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 (Runtime)

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

A subset of runtime tests can be the same tests written for P Compiler and Zing. There’s an issue with the tests for which the runtime program is non-terminating (for example, Zing “passes”). Possible solutions:

* Use a timeout for runtime (for example, for liveness tests)
* Modify the original test by adding an assert, thus changing Zing result from “pass” to “fail”

For the first set of runtime tests, we are assuming single-threaded model of execution: one thread per machine (even for multiple instances of the same machine).

Coverage of runtime tests:  
1. Coverage for the tests identical to the tests for statics checks and Zing, when Zing results in “fails”:

(Tests in yellow have to be converted such that Zing result changes from “pass” to “fail”)

|  |  |
| --- | --- |
| SEM\_OneMachine\_1\SendRaiseInEntry.p | Integration: P semantics test: one machine, "send" to itself and "raise" in entry actions |
| SEM\_OneMachine\_2\RaiseSendInEntry.p | Integration: P semantics test: one machine, 'raise" and "send" to itself in entry actions |
| SEM\_OneMachine\_3\SendInEntry.p | Integration: P semantics test: one machine, "send" to itself in entry actions |
| SEM\_OneMachine\_4\SendInExitNotExecuted.p | Integration: P semantics test: one machine,"send" to itself in exit not executed |
| SEM\_OneMachine\_5\SendInExitUnhandledEvent.p | Integration: P semantics test: one machine, "send" to itself in exit function |
| SEM\_OneMachine\_6\SendInExitHandledEvent.p | Integration: P semantics test: one machine, "send" to itself in exit function |
| SEM\_OneMachine\_7\SendInExitUnhandledHalt.p | Integration: P semantics test: one machine, "send" to itself in exit function |
| SEM\_OneMachine\_8\GotoToItself.p | Integration: P semantics test: one machine, "goto" to the same state; "send" in entry and exit |
| SEM\_OneMachine\_9\PushItself.p | Integration: P semantics test: one machine, "push" and "goto" to the same state; "send" in entry and exit |
| SEM\_OneMachine\_10\Push.p | Integration: P semantics test: one machine, "push", "send" in entry and exit |
| SEM\_OneMachine\_11\PushImplicitPopWithSend.p | Integration: P semantics test: one machine, "push" with implicit "pop" when the unhandled event was sent |
| SEM\_OneMachine\_12\PushExplicitPop.2 | Integration: P semantics test: one machine, "push" with explicit "pop" |
| SEM\_OneMachine\_13\PushTransInheritance.p | Integration: P semantics test: one machine, "push" transition, action inherited by the pushed state |
| SEM\_OneMachine\_14\GotoTransInheritance.p | Integration: P semantics test: one machine, "goto" transition, action is not inherited by the destination state |
| SEM\_OneMachine\_15\ImplicitPopExit.p | Integration: P semantics test: one machine, exit actions executed upon implicit "pop" |
| SEM\_OneMachine\_16\ExplicitPopExit.p | Integration: P semantics test: one machine, exit actions executed upon explicit "pop" |
| SEM\_OneMachine\_17\PushImplicitPopWithRaise.p | Integration: P semantics test: one machine, "push" with implicit "pop" when the unhandled event was raised |
| SEM\_OneMachine\_18\UnhandledEventDeferred.p | Integration: P semantics test: one machine, deferral of an unhandled event |
| SEM\_OneMachine\_19\UnhandledEventIgnored.p | Integration: P semantics test: one machine, "ignore" of an unhandled event |
| SEM\_OneMachine\_20\BugReproIdenticalTerms.p | Integration: P semantics test: one machine, exit function performed while explicitly popping the state |
| SEM\_OneMachine\_21\BugReproIdenticalTerms\_workaround.p | Integration: P semantics test: one machine, exit function performed while explicitly popping the state |
| SEM\_OneMachine\_22\AlonBug\_1.p | Integration: P semantics test: one machine, deadlocked state is not the only state on the stack |
| SEM\_TwoMachines\_1\EventSentAfterSentHalt.p | Integration: P semantics test: event sent after machine is halted |
| SEM\_TwoMachines\_2\EventSentAfterSentHalt\_v.p | Integration: P semantics test: event sent after machine is halted: validating test |

AlonBug\_fails: 1.6.5.3. Exit actions: exit actions not executed

MaxInstances\_1: 1.10. Assert Max Instances of an event: number of instances greater than asserted

Actions\_1: Feature Integration: 1.6.5.2. basic semantics of actions

Actions\_2\_fails, Actions\_3: Feature Integration: 2.1.2. Deferred-by-default semantics of push statement

Actions\_4: Feature Integration: 1.7.2.3. Push transition: overriding of action handlers by a pushed state

Actions\_5: Feature Integration: 1.7.2.3. Inheritance of actions and not deferred-by-default

Actions\_6: Feature Integration: payload with push transition

BangaloreToRedmond: Feature Integration: protocol sample

Push\_Pop\_1, Push\_Pop\_2, PushStatement\_1: Feature Integration

Liveness\_2\_bugFound: Liveness: simplest sample demonstrating liveness error found

BoundedChoice\_1: Testing Bounded Choice Operator in Zing

## 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**
   1. Bounded choice operator in Zing:
      1. “-bc:2”: [test1](../../Tst/RegressionTests/Zinger/Correct/BoundedChoice_1/BoundedChoice_1.p)
      2. Default “-bc”: [test2](../../Tst/RegressionTests/Zinger/DynamicError/BoundedChoice_1/BoundedChoice_1.p)
   2. DFS Stack Bound:
      1. "-maxdfsstack:10", "-ibound:100": [test1](../../Tst/RegressionTests/Zinger/DynamicError/DFSStackBound/DFSStackBound.p)
4. Interaction with Liveness algorithms (MAP and MACE): refer to section [#Liveness tests](#_Liveness_tests)

