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 Logical Micro Widget

verze z 4. prosince 2025

Knihovnu tvoří soubory `logic.py`, `micro_widget.py` a `lmw.py`. Soubory se musí nalézat v témže adresáři jako program, který je používá. Pro použití stačí importovat modul:

```
from lmw import *
```

Ovládací prvky jsou datové struktury reprezentované polem, kde první prvek je řetězec určující typ ovládacího prvku. Typy přehledně zachycuje následující tabulka:

popisek	`label`
tlačítko	`button`
textové pole	`entry`
zaškrtávací pole	`checkbox`
přepínač	`radiobutton`

Typ *`widget_type`* je určen řetězcem `"`*`widget_type`*`"`. Ostatní prvky pole odpovídají položkám datové struktury. Druhý a třetí prvek prvek pole odpovídá položkám `x` a `y` určujícím souřadnice ovládacího prvku. Položka `text` určující text tlačítka a popisku se nalézá ve čtvrtém prvku pole. Hodnotu textového pole, zaškrtávacího pole a přepínače určuje položka `value` ve čtvrtém prvku pole. Akce tlačítka, textového pole, zaškrtávacího pole a přepínače je určena položkou `action` nalézající se v pátém prvku pole. Vyplývá, že popisek je čtyřprvkové pole. Ostatní ovládací prvky jsou pětiprvková pole.

Pro každý typ ovládacích prvků *`widget_type`* máme predikát *`widget_type`*`_props` očekávají ovládací prvek *`widget`* a typu odpovídající počet položek *`a1`*, ..., *`an`*. Predikát určí, že *`widget`* je ovládací prvek typu *`widget_type`* a *`a1`*, ..., *`an`* jsou postupně jeho položky. Dále volání predikátu *`widget_type`*`(`*`widget`*`)` určí, že *`widget`* je ovládací prvek typu *`widget_type`*.

Pro každý typ ovládacích prvků *`widget_type`* a jeho položku *`attribute`* máme predikát *`widget_type_attribute`* očekávající dva argumenty *`widget`* a *`value`*. Predikát určí, že ovládací prvek *`widget`* je typu *`widget_type`* a hodnota atributu *`attribute`* je *`value`*.

Pro každou výše uvedenou položku *`attribute`* máme predikát `widget_`*`attribute`* očekávající argumenty *`widget`* a *`value`*. Predikát určí, že *`widget`* je ovládací prvek, který může mít atribut *`attribute`*, a ten má hodnotu *`value`*.

Knihovna zavádí proměnné: `content`, `widget`, `widget1`, `widget2`, `widget3`, `value`, `next_x`, `next_y`, `next_z` a `next_value`.

```
display(content_pattern,
        content_goal=taut,
        init_goal=taut,
        trace=False) => None
```

content_pattern: hodnota
content_goal: cíl
init_goal: cíl
trace: hodnota `True` nebo `False`

Volání vytvoří novou aplikaci pro každou možnost splnění cíle *init_goal*. Možnost splnění bude výchozím stavem aplikace. Vytvořená aplikace otevře okno pro každou možnost splnění cíle *content_goal* ze stavu aplikace. Možnost splnění určuje obsah okna. Při změně stavu aplikace dojde k opětovnému pokusu o splnění cíle *content_goal* a aktualizaci aplikací otevřených oken.

Při stisknutí tlačítka se zpracuje jeho akce. Při změně hodnoty textového pole, přepínače nebo zaškrtávacího pole dojde k zpracování jeho akce s novou hodnotou.

Pokud je hodnota *trace* pravda, tisknou se při každé změně stavy aplikací a obsahy oken.

Zpracování akce *action* probíhá tak, že se nejprve systém pokusí splnit cíl *action*. Cíl musí mít právě jednu možnost splnění, jinak systém vyvolá výjimku. Výpočet nového stavu aplikace probíhá v těchto krocích:

1. Položíme nový stav roven substituci `next_state_sub`.
2. V novém stavu zanecháme jen ty vazby, jejichž proměnné začínají prefixem `next_`.
3. Z proměnných vazeb v novém stavu se odstraní prefix `next_`.
4. Do substituce se přidají ty vazby ze starého stavu, jejichž proměnné se nenachází v žádné vazbě v novém stavu.

Zpracování akce *action* s hodnotou *value* probíhá stejně jako zpracování akce *action* až na to, že se cíl *action* pokoušíme splnit z aktuálního stavu, do kterého jsme přidali vazbu proměnné *changed_value* na *value*.