論理と計算

第１３回
高次推論：帰納推論の発展

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

- 標準論理プログラムを対象とした帰納推論
  - 概要
  - いくつかの問題設定
    - Cautious Induction / Brave Induction / Induction of Stable Models

- ILASP（Inductive Learning of Answer Set Programs）
  - 問題設定(learning from Answer Sets)
  - 学習アルゴリズム

- ILASPシステム
  - 問題設定の拡張
  - 問題の与え方（言語バイアス）
  - 例題

標準論理プログラムの帰納推論

準備：標準論理プログラムの伴意と帰納推論の問題設定

- 標準論理プログラムPは，複数の安定モデル（解集合）を持つ.
  - AS(P)：プログラムPの安定モデル（解集合）の全体集合

- 基礎原子式（基礎アトム）e に対する 2 種類の伴意関係
  - Cautious Entailment：すべての解集合（安定モデル）がeを含む

    $$P \models_c e \Leftrightarrow \forall A \in AP(P)[e \in A]$$
  - Brave Entailment：eを含む解集合（安定モデル）が存在する

    $$P \models_b e \Leftrightarrow \exists A \in AP(P)[e \in A]$$

- 標準倫理プログラムの帰納推論に関する問題設定（論理設定）の例
  - Cautious Induction
  - Brave Induction
  - Induction of Stable Models
  - Learning from Answer Sets

- 帰納推論実現に必要なこと
  - 問題の形式的な表現：BKは標準論理，正事例・負事例は？
  - 被覆（仮説によって事例が説明されるか否か）のチェック
  - パターンの生成：今回は明示的に与えます
  - パターンの評価：仮説長（基数制約の扱いに注意）など

例：

P = { 1 { p ; q } 2 :- r.,
r.,
:- not p, r. }

AS(P) = { {r,p}, {r, p, q} }

$P \models_c r$

$P \models_c p$

$P \models_b q$

Cautious Induction・Brave Induction

Cautious Inductionの論理設定

- 入力：
  - B：背景知識，S_M：仮説空間（ルール集合）
  - E^+：正例集合(基礎アトム)，E^-：負例集合(基礎アトム)
- 出力：以下の条件を満たす仮説H
  - $H \subseteq S_M$ … Hはルール集合
  - $AS(B \cup H) \neq \{\}$ … Bと併せたときに解集合を持つ
  - $\forall A \in AP(B \cup H)[E^+ \subseteq A, E^- \cap A = \{\}]$
    … すべての解集合は正例集合を含み，負例を含まない

Cautious Induction：条件が厳しすぎる場合がある
Brave Induction：制約が学習できない場合がある

背景知識，仮説はNLP
事例は基礎アトム

事前に与えた仮説空間から
条件に合う部分集合を獲得
※仮説はルールの集合

解集合に含まれる＝被覆される（説明される）

- Brave Inductionの論理設定
- 入力：Cautious Inductionと同じ
- 出力：以下の条件を満たす仮説H
  - $H \subseteq S_M$ … Hは仮説集合の部分集合
  - $\exists A \in AP(B \cup H)[E^+ \subseteq A, E^- \cap A = \{\}]$
    … 正例集合を含み，負例を含まない解集合が存在する

Cautiousでは解集合の全部が条件を満たす
Braveでは解集合の一つが条件を満たす

C. Sakama and K. Inoue: Brave induction: a logical framework for learning from incomplete information, Machine Learning 76(1):3–35, 2009

B = { bird(X):- penguin(X). bird(X):- sparrow(X).
penguin(penguin). sparrow(sparrow). }

E+ = { flies(sparrow) }
E- = { flies(penguin) }

SM = {
h1 .. flies(X):- bird(X).
h2 .. flies(X):- bird(X), not penguin(X).
h3 .. 0{ flies(X) }1 :- bird(X).
h4 .. 0{ flies(X) }1 :- bird(X), not penguin(X).
}

基本：
事例に現れる述語を頭部に，
背景知識に現れる述語を本体部に
持つルールを考える

{ h1, h2, h3, h4 }の部分集合が仮説の候補
以下では， 4候補のみを示す

A = { bird(penguin), bird(sparrow), penguin(penguin), sparrow(sparrow) }
AS(B) = { A }

