情報論理学
第14回：論理型プログラミング言語

基礎工学部情報科学科　中川 博之

レポート課題

レポート課題: 問13-1 (解答)

- 以下のそれぞれの組み合わせに対してmguを求めよ.
  - (1) p(x, u) と p(g(y), y)
  - (2) p(f(x), y, g(z)) と p(u, u, v)
  - (3) p(a, x, h(g(z))) と p(z, h(y), h(y))

  - (1) p(x, u)とp(g(y), y)
    - x←g(y) で p(g(y), u)とp(g(y), y)
    - u←y で p(g(y), y)とp(g(y), y)
    - よって, mguは θ =<x←g(y), u←y>

レポート課題

レポート課題: 問13-1 (解答)

- (2) p(f(x), y, g(z)) と p(u, u, v)
  - u←f(x) で p(f(x), y, g(z)) と p(f(x), f(x), v)
  - y←f(x) で p(f(x), f(x), g(z)) と p(f(x), f(x), v)
  - v←g(z) で p(f(x), f(x), g(z)) と p(f(x), f(x), g(z))
  - よって, mguは θ =< u←f(x), v←g(z), y←f(x)>

- (3) p(a, x, h(g(z)))と p(z, h(y), h(y))
  - 変数が{x, z}と{y, z}でzが重複
    → 第2項の変数 z を新規変数 u に替える (z←u)
  - p(a, x, h(g(z)))と p(u, h(y), h(y))
  - u←a で p(a, x, h(g(z)))と p(a, h(y), h(y))
  - x←h(y) で p(a, h(y), h(g(z)))と p(a, h(y), h(y))
  - y←g(z) で p(a, h(g(z)), h(g(z)))と p(a, h(g(z)), h(g(z)))
  - よってmguは 第1項θ_1=< x←h(g(z))>
    第2項θ_2=< z←a, y←g(z)>

レポート課題

レポート課題: 問13-2 (解答)

- 先週のレポート課題 問12-1 (2)で得られたスコーレム標準形に対し，mguを用いた導出原理により充足不能であることを示せ.
  - 問12-1 (2)の解答は本日のスライドを参照のこと
  - スコーレム標準形は

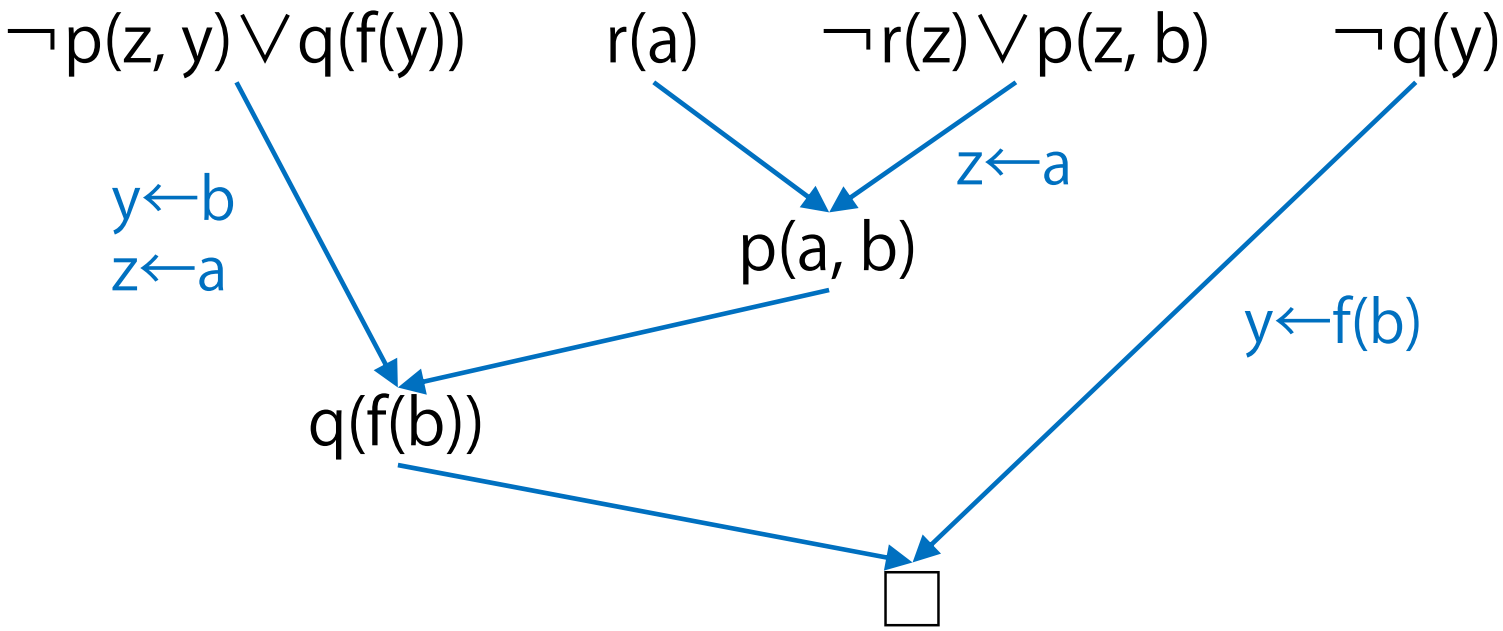
$$\forall z \forall y((\neg p(z, y) \vee q(f(y))) \wedge r(a) \wedge (\neg r(z) \vee p(z, b)) \wedge \neg q(y))$$

