Coordinating Distributed Requirements Conformance Tests

Bill Mitchell

w.mitchell@surrey.ac.uk,
Department of Computing, University of Surrey
Guilford, Surrey GU2 7XH, UK

Abstract. Message sequence charts (MSC) requirements specifications are often used to generate TTCN conformance tests for telecommunications protocols. There are difficulties with constructing concurrent TTCN test scripts from an MSC used to describe test purposes. In [4] they demonstrate that without any timing restrictions it is not possible to automatically construct a concurrent test script from an MSC.
They do not discuss in their paper when it is meaningful to construct concurrent conformance tests from the MSC. In this paper we give a characterisation of when coordinating messages added to an MSC produce a faithful and sound concurrent test. These reults can be implemented in order to automatically generate such tests directly from the MSC.

1 Introduction

The specification of the air interface of a wireless telecommunications system is defined in the form of a communication protocol for a set of asynchronous concurrent communicating processes. Once such a protocol has been defined a suite of conformance tests is constructed from the specifications which are used to verify any implementation with respect to this protocol.

There are various standardised formal languages which are used in this process of specifying and defining tests for an air interface which are commonly used. Requirements specifications are often captured in the form of message sequence charts MSC-s [9]. These specifications can be refined and implemented in SDL [8] or some other suitable language [1], [7]. Conformance tests can be constructed in TTCN [6] from the MSC requirements and evaluated through simulation of the SDL model [3].

Concurrent test scripts were incorporated into TTCN in 1996. The TTCN model of concurrency allows the implementation under test (IUT) to be interrogated by a collection of parallel test components (PTC-s) to determine if the IUT passes some given concurrent test. The PTC-s can be autonomous processes running over a distributed system. In order for the PTC-s to synchronise with each other they use coordinating messages (CM-s).

The standard formal semantics for MSC-s in [9] define messages to have arbitrary latency. In [4] they show that if the coordinating messages also have arbitrary latency then it will not be possible to automatically generate suitable coordinating messages which allow the PTC-s to conduct a given concurrent test. This makes it very difficult to decide how to generate concurrent conformance tests from a set of MSC-s which define the requirements specifications. These difficulties occur because the PTC-s are asynchronously communicating concurrent processes.

Clearly the PTC-s need to coordinate so that they are correctly performing the test, but it is not clear that any given configuration of coordinating messages is correct with respect to a given test. We make a modest assumption about the latency of CM-s which allows us to decide when a set of coordinating messages are correct with respect to a given test.

A concurrent test partially defines what order events on the PTC-s will occur during a test run. After a test run the IUT passes the test if the observed order that the events occur in agrees with the test partial ordering. The PTC-s effectively decide if the IUT passes a test. They use coordinating messages to warn each other of the intended next event so that they can judge whether this occurs or not. That judgment is then stored in the master test controller test verdict.

The coordinating messages therefore have two purposes. First to ensure that the PTC-s are correctly ordering events relative to the test partial order. Second, to give warning, when necessary, of which message should occur next so that the test verdict can be updated at relevant moments. It is therefore vital that the coordinating messages do their job correctly.

Before we can decide if the CM-s are correctly placed relative to a concurrent test we need to know that the test is sound. A test is unsound if it expects events to occur in an order which may be impossible to guarantee in practice. We may then have a situation where the test terminates with a fail verdict because it is not possible for the IUT to comply with the test. Conversely a test is sound if it it is possible for the IUT to comply with the test ordering of events and when it does so the test run terminates with a pass verdict.

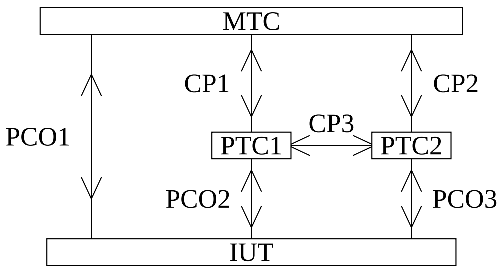
We provide a criteria for determining if a test will be sound. We define viable (definition 4) tests and show (theorem 5) that this is a necessary condition for a test to be sound.

