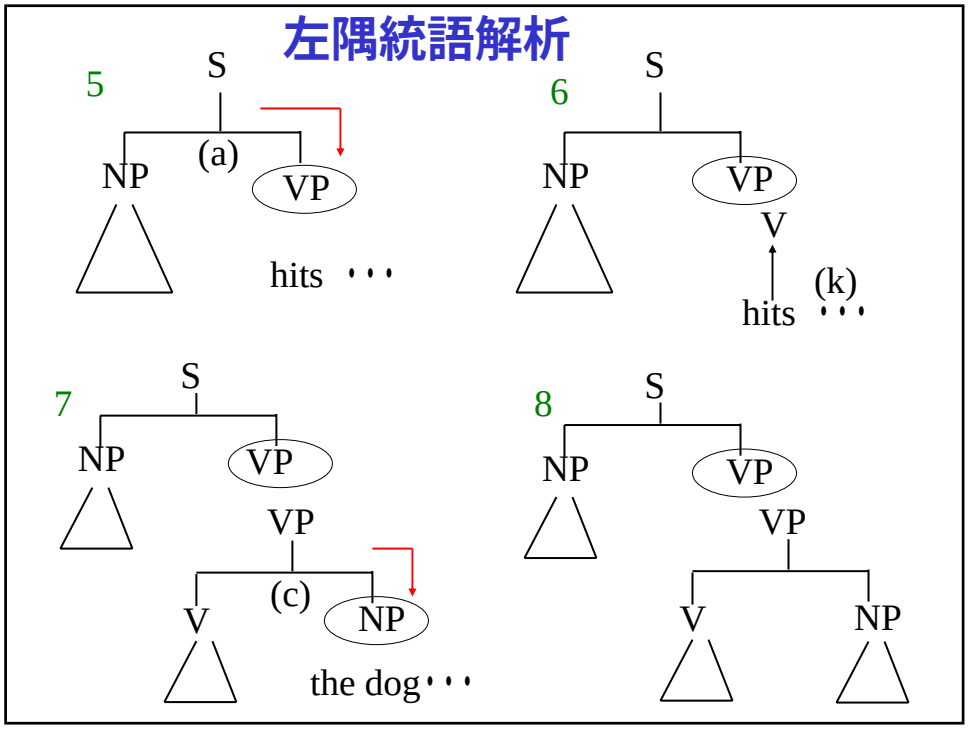
Top-Down Parsing

1. **S**	the boy hits the dog with a rod.
2. **NP** VP	the boy hits the dog with a rod.
3. **DET** N VP	**the** boy hits the dog with a rod.
4. **N** VP	**boy** hits the dog with a rod.
5. **VP**	hits the dog with a rod.
6. **V** NP	**hits** the dog with a rod.
7. **NP**	the dog with a rod.
8. **DET** N	**the** dog with a rod.
9. **N**	**dog** with a rod.

Top-Down Parsing

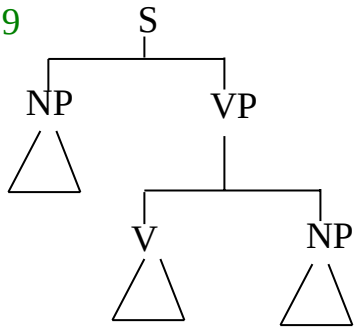
1. **S**	the boy hits the dog with a rod.
2. **NP** VP	the boy hits the dog with a rod.
3. **DET** N VP	**the** boy hits the dog with a rod.
4. **N** VP	**boy** hits the dog with a rod.
5. **VP**	hits the dog with a rod.
10. **VP** PP	**hits the dog** with a rod
11. ……	……..
12. **PP**	with a rod.
13. **PREP** np	**with** a rod.
14. **DET** n	a rod.


左隅統語解析
1
DET
(f)
the boy ・・・
2
NP
(b)
DET
N
予測
the
boy ・・・
3
NP
DET
N
the
N
(h)
boy ・・・
4
NP
DET
N
the
boy ・・・


左隅統語解析
5
S
(a)
NP
VP
hits ・・・
6
S
NP
VP
V
(k)
hits ・・・
7
S
NP
VP
VP
(c)
V
NP
the dog・・・
8
S
NP
VP
VP
V
NP

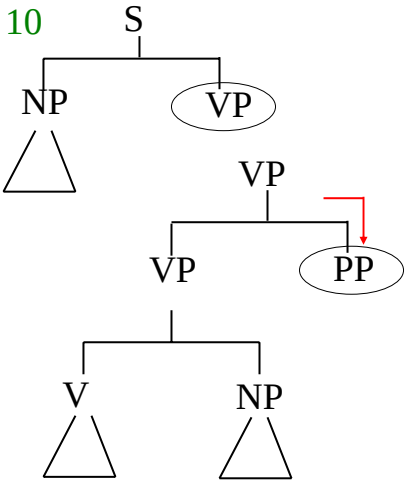
左隅統語解析

9



sentence が完成するが、
文が残っている

10



8へ後戻りし、解析を続ける

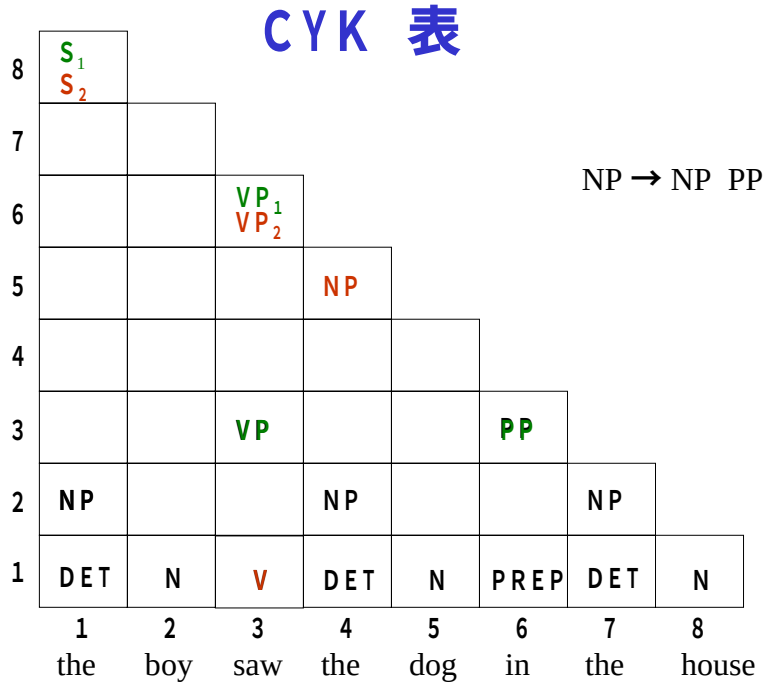
Bottom-Up Parsing

CYK 表

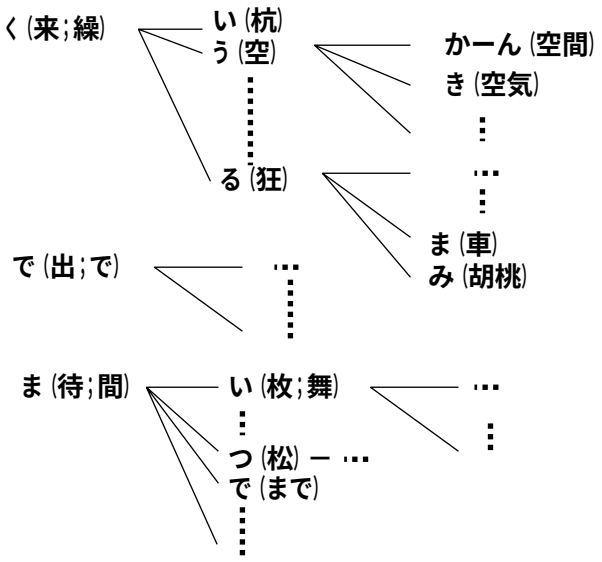
NP → NP PP

	1	2	3	4	5	6	7	8
8								
7								
6								
5								
4								
3								
2	NP			NP			NP	
1	DET	N	V	DET	N	PREP	DET	N
	the	boy	saw	the	dog	in	the	house

CYK 表



形態素解析のための辞書



形態素辞書に基づく CYK 表の途中結果

3			話 歯無し					
2	今		花 鼻	梨 無し	志賀		或	
1	医 位	間 真	は 歯	名 菜	四 死	が 我	有 亜	る
	1	2	3	4	5	6	7	8
	い	ま	は	な	し	が	あ	る

形態素解析結果とCYK表

8	今Hが有る 今は梨が有る
7	今Hが有 今は梨が有
6	今Hが 今は梨が
5	今H 今は梨
4	今はG 今F
3	今は 今歯

H=[話
歯無し]

G=[名
菜]

F=[花
鼻]

