Paradigmata programování 4

Přednáška 9. Usuzování s výroky

verze z 8. dubna 2025

Jan Laštovička

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Obsah

Bezpečností systém

Bezpečností systém

Pravidla:

1. Jestliže je zamčeno a je detekován pohyb, pak je alarm spuštěný.
2. Jestliže je detekován kouř, pak je alarm spuštěný.

Bezpečností systém

Pravidla:

1. Jestliže je zamčeno a je detekován pohyb, pak je alarm spuštěný.
2. Jestliže je detekován kouř, pak je alarm spuštěný.

Pravidla mají tvar implikace, kde

- předpoklad implikace je konjunkce atomických výroků
- a závěr implikace je jediný atomický výrok.

Bezpečností systém

Pravidla:

1. Jestliže je zamčeno a je detekován pohyb, pak je alarm spuštěný.
2. Jestliže je detekován kouř, pak je alarm spuštěný.

Pravidla mají tvar implikace, kde

- předpoklad implikace je konjunkce atomických výroků
- a závěr implikace je jediný atomický výrok.

Fakta:

1. Je zamčeno.
2. Je detekován kouř.

Bezpečností systém

Pravidla:

1. Jestliže je zamčeno a je detekován pohyb, pak je alarm spuštěný.
2. Jestliže je detekován kouř, pak je alarm spuštěný.

Pravidla mají tvar implikace, kde

- předpoklad implikace je konjunkce atomických výroků
- a závěr implikace je jediný atomický výrok.

Fakta:

1. Je zamčeno.
2. Je detekován kouř.

Fakta jsou atomické výroky.

Bezpečností systém

Pravidla:

1. Jestliže je zamčeno a je detekován pohyb, pak je alarm spuštěný.
2. Jestliže je detekován kouř, pak je alarm spuštěný.

Pravidla mají tvar implikace, kde

- předpoklad implikace je konjunkce atomických výroků
- a závěr implikace je jediný atomický výrok.

Fakta:

1. Je zamčeno.
2. Je detekován kouř.

Fakta jsou atomické výroky.

Dotaz:

1. Je alarm spuštěný?

Bezpečností systém

Pravidla:

1. Jestliže je zamčeno a je detekován pohyb, pak je alarm spuštěný.
2. Jestliže je detekován kouř, pak je alarm spuštěný.

Pravidla mají tvar implikace, kde

- předpoklad implikace je konjunkce atomických výroků
- a závěr implikace je jediný atomický výrok.

Fakta:

1. Je zamčeno.
2. Je detekován kouř.

Fakta jsou atomické výroky.

Dotaz:

1. Je alarm spuštěný?

Dotaz je konjunkce atomických výroků.

Obsah

Reprezentace atomických výroků a pravidel

Reprezentace atomických výroků a pravidel

Atomický výrok reprezentujeme necyklickou strukturou tečkových párů.

Reprezentace atomických výroků a pravidel

Atomický výrok reprezentujeme necyklickou strukturou tečkových párů.

Zatím si vystačíme pouze se symboly:

1. alarm je spuštěný ... `alarm-on`
2. místnost je zamčená ... `room-locked`
3. je detekován pohyb ... `move-detected`
4. je detekován kouř ... `smoke-detected`

Reprezentace atomických výroků a pravidel

Atomický výrok reprezentujeme necyklickou strukturou tečkových párů.

Zatím si vystačíme pouze se symboly:

1. alarm je spuštěný ... `alarm-on`
2. místnost je zamčená ... `room-locked`
3. je detekován pohyb ... `move-detected`
4. je detekován kouř ... `smoke-detected`

Pravidlo reprezentujeme seznamem `(<- `*`head`*` . `*`body`*`)`, kde

- *`head`* je atomický výrok (hlava pravidla),
- *`body`* je seznam atomických výroků (tělo pravidla).

Reprezentace atomických výroků a pravidel

Atomický výrok reprezentujeme necyklickou strukturou tečkových párů.

Zatím si vystačíme pouze se symboly:

1. alarm je spuštěný ... `alarm-on`
2. místnost je zamčená ... `room-locked`
3. je detekován pohyb ... `move-detected`
4. je detekován kouř ... `smoke-detected`

Pravidlo reprezentujeme seznamem `(<- head . body)`, kde

- *`head`* je atomický výrok (hlava pravidla),
- *`body`* je seznam atomických výroků (tělo pravidla).