## P Features to test (mostly syntax checks)

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”), [test3](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_9/PushItself.p)

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: see section 4.3 below.
  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, null trans, “this”: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/monitors/monitors.p)
     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: same names, different names: [test1](../../Tst/RegressionTests/Combined/StaticError/Duplicates/Duplicates.p)
     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),
     6. Functions that mutate parameters and have a scheduling event between parameter mutation and usage: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/DynamicError/BugRepro/PingPongBugRepro.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/BugRepro1/FunctionParMutated.p)
  4. Anonymous Function Declaration
     1. Return values of anonymous functions: [cannot return value](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p), [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/functionAnyAnon/functionAnyAnon.p)
  5. State Declaration
     1. Function as entry/exit action: [with arguments](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/entryExit_2/entryExit_2.p)
     2. Undefined function (in entry actions, in exit functions): [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/entryExit_2/entryExit_2.p)
     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: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/monitors/monitors.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/DeferIgnoreSameEvent/DeferIgnore1.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:

[test1](../../Tst/RegressionTests/Feature1SMLevelDecls/DynamicError/AlonBug_fails/AlonBug_fails.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_6/SendInExitHandledEvent.p)

exit actions are executed upon explicit/implicit pop: <#ExitSemanticsTests>

exit actions are not executed: [test1](file:///D:\PLanguage\PLang1001\plang\Tst\RegressionTests\Feature1SMLevelDecls\Correct\AlonBug\AlonBug.p); [test2;](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_4/SendInExitNotExecuted.p)

* 1. Transition Declaration
     1. Different types of events {String, null, halt} : [test1](../../Tst/RegressionTests/Integration/Correct/SEM_TwoMachines_3/EventSentAfterSentHaltHandled.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/Push_Pop_1/Push_Pop_1.p)
     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)

[Tests in the “Integration Tests” section](#Integration_GotoTrans)

* + - 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);

[Tests in the “Integration tests” section](#Integration_PushTrans)

transition function not defined: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p)

* + 1. Functions on transitions Anon or Named: <#FunctionStmts>
    2. Transition to undefined state: goto transition, push transtion: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/AnonFuns/anonFunction.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p)
    3. Transition on an undefined event: goto transition, push transition: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature1SMLevelDecls\StaticError\TransDelc_DoDecl\TransDecl_DoDecl.p)
    4. Function as action in transition cannot take arguments: goto transition, do declaration: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p)
    5. “null” event:

cannot be sent/raised/declared/deferred/ignored: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/RaisedNullEvent/RaisedNullEvent.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/SentNullEvent/SentNullEvent.p), [test3](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/NullEventDecl/NullEventDecl.p), [test4](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/DeferredNullEvent/DeferredNullEvent.p), [test5](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/IgnoredNullEvent/IgnoredNullEvent.p)

* + 1. “halt” event: <#HaltTwoMachines>
  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);[test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/EventDeferredHandledSameState/EventDeferredHandledSameState.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:
      1. Basic syntax/semantics: [test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p)

“Undeclared state name”: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.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)

Errors:

**“function contains bad labeling of push statements"**

* 1. [Dynamic creation of machines using New;](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p" \o "Actions_1)

**[“Undeclared machine” error: how to enable this message? See monitors.p.](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p" \o "Actions_1)**

* 1. Raise Statement:
     1. Basic tests: [test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p);

[parameter should be an event](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p)

[“raise” in entry handled in the same state](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_1/SendRaiseInEntry.p" \o "SendRaiseInEntry)

* + 1. raise with non-constant event expression (with/without payload): [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/eventExprSendRaise/EventExprSendRaise.p), [test2](../../Tst/RegressionTests/Zinger/DynamicError/RaisedNull/RaisedNullDetected.p); tests in “Integration Tests” section, 3.1.1.4.
    2. raise with constant event expression: raise has NIL payload, raise has non-NIL payload;

“[argument 1 of "raise" expects an event value](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p)"

“[invalid payload type in raise (cannot send null value)](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/monitors/monitors.p)” (for raise with constant event expression, NIL payload)

“invalid payload type in raise" (for raise with constant event expression, NIL and non-NIL payload):[test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/monitors/monitors.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/StaticError/payloads/payloads.p)

* 1. Send Statement:
     1. Basic semantics tests:

[“send” in entry handled in the same state](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_1/SendRaiseInEntry.p" \o "SendRaiseInEntry)

* + 1. Send with constant event expression: NIL and non-NIL payload cases: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature4DataTypes\StaticError\payloads\payloads.p), [test2](../../Tst/RegressionTests/Zinger/DynamicError/SentNull/SentNullDetected.p)
    2. Send with non-constant event expression (with/without payload): [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\StaticError\eventExprSendRaise\EventExprSendRaise.p), [test2](../../Tst/RegressionTests/Zinger/DynamicError/SentNull/SentNullDetected.p); tests in “Integration Tests” section, 3.1.1.4.
    3. Error rules for send:  
       “argument 1 of ”send” expects a machine value”: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\StaticError\sends\sends.p)

“[argument 2 of "send" expects an event value](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p)"

“invalid payload type in send (cannot send null value)": [test1](../../Tst/RegressionTests/Feature4DataTypes/StaticError/payloads/payloads.p)

“invalid payload type in send": [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/sends/sends.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/StaticError/function_Typos/function_Typos.p)

* 1. Monitor Invocation:
     1. Rules for monitor with non-constant event expression: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\StaticError\sends\sends.p), [test2](../../Tst/RegressionTests/Zinger/DynamicError/MonitorNull/MonitorNullDetected1.p); tests in “Integration Tests” section, 3.1.1.10.
     2. Rules for monitor with constant event expression: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\StaticError\sends\sends.p)
     3. Error rules for monitor:

“Undeclared monitor”: [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/sends/sends.p)

“argument 2 of "monitor" expects an event value": [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\StaticError\sends\sends.p)

“invalid payload type in monitor (cannot send null value)": [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/sends/sends.p)

“invalid payload type in monitor”: [test1](file:///D:\PLanguage\PLang1001\plang\Tst\RegressionTests\Feature2Stmts\StaticError\sends\sends.p)

* 1. Function Statement (FunStmt):

rules about conformance of function stmts to function decls: 2 cases

errors (make sure the tests are for FunStmt, rather then for FunApp, as in 4.4.3.12):

“function not defined" (not the case of DoDecl, as in 1.8.3. above): [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInEntry/ControlImpureInEntry.p)

“function requires arguments": [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/lvalues/lvalues.p)

“function arguments have incorrect types": [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\StaticError\lvalues\lvalues.p)

* 1. Assertions: [single assert](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), two asserts in a ro1w: [pass/fail](../../Tst/RegressionTests/Integration/DynamicError/Actions_4/Actions_4.p), [with complimentary conditions](../../Tst/RegressionTests/Integration/DynamicError/PushStatement_1/PushStatement_1.p);

“"assert" expects a boolean value”: [test1](../../Tst/RegressionTests/Feature4DataTypes/StaticError/nonAtomicDataTypes/nonAtomicDataTypes.p)

* 1. Mutating Statements (ASSIGN, REMOVE, INSERT)
     1. ASSIGN:

rule about subtype relation between LHS and RHS

error: “invalid assignment. right hand side is not a subtype of left hand side":

cases of LHS/RHS types: [event/NIL](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p), [NIL/ERROR](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/monitors/monitors.p), [SeqType(ANY)/INT](../../Tst/RegressionTests/Feature2Stmts/StaticError/sends/sends.p), [INT/ERROR](../../Tst/RegressionTests/Feature3Exprs/StaticError/fields/fields.p), [NmdTupType/NmdTupType, wrong names](../../Tst/RegressionTests/Feature2Stmts/StaticError/nmdType/nmdType.p); **etc.**

[impure functions in LHS](../../Tst/RegressionTests/Feature2Stmts/StaticError/lvalues/lvalues.p)

errors:

“this function must be pure. Data impure functions can only be called like "foo(...);" or "x = foo(...);"

“new M(...) can only be called like "new M(...);" or "x = new M(...);" : [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/purity/purity.p) (only BinApp is tested; **TODO: test all other contexts**)

* + 1. REMOVE:

rule about types of arguments: seq. and map cases: tests from section 4.1 below.

errors: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature4DataTypes\StaticError\nonAtomicDataTypes\nonAtomicDataTypes.p)

“remove must be applied to a sequence or map"

“index must be an integer"

“Index may not be in the domain of the map"

* + 1. INSERT: refer to tests from section 4.1 below.

rule about types of arguments: 2 cases

errors: [test1](../../Tst/RegressionTests/Feature4DataTypes/StaticError/nonAtomicDataTypes/nonAtomicDataTypes.p), [test2](../../Tst/RegressionTests/Feature3Exprs/StaticError/ExprsOperators/ExprOperators.p)

“insert must be applied to a sequence or a map"

“for insert right syntax is seq += (index value) or map += (key value)"

“key must be an integer”

“value must be a subtype of sequence type"

“key not in the domain of the map"

“value not in the codomain of the map"

* 1. Return statement:

“[function must return a value](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p)”;

“return value has incorrect type”: declared/used cases: [int/bool](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p); [NIL/bool](../../Tst/RegressionTests/Feature4DataTypes/StaticError/function_Typos/function_Typos.p); **etc.**

[**“**anonymous function cannot return a value”](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p)

* 1. While Statement: [test1](../../Tst/RegressionTests/Feature4DataTypes/Correct/nonAtomicDataTypes/nonAtomicDataTypes.p)

Error:

“"while (...)" expects a boolean value";

* 1. ITE Statement: “"if (...)" expects a boolean value": [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/entryExit_1/entryExit_1.p); [test2](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p);

1. P Expressions (excluding type-checking – see #4 below).
   1. New expression: [test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p) (and many others)
   2. Function Application (FunApp): see 4.4.3.12 below.
   3. Different Primitive expressions: refer to 3.4. below
      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: initialization of variable
         1. For all data types: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature4DataTypes\StaticError\nonAtomicDataTypes\nonAtomicDataTypes.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/Correct/nonAtomicDataTypesAllAsserts/nonAtomicDataTypesAllAsserts.p)
      6. Cast operator (as[): payload as machine](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p), …
      7. NONDET expressions: [nondet choice in real/model/monitor machines and functions](file:///D:\PLanguage\PLang1001\plang\Tst\RegressionTests\Feature2Stmts\StaticError\entryExit_1\entryExit_1.p);
      8. FAIRNODET expressions (liveness-related): [test1](../../Tst/RegressionTests/Liveness/Correct/Liveness_FAIRNONDET/Liveness_FAIRNONDET.p), [test2](../../Tst/RegressionTests/Liveness/Correct/Liveness_FAIRNONDET2/Liveness_FAIRNONDET.p), [test3](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_NONDET/Liveness_NONDET.p), [test4](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_NONDET2/Liveness_NONDET.p)
   4. Types of operands in expressions have each operator tested at least once:  
      3.4.1. Nullary expressions: Integer + Boolean + { THIS, TRIGGER, PAYLOAD, NONDET, FAIRNONDET, NULL, HALT }: refer to 3.3 above and other tests.
      1. Unary expressions: for all all operators and all applicable types: BOOL, INT, EVENT, MACHINE, ANY:

