Select a drink:

- Water (C50p)
- Soft Drink
- Tea
- Fill up

Ensure there is water in stock before dispensing:

- If stock is sufficient, dispense water.
- If not, alert user.

Forbidden Scenario Example: Don't Give Two Drinks

- Ensure only one drink is dispensed per transaction.

Universal LSC Example:

- Ensure water is dispensed only when stock is sufficient.

Note: Existential LSCs may hint at test-cases for the acceptance test.

Universal LSCs (and negative/anti-scenarios) in general need exhaustive analysis.

Exhaustive analysis ensures that the software never ever exhibits the unwanted behaviour.

Course Map

- UML
- Model-Based/-Driven Software Engineering
Statement/branch coverage canonically extends to test suite
• negative (or unsuccessful execution of
  \( p \))
• successful (coverage of
  \( p \)).

In a description of

\( S_{\text{testintro}} \)

• test suites and
  \( S_{\text{cover}} \)

\( \{ \text{redefines name} \} \)

\( \text{shape: Square} \) (height = 7, width)

\( \text{System Model} \)

\( \text{Behaviour} \)

\( \text{Structure} \)

\( \text{Diagram} \)

\( \text{Class} \)

\( \text{Diagram} \)

\( \text{Idea} \)

\( \text{Diagram} \)
int f(int x, int y, int z)
{
    if (x > 100/7)
        z = z/5;
    else
        z = z/2;

    if (x > 50/5)
    {
        z = z/5;
    }
}

• Requirement:

\[ \text{true} \] \Rightarrow \text{true} \text{ (no abnormal termination)}, i.e. \ \dfrac{S_{\text{true}}}{S} = 1.

In %
% i 2 / % x, y, z i 1 / t i 1 / f s 1 s 2 i 2 / t i 2 / f c 1 c 2 s 3 s 4

State machine of C: State machine of D:

\[ E \rightarrow \text{GEN(F)} \]

\[ F \rightarrow \text{GEN(F)} \]

\[ \[ \text{params->x.GetLength()} == 7 \] \]

Excursion: Automatic Test Generation

Model-based Testing

• Given a set of test cases passing for the model, • and an implementation of the model (maybe hand-written). • Execute the test cases on the implementation (or the final system). This may need an appropriate interpretation. For example, if the test case says • send "C50" to the CoinValidator, • rather insert a 50 Cent coin into the vending machine. • If the vending machine does not behave according to the test, • then there's something wrong (wrong test conduction, wrong implementation, etc.). • If the vending machine does behave according to the test, • then we know that this scenario works — not more.


