Prolog
(9. přednáška)

Syntax prologu — stavební kameny

predikáty začínají malým písmenem, mohou obsahovat číslice a podtržítka

Syntax prologu — stavební kameny

predikáty začínají malým písmenem, mohou obsahovat číslice a podtržítka

funkce začínají malým písmenem, mohou obsahovat číslice a podtržítka

Syntax prologu — stavební kameny

predikáty	začínají malým písmenem, mohou obsahovat číslice a podtržítka
funkce	začínají malým písmenem, mohou obsahovat číslice a podtržítka
konstanty	(nulární predikáty)

Syntax prologu — stavební kameny

predikáty	začínají malým písmenem, mohou obsahovat číslice a podtržítka
funkce	začínají malým písmenem, mohou obsahovat číslice a podtržítka
konstanty	(nulární predikáty)
proměnné	začínají velkým písmenem nebo podtržítkem

Syntax prologu — stavební kameny

predikáty	začínají malým písmenem, mohou obsahovat číslice a podtržítka
funkce	začínají malým písmenem, mohou obsahovat číslice a podtržítka
konstanty	(nulární predikáty)
proměnné	začínají velkým písmenem nebo podtržítkem
spojky	`:-` (implikace)

Syntax prologu — stavební kameny

predikáty	začínají malým písmenem, mohou obsahovat číslice a podtržítka
funkce	začínají malým písmenem, mohou obsahovat číslice a podtržítka
konstanty	(nulární predikáty)
proměnné	začínají velkým písmenem nebo podtržítkem
spojky	`:-` (implikace), `,` (konjunkce)

Syntax prologu — stavební kameny

predikáty	začínají malým písmenem, mohou obsahovat číslice a podtržítka
funkce	začínají malým písmenem, mohou obsahovat číslice a podtržítka
konstanty	(nulární predikáty)
proměnné	začínají velkým písmenem nebo podtržítkem
spojky	`:-` (implikace), `,` (konjunkce), `;` (disjunkce)

Syntax prologu — stavební kameny

predikáty	začínají malým písmenem, mohou obsahovat číslice a podtržítka
funkce	začínají malým písmenem, mohou obsahovat číslice a podtržítka
konstanty	(nulární predikáty)
proměnné	začínají velkým písmenem nebo podtržítkem
spojky	`:-` (implikace), `,` (konjunkce), `;` (disjunkce), `=` (rovnost)

Syntax prologu — statements

Fakta

Syntax prologu — statements

Fakta

```
muz(adam).
zena(eva).
manzel(adam,eva).
rodic(adam,abel).
rodic(eva,abel).
```

Syntax prologu — statements

Fakta

```
muz(adam).
zena(eva).
manzel(adam,eva).
rodic(adam,abel).
rodic(eva,abel).
```

Procedury/Pravidla

Syntax prologu — statements

Fakta

```
muz(adam).
zena(eva).
manzel(adam,eva).
rodic(adam,abel).
rodic(eva,abel).
```

Procedury/Pravidla (`Hlava :- Klauzule `)

Syntax prologu — statements

Fakta

```
muz(adam).
zena(eva).
manzel(adam,eva).
rodic(adam,abel).
rodic(eva,abel).
```

Procedury/Pravidla (`Hlava :- Klauzule`)

```
matka(M,D) :- rodic(M,D), zena(M).
otec(O,D) :- rodic(O,D), muz(O).
```

Syntax prologu — statements

Fakta

```
muz(adam).
zena(eva).
manzel(adam,eva).
rodic(adam,abel).
rodic(eva,abel).
```

Procedury/Pravidla (Hlava :- Klauzule)

```
matka(M,D) :- rodic(M,D), zena(M).
otec(O,D) :- rodic(O,D), muz(O).
sourozenec(X,Y) :- rodic(R,X), rodic(R,Y).
```

Syntax prologu — statements

Fakta

```
muz(adam).
zena(eva).
manzel(adam,eva).
rodic(adam,abel).
rodic(eva,abel).
```

