Formal Methods Lab

Google Colab is a great platform for implementing formal methods-related simulations and verifying models using Python and other compatible tools. Below are **program examples** and steps for implementing each assignment in Colab. These examples focus on Python-based libraries like **SimPy**, **Z3 Solver**, and **PyNuSMV**, as well as basic coding techniques for formal methods.

Setting Up Google Colab

1. Open Google Colab.
2. Create a new notebook.
3. Use the Python interpreter to install any required libraries with `!pip install` commands.

Lab 1: Introduction to Formal Methods Tools

Objective: Simulate a vending machine using Python.

```
# Install SimPy (if not installed already)
!pip install simpy

import simpy

# Vending Machine Model
def vending_machine(env):
    print(f"Vending Machine Ready at {env.now}")
    while True:
        print(f"Waiting for Customer at {env.now}")
        yield env.timeout(5)  # Time until next customer
        print(f"Serving Customer at {env.now}")

# Run the simulation
env = simpy.Environment()
env.process(vending_machine(env))
env.run(until=20)  # Run the simulation for 20 time units
```

Lab 2: Modeling Systems with CCS

Python doesn't directly support CCS, but **SimPy** can simulate concurrent processes.

```
import simpy
```

```
def producer(env, buffer):
    while True:
        yield env.timeout(1)  # Time to produce
        buffer.append(1)
        print(f"Produced an item at {env.now}, Buffer: {len(buffer)}")

def consumer(env, buffer):
    while True:
        if buffer:
            buffer.pop(0)
            print(f"Consumed an item at {env.now}, Buffer: {len(buffer)}")
        yield env.timeout(2)  # Time to consume

# Run the simulation
env = simpy.Environment()
buffer = []
env.process(producer(env, buffer))
env.process(consumer(env, buffer))
env.run(until=10)
```

Lab 3: Pi-Calculus and Dynamic Systems

For dynamic systems, use Python functions to model communication and dynamic behavior.

```
def client_server(env):
    print(f"Client sends request at {env.now}")
    yield env.timeout(1)
    print(f"Server processes request at {env.now}")
    yield env.timeout(2)
    print(f"Server sends response at {env.now}")

env = simpy.Environment()
env.process(client_server(env))
env.run(until=5)
```

Lab 4: Bisimulation Equivalence with Z3 Solver

Z3 Solver can be used to verify equivalence. Install Z3 using Colab:

```
!pip install z3-solver

from z3 import *

# Bisimulation Example: Compare two simple state transitions
x1, x2 = Ints('x1 x2')
solver = Solver()

# Define transitions
solver.add(x1 + 1 == x2 + 1)  # Both states increment similarly

if solver.check() == sat:
```

```
    print("The states are bisimilar.")
else:
    print("The states are not bisimilar.")
```

Lab 5: Fixed Points and Behavioral Properties

Use Python to calculate a fixed point for a simple example.

```
def fixed_point(func, x0, max_iter=10):
    for _ in range(max_iter):
        x1 = func(x0)
        if x1 == x0:
            return x1
        x0 = x1
    return None

# Example: Fixed point of f(x) = x^2 for initial value x0 = 1
f = lambda x: x**2
x0 = 1
print(f"Fixed point: {fixed_point(f, x0)}")
```

Lab 6: Modal Logic and Temporal Properties

PyNuSMV can be used to verify temporal properties:

```
# Install PyNuSMV
!pip install pynusmv

from pynusmv import *

# Define a simple FSM and verify properties
init(["example.smv"])  # Load an SMV file
if glob.prop_database.size > 0:
    print(glob.prop_database.get(0).evaluate())
else:
    print("No properties found.")
```

You'll need an .smv file containing the NuSMV model uploaded to Colab.

Lab 7: CTL Model Checking

Write CTL properties in NuSMV and verify them. Use the pynusmv package as shown in Lab 6.

Lab 8: Advanced Temporal Verification

Model a banking transaction system and verify fairness properties using SPIN or Python.

```
# Banking Transaction Example
def bank_transaction(env):
    print(f"Transaction Initiated at {env.now}")
    yield env.timeout(2)
    print(f"Transaction Approved at {env.now}")

env = simpy.Environment()
env.process(bank_transaction(env))
env.run(until=5)
```

Lab 9: Real-World Protocol Verification

Simulate the **TCP three-way handshake** using Python:

```
def tcp_handshake(env):
    print("SYN sent")
    yield env.timeout(1)
    print("SYN-ACK received")
    yield env.timeout(1)
    print("ACK sent")
    yield env.timeout(1)
    print("Connection Established")

env = simpy.Environment()
env.process(tcp_handshake(env))
env.run(until=5)
```

Lab 10: Comprehensive System Modeling and Verification

Model an elevator control system in Python:

```
def elevator(env, floors):
    current_floor = 0
    for target_floor in floors:
        print(f"Elevator moving from {current_floor} to {target_floor} at {env.now}")
        yield env.timeout(abs(target_floor - current_floor))  # Time to move
        current_floor = target_floor
        print(f"Elevator arrived at {current_floor} at {env.now}")

env = simpy.Environment()
floors = [0, 2, 5, 1]
env.process(elevator(env, floors))
env.run()
```

These programs are designed for hands-on implementation of the assignments. You can extend or adapt them as needed. Let me know if you'd like detailed examples for specific tools or tasks!