仮説 {h1}： AS(B∪{h1}) = { { flies(penguin), flies(sparrow) }∪A } → ×（解ではない）

仮説{h2}： AS(B∪{h2}) = { { flies(sparrow) }∪A } → { cautious, brave } inductionの解

仮説{h3}： AS(B∪{h3}) = { **A**, { flies(sparrow) }∪A}, { flies(penguin)}∪A},
{flies(sparrow),flies(penguin) }∪A } → brave inductionの解

仮説{h4}： AS(B∪{h4}) = { **A**, {flies(sparrow) }∪A } → brave inductionの解

Induction of stable models

解釈：プログラムを真にするアトムの集合
e^{inc}：解釈に含まれるべきアトムの集合
e^{exc}：解釈に含まれるべきではないアトムの集合

- 部分解釈$e = \langle e^{inc}, e^{exc} \rangle$

背景知識，仮説はNLP
事例は部分解釈

- 解釈I が条件$[e^{inc} \subseteq I, e^{exc} \cap I = \{\}]$を満たすとき，Iはe を拡大する（満たす）という

Induction of stable modelの論理設定

- 入力：
  - B：背景知識，S_M：仮説空間（ルール集合）
  - E：部分解釈の集合
- 出力：以下の条件を満たす仮説H
  - $H \subseteq S_M$　　…Hはルール集合
  - $\forall \langle e^{inc}, e^{exc} \rangle \in E,\ \exists A \in AP(B \cup H)[e^{inc} \subseteq A, e^{exc} \cap A = \{\}]$

    … 事例である各部分解釈に対し，それぞれ，それを拡大する（ある）解集合が存在する

※Brave Inductionは，
$E = \{e\}$，　$e = \langle e^{inc} = E^{+}, e^{exc} = E^{-} \rangle$とする
Induction of stable models の特殊なケース

Induction of stable models は
Brave Inductionの一般化
しかし，制約の学習ができない場合も．．

- Brave Inductionとの違いに注意
  - Brave Induction：ある一つの解集合が，すべての事例を説明する(∃∀)
  - Induction of stable models：事例毎に，それを拡張する解集合は異なっても良い(∀∃)

R. P. Otero: Induction of stable models,
Proc. of the 11th International Conference on Inductive Logic Programming，pp.193–205, 2001.

B = {　　} #背景知識なし
E = {
　e1 = <{p}, {q} >, #pはモデルに含まれる，qはモデルに含まれない
　e2 = <{q}, {p} >　#qはモデルに含まれる，pはモデルに含まれない
}

SM = {
　h1 .. p :- not q.
　h2 .. q :- not p.
}

{ h1, h2 }の部分集合が仮説の候補

AS(B ∪ {h1}) = { { p } } → × ({ p } extends e1, but not e2)

AS(B ∪ {h2}) = { { q } } → × ({ q } extends e2, but not e1)

AS(B ∪ {h1,h2}) = { {p}, {q} } → ○ ({ p } extends e1, and { q } extends e2)

解は，{h1,h2} = { p :- not q.,　q :- not p. }

Inductive Learning of Answer Set Programs

Learning from Answer Sets

Learning from Answer Setsの論理設定

Induction of Stable Models に負事例を追加

- 入力：
  - B：背景知識，S_M：仮説空間（ルール集合），
  - E^+：部分解釈の集合，E^-：部分解釈の集合

背景知識，仮説はNLP
事例は部分解釈

- 出力：以下の条件を満たす仮説H
  - $H \subseteq S_M$ … Hはルール集合
  - $\forall \langle e^{inc}, e^{exc} \rangle \in E^+, \exists A \in AP(B \cup H)[e^{inc} \subseteq A, e^{exc} \cap A = \{\}]$
    … 各正事例（部分解釈）に対し，それぞれそれを拡大する解集合が存在する
  - $\forall \langle e^{inc}, e^{exc} \rangle \in E^-, \nexists A \in AP(B \cup H)[e^{inc} \subseteq A, e^{exc} \cap A = \{\}]$
    … 各負事例（部分解釈）に対し，それを拡大する解集合が存在しない

- Learning from Answer Sets ≒ Cautious Induction + Brave Induction
  - 正事例は，bravely entailed
  - 負事例は，その否定がcautiously entailed

M. Law, A. Russo, and K. Broda : Inductive learning of answer set programs,
Proc. of the 14th European Conference on Logics in Artificial Intelligence, pp.311-325, 2014.

