Chapter

9

Tutorial: The Autolink Tool

This tutorial is intended to guide you through some of the features of Autolink. There are some aspects of Autolink which are not covered by this tutorial. For a complete description of all Autolink facilities, please see "Using Autolink" on page 1393 in chapter 36, *TTCN Test Suite Generation, in the User's Manual*.

To be able to follow the tutorial, some experience of the SDL Validator is needed. Therefore you should have read and practised *chapter 5, Tutorial: The SDL Validator, in the SDL Suite Getting Started*. You also need to have basic knowledge of the TTCN suite.

The example protocol used in this tutorial is the `inres` system, the same as the one used in the TTCN Link tutorial.

Purpose of This Tutorial

The purpose of this tutorial is to make you familiar with Autolink. The tutorial is split into three parts. You are strongly recommended to read the tutorial sequentially as each exercise is founded on the previous one.

In the first exercise, you will start by creating a simple MSC. Based on this MSC, you will generate a TTCN test case and save it in a test suite. You will also learn how to manipulate constraints.

In the second session, you will develop a structured MSC. Once again a TTCN test case is generated, but this time by a direct MSC into TTCN translation. Additionally, so-called translation rules are introduced as a simple example for an Autolink configuration.

In the third part, you will use the Tree Walk algorithm to create a large number of MSC test cases automatically. You will also learn how to save and load generated test cases in Autolink. This facility will allow you to compute test cases separately and distribute test generation over several computers (provided that you have more than one license).

It is assumed that you know how to generate an SDL Validator. In addition, you should be familiar with the basic functionality of the SDL Validator, the MSC Editor and the TTCN suite. You may also find it useful to have knowledge of MSC diagrams.

Note: Platform differences

This tutorial is possible to run on both the UNIX and Windows platform, and is described in a way common to both platforms. In case there are differences between the platforms, this is indicated by texts like “on UNIX”, “Windows only”, etc.

When such platform indicators are found, please pay attention only to the instructions for the platform you are running on.

Basics of Autolink

Autolink assists you in the automatic generation of TTCN test suites based on an SDL specification. In a simple approach, a TTCN test suite is generated in three steps:

1. One or more *system level Message Sequence Charts* have to be defined. System level MSCs contain the externally visible events of a path. These events describe the correct test sequences of a TTCN test case.
2. The MSCs defined in step 1 are used to generate internal test case representations.
3. All internal test case representations are saved as a test suite in a file in TTCN-MP format.

Constraints can be added, modified and deleted between any of the above steps.

Getting Ready to Use Autolink

In order to use Autolink, an SDL Validator application has to be generated and started. In this exercise, you will use the `inres` system, which is found in `$telelogic/sdt/examples/inres`.

"Inres" is short for Initiator-Responder. The `inres` system is an example of a simplified communication protocol intended to give a secure transfer of information over an unsafe communication medium. It provides a one-way, connection-oriented communication service that uses sequence numbers and retransmission to ensure that messages sent from the initiator side are correctly received at the responder side.

Note:

In order to generate a validator application that behaves as stated in the exercises, you should copy all files of the `inres` protocol from the release to your working directory.

You also have to create two directories where MSC test steps and test cases will be stored.

1. In your current working directory, create two subdirectories called `TC` (used for test cases) and `TS` (used for test steps).
2. Start Telelogic Tau and open the `inres` system in the Organizer.
3. Generate an `inres` validator and open it in the Validator UI.

Exercise 1: Basic Concepts

What You Will Learn

- To set up value definitions
- To create an MSC test case definition
- To generate a test case
- To work with constraints

Preparations

First, you should define the location of the test cases and test steps directories which you have created prior to opening the validator:

1. Select *Autolink: Test Cases Directory* in the *Options2* menu.

   The *Directory name* dialog is displayed.

2. Double-click the `TC` directory.

3. Click the *OK* button.

4. Select *Autolink: Test Steps Directory* in the *Options2* menu.

   Once again, the *Directory name* dialog is displayed.

5. Choose the `TS` directory by first clicking the *Current* button and then double-clicking the `TS` directory.

6. Click the *OK* button.

> **Note:**
> The directory settings can be saved when you leave the validator.

You should also set up proper test value definitions for the `inres` system:

1. Open the *TEST VALUES* module by clicking on the box beside.

   A number of new command buttons appear.

2. Click the *Clear Value* button in the *TEST VALUES* module to clear all integer test values.

   A *Select* dialog is displayed.

3. Choose `integer`.
4. Click the *OK* button.

   A *Prompt* dialog is displayed.
5. Click the *OK* button, without typing anything in the text field.
6. Click the *Def Value* button in the *TEST VALUES* group to define a new test value for integers.

   Another *Select* dialog is displayed.
7. Choose `integer`.
8. Click the *OK* button.

   A *Prompt* dialog is displayed.
9. Type `55` as the new test value in the *Value* field.
10. Click the *OK* button.

Creating an MSC Test Case

The first step in the test case generation process is the choice of a path. A path describes a trace through the SDL system. This trace consists of internal and external events. External events are signals sent to or from the environment.

Autolink uses system level MSCs to abstract from a path by only considering the external events. System level MSCs consist of exactly one instance axis for the SDL system and one or more instance axes for the system environment. Figure 71 shows the MSC which you are going to create in this exercise.

1. Click the *Navigator* button in the *EXPLORE* module to start the Navigator.

   The *Navigator* window is displayed.

