論理と計算

第８回
論理プログラム：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

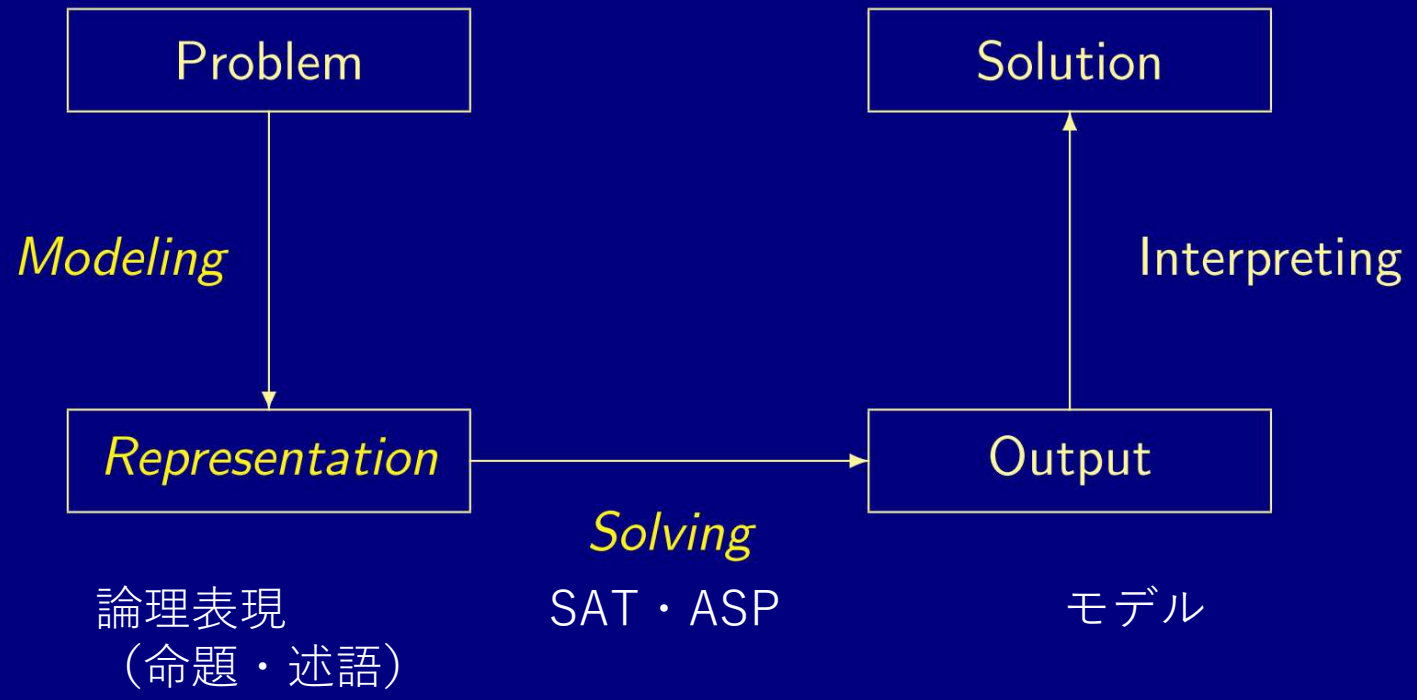
目次：今回の授業の内容

- （復習）論理による問題解決
- （復習）述語論理の節形式
- 述語論理の基礎化
- 論理プログラムのクラス
  - 確定論理プログラムと最小モデル
  - 標準論理プログラムと安定モデル
  - 選言論理プログラムと安定モデル
  - 拡張論理プログラムと解集合
  - その他の論理プログラム

論理に基づく問題解決

Declarative problem solving

"What is the problem?"



Potassco

https://github.com/potassco-asp-course/course/releases/download/v1.1.0/motivation.pdf より

命題論理の解釈・モデル

- 解釈（Interpretation）
  - 命題文 α に現れる原子文（命題記号）を$\alpha_1, \alpha_2 \cdots \alpha_n$とする.
  - 各原子文α_iへの真理値(true,false)の割り当てを α の解釈と呼ぶ
- モデル（model）
  - 命題文 α の真理値を真とする解釈

- 問題解決とモデル
  - 論理式：対象とする問題の表現
    - 既知の事実やルール，制約を論理式で表現する
    - 知りたいこと（問題の解）も命題で表現する
  - モデル：既知の事実やルール，制約を満たす「真となる命題の集合」
    - 論理式を成立させるために，真（または偽）とする必要のある命題が分かる
    - 「知りたいこと」（問題の解）に相当する命題の真・偽が分かる（＝問題が解決できた！）

