関数・論理型プログラミング実験
Prolog演習第2回
（通算第12回）

松田 一孝

TA: 武田広太郎 寺尾拓

今日からProlog演習

- 論理プログラミング演習の予定
  - 第1回（6/24）
    - Prologの使い方
  - 第2回（7/1）
    - Prologの評価メカニズム
  - 第3回（7/8）
    - 探索について（仮）
  - 第4回（7/15）
    - 関数論理型言語Curry

今日の話

- Prologはどうやって動いているの？
  - SLD導出（SLD resolution）
    - S: Selective
      L: Linear
      - $P_1, \ldots, P_n$を確認するのに
        P_iを「いっこ」「選ぶ」
    - D: Definite clauses
      - $Q \leftarrow P_1, \ldots, P_n$の形の節（要はruleとfact）のこと
        - 正確には$Q \vee \neg P_1 \vee \cdots \vee \neg P_n$

例

```
male(kobo).
male(kojì).
male(iwao).
female(sanae).
female(mìne).
parent(kobo, kojì).
parent(kobo, sanae).
parent(sanae, ìwao).
parent(sanae, mìne).
father(X, Y)  :- parent(X, Y), male(Y).
mother(X, Y)  :- parent(X, Y), female(Y).
```

動作例

```
?- father(kobo,koji)
```

Goal:
成否を確認したい式

動作例

```
?- father(kobo,koji)
```

Goal:
成否を確認したい式

```
father(X,Y) :- parent(X,Y),male(Y).
```

動作例

```
?- father(kobo, koji)
```

Goal:
成否を確認したい式

```
father(X, Y) :- parent(X, Y), male(Y).
```

```
?- parent(kobo, koji), male(koji)
```

動作例

```
?- father(kobo, koji)
```

Goal:
成否を確認したい式

```
father(X, Y) :- parent(X, Y), male(Y).
```

```
?- parent(kobo, koji), male(koji)
```

```
parent(kobo, koji).
```

動作例

```
?- father(kobo, koji)
```

Goal:
成否を確認したい式

```
father(X, Y) :- parent(X, Y), male(Y).
```

```
?- parent(kobo, koji), male(koji)
```

```
parent(kobo, koji).
```

```
?- male(koji)
```

動作例

```
?- father(kobo, koji)
```

Goal:
成否を確認したい式

```
father(X, Y) :- parent(X, Y), male(Y).
```

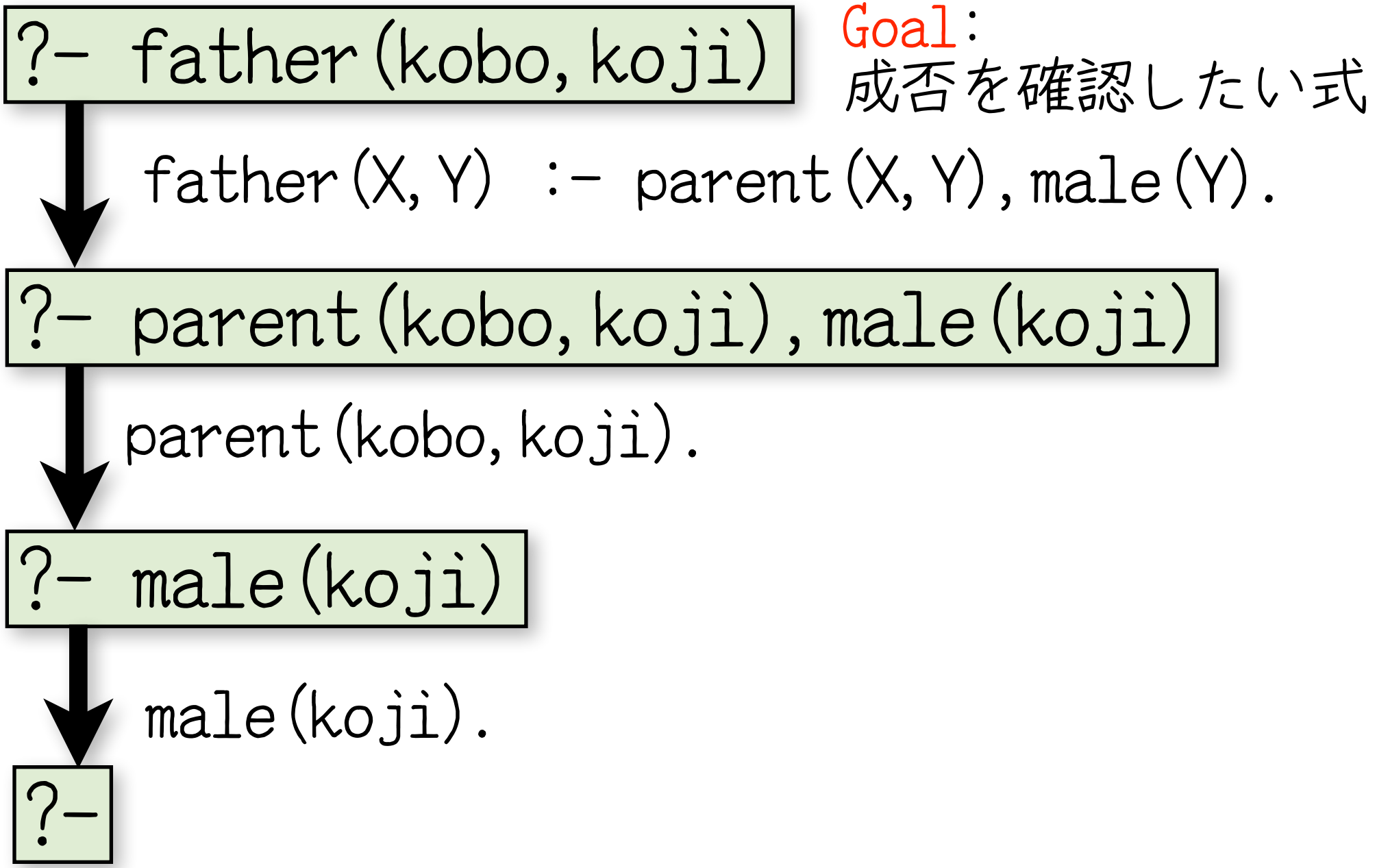
```
?- parent(kobo, koji), male(koji)
```

```
parent(kobo, koji).
```

```
?- male(koji)
```

```
male(koji).
```

動作例


?- father(kobo, koji)
Goal:
成否を確認したい式
father(X, Y) :- parent(X, Y), male(Y).
?- parent(kobo, koji), male(koji)
parent(kobo, koji).
?- male(koji)
male(koji).
?-

動作例

```
?- father(kobo, koji)
```

Goal:
成否を確認したい式

```
father(X, Y) :- parent(X, Y), male(Y).
```

```
?- parent(kobo, koji), male(koji)
```

```
parent(kobo, koji).
```

```
?- male(koji)
```

```
male(koji).
```

```
?-
```

成功：Goalの式が全て成り立つことを確認

動作例

```
?- father(kobo, X)
```

動作例

```
?- father(kobo, X)
```

```
father(X, Y) :- parent(X, Y), male(Y).
```

動作例

```
?- father(kobo, X)
```

```
father(X, Y) :- parent(X, Y), male(Y).
```

```
?- parent(kobo, X), male(X)
```

動作例

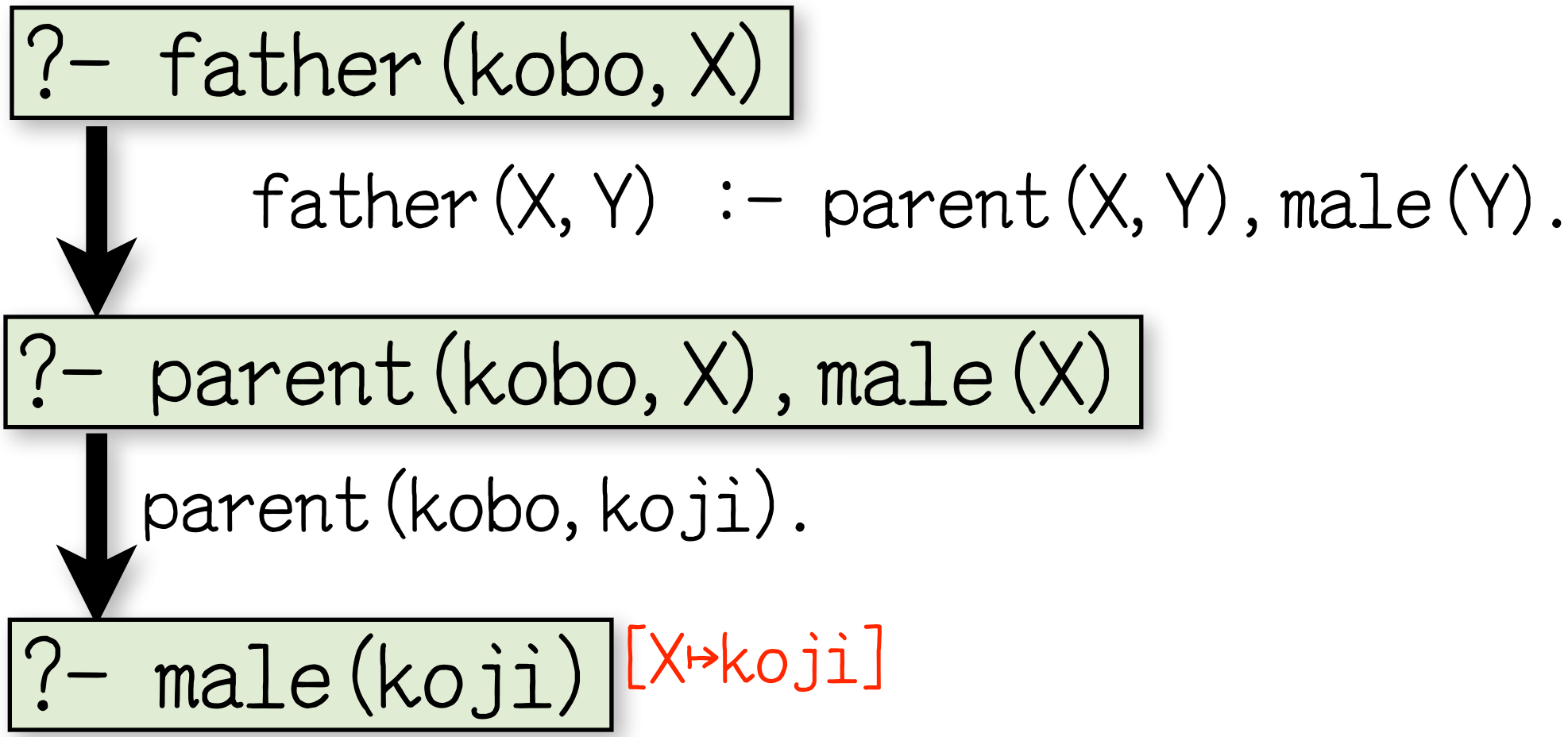
```
?- father(kobo, X)
```

```
father(X, Y) :- parent(X, Y), male(Y).
```

```
?- parent(kobo, X), male(X)
```

```
parent(kobo, koji).
```