2. Navigate through the system by double-clicking in turn on the following nodes:

   1, 1, 1, 1, 1, 1, 1, 1 (i.e. 8 times transition #1),
   3,
   1, 1, 1, 1, 1, 1, 1 (i.e. 7 times transition #1),
   10,
   1, 1, 1 (i.e. 3 times transition #1),
   2,
   1, 1, 1 (i.e. 3 times transition #1)

3. Select *MSC*: *Save Test Case* in the *Autolink1* menu to save the current path as a system level MSC.

   A *Prompt* dialog is displayed.

4. Type **`Example1`** in the *Test case name* field.

   The test case name is identical to the name of the generated MSC.

5. Click the *OK* button.

   Autolink saves a file called `Example1.mpr` in the MSC test cases directory.

You have now created your first MSC test case (see Figure 71). You may start the MSC Editor and take a look at it.

Note:

You can create as many MSC test cases as you like, provided that they share the same root node.

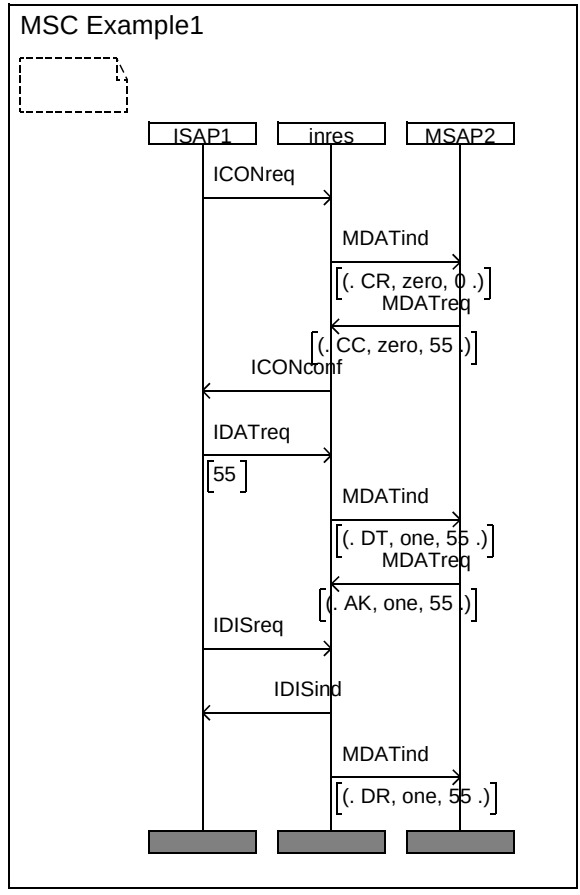

Figure 71: MSC Example1

It is possible to get a list of all MSC test cases (and test steps) which are currently defined:

- Select *MSC: List* in the *Autolink1* menu.

  A message similar to the one below should be given in the text area.

```
MSC test cases:
Example1.mpr

No MSC test steps found in directory
'/home/tutorial/inres/TS/'.
```

Generating the Test Case

In the next step, an internal test case representation has to be created which is based on the MSC test case defined above.

1. Select *Test Case: Generate* in the *Autolink1* menu.

   A *Select* dialog is displayed.

2. Select `Example1.mpr`.

3. Click the *OK* button.

   Autolink performs a bit state exploration, which produces the following output in the text area of the Validator UI:

```
Autolink state space options are set.

Define-Scheduling All
Define-Transition Symbol-Sequence
Define-Symbol-Time Zero
Define-Priorities 2 1 3 2 2
Define-Channel-Queue ISAP1 On
Define-Channel-Queue MSAP2 On
Define-Max-Input-Port-Length 10

Current state is reset to root.

MSC 'Example1' loaded.

** Test case generation statistics **
No of reports: 2.
  Deadlock         : 1 report
  MSCVerification  : 1 report
Generated states: 2089.
Truncated paths: 0.
Unique system states: 790.
Size of hash table: 8000000 (1000000 bytes)
No of bits set in hash table: 1411
```

```
Collision risk: 0 %
Max depth: 49
Current depth: -1
Min state size: 388
Max state size: 524
Exploration started at: Sat Jan 29 13:39:14 2000
Exploration ended at:   Sat Jan 29 13:39:15 2000
Exploration time: 000 h, 00 m, 01 s

Constraints are checked for naming conflicts...

Constraint 'cExample1' is renamed to
'cExample1_001'.
Constraint 'cExample1' is renamed to
'cExample1_002'.
Constraint 'cExample1' is renamed to and merged with
'cExample1_002'.
Constraint 'cExample1' is renamed to and merged with
'cExample1_001'.
Constraint 'cExample1' is renamed to
'cExample1_003'.
Constraint 'cExample1' is renamed to
'cExample1_004'.
Constraint 'cExample1' is renamed to
'cExample1_005'.
Constraint 'cExample1' is renamed to
'cExample1_006'.
Constraint 'cExample1' is renamed to
'cExample1_007'.
Constraint 'cExample1' is renamed to
'cExample1_008'.
Constraint 'cExample1' is renamed to
'cExample1_009'.
Constraint 'cExample1' is renamed to
'cExample1_010'.

Constraints with identical signal definitions are
merged automatically...

... done.

State space options are restored.
```

You have now generated your first test case. This test case is kept in memory. You will save it in a TTCN test suite soon.

It is possible to get a list of all test cases which have been generated so far.

- Select *Test Case: List* in the *Autolink1* menu.

  The output will simply be:

```
Example1
```

You might also want to take a look at the internal test case representation before you create the TTCN test suite.

1. Select *Test Case: Print* in the *Autolink1* menu.

   A *Select* dialog is displayed.

2. Choose `Example1`.

3. Click the *OK* button.

   The resulting output is:

```
Example1

  0     ISAP1 ! ICONreq - cExample1_010
  1      MSAP2 ? MDATind - cExample1_009
  2       MSAP2 ! MDATreq - cExample1_008
  3        ISAP1 ? ICONconf - cExample1_007
  4         ISAP1 ! IDATreq - cExample1_006
  5          ISAP1 ! IDISreq - cExample1_005
  6           MSAP2 ? MDATind - cExample1_004
  7            MSAP2 ! MDATreq - cExample1_003
  8             ISAP1 ? IDISind - cExample1_001
  9  P           MSAP2 ? MDATind - cExample1_002
  8             MSAP2 ? MDATind - cExample1_002
  9  P           ISAP1 ? IDISind - cExample1_001
```

Modifying the Constraints

During test case generation, a number of constraints have been created. These constraints are also stored in memory and can be manipulated in several ways.

Listing the Constraints

First of all, it is possible to print the list of all constraints:

- Select *Constraint: List* in the *Autolink2* menu.

  This should be the output in the text area:

```
cExample1_001 :
  IDISind {  }
cExample1_002 :
  MDATind { mSDUType1 { id DR, num one, data 55 } }
cExample1_003 :
  MDATreq { mSDUType1 { id AK, num one, data 55 } }
cExample1_004 :
  MDATind { mSDUType1 { id DT, num one, data 55 } }
```

```
cExample1_005 :
  IDISreq {  }
cExample1_006 :
  IDATreq { iSDUType1 55 }
cExample1_007 :
  ICONconf {  }
cExample1_008 :
  MDATreq { mSDUType1 { id CC, num zero, data 55 } }
cExample1_009 :
  MDATind { mSDUType1 { id CR, num zero, data 0 } }
cExample1_010 :
  ICONreq {  }
```

As you can see, 10 constraints have been created during test generation.

Adding a Constraint

Next, you can add a new constraint:

1. Select *Constraint: Define* in the *Autolink2* menu.

   A *Prompt* dialog is displayed.

2. Type **`RequestCon`** in the *Constraint name* field.

3. Click the *OK* button.

   A *Select* dialog is displayed, listing all signals in the system.

4. Select `ICONreq`.

5. Click the *OK* button.

   In the text area of the Validator user interface, the following message appears:

   ```
   Constraints are checked for naming conflicts...
   ```

   Of course, there is no naming conflict, since there does not exist another constraint with the same name.

Renaming a Constraint

You might want to assign a more reasonable name to constraint `cExample1_007` containing the signal `ICONconf`:

1. Select *Constraint: Rename* in the *Autolink2* menu.

   A *Select* dialog is displayed, listing all constraints currently defined.

2. Choose `cExample1_007`.

3. Click the *OK* button.

   A *Prompt* dialog is displayed.

4. Enter **`ConfirmCon`** in the *New constraint name* field.

5. Click the *OK* button.

Parameterize a Constraint

Constraint `cExample1_006` contains a signal parameter (value `55`). You can define this signal parameter to be a parameter of the constraint.

1. Select *Constraint: Parameterize* in the *Autolink2* menu.

   A *Select* dialog is displayed.

2. Select `cExample1_006`.

3. Click the *OK* button.

   A *Select* dialog is displayed.

4. Select `1` for the first signal parameter.

5. Click the *OK* button.

   A *Prompt* dialog is displayed.

6. Enter **`Data`** in the *Formal parameter* field.

7. Click the *OK* button.

   The *Select* dialog is displayed again.

8. This time, select `0` in order to finish parameterization.

9. Click the *OK* button.

Merging Constraints

The constraints `cExample1_002` and `cExample1_004` only differ in the first signal parameter (which is a struct). Therefore, you might want to merge them.

1. Select *Constraint: Merge* in the *Autolink2* menu.

   A *Select* dialog is displayed, listing all currently defined constraints.

2. Choose `cExample1_002` as the first constraint.

3. Click the *OK* button.

   Another *Select* dialog is displayed.

4. Choose `cExample1_004` as the second constraint.

   A *Prompt* dialog is displayed.

5. Enter **`ProtDataUnit`** in the *Formal parameter name* field.

6. Click the *OK* button.

Listing the Constraints – A Second Time

To see the effect of all your operations, you can list the constraints again:

1. Select *Constraint: List* in the *Autolink2* menu.

   This time, the following output should be displayed in the text area:

```
ConfirmCon :
  ICONconf {  }
RequestCon :
  ICONreq {  }
cExample1_001 :
  IDISind {  }
cExample1_003 :
  MDATreq { mSDUType1 { id AK, num one, data 55 } }
cExample1_004(ProtDataUnit) :
  MDATind { mSDUType1 ProtDataUnit }
cExample1_005 :
  IDISreq {  }
cExample1_006(Data) :
  IDATreq { iSDUType1 Data }
cExample1_008 :
  MDATreq { mSDUType1 { id CC, num zero, data 55 } }
cExample1_009 :
  MDATind { mSDUType1 { id CR, num zero, data 0 } }
cExample1_010 :
  ICONreq {  }
```

In the test case, all references to the constraints have been updated appropriately:

2. Select *Test Case: Print* in the *Autolink1* menu.

   A *Select* dialog is displayed.

