BPEL2oWFN

Translating BPEL Processes to Open Workflow Nets
BPEL2oWFN Version 1.0, 16 January 2006

User's Manual

Niels Lohmann

About this document:

This manual is for BPEL2oWFN, Version 1.0, a tool translating business processes described in BPEL to open workflow nets (oWFN), last updated 16 January 2006. This manual does not explain how to setup or install BPEL2oWFN. For this information please read the Installation Manual which is part of the distribution or can be downloaded from the website of BPEL2oWFN (http://www.informatik.hu-berlin.de/top/tools4bpel/bpel2owfn).

Tools 4 BPEL

BPEL2oWFN is part of the Tools4BPEL project funded by the Bundesministerium für Bildung und Forschung. See http://www.informatik.hu-berlin.de/top/tools4bpel for details.

Table of Contents

1 Overview

1.1 Introduction

BPEL2oWFN is a compiler translating a business processes expressed in BPEL (Business Process Execution Language for Web Services) [ACD+03] to an oWFN (open Workflow Net) [MRS05]. This oWFN can be used to:

- check controllability [Mar03, Wei04],
- generate the operating guidelines [MRS05],
- check for deadlocks, or
- check any temporal logic formula expressed in CTL (Computation Tree Logic) with the model checking tool LoLA (Low-Level Analyzer) [Sch00].

In future versions of BPEL2oWFN static analysis is used to find the smallest oWFN to analyze a chosen property. This is called *flexible model generation* (see Chapter 7 [Future Work], page 13).

BPEL2oWFN was written by Niels Lohmann, Christian Gierds and Dennis Reinert. It is part of the Tools4BPEL project funded by the Bundesministerium für Bildung und Forschung. See `http://www.informatik.hu-berlin.de/top/tools4bpel` for details.

1.2 Translation Process

The translation process of the BPEL business process is performed in six steps which we describe briefly in this section:

1. **Lexical and syntactical analysis**. BPEL2oWFN parses the input process according to the specification of BPEL4WS version 1.1 [ACD+03]. All information about the process is collected in a symbol table for further use.
2. **Semantic analysis**. The input file is checked against the constraints of the specification, e.g. that each defined link has to be used as source and target exactly once. BPEL processes violating these constraints are rejected.
3. **AST generation**. For further analysis steps the exact syntax (indentation etc.) is not used any more. The input process is represented as an AST (abstract syntax tree). While generating the AST, the implicit transformation rules of BPEL (e.g. the presence of an '`otherwise`'-branch with an empty activity) are applied.
4. **Net generation**. The nodes of the AST are used to create the Petri net using the pattern database by applying 'unparse'-rules (rules associating each node with a pattern).
5. **Net optimization** (optional). To reduce the generated net several structural reduction rules can be applied, e.g. to merge sequences.
6. **Net output**. The generated Petri net can be exported in several file formats.

1.3 Concepts of BPEL2oWFN

In this section we describe the main concepts of BPEL2oWFN used to realize the translation. Reading this section is not necessary for using BPEL2oWFN, yet knowing the underlying algorithms and data structures not only helps to locate bugs, but also helps you to customize BPEL2oWFN or request a feature.

1.3.1 Abstract Syntax Tree

The AST (abstract syntax tree) is an abstraction of the syntax tree generated while parsing the BPEL process: any unnecessary information (e.g. indention, brackets or other "syntax-supporting" elements) is omitted. It is the central data structure of BPEL2oWFN. The nodes of the AST are annotated during the analysis steps. These annotations are used to select the most compact Petri net pattern from the pattern database to check a given property.

1.3.2 Pattern Database

The idea of flexible model generation is to find the most compact model to check a given property. The patterns of the Petri net semantics of [Sta05] are designed to fit in any given context. However when the context is known some behavior modeled in the patterns (i.e. some of the nodes) can be safely removed without changing its semantics. BPEL2oWFN is designed to hold several sets of Petri net patterns each suitable in certain contexts. These patterns are collected in a pattern database.