If the CM-s are introduced in the wrong way then the test might result in a pass verdict when it should have failed. When this happens we say the CM-s are unfaithful. This occurs when the CM-s have not restricted the observable behavior strongly enough relative to the test criteria. Conversely then, the CM-s are faithful to the test if a pass verdict can only occur when the observable behavior complies with the test ordering.

We define the notion of a tightly coupled test (definition 2). We also define the observable partial ordering (definition 6) of a set of CM-s, and prove (corollary 9) that when this ordering is of the right kind, and the test is viable and tightly coupled then the test is sound and the CM-s are faithful to the concurrent test. Motorola UK Research labs have implemented these results in their *ptk* tool [2], which generates conformance test suits from requirements specifications in the form of MSC-s.

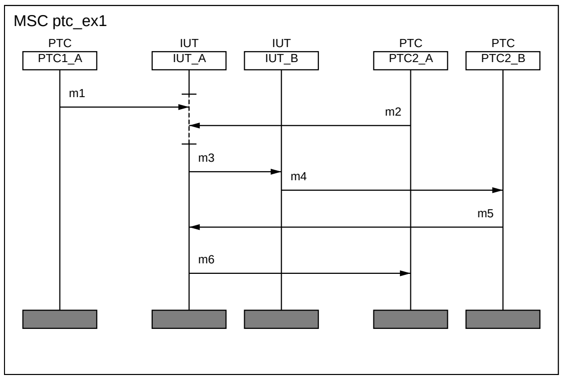
2 PCO-s and CP-s in TTCN

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. The MTC is in overall control and it is also responsible for storing the test verdict. This verdict can be pass, fail or inconclusive. The TTCN model supposes there are dedicated communication channels between the test components. Points of control and observation (PCO) are the channels for the PTC-s and MTC to interact with the IUT, the coordination points (CP-s) are channels between the PTC-s and MTC which are used to coordinate their activity via special messages known as coordinating messages (CM-s).



We will assume all concurrent tests are conformance tests derived from MSC-s which capture the requirements specifications. Each PTC and IUT may be defined as a collection of instances within the MSC. A concurrent test defines how the observed events which are either sent or received by the PTC-s should occur.

2.1 Examples



In this example we define PTC1 to consist of the instance PTC1_A, PTC2 to be instances PTC2_A and PTC2_B, and the IUT to be instances IUT_A and IUT _B. From now on we assume that any instance which is marked as a PTC is a separate PTC in its own right unless we state otherwise.

The MTC is not shown in this MSC since it is not used to explicitly send or receive any messages. Its only role is to act as a repository for the test verdict. We suppose this to be true for all the examples in the paper.

For a message m let $!m$ be the send event for this message and $?m$ be the receive event for the message.

Definition 1
Define a PTC event to be an event which is either sent by or received by a PTC.

The PTC events in this example are $!m1$, $!m2$, $?m4$, $!m5$, and $?m6$. The only PTC events which are not ordered in this example are $!m1$ and $!m2$. The dotted line on process IUT_A denotes a co-region. This is a region of the time line where the order of events can be arbitrary. A concurrent test specifies how the PTC events should be observed during the test, it also defines a set of CM-s which force the PTC-s to synchronise correctly.

For conformance testing we must show that no matter what order the events in the MSC occur, so long as that is in accordance with the partial ordering defined by the semantics of the MSC, the IUT performs correctly. Each particular interleaving of the PTC and IUT defines a particular conformance test.

