Paradigmata programování 4

Přednáška 10. Proměnné v pravidlech

verze z 15. dubna 2025

Jan Laštovička

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Obsah

Motivace

Motivace

Fakta:

1. Abel je dítětem Evy.
2. Abel je dítětem Adama.
3. Adam je muž.
4. Abel je muž.
5. Eva je žena.

Motivace

Fakta:

1. Abel je dítětem Evy.
2. Abel je dítětem Adama.
3. Adam je muž.
4. Abel je muž.
5. Eva je žena.

Pravidlo:

- Jestliže y je dítětem x a x je muž, pak x je otcem y.

Motivace

Fakta:

1. Abel je dítětem Evy.
2. Abel je dítětem Adama.
3. Adam je muž.
4. Abel je muž.
5. Eva je žena.

Pravidlo:

- Jestliže y je dítětem x a x je muž, pak x je otcem y.

Dotaz: Je Adam otcem Abela?

Atomické výrokové formy

Atomické výrokové formy

Objekt popíšeme necyklickou strukturou tečkových párů.

Atomické výrokové formy

Objekt popíšeme necyklickou strukturou tečkových párů.

Zatím si vystačíme se symboly:

- Adam ... `adam`
- Eva ... `eva`
- Abel ... `abel`

Atomické výrokové formy

Objekt popíšeme necyklickou strukturou tečkových párů.

Zatím si vystačíme se symboly:

- Adam ... `adam`
- Eva ... `eva`
- Abel ... `abel`

Atomickou výrokovou formu zapíšeme seznamem (*`predicate arg1 arg2 ...`*), kde:

- *`predicate`* je predikát
- *`argi`* jsou objekty zvané argumenty

Atomické výrokové formy

Objekt popíšeme necyklickou strukturou tečkových párů.

Zatím si vystačíme se symboly:

- Adam ... `adam`
- Eva ... `eva`
- Abel ... `abel`

Atomickou výrokovou formu zapíšeme seznamem (*`predicate arg1 arg2 ...`*), kde:

- *`predicate`* je predikát
- *`argi`* jsou objekty zvané argumenty

Například:

1. Abel je dítětem Evy ... `(child abel eva)`
2. Eva je otcem Abela ... `(father eva abel)`
3. Adam je muž ... `(male adam)`
4. Abel je žena ... `(female abel)`

Atomické výroky

Atomické výroky

Proměnná je symbol začínající otazníkem,
za kterým následuje aspoň jeden další znak.

Atomické výroky

Proměnná je symbol začínající otazníkem,
za kterým následuje aspoň jeden další znak.

Například: `?x`, `?person`

Atomické výroky

Proměnná je symbol začínající otazníkem,
za kterým následuje aspoň jeden další znak.

Například: `?x`, `?person`

Atomické výrokové formy mohou obsahovat proměnné.

Atomické výroky

Proměnná je symbol začínající otazníkem,
za kterým následuje aspoň jeden další znak.

Například: `?x`, `?person`

Atomické výrokové formy mohou obsahovat proměnné.

Například:

- `?y` je dítětem `?x` ... `(child ?y ?x)`
- `?x` je muž ... `(male ?x)`
- `?x` je otcem `?y` ... `(father ?x ?y)`

Atomické výroky

Proměnná je symbol začínající otazníkem,
za kterým následuje aspoň jeden další znak.

Například: `?x`, `?person`

Atomické výrokové formy mohou obsahovat proměnné.

Například:

- `?y` je dítětem `?x` ... `(child ?y ?x)`
- `?x` je muž ... `(male ?x)`
- `?x` je otcem `?y` ... `(father ?x ?y)`

Atomický výrok je

- atomická výroková forma bez proměnných.

Například: `(child abel eva)`

Atomické výroky

Proměnná je symbol začínající otazníkem,
za kterým následuje aspoň jeden další znak.

Například: `?x`, `?person`

Atomické výrokové formy mohou obsahovat proměnné.