- SATソルバー
  - 命題論理（CNF）で表現された論理式を対象としたモデル発見器

- 述語論理でも同じようなことを行いたい→論理プログラム（解集合プログラミング）

述語論理から命題論理へ

述語論理の基礎化

- 基礎：「変数を含まない」
- 基礎化：変数を含む論理式を，変数を含まない論理式へ変換すること

- 基礎原子文（ground atom）
  - すべての引数が基礎項（変数を含まない項）である原子文（変数を含まない原子文）
  - 命題論理の命題記号に相当

- 節集合の基礎化
  - 節集合：節（リテラルの∨結合）の集合（∧結合）
  - 各節に現れる変数は全称限量されている（存在限量は現れない）
  - 基礎化
    - 各変数を考えられる定数に置き換えた節を作ればよい
      - 「考えられる定数」の集合＝エルブラン領域
    - 全称限量：変数を定数に置き換えた節の∧結合＝集合
  - 基礎節集合：変数を含まない節の集合（∧結合）

エルブラン領域と基礎節集合

- エルブラン領域（Herbrand Universe）：考えられる項の集合
  - 節集合に現れるすべての定数と関数記号から生成される基礎項の全体集合
  - 定数が存在しない場合は，適当に一つの定数aを導入する
  - 関数記号がある場合は，無限の集合になる
  - 例：{ dog(pochi), cat(mike) } のエルブラン領域：{ pochi, mike }
  - 例：{ ¬dog(X) ∨ animal(X) } のエルブラン領域：{ a }
  - 例：{ nat(0), ¬nat(X) ∨ nat(s(X)) } のエルブラン領域：{ 0, s(0), s(s(0)), s(s(s(0))), … }

- 基礎節集合：
  - 節集合に対して，全称限量変数にエルブラン領域中の項を代入して得られる文の全体集合
  - （変数はないので）全称限量子は削除
  - （節を構成する）「基礎原子文を命題記号とする命題論理式」と見做すことが可能

  - 例：{ ¬dog(X) ∨ animal(X) } の基礎節集合：{ ¬dog(a) ∨ animal(a) }
  - 例：{ ¬dog(X) ∨ animal(X), dog(pochi), cat(mike) } の
    基礎節集合：{ ¬dog(pochi) ∨ animal(pochi), ¬dog(mike) ∨ animal(mike),
    dog(pochi), cat(mike) }

エルブラン領域が無限集合の場合基礎節集合も無限集合になる

エルブラン基底とエルブラン解釈

- エルブラン基底（Herbrand base）：考えられる基礎原子式の集合
  - エルブラン領域の要素を引数とする基礎原子式の全体集合
  - 例：{p(X,Y), q(a), r(b) }
    定数：{a, b}，述語記号：{p, q, r }
    エルブラン領域：{ a, b }
    エルブラン基底：{ p(a,a), p(a,b), p(b,a), p(b,b), q(a), q(b), r(a), r(b) }

- エルブラン解釈（Herbrand Interpretation）：エルブラン基底に対する解釈

- 補題：節集合Sの基礎節集合Γ が，命題論理式の集合として充足可能であれば，S は充足可能である（Γ は変数を含まないので，各原子式を命題記号に置き換えれば，Γ は命題論理式集合となる）

- エルブランの定理
  - 任意の節集合Sが充足不能なら，その基礎節集合Γ の有限部分集合の中に，充足不能なものが存在する

述語論理と論理プログラム

※詳細は後述：ざっくりイメージを掴んでください

述語論理から論理プログラムへ

- 節（clause）からルール（rule）へ
  - 述語論理：「節」の集合
  - 論理プログラム：ルールの集合

- ルール：「頭部（ヘッド）←本体部（ボディ）」
  - 頭部，本体部の構成要素は，プログラムのクラスに依存（詳細は後述）
    - 頭部＝ヘッド＝帰結は選言，本体部＝ボディ＝条件は連言
  - 直観的な意味：「本体部が成り立つならば頭部が成り立つ」（詳細は後述）
  - ※一般には，関数フリーで，値域限定条件を満たすことを前提とする

- 「節」（アトムの選言）と「ルール」は別物
  - 古典論理の含意（⇒）と，論理プログラミングの首（←）は，意味が異なる
  - ルールでは「ボディ（条件部）が成り立てば，ヘッド（帰結部）が成り立つ」と考える
    - 明確な方向を考えている

- ルールでは，頭部と本体部に明確な違いがある
  - ルール１：penguin(X); fly(X) ←bird(X)　　：Xが鳥であれば，Xはペンギンか空を飛ぶ
  - ルール２：fly(X) ← bird(X), ¬penguin(X)　：Xがペンギンではない鳥であれば，空を飛ぶ
  - 述語論理では両方とも penguin(X) ∨ fly(X) ∨ ¬bird(X)