The partial order semantics for this example define this partial order on the MSC events:

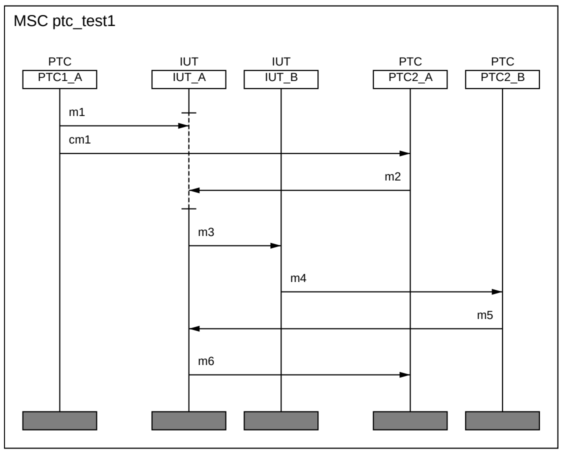
$$\begin{tikzcd}
 & !m1 \arrow[r, dash] & ?m1 \arrow[dr, dash] & & & & & & & & & & & \\
\text{Start} \arrow[ur, dash] \arrow[dr, dash] & & & !m3 \arrow[r, dash] & ?m3 \arrow[r, dash] & !m4 \arrow[r, dash] & ?m4 \arrow[r, dash] & !m5 \arrow[r, dash] & ?m5 \arrow[r, dash] & !m6 \arrow[r, dash] & ?m6 \arrow[r, dash] & \text{Stop} \\
 & !m2 \arrow[r, dash] & ?m2 \arrow[ur, dash] & & & & & & & & & &
\end{tikzcd}$$

Therefore the possible interleavings of the PTC events which may be observed in practise are:

$$\begin{array}{ll} (1) & !m1 < !m2 < ?m4 < !m5 < ?m6 \\ (2) & !m2 < !m1 < ?m4 < !m5 < ?m6 \end{array}$$

These two orderings only differ in the order in which $!m1$ and $!m2$ occur. Thus there are two conformance tests, one for each order.

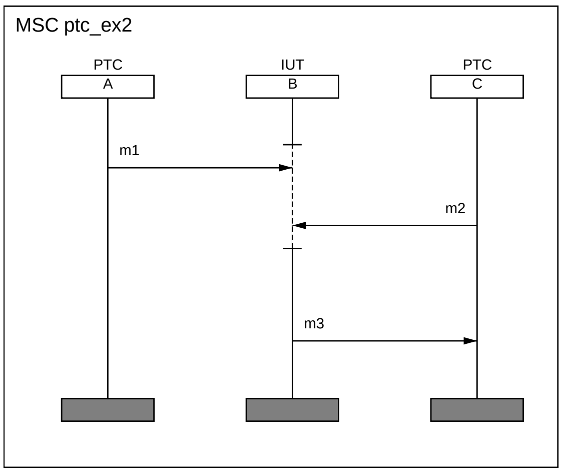
The following MSC depicts the first test where a new coordinating message $cm1$ is introduced to force $!m2$ to occur after $!m1$.



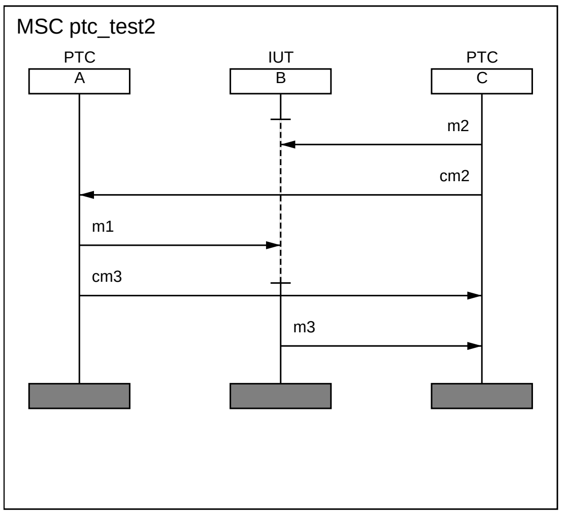
The test to force $!m1$ to occur after $!m2$ is constructed analogously.

There is one other function the PTC-s must perform. They must decide what test verdict to reach. To do this more CM-s may be necessary so that the PTC-s can decide if the IUT has responded correctly to messages from the PTC-s.

Consider a more simple example.



