

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



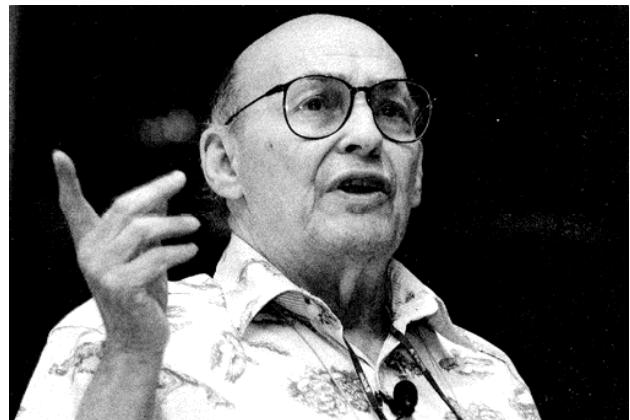
初期のAIの歴史

- 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

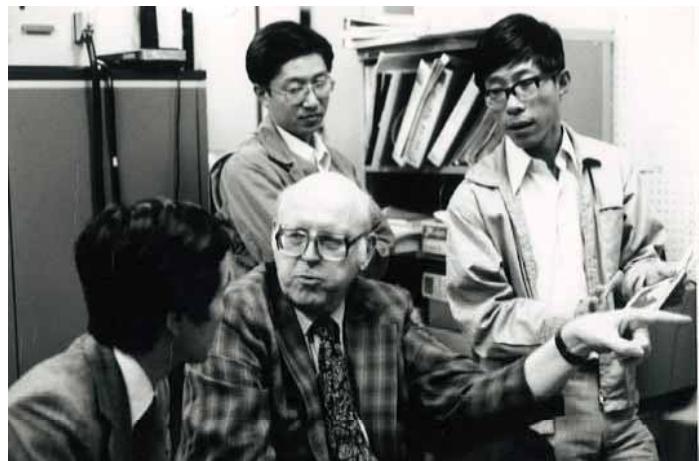
2005年米国AI学会での講演



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

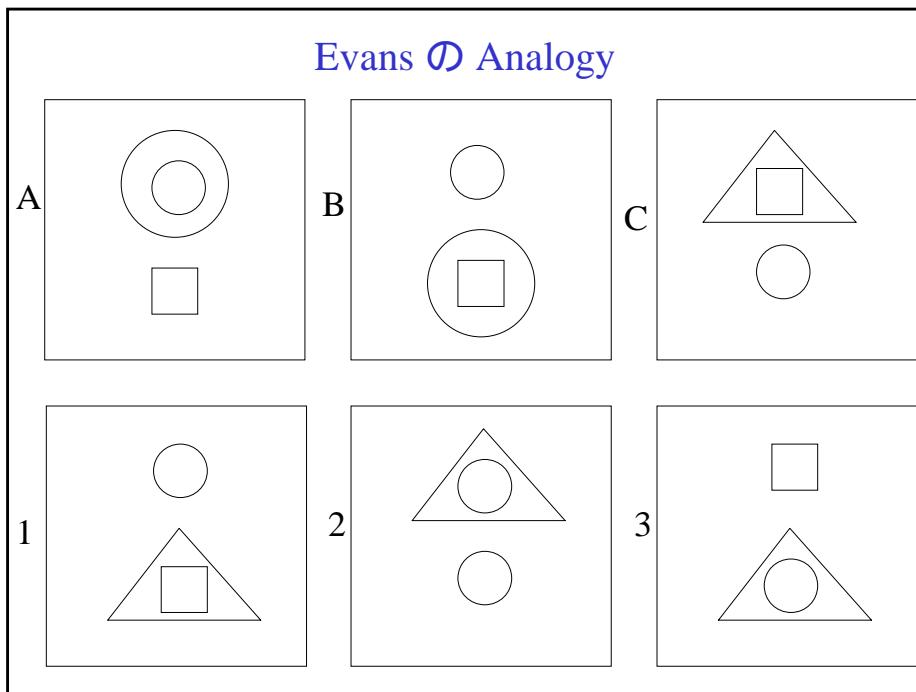
- Green et al: “BASEBALL” An Automatic Question-Answer, WJCC, 1961
- Newell & Simon: Computer Simulation of Human Thinking and Problem-Solving, Computer and Automation, 10, 1961
- Slagle: A Computer Program for Solving Problems in Freshman Calculus (SAINT), Thesis M.I.T., 1961
- Minsky: Steps Toward 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



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)

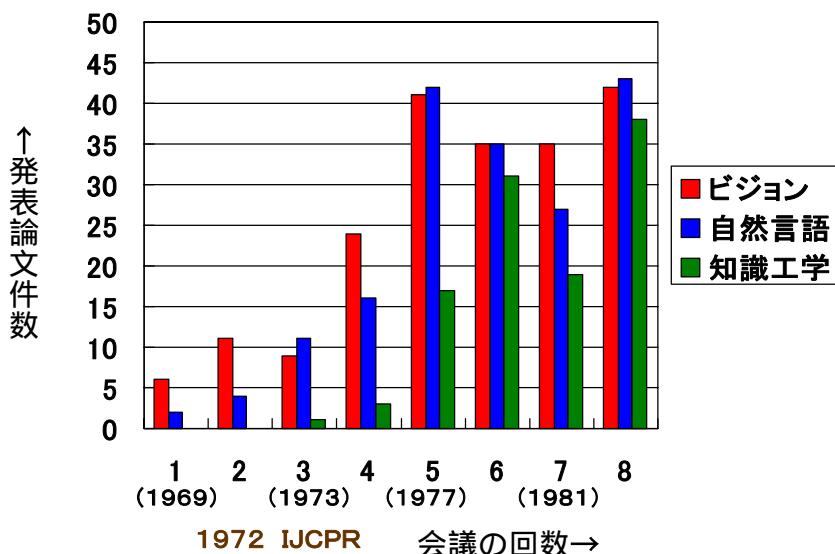
1963 Computer and Thought (Feigenbaumなど)

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



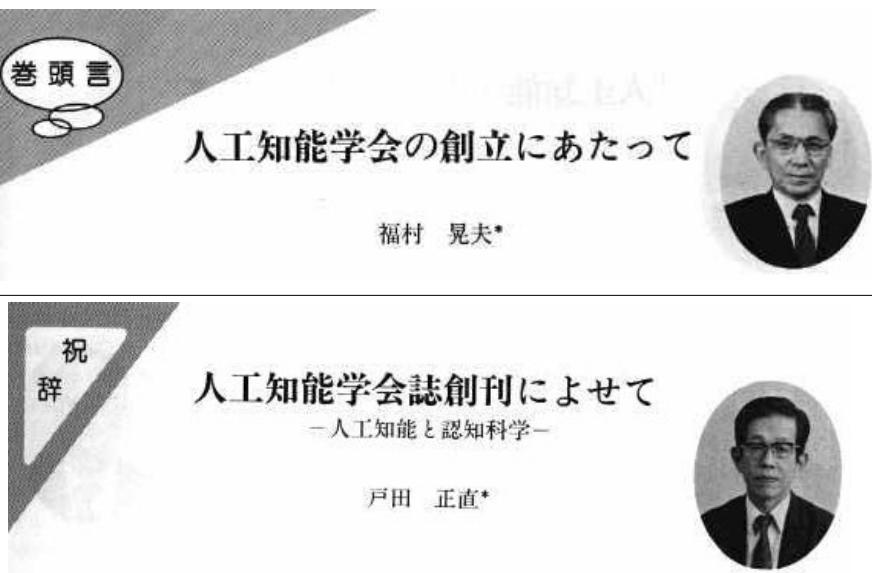
論文数比較

	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)	—	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)	—	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

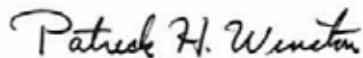
人工知能学会誌創刊号(1986年9月)



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.

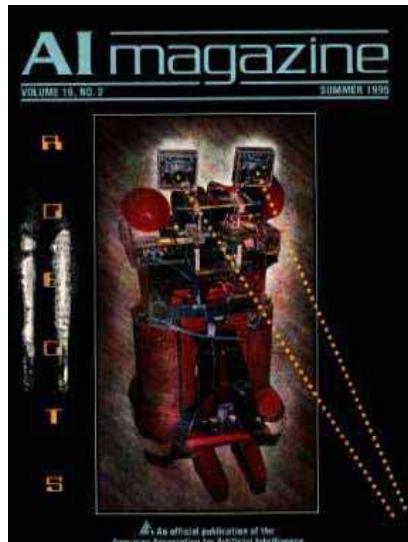


Professor Patrick H. Winston
AAAI President(1985-1987)

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

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。

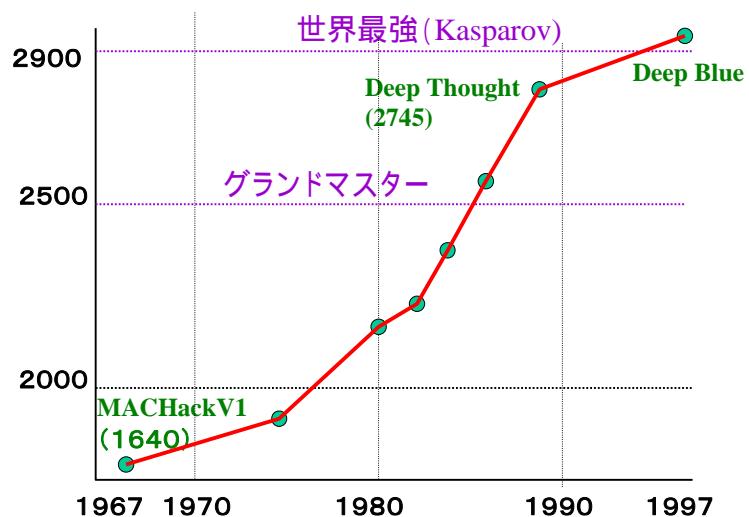
形式重視 → 内容重視

The Hottest Jobs of the Future

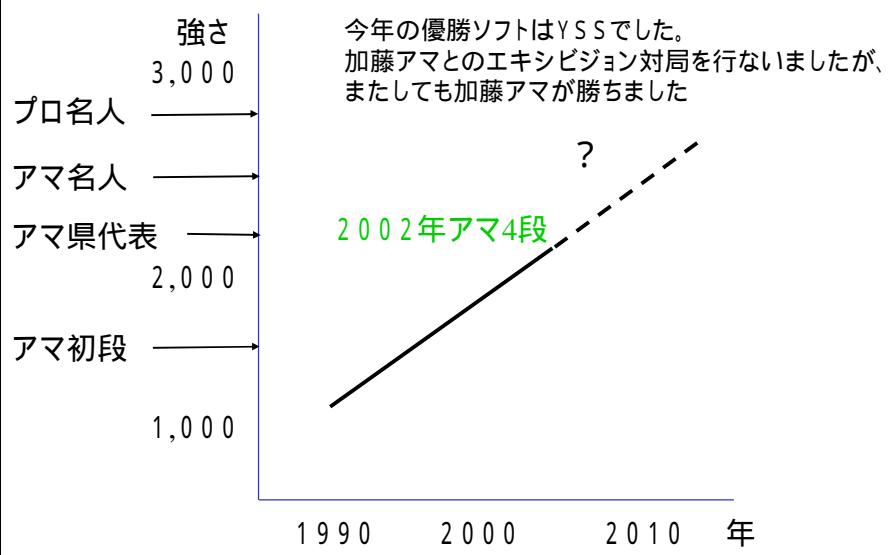
Time May 11, 2000

- **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



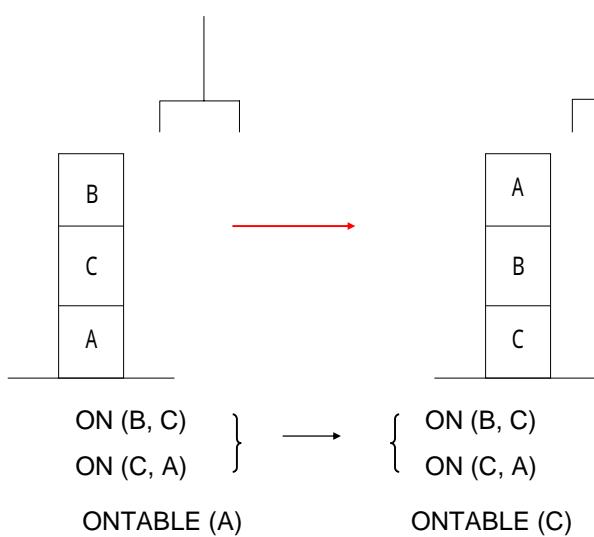
コンピュータの将棋の能力



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

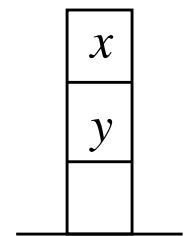
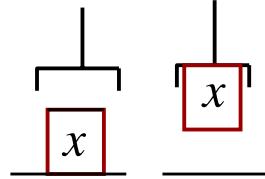
追加リスト: CLEAR(y), HOLD(x)

PUTON(x,y) ; x を y の上へ置く

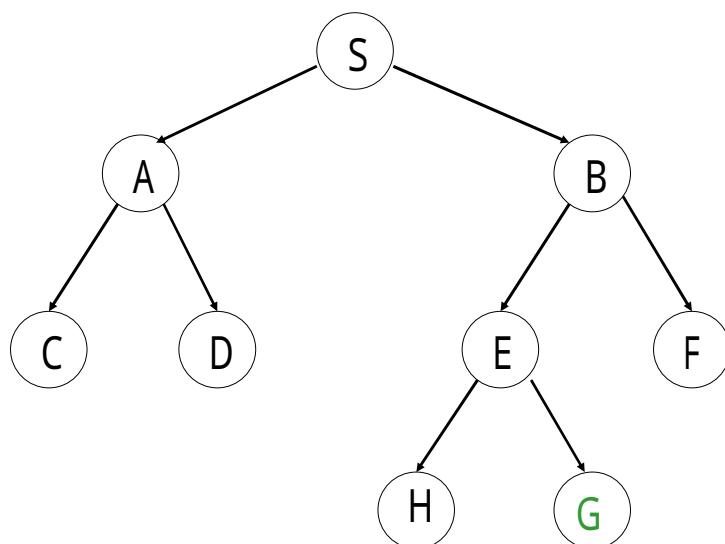
前提条件: HOLD(x), CLEAR(y)

削除リスト: HOLD(x), CLEAR(y)

追加リスト: ON(x,y), CLEAR(x), EMPTY



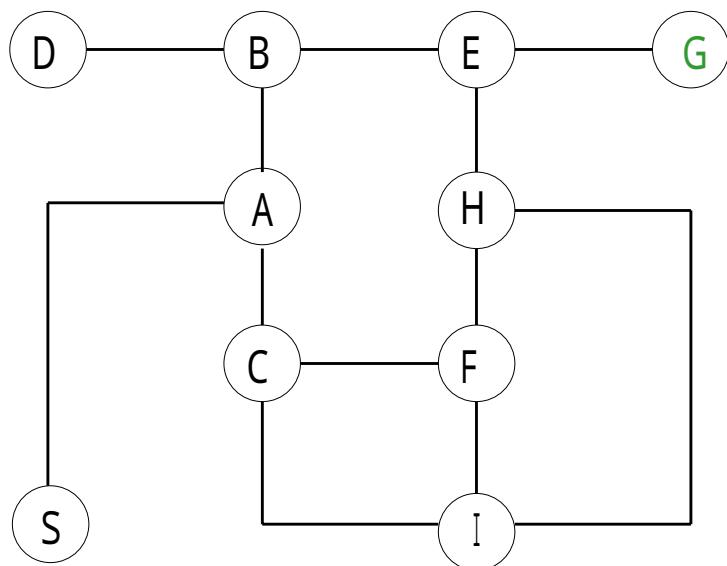
木の探索



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



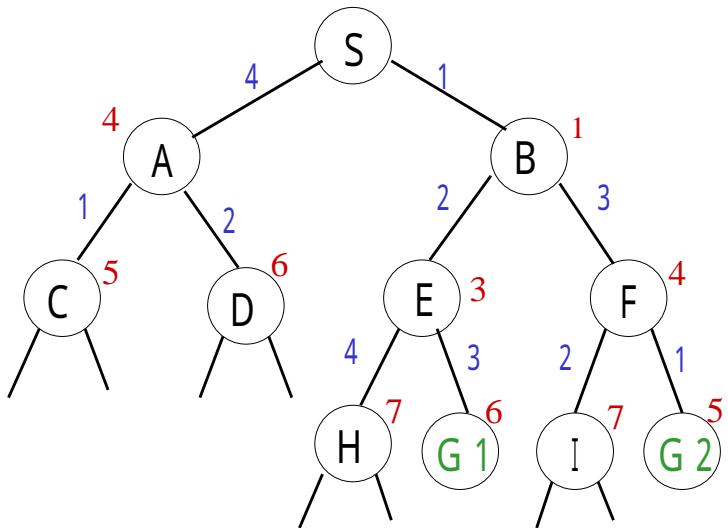
procedure depth-first ~~tree~~ search

- 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.

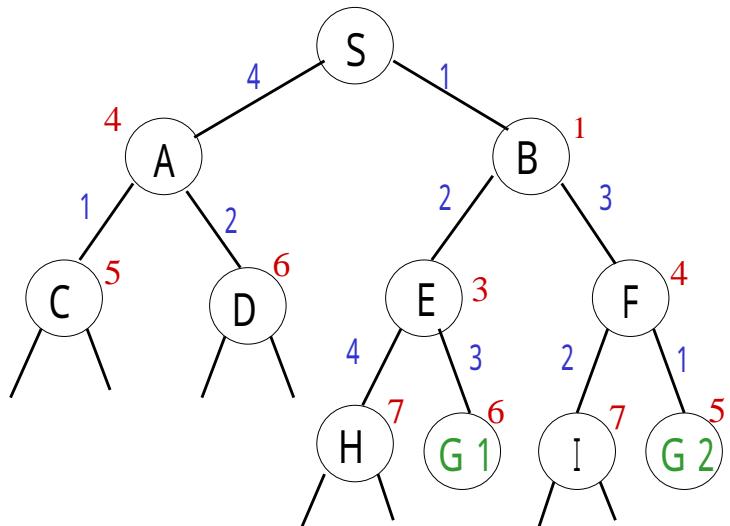
最適経路の探索



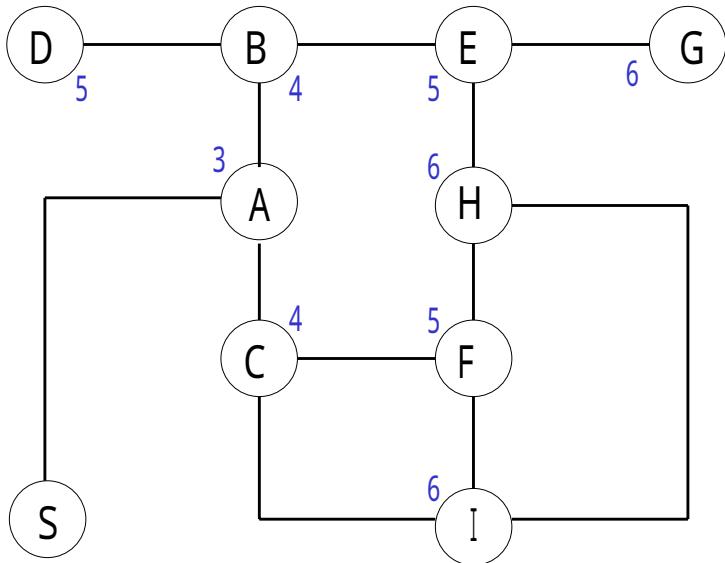
procedure optimal tree search

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

最適経路の探索



グラフ最適経路の探索



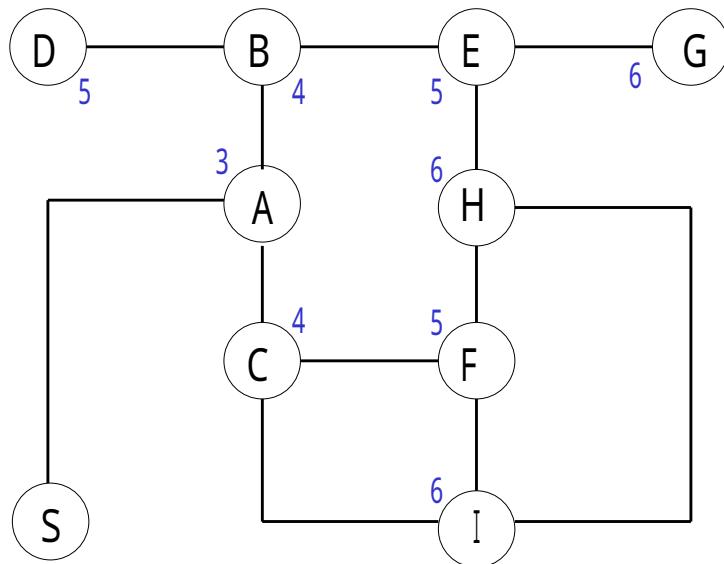
procedure optimal ~~tree~~ search

- add (s, **open**); $g^\wedge(s) := 0$;
- LOOP: if **open** = null then exit(fail);
- $n := \text{first}(\text{open})$;
- If $\text{goal}(n)$ 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^\wedge ;
- Go to LOOP.