B = { p :- r. }
E+ = {
  e1 = <{p}, {q} >, #pはモデルに含まれる，qはモデルに含まれない
  e2 = <{q}, {p} > #qはモデルに含まれる，pはモデルに含まれない
}
E- = {
  e3 = <{ }, { p,q } >,
  e4 = <{ p,q }, { } >
}
SM = {
  h1 .. q :- not r.
  h2 .. r :- not q.
}

{ h1, h2 }の部分集合が仮説の候補

AS(B ∪ {h1}) = { { q } } → × ({ q } extends e2, but not e2, e3, e4)

AS(B ∪ {h2}) = { { r, p } } → × ({ r, p } extends e1, but not e1, e3, e4)

AS(B ∪ {h1,h2}) = { { r, p }, {q} } → ○ ({ r, p } extends e1, but not e2, e3, e4
{ q } extends e2, but not e2, e3, e4)

解は，{h1,h2} = { q :- not r., r :- not q. }

仮説の評価

- 条件を満たす仮説（ルール集合）は複数考えられる
  - 何らかの順位付けが必要
    → 記述長最小原理（Minimum Description Length Principle ）≒ 短い方が良い

- Inductive Learning of Answer Set Programs における仮説長
  - 仮説（＝ルール集合）を構成するリテラル数＝各ルールを構成するリテラル数の総和
  - 仮説 { q :- not r., r :- not q. }の仮説長 → 2+2=4

    q :- not r. → 2

    r :- not q. → 2

- 注意：チョイスルール（基数制約）の扱い
  - 展開してからリテラル数を数える
  - 1 {p ; q} 1. → (p ∧ not q) ∨ (not p ∧ q) → 4
  - 0 {p ; q} 2. → (p ∧ q) ∨ (p ∧ not q) ∨ (not p ∧ q) ∨ (not p ∧ not q) → 8

仮説の導出

- Positive Solution：背景知識と共に各正事例を説明する仮説（ルール集合）
  - 条件　$\forall\langle e^{inc}, e^{exc}\rangle \in E^{+},\ \exists A \in AP(B \cup H)\left[e^{inc} \subseteq A, e^{exc} \cap A = \{\}\right]$　を満たすH
- Violating Solution：Positive Solutionのうち，負事例も説明してしまう仮説（ルール集合）
  - 条件　$\forall\langle e^{inc}, e^{exc}\rangle \in E^{+},\ \exists A \in AP(B \cup H)\left[e^{inc} \subseteq A, e^{exc} \cap A = \{\}\right]$
    $\wedge\ \exists\langle e^{inc}, e^{exc}\rangle \in E^{-},\ \exists A \in AP(B \cup H)\left[e^{inc} \subseteq A, e^{exc} \cap A = \{\}\right]$　を満たすH
- タスクT $= < B, S_M, E^{+}, E^{-} >$に対し，
  Positive Solutionの集合を pos_sol(T) , Violating Solutionの集合をvio_sol (T)と表記
  - 求めるべき仮説H*は，　pos_sol(T) \ vio_sol(T) の中で記述長が最小のもの
- ナイーブな方法：仮説空間（ルール集合）S_M のべき集合を一つずつ調べる
  - べき集合のサイズは$2^{|S_M|}$ なので，非現実的
- ILASPのアプローチ
  - 帰納推論の問題を，解集合プログラムへ変換する（clingoに解いてもらう！）
    - 背景知識，仮説，事例のそれぞれを解集合プログラムへ変換
      - メタなアプローチ
    - 仮説長を指定し，条件に合うルール集合を獲得（指定する仮説長を徐々に長くする）
      - Cf: Cover-set Algorithm (separate and conquer)は，一つずつルールを導出する