Procedury/Pravidla (Hlava :- Klauzule)

```
matka(M,D) :- rodic(M,D), zena(M).
otec(O,D) :- rodic(O,D), muz(O).
sourozenec(X,Y) :- rodic(R,X), rodic(R,Y).
vlastni_sourozenec(X,Y) :-
     otec(O,X), otec(O,Y),
     matka(M,X), matka(M,Y).
```

Syntax prologu — statements

Fakta

```
muz(adam).
zena(eva).
manzel(adam,eva).
rodic(adam,abel).
rodic(eva,abel).
```

Procedury/Pravidla (Hlava :- Klauzule)

```
matka(M,D) :- rodic(M,D), zena(M).
otec(O,D) :- rodic(O,D), muz(O).
sourozenec(X,Y) :- rodic(R,X), rodic(R,Y).
vlastni_sourozenec(X,Y) :-
     otec(O,X), otec(O,Y),
     matka(M,X), matka(M,Y).
```

Prologovský program je seznam (databáze) faktů a procedur.

Interakce s prologem

- Načteme databázi faktů (`consult`).

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?-
```

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?- consult(genealogy).
```

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?- consult(genealogy).
% genealogy compiled 0.00 sec, 10 clauses
true.

2 ?-
```

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?- consult(genealogy).
% genealogy compiled 0.00 sec, 10 clauses
true.

2 ?- rodic(adam,abel).
```

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?- consult(genealogy).
% genealogy compiled 0.00 sec, 10 clauses
true.

2 ?- rodic(adam,abel).
true.

3 ?-
```

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?- consult(genealogy).
% genealogy compiled 0.00 sec, 10 clauses
true.

2 ?- rodic(adam,abel).
true.

3 ?- rodic(X,abel).
```

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?- consult(genealogy).
% genealogy compiled 0.00 sec, 10 clauses
true.

2 ?- rodic(adam,abel).
true.

3 ?- rodic(X,abel).
X = adam
```

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?- consult(genealogy).
% genealogy compiled 0.00 sec, 10 clauses
true.

2 ?- rodic(adam,abel).
true.

3 ?- rodic(X,abel).
X = adam ;
```

Interakce s prologem

- Načteme databázi faktů (`consult`).
- Klademe dotazy

Příklad interakce

```
1 ?- consult(genealogy).
% genealogy compiled 0.00 sec, 10 clauses
true.

2 ?- rodic(adam,abel).
true.

3 ?- rodic(X,abel).
X = adam ;
X = eva.

4 ?-
```

Hledání důkazu

- unifikace

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
```

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
```

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
        Cíl:  rodic(R,X), rodic(R, Y) / [Y=abel]
```

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
        Cíl:  rodic(R,X), rodic(R, Y) / [Y=abel]
            rodic(R,X) = rodic(adam,kain) / [Y=abel,R=adam,X=kain]
```

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
        Cíl:  rodic(R,X), rodic(R, Y) / [Y=abel]
            rodic(R,X) = rodic(adam,kain) / [Y=abel,R=adam,X=kain]
                Cíl:  rodic(R,Y) / [Y=abel,R=adam,X=kain]
```

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
        Cíl:  rodic(R,X), rodic(R, Y) / [Y=abel]
            rodic(R,X) = rodic(adam,kain) / [Y=abel,R=adam,X=kain]
                Cíl:  rodic(R,Y) / [Y=abel,R=adam,X=kain]
                    rodic(R,Y) = rodic(adam,abel) / [Y=abel,R=adam,X=kain]
```

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
        Cíl:  rodic(R,X), rodic(R, Y) / [Y=abel]
            rodic(R,X) = rodic(adam,kain) / [Y=abel,R=adam,X=kain]
                Cíl:  rodic(R,Y) / [Y=abel,R=adam,X=kain]
                    rodic(R,Y) = rodic(adam,abel) / [Y=abel,R=adam,X=kain]
                    ÚSPĚCH
```

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
        Cíl:  rodic(R,X), rodic(R, Y) / [Y=abel]
            rodic(R,X) = rodic(adam,kain) / [Y=abel,R=adam,X=kain]
                Cíl:  rodic(R,Y) / [Y=abel,R=adam,X=kain]
                    rodic(R,Y) = rodic(adam,abel) / [Y=abel,R=adam,X=kain]
                    ÚSPĚCH
```