動作例


?- father(kobo, X)
father(X, Y) :- parent(X, Y), male(Y).
?- parent(kobo, X), male(X)
parent(kobo, koji).
?- male(koji)
[X↦koji]

動作例

```
?- father(kobo, X)
```

```
father(X, Y) :- parent(X, Y), male(Y).
```

```
?- parent(kobo, X), male(X)
```

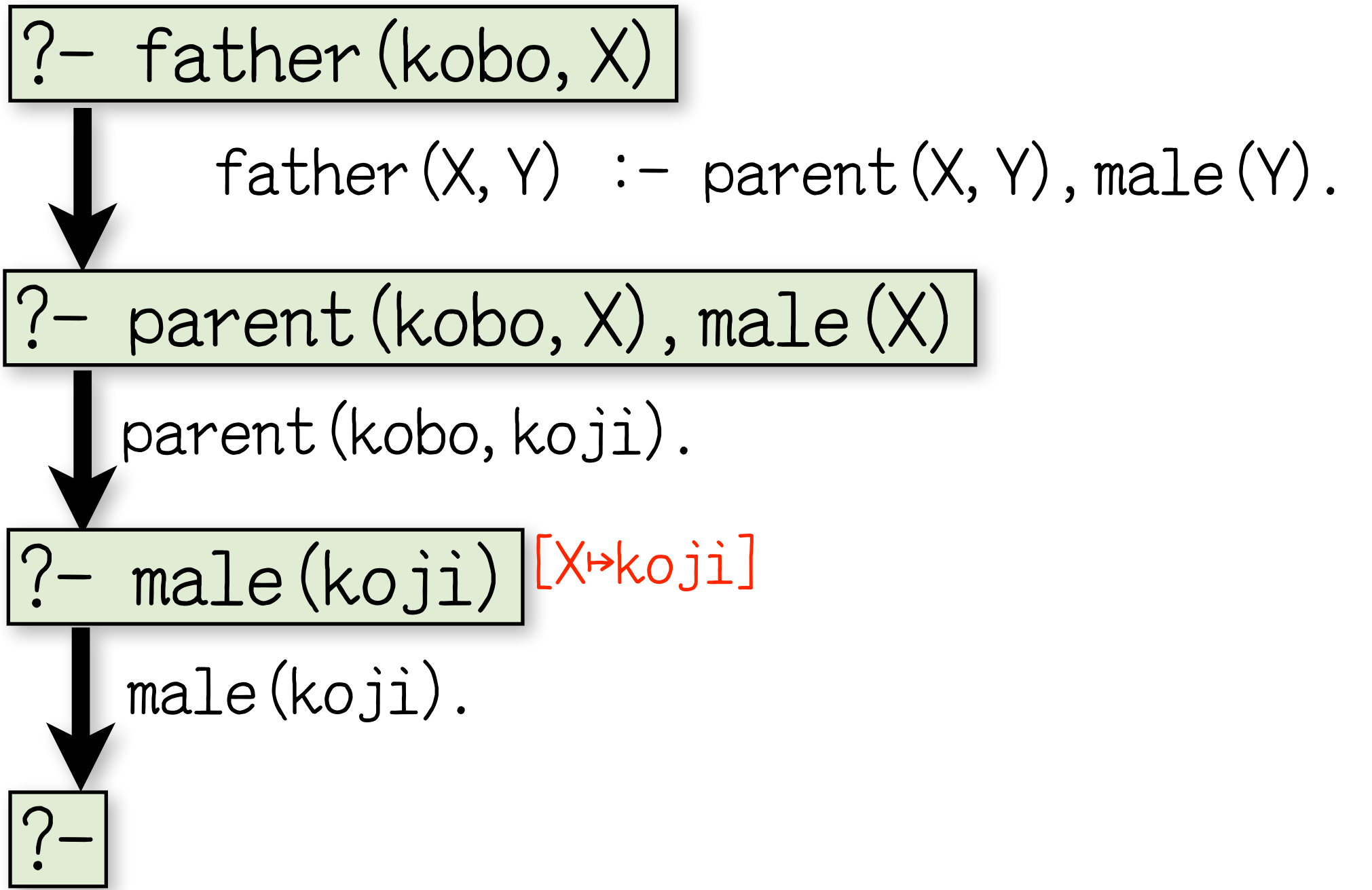
```
parent(kobo, koji).
```

```
?- male(koji)
```