いまはなしがある

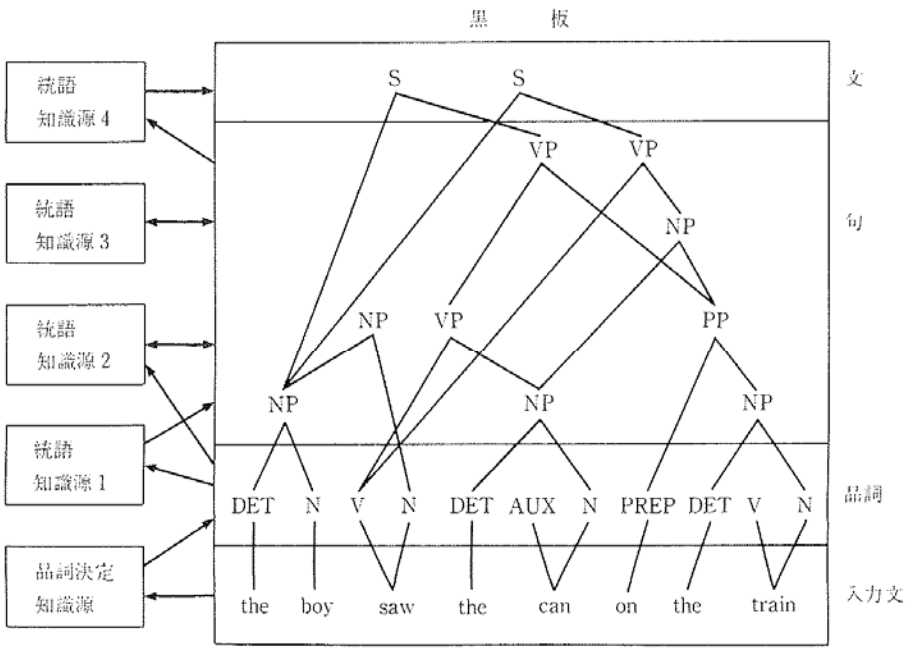


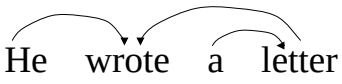
図 8.3　黒板モデルによる統語解析

コーパスに基づく統語解析
(統語解析を規定する)

確率文脈自由文法に基づく統計モデル

$$P(\alpha \mid A) = \frac{C(A \rightarrow \alpha)}{\sum_i C(A \rightarrow \alpha_i)}$$

依存関係に基づく統計モデル



$$P(w_i \rightarrow w_j) = \frac{C(w_i, w_j, \text{依存})}{C(w_i, w_j, \text{依存}) + C(w_i, w_j, \text{非依存})}$$

$$Score(D) = \Pi_{w_i \rightarrow w_j \in D} P(w_i \rightarrow w_j)$$

同音語

(1) 自立語単語
しょうひん：商品、賞品、小品

(2) 活用形
いった　　：行った、言った、入った

(3) 文節
ひとで　　：人手、人で、火とで

(4) 接辞付き
しんぶんや：新分野、新聞屋

(5) べたがき
きょうはきものの：今日は着物の、今日履物の

文節

<文節> := < 通常文節> |<数詞文節> | <固有名詞文節>

<通常文節>:= (〔接頭辞〕 自立語 〔接尾辞〕 *)*
〔付属語〕 *

<数詞文節> := 〔前置助数詞〕 数詞 〔後置助数詞〕
〔接尾語〕 * 〔付属語〕 *

<固有名詞文節> := 〔接頭語〕 固有名詞
〔接尾語〕 * 〔付属語〕 *

〔〕は省略可、 * は繰り返し可を表す。

文節単位変換(分かち書きあり)

(1) 自立語と付属語、付属語と付属語の接続行列

C(i,j)=1：行 i が列 j に接続可能　(大きさは250程)

C(i,j)=0：行 i が列 j に接続不可能

(2) 文節終端条件

T(i,j)=1：文節終端可能

T(i,j)=0：文節終端不可能

例　「おもったが」

尾もったが(尾もから後が接続不可能)
重ったが(用言の語尾変化が接続不可能)

連文節変換

(1) 文節最長一致法
ていあんしたけいかくを　　(失敗したら次の候補へ)

(2) 2文節最長一致法 (最初の文節を決めるだけ)
けんきゅうの もくてきは　　研究のも 九

(3) 文節数最小法
にほんの れきしを まなぶ　　日本 乗れ 岸を 学ぶ

(4) 前処理法 (特徴的な部分を抽出してから解析)
ぶんしょうの にゅうりょくに (熟語の熟語に)

(5) 共通区切り探索法 (n文節最長一致法の共通区切り)
みせではかった せいひんの
(店では買った、店で測った) 共通区切りで分割

あいまい性への対処

(1) 体言に直接動詞がつくものはX
私は/知っていると　　(私/走っていると)

(2) 1字語名詞はX
増えてきたと/聞く　　(増えてきたとき/区)

(3) 漢字熟語の結合○
行政改革　　(行政か/威嚇)

(4) 意味情報の利用 (分類語彙表や類語辞典を参考にして辞書に付加される)
本を/読んだ　　(本を/呼んだ)
先生と/生徒が　　(宣誓と/生徒が)

第一階述語論理(FPC)

「私は本を持つ」

$$\exists x[have(I,x) \wedge book(x)]$$

「私は本かノートを持つ」

$$\exists x[have(I,x) \wedge book(x)] \vee \exists x[have(I,x) \wedge notebook(x)]$$

「すべての女性はケーキが好きだ」

$$\forall x[girl(x) \rightarrow \exists y[loves(x,y) \wedge cake(y)]]$$

「誰もそれをできない」

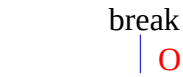
$$\neg\exists x[human(x) \wedge do-it(x)]$$

「ペンギン以外の鳥は飛ぶ」

$$\forall x[bird(x) \wedge \neg penguin(x) \rightarrow fly(x)]$$

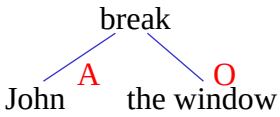
NL ⟶ parser ⟶ FPC ⟶ Database

格文法



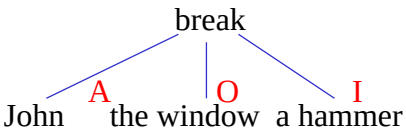
(a) ”The window broke”

O：**対象格**



(b) ”John broke the window”

A：**動作主格**



(c) ”John broke the window with a hammer”

I：**道具格**

Fillmore の与えた深層格の集合
(1971年当時)

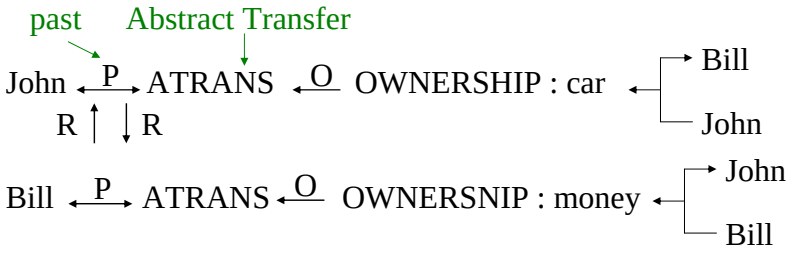
動作主格 (A)：動作を引き起こす者
経験者格 (E)：心理事象を体験する者
道具格 (I)：出来事の直接原因
対象格 (O)：移動する対象物や変化する対象物など
源泉格 (S)：対象物の移動の起点、および最初の状態
目標格 (G)：対象物の移動の終点、および最終的な状態
場所格 (L)：出来事が起こる場所
時間格 (T)：出来事が起こる時間

概念依存理論
(Conceptual Dependency Theory)

Schank, R. C. : Conceptual Information Processing,
North-Holland, Amsterdam and American Elsevier ,
New York , 1975

一つのまとまりを概念化　　Conceptualization

“John sold his car to Bill.”
に対応する概念依存構造



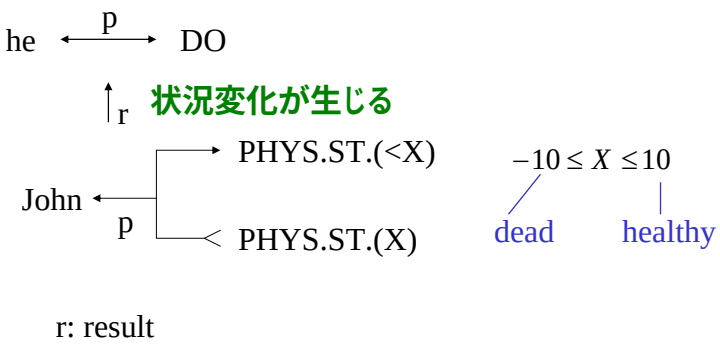
$\alpha_1 \longleftrightarrow \alpha_2$
$\beta_1 \longleftrightarrow \beta_2$ **が理由･原因** (Reason) **となって**

R

$\beta_1 \longleftrightarrow \beta_2$
$\alpha_1 \longleftrightarrow \alpha_2$ **なる行為が引き起こされること**

