TPT 18 Severe Issues

Introduction

The following document contains a list of known severe issues of TPT. By severe issues we mean issues/bugs
in particular versions of TPT that:

1. might cause malfunctions in the behavior of TPT
2. are hard or even impossible to find by the TPT user herself/himself
3. cause the risk that bugs/defects in a SUT (system under test) are not detected by TPT in cases where
   TPT would have been able to reveal these bugs/defects in the SUT without the aforementioned
   malfunction in the behavior of TPT.

Usually these severe issues address the situations where the problem might appear and have well-defined workarounds.

ISSUE # P90208485-43040

TITLE: Enum import with duplicate constant can lead to an incorrect value for some of the constants to be imported to TPT

ISSUE DETECTION:
06/18/2024

AFFECTED VERSIONS OF TPT:
TPT 16 to TPT 20

PRECONDITIONS:
Enumeration data type with multiple constants using the same value present in C/C++ example or other data-source for interface import capable of importing enumeration data types to TPT.

DETAILS:
On interface import of an enumeration data type with more than one constant for the same value, the duplicate constants may get imported with an incorrect value.

EFFECT OF THE ISSUE:
The declared enumeration data type in TPT has incorrect values for the affected constants.

WORKAROUND:
Ensure that this use-case does not occur within the data source of the interface import before import the interface to TPT or manually review the constants for imported enumeration data types.

RESOLVED IN:
TPT 2024.12

ISSUE # 35247
==========
TITLE: Incorrect coverage results for MC/DC when using TPT Coverage with complex left-bound logical operations.

ISSUE DETECTION:
25-May-2023

AFFECTED VERSIONS OF TPT:
TPT 18 to TPT 18u3, TPT 19

PRECONDITIONS:
The user-code (C/C++ Platform) or a Stateflow transition (MATLAB platform) contains a complex left-bound inter logical condition - e.g. within a single decision.
TPT Coverage (TASMO) is enabled for the platform or the TASMO test data generation is used.

DETAILS:
The MC/DC coverage for some goals may be evaluated as covered although it is not.

EFFECT OF THE ISSUE:
Some coverage goals may be marked as covered although they are not. The overall statistics may have incorrect numbers.

WORKAROUND:
Do not measure the MC/DC Coverage with TPT coverage (TASMO) if this use-case occurs.

RESOLVED IN:
TPT 18u4, TPT 19u1

ISSUE # 34814
==========
TITLE: Compiler-Bug when compiling index-based array-accesses for declared constants that are arrays or structs containing arrays.

ISSUE DETECTION:
25-February-2023

AFFECTED VERSIONS OF TPT:
TPT 17 to TPT18u2

PRECONDITIONS:
A constant must be declared which has an array type, matrix type or a struct type that contains an array.
In the test model there is at least an index-based access to this array constant, i.e. "myarray[index]" where "index" differs from "0".

DETAILS:

Assume that the above conditions hold in a TPT model. Then the TPT compiler has produced faulty code, so that instead of the access
"myarray[index]" always "myarray[0]" was used. Note that this error only occurs with declared constants and **not** with declared
channels or declared parameters.

EFFECT OF THE ISSUE:
The wrong values of declared constant arrays in the test model are accessed and, if used, might result in wrong test results.

WORKAROUND:
Convert all declared array constants to local array parameters. The mentioned error does not occur with declared array parameters or array channels.

RESOLVED IN:
TPT 17u5, TPT18u3

ISSUE # 33962
==========
TITLE:
Assessment variables in *Global Assesselts* shall forbid using numeric operations (+,-,*,/,%) and
comparisons (>,>=,<,<=,==,!=) to avoid misunderstandings about the semantics.

ISSUE DETECTION:
01-November-2022

AFFECTED VERSIONS OF TPT:
TPT 18u1

PRECONDITIONS:
when using Global Assesslets with global variables in conjunction with numeric or comparison operators.

DETAILS:
Assessment variables in *global* assessment contain the values of all contributing test cases as a
list-of-values. For example: If an assessment varible "a" is defined in 5 test cases as 1,2,3,4, and 5, the
variable "a" in Global assesslet has the list-of-values [1,2,3,4,5].
This might cause confusions when using these variables in numeric operations (+,-,*,/,%) or in comparisons
(>,>=,<,<=,==,!=) because the user might expect operations of the *individual* values instead of list
operations.
Therefore, the mentioned operators shall be forbidden in Global Assesslets at all.
Note that this change might cause existing test models to fail in global assesslet (compatibility issue!).

EFFECT OF THE ISSUE:
misunderstandings in the semantics of the aforementioned operators.

