論理と計算

第１２回
高次推論：帰納推論の基礎

担当：尾崎　知伸
ozaki.tomonobu@nihon-u.ac.jp

講義予定　※一部変更（前倒し）になる可能性があります

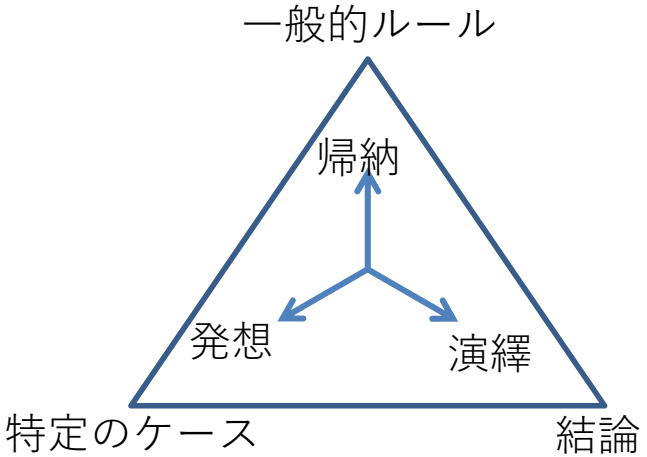
09/22	01. オリエンテーション と 論理を用いた問題解決の概要
09/29	02. 命題論理：構文・意味・解釈
10/06	03. 命題論理：推論
10/13	04. 命題論理：充足可能性問題
10/20	05. 命題論理：振り返りと演習（課題学習）
10/27	06. 述語論理：構文・意味・解釈
11/03	07. 述語論理：推論　※文化の日，文理学部授業日
11/10	08. 述語論理：論理プログラムの基礎
11/17	09. 述語論理：論理プログラムの発展
11/24	10. 述語論理：振り返りと演習（課題学習）
12/01	11. 高次推論：発想推論
12/08	12. 高次推論：帰納推論の基礎
12/15	13. 高次推論：帰納推論の発展
12/22	14. 高次推論：振り返りと演習（課題学習）
01/19	15. まとめと発展的話題

目次：今回の授業の内容

- 帰納推論
  - 概要
  - 問題設定
- 伴意からの学習
- 帰納推論（伴意からの学習）の応用例

帰納推論の概要

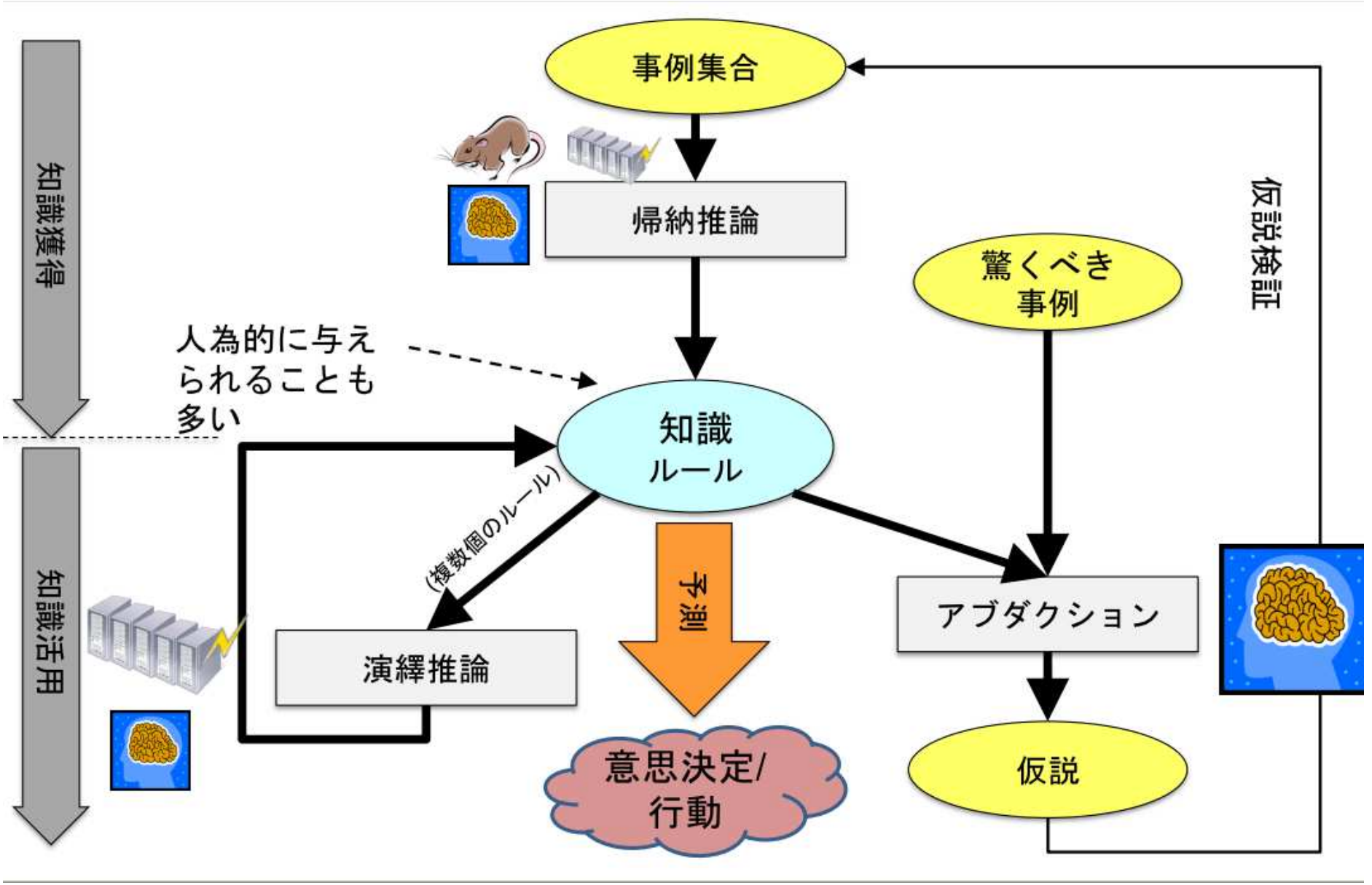
推論の種類



- 推論：知識をもとに，新しい結論を得ること
  - 既に分かっている情報・知識から，
    新しい情報・知識を導き出すこと
- 推論にはいくつかの種類がある
  - 演繹・発想・帰納・類推など
- 演繹推論（Deduction）
  - 一般的ルールを特定のケースに当てはめて結論を得る分析的過程
  - ルール「aならばb」とケース「a」から，結論「b」を導出
- 帰納推論（Induction）
  - 特定のケースと結論から，ルールを推論する合成的過程
  - ケース「a」と結論「b」の対から，ルール「a→b」を導出
  - いわゆる学習に相当
- 発想推論（Abduction）
  - 一般的ルールと結論化から，特定のケースを推論するもう一つの合成的過程
  - ルール「aならばb」と結論「b」から，ケース「a」を導出
  - ex. 病名診断．犯人捜し

常に正しい結果が得られる推論 vs 結果の正しさが保証されない推論

これまでの資料より



http://www.eccse.kobe-u.ac.jp/assets/images/distance_learning/life_science3/d84dedb94c7e352fbdc0c458b814a2ea59ecd46f.pdf より

発想推論の概要

- パースによる定義
  - 演繹推論：一般的ルールを適当のケースに当てはめて，結論を得る分析的過程
  - 発想推論：ルールと結論から，特定のケースを推論する合成的過程
  - 帰納推論：特定のケースと結論から，ルールを推論する合成的過程

- 発想推論（abduction）
  - 「大前提 $\alpha \Rightarrow \beta$ および結論 β から小前提 α を導く」 $\dfrac{\beta, \alpha \Rightarrow \beta}{\alpha}$
  - 仮説推論・辻褄合わせの推論（状況を上手く説明できる根拠を導出する）
    - そう考えると，矛盾なく説明できる
    - 故障診断・医療診断・犯人捜し