Tedy:

1. Jestliže je zamčeno a je detekován pohyb, pak je alarm spuštěný.
   `... (<- alarm-on room-locked move-detected)`
2. Jestliže je detekován kouř, pak je alarm spuštěný.
   `... (<- alarm-on smoke-detected)`

Reprezentace faktů a dotazů

Reprezentace faktů a dotazů

Pozorování: Konjunkce prázdné množiny výroků je vždy splněna.

Reprezentace faktů a dotazů

Pozorování: Konjunkce prázdné množiny výroků je vždy splněna.

Fakt reprezentujeme jako pravidlo s prázdným tělem.

Reprezentace faktů a dotazů

Pozorování: Konjunkce prázdné množiny výroků je vždy splněna.

Fakt reprezentujeme jako pravidlo s prázdným tělem.

Tedy:

1. Je zamčeno. ... `(<- room-locked)`
2. Je detekován kouř. ... `(<- smoke-detected)`

Reprezentace faktů a dotazů

Pozorování: Konjunkce prázdné množiny výroků je vždy splněna.

Fakt reprezentujeme jako pravidlo s prázdným tělem.

Tedy:

1. Je zamčeno. ... `(<- room-locked)`
2. Je detekován kouř. ... `(<- smoke-detected)`

Dotaz formujeme jako cíl.

Reprezentace faktů a dotazů

Pozorování: Konjunkce prázdné množiny výroků je vždy splněna.

Fakt reprezentujeme jako pravidlo s prázdným tělem.

Tedy:

1. Je zamčeno. ... `(<- room-locked)`
2. Je detekován kouř. ... `(<- smoke-detected)`

Dotaz formujeme jako cíl.

Cíl je seznam `(? subgoal1 subgoal2 ...)`, kde

- *subgoal1*, *subgoal2* ... jsou atomické výroky (podcíle).

Reprezentace faktů a dotazů

Pozorování: Konjunkce prázdné množiny výroků je vždy splněna.

Fakt reprezentujeme jako pravidlo s prázdným tělem.

Tedy:

1. Je zamčeno. ... `(<- room-locked)`
2. Je detekován kouř. ... `(<- smoke-detected)`

Dotaz formujeme jako cíl.

Cíl je seznam `(? subgoal1 subgoal2 ...)`, kde

- *subgoal1*, *subgoal2* ... jsou atomické výroky (podcíle).

Tedy:

- Je alarm spuštěný? ... `(? alarm-on)`

Reprezentace faktů a dotazů

Pozorování: Konjunkce prázdné množiny výroků je vždy splněna.

Fakt reprezentujeme jako pravidlo s prázdným tělem.

Tedy:

1. Je zamčeno. ... `(<- room-locked)`
2. Je detekován kouř. ... `(<- smoke-detected)`

Dotaz formujeme jako cíl.

Cíl je seznam `(? subgoal1 subgoal2 ...)`, kde

- *subgoal1*, *subgoal2* ... jsou atomické výroky (podcíle).

Tedy:

- Je alarm spuštěný? ... `(? alarm-on)`

Cíl bez výroků nazýváme prázdný cíl.

Program

Logický program je seznam pravidel.

Program

Logický program je seznam pravidel.

Bezpečnostní systém v určitém čase:

```
((<- alarm-on room-locked move-detected)
 (<- alarm-on smoke-detected)
 (<- room-locked)
 (<- smoke-detected))
```

Obsah

Úsudek

Úsudek

Na cíl:

- `(? subgoal1 subgoal2 ...)`

lze použít pravidlo:

- `(<- head body1 ... bodyn)`

pokud *subgoal1* je rovno *head*.

Úsudek

Na cíl:

- `(? subgoal1 subgoal2 ...)`

lze použít pravidlo:

- `(<- head body1 ... bodyn)`

pokud *subgoal1* je rovno *head*.

Použitím obdržíme cíl:

- `(? body1 ... bodyn subgoal2 ...)`

Úsudek

Na cíl:

- `(? subgoal1 subgoal2 ...)`

lze použít pravidlo:

- `(<- head body1 ... bodyn)`

pokud *subgoal1* je rovno *head*.

Použitím obdržíme cíl:

- `(? body1 ... bodyn subgoal2 ...)`

Například:

1. Použitím pravidla `(<- alarm-on room-locked move-detected)`
   na `(? alarm-on)`
   obdržíme `(? room-locked move-detected)`.

Úsudek

Na cíl:

- `(? subgoal1 subgoal2 ...)`

lze použít pravidlo:

- `(<- head body1 ... bodyn)`

pokud *subgoal1* je rovno *head*.

Použitím obdržíme cíl:

- `(? body1 ... bodyn subgoal2 ...)`

Například:

1. Použitím pravidla `(<- alarm-on room-locked move-detected)`
   na `(? alarm-on)`
   obdržíme `(? room-locked move-detected)`.
2. Použitím pravidla `(<- room-locked)`
   na `(? room-locked move-detected)`
   obdržíme `(? move-detected)`.

Strom úsudků

Strom úsudků

Strom úsudků (také rezoluční strom) pro dotaz *query* vzhledem k programu *rules* je strom, kde platí následující.

Strom úsudků

Strom úsudků (také rezoluční strom) pro dotaz *query* vzhledem k programu *rules* je strom, kde platí následující.

- Uzly stromu jsou ohodnoceny cíli.

Strom úsudků

Strom úsudků (také rezoluční strom) pro dotaz *query* vzhledem k programu *rules* je strom, kde platí následující.

- Uzly stromu jsou ohodnoceny cíli.
- Kořen stromu má cíl *query*.

Strom úsudků

Strom úsudků (také rezoluční strom) pro dotaz *query* vzhledem k programu *rules* je strom, kde platí následující.

- Uzly stromu jsou ohodnoceny cíli.
- Kořen stromu má cíl *query*.
- Uzel stromu má tolik následníků, kolik pravidel z *rules* lze použít na jeho cíl.

Strom úsudků

Strom úsudků (také rezoluční strom) pro dotaz $query$ vzhledem k programu $rules$ je strom, kde platí následující.

- Uzly stromu jsou ohodnoceny cíli.
- Kořen stromu má cíl $query$.
- Uzel stromu má tolik následníků, kolik pravidel z $rules$ lze použít na jeho cíl.
- Pokud lze použít pravidlo $rule$ z $rules$ na cíl $goal$ uzlu stromu, pak má uzel následníka s cílem rovným výsledku použití pravidla $rule$ na $goal$.

Příklad stromu úsudků

Program:

```
((<- alarm-on room-locked move-detected)
 (<- alarm-on smoke-detected)
 (<- room-locked)
 (<- smoke-detected))
```

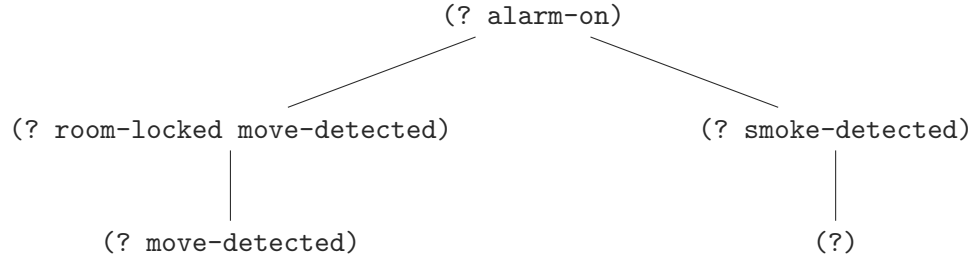
Strom pro `(? alarm-on)`:

Příklad stromu úsudků

Program:

```
((<- alarm-on room-locked move-detected)
 (<- alarm-on smoke-detected)
 (<- room-locked)
 (<- smoke-detected))
```

Strom pro `(? alarm-on)`:



Příklad stromu úsudků

Program:

```
((<- danger alarm-on)
 (<- alarm-on danger))
```

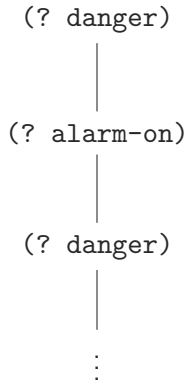
Strom úsudků pro `(? danger)`

Příklad stromu úsudků

Program:

```
((<- danger alarm-on)
 (<- alarm-on danger))
```

Strom úsudků pro `(? danger)` je nekonečný:



Splnění cíle

Splnění cíle

Cíl *query* je vzhledem k programu *rules* splněn, jestliže jeho strom úsudků obsahuje prázdný cíl.