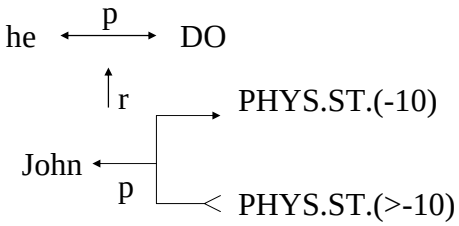
P: past

“He hurt John.” に対する概念依存構造

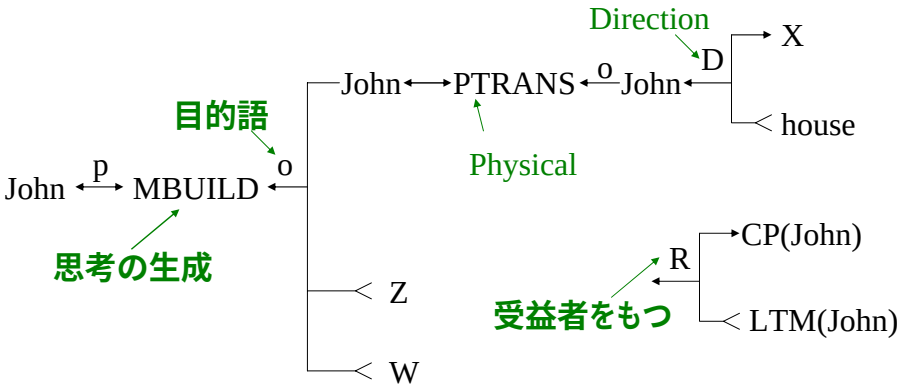


r: result

“He killed John.” に対する概念依存構造

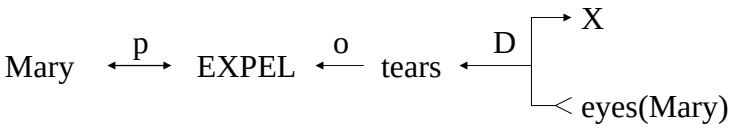


“John decided to leave the house.” の概念依存構造



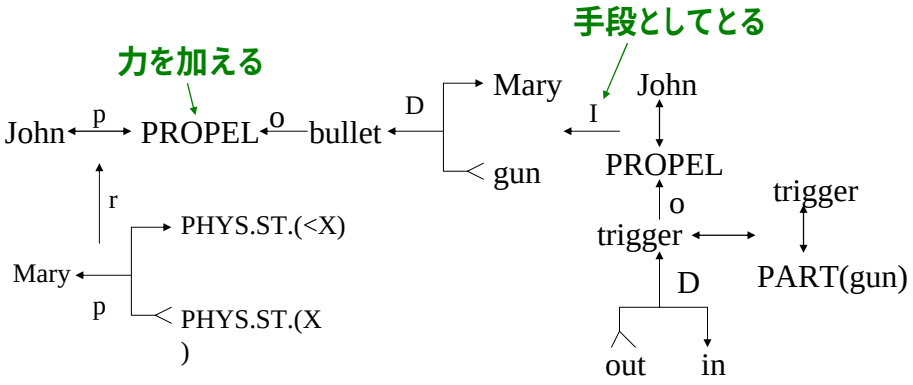
「ジョンは、ジョンがジョンを家からXに移動させる（PTRANS）という考えを、LTM（長期記憶）からCP（概念プロセッサ）に移動した。」

“Mary cried.”の概念依存構造



「メアリーはメアリーの目から涙をXに排出した。」

“John shot Mary.”の概念依存構造



「ジョンは、gun の一部である trigger に力を加えてout から in にすることによって鉄砲からメアリーに bullet を動かした。その結果メアリーの物理的な状態がXからX 未満に悪化した。」

“John went to New York from Texas.” に対するMARGIEの出力

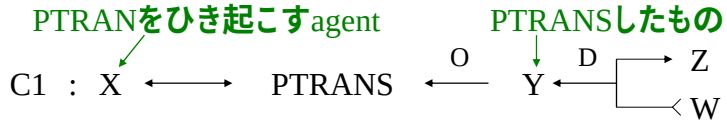
(1) JOHN CAME TO BE IN NEW YORK.

(2) JOHN CEASED TO BE IN TEXAS.

(3) JOHN WANTED TO DO SOMETHING IN NEW YORK.

(4) JOHN THOUGHT HE WOULD ENJOY BEING IN NEW YORK.

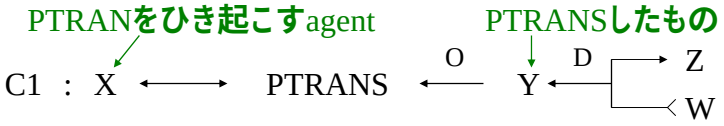
PTRANSの概念依存構造での推論規則



上の概念依存構造から次のようなことを推論する。

- (1) Yは今Zに位置している。
  JOHN CAME TO BE IN NEW YORK.
- (2) Yはもはや位置Wにはない。
  JOHN CEASED TO BE IN TEXAS.
- (3) もしXが人間でC1を要求したか、XとYが同じであれば、Xは通常Zでできることをすると思われる。
  JOHN WANTED TO DO SOMETHING IN NEW YORK.
- (4) (3)をすることがXを喜ばせるだろう。
  JOHN THOUGHT HE WOULD ENJOY BEING IN NEW YORK.

PTRANSの概念依存構造での推論規則



上の概念依存構造から次のようなことを推論する。

(1) Yは今Zに位置している。

(2) Yはもはや。位置Wにはない

(3) もしXが人間でC1を要求したか、XとYが同じであれば、Xは通常Zでできることをすると思われる。

(4) (3) をすることがXを喜ばせるだろう。

命題論理

素命題：P, Q, ...
論理記号：∧, ∨, ~, →
命題：P ∧ Q, P ∨ Q, ~P, P → Q, P ≡ Q

P ∧ Q：**論理積、連言、**and
P ∨ Q **を論理和、選言、**or
~ P：**否定、**not
P → Q: **含意、**implication
Q ≡ Q: **同値、** equivalence

P → Q ≡ ~P ∨Q

P	Q	P ∧ Q	P ∨ Q	P → Q	P ≡ Q
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

述語論理

ON(monkey, box)　　ON: **述語記号**(predicate symbol) T or F、
monkey, box: **定数記号**
AT(monkey, x)　　x: **変数記号**
child(x): x **の子供**　　child: **関数記号**

結合記号：∧, ∨, ~, →, ≡
量記号：　∀ (**全称記号**) 、∃ (**存在記号**)
項(term): **定数記号、変数記号、関数、**T, F
素命題：　**述語記号と項から成る。**
例：　ON(x, y), AT(child(x), a)
命題 (well-formed-formula, wff): **素命題、命題を結合記号で**
つないだもの、変数を量記号で指定したもの
P,Q**を命題とすると、**P ∧ Q, P ∨ Q, ~P, P → Q, P ≡ Q
(∀x)[(∃y) LESS(x, y)], ∀x∃y LESS(x, y),
∃ x ∀ y LESS(x, y)

述語論理

命題の例: ∀x ∀y ∀z [LESS(x, y) ∧ LESS(y, z) →LESS(x, z)]

妥当命題: **述語記号にどのような解釈を与えても真**
Valid wff

∀x P(x) →P(a)

充足不能命題:**述語記号にどのような解釈を与えても偽**
Unsatisfiable wff
∀x P(x) →~P(a)

節形式

リテラル(literal): **素命題、素命題の否定**
節形式(clause) : **リテラル、リテラルの選言**

∀x ∀y ∀z [P ∧Q ∧R]　　[ ]**内を母式**(matrix)

節形式への変換
1. P → Q を ~P∨Q, P ≡ Q を(~P ∨ Q)∧(~Q ∨ P)
2. ∀x[~P(x) ∨ ∃xQ(x)] を ∀x[~P(x) ∨ ∃yQ(y)]
3. ~∀x [P(x) ∧~(Q (x) ∧R (x))] を ∃x [~P(x) ∨(Q (x) ∧R (x))]
4. ∀x[∃y P(x, y)] を ∀x[P(x, f(x))]　Skolem-function
   ∃x P(x) をP(a)
5. ∀x[P(x) ∨ ∀y ~Q(y)] を ∀x ∀y [P(x) ∨ ~Q(y)]
6. (A ∧ B)∨(B ∧C) を (A ∨ B)∧(A ∨ C) ∧B∧(B ∨C)

(A ∨ C)
B

推論の基本

1．PとP → Q からQを導く（P, P → Q ├ Q と書く)。
2．P → Q とQ → R からP → Rを導く。

$C_1 = P \vee Q_1 \vee Q_2 \vee \ldots \vee Q_m$ と$C_2 = \sim P \vee R_1 \vee R_2 \vee \ldots \vee R_n$ から

$C_r = Q_1 \vee Q_2 \vee \ldots \vee Q_m \vee R_1 \vee R_2 \vee \ldots \vee R_n$ を導く。

C_1 と C_2 を親節、C_r を導出節(resolvent clause)という。

Procedure unify (P_1 ,P_2)

1 s:=NIL $Q_1 := P_1$, $Q_2 := P_2$

2 LOOP: $D_i = Q_1$ とQ_2の不一致集合

3 if empty(D) then return (s)

4 t_1:=first(D), t_2:=last(D)

5 if variable(t_1) and not-contain(t_1 , t_2)
then s1: = (t1 / t2)
else if variable(t2) and not-contain(t2 , t1)
then s1: =(t2 / t1)
else return(fail)

6 $Q_1 := Q_1 s_1$, $Q_2 := Q_2 s_1$, $s := s s_1$

7 go to LOOP

導出原理による証明

前提： $\forall x \exists v\, P(x,v) \wedge \forall y \forall z [P(a,y) \rightarrow \sim Q(y,z)]$

