Testing Framework for P

# Components in p

There are three main components in the P framework.

1. P Compiler.
2. Executable Code.
3. Verification of P program.

It is important to test each of these components separately and rigorously.

## P Compiler

The P compiler consist of following important components that needs to be tested properly:

1. Syntax Checking
2. Static Checking
3. Type Checking and Type Inference
4. Generated C code
5. Generated Zing code

## Executable Code

Testing the runtime and its interaction with the generated code consists of writing test cases for the following.

1. Semantics of all the operations in P.
2. Dynamic type checking
3. Tests all the 3 platforms that are supported right now:
   1. User
   2. Driver
   3. Distributed

## P + Zing

Testing of P+ZING consists of writing test cases for the following:

1. Semantics of all the operations in P.
2. Dynamic type checking in Zing
3. Interaction with DelayBounding and other Zing optimizations
4. Interaction with Liveness algorithms (MAP and MACE).

## P Features to test

Test all the P constructs listed in P.4ml:

1. State Machine Level Declarations
   1. Event Declaration
      1. Cardinality of the event:
         1. Assume Max Instances of an event: [syntax](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/MaxInstances_2/MaxInstances_2.p);

number of instances greater than assumed: ["assume 0" case](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/MaxInstances_3/MaxInstances_3.p); “assume 1” case;

number of instances are not greater than assumed: [test1](../../Tst/RegressionTests/Integration/Correct/BangaloreToRedmond_Liveness/BangaloreToRedmond_Liveness.p) (“assume 1”), [test2](../../Tst/RegressionTests/Integration/Correct/Elevator/Elevator.p) (“assume 1”)

* + - 1. Assert Max Instances of an event: [syntax](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/MaxInstances_2/MaxInstances_2.p);

number of instances are greater than asserted: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/DynamicError/MaxInstances_1/MaxInstances_1.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/TokenRing/TokenRing.p) (“assert 1”), …

number of instances are not greater than asserted: [test1](../../Tst/RegressionTests/Integration/DynamicError/BangaloreToRedmond/BangaloreToRedmond.p) (“assert 2”), [test2](../../Tst/RegressionTests/Integration/Correct/BangaloreToRedmond_Liveness/BangaloreToRedmond_Liveness.p) (“assert 2”),

[test3](../../Tst/RegressionTests/Integration/Correct/PingPong/PingPong.p) (“assert 1”), [test4](../../Tst/RegressionTests/Integration/Correct/PingPongDingDong/PingPongDingDong.p) (“assert 1”), [test5](../../Tst/RegressionTests/Integration/Correct/Elevator/Elevator.p) (“assert 1”)

* + 1. Payload Type: [invalid payload type (event expects no payload)](../../Tst/RegressionTests/Feature4DataTypes/StaticError/function_Typos/function_Typos.p);

**incompatible payload type**

* 1. Machine Declaration
     1. Queue Size Constraint: [test1](../../Tst/RegressionTests/Integration/DynamicError/TokenRing/TokenRing.p)
     2. Different types of Machines: Real ([test](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p)1), Model ([test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/TokenRing/TokenRing.p)), Monitor ([test1](../../Tst/RegressionTests/Integration/Correct/Elevator/Elevator.p))).
     3. Start state [defined](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p) / [not defined](../../Tst/RegressionTests/Combined/StaticError/Duplicates/Duplicates.p).
     4. Monitors:
        1. Hot/cold states: [test1](../../Tst/RegressionTests/Integration/Correct/PingPongMonitor/PingPongMonitor.p)
        2. Not allowed constructs: [new, send, push, pop, model functions, defer, default trans](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/monitors/monitors.p); **monitors cannot refer to “this”**
     5. No main machine declared: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/purity/purity.p), [test2](../../Tst/RegressionTests/Feature2Stmts/StaticError/lvalues/lvalues.p), [test3](../../Tst/RegressionTests/Feature3Exprs/StaticError/fields/fields.p)
     6. Multiple main machines declared: **test1**
     7. No start state in machine: [test](../../Tst/RegressionTests/Combined/StaticError/Duplicates/Duplicates.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/purity/purity.p), [test3](../../Tst/RegressionTests/Feature2Stmts/StaticError/lvalues/lvalues.p), [test4](../../Tst/RegressionTests/Feature3Exprs/StaticError/fields/fields.p), [test5](../../Tst/RegressionTests/Feature4DataTypes/StaticError/function_Typos/function_Typos.p)
  2. Variable Declaration
  3. Function Declaration
     1. Model functions: can only be declared in real machines ([test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/entryExit_1/entryExit_1.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/monitors/monitors.p))
     2. [Wrong type/number of function parameters or return value](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p)
     3. [Return value has incorrect type (undeclared)](../../Tst/RegressionTests/Feature4DataTypes/StaticError/function_Typos/function_Typos.p)
     4. Data impure functions: [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/lvalues/lvalues.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/purity/purity.p)
     5. Types of formal and actual parameters: [any/int](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/functionAny/functionAny.p),
  4. Anonymous Function Declaration
     1. Return values of anonymous functions: [cannot return value](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p)
  5. State Declaration
     1. Function as entry/exit action: [with arguments](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/entryExit_2/entryExit_2.p)
     2. Undefined function: [for exit](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/entryExit_2/entryExit_2.p), **for entry**
     3. Function cannot take arguments:[for entry**,** for exit](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/entryExit_2/entryExit_2.p)
     4. [Hot/cold states](../../Tst/RegressionTests/Integration/Correct/Elevator/Elevator.p)
     5. Groups of states[: no error](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/Groups/Groups.p); error
     6. Actions:
        1. [multiple actions over the same event](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/monitors/monitors.p)
        2. [Basic semantics of actions](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p" \o "Actions_1)
        3. Exit actions: [exit actions are executed before "goto" transition,](../../Tst/RegressionTests/Feature1SMLevelDecls/DynamicError/AlonBug_fails/AlonBug_fails.p)

