Boosting LLM-Based Software Generation by Aligning Code with Requirements

Tom Yaacov, Achiya Elyasaf, Gera Weiss

14th International Model-Driven Requirements Engineering (MoDRE) workshop

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

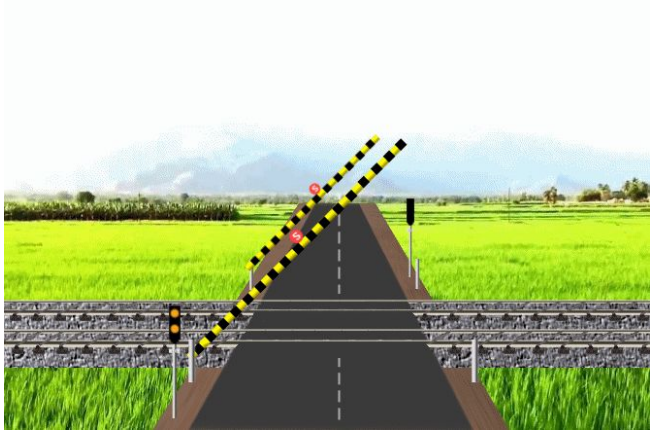
Level Crossing System Example

1) When a train passes, the sensor system activates the exact event order: **approaching**, **entering**, and **leaving**.

2) The barriers are **lowered** when a train **approaches** and then **raised**.

3) A train may not **enter** while barriers are **raised**.

4) The barriers may not be **raised** while a train passes, i.e., it **approached** but did not **leave**.



N. Leveson and J. Stolzy, “Safety Analysis Using Petri Nets,” IEEE Transactions on Software Engineering, vol. SE-13, no. 3, pp. 386–397, 1987.

Level Crossing System Example

A general Python implementation by GPT 4:

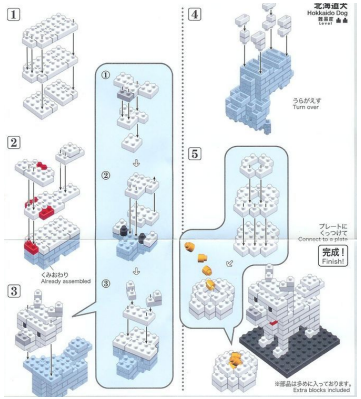
```
def railway_crossing_events():
    evts = ["Approaching", "Entering", "Leaving", "Lower", "Raise"]
    sequence = []
    mandatory_sequence = ["Approaching", "Lower", "Entering",
                          "Leaving", "Raise"]
    sequence.extend(mandatory_sequence)
    pre_evts = random.sample(evts, k=random.randint(0, len(evts)))
    sequence = pre_evts + sequence
    post_evts = random.sample(evts, k=random.randint(0, len(evts)))
    sequence += post_evts
    if sequence[-1] != "Raise":
        sequence.append("Raise")
    return sequence
```

Challenges of LLM-Based Code Generation

- We identify two main factors:
  - Programmers must **manually design** the software and LLMs only implement **parts of it**.
  - LLMs introduce **errors** that are challenging for programmers and stakeholders to **identify**.

Behavioral Programming (BP)

Requirements



Execution engine

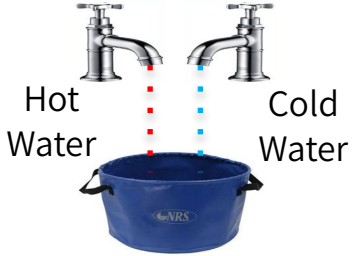
Behavior

David Harel, Assaf Marron, Gera Weiss. "Behavioral programming." Communications of the ACM 55.7 (2012): 90-100.

Behavioral Programming (BP)

1) "Pour some small amount of hot water three times."

```
@b_thread
def pour_3_hot():
    for i in range(3):
        yield {request: Hot}
```



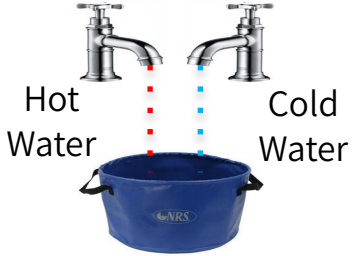
Yaacov, Tom. "BPpy: Behavioral Programming in Python." SoftwareX 24 (Dec. 2023), 101556.