  - na：アリバイがない， c：犯人である，c⇒na：犯人にはアリバイがない
    - na と c⇒na から cを導く：アリバイが無くても犯人ではない
    - { c, c⇒na } |= na は成り立つ，{ na, c⇒na } |= c は成り立たない
      （必ずしも正しい推論ではない）

※必ずしも正しい推論ではない＝得られる結果が正しい（真実）とは限らない

帰納推論と概念学習

- 観測された事実から，その一般的な概念を得る

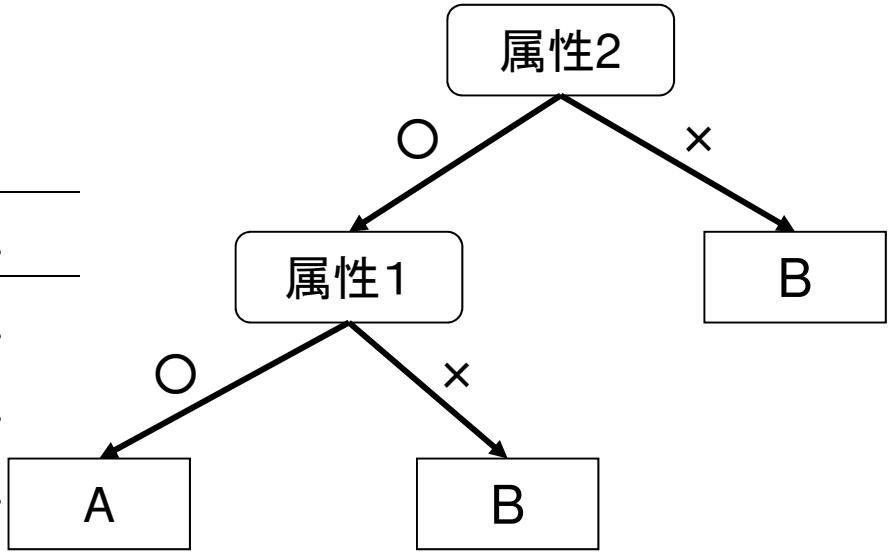
- 人間の絵
  - 細部（衣服，持ち物，格好，動作など）の相違を無視して，事例を一般化する
- スポーツをしている人間の絵
  - 細部の相違の無視と，他のクラスの事例との違いの利用

- 帰納推論・概念学習
  - 「個々の事例の細部を無視し，事例に見られる共通部分を取り出すことで，一般的な概念を得る」推論方式
  - 過度の一般化を避けるために，異なるクラスに属する事例を与える
    - ex:スポーツの絵を一般化するのに，スポーツではない絵も利用する
    - ある意味での「あるなしクイズ」

決定木：命題論理における帰納学習

- 属性＝値表（単一の表）からのルールの構築
- 木構造を使ったルール表現
  - IF “属性2 == ○” AND “属性1==○” THEN “カテゴリ=A”
  - IF “属性2 == ○” AND “属性1==×” THEN “カテゴリ=B”
  - IF “属性2 == ×” THEN “カテゴリ=B”
- ノード：属性のテスト（分割テスト）
- 枝ラベル：テストの結果
- 葉：カテゴリ（or クラス分布）

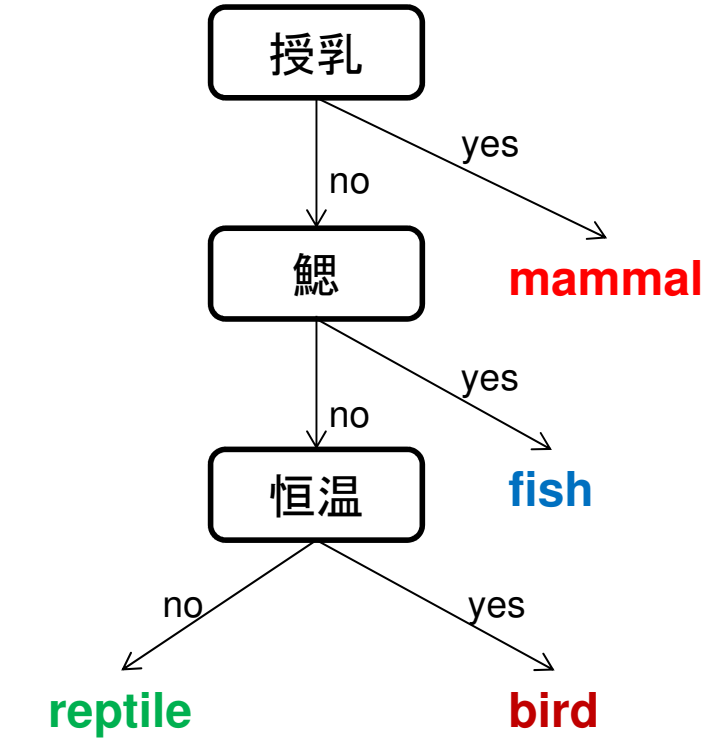
事例	カテゴリ	属性 1	属性 2	…
1	A	○	○	…
2	B	×	○	…
…	…	…	…	…



決定木：命題論理における帰納学習

- 例：動物分類
- 生物の特徴から，その「類」を推測する
  - 属性：｛　授乳，　鰓（えら），　体表，　足の数，　恒温？，　産卵？，　住処　｝
  - カテゴリ：｛　哺乳類，　魚類，　爬虫類，　鳥類　｝

	授乳	鰓	体表	足	恒温	産卵	住処	カテゴリ
dog	yes	no	hair	4	yes	no	land	mammal
dolphin	yes	no	none	0	yes	no	water	mammal
platypus	yes	no	hair	2	yes	yes	water	mammal
bat	yes	no	hair	2	yes	no	air	mammal
trout	no	yes	scale	0	no	yes	water	fish
herring	no	yes	scale	0	no	yes	water	fish
shark	no	yes	none	0	no	yes	water	fish
eel	no	yes	none	0	no	yes	water	fish
lizard	no	no	scale	4	no	yes	land	reptile
crocodile	no	no	scale	4	no	yes	water	reptile
t_rex	no	no	scale	4	no	yes	land	reptile
turtle	no	no	scale	4	no	yes	water	reptile
snake	no	no	scale	0	no	yes	land	reptile
eagle	no	no	feathers	2	yes	yes	air	bird
ostrich	no	no	feathers	2	yes	yes	land	bird
penguin	no	no	feathers	2	yes	yes	water	bird