Splnění cíle

Cíl *query* je vzhledem k programu *rules* splněn,
jestliže jeho strom úsudků obsahuje prázdný cíl.

Například cíl `(? alarm-on)` je vzhledem k

```
((<- alarm-on room-locked move-detected)
 (<- alarm-on smoke-detected)
 (<- room-locked)
 (<- smoke-detected))
```

splněný,

Splnění cíle

Cíl *query* je vzhledem k programu *rules* splněn,
jestliže jeho strom úsudků obsahuje prázdný cíl.

Například cíl `(? alarm-on)` je vzhledem k

```
((<- alarm-on room-locked move-detected)
 (<- alarm-on smoke-detected)
 (<- room-locked)
 (<- smoke-detected))
```

splněný, ale `(? room-locked move-detected)` není.

Splnění cíle

Cíl *query* je vzhledem k programu *rules* splněn,
jestliže jeho strom úsudků obsahuje prázdný cíl.

Například cíl `(? alarm-on)` je vzhledem k

```
((<- alarm-on room-locked move-detected)
 (<- alarm-on smoke-detected)
 (<- room-locked)
 (<- smoke-detected))
```

splněný, ale `(? room-locked move-detected)` není.

Dále cíl `(? danger)` vzhledem k

```
((<- danger alarm-on)
 (<- alarm-on danger))
```

Splnění cíle

Cíl *query* je vzhledem k programu *rules* splněn,
jestliže jeho strom úsudků obsahuje prázdný cíl.

Například cíl `(? alarm-on)` je vzhledem k

```
((<- alarm-on room-locked move-detected)
 (<- alarm-on smoke-detected)
 (<- room-locked)
 (<- smoke-detected))
```

splněný, ale `(? room-locked move-detected)` není.

Dále cíl `(? danger)` vzhledem k

```
((<- danger alarm-on)
 (<- alarm-on danger))
```

splněný není.

Obsah

Cíl

- (? . *subgoals*)

```
(defun goal (subgoals)
  (cons '? subgoals))

(defun goal-subgoals (goal)
  (cdr goal))

(defun empty-goal-p (goal)
  (null (goal-subgoals goal)))
```

Pravidlo

- (<- *head* . *body*)

```
(defun rule (head body)
  `(<- ,head ,@body))

(defun rule-head (rule)
  (cadr rule))

(defun rule-body (rule)
  (cddr rule))
```

Úsudek

Je pravidlo použitelné na cíl?

```
(defun applicable-rule-p (rule goal)
  (and (not (empty-goal-p goal))
       (equalp (rule-head rule)
               (car (goal-subgoals goal)))))
```

Výsledek použití pravidla:

```
(defun rule-resolvent (rule goal)
  (goal (append (rule-body rule)
                (cdr (goal-subgoals goal)))))
```

. . . pravidlo musí jít použít.

Nedeterminismus

Nedeterminismus

Použitelná pravidla:

```
(defun applicable-rules (goal rules)
  (remove-if-not (lambda (rule)
                   (applicable-rule-p rule goal))
                 rules))
```

Nedeterminismus

Použitelná pravidla:

```
(defun applicable-rules (goal rules)
  (remove-if-not (lambda (rule)
                   (applicable-rule-p rule goal))
                 rules))
```

Výsledky použití možných pravidel:

```
(defun goal-resolvents (goal rules)
  (mapcar (lambda (rule)
            (rule-resolvent rule goal))
          (applicable-rules goal rules)))
```

Stromy

Uzel stromu:

- (node *value* . *children*)

```
(defun tree-node (value children)
  (cons 'node (cons value children)))

(defun node-value (node)
  (cadr node))

(defun node-children (node)
  (cddr node))
```