- chtěli bychom, aby neplatilo `sourozenec(abel,abel)`;

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
        Cíl:  rodic(R,X), rodic(R, Y) / [Y=abel]
            rodic(R,X) = rodic(adam,kain) / [Y=abel,R=adam,X=kain]
                Cíl:  rodic(R,Y) / [Y=abel,R=adam,X=kain]
                    rodic(R,Y) = rodic(adam,abel) / [Y=abel,R=adam,X=kain]
                    ÚSPĚCH
```

- chtěli bychom, aby neplatilo `sourozenec(abel,abel)`; to však nelze zapsat hornovskou klauzulí;

Hledání důkazu

- unifikace
- prohledávání do hloubky a “backtracking”

```
sourozenec(X,abel)
```

- prochází databázi a snaží se unifikovat sourozenec(X,abel) s “hlavou” nějakého faktu, nebo pravidla (procedury).
- pokud se mu podaří unifikace s faktem, uspěl.
- pokud se mu podaří unifikace s hlavou pravidla, pokusí se rekurzivně provést totéž s jednotlivými klauzulemi pravidla

```
Cíl:  sourozenec(X,abel)
    sourozenec(X,abel) = sourozenec(X,Y) / [Y=abel]
        Cíl:  rodic(R,X), rodic(R, Y) / [Y=abel]
            rodic(R,X) = rodic(adam,kain) / [Y=abel,R=adam,X=kain]
                Cíl:  rodic(R,Y) / [Y=abel,R=adam,X=kain]
                    rodic(R,Y) = rodic(adam,abel) / [Y=abel,R=adam,X=kain]
                    ÚSPĚCH
```

- chtěli bychom, aby neplatilo `sourozenec(abel,abel)`; to však nelze zapsat hornovskou klauzulí; je třeba použít operátor řezu (viz dále).

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

```
predek(adam,tubal)
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

```
predek(adam,tubal)
```

```
Cíl:  predek(adam,tubal)
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

```
predek(adam,tubal)
```

```
Cíl:  predek(adam,tubal)
     predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

```
predek(adam,tubal)
```

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

```
predek(adam,tubal)
```

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(Z,Y), predek(X,Z) / [X=adam,Y=tubal]
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(Z,Y), predek(X,Z) / [X=adam,Y=tubal]
            rodic(Z,Y) = rodic(kain,tubal) / [Z=kain,Y=tubal,X=adam]
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(Z,Y), predek(X,Z) / [X=adam,Y=tubal]
            rodic(Z,Y) = rodic(kain,tubal) / [Z=kain,Y=tubal,X=adam]
                O.K.
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(Z,Y), predek(X,Z) / [X=adam,Y=tubal]
            rodic(Z,Y) = rodic(kain,tubal) / [Z=kain,Y=tubal,X=adam]
                O.K.
            predek(X,Z) = predek(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(Z,Y), predek(X,Z) / [X=adam,Y=tubal]
            rodic(Z,Y) = rodic(kain,tubal) / [Z=kain,Y=tubal,X=adam]
                O.K.
            predek(X,Z) = predek(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
                Cíl:  rodic(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(Z,Y), predek(X,Z) / [X=adam,Y=tubal]
            rodic(Z,Y) = rodic(kain,tubal) / [Z=kain,Y=tubal,X=adam]
                O.K.
            predek(X,Z) = predek(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
                Cíl:  rodic(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
                    rodic(X,Y1) = rodic(adam,kain)
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(Z,Y), predek(X,Z) / [X=adam,Y=tubal]
            rodic(Z,Y) = rodic(kain,tubal) / [Z=kain,Y=tubal,X=adam]
                O.K.
            predek(X,Z) = predek(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
                Cíl:  rodic(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
                    rodic(X,Y1) = rodic(adam,kain)
                        O.K.
```

Rekurze — dobrý sluha

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-rodic(Z,Y),predek(X,Z).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(Z,Y), predek(X,Z) / [X=adam,Y=tubal]
            rodic(Z,Y) = rodic(kain,tubal) / [Z=kain,Y=tubal,X=adam]
                O.K.
            predek(X,Z) = predek(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
                Cíl:  rodic(X,Y1) / [ Z=kain,X=adam, Y1=Z, Y=tubal ]
                    rodic(X,Y1) = rodic(adam,kain)
                        O.K.

            ÚSPĚCH
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

```
predek(adam,tubal)
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

```
predek(adam,tubal)
```

```
Cíl:  predek(adam,tubal)
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

```
predek(adam,tubal)
```

```
Cíl:  predek(adam,tubal)
     predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

```
predek(adam,tubal)
```

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  predek(X,Z1), rodic(Z1,Y) / [X=adam,Y=tubal]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  predek(X,Z1), rodic(Z1,Y) / [X=adam,Y=tubal]
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  predek(X,Z1), rodic(Z1,Y) / [X=adam,Y=tubal]
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
                        Cíl:  rodic(X,Y1) / [ X=adam,Y=tubal ]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  predek(X,Z1), rodic(Z1,Y) / [X=adam,Y=tubal]
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
                        Cíl:  rodic(X,Y1) / [ X=adam,Y=tubal ]
                            FAIL
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  predek(X,Z1), rodic(Z1,Y) / [X=adam,Y=tubal]
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
                        Cíl:  rodic(X,Y1) / [ X=adam,Y=tubal ]
                            FAIL
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  predek(X,Z1), rodic(Z1,Y) / [X=adam,Y=tubal]
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
                        Cíl:  rodic(X,Y1) / [ X=adam,Y=tubal ]
                            FAIL
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
                        Cíl:  predek(X,Z2),rodic(Z2,Y1) / [ X=adam,Y=tubal ]
```

Rekurze — zlý pán

```
predek(X,Y):-rodic(X,Y).
predek(X,Y):-predek(X,Z),rodic(Z,Y).
```

predek(adam,tubal)

```
Cíl:  predek(adam,tubal)
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  rodic(X,Y) / [X=adam,Y=tubal]
            FAIL
    predek(adam,tubal) = predek(X,Y) / [X=adam,Y=tubal]
        Cíl:  predek(X,Z), rodic(Z,Y) / [X=adam,Y=tubal]
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  rodic(X,Y1) / [X=adam,Y=tubal]
                    FAIL
            predek(X,Z) = predek(X,Y1) / [X=adam,Y=tubal]
                Cíl:  predek(X,Z1), rodic(Z1,Y) / [X=adam,Y=tubal]
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
                        Cíl:  rodic(X,Y1) / [ X=adam,Y=tubal ]
                            FAIL
                    predek(X,Z1) = predek(X,Y1) / [ X=adam, Y=tubal ]
                        Cíl:  predek(X,Z2),rodic(Z2,Y1) / [ X=adam,Y=tubal ]

                            ...
```

Seznamy

- `[1,2,3,4,5]`

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Příklad: Predikát býti prvkem.

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Příklad: Predikát býti prvkem.

```
contains(X,[X]).
```

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Příklad: Predikát býti prvkem.

```
contains(X,[X]).
contains(X,[X|_]).
```

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Příklad: Predikát býti prvkem.

```
contains(X,[X]).
contains(X,[X|_]).
contains(X,[_|T]):-contains(X,T).
```

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Příklad: Predikát býti prvkem.

```
contains(X,[X]).
contains(X,[X|_]).
contains(X,[_|T]):-contains(X,T).
```

Cvičení

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Příklad: Predikát býti prvkem.

```
contains(X,[X]).
contains(X,[X|_]).
contains(X,[_|T]):-contains(X,T).
```

Cvičení

1. Predikát konkatenace dvou seznamů.

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Příklad: Predikát býti prvkem.

```
contains(X,[X]).
contains(X,[X|_]).
contains(X,[_|T]):-contains(X,T).
```

Cvičení

1. Predikát konkatenace dvou seznamů.
2. Predikát otočení seznamu.

Seznamy

- `[1,2,3,4,5]`
- `[H|T]`

Příklad: Predikát býti prvkem.

```
contains(X,[X]).
contains(X,[X|_]).
contains(X,[_|T]):-contains(X,T).
```

Cvičení

1. Predikát konkatenace dvou seznamů.
2. Predikát otočení seznamu.
3. Predikát vybrání lichých prvků.

Operátor řezu

- predikát sourozenci nebyl správně

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

```
test(X,Y):-test1(X,Y),!,test2(X,Y)
```

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

```
test(X,Y):-test1(X,Y),!,test2(X,Y)
```

Význam

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

```
test(X,Y):-test1(X,Y),!,test2(X,Y)
```

Význam

- nebacktrackuj přes `!`

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

```
test(X,Y):-test1(X,Y),!,test2(X,Y)
```

Význam

- nebacktrackuj přes `!`
- t.j. pokud v téhle větvi neuspěješ, nesnaž se hlavu splnit jinak

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

```
test(X,Y):-test1(X,Y),!,test2(X,Y)
```

Význam

- nebacktrackuj přes `!`
- t.j. pokud v téhle větvi neuspěješ, nesnaž se hlavu splnit jinak

```
different(X,X):-!,fail.
```

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

```
test(X,Y):-test1(X,Y),!,test2(X,Y)
```

Význam

- nebacktrackuj přes `!`
- t.j. pokud v téhle větvi neuspěješ, nesnaž se hlavu splnit jinak

```
different(X,X):-!,fail.
different(_,_).
```

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

```
test(X,Y):-test1(X,Y),!,test2(X,Y)
```

Význam

- nebacktrackuj přes !
- t.j. pokud v téhle větvi neuspěješ, nesnaž se hlavu splnit jinak

```
different(X,X):-!,fail.
different(_,_).
sourozenec(X,Y) :- rodic(R,X), rodic(R,Y),
different(X,Y).
```

Operátor řezu

- predikát sourozenci nebyl správně
- chtěli bychom klauzuli zajišťující $X \neq Y$
- lze zařídit pomocí operátoru řezu a predikátu `fail`

```
test(X,Y):-test1(X,Y),!,test2(X,Y)
```

Význam

- nebacktrackuj přes !
- t.j. pokud v téhle větvi neuspěješ, nesnaž se hlavu splnit jinak

```
different(X,X):-!,fail.
different(_,_).
sourozenec(X,Y) :- rodic(R,X), rodic(R,Y),
different(X,Y).
```

Robinsonova Aritmetika

1. numerály (za pomoci 0, s)
2. $+, *$
3. $\leq$
4. prvočíselnost
5. rozklad na prvočinitele

NLP — Natural Language Processing

- Rozdílové seznamy
- Sentence –> NounPhrase VerbPhrase
- NounPhrase –> Adjective1, Adjective2, ..., Adjectiven, Noun
- VerbPhrase –> Verb
- VerbPhrase –> Verb NounPhrase

Více . . .

Výuka

Více . . .

Výuka

P. Švarný: Logical Programming

Více . . .

Výuka

P. Švarný: Logical Programming

Internetové zdroje

Více . . .

Výuka

P. Švarný: Logical Programming

Internetové zdroje

`http://www.learnprolognow.org/`

Více . . .

Výuka

P. Švarný: Logical Programming

Internetové zdroje

`http://www.learnprolognow.org/`
`http://www.amzi.com/AdventureInProlog/`

Více . . .

Výuka

P. Švarný: Logical Programming

Internetové zdroje

`http://www.learnprolognow.org/`
`http://www.amzi.com/AdventureInProlog/`
`http://ksvi.mff.cuni.cz/~kryl/prolog.pdf`

Více . . .

Výuka

P. Švarný: Logical Programming

Internetové zdroje

`http://www.learnprolognow.org/`
`http://www.amzi.com/AdventureInProlog/`
`http://ksvi.mff.cuni.cz/~kryl/prolog.pdf`

Knižní zdroje

Více . . .

Výuka

P. Švarný: Logical Programming

Internetové zdroje

`http://www.learnprolognow.org/`
`http://www.amzi.com/AdventureInProlog/`
`http://ksvi.mff.cuni.cz/~kryl/prolog.pdf`

Knižní zdroje

Jirků, P. a kol.: Programování v jazyku Prolog, Praha 1991
Ivan Bratko: Prolog Programming for Artificial Intelligence, 1986