R1：授乳⇒mammal
R2：¬授乳∧鰓⇒fish
R3：¬授乳∧¬鰓∧恒温⇒bird
R4：¬授乳∧¬鰓∧¬恒温⇒reptile

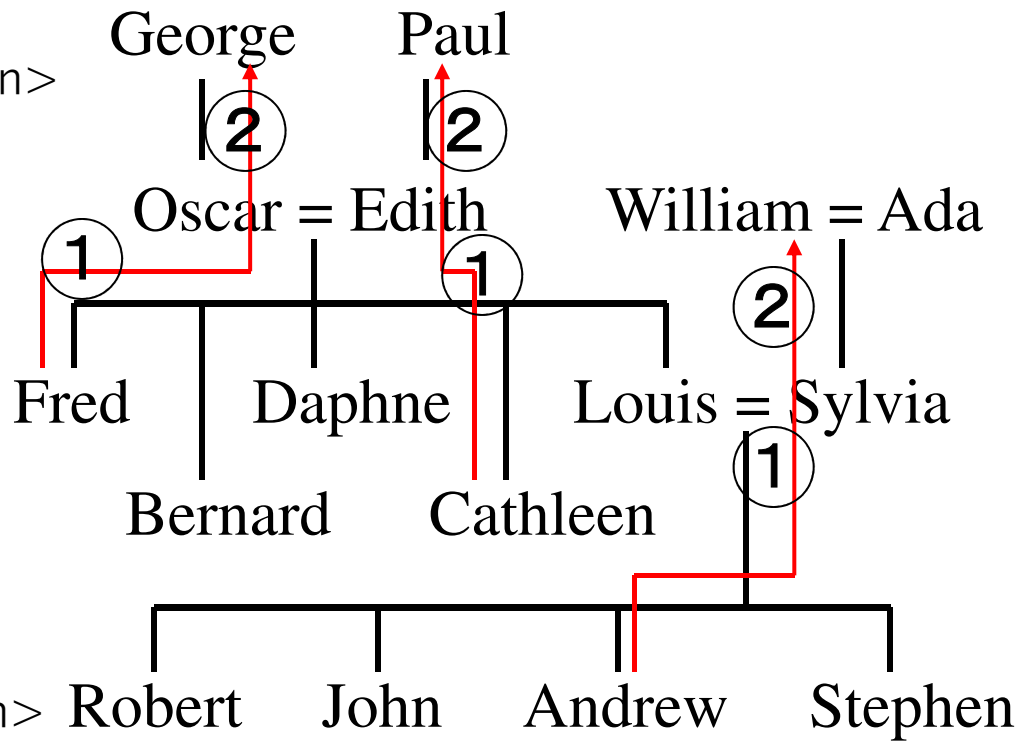
命題論理学習器の限界と帰納論理プログラミング

- 決定木：命題論理に基づく学習器
  - （拡張により）記号データと数値データの両方を扱うことができる.
  - 一つの表しか扱うことができない
    - 複数の関係表がある場合には，（データベースの）結合演算によって単一の表に変換する必要があるが，一般に，表が大きくなる／欠損が増えるなど，現実的ではない.
  - ルールとして表現された関連知識を扱うことができない：漸増的な学習ができない
  - 複雑な構造を扱うことができない

- 帰納論理プログラミング：述語論理に基づく学習器
  - 応用対象に依存しない表現／複数の表からの学習
  - 関連知識（背景知識）を利用した事例の一般化
  - 事例に間接的に関係する知識や公理などを利用可能
  - 漸増的な学習可能＝ある学習結果を，他の学習に利用することが可能
  - 共通したパターンの抽出
    - 関係（＝構造）のレベルでの共通パターンの抽出

関係の学習の例：祖父母

祖父母の関係の事例
< George, Fred> <George, Daphne>
<George, Bernard> <George, Cathleen>
<George, Louis> <Paul, Fred>
<Paul, Daphne> <Paul, Bernard>
<Paul, Cathleen> <Paul, Louis>
<Oscar, Robert> <Oscar, John>
<Oscar, Andrew> <Oscar, Stephen>
<Edith,Robert> <Edith, John>
<Edith, Andrew> <Edith, Stephen>
<Ada,Robert> <Ada, John>
<Ada, Andrew> <Ada, Stephen>
<William,Robert> <William, John>
<William, Andrew> <William, Stephen>



祖父母関係ではない例
<George, Oscar> <George, Paul>
<Oscar, Edith> <Edith, Louis> …

親の親が祖父母である
（上に２つ進むとたどり着く）

ルールの利用：論理プログラムの自動合成

```
% Positive examples
reverse([],[]).
reverse([1],[1]).
reverse([1,2,3],[3,2,1]).
reverse([0,1,2],[2,1,0]).
reverse([2],[2]).
reverse([3],[3]).
reverse([4],[4]).
reverse([1,2],[2,1]).
reverse([1,3],[3,1]).
reverse([1,4],[4,1]).
reverse([2,2],[2,2]).
reverse([2,3],[3,2]).
reverse([2,4],[4,2]).
```

```
% Negative examples
:- reverse([1],[]).
:- reverse([],[0]).
:- reverse([0,1],[0,1]).
:- reverse([0,1,2],[2,0,1]).
:- reverse([1,2,3],[2,3,1]).
:- reverse([1,2,3],[3,2,4]).
:- reverse([1,2,3],[4,2,1]).
```

```
% Background Knowledge
append( [],Y,Y).
append( [W|X],Y,[W|Z] ):-
        append( X,Y,Z ).
```

```
% Hypotheses
reverse([],[]).
reverse([A|B],C):-
    reverse(B,D), append(D,[A],C).
```

抽出されるパターン

reverseの第一引数の値を，
先頭要素Aと残りのリストBに分け，
Bを反転したリストDの末尾にAを追加した
リストEが，reverseの第二引数Cと同じ．

reverse([A | B], C) ← reverse(B , D),append(D ,[A], E), E=C.
reverse([1 | []], [1]) ← reverse([] , []),append([] ,[1], [1]), [1]=[1].
reverse([2 | []], [2]) ← reverse([] , []),append([] ,[2], [2]), [2]=[2].
reverse([3 | []], [3]) ← reverse([] , []),append([] ,[3], [3]), [3]=[3].
reverse([1 | [2]], [2,1]) ← reverse([2] , [2]),append([2] ,[1], [2,1]), [2,1]=[2,1].
reverse([2 | [3]], [3,2]) ← reverse([3] , [3]),append([3] ,[2], [3,2]), [3,2]=[3,2].
reverse([1 | [2,3]], [3,2,1]) ← reverse([2,3] , [3,2]),append([3,2] ,[1], [3,2,1]), [3,2,1]=[3,2,1].
reverse([1 | []], []) ← reverse([] , []),append([] , [1], [1]), [1]≠[].
reverse([] , [0]).
reverse([0 | [1]], [0,1]) ← reverse([1] , [1]),append([1] , [0], [1,0]), [1,0]≠[0,1].
reverse([1 | [2,3]], [2,3,1]) ← reverse([2,3] , [3,2]),append([3,2] , [1], [3,2,1]), [3,2,1]≠[2,3,1]
reverse([0 | [1,2]], [2,0,1]) ← reverse([1,2] , [2,1]),append([2,1] , [0], [2,1,0]), [2,1,0]≠[2,0,1]

帰納推論の問題設定

帰納論理プログラミングの問題設定（Logical Settings）

- 問題設定（論理的設定：Logical Settings）
  - 入力情報と出力情報に関する論理的な関係性を表したもの
  - 解が満たすべき論理的な性質を定めるもの

- 様々な設定が提案されている
  - Learning from Entailment （伴意からの学習）
  - Learning from Interpretation（解釈からの学習）
  - Learning from Satisfiability（充足可能性からの学習）
  - Learning from Answer sets（解集合からの学習）

- ちなみに．．．
  - ILPが始まったころの問題設定：Learning from Entailment
  - 決定木などの命題論理学習：Learning from Interpretation

伴意からの学習

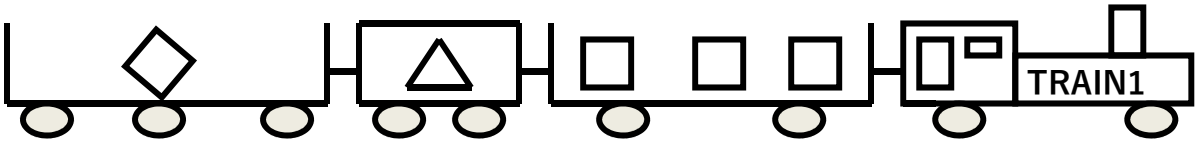
伴意からの学習：Learning from Entailment

これが「問題設定」

- 入力：（述語論理で表現された）正例E^+，負例E^-，関連知識（背景知識）BK
  - $BK \not\models E^+$（背景知識だけでは正例が説明できない）
- 出力：（述語論理で表現された）仮説H
  - $BK \cup H \models E^+$（背景知識と共に正例を導出（説明）できる）
  - $BK \cup H \not\models E^-$（背景知識と合わせても負例を導出（説明）できない）