WORKAROUND:
do not use the aforementioned operators for global assessment variables. Use the operators *only* for individual
values of these variables (example: use "if a[idx] < 2:" instead of "if a < 2:")

RESOLVED IN:
TPT 18u2

ISSUE # 33511
==========
TITLE:
Wrong times used for TPT.getConstant(...) and TPT.isConstant(...) with local time

ISSUE DETECTION:
22-July-2022

AFFECTED VERSIONS OF TPT:
TPT 18, TPT 18u1

PRECONDITIONS:
Using of TPT.getConstant(timed_float expr, time starttime, time endtime) or TPT.getConstant(timed_float expr, time starttime, time endtime).
Please note that TPT.getConstant was falsely documented as expecting a timed_bool instead of a timed_float for 'expr'.

DETAILS:
When using TPT.getConstant(timed_float expr, time starttime, time endtime) or TPT.getConstant(timed_float expr, time starttime, time endtime) with a local time context, e.g. inside a during,
the time values passed to the methods were interpreted as global time values instead of local time values. So the
methods use the wrong time interval to calculate the constant value for the given timed expression.

EFFECT OF THE ISSUE:
The wrong time intervall is used to calculate the constant value. So a wrong result can be returned:
- TPT.getConstant() can return a wrong constant value for the given (local) time interval
- TPT.getConstant() can return a constant value instead of None if the given expression is not constant in the
  given (local) time interval or vice versa
- TPT.isConstant() can return True instead of False if the given expression is not constant in the given (local)
  time interval or vice versa

WORKAROUND:
As a workaround the starttime and endtime can be manually translate to global time by using TPT.toGlobal(time localtime).
However, this workaround will lead to wrong results when using a TPT version where the issue is fixed (TPT18u2 and higher versions).

RESOLVED IN:
TPT 18u2

ISSUE # 33413
=============
TITLE:
Assessment function REQUIREMENTS.checked() does not work correctly, if the second argument is a signal
or type boolean or int and not a verdict.

ISSUE DETECTION:
07-June-2022

AFFECTED VERSIONS OF TPT:
TPT 13 to TPT 18

PRECONDITIONS:
TPT Script Assessment contains REQUIREMENTS.checked("REQ-1",check_sig) and the second argument is a
signal and not the result direct via verdict.

DETAILS:
If inside a script Assesslet the function
REQUIREMENTS.checked("REQ-1",check_sig) is used with a signal
with a result, the function does not use the result of signal check_sig, instead the function uses the value
of this signal.

```
check_sig = TPT.UInt8X()
check_sig := 0
check_sig.setResult(TPT.PASSED)
REQUIREMENTS.checked("REQ-1",check_sig)
```
In this case "REQ-1" is marked as FAILED, instead of PASSED.

EFFECT OF THE ISSUE:
Requirements get the wrong result.

WORKAROUND:
Use instead
```
REQUIREMENTS.checked("REQ-1",check_sig.getResult())
```

RESOLVED IN:
TPT 17u4, TPT 18u1

ISSUE # 33174
=============
TITLE:
TPT trigger rule, implausible interval discarding, when using "t" in 'Abort'/'Ignore intervals if' conditions.

ISSUE DETECTION:

02-May-2022

AFFECTED VERSIONS OF TPT:
TPT 13 to TPT 18

PRECONDITIONS:
The TPT project contains Trigger Rule assesslets using "t" in 'Abort'/'Ignore intervals if' conditions.

DETAILS:
In general, abort conditions influence whether an interval is used for analyzing THEN/ELSE-checks or not.
If an 'Abort'/'Ignore intervals if' condition is true within such an interval, the interval is discarded and
no THEN/ELSE checks take place in this particular interval.
The inconsistency arises from the use of "t" in the 'Abort condition' of "Trigger condition"-checks because
"t" refers for THEN-intervals to the *global* time (time elapsed since the trigger rule evaluation started),
but for ELSE-intervals to the *local* time (time since the beginning of the respective interval).
Furthermore, in the 'Ignore intervals if' condition of "While condition is true"-checks "t" refers for both,
THEN- and ELSE-intervals, to the *local* time (time since the beginning of the respective interval).
Since "t" in the START/STOP conditions always refer to the global time, this should also be uniformly
corrected to the *global* time in all 'Abort'/'Ignore intervals if' conditions. To avoid incompatiblity for
existing models, this behavior shall be able to explicitly be turned on/off in the TPT Tool Preferences
(settings per model).

EFFECT OF THE ISSUE:
The intervals discarded according to 'Abort'/'Ignore intervals if' conditions containing "t" might be
discarded or retained under implausible (but deterministic and well-defined) conditions.

WORKAROUND:
There is no real workaround, since the old behavior is not wrong! It is simply implausible or inconsistent
and should therefore be adjusted to avoid misunderstandings. For this reason, the behavior was only
adjusted via a preference option (configurable via TPT Tool Preferences): As long as this option is not
selected in old TPT models, the behavior remains unchanged even after the fix to maintain compatibility.
In new TPT models, however, the TPT Tool Preference option is automatically preset.

RESOLVED IN:
TPT 17u4, TPT 18u1