One approach directly supports this approach. Rhapsody for example the behaviour in the software.

Check whether for 'drink_ready' more interaction is necessary.

- **Edge Coverage**
  - Analogously to check for 'drink_ready', more interaction is necessary.

- **Location Coverage**
  - If the LSC has designated a test driver & monitor role, we may need to glue logic into the environment (i.e., test unsuccessful) if and only if TBA state is reached.

- **Test case, test suite, test execution**
  - When testing, the environment instance (i.e., test unsuccessful) is automatically be generated from the model. For every edge in the given CFA model of the CoinValidator, the environment instance can originate from the model there is corresponding configuration of the model there is.

- **Input sequences can**
  - Check whether for 'drink_ready', more interaction is necessary.

- **Software quality assurance wrap-up**
  - Review of deterministic programs correctness.

- **Model-based testing**
  - Limits of software testing.

- **The verifier for Concurrent C**
  - Partial and total correctness.

- **Proof system PD**
  - Positive and negative outcomes.

- **Syntax, Semantics, Termination, Divergence**
  - Deterministic programs correctness.

- **Testing in the development process**
  - Testing in the development process.

- **Software verification**
  - Introduction and vocabulary.
When to Stop Testing?

- There need to be defined criteria for when to stop testing; project planning should consider these criteria (and previous experience).

  - Possible "testing completed" criteria:
    - all (previously) specified test cases have been executed with negative result, (Special case: All test cases resulting from a certain strategy, like maximal statement coverage have been executed.)
    - testing effort time sums up to \( x \) (hours, days, weeks),
    - testing effort sums up to \( y \) (any other useful unit),
    - \( n \) errors have been discovered,
    - no error has been discovered during the last \( z \) hours (days, weeks) of testing,

  - Values for \( x, y, n, z \) are fixed based on experience, estimation, budget, etc.

  - Of course: not all criteria are equally reasonable or compatible with each testing approach.

Another Criterion

- Another possible "testing completed" criterion:
  - The average cost per error discovery exceeds a defined threshold \( c \).

\[
\text{cost per discovered error} = \frac{\text{cost}}{\# \text{errors}}
\]

Value for \( c \) is again fixed based on experience, estimation, budget, etc.
• Test Gear: (may need to be developed in the project!)

  test driver — A software module used to invoke a module under test and, often, provide test inputs, control and monitor execution, and report test results.

  Synonym: test harness.

  IEEE 610.12 (1990)

  stub —
  (1) A skeletal or special-purpose implementation of a software module, used to develop or test a module that calls or is otherwise dependent on it.
  (2) A computer program statement substituting for the body of a software module that is or will be defined elsewhere.

  IEEE 610.12 (1990)

  • Roles:
    tester and developer should be different persons!

  Content

  • Testing:
    • Model-Based Testing
    • When to Stop Testing?
    • Testing in the Development Process
    • Formal Program Verification
      • Deterministic Programs
        • Syntax
        • Semantics
        • Termination, Divergence
        • Correctness of deterministic programs
          • partial correctness,
          • total correctness.

  • Proof System PD
    The Verifier for Concurrent C

  Concepts of Software Quality Assurance

  software quality assurance
  project management
  organisational software examination
  analytic examination by humans non-mech.
  inspection review manual proof
  comp. aided human exam.
  semi-mech.
  e.g. interactive prover examination with computer mechanical static checking analyse check against rules consistency checks quantitative examination dynamic checking (test)
  execute formal verification prove constructive software engineering constructive e.g. code generation

(Σ × A) ω all computation paths satisfying the specification

LSC: buy water
AC: true
AM: invariant
I: strict

User CoinValidator ChoicePanel Dispenser
C 50 WATER
¬ (C 50 ! ∨ E 1 ! ∨ pSOFT ! ∨ pTEA ! ∨ pFILLUP !)

water _ in _ stock dWATER
OK ¬ (dSoft ! ∨ dTEA !)
is a program \((S, \sigma)\) where \(\sigma = S\) with \(S\) as the first component of \(vii\).

\[
\begin{align*}
\text{do} &\rightarrow \langle \sigma \rangle \\
\text{while} &\rightarrow \langle \sigma \rangle
\end{align*}
\]
We say \( S \subseteq p \) and a computation is said to be successful (starting in \( S \)) if eventually \( S \) is reachable and the value \( \sigma \) of the program variable is such that \( \tau \sigma = \langle i, v \rangle \) for some \( i \). Let \( S \) be a deterministic program.

Example: Computing squares (of numbers \( x \))

(Recall: \( x = x \cdot x \); \( x = x + x \);
(\( x \equiv 0 \))

Another Example
\[ x = b \cdot \{ y \geq 0 \} \\\]
inside the specification!

For example, Rocky Mountain Brittle Power - Energy Strategy for National Security testing than just to software quality assurance.

We have shown:

1. \( p = b + y \)
2. \( \frac{a}{b} = b \)
3. \( x = b \cdot y \)
4. \( \frac{a}{b} = b + 1 \)
5. \( y = 0; \)
6. \( y = b + y \)
7. \( x = b \cdot y \)

We have shown:

1. \( p = b + y \)
2. \( \frac{a}{b} = b \)
3. \( x = b \cdot y \)
4. \( \frac{a}{b} = b + 1 \)
5. \( y = 0; \)
6. \( y = b + y \)
7. \( x = b \cdot y \)