

Large Language Model for Software Variability and Configurations

Mathieu Acher
@acherm



institut
universitaire
de France



PhD at University of Nice Sophia Antipolis (MESR, 2008-2011)

Postdoc at University of Namur, Belgium (Marie Curie fellowship, 2011-2012)

Associate Professor at University of Rennes 1/ISTIC (2012-2022)

Junior research fellow at IUF "mediation" (september 2021, 5 years)

Professor at INSA Rennes since september 2022

Love chess (FIDE master, 2341 Elo)

Love football (coach of Vétérans in Saint Aubin du Cormier)

Love software

Love science

can you depict a map of France with all locations?

```
python Copy code

import folium

# Create a map centered around France
m = folium.Map(location=[46.603354, 1.888334], zoom_start=6)

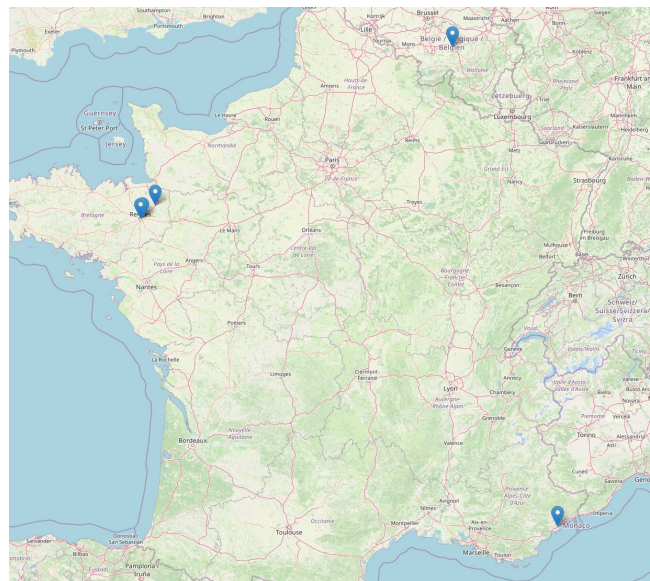
# Add markers for each location
locations = {
    'University of Nice Sophia Antipolis': [43.6155, 7.0719],
    'University of Namur, Belgium': [50.4652, 4.8676],
    'University of Rennes 1/ISTIC': [48.1172, -1.6778],
    'INSA Rennes': [48.1215, -1.6366],
    'Saint Aubin du Cormier': [48.2972, -1.3746]
}

for place, coord in locations.items():
    folium.Marker(location=coord, popup=place).add_to(m)

# Save the map to an HTML file
m.save('map.html')
```



institut
universitaire
de France



Plan

3 AI for software variability and configurations

LLM for programming variability

Discussion

Synthesizing **variants with software** is a holy grail in many application domains and fields

exploration of different designs, ideas, hypothesis (e.g., in computational science or avionics)

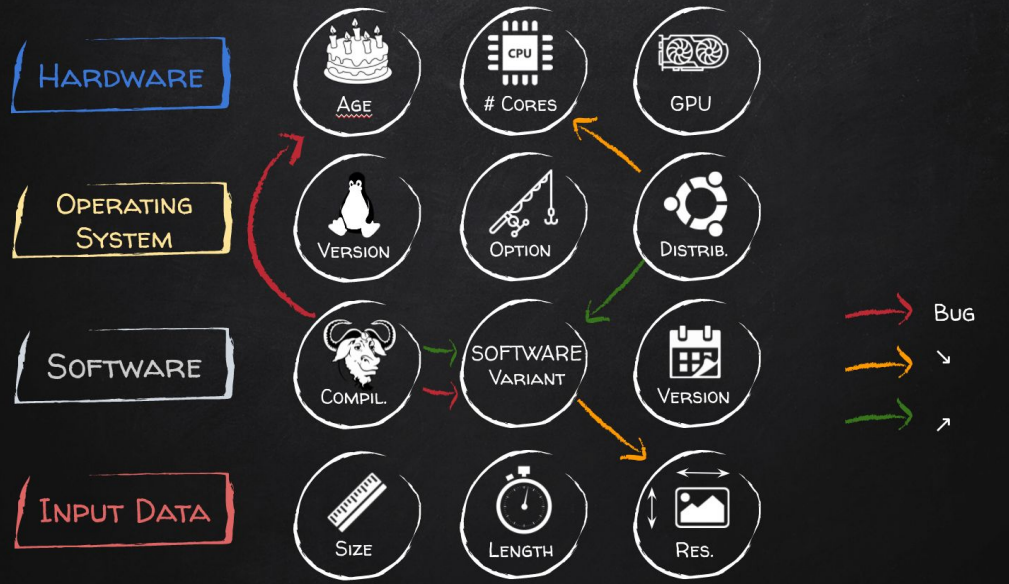
fitting a variety of contexts, users, requirements (e.g., wrt security or energy consumption)

highly-configurable systems, software product lines, generators

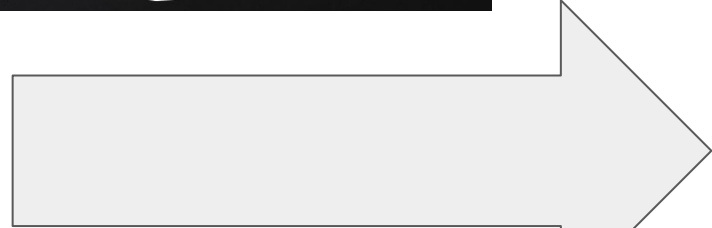
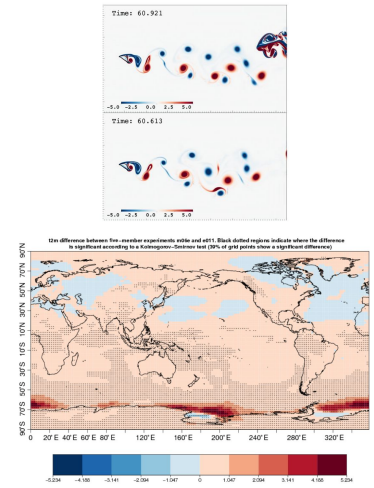
configuration, A/B testing, tuning, hyperparameters, diversification, toggles, etc.



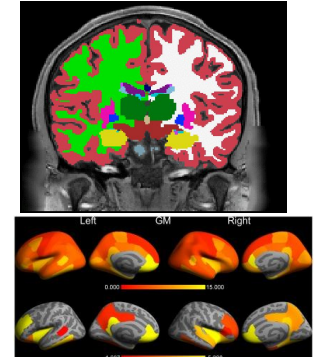
DEEP VARIABILITY



(computational) science



from a set of scripts to automate the deployment to... a comprehensive system containing several features that help researchers exploring various hypotheses



Can we accurately predict configurations' properties?

Challenge 1+2: huge configuration space + Linux evolves a lot

Variability in space

```
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDMISH=y
# CONFIG_INTEL_PUNIT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

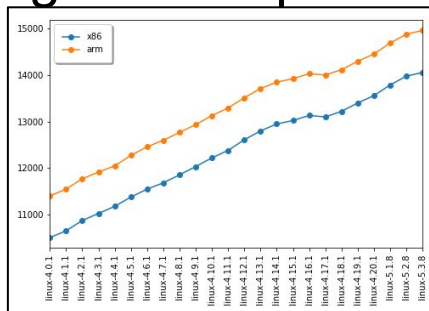
7.1Mb

```
CONFIG_PM_WAKELOCKS=y
CONFIG_PM_WAKELOCKS_LIMIT=100
CONFIG_PM_WAKELOCKS_GC=y
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
CONFIG_PM_CLK=y
CONFIG_PM_GENERIC_DOMAINS=y
CONFIG_WQ_POWER_EFFICIENT_DEFAULT=y
CONFIG_PM_GENERIC_DOMAINS_SLEEP=y
CONFIG_PM_GENERIC_DOMAINS_OF=y
CONFIG_ENERGY_MODEL=y
CONFIG_ARCH_SUPPORTS_ACPI=y
```

176.8Mb