Again two tests are possible, consider the case where we choose to make $!m2$ occur before $!m1$. In this case we need one coordinating message to ensure that the send events occur in the correct order and another to ensure that instance C can decide if $?m3$ has occurred after $!m1$.

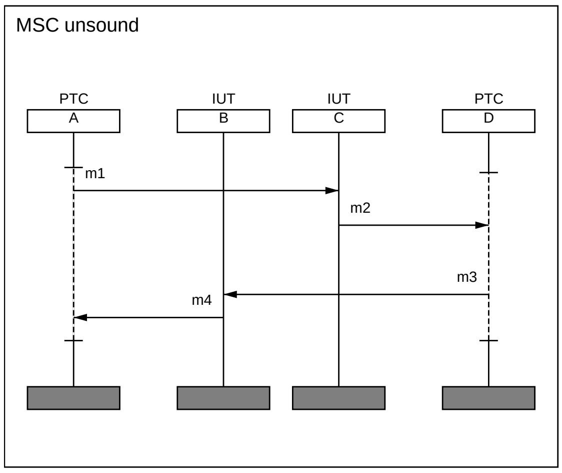


The purpose of $cm2$ is to force $!m1$ to be after $!m2$. The purpose of $cm3$ is to advise C that A is has sent $m1$ to the IUT. So long as $?cm3$ arrives before $?m3$ C can report that the test has passed.

One subtlety to notice is that just because C reports the PTC events have occurred in the right order does not mean that the IUT is acting correctly.

For example it is possible for B to send $m3$ as soon as it receives $m2$ but before it receives $m1$. This should result in a fail result. However it is possible that $m3$ travels so slowly compared to $cm3$, that C still receives $cm3$ before $m3$ and reports a pass. There is nothing that can be done to prevent this in practice, although repeating the test may be helpful.

This illustrates that a passed test does not necessarily reflect that the IUT is behaving correctly. A weaker property that we absolutely want to hold is that if the IUT does behave correctly then it will pass the test. This is the soundness property for a test. The next MSC although harmless in itself can lead to unsound conformance tests if we are not careful.



According to the MSC semantics one possible ordering for the PTC events is $!m1$ then $!m3$, and then $?m4$ and $?m2$ in either order. However no matter what coordinating messages we introduce between A and D we can never guarantee that in practise $?m2$ occurs after $!m3$. So no matter how we add coordinating messages the resultant concurrent test will always be unsound.

3 Tightly Coupled Partial Order

Intuitively we would like a test to define a total ordering on the set of PTC events. That is we would like it to define exactly how the PTC events will interleave during the test. For many MSC this is not possible. In the MSC unsound example above, we will never be able to force $?m2$ and $?m4$ to occur in any given order.

Clearly a test is not meant to be some arbitrary partial order of the PTC events and we can not force it to be a total order on the PTC events. What then do we mean by a test? We want to impose a restriction which is as near as possible to a total ordering on the PTC events without trying to impose absurd restrictions on the order of events. This is the intuition behind the idea of a tightly coupled partial order. This definition will remove some of the tests which clearly can lead to false results, but it will not exclude all unsound tests. For this we will need the property of a viable test which we introduce later. The two conditions together can be used to check if a given set of coordinating messages construct a sound test.

For this paper we define a partial order $<$ on a set X to be an irreflexive asymmetric transitive relation. That is

1. $\forall\ x \in X,\ (x \not< x)$
2. $\forall\ x, y \in X,\ (x < y) \Rightarrow (y \not< x)$
3. $\forall\ x, y, z \in X,\ (x < y)$ and $(y < z) \Rightarrow (x < z)$

When K is an instance, a set of instances, or an MSC M, let $E(K)$ be the set of events contained in K. Let $S(K)$ be the send events in K, and $R(K)$ be the set of receive events. For an MSC M let PTC be the set of instances which occur in the PTC-s. So, for example, $S(PTC)$ is the set of send events which occur on an instance for some PTC. Let $<_M$ denote the partial ordering which the MSC semantics of M impose on the events $E(M)$.

