

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

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

---

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

September, 2019 @ ERK Portorož Slovenia

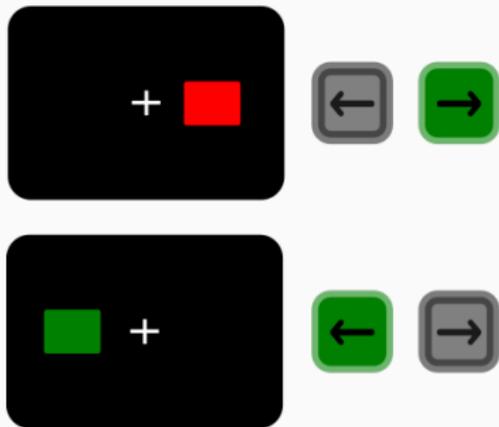
# Introduction

---

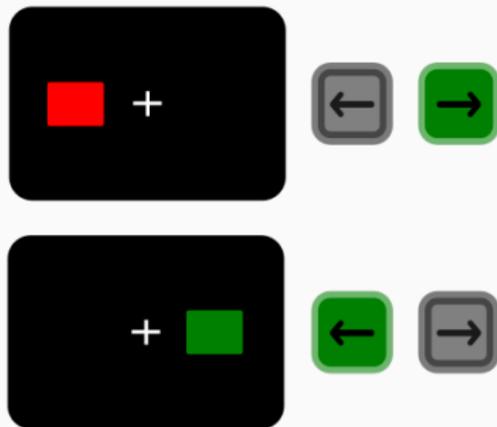
# Experiments in Psychology

# An example – the Simon effect test

congruent



incongruent



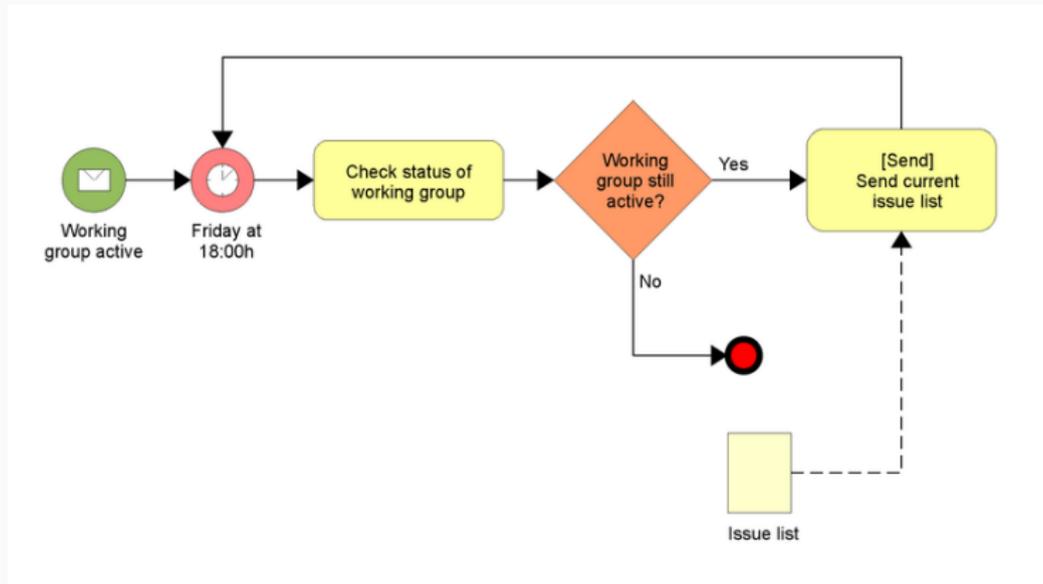
# Motivation

# Domain-Specific Languages

---

```
SELECT player, stadium  
FROM game JOIN goal ON (id=matchid)
```

# Business processes - BPMN



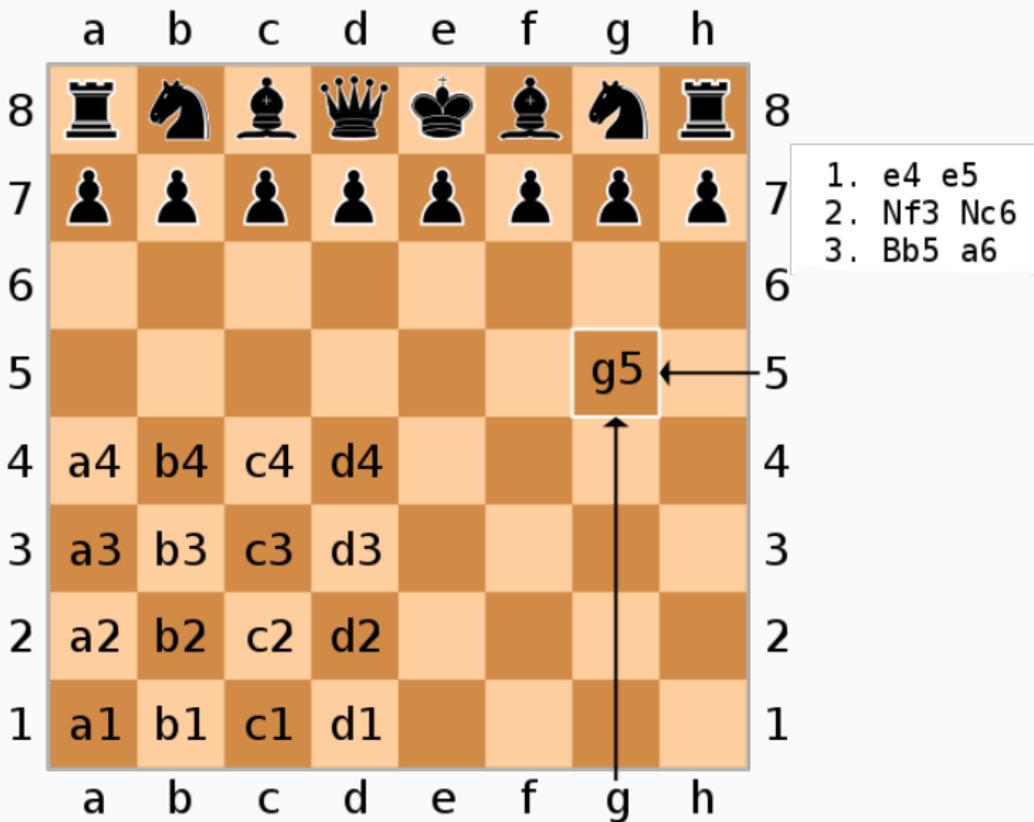
But also...

**Allegro Vivace**

*P sempre leggiero*

The image displays a musical score for piano, consisting of two systems. The tempo is marked "Allegro Vivace" and the dynamics are "P sempre leggiero". The key signature has three flats (B-flat, E-flat, A-flat) and the time signature is 12/8. The