Chapter

56 *Analyzing a System*

Analyzing a system means checking that its SDL description obeys the syntax and semantic rules as defined in the Z.100 recommendation (some syntax checking is performed by the SDL Editor at editing time).

The SDL Analyzer allows you to perform a complete syntactic and semantic check of an SDL system. This chapter describes the Analyzer and how you may use it to analyze an SDL system.

For a reference to the Analyzer commands and the Analyzer functionality and restrictions, see *chapter 55, The SDL Analyzer*.

General Description

The Analyzer's main task is to perform syntactic and semantic analysis of SDL-92 definitions and diagrams, and to serve as a front end to code generators. You may perform full syntactic and nearly full semantic analysis of complete system definitions.

Analysis of separate units (block, process, substructure, service, and procedure) is also supported. Syntactic analysis may be performed on a unit, while restricted semantic analysis of a unit may only be performed if the context of the unit is provided. The context is the enclosing units and their definitions (for a detailed description, see the section *"Separate Analysis" on page 2435 in chapter 55, The SDL Analyzer*).

The Analyzer works in a number of passes:

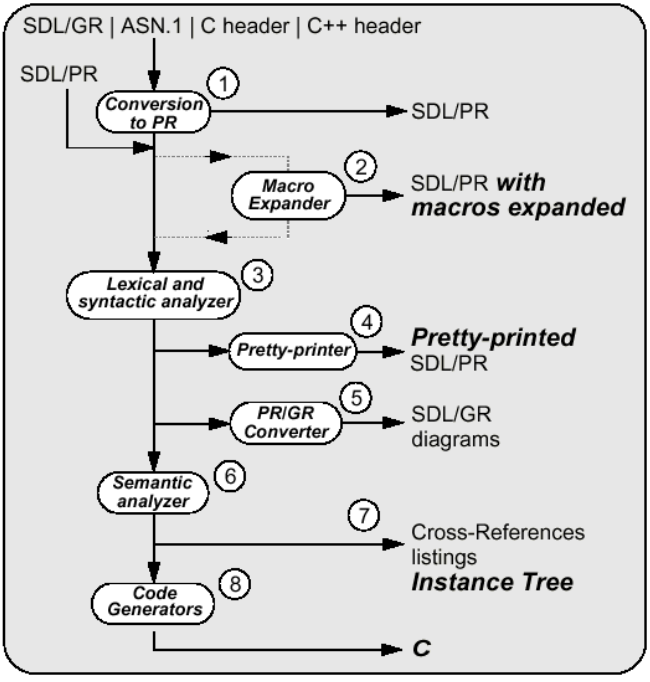

Figure 483: Analyzer passes

1. *Conversion to PR*. This pass is needed when input is not SDL/PR, i.e. for SDL/GR diagrams, ASN.1 files, C, and C++ header files. The result is a PR file with a "raw" layout, that is submitted as input to the next pass. This PR file is not intended to be read by the human. See *"Conversion to PR" on page 2431 in chapter 55, The SDL Analyzer* for a more thorough description.

2. *Macro expansion.* This pass is needed when input contains references to macros (which need to be expanded). The result is a PR file where SDL macros are expanded. See *"The Macro Expander" on page 2432 in chapter 55, The SDL Analyzer* for a more complete description of the Macro Expander.

3. *Lexical and syntactic analysis.* This pass checks that the input follows the SDL definition with respect to syntactic rules. During this pass, the Analyzer builds an abstract syntax tree that is used by the following passes. See *"The Lexical and Syntactic Analyzer" on page 2434 in chapter 55, The SDL Analyzer* for a more thorough description.

4. *Pretty-printing*. This optional pass uses the pretty-printer to produce an SDL/PR file with a nice layout, easy to read and understand by the human. The pretty-printed PR file contains the original SDL specification formatted according to specific layout rules.

5. *PR to GR conversion.* The input PR files are translated to SDL/GR diagrams, that you may open in the SDL Editor. This pass allows you to for instance import PR files from other tools supporting SDL. See also section *"The PR to GR Converter" on page 2437 in chapter 55, The SDL Analyzer*.