Definition 2
Let $<_T$ be a partial order defined on the events $E(PTC)$. Define $<_T$ to be a tightly coupled order if

1. $\forall\ x, y \in E(PTC)$ if $x <_M y$ then $x <_T y$,
   i.e. $<_T$ is consistent with $<_M$.
2. $<_T$ is a total ordering on the events $S(PTC)$
3. $\forall\ e \in S(PTC),\ \forall\ r \in R(PTC)$, either $e <_T r$ or $r <_T e$.

This is as close as we can get to a total ordering of $E(PTC)$ without trying to force two receive events to be ordered where it is not possible for this to happen in practise. It may still be that two particular receive events r_1 and r_2 are ordered by $<_T$, since it is a transitive ordering. For example if we have $r_1 <_T e$, and $e <_T r_2$ for some $e \in S(PTC)$, then $r_1 <_T r_2$ by transitivity.

Although a tightly coupled test does not contain absurd restrictions on the PTC event ordering it is not necessarily sound. For example the test we proposed for the MSC unsound example is tightly coupled. It is not total since we allow $?m2$ and $?m4$ to occur in either order. As we saw that test is unsound.

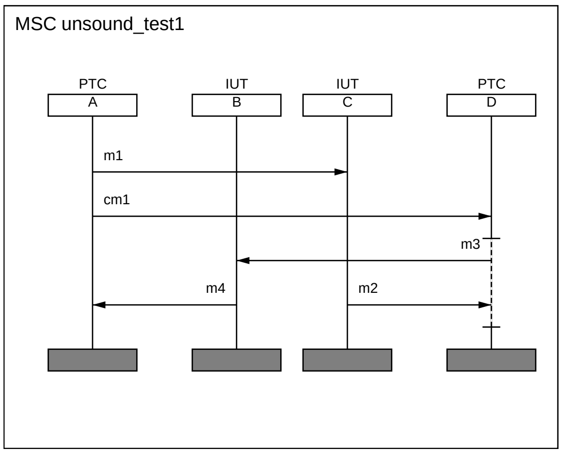
From now on we only consider tests which define tightly coupled partial orders on the events in $E(PTC)$.

Definition 3
Let T be a test on the MSC M, and let $<_T$ be the partial order which T defines on the events $E(PTC)$. When $<_T$ is a tightly coupled partial order define T to be a tightly coupled test.

4 Coordinating Messages

Let T_u be the unsound test we proposed for the MSC unsound example. Although T_u is tightly coupled it failed to be sound because it ordered $!m3 <_{T_u} ?m2$. If we had wanted to construct a concurrent test from this partial order we would have needed to introduce additional CM-s which imposed this order on $E(PTC)$.

The nearest we can get to imposing the $<_{T_u}$ order is with an additional CM $cm1$ added to the MSC, as dipicted in the MSC unsound_test1 below.



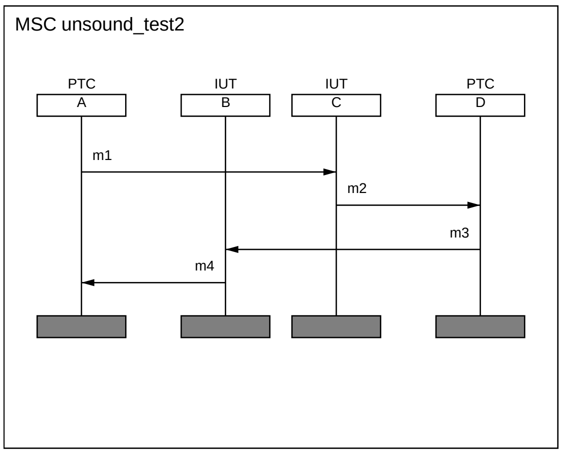
First note that the ordering of events in the PTC is still tightly coupled. With this coordinating message we have forced $!m2$ to be earlier than $!m3$. Also it is clear that there is no other way we can force $!m2 <_T !m3$ to occur, so such a CM is the minimum necessary additional structure we can add.