first system shows a whole rest in the right hand, followed by a rhythmic pattern of eighth notes in the left hand. The second system begins with a triplet of chords in the right hand, followed by a rhythmic pattern in the left hand, marked with "[simile]".

Or...

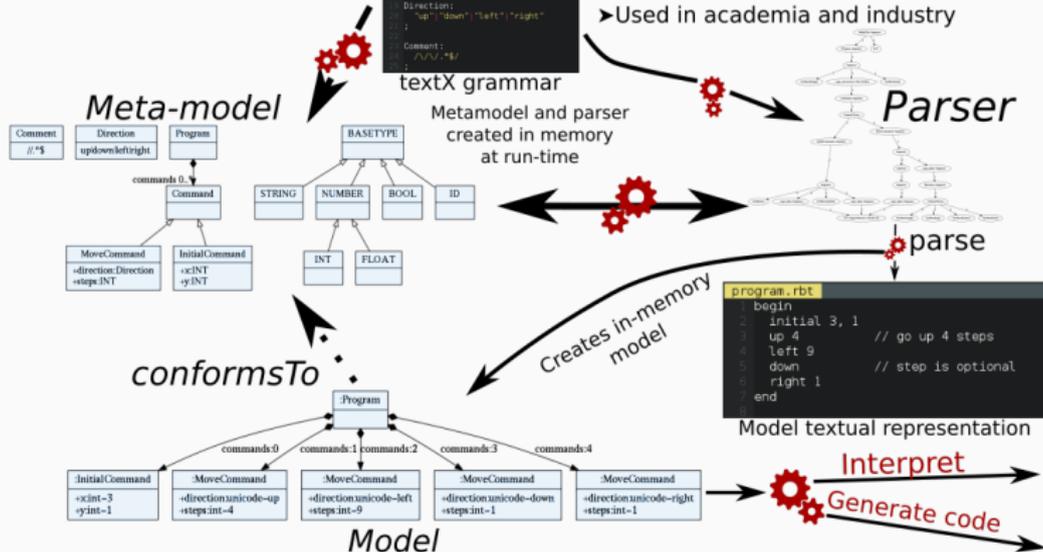


# textX

Domain-Specific Languages  
made easy

```
textX grammar
Program:
  begin
    commands* Command
  end
Command:
  InitialCommand | MoveCommand
InitialCommand:
  initial: INT ; y: INT
MoveCommand:
  direction: Direction (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. Vaderna, G. Milosavljević, Ž. Vuković, *TextX: A Python tool for Domain-Specific Languages implementation*, *Knowledge-Based Systems* 115, 1-4, 2017.

pyFlies

---

# pyFlies - DSL for psychological RT experiments

pyFlies

```
Simon.pf | PosnerCueing.pf | Parity.pf
```

```
experiment "Simon"
=
A simple behavioural task to assess a Simon effect.