後ほど「モデル」を求めてみましょう

※詳細は後述：ざっくりイメージを掴んでください

述語論理から論理プログラムへ

- 論理プログラムでは「二種類の否定」を考える
  - 明示的否定（論理否定）：「ｘ ｘ ｘではない」ことを意味する
  - デフォルトの否定：「ｘ ｘ ｘであるか不明」ことを意味する
    - 例：alive(tom)：「tomは生きている」
      - ¬alive(tom)：tomは生きていない（＝死んでいる）
      - not alive(tom)：tomが生きているか不明（＝生死不明）
- デフォルトの否定を利用すると，曖昧・不確実なことが表現できる
- 論理プログラムの意味論（モデル）

プログラムを満たす
＝各ルールを満たす（真にする）

  - そのプログラムを満たす「モデル」＝「基礎リテラルの集合」（アトムではなくリテラル）
    - 確定プログラム：最小モデル
    - 標準論理プログラム：安定モデル
    - 解集合プログラム：解集合

論理表現した問題の解に相当する

- モデルに含まれないリテラルの真偽：2つの立場

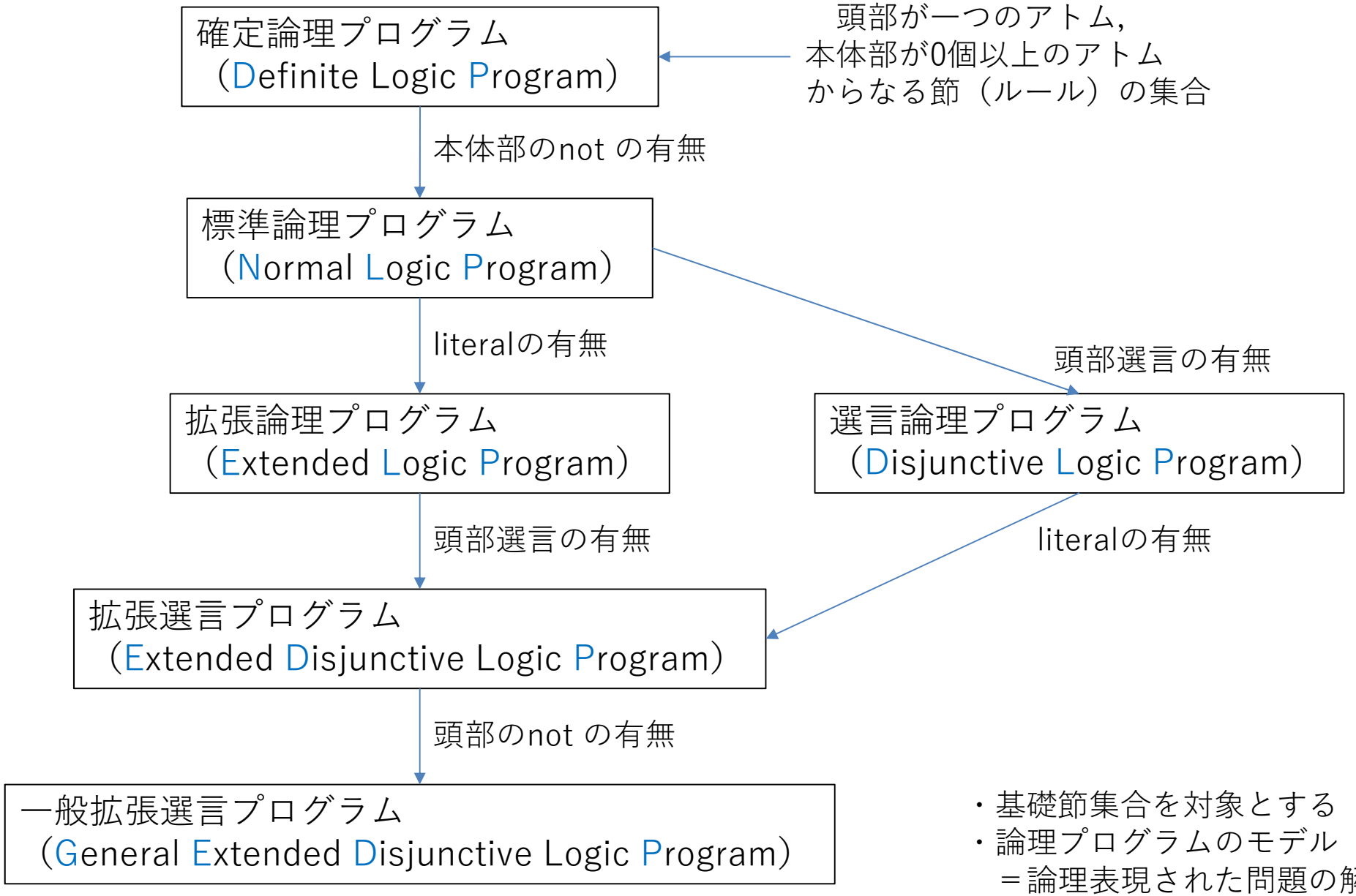
どの様な条件で，
「明示的否定（論理否定）が成り立つと判断するか？」という話です