  - $C_1 = \neg p(z, y) \vee q(f(y))$
  - $C_2 = r(a)$
  - $C_3 = \neg r(z) \vee p(z, b)$
  - $C_4 = \neg q(y)$

レポート課題

レポート課題: 問13-2 (解答)

- mguによる導出
- $C_1 = \neg p(z, y) \vee q(f(y))$
- $C_2 = r(a)$
- $C_3 = \neg r(z) \vee p(z, b)$
- $C_4 = \neg q(y)$



[復習] Prolog

- Prolog: 論理型プログラミング言語の1つ
  - PROgrammation en LOGique (Programming in Logic)
  - 論理式の恒真性を問う形で記述する

$$\forall x_1 \forall x_2 \ldots (B_{11} \wedge B_{12} \wedge \ldots \wedge B_{1n1} \rightarrow A_1)$$
$$\wedge \ \forall x_1 \forall x_2 \ldots (B_{21} \wedge B_{22} \wedge \ldots \wedge B_{2n2} \rightarrow A_2)$$
$$\ldots$$
$$\wedge \ \forall x_1 \forall x_2 \ldots (B_{m1} \wedge B_{m2} \wedge \ldots \wedge B_{mnm} \rightarrow A_m)$$
$$\rightarrow \exists x_1 \exists x_2 \ldots (C_1 \wedge C_2 \wedge \ldots \wedge C_k)$$

  - ただし, n1, n2, ..., nm $\geqq$ 0, 各B, A, C は原始論理式

[復習] Prolog

- Prologでは下記のように書く
  - 帰結部を先頭に．限定作用素省略．“∧”を”,”に．各行ごとに記述

$A_1 \leftarrow B_{11}, B_{12}, \ldots, B_{1n1}$.
$A_2 \leftarrow B_{21}, B_{22}, \ldots, B_{2n2}$.
…
$A_m \leftarrow B_{m1}, B_{m2}, \ldots, B_{mnm}$.
$\leftarrow C_1, C_2, \ldots, C_k$.

・実際のプログラムでは，
“←” は ”:-” とする場合が多い
・本講義では各行最後の
ピリオドは省略する

- Prologにおける意味付け
  - B_{11} かつ B_{12} かつ … B_{1n1} が成り立つならA_1が成り立つ
  - B_{21} かつ B_{22} かつ … B_{2n2} が成り立つならA_2が成り立つ
  - …
  - これらの状況下でC_1 かつ C_2 かつ… C_k が成り立つか？
    - あるいは，C_1… C_k を真にする変数の割り当てが存在するか？

[復習] Prolog

- $\leftarrow C_1 , C_2 , \ldots , C_k$: ゴール節 (GC), 目標節
- $A_i \leftarrow B_{i1} , B_{i2} , \ldots , B_{ini}$: プログラム節 (PC), 確定節

  特に,
  - $A_i \leftarrow B_{i1} , B_{i2} , \ldots , B_{ini}$ (ni>0): 規則節 (ルール節)
  - $A_i \leftarrow$ (ni=0): 事実節 (ファクト節)

  と呼ばれる

- GC, PCはホーン節 (Horn clause) と呼ばれる
  - 肯定の原子論理式(リテラル)が高々一つの節

Prologでの導出のイメージ

$P \leftarrow Q, Q_1, \ldots, Q_n$

$Q \leftarrow R_1, \ldots, R_n$

から

$P \leftarrow Q_1, \ldots, Q_n, R_1, \ldots, R_n$

を得る

$P \vee \neg Q \vee \neg Q_1 \vee \ldots \vee \neg Q_n$
$Q \vee \neg R_1 \vee \ldots \vee \neg R_n$
が対応

- 最終的に　←　(空節)　が得られると良い
- 空節が得られれば，そのときのゴール節への代入が解
  - 解代入と呼ばれる

例6.1.1

- Q(f(y)) ← R(y)　(1)
- R(g(z)) ←　(2)
- ← Q(x)　(3) (ゴール)

- (3)の変数xにf(y)を代入すると ← Q(f(y))　(4)
- (1)と(4)より ← R(y)　(5)
- (5)の変数yにg(z)を代入して ← R(g(z))　(6)
- (2)と(6) より　空節□ (←) が得られる

- そのときの解代入θは
  θ = <x ← f(y)> ・ <y ← g(z)> = <x ← f(g(z))>

レポート課題

レポート課題: 問14-1

- 以下の論理プログラムにおけるゴールの解を求めよ
  ［問6.1.1］
  - q(f(y), x) ← p(x, y)
  - q(h(y), f(a)) ← p(a, y), q(f(y), a)
  - p(a, b) ←
  - ← q(h(b), x)

例6.1.2

- CFG(文脈自由文法) $G = (\{S, A, B\}, \{a, b, c\}, P, S)$
- P: $S \rightarrow AB$

  $A \rightarrow Aa,\ A \rightarrow a$

  $B \rightarrow bBb,\ B \rightarrow c$

において，記号列 aabcb がGで生成される文であるかどうかをチェックする論理プログラムを考える

例6.1.2

- a(0, 1) ←
- a(1, 2) ←
- b(2, 3) ←
- c(3, 4) ←
- b(4, 5) ←
- S(x, y) ←A(x, z), B(z, y)
- A(x, y) ← A(x, z), a(z, y)
- A(x, y) ← a(x, y)
- B(x, y) ← b(x, u), B(u, v), b(v, y)
- B(x, y) ← c(x, y)

区間[0, 1]にアルファベット a が存在する

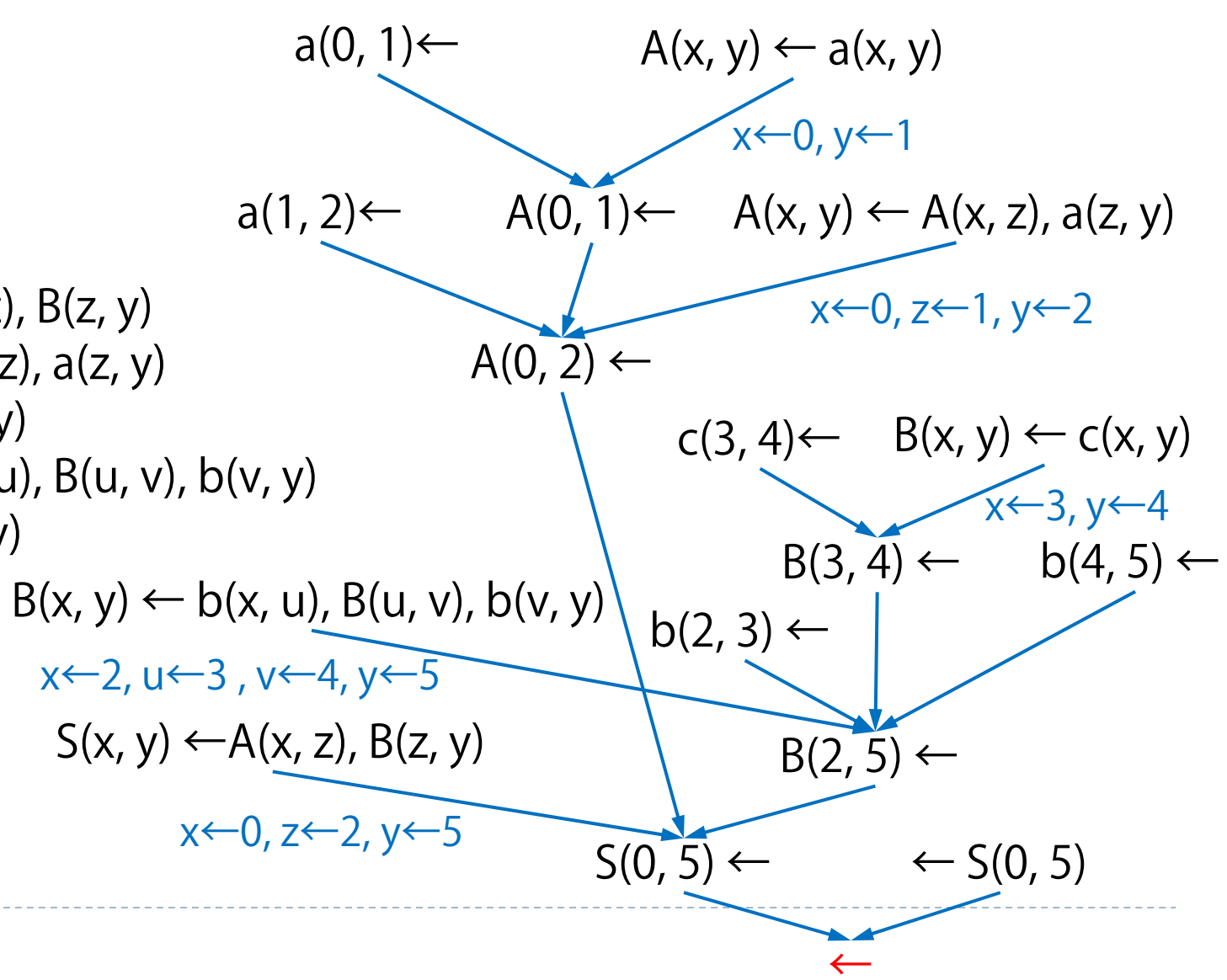
区間[x, z]の文字列がAから生成(導出)でき, かつ, 区間[z, y]の文字列がBから生成できれば,
区間[x, y]の文字列がSから生成できる

