This poster outlines a study that analyzes the effectiveness of enhancing pair-wise testing with branch coverage by incorporating execution path tracing using Control Flow Graphs (CFGs). The analysis will investigate the use of execution paths to frame the location of errors in source code and determine whether the combination facilitates regression testing.

**Motivation**

Software specifications refer to properties that the system must exhibit. Informal specifications are written in natural language. Formal specifications are mathematically based and allow the use of formal verification techniques such as model checking and runtime monitoring [1][2].

The Specification Pattern System (SPS) guides the practitioner through the formal specification process[3][4]. SPS defines a set of six patterns to represent the most commonly used formal properties and a set of five scopes of system execution where the pattern of interest must hold. Eight composite propositions (CP) expand the expressiveness of patterns and scopes to include the specification of sequential and concurrent behaviors [7].

The Prospec algorithm defines a set of templates that can be used to ease the generation of LTL specifications for all pattern, scope, and CP combinations [5][6]. The templates generate properties by selecting one out of six possible pattern types, one out of five possible scopes, and each pattern and scope can have between zero and two CPs associated with it, where each CP can be one out of eight possible types, as shown in Table 1.

**Pair-Wise Testing**

Software systems of any significance are typically too large to be tested for all possible inputs. For these systems, pair-wise testing is useful because it focuses on testing all pairs of test case inputs instead of all combinations of inputs [9].

Consider a hypothetical system with three variables a, b, and c, in which each variable can assume one of four values. The task of verifying 32120 properties generated by the Prospec algorithm is daunting. While a white-box testing strategy is optimal to locate defects, it is difficult to identify the test cases that provide appropriate coverage [10][11]. The pair-wise testing strategy addresses test-case selection [12]; however, it is unclear whether it provides adequate branch coverage [13][14]. This work is expected to instill confidence that the pair-wise strategy generates a test suite that provides sufficient coverage to verify the Prospec implementation.

**Summary and Future Work**

This poster outlines a study that analyzes the effectiveness of enhancing pair-wise testing with branch coverage by incorporating execution path tracing using CFGs. This work will evaluate the approach on a set of six complex LTL-Generation implementations created independently of each other. The analysis will investigate the use of execution paths to frame the location of errors in source code and determine whether the combination facilitates regression testing. By creating a link between tests and source code when a modification to a section of the code is made, the affected tests can easily be discarded and re-executed to facilitate regression testing.

Future work includes augmenting the pair-wise testing strategy with a process that includes execution-path traces to locate defects. Another important effort will be to use the same approach to evaluate pair-wise testing when applied to different domains.

**References**


