KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

Jan Laštovička
https://www.inf.upol.cz/lide/jan-lastovicka
jan.lastovicka@upol.cz
17. listopadu 12, 771 46 Olomouc

Úvod do programovacích stylů

Manuál ke knihovně `logic`

verze z 3. prosince 2025

Knihovnu tvoří jediný soubor `logic.py`. Soubor se musí nalézat v témže adresáři jako program, který jej používá. Pro použití stačí importovat modul:

```
from logic import *
```

Instanciací `Var(`*`name`*`=None)` vznikají nové proměnné, Pokud zadáme *`name`*, pak vznikne proměnná se zadaným jménem, jinak se jako jméno použije dosud nepoužité systémem vybrané číslo. Knihovna zavádí proměnné `x`, `y` a `z`.

```
walk(val, sub) => val'

  val: hodnota
  sub: substituce
  val': hodnota
```

Hodnota *`val'`* vznikne průchodem substitucí *`sub`* z hodnoty *`val`*.

```
reify(val, sub) => val'

  val: hodnota
  sub: substituce
  val': hodnota
```

Hodnota *`val'`* vznikne dosazením substituce *`sub`* do hodnoty *`val`*.

```
unify(val1, val2, sub) => res

  val1: hodnota
  val2: hodnota
  sub: substituce
  res: substituce nebo None
```

Pokud se unifikace hodnot *`val1`* a *`val2`* vzhledem k substituci *`sub`* zdaří, pak *`res`* je výsledná substituce, jinak hodnota `None`.

Cíl `tatu` je vždy splněný a cíl `contr` není splněný nikdy.

```
eq(val1, val2) => goal
```

val1: hodnota
val2: hodnota
goal: cíl

Predikát stanovuje, že se hodnoty *val1* a *val2* rovnají.

```
add(m, n, k) => goal
```

m, *n*, *k*: hodnoty
goal: cíl

Predikát stanovuje, že součet hodnot *m* a *n* je *k*. V době pokusu o splnění cíle *goal* musí být aspoň dvě z hodnot *m*, *n*, *k* známé.

```
conj(goal1, ..., goaln) => goal
```

goal1, ..., *goaln*: cíle
goal: cíl

Spojka stanovuje, že všechny cíle *goal1*, ..., *goaln* musí být splněny.

```
disj(goal1, ..., goaln) => goal
```

goal1, ..., *goaln*: cíle
goal: cíl

Spojka stanovuje, že aspoň jeden z cílů *goal1*, ..., *goaln* je splněn.

```
neg(goal) => goal'
```

goal: cíl
goal': cíl

Spojka vrací cíl *goal'*, který je splněn právě tehdy, když cíl *goal* splněn není.

```
lazy(pred, arg1, ..., argn) => goal
```

pred: predikát
arg1, ..., *argn*: hodnoty
goal: cíl

Při pokusu o splnění cíle *goal* se aplikuje *pred* na *arg1*, ..., *arg*n, čímž se získá cíl, který se systém pokusí splnit místo původního cíle.

```
run(pattern, goal1, ..., goaln) => values

  pattern: hodnota
  goal1, ..., goaln: cíle
  values: pole hodnot
```

Pole *values* je tvořené hodnotami vzniklými dosazením možností splnění konjunkce cílů *goal1*, ..., *goal*n do hodnoty *pattern*.

```
run_lazy(pattern, goal1, ..., goaln) => iterator

  pattern: hodnota
  goal1, ..., goaln: cíle
  iterator: iterátor hodnot
```

Iterátor *iterator* postupně vrací hodnoty vzniklé dosazením možností splnění konjunkce cílů *goal1*, ..., *goal*n do hodnoty *pattern*.

```
relation(tuples) => predicate

  tuples: pole polí se stejnou délkou n
  predicate: predikát
```

Predikát *predicate* očekává n argumenů *val1*, ..., *val*n a vrací cíl, který je splněn, jestliže se pole [*val1*, ..., *val*n] nalézá mezi prvky pole *tuples*.