- 必要なこと
  1. 問題の形式的な記述：入力データの表現（BK, E^+, E^- の表記）
  2. 共通パターンの形式的な記述：仮説の表現（H の表記）
  3. 各事例に特定のパターンが存在するかの機械的なチェック：被覆計算
     - $BK \cup H \models E^+$ と $BK \cup H \not\models E^-$ の確認
  4. 機械的なパターンの生成・列挙：仮説空間の設定と探索
     - どうやってHを作るのか？
     - 種々の方法が考えられるが，今回は探索のコンテクストで．
  5. 機械的なパターンの評価：仮説に対する評価関数
     - 条件を満たす仮説は複数存在する可能性がある．その順位付けが必要

問題の形式的な表現：この授業では確定節プログラムを前提とする



正例：基礎アトム

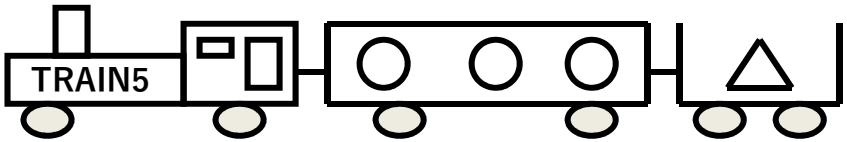
east(train1).	train1は東に向かっている

背景知識（任意の論理プログラム）

has_car(train1, car11).	train1には，car11が連結されている
shape(car11, rectangle).	car11の形は四角い
open(car11).	car11は屋根がない
long(car11).	car11 の長さは長い
wheels(car11, 2).	car11にはタイヤが２つついている

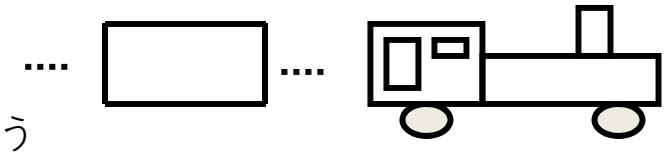
負例：基礎アトムの否定

:- east(train5).	train5は東に向かっていない



仮説（共通パターン）の形式的な表現：（変数を伴う）ルール

east(A) :- has_car(A,B), short(B), closed(B).



長さが短く屋根が閉じている車がある列車は東へ向かう

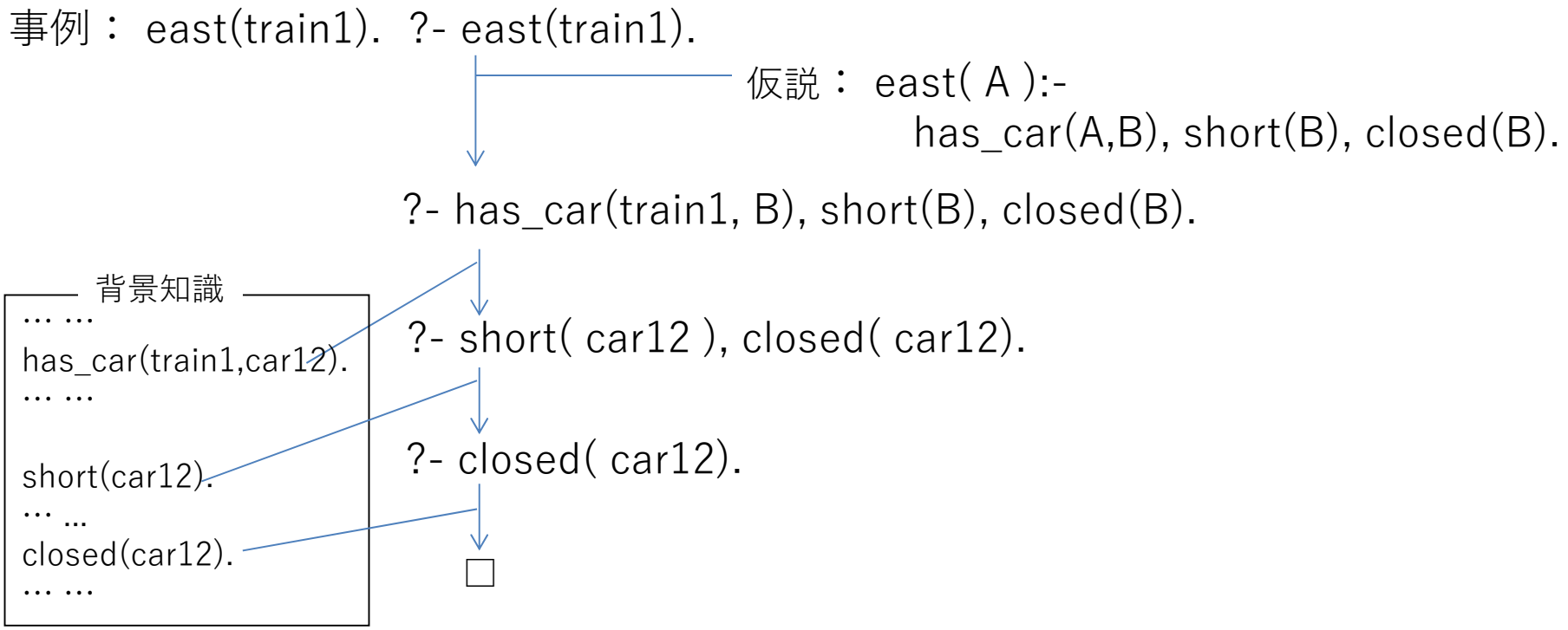
やりたいこと（帰納推論）：
背景知識を利用して事例を説明する（しない）ルールを作る

※頭部：事例と同じ述語と持つアトム
本体部：背景知識中に現れる述語の連言

各事例に特定のパターンが存在するかの機械的なチェック

- 被覆計算：各事例が共通パターンを持っているか否かのチェック？
  - $BK \cup H \models e^+ \in E^+$
  - $BK \cup H \not\models e^- \in E^-$
- 演繹計算で確認可能
  - 確定プログラムの場合，例えば，事例をゴールとした融合法による反駁証明

融合法の反駁証明は，
完全かつ健全



機械的なパターンの生成・列挙：仮説空間の設定と探索

- 枚挙法：あり得る仮説を次々と生成し，その中から条件に合う仮説を選択する
  - 仮説生成：（確定節プログラムの場合）ボディに現れる可能性のあるリテラルの並びを生成
  - 仮説の評価・選択
    - なるべく多くの正例を説明でき，なるべく少ない数の負例を説明する仮説が良い
    - 長さ（＝本体部のリテラル長）が短い仮説が良い　など

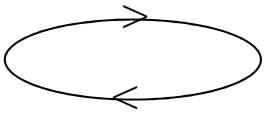
→仮説の生成・選択は，仮説空間における最適仮説の「探索問題」

```
east(t1).
east(t2).
 ….
has_car(t1,car11).
shape(car11,rectangle).
long(car11).
open(car11).
load(car11,rectangle,3).
wheels(car11,2).
 ….
:- east(t5).
:- east(t6).
 ….
```

※頭部：事例と同じ述語と持つアトム
本体部：背景知識中に現れる述語の連言

本体部リテラルの組み合わせを考えればよい

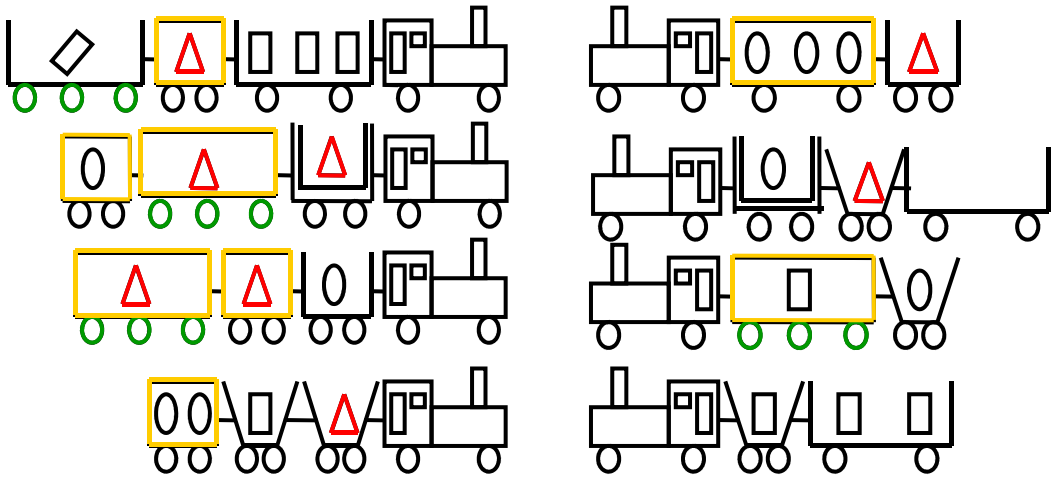
仮説(パタン)の生成
（背景知識を組み合わせて一般化する）

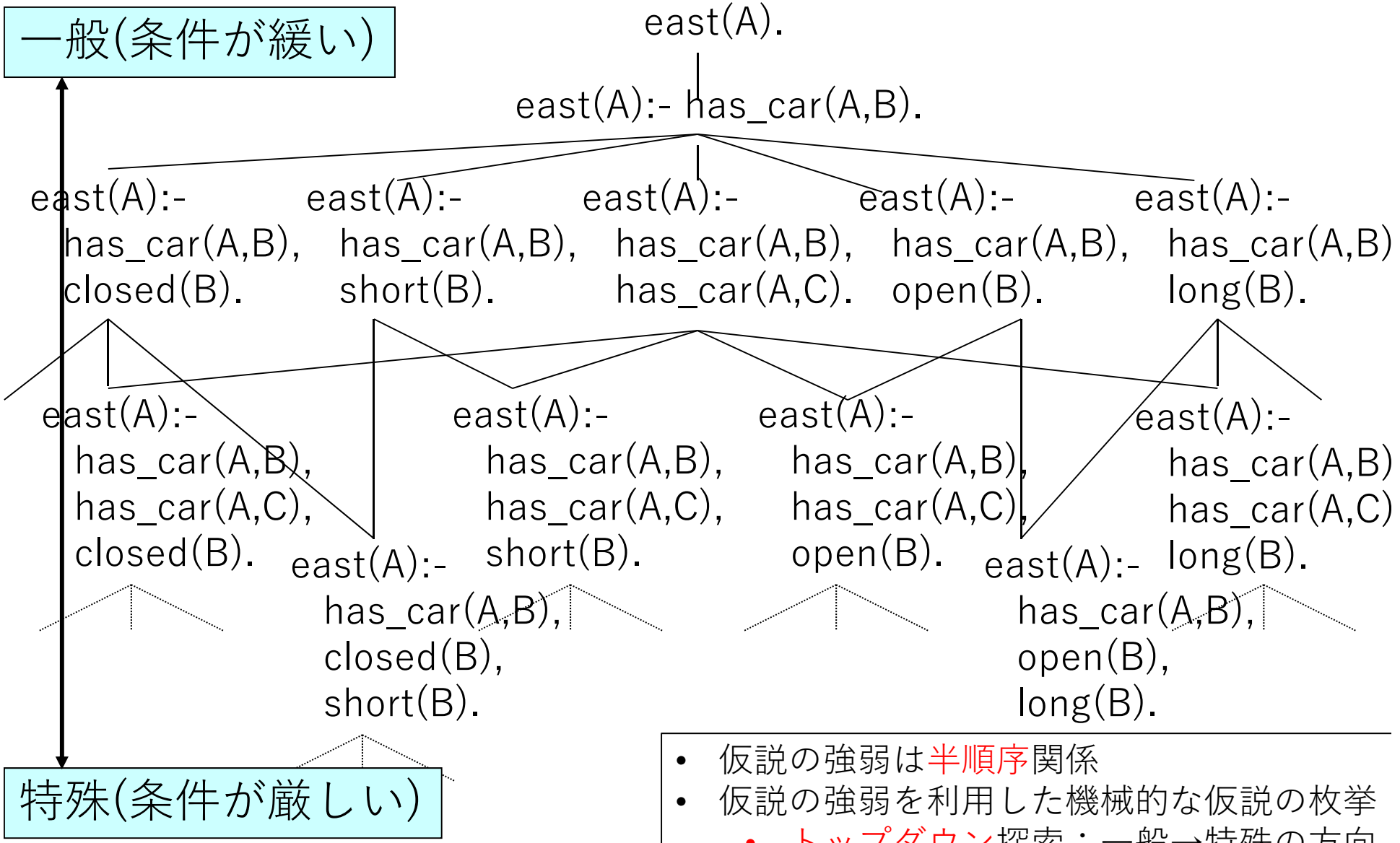


仮説の評価
（説明される(パタンをもつ)正負事例の数）

```
east(A):-has_car(A,B),load(B,triangle,1).   P=4,N=2
east(A):-has_car(A,B),wheels(B,3).          P=3,N=1
east(A):-has_car(A,B),closed(B).            P=4,N=2
   …..  ….
```

→ 最も評価値の高い組み合わせが最終的な解





- 仮説の強弱は半順序関係
- 仮説の強弱を利用した機械的な仮説の枚挙
  - トップダウン探索：一般→特殊の方向
  - ボトムアップ探索：特殊→一般の方向
- 探索問題として定式化できる

帰納推論の困難性

- 多くの事実から，どうやって特徴的なパターンを抽出すればよいのか
- 基本的な考え方
  - 仮説の生成･検査法(枚挙法)
  - 仮説の「形」を定めて，その形に合う仮説を系統的に生成し，検査する

- 仮説空間： 一階述語論理を表現の枠組みとすると，その仮説空間は膨大になる

  →空間を限定する（狭める）工夫の導入

  1. 仮説の強弱の利用
  2. 言語バイアス
  3. ボトムの導入など

概念の強弱

- 概念が強い ＝ 一般＝多くの事例を説明できる
- 概念が弱い ＝ 特殊＝少ない事例しか説明できない

事例： f(1, 1). f(2, 2). f(1, 2). f(2, 3).
背景知識：h(1, 2). h(1, 3). h(2, 1). h(2, 3). h(3, 1). h(3, 2).
仮説

H1: f(A, B). → 被覆される事例 { f(1, 1). f(2, 2). f(1, 2). f(2, 3). }
H2 : f(A, A). → 被覆される事例 { f(1, 1). f(2, 2). }
H3 : f(A, B):- h(A, B). → 被覆される事例 { f(1, 2). f(2, 3). }