3. Select `Example1`.

4. Click the *OK* button.

   The output is:

```
Example1

  0      ISAP1 ! ICONreq - cExample1_010
  1       MSAP2 ? MDATind - cExample1_009
  2        MSAP2 ! MDATreq - cExample1_008
  3         ISAP1 ? ICONconf - ConfirmCon
  4          ISAP1 ! IDATreq - cExample1_006(55)
  5           ISAP1 ! IDISreq - cExample1_005
  6            MSAP2 ? MDATind - cExample1_004
                    ({ id DT, num one, data 55 })
  7             MSAP2 ! MDATreq - cExample1_003
  8              ISAP1 ? IDISind - cExample1_001
  9  P            MSAP2 ? MDATind - cExample1_004
                     ({ id DR, num one, data 55 })
  8              MSAP2 ? MDATind - cExample1_004
                     ({ id DR, num one, data 55 })
  9  P            ISAP1 ? IDISind - cExample1_001
```

Saving the Constraints

Since all constraints are deleted if they are no longer referred to by any generated test case or if you leave the Validator, they can be saved in a file:

1. Select *Constraint: Save* in the *Autolink2* menu.

   A *Select* dialog is displayed.

2. Click the *OK* button without entering any text before.

   By this you indicate that you want to store all constraints.

   The *File name* dialog is displayed.

3. Type **`Example1.con`** in the *File* field.

4. Click the *OK* button.

The constraint definitions are saved. They can be restored by the *Constraint: Load* command in the *Autolink2* menu but you should not do that at the moment.

Removing a Constraint

A constraint can be removed:

1. Select *Constraint: Clear* in the *Autolink2* menu.

   A *Select* dialog is displayed.

2. Select `cExample1_003`.

3. Click the *OK* button.

   Since this constraint is currently used in a test case, an error message is displayed in the text area:

   ```
   Error clearing constraint 'cExample1_003'.
   The constraint is currently used in a test case.
   ```

4. Once again, select *Constraint: Clear* in the *Autolink2* menu.

   The same *Select* dialog as above is displayed.

5. Select `RequestCon`.

6. Click the *OK* button.

   This time the specified constraint is removed, since it is not referred to in any test case.

Saving the TTCN Test Suite

In a final step, all generated test cases can be saved in a TTCN test suite. In this tutorial there is only one test case, but it is also possible to put several test cases into one test suite.

1. Select *Test Suite: Save* in the *Autolink1* menu.

   A *Prompt* dialog is displayed.

2. Type **`inres`** in the *Test suite name* field.

3. Click the *OK* button.

   A *File name* dialog is displayed.

4. Type **inres.mp** in the *File* field.
5. Click the *OK* button.

   Autolink saves the test suite in file `inres.mp`.

   During saving, Autolink performs a few checks to ensure the consistent use of various test suite elements. You can ignore the message about test steps, since your test case does not have any test steps at all.

You can exit the Validator now:

1. Select *Exit* in the *File* menu.

   A *Select* dialog is displayed.

2. Select `Yes`.
3. Click the *OK* button.

   The current options are saved. You will need them in the next exercise.

Viewing the TTCN Test Suite

Up to now, you have created a TTCN test suite. This test suite is stored in MP format (*m*achine *p*rocessable) which is somewhat hard to read. Therefore you should add it to the Organizer and open it in the TTCN suite.

1. In the Organizer, select the *TTCN Test Specification* chapter.
2. Select *Add Existing* in the *Edit* menu.

   A corresponding dialog is displayed.

3. Change the filter from "`*.txt`" to "`*.mp`" and click the *Filter* button.

   Provided that you are in the right directory, the file `inres.mp` is shown in the *Files* box.

4. Select the file `inres.mp` in the *Files* section.
5. Click the *OK* button.

The file is added to the Organizer and the TTCN suite is opened with the test suite.

Completing the Test Suite

The TTCN test suite you have created so far contains information in three parts: the declarations part, the constraints part and the dynamic part.

In Windows, the test suite overview part will be generated automatically for example when you open one of the overview tables or before you print the test suite. After that, it will be kept updated.

On UNIX, you have to generate and update the overview explicitly:

1. Select *Generate Overview* in the *Tools* menu in the Browser.

   A dialog windows appears.

2. Click the *Generate* button.

   The test case index is generated.

Exercise 2: Advanced Concepts

What You Will Learn

- To create a structured MSC test case
- To define a simple Autolink configuration
- To generate a structured TTCN test case by a direct translation
- To merge new test cases with an existing test suite

Preparations

If you have closed the SDL suite or the Validator UI temporarily, perform the following steps:

1. Open the `inres` system in the Organizer.
2. Start the Validator UI.
3. Open the `inres` validator again.

If you have **not** quit the `inres` validator, bring the application to its initial state now:

- Select *Reset* in the *Options1* menu.

Creating a Structured MSC Test Case

For a better understanding, a test case can be structured into different test steps. Typically, a test case consists of three parts:

- The *preamble* comprises a sequence of test events that drives the system from the stable start state to the initial state from which the test body starts.
- The *test body* comprises a sequence of test events that achieve the test purpose.
- The *postamble* comprises a sequence of test events that drives the system from the state reached by the test body to a stable end state.