Například:

- `?y` je dítětem `?x` ... `(child ?y ?x)`
- `?x` je muž ... `(male ?x)`
- `?x` je otcem `?y` ... `(father ?x ?y)`

Atomický výrok je

- atomická výroková forma bez proměnných.

Například: `(child abel eva)`

Atomy lispu, které nejsou proměnné, nazýváme konstanty.

Pravidla

Pravidla

Pravidlo je výrok tvaru:

- Pro každé $var1, \ldots, varm$ platí, že jestliže $body1$ a $\ldots$ a $bodyn$, pak $head$,

kde

- $bodyi$ a $head$ jsou atomické výrokové formy
- a $vari$ jsou všechny proměnné, které se v nich vyskytují.

Pravidla

Pravidlo je výrok tvaru:

- Pro každé *var1*, ..., *varm* platí, že jestliže *body1* a ... a *bodyn*, pak *head*,

kde

- *bodyi* a *head* jsou atomické výrokové formy
- a *vari* jsou všechny proměnné, které se v nich vyskytují.

Opět zapisujeme: (<- *head body1 ... bodyn*)

Pravidla

Pravidlo je výrok tvaru:

- Pro každé *var1*, ..., *varm* platí, že jestliže *body1* a ... a *bodyn*, pak *head*,

kde

- *bodyi* a *head* jsou atomické výrokové formy
- a *vari* jsou všechny proměnné, které se v nich vyskytují.

Opět zapisujeme: (<- *head body1 ... bodyn*)

Omezení: Proměnné v těle pravidla se musí vyskytovat v hlavě pravidla.

Pravidla

Pravidlo je výrok tvaru:

- Pro každé *var1*, ..., *varm* platí, že jestliže *body1* a ... a *bodyn*, pak *head*,

kde

- *bodyi* a *head* jsou atomické výrokové formy
- a *vari* jsou všechny proměnné, které se v nich vyskytují.

Opět zapisujeme: `(<- head body1 ... bodyn)`

Omezení: Proměnné v těle pravidla se musí vyskytovat v hlavě pravidla. Například:

- Pro každé `?x` a `?y` platí, že jestliže `?y` je dítětem `?x` a `?x` je muž, pak `?x` je otcem `?y`.

```
... (<- (father ?x ?y) (child ?y ?x) (male ?x))
```

Cíle

Cíle

Cíl má stále tvar:

- *atom1* a ... a *atomn*

Kde:

- *atomi* jsou atomické výroky

Cíle

Cíl má stále tvar:

- *atom1* a ... a *atomn*

Kde:

- *atomi* jsou atomické výroky

Zapíšeme: `(? atom1 ... atomn)`

Cíle

Cíl má stále tvar:

- *atom1* a ... a *atomn*

Kde:

- *atomi* jsou atomické výroky

Zapíšeme: `(? atom1 ... atomn)`

Například:

- Adam je otcem Abela.
  ... `(? (father adam abel))`

Cíle

Cíl má stále tvar:

- *atom1* a ... a *atomn*

Kde:

- *atomi* jsou atomické výroky

Zapíšeme: `(? atom1 ... atomn)`

Například:

- Adam je otcem Abela.
  ... `(? (father adam abel))`

Cíl tedy nesmí obsahovat proměnné.

Obsah

Substituce

Substituce

Substituce je seznam: `(sub (`*var1* `.` *val1*`) ... (`*varn* `.` *valn*`))`
Kde:

- *vari* jsou proměnné
- *vali* jsou objekty

Substituce

Substituce je seznam: `(sub (`*var1* `.` *val1*`) ... (`*varn* `.` *valn*`))`
Kde:

- *vari* jsou proměnné
- *vali* jsou objekty

Podmínky:

1. proměnné v *car* párů se neopakují
2. proměnné z *car* párů se nevyskytují v *cdr* žádného páru

Substituce

Substituce je seznam: `(sub (`*var1* `.` *val1*`) ... (`*varn* `.` *valn*`))`
Kde:

- *vari* jsou proměnné
- *vali* jsou objekty

Podmínky:

1. proměnné v *car* párů se neopakují
2. proměnné z *car* párů se nevyskytují v *cdr* žádného páru