目標：$\sim \forall u \exists w Q(u,w)$

目標の否定： $\forall u \exists w Q(u,w)$

節形式： $Q(u,g(u))$

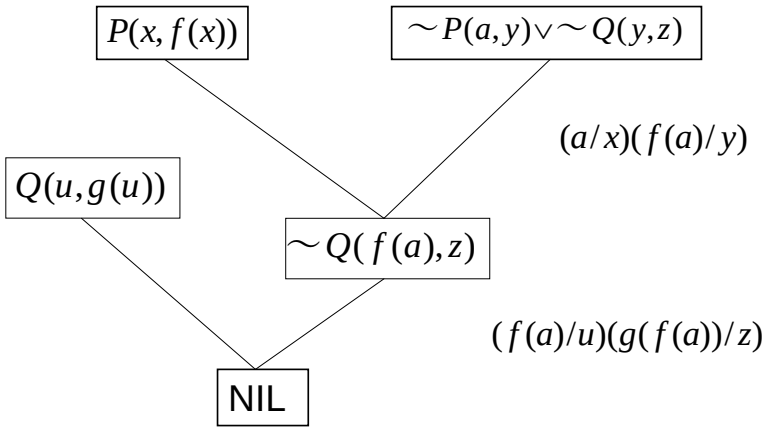
前提の節形式：

$$P(x, f(x))$$
$$\sim P(a,y) \vee \sim Q(y,z)$$

導 出 過 程

目標の否定： $Q(u,g(u))$　　前提：

$$P(x, f(x))$$
$$\sim P(a,y) \vee \sim Q(y,z)$$



証明の制御

目標の否定 $\sim P(x,y) \vee Q(x,y)$

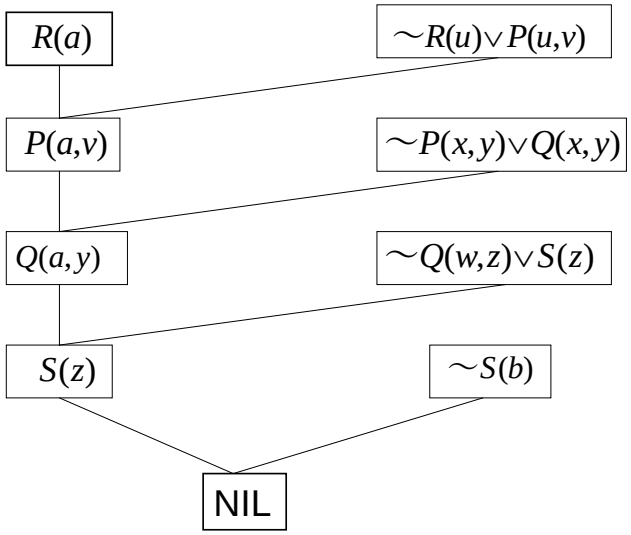
前提 $\sim R(u) \vee P(u,v)$

$\sim Q(w,z) \vee S(z)$

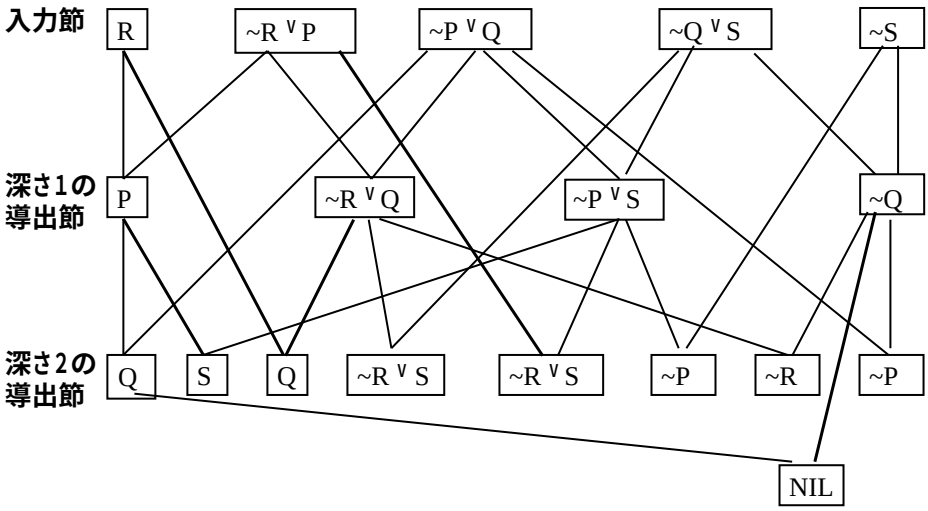
$R(a)$

$\sim S(b)$

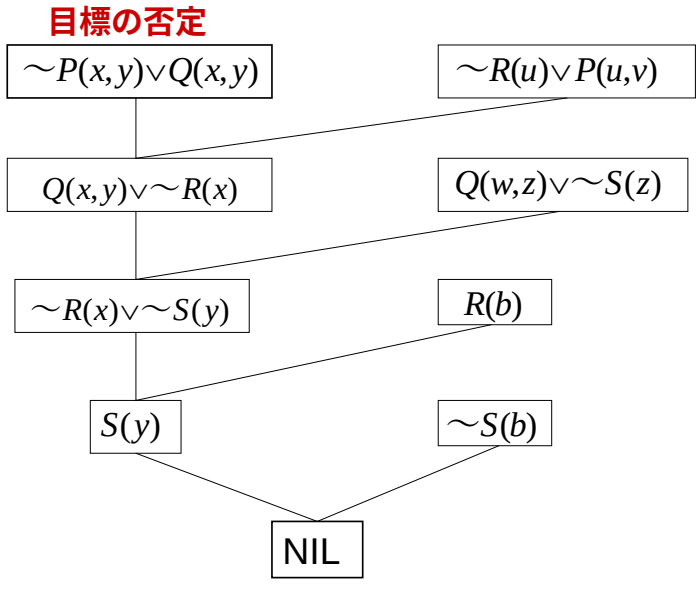
線形戦略による導出過程



幅優先戦略による導出過程



支持集合戦略による導出過程



答えの表現

目標の否定 $\sim P(x,y) \vee Q(x,y) \ \vee (P(x,y) \wedge \sim Q(x,y))$

前提 $\sim R(u) \vee P(u,v)$

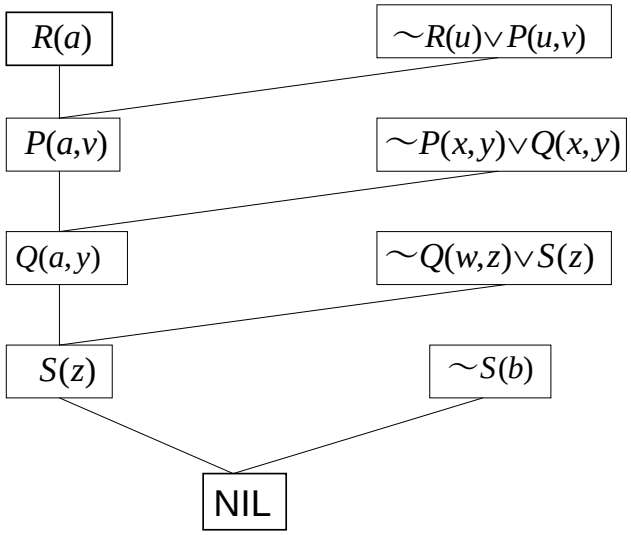
$\sim Q(w,z) \vee S(z)$

$R(a)$

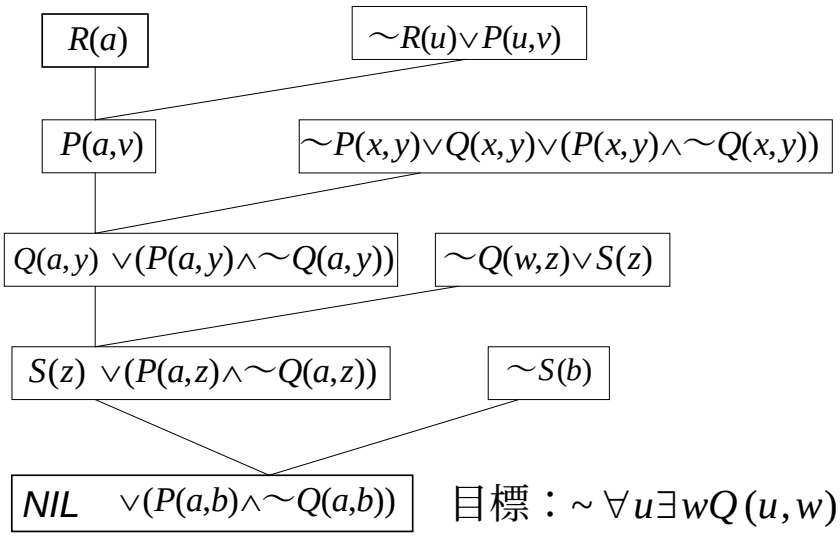
$\sim S(b)$

目標 $(P(x,y) \wedge \sim Q(x,y))$

線形戦略による導出過程



線形戦略による導出での答の表現



答の表現の具体例

命題：誰にも祖母父がいる

$\forall x \exists y G(x,y)$

命題の否定：

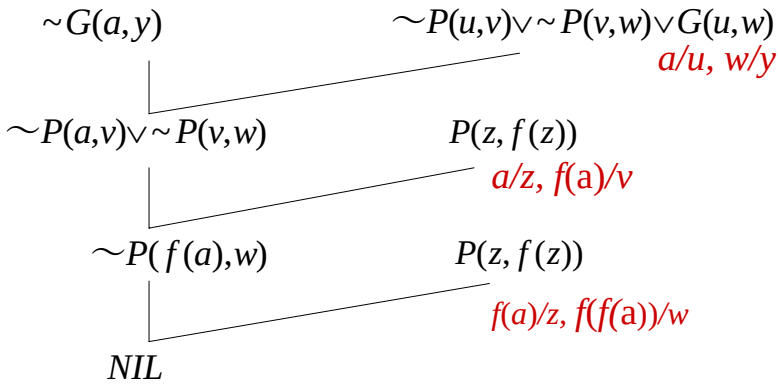
$\exists x \forall y \sim G(x,y) \equiv \sim G(a,y)$