```
CONFIG_VM_EVENT_COUNTERS=y
CONFIG_SLUB_DEBUG=y
# CONFIG_SLUB_MEMCG_SYSSFS_DM is not set
# CONFIG_COMPAT_BRK is not set
# CONFIG_SLAB is not set
CONFIG_SLUB=y
# CONFIG_SLOB is not set
# CONFIG_SLAB_MERGE_DEFAULT is not set
# CONFIG_SLAB_FREELIST_RANDOM is not set
# CONFIG_SLAB_FREELIST_HARDENED is not set
CONFIG_SHUFFLE_PAGE_ALLOCATOR=y
CONFIG_SLUB_CPU_PARTIAL=y
CONFIG_SYSTEM_DATA_VERIFICATION=y
```

?



3 years later...

?

?

?

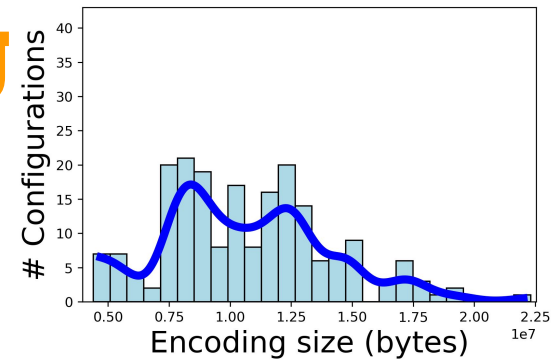
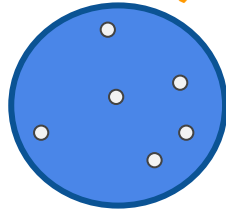
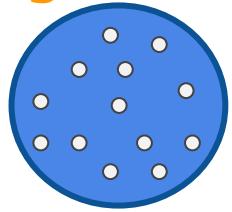
v4.13

...Variability in time...

v5.8

Statistical machine learning

(regression, classification)

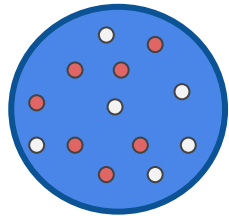


Whole Population of Configurations

Training Sample

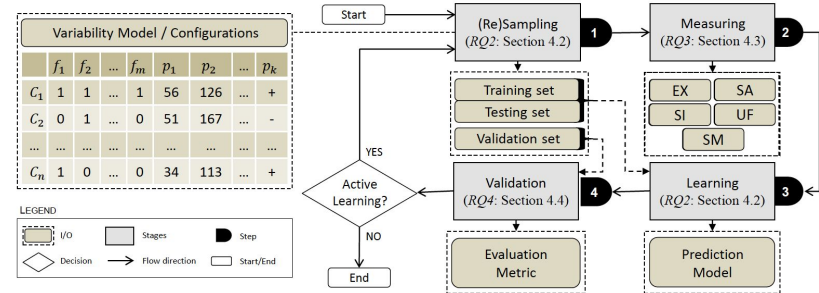
Performance Measurements

Prediction Model

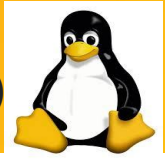


Performance Prediction

$f : \mathbb{C} \rightarrow \mathbb{R}^+$ the function affecting to any configuration $c \in \mathbb{C}$ its performance $f(c) \in \mathbb{R}^+$,



Statistical machine learning (eg transfer learning)



cid	compilation_date	compilation_time	config_file	stdout_log_file	stderr_log_file	user_output_file	compiled_kernel_size	compressed_compiled_kernel_size
74882	2019-08-12 17:09:42	399.856	[BLOB - 24,3 Kio]	[BLOB - 33,7 Kio]	[BLOB - 14 o]	[BLOB - 702 o]	74559280	GZIP-bzImage : 8855504 , GZIP-vmlinux : 10943304 , ...
74881	2019-08-12 16:58:09	460.392	[BLOB - 25,8 Kio]	[BLOB - 34,7 Kio]	[BLOB - 14 o]	[BLOB - 704 o]	81377768	GZIP-bzImage : 18375632 , GZIP-vmlinux : 20462408 ...
74880	2019-08-12 16:47:28	301.775	[BLOB - 22 Kio]	[BLOB - 24,2 Kio]	[BLOB - 14 o]	[BLOB - 705 o]	83004496	GZIP-bzImage : 14365648 , GZIP-vmlinux : 16452424 ...
74879	2019-08-12 16:46:14	1393.61	[BLOB - 24,1 Kio]	[BLOB - 50 Kio]	[BLOB - 571 o]	[BLOB - 712 o]	109098328	GZIP-bzImage : 17183792 , GZIP-vmlinux : 19272160 ...
74878	2019-08-12 16:45:03	305.705	[BLOB - 26,1 Kio]	[BLOB - 28,8 Kio]	[BLOB - 14 o]	[BLOB - 703 o]	55523752	GZIP-bzImage : 14767568 , GZIP-vmlinux : 16852088 ...

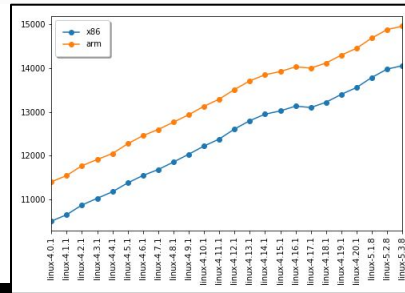
Console de requêtes SQL

74K+ configurations for Linux 4.15

95K+ configurations for Linux 4.13.3

(and 15K hours of computation on a grid computing)

H. Martin, M. Acher, J. A. Pereira, L. Lesoil, J. Jézéquel and D. E. Khelladi, "Transfer learning across variants and versions: The case of linux kernel size" Transactions on Software Engineering (TSE), 2021



Symbolic reasoning

Modelling and reverse engineering variability/configuration space

Statistical machine learning

Predicting properties of configurations and variants

Generative AI and large language model

Programming software variability

Plan

3 AI for software variability and configurations

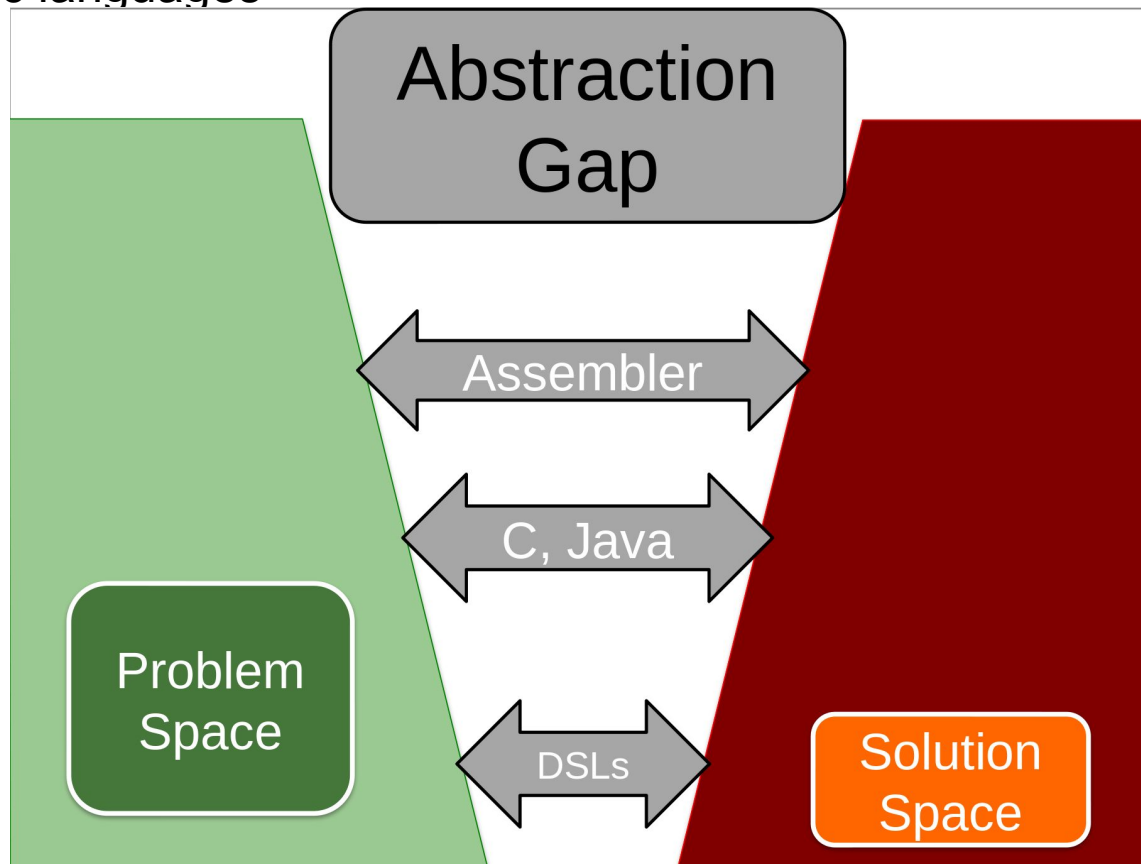
LLM for programming variability

Discussion

Generative programming [Czarnecki2000], Model-driven engineering:
automatically generate variants from a specification written in one or more textual or graphical domain-specific languages

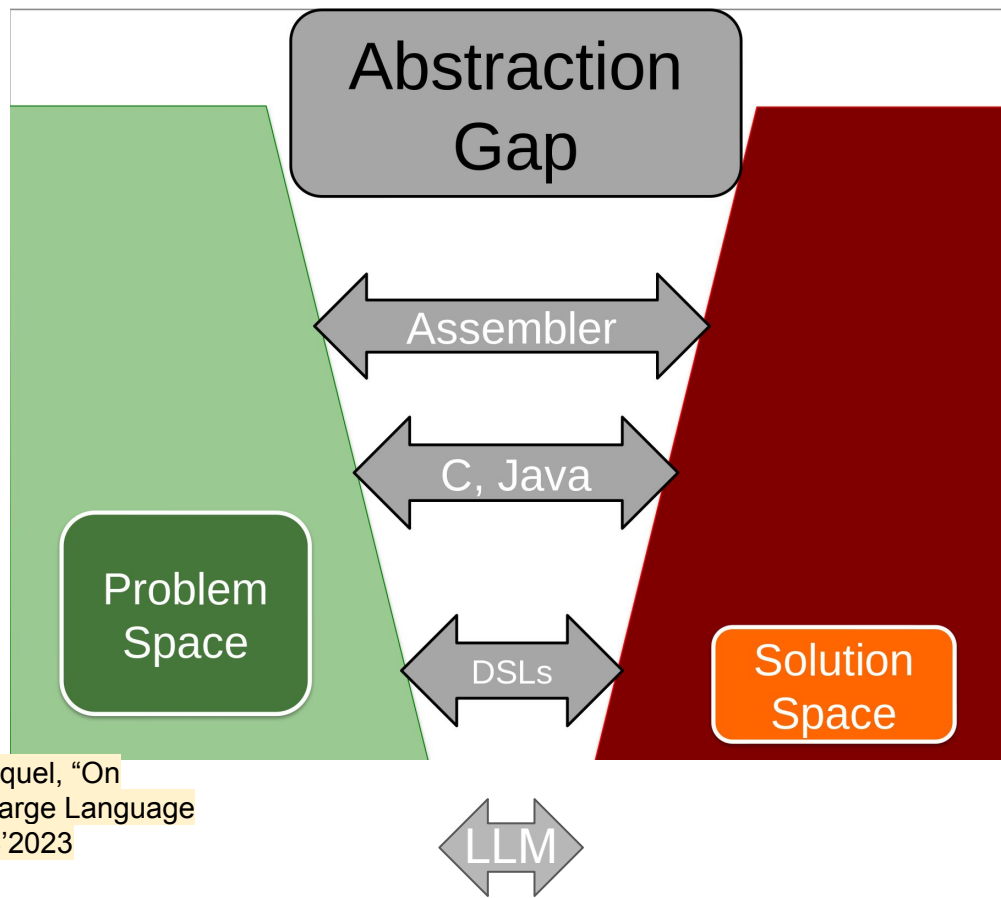
Accidental and essential complexity

Variability further increases this software complexity: multiple features, code variations, and an exponential number of possible variants



Hypothesis: **Large language models (LLMs)** act as a new variability compiler capable of transforming a high-level specification (“prompt”) into variable code, features, generators, configurable systems, etc. written in a given technological space.

Motto:
“features
as
prompts”



M. Acher, J. Galindo, J.M Jézéquel, “On Programming Variability with Large Language Model-based Assistant”, SPLC’2023

Remainder of the talk

How LLMs can be concretely and originally used for programming software variability

- Variability implementations of a classical Hello world

- Variability for reproducibility and floating-points

- Transforming an unfamiliar code with an end-user, Web-based customization tool

Discussions (potential and limitations)

Remainder of the talk

How LLMs can be concretely and originally used for programming software variability

Variability implementations of a classical Hello world

Variability for reproducibility and floating-points

Transforming an unfamiliar code with an end-user, Web-based customization tool

Discussions (potential and limitations)


Write a configurable **Rust** program that implements four features and their selections, namely, Hello, Beautiful, Wonderful, and World... If the feature is selected, then we print the name of the feature. For instance, we could well print "Hello Beautiful World", "Hello World", "Hello", "World" etc. The four features should be implemented as compile-time features.



(LLM) [...] you can use the 'cfg' attribute to enable or disable the features at compile-time [...] in your 'Cargo.toml' file [...].

```
fn main() {  
    #[cfg(feature = "hello")]  
    print!("Hello ");  
    #[cfg(feature = "beautiful")]  
    print!("Beautiful ");  
    #[cfg(feature = "wonderful")]  
    print!("Wonderful ");  
    #[cfg(feature = "world")]  
    print!("World");  
    println!();  
}
```


Write a configurable Java program that implements [...] The four features should be implemented as compile-time features. **As there is no preprocessor and built-in support for conditional compilation in Java, it is needed to leverage a templating engine for customization, acting as a preprocessor for Java. You can use Jinja (Python)**



(LLM) To create a configurable Java program that implements the four features as compile-time features using Jinja (Python) templating engine, you need to follow these steps: ... Create a Jinja template (main_template.java.j2) for your Java program [...] ... Create a Python script (generate_java.py) that will generate the Java source file based on the provided features:...

```
public class Main {
    public static void main(String[] args) {
{% if features.Hello %}
        System.out.print("Hello ");
{% endif %}
{% if features.Beautiful %}
        System.out.print("Beautiful ");
{% endif %}
{% if features.Wonderful %}
        System.out.print("Wonderful ");
{% endif %}
{% if features.World %}
        System.out.print("World");
{% endif %}
        System.out.println();
    }
}
```

Write a configurable Java program that implements [...] The four features should be implemented as compile-time features. **As there is no preprocessor and built-in support for conditional compilation in Java, it is needed to leverage a templating engine for customization, acting as a preprocessor for Java. You can use Jinja (Python)**



(LLM) To create a configurable Java program that implements the four features as compile-time features using Jinja (Python) templating engine, you need to follow these steps: ... Create a Jinja template (main_template.java.j2) for your Java program [...] ... Create a Python script (generate_java.py) that will generate the Java source file based on the provided features:...

```
public class Main {
    public static void main(String[] args) {
{% if features.Hello %}
        System.out.print("Hello ");
{% endif %}
{% if features.Beautiful %}
        System.out.print("Beautiful ");
{% endif %}
{% if features.Wonderful %}
        System.out.print("Wonderful ");
{% endif %}
{% if features.World %}
        System.out.print("World");
{% endif %}
        System.out.println();
    }
}
```

```
import sys
from jinja2 import Template

def generate_java_code(features):
    with open('main_template.java.j2', 'r') as template_file:
        template = Template(template_file.read())
        java_code = template.render(features=features)
    return java_code

def write_java_code(java_code, output_file):
    with open(output_file, 'w') as java_file:
        java_file.write(java_code)

if __name__ == '__main__':
    features = {
        "Hello": True,
        "Beautiful": False,
        "Wonderful": True,
        "World": True
    }

    java_code = generate_java_code(features)
    output_file = 'Main.java'
    write_java_code(java_code, output_file)
    print(f"Java code generated in {output_file}")
```

https://github.com/acherm/progvary-withgpt/blob/main/featuretoggle-helloworld/ChatGPT-Unleash_JS_Feature_Toggles.md

Write a configurable JavaScript program that implements [...]
The four features should be implemented as feature toggles. I'd like to use Unleash, a feature toggle service, integrated with Gitlab



```
const { initialize } = require('unleash-client');

const unleash = initialize({
  appName: 'my-app-name',
  instanceId: 'my-instance-id',
  url: 'https://unleash.example.com/api/',
});

function printFeatures() {
  let output = '';

  if (unleash.isEnabled('hello')) {
    output += 'Hello';
  }

  if (unleash.isEnabled('beautiful')) {
    output += (output ? ' ' : '') + 'Beautiful';
  }

  if (unleash.isEnabled('wonderful')) {
    output += (output ? ' ' : '') + 'Wonderful';
  }

  if (unleash.isEnabled('world')) {
    output += (output ? ' ' : '') + 'World';
  }

  console.log(output);
}

unleash.on('ready', printFeatures);
unleash.on('update', printFeatures);
```

https://github.com/acherm/progvary-withgpt/blob/main/featuretoggle-helloworld/ChatGPT-Unleash_JS_Feature_Toggles.md

Write a configurable JavaScript program that implements [...] **The four features should be implemented as feature toggles. I'd like to use Unleash, a feature toggle service, integrated with Gitlab**

“would it be possible to adapt the program as a Web application... and provide a server-side that returns the output while the client side prints it”



```
const express = require('express');
const { initialize } = require('unleash-client');
const cors = require('cors');

const app = express();
const unleash = initialize({
  appName: 'my-app-name',
  instanceId: 'my-instance-id',
  url: 'https://unleash.example.com/api/',
});

app.use(cors());

app.get('/features', (req, res) => {
  let output = '';

  if (unleash.isEnabled('hello')) {
    output += 'Hello';
  }

  if (unleash.isEnabled('beautiful')) {
    output += (output ? ' ' : '') + 'Beautiful';
  }

  if (unleash.isEnabled('wonderful')) {
    output += (output ? ' ' : '') + 'Wonderful';
  }

  if (unleash.isEnabled('world')) {
    output += (output ? ' ' : '') + 'World';
  }

  res.json({ output });
});
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Feature Toggles Web App</title>
</head>
<body>
  <h1>Enabled Features:</h1>
  <div id="output"></div>

  <script>
    async function fetchFeatures() {
      const response = await fetch('http://localhost:3000/features');
      const data = await response.json();
      document.getElementById('output').innerText = data.output;
    }

    fetchFeatures();
  </script>
</body>
</html>
```

Remainder of the talk

How LLMs can be concretely and originally used for programming software variability

Variability implementations of a classical Hello world

Variability for reproducibility and floating-points

Transforming an unfamiliar code with an end-user, Web-based customization tool

Discussions (potential and limitations)

I have the following program. I'd like to create variations over this program using conditional compilation. Specifically, I'd like to create a first variation that consists in replacing the type long double by double... and another alternative variation that consists in replacing long double by float.