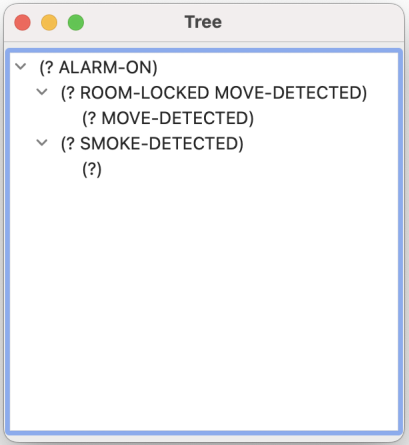
Strom úsudků

```
(defun resolution-tree (goal rules)
  (tree-node goal
             (mapcar (lambda (g)
                       (resolution-tree g rules))
                     (goal-resolvents goal rules))))
```

```
(NODE (? ALARM-ON)
      (NODE (? ROOM-LOCKED MOVE-DETECTED)
            (NODE (? MOVE-DETECTED)))
      (NODE (? SMOKE-DETECTED)
            (NODE (?))))
```

Zobrazení stromu

Funkce `(view-tree` *`tree`*`)`



Prohledávání do hloubky

```
(defun tree-values-dfs (root)
  (cons (node-value root)
        (reduce #'append
                (mapcar #'tree-values-dfs
                        (node-children root))
                :initial-value '())))
```

Prohledávání do šířky

```
(defun tree-values-bfs (root)
  (tvb (list root)))

(defun tvb (roots)
  (if (null roots)
      '()
    (append (mapcar #'node-value roots)
            (tvb (reduce #'append
                         (mapcar #'node-children roots)
                         :initial-value '())))))
```

Splnění cíle

```
(defun fulfilledp (goal rules &optional breadthp)
  (find-if #'empty-goal-p
           (funcall (if breadthp
                        #'tree-values-bfs
                      #'tree-values-dfs)
                    (resolution-tree goal rules))))
```

Splnění cíle

```
(defun fulfilledp (goal rules &optional breadthp)
  (find-if #'empty-goal-p
           (funcall (if breadthp
                        #'tree-values-bfs
                      #'tree-values-dfs)
                    (resolution-tree goal rules))))
```

Nefunguje na nekonečné stromy:

```
(fulfilledp '(? danger)
            '((<- danger alarm-on)
              (<- alarm-on danger)))
```

Obsah

Pravidla ve výrokové logice

Pravidla ve výrokové logice

Atomické výroky jsou výrokové symboly.

Pravidla ve výrokové logice

Atomické výroky jsou výrokové symboly.

Příklad:

1. alarm je spuštěný ... p_a
2. místnost je zamčená ... p_l
3. je detekován pohyb ... p_m
4. je detekován kouř ... p_s

Pravidla ve výrokové logice

Atomické výroky jsou výrokové symboly.

Příklad:

1. alarm je spuštěný ... p_a
2. místnost je zamčená ... p_l
3. je detekován pohyb ... p_m
4. je detekován kouř ... p_s

Pravidlo (<- p q_1 ... q_n) je formule $(q_1 \wedge \cdots \wedge q_n) \rightarrow p$.

Pravidla ve výrokové logice

Atomické výroky jsou výrokové symboly.

Příklad:

1. alarm je spuštěný ... p_a
2. místnost je zamčená ... p_l
3. je detekován pohyb ... p_m
4. je detekován kouř ... p_s

Pravidlo (<- p q_1 ... q_n) je formule $(q_1 \wedge \cdots \wedge q_n) \rightarrow p$.

Příklad:

1. Jestliže je zamčeno a je detekován pohyb, pak je alarm spuštěný.
   ... $(p_l \wedge p_m) \rightarrow p_a$
2. Jestliže je detekován kouř, pak je alarm spuštěný.
   ... $p_s \rightarrow p_a$

Program a cíl ve výrokové logice

Program a cíl ve výrokové logice

Fakt (<- p) je formule $(q_1 \wedge \cdots \wedge q_0) \to p = 1 \to p \equiv p$.

Program a cíl ve výrokové logice

Fakt (<- p) je formule $(q_1 \wedge \cdots \wedge q_0) \rightarrow p = 1 \rightarrow p \equiv p$.

Příklad:

1. Místnost je zamčená ... p_l
2. Je detekován kouř ... p_s

Program a cíl ve výrokové logice

Fakt (<- p) je formule $(q_1 \wedge \cdots \wedge q_0) \to p = 1 \to p \equiv p$.

Příklad:

1. Místnost je zamčená ... p_l
2. Je detekován kouř ... p_s

Program je konečná množina formulí T (teorie).