誰にも親がいる $P(z, f(z))$

祖父母の定義

$P(u,v) \wedge P(v,w) \rightarrow G(u,w)$

親



答の表現の具体例

命題：誰にも祖父母がいる

$\forall x \exists y G(x,y)$

命題の否定：　$\sim G(a,y)$

命題

$\sim G(a,y) \vee G(a,y)$

$\sim P(u,v) \vee \sim P(v,w) \vee G(u,w)$

a/u, w/y

$\sim P(a,v) \vee \sim P(v,w) \vee G(a,w)$

$P(z, f(z))$

a/z, f(a)*/v*

$\sim P(f(a),w) \vee G(a,w)$

$P(z, f(z))$

f(*a*)*/z,*

f(*f*(a))*/w*

$G(a, f(f(a))$

第一階述語論理（ＦＰＣ）

「私は本を持つ」

$\exists x[have(I,x) \wedge book(x)]$

「私は本かノートを持つ」

$\exists x[have(I,x) \wedge book(x)] \vee \exists x[have(I,x) \wedge notebook(x)]$

「すべての女性はケーキが好きだ」

$\forall x[girl(x) \rightarrow \exists y[loves(x,y) \wedge cake(y)]]$

「誰もそれをできない」

$\neg \exists x[human(x) \wedge do-it(x)]$

「ペンギン以外の鳥は飛ぶ」

$\forall x[bird(x) \wedge \neg penguin(x) \rightarrow fly(x)]$

NL ⟶ parser ⟶ FPC ⟶ Database

非単調論理による推論

公理の集合 A から導かれる定理の集合：Th (A)

$$A \subset B \Rightarrow Th(A) \subseteq Th(B)$$

これを論理体系の単調性という

単調論理 ⟷ 非単調論理

1 閉世界仮説

2 サーカムスクリプション

3 デフォールト推論

デフォールト推論 (default reasoning)

P が成立ち、~*Q* が証明されていないならば、
Q が成り立つ.

$$\frac{P : \mathrm{M}Q}{Q}$$

これは**正規デフォールト規則**
P を前提、*Q* を仮定、*R* を結論とよぶ.

$$\frac{BIRD(x) : MFLY(x)}{FLY(x)}$$

“Most birds fly”という意味

正規デフォールト体系

命題 (well-formed-formula) の集合 W と
正規デフォールト規則の集合 D からの
論理体系 (W, D) を正規デフォールト体系,

デフォールト論理体系から得られる命題の集合を
拡張 (extension) とよぶ。

デフォールト規則: Most birds fly と、

W＝ ~PENGUIN(x) ∨ ~FLY(x),

BIRD(a), BIRD(b), PENGUIN(b)

から構成される論理体系の拡張Eは

E＝ W, FLY(a)

デフォールト推論のアルゴリズム

1. W と CON(D) から g を証明する。

S:＝W

2. Ds:＝証明に用いたDの規則の集合
If Ds＝{}, go to 4.
Else S:＝S U CON(Ds)

3. W と CON(D) からPRE(Ds)を証明する。
go to 2

4. Sが充足可能であることを示す。

非 単 論 論 理

推論システム

デフォールト推論

サーカムスクリプション

推論結果 →

← 信念の状態

データベース管理システム

TMS

(Truth Maintenance System)

ATMS

(Assumption-Based TMS)

真理性維持システム （TMS）

信念の表現：節点　意味　（正当化）

1．支持リスト正当化（SL）
（SL＜In リスト＞＜Out リスト＞）

2．条件つき正当化（CP）
（CP＜結果＞＜In 仮説リスト＞＜Out リスト＞）

1：矛盾の原因の候補となる信念を求める。
2：矛盾の原因の候補がよくないという意味の節点を生成する。
3：矛盾の原因の候補から適当な信念を選んでOut 状態にして矛盾を解消する。

TMS (Truth Maintenance System)
by Doyle ('79)

信念の他の信念との依存関係を正当化 (justification)

代表的正当化

[1]支持リスト正当化 (SL)
(Support List)
<節点> (SL <In リスト> <Out リスト>
↑ ↑ ↑
成立 ← すべて In すべてOut

[2]条件つき正当化 (CP)
(Conditional Proof)
<節点> (CP <結果> <In リスト> <Out リスト>)
↑ ↑ ↑
成立 ← すべて In すべてOut

仮説に基づくTMS (ATMS)

Assumption－based TMS by DeKleer ('86)

節点の表現: n: <**データ、ラベル、正当化**>
データと正当化は、推論システムから供給
ラベルは、節点が成立する環境の集合
ラベルは ATMS が維持 ↑ $\{E_1、E_2、\cdots\}$

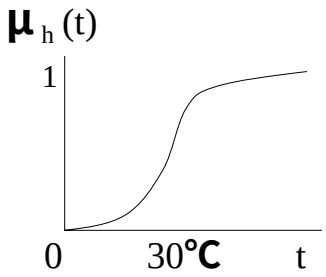
ATMSのラベル更新アルゴリズム

1. 節点nの正当化が与えられると、ラベルを求める
2. If 新しいラベル＝もとのラベル、終了
3. If n が矛盾節点、
そのラベルの環境を NOGOOD とし、各節点のラベルから
矛盾する環境を除く Else, n を正当化に含む節点に対して、
このラベルの変化に伴う変更を行う

Fuzzy 理論

すべての温度の集合: U
水の沸騰する温度集合: 100℃のみ
暑い温度集合 h は定義が困難

t の hである確からしさ: $\mu_h(t)$を
t の hへの所属度、あるは
メンバーシップ関数という



1965年 L.A. Zadeh:
Fuzzy Sets, Information and Control

Fuzzy 推論

$\mu(A \wedge B) = \min\{\mu(A), \mu(B)\}$
$\mu(A \vee B) = \max\{\mu(A), \mu(B)\}$
$\mu(\sim A) = 1 - \mu(A)$

Kleene の A→B

$\mu(A \rightarrow B)$
$= \mu(\sim A \vee B)$
$= \max\{\mu(\sim A), \mu(B)\}$

A / B	0	1/2	1
0	1	1/2	0
1/2	1	1/2	1/2
1	1	1	1

$\mu(A \rightarrow A)=1$としたいが
$\mu(A)=1/2$で1/2となる。

Fuzzy 推論

Lukaszewicz の A→B

μ(A → B)=

min{1, 1− μ(A)+ μ(B)}

A / B	0	1/2	1
0	1	1/2	0
1/2	1	1	1/2
1	1	1	1

μ(~A → A)= μ(A)と
したいがμ(A)=1/2で1

Goedel の A→B

μ(A → B)=1 μ(A) =< μ(B)

μ(B) otherwise

A / B	0	1/2	1
0	1	0	0
1/2	1	1	1/2
1	1	1	1

同左

Fuzzy 推論

Mamdani の A→B

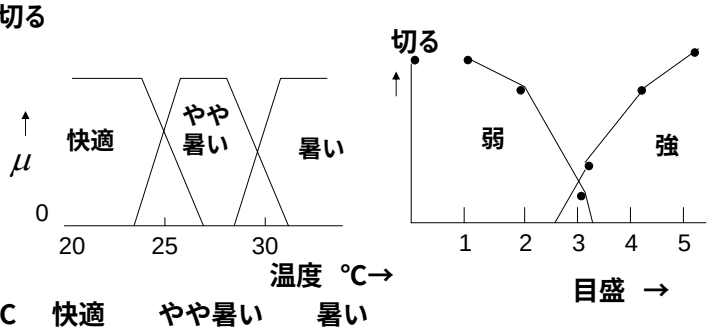
μ(A → B)= min{ μ(A), μ(B)}

(= μ(A∧B))

A / B	0	1/2	1
0	0	0	0
1/2	0	1/2	1/2
1	0	1/2	1

Fuzzy 推論

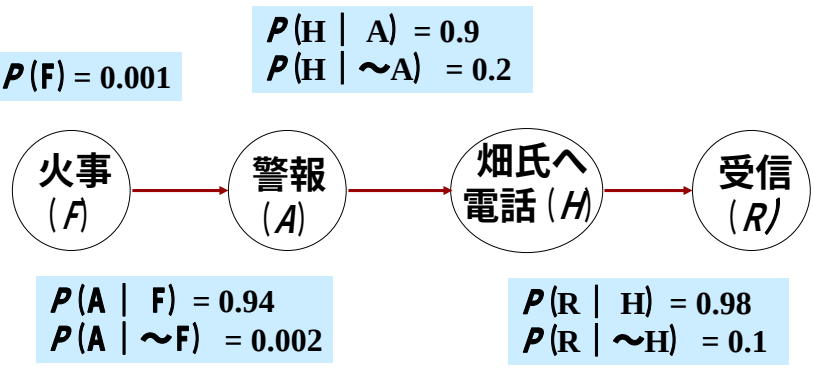
(1) 快適→空調を切る
(2) やや暑い→弱
(3) 暑い→強



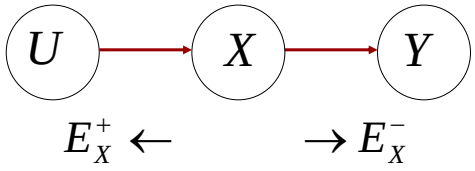
温度30℃	快適		やや暑い	暑い	
	0		0.4	0.6	
目盛	弱	強	(2)	(3)	(2) と (3)
1	1.0	0	0.4	0	0.4
2	0.8	0	0.4	0	0.4
3	0.1	0.1	0.1	0.1	0.1
4	0	0.8	0	0.6	0.6
5	0	1.0	0	0.6	0.6