  - 閉世界仮説（closed world assumption）：偽と解釈する立場
  - 開世界仮説（open world assumption）：真とも偽とも解釈しない立場
    - 「真」または「偽」が明示的に示されない場合は「不明」と考える
      - 示される＝モデルに含まれる
  - P∨¬P：CWAだとトートロジー，OWAだとトートロジーではない

※詳細は後述：ざっくりイメージを掴んでください

本体部のnot : normal
頭部の選言：disjunctive
リテラルの採用：extended

様々な論理プログラムのクラス



- 基礎節集合を対象とする
- 論理プログラムのモデル
  ＝論理表現された問題の解

確定論理プログラム

確定論理プログラム（definite logic program）

- 確定論理プログラム（definite logic program）：確定節の集合
  - 確定節（definite clause）：ちょうど一つの正リテラルを含む節
  - 正リテラルのみから確定節：事実
  - 正負リテラルからなる確定節：ルール $\neg b_1 \vee \cdots \vee \neg b_n \vee h$ → $h \leftarrow (b_1, \cdots, b_n)$
- 確定プログラムの意味論（モデル）

表記：Cn(P)：確定プログラム P の最小モデル

  - 最小エルブランモデル（そのプログラムが成り立つ最小のエルブランモデル）
    - 参考：最小モデルに含まれる基礎アトム集合とSLD反駁（証明したいことの否定を起点とする入力融合法による反駁証明）によって証明される基礎アトムの集合が一致する

モデル： プログラム（ルール）を真にする割り当て＝真となるアトムの集合
最小：集合の包含関係で最小

- Tpオペレータ：確定プログラムPとその基礎例の集合ground(P) および エルブラン基底Bpの任意の部分集合Iに対し，Tpオペレータを以下のように定義する

$$\mathrm{Tp(I)} = \{\mathrm{A} \in Bp \mid \exists C \in ground(P), C = A \leftarrow L_1 \wedge \cdots \wedge L_n,\ \{L_1, \ldots, L_n\} \subseteq I\}$$

- 不動点：条件 $\mathrm{Tp(I)} = \mathrm{I}$ を満たす集合 I をTpオペレータの不動点と呼ぶ．
  - I = { } から始め，不動点が見つかるまで再帰的にTp(I)を計算する

難しく言っているだけ
「事実集合から始め，
ルールを繰り返し
適用していく」

```
I = { }
while( Tp(I) ≠ I ) {
    I = Tp(I)
}
return I
```

- 定理：Characterization Theorem
  - 任意の確定プログラムの最小エルブランモデルと，
    Tpオペレータの不動点は等しい

最小モデル計算の例

プログラム{ p(a).　r(b).　q(X)←p(X).　s(A,B)← q(A),r(B). }
→基礎化　{ p(a).　r(b). q(a)←p(a). q(b)←p(b).
　　s(a,a)← q(a),r(a). s(a,b)← q(a),r(b). s(b,a)← q(b),r(a). s(b,b)← q(b),r(b). }

事実集合
I = { p(a), r(b) }
　　.. I で本体部が（新たに）成立するルール：{q(a)←p(a). }
　　.. 頭部を I に追加する（=本体部が成り立ったのだから頭部が成り立つ）

I = { p(a), r(b),　q(a) }
　　.. I で本体部が（新たに）成立するルール：{s(a,b)←q(a),r(b). }
　　.. 頭部を I に追加する（=本体部が成り立ったのだから頭部が成り立つ）

I = { p(a), r(b), q(a), s(a, b) }
　　.. I で本体部が（新たに）成立するルール：{}
　　.. Iは変化しない → Iが最小モデル

※単なる「ルールの適用」の繰り返し
　　「本体部を満たす（= I の部分集合）なら頭部を I に追加する」の繰り返し

ルールを真にするため，順次「本体部が成り立つルールの頭部アトムをモデルに追加」
　　→（集合の包含関係の意味で）最小な集合（最小モデル）が得られる

確定論理プログラムの欠点と拡張

- 確定情報しか記述することができない
  - 実世界における不確定・不完全な知識を表現するには限界がある
  - →表現能力の拡張
    - 否定（negation）：明示的否定（論理否定），デフォルトの否定
    - 選言（disjunction）：選言論理プログラム（disjunctive logic program）

- 確定プログラムの拡張１：Prolog
  - 表現言語：確定論理プログラム＋NAF
    - NAF : Negation as Failure, 証明が失敗した場合に否定が成り立つと考える立場
      - NAF ≒ デフォルトの否定
  - トップダウン，証明手続き（入力融合法）が基礎 → 解はyes/no または 変数への代入

- 確定プログラムの拡張２：解集合プログラミング（answer set programming）
  - 表現言語：一般拡張選言プログラム（general extended disjunctive program）
    - 一般：ヘッドにデフォルトの否定＋ボディにデフォルトの否定（標準）
    - 拡張：（アトムではなく）リテラルが基本要素
    - 選言：ヘッドにdisjunction

    ※一般には，関数記号なしで，値域限定条件を満たすことを前提とする
  - 要は，二種類の否定＋選言を扱うことのできる論理プログラム
  - ボトムアップ計算が基礎 → 解は（複数の）「モデル」（＝基礎リテラルの集合）

標準論理プログラム

標準論理プログラム（Normal Logic Program）

- 以下の形式をした「ルール」の集合を標準論理プログラムと呼ぶ
  - $A \leftarrow A_1, \ldots, A_m,\ not\ A_{m+1},\ \ldots,\ not\ A_n \quad (A, A_1, \ldots, A_n$はアトム$)$
  - 確定論理プログラムの本体部にnot を許す（not A は，「Aが成り立たない」の意味）
- ルールが表す直感的な意味
  - 「A1～Amが成り立ち，Am+1～Anが成り立たないならば，Aが成り立つ」
- 成り立つ＝「モデルに含まれる」
- 閉世界仮説：
  - モデルに含まれるアトムは成り立つ
  - モデルに含まれないアトムは成り立たない
- モデル： プログラム（ルール集合）を真にする割り当て＝真となるアトムの集合
  - ルールが真（ケース１）：本体部が成り立つときに頭部が成り立つ
  - ルールが真（ケース２）：本体部が成り立たない
  - すべてのルールを真にするアトムの集合のうち，極小（最小）のものを見つける

安定モデル

- 以下の形式をした「ルール」の集合を標準論理プログラムと呼ぶ
  - $A \leftarrow A_1, \ldots, A_m,\ not\ A_{m+1},\ \ldots,\ not\ A_n \quad (A, A_1, \ldots, A_n$はアトム$)$
- 安定モデル導出の直感的な考え方
  - モデル候補（アトム集合）を仮定する．仮定したモデルにおけるルール本体部の成否を考え，
    "ルール自体を削除" or "not Aを削除" することで，確定論理プログラム変換する
    仮定（モデル候補） と 変換後の確定論理プログラムのモデル の一致を確認する
- 意味論：安定モデル（stable model）
  - 標準論理プログラムPと基礎アトムの集合Sに対し，Pを基礎化した上で，
    以下の操作を行うことで得られる確定論理プログラムをP^Sとする
    - 条件：$\{A_{m+1}, \cdots, A_n\} \cap S \neq \{\ \}$を満たすルールをPから削除する
      - すなわちデフォルトの否定が付与されたアトムがモデルに含まれるルールを削除する
    - 残ったルール中から，$not\ A_{m+1}, \cdots, not\ A_n$を削除する
  - S と確定論理プログラムP^Sの最小モデル$Cn(P^S)$とが一致するとき，SをPの安定モデルと呼ぶ
    - P^Sを，PのSに関するreductと呼ぶ

  - 難しく書いてありますが．．．．
    - 「モデルにA_xが含まれる場合，not A_x を含むルールは本体部を満たさない」ので削除
      - ※本体部を満たさない＝ルールとしては真となるに注意
    - 残ったルールの本体部に現れる not A_x は，モデルに含まれないという条件を満たしたので削除
    - notがなくなったので，確定論理プログラムとしてモデルを計算．一致を確認

安定モデル計算の例

a.
c ← *not* b, *not* d.
d ← a, *not* c.

- エルブラン基底： {a, b, c, d }
  の部分集合を一つずつ確認する

- 実際には（エルブラン基底ではなく）ルールPの頭部アトム集合を対象とすればよい
  - $S \subseteq Cn(P^S) \subseteq head(P^S) \subseteq head(P)$が成立
  - 今回，b はルール頭部に現れないので無視
  - ※もっと効率の良い方法も提案されている
  - 右図より安定モデルは {a,c} と {a,d}の2つ

S	P^S	$Cn(P^S)$	S == $Cn(P^S)$
{ }	a. c ← ~~*not* b, *not* d~~ d ← a~~, *not* c.~~	{ a, c, d }	No
{a}	a. c ← ~~*not* b, *not* d~~ d ← a~~, *not* c.~~	{ a, c, d }	No
{c}	a. c ← ~~*not* b, *not* d~~ ~~d ← a, *not* c.~~	{ a, c }	No
{d}	a. ~~c ← *not* b, *not* d~~ d ← a~~, *not* c.~~	{ a, d }	No
{a,c}	a. c ← ~~*not* b, *not* d~~ ~~d ← a, *not* c.~~	{ a, c }	Yes
{a,d}	a. ~~c ← *not* b, *not* d~~ d ← a~~, *not* c.~~	{ a, d }	Yes
{c,d}	a. ~~c ← *not* b, *not* d~~ ~~d ← a, *not* c.~~	{ a }	No
{a, c, d}	a. ~~c ← *not* b, *not* d~~ ~~d ← a, *not* c.~~	{ a }	No

安定モデル計算の例

安定モデルは「プログラムを成立させる解」
安定モデルの数は，プログラムに依存する

$$P = \left\{ \begin{array}{l} p \leftarrow p \\ q \leftarrow not\, p \end{array} \right\}$$

S	P^S	$Cn(P^S)$	$S = Cn(P^S)$
$\{\ \ \}$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	No
$\{p\ \ \}$	$p \leftarrow p$	$\{\ \ \}$	No
$\{\ \ q\}$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	Yes
$\{p, q\}$	$p \leftarrow p$	$\{\}$	No

※not を含まない標準論理プログラム，
すなわち確定論理プログラムは
必ず（唯一の）最小モデルを持ち，
最小モデルと安定モデルは一致する

∵ reduct(P^S) = P より Cn(P^S) =Cn(P)

$$P = \left\{ \begin{array}{l} p \leftarrow not\, q \\ q \leftarrow not\, p \end{array} \right\}$$ 2 者択一

S	P^S	$Cn(P^S)$	$S = Cn(P^S)$
$\{\ \ \}$	$p \leftarrow$ $q \leftarrow$	$\{p, q\}$	No
$\{p\ \ \}$	$p \leftarrow$	$\{p\}$	Yes
$\{\ \ q\}$	$q \leftarrow$	$\{q\}$	Yes
$\{p, q\}$		$\{\}$	No

$$P = \{\ p \leftarrow not\, p\ \}$$ 一貫性制約

S	P^S	$Cn(P^S)$	$S = Cn(P^S)$
$\{\ \ \}$	$p \leftarrow$	$\{p\}$	No
$\{p\ \ \}$		$\{\ \ \}$	No

その他の論理プログラム

NLPとの差
アトム →リテラル

拡張論理プログラム（Extended Logic Program）

- 以下の形式をした「ルール」の集合を拡張選言プログラムと呼ぶ
  - $L \leftarrow L_1, \ldots, L_m,\ not\ L_{m+1},\ \ldots,\ not\ L_n \quad (L, L_1, \ldots, L_n$はリテラル$)$
  - リテラル：アトム A とその否定 ¬A（Aの明示的否定・論理否定）

- ルールの直感的解釈
  - 「L_1～L_mが成り立ち，L_{m+1}～L_nが成り立たなければ，Lが成り立つ」

- 2種類の否定
  - 明示的否定（論理否定）：直感的には「x x x ではない」ことを意味する
  - デフォルトの否定：直感的には「x x x であるか不明」ことを意味する
    - 例：alive(tom)：「tomは生きている」
      - ¬alive(tom)：tomは生きていない（＝死んでいる）
      - not alive(tom)：tomが生きているか不明（＝生死不明）
  - 併用の例： P ← not ¬P.（¬Pが成り立たなければ，Pが成り立つ＝デフォルトの成立）
  - 併用の例：¬P ← not P.（Pが成り立たなければ，¬Pが成り立つ＝閉世界仮説）

- 開世界仮説の採用
  - リテラルが成立＝モデルに含まれる / リテラルが不成立＝モデルに含まれない
  - ¬Aの成否：モデルに¬Aが含まれる場合に成立
  - not Bの成否：モデルにBが含まれない場合に成立
  - not ¬Cの成否：モデルに¬Cが含まれない場合に成立

NLPとの差
アトム →リテラル

拡張論理プログラム（Extended Logic Program）

- 以下の形式をした「ルール」の集合を拡張選言プログラムと呼ぶ
  - $L \leftarrow L_1, \ldots, L_m,\ not\ L_{m+1},\ \ldots,\ not\ L_n \quad (L, L_1, \ldots, L_n$はリテラル$)$
- 解集合導出の直感的な考え方
  - 各アトムAに対して新たなアトム¬Aというが追加されたと考えることで
    拡張論理プログラムを標準論理プログラムに還元
    - ただし，¬AとAを同時に含む場合は排除する
- 意味論：解集合（answer set）
  - 解集合＝拡張論理プログラム版の安定モデル
  - 拡張論理プログラムPと基礎アトムとその否定の集合Sに対し，Pを基礎化した上で，
    以下の操作を行うことで得られる（リテラルをアトムと見做した）確定論理プログラムをP^Sとする
    - 条件：$\{L_{m+1}, \cdots, L_n\} \cap S \neq \{\}$を満たすルールをPから削除する
    - 残ったルール中から，$not\ L_{m+1}, \cdots, not\ L_n$を削除する
  - S とP^Sの最小モデルCn(P^S)とが一致するとき，SをPの解集合と呼ぶ
  - ただし，SがアトムAとその否定¬Aを同時に含む場合，S==Litとする（でなければならない）
    - Litはプログラム中に現れる基礎アトムとその否定の全体集合
    - 深く考えなくてOKです．（Aと¬Aを含むという）矛盾を含む集合を排除する

選言論理プログラム（Disjunctive Logic Program）

NLPとの差
ヘッドを Disjunction

- 以下の形式をした「ルール」の集合を選言論理プログラムと呼ぶ
  - $A_1; \ldots; A_l \leftarrow A_{l+1}, \ldots, A_m,\ not\ A_{m+1},\ \ldots,\ not\ A_n \quad (A_1, \ldots, A_n$はアトム$)$
- ルールの直感的解釈
  - 「A_{l+1}～A_mが成り立ち，A_{m+1}～A_nが成り立たなければ，
    A_1～A_lのいずれかが成り立つ」

本体部が成り立つときに，少なくとも一つの頭部アトムが成立

P;¬Pは，古典論理なら恒真なので無視できる．
選言論理プログラムでは，
P または¬Pがモデルに含まれることを要請

- 意味論：安定モデル
- 選言論理プログラムの解集合（notなしの場合）
  - （基礎化済みの）選言論理プログラムPと基礎リテラルS の集合に対し，Sが条件
    1. $\{L_{l+1}, \cdots, L_m\} \subseteq S$ ならば$\{L_1, \cdots, L_l\} \cap S \neq \{\}$

  を満たす（極小の）集合であるとき，SをPの安定モデルと呼ぶ

各ルールに対し
Sにおいて，本体部が成り立つときに，
頭部が成り立つことを確認している

選言があるので確定節になりません．
複数の極小モデルを持つ可能性があります．
モデル計算も複雑になります

- 選言論理プログラムの安定モデル（notありの場合）
  - 選言論理プログラムPと基礎アトムS の集合に対し，Pを基礎化した上で，
    以下の操作を行うことで得られる選言論理プログラムをP^Sとする
    - 条件：$\{L_{m+1}, \cdots, L_n\} \cap S \neq \{\}$を満たすルールをPから削除する
    - 残ったルール中から，$not\ L_{m+1}, \cdots, not\ L_n$を削除する
  - Sと（notなしの）選言論理プログラムP^Sの安定モデルとが一致するとき，SをPの安定モデルと呼ぶ

解集合・安定モデル

- ルールでは，頭部と本体部に明確な違いがある
- 述語論理の　$\mathrm{penguin}(X) \vee \mathrm{fly}(X) \vee \neg \mathrm{bird}(X)$

- ファクト：{ bird(a)}
- ルール１：penguin(X); fly(X) ←bird(X)　　：鳥は，ペンギンか空を飛ぶ
  - ファクト＋ルール１のモデル
  - { bird(a) fly(a) } { bird(a) penguin(a) }

- ルール２：fly(X) ← bird(X)，¬penguin(X)　　：ペンギンではない鳥は，空を飛ぶ
  - ファクト＋ルール２のモデル
  - { bird(a) }　　∵「aがペンギンでない」は記述されていない

- ルール３：fly(X) ← bird(X)，*not* penguin(X)　：ペンギンと確認できない鳥は，空を飛ぶ
  - ファクト＋ルール３のモデル
  - { bird(a) fly(a) }　∵「aがペンギンか確認できない」

拡張選言プログラム（Extended Disjunctive Program）

NLPとの差
アトム をリテラルに
ヘッドを Disjunction

- 以下の形式をした「ルール」の集合を拡張選言プログラムと呼ぶ
  - $L_1; \ldots; L_l \leftarrow L_{l+1}, \ldots, L_m, \ not\ L_{m+1}, \ \ldots\ , \ not\ L_n \quad (L_1, \ldots, L_n$はリテラル$)$