exit actions are not executed: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/AlonBug/AlonBug.p), …

* 1. Transition Declaration
     1. Different types of events {String, default, halt}
     2. Different types of transitions: normal or push:
        1. [Pairwise push/pop in a loop](../../Tst/RegressionTests/Integration/DynamicError/PushStatement_1/PushStatement_1.p" \o "PushStatement_1)
        2. Goto transition: [basic syntax](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), [undefined event](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p),

Goto with action: [function cannot take arguments](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p),

[transition function not defined](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p), [assigment as action](../../Tst/RegressionTests/Integration/DynamicError/Push_Pop_1/Push_Pop_1.p),

* + - 1. Push transition: [overriding of action handlers by a pushed state](../../Tst/RegressionTests/Integration/DynamicError/Actions_4/Actions_4.p);

[inheritance of actions and not deferred-by-default](../../Tst/RegressionTests/Integration/DynamicError/Actions_5/Actions_5.p); [inheritance of actions but not of transitions](../../Tst/RegressionTests/Integration/DynamicError/Actions_5/Actions_5.p); [payload with push transitions](../../Tst/RegressionTests/Integration/DynamicError/Actions_6/Actions_6.p);

**transition function not defined**

* + 1. Functions on transitions Anon or Named.
    2. Transition to undefined state: goto transition ([test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/AnonFuns/anonFunction.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p)); **push transition;**
    3. **Transition on an undefined event: goto transition, push transition**
    4. Function as action in transition cannot take arguments: goto transition ([test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p)); **push transition**
  1. Do Declaration
     1. Actions on different types of events {String, [default](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/AnonFuns/anonFunction.p), halt}
     2. Different types of actions defer, ignore or anon function or named function.
        1. Do decl: action on named function: [test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p),
        2. Explicit “defer”: [test1](../../Tst/RegressionTests/Integration/Correct/BangaloreToRedmond_Liveness/BangaloreToRedmond_Liveness.p)
     3. Do declaration: [action on undefined event](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p), [transition function not defined](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p),

[function cannot take arguments](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p)

* 1. Annotations in P language
     1. Zing Annotation for seal/unseal
     2. Zing Annotation for state coverage information.

1. P Statements
   1. Push and pop statements:

2.1.1. Basic syntax/semantics: [test1,](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p)

2.1.2. Deferred-by-default semantics: action (“do”) vs transition (“goto”): [test passes,](../../Tst/RegressionTests/Integration/Correct/Actions_2/Actions_2.p) test fails ([test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_2_fails/Actions_2_fails.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/Actions_3/Actions_3.p)); [push with pop](../../Tst/RegressionTests/Integration/DynamicError/Actions_3/Actions_3.p)

2.1.3. [Overriding of action handlers by a pushed state](../../Tst/RegressionTests/Integration/DynamicError/Actions_4/Actions_4.p)

2.1.4. [push and pop statements with if-while control flow](../../Tst/RegressionTests/Integration/DynamicError/PushStatement_1/PushStatement_1.p)