```
#include <stdio.h>
int main() {
    // Declare and initialize variables
    long double x = 1.0;
    long double y = 0x1p-53 + 0x1p-64;
    long double z = x + y;

    // Print the value of z
    printf("z = %a\n", z);
    return 0;
}
```



```
#include <stdio.h>

// Define the type to use as an alias
#ifdef USE_DOUBLE
    typedef double real;
#elif defined(USE_FLOAT)
    typedef float real;
#else
    typedef long double real;
#endif

int main() {
    // Declare and initialize variables
    real x = 1.0;
    real y = 0x1p-53 + 0x1p-64;
    real z = x + y;

    // Print the value of z
#ifdef USE_DOUBLE
    printf("z (using double) = %a\n", z);
#elif defined(USE_FLOAT)
    printf("z (using float) = %a\n", z);
#else
    printf("z (using long double) = %La\n", z);
#endif

    return 0;
}
```

```
#include <stdio.h>

// Define the type to use as an alias
#ifdef USE_DOUBLE
    typedef double real;
#elif defined(USE_FLOAT)
    typedef float real;
#else
    typedef long double real;
#endif

int main() {
    // Declare and initialize variables
    real x = 1.0;
    real y = 0x1p-53 + 0x1p-64;
    real z = x + y;

    // Print the value of z
#ifdef USE_DOUBLE
    printf("z (using double) = %a\n", z);
#elif defined(USE_FLOAT)
    printf("z (using float) = %a\n", z);
#else
    printf("z (using long double) = %La\n", z);
#endif

    return 0;
}
```

Write a Python script that compiles and executes the three variants above with and without the `-mfpmath=387` of `gcc`... Compile also with `clang`. Synthesize all results in a CSV containing the configuration (flag, with or without `-mfpmath=387`, compiler) and the different outputs. Please use `pandas` for writing the CSV... also the fieldnames should be `compiler`, `flags`, `type`, and `result`. The C file to compile is `assoc.c`

We then refine and ask:

Can you think about `gcc` flags or `clang` flags that could be combined/used and have an influence on floating point operations? Please implement them in the Python code.



```
def compile_and_run(compiler, flags):
    commands = [
        f"{compiler} {flags} -o original approx.c",
        f"{compiler} {flags} -DUSE_DOUBLE -o double_variation approx.c",
        f"{compiler} {flags} -DUSE_FLOAT -o float_variation approx.c",
    ]

    results = []
    for cmd in commands:
        subprocess.run(cmd, shell=True, check=True)

        binary = cmd.split()[-2]
        output = subprocess.check_output(f'./{binary}', shell=True).decode().strip()
        results.append(output)

    return results

def main():
    configurations = [
        ('gcc', ''),
        ('gcc', '-mfpmath=387'),
        ('gcc', '-ffast-math'),
        ('gcc', '-fno-signed-zeros'),
        ('gcc', '-fno-trapping-math'),
        ('gcc', '-fno-rounding-math'),
        ('clang', ''),
        ('clang', '-mno-sse'),
        ('clang', '-ffast-math'),
        ('clang', '-fno-signed-zeros'),
        ('clang', '-fno-trapping-math'),
        ('clang', '-fno-rounding-math'),
    ]

    data = []

    for compiler, flags in configurations:
        results = compile_and_run(compiler, flags)
        data.extend([
            {'compiler': compiler, 'flags': flags, 'type': 'original', 'result': results[0]},
            {'compiler': compiler, 'flags': flags, 'type': 'double', 'result': results[1]},
            {'compiler': compiler, 'flags': flags, 'type': 'float', 'result': results[2]},
        ])

    return data
```



```

def compile_and_run(compiler, flags):
    commands = [
        f"{compiler} {flags} -o original approx.c",
        f"{compiler} {flags} -DUSE_DOUBLE -o double_variation approx.c",
        f"{compiler} {flags} -DUSE_FLOAT -o float_variation approx.c",
    ]

    results = []
    for cmd in commands:
        subprocess.run(cmd, shell=True, check=True)

        binary = cmd.split()[-2]
        output = subprocess.check_output(f'./{binary}', shell=True).decode().strip()
        results.append(output)

    return results

def main():
    configurations = [
        ('gcc', ''),
        ('gcc', '-mfpmath=387'),
        ('gcc', '-ffast-math'),
        ('gcc', '-fno-signed-zeros'),
        ('gcc', '-fno-trapping-math'),
        ('gcc', '-fno-rounding-math'),
        ('clang', ''),
        ('clang', '-mno-sse'),
        ('clang', '-ffast-math'),
        ('clang', '-fno-signed-zeros'),
        ('clang', '-fno-trapping-math'),
        ('clang', '-fno-rounding-math'),
    ]

    data = []

    for compiler, flags in configurations:
        results = compile_and_run(compiler, flags)
        data.extend([
            {'compiler': compiler, 'flags': flags, 'type': 'original', 'result': results[0]},
            {'compiler': compiler, 'flags': flags, 'type': 'double', 'result': results[1]},
            {'compiler': compiler, 'flags': flags, 'type': 'float', 'result': results[2]},
        ])

```

compiler	flags	type	result
gcc		original	z (using long double) = 0x8.0000000000004p-3
gcc		double	z (using double) = 0x1.0000000000001p+0
gcc		float	z (using float) = 0x1p+0
gcc	-mfpmath=387	original	z (using long double) = 0x8.0000000000004p-3
gcc	-mfpmath=387	double	z (using double) = 0x1p+0
gcc	-mfpmath=387	float	z (using float) = 0x1p+0
gcc	-ffast-math	original	z (using long double) = 0x8.0000000000004p-3
gcc	-ffast-math	double	z (using double) = 0x1.0000000000001p+0
gcc	-ffast-math	float	z (using float) = 0x1p+0
gcc	-fno-signed-zeros	original	z (using long double) = 0x8.0000000000004p-3
gcc	-fno-signed-zeros	double	z (using double) = 0x1.0000000000001p+0
gcc	-fno-signed-zeros	float	z (using float) = 0x1p+0
gcc	-fno-trapping-math	original	z (using long double) = 0x8.0000000000004p-3
gcc	-fno-trapping-math	double	z (using double) = 0x1.0000000000001p+0
gcc	-fno-trapping-math	float	z (using float) = 0x1p+0
gcc	-fno-rounding-math	original	z (using long double) = 0x8.0000000000004p-3
gcc	-fno-rounding-math	double	z (using double) = 0x1.0000000000001p+0
gcc	-fno-rounding-math	float	z (using float) = 0x1p+0
clang		original	z (using long double) = 0x8.0000000000004p-3
clang		double	z (using double) = 0x1.0000000000001p+0

retrieve the result of S. Boldo et al.

Remainder of the talk

How LLMs can be concretely and originally used for programming software variability

- Variability implementations of a classical Hello world

- Variability for reproducibility and floating-points

- Transforming an unfamiliar code with an end-user,
Web-based customization tool**

Discussions (potential and limitations)

Here is a TikZ code:

```
\tikzset{%
something/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={((0,-1/3))}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={((1/2*\i,9/4))}, rotate=45*\i]{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{[shift={(-1/2,-2)}, rotate= 10] ellipse [x radius=1/3, y radius=5/4]}
{[shift={ (1/2,-2)}, rotate=-10] ellipse [x radius=1/3, y radius=5/4]};
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
( 1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0,12/8) ellipse [x radius=1/10, y radius=1/5]
(0,12/8+1/9) ellipse [x radius=1/5, y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={((1/2*\i,2))}, rotate=35*\i]{
\clip [overlay] (0, 1/7) ellipse [radius=2/7];
\clip [overlay] (0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
};
\scoped{
\clip (-1,-2) rectangle ++(2,1);
\fill [BlueGrey900] (0,-2) ellipse [radius=1/2];
\fill [Grey100]
(-1/2,-2) ellipse [x radius=1/3, y radius=1/4]
( 1/2,-2) ellipse [x radius=1/3, y radius=1/4];
};
\foreach \i in {-1,1}
\foreach \j in {-1,0,1}
\fill [Grey100, shift={((0,11/8))}, xscale=\i, rotate=\j*15,
shift=(0:1/2)]
ellipse [x radius=1/3, y radius=1/64];
},
```

Here is a TikZ code:

```
\tikzset{%
something/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={(0,-1/3)}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,9/4)}, rotate=45*\i]{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{[shift={(-1/2,-2)}, rotate= 10] ellipse [x radius=1/3, y radius=5/4]}
{[shift={( 1/2,-2)}, rotate=-10] ellipse [x radius=1/3, y radius=5/4]};
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
```

Here is a TikZ code:

```
\tikzset{%
cat.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={(0,-1/3)}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:1/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,9/4)}, rotate=45*\i]{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0, 15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{{shift={(-1/2,-2)}, rotate= 10} ellipse [x radius=1/3, y radius=5/4]}
{{shift={( 1/2,-2)}, rotate=-10} ellipse [x radius=1/3, y radius=5/4]};
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
( 1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0, 12/8) ellipse [x radius=1/10, y radius=1/5]
(0, 12/8+1/9) ellipse [x radius=1/5, y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,2)}, rotate=35*\i]{
\clip [overlay] (0, 1/7) ellipse [radius=2/7];
\clip [overlay] (0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
};
\scoped{
\clip (-1,-2) rectangle ++(2,1);
\fill [BlueGrey900] (0,-2) ellipse [radius=1/2];
\fill [Grey100]
(-1/2,-2) ellipse [x radius=1/3, y radius=1/4]
( 1/2,-2) ellipse [x radius=1/3, y radius=1/4];
};
\foreach \i in {-1,1}
\foreach \j in {-1,0,1}
\fill [Grey100, shift={(0,11/8)}, xscale=\i, rotate=\j*15,
shift=(0:1/2)]
ellipse [x radius=1/3, y radius=1/64];
},
```



cat

Here is a TikZ code:

```
\tikzset{%
cat.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={(0,-1/3)}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,9/4)}, rotate=45*\i]{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{[shift={(-1/2,-2)}, rotate= 10] ellipse [x radius=1/3, y radius
{[shift={( 1/2,-2)}, rotate=-10] ellipse [x radius=1/3, y radius
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
( 1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0,12/8) ellipse [x radius=1/10, y radius=1/5]
(0,12/8+1/9) ellipse [x radius=1/5 , y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,2)}, rotate=35*\i]{
\clip [overlay] (0, 1/7) ellipse [radius=2/7];
\clip [overlay] (0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
```

I'd like to change the color of the cat's tail



cat

I'd like to change the color of the cat's tail

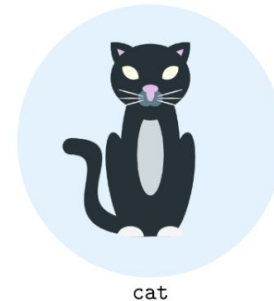
Here is a TikZ code:

```
\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={{(0,-1/3)}}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={{(1/2*\i,9/4)}, rotate=45*\i}{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{[shift={{(-1/2,-2)}, rotate= 10} ellipse [x radius=1/3, y radius=5/4]}
{[shift={{( 1/2,-2)}, rotate=-10} ellipse [x radius=1/3, y radius=5/4]};
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
(1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
}
```

(LLM) To change the color of the tail, you need to modify the fill color specified in the first 'fill' command, which draws the tail shape. In the provided code, the tail color is set to 'BlueGrey900'. You can replace this color with any other color you'd like. Here's an example with the tail color changed to 'red':

```
\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={{(0,-1/3)}}}
\useasboundingbox (-1,-1) (1,2);
\fill [red] (0,-2) % Change the color here
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
% The rest of the code remains the same
```

added by
ChatGPT!



change as much as possible elements, applying different styles, colors, and transformations. Give me the new TikZ code and explain me the changes

Can you write a Python program that would apply the 9 proposed changes? I recommend using a templating engine in such a way the 9 changes can be applied and configured

```
cat/.pic={
  \tikzset{x=3cm/5,y=3cm/5,shift={{(0,-1/3)}}}
  \useasboundingbox (-1,-1) (1,2);
  \fill [<< tail_color >>] (0,-2)
    .. controls ++(180:3) and ++(0:5/4) .. (-2,0)
    arc (270:90:1/5)
    .. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
  ...
  \fill [<< central_body_color >>] ellipse [x radius
    =3/4, y radius=2];
  \fill [<< inner_body_color >>] ellipse [x radius=1/3,
    y radius=1];
  \fill [<< head_paws_color >>]
  ...
}
```

```
import random

def random_rgb_color():
    return f"rgb({random.randint(0, 255)},{random.randint(0, 255)},{random.randint(0, 255)})"

def random_scale(min_value=0.5, max_value=2.0):
    return random.uniform(min_value, max_value)

output = template.render(
    main_body_color=random_rgb_color(),
    head_stripe_color=random_rgb_color(),
    inner_stripe_color=random_rgb_color(),
    head_stripe_scale=random_scale(),
    head_surround_color=random_rgb_color(),
    head_inner_color=random_rgb_color(),
    nose_main_color=random_rgb_color(),
    eyes_color=random_rgb_color(),
    nose_color=random_rgb_color(),
    whisker_spots_color=random_rgb_color(),
    whisker_spots_scale=random_scale(),
    face_lower_part_color=random_rgb_color(),
    face_inner_ellipses_color=random_rgb_color(),
    whiskers_color=random_rgb_color(),
)

print(output)
```



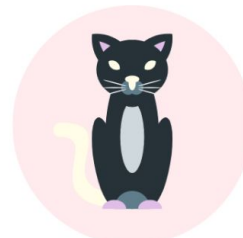
change as much as possible elements, applying different styles, colors, and transformations. Give me the new TikZ code and explain me the changes

Can you write a Python program that would apply the 9 proposed changes? I recommend using a templating engine in such a way the 9 changes can be applied and configured

```
cat/.pic={
  \tikzset{x=3cm/5,y=3cm/5,shift={{(0,-1/3)}}}
  \useasboundingbox (-1,-1) (1,2);
  \fill [<< tail_color >>] (0,-2)
    .. controls ++(180:3) and ++(0:5/4) .. (-2,0)
    arc (270:90:1/5)
    .. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
  ...
  \fill [<< central_body_color >>] ellipse [x radius
    =3/4, y radius=2];
  \fill [<< inner_body_color >>] ellipse [x radius=1/3,
    y radius=1];
  \fill [<< head_paws_color >>]
  ...
}
```

Customization			
Ears, tail, and whisker	Body	Head and paws	Ear and eyes (scale)
tail : <input type="text" value="Yellow50"/>	central body : <input type="text" value="BlueGrey900"/>	head paws : <input type="text" value="BlueGrey900"/>	outer ear scale: <input type="range" value="50"/>
outer ear : <input type="text" value="BlueGrey900"/>	inner body : <input type="text" value="BlueGrey100"/>	bottom paws : <input type="text" value="Purple100"/>	eyes scale: <input type="range" value="50"/>
inner ear : <input type="text" value="Purple100"/>	lower part : <input type="text" value="BlueGrey500"/>	outer nose : <input type="text" value="BlueGrey500"/>	
whiskers : <input type="text" value="Grey100"/>		nose : <input type="text" value="Yellow50"/>	
		eyes : <input type="text" value="Yellow50"/>	

Beautify the cat!



cat

vs original



cat

Replication

```

\tikzset{%
chimpanzee/.pic={
\tikzset{x=3cm/3.5,y=3cm/3.5, shift={(0,1/4)}, rotate=<< rotation_angle >>, scale=<< scaling_factor >>}
\fill [<< left_ear_outer_color >>] (165:5/4 and 1) ellipse [radius=1/2];
\fill [<< left_ear_inner_color >>] (165:5/4 and 1) ellipse [radius=1/3];
\fill [<< left_overlay_color >>]
(0,-1/2) ellipse [x radius=1, y radius=4/3]
(0, 0) ellipse [x radius=4/3, y radius=1];
\scoped{
\clip(0,-2) rectangle(9/5,1);
\fill [<< right_ear_outer_color >>] (15:5/4 and 1) ellipse [radius=1/2];
\fill [<< right_ear_inner_color >>] (15:5/4 and 1) ellipse [radius=1/3];
\fill [<< right_overlay_color >>]
(0,-1/2) ellipse [x radius=1, y radius=4/3]
(0, 0) ellipse [x radius=4/3, y radius=1];
}
\fill [<< left_face_color >>] (0,-2/3) ellipse [x radius=5/6, y radius= 1];
\fill [<< left_face_color >>, xscale=-1] (1/2, 0) ellipse [x radius=3/5, y radius=2/3];
\fill [<< left_face_color >>, xscale= 1] (1/2, 0) ellipse [x radius=3/5, y radius=2/3];
\scoped{
\clip(0,-2) rectangle(9/5,1);
\fill [<< right_face_color >>] (0,-2/3) ellipse [x radius=5/6, y radius= 1];
\fill [<< right_face_color >>, xscale=-1] (1/2, 0) ellipse [x radius=3/5, y radius=2/3];
\fill [<< right_face_color >>, xscale= 1] (1/2, 0) ellipse [x radius=3/5, y radius=2/3];
}
}
\foreach \i in {-1,1}
\fill [<< eye_color >>, xscale=\i] (1/2,0) ellipse [radius=1/4];
\fill [<< nose_color >>]
(-1/7,-2/3) ellipse [radius=1/9]
( 1/7,-2/3) ellipse [radius=1/9];
\fill [<< mouth_outline_color >>]
(-1/2,-1) arc (180:360:1/2 and 1/8) arc (360:180:1/2 and 1/4);
}
}

```



Replication (beyond TikZ!)

Customize Your Squid

- Crazy Tentacles
- Prominent Mouth
- Aggressive Shading
- Original Tentacles

Outer Eye Color:

Inner Eye Color:

Number of Extra Spots:

Random color

for spots?

Generate Squid

Your Squid

```
\documentclass[tikz, border=5]{standalone}
\usepackage[prefix={xcolor-material}]
\pgfmathdeclarerandomlist{material}{%
{Red}{Blue}{Green}}
\tikzset{%
half clip/.code={
\clip (0, -256) rectangle (256, 256);
},
color/.code=\colorlet{fill color}{#1},
```

Show More

HTML Configuration

Layout:

Horizontal

Border for Image:

SVG Configuration

Moving the squid...

Size of the squid:

Padding:

Flip/Mirror:

Rotation:

Figure 2: End-user customization of (1) TikZ code (squid) (2) HTML/CSS (Website layout) (3) SVG (post-process over the resulting image). The Python template-based generator produces a variant of the TikZ code, which is displayed, but end-users do not interact with it.

Wait wait... **Feature location:** Despite a positive experience here with the cat TikZ, replications over different TikZ code representing squid, chimpanzee, or dogs show that LLMs are in fact inaccurate. LLMs can be lucky to guess which portion of code is associated to a feature, but clearly not 100% of the time. It is problematic since then the mapping between features and code element should then be revised, after manual investigation.

I'd like to change the color of the cat's tail

(LLM) To change the color of the tail, you need to modify the color specified in the first 'fill' command, which draws the shape. In the provided code, the tail color is set to 'BlueGrey'. You can replace this color with any other color you'd like. Here is an example with the tail color changed to 'red':

```
\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={{(0,-1/3)}}}
\useasboundingbox (-1,-1)(1,2);
\fill [red](0,-2) % Change the color here
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5)
% The rest of the code remains the same
```



cat