次ページ参照

解集合プログラムへの変換

B = { q :- r. }
E+ = { < {p, q}, {r, s } >, <{q},{ }> }
E- = { <{ p }, {q, t } > }
SM = { p., q :- r, not s. }

- B, S_M, E^+, E^-をそれぞれ変換する（meta変換）
- T = <B, S_M, E^+, E^-> に対し，変換で得られるルール集合をT^n_{meta}と表記する．

- 背景知識の変換
  - 各ルールに対し，ルール中の各アトムAを e(A,X) に置き換え，本体部にex(X) を追加したルールを生成
  - 変換後：e(**q**, X)**:-** e(**r**, X), ex(X)**.**

- 仮説空間の変換
  - 識別子 id を持つルールに対し，
    ルール中の各アトムAを e(A,X) に置き換え，
    本体部にactive(id) 及び ex(X) を追加したルールを生成
  - 識別子 id を持つルールRに対し，その長さ|R|を表すファクト length(id, |R|)を生成
  - 変換後：{ e(**p**, X):- active(h1), ex(X)**.**, length(h1, 1).,
    e(**q**, X)**:-** e(**r**, X), **not** e(**s**, X), active(h2), ex(X)**.**, length(h2, 3). }

- 仮説長をnに限定するための補助ルールを生成
  n #sum { active(R) = X : length(R, X) } n.　　※ASP実行時にnを指定する

※それぞれの変換の意味・意図・正当性を考えてみよう

B = { q :- r. }
E+ = { < {p, q}, {r, s } >, <{q},{ }> }
E- = { <{ p }, {q, t } > }
SM = { p., q :- r, not s. }

- 正事例の変換
  - 識別子がidである各正事例$\langle e^{inc}, e^{exc} \rangle$に対し，
    以下の3ルールを生成
  - ルール1 : ex(id)
  - ルール2： :- not covered(id).
  - ルール3：頭部はcovered(id)，本体部はe^{inc}中の各アトムAに対するe(A,id)とe^{exc}中の各アトムBに対するnot e(B,id)の連言
  - 変換後：{ ex(pos1).,
    :- not covered(pos1).,
    covered(pos1):- e(**p**, pos1), e(**q**, pos1), not e(**r**, pos1), not e(**s**, pos1).,
    ex(pos2).,
    :- not covered(pos2).,
    covered(pos2):- e(**q**, pos2). }
- 負事例の変換
  - 各負事例$\langle e^{inc}, e^{exc} \rangle$に対し，以下の1ルールを生成
  - ルール1：頭部はvoilating，本体部はe^{inc}中の各アトムAに対するe(A,neg)とe^{exc}中の各アトムBに対するnot e(B,neg)の連言（※事例の識別子ではなくすべてneg）
  - 変換後：{ violating :- e(**p**, neg), not e(**q**, neg), not e(**t**, neg). }