1.3.3 Petri Net Class

BPEL2oWFN provides many algorithms and data structures to build, represent, modify and simplify Petri nets and open workflow nets, resp. They are the interface between the pattern database and the file output for the model checking tool. The functions are collected in an extensible class allowing to add more output file formats (e.g. PNML (Petri Net Markup Language), PEP notation, etc.), structural simplification rules (optimized to preserve certain properties such as deadlock freedom or liveness) or abstractions (e.g. abstraction from variables, abstraction from external behavior).

2 Invoking BPEL2oWFN

The standard invocation of BPEL2oWFN is:

```
bpel2owfn -f inputfile.bpel -O2F
```

where '`inputfile.bpel`' is a BPEL process. The option `-O2F` causes BPEL2oWFN to generate an open workflow net named '`inputfile.owfn`' and a file '`inputfile.info`' describing the places and transitions of the net. For more examples, see [Examples], page 4.

2.1 Options

BPEL2oWFN supports the following command-line options:

'`--file` *`filename`*`.bpel`'
'`-f` *`filename`*`.bpel`'
: Read BPEL input from file '*`filename`*`.bpel`'. The filename is used to name any files generated by BPEL2oWFN. If this parameter is omitted, input is read from standard input.

'`--help`'
'`-h`'
: Print an overview of the command-line options and exit.

'`--version`'
'`-v`'
: Print version information and exit.

2.1.1 Output modes

When invoking BPEL2oWFN several output modes are possible. While the '`-pn`'-mode can be implied by other options the other modes have to be set explicitly. Either way, at most one of these options can be chosen.

'`--ast`'
'`-a`'
: Outputs the AST (abstract syntax tree) generated while parsing the input file to standard output. This option is mostly used for debugging reasons.

'`--xml`'
'`-x`'
: Outputs the parsed BPEL file in XML representation. Any unnecessary attributes are omitted. This option is mostly used for debugging reasons.

'`--petrinet`'
'`-pn`'
: Generates a Petri net representing the semantics of the given process. Other options can be added to simplify, abstract or modify that Petri net (see below).

2.1.2 Petri net related options

These options imply the '`-pn`'-option. When BPEL2oWFN is run with the '`-pn`'-option reading a file '`input.bpel`', a file '`input.info`' holding the information about the nodes of the created Petri net is generated. See [Naming Conventions], page 7 for more information.

'`--simplify`'
'`-s`'
: Uses several rules to structurally simplify the generated Petri net. See Chapter 5 [Petri Net related Functions], page 11 for more details.

'`--low-level`'
'`-ll`'
: Abstracts the generated Petri net to low-level representation. See Chapter 5 [Petri Net related Functions], page 11 for more details.

'`--lola`'
'`-L`'
: Outputs a low-level Petri net representation in the LoLA format (see Chapter 3 [File Formats], page 6). This option should not be used together with '`-D`', '`-D2F`', '`-O`', '`-O2F`'. This option implies '`-ll`'.

'--lola2file'
'-L2F' The same effect as the '--lola' option, yet an output file is generated: If the input BPEL process is read from 'file.bpel', the LoLA net is named 'file.lola'.

'--owfn'
'-O' Outputs an oWFN representing the semantics of the input process (see Chapter 3 [File Formats], page 6). This option should not used together with '-D', '-D2F', '-L' or '-L2F'.

'--owfn2file'
'-O2F' The same effect as the '--owfn' option, yet an output file is generated: If the input BPEL process is read from 'file.bpel', the oWFN is named 'file.owfn'.

'--dot'
'-D' Outputs a Dot representation of the Petri net (see Chapter 3 [File Formats], page 6). Should not used together with options '-L', '-L2F', '-O', '-O2F'. Implies option '-pn'.