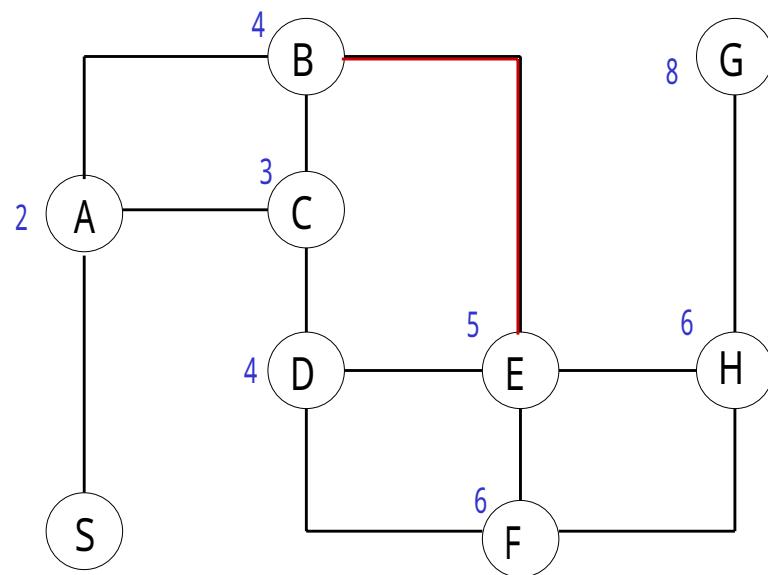
procedure optimal ~~tree~~ search

- add (s, **open**); $g^\wedge(s) := 0$;
- LOOP: if **open** = null then exit(fail);
- $n := \text{first}(\text{open})$;
- If $\text{goal}(n)$ 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 $g^\wedge(n, j) < g^\wedge(j)$,
then attach pointers to n ;
- Sort nodes in **open** in increasing order of g^\wedge ;
- Go to LOOP.

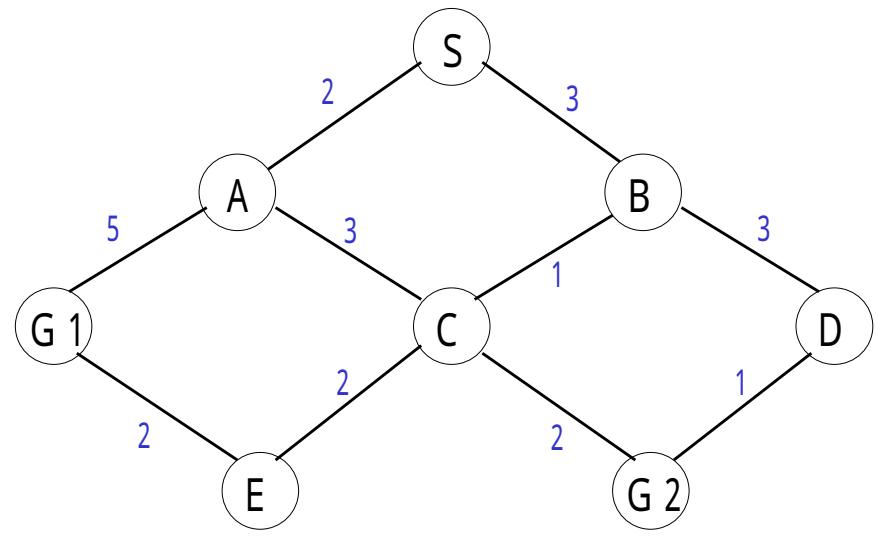
グラフ最適経路の探索



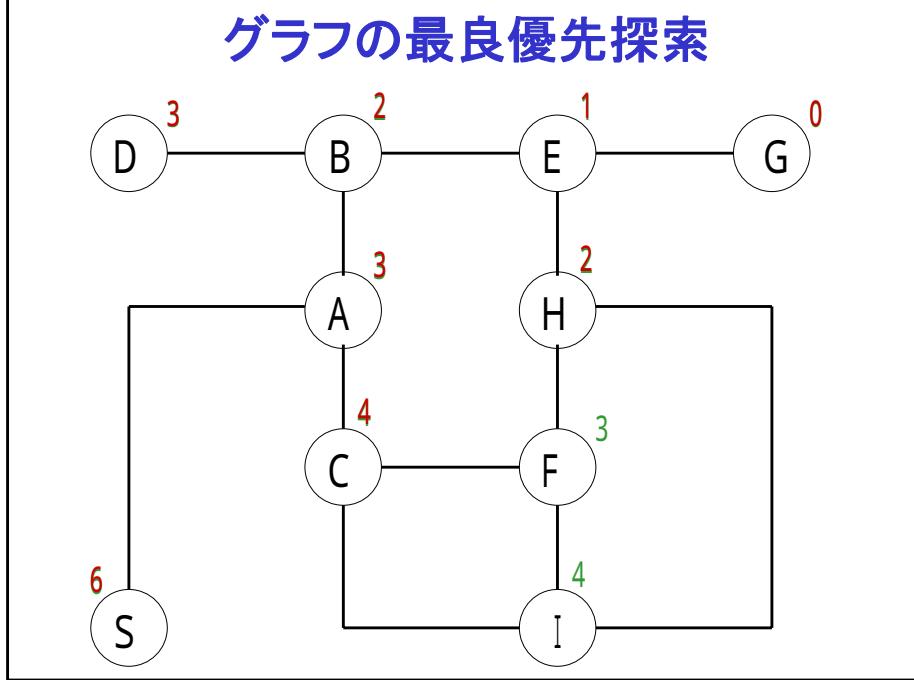
グラフ最適経路の探索



ゴールをopenに入れる例



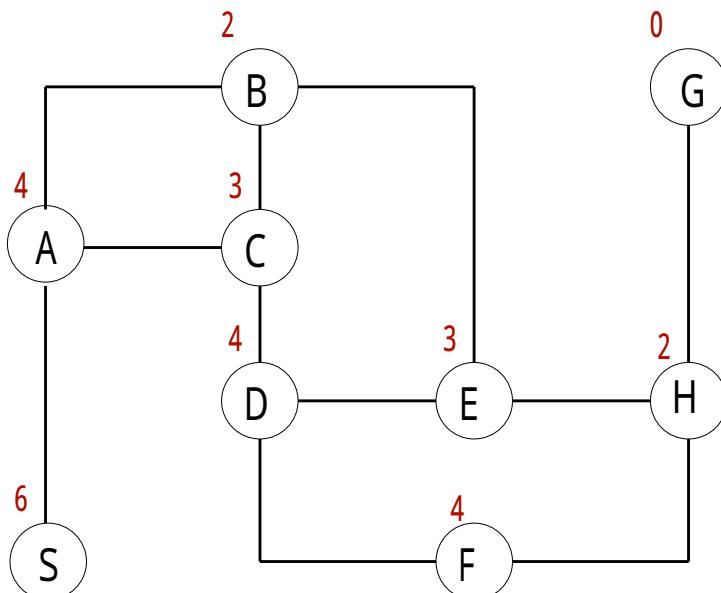
グラフの最良優先探索

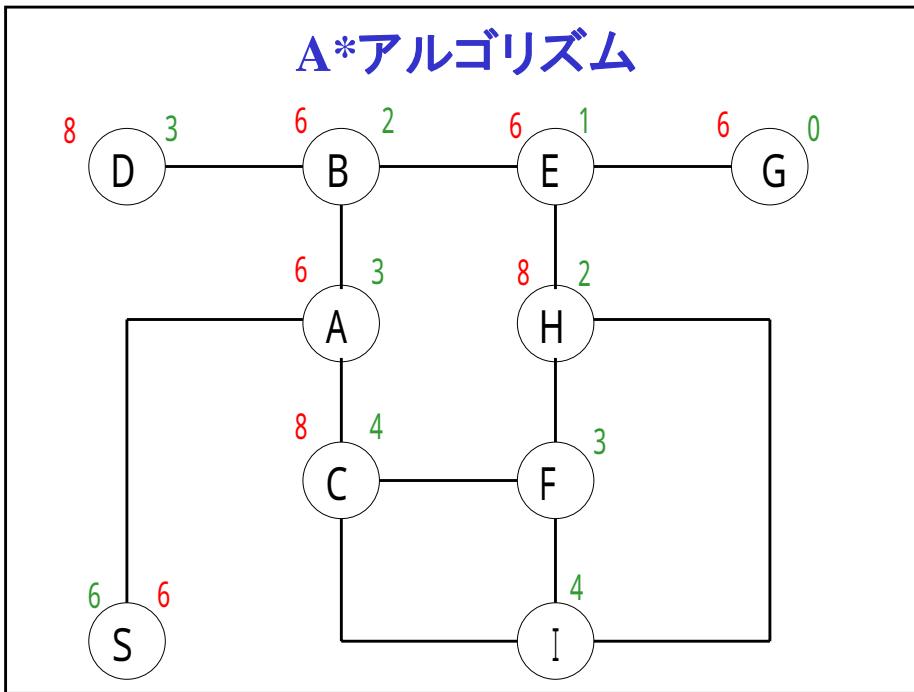


procedure best-first search

- 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^{\wedge} ;
- Go to LOOP;

グラフの最良優先探索





procedure optimal ~~tree~~ search

- add (s, **open**); $g^\wedge(s) := 0$;
- LOOP: if **open** = null then exit(fail);
- $n := \text{first}(\text{open})$;
- If $\text{goal}(n)$ 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 $\min\{g^\wedge(j), g^\wedge(n, j)\}$,
 and attach pointers to n ;
- Sort nodes in **open** in increasing order of g^\wedge ;
- Go to LOOP.

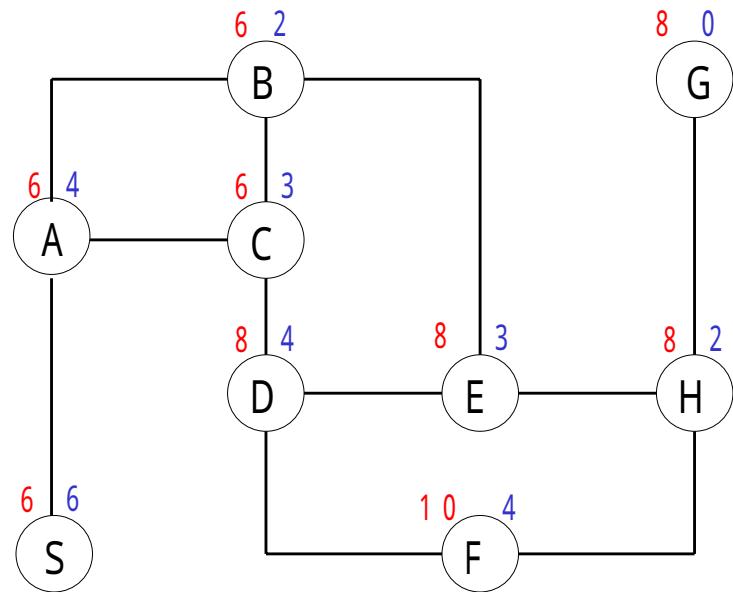
procedure optimal search

- add (s, **open**); $g^\wedge(s) := 0$;
- LOOP: if **open** = null then exit(fail);
- $n := \text{first}(\text{open})$;
- If $\text{goal}(n)$ 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 $\min\{g^\wedge(j), g^\wedge(n, j)\}$, and attach pointers to n ;
- Sort nodes in **open** in increasing order of g^\wedge ;
- Go to LOOP.

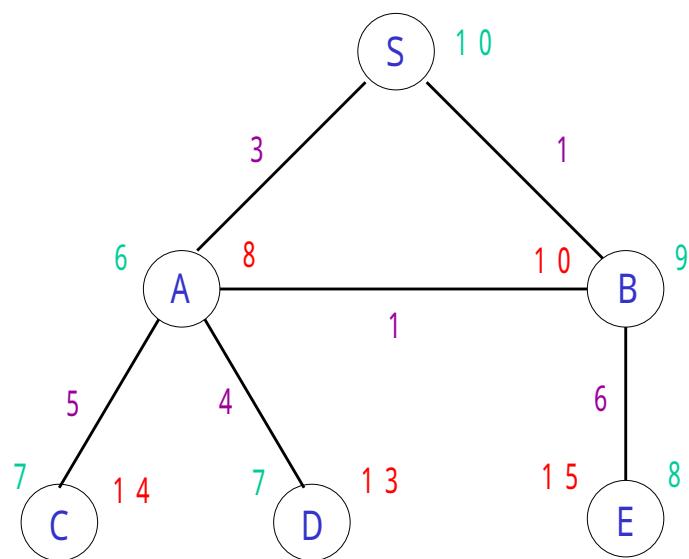
procedure A-algorithm search

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

A*アルゴリズム



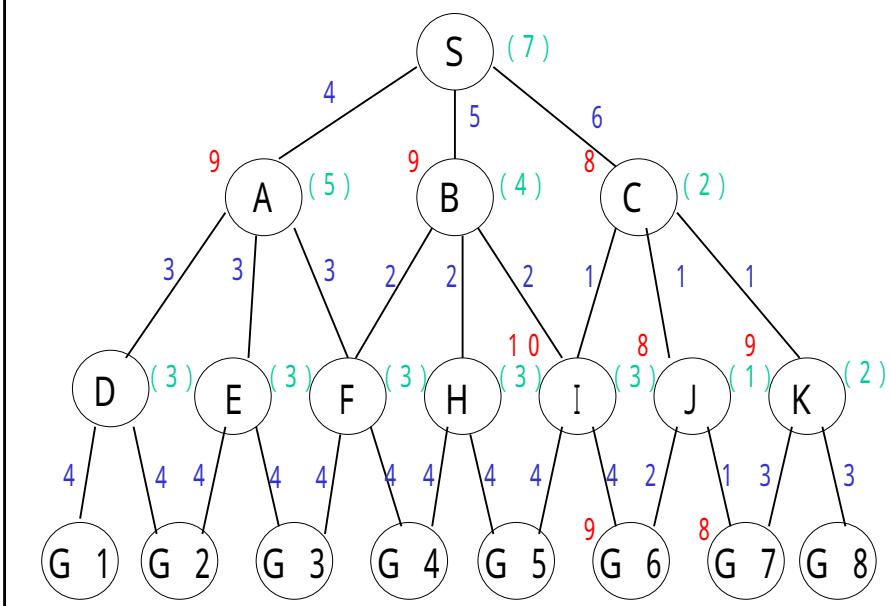
CLOSED, OPEN を用いる例



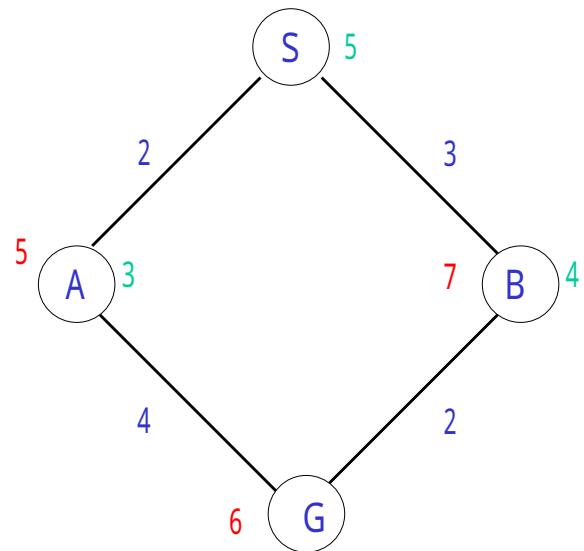
procedure A-algorithm search

- add (s, **open**); $f^\wedge(s) := h^\wedge(s)$;
- LOOP: if **open** = null then exit(fail);
- $n := \text{first}(\text{open})$;
- 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 ;
 $f^\wedge(n, j) := g^\wedge(n, j) + h^\wedge(n)$; if node j is in **open** or **closed**, select $\min\{f^\wedge(j), f^\wedge(n, j)\}$ and put into **open**;
- Sort nodes in **open** in increasing order of f^\wedge ;
- Go to LOOP;

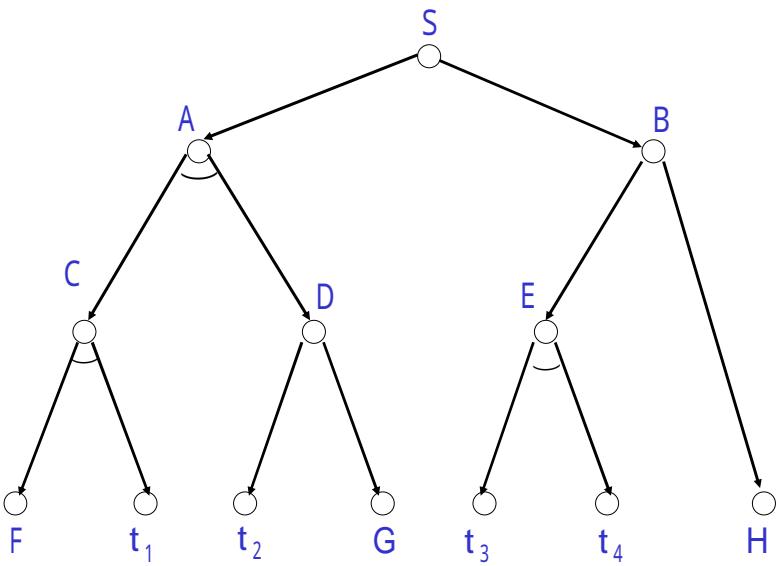
A*アルゴリズムが有効な例

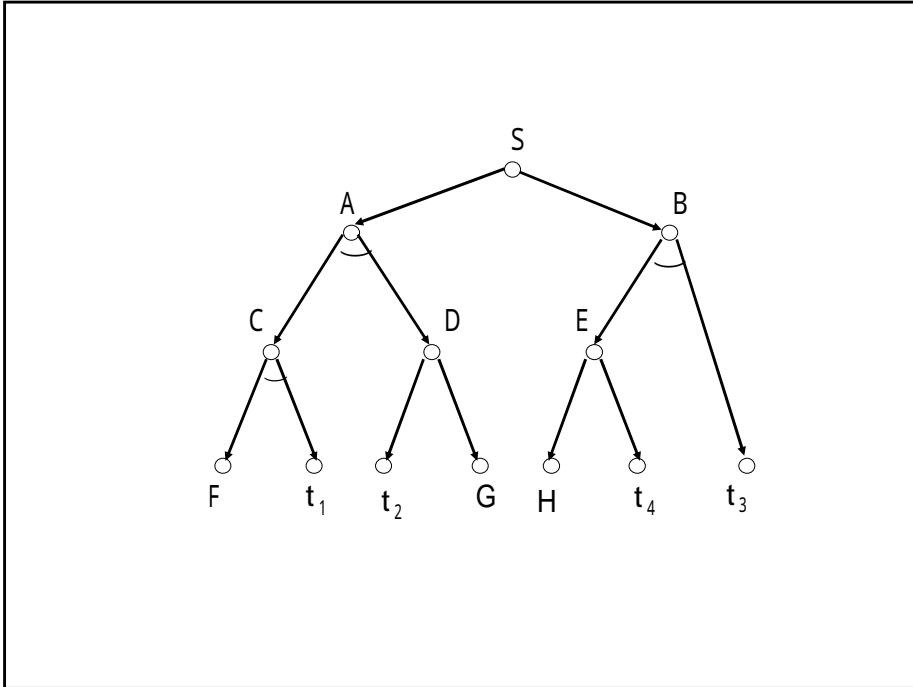


A*アルゴリズムとならない例



AND-OR木の探索



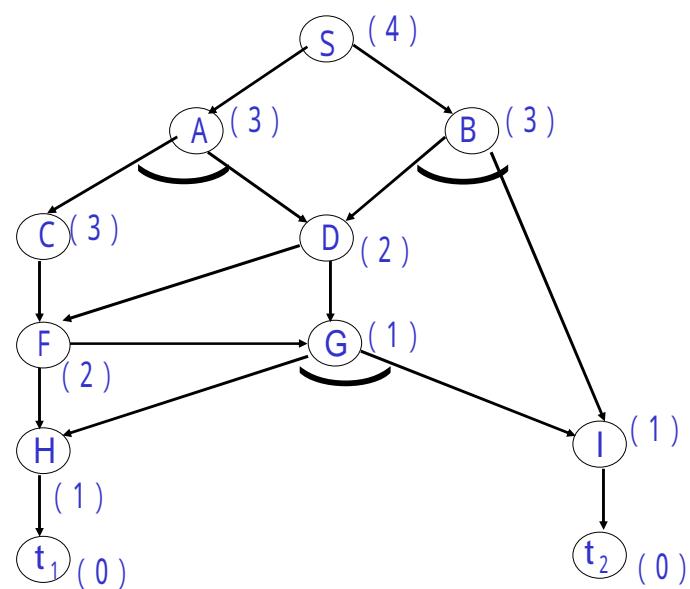


procedure depth-first and-or

- 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);
- $p:= \text{first}(\text{open})$;
- 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