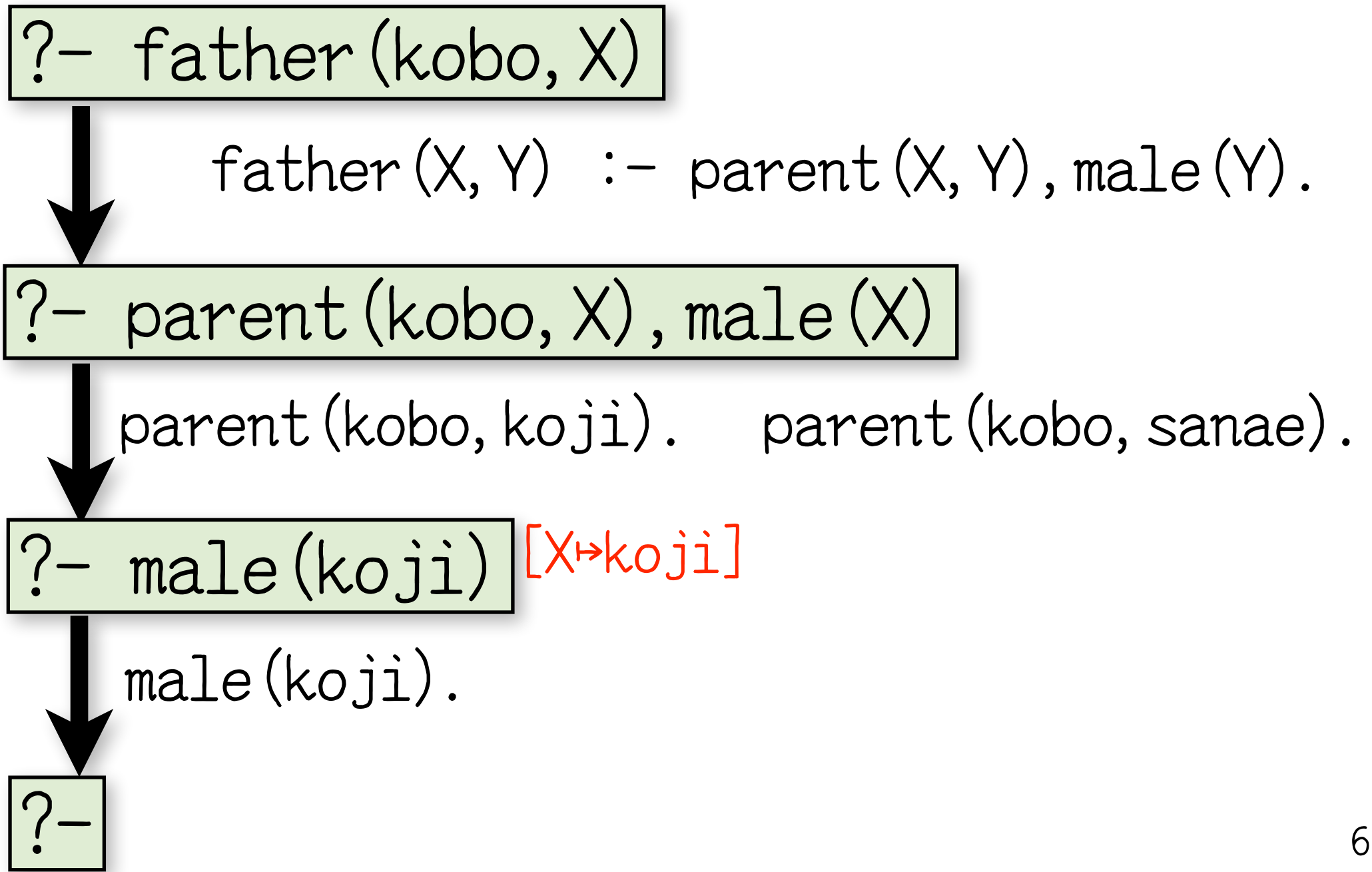
[X↦koji]

```
male(koji).
```

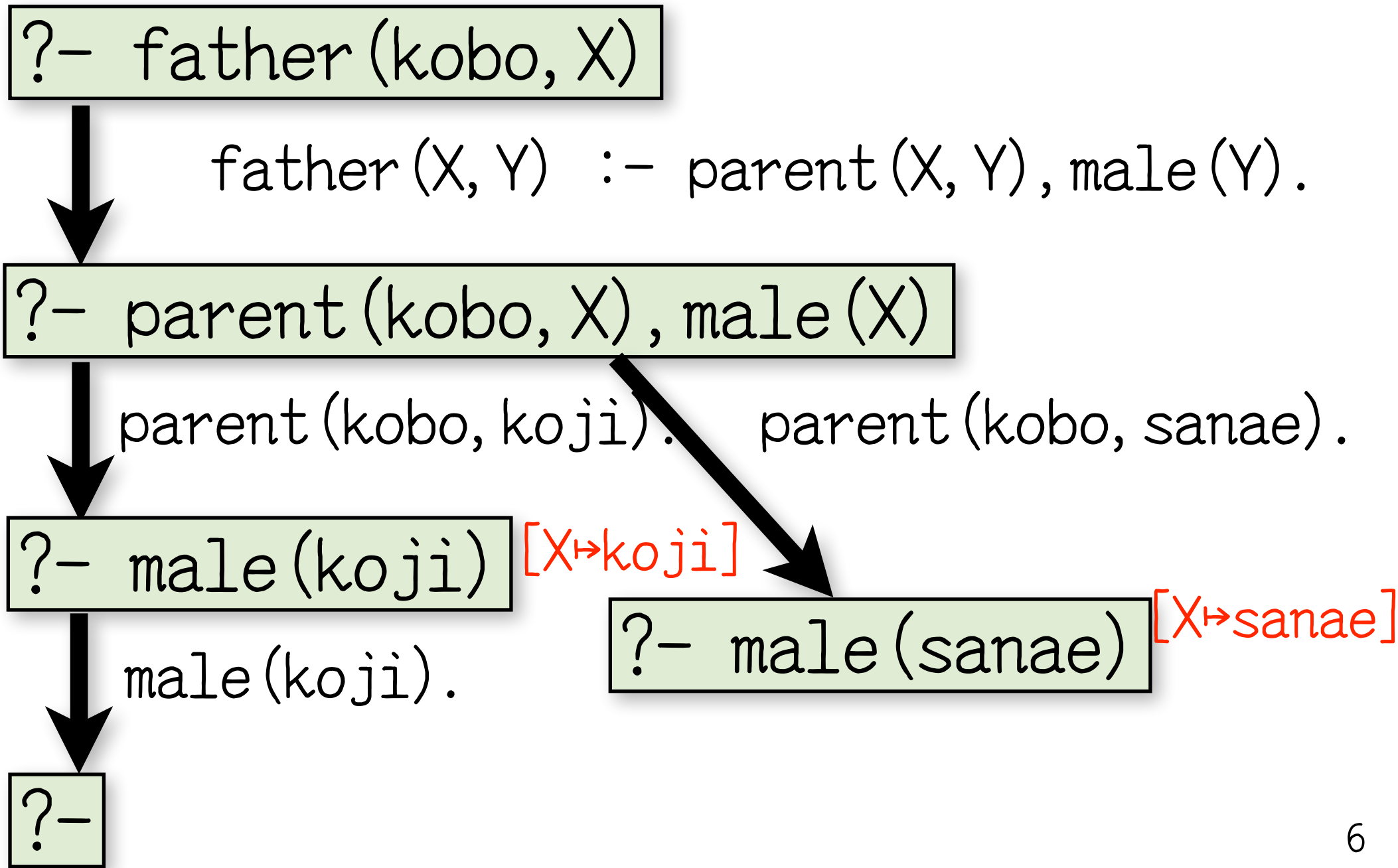
動作例


?- father(kobo, X)
father(X, Y) :- parent(X, Y), male(Y).
?- parent(kobo, X), male(X)
parent(kobo, koji).
?- male(koji)
[X↦koji]
male(koji).
?-

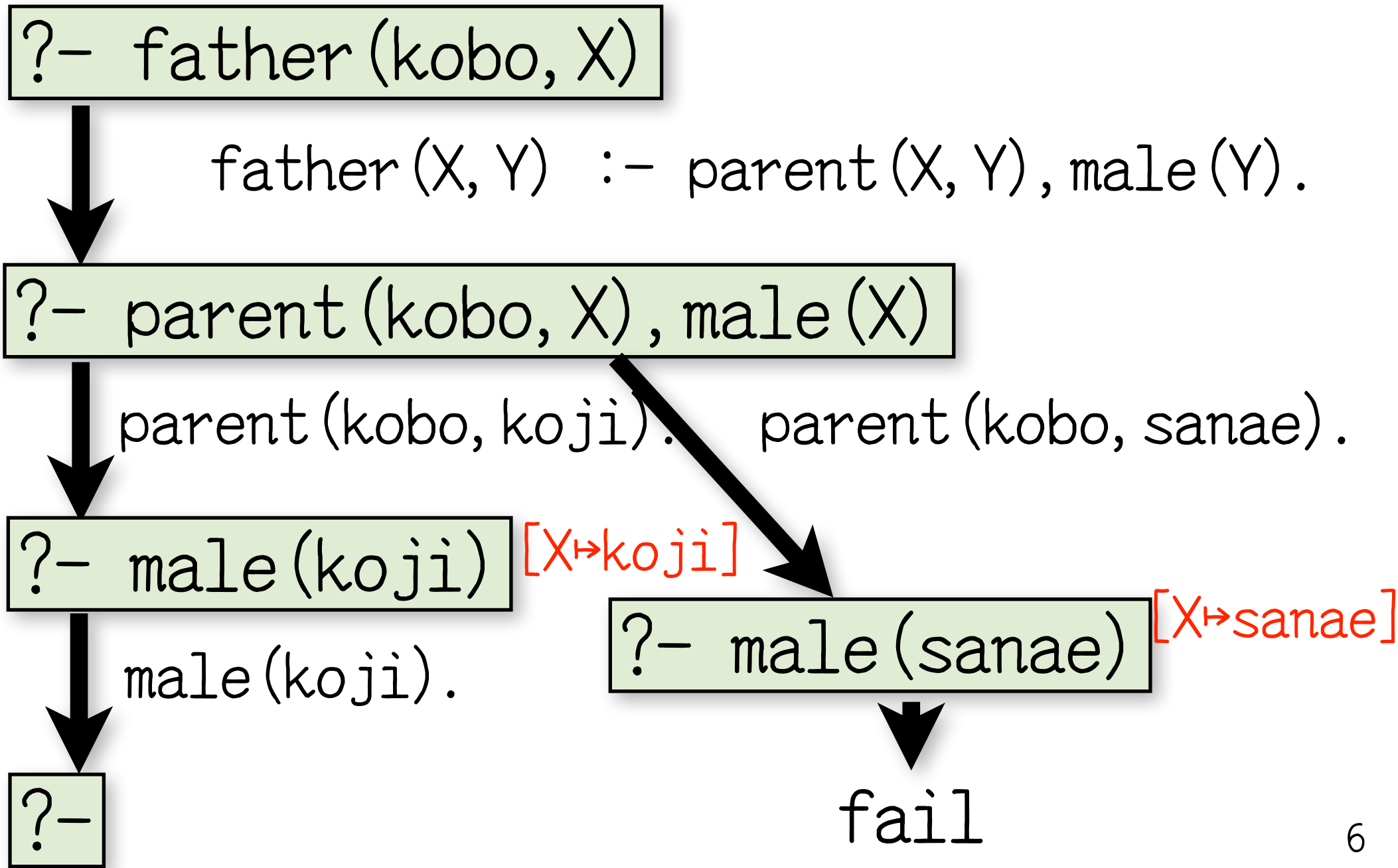
動作例


?- father(kobo, X)
father(X, Y) :- parent(X, Y), male(Y).
?- parent(kobo, X), male(X)
parent(kobo, koji). parent(kobo, sanae).
?- male(koji)
[X↦koji]
male(koji).
?-

動作例


?- father(kobo, X)
father(X, Y) :- parent(X, Y), male(Y).
?- parent(kobo, X), male(X)
parent(kobo, koji).
parent(kobo, sanae).
?- male(koji)
[X↦koji]
?- male(sanae)
[X↦sanae]
male(koji).
?-

動作例


?- father(kobo, X)
father(X, Y) :- parent(X, Y), male(Y).
?- parent(kobo, X), male(X)
parent(kobo, koji).
parent(kobo, sanae).
?- male(koji)
[X↦koji]
?- male(sanae)
[X↦sanae]
male(koji).
?-
fail

動作例

```
?- mother(kobo, X)
```

動作例

```
?- mother(kobo, X)
```

```
mother(X, Y) :- parent(X, Y), female(Y).
```

動作例

```
?- mother(kobo, X)
```

```
mother(X, Y) :- parent(X, Y), female(Y).
```

```
?- parent(kobo, X), female(X)
```

動作例

```
?- mother(kobo, X)
```

```
mother(X, Y) :- parent(X, Y), female(Y).
```

```
?- parent(kobo, X), female(X)
```

```
parent(kobo, koji).
```

動作例

```
?- mother(kobo, X)
```

```
mother(X, Y) :- parent(X, Y), female(Y).
```