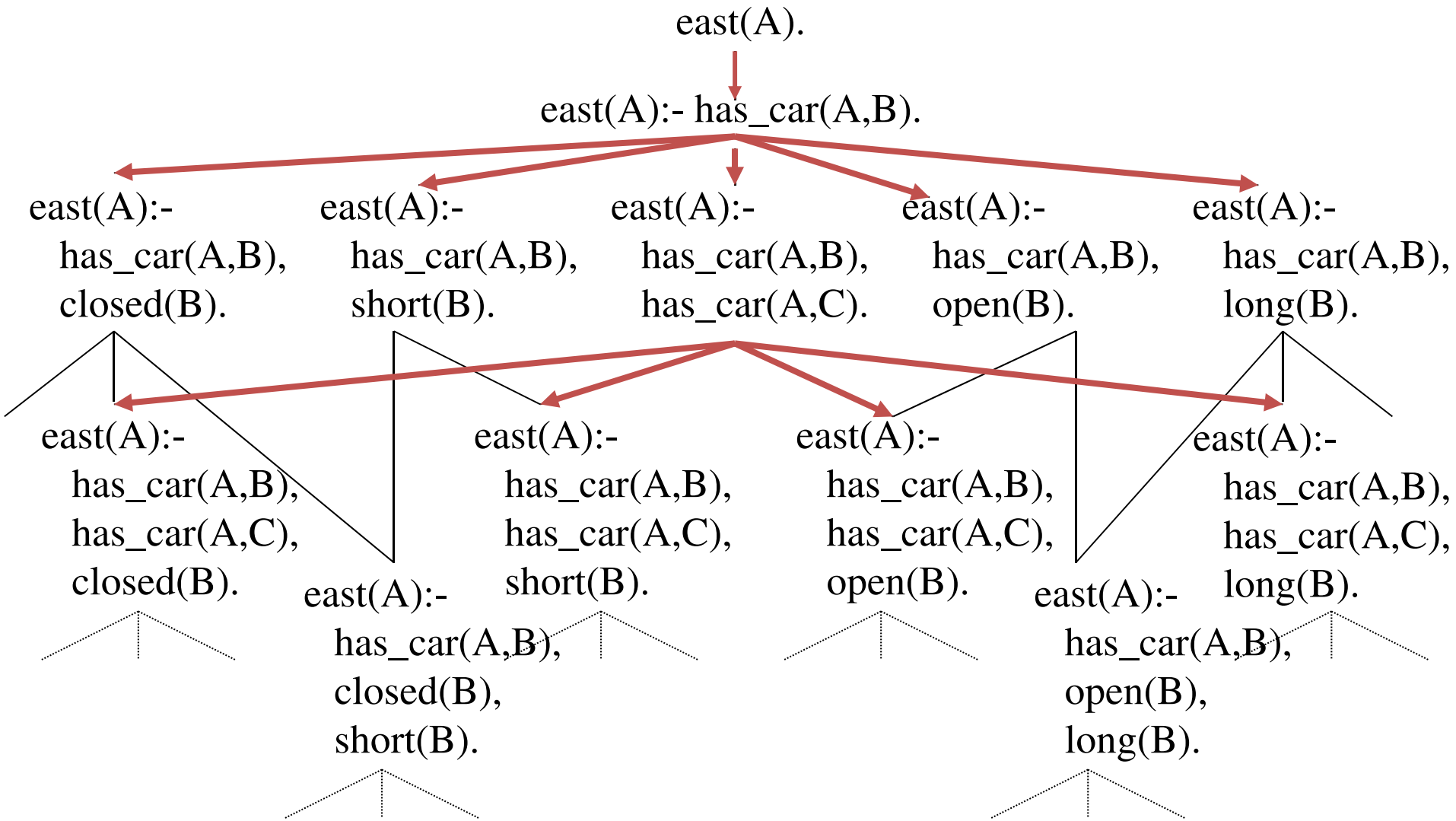
これより．．．

H1: f(A, B). もっとも一般的，変数に関する制限がない
H2: f(A, A). “f(A,B).”より特殊/ ←変数に関する制約
H3: f(A, B):-h(A, B). “ f(A,B).”より特殊/ ←本体部へのリテラルの追加

- 概念の特殊化：「代入による制約」と「リテラルの追加」＝ 包摂
  - 定義：包摂（subsumption）
    - 二つの節PとQに対し，条件$P\theta \subseteq Q$を満たす代入が存在するとき，節PはQを包摂（subsume）するといい， $P \succcurlyeq Q$と表す． （各節はリテラルの集合と見做す）
  - 例：$(1)\ p(X) \succcurlyeq p(a),\quad (2)\ p(X) \vee \neg q(X) \succcurlyeq p(X) \vee \neg q(X) \vee \neg r(X),\quad (3)\ p(X) \vee p(Y) \succcurlyeq p(Z)$

探索空間のトップダウン探索

- 最も一般的な仮説からはじめて，だんだんに特殊化な仮説を生成する
- 精密化演算子(Refinement Operator)の適用：現在の仮説に対する，変数の特化とリテラルの追加



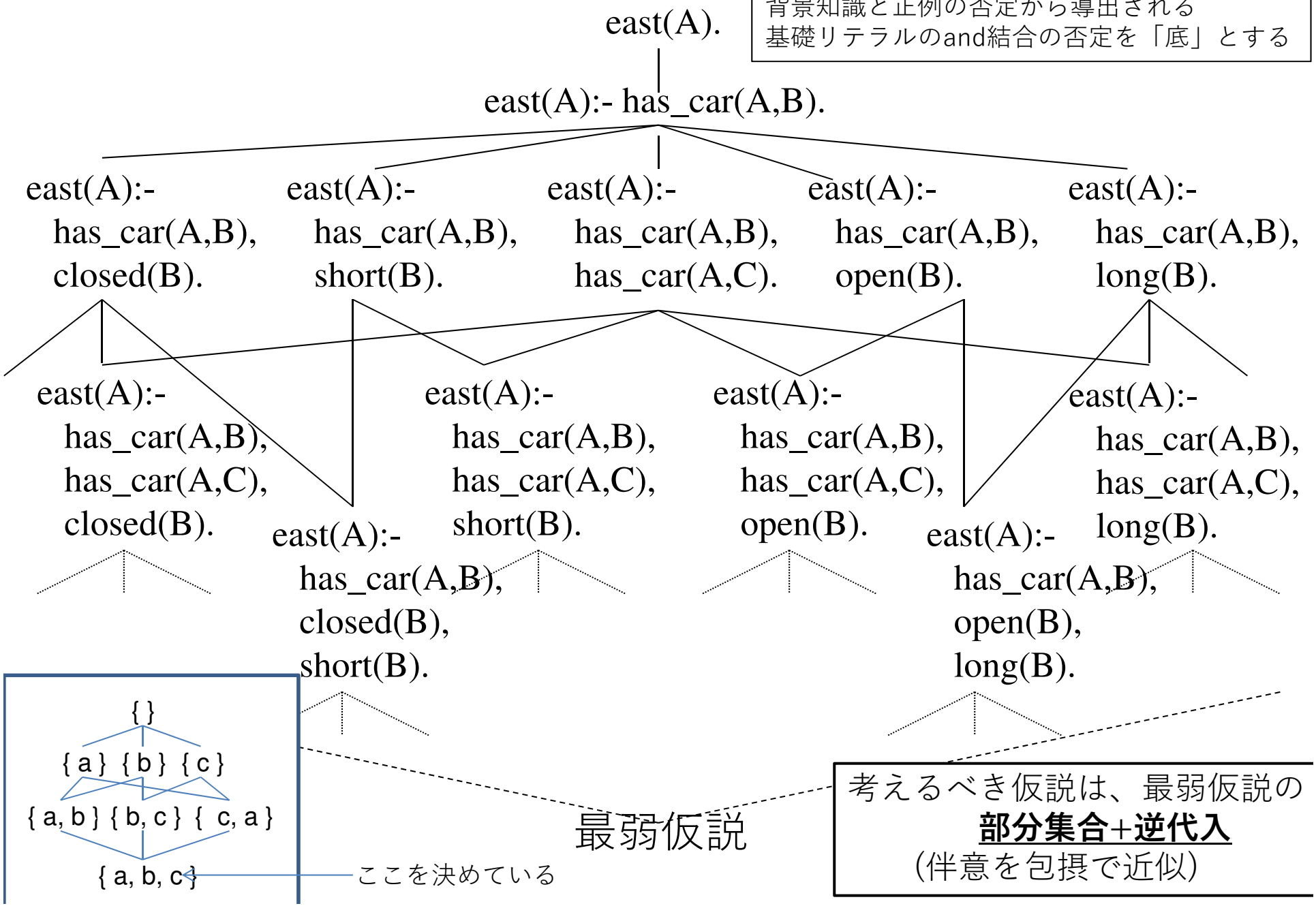
仮説空間の探索

- 仮説空間は，背景知識によって構成される．
  - 引数の組み合わせ爆発の可能性
- 探索における制限の必要性
  1. 探索空間の実質的な縮小→ボトムの決定
  2. （全解探索ではなく）Greedy探索やBeam探索の採用
- それだけでは不十分／さらなる空間の制限が必要
  →言語バイアス
  - 仮説の長さ／変数の数／ルールの形に関する制限
- 宣言バイアス
  - 言語の宣言的意味上の制約
    - モード宣言／タイプ情報／長さ制限／変数深度　など
- 手続きバイアス
  - 言語で表現された分類判定プログラムの実行時に関する制約
    - 推論深度 (h-easy) など

p(A):- q(A,A,A).
p(A):- q(A,A,B).
p(A):- q(A,B,A).
p(A):- q(A,B,B).
p(A):- q(B,A,A).
p(A):- q(B,A,B).
p(A):- q(B,A,C).
p(A):- q(B,B,A).
p(A) : q(B,C,A).
p(A):- q(A,B,C).
⋮

逆伴意：探索空間の「底」の決定

直感的には．．
$BK \cup H \models e^+$より$BK \cup \{\neg e^+\} \models \neg H$に着目し，
背景知識と正例の否定から導出される
基礎リテラルのand結合の否定を「底」とする



考えるべき仮説は、最弱仮説の
<u>部分集合+逆代入</u>
(伴意を包摂で近似)

仮説の評価

- 仮説は負例を説明してはいけない：負例を用いて過学習を避ける
- 正例のみを被覆し，負例を説明しない仮説は複数考えられる
  - これらの中から，適切な基準を用いて仮説を選択（評価）する必要がある

- これまでにいくつかの評価関数が提案されている
- 記述長最小原理
  - 仮説評価のために，記述長を利用
  - Compression Gainがもっとも大きい（負事例を説明しない）仮説が最良の仮説

[非常に簡易に近似すると．．]

正事例を（仮説（ルール）＋例外）に置き換える
その記述量の差を計算する

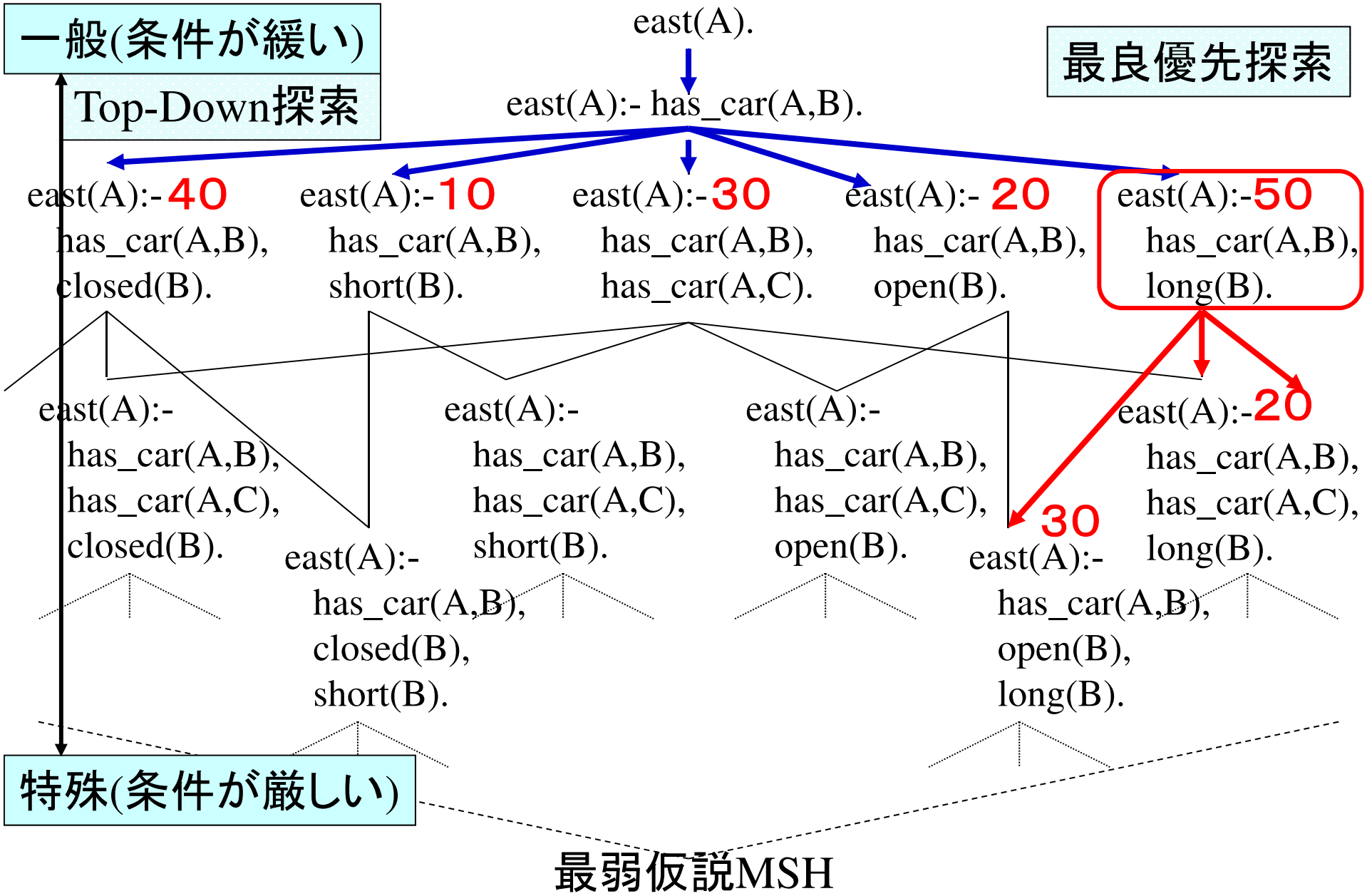
Compression Gain ＝

説明される正事例の数 － 仮説のリテラル長　－ 説明される負事例の数

＝｜概念の外延表現｜ －　｜概念の内包表現｜－　｜例外表現｜

- H1: east(A):- has_car(A,B), closed(B), short(B).
  {train1, train2, train3, train4} Compression Gain 4 - 3 = 1.
- H2: east(A):- has_car(A,B), long(B), load(B,circle, 2).
  {train4} Compression Gain 1-3 = -2
- H3: east(A):- has_car(A,B), long(B), open(B), has_car(A,C), long(C), open(C), infront(C,B).
  {train1} Compression Gain 1-7 = -6
- H4: east(A):- has_car(A,B),load(B,triangle,1).
  {train1, train2, train3, train4, train5, train6}　Compression Gain 4 - 2 - 2 = 0

評価値を用いたトップダウン・最良優先探索



集合被覆アルゴリズム（カバーセットアルゴリズム）

- 単一の仮説では，全ての正事例を説明しきれない場合，複数の仮説を用いる
- Separate and Conquer（分離統治？）vs Divide and Conquer（分割統治）
  - Divide and Conquer：データを分割して，そのそれぞれに対してルールを作る（決定木学習など）
  - Separate and Conquer ：データ全体からルールを一つ作り，得られたルールを利用してデータを更新することを繰り返す→集合被覆アルゴリズム

- 集合被覆アルゴリズム
  - 正事例集合の中から一つの事例を選択する
  - 仮説空間の探索によりその一般化を試みる
  - 最適な仮説が見つかったら，説明される正事例を元の正事例集合から取り除く

```
1:  H := ∅
2:  while E+ ≠ ∅
3:      e := select(E+)
4:      h := search(e, E+, E−, BK ∪ H)
5:      H := H ∪ {h}
6:      E+ := E+ \ {e ∈ E+ | BK ∪ H ⊨ e}
7:  return H
```