- ← S(0, 5)

区間[0, 5]の文字列がSから生成可能？

例6.1.2

- a(0, 1) ←
- a(1, 2) ←
- b(2, 3) ←
- c(3, 4) ←
- b(4, 5) ←
- S(x, y) ←A(x, z), B(z, y)
- A(x, y) ← A(x, z), a(z, y)
- A(x, y) ← a(x, y)
- B(x, y) ← b(x, u), B(u, v), b(v, y)
- B(x, y) ← c(x, y)
- ← S(0, 5)



SLD導出

- 反駁の実行は, 選択する節により選択肢がある
  - 選択する節により, 実行効率が大きく異なる場合もある
- では, どのように節を選択するか？
  - どのような戦略で導出を進めるか？
- SLD導出
  - SL resolution (=Linear resolution with selection function) for Define clause
    - その後, Selective Linear resolution for Definite clauseが略となった
  - ゴールと確定節の間で導出することを繰り返す戦略

SLD導出の自由度

- SLD導出には自由度がある

  1. ゴールのどのリテラルを次の導出(反駁)に用いるか
  2. どの確定節を導出の相手とするか
  3. 導出時にどの単一化代入を用いるか

- これらを定めてSLD導出の手続きが確定

- [定理] SLD導出は健全であり完全
  - 健全: 導き出した解は正しい(解を代入すると恒真式になる)
  - 完全: 恒真のものについては解を導き出すことができる
    - 完全性については, 上手くSLD導出を決定する必要がある