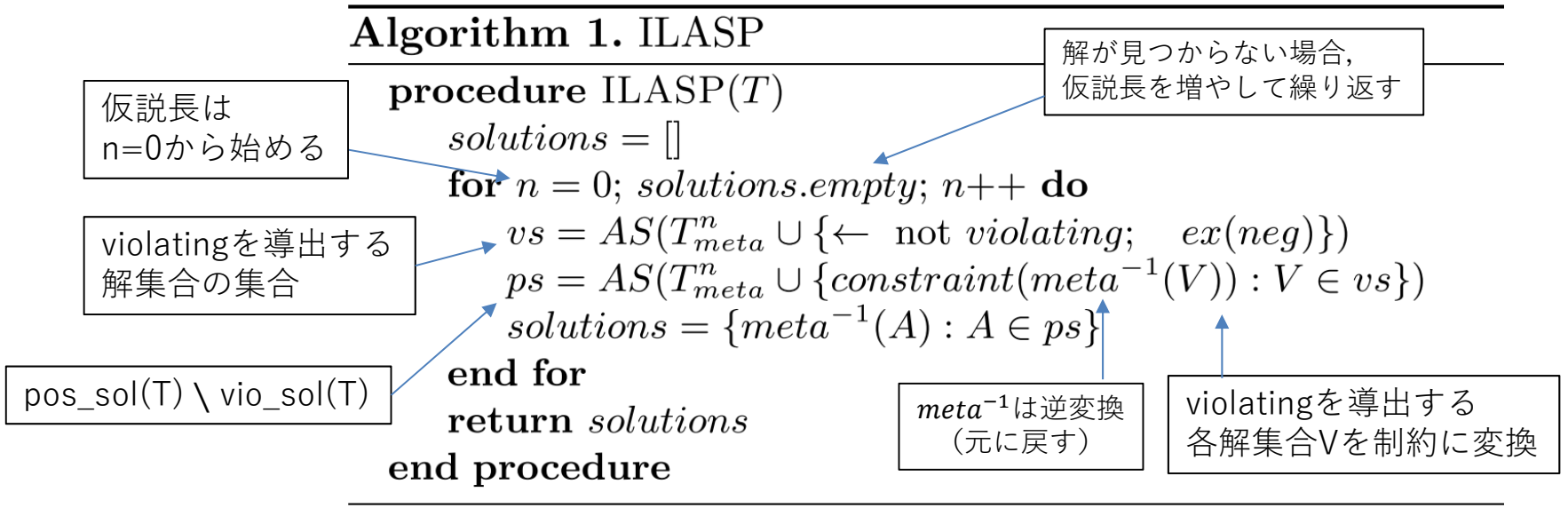
※それぞれの変換の意味・意図・正当性を考えてみよう

アルゴリズム ILASP

- タスクT = < B, S_M, E^+, E^- >に対し，
  Positive Solutionの集合を pos_sol(T) , Violating Solutionの集合をvio_sol (T)と表記
  - 求めるべき仮説H*は， pos_sol(T) \ vio_sol(T) の中で記述長が最小のもの

- まず仮説長nのvio_sol(T)を算出： $T^n_{meta} \cup \{: -not\ violating.,\ ex(neg).\ \}$ の解集合を計算
  - その解集合を制約として， pos_sol(T) \ vio_sol(T)を計算
  - 制約："各"解集合{ active(h^i_1), …, active(h^i_n) } に対して，
    制約 :- active(h^i_1),…,active(h^i_n). を準備
  - T^n_{meta} と制約から得られる解集合がpos_sol(T) \ vio_sol(T)

※T^n_{meta}は，各B, S_M, E^+, E^-の変換の集合（連言）



ILASPシステム

ILASP：Inductive Learning of Answer Set Programs

- http://www.ilasp.com/
- 種々の応用：一般ゲーム / 文法学習 / 強化学習

- 論理設定の拡張
  - 各事例に対する背景知識
  - ノイズの許容（事例に対するpenalty）
  - 弱い制約の学習（選好学習）と順序事例　など

e^{inc}, e^{exc}：**アトム集合の対**
e^{ctx}：**解集合プログラム**

- 各事例に対する背景知識
  - $\langle e^{inc}, e^{exc} \rangle$を$\langle e^{inc}, e^{exc}, e^{ctx} \rangle$に拡張．　e^{ctx}は，弱い制約を含まない解集合プログラム
  - 背景知識Bだけでなく，　e^{ctx}と併せたときの解集合を考える
    - $\forall \langle e^{inc}, e^{exc}, e^{ctx} \rangle \in E^{+},\ \exists A \in AP(B \cup H \cup e^{ctx}) [e^{inc} \subseteq A, e^{exc} \cap A = \{\}]$
    - $\forall \langle e^{inc}, e^{exc}, e^{ctx} \rangle \in E^{-},\ \nexists A \in AP(B \cup H \cup e^{ctx}) [e^{inc} \subseteq A, e^{exc} \cap A = \{\}]$

- ノイズの許容（事例に対するpenalty）
  - 各事例に対して，ペナルティ値を設定
  - 「仮説長 + コスト」を最小化する仮説の導出
    - コスト＝条件を満たさない事例に付与されたペナルティの総和