In this section, you will create a test case with a preamble, a test body and a postamble. This test case will be identical to the one developed in "Exercise 1: Basic Concepts" on page 154 except for its structural information.

Creating the Preamble

You can define structured MSC test cases in a similar way as you did in "Exercise 1: Basic Concepts" on page 154.

1. Click the *Navigator* button in the *EXPLORE* module to start the Navigator.

   The *Navigator* window is displayed.

2. Navigate through the system by double-clicking in turn on the following nodes:

   1, 1, 1, 1, 1, 1, 1, 1 (i.e. 8 times transition #1),
   3,
   1, 1, 1 (i.e. 3 times transition #1)

   Do not close the Navigator yet.

3. Select *MSC: Save Test Step* in the *Autolink1* menu to save the current path as an MSC test step.

   A prompt dialog is displayed.

4. Type **`Preamble`** in the *Test step name* field.

5. Click the *OK* button.

   Autolink saves a file called `Preamble.mpr` in the MSC test steps directory.

The preamble which you have just created is shown in Figure 73 on page 171.

Creating the Test Body

You should also create the body of the MSC test case:

1. Type **`define-root current`** in the input line to set the current root.

   The following message appears in the text area:

   ```
   Root of behaviour tree set to current system state
   ```

2. Double-click the following transitions in the *Navigator* window:

   1, 1, 1, 1 (i.e. 4 times transition #1),
   10,
   1, 1, 1 (i.e. 3 times transition #1).

   Again, do not close the Navigator.

3. Select *MSC: Save Test Case* in the *Autolink1* menu to save the path as an MSC test case.

   Make sure that you save the path as a test case, not as test step!

   A *Prompt* dialog is displayed.

4. Type **`Example2`** in the *Test case name* field.

5. Click the *OK* button.

   Autolink saves a file called `Example2.mpr` in the MSC test cases directory.

Now you have defined the body of the test case. See Figure 72.

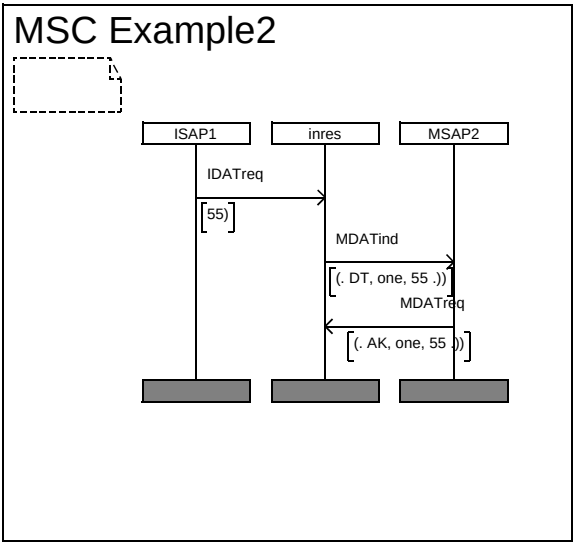

Figure 72: The MSC Example2 – test body

Creating the Postamble

Finally, you create the postamble of the MSC.

1. Again, type **`define-root current`** in the input line to set the root.

   The following message appears in the text area:

   ```
   Root of behaviour tree set to current system state
   ```

2. Navigate through the system by double-clicking in turn on the following nodes:

   2,
   1, 1, 1.

3. Select *MSC: Save Test Step* in the *Autolink1* menu.

   A *Prompt* dialog is displayed.

4. Type **`Postamble`** in the *Test step name* field.

5. Click the *OK* button.

   Autolink saves a file called `Postamble.mpr` in the MSC test steps directory.

You have now created the postamble of the test case, see Figure 73.

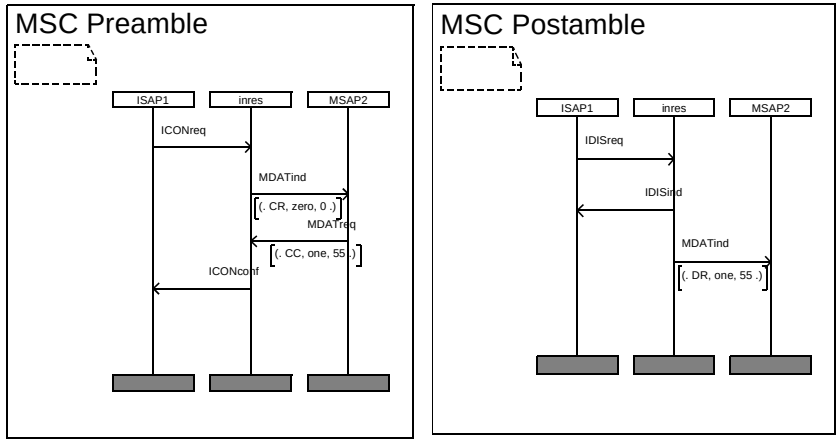

Figure 73: Preamble and Postamble MSCs

Inserting MSC References

Before you can use MSC *Example2* to generate a TTCN test case, you have to insert two global MSC references for the preamble and the postamble. To do this:

1. Select *Toggle MSC Trace* in the *Commands* menu.

   The MSC Editor will start, showing a complete Validator Trace.

2. Open MSC `Example2.mpr` stored in directory `TC`.

3. Make the MSC look like Figure 74, that is:
   - Insert two global MSC references, one above the first message and one after the last message.
   - Type **`Preamble`** and **`Postamble`** (**without** the file extension `.mpr`) as reference names.