NOT, NEG: [test1](../../Tst/RegressionTests/Feature3Exprs/StaticError/ExprsOperators/ExprOperators.p), [test3](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\Correct\ExprOperatorsAsserts\ExprOperatorsAsserts.p)

Operators’ types:

BOOL, INT (error), EVENT (error), ANY (error)

KEYS, VALUES,SIZEOF**:** [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature4DataTypes\StaticError\nonAtomicDataTypes\nonAtomicDataTypes.p)

Error messages to test:

“Operator expected a [boolean/integer/map(keys,values)/map or sequence(sizeof)] value”: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature4DataTypes\StaticError\nonAtomicDataTypes\nonAtomicDataTypes.p)

* + 1. Binary expressions:
       1. Cast in expressions: [test1](../../Tst/RegressionTests/Feature4DataTypes/StaticError/CastInExprs/CastInExprs.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/Correct/CastInExprsAsserts/CastInExprsAsserts.p), [test3](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/CastInExprs1/CastInExprsDynError.p), [test4](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/CastInExprs2/CastInExprsDynError.p), [test5](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/CastInExprs3/CastInExprsDynError.p), [test6](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/CastInExprs4/CastInExprsDynError.p), [test7](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/CastInExprs5/CastInExprsdynError.p), [test8](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/CastInExprs6/CastInExprsDynError.p)

error:

“Cast can never succeed”

* + - 1. “New” expressions:

errors:[test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature1SMLevelDecls\StaticError\monitors\monitors.p)

“Monitors cannot be created with "new""”

“Undefined machine type”

* + - 1. “Field” operator:

errors:

“Bad field name" ([test1](../../Tst/RegressionTests/Feature3Exprs/StaticError/fields/fields.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/StaticError/nonAtomicDataTypes/nonAtomicDataTypes.p))

“Operator expected a (named) tuple value": [test1](../../Tst/RegressionTests/Feature4DataTypes/StaticError/payloadEntry/payloadEntry.p) and other tests in the same subfolder

* + - 1. { ADD, SUB, MUL, INTDIV }: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\StaticError\ExprsOperators\ExprOperators.p), [test2](../../Tst/RegressionTests/Feature3Exprs/StaticError/fields/fields.p), [test3](../../Tst/RegressionTests/Feature3Exprs/Correct/ExprOperatorsAsserts/ExprOperatorsAsserts.p)

Operands’ types tested:

ADD: INT, INT

ADD: INT, BOOL (error)

ADD: INT, ANY (error, even if var of type ANY is assigned an int value)

SUB: INT, INT

SUB: BOOL, INT (error)

MUL: INT, INT

MUL: INT, BOOL (error)

MUL: INT, ANY (error, even if var of type ANY is assigned an int value)

INTDIV: INT, INT

INTDIV: INT, BOOL (error)

errors:

[“Operator expected first argument to be int”](../../Tst/RegressionTests/Feature3Exprs/StaticError/fields/fields.p)

“Operator expected second argument to be int”

* + - 1. { LT, LE, GT, GE }:[test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\StaticError\ExprsOperators\ExprOperators.p) , [test3](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\Correct\ExprOperatorsAsserts\ExprOperatorsAsserts.p)

Operands’ types tested:

INT, INT

INT, BOOL (error)

ANY, INT (error)

errors:

“Operator expected first argument to be int”

“Operator expected second argument to be int”

* + - 1. { AND, OR }: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\StaticError\ExprsOperators\ExprOperators.p), [test3](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\Correct\ExprOperatorsAsserts\ExprOperatorsAsserts.p)

Operands’ types tested:

BOOL, BOOL

INT, BOOL (error)

BOOL, EVENT (error)

ANY, BOOL (error, even if var of type ANY is assigned a bool value)

errors:

“Operator expected first argument to be bool”

“Operator expected second argument to be bool”

* + - 1. { EQ, NEQ }: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\StaticError\ExprsOperators\ExprOperators.p), [test2](file:///D:\PLanguage\plang\Tst\RegressionTests\Integration\StaticError\TokenRing_Typos\TokenRing_Typos.p), [test3](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\Correct\ExprOperatorsAsserts\ExprOperatorsAsserts.p)

Operands’ types tested:

BOOL, BOOL

INT, INT

BOOL, ANY (OK??)

INT, ANY (OK??)

EVENT, ANY (OK??)

MACHINE, ANY

INT, BOOL (error)

EVENT, BOOL (error)

INT, EVENT (error)

errors:

“Values cannot be compared because their types are incompatible": [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\StaticError\ExprsOperators\ExprOperators.p)

error vs int

error vs bool

error vs event

[bool vs int](../../Tst/RegressionTests/Integration/StaticError/TokenRing_Typos/TokenRing_Typos.p)

event vs int

event vs bool

* + - 1. IDX: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature4DataTypes\StaticError\nonAtomicDataTypes\nonAtomicDataTypes.p)

errors:

“Indexer must be applied to a sequence or map"

“Index must be an integer"

“Index may not be in the domain of the map"

* + - 1. IN: [test1](../../Tst/RegressionTests/Feature4DataTypes/StaticError/nonAtomicDataTypes/nonAtomicDataTypes.p)

errors:

“Value can never be in the map"

“"in" expects a map"

* + - 1. Expr produces a tuple type: [test1](../../Tst/RegressionTests/Feature4DataTypes/Correct/nonAtomicDataTypes13/SeqAssgnToAny.p)
      2. NamedExprs produces a named tuple type: [test1](../../Tst/RegressionTests/Feature4DataTypes/Correct/nonAtomicDataTypes13/SeqAssgnToAny.p)
      3. Function application: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature4DataTypes\Correct\nonAtomicDataTypes13\SeqAssgnToAny.p), [test2](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p)

errors:

“function must return a value”  
[“function not defined](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/TransDelc_DoDecl/TransDecl_DoDecl.p)”

“function does not return a value”

“[function requires arguments](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p)”

“function arguments have incorrect types”

“return value has incorrect type”

* + 1. “null” in expressions: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature3Exprs\StaticError\ExprsOperators\ExprOperators.p), [test2](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature4DataTypes\Correct\nonAtomicDataTypesAllAsserts\nonAtomicDataTypesAllAsserts.p)
    2. **REAL and FOREIGN as base types**
    3. **Division by zero, integer underflow/overflow.**

1. Data types in P
   1. Complex (non-atomic) data types in assign/remove/insert: sequences, tuples, maps
      1. “StaticError” tests: [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/lvalues/lvalues.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/StaticError/function_Typos/function_Typos.p), [test3](../../Tst/RegressionTests/Feature4DataTypes/StaticError/nonAtomicDataTypes/nonAtomicDataTypes.p)
      2. “DynamicError” tests: [test1](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes1/nonAtomicDataTypes.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes10/nonAtomicDataTypes.p), [test3](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes11/nonAtomicDataTypes.p), [test4](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes12/StackOverflowException.p), [test5](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes2/nonAtomicDataTypes.p), [test6](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes3/nonAtomicDataTypes.p), [test7](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes4/nonAtomicDataTypes.p), [test8](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes5/nonAtomicDataTypes.p), [test9](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes6/nonAtomicDataTypes.p), [test10](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes7/nonAtomicDataTypes.p), [test11](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes8/nonAtomicDataTypes.p), [test12](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes9/nonAtomicDataTypes.p), [test13](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes14/nonAtomicDataTypes.p), [test15](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/nonAtomicDataTypes15/Insert4Map.p),
      3. “Correct” tests: [test1](../../Tst/RegressionTests/Feature4DataTypes/Correct/nonAtomicDataTypes/nonAtomicDataTypes.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/Correct/nonAtomicDataTypes13/SeqAssgnToAny.p), [test3](../../Tst/RegressionTests/Feature4DataTypes/Correct/nonAtomicDataTypesAllAsserts/nonAtomicDataTypesAllAsserts.p), [test4](../../Tst/RegressionTests/Feature4DataTypes/Correct/nonAtomicDataTypes16/nonAtomicDataTypes.p)

Errors:

“same name appears more than once in name tuple or named tuple type"

* 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. Other non-atomic types: refer to 4.1 above.
  2. Subtype relation among data types: [assignments](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/function/function.p), tests from section 4.1 above.
  3. Paylod type inference:

“invalid payload type in send”: [test1](../../Tst/RegressionTests/Feature4DataTypes/StaticError/function_Typos/function_Typos.p), [test2](../../Tst/RegressionTests/Feature2Stmts/StaticError/sends/sends.p), [test3](../../Tst/RegressionTests/Feature4DataTypes/StaticError/payloads/payloads.p)

“invalid payload type in monitor (cannot send null value)”: [test1](../../Tst/RegressionTests/Feature2Stmts/StaticError/sends/sends.p),

"invalid payload type in send (cannot send null value)": [test1](../../Tst/RegressionTests/Feature4DataTypes/StaticError/payloads/payloads.p)

"invalid payload type in raise": [test1](file:///D:\PLanguage\PLang1001\plang\Tst\RegressionTests\Feature4DataTypes\StaticError\payloads\payloads.p)

Tests under \Tst\RegressionTests\Feature4DataTypes\StaticError with the name “payloadXXX”: these test payload type inference which results in error messages related to assignments, maps, tuples, etc.

* 1. Types of operands in expressions: refer to 3.4. above.
  2. “any” type: [test1](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/anyType/anyType.p), [test2](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/anyType1/anyType1.p), [test3](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/anyType2/anyType2.p), [test4](../../Tst/RegressionTests/Feature4DataTypes/DynamicError/anyType3/anyType3.p), [test5](../../Tst/RegressionTests/Feature4DataTypes/Correct/anyTypeNullValue/anyTypeNullValue.p)

## 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) **}: tests exist?**
2. No raise, pop or call in functions: refer to 4.1.7 below.
3. **Basic semantics tests**

**Note 1:**  For the purposes of testing P semantics, we use delay-bounding scheduling (to avoid randomness). To deal with explicit non-determinism, we use choice bounding option: -bc:1

**Note 2:** The tests should be written in such a way that **regardless of scheduling**, the (preferably single) explicit "assert" in the test case, or Zing exception, should always be reachable.

**Note 3:** We assume single-threaded model of execution: one thread per machine (even for multiple instances of the same machine).

**Note 4:** To test basic semantics of P constructs, re-use static error tests from the section “P Features to test”

**Note 5:** For testing some semantical features, it is necessary to have a pair of tests for a single feature: to confirm that the feature works as expected, comparison between the results of the two tests is needed. For example, to check that atfer the “halt” event, the receiving machine M does not process any event E sent to it afterwards, there are two tests: one test demonstrates that “assert(false)” that would be enabled if the E event got processed, is not in fact enabled (and Zing “passes”); in another test – which only differs in that no “halt” event is ever sent – the “assert” is in fact enabled (and Zing “fails”).

Such paired tests have related names: XXX.p and XXX\_v.p (“v” for “validate”).

Examples: EventSentAfterSentHalt.p and EventSentAfterSentHalt\_v.p; EventSentAfterRaisedHalt.p and EventSentAfterRaisedHalt\_v.p.