See also:
http://en.wikipedia.org/wiki/Simon_effect
=

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, fillColor color)
    /* Or in extended form
    1: shape(rectangle, position left, color green, fillColor green)
    2: shape(rectangle, position left, color red, fillColor red)
    3: shape(rectangle, position right, color green, fillColor green)
    4: shape(rectangle, position right, color red, fillColor red)
    */
    error: sound(1000, duration 300)
    correct: sound(500, duration 300)
    fixation: shape(cross)
  }
}

screen Practice {
  Simon test
  -----

  You will be presented with a colored rectangle positioned
  left or right.
  Press LEFT for the GREEN rectangle and right for the red.
```

```
graph TD
  Start(( )) --> Test1[Simon test  
You will be presented with a colored rectangle positioned  
left or right.  
Press LEFT for the GREEN rectangle and right for the red.  
Press SPACE for the practice block.]
  Test1 --> Decision1{{Simon  
conditions: 4  
variables: congruency, color, position, response  
duration: [2000, 4000]  
randomize}}
  Decision1 --> Test2[Simon test  
Now a REAL testing will be performed.  
Press SPACE for the real block.]
  Test2 --> Decision2{{Simon  
conditions: 4  
variables: congruency, color, position, response  
duration: [2000, 4000]  
randomize}}
  Decision2 --> End(( ))
```

## pyFlies code for the Simon effect test

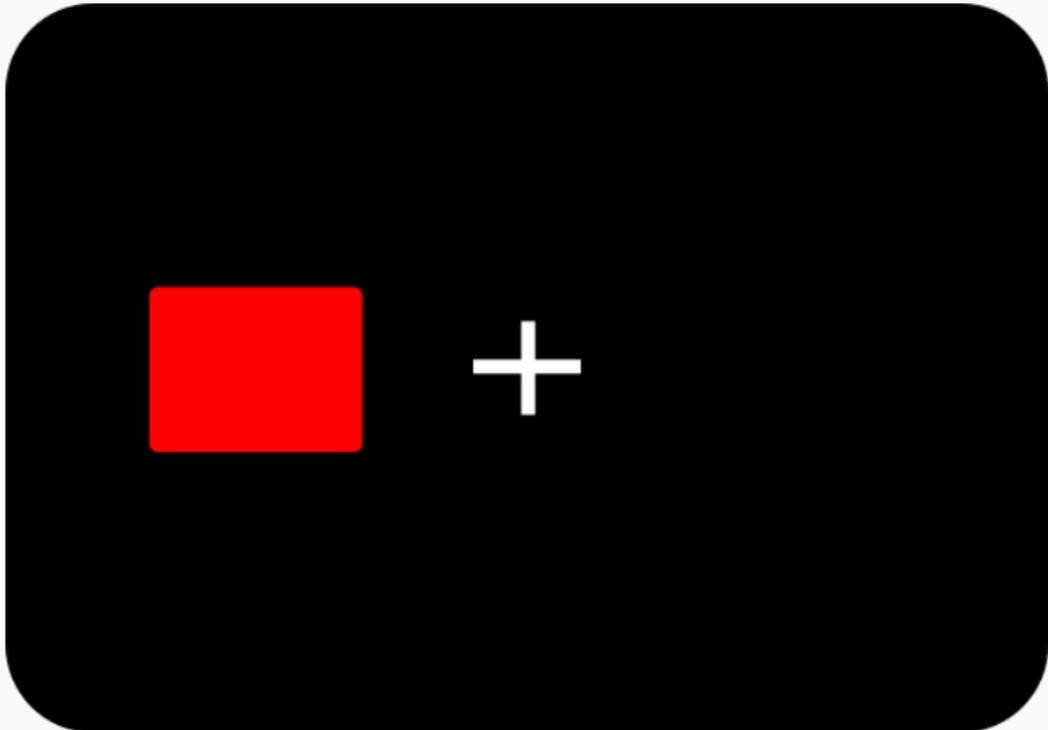
```
test Simon {
  conditions {
    position  color  congruency  response

    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)
  }
}
```

# Connecting stimuli and conditions

```
parity=odd and position=left: image('red_square.png', position position)  
position=left and color=red
```





# 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)}}_{{(e.practice)}}_{{(e.randomize)}}_{{(e.type.name)}}_fixation, {{(e.type.name)}}_error,
          {{(e.type.name)}}_correct)
{% endif %}
{% endfor %}
```

Template

## Template engine

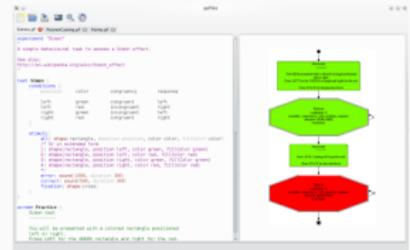


```
# Run experiment
# Create folder for experiment results.
timestr = time.strftime("%Y%m%d-%H%M%S")
exp_folder = 'Simon-s' % timestr
os.mkdir(exp_folder)

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?

---