Characterising Concurrent Tests Based On Message Sequence Chart Requirements

Bill Mitchell
b.mitchell@motorola.com
UK Motorola Research Lab
Viables Industrial Estate
Hampshire RG22 4DP, UK

Keywords Message Sequence Charts, Concurrent TTCN, Coordinating Messages.

Abstract

MSC-s specifying system requirements can be used to generate test scripts in languages such as TTCN. In 1996 TTCN was extended in order to describe concurrent test scripts.

We give a characterisation of which concurrent tests can be executed by parallel test components (PTC-s) acting in concert. We also characterise when the PTC-s give the correct test result verdict.

1 Introduction

Often half the development life cycle of a telecommunications system is spent in testing. Distributed systems are sometimes tested using parallel test components (PTC-s). These are distributed autonomous processes which run in parallel. They interrogate the system being tested to ensure that it's observed behavior matches the requirements specification.

In 1996 the TTCN (Tree and Tabular Combined Notation, a testing description language) test pupose was extended to describe concurrent test scripts. Subsequently it was realised that there are various problems with concurrent testing.

Firstly, how to ensure that the PTC-s are cooperating correctly with respect to the test specification, and secondly how to ensure that they are reporting the correct verdict after the test. There is also the significant problem of what concurrent tests are possible. Is it the case that any possible test can be executed via correctly coordinated PTC-s?

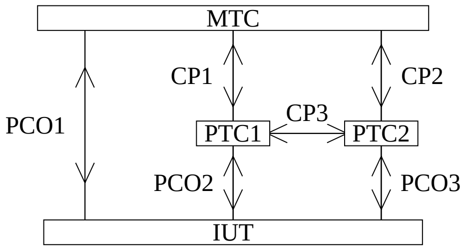
In the paper we look at these difficulties in the context of conformance tests which are driven by requirements captured using MSC-s (Message Sequence Chart-s). As discussed in [1] it is not possible to derive sound concurrent tests from an MSC which contains a race. We discuss racing in more detail later, but roughly a race occurs in an MSC where two events are defined to occur in a particular order which is not possible to guarantee in practise.

We characterise tests in terms of tightly coupled partial orders. We give an exact characterisation of those concurrent tests that can be executed by PTC-s acting in concert, and also when it is possible for the PTC to come to a correct collective verdict on the test result.

Clearly a concurrent test can be converted into a non-concurrent test. Thus in some sense they represent a subclass of the non-concurrent tests. This paper characterises the class of non-concurrent tests which can be converted into concurrent tests. This characterisation holds for the class of sound tests (that is the test will not give false results). Given a specific test it is not obvious whether it will be sound or not. We define the notion of a viable test, which can be effectively established for a given test and is equivalent to soundness when the MSC is race free.

2 The TTCN Model

The TTCN model of concurrent testing assumes that the IUT is running in parallel with a unique master test controller (MTC) and a number of PTC-s. The MTC is in overall control and it is also responsible for storing the test verdict. This verdict can be none, pass, fail or inconclusive. The TTCN model supposes there are dedicated communication channels between the test components. The PTC-s and MTC interact with the IUT through points of control and observation (PCO). The PTC-s and MTC coordinate their activity via special messages known as coordination messages (CM-s) which are exchanged through coordination points (CP-s). We view a PCO or CP as a communication channel between two components.

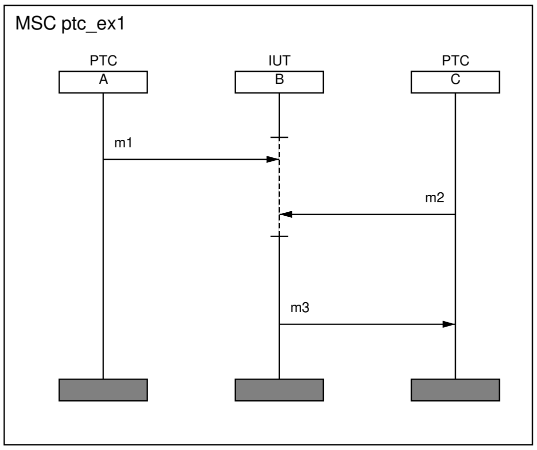


We will assume the concurrent tests are conformance tests derived from MSC-s which capture the requirements specifications. To simplify matters we will assume that an MSC intended for generating tests will use a separate instance for each PTC and the IUT. This is not normally the case, frequently a PTC is characterised by a group of instances, as will be the IUT, however the restriction will not affect any of the results in the paper, but is helpful in keeping space to a minimum.