目盛の重心
≈ 3.3

単純ベイズネットの例



単純ベイズネット

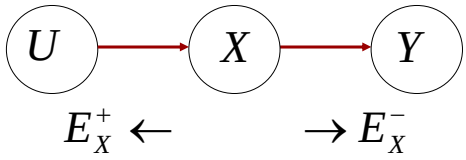


$$P(X|E) = P(X|E_X^-, E_X^+)$$

$$= \frac{P(E_X^-|X, E_X^+)P(X|E_X^+)}{P(E_X^-|E_X^-)} = \alpha P(E_X^-|X)P(X|E_X^+)$$

$$= \alpha\lambda(X)\pi(X)$$

単純ベイズネットのπとλの更新



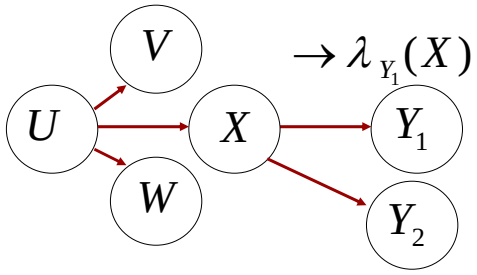
$$\pi(x) = \sum_u P(X|u, E_X^+)P(u|E_X^+)$$

$$= \sum_u P(X|u)P(u|E_U^+) = \sum_u P(X|u)\pi(u)$$

$$\lambda(X) = P(E_X^-, Y|X) = \sum_y P(E_X^-|X, y)P(y|X)$$

$$= \sum_y P(E_Y^-|y)P(y|X) = \sum_y \lambda(y)P(y|X)$$

木の場合のπとλの更新

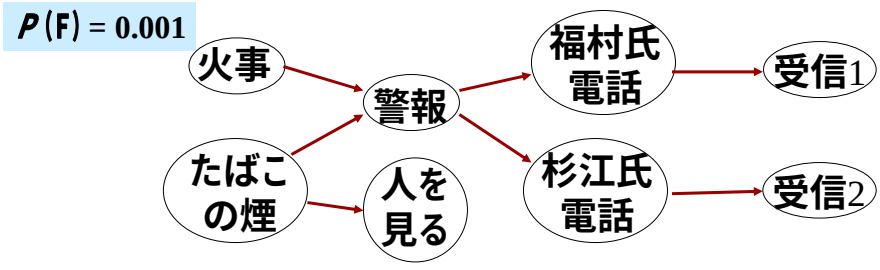


$$\pi(x) = \sum_u P(X|u)P(u|E^+_{U\neg X})$$

$$\lambda(X) = \prod_i P(E_{Y_i\neg X}|X) = \prod_i \lambda_{Y_i}(X)$$

$$= \prod_i \sum_{y_i} \lambda(y_i)P(y_i|X)$$

ベイズネット



警報(A)に対して

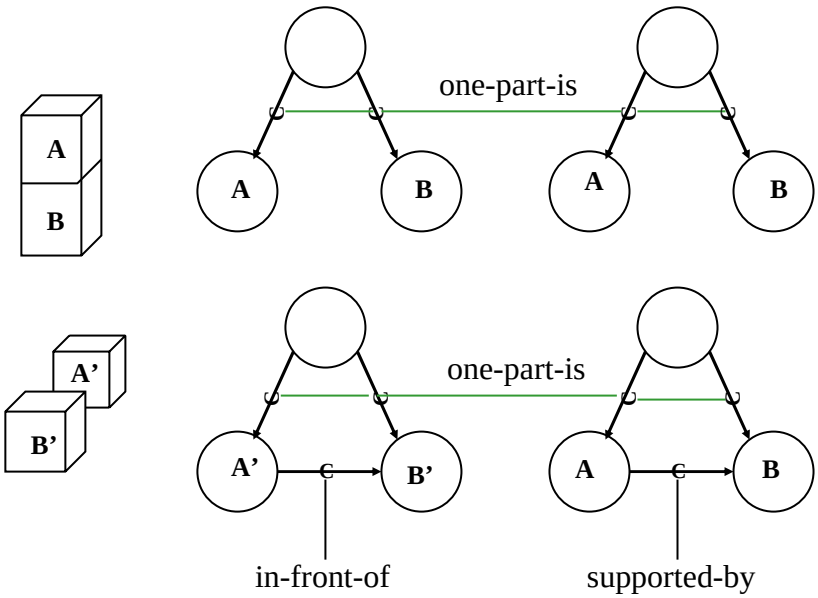
P(A | F, S) = 0.95
P(A | F, ～S) = 0.94
P(A | ～F, S) = 0.2
P(A | ～F, ～S) =0.01

Learning Structural 4 from Examples

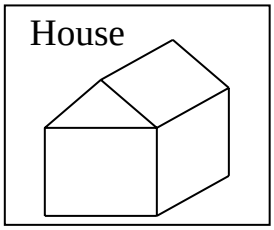
Patrick H. Winston,
The Psychology of Computer Vision,
McGraw-Hill Book Co. 1975

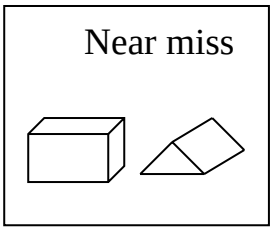
積木の線画から、積木の構造を学習する

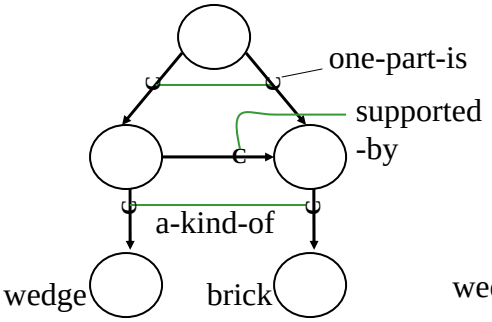
Tower

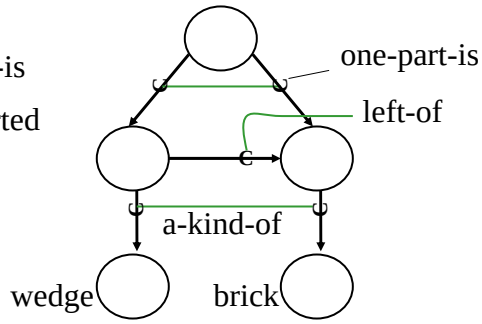


House

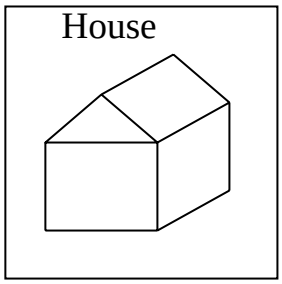

House

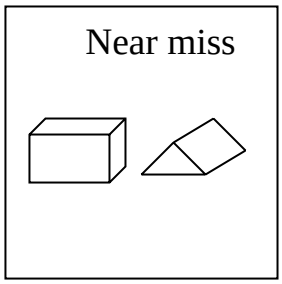

Near miss

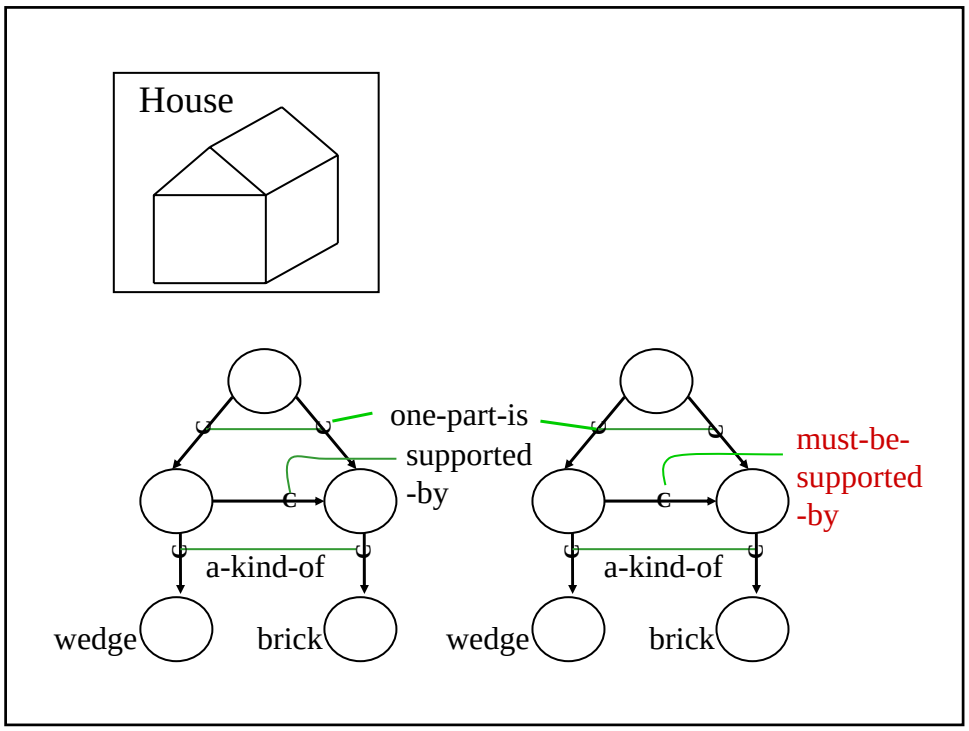

one-part-is
supported
-by
a-kind-of
wedge
brick

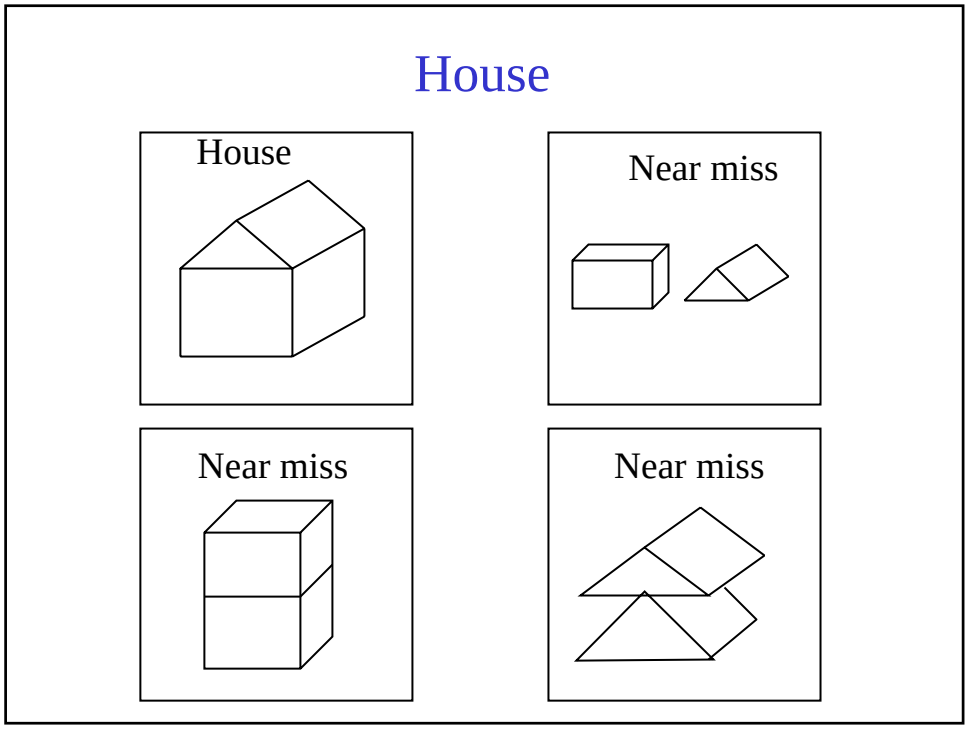

one-part-is
left-of
a-kind-of
wedge
brick

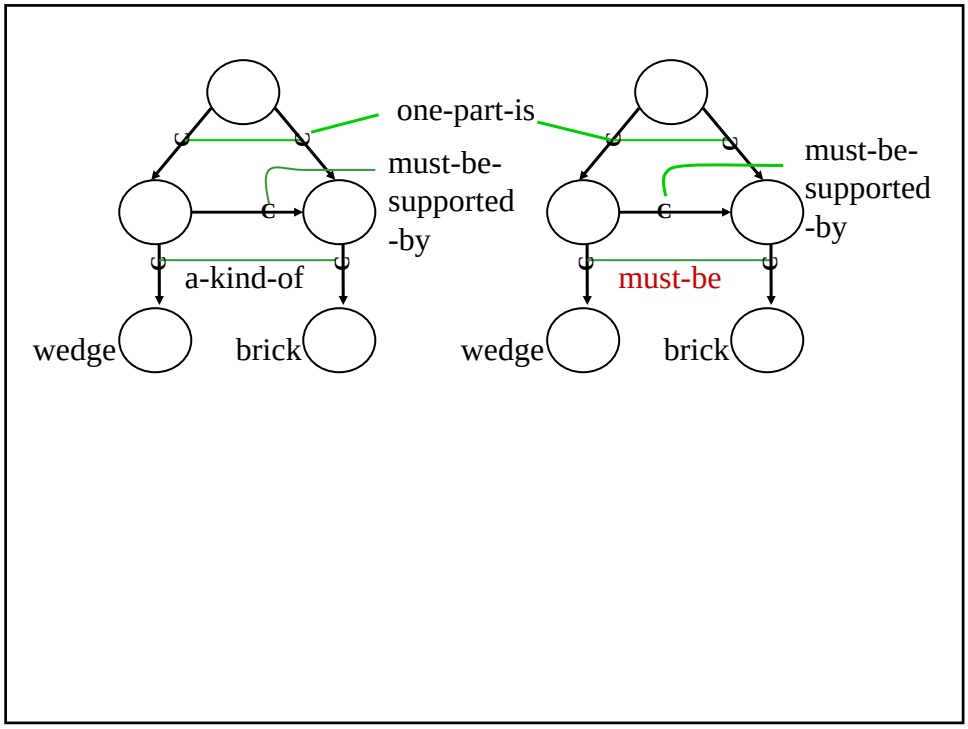
House


House

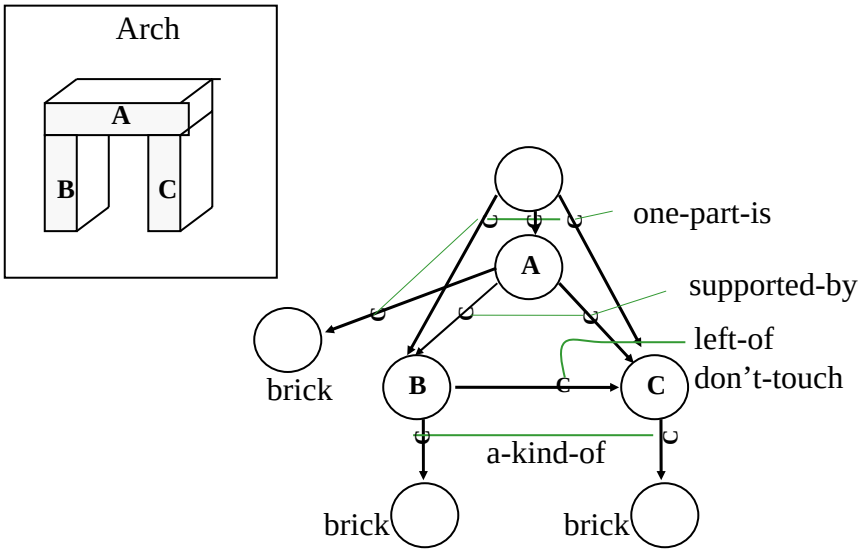

Near miss


House
one-part-is
supported
-by
must-be-
supported
-by
a-kind-of
a-kind-of
wedge
brick
wedge
brick

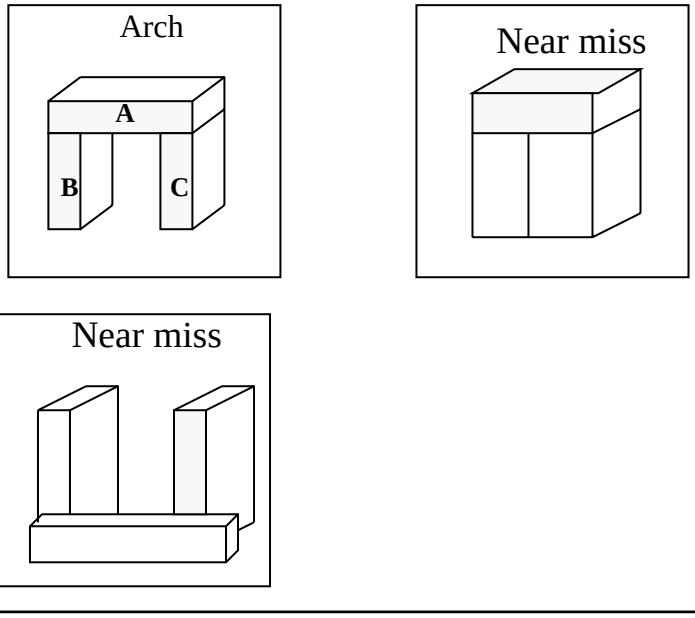

House
House
Near miss
Near miss
Near miss

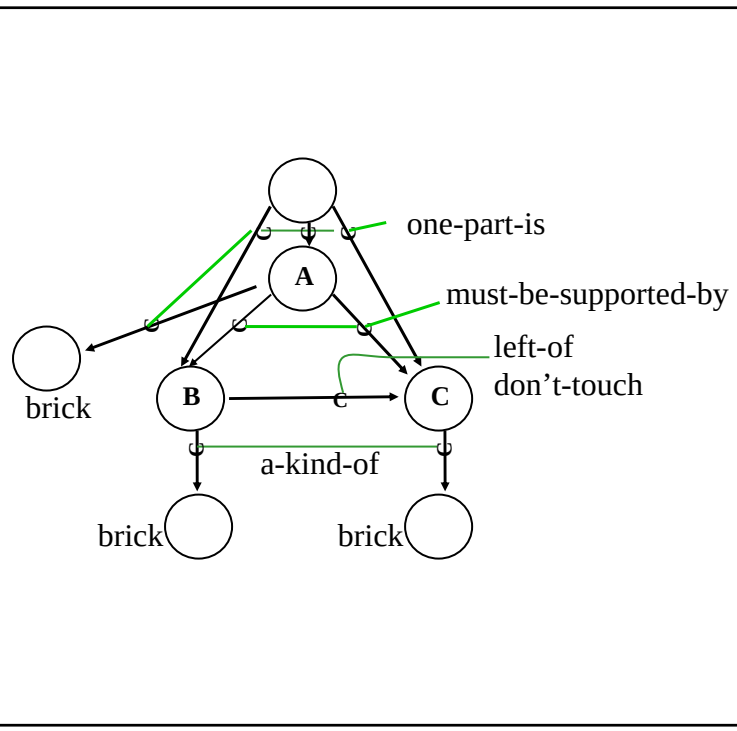

one-part-is
must-be-
supported
-by
a-kind-of
wedge
brick
must-be-
supported
-by
must-be
wedge
brick

Arch

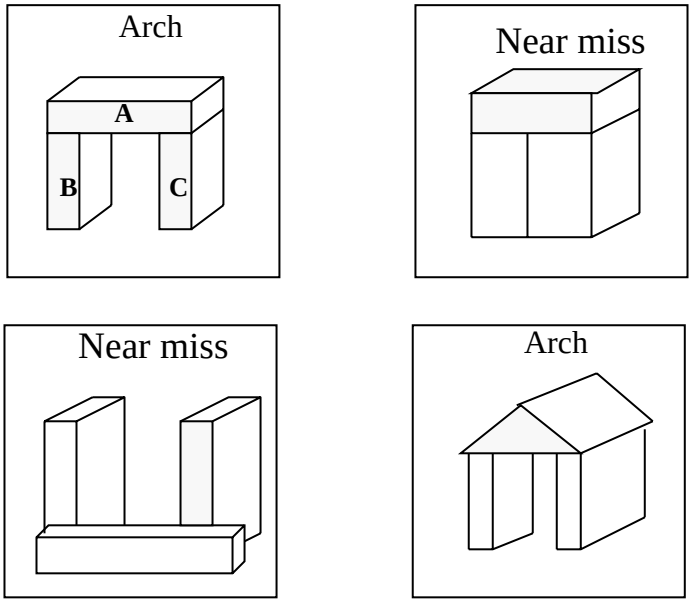

Arch
A
B
C
one-part-is
supported-by
left-of
don’t-touch
a-kind-of
brick
brick
brick

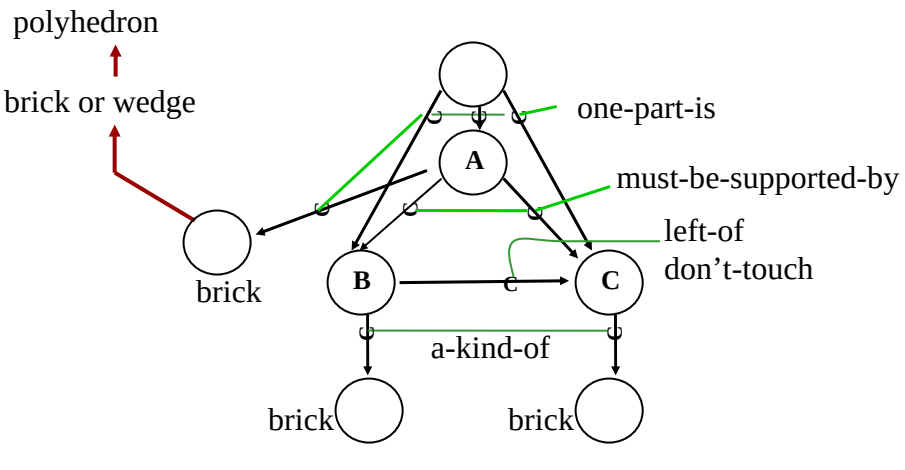
Arch


Arch
A
B
C
Near miss
Near miss


one-part-is
A
must-be-supported-by
left-of
don’t-touch
brick
B
C
a-kind-of
brick
brick

Arch


Arch
A
B
C
Near miss
Near miss
Arch


polyhedron
brick or wedge
brick
A
B
C
one-part-is
must-be-supported-by
left-of
don’t-touch
a-kind-of
brick
brick

ニューロコンピュータ

- 1958 Rosenblatt: Principles of Neurodynamics, Spartan Books
- 1960 **バイオニクス学会設立**
- 1965 Nilsson: Learning Machine
  (1967 **学習機械**)
- 1969 Minsky & Papert: Perceptron, MIT Press

ニューロコンピュータ

- 1975 **福島： 神経回路を持つ多層回路**, **信学論**, 58-D
- 1978 **甘利 神経回路網の数理**, **産業図書**
- 1982 Hopfield: Neural Networks & Physical Systems with Emergent Collective Computational Abilities, Pric. Nat. Academy of Science
- 1986 Rumelhart, et al: Learning Internal Representations by Error Propagation, PDP, 1, MIT Press

多層ネットワークの誤差逆伝播学習

学習望ましい出力　y_{dj}

$r = \sum_j (O_j^m - y_j)^2$　を最小化

$\frac{\partial r}{\partial W_{i,j}^{k-1,k}}$　を用いる

ユニットへの出力

$i_j^k = \sum_i W_{i,j}^{k-1,k} O_i^{k-1}$

出力

$O_j^k = f(i_j^k)$

$f(x) = (1 + e \times p(-x + \vartheta))^{-1}$

$W_{i,j}^{m-1,m} \rightarrow W_{i,j}^{n-1,m-2} \rightarrow \cdots W_{i,j}^{1,2}$

$$\Delta W_{ij}^{k-1\ k} = -\varepsilon \frac{\partial r}{\partial w_{ij}^{k-1\ k}} = -\varepsilon \frac{\partial r}{\partial i_j^k} \quad \frac{\partial r_j^k}{\partial w_{ij}^{k-1\ k}}$$

$k \neq m$　では　　$i_j^k = \Sigma \cdots$　より

O_i^{k-1}となる

$$\sum_\ell \frac{\partial r}{\partial i_\ell^{k+1}} \quad \frac{\partial i_\ell^{k+1}}{\partial o_j^k} \quad \frac{\partial o_\ell^k}{\partial i_j^k}$$

$$= \sum_\ell \frac{\partial r}{\partial i_\ell^{k+1}} \quad W_{j\ell}^{k\ k+1} \quad f'(i_j^k) \qquad \frac{\partial r}{\partial i_j^k} = d_j^k \quad \text{とおけば}$$

$k = m$では

$$\Delta W_{ij}^{k-1\ k} = -\varepsilon \quad d_j^k \quad O_i^{k-1}$$

$$d_j^k = (\sum_\ell \ W_{j\ell}^{k\ k+1} \quad d_\ell^{k+1}) f'(i_j^k)$$

$$d_j^m = 2(O_j^m - y_j) \quad f'(i_j^m)$$

$$\frac{\partial r}{\partial i_j^k} = \frac{\partial r}{\partial o_j^m} \frac{\partial o_j^m}{\partial i_j^m}$$

まず　$\Delta W_{ij}^{m-1\ m} = -\varepsilon \quad d_j^m \quad O_i^{m-1}$

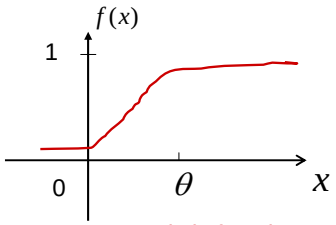
計算可 or 観測可

$W_{ij}^{k\ k+1}$,　d_ℓ^{k+1}がわかれば　d_j^kは計算可

$$f(x) = \frac{1}{1 + e \times p(-x + \vartheta)}$$

$$f'(x) = f(x)(1 - f(x))$$

$$f'(i_j^k) = O_j^k(1 - O_j^k)$$



Logistic function

データマイニング(Data Mining)

- **あるいは、**
  KDD(Knowledge Discovery in Databases)
- **データマイニングのプロセス**
  データ獲得：問題を理解して目標データを決める、
  データ選択：データ集合から選択、
  前処理：ノイズ除去、
  データ変換、
  狭義のデータマイニング：ルールや決定木やクラスタを得る、
  解釈と評価：最終的に知識を得る

相関ルールマイニング

データマイニング手法の1つに相関ルールマイニング（またはマーケットバスケット分析）がある。
相関ルールとは、例えばデパートのデータベース（トランザクションデータベース）において、
『枕を購入した客のうちの98%は枕カバーも購入しており、枕と枕カバーの商品（アイテム）の組み合わせ（アイテムセット）を 購入した件数は買い物件数全体の8%である』というようなルールである。

$I = \{ i_1, i_2, \ldots, i_N \}$ をアイテム全体の集合とし、D をトランザクションデータベースとする。D 中の各トランザクション t は, $t \subseteq I$ となるアイテム集合である。
相関ルールとは、アイテムセット X ($\subseteq I$) と Y ($\subseteq I$) により、

$$X \Rightarrow Y$$

の形で記述される関係である。ここで X と Y は $X \cap Y = \phi$ である。
$X \Rightarrow Y$ の関係は、『トランザクションがアイテムセット X を含むならば、アイテムセット Y も含む』ということを表す。

あるアイテムセット X について、D のうちの s % のトランザクションが X を含むとき、アイテムセット X は s % の**支持度** (support) を持つという。また、相関ルール $X \Rightarrow Y$ については、アイテムセット $X \cup Y$ の支持度を相関ルール $X \Rightarrow Y$ の支持度と定義する。
さらに、X を含むトランザクションのうちの c % のトランザクションが Y も含むとき、相関ルール $X \Rightarrow Y$ は c % の**確信度** (confide) を持つという。