Příklady:

1. `(sub)...` prázdná substituce
2. `(sub (?x . abel))`
3. `(sub (?x . abel) (?y . ?z))`

Aplikace substituce

Aplikace substituce

Výrazy:

1. proměnné
2. konstanty
3. páry

Aplikace substituce

Výrazy:

1. proměnné
2. konstanty
3. páry

Aplikace substituce *sub* na výraz *expr*:

1. je-li *expr* konstanta, výsledkem je *expr*,
2. je-li *expr* proměnná, zjistí se, zda není rovna některé proměnné *vari* substituce *sub*. Pokud ano, výsledkem je *vali*, pokud ne, je jím *expr*,
3. je-li *expr* pár `(a1 . a2)`, výsledkem je pár `(b1 . b2)`, kde každé *bi* vzniklo rekurzivně aplikací substituce *sub* na *ai*.

Aplikace substituce

Výrazy:

1. proměnné
2. konstanty
3. páry

Aplikace substituce *sub* na výraz *expr*:

1. je-li *expr* konstanta, výsledkem je *expr*,
2. je-li *expr* proměnná, zjistí se, zda není rovna některé proměnné *vari* substituce *sub*. Pokud ano, výsledkem je *vali*, pokud ne, je jím *expr*,
3. je-li *expr* pár `(`*a1* `.` *a2*`)`, výsledkem je pár `(`*b1* `.` *b2*`)`, kde každé *bi* vzniklo rekurzivně aplikací substituce *sub* na *ai*.

Například výsledkem aplikace `(sub (?x . abel))`

- na `(male ?x)` je `(male abel)`,
- na `(father ?x ?y)` je `(father abel ?y)`
- a na `(father ?x ?x)` je `(father abel abel)`.

Instance výrazů

Instance výrazů

Substituce *sub* je unifikátor výrazů *expr1* a *expr2*,

- jestliže aplikací *sub* na *expr1* a *expr2* dostaneme týž výraz.

Instance výrazů

Substituce *sub* je unifikátor výrazů *expr1* a *expr2*,

- jestliže aplikací *sub* na *expr1* a *expr2* dostaneme týž výraz.

Výraz *expr1* je instancí výrazu *expr2*,

- jestliže existuje substituce *sub* taková,
- že *expr1* je výsledek aplikace *sub* na *expr2*.

Instance výrazů

Substituce *sub* je unifikátor výrazů *expr1* a *expr2*,

- jestliže aplikací *sub* na *expr1* a *expr2* dostaneme týž výraz.

Výraz *expr1* je instancí výrazu *expr2*,

- jestliže existuje substituce *sub* taková,
- že *expr1* je výsledek aplikace *sub* na *expr2*.

Například:

- `(father adam abel)` je instancí `(father ?x ?y)`
- unifikátor: `(sub (?x . adam) (?y . abel))`

Je výraz instance?

Je výraz instance?

Rozhodneme,

- zda je výraz bez proměnných *expr1* instancí výrazu *expr2*,
- na který jsme aplikovali substituci *sub*,
- a v kladném případě vrátíme jejich unifikátor.

Je výraz instance?

Rozhodneme,

- zda je výraz bez proměnných *expr1* instancí výrazu *expr2*,
- na který jsme aplikovali substituci *sub*,
- a v kladném případě vrátíme jejich unifikátor.

Postup rozhodování, zda je *expr1* instancí *expr2* se substitucí *sub*:

1. Vytvoříme nový výraz *expr2'* vzniklý aplikací *sub* na *expr2*.
2. Jestliže *expr1* = *expr2'*, výsledkem je substituce *sub*.
3. Je-li *expr2'* proměnná, výsledkem je substituce *sub* s přidaným párem (*expr2'* . *expr1*).
4. Jsou-li *expr1* a *expr2'* páry, výsledkem je substituce vzniklá při rozhodování, zda pár *expr1* je instancí páru *expr2'* se *sub*.
5. Jinak zamítáme.

Je výraz instance?