* 1. **Single instance of a machine.**
     1. Entry function
        1. “send to itself: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_3/SendInEntry.p)
        2. “send” to itself and “raise”: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_1/SendRaiseInEntry.p)
        3. “raise” and “send” to itself: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_2/RaiseSendInEntry.p)
        4. “raise” and “send” with non-constant event expression: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_9/NonConstantEventExpr.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_10/NonConstantEventExpr1.p), [test3](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_16/NonConstantEventExpr2.p)
        5. Entry function is not executed when the state is re-entered: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_11/PushImplicitPopWithSend.p)
        6. “new”: [test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_1/Actions_1.p) , test2, test3 (and many others)
        7. “push”: [test1](../../Tst/RegressionTests/Integration/DynamicError/Actions_2_fails/Actions_2_fails.p) (and many others)
        8. “pop”: [test1](../../Tst/RegressionTests/Integration/Correct/Multi_Paxos_3/Multi_Paxos_3.p) (and many others)
        9. “monitor”: [test1](../../Tst/RegressionTests/Integration/Correct/Elevator/Elevator.p) (and other tests)
        10. “new” for monitor: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_13/NewMonitor.p), test2
        11. monitor invocation with non-constant event expression: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_11/NonConstantEventExprMonitor.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_12/NonConstantEventExprMonitor1.p), [test3](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_15/MonitorInvocationEventExprPayload.p)
        12. “assert”: [test1](../../Tst/RegressionTests/Integration/Correct/Actions_2/Actions_2.p) (and many others)
        13. Remove/assign/insert: see static checks: see 4.1.7. below: <#FunctionStmts>
     2. Exit function
        1. “send to itself: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_4/SendInExitNotExecuted.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_5/SendInExitUnhandledEvent.p), [test3](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_6/SendInExitHandledEvent.p), [test4](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_7/SendInExitUnhandledHalt.p)
        2. “raise”: [tests in the “Combined Tests” section](#Combined_RaiseInExit) (error)
        3. Exit function executed upon implicit “pop”: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_15/ImplicitPopExit.p)
        4. Exit function executed upon explicit “pop”: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_16/ExplicitPopExit.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_20/BugReproIdenticalTerms.p), [test3](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_21/BugReproIdenticalTerms_workaround.p)
        5. “new”: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_25/NewInExit.p)
        6. “push”, “pop” not allowed in “exit” function: <#ControlImpureTests>
        7. **“monitor”: allowed in exit function**
        8. **Function stmt:**
        9. **“assert”:**
        10. **Remove/assign/insert:**
        11. **“return”:**

**in named function:**

**In anonymous function:**

* + - 1. **While/ite/seq:**
    1. Transitions
       1. Goto transition: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_6/SendInExitHandledEvent.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_8/GotoToItself.p), [test3](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_14/GotoTransInheritance.p),
       2. Push transition: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_9/PushItself.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_10/Push.p),

“push” with implicit “pop”: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_11/PushImplicitPopWithSend.p) [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_17/PushImplicitPopWithRaise.p)

“push” with explicit “pop”: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_12/PushExplicitPop.p)

“push” with action inherited by the pushed state: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_13/PushTransInheritance.p)

* + - 1. “null” as a transition trigger: refer to 4.1.5. below
      2. “halt” as a transition trigger: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_TwoMachines_3/EventSentAfterSentHaltHandled.p)
      3. Statements in transitions (on E goto S;{ stmts }): [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureEnclosedFunCalls/EnclosedFunCalls.p) (and many others in the same subfolder)
    1. “Do” declaration:
       1. “null” as a trigger: refer to 4.1.5. below
       2. “halt” as a trigger: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_TwoMachines_8/HaltTrigger4Do.p) (two machine instances)
       3. “push”, “pop”, “raise” not allowed: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Combined\StaticError\ControlImpureInDoDecl\ControlImpureDoDecl.p)
       4. “defer” as an action: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_18/UnhandledEventDeferred.p)

event both deferred and handled in the same state (static error): [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/StaticError/EventDeferredHandledSameState/EventDeferredHandledSameState.p)

event deferred in a state, but has an inherited handler (top handler – “defer” overrides): <#TopHandlerOverridesInherited>

“defer” and state stack: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_28/DeferIgnore2.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_29/DeferIgnore3.p), [test3](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_30/DeferIgnore4.p)

* + - 1. “ignore” as an action: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_19/UnhandledEventIgnored.p)

event ignored in a state, but has an inherited handler (top handler – “ignore” overrides): <#TopHandlerOverridesInheritedIgnore>

* + - 1. defer/ignore interaction:

two events in front of the queue: 1st is ignored, 2nd is deferred: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Integration\DynamicError\SEM_OneMachine_28\DeferIgnore2.p)

two events in front of the queue: 1st is deferred, 2nd is ignored: [test2](file:///D:\PLanguage\plang\Tst\RegressionTests\Integration\DynamicError\SEM_OneMachine_29\DeferIgnore3.p)

* + - 1. Anonymous function as an action: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\Correct\nonAtomicDataTypes\nonAtomicDataTypes.p), [test2](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInDoDecl/ControlImpureDoDecl.p)
      2. Named function as an action: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\Correct\nonAtomicDataTypes\nonAtomicDataTypes.p), [test2](file:///D:\PLanguage\plang\Tst\RegressionTests\Combined\StaticError\ControlImpureInDoDecl\ControlImpureDoDecl.p)
      3. **Statements in “do” declaration:**

**“push” statement**

**“raise” statement**

**“send” statement**

**Monitor invocation**

**Function statement**

**“assert”:**

**Remove/assign/insert:**

**“return”:**

While/ite/seq:

* + 1. “Special” events: “halt”, “null”.
       1. no “unhandled exception” for unhandled “halt”: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_7/SendInExitUnhandledHalt.p)
       2. raised “halt”: [unhandled](file:///D:\PLanguage\plang\Tst\RegressionTests\Integration\Correct\SEM_OneMachine_31\RaisedHalt.p), [handled](file:///D:\PLanguage\plang\Tst\RegressionTests\Integration\DynamicError\SEM_OneMachine_32\RaisedHaltHandled.p)
       3. simple semantcis of “null” event: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_36/NullEventHandler1.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_37/NullEventHandler2.p" \o "NullEventHandler2)
       4. inherited “null” handler: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_38/NullHandlerInheritedByPushTr.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_39/TopNullHandlerOverridesInherited.p)
       5. “null” handler in a loop: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_41/NullHandlerInLoop.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/BangaloreToRedmond/BangaloreToRedmond.p) (to set priorities between received events), [test3](../../Tst/RegressionTests/Integration/Correct/LinearTopology_5/LinearTopology_5.p) (for waiting in a state)
    2. Hot/warm/cold states:
       1. Hot/cold states in liveness: see section [“Liveness Tests”](#LivenessTests)
       2. Warm states: (get from liveness tests)

For passing liveness test: remove “cold” – the test now fails: [test1](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_1_WarmState/Liveness_1_WarmState.p)

For failing liveness test: remove “hot” – the test passes: [test1](../../Tst/RegressionTests/Liveness/Correct/WarmState_1/Liveness_2_WarmState.p)

* + 1. Function stmt (in any context):
       - 1. with explicit “return”, named function: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/functionAny/functionAny.p) (and many others)

no return, named function: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/MaxInstances_2/MaxInstances_2.p) (and many others)

with explicit “return”, anonymous function (not allowed): <#AnonFunDecl>

no “return”, anonymous function: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_2/EventSentAfterSentHalt_v.p) (and many others)

* + - * 1. While/ite/seq:

While stmt: [test1](../../Tst/RegressionTests/Integration/DynamicError/PushStatement_1/PushStatement_1.p), [test4](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\Correct\SEM_OneMachine_35\bubble_sort.p)

If-then-else: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/DynamicError/AlonBug_fails/AlonBug_fails.p), [test2](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\Correct\SEM_OneMachine_35\bubble_sort.p)

Seq: [test1](../../Tst/RegressionTests/Feature2Stmts/Correct/nonAtomicDataTypes/nonAtomicDataTypes.p), [test2](../../Tst/RegressionTests/Feature2Stmts/Correct/SEM_OneMachine_33/seq_tuple.p), [test3](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\Correct\SEM_ONeMachine_34\payloadActionsFuns.p), [test4](../../Tst/RegressionTests/Feature2Stmts/Correct/SEM_OneMachine_35/bubble_sort.p)

* + - * 1. “assert”: [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/DynamicError/AlonBug_fails/AlonBug_fails.p) (in anonymous function, many other tests); [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_28/DeferIgnore2.p) (named function)
        2. “remove”/”assign”/”insert”

Assignment: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\Correct\nonAtomicDataTypes\nonAtomicDataTypes.p)  
asgn to uninitialized sequence (runtime error): [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_40/lvalues_runtimeError.p)

Remove/Insert: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Feature2Stmts\Correct\nonAtomicDataTypes\nonAtomicDataTypes.p), [test2](../../Tst/RegressionTests/Feature2Stmts/Correct/SEM_ONeMachine_34/payloadActionsFuns.p)

* + - * 1. “control impure” stmts “push”, “pop”, “raise” not allowed:

named function: [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureEnclosedFunCalls/EnclosedFunCalls.p)

anonymous function: [test1](file:///D:\PLanguage\plang\Tst\RegressionTests\Combined\StaticError\ControlImpureEnclosedFunCalls\EnclosedFunCalls.p)

also: <#ControlImpureTests>

* + 1. Deadlocked state:
       1. Queue is empty: [there’s no diff between the case when the deadlocked state is the only one left on the stack,

or not the only one]

Deadlocked state is the only one on the stack**:** [test1](../../Tst/RegressionTests/Feature1SMLevelDecls/Correct/AlonBug/AlonBug.p)

There are other states on the stack:[test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_22/AlonBug_1.p)

All events are deferred: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_18/UnhandledEventDeferred.p)

* + 1. **Priorities among transitions and actions:**
       1. For an event, there’s both a handler defined in a state, and an inherited handler (top – local - handler overrides): [test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_24/TopHandlerOverridesInherited.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_23/TopHandlerOverridesInherited_v.p)
       2. Event ignored in a state, but has an inherited handler (top handler – “ignore” overrides): [test1](../../Tst/RegressionTests/Integration/Correct/SEM_OneMachine_26/TopHandlerOverridesInherited.p" \o "TopHandlerOverrides), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_OneMachine_27/TopHandlerOverridesInherited_v.p)
       3. **(Local) step or push transition has higher priority over actions (paper, p. 6) TODO: write a test, compiler should catch**

**Quote from the paper: “**if Step(m; n; e) or

Call (m; n; e) is defined, it takes higher priority over actions.”