'--dot2file'
'-D2F' The same effect as the '--dot' option, yet an output file is generated: If the input BPEL process is read from 'file.bpel', the Dot representation is named 'file.dot'.

2.2 Examples

In this section we show some examples how BPEL2oWFN can be invoked.

'bpel2owfn -f sample.bpel -L2F -s'
Reads the file 'sample.bpel', generates a structural simplified low-level Petri net and saves it in a LoLA file 'sample.lola'. For further information a file 'sample.info' is generated.

'bpel2owfn -f sample.bpel -O2F -ddd'
Reads the file 'sample.bpel', generates a low-level open workflow net and saves it in an oWFN file 'sample.owfn'. For further information a file 'sample.info' is generated. During the conversion several debug messages are printed to standard output.

'*prog* | bpel2owfn -D -ll | dot -Tpng -osample.png'
Runs the program *prog* and reads its output as BPEL process, generates a low-level Petri net and outputs its Dot representation. This stream is read by Dot which layouts the Petri net and creates an output PNG (Portable Network Graphic) file 'sample.png'.

'bpel2owfn -f sample.bpel -a'
Reads the file 'sample.bpel' and prints the abstract syntax tree (AST) to standard output.

2.3 Exit Values

When BPEL2oWFN is invoked and run without any error, the exit value is 0.

0 **No error.** The input file could be correctly opened, parsed and the output file(s) could be generated without any error.

1 **Lexical or syntax error.** This error occurs while lexing or parsing the input file. It is thrown by the lexer or the parser, resp. Usually the 'source' of the error (i.e. the filename and line number) is indicated together with the unexpected (last read) and expected token.

An example:

```
Error while parsing

syntax error, unexpected X_SLASH, expecting X_OPEN
Error in ‘example.bpel’ in line 12:
  token/text last read was ‘/’
```

Please note that the indicated position (i.e. the line number) may be fuzzy — it should be understood as a hint to the erroneous line.

2 **‘File not found’ exception.** The given input file was not found resp. could not be opened.

An example:

```
An error has occurred while parsing "example.bpel"!

Exception #2 occurred!

   File ‘example.bpel’ not found.
```

10 **Option mismatch.** The given command-line options cannot be processed together.

An example:

```
An error has occurred while parsing "example.bpel"!

Exception #10 occurred!

   Chosen parameters cannot work together (see parameter -h).

 Additional information:
   LoLA and dot output on stdout are confusing, chose one!
```

30 **‘Dynamic cast error’ exception.** While building an internal scope tree an unexpected error has occurred.

40 **Node not found.**

41 **‘Node already defined’ exception.** While generating the Petri net a node was found having a history entry covered by another node before.

An example:

```
An error has occurred while parsing "example.bpel"!

Exception #41 occurred!

   Place with role ‘1.internal.final’ already defined.
```

42 **‘Merging error’ exception.** While generating the Petri net an error occurred while merging two nodes. It happens either when one of the nodes was not found or one of the nodes is a guarded transition—the merging of guarded transitions is not yet supported.

43 **‘Arc error’ exception.** While generating the Petri net an error occurred while adding an arc to the net. It happens either on type errors — i.e. an arc between two transitions (or two places, resp.) should be drawn — or when the source or target node of an arc was not found.

Please report the occurrence of any exception with numbers 30–50 since it indicates a bug in BPEL2oWFN we would like to fix immediately (see [Reporting Bugs], page 12).

3 File Formats

BPEL2oWFN can generate several file formats:

3.1 LoLA place/transition net

A (low-level) place/transition net as described in [LoLA]. The places and transitions are named using the internal (numeric) name of each node. Furthermore, the first entry of the history of each node is added as a comment.

```
{ Petri net created by BPEL2oWFN reading filename.bpel }

PLACE
  a comma-separated list of places

MARKING
  the initial marking

a comma-separated list of transitions

{ END OF FILE }
```

For more information on the node naming conventions of BPEL2oWFN, see [Naming Conventions], page 7.