Rozhodneme,

- zda je výraz bez proměnných *expr1* instancí výrazu *expr2*,
- na který jsme aplikovali substituci *sub*,
- a v kladném případě vrátíme jejich unifikátor.

Postup rozhodování, zda je *expr1* instancí *expr2* se substitucí *sub*:

1. Vytvoříme nový výraz *expr2'* vzniklý aplikací *sub* na *expr2*.
2. Jestliže *expr1* = *expr2'*, výsledkem je substituce *sub*.
3. Je-li *expr2'* proměnná, výsledkem je substituce *sub* s přidaným párem (*expr2'* . *expr1*).
4. Jsou-li *expr1* a *expr2'* páry, výsledkem je substituce vzniklá při rozhodování, zda pár *expr1* je instancí páru *expr2'* se *sub*.
5. Jinak zamítáme.

Za *sub* můžeme vzít prázdnou substituci.

Přidání páru

Přidání páru

Přidání páru (*var* . *val*) k substituci *sub* probíhá takto:

1. zkontroluje se, zda *var* není rovno některému *vari* ze substituce.
   Pokud je, je to chyba,
2. zkontroluje se, zda se *var* nevyskytuje v *val*.
   Pokud se vyskytuje, je to chyba,
3. na všechna *vali* se aplikuje substituce (sub (*var* . *val*)),
4. pár se přidá zepředu k *sub*.

Přidání páru

Přidání páru (*var* . *val*) k substituci *sub* probíhá takto:

1. zkontroluje se, zda *var* není rovno některému *vari* ze substituce.
   Pokud je, je to chyba,
2. zkontroluje se, zda se *var* nevyskytuje v *val*.
   Pokud se vyskytuje, je to chyba,
3. na všechna *vali* se aplikuje substituce (sub (*var* . *val*)),
4. pár se přidá zepředu k *sub*.

Například přidáním páru (?x . abel) k substituci (sub (?y . ?x)) vznikne:

- (sub (?x . abel) (?y . abel))

Je pár instance?

Je pár instance?

Postup rozhodování, zda pár (*a1 . a2*) je instancí páru (*b1 . b2*) se substitucí *sub*:

1. Rozhodneme, zda *a1* je instancí *b1* se *sub*, pokud ano, obdržíme substituci *sub2*, jinak zamítáme.
2. Rozhodneme, zda *a2* je instancí *b2* se *sub2*, pokud ano, obdržíme substituci *sub3*, jinak zamítáme. Výsledkem je substituce *sub3*.

Je pár instance?

Postup rozhodování, zda pár (*a1* . *a2*) je instancí páru (*b1* . *b2*) se substitucí *sub*:

1. Rozhodneme, zda *a1* je instancí *b1* se *sub*, pokud ano, obdržíme substituci *sub2*, jinak zamítáme.
2. Rozhodneme, zda *a2* je instancí *b2* se *sub2*, pokud ano, obdržíme substituci *sub3*, jinak zamítáme. Výsledkem je substituce *sub3*.

Například:

1. `(a . a)` je instancí `(?x . ?x)`
   unifikátor: `(sub (?x . a))`
2. ale `(a . b)` instancí `(?x . ?x)` není

Obsah

Programy

Programy

Program je stále seznam pravidel.

Programy

Program je stále seznam pravidel.

Například:

```
((<- (child abel eva))
 (<- (child abel adam))
 (<- (male adam))
 (<- (male abel))
 (<- (female eva))
 (<- (father ?x ?y) (child ?y ?x) (male ?x)))
```

Úsudek

Úsudek

Na cíl:

- `(? atom1 ... atomm)`

lze použít pravidlo:

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

pokud *atom1* je instancí *head*.

Úsudek

Na cíl:

- `(? atom1 ... atomm)`

lze použít pravidlo:

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

pokud *atom1* je instancí *head*.