6. *Semantic analysis.* During this pass, the Analyzer checks that your SDL diagrams obey the static semantic rules as defined in the Z.100 recommendation. During this pass, the Analyzer builds and uses a symbol table, which can also be used later by the Code Generators. The semantic analysis is likely to be the "bottleneck" when analyzing a system. See "Optimizing a System to Reduce Analysis Time" on page 2551.

7. After the semantic analysis pass, you may optionally generate*:*
   - *cross-references* listings of SDL entities; where they are defined in an SDL system, and where they are used (referred). The result from this pass is a file which contents may be displayed graphically in the Index Viewer. See chapter 47, *The SDL Index Viewer* for more information.
   - *instance tree* information about the SDL system. The result from this pass is an instance information file consisting of records that describe the SDL entities that are present in the system after instantiation of all types. The file is a plain text file with a simple format. See *"File Syntax" on page 2443 in chapter 55, The SDL Analyzer* for more information.

8. If your configuration includes a Code Generator[1] you may include:
   - A *C Code Generation* pass, in order to generate a C description of your SDL system. This C code is then compiled and linked to generate a simulator, a validator or an application. The SDL to C Compiler is available as a Cbasic and as a Cadvanced code generator.
   - A *Cmicro Code Generation* pass. The generated C code is optimized with respect to memory requirements, making it suitable for generating applications for systems with limited resources.

Analyzer Input and Output

The input to the Analyzer consists of SDL-92 specifications, that is, SDL/GR diagrams that are stored using the storage format of the SDL suite, or SDL/PR files, or a combination of both.

The output consists mainly of PR files, error and warning messages. These messages are presented on the screen in a log window and may be stored on file. See "Error and Warning Messages" on page 2461 in chapter 55, *The SDL Analyzer* for a description of these messages.

It is also possible to obtain a pretty printed SDL/PR file of the input and to transform SDL/PR files into SDL/GR diagrams.

1. Although technically built into the Analyzer binary executable (sdtsan), the Code Generators are additional optional tools that are licensed separately.

The Analyzer User Interfaces

The Analyzer provides the following user interfaces.

Graphical UI

When started from the Organizer with the *Analyze* command, the Analyzer takes advantage of the graphical user interface and integration mechanism of the SDL suite.

For instance:

- Graphical references between source documents and error reports is supported, which facilitates locating and correcting errors in the source SDL diagrams.
- An SDL-Make facility, managed by the Organizer, controls the Analyzer and brings down the analysis work that needs to be done.
- On-line help on analysis diagnostics is available (provided the prerequisites for the on-line help are satisfied).

Batch UI

Suitable for running the Analyzer non-interactively. See "Batch Facilities" on page 203 in chapter 2, *The Organizer*.

Command-Line UI

Suitable for working on the file level with detailed control. See "The Analyzer Command-Line UI" on page 2404 in chapter 55, *The SDL Analyzer*.

Using the Analyzer

This section describes how you operate the Analyzer from the Organizer. We will discuss topics related to the various ways you may analyze an SDL structure. With the SDL suite, you may for instance perform syntax check on an SDL structure or check an entire SDL system with respect to the semantic rules.

Analyzing Using Default Options

To analyze an SDL structure using default options:

1. In the Organizer, select the root node for the subtree that is the subject to be input to the Analyzer.

2. Click the quick button for Analyze. The Organizer first checks if there are any unsaved diagrams; if any, the Organizer will prompt you to save these before analyzing them (since the Analyzer operates on the latest saved copy of a diagram).

3. The Organizer determines what diagrams need to be analyzed and what passes need to be run, by looking at the time the diagrams were saved on file and by monitoring the Analyzer's work.

   – To perform an analysis, you may either "touch" the SDL diagram files or force the Analyzer by clicking the *Full Analyze* button (see Figure 484).

4. The analysis job is submitted the to the Analyzer, using the options as they are currently defined in the Analyzer options dialog (see Figure 484 on page 2549).