With the addition of $cm1$ it is clear that we can never force $?m2$ to be ordered relative to $?m3$. The reason is that we are trying to force a send event to occur before a receive event when there is no linkage between the events within the MSC.

Consider a second test on the MSC unsound example. This time we want to construct a concurrent test which imposes the order

$$!m1 <?m2 <!m3 <?m4$$

This is a tightly coupled partial order since it is total. In this case we have a sound test which is represented by this MSC:



In this case we do not need any coordinating messages to force this order to occur. The test is achieved by PTC D waiting for $?m2$ before sending $!m3$. That is all that is required to enforce the order. In this case the test is sound because we force a send event to wait until a receive event had occurred, in the unsound case we were trying to force a receive event to wait until after a send event.

The intuition behind these examples leads us to define a viable test.

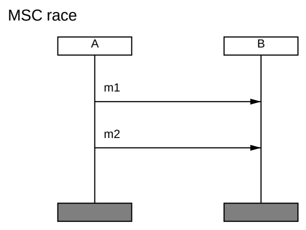
Definition 4
Let $<_T$ be the partial order defined on $E(PTC)$ by a test T. We define T to be viable if

$$\forall e \in S(PTC), \forall r \in R(PTC)\,(e <_T r \Rightarrow (\exists e' \in S(PTC)\ : e \leq_T e' <_M r))$$

That is, a viable test only orders a send event e to occur before a receive event r, if this is a direct consequence of ordering e before some send event e' which by the structure of M occurs before r.

It will always be the case that we can force two send events to be ordered the way we want them for a particular test, so long as the MSC M is race free. Thus we can safely make e occur before r by finding some other send event $e' <_M r$ and forcing e' to wait until after e.

We say an MSC contains a race if it defines an ordering of events which can not be guaranteed to occur in practise. Here is a simple example of a race.



In this example the MSC specifies that $?m1 <_M ?m2$. However, the standard MSC semantics state that the latency in a message is arbitrary. Therefore the latency of $m1$ could be larger than the latency of $m2$. If this difference is large enough then it will be perfectly possible for $?m2$ to occur before $?m1$. Moreover in a test system where data may be traveling over an IP network the latency may vary every time the message is sent. In such circumstances it will not be possible to know the relative latency of $m1$ and $m2$ and it will be impossible to impose the specified MSC ordering.

This example is by no means the only way a race can occur in an MSC, for a full analysis of racing events in an MSC see [5].

Even with a test which is viable (but not necessarily tightly coupled) we can say that a test which is not viable is not sound.

Theorem 5
Suppose that M is a race free MSC, and T is a concurrent test on M which is not viable, then T is unsound.

Proof Since the test is unsound there is some $e \in \mathrm{S(PTC)}$ and some $r \in \mathrm{R(PTC)}$ such that $e <_T r$, and there is no event $e' \in \mathrm{S(PTC)}$ such that $e <_T e' <_M r$.

Therefore for every event $e' \in \mathrm{S(PTC)}$ such that $e' <_M r$ it must be that $e' \not<_T e$. Test T must be a total ordering on S(PTC), therefore for every $e' \in \mathrm{S(PTC)}$ we have $e' <_M r \Rightarrow e' <_T e$.

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. $\square$

We now consider what it is that a concurrent test actually tests. For this we need to define the idea of the observable ordering for the PTC-s in the test. Given this we can state our main theorem which describes when a set of coordinating messages create a sound test which faithfully verify the test.

5 The Observable Ordering of a Concurrent Test

We have defined a concurrent test in terms of the partial order it defines over the events in the PTC-s. Implicitly the ordering $<_M$ is contained in any test in so far as it orders elements of $E(PTC)$, since the test must be consistent with $<_M$. However the orderings $<_T$, and $<_M$ are transitive, and so much of the ordering structure is determined by how events within the IUT are related to events on the PTC-s.

A concurrent test uses coordinating messages to ensure that the correlation between the IUT events and the PTC events can be verified by an observer. The coordinating messages define a second partial order which determines how the events external to the IUT are ordered without reference to the internal structure of the IUT. We call this the observable ordering of the test. Essentially the observable ordering is the order which would be given by the MSC if we delete the instances in the IUT and all the events they contain.

It is the observable ordering that any given test run actually measures. That is a test run will report a pass exactly when the observable ordering occurs. This does not necessarily mean the test ordering has occurred. Therefore if the coordinating messages are not set up correctly a test can report a false pass verdict.

For a binary relation $\mathcal{R}$ let $\mathcal{R}^*$ denote the transitive closure of the relation.

Definition 6
For a set of instances K, which define a PTC, let $<_K$ be the partial order of events in K given by restricting $<_M$ to K. Let $\mathcal{C}$ be a binary relation defined on $E(PTC)$ by $x\mathcal{C}y$ if and only if $x \in S(PTC)$, $y \in R(PTC)$ and x and y are part of the same message (which may be a CM or an ordinary message). Define the binary relation $\mathcal{P}$ by $x\mathcal{P}y$ if x and y belong to some PTC, K, and $x <_K y$.

Define the observable partial ordering $<_{PTC}$ for a concurrent test T by

$$<_{PTC} = \Big(\mathcal{P} \cup \mathcal{C}\Big)^*$$

Return now to the MSC unsound_test2. This characterised a conformance test for the MSC unsound example. In this case there are no CM-s but we can still examine the observable ordering and compare it with the test ordering. Rather surprisingly the observable ordering is completely defined by these two inequalities

$$(!m1) <_{PTC} (?m4), \ (?m2) <_{PTC} (!m3)$$

This is not the same as the test ordering, which in particular states that $!m1 <_T ?m2$, which most certainly does not hold in $<_{PTC}$.

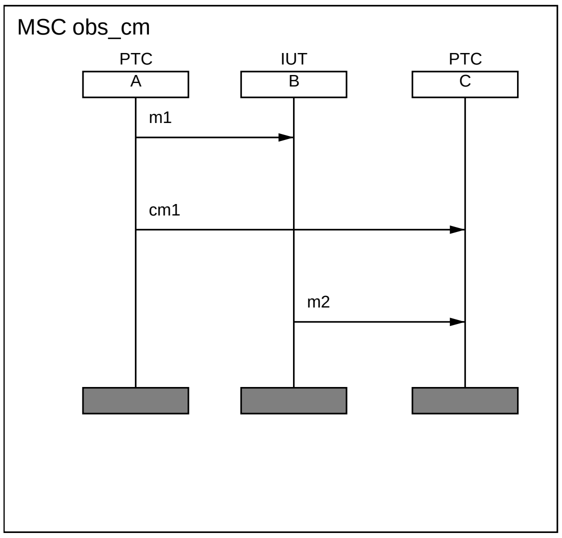
Theorem 7
Any test run of a concurrent test can only verify that the observable partial ordering occurs.

Since the observable partial ordering may not be the same as the test partial ordering, it is not the case that a given concurrent test script actually verifies that the test ordering has occurred correctly.

For a concurrent test to be faithful we need to know that the observed ordering is the same as the test ordering. Before we can prove any results along these lines we need to make an assumption about the latency of a coordinating message.

Latency Assumption We make the assumption that the latency in each CP channel is less than the latency in any of the PCO channels.

This constraint is fairly reasonable. It means that the test system must be constructed so that CP-s are fast compared to CPO-s. To understand why we need this restriction consider the following MSC.



The coordinating message here ensures that the observable ordering is the same as the test ordering, provided that the above latency assumption holds. For the observation ordering to be faithful to the test ordering we have to be able to verify that $!m1 <?m2$. If the latency in $cm1$ is less than the latency in $m2$ and $?m1 <!m2$ then we will have that $?cm1 <?m2$. In this case the only way that $?m2$ could be before $?cm1$ is if $!m2$ is before $?m1$.