```
Here is a TikZ code:
\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={{(0,-1/3)}}}
\useasboundingbox (-1,-1)(1,2);
\fill [BlueGrey900](0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={{1/2*\i,9/4}}, rotate=45*\i]{
\clip [overlay](0,5/9) ellipse [radius=8/9];
\clip [overlay](0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay](0,7/9) ellipse [radius=10/11];
\clip [overlay](0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0,8/6) ellipse [x radius=1/2, y radius=1/2]
{shift={{(-1/2,-2)}, rotate=10} ellipse [x radius=1/3, y radius=5/4]}
{shift={{1/2,-2)}, rotate=-10} ellipse [x radius=1/3, y radius=5/4]}
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
(1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0,12/8) ellipse [x radius=1/10, y radius=1/5]
(0,12/8+1/9) ellipse [x radius=1/5, y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={{1/2*\i,2}}, rotate=35*\i]{
\clip [overlay](0,1/7) ellipse [radius=2/7];
\clip [overlay](0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
};
\scoped{
\clip (-1,-2) rectangle ++(2,1);
\fill [BlueGrey900](0,-2) ellipse [radius=1/2];
\fill [Grey100]
(-1/2,-2) ellipse [x radius=1/3, y radius=1/4]
(1/2,-2) ellipse [x radius=1/3, y radius=1/4];
};
\foreach \i in {-1,1}
\foreach \j in {-1,0,1}
\fill [Grey100,shift={{0,11/8}}, xscale=\i, rotate=\j*15,
shift=(0:1/2)]
ellipse [x radius=1/3, y radius=1/64];
};
```



cat

Here is a TikZ code:

```

\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={(0,-1/3)}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,9/4)}, rotate=45*\i]{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{shift={(-1/2,-2)}, rotate= -10} ellipse [x radius=1/3, y radius=5/4];
{shift={( 1/2,-2)}, rotate=-10} ellipse [x radius=1/3, y radius=5/4];
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
( 1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0,12/8) ellipse [x radius=1/10, y radius=1/5]
(0,12/8+1/9) ellipse [x radius=1/5, y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,2)}, rotate=35*\i]{
\clip [overlay] (0, 1/7) ellipse [radius=2/7];
\clip [overlay] (0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
};
\scoped{
\clip (-1,-2) rectangle ++(2,1);
\fill [BlueGrey900] (0,-2) ellipse [radius=1/2];
\fill [Grey100]
(-1/2,-2) ellipse [x radius=1/3, y radius=1/4]
( 1/2,-2) ellipse [x radius=1/3, y radius=1/4];
};
\foreach \i in {-1,1}
\foreach \j in {-1,0,1}
\fill [Grey100, shift={(0,11/8)}, xscale=\i, rotate=\j*15,
shift=(0:1/2)]
ellipse [x radius=1/3, y radius=1/64];
}

```

due to inaccurate feature location or/and incorrect code synthesis



I'd like to change the color of the cat's tail in red



cat



cat

I'd like to change the color of the cat's tail in red

MA You



cat

visual abilities to implement a "testing oracle"

what is the color of the cat's tail? is it red?

🌀 ChatGPT

The cat's tail is black, not red. The only red part on the cat is its nose.

MA You

what are eyes colors?

🌀 ChatGPT

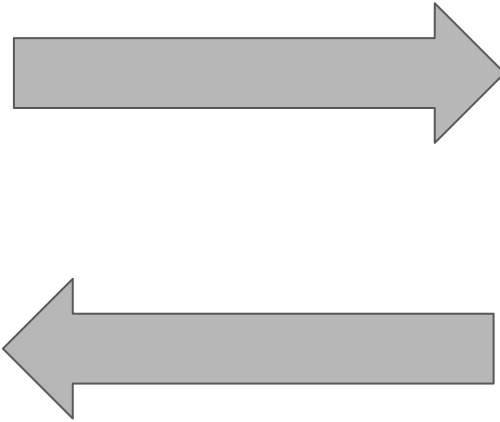
The cat's eyes are red.

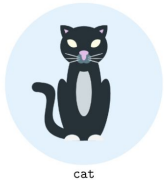
Here is a TikZ code:

```

\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={(0,-1/3)}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,9/4)}, rotate=45*\i]{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{shift={(-1/2,-2)}, rotate= 10} ellipse [x radius=1/3, y radius=5/4];
{shift={( 1/2,-2)}, rotate=-10} ellipse [x radius=1/3, y radius=5/4];
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
( 1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0,12/8) ellipse [x radius=1/10, y radius=1/5]
(0,12/8+1/9) ellipse [x radius=1/5, y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,2)}, rotate=35*\i]{
\clip [overlay] (0, 1/7) ellipse [radius=2/7];
\clip [overlay] (0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
};
\scoped{
\clip (-1,-2) rectangle ++(2,1);
\fill [BlueGrey900] (0,-2) ellipse [radius=1/2];
\fill [Grey100]
(-1/2,-2) ellipse [x radius=1/3, y radius=1/4]
( 1/2,-2) ellipse [x radius=1/3, y radius=1/4];
};
\foreach \i in {-1,1}
\foreach \j in {-1,0,1}
\fill [Grey100, shift={(0,11/8)}, xscale=\i, rotate=\j*15,
shift=(0:1/2)]
ellipse [x radius=1/3, y radius=1/64];
}

```

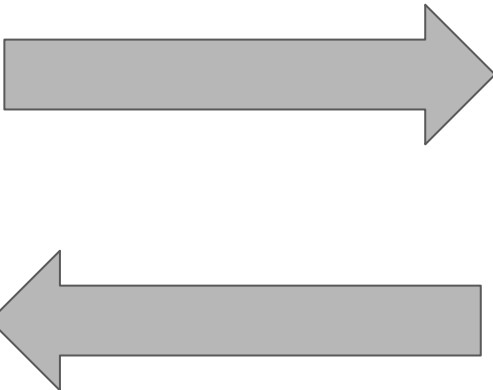




I'd like to change the color of the cat's tail in red

Here is a TikZ code:

```
\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={(0,-1/3)}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,9/4)}, rotate=45*\i]{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{shift={(-1/2,-2)}, rotate= 10} ellipse [x radius=1/3, y radius=5/4];
{shift={( 1/2,-2)}, rotate=-10} ellipse [x radius=1/3, y radius=5/4];
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
( 1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0,12/8) ellipse [x radius=1/10, y radius=1/5]
(0,12/8+1/9) ellipse [x radius=1/5, y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,2)}, rotate=35*\i]{
\clip [overlay] (0, 1/7) ellipse [radius=2/7];
\clip [overlay] (0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
};
\scoped{
\clip (-1,-2) rectangle ++(2,1);
\fill [BlueGrey900] (0,-2) ellipse [radius=1/2];
\fill [Grey100]
(-1/2,-2) ellipse [x radius=1/3, y radius=1/4]
( 1/2,-2) ellipse [x radius=1/3, y radius=1/4];
};
\foreach \i in {-1,1}
\foreach \j in {-1,0,1}
\fill [Grey100, shift={(0,11/8)}, xscale=\i, rotate=\j*15,
shift=(0:1/2)]
ellipse [x radius=1/3, y radius=1/64];
};
```



```
[Loading MPS to PDF converter (version 2006.09.02).]
) (/usr/share/texlive/texmf-dist/tex/latex/epstopdf-pkg/epstopdf-base.sty
(/usr/share/texlive/texmf-dist/tex/latex/latexconfig/epstopdf-sys.cfg))
(/usr/share/texlive/texmf-dist/tex/generic/pgf/frontendlayer/tikz/libraries/tik
zlibrarypositioning.code.tex)
! Argument of \tikz@scan@no@calculator has an extra }.
<inserted text>
\par
\l.133 }}
? □
```

compilers/interpreters'
errors, logs,
stacktraces, etc.



cat

Here is a TikZ code:

```

\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={(0,-1/3)}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (0,-2+2/5);
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,9/4)}, rotate=45*\i]{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{shift={(-1/2,-2)}, rotate= 10} ellipse [x radius=1/3, y radius=5/4];
{shift={( 1/2,-2)}, rotate=-10} ellipse [x radius=1/3, y radius=5/4];
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
( 1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0,12/8) ellipse [x radius=1/10, y radius=1/5]
(0,12/8+1/9) ellipse [x radius=1/5, y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={(1/2*\i,2)}, rotate=35*\i]{
\clip [overlay] (0, 1/7) ellipse [radius=2/7];
\clip [overlay] (0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
};
\scoped{
\clip (-1,-2) rectangle ++(2,1);
\fill [BlueGrey900] (0,-2) ellipse [radius=1/2];
\fill [Grey100]
(-1/2,-2) ellipse [x radius=1/3, y radius=1/4]
( 1/2,-2) ellipse [x radius=1/3, y radius=1/4];
};
\foreach \i in {-1,1}
\foreach \j in {-1,0,1}
\fill [Grey100, shift={(0,11/8)}, xscale=\i, rotate=\j*15,
shift=(0:1/2)]
ellipse [x radius=1/3, y radius=1/64];
}

```



“I’d like to have a better red, more dark...” (kids)

Customization

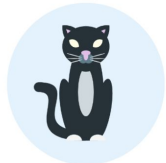
Ears, tail, and whisker

tail :	<input type="text" value="red"/>
outer ear :	<input type="text" value="BlueGrey900"/>
inner ear :	<input type="text" value="Purple100"/>
whiskers :	<input type="text" value="Grey100"/>

Beautify the cat!



cat



cat

multi-modal, end-user programming of variability



self-improvements