* + 1. Data types semantics: refer to sections 3 and 4 in <#P_Features>
    2. **Multi-threaded runtime tests: 1 thread per machine instance**
       1. **One instance of the machine, 1 thread**
       2. **Multiple istances of the machine, multiple threads**
  1. **Two instances of machine(s)**
     1. **Basic features for communications between instances**
        1. “send”, “new”, “ignore”, “null”, handling all possible events in all states; non-constant, non-atomic event expressions, events as payloads: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_17/EventExprSemantics.p)

“unhandled event” exception in case of non-constant even expressions: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_18/EventExprSemantics1.p)

“trigger” in event expressions: [test1](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_19/EventExprSemantics2.p)

* + - 1. **extend tests for one instance from 4.1.: for stack, inheritance, etc. for the two-instances case**
    1. **Various number of instances of two machines**
    2. “halt” event in the two-instances context:
       1. sent “halt”, with “halt” both handled and unhandled: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_TwoMachines_1/EventSentAfterSentHalt.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_2/EventSentAfterSentHalt_v.p), [test3](../../Tst/RegressionTests/Integration/Correct/SEM_TwoMachines_3/EventSentAfterSentHaltHandled.p), [test4](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_4/EventSentAfterSentHaltHandled_v.p), [test5](../../Tst/RegressionTests/Integration/Correct/SEM_TwoMachines_8/HaltTrigger4Do.p)
       2. raised “halt”, with “halt” both handled and unhandled: [test1](../../Tst/RegressionTests/Integration/Correct/SEM_TwoMachines_5/RaisedHalt.p), [test2](../../Tst/RegressionTests/Integration/DynamicError/SEM_TwoMachines_6/RaisedHaltHandled.p), [test3](../../Tst/RegressionTests/Integration/Correct/SEM_TwoMachines_7/RaisedHalt_bugFound.p),
    3. “null” event in the multi-instances context: [test2](file:///D:\PLanguage\plang\Tst\RegressionTests\Integration\DynamicError\BangaloreToRedmond\BangaloreToRedmond.p) (to set priorities between received events), [test3](file:///D:\PLanguage\plang\Tst\RegressionTests\Integration\Correct\LinearTopology_5\LinearTopology_5.p) (for waiting in a state)
    4. **Various protocols for communications between instances:**
       1. **Main machine instance and ghost machine instance (env): client/server protocol**

**non-det case (**use choice bounding option: -bc:1)

**deterministic case**

* + - 1. **Async mode of communication**
      2. **Sync mode of communication (PingPong sample?)**
    1. **Multi-threaded runtime tests: 1 thread per machine instance**
       1. **Single instance of each machine**
       2. **Single instance of one machine, multiple instances of another machine**
  1. **Test cases that exercise semantical errors**
     1. **Exercise all values of PRT\_STEP, PRT\_STATUS in Prt.h**
     2. **Exercise all error logging messages in PtoZing.cs (search for:** "\"<")
  2. **Test “model” machines and functions (for example: PingPong.p for TechFest demo)**

## Liveness tests

1. Simple sampes demonstrating liveness checking by Zing
   1. “Pass” result from Zing
      1. Actual “pass” result: [test1](../../Tst/RegressionTests/Liveness/Correct/Liveness_1/Liveness_1.p)
      2. False “pass” result: [test1](../../Tst/RegressionTests/Liveness/Correct/Liveness_1_falsePass/Liveness_1_falsePass.p)
   2. “Fail” result from Zing:
      1. Infinite loop in the “hot” state: [test1](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_2/Liveness_2.p), [test2](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_2_LoopMachineAdded/Liveness_2_LoopMachineAdded.p)
      2. Deadlock in the “hot” state: (bug!) [test1](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_2_bugFound/Liveness_2_bugFound.p)
      3. Liveness violation but no deadlock, at least one thread is making progress: [test1](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_3/Liveness_3.p)
   3. FAIRNODET expressions (liveness-related): [test1](../../Tst/RegressionTests/Liveness/Correct/Liveness_FAIRNONDET/Liveness_FAIRNONDET.p), [test2](../../Tst/RegressionTests/Liveness/Correct/Liveness_FAIRNONDET2/Liveness_FAIRNONDET.p), [test3](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_NONDET/Liveness_NONDET.p), [test4](../../Tst/RegressionTests/Liveness/DynamicError/Liveness_NONDET2/Liveness_NONDET.p)
2. Hot/warm/cold states: see <#HotWarmColdStates>
3. **More complex liveness tests: after the paper is finished, ask Ankush to populate real protocols into SampleProtocols regression suite.**

## Combined Tests

1. Duplicates:

Errors detected during parsing: [test1](../../Tst/RegressionTests/Combined/StaticError/Duplicates/Duplicates.p)

Static errors: [test1](../../Tst/RegressionTests/Combined/StaticError/Duplicates-1/Duplicates-1.p)

1. 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**
2. **Missing declarations**: undefined symbol (must be declared as formal par, variable or event)
3. “Control Impure” statements (“push”, “pop”, “raise”) tests. Contexts:
   1. In exit function (anon and named): [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInExit/ControlImpureInExit.p), [test2](file:///D:\PLanguage\plang\Tst\RegressionTests\Combined\StaticError\RaiseInExitFun\RaiseInExitFun.p), [test3](file:///D:\PLanguage\plang\Tst\RegressionTests\Combined\StaticError\PushInExitFun\PushInExitFun.p), [test4](file:///D:\PLanguage\plang\Tst\RegressionTests\Combined\StaticError\PopInExitFun\PopInExitFun.p)
   2. In “goto” function (anon and named): [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInGoto/ControlImpureErrorsGoto.p)
   3. In function application or function stmt used in:
      1. exit or “goto”: [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInExpr/ControlImpureInExpr.p), [test2](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInExit/ControlImpureInExit.p), [test3](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInGoto/ControlImpureErrorsGoto.p)
      2. entry action: [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInEntry/ControlImpureInEntry.p)
      3. “do” declaration: [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureInDoDecl/ControlImpureDoDecl.p)
   4. Enclosed function calls in various contexts: [test1](../../Tst/RegressionTests/Combined/StaticError/ControlImpureEnclosedFunCalls/EnclosedFunCalls.p)