再掲

伴意からの学習：Learning from Entailment

これが「問題設定」

- 入力：（述語論理で表現された）正例E^+，負例E^-，関連知識（背景知識）BK
  - $BK \not\models E^+$（背景知識だけでは正例が説明できない）
- 出力：（述語論理で表現された）仮説H
  - $BK \cup H \models E^+$（背景知識と共に正例を導出（説明）できる）
  - $BK \cup H \not\models E^-$（背景知識と合わせても負例を導出（説明）できない）

- 必要なこと
  1. 問題の形式的な記述：入力データの表現（BK, E^+, E^- の表記）
  2. 共通パターンの形式的な記述：仮説の表現（H の表記）
  3. 各事例に特定のパターンが存在するかの機械的なチェック：被覆計算
     - $BK \cup H \models E^+$ と $BK \cup H \not\models E^-$ の確認
  4. 機械的なパターンの生成・列挙：仮説空間の設定と探索
     - どうやってHを作るのか？
     - 種々の方法が考えられるが，今回は探索のコンテクストで．
  5. 機械的なパターンの評価：仮説に対する評価関数
     - 条件を満たす仮説は複数存在する可能性がある．その順位付けが必要

帰納論理プログラミングの応用

帰納論理プログラミングの応用領域の例

- bio-chemistry
- protein engineering
- drug-design
- natural language processing
- finite element mesh design
- satellite diagnosis
- games
- medicine
- robotics
- skill acquisition
- planning
- software engineering
- music analysis
- agents
- ecology
- traffic analysis
- network management
- Deductive database design

応用例1：突然変異誘発性物質の判定

- R.D.King, A.Srinivasan and M.J.E.Sternberg,
  “Relating chemical activity to structure: an examination of ILP successes”,
  New Generation Computing, Vol.13, pp411-433, 1995.

- 問題 ：２３０種の化合物について，それらが突然変異性を示すか否かを例として与え，分類規則を発見する（化合物は，複雑な構造をしているため，命題論理では表現が困難）

- 化学者による解　：　重回帰分析式による判別
  - １８８の化合物に関しては，線形回帰モデルで判別可能
  - 残りの４２の化合物は，対象外とした
- 説明変数
  - log P ：化合物の疎水性係数のlog
  - ε LUMO : 化合物の最低空分子軌道のエネルギー
  - I1 : 3つ以上の連続したベンゼン核を持っているか否かを示す指示変数
  - Ia : 5つの予想外に低い誘発性を示すアセン系化合物を示す指示変数

- 得られた線形回帰式

$$\log M = 0.65(\pm 0.16) \log P - 2.90(\pm 0.59) \log(\beta 10 \log P + 1) - 1.38(\pm 0.25)\ \varepsilon LUMO + 1.88(\pm 0.39)\ I1 - 2.89(\pm 0.81)\ Ia - 4.15(\pm 0.58)$$

$$(\log M = \log \text{mutagenecity},\quad \log \beta = -5.48)$$

帰納論理プログラミングによる判定

- 結合レベルのデータ
  - 標準分子モデリングパッケージQUANTAから，230の化合物に関する原子と結合構造の情報を抽出
  - 化合物を構成する原子
    - 原子のタイプ（芳香族炭素，アリール族炭素など２３３タイプ）
    - 原子の部分電荷量
  - 結合のタイプ（芳香，単，複など８タイプ）

- 正負事例
  - Case1 :Regression Friendly 188個の化合物のうち，125個を正例，63個を負例とする
  - Case2 :Regression Unfriendly 42個の化合物のうち，13個を正例，29個を負例とする

- 背景知識
  - bond(化合物, 原子1, 原子2, 結合タイプ)
    - “化合物”が“原子1”と“原子2”の間に“結合タイプ”の結合を持っている
  - atm (化合物, 原子, 原子記号, 原子タイプ, 電荷)
    - “化合物”中の“原子”は“原子タイプ”の型の“原子記号”で，電荷は“電荷”である

得られたルール

- Rule1 active(A) :- atm(A,B,c,195,C).
  Accuracy = 100%, Coverage=10%
- Rule2 active(A):- atm(A,B,c,10,C), atm(A,D,c,22,E), bond(A,D,B,1).
  Accuracy = 84%, Coverage=30%
- Rule3 active(A):- atm(A,B,c,27,C), bond(A,D,E,1), bond(A,B,E,7).
  Accuracy = 90%, Coverage = 58%
- Rule4 active(A) :- atm(A,B,o,40,C), atm(A,D,n,32,C).
  Accuracy = 71%, Coverage = 8%
- Rule5 active(A):- atm(A,B,o,40, -0.383).
  Accuracy = 82%, Coverage = 7%
- Rule6 active(A) :- atm(A,B,o,40, -0.384).
  Accuracy = 89%, Coverage = 13%
- Rule7 active(A) :- atm(A,B,o,40,-0.378).
  Accuracy = 100%, Coverage = 5%
- Rule8 active(A) :- atm(A,B,h,3,0.149)
  Accuracy = 88%, Coverage = 6%
- Rule9 active(A) :- atm(A,B,h,3,0.144)
  Accuracy = 89%, Coverage = 6%

Regression Unfriendly Data
active(A) :- bond(A,Y,Z,2),
bond(A,D,Y,1), atm(A,D,c,21,E).
Accuracy = 100%, Coverage = 62%

応用例2：電子メールの分類システム

- K.Shimazu and K.Furukawa：“Knowledge Discovery in Database by PROGOL - Design,Implementation and its Application to Expert System Building-”,Proc. of the 1st Int. Conf. on the Practical Application of Knowledge Discovery and Data Mining, 1997.
- 問題
  - 企業における定常的な商品の利用に関する電子メールによる質問文の分類
  - 不定長のリストを扱う必要があるため，命題論理学習器では扱いが困難

事例：質問文から抽出したキーワードのリスト

form85(['XXX', '自動操作', '電源切', 'モード', '切り替え', '単価']).
form85(['XXX', '月間', '表示', '電気', '優先', '自動操作', '電源切']).
form85(['自動操作', 'XXX', '紹介', '電源切', '登録', '方法', '表示']).

背景知識

- Listにキーワード KWが含まれる． have(List,KW):-!,member(KW,List).
- Listのキーワード KWが含まれない not_have(List,KW):- keyword(KW),¥+have(List,KW).
- リストで，キーワードAがキーワードBの前に現れる
  - in_order([A|T],A,B):-member(B,T).
  - in_order([_|T],A,B):-in_order(T,A,B).

仮説の例

form85(A):- have(A,'方法'),in_order(A,'XXX','モード'), not_have(A,'YYY').
form85(A):-in_order(A,'電源', '方法').

ILPシステムAleph

ILP システムAleph

- ILPシステムProgol (by S. Muggleton) + αのProlog実装
  - 種々の探索手法・種々の論理設定（LogicalDT/AR）・ユーザ定義評価関数など，多くの拡張が施されている
  - （余談ですが．．）私が学生のころからあるシステムです

- オリジナルサイト：https://www.cs.ox.ac.uk/activities/programinduction/Aleph/aleph.html
  - Yap-Prolog上での実装
    - Yap自体は，Ubuntuなら　$ apt install yap　で楽々インストール可能

- 入手先：https://github.com/friguzzi/aleph
  - SWI-Prologへの移植済み．こちらの方が使いやすいかも？

- マニュアルを見て動かしてみよう
  - $ yap　　#Yapの起動
  - ?- ['aleph.pl'].　　#Alephのロード（必要に応じて相対パスで指定）
  - ?- read_all(問題)　　#問題のロード（必要に応じて相対パスで指定）
  - ?- induce.　　#推論開始 → 結果が表示される