Behavioral Programming (BP)

2) "Pour some small amount of cold water three times."

```python
@b_thread
def pour_3_hot():
    for i in range(3):
        yield {request: Hot}
```

```python
@b_thread
def pour_3_cold():
    for i in range(3):
        yield {request: Cold}
```



Possible system traces:

```
Cold
   Hot
   Hot
Cold
   Hot
Cold
```

```
   Hot
Cold
   Hot
Cold
   Hot
Cold
```

```
   Hot
   Hot
   Hot
Cold
Cold
Cold
```

Yaacov, Tom. "BPpy: Behavioral Programming in Python." SoftwareX 24 (Dec. 2023), 101556.

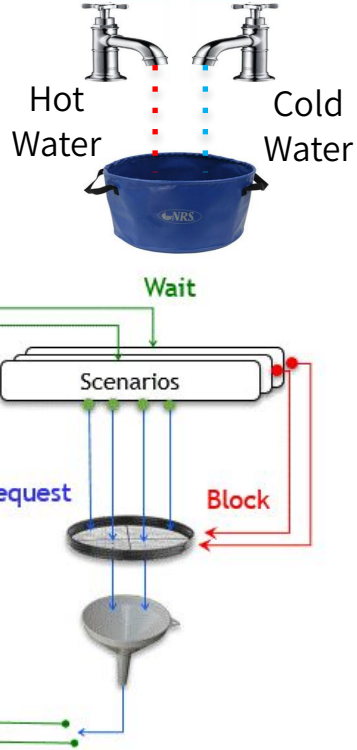
Behavioral Programming (BP)

3) "Cold water should be poured between any two pouring of hot water."

```
@b_thread
def pour_3_hot():
    for i in range(3):
        yield {request: Hot}
```

```
@b_thread
def pour_3_cold():
    for i in range(3):
        yield {request: Cold}
```

```
@b_thread
def prevent_consecutive_hot():
    While True:
        yield {waitFor: Hot}
        yield {waitFor: Cold, block: Hot}
```

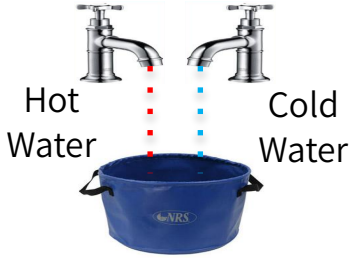


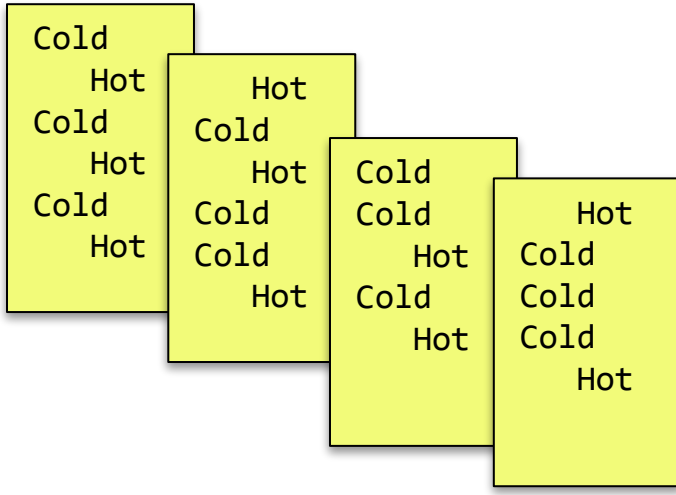
Behavioral Programming (BP)

```
@b_thread
def pour_3_hot():
    for i in range(3):
        yield {request: Hot}
```

```
@b_thread
def pour_3_cold():
    for i in range(3):
        yield {request: Cold}
```

```
@b_thread
def prevent_consecutive_hot():
    While True:
        yield {waitFor: Hot}
        yield {waitFor: Cold, block: Hot}
```



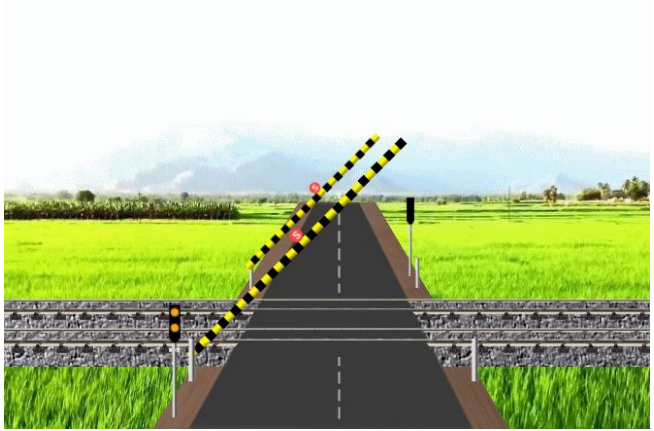


Level Crossing System Example

A BPpy implementation by GPT 4:

```
@b_thread
def requirement_1():
    while True:
        yield {waitFor: Approaching}
        yield {waitFor: Entering}
        yield {waitFor: Leaving}
@b_thread
def requirement_2():
    while True:
        yield {waitFor: Approaching}
        yield {request: Lower}
        yield {waitFor: Leaving}
        yield {request: Raise}
```

```
@b_thread
def requirement_3():
    while True:
        yield {waitFor: Approaching}
        yield {block: Entering,
               waitFor: Leaving}
@b_thread                    Lower
def requirement_4():
    while True:
        yield {waitFor: Approaching}
        yield {block: Raise,
               waitFor: Leaving}
```



Initial Evaluation

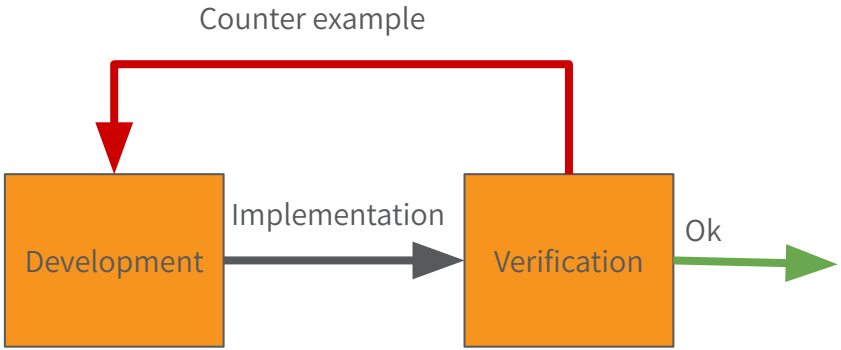
- The initial experiment involved **20** system with a total of **149** requirements.
- We compared GPT's BP and General Python implementation based on sampled system traces.
- BP implementation showed better alignment in **52** requirements, while the general implementation in **37**.
- Statistical significance - probability of a random Bernoulli variable to produce such advantage is **95.5%**

Specification	#Requirements	General	BP
r1	8	1	3
r2	7	2	1
r3	8	2	4
r4	8	5	1
r5	5	2	3
r6	8	2	4
r7	8	2	2
r8	5	0	5
r9	9	3	5
r10	6	1	3
rs1	4	1	3
rs2	4	1	2
rs3	8	3	0
rs4	10	4	4
rs5	8	1	1
rs6	9	1	3
rs7	9	3	0
rs8	9	0	3
rs9	8	3	1
rs10	8	0	4
Total	**149**	**37**	**52**

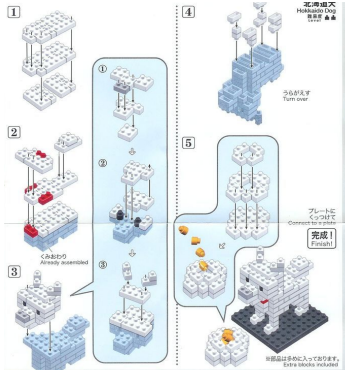
https://github.com/bThink-BGU/Papers-2024-MoDRE-BP-LLM

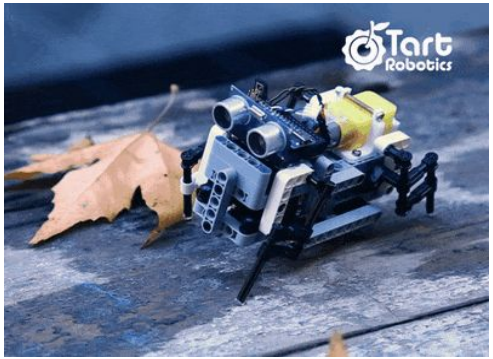
Verification and Validation Support

- BPpy support model checking through both **explicit** and **symbolic** modes.
- **Explicit mode** - program is validated through assertions in b-threads using DFS.
- **Symbolic mode (**NuSMV):
  - Provides general LTL support.
  - Avoids explicit enumeration of all program states.



Yaacov, Tom. "BPpy: Behavioral Programming in Python." SoftwareX 24 (Dec. 2023), 101556.





Thank you for listening

tomya@post.bgu.ac.il