PrologにおけるSLD導出

- Prologでは次のようなSLD導出を用いる

1. ゴールのどのリテラルを次の導出(反駁)に用いるか
   → 最左リテラルを選択

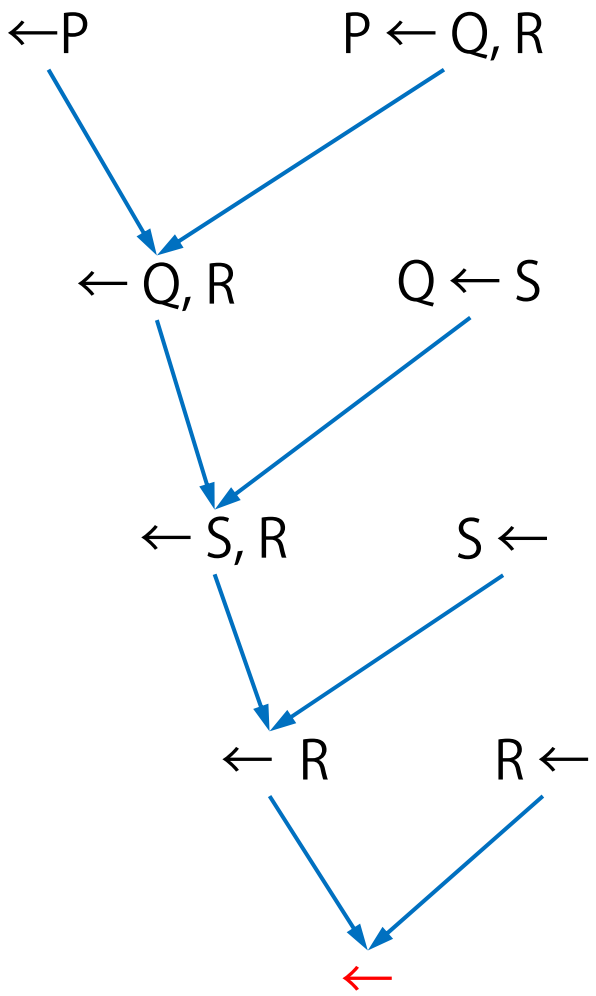
2. どの確定節を導出の相手とするか
   → プログラムに記述された順

3. 導出時にどの単一化代入を用いるか
   → mguを利用

- 導出結果は最左リテラルとして挿入する
- 失敗時にはバックトラックする
  - 直近の適用可能なポイントまで戻る

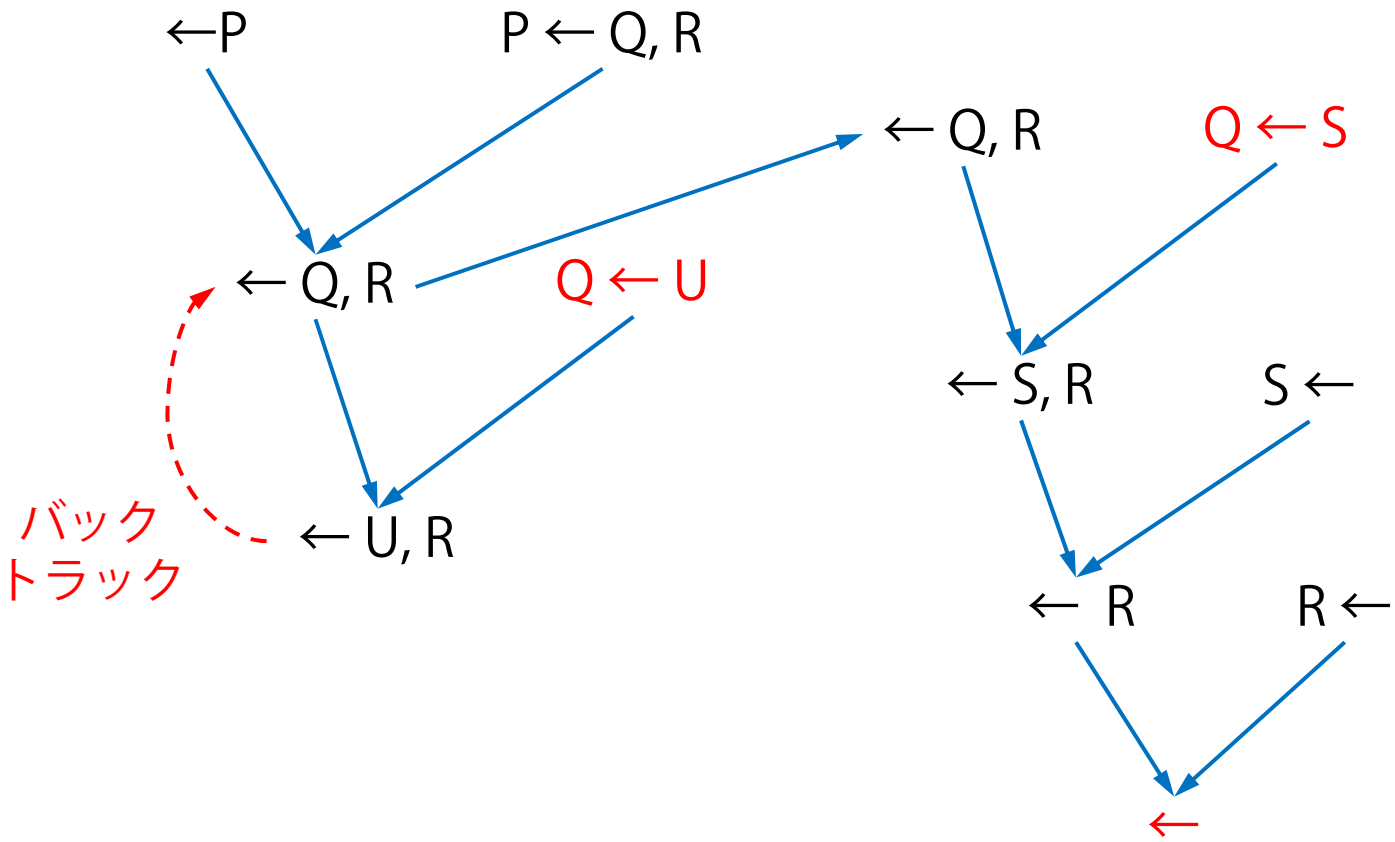
Prologの実行例

- (1) P ← Q, R
- (2) Q ← S
- (3) Q ← U
- (4) S ←
- (5) R ←
- (goal) ←P



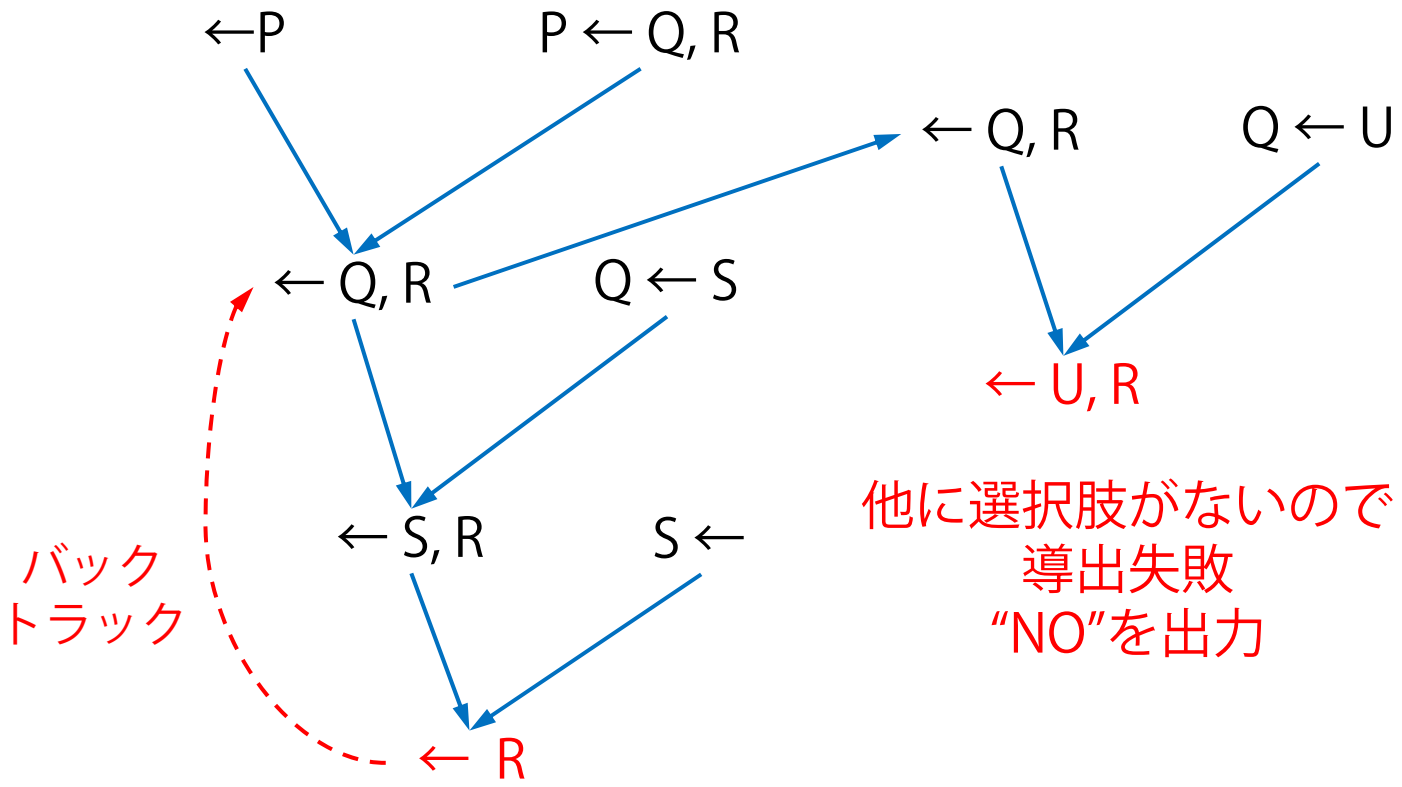
Prologの実行例

- (1) P ← Q, R
- (2) Q ← U
- (3) Q ← S
- (4) S ←
- (5) R ←
- (goal) ←P



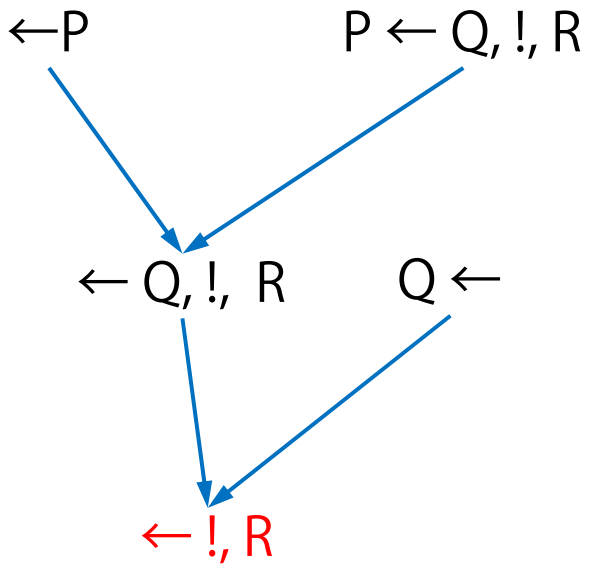
Prologの実行例

- (1) P ← Q, R
- (2) Q ← S
- (3) Q ← U
- (4) S ←
- (goal) ←P



Prologその他

- カット
  - 効率化のためにバックトラックを制御する演算子
- 例:
  - P ← Q, !, R
  - P ← S
  - Q ←
  - S ←
  - ← P