* 1. [Dynamic creation of machines using New](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p" \o "Actions_1)
  2. Raise Statement: [test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), [parameter should be an event](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p),
  3. Send Statement[: payload type checking,](../../Tst/RegressionTests/Feature2Stmts/StaticError/sends/sends.p) [2nd argument should be an event](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p),
  4. [Monitor Invocation](../../Tst/RegressionTests/Integration/Correct/Elevator/Elevator.p)
  5. Function Call
  6. Skip and Pop Statement
  7. Assertions: [single assert](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), two asserts in a row: [pass/fail](../../Tst/RegressionTests/Integration/DynamicError/Actions_4/Actions_4.p), [with complimentary conditions](../../Tst/RegressionTests/Integration/DynamicError/PushStatement_1/PushStatement_1.p)
  8. Binary Statements (remove, insert and assign)
     1. On all data types: [bool](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), [int](../../Tst/RegressionTests/Integration/DynamicError/PushStatement_1/PushStatement_1.p), …
     2. Assign:
        1. invalid LHS ([test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p), [test2](../../Tst/RegressionTests/Feature2Stmts/StaticError/lvalues/lvalues.p)),
        2. [impure functions in LHS](../../Tst/RegressionTests/Feature2Stmts/StaticError/lvalues/lvalues.p), …
  9. Return statement
  10. While Statement
  11. ITE Statement: [type of the condition](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p),

1. P Expressions
   1. New expression.
   2. Function call as an expression
   3. Different Primitive expressions
      1. This, [trigger](../../Tst/RegressionTests/Integration/DynamicError/BangaloreToRedmond/BangaloreToRedmond.p)[, payload](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), nondet, fair nondet, null and halt
      2. Unary Expression
         1. Not, neg, keys, values, sizeof
      3. Binary Expression
         1. [Add](../../Tst/RegressionTests/Integration/DynamicError/PushStatement_1/PushStatement_1.p), sub, mul, division, and, or, eq([test1](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p)), neq, lt, le, gt, ge, idx and in
      4. Field Access
         1. Tuples and NamedTuples
         2. [Bad field names](../../Tst/RegressionTests/Feature3Exprs/StaticError/fields/fields.p)
      5. Default Operation
         1. For all data types
      6. Cast operator (as[): payload as machine](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), …
      7. NONDE expressions: [nondet choice in real/model/monitor machines and functions](file:///D:\PLanguage\PLang1001\plang\Tst\RegressionTests\Feature2Stmts\StaticError\entryExit_1\entryExit_1.p);

**same test for FAIRNONDET case (modify entryExit\_1)**

1. Data types in P
   1. Creation of complex data types
      1. Tuples: [duplicate names in seq decl, in func decl, in payload](../../Tst/RegressionTests/Feature4DataTypes/StaticError/function_Typos/function_Typos.p)
   2. Subtype relation among data types: [assignments](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p), …
   3. Passing variables as payloads: [“ghost machine” type](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), …
   4. **Types of operands in expressions**
      1. **Unary expressions: for all all operators: NOT, NEG, KEYS, VALUES, SIZEOF and all applicable types: BOOL, INT, SeqType(dom), SeqType(cod), INT; (lines 246-265 in P.4ml)**

**Error messages to test: “Operator expected a [boolean/integer/map(4 cases)] value”**

* + 1. **Binary expressions:** 
       1. **Cast expressions: “Cast can never succeed”**
       2. **“New” expressions: “Monitors cannot be created with "new""”, “Undefined machine type”**
       3. **“Field” operator:** “Bad field name" ([test1](../../Tst/RegressionTests/Feature3Exprs/StaticError/fields/fields.p), …); **“Operator expected a (named) tuple value"**
       4. **{ ADD, SUB, MUL, INTDIV }**

## INTEGRATION TESTS: Interaction between featURES

1. Precedence Relations
   1. Local variables > Variables > Events
   2. Transitions > Actions > [{ Ignore](../../Tst/RegressionTests/Integration/Correct/Elevator/Elevator.p), [defer](../../Tst/RegressionTests/Integration/Correct/Elevator/Elevator.p) }
2. No raise, or pop or call in functions.
3. Calls should always terminate with a pop.

## Combined Tests

1. Duplicates: [event defintions, machine declarations, variable declarations, transitions over the same event, actions over the same event](../../Tst/RegressionTests/Combined/StaticError/Duplicates/Duplicates.p" \o "Duplicates), **multiple states with the same name**, **multiple functions with the same name**.
2. Re-definition of variables: [int and event](../../Tst/RegressionTests/Combined/Correct/variableType/variableType.p) (variable hides the name of event); **formal par hides local varialble**
3. **Missing declarations**: undefined symbol (must be declared as formal par, variable or event)

## Test cases architecture

The folder structure for the test cases would look like following

* Root
  + State Machine Level Declarations
  + Statements
  + Expressions
  + Data types
  + Integration Tests
  + Combined Tests

Each subfolder listed above will have the following subfolders (“type of error” level):

* Static Error: Static analysis reports an error (as listed in “P COMPILER” section)
* Dynamic Error: Zinger reports an error
* Correct: Zinger does not report an error

Folders at the “type of error” level might change, according to the requirements in the “Components of P” section of this document. Also, we may want to add more granularity to the “Correct” folder, for example, to separate tests for runtime from tests for (P + Zing).