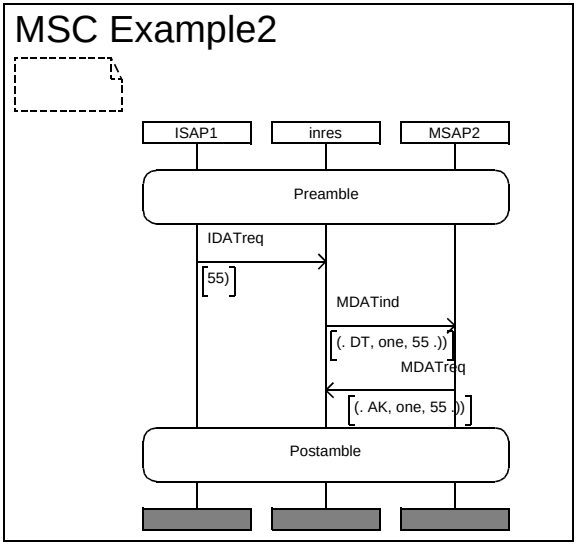

Figure 74: Example2

4. Save the MSC diagram as `Example2.msc`.

   You have now created a complete structured MSC test case with a pre- and a postamble and you are ready to generate a TTCN test case from it.

In the previous steps, you have changed the root of the behavior tree to the beginning of the postamble. Even though it is not necessary for a direct translation, you should better reset it before continuing.

1. Switch back to the *Validator UI* window.
2. Type **`Define-Root Original`** at the command line.

   The MSC Editor is started, but you do not have to bother about that. Select *Toggle MSC Trace* in the *Commands* menu to quit it.

You can list all MSC test cases and test steps currently stored on disk:

3. Select *MSC: List* in the *Autolink1* menu.

   The following output should appear in the text area:

```
MSC test cases:
Example1.mpr Example2.mpr Example2.msc

MSC test steps:
Postamble.mpr Preamble.mpr
```

It is also possible to remove test cases and test steps from disk. Since you do not need the temporary MSC *Example2.mpr* any more, you can delete it:

1. Select *MSC*: *Clear Test Case* in the *Autolink1* menu.

   A *Select* dialog is displayed.
2. Select `Example2.mpr`.
3. Click the *OK* button.

   Another *Select* dialog is displayed, asking you whether you really want to delete this MSC.
4. Choose `Yes`.
5. Click the *OK* button.

Defining an Autolink Configuration

Autolink allows to define a configuration which guides the naming and parameterization of constraints, the introduction of test suite parameters and constants and the grouping of test cases and test steps into a hierarchy of test groups. An Autolink configuration consists of a set of rules which have to be written by the user.

In this tutorial you will not learn how to write configuration rules. Instead you will use a pre-defined configuration that covers *some* aspects of an Autolink configuration. For a detailed descriptions of configuration rules and corresponding examples see "Translation Rules" on page 1421 in chapter 36, *TTCN Test Suite Generation* and "Test Suite Structure Rules" on page 1427 in chapter 36, *TTCN Test Suite Generation, in the User's Manual*.

In a first step you have to create a new command file which contains the Autolink configuration:

1. Open a text editor.

   The choice of text editor depends on the operating system of your computer.

   Create a new file called `config.com`.

2. Type in the following text:

```
# -------------------------------
#   Autolink configuration file
#     for the inres protocol
# -------------------------------

Define-Autolink-Configuration

TRANSLATE "ICONreq"
  CONSTRAINT NAME   "Connection_Request"
END

TRANSLATE "ICONconf"
  CONSTRAINT NAME   "Connection_Confirmation"
END

TRANSLATE "IDATreq"
  CONSTRAINT NAME   "Data_Request"
             PARS   $1="Data"
  TESTSUITE  PARS   $1=”DataValue”
END
```

```
TRANSLATE "IDISreq"
  CONSTRAINT NAME   "Disconnection_Request"
END

TRANSLATE "IDISind"
  CONSTRAINT NAME   "Disconnection_Indication"
END

TRANSLATE "MDATind" | "MDATreq"
  IF $1 == "{ id CC, *, * }" THEN
    CONSTRAINT NAME "Medium_Connection_Confirmation"
  END
  IF $1 == "{ id AK, *, * }" THEN
    CONSTRAINT NAME   "Medium_Acknowledge"
  END
  CONSTRAINT NAME   "c_" + $0
END

End
```

3. Save the file in your current working directory.

Next, you have to load this configuration into the Validator:

1. Select *Configuration: Load* in the *Autolink2* menu.

   A *File name* dialog is displayed.

2. Select `config.com` in the *Files* section.

3. Click the *OK* button.

If an error message is displayed, you have probably made a spelling mistake in the command file. In this case, correct your file and try to re-load the configuration.

Translating the MSC into TTCN

In "Generating the Test Case" on page 158, you have learned how to generate an internal test case representation based on an MSC. For that purpose, a state space exploration has been started.

However, in some cases it is not possible to simulate an MSC, e.g. if the internal processes of the SDL system are not fully specified. In these cases, you have to translate an MSC test case directly into an internal test case representation (without performing a state space exploration).

Even though it should be possible to simulate the MSC `Example2.msc`, you will apply a direct translation in this exercise:

1. Select *Test Case: Translate* in the *Autolink1* menu.

   A *Select* dialog is displayed.