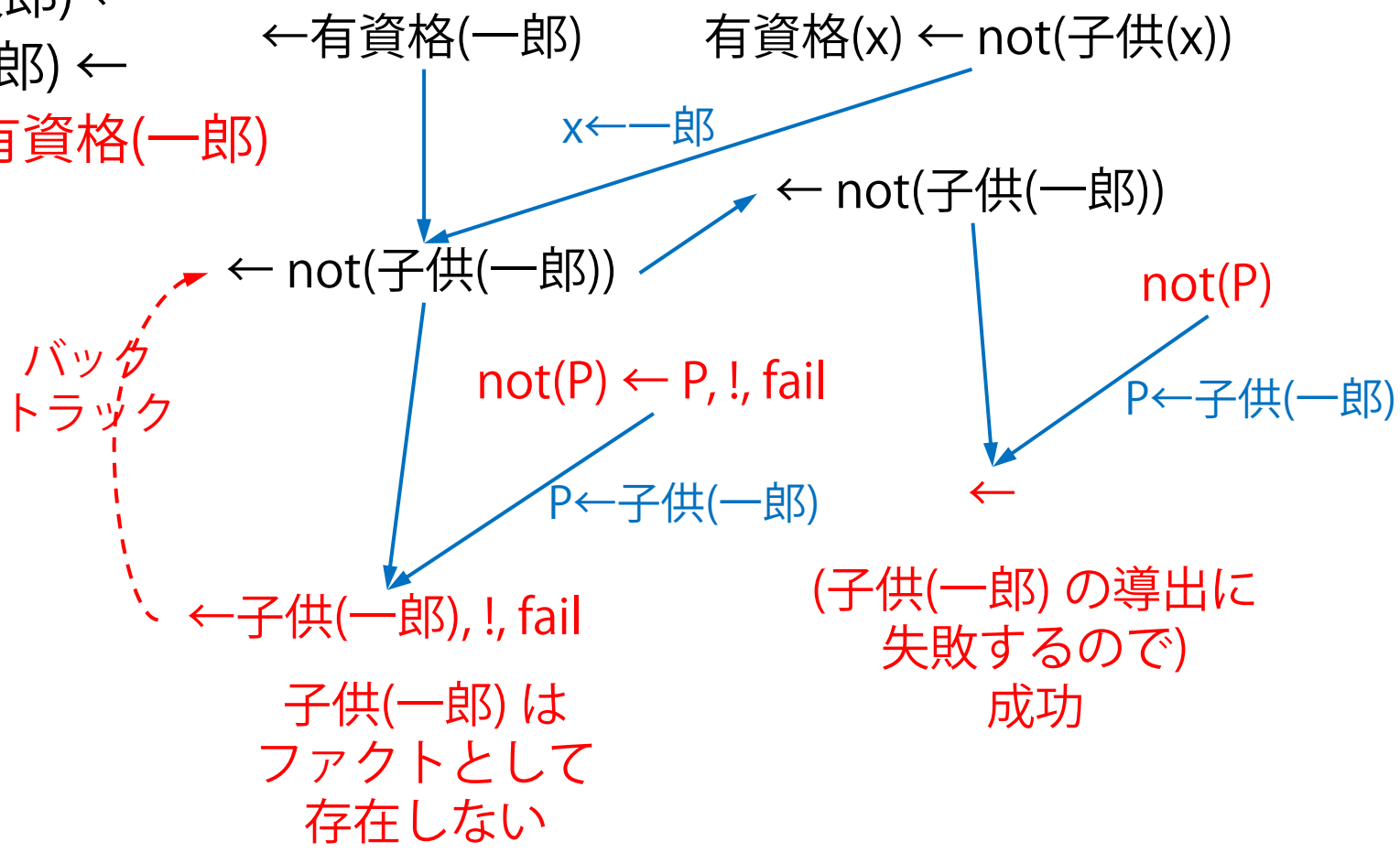
! より右側のリテラルに対して導出を試みた場合,
失敗しても !より左側の導出には戻らず失敗とする

