

# Generating program code for psychological experiments from high-level descriptions

University of Novi Sad<sup>1</sup>, University of Priština<sup>2</sup>

---

Igor Dejanović<sup>1</sup>, Mirjana Dejanović<sup>2</sup>

September, 2019 @ ERK Portorož Slovenia

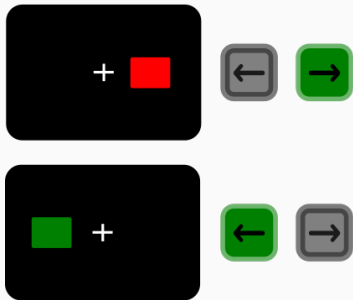
# Introduction

---

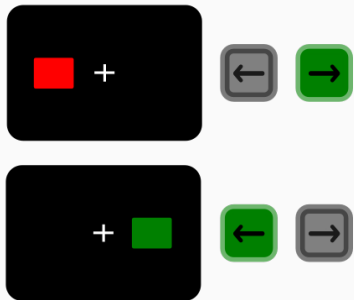


## An example – the Simon effect test

congruent



incongruent



# Motivation

# Domain-Specific Languages

---

# What are they?

# textX

Domain-Specific Languages  
made easy

```
robot.txt
Program:
begin
  commands+=Command
end

Command:
InitialCommand | MoveCommand

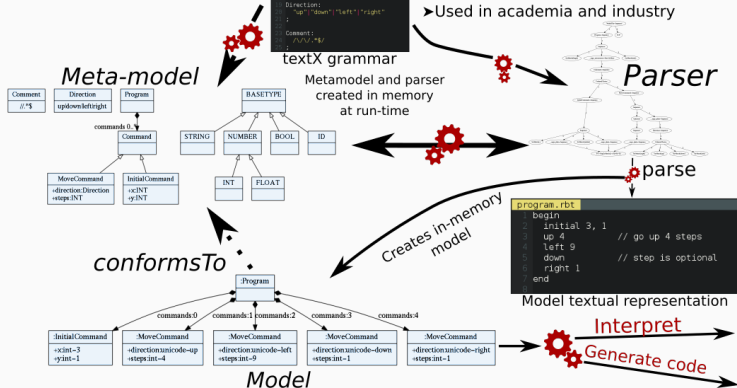
InitialCommand:
initials+=INT '/' y=INT

MoveCommand:
directionDirection (steps=INT)

Direction:
'up' | 'down' | 'left' | 'right'

Comment:
//.*$
```

- 100% Python
- Built on top of Arpeggio
- Grammar is interpreted
- Only dependency - Arpeggio
- Simple, light-weight, powerful
- Meta-model and model visualization
- Available at GitHub - MIT license
- Used in academia and industry



<https://github.com/textX/textX>

I. Dejanović, R. Vadera, G. Milosavljević, Ž. Vuković, TextX: A Python tool for Domain-Specific Languages implementation, Knowledge-Based Systems 115, 1-4, 2017.



pyFlies

---

# pyFlies code for the Simon effect test

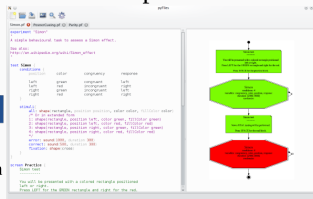
```
test Simon {  
  conditions {  
    position  color  congruency  response  
  
    left      green  congruent   left  
    left      red    incongruent right  
    right     green  incongruent left  
    right     red    congruent   right  
  }  
  
  stimuli{  
    all: shape(rectangle, position position,  
               color color)  
    error: sound(1000)  
    fixation: shape(cross)  
  }  
}
```

# Target code generators



Direct  
interpretation

## Abstract experiment model



Automatic model  
transformation  
(code generation)

Existing platforms

PsychoPy

Epyriment

jsPsych

OpenSesame

Tatool

...

# Template engines

```
# Run experiment
# Create folder for experiment results.
timestr = time.strftime("%Y%m%d-%H%M%S")
exp_folder = '{(e.name)}-%s' % timestr
os.mkdir(exp_folder)

{% for e in n.structure.elements %}
{% if e.__class__.__name__ == "ScreenInstance" %}
present_stimuli(screen,{(e.type.name)})
{% elif e.__class__.__name__ == "TestInstance" %}
run_block(exp_folder, "{(e.type.name)}", {(e.type.name)}.varNames,
           {(e.type.name)}.conditions, {(e.type.name)}.condition_stimuli,
           {(e.type.name)}.fixation, {(e.type.name)}.error,
           {(e.type.name)}.correct)
{% endif %}
{% endfor %}
```

Template

## Template engine

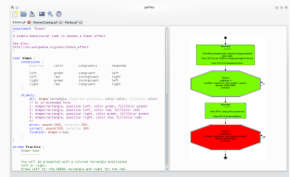


```
# Run experiment
# Create folder for experiment results.
timestr = time.strftime("%Y%m%d-%H%M%S")
exp_folder = 'Simon-%s' % timestr
os.mkdir(exp_folder)

present_stimuli(screen_Practice)

run_block(exp_folder, "Simon", Simon_varNames,
          Simon_conditions, Simon_condition_stimuli,
          1, True, True,
          Simon_fixation, Simon_error, Simon_correct)
```

Generated source code



pyFlies model

## Conclusion

---

Thanks! Q&A?

---