2. Select `Example2.msc`.

3. Click the *OK* button.

   Autolink analyses the preamble, the test body and the postamble and generates a complete, structured test case. The following output should appear in the text area:

```
Current state is reset to root.

MSC 'Example2' loaded.

Constraints are checked for naming conflicts...

Constraint 'c_MDATind' is renamed to
'c_MDATind_001'.
Constraint 'c_MDATind' is renamed to
'c_MDATind_002'.
Constraint 'c_MDATind' is renamed to
'c_MDATind_003'.

Constraints with identical signal definitions are
merged automatically...

... done.
```

Take a look at the internal test case representation to see the difference with respect to test case `Example1`:

1. Select *Test Case: Print* in the *Autolink1* menu.

   A *Select* dialog is displayed.

2. Choose `Example2`.

3. Click the *OK* button.

4. The output is:

```
Example2

  0      IN Preamble
  1       ISAP1 ! ICONreq - Connection_Request
  2        MSAP2 ? MDATind - c_MDATind_003
  3         MSAP2 ! MDATreq -
               Medium_Connection_Confirmation
  4          ISAP1 ? ICONconf -
                Connection_Confirmation
  5           OUT Preamble
  6            ISAP1 ! IDATreq -
                  Data_Request(DataValue)
  7  (P)        MSAP2 ? MDATind - c_MDATind_002
  8              MSAP2 ! MDATreq - Medium_Acknowledge
  9               IN Postamble
 10                ISAP1 ! IDISreq -
                      Disconnection_Request
 11                ISAP1 ? IDISind -
                      Disconnection_Indication
 12   P             MSAP2 ? MDATind - c_MDATind_001
 13                  OUT Postamble
 11                MSAP2 ? MDATind - c_MDATind_001
 12   P             ISAP1 ? IDISind -
                      Disconnection_Indication
 13                  OUT Postamble
```

Also take a look at the constraints to see what the translation rules in your configuration file have affected:

5. Select *Constraint: List* in the *Autolink2* menu.

The following output will appear in the text area:

```
Connection_Confirmation :
  ICONconf {  }
Connection_Request :
  ICONreq {  }
Data_Request(Data) :
  IDATreq { iSDUType1 Data }
Disconnection_Indication :
  IDISind {  }
Disconnection_Request :
  IDISreq {  }
Medium_Acknowledge :
  MDATreq { mSDUType1 { id AK, num one, data 55 } }
Medium_Connection_Confirmation :
  MDATreq { mSDUType1 { id CC, num zero, data 55 } }
c_MDATind_001 :
  MDATind { mSDUType1 { id DR, num one, data 55 } }
c_MDATind_002 :
  MDATind { mSDUType1 { id DT, num one, data 55 } }
c_MDATind_003 :
```

```
MDATind { mSDUType1 { id CR, num zero, data 0 } }
```

Saving the TTCN Test Suite

When a TTCN test suite is created, test steps can be stored in different formats. By default, test steps are put into the test step library. However, you might want to save them locally, i.e. attached to their test cases:

1. Select *Autolink: Test Steps Format* in the *Options2* menu.

   A *Select* dialog is displayed.

2. Select `Local`.

3. Click the *OK* button.

Now, you are ready to save the test suite:

1. Select *Test Suite: Save* in the *Autolink1* menu.

   A *Prompt* dialog is displayed.

2. Type **`inres2_local`** in the *Test suite name* field.

3. Click the *OK* button.

   A *File name* dialog is displayed.

4. Type **`inres2_local.mp`** in the *File* field.

5. Click the *OK* button.

   Autolink saves the test suite in file `inres2_local.mp`.

You may want to save the test suite in different formats. To do this:

1. Select *Autolink: Test Steps Format* in the *Options2* menu.

   A *Select* dialog is displayed.

2. Select `Global` or `Inline`.

3. Click the *OK* button.

4. Save the test suite as above. You can save the current test suite as often as you like.

Merging With the Old Test Suite

It is possible to merge the new TTCN test case (with its constraints) with an existing test suite.

1. Make sure that the `inres` test suite you created in the first part of this tutorial is opened in the TTCN suite. This test suite contains test case `Example1`.
2. **On UNIX**, select *Autolink Merge* from the *SDT Link* menu in the Browser.

   In Windows, select *Autolink Merge* from the *File* menu.
3. In the dialog that is opened, select `inres2_local.mp`.
4. Click *Merge* **(UNIX)** or *OK* **(Windows)**.

   The test case and the constraints from the `inres2_local.mp` file will now be merged into the old test suite.

There are more possibilities to merge test cases: Autolink also provides commands to combine separately generated test cases within the Validator. For a detailed description see "Computing Test Cases" on page 1412 in chapter 36, *TTCN Test Suite Generation*, *in the User's Manual*.

Exercise 3: Test Generation with Tree Walk

What You Will Learn

- To create a large number of MSC test cases automatically
- To save and load generated test cases in Autolink
- To distribute test case generation

Preparations

If you have closed the SDL suite or the Validator UI temporarily, perform the following steps:

1. Open the `inres` system in the Organizer.
2. Start the Validator UI.
3. Open the `inres` validator again.

If you have **not** quit the `inres` validator, bring the application to its initial state now:

4. Select *Reset* in the *Options1* menu.

Creating MSC Test Cases Automatically

In the previous exercises, you had to specify your MSC test cases manually. Creating test cases by hand gives you full control over what is tested.

However, you might not be interested in defining tests for particular aspects of your SDL system. Instead, you only wish to create a large number of test cases randomly which cover most symbols of the SDL system.

In this case, you can use the Tree Walk algorithm provided by Autolink to generate test cases automatically.

In the first step, a number of *TreeWalk* reports have to be computed:

1. Click the *Tree Walk* button in the *EXPLORE* module.

   A *Prompt* dialog is displayed.

2. Enter the value **10**.

   This is the maximum number of minutes given to Autolink for computing reports. In fact, the state space exploration is much faster for simple protocols such as Inres.

   Another *Prompt* dialog is displayed.

3. Enter the value **100**.

   This is the percentage of symbol coverage that you want to reach when executing your test cases.

   A message similar to the one below will appear in the text area:

```
Tree Walk will stop after 10 minutes or after
reaching 100% symbol coverage.
Reports and symbol coverage table cleared.
States: 2109. Reports: 3. Tree walk reports: 0
(+31). Coverage: 100.00%. Time: 000:00:01

Target symbol coverage is reached.
Tree Walk is stopped.
Test report #1 of length 7 added.
Test report #2 of length 8 added.
Test report #3 of length 12 added.
Test report #4 of length 16 added.
Test report #5 of length 16 added.
Test report #6 of length 20 added.
Test report #7 of length 22 added.
Test report #8 of length 30 added.
Test report #9 of length 32 added.
```

The output above states that nine reports have been generated. When traversing their corresponding paths, a total symbol coverage of 100 percent is reached. The Tree Walk command stops immediately when it reaches the targeted symbol coverage.

Since Autolink requires MSC test cases as input, you have to convert the TreeWalk reports into MSCs:

1. Select *MSC: Save Reports* in the *Autolink1* menu.

   A *Select* dialog is displayed.

2. Select `TreeWalk` in order to save all TreeWalk reports.
3. Click the *OK* button.

   A Prompt dialog is displayed.
4. Enter **Tutorial** as test case name prefix.

   Autolink saves all *TreeWalk* reports as system level MSCs in distinct files in the test cases directory:

```
MSC test case is saved in file
'/home/tutorial/inres/TC/Tutorial_Tree_00001.mpr'.
MSC test case is saved in file
'/home/tutorial/inres/TC/Tutorial_Tree_00002.mpr'.
MSC test case is saved in file
...
MSC test case is saved in file
'/home/tutorial/inres/TC/Tutorial_Tree_00009.mpr'.
```

Generating the Test Cases

In the previous exercises you have learned how to generate internal test case representations from MSCs. If you want more than one test case, you can either compute all test cases in the test cases directory at once – by a single command – or compute one test case after another.

For large SDL systems, test generation may take some time. In this case, it is advantageous not to compute all test cases in a single validator session. Instead, each test case (or a small number of test cases) should be computed separately, preferably in parallel on different machines.

To enable distributed test generation, Autolink allows to store internal test case representations on disk. After all test cases have been computed, their internal representations can be reloaded from disk and saved in a single test suite. While loading the test cases, Autolink checks them for consistency and merges identical constraints.

Generating and Saving a Single Test Case

1. Select *Reset* in the *Options1* menu.

   All options are reset and all generated test cases and reports are cleared.
2. Select *Test case: Generate* in the *Autolink1* menu.
3. A *Select* dialog is displayed.

4. Select **Tutorial_Tree_00007.mpr**.
5. Click the *OK* button.
6. Select *Test case: Save* in the *Autolink1* menu.

   A *Select* dialog is displayed.
7. Select `Tutorial_Tree_00007`.
8. Click the *OK* button.

   A *File name* dialog is displayed.
9. Select the `TC` directory by double-clicking.
10. Type **Tutorial_Tree_00007.gen** in the *File* field.
11. Click the *OK* button.

    The generated test case is stored on disk.

Repeat steps 1 to 11 for other TreeWalk test cases.

Saving the Test Suite

Once you have created all test cases and saved them in files, you can combine them and create a test suite.

1. Select *Reset* in the *Options1* menu.
2. Select *Test case: Load* in the *Autolink1* menu.

   A *File name* dialog is displayed.
3. Select file `Tutorial_Tree_00007.gen` in the *Files* section.
4. Click the *OK* button.

   The generated test case is loaded and identical constraints are merged (if there are identical constraints).
5. Repeat steps 2 to 4 for all other test cases.
6. Select *Test suite: Save* in the *Autolink1* menu.

   A *Prompt* dialog is displayed.
7. Type **`TreeWalk`** in the *Test suite name* field.

8. Click the *OK* button.

   A *File name* dialog is displayed.

9. Type **`TreeWalk.mp`** the *File* field.

10. Click the *OK* button.

    Autolink saves the test suite file `TreeWalk.mp`.

    Now you may exit the Validator:

11. Select *Exit* in the *File* menu.

In order to take a look at the TTCN test suite, add the test suite to the Organizer just like you did in the first and second example above.

So Far...

You should have learned how to create structured system level MSCs, and how to generate TTCN test cases from those MSCs. Especially you have become acquainted with two ways of producing TTCN test cases: Test generation by state space search and test generation by a direct translation. While doing this, you have discovered many of the commands in the *Autolink* menus of the Validator. In addition, you should also know how to view the test suite in the TTCN suite, how to generate the test suite overview and how to merge new test cases and constraints into an existing test suite in the TTCN suite.