問合せ言語の意味論 2 — 否定記述 —

否定記述

- EDB 述語

  item(X): 商品

  broken(X): 壊れた商品（全体のごく一部）

  EDB ファクト: item(a), item(b), item (c), item(d), item(e), broken(e)

  ごく一部の壊れた商品を記述すればよいので, 記憶スペースを食わずに効率的.

- IDB 述語

  canSell(X): 売ることができる商品

  canSell(X) :- item(X) & NOT broken(X)

- 推論の例:

  canSell(a) :- item(a) & NOT broken(a)

  broken(a) が EDB ファクトでないので, NOT broken(a) は真であるとみなし, canSell(a) を推論.

  canSell(e) :- item(e) & NOT broken(e)

  broken(e) が存在するので, NOT broken(e) は真でない. canSell(e) は推論できない.

否定の誤った使用例

- 「X が 任意の Y と結婚していないとき, X は独身である」
  を否定を使って表現したい.
- EDB ファクト
  person(a), person(b), person(c), married(a,b)
  a は b と結婚している. a は独身という結論を導いてはいけない.
- ルール
  bachelor(X) :- person(X) & NOT married(X,Y)
  でうまく表現できているのか?
- 代入 {X := a, Y := c} をルールに適用
  bachelor(a) :- person(a) & NOT married(a,c)
- married(a,c) は EDB ファクトとして存在しないので偽とみなす. bachelor(a)
  を推論してしまう. どこがおかしいのか ?

論理式の等価変形を考えると...

- bachelor(X) :- person(X) & NOT married(X,Y)
  中の X と Y は全称記号により束縛されている
- ∀ X ∀ Y (bachelor(X) :- person(X) & NOT married(X,Y))
  ⇔ ∀ X ∀ Y (bachelor(X) ∨ NOT(person(X) & NOT married(X,Y)))
  ⇔ ∀ X (bachelor(X) ∨ NOT(∃ Y (person(X) & NOT married(X,Y))))
  ⇔ ∀ X (bachelor(X) :- person(X) & ∃ Y NOT married(X,Y))
- 「X が ある人 Y と結婚していないとき X は独身である」
  という意味になってしまっている!
- 「任意の Y」すなわち
  ∀ X (bachelor(X) :- person(X) & ∀ Y NOT married(X,Y))
  を記述するには...

否定の安全な適用

- $\forall$ X (bachelor(X) :- person(X) & $\forall$ Y NOT married(X,Y))
  $\Leftrightarrow$ $\forall$ X (bachelor(X) :- person(X) & $\underline{\text{NOT } \exists \text{ Y married(X,Y)}}$)
- $\forall$ X (spouse(X) :- $\exists$ Y married(X,Y))
  spouse(X) は X に配偶者がいることを意味する
- $\forall$ X (bachelor(X) :- person(X) & NOT spouse(X))
- spouse(X) :- married(X,Y)
  bachelor(X) :- person(X) & NOT spouse(X)
- spouse(a) : - married(a,b)
  bachelor(a) :- person(a) & NOT spouse(a)
  spouse(a) は推論されるが, bachelor(a) は推論できない.
- Tips: NOT が適用される原子式に出現するすべての変数は,
  他の否定されていない原子式に出現するように工夫する.

IDB 述語の否定

- いままでの例では EDB 述語に否定記号を適用
- 有向グラフ中にある一方通行の道を表現
  EDB ファクト: arc(a,b)
  path(X,Y) :- arc(X,Y)
  path(X,Y) :- arc(X,Z) & path(Z,Y)
  one-way(X,Y) :- path(X,Y) & NOT path(Y,X)

エルブランモデル

- M を基礎原子式の集合, A を基礎原子式.
- 原子式およびその否定をリテラルと呼ぶ. A や NOT A はリテラル.
- $A \in M$ のとき, A は M で真. $A \notin M$ のとき, NOT A は M で真
- M が論理プログラム P のエルブランモデル iff
  P のルールの任意の基礎代入例について，本体のリテラルが M で真のとき，頭部の基礎原子式も M の元である.
- 例: {arc(a,b), path(a,b), one-way(a,b) }
  path(a,b) :- arc(a,b)
  one-way(a,b) :- path(a,b) & NOT path(b,a)
- 例: {arc(a,b), path(a,b), path(b,b), one-way(a,b) }
  path(a,b) :- arc(a,b) & path(b,b)
  path(b,b) を追加しても推論可能な IDB ファクトに変化がない
- 例: {arc(a,b), path(a,b), path(b,a), path(a,a) }
  path(a,a) :- arc(a,b) & path(b,a) が新たに推論可能
  one-way(a,b) :- path(a,b) & NOT path(b,a) は推論不可能

複数の極小エルブランモデル

- arc(a,b)

  path(X,Y) :- arc(X,Y)

  path(X,Y) :- arc(X,Z) & path(Z,Y)

  one-way(X,Y) :- path(X,Y) & NOT path(Y,X)

- 下線部がエルブランモデル.

  {arc(a,b), path(a,b)}

  $\subseteq$ <u>{arc(a,b), path(a,b), one-way(a,b)}</u>

  $\subseteq$ <u>{arc(a,b), path(a,b), path(b,b), one-way(a,b) }</u>

- {arc(a,b), path(a,b)}

  $\subseteq$ {arc(a,b), path(a,b), path(b,a)}

  $\subseteq$ <u>{arc(a,b), path(a,b), path(b,a), path(a,a) }</u>

- 複数の極小なモデルが得られる. 理由は？

- $\forall$ X, Y (one-way(X,Y) :- path(X,Y) & NOT path(Y,X))

  $\Leftrightarrow$ $\forall$ X, Y (one-way(X,Y) $\vee$ NOT(path(X,Y) & NOT path(Y,X)))

  $\Leftrightarrow$ $\forall$ X, Y (one-way(X,Y) $\vee$ NOT path(X,Y) $\vee$ path(Y,X))

  $\Leftrightarrow$ $\forall$ X, Y (one-way(X,Y) $\vee$ path(Y,X) :- path(X,Y))

- one-way(a,b) $\vee$ path(b,a) :- path(a,b)

  path(a,b) が真のとき, one-way(a,b) もしくは path(b,a) のどちらかを真として選択する余地がある

- one-way(X,Y) :- path(X,Y) & NOT path(Y,X)

  path(X,Y) のとき path(Y,X) でない限り one-way(X,Y) を推論するという意図が込められている.

- {arc(a,b), path(a,b), one-way(a,b)}

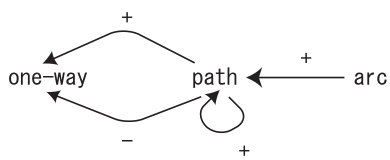
  を意図するモデルとして自然に選択できる意味論をつくれないか？

  層状エルブランモデルの構築へ

依存グラフ (Dependency Graph)

- ノード: 述語
- １つのルールについて, p を頭部の述語, q を本体の述語とするとき, q から p へ有向辺 $p \leftarrow q$ を描く.
- q が否定されていないとき “+” のラベルを,
  否定されているとき “-” のラベルを辺 $p \leftarrow q$ につける.

例　one-way プログラムの依存グラフ



層と層状論理プログラム

層 $stratum$ とは, 次の性質を満たす各述語 p に対応する非負整数 $stratum(p)$.

- １つのルールについて, p を頭部の述語, q を本体の述語とする.
- q が否定されていないならば $stratum(p) \geq stratum(q)$
- q が否定されているならば $stratum(p) > stratum(q)$

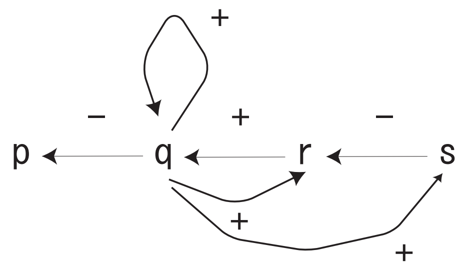
層が存在するとき層状論理プログラムと呼ぶ

層の例 — one-way プログラム —

- one-way = 1, path = 0, arc = 0
- one-way= 2, path=1, arc=0
- 層の入れ方は複数ありうる.

層状論理プログラムでない例

- p :- NOT q,　　q :- q & r,　　r :- q & NOT s,　　s :- q



定理

論理プログラム P に関する次の 2 つは同値

1. "-" のラベルの有向辺を含む閉路が P の依存グラフに存在しない.
2. P は層状.

証明 (スケッチ)

- $1 \Rightarrow 2$:
  依存グラフを極大な強連結成分に分解
  強連結成分内の述語に同一の非負整数値を層として割当てる.
  強連結成分 a から b へ有向辺がある時は, 条件を満たすように層を割当てる

- $2 \Rightarrow 1$:
  背理法. "-" のラベルの辺を含む閉路があると層を構成できない.

層状エルブランモデル

- 層状論理プログラム P を同一の層 i の述語を頭部にもつルールの集合 P_i に分解.
- 下の層から上の層へとボトムアップにモデルを構成.
- P_0 は否定を含まないので通常の最小不動点 $(T_{P_0} \uparrow \omega)$ をモデル (M_0) とする.
- P_i のモデルを計算する場合.
  P_0 から P_{i-1} までで計算したモデルの和集合を M_{i-1}.
- M_{i-1} の元を真と仮定して 1 ステップ推論する写像 $T_{P_i}[M_{i-1}]$
  $$T_{P_i}[M_{i-1}](I) = \{A \mid A \text{ はルールの基礎代入例の頭部で,} \\ \text{本体のすべてのリテラルが } M_{i-1} \text{ または } I \text{ で真}\}$$
- P_i のモデルの計算：$M_i = T_{P_i}[M_{i-1}] \uparrow \omega \cup M_{i-1}$

例

- P_0: path(X,Y) :- arc(X,Y)
  path(X,Y) :- arc(X,Z) & path(Z,Y)
  arc(a,b)
- P_1 : one-way(X,Y) :- path(X,Y) & NOT path(Y,X)
- M_0 = {arc(a,b), path(a,b)}
- M_1 = {one-way(a,b)} $\cup M_0$

定理　層状論理プログラムには複数の層を定義可能であるが, 層の選択に依存しないで層状エルブランモデルは一意に定まる.

(証明：K. R. Apt, H. A. Blair, A. Walker: Towards a Theory of Declarative Knowledge, *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 89-148, 1988.)

デフォールト 推論

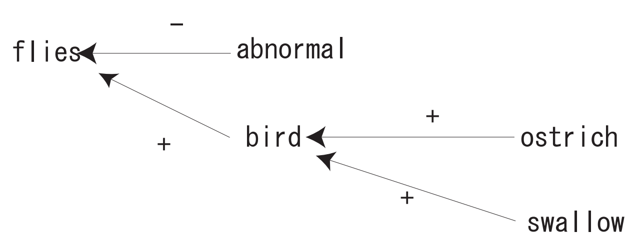
- 例外を除いて、デフォールトでは鳥は飛ぶ.
  例外である飛ばない鳥だけを記述したい

- abnormal(X) :- ostrich(X)
  bird(X) :- ostrich(X)
  ostrich(john)
  bird(X) :- swallow(X)
  swallow(tom)

  flies(X) :- bird(X) & NOT abnormal(X)

- 層 0 abnormal, bird, ostrich, swallow
  層 1 flies

- M_0 = {swallow(tom), bird(tom), ostrich(john), bird(john), abnormal(john)}
  M_1 = {flies(tom)}

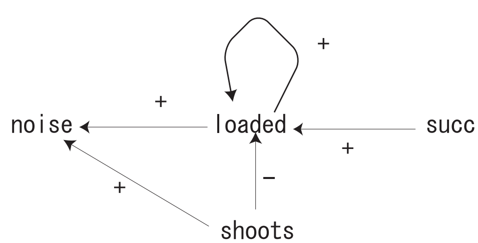


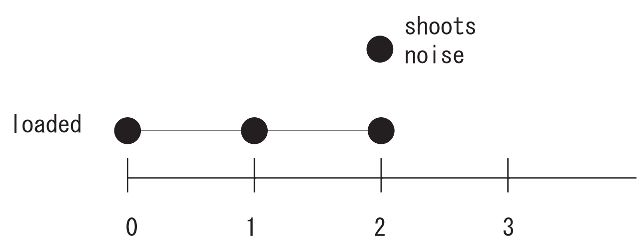
フレーム問題

- 世界の状態は何か動きが起こらない限り, 元の状態を保っている
  動きが起こった事実だけを記述したい

- shoots(2)
  succ(0,1)
  succ(1,2)
  succ(2,3)
  succ(3,4)
  ⋮

  loaded(0)
  loaded(T1) :- loaded(T0) & NOT shoots(T0) & succ(T0,T1)
  noise(T) :- loaded(T) & shoots(T)





- 層 0 shoots, succ
  層 1 loaded, noise

- M_0 = { shoots(2), succ(0,1), succ(1,2), ... }
  M_1 = { loaded(0), loaded(1), loaded(2), noise(2) }