Použitím obdržíme cíl:

```
(? body1' ... bodyn' atom2 ... atomm)
```

Kde:

- *unifier* je unifikátor *atom1* a *head*
- *bodyi'* je výsledek aplikace *unifier* na *bodyi*

Příklady úsudků

Příklady úsudků

Na cíl `(? (father adam abel))`
lze použít pravidlo: `(<- (father ?x ?y) (child ?y ?x) (male ?x))`
unifikátor: `(sub (?x . adam) (?y . abel))`
výsledek: `(? (child abel adam) (male adam))`

Příklady úsudků

Na cíl `(? (father adam abel))`
lze použít pravidlo: `(<- (father ?x ?y) (child ?y ?x) (male ?x))`
unifikátor: `(sub (?x . adam) (?y . abel))`
výsledek: `(? (child abel adam) (male adam))`

Na cíl `(? (child abel adam) (male adam))`
lze použít pravidlo: `(<- (child abel adam))`
unifikátor: `(sub)`
výsledek: `(? (male adam))`

Strom úsudků a splnění cíle

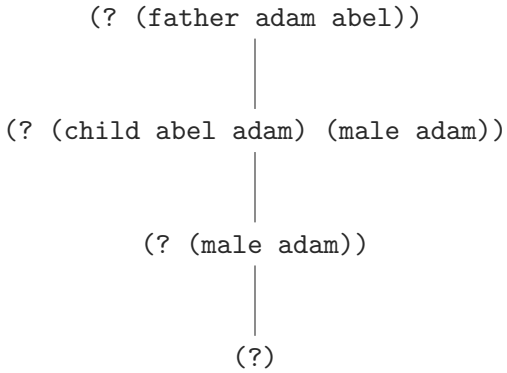
Strom úsudků a splnění cíle

Definice zůstávají stejné jako v minulé přednášce.

Strom úsudků a splnění cíle

Definice zůstávají stejné jako v minulé přednášce.

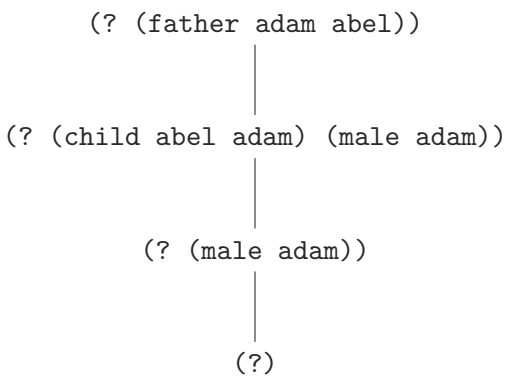
Například:



Strom úsudků a splnění cíle

Definice zůstávají stejné jako v minulé přednášce.

Například:



Cíl `(? (father adam abel))` je splněn.

Obsah

Objekty a predikáty

Objekty a predikáty

Proměnným odpovídají objektové proměnné.

Objekty a predikáty

Proměnným odpovídají objektové proměnné.

Objektům odpovídají termy.

Objekty a predikáty

Proměnným odpovídají objektové proměnné.

Objektům odpovídají termy.

Například:

- `adam` ... $adam$
- `eva` ... eva
- `abel` ... $abel$

Objekty a predikáty

Proměnným odpovídají objektové proměnné.

Objektům odpovídají termy.

Například:

- `adam` ... $adam$
- `eva` ... eva
- `abel` ... $abel$

Predikátům odpovídají relační symboly.

Objekty a predikáty

Proměnným odpovídají objektové proměnné.

Objektům odpovídají termy.

Například:

- `adam` ... $adam$
- `eva` ... eva
- `abel` ... $abel$

Predikátům odpovídají relační symboly.

Například:

- `father` ... $father$
- `male` ... $male$
- `female` ... $female$
- `child` ... $child$

Atomické výrokové formy a pravidla

Atomické výrokové formy a pravidla

Atomickým výrokovým formám odpovídají atomické formule.

Atomické výrokové formy a pravidla

Atomickým výrokovým formám odpovídají atomické formule.

Například:

- `(father adam abel)` ... $father(adam, abel)$
- `(male ?x)` ... $male(x)$

Atomické výrokové formy a pravidla

Atomickým výrokovým formám odpovídají atomické formule.

Například:

- `(father adam abel)` ... $father(adam, abel)$
- `(male ?x)` ... $male(x)$

Pravidlu `(<-` φ ψ_1 `...` ψ_n`)` odpovídá formule $(\forall x_1, \ldots, x_m)((\psi_1 \wedge \cdots \wedge \psi_n) \rightarrow \varphi)$, kde $x_1, \ldots, x_m$ jsou všechny proměnné v φ, ψ_1, ..., ψ_n.

Atomické výrokové formy a pravidla

Atomickým výrokovým formám odpovídají atomické formule.

Například:

- `(father adam abel)` ... $father(adam, abel)$
- `(male ?x)` ... $male(x)$

Pravidlu `(<-` φ ψ_1 `...` ψ_n`)` odpovídá formule $(\forall x_1, \ldots, x_m)((\psi_1 \wedge \cdots \wedge \psi_n) \rightarrow \varphi)$, kde $x_1, \ldots, x_m$ jsou všechny proměnné v φ, ψ_1, ..., ψ_n.

Například:

- `(<- (father ?x ?y) (child ?y ?x) (male ?x))`
  ... $(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))$

Program a cíl

Program a cíl

Programu odpovídá konečná množina formulí (teorie).

Program a cíl

Programu odpovídá konečná množina formulí (teorie).

$$
\begin{aligned}
T = \{&child(abel, eva), child(abel, adam),\\
&male(adam), male(abel), female(eva),\\
&(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\}
\end{aligned}
$$

Program a cíl

Programu odpovídá konečná množina formulí (teorie).

$$\begin{aligned} T = \{&child(abel, eva), child(abel, adam),\\ &male(adam), male(abel), female(eva),\\ &(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\} \end{aligned}$$

Cíli (? ψ_1 ... ψ_n) odpovídá formule $\varphi = \psi_1 \wedge \cdots \wedge \psi_n$.

Program a cíl

Programu odpovídá konečná množina formulí (teorie).

$$\begin{aligned} T = \{&child(abel, eva), child(abel, adam),\\ &male(adam), male(abel), female(eva),\\ &(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\} \end{aligned}$$

Cíli `(?` ψ_1 `...` ψ_n`)` odpovídá formule $\varphi = \psi_1 \wedge \cdots \wedge \psi_n$.

Například:

- `(? (father adam abel))` ... $\varphi = father(adam, abel)$

Modely teorie

Modely teorie

Struktura M je modelem T, jestliže je každá formule z T v M pravdivá.

Modely teorie

Struktura M je modelem T, jestliže je každá formule z T v M pravdivá.

$$T = \{child(abel, eva), child(abel, adam), male(adam), male(abel), female(eva), \\ (\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\}$$

Modely teorie

Struktura M je modelem T, jestliže je každá formule z T v M pravdivá.

$$\begin{aligned} T = \{&child(abel, eva), child(abel, adam), male(adam), male(abel), female(eva), \\ &(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\} \end{aligned}$$

Struktury:

$$\begin{aligned} M_1 = \{&0, 1, 2\}, adam^{M_1} = 0, eva^{M_1} = 1, abel^{M_1} = 2, \\ &child^{M_1} = \{\langle 2, 1\rangle, \langle 2, 0\rangle\}, male^{M_1} = \{0, 2\}, female^{M_1} = \{1\}, father^{M_1} = \{\langle 0, 2\rangle\}; \\ M_2 = \{&0, 1, 2\}, adam^{M_2} = 0, eva^{M_2} = 1, abel^{M_2} = 2, \\ &child^{M_2} = \{\langle 2, 1\rangle, \langle 2, 0\rangle\}, male^{M_2} = \{0, 2\}, female^{M_2} = \{1\}, father^{M_2} = \{\langle 0, 1\rangle\}. \end{aligned}$$

Modely teorie

Struktura M je modelem T, jestliže je každá formule z T v M pravdivá.

$$\begin{aligned} T = \{&child(abel, eva), child(abel, adam), male(adam), male(abel), female(eva), \\ &(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\} \end{aligned}$$

Struktury:

$$\begin{aligned} M_1 = \{&0, 1, 2\}, adam^{M_1} = 0, eva^{M_1} = 1, abel^{M_1} = 2, \\ &child^{M_1} = \{\langle 2, 1\rangle, \langle 2, 0\rangle\}, male^{M_1} = \{0, 2\}, female^{M_1} = \{1\}, father^{M_1} = \{\langle 0, 2\rangle\}; \\ M_2 = \{&0, 1, 2\}, adam^{M_2} = 0, eva^{M_2} = 1, abel^{M_2} = 2, \\ &child^{M_2} = \{\langle 2, 1\rangle, \langle 2, 0\rangle\}, male^{M_2} = \{0, 2\}, female^{M_2} = \{1\}, father^{M_2} = \{\langle 0, 1\rangle\}. \end{aligned}$$

M_1 je model T, ale M_2 není

Modely teorie

$$T = \{child(abel, eva), child(abel, adam), male(adam), male(abel), female(eva),$$
$$(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\}$$

Modely teorie

$$T = \{child(abel, eva), child(abel, adam), male(adam), male(abel), female(eva),$$
$$(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\}$$

Každá teorie programu má model:

$$M_3 = \{0\}, adam^{M_3} = eva^{M_3} = abel^{M_3} = 0,$$
$$child^{M_3} = father^{M_3} = \{\langle 0, 0\rangle\}, male^{M_3} = female^{M_3} = \{0\}$$

Modely teorie

$$T = \{child(abel, eva), child(abel, adam), male(adam), male(abel), female(eva), \\ (\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\}$$

Každá teorie programu má model:

$$M_3 = \{0\}, adam^{M_3} = eva^{M_3} = abel^{M_3} = 0, \\ child^{M_3} = father^{M_3} = \{\langle 0, 0\rangle\}, male^{M_3} = female^{M_3} = \{0\}$$

Zajímavý model:

$$M_4 = \{adam, eva, abel\}, adam^{M_4} = adam, eva^{M_4} = eval, abel^{M_4} = abel, \\ child^{M_4} = \{\langle abel, eva\rangle, \langle abel, adam\rangle\}, male^{M_4} = \{adam, abel\}, \\ female^{M_4} = \{eva\}, father^{M_4} = \{\langle adam, abel\rangle\}$$

Rozhodnutí sémantického vyplývání

Rozhodnutí sémantického vyplývání

Formule φ sémanticky vyplývá z T, jestliže je φ pravdivá v každém modelu T.

Zápis: $T \vDash \varphi$

Rozhodnutí sémantického vyplývání

Formule φ sémanticky vyplývá z T, jestliže je φ pravdivá v každém modelu T.

Zápis: $T \models \varphi$

Platí:

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

Rozhodnutí sémantického vyplývání

Formule φ sémanticky vyplývá z T, jestliže je φ pravdivá v každém modelu T.

Zápis: $T \models \varphi$

Platí:

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

Například:

$$\{child(abel, eva), child(abel, adam),$$
$$male(adam), male(abel), female(eva),$$
$$(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\} \models father(adam, abel)$$

Rozhodnutí sémantického vyplývání

Formule φ sémanticky vyplývá z T, jestliže je φ pravdivá v každém modelu T.

Zápis: $T \models \varphi$

Platí:

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

Například:

$$\{child(abel, eva), child(abel, adam),$$
$$male(adam), male(abel), female(eva),$$
$$(\forall x, y)((child(y, x) \wedge male(x)) \rightarrow father(x, y))\} \models father(adam, abel)$$

Připomenutí zamýšleného modelu T:

$$M_1 = \{0, 1, 2\}, adam^{M_1} = 0, eva^{M_1} = 1, abel^{M_1} = 2,$$
$$child^{M_1} = \{\langle 2, 1\rangle, \langle 2, 0\rangle\}, male^{M_1} = \{0, 2\}, female^{M_1} = \{1\}, father^{M_1} = \{\langle 0, 2\rangle\}$$

Obsah