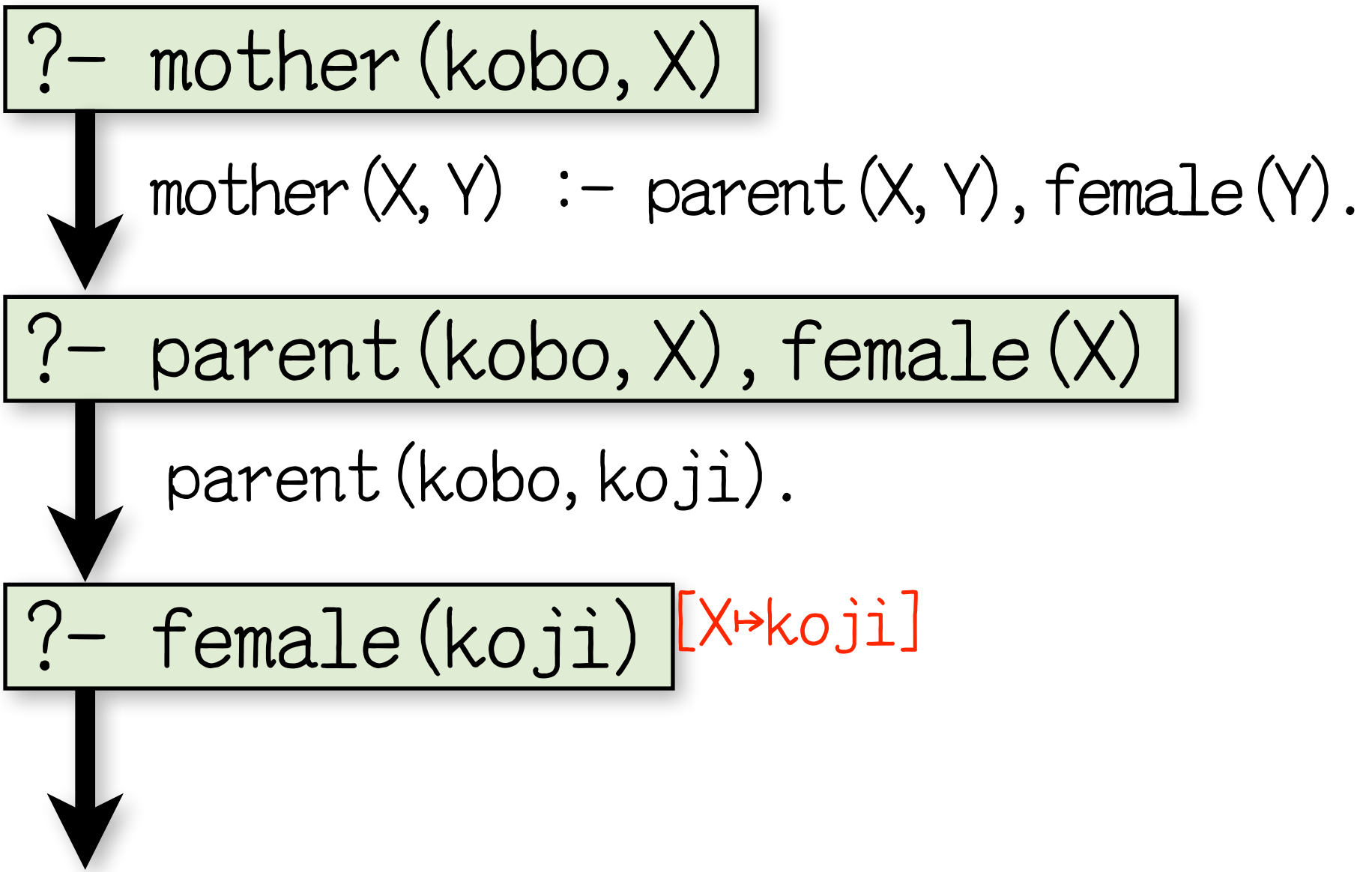
```
?- parent(kobo, X), female(X)
```

```
parent(kobo, koji).
```

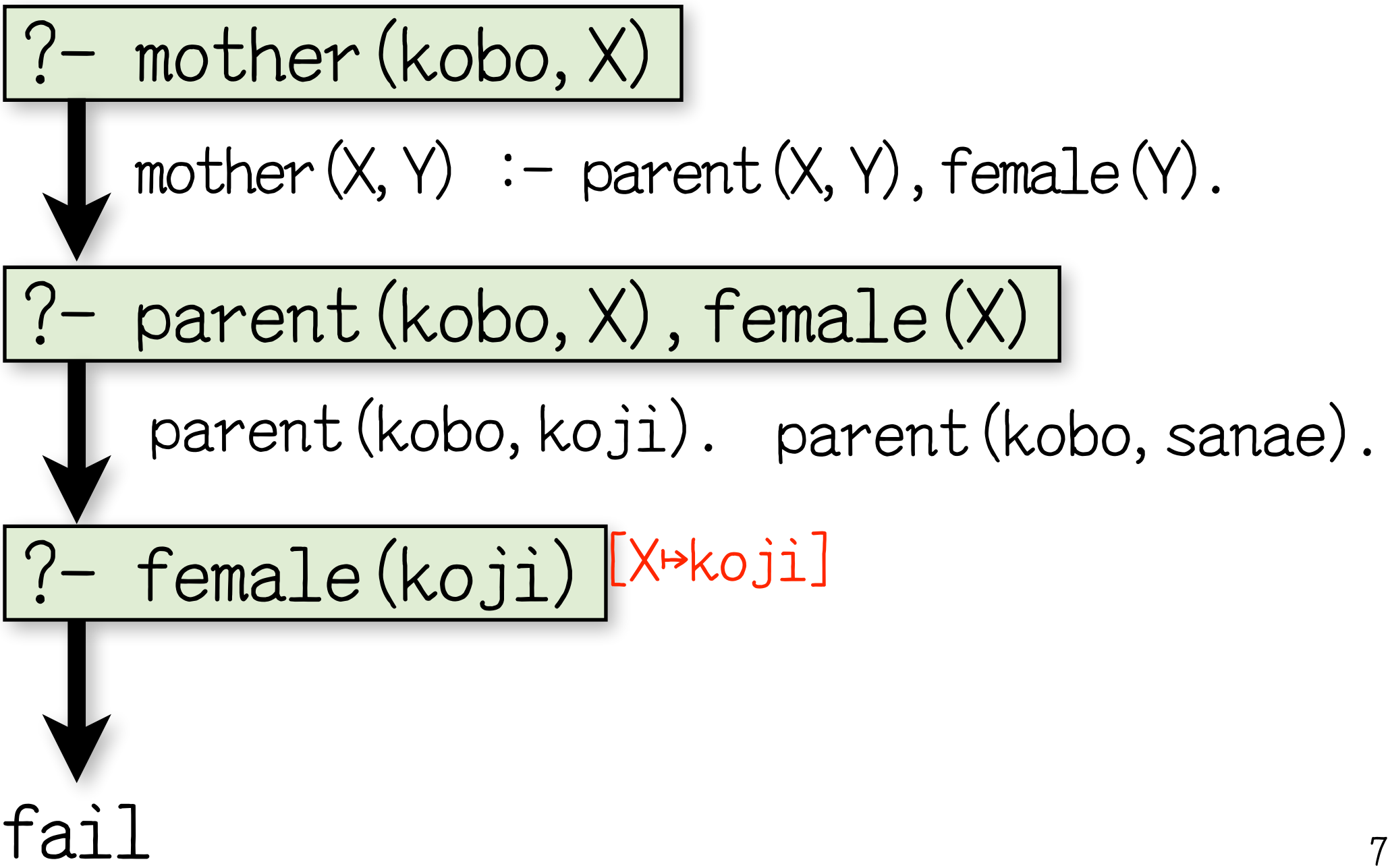
```
?- female(koji)
```

[X↦koji]

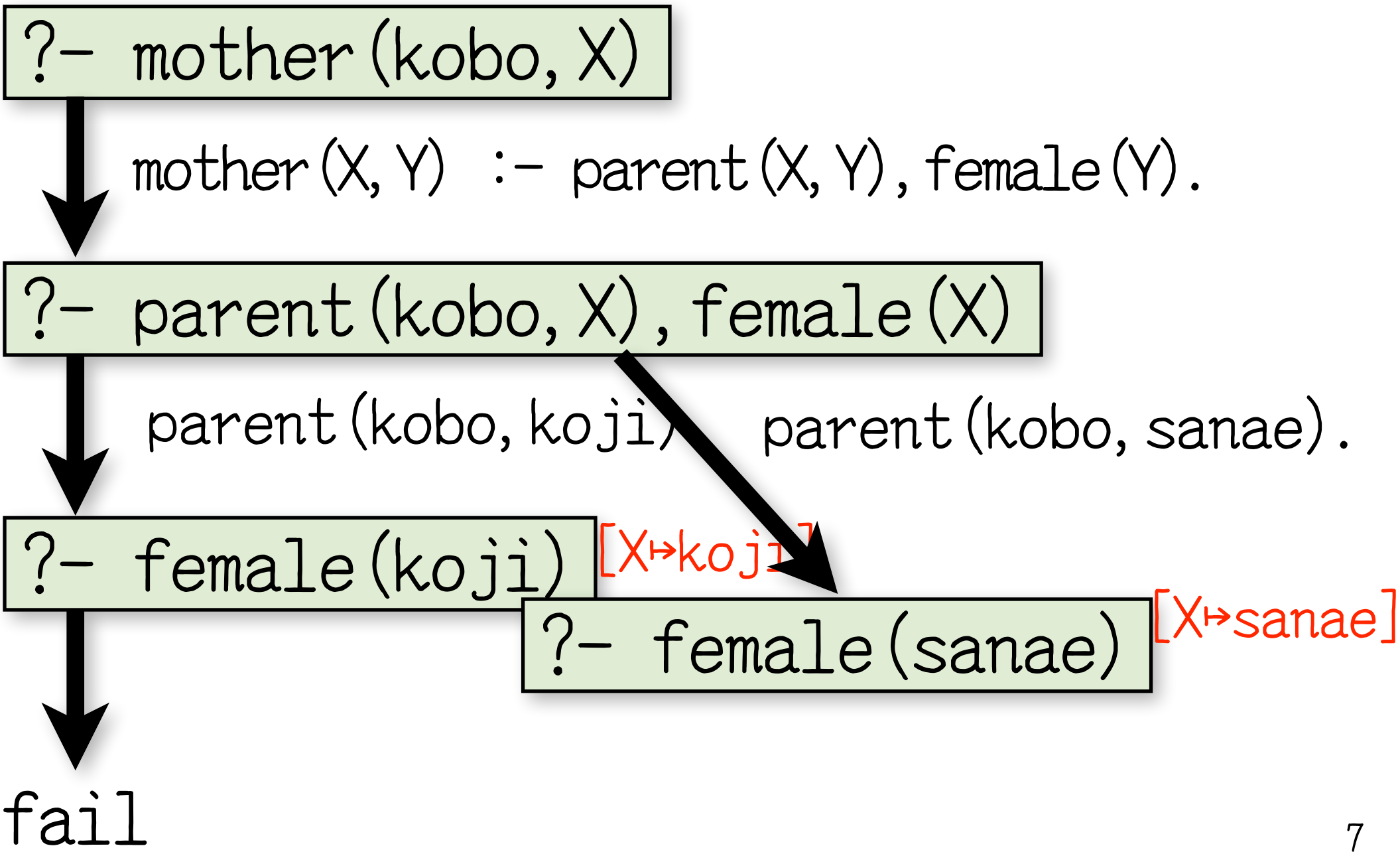
動作例


?- mother(kobo, X)
mother(X, Y) :- parent(X, Y), female(Y).
?- parent(kobo, X), female(X)
parent(kobo, koji).
?- female(koji)
[X↦koji]
fail

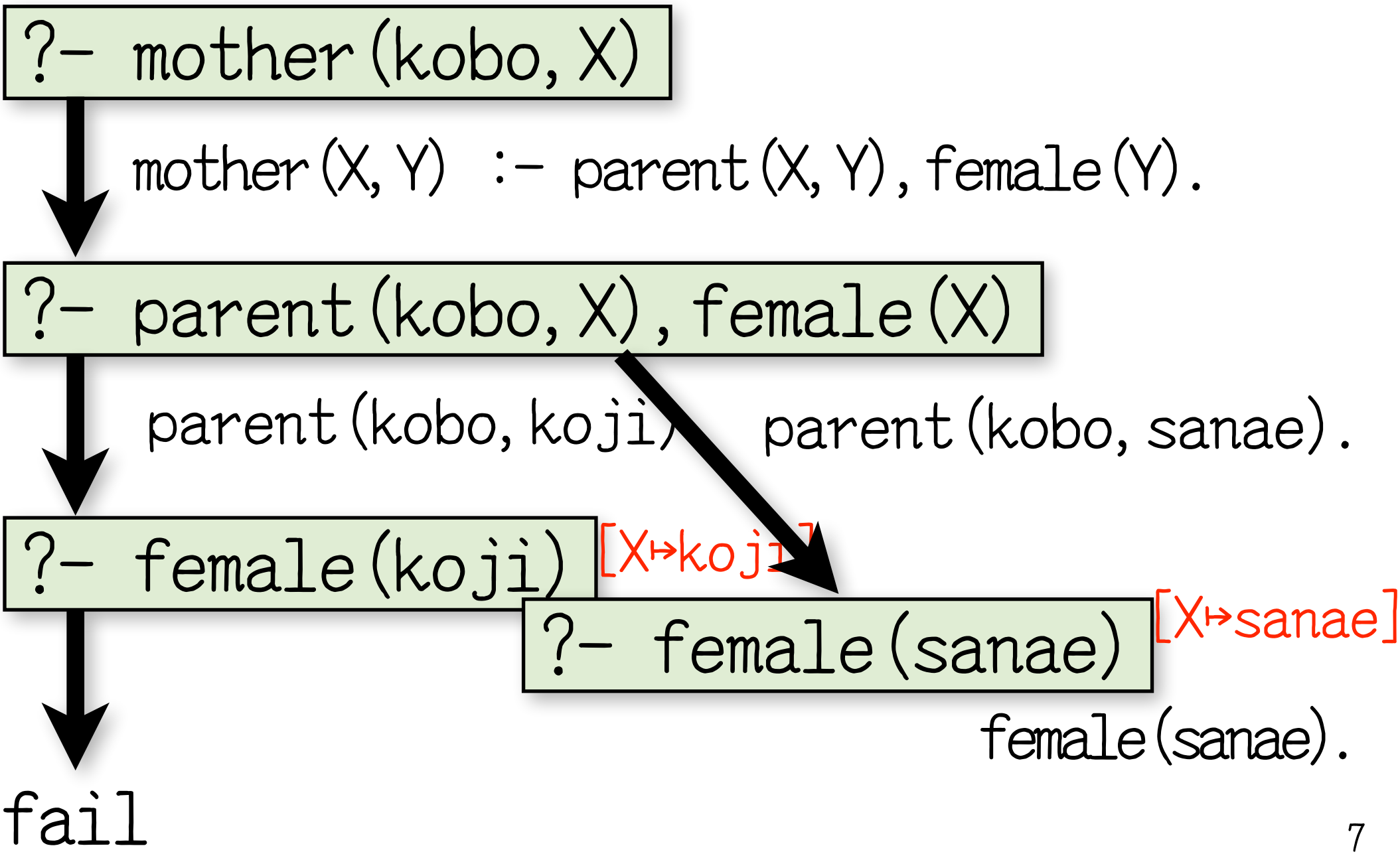
動作例


?- mother(kobo, X)
mother(X, Y) :- parent(X, Y), female(Y).
?- parent(kobo, X), female(X)
parent(kobo, koji). parent(kobo, sanae).
?- female(koji)
[X↦koji]
fail

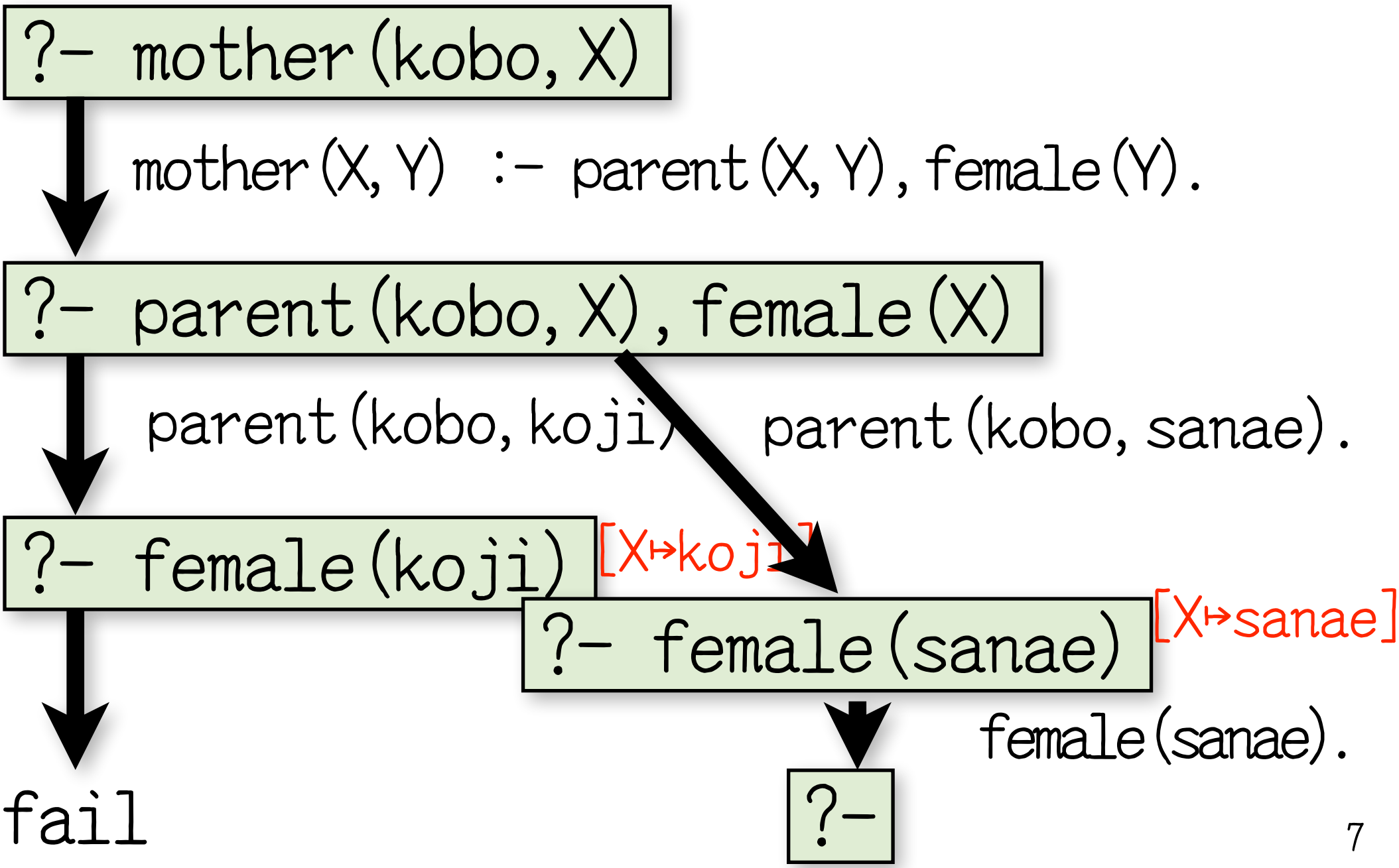
動作例


?- mother(kobo, X)
mother(X, Y) :- parent(X, Y), female(Y).
?- parent(kobo, X), female(X)
parent(kobo, koji).
parent(kobo, sanae).
?- female(koji)
[X↦koji]
?- female(sanae)
[X↦sanae]
fail

動作例


?- mother(kobo, X)
mother(X, Y) :- parent(X, Y), female(Y).
?- parent(kobo, X), female(X)
parent(kobo, koji).
parent(kobo, sanae).
?- female(koji)
[X↦koji]
?- female(sanae)
[X↦sanae]
female(sanae).
fail

動作例


?- mother(kobo, X)
mother(X, Y) :- parent(X, Y), female(Y).
?- parent(kobo, X), female(X)
parent(kobo, koji).
parent(kobo, sanae).
?- female(koji)
[X↦koji]
?- female(sanae)
[X↦sanae]
female(sanae).
fail
?-

やっていること

- Goalの一部をruleやfactにより置き換え，Goalが空か確認
  - また，置き換え時には[X↦koji]や[X↦sanae]などの代入が生じる

- 複数の置き換えが行える場合があり，その場合は一つずつ試す
  - 適用できるruleやfactが複数ある場合
    - 例：parent(kobo,koji)とparent(kobo, sanae)

単一化&最汎単一化子

- SLDについて述べる前に，
  単一化と最汎単一化子について
  復習/確認する
  - 第6回（型推論）で出てきた
    - 第6回のunifyの
      やることが単一化，
      求めるものが（最汎）単一化子

単一化（unification）

- 自由変数を含む項sとtに対し，
  sθ = tθとなる代入θを求めること
  - sθ：s中の全ての自由変数Xをθ(X)で置き換えたもの
  - θ：単一化子，単一化代入（unifier）
  - 例：
    - Xとkojiについて[X↦koji]
    - s(X)とYについて[Y↦s(X)]や
      [Y↦s(s(z)),X↦s(z)]
    - s(X)とzとはunifiableでない
    - n(X,X)とn(l,n(l,l))もunifiableでない

最汎単一化子

- Most General Unifier (MGU)
  - 単一化子の中でもっとも一般的なもの
    - θ_1がθ_2より一般的：代入ηが存在して任意のXについて$\theta_2(X) = \eta(\theta_1(X))$.
      - 代入の上の前順序になる
  - 例：s(X)とYに対するMGUは[Y↦s(X)]
  - MGUは存在すれば唯一
    - ただし，変数の名前換えを同一視
      - 上では[Y↦s(X)]と[Y↦s(Z),X↦Z]を同一視

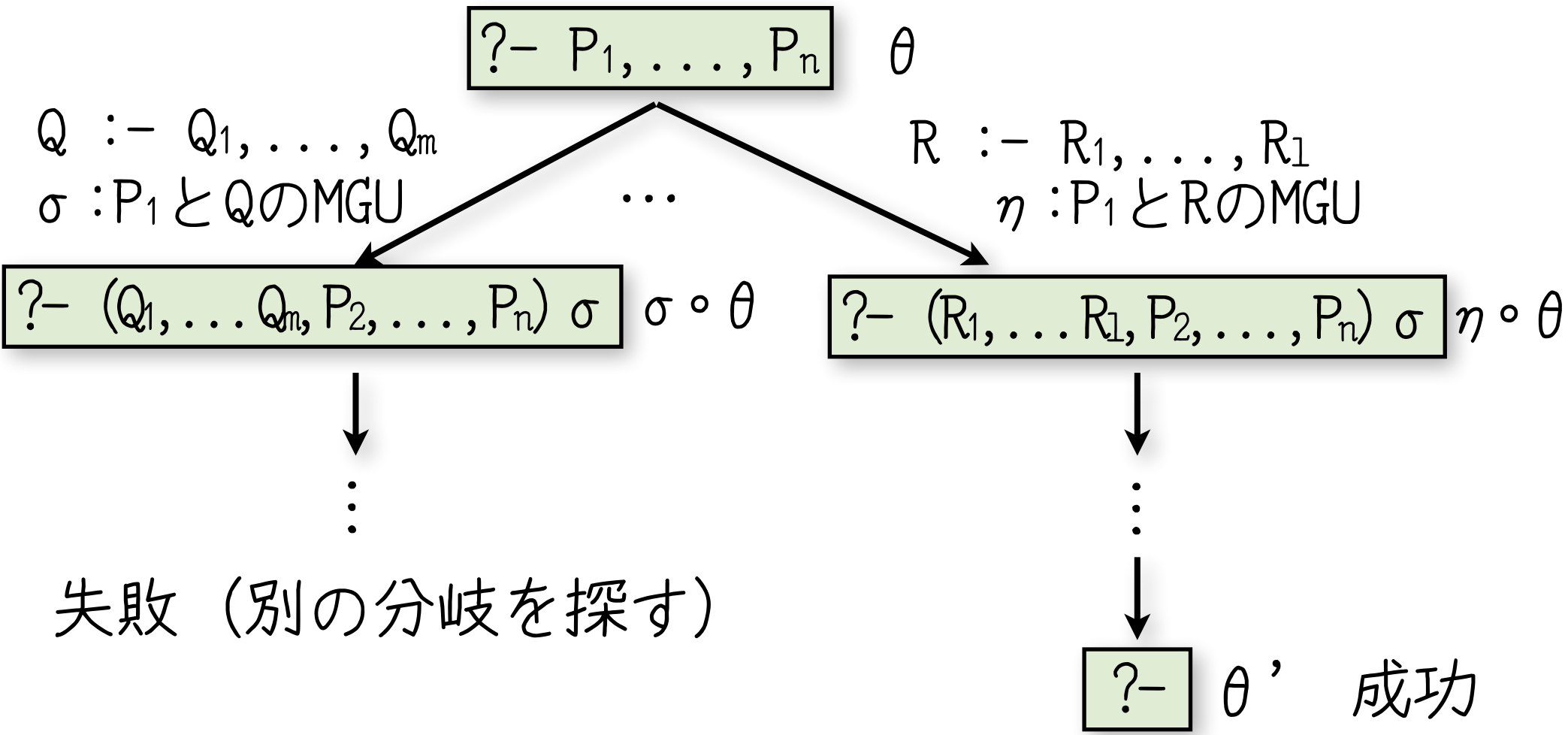
SLD導出 in Prolog (1/2)

- 入力：
  - Goal:$P_1,...,P_n$と
  - Q :- $Q_1,...,Q_m$の形のruleのリスト
    - m=0がfactに相当
- 出力：
  - 成否 + α
    - 成功ならば，$P_1,...,P_n$中の自由変数への割り当ても求める

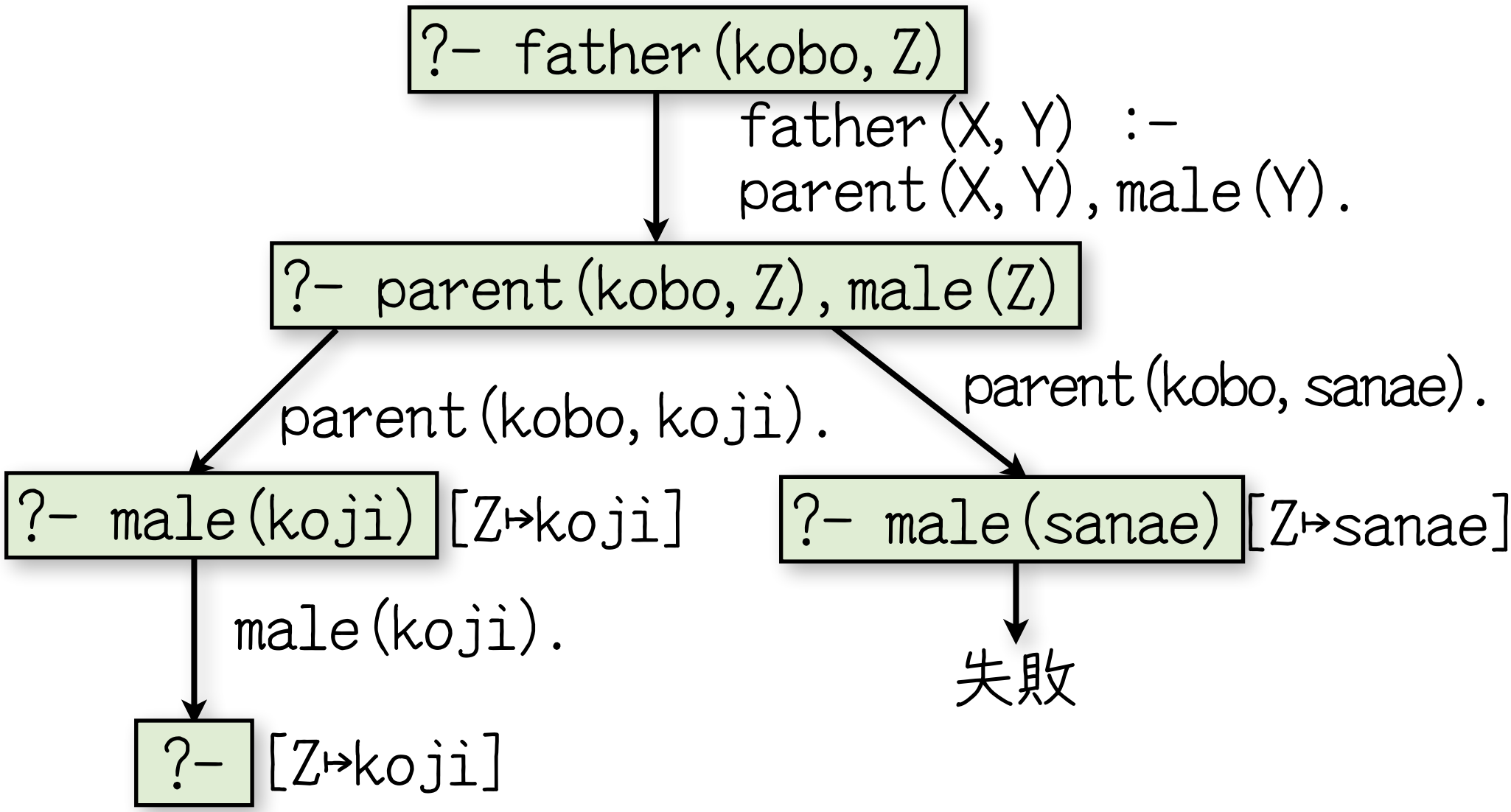
SLD導出 in Prolog (2/2)

- 手続き
  - Goal $P_1, ..., P_n$と代入θについて
    - Goalが空なら成功（+代入θ）
    - そうでなければ，最初から順にruleを調べ
      - Q :- $Q_1, ..., Q_m$が
        MGU σについて$P_1\sigma = Q\sigma$となれば
        - 新たなゴール
          $(Q_1, ..., Q_m, P_2, ..., P_n)\sigma$と$\sigma \circ \theta$に
          再帰的にこの手続きを適用
      - 上記のようなruleがなければ失敗

イメージ


?- P1, . . . , Pn
θ
Q :- Q1, . . . , Qm
σ :P1とQのMGU
…
R :- R1, . . . , Rl
η :P1とRのMGU
?- (Q1, . . . Qm, P2, . . . , Pn) σ
σ ∘ θ
?- (R1, . . . Rl, P2, . . . , Pn) σ
η ∘ θ
失敗（別の分岐を探す）
?-
θ’ 成功

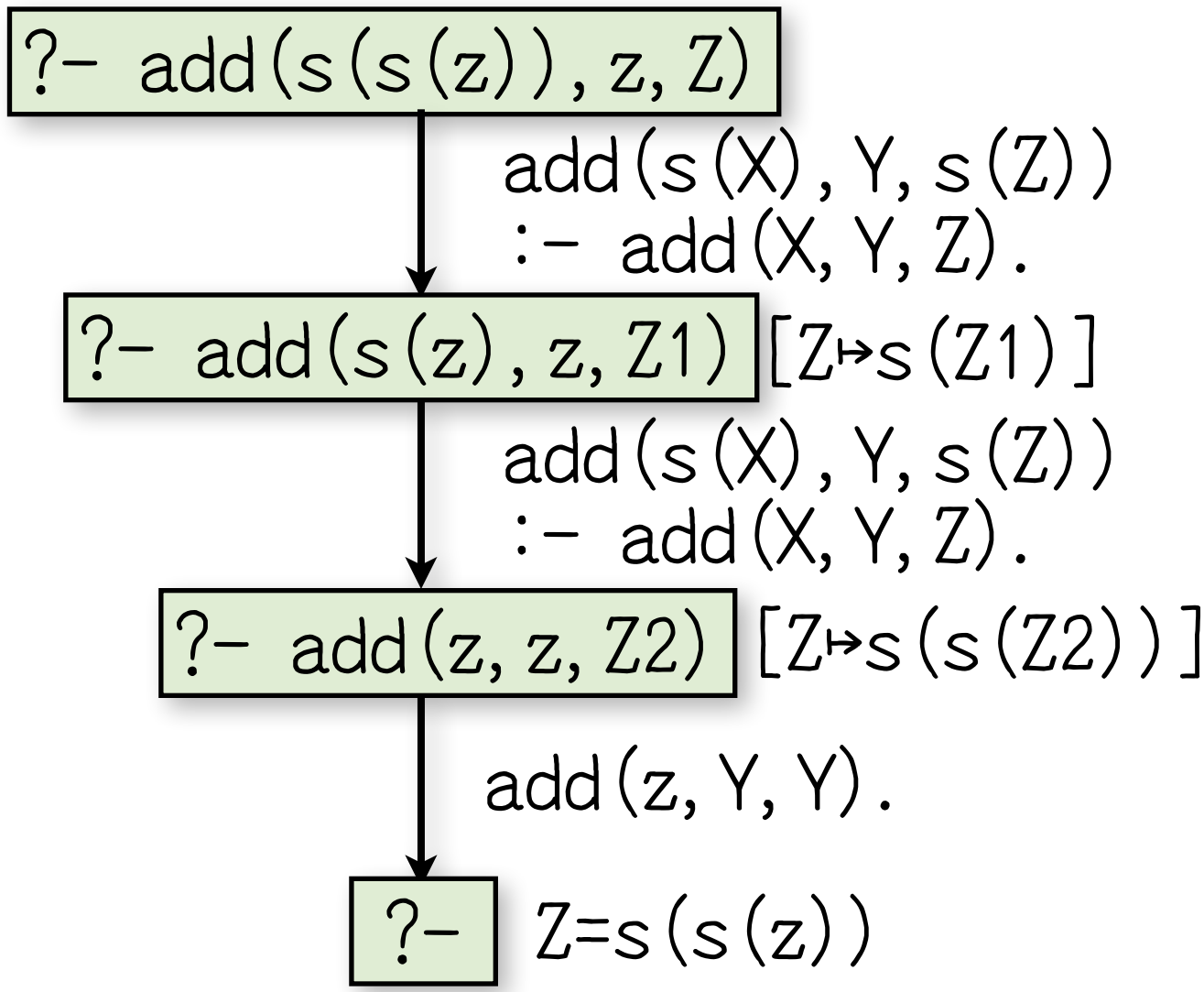
例



成功（.なら終わり，；なら次を探す）

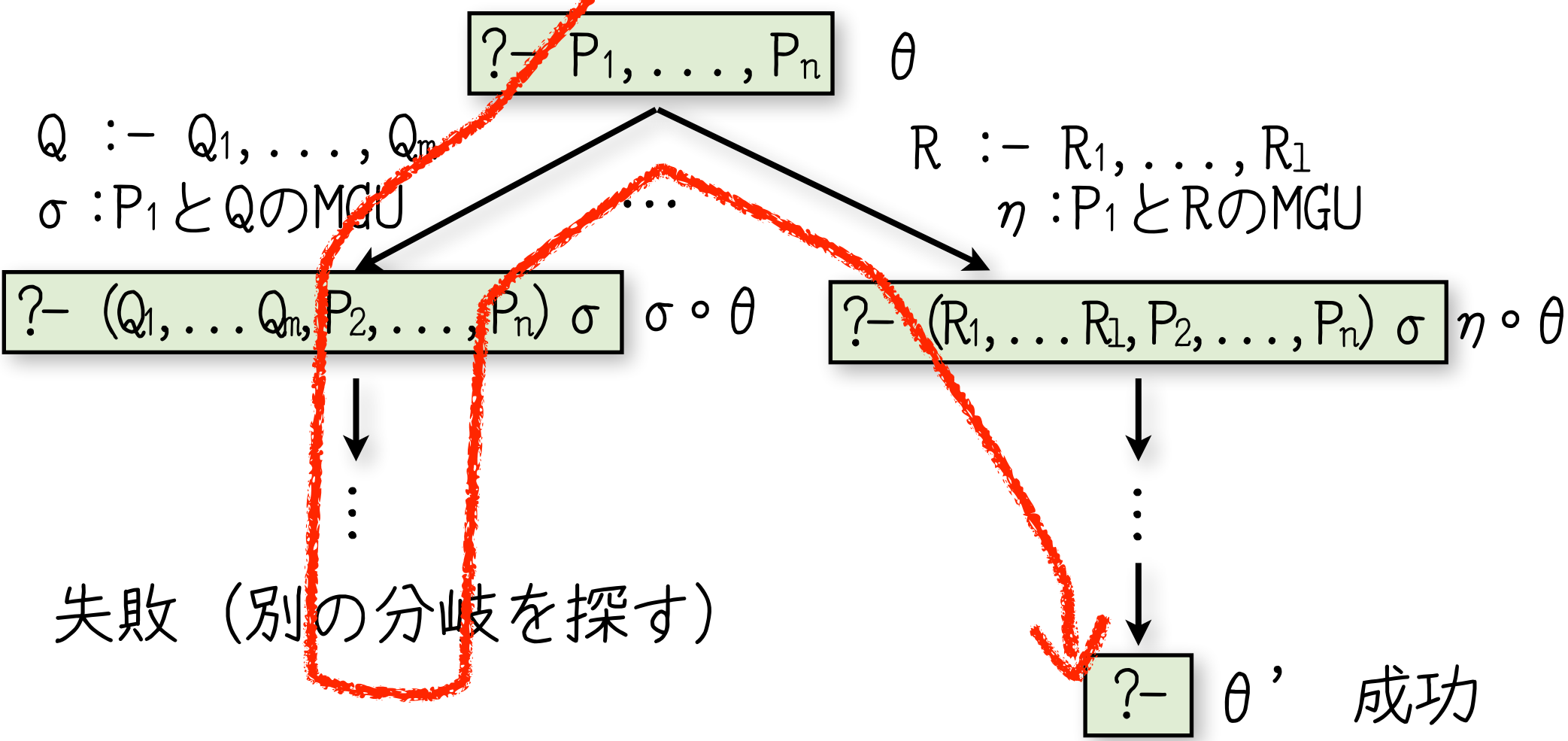
別の例

```
add(z, Y, Y).
add(s(X), Y, s(Z)) :- add(X, Y, Z)
```