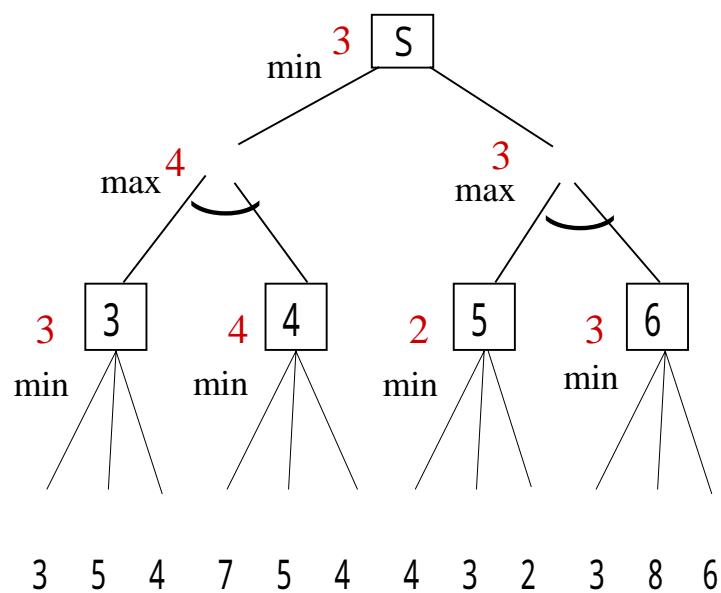
例3.1 この手続きを図3.3AND/ORグラフに適用して時のopenリスト内の部分解グラフを表3.1に示す。3回目の繰返しのときに、...のところ、3回目を4回目に

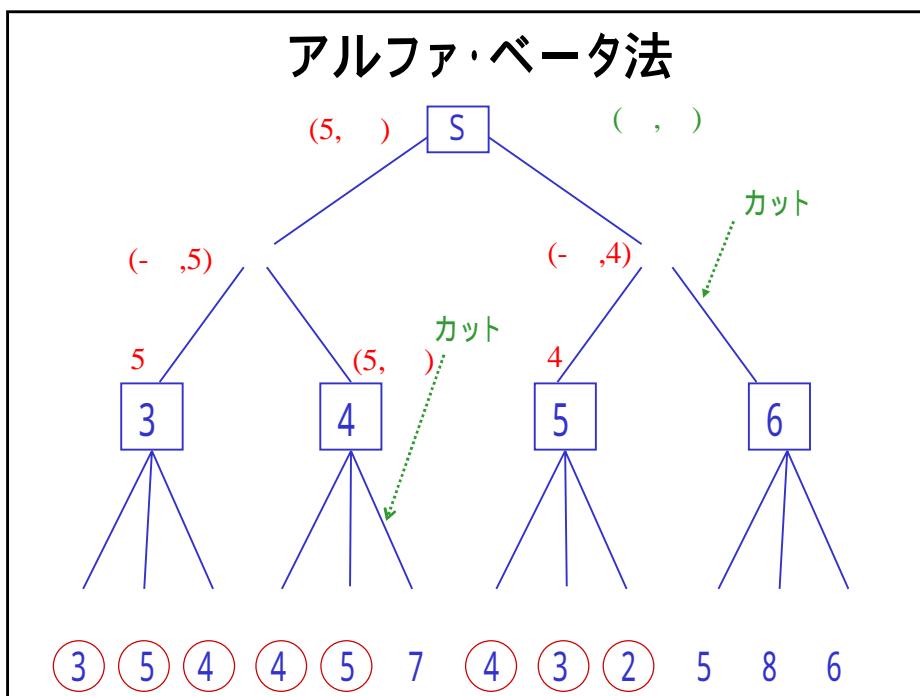
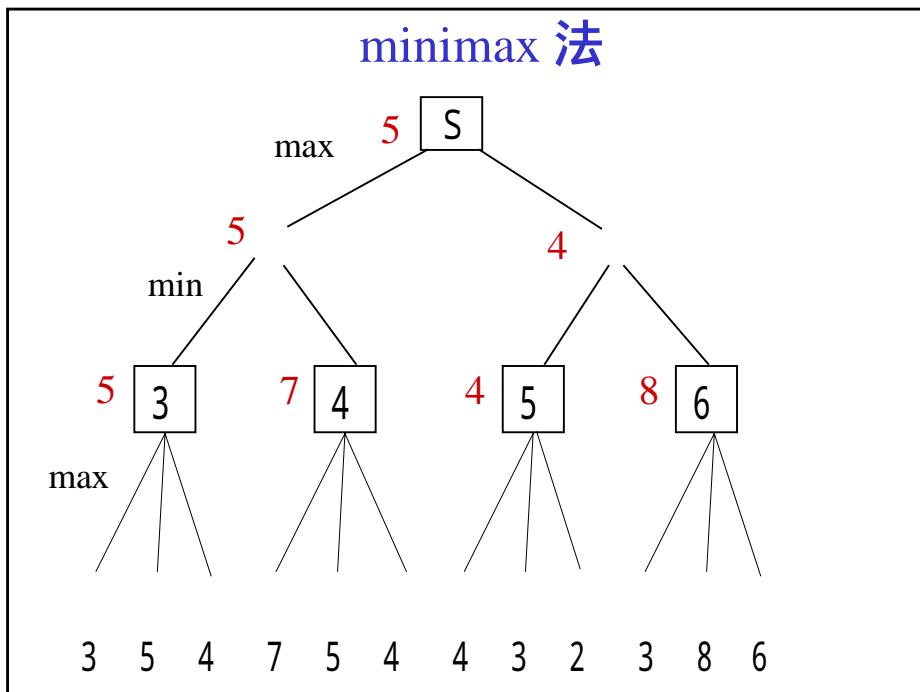
...

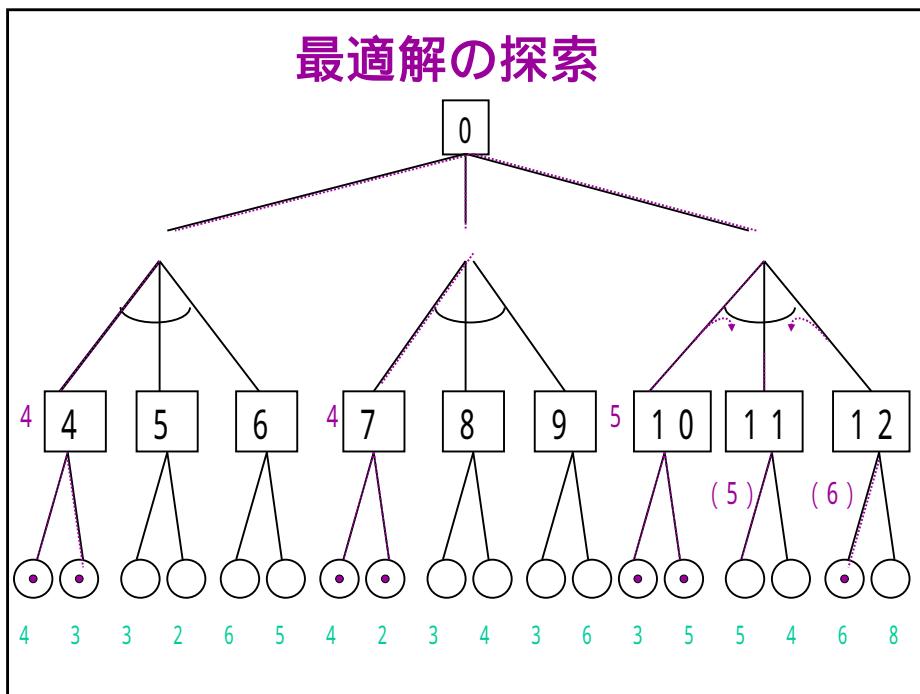
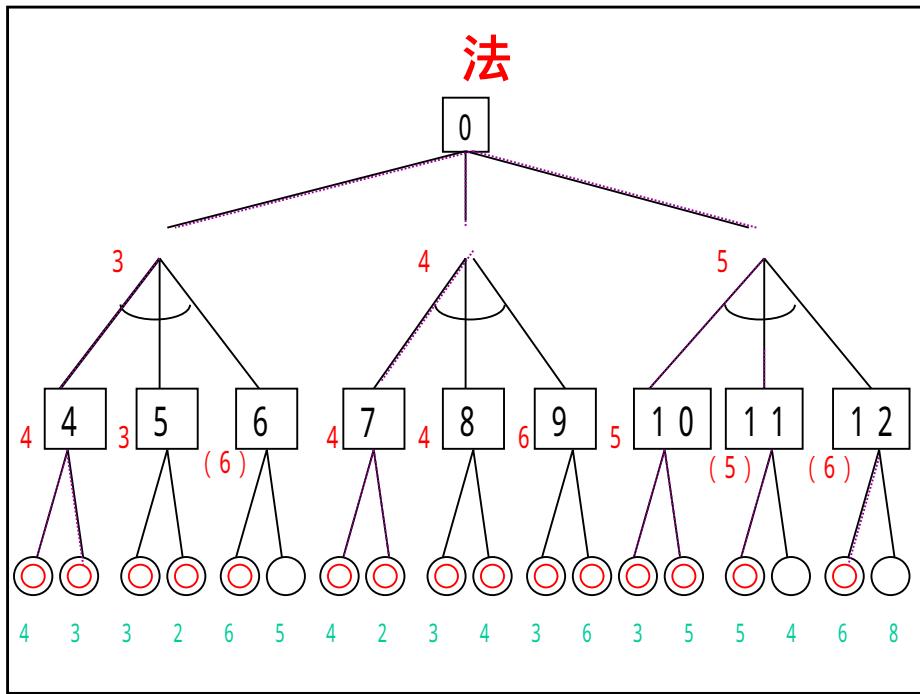
フはopenに入れられない。6回目の...の6回目を7回目に

2) 表3.1は、correction に修正。

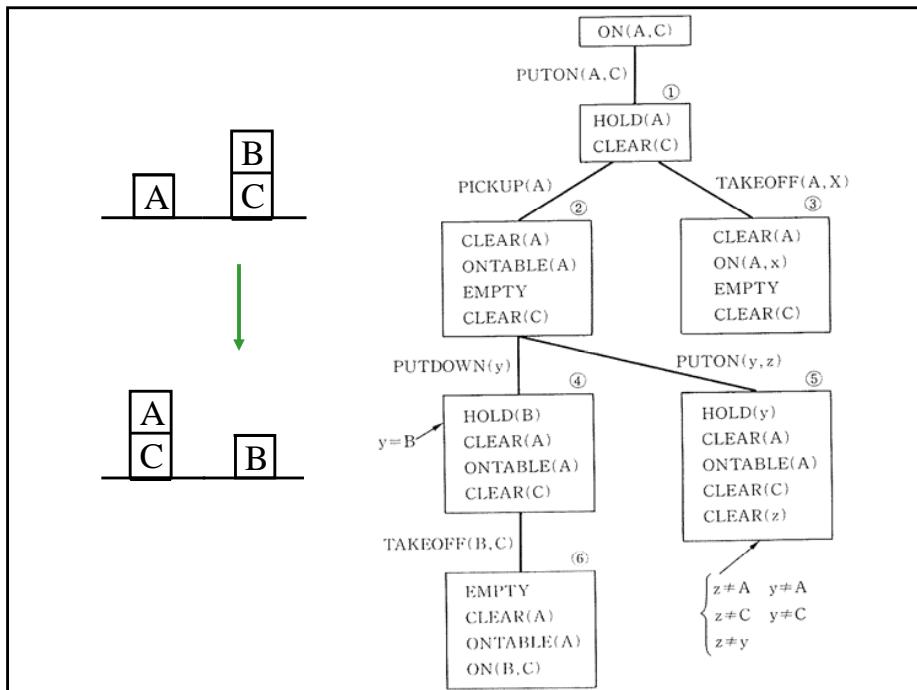
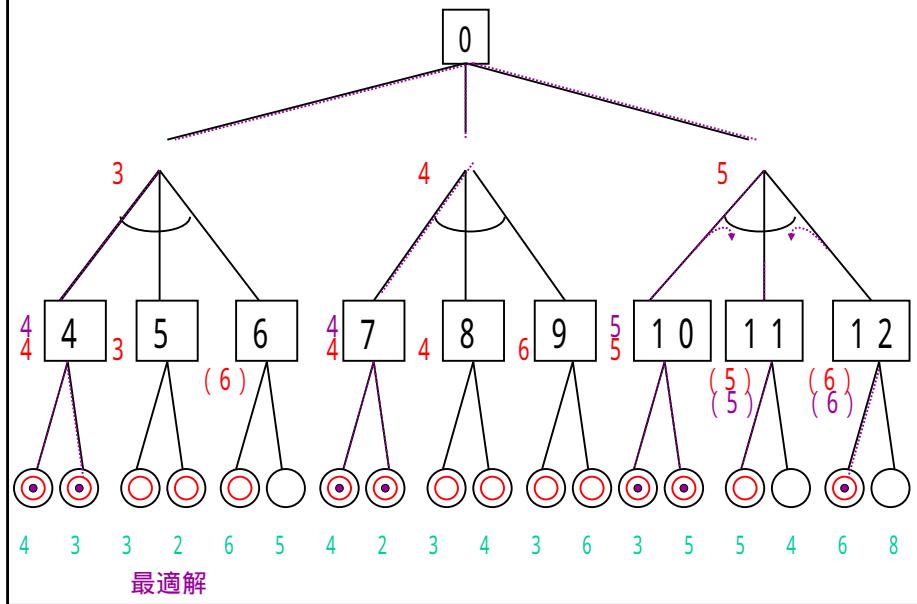
道のコストがない場合







法と最適解の探索との比較



(出所) 『人口知能の理論』白井良明 (2001) コロナ社

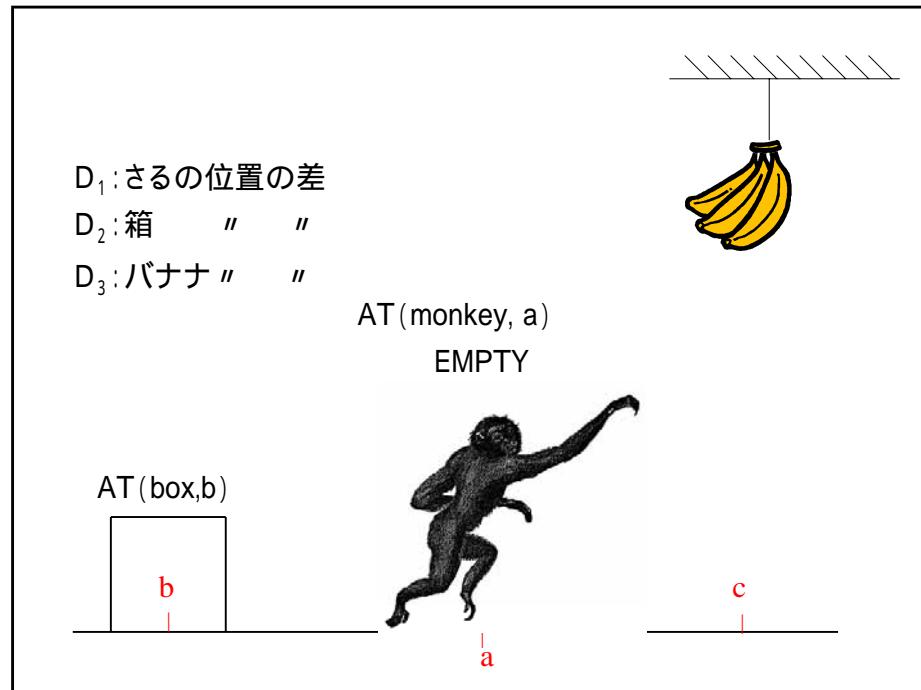


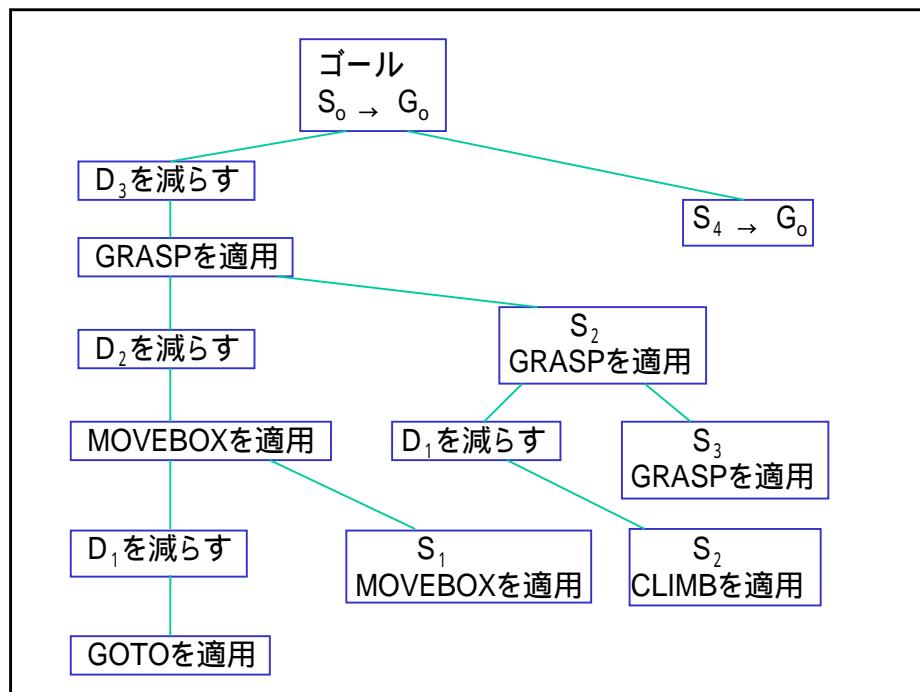
表4.1 差異と差異を縮めるオペレータ

差異 \ オペレータ	GOTO	MOVE BOX	CLIMB	GRASP
D_1 (猿の位置)	0	0	0	
D_2 (箱の位置)		0		
D_3 (猿の手の状態)				0