5. From now on, the status bar reads "Analyzer working". When done, the status bar will read "Analyzer ready".

Analyzing Using Customized Options

To analyze an SDL structure with customized options:

1. In the Organizer, select the root node for the subtree that is the subject to be input to the Analyzer.

2. From the Organizer's *Generate* menu, select the *Analyze* command. The Analyzer Options dialog is displayed.

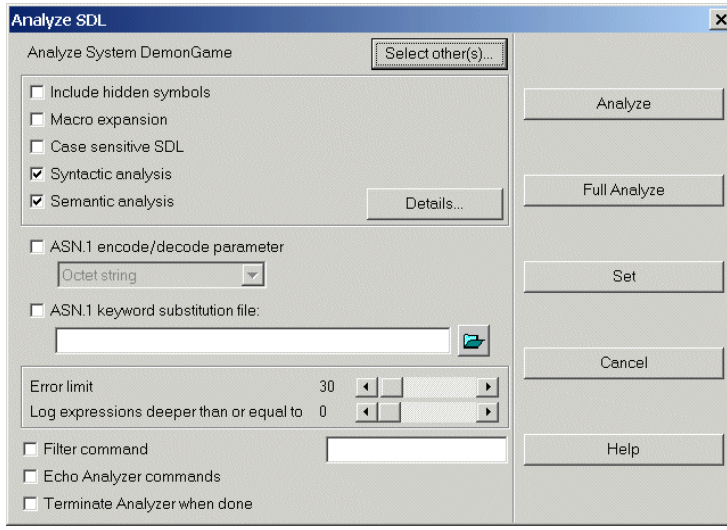

Figure 484: The Analyzer dialog

3. Adjust, if required, some of the Analyzer options to their required values. These options govern what passes should be performed by the Analyzer, see Figure 483 on page 2544, and what output should be produced by the Analyzer. (The code generation pass options are however not controlled from this dialog.)

4. Click the *Analyze* button. First, the Organizer checks if there are any unsaved diagrams; if any, the Organizer will prompt you to save these before analyzing them (since the Analyzer operates on the latest saved copy of a diagram).

5. The Organizer determines what diagrams need to be analyzed and what passes need to be run, by looking at the time the diagrams were saved on file and by monitoring the Analyzer's work.

   – To perform an analysis, you may either "touch" the SDL diagram files or force the Analyzer by clicking the *Full Analyze* button.

6. The Analyzer is initialized and starts executing (this is indicated in the Organizer status bar which now reads "Analyzer working").

7. When the Analyzer has analyzed the input as specified in the options, the Organizer status bar reads “Analyzer ready”. The results of the analysis are reported in the Organizer log window.

Performing Syntax Check

1. If the input contains SDL macros, you should expand these before proceeding with the syntax analysis in the resulting SDL/PR file.
   – Turn the *Macro expansion* button on to order expansion of SDL macros.
2. Turn the *Syntactic analysis* button on to include the syntax checking pass.
3. Click the *Analyze* button.

Performing Semantic Check

Turn the *Semantic analysis* button on to include the semantic check pass. This option cannot be turned on unless the *Syntactic analysis* pass is enabled. The semantic checker may be set up with by a number of options, each one individually activated by a toggle button.

Note: Optimizing a System to Reduce Analysis Time

There are two components that, to a large extent, affect the performance of the "resolution by context" in the semantic analysis pass:

- The depth of the expressions, because all possible combinations must be tried.
- The size of the system, because the context is all the visible identifiers.

Deeply nested expressions may cause a **significant** degradation of performance when performing the semantic analysis pass. It is therefore recommended to break down complex expressions into multiple, less complex expressions. *Checking for Deep Expressions* (see below) can assist you doing this.

If it is an option to modify your system, it might be worthwhile to go through all synonyms, newtypes and syntypes at the system level and move them as far as possible down the system structure. This makes the context visible to every expression smaller and reduces the time spent in "resolution by context". (Doing the same thing at lower levels will also improve performance, but not as much as on higher levels.) The Index Viewer might be useful in accomplishing this.

Checking Output Semantics

The *Check output semantics* option controls whether to issue warnings when SDL signal sendings, where the semantics is different in SDL-88 and in SDL-92, are detected. These warnings are particularly valuable when the input consists of SDL diagram that were designed in SDL-88 (for instance with SDT 2.X).

Detecting Not Used Definitions

The *Check unused definitions* option, when turned on, orders the Analyzer to report definitions that are not used (for instance variables that are declared but neither written or read).

Checking Optional Parameters

The *Check optional parameters* option controls whether to issue warnings when an optional parameter is omitted from a procedure call, an output, a create request, or an input. Note that in/out parameters are not

optional. An omitted parameter is indicated by empty parenthesis or a comma. See also *Checking Trailing Parameters*, below.

Checking Trailing Parameters

The *Check trailing parameters* option controls whether to issue warnings when the number of actual parameters is not equal to the number of formal parameters in a procedure call, an output, a create request, or an input. See also *Checking Optional Parameters*, above.

Checking Unique References

When the *Check references* option is activated, the Analyzer will check that each remote definition is referred only once. Turn this button off to disable this check.

Checking Missing Else Answer

The *Check missing else answers* option controls whether to issue warnings when an else part is expected but does not exist among the branches in a decision or transition option statement.

Checking Missing Answer Values

The *Check missing answer values* option controls whether to issue warnings when there are values not covered by any of the branches in a decision or transition option statement.

Allowing Implicit Type Conversions

The option *Allow implicit type conversion* controls whether implicit type conversions of reference data types (generators Own, ORef, and Ref) are allowed. For more information, see "Implicit Type Conversions" on page 134 in chapter 3, *Using SDL Extensions, in the SDL Suite Methodology Guidelines*.

Note:

Analyzing large expressions with this option on is slow.

Generating a Cross Reference File

Turn the *Generate a cross reference file* option on to have the Analyzer generate a file with a list of definitions and references to SDL entities, as an supplementary result from the semantic pass. You may also want

to specify another file name than the suggested one. The contents of this file may be read and visualized graphically with the Index Viewer.

Generating a Complexity Measurement file

Turn the *Generate a complexity measurement file* option on to have the Analyzer generate a file containing characteristics of the system. You may change the file name in the field below. See chapter 49, *Complexity Measurements* for more information.

Generating an Instance Information File

Turn the *Generate an instance information file* on to have the Analyzer generate a file with instance information about the analyzed system. The name of the file can be set in the field below. The file is produced after the semantic pass. Read more about the contents of an instance information file in *"SDL Instance Information" on page 2442 in chapter 55, The SDL Analyzer*.

Checking for Deep Expressions

Adjust the *Expression depth* parameter to specify the depth limit that the Analyzer should check for when evaluating expressions. Expressions which depth exceed the specified limit will be reported. Where possible, try to break down deep expressions since they require advanced calculations and slow down the Analyzer.

Specifying the Error Limit

Adjust the *Error limit* parameter to an adequate value (drag the slider for coarse adjustments, click left or right on the bar for fine adjustments). The Analyzer will stop its execution once this error limit has been reached.

Using a Filter

With the *Filter command* files can be filtered or preprocessed before they are read by the Analyzer. The specified executable file will be called with two arguments: the file to be processed by the Analyzer, and the Analyzer pass to be executed (import, macro, or parse). Try a simple OS command and look in the Organizer log to find out exactly how it is called.

Tracing the Analyzer Execution

You can trace the execution of the Analyzer by turning on the option *Echo Analyzer commands*. All Analyzer commands are then printed in the Organizer Log as they are executed.

Terminating the Analyzer Process

Turn on the option *Terminate Analyzer when done* if you want the Analyzer process to terminate after analysis is done. Otherwise, the Analyzer process is left running in the background.

Locating and Correcting Analysis Errors

The results of the Analyzer are appended to the Organizer Log Window. (You may save the window contents on any file at any time). The results of the last run are also saved on a file with the predefined name `analyzer.err`

The SDL suite provides a nice feature for displaying the source of an analysis error:

1. Locate the Organizer log (select *Organizer Log* if required).
2. Select the error (or warning) message by dragging the mouse.

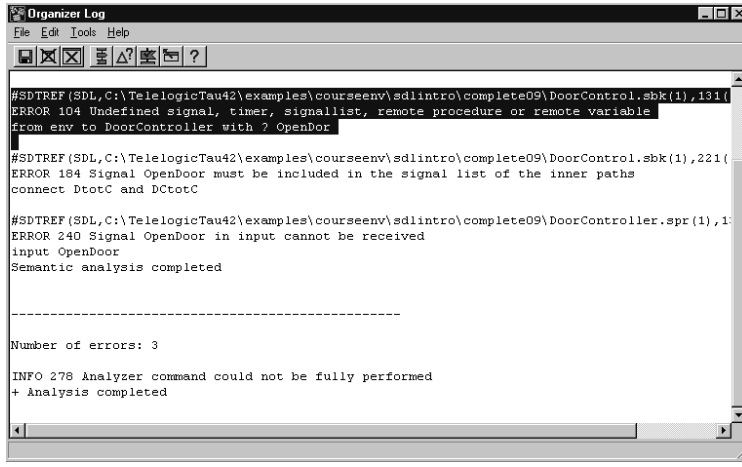

Figure 485: Selecting an error message

– The selection must contain at least the line starting with the text #SDTREF (for more information on the format of references, see chapter 19, *SDT References*).

3. Select the menu choice *Show Error* from the *Tools* menu.

4. The symbol where the error has been detected is displayed in an SDL Editor window. When the reference also contains a line and a column reference, the text cursor is placed on the line and column of text where the error was detected.

5. If you need additional explanations to understand the error message, select *Help on Error* from the *Tools* menu.

Producing a Pretty-Printed SDL/PR File

You may produce a pretty-printed PR file with the Analyzer, with a single command. The layout used by this PR makes it easy to read by the human. You may either submit SDL diagrams or SDL/PR files as input to the pretty-printer.

To be able to pretty-print SDL/PR, the SDL suite requires that the input must be syntactically correct. If the input does not fulfill this requirement, the SDL suite will however produce a PR file with a "raw" layout. This file is used by the SDL suite as a temporary file and it does not address the human reader.

1. In the Organizer, select the root node of the SDL structure to be pretty-printed.
2. From the Generate menu, select *Convert to PR/MP*. The *Convert to PR* dialog is displayed:

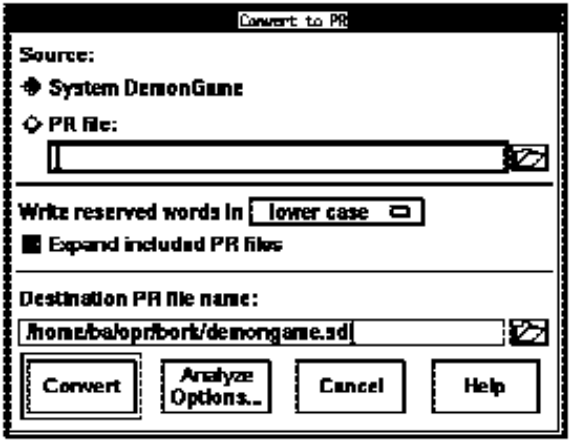

Figure 486: The Convert to PR dialog

3. Adjust, if required, the conversion options:
   - You may specify a PR file as alternative input. (The default input is the SDL structure designated by the object selected in the

Organizer.) To do this, type in the name of a file into the *PR file* text field.

- You may order the Analyzer to capitalize SDL reserved words or not; select the appropriate option from the *Write reserved words in* option menu.
- If the SDL structure contains #INCLUDE references to SDL/PR files, you may want to expand these and include them in the resulting pretty-printed PR file. Turn the *Expand included PR files* button on to order expansion of included PR files.

4. An output file where the resulting SDL/PR code will be stored is suggested. You may specify any other file to store the results on in the *Destination PR file name* text field.
5. If the input contains SDL macros, you may want to expand these in the resulting SDL/PR file.
   - Click the *Analyze Options* button to gain access to the *Analyzer options* dialog, where you turn the *Macro expansion* button on and click the *Set* button.
6. Click the *Convert* button to close the dialog and have the Analyzer start the conversion to SDL/PR.

Converting SDL/PR to SDL/GR

You may convert SDL/PR files to SDL/GR diagrams, storing them on files using the native format of the SDL suite. Once converted, these SDL descriptions may be managed, edited and printed by the graphical tools of the SDL suite just as if they were created using the SDL suite.

To convert an SDL/PR file to SDL/GR diagrams:

1. From the Organizer, *Generate* menu, select *Convert to GR*. A dialog is displayed:

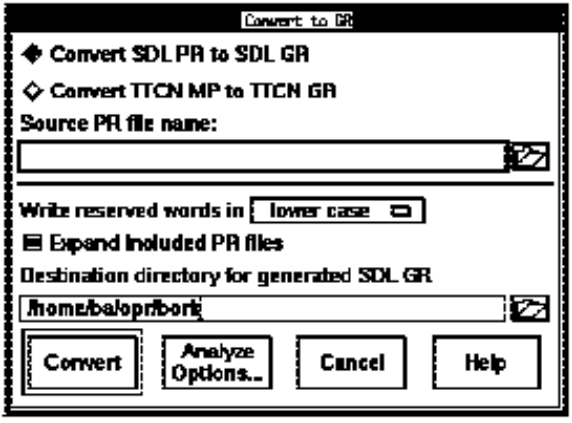

Figure 487: The Convert PR to GR dialog

2. Specify the PR file to convert from. You may either type the file specifier or click the folder button to issue a File Selection Dialog.

3. Specify the destination directory, where to store the results of the conversion. You may either type the name of a directory or click the folder button to issue a directory selection dialog.

4. You may specify other options:

   – Whether to capitalize SDL reserved words or not, by selecting the corresponding option from the option menu.

   – Whether to expand any included PR files or not (PR files are included using a #INCLUDE directive inside an SDL comment statement).

5. If required, click the *Analyze Options* button to modify the Analyze options (typically, you may need to turn the *Macro Expansion* option on). Close the *Analyze Options dialog* with the Set button (see Figure 484 on page 2549).

6. Click the *Convert* button to order the conversion. The Organizer's status bar reads "Analyzer ready" when the conversion is completed.

   – Each diagram that is specified (or included and expanded) in the source PR file is transformed to one file. These files are assigned default file names (see "Save in file" on page 62 in chapter 2, *The Organizer*).

   – The generated diagrams are assigned a layout by the SDL Editor.

   – Error and warning messages, if any, are appended to the Organizer log window.

7. To look at the resulting diagrams, you may either open the resulting files in the SDL Editor.

   – Alternatively, identify the root diagram file that has been produced. (You may need to look at the input SDL/PR file to determine what file to look for.) Then, import this file into the Organizer, with *Import SDL* from the *File* menu. Specify the root diagram file as *Root document*.

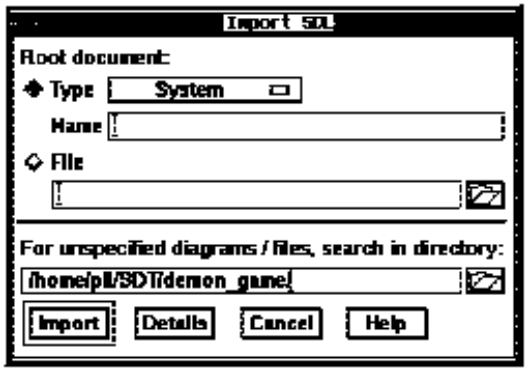

Figure 488: Importing an SDL structure

Chapter 56 Analyzing a System