```

[Loading MPS to PDF converter (version 2006.09.02).]
) (/usr/share/texlive/texmf-dist/tex/latex/epstopdf-pkg/epstopdf-base.sty
(/usr/share/texlive/texmf-dist/tex/latex/latexconfig/epstopdf-sys.cfg)
(/usr/share/texlive/texmf-dist/tex/generic/pgf/frontendlayer/tikz/libraries/tik
zlibrarypositioning.code.tex)
! Argument of \tikz@scan@node@calculator has an extra }.
<inserted text>
\par
l.133 }}
? []

```



cat

what is the color of the cat's tail? it is red?

ChatGPT

The cat's tail is black, not red. The only red part on the cat is its nose.

You

what are eyes colors?

ChatGPT

The cat's eyes are red.

“I'd like to have a better red, more dark...” (kids)

```

Here is a TikZ code:
\tikzset{%
cat/.pic={
\tikzset{x=3cm/5,y=3cm/5,shift={{(0,-1/3)}}
\useasboundingbox (-1,-1) (1,2);
\fill [BlueGrey900] (0,-2)
.. controls ++(180:3) and ++(0:5/4) .. (-2,0)
arc (270:90:1/5)
.. controls ++(0:2) and ++(180:11/4) .. (-2,0+2/5);
\foreach \i in {-1,1}
\scoped[shift={{(1/2*\i,9/4)}, rotate=45*\i}{
\clip [overlay] (0, 5/9) ellipse [radius=8/9];
\clip [overlay] (0,-5/9) ellipse [radius=8/9];
\fill [BlueGrey900] ellipse [radius=1];
\clip [overlay] (0, 7/9) ellipse [radius=10/11];
\clip [overlay] (0,-7/9) ellipse [radius=10/11];
\fill [Purple100] ellipse [radius=1];
};
\fill [BlueGrey900] ellipse [x radius=3/4, y radius=2];
\fill [BlueGrey100] ellipse [x radius=1/3, y radius=1];
\fill [BlueGrey900]
(0,15/8) ellipse [x radius=1, y radius=5/6]
(0, 8/6) ellipse [x radius=1/2, y radius=1/2]
{shift={{(-1/2,-2)}, rotate=10} ellipse [x radius=1/3, y radius=5/4];
{shift={{ 1/2,-2)}, rotate=-10} ellipse [x radius=1/3, y radius=5/4];
\fill [BlueGrey500]
(-1/9,11/8) ellipse [x radius=1/5, y radius=1/5]
( 1/9,11/8) ellipse [x radius=1/5, y radius=1/5];
\fill [Purple100]
(0,12/8) ellipse [x radius=1/10, y radius=1/5]
(0,12/8+1/9) ellipse [x radius=1/5 , y radius=1/10];
\foreach \i in {-1,1}
\scoped[shift={{(1/2*\i,2)}, rotate=35*\i}{
\clip [overlay] (0, 1/7) ellipse [radius=2/7];
\clip [overlay] (0,-1/7) ellipse [radius=2/7];
\fill [Yellow50] ellipse [radius=1];
};
\scoped{
\clip (-1,-2) rectangle ++(2,1);
\fill [BlueGrey900] (0,-2) ellipse [radius=1/2];
\fill [Grey100]
(-1/2,-2) ellipse [x radius=1/3, y radius=1/4]
( 1/2,-2) ellipse [x radius=1/3, y radius=1/4];
};
\foreach \i in {-1,1}
\foreach \j in {-1,0,1}
\fill [Grey100, shift={{(0,11/8)}, xscale=\i, rotate=\j*15,
shift=(0:1/2)}
ellipse [x radius=1/3, y radius=1/64];
};
}

```

Plan

3 AI for software variability and configurations

LLM for programming variability

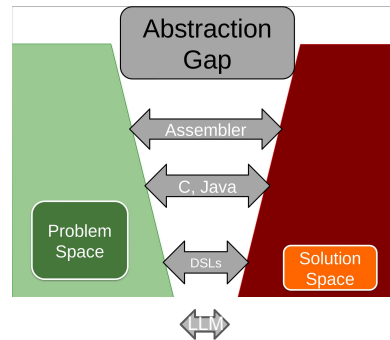
Discussion

Discussions (potential)

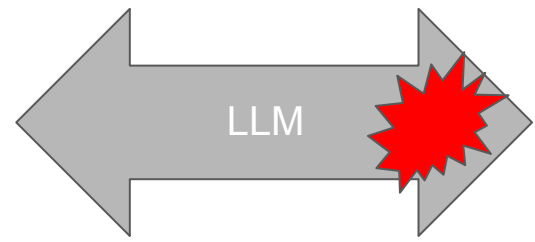
LLMs can assist developers in implementing variability in different programming languages (C, Rust, Java, TikZ, etc.) and mechanisms (conditional compilation, feature toggles, command-line parameters, template, etc.).

Synthesis of configuration files/deployment script/... (more for CI/CD/DevOps/Cloud)

With "features as prompts", there is hope to raise the level of abstraction, increase automation, and bring more flexibility when synthesizing and exploring software variants.



Discussions (potential/limitations)



LLMs are by construction stochastic, non-determinist and highly sensitive to prompt variations – and so are corresponding implementations of features and variability.

I have the following program. I'd like to create variations over this program using conditional compilation. Specifically, I'd like to create a first variation that consists in replacing the type long double by double... and another alternative variation that consists in replacing long double by float.

```
#include <stdio.h>
int main() {
    // Declare and initialize variables
    long double x = 1.0;
    long double y = 0x1p-53 + 0x1p-64;
    long double z = x + y;

    // Print the value of z
    printf("z = %a\n", z);
    return 0;
}
```

```
#include <stdio.h>

// Define a macro to choose the type of variable
// 1: float
// 2: double
// 3: long double
#define VARIABLE_TYPE 3

int main() {
    // Declare and initialize variables
    #if VARIABLE_TYPE == 1
        float x = 1.0f;
        float y = 0x1p-24f + 0x1p-32f;
    #elif VARIABLE_TYPE == 2
        double x = 1.0;
        double y = 0x1p-53 + 0x1p-64;
    #elif VARIABLE_TYPE == 3
        long double x = 1.0;
        long double y = 0x1p-53 + 0x1p-64;
    #endif

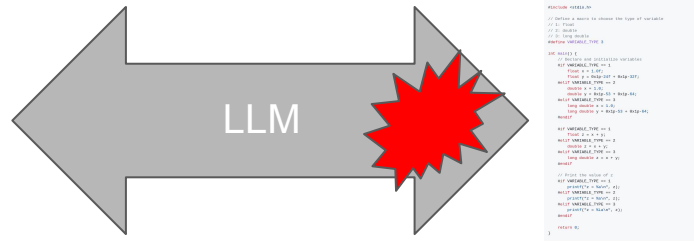
    #if VARIABLE_TYPE == 1
        float z = x + y;
    #elif VARIABLE_TYPE == 2
        double z = x + y;
    #elif VARIABLE_TYPE == 3
        long double z = x + y;
    #endif

    // Print the value of z
    #if VARIABLE_TYPE == 1
        printf("z = %a\n", z);
    #elif VARIABLE_TYPE == 2
        printf("z = %a\n", z);
    #elif VARIABLE_TYPE == 3
        printf("z = %La\n", z);
    #endif

    return 0;
}
```

https://github.com/acherm/progvary-withgpt/blob/main/varyfloatinC/ChatGPT-Variable_Type_Alternatives.md

Discussions (limitations)

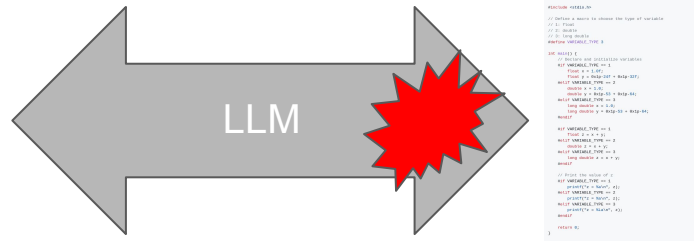


LLMs are by construction stochastic, non-determinist and highly sensitive to prompt variations – and so are corresponding implementations of features and variability.

In particular, LLMs can produce different programs out of the same exact prompt. Unfortunately, some of these programs can be incomplete or simply wrong, introducing serious variability errors.

Potential: LLMs act as **assistant** that recommend some possible implementations of variability. It is the role of programmers to review the recommendation and either selects another recommendation or manually implement variability.

Discussions (limitations)

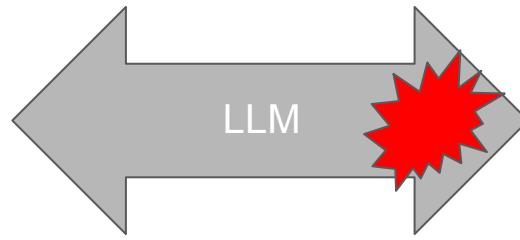


LLMs are by construction stochastic, non-determinist and highly sensitive to prompt variations – and so are corresponding implementations of features and variability.

#1 A variability benchmark for LLM-based assistant (for assessing accuracy and robustness)

Challenge: In our case, we target comprehensive programming scenarios that (1) require combining different pieces of code and (2) call to combine different languages and technologies for realizing variability. Not easy to automate the “goodness” assessment of some solutions

Discussions (limitations)



```
FUNCTION generate_text(prompt, temperature, max_tokens):
    """Generate text using the OpenAI GPT-3 API"""
    # OpenAI API endpoint
    url = "https://api.openai.com/v1/completions"

    # Headers
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {api_key}"
    }

    # Payload
    payload = {
        "model": "gpt-3.5-turbo",
        "prompt": prompt,
        "max_tokens": max_tokens,
        "temperature": temperature
    }

    # Send request
    response = requests.post(url, headers=headers, json=payload)

    # Parse response
    data = response.json()

    # Extract text
    text = data["choices"][0]["text"]

    return text
```

LLMs are by construction stochastic, non-determinist and highly sensitive to prompt variations – and so are corresponding implementations of features and variability.

#2 Improving the robustness of LLMs

Challenge: how to properly formulate instructions/prompts for programming variability... informality of NLP can backfire. Dedicated “variability language” for LLM? Specialization of LLMs for variability tasks?

Symbolic reasoning

Modelling and reverse engineering variability/configuration space

Statistical machine learning

Predicting properties of configurations and variants

Generative AI and large language model

Programming software variability

```

// Define the type to use as an alias
using var_double = double;

// Define the struct
struct Config {
    bool fast_math;
    bool use_double;
    bool use_float;
};

// Define the function
void compile(Config c) {
    // ...
}

// Define the main function
int main() {
    // ...
}

```

Write a Python script that compiles and executes the three variants above with and without the `-mfloat=387` of gcc... Compile also with clang. Synthesize all results in a CSV containing the configuration (flag, with or without `-mfloat=387`, compiler) and the different outputs. Please use pandas for writing the CSV... also the fieldnames should be compiler, flag, type, and result. The C file to compile is assoc.

We then refine and ask:
Can you think about gcc flags or clang flags that could be combined/used and have an influence on floating point operations? Please implement them in the Python code.