(出所) 『人口知能の理論』白井良明 (2001) コロナ社

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

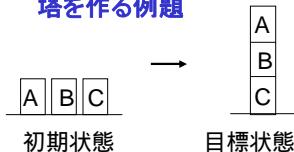
階層的計画のオペレータ

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

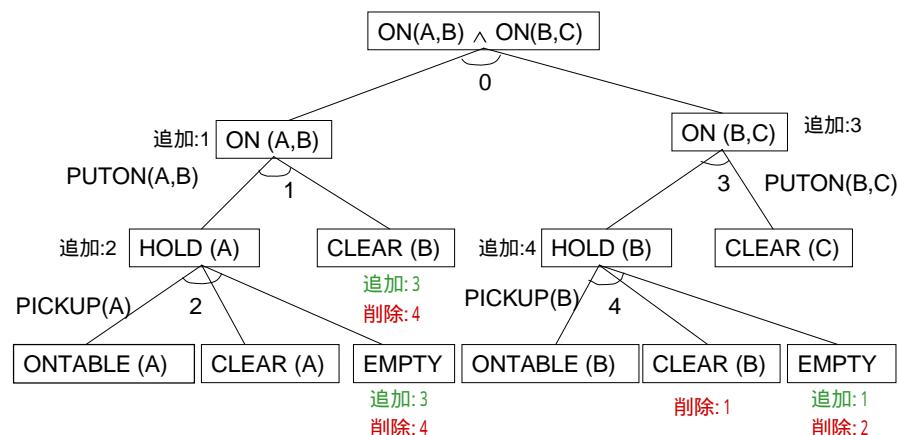
階層的計画のオペレータ

- | | |
|-----------------------|------------|
| • GOTO(u) | 計画 |
| • MOVEBOX(v) | GOTO(b) |
| – 前提条件: (3)AT(box, x) | MOVEBOX(c) |
| – (1)AT(monkey, x) | |
| • CLIMB | CLIMB |
| – 前提条件: (3)AT(box, x) | |
| – (1)AT(monkey, x) | GRASP |
| • GRASP | |
| – 前提条件: (3)AT(box, c) | |
| – (2)ON(monkey, box) | |

塔を作る例題



複数の目標のAND/OR木



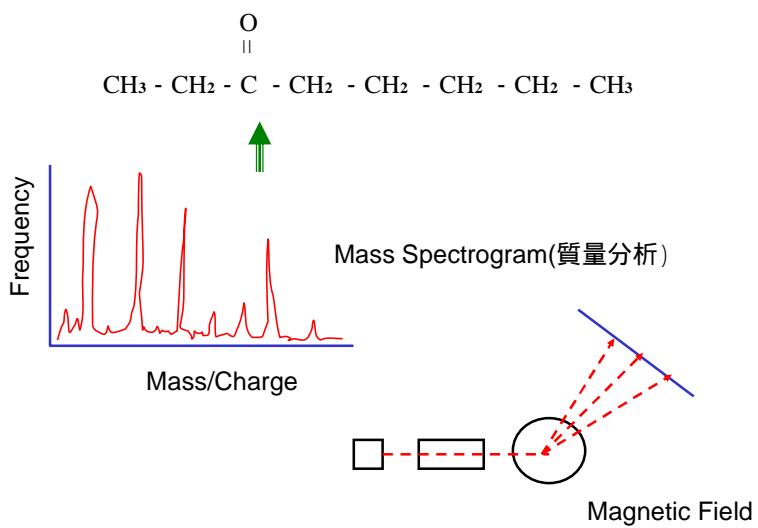
初期のAIの歴史

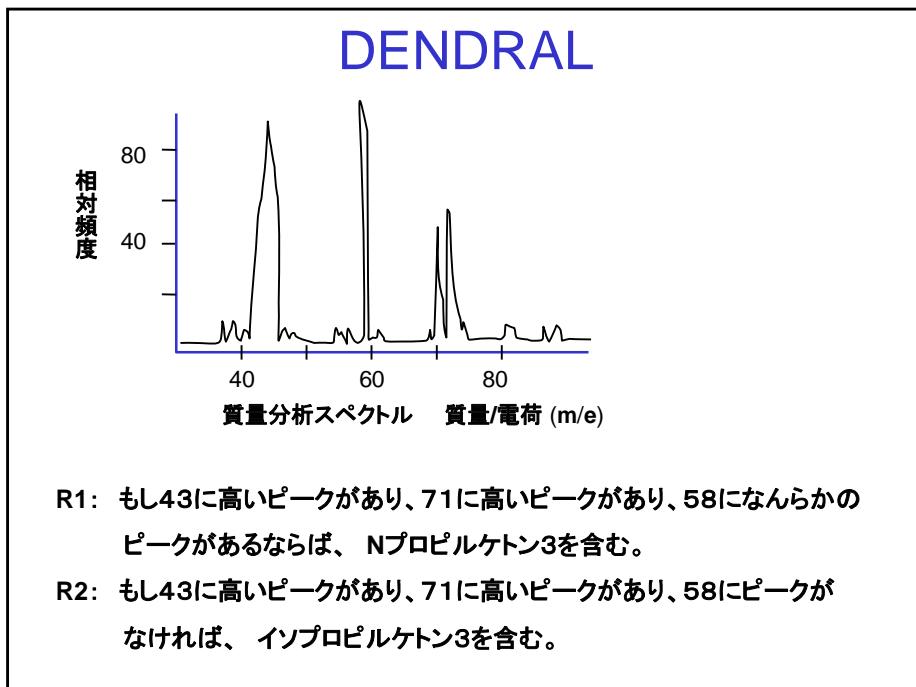
- 1956 AIの研究会 (Minsky, Shannon, Newell, Simonなど)
- 1959 AI Lab . 設立 (Minsky, McCarthy)
- 1961 AI へのステップ (Minsky)
- 1963 Computer and Thought (Feigenbaumなど)



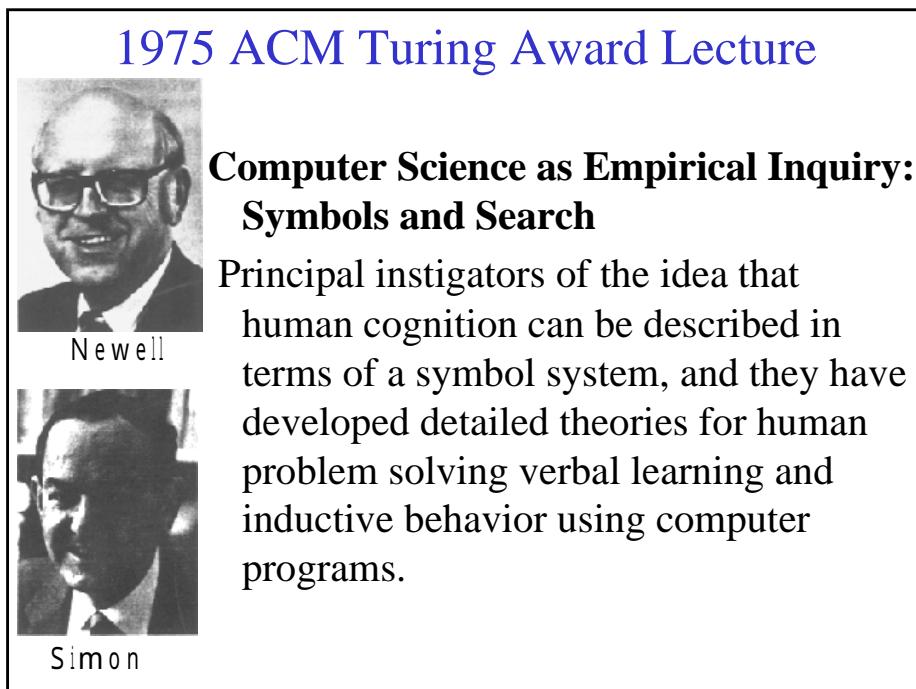
リンダーバーグ(遺伝学)

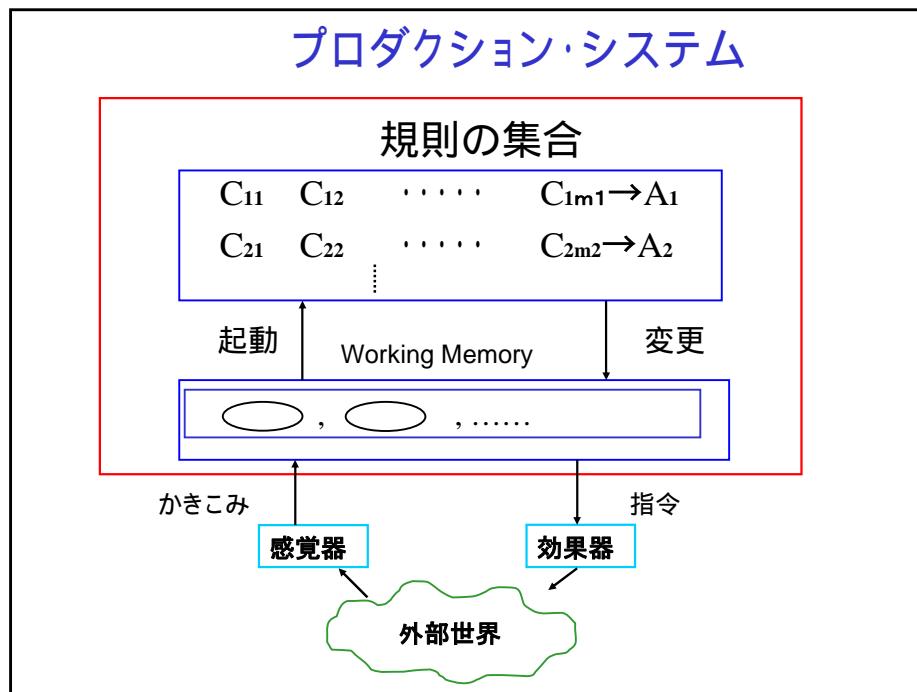
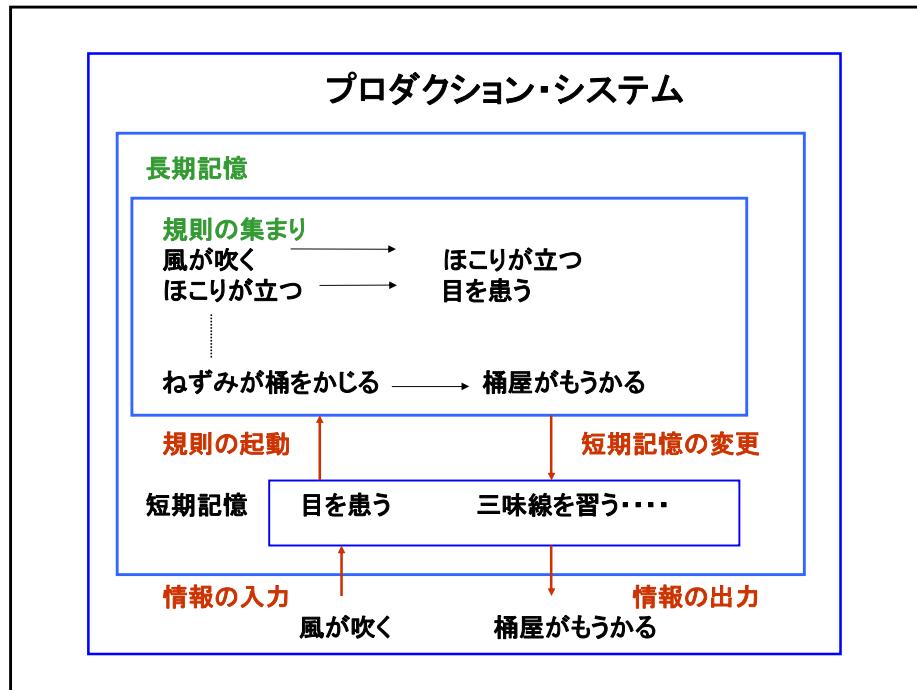
知識工学の誕生 (1970-1979)



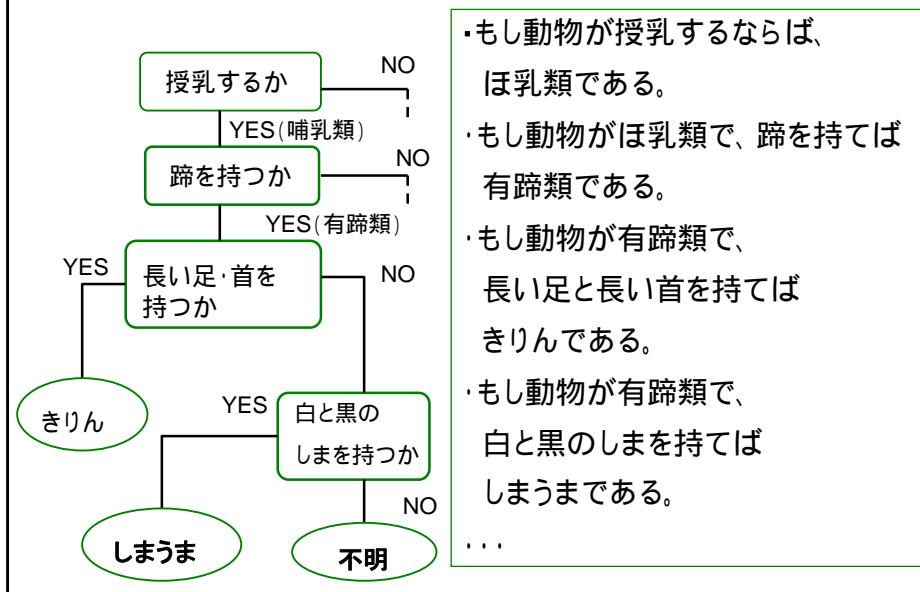


(出所) 『人口知能の理論』白井良明 (2001) コロナ社

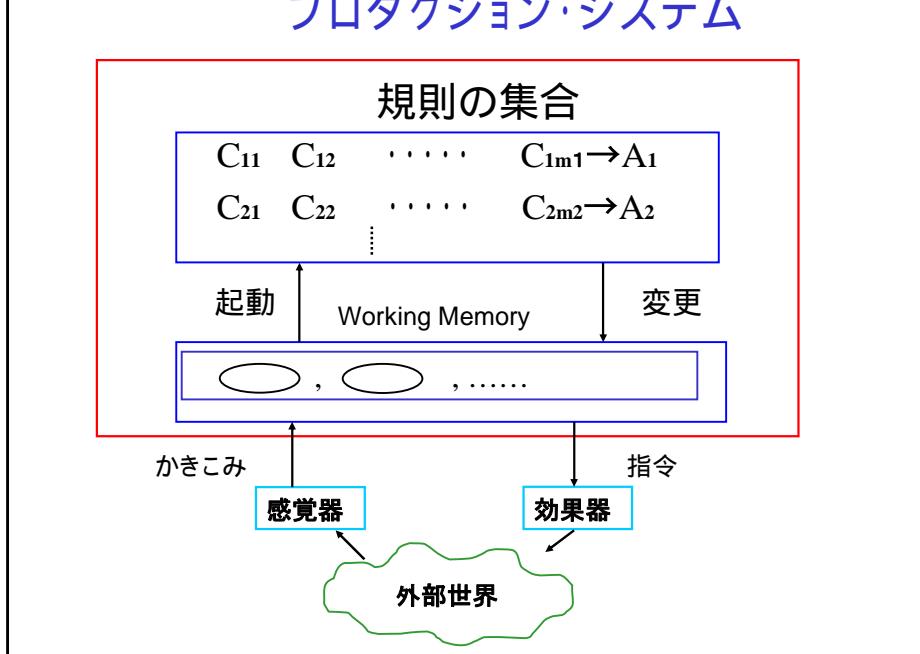




プロダクションシステムと従来プログラムとの比較



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。項目がついた規則は捨てる。

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

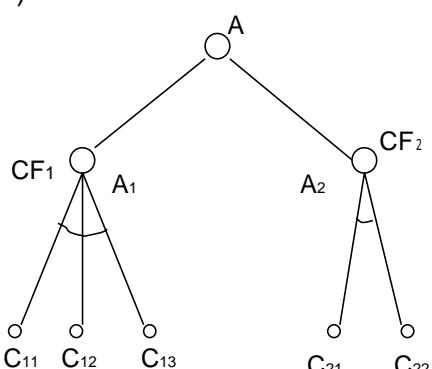
$R_1: C_{11} \ C_{12} \ \dots \ C_{1n} \rightarrow A_1(CF_1)$

事実の確信度

C_{ij} (CF_{ij})

確からしさ
Certainty Factor
確信度
信頼性係数

AND/OR木



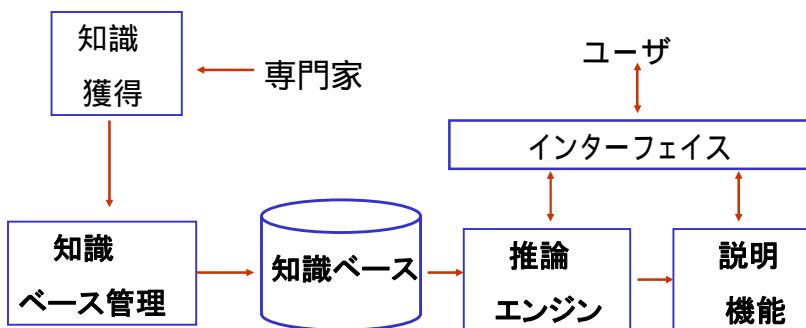
e-Learning の教材 (上野@情報学研究所)

- 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、見たい科目的「見る」をクリック。

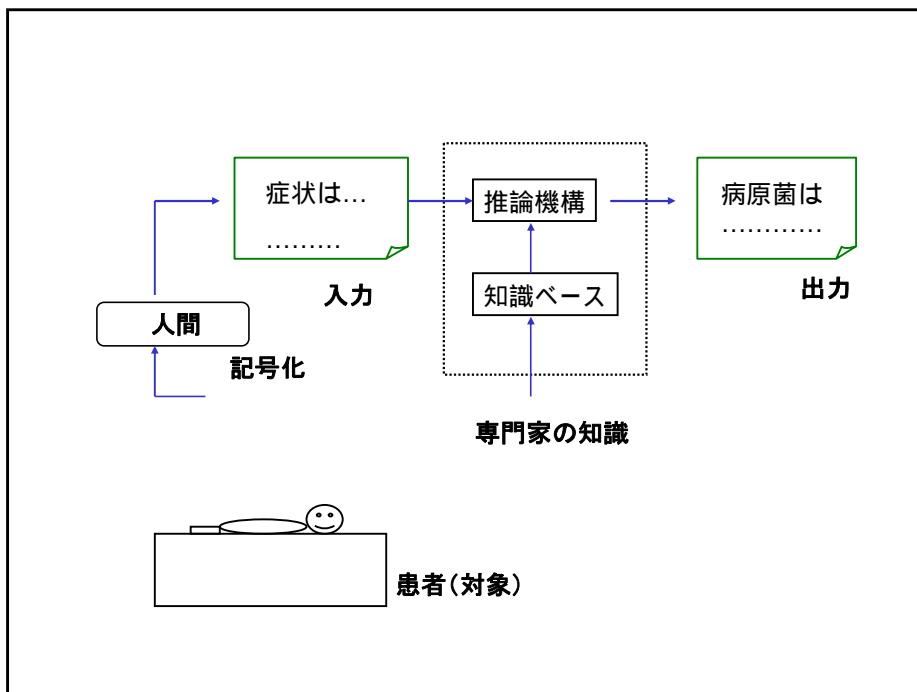
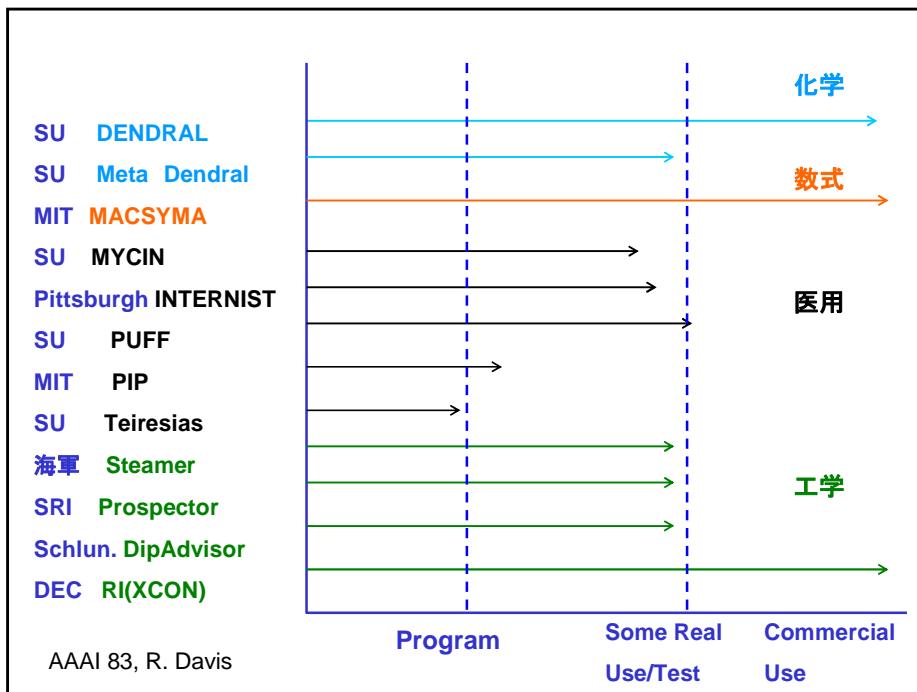
Knowledge Modeling and Reasoning I (情報学専攻 上野)
(AIとは)

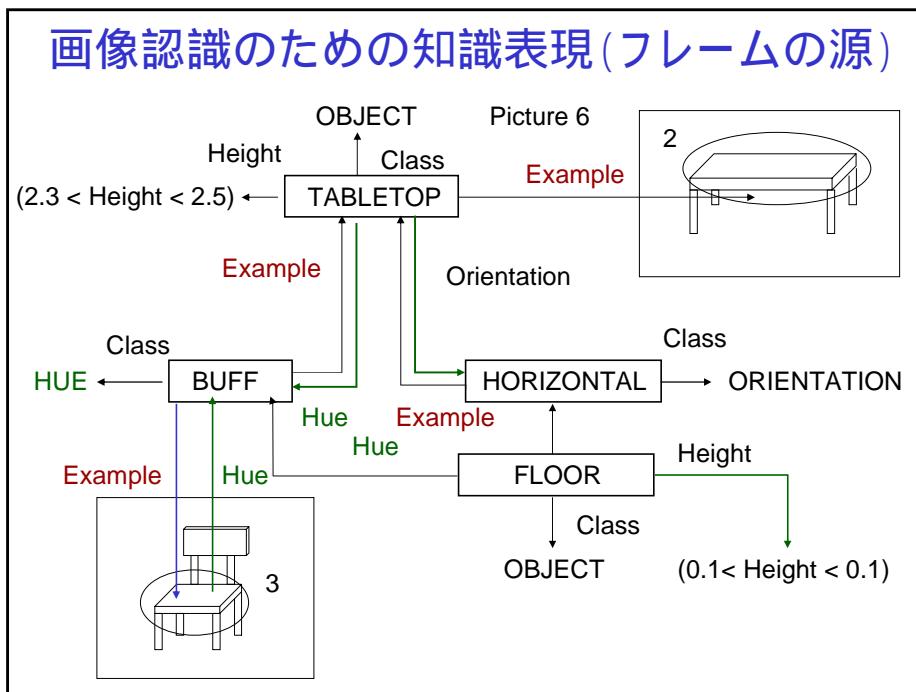
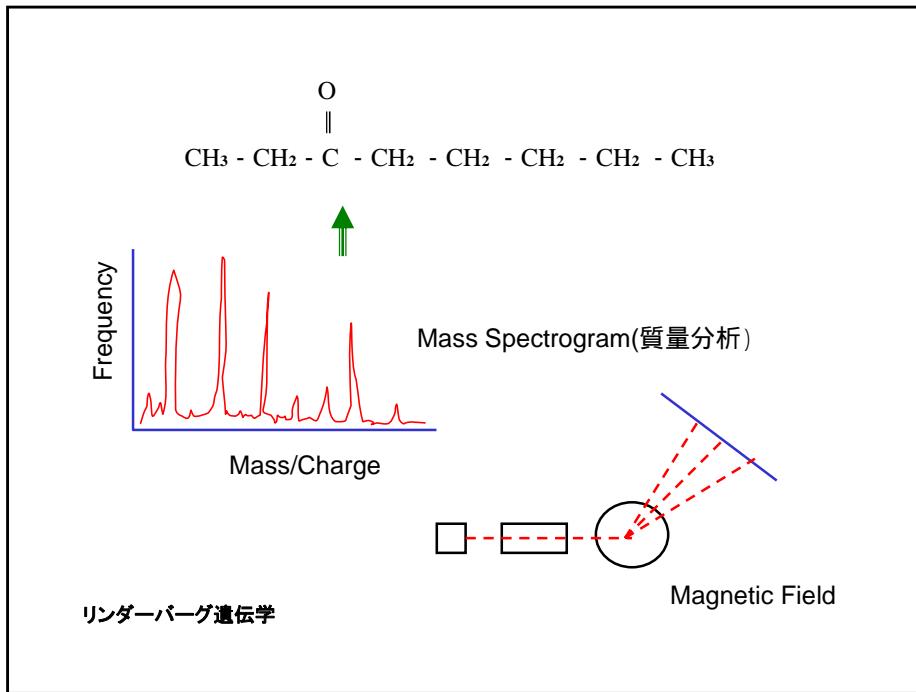
Knowledge Modeling and Reasoning II (情報学専攻 上野)
(知識表現と推論)

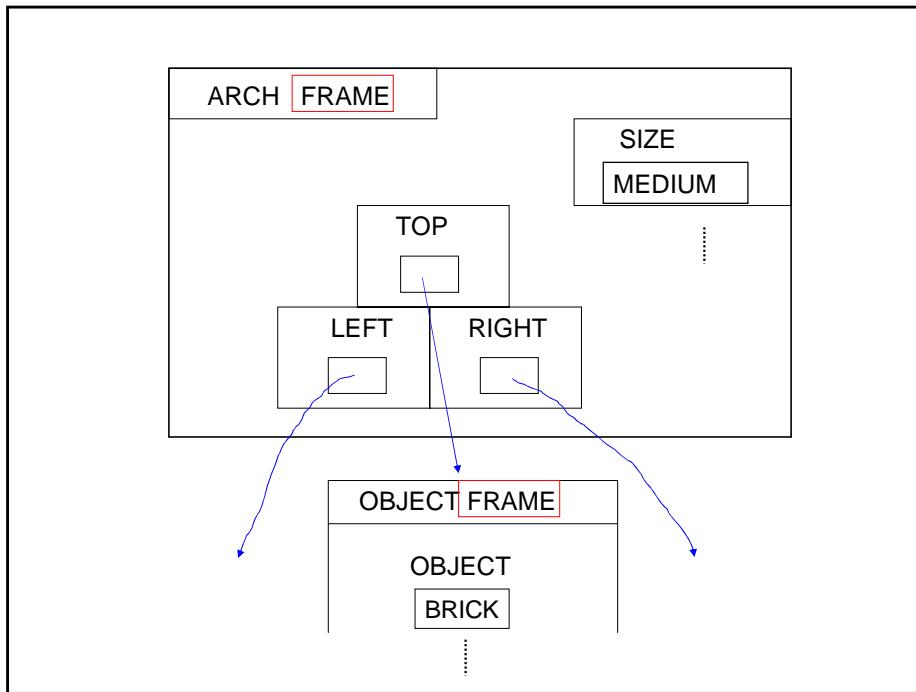
エキスパート・システムの構成



EMYCIN へ







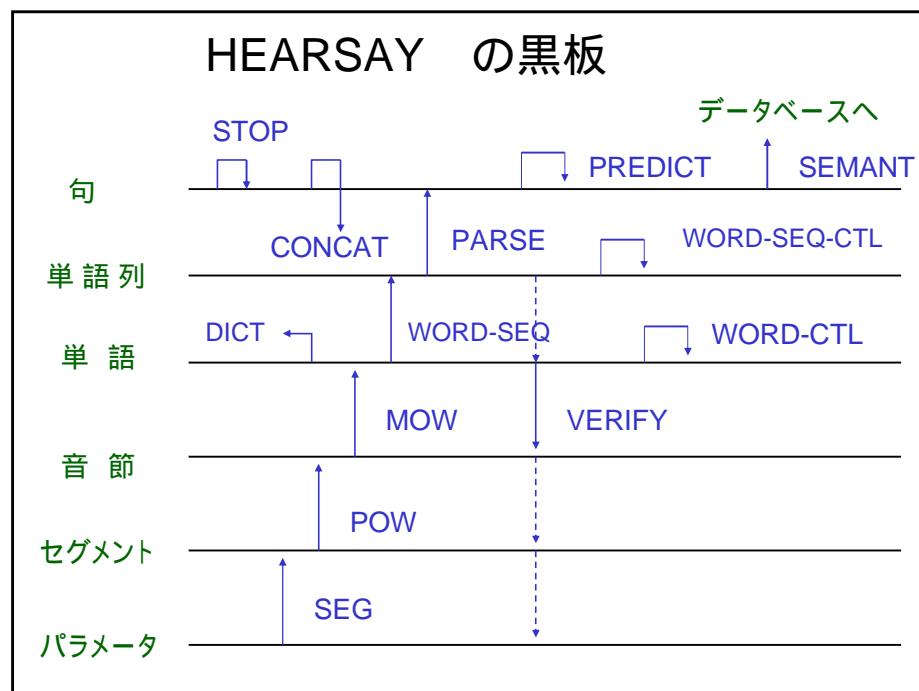
フレーム (Frame)の例

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

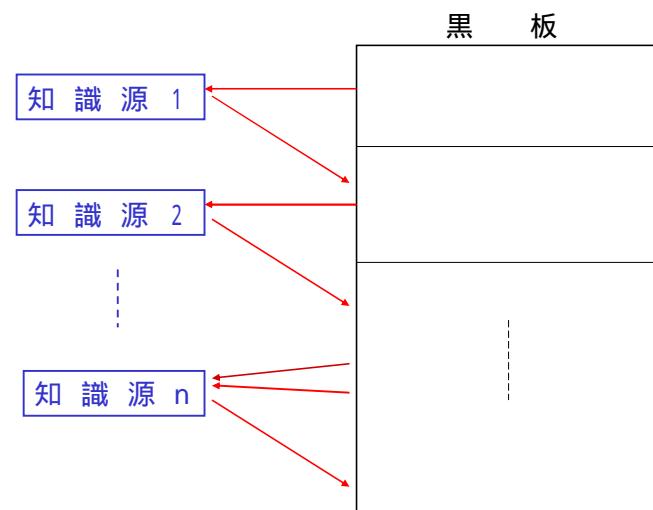
スズメフレーム
鳥の一種
食物 : 昆虫、穀物
大きさ : 5 ~ 10 cm

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

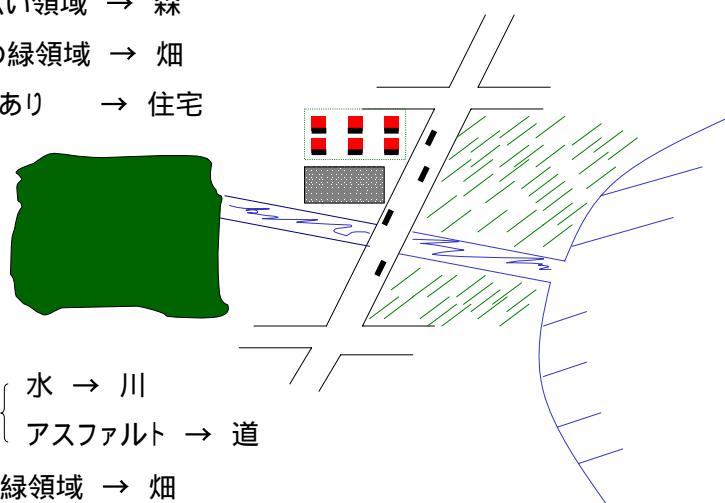
長方形の緑領域 → 畑

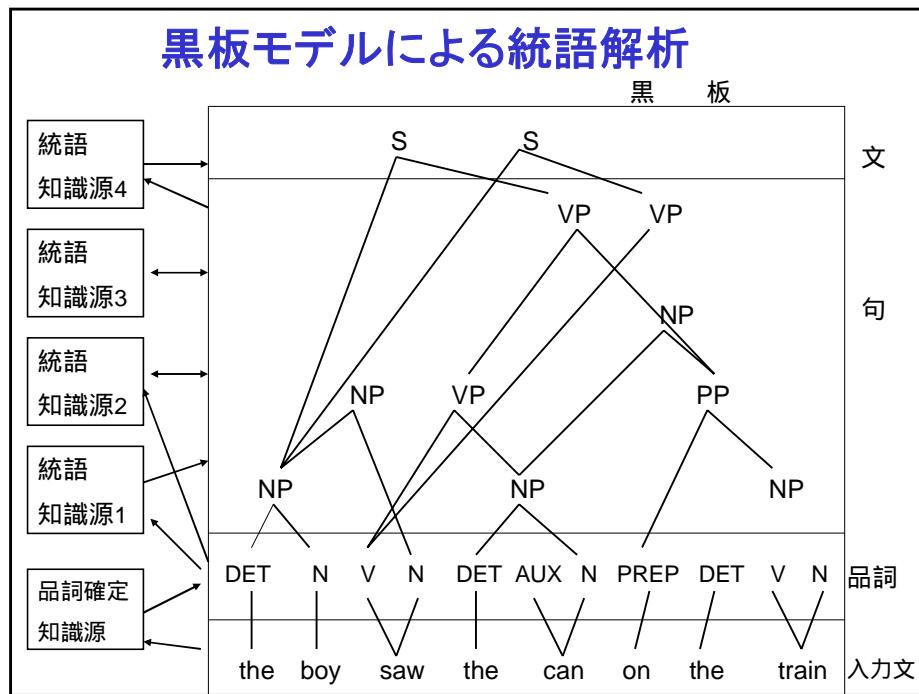
混在、影あり → 住宅

細長い { 水 → 川
アスファルト → 道

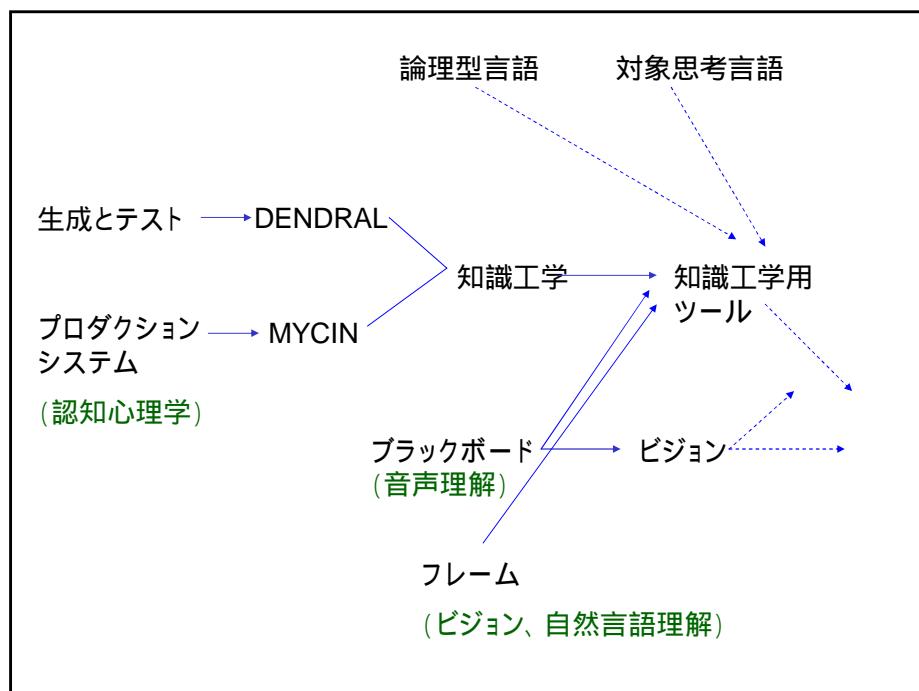
長方形の緑領域 → 畑

道と川の交点 → 橋





(出所) 『人口知能の理論』白井良明 (2001) コロナ社



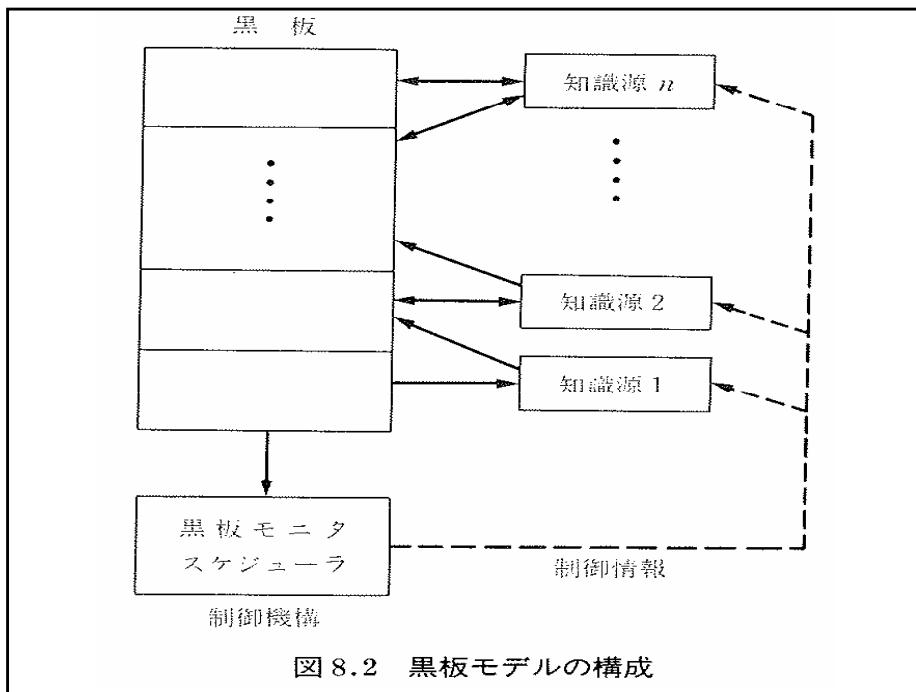
```
(suzuki-ken (date-of-birth (1967 8 9))
  (age (26))
  (sex (male))
  (institution (Naniwa University))
  (address (1-1, Naniwaku Osaka))
  (fields ((electronics) (artificial-intelligence))))
```

フレームの構造

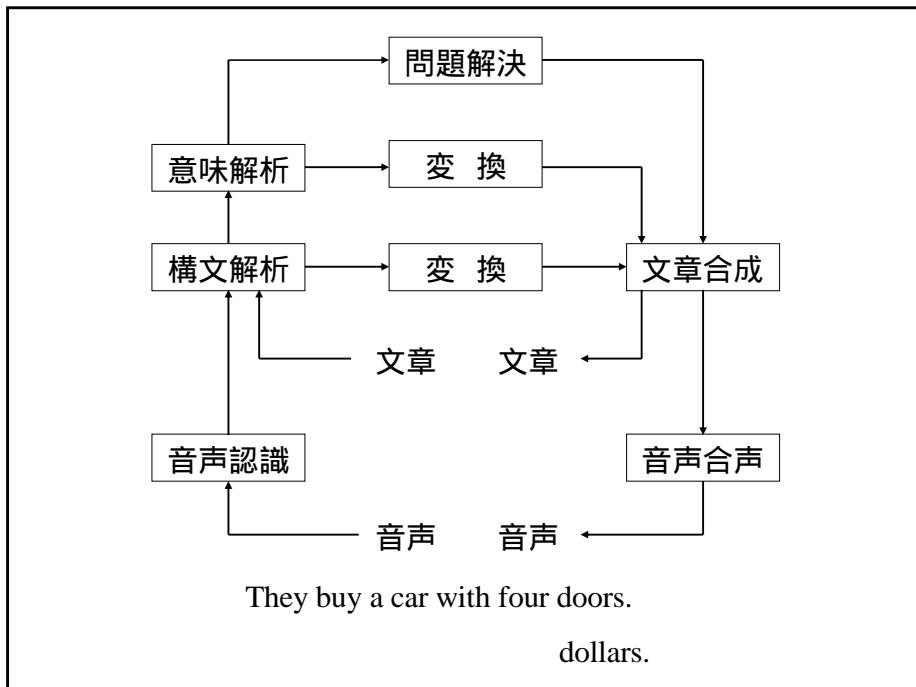
```
(suzuki-ken (date-of-birth (VALUE (1967 8 9))
  (IF-ADDED (年齢を求める手続き)))
  (age (VALUE (26)))
  (sex (DEFAULT (male)))
  (institution (VALUE (Naniwa University))))
  (address (IF-NEEDED(所属の住所を求める手続き)))
  (fields (VALUE (electronics) (artificial-intelligence)))))
```

```
(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)))
  .....
  .....
  .....)
```



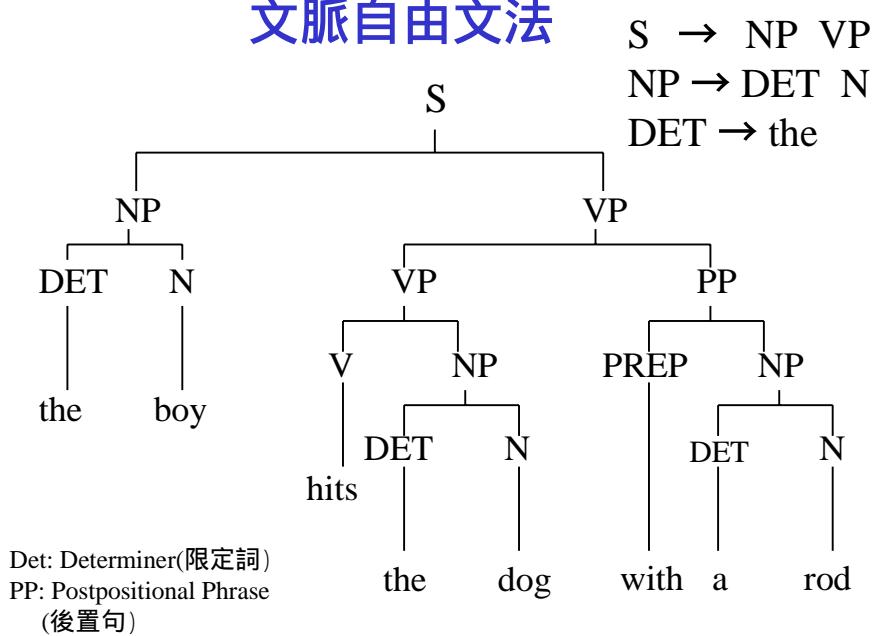
(出所) 『人口知能の理論』白井良明 (2001) コロナ社



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) , ...
- 正規文法(regular grammar)
 $A \rightarrow a, \quad A \rightarrow aB$
- 文脈自由文法(context-free grammar)
 $A \rightarrow$
- 文脈依存文法(context-sensitive grammar)
 \rightarrow
 - 1型: $|\alpha| \leq |\beta|$
 - 0型: 制限なし

文脈自由文法



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

DET (b)

the

N boy ...

予測

3 NP

DET

the

N

boy ...
↑ (h)

4 NP

DET

the

N

boy ...

左隅統語解析

5 S

NP (a) VP

hits ...

6 S

NP VP

V hits ...
↑ (k)

7 S

NP VP

the

VP

V

dog ...
↑ (c)
NP

8 S

NP VP

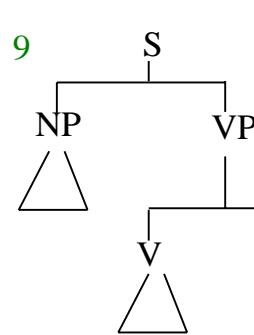
the

VP

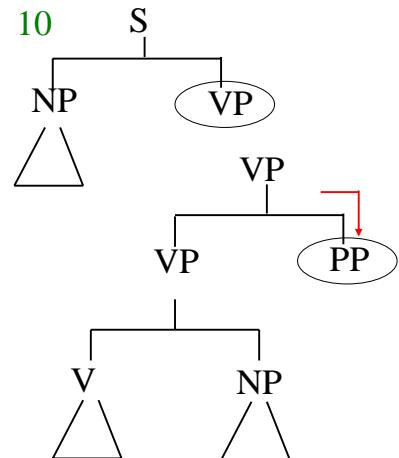
V

dog ...
NP

左隅統語解析



sentence が完成するが、文が残っている



8へ戻りし、解析を続ける

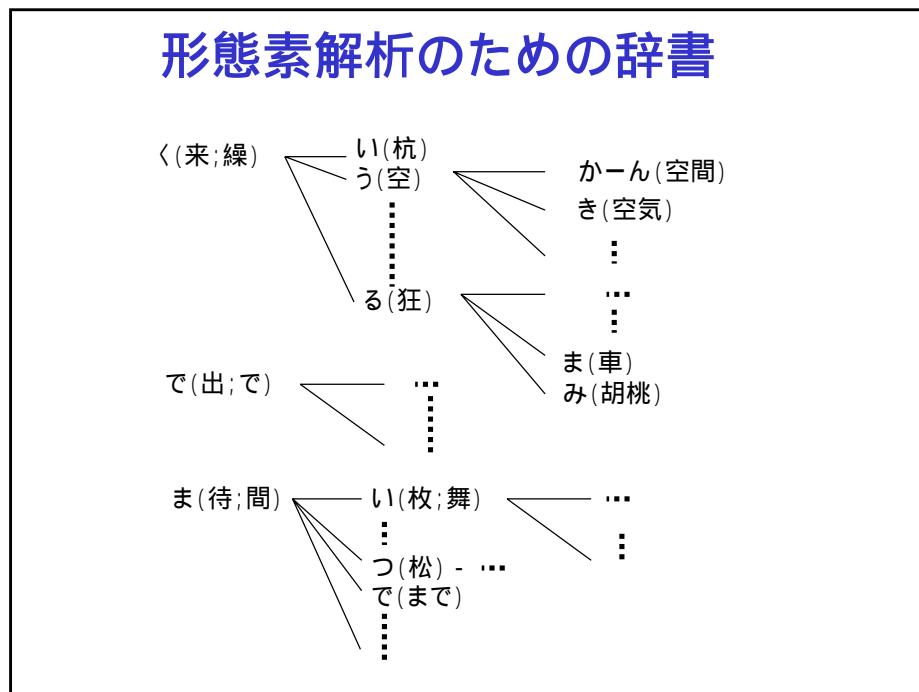
Bottom-Up Parsing

C Y K 表

$NP \rightarrow NP \ PP$

								8
								7
								6
								5
								4
								3
								2
								1
1	2	3	4	5	6	7	8	
the	boy	saw	the	dog	in	the	house	

CYK 表							
S_1							
S_2							
			VP_1	VP_2			
					NP		
			VP			PP	
NP				NP			NP
DET	N	V	DET	N	$PREP$	DET	N
the	boy	saw	the	dog	in	the	house



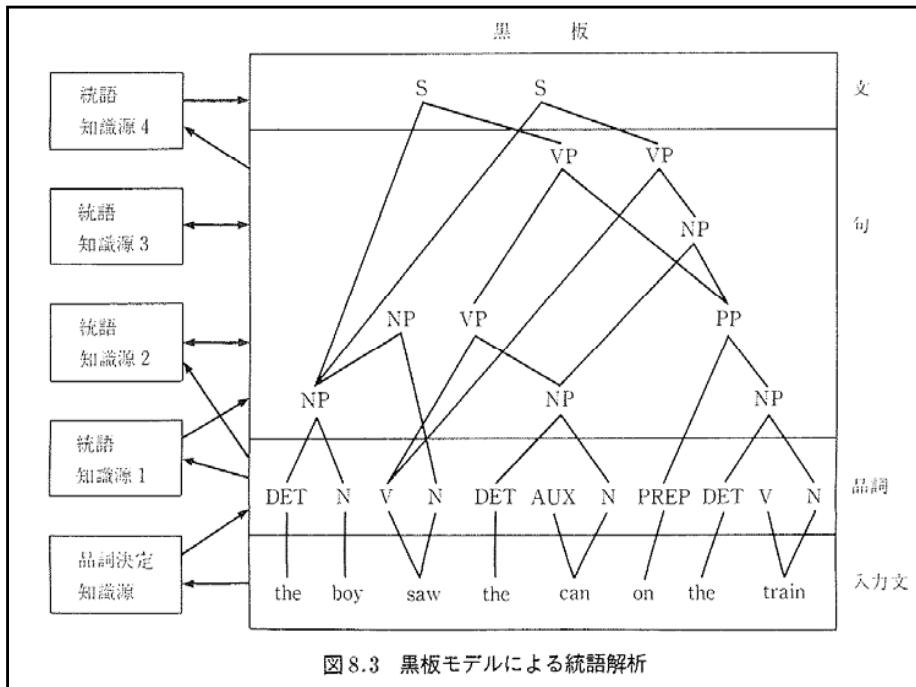
形態素辞書に基づく CYK 表の途中結果

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 亞	る

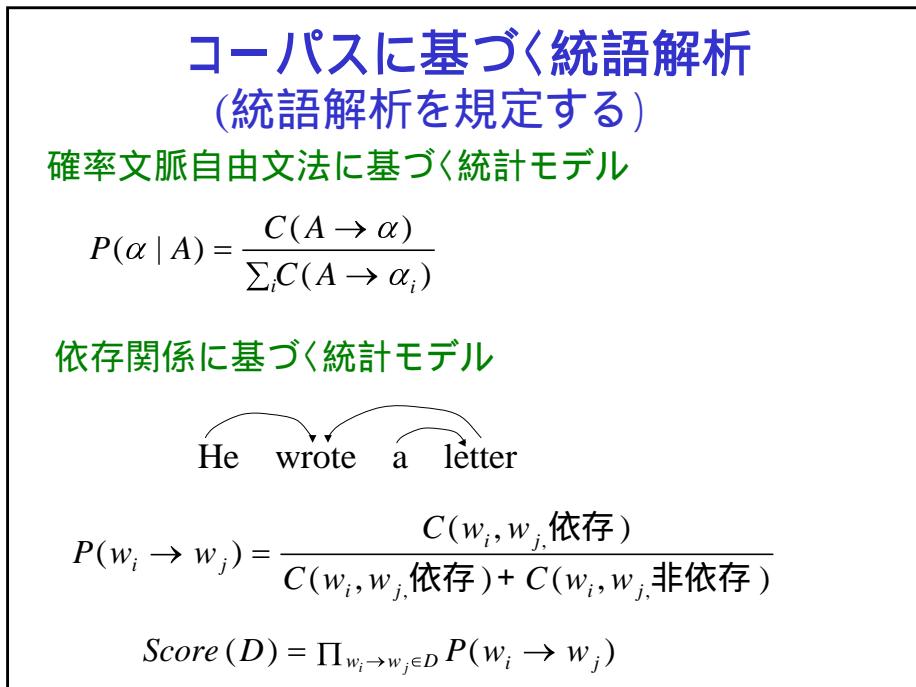
1 2 3 4 5 6 7 8
い ま は な し が あ る

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	$H = [\text{話歯無し}]$
6	今Hが 今は梨が	$G = [\text{名菜}]$
5	今H 今は梨	$F = [\text{花鼻}]$
4	今はG 今F	
3	今は 今歯	いまはなしがある



(出所) 『人口知能の理論』白井良明 (2001) コロナ社



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：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> := ([接頭語] 自立語 [接尾語] *)*

[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 もくべきは 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

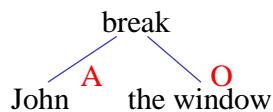
格文法

break

O

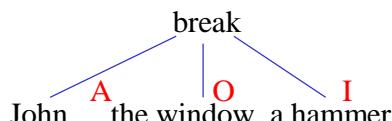
(a) "The window broke"

O: 対象格



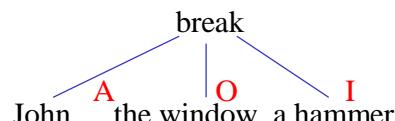
(b) "John broke the window"

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

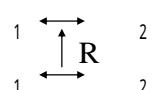
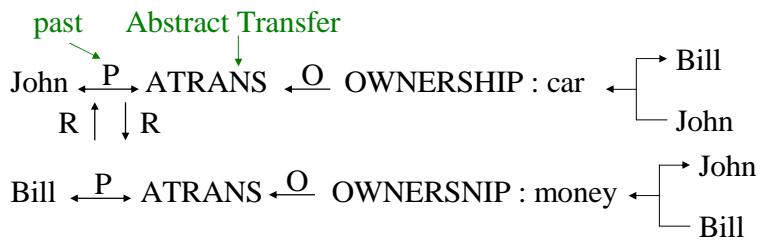
時間格 (T): 出来事が起こる時間

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

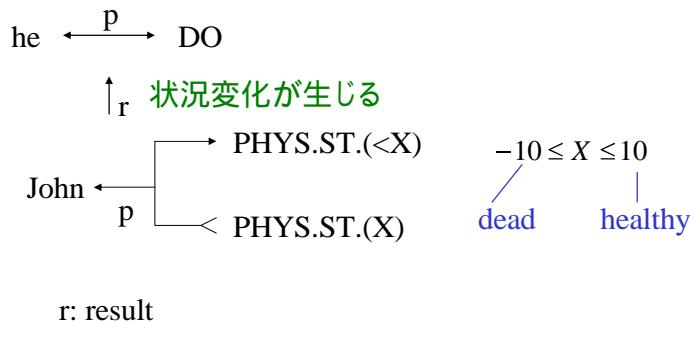
“John sold his car to Bill.”
に対応する概念依存構造



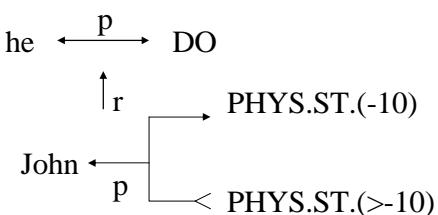
1 ↔ 2 1 ↔ 2 が理由・原因 (Reason) となって
1 ↔ 2 なる行為が引き起こされること

P: past

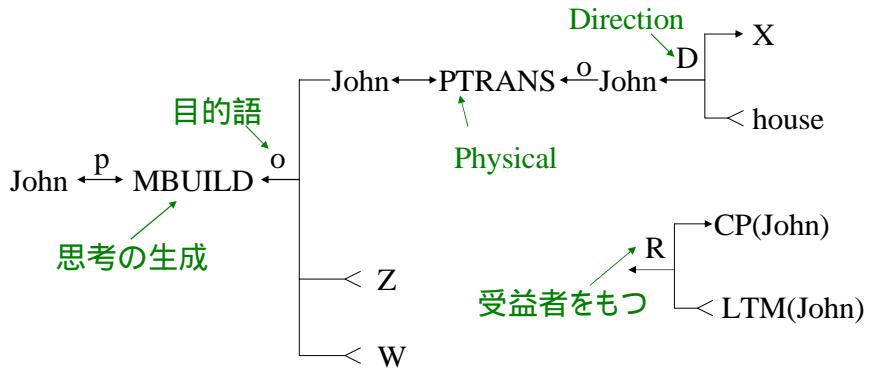
“He hurt John.”
に対する概念依存構造



“He killed John.”
に対する概念依存構造

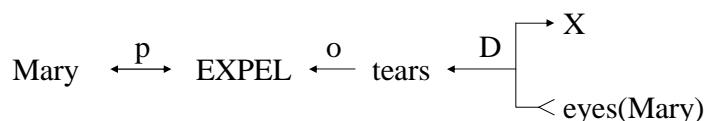


“John decided to leave the house.” の概念依存構造



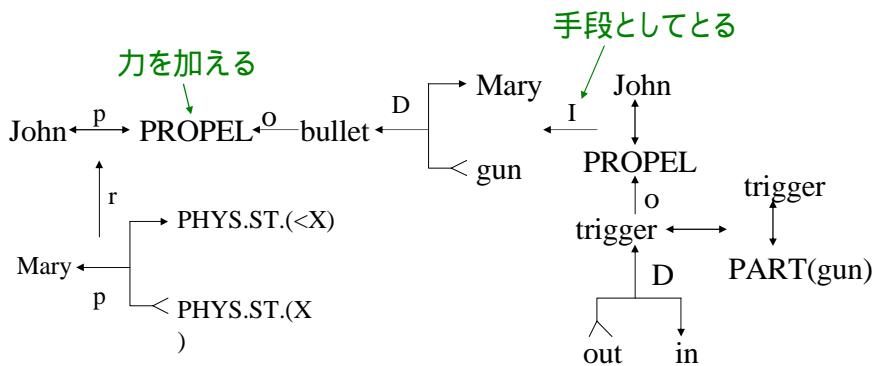
「ジョンは、ジョンがジョンを家からXに移動させる(PTRANS)という考えを、LTM(長期記憶)からCP(概念プロセッサ)に移動した。」

“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, \dots

論理記号: $, , \sim, \rightarrow$

命題: $P \wedge Q, P \vee Q, \sim P, P \rightarrow Q, P \leftrightarrow Q$

$P \wedge Q$: 論理積、連言、and

$P \vee Q$ を論理和、選言、or

$\sim P$: 否定、not

$P \rightarrow Q$: 含意、implication

$P \leftrightarrow Q$: 同値、equivalence

$P \rightarrow Q, \sim P, Q$

P	Q	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
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: $, , \sim, \rightarrow,$

量記号: \forall (全称記号)、 \exists (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、 $P \wedge Q, P \vee Q, \sim P, P \rightarrow Q, P \leftrightarrow Q$
 $(\forall x)[(\forall y) LESS(x, y)], \forall x \forall y LESS(x, y),$
 $\forall x \forall y LESS(x, y)$

述語論理

命題の例: $x \ y \ z [LESS(x, y) \wedge LESS(y, z) \rightarrow LESS(x, z)]$

妥当命題: 述語記号にどのような解釈を与えても真

Valid wff

$$x P(x) \rightarrow P(a)$$

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

$$x P(x) \rightarrow \neg P(a)$$

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

$$x \ y \ z [P \vee Q \vee R] \quad [] \text{内を母式(matrix)}$$

節形式への変換

1. $P \rightarrow Q$ を $\neg P \vee Q$, $P \wedge Q$ を $(\neg P \vee Q) \wedge (\neg Q \vee P)$
2. $x[\neg P(x) \vee Q(x)]$ を $x[\neg P(x) \wedge yQ(y)]$
3. $\neg x[P(x) \wedge \neg(Q(x) \wedge R(x))]$ を $x[\neg P(x) \vee (Q(x) \wedge R(x))]$
4. $x[\neg y P(x, y)]$ を $x[P(x, f(x))]$ Skolem-function
 $x P(x)$ を $P(a)$
5. $x[P(x) \wedge y \neg Q(y)]$ を $x \ y [P(x) \wedge \neg Q(y)]$
6. $(A \wedge B) \vee (B \wedge C)$ を $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$

$$\begin{matrix} (A \wedge C) \\ B \end{matrix}$$

推論の基本

1. $P \wedge P \rightarrow Q$ から Q を導く ($P, P \rightarrow Q \vdash Q$ と書く)。
2. $P \rightarrow Q$ と $Q \rightarrow R$ から $P \rightarrow R$ を導く。

$C_1 = P \quad Q_1 \quad Q_2 \quad \dots \quad Q_m \quad \wedge C_2 = \neg P \quad R_1 \quad R_2 \quad \dots \quad R_n$ から
 $C_r = Q_1 \quad Q_2 \quad \dots \quad Q_m \quad R_1 \quad R_2 \quad \dots \quad R_n$ 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 $s := \text{NIL}$ $Q_1 := P_1$, $Q_2 := P_2$
- 2 LOOP: $D_i = Q_1$ と Q_2 の不一致集合
- 3 **if** empty(D) **then return** (s)
- 4 $t_1 := \text{first}(D)$, $t_2 := \text{last}(D)$
- 5 **if** variable(t_1) and not-contain(t_1, t_2)
 then $s1 := (t_1 / t_2)$
 else if variable(t_2) and not-contain(t_2, t_1)
 then $s1 := (t_2 / t_1)$
 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 : $\forall x \exists v P(x, v) \wedge \forall y \forall z [P(a, y) \rightarrow \neg Q(y, z)]$

目標 : $\neg \forall u \exists w Q(u, w)$

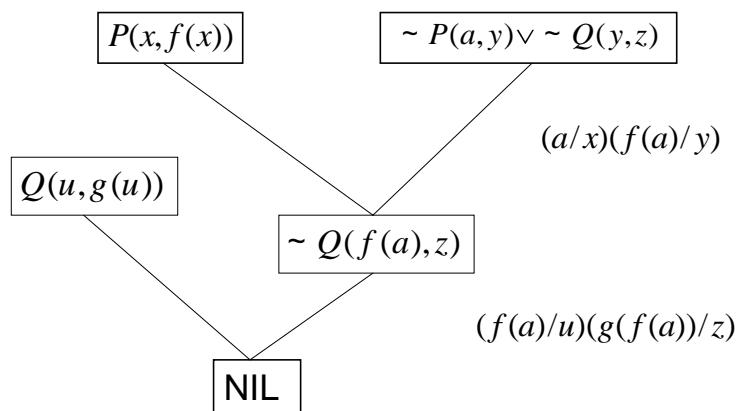
目標の否定 : $\forall u \exists w Q(u, w)$

節形式 : $Q(u, g(u))$

前提の節形式 : $P(x, f(x))$
 $\neg P(a, y) \vee \neg Q(y, z)$

導出過程

目標の否定 : $Q(u, g(u))$ 前提 : $P(x, f(x))$
 $\neg P(a, y) \vee \neg Q(y, z)$



証明の制御

目標の否定 $\sim P(x, y) \vee Q(x, y)$

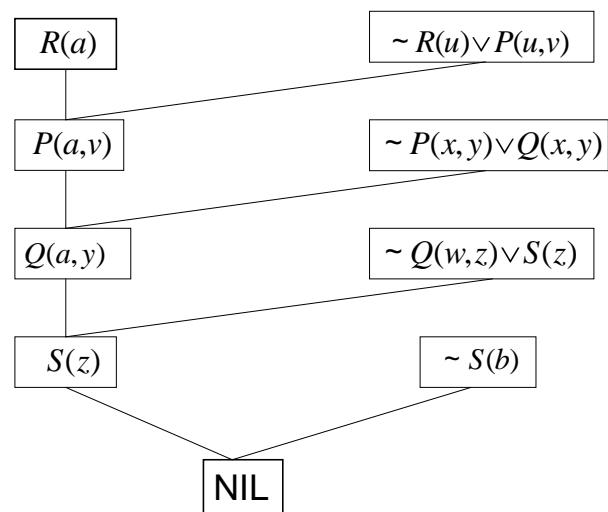
前提 $\sim R(u) \vee P(u, v)$

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

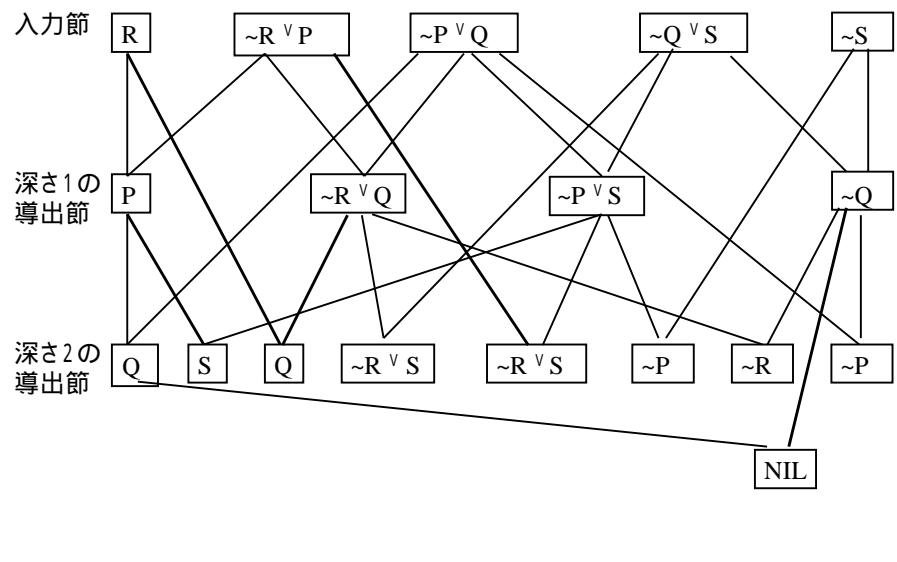
$R(a)$

$\sim S(b)$

線形戦略による導出過程

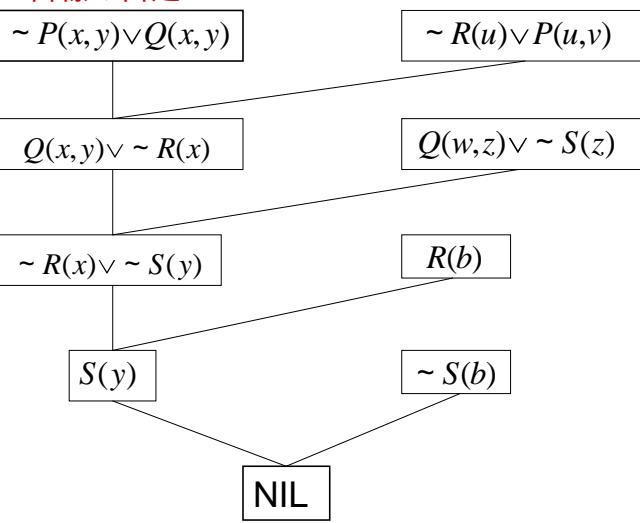


幅優先戦略による導出過程



支持集合戦略による導出過程

目標の否定



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 $\sim R(u) \vee P(u,v)$

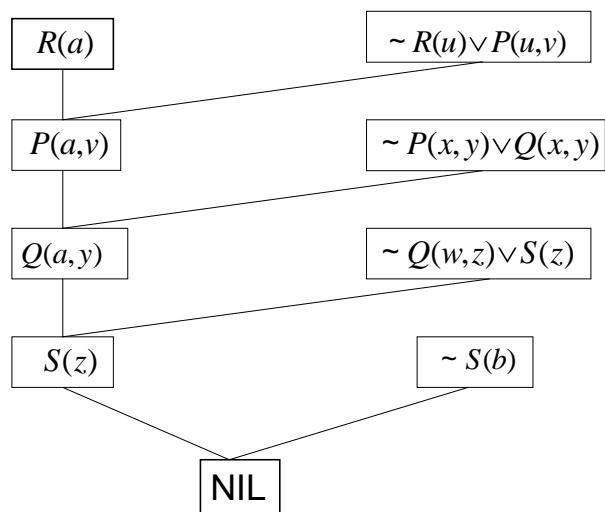
$\sim Q(w,z) \vee S(z)$

$R(a)$

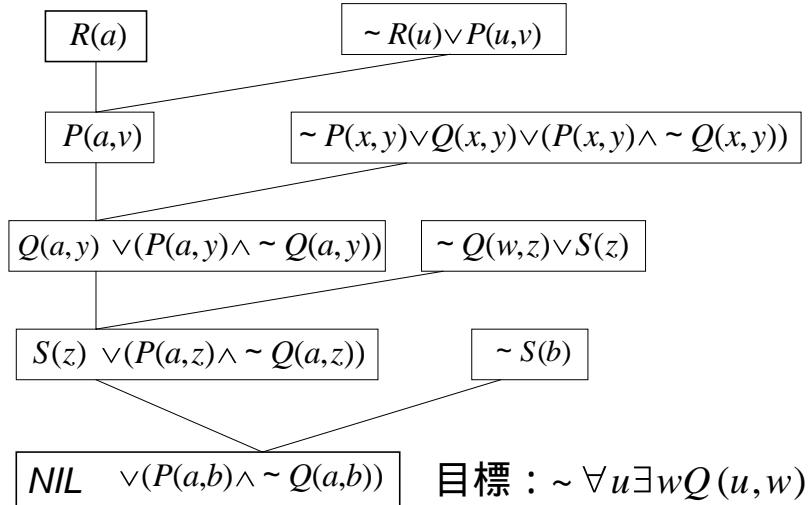
$\sim S(b)$

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: 誰にも祖父母がいる

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

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

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

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

a/u, w/y

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

$$P(z, f(z))$$

a/z, f(a)/v

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

$$P(z, f(z))$$

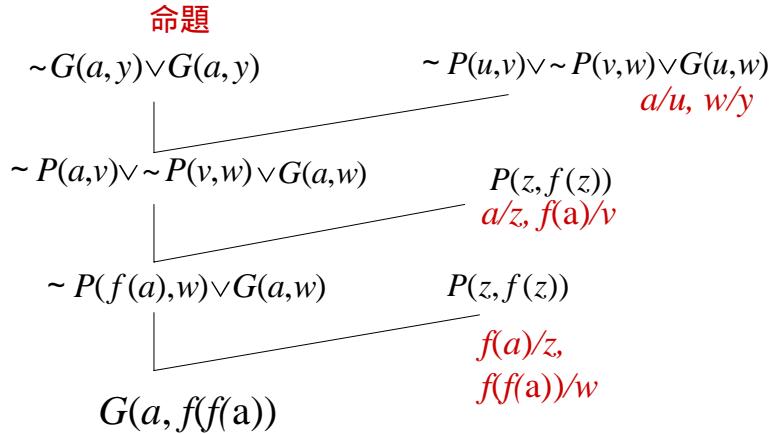
f(a)/z, f(f(a))/w

$$NIL$$

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: $\sim G(a, y)$



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 \longleftrightarrow 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 が成立し、 $\sim Q$ が証明されていないならば、
 Q が成り立つ。

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

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

正規デフォルト体系

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

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

デフォルト規則: Most birds fly と、

$W = \neg \text{PENGUIN}(x) \vee \neg \text{FLY}(x),$

$\text{BIRD}(a), \text{BIRD}(b), \text{PENGUIN}(b)$

から構成される論理体系の拡張 E は

$E = W, \text{FLY}(a)$

デフォルト推論のアルゴリズム

1. W と $\text{CON}(D)$ から g を証明する。

$S := W$

2. $D_S :=$ 証明に用いた D の規則の集合

If $D_S = \{\}$, go to 4.

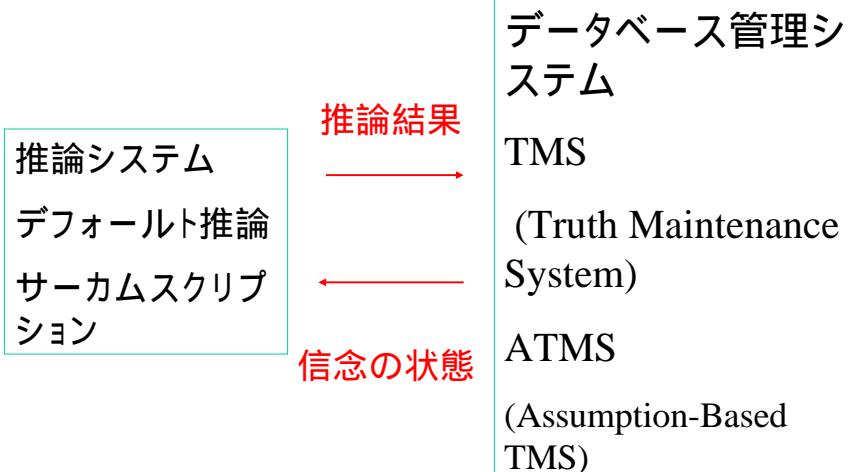
Else $S := S \cup \text{CON}(D_S)$

3. W と $\text{CON}(D)$ から $\text{PRE}(D_S)$ 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



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: $n: <\text{データ}, \text{ラベル}, \text{正当化}>$

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 $\uparrow \{E_1, E_2, \dots\}$

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

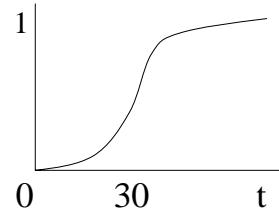
3. If n が矛盾節点、

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、あるいは
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

$$\mu(A \cap B) = \min\{\mu(A), \mu(B)\}$$

$$\mu(A \cup B) = \max\{\mu(A), \mu(B)\}$$

$$\mu(\sim A) = 1 - \mu(A)$$

Kleene の $A \rightarrow B$

$$\mu(A \rightarrow B)$$

$$= \mu(\sim A \cap B)$$

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

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

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

Fuzzy 推論

Lukaszewicz の $A \rightarrow B$

$$\mu(A \rightarrow B) = \min\{1, 1 - \mu(A) + \mu(B)\}$$

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

Goedel の $A \rightarrow B$

$$\mu(A \rightarrow B) = 1 \quad \mu(A) = < \mu(B)$$

$$\mu(B) \text{ otherwise}$$

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

$\mu(\sim A \rightarrow A) = \mu(A)$ と
したいが $\mu(A) = 1/2$ で 1

Fuzzy 推論

Mamdani の $A \rightarrow B$

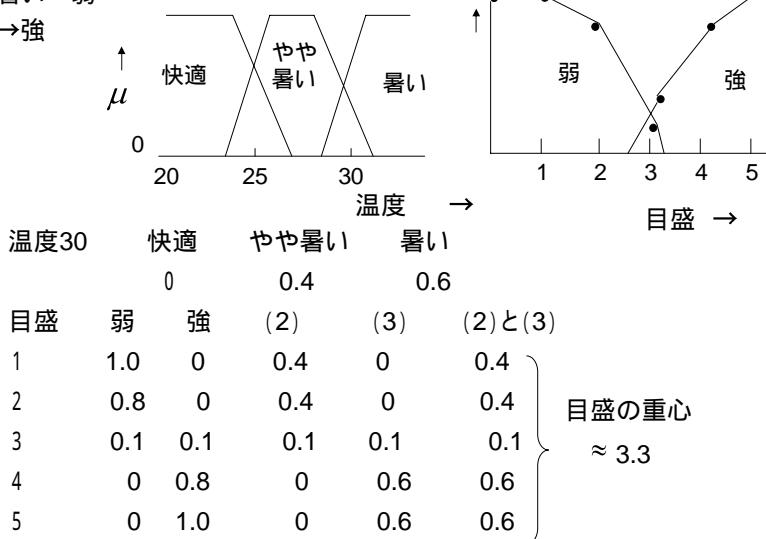
$$\mu(A \rightarrow B) = \min\{ \mu(A), \mu(B) \}$$

$$(\equiv \mu(A \cap B))$$

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

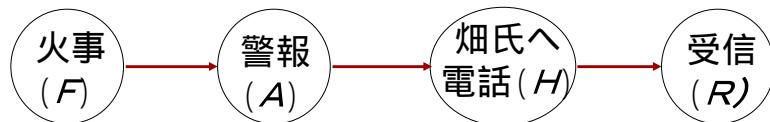
- (1) 快適 → 空調を切る
- (2) やや暑い → 弱
- (3) 暑い → 強



単純ベイズネットの例

$$P(F) = 0.001$$

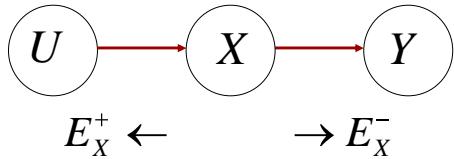
$$\begin{aligned} P(H | A) &= 0.9 \\ P(H | \sim A) &= 0.2 \end{aligned}$$



$$\begin{aligned} P(A | F) &= 0.94 \\ P(A | \sim F) &= 0.002 \end{aligned}$$

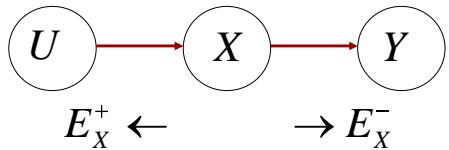
$$\begin{aligned} P(R | H) &= 0.98 \\ P(R | \sim H) &= 0.1 \end{aligned}$$

単純ベイズネット



$$\begin{aligned} P(X|E) &= P(X|E_X^-, E_X^+) \\ &= \frac{P(E_X^-|X, E_X^+)P(X|E_X^+)}{P(E_X^-|E_X^-)} = \alpha P(E_X^-|X)P(X|E_X^+) \\ &= \alpha \lambda(X)\pi(X) \end{aligned}$$

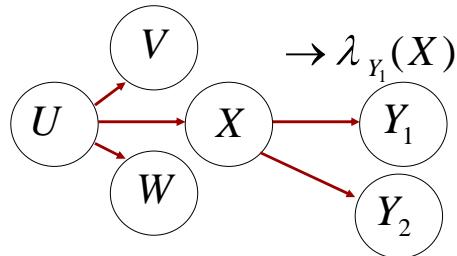
単純ベイズネットの π と λ の更新



$$\begin{aligned} \pi(x) &= \sum_u P(X|u, E_X^+)P(u|E_X^+) \\ &= \sum_u P(X|u)P(u|E_X^+) = \sum_u P(X|u)\pi(u) \end{aligned}$$

$$\begin{aligned} \lambda(X) &= P(E_X^-, Y|X) = \sum_y P(E_X^-|X, y)P(y|X) \\ &= \sum_y P(E_X^-|y)P(y|X) = \sum_y \lambda(y)P(y|X) \end{aligned}$$

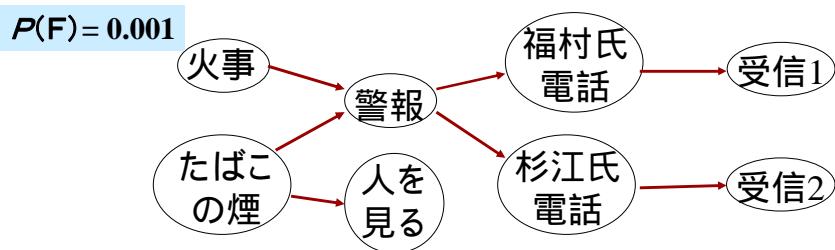
木の場合のとの更新



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

$$\begin{aligned}\lambda(X) &= \prod_i P(E_{Y_i \neg X} | X) = \prod_i \lambda_{Y_i}(X) \\ &= \prod_i \sum_{y_i} \lambda(y_i) P(y_i | X)\end{aligned}$$

ベイズネット



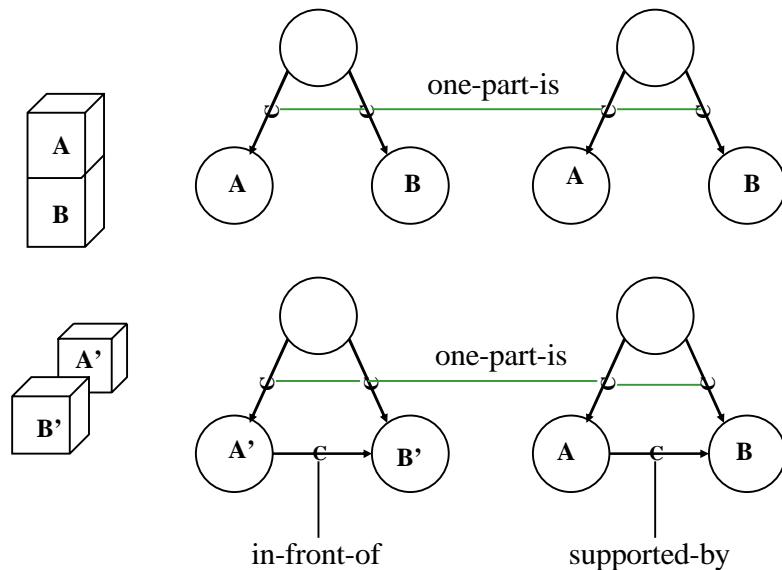
警報 (A)に 対して	$P(A F, S) = 0.95$ $P(A F, \neg S) = 0.94$ $P(A \neg F, S) = 0.2$ $P(A \neg F, \neg S) = 0.01$
-------------------	---

Learning Structural 4 from Examples

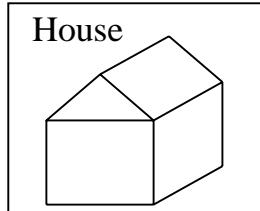
Patrick H. Winston,
The Psychology of Computer Vision,
McGraw-Hill Book Co. 1975

積木の線画から、積木の構造を学習する

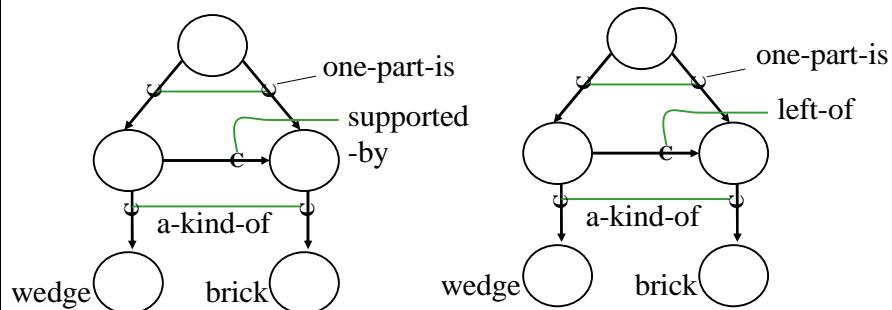
Tower



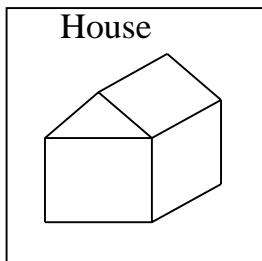
House



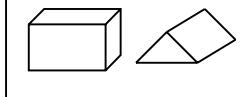
Near miss

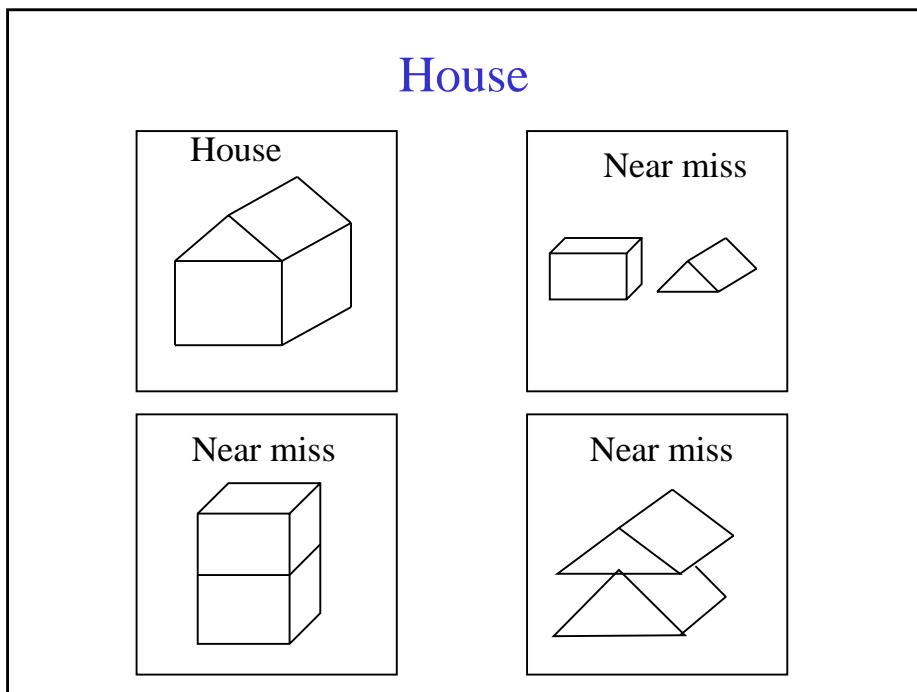
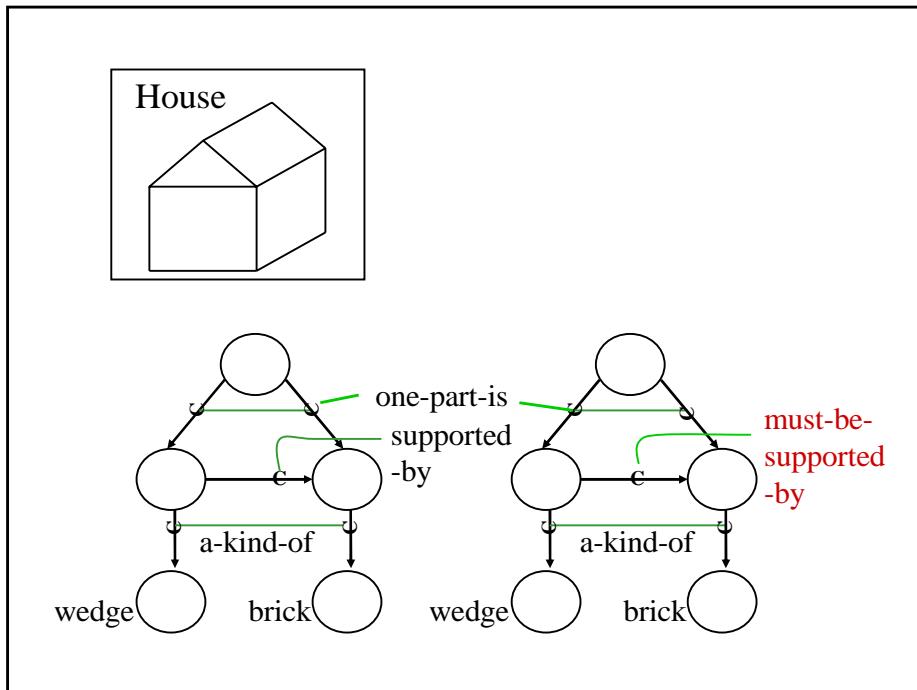


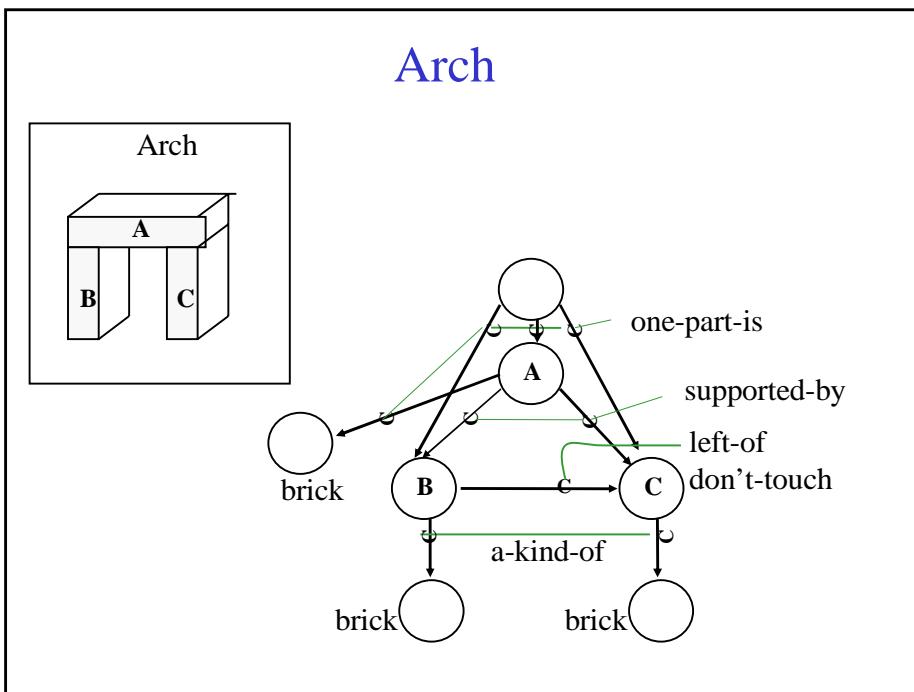
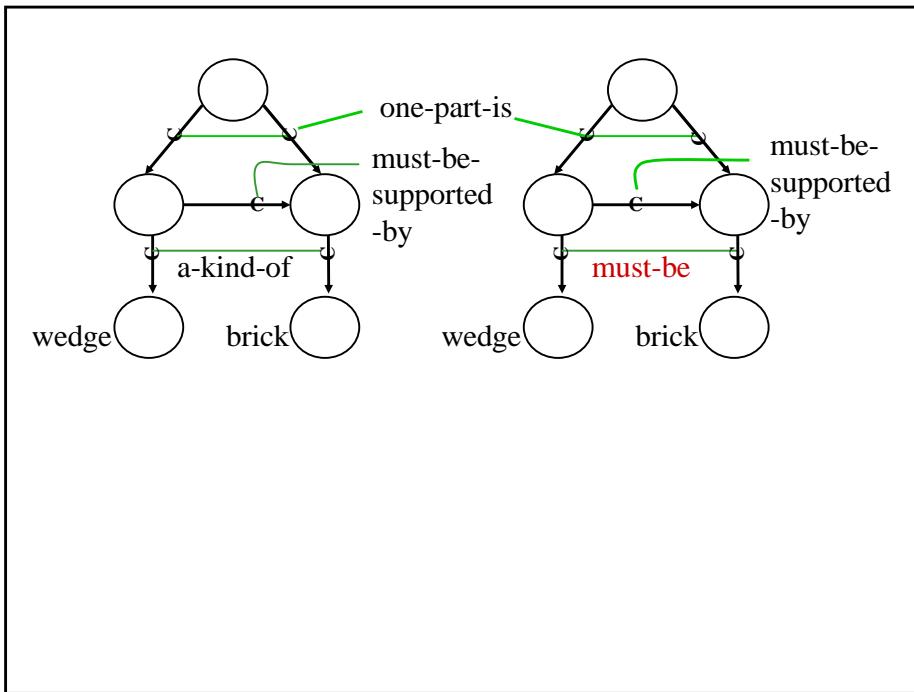
House

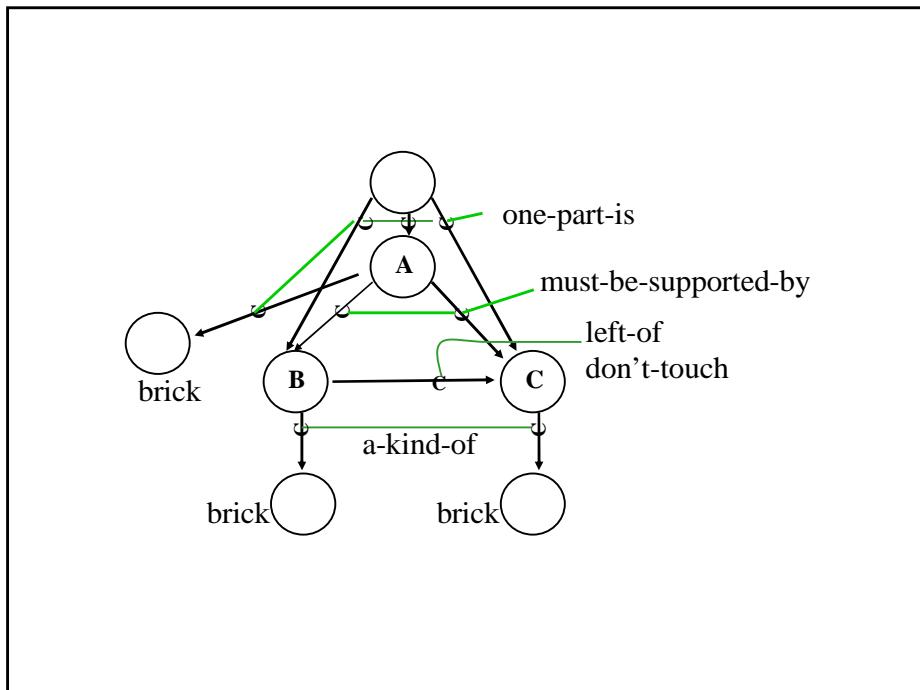
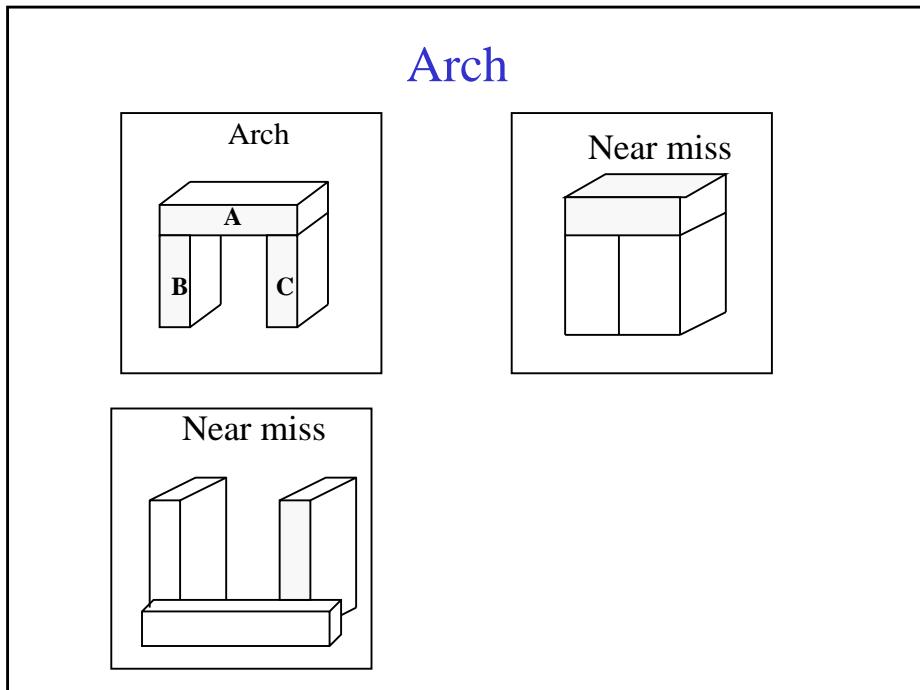


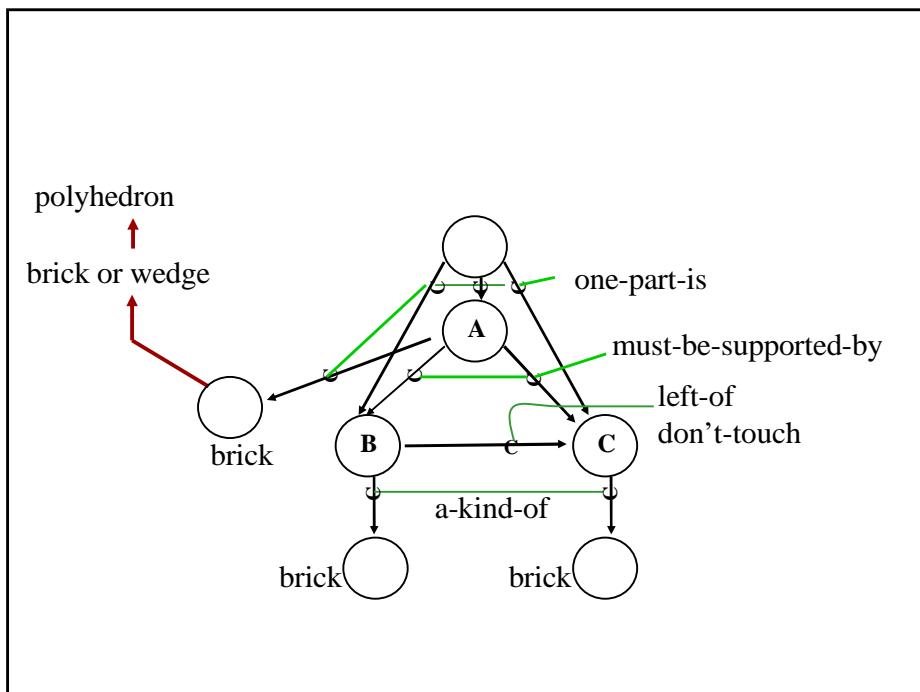
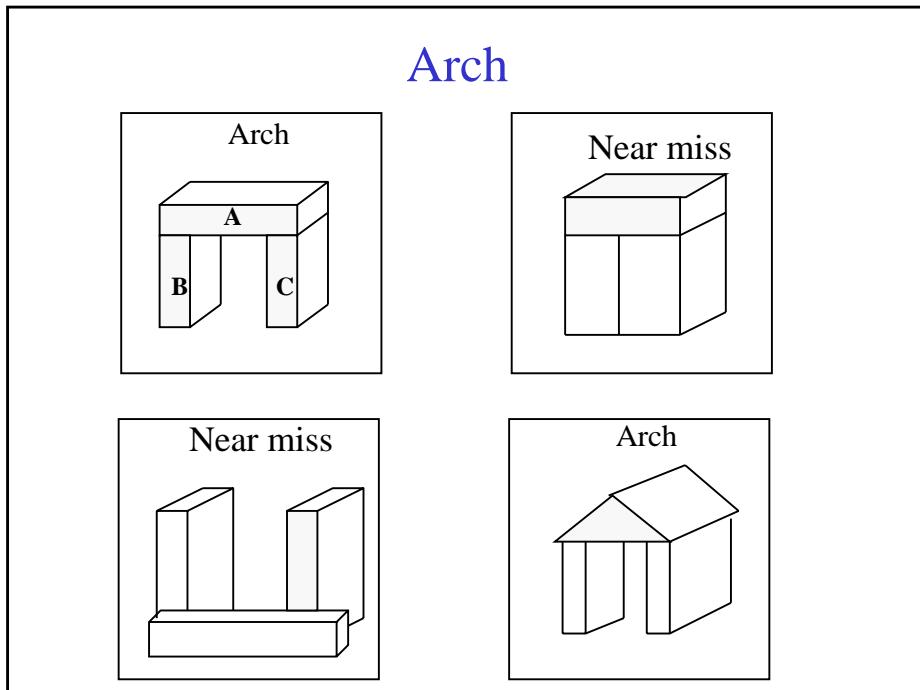
Near miss











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

ユニットへの出力

学習望ましい出力 y_{d_j}

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

$$r = \sum_j (O_j^m - y_j)^2 \text{ を最小化}$$

出力

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

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

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

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

$$\Delta W_{i,j}^{k-1,k} = -\varepsilon \frac{\partial r}{\partial w_{i,j}^{k-1,k}} = -\varepsilon \frac{\partial r}{\partial i_j^k} \frac{\partial r_j^k}{\partial w_{i,j}^{k-1,k}}$$

$\swarrow \quad k \neq m \quad \text{では} \quad i_j^k = \sum \cdots \text{より} \quad \uparrow \quad O_i^{k-1} \text{となる}$

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

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

$$\Delta W_{i,j}^{k-1,k} = -\varepsilon d_j^k O_i^{k-1}$$

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

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

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

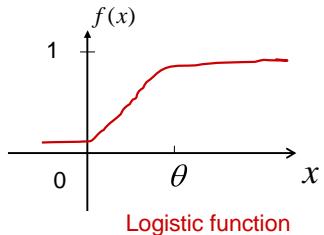
$$\text{まず } \Delta W_i^{m-1 \rightarrow m} = -\varepsilon \quad d_j^m \quad O_i^{m-1}$$

計算可 or 観測可

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

$$f(x) = \frac{1}{1 + e^{-p(-x + \theta)}}$$

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



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。

$I = \{ i_1, i_2, \dots, i_N \}$ 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 $X \subseteq Y = I$ 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 $X \rightarrow Y$ 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。