Prologその他

- 否定
  - ホーン節では節内に既に否定演算子が含まれている
  - 従って従来の否定を扱うことはできない
- Prologでは，おおよそ否定を意味するnot演算子を用意している
  - not(P) ← P, !, fail
  - not(P)
    - Pのマッチングに成功すれば，カット演算子を越えて失敗するので失敗となる
    - Pのマッチングに失敗したら，2つ目の確定節に移り，成功を返す
- 失敗による否定(Negation as failure)と呼ばれている

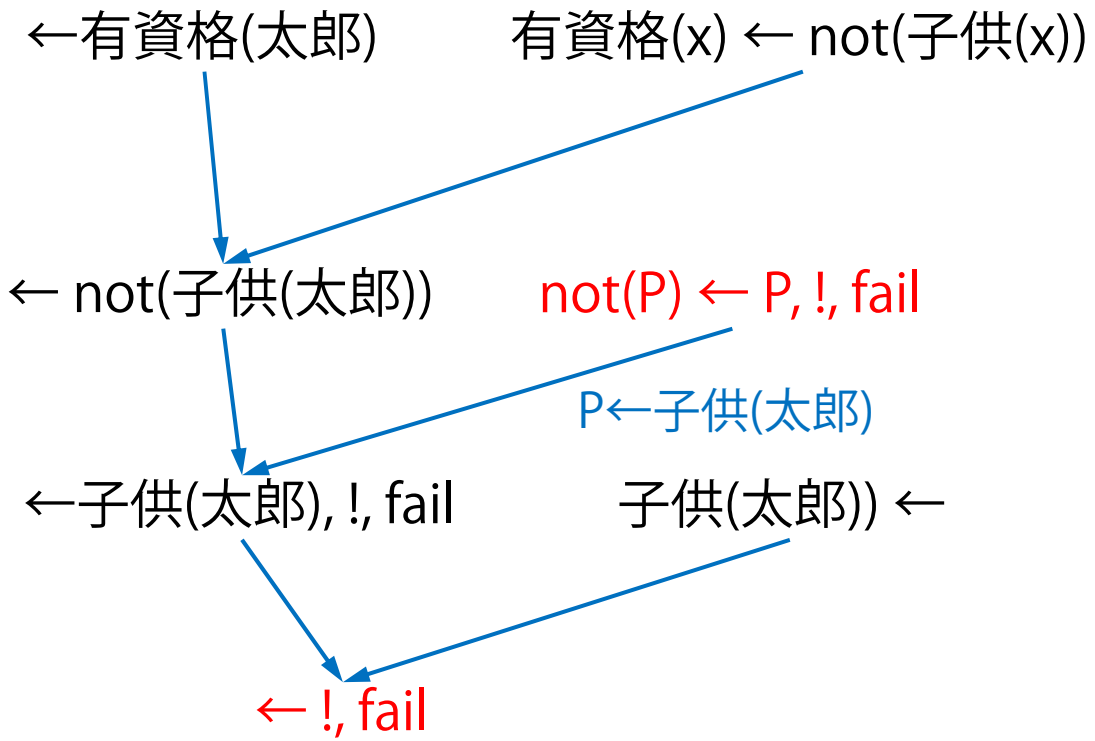
Prologの実行例

- (1) 有資格(x) ← not(子供(x))
- (2) 子供(太郎) ←
- (3)子供(次郎) ←
- (goal) ← 有資格(一郎)



Prologの実行例

- (1) 有資格(x) ← not(子供(x))
- (2) 子供(太郎) ←
- (3)子供(次郎) ←
- (goal) ← 有資格(太郎)



(子供(太郎) の導出に成功するので)
バックトラックできずに失敗となる

Prologの特徴

- 論理型プログラミング言語の一つ
  - ルール節, ファクト節が存在する状況下で, ゴール節が成り立つかどうかを問い合わせる形式
  - 全ての節はホーン節として記述される
- SLD導出を採用
- 理論(理想)どおりに行かないところもある
  - SLD導出の手順がfix
    - 健全ではあるが完全ではない
    - プログラムの記述, リテラルの記述順に依存
      - 永久ループに陥りやすい
  - 否定を扱うことができない: notで代用
  - writeなど副作用を伴うオペレーションを導入

まとめ

- 目標 [再掲]
  - 述語論理式を理解する，記述できる
    - 限定作用素により量について言及
    - 自由変数, 束縛, 代入, 証明法
  - (一部の) 述語論理式の恒偽性の確認手段を習得する
    - あるクラスの論理式であれば, コンピュータを用いて自動的に恒偽 (充足不能)であることを確認できる
    - 導出原理により確認
      - 導出原理適用のために: 冠頭標準形, スコーレム連言標準形, エルブランの定理, mgu
    - 論理型プログラミング言語 Prolog