Phased Coordinating Messages Let T be a concurrent test with observable partial order $<_{PTC}$. Suppose that the CM-s associated with T have been added to the MSC M. We say the CM-s in T are *in phase* when

1. $<_{PTC}$ is a total order on $S(PTC)$
2. when $e \in S(PTC)$, $r \in R(PTC)$ and $e <_M r$ then $e <_{PTC} r$
3. when $e \in S(PTC)$, $r \in R(PTC)$ and $r <_T e$ then $r <_{PTC} e$

This looks very like the definition of tightly coupled but is significantly different in parts 2 and 3. In the second part we only try to force a send event to occur before a receive event when this is part of the MSC structure. Provided that the MSC is race free it will be possible to introduce suitable CM-s to achieve this purpose. Part three means we may have some CM-s present which cause a send event to wait until after a receive event has occurred. This will always be possible and so does not cause a problem in practise.

Armed with the definitions we have we can state our main theorem for testing if a concurrent test is sound and faithful.

Theorem 8
Let M be a race free MSC, let T be a concurrent test where the CM-s are in phase, and let T be tightly coupled and viable. Then $<_{PTC}$ is tightly coupled and viable.

Proof Since the coordinating messages are in phase we have that $S(PTC)$ is totally ordered by $<_{PTC}$. Thus to demonstrate that $<_{PTC}$ is tightly coupled we only need concern ourselves with $e \in S(PTC)$ and $r \in R(PTC)$.

– If $e <_M r$ or $r <_M e$ it follows from the definition of $<_{PTC}$ that $e <_{PTC} r$ or $r <_{PTC} e$.
– If $r <_T e$ then again it follows from the definition of $<_{PTC}$ that $r <_{PTC} e$.
– Finally consider the case where $e <_T r$. Since M is race free the only events which can occur immediately before r must be send events $e' \in$ S(PTC) and $e' <_M r$. Since $<_T$ is viable there must be at least one such event.
  Events e and e' are ordered by $<_{PTC}$. Since we know the coordinating messages are in phase we know $S(PTC)$ is totally ordered by $<_{PTC}$.
  If $e <_T e'$ then also $e <_{PTC} e'$ and so $e <_{PTC} e' <_M r$.
  If $e' <_T e$ for every such e' we contradict that $<_T$ is viable, because this implies there is not $e'' \in S(PTC)$ where $e <_T e'' <_M r$.
  Hence there is at least one e' where $e <_{PTC} e' <_M r$, and therefore $e <_{PTC} r$. This completes the proof that $<_{PTC}$ is tightly coupled.

Note we have also proved that $<_{PTC}$ is viable at the same time. This completes the proof. □

Corollary 9
Let M be a race free MSC, let T be a concurrent test where the CM-s are in phase, and let T be tightly coupled and viable. Then $<_{PTC}$ is sound and faithful to T.

References

1. R. Alur and M. Yannakakis, Model checking of message sequence charts, Proceedings of the Tenth International Conference on Concurrency Theory, Springer Verlag, 1999.
2. P. Baker, P. Bristow, C. Jervis, D. King, B. Mitchell, Automatic Generation of Conformance Tests From Message Sequence Charts, Proceedings of 3rd SAM Workshop 2002, Telecommunications and Beyond: The Broader Applicability of MSC and SDL, pp 170-198, LNCS 2599.

3. Jens Grabowski, D. Hogrefe, R. Nahm, Test Case Generation with Test Purpose Specification by MSC-s, SDL'93 - Using Objects, Ed. O Faergemand, A. Sarma, North Holland, October 1993.
4. 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
5. Gerard J. Holzmann, Doron A. Peled, and Margaret H. Redberg, Design Tools for Requirements Engineering, Bell Labs Technical Journal, Winter 1997
6. ISO Information Technology - OSI-Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation (TTCN) ISO IS 9649-3, 1996
7. Stefan Leue, Lars Mehrmann, Mohammad Rezai, Synthesizing ROOM Models from Message Sequence Chart Specifications, http://fee.uwaterloo.ca/ sleue/msc.html#tr98-06
8. 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
9. Z.120 (11/99)ITU-T Recommendation - Message Sequence Chart (MSC) http://www.itu.int/itudoc/itu-t/approved/z/index.html