```

def compile_and_run(compiler, flags):
    commands = [
        f"{compiler} {flags} -o original approx.c",
        f"{compiler} {flags} -DUSE_DOUBLE -o double_variation approx.c",
        f"{compiler} {flags} -DUSE_FLOAT -o float_variation approx.c",
    ]

    results = []
    for cmd in commands:
        subprocess.run(cmd, shell=True, check=True)

    binary = os.path.join(".", "binary")
    output = subprocess.check_output(f"{binary}, shell=True).decode('strip')
    results.append(output)

    return results

def main():
    configurations = [
        ('gcc', ''),
        ('gcc', '-mfloat=387'),
        ('gcc', '-fast-math'),
        ('gcc', '-fno-signed-zero'),
        ('gcc', '-fno-trapping-math'),
        ('gcc', '-fno-rounding-math'),
        ('clang', ''),
        ('clang', '-fno-math-errno'),
        ('clang', '-fast-math'),
        ('clang', '-fno-signed-zero'),
        ('clang', '-fno-trapping-math'),
        ('clang', '-fno-rounding-math'),
    ]

    data = []

    for compiler, flags in configurations:
        results = compile_and_run(compiler, flags)
        data.append([
            compiler, flags, 'type': 'original', 'result': results[0],
            compiler, flags, 'type': 'double', 'result': results[1],
            compiler, flags, 'type': 'float', 'result': results[2],
        ])

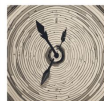
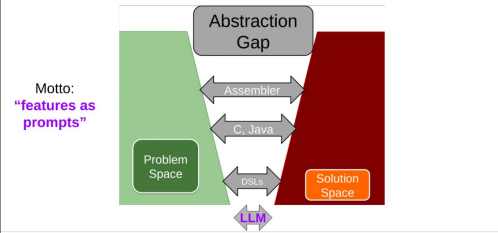
```



Generation of software variants

LLMs can assist developers in implementing variability in different programming languages (C, Rust, Java, TikZ, etc.) and mechanisms (conditional compilation, feature toggles, command-line parameters, template, etc.). With "features as prompts", there is hope to raise the level of abstraction, increase automation, and bring more flexibility when synthesizing and exploring software variants.

Hypothesis: Large language models (LLMs) act as a new variability compiler capable of transforming a high-level specification ("prompt") into variable code, features, generators, configurable systems, etc. written in a given technological space.

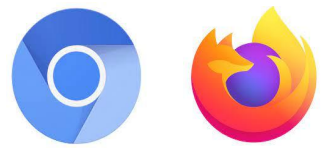


multi-modal, end-user programming of variability



self-improvements

"I'd like to have a better red, more dark..." (kids)



deep software variability

hardware variability

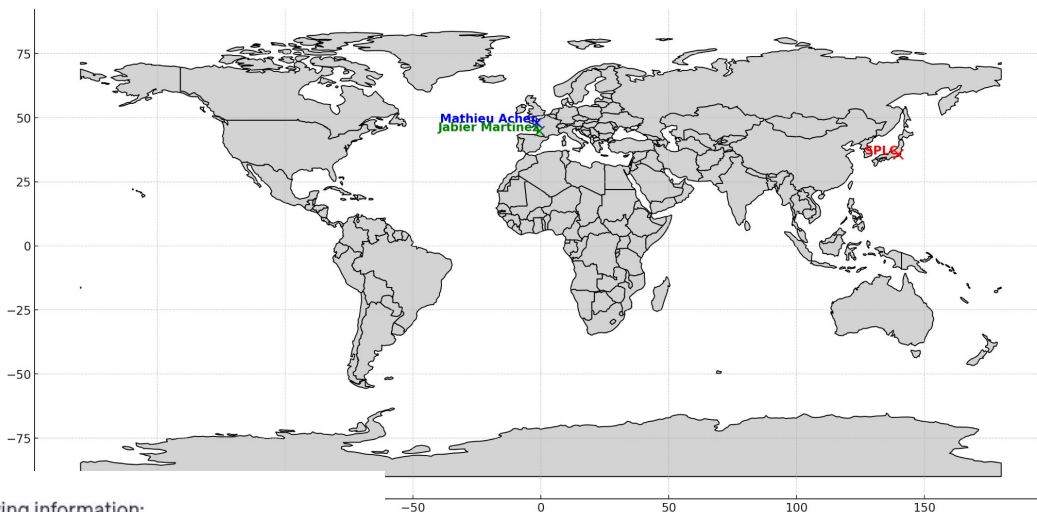
15,000+ options

thousands of compiler flags
and compile-time options

dozens of preferences

100+ command-line parameters

1000+ feature toggles



MA

I will present a paper with the following information:

Generative AI for Reengineering Variants into Software Product

Lines: An Experience Report

Mathieu Acher

Univ Rennes, IRISA, Inria, CNRS, IUF

Rennes, France

mathieu.acher@irisa.fr

Jabier Martinez

Tecnalia, Basque Research and

Technology Alliance (BRTA)

Pessac, France

jabier.martinez@tecnalia.com

The presentation will happen in Tokyo (Japan).

Can you depict a map showing the location of the authors as well as Tokyo?

The map should somehow include the title of the paper as well

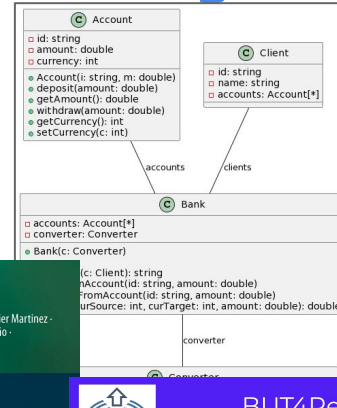
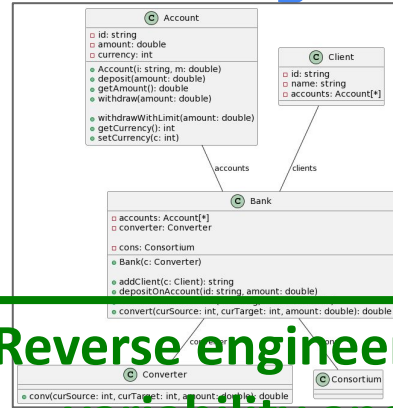
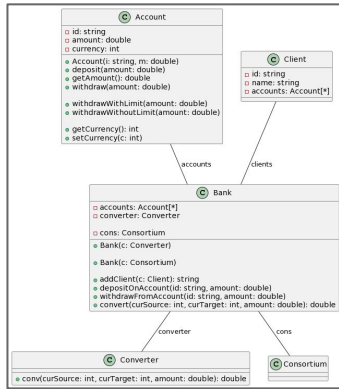
```
# Plot the map with enhanced visual features
fig, ax = plt.subplots(figsize=(15, 10))
world.plot(ax=ax, color='lightgray', edgecolor='black')

# Plot the locations with custom colors
colors = ['blue', 'green', 'red']
for i, row in gdf_approx.iterrows():
    ax.scatter(row['Longitude'], row['Latitude'], color=colors[i], s=100)
    ax.text(row['Longitude'], row['Latitude'], row['Location'], fontsize=12,

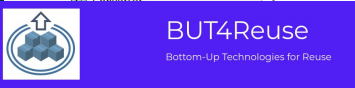
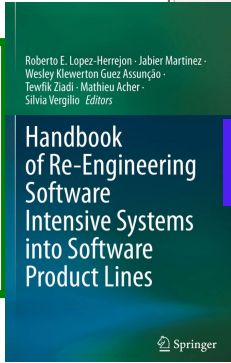
# Add title with bold font and larger font size
plt.title("Generative AI for Reengineering Variants into Software Product Li

# Show the enhanced map
plt.tight_layout()
plt.show()
```

Problem: Given a set of variants, how to synthesize a program that can be configured to retrieve original variants?



Reverse engineering
variability and
reusable assets



e4CompareFramework
ECCO tool

Error-prone
Time-consuming

SPL = feature model + annotated program (eg template-based generator)
Expected properties: soundness/completeness/meaningful set of features
(for configuration or maintenance/evolution/expansion)