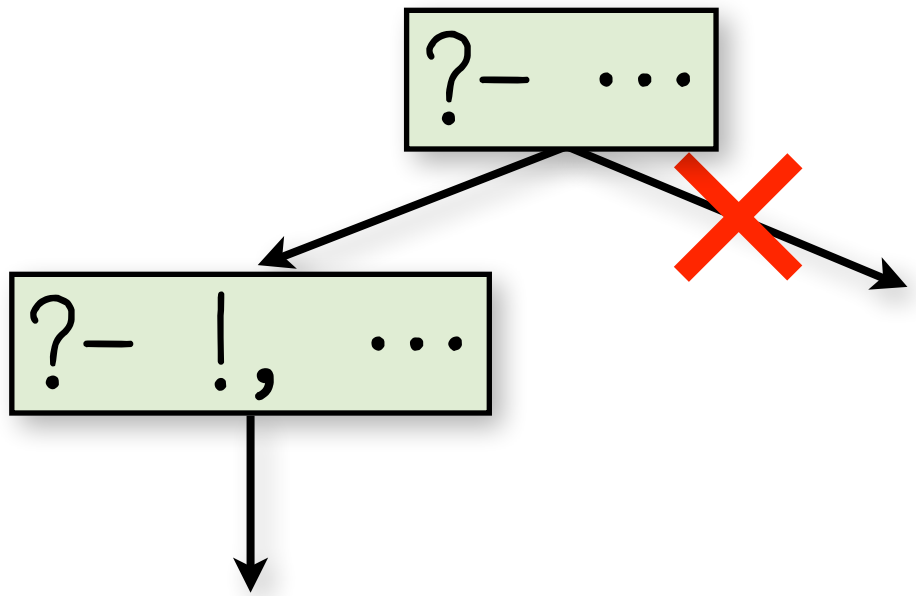
Prologの探索順

- 深さ優先（DFS）



カット

- !
  - 必ず成功，それ以下の導出で失敗してもそれより上に戻らない



カットの用法

- 探索の効率化に

```
max(X, Y, X)  :- X > Y.
max(X, Y, Y)  :- X =< Y.
```

X>Yを調べた後
X<=Yを調べるのは無駄

```
max(X, Y, X)  :- X > Y, !.
max(X, Y, Y).
```

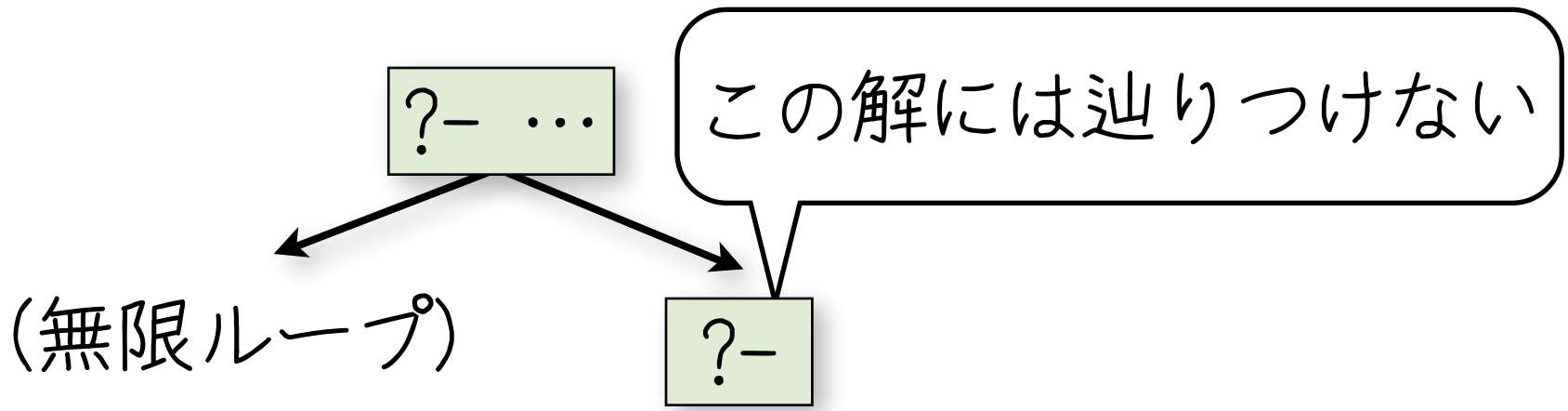
X>Yを調べたら，
X<=Yを調べない

  - 注意：カットは意味を変えうる
    - 二つ目ので定義ではmax(1, 0, 0)は成功
      下のようにすれば防げる

```
max(X, Y, Z)  :- X > Y,  !,  Z is X.
max(X, Y, Y).
```

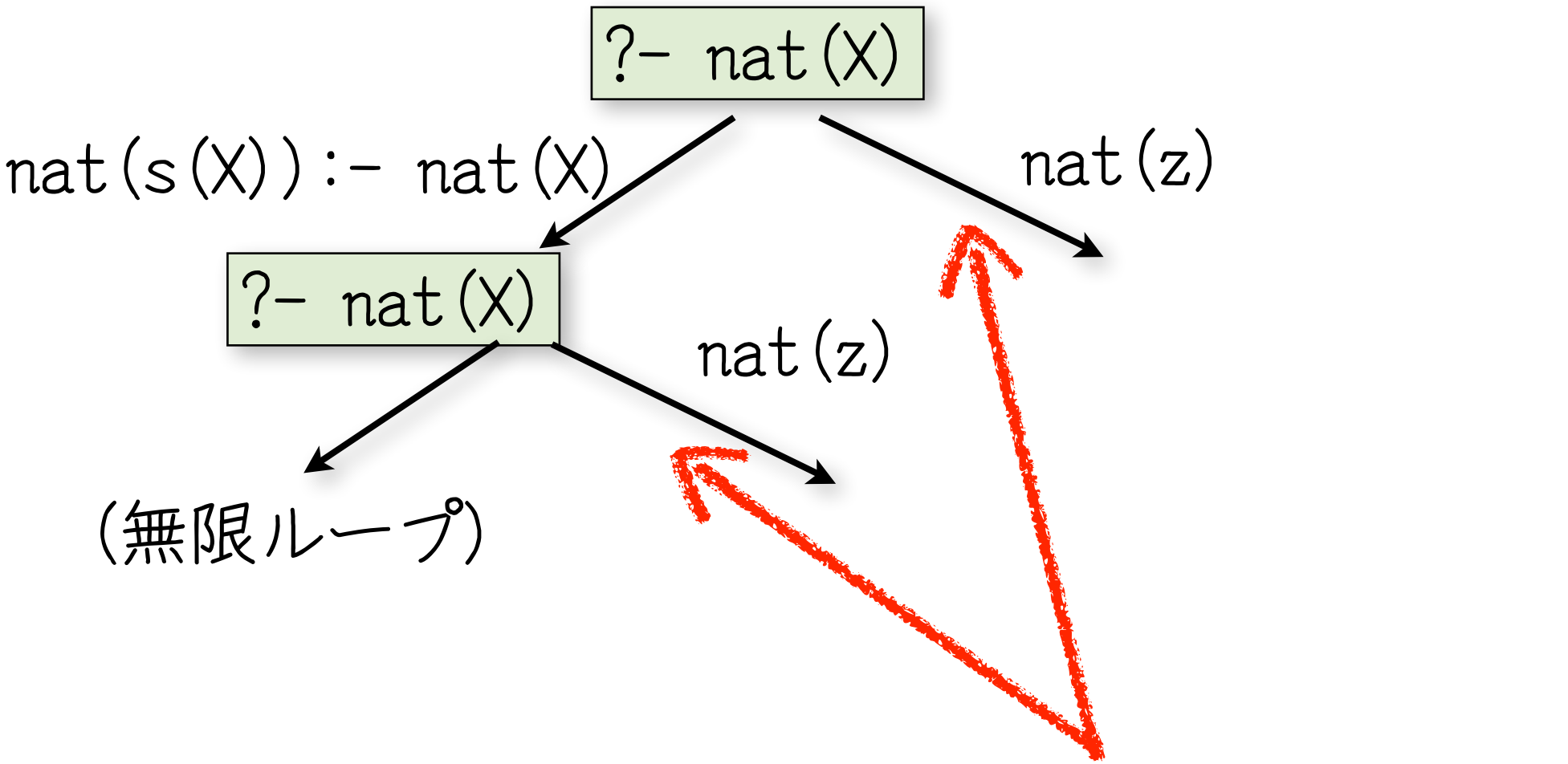
深さ優先探索の特徴

- 復習：PrologではSLD導出の際の探索順は深さ優先（DFS）
  - ＋ メモリ消費が少ない cf. 幅優先
  - － あるはずの解が見付けられない



例

```
nat(s(X)) :- nat(X).
nat(z).
```



これらの分岐は試みられない

まとめ

- Prologの動作原理を確認した
  - SLD導出

第12回レポート課題
締切 7/15 13:00

問1

- 以下の述語がうまく動作しない理由を述べよ

```
ancestor(X, Y)  :- ancestor(Z, Y), parent(X, Z).
ancestor(X, Y)  :- parent(X, Y).
```

  - たとえば，ancestor(kobo, iwao)という問い合わせはどうなるか？

問2

- 次のプログラムについて，nat_list([z,s(z),z])やnat_list([z,X])は期待通り動作するがnat_list(X)はそうでない．何故か
  - たとえば，nat_list(X),X=[s(Y)]は解を一つも返さず無限ループする

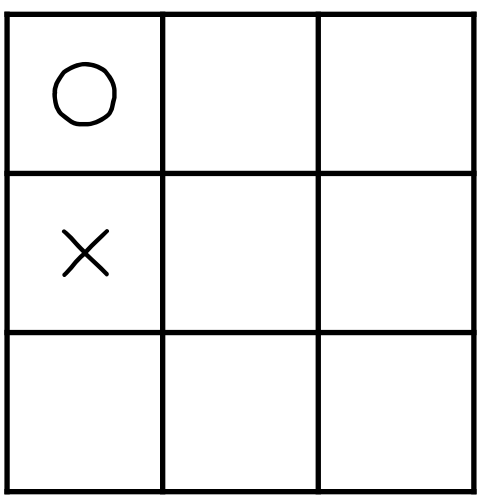
```
nat(z).
nat(s(X)) :- nat(X).
nat_list([]).
nat_list([A|X]) :- nat(A), nat_list(X).
```

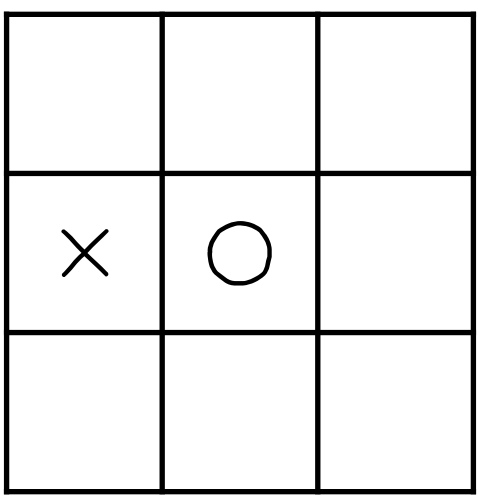
問3（1/2）

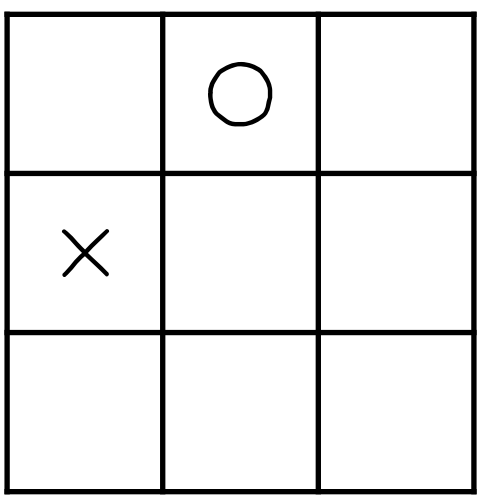
- tic-tac-toe（3×3の○×ゲーム）が双方最善をつくすと引き分けであることをPrologにより確認せよ
  - 以下の述語を使う？
    - win(P,B)
      - 盤面がBのときPの手番なら勝ち
    - lose(P,B)
      - 盤面がBのときPの手番なら負け
    - tie(P,B)
      - 盤面がBのときPの手番ならよくて引き分け

問3（2/2）

- 同じプログラムを用いて
  以下の盤面からは先手必勝であること
  を確認せよ
  - 先手：○，後手：×







ヒント

- win(P,B)は，相手をQとすると
  - 自分のマークが3つ並んでいるか
  - lose(Q,B')なるBをB'にする手がひとつ有ればOK
- lose(P,B)は，相手をQとすると
  - 相手のマークが既に3つ並んでいるか
  - lose(Q,B')∨tie(Q,B')なるBをB'にする手がないとき
- tie(P,B)はwin(P,B)でもlose(P,B)でもないとき

発展

- Prologライクな論理型言語を実装せよ
  - パーサを準備する必要はない
  - カットはなくてよい