- ※余力があったら，これらを考慮した場合のmeta変換（ASPへの変換）を考えてみよう

背景知識・事例の与え方

- 背景知識：ASPの形式で与える
- 正事例$\langle e^{inc}, e^{exc}, e^{ctx} \rangle$：#pos($e^{inc}$, e^{exc}, e^{ctx}).
- 負事例$\langle e^{inc}, e^{exc}, e^{ctx} \rangle$：#neg($e^{inc}$, e^{exc}, e^{ctx}).
  - e^{inc}とe^{exc}は，｛要素1, 要素2, …, 要素n｝※各要素をカンマで区切り，｛｝で囲む
  - e^{ctx}は，｛プログラム｝※{ }の中にプログラムを書く

数字1..4を使ったミニ数独の例

```
cell((1..4,1..4)).

block((X, Y), tl) :- cell((X, Y)), X < 3, Y < 3.
block((X, Y), tr) :- cell((X, Y)), X > 2, Y < 3.
block((X, Y), bl) :- cell((X, Y)), X < 3, Y > 2.
block((X, Y), br) :- cell((X, Y)), X > 2, Y > 2.

same_row((X1,Y),(X2,Y)) :- X1 != X2, cell((X1,Y)), cell((X2, Y)).
same_col((X,Y1),(X,Y2)) :- Y1 != Y2, cell((X,Y1)), cell((X, Y2)).
same_block(C1,C2) :- block(C1, B), block(C2, B), C1 != C2.
```

背景知識

```
#pos({
  value((1, 1), 1),
  value((1, 2), 2),
  value((1, 3), 3),
  value((1, 4), 4),
  value((2, 3), 2)
}, {
  value((1, 1), 2),
  value((1, 1), 3),
  value((1, 1), 4)
}, {

}).
```

正事例

```
% This positive examples says that there should be at
% least one answer set of B union H which represents a
% board extending the partial board:
%
%         #-------#-------#
%         | 1 : 2 | 3 : 4 |
%         | - + - | - + - |  such that the first cell
%         |   :   | 2 :   |  does not also contain a
%         #-------#-------#  value other than 1.
%         |   :   |   :   |
%         | - + - | - + - |
%         |   :   |   :   |
%         #-------#-------#
```

```
#neg({ value((1, 1), 1), value((1, 3), 1) },{ }, { }).
#neg({ value((1, 1), 1), value((3, 1), 1) },{ }, { }).
#neg({ value((1, 1), 1), value((2, 2), 1) },{ }, { }).
```

負事例

ハミルトン閉路を持つか否か？
背景知識はグラフそのもの
e^inc, e^exc　は空集合

```
#neg({},{},{node(1..3).
        edge(1,1). edge(1,3). edge(2,2). edge(2,3). edge(3,1).
        }).
#neg({},{},{node(1..3).
        edge(1,1). edge(1,2). edge(3,1). edge(3,2). edge(3,3).
        }).
#neg({},{},{node(1..3).
        edge(1,1). edge(1,2). edge(1,3). edge(2,2). edge(2,3).
        }).
#neg({},{},{node(1..4).
        edge(1,1). edge(1,2). edge(2,1). edge(2,2). edge(2,4). edge(3,2).
        edge(3,3). edge(3,4). edge(4,1). edge(4,4).
        }).
#neg({},{},{node(1..3).
        edge(2,3). edge(3,1). edge(3,2).
        }).

#pos({},{},{node(1..3).
        edge(1,1). edge(1,2). edge(1,3). edge(2,1). edge(3,1). edge(3,2).
        }).
#pos({},{},{node(1..2).
        edge(1,2). edge(2,1). edge(2,2).
        }).
#pos({},{},{node(1..4).
        edge(1,3). edge(1,4). edge(2,1). edge(2,3). edge(3,1). edge(3,2).
        edge(3,3). edge(4,1). edge(4,2).
        }).
#pos({},{},{node(1..3).
        edge(1,1). edge(1,3). edge(2,1). edge(2,2). edge(2,3). edge(3,2).
        }).
#pos({},{},{node(1..4).
        edge(1,2). edge(2,1). edge(2,3). edge(2,4). edge(3,2). edge(3,4).
        edge(4,1). edge(4,2). edge(4,4).
        }).
```