Program a cíl ve výrokové logice

Fakt (<- p) je formule $(q_1 \wedge \cdots \wedge q_0) \to p = 1 \to p \equiv p$.

Příklad:

1. Místnost je zamčená ... p_l
2. Je detekován kouř ... p_s

Program je konečná množina formulí T (teorie).

Příklad:

$$T = \{(p_l \wedge p_m) \to p_a, p_s \to p_a, p_l, p_s\}$$

Program a cíl ve výrokové logice

Fakt (`<-` p) je formule $(q_1 \wedge \cdots \wedge q_0) \to p = 1 \to p \equiv p$.

Příklad:

1. Místnost je zamčená ... p_l
2. Je detekován kouř ... p_s

Program je konečná množina formulí T (teorie).

Příklad:

$$T = \{(p_l \wedge p_m) \to p_a, p_s \to p_a, p_l, p_s\}$$

Cíl (`?` p_1 `...` p_n) je formule $\varphi = p_1 \wedge \cdots \wedge p_n$.

Program a cíl ve výrokové logice

Fakt (`<-` p) je formule $(q_1 \wedge \cdots \wedge q_0) \to p = 1 \to p \equiv p$.

Příklad:

1. Místnost je zamčená ... p_l
2. Je detekován kouř ... p_s

Program je konečná množina formulí T (teorie).

Příklad:

$$T = \{(p_l \wedge p_m) \to p_a, p_s \to p_a, p_l, p_s\}$$

Cíl (`?` p_1 `...` p_n) je formule $\varphi = p_1 \wedge \cdots \wedge p_n$.

Příklad:

- Je alarm spuštěný? ... $\varphi = p_a$

Rozhodnutí sémantického vyplývání

Rozhodnutí sémantického vyplývání

Chceme výpočtem rozhodnout, zda $T \models \varphi$.

Rozhodnutí sémantického vyplývání

Chceme výpočtem rozhodnout, zda $T \vDash \varphi$.

Příklad:

$$\{(p_l \wedge p_m) \to p_a, (p_l \wedge p_s) \to p_a, p_l, p_s\} \models p_a$$

Rozhodnutí sémantického vyplývání

Chceme výpočtem rozhodnout, zda $T \models \varphi$.

Příklad:

$$\{(p_l \wedge p_m) \to p_a, (p_l \wedge p_s) \to p_a, p_l, p_s\} \models p_a$$

Použitím pravidla $(q_1 \wedge \cdots \wedge q_n) \to p_1 \in T$
na cíl $\varphi_1 = p_1 \wedge \cdots \wedge p_m$
obdržíme cíl $\varphi_2 = q_1 \wedge \cdots \wedge q_n \wedge p_2 \wedge \cdots \wedge p_m$.

Rozhodnutí sémantického vyplývání

Chceme výpočtem rozhodnout, zda $T \models \varphi$.

Příklad:

$$\{(p_l \wedge p_m) \to p_a, (p_l \wedge p_s) \to p_a, p_l, p_s\} \models p_a$$

Použitím pravidla $(q_1 \wedge \cdots \wedge q_n) \to p_1 \in T$
na cíl $\varphi_1 = p_1 \wedge \cdots \wedge p_m$
obdržíme cíl $\varphi_2 = q_1 \wedge \cdots \wedge q_n \wedge p_2 \wedge \cdots \wedge p_m$.

Platí:

Pokud $T \models \varphi_2$, *pak* $T \models \varphi_1$.

Rozhodnutí sémantického vyplývání

Chceme výpočtem rozhodnout, zda $T \models \varphi$.

Příklad:

$$\{(p_l \wedge p_m) \to p_a, (p_l \wedge p_s) \to p_a, p_l, p_s\} \models p_a$$

Použitím pravidla $(q_1 \wedge \cdots \wedge q_n) \to p_1 \in T$
na cíl $\varphi_1 = p_1 \wedge \cdots \wedge p_m$
obdržíme cíl $\varphi_2 = q_1 \wedge \cdots \wedge q_n \wedge p_2 \wedge \cdots \wedge p_m$.

Platí:

Pokud $T \models \varphi_2$, pak $T \models \varphi_1$.

Platí:

Cíl φ je vzhledem k T splněný, právě když $T \models \varphi$.