- ルールの直感的解釈
  - 「L_{l+1}～L_mが成り立ち，L_{m+1}～L_nが成り立たなければ，
    L_1～L_lのいずれかが成り立つ」

リテラルを単位とする → 開世界仮説
「成り立つ」＝「モデルに含まれる」
「成り立たない」＝「モデルに含まれない」

- 意味論：解集合（answer set）
- 選言論理プログラムの解集合（notなしの場合）
  - （基礎化済みの）選言論理プログラムPと基礎リテラルS の集合に対し，Sが条件
    1. $\{L_{l+1}, \cdots, L_m\} \subseteq S$ ならば$\{L_1, \cdots, L_l\} \cap S \neq \{\ \}$
    2. SがLと¬Lを同時に含むならS = Lit

  を満たす（極小の）集合であるとき，SをPの解集合と呼ぶ

条件1. は，各ルールに対し
Sにおいて，本体部が成り立つときに，
頭部が成り立つことを確認している

条件2. におけるLitは全基礎リテラルの集合
ただし，矛盾する集合を含んでいるので，
実質的な意味をなさない．

- 拡張選言プログラムの解集合（notありの場合）
  - 拡張選言プログラムPと基礎アトムS の集合に対し，Pを基礎化した上で，
    以下の操作を行うことで得られる選言論理プログラムをP^Sとする
    - 条件：$\{L_{m+1}, \cdots, L_n\} \cap S \neq \{\ \}$を満たすルールをPから削除する
    - 残ったルール中から，$not\ L_{m+1}, \cdots, not\ L_n$を削除する
  - Sと拡張論理プログラムP^Sの解集合とが一致するとき，S を Pの解集合と呼ぶ

標準論理プログラム
（安定モデル）の
場合と同じ

一般拡張選言プログラム

EDPとの差
ヘッドにデフォルトの否定

- $L_1; \ldots; L_k; not\ L_{k+1}; \ldots; not\ L_l \leftarrow L_{l+1}, \ldots, L_m,\ not\ L_{m+1},\ \ldots,\ not\ L_n$
- 直感的な意味
  - 本体部の$L_{l+1}, \ldots, L_m$がすべて成立ち，$L_{m+1}, \ldots, L_n$がそれぞれ成り立たないならば，
  - 頭部の$L_1, \ldots, L_k$のどれかが成り立つか，または$L_{k+1}, \ldots, L_l$のどれかが成り立たない
- 意味論：解集合（answer set）
  - 一般拡張選言プログラムPと基礎アトムS の集合に対し，Pを基礎化した上で，
    以下の操作を行うことで得られる選言論理プログラムをP^Sとする
    - 条件1：$\{L_{m+1}, \cdots, L_n\} \cap S \neq \{\}$　（＝ボディのnotの内，Sに含まれるものがある）
      または
    - 条件2：$\{L_{k+1}, \cdots, L_l\} \not\subseteq S$　（＝ヘッドのnotの内，Sに含まれないものがある）
      を満たすルールをPから削除する
    - 残ったルール中から，$not\ L_{k+1}, \cdots, not\ L_l$ と $not\ L_{m+1}, \cdots, not\ L_n$を削除する
  - Sと拡張論理プログラムP^Sの解集合とが一致するとき，S を Pの解集合と呼ぶ
- 条件1：ボディの条件を満たさないので無視する
- 条件２：ヘッドの条件を満たしたので無視する
  - ボディの条件を満たさない場合は，使われないので無視できる
  - ボディの条件を満たす場合でも，ヘッドの条件を満たしている（not 部の一つ以上が，モデルに含まれないのでヘッドが成り立つ）ので無視できる

解集合プログラミングシステム clingo

解集合プログラミングシステム clingo

- 表記
  - " ; "　　選言はセミコロン
  - " , "　　連言はカンマ
  - " :- "　　←はコロンマイナス（メダカマークですね）
  - " not "　デフォルトの否定はnot
  - " - "　　論理否定（負リテラル）はマイナス

  - 注意１：ボディが空の場合は，:- は記述しない.
  - 注意２：各ルールは，"." で終わる.

ルール：{ p ; not p.　}
clingo：　p ; not p .

ルール：{ p. ¬ p }
clingo：　p.
　　-p.

ルール：{ p ← not q.
　　q ← not p. }
clingo:　p :- not q.
　　q :- not p.

- 使い方　$ clingo　0　入力ファイル
  - clasp と同じ使い方 (0 はすべてのモデルを表示するためのオプション)
  - Clingoは，基礎化器（grounder）gringoを用いてプログラムを基礎化&変換し，claspを用いて解集合を求めている

- やってみよう1 : clingoを用いて資料中の「安定モデルの例・解集合の例」を計算してみよう
- やってみよう2：以下のプログラムの解集合を計算し，だれが飛ぶのか確認しよう

fly(X) ← bird(X), *not* abnormal(X).
abnormal(X) ← penguin(X).
bird(john).
bird(tweety).
penguin(tweety).