仮説空間・言語バイアスの指定：方法１

- 形式「*Length*　～仮説.」の形式で，候補となる仮説を列挙する.
  - 1 ~ :- edge(V0, V0).
  - 1 ~ :- edge(V0, V1).
  - 1 ~ :- in(1,V0).
  - 1 ~ :- in(V0,V1).
  - 1 ~ :- reach(V0).
  - 2 ~ reach(V0) :- in(1,V0).
  - 2 ~ reach(V0) :- in(V0,V1).
  - 2 ~ reach(V1) :- in(V0,V1).
  - 3 ~ 0 {in(V0,V0) } 1 :- edge(V0, V0).
  - 3 ~ 0 {in(V0,V0) } 1 :- edge(V0, V1).
  - 3 ~ reach(V2) :- in(V0,V1), in(V0,V2).
  - 3 ~ reach(V2) :- in(V0,V1), in(V1,V2).

仮説空間・言語バイアスの指定：方法２

- 頭部リテラル： #mode**h**(*literal*(arg1, …, argn)).
- 本体部リテラル： #mode**b**(*literal*(arg1, …, argn)).
- 集約頭部リテラル（チョイスルール） ：#mode**ha**(*literal*(arg1, …, argn)).
  - #minhl(Min)：頭部リテラルの最小数をMinにする
  - #maxhl(Max)：頭部リテラルの最大数をMaxにする
- 一貫性制約の本体部リテラル：#mode**c**(*literal*(arg1, …, argn)).
- argN = var(*type*) | const(*type*)

- ※モード宣言は，第一引数にrecall数（各ルールにおける出現数上限）を取ることもある
- ※モード宣言は，第三引数にオプションを取ることもある
  - オプションの例1： (positive)を指定すると，not literal(..) は生成されない
  - オプションの例2：引数2の述語に対したのみ有効．(anti_reflexive)を指定すると，同じ引数を取ることはなくなる

- type宣言(constant宣言)：#constant(type, 値).

- 頭部に複数の変数が現れることを許さない．#disallow_multiple_head_variables.

数字1..4を使ったミニ数独の例

```
#modeha(value(var(cell),const(number))).

#modeb(2,value(var(cell),var(val))).
#modeb(1,same_row(var(cell), var(cell)), (anti_reflexive)).
#modeb(1,same_block(var(cell), var(cell)), (anti_reflexive)).
#modeb(1,same_col(var(cell), var(cell)), (anti_reflexive)).
#modeb(1,cell(var(cell))).

#constant(number, 1).
#constant(number, 2).
#constant(number, 3).
#constant(number, 4).

#minhl(4).
#maxhl(4).
#disallow_multiple_head_variables.
```

得られる仮説　※本体部の ";"は，論理的には"," の意味

```
 :- same_block(V1,V2); value(V1,V3); value(V2,V3).
 :- same_row(V1,V2); value(V1,V3); value(V2,V3).
 :- same_col(V1,V2); value(V1,V3); value(V2,V3).
1 {value(V1,1);value(V1,2);value(V1,3);value(V1,4) } 1 :- same_row(V2,V1).
```

ILASPの実行

- 基本： % ilasp --version=4 ファイル1, .., ファイルn
  - 実行例１： % ilasp --version=4 sudoku.las
  - 実行例２： % ilasp --version=4 hamiltonX.las hamiltonY.las

- 仮説空間の確認：%ilasp –sファイル1, .., ファイルn
  - 仮説空間が表示されます．流れてしまうので， less などで受けましょう
  - 学習前に，(wc コマンドで) 仮説空間の大きさを確認しよう（大きいと終わりません！）
- 実行時オプション
  - 詳細は ilasp --helpで確認（こちらも流れてしまうので，lessなどで受けましょう）
  - -nc ：探索空間からの制約の排除
  - -na : 探索空間からのチョイスの排除
  - -ml=[integer]：各節の本体部リテラル数上限（default 3）
  - --max-rule-length=[integer]：各節に含まれるリテラル数の上限（default 5）

  - 実行例３： %ilasp --version=4 –nc animalB.las

- 例題を確認してみよう
  - ハミルトン閉路：hamiltonX.las, hamiltonY.las
  - 数独：sudoku.las
  - 動物分類：animalB.las

参考：動物分類の元データ

	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