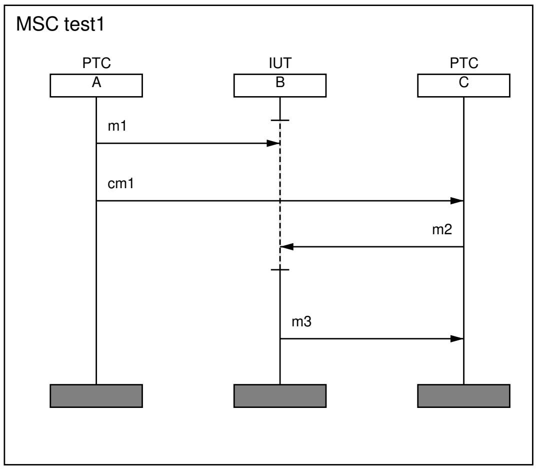
Any test which can be performed by the explicit intervention of the MTC can be performed solely by the PTC-s without any explicit intervention from the MTC. Again for simplicity we will assume that tests are always conducted in this way. Thus we can assume that a test consists of a number of PTC-s which coordinate their activity through CM-s exchanged through CP-s. It is irrelevant to our discussion how many CP-s there are as long as their are enough. We will assume that there is a CP between each pair of PTC-s and a PCO between each PTC and IUT.

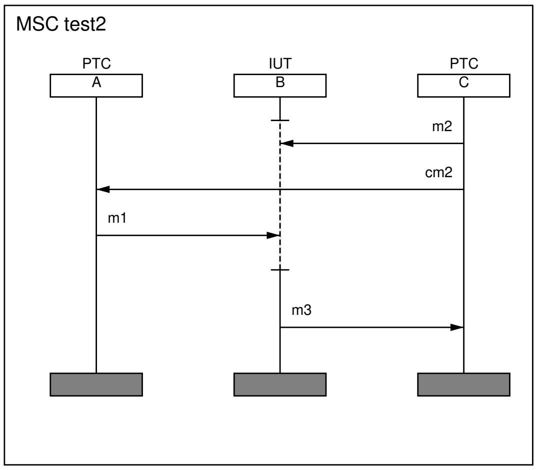
Before considering the technicalities involved in concurrent tests we look at a simple example. Suppose the requirements specifications are in part given by the MSC ptc_ex1 below. Note there is a co-region within the IUT. A co-region defines a region where there is no constraint on the order in which the events in that region occur. Therefore B can validly receive $m1$ and $m2$ in any order.

This MSC determines two possible tests which are required for conformance testing. First we test that if m1 is sent before m2, then B correctly replies with m3. The second test is given by reversing the order, first m2, then m1, then check that m3 is received after that.



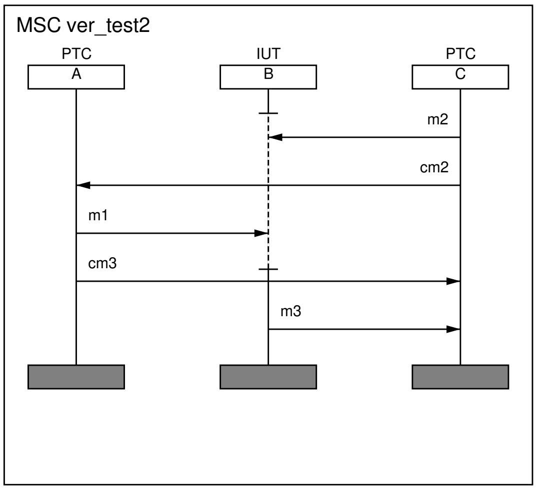
Given that the two PTC-s will run in parallel how can we implement each of these two tests? The two MSC below have introduced new coordinating messages that force the PTC-s to enact these two conformance tests.





In each of these MSC we have introduced CM-s which enable the PTC to coordinate their actions in accordance with a specific test they are implementing. In MSC test1 we have forced PTC C to wait for the coordinating message cm1 before it can send m2. If the behavior of this MSC is implemented in the form of a TTCN test script it will correctly perform the first test necessary for conformance testing. Similarly MSC test2 forces PTC A to wait for cm2 which means m2 is sent before m1.

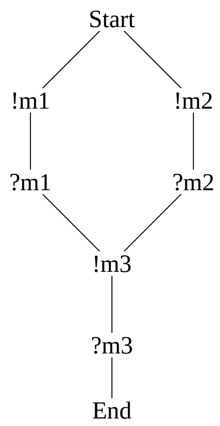
There is one other function the PTC must perform. They must decide after the test is run what verdict to reach. To do this more CM-s are necessary so that the PTC-s can decide if the IUT has responded correctly to messages from the PTC-s. In the case of MSC test2, we need to add a new CM, as drawn in MSC ver_test2, to allow PTC C to judge the verdict of the test.



This new CM allows PTC C to check that m3 is received after m1 is sent. With test1 there is no need for an additional CM since m1 is the final trigger for m3. The receipt of cm1 is proof that PTC A has sent m1. Thus as long as PTC C receives m3 after it receives cm3 it will give the test a pass verdict.

3 What is a test

The standard MSC semantics [5] defines a partial temporal order on the events in the MSC. Each message m is split into a send event $!m$ and a receive event $?m$, the partial order is defined over the set of these events. For example the partial order given by MSC ptc_ex1 is



The partial order $<_M$ defined by the MSC M is the set of temporal constraints which is imposed by the MSC on the order of the events within it. Note, in the example because of the co-region $?m1$ and $?m2$ are unordered. Also, even though $!m2$ is drawn visually below $!m1$, they are not forced to be temporally ordered in that way since the time line represented by A is not comparable to that of C.

Concurrent (and non-concurrent) testing is black box testing. If events sent and received by the PTC-s are ordered consistently with the ordering of events specified by a given test then the IUT passes that test. Let S(PTC/IUT) denote the set of events sent by the PTC-s to the IUT. Let R(PTC/IUT) be the set of events received by the PTC-s from the IUT.

Let $X = S \cup R$ be a set of events from MSC M, where S is a set of send events, and R is a set of receive events. A partial order $<_P$ over X is tightly coupled if

1. $<_P$ is consistent with $<_M$,
   i.e $\forall\ x,y \in X,(x <_M y \Rightarrow x <_P y)$
2. $<_P$ is a total ordering of S
3. $\forall\ e \in S, \forall\ r \in R,\ e <_P r$ or $r <_P e$

Define a test to be a partial ordering $<_T$ which is a tightly coupled ordering over S(PTC/IUT) $\cup$ R(PTC/IUT). The intuition is that a tightly coupled ordering is as close as we can get to a total ordering, without explicitely ordered receive events relative to one another.

Notice a test does not have to state an explicit ordering on R(PTC/IUT). It would not be possible in general for a test to force receive events to occur in a particular order. It may well implicitly do so since, for example, it may be that the test forces $r_1 < e_1$ and $e_1 < r_2$, hence we must have $r_1 < r_2$. It is not the case that every test can be correctly implemented. A test is defined to be unsound if a test run can result in a false result. That is it is possible for each PTC and the IUT to be acting in accordance with the test partial ordering, but for the final verdict to be incorrect.

Definition of concurrent test

Suppose we have an MSC M which contains various CM-s passing between the PTC-s. We define the CM ordering $<_{\mathsf{cm}}$ on the events in M which are not contained in the IUT as follows. When K is an instance or an MSC we write $E(K)$ to denote the set of events which occur in K. When K is an instance in M we write $<_K$ to denote the partial order $<_M$ restricted to $E(K)$.

Define the union $<_{1,2}$ of two partial orders $<_1$ and $<_2$ by $x <_{1,2} y$ if $x <_1 y$ or $x <_2 y$. Let $<'$ be the union of all the partial orders $<_K$ where K is any PTC instance. Define $x <'' y$ whenever x is a send event $!m$, y is the corresponding receive event $?m$, and m is a message from one PTC to another PTC (e.g. a coordinating message). Finally define $<_{\mathsf{cm}}$ to be the transitive closure of $<'$ union $<''$.

A concurrent test of an MSC M is a set of coordinating messages which when added to M construct a new MSC M' whose relation $<_{\mathsf{cm}}$ is tightly coupled over S(PTC/IUT) $\cup$ R(PTC/IUT).

For example the ordering specified in MSC ver_test2 is a test since the events in S(PTC/IUT) $\cup$ R(PTC/IUT) are totally ordered, and it is a concurrent test since cm2 forces the $<_{\mathsf{cm}}$ to be tightly coupled.

Latency Assumption

In [1] the authors show that it is not possible to generate sound concurrent tests from an MSC when no assumptions are made about the latency of the coordinating messages. In this paper we make the assumption that the latency of messages sent along CP channels is less than the latency of any message sent along a PCO channel. Thus coordinating messages always travel faster than other messages.

This assumption ensures that it will be possible to introduce coordinating messages without creating racing within the MSC when we want to construct a concurrent test.

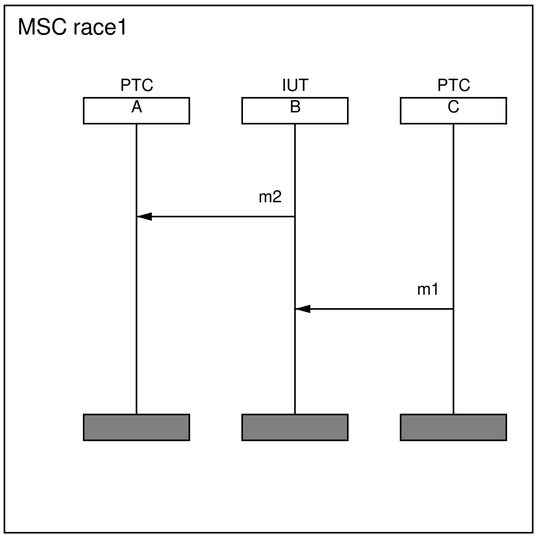
4 Viable Tests

Intuitively a test is sound if a final pass verdict is always reached when the IUT and each PTC act correctly with respect to the test partial order. It is not clear how to establish that a given test is sound. We need a simple criteria which tells us when a test is sound.

For a test T let $<_T$ be the test partial order on the events $\mathrm{E}(M)$. We define a test to be viable if for every e in S(PTC/IUT) and every r in R(PTC/IUT),

$$e <_T r \Rightarrow \exists e' \in \mathrm{S(PTC/IUT)} : e <_T e' <_M r$$

When the MSC is race free a concurrent test is sound if and only if it is viable. For a full discussion of racing in MSC-s see [2]. Crudely one can say an MSC contains a race if the MSC specifies a partial order of events which is not possible to guarantee in practise. Here is a simple example of a race.



This contains a race between $!m2$ and $?m1$. The MSC has specified that B must send $m2$ before it receives $m1$ which is clearly absurd. The MSC contains all observable events and there is no mechanism which informs C that B has sent $m1$ and it is safe to send $m2$.

We do not give a formal proof of the above viability proposition, however we briefly indicate part of a sketch proof. Suppose that we have an MSC M, which is race free, and we have a concurrent test which is *not* viable. We can show that this test must be unsound.

Since the test is unsound there is some $e \in$ S(PTC/IUT) and some $r \in$ R(PTC/IUT) such that $e <_T r$, and there is no event $e' \in$ S(PTC/IUT) such that $e <_T e' <_M r$. Therefore for every event $e' \in$ S(PTC/IUT) such that $e' <_M r$ it must be that $e' \not<_T e$. Test T must be a total ordering on S(PTC/IUT), therefore for every $e' \in$ S(PTC/IUT) we have $e' <_M r$, $e' <_T e$.

Notice that if $e' <_T r' <_M r$, then r' must have this same property. Hence we may suppose r is minimal with this property. We have now constructed an unsound occurrence within the test. The test requires $e <_T r$, and yet there is no linkage between the two events, so in practice it will be possible for r to occur before e in any given test run.

To prove that a viable concurrent test is sound when M is race free, it is enough to apply the algorithm in [1] to the test with suitable coordinating messages introduced to ensure the PTC-s are correctly synchronised.

5 Classifying concurrent tests

In this section we look at what concurrent tests are possible. To do this we give a theorem which shows when a (non-concurrent) test can be converted into a concurrent test.

Suppose we have an MSC M which is race free and a tightly coupled viable test T. The order $<_T$ is total on $S(PTC/IUT)$. Introduce suitable CM-s which force this ordering on $S(PTC/IUT)$ to occur in practise. Add suitable CM-s so that $e <_{\mathsf{cm}} r$ whenever $e \in$ S(PTC/IUT) and $r \in$ R(PTC/IUT) and $e <_M r$. Finally add suitable CM-s so that for each pair $e \in$ S(PTC/IUT) and $r \in$ R(PTC/IUT) where $r <_T e$ then $r <_{\mathsf{cm}} e$.

Theorem 5.1
The order $<_{\mathsf{cm}}$ is tightly coupled over S(PTC/IUT) $\cup$ R(PTC/IUT) and is viable.

In other words the CM-s added to M force the PTC-s to be synchronised so that they correctly implement the non-concurrent test T as a concurrent test. It is always possible to add coordinating messages to a viable test in the way described above, hence converting it into a concurrent viable test. In fact this represents the minimal set of CM-s which are required for such a conversion.

Proof of Theorem

MSC M_{cm} is explicitly constructed to force S(PTC/IUT) to be totally ordered as defined by T. Thus we only need concern ourselves with $e \in$ S(PTC/IUT) and $r \in$ R(PTC/IUT).

If $e <_M r$ or $r <_M e$ it follows from the definition of $<_{\mathsf{cm}}$ that $e <_{\mathsf{cm}} r$ or $r <_{\mathsf{cm}} e$.

If $r <_T e$ then again it follows from the definition of $<_{\mathsf{cm}}$ that $e <_{\mathsf{cm}} r$.

The only case left to consider is if $e <_T r$. Since M is race free the only events which can occur immediately before r must be send events $e' \in$ S(PTC/IUT) and $e' <_M r$. Also there must be at least one such event since $<_T$ is viable.

Events e and e' are ordered by $<_T$, and therefore by $<_{\mathsf{cm}}$. If $e <_T e'$ then also $e <_{\mathsf{cm}} e'$ and so $e <_{\mathsf{cm}} e' <_M r$. If $e' <_T e$ for every such e' we contradict that $<_T$ is viable, because this implies there is not $e'' \in$ S(PTC/IUT) where $e <_T e'' <_M r$. Hence there is at least one e' where $e <_{\mathsf{cm}} e' <_M r$, which proves that $<_{\mathsf{cm}}$ is tightly coupled. Note we have also proved that $<_{\mathsf{cm}}$ is viable at the same time.

6 Conclusion

It is not immediately obvious what constitutes a valid conformance test derived from an MSC. Intuitively, and incorrectly, we would like to think a test is a total ordering on events from the MSC which represents one particular way that the PTC can interleave with each other and the IUT. This intuition is wrong since sometimes it just is not possible to force particular events to occur in a certain order when the IUT is not under direct control by the observer. The best that is possible is to consider partial orders of events which are tightly coupled and viable.

The viable tightly coupled partial orders turn out to characterise those non-concurrent tests which can be converted into sound concurrent tests.

References

[1] Jens Grabowski, Beat Koch, Michael Schmitt, Dieter Hogrefe, SDL and MSC Based Test Generation for Distributed Test Architectures. In: 'SDL'99 - The next Millenium' (Editors: R. Dssouli, G. v. Bochmann, Y. Lahav), Elsevier, June 1999. http://www.itm.mu-luebeck.de/english/publications/grabowski.html

[2] Gerard J. Holzmann, Doron A. Peled, and Margaret H. Redberg, Design Tools for Requirements Engineering, Bell Labs Technical Journal, Winter 1997

[3] ISO Information Technology - OSI-Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN) ISO IS 9649-3, 1996

[4] Z.100 (11/99) ITU-T Recommendation - Languages for telecommunications applications - Specification and description language http://www.itu.int/itudoc/itu-t/approved/z/index.html

[5] Z.120 (11/99)ITU-T Recommendation - Message Sequence Chart (MSC) http://www.itu.int/itudoc/itu-t/approved/z/index.html