3.2 oWFN in LoLA format

An open workflow net is a Petri net with an *interface*, i.e. two sets of places: *input places* and *output places*. To represent oWFNs the LoLA format was extended to implement this categorization:

```
{ oWFN created by BPEL2oWFN reading filename.bpel }

PLACE
  INPUT
    a comma-separated list of input places
  OUTPUT
    a comma-separated list of output places
  INTERNAL
    a comma-separated list of internal places

MARKING
  the initial marking

a comma-separated list of transitions

{ END OF FILE }
```

Additionally an open workflow net has a set of final markings. Since there exist no tools reading oWFNs yet, the representation of final markings is not implemented.

3.3 Info-files

The Info-files are generated when any command-line option is used which imply Petri net-generation. When reading from a file '`process.bpel`' a file '`process.info`' is generated. This file sums up all places and transitions together with their internal (numeric) name and their complete history:

```
PLACES:
ID  TYPE        ROLES
a list of places

TRANSITIONS:
ID  GUARD       ROLES
a list of transitions
```

These files are generated to document the connection between the generated output file and the chosen Petri net patterns. In future distributions of BPEL2oWFN the Info-files will be used to annotate witness and counter-example paths, resp. and to "re-translate" Petri net properties (e.g. a dead transition) to the input BPEL process.

For more information on the node naming conventions of BPEL2oWFN, see [Naming Conventions], page 7.

3.3.1 Naming Conventions

BPEL2oWFN generates the output Petri net by creating and merging parameterized patterns of the Petri net semantics defined in [Sta05]. Due to merging and simplifying the Petri net nodes "belong" to more than one pattern. For example, in a sequence the initial place of the sequence and the initial place of its first activity are merged so that the final Petri net contains one place with two *roles*.

The roles of each place are collected during the Petri net generation. They form the *history* of the node. It is used to locate errors of the modeled business process: If, for example, BPEL2oWFN generates a Petri net of a business process and the model checker LoLA finds a dead transition, its history helps to find which BPEL constructs are affected and in this case will never be executed.

The roles are named using the following conventions:

- Each BPEL activity has been assigned an identifier during the syntactic analysis of the input process. Each node added to the Petri net from the Petri net pattern of that activity begins with that identifier.

  For example, BPEL's activity process has the identifier '`1`', so that all nodes of the process pattern begin with '`1.`'. To find out the identifiers of a given process use the '`--xml`' command-line option which prints the id of each activity as an XML attribute.
- In most cases each BPEL activity can be source or target of links. The semantics defined in [Sta05] organizes this link concept by several wrappers. For an activity with the identifier *id* the nodes of the wrapper begin with '*`id`*`.`' whereas the nodes of the actual activity begin with '*`id`*`.internal.`'.
- The roles of nodes of the stop pattern of a process or scope with identifier '*`id`*' begin with '*`id`*`.internal.stop.`'.

  The same schema is used for fault handlers ('*`id`*`.internal.faultHandler.`'), compensation handlers ('*`id`*`.internal.compensationHandler.`') and event handlers ('*`id`*`.internal.eventHandler.`'), resp.
- Labels (e.g. '`initial`') in a figure of [Sta05] are appended to the id string (e.g. '*`id`*`.internal.initial`'). If both numeric (e.g. '`p1`') and textual (e.g. '`initial`') labels are depicted in a figure, the latter is used.
- The labels of fault-throwing transitions also contain the last place of the positive control flow: If, for example, a reply activity throws a fault, the fault-throwing transition reads from the place labeled '*`id`*`.internal.running`' and is labeled '*`id`*`.internal.throwFault.running`'.
- In parameterized patterns (e.g. an assign activity or all structured activities) the labels of the figures of [Sta05] are trailed by an numeration (e.g. '*`id`*`.internal.copy.`*`number`*`.running`').

3.4 Dot Graph

To bugfix[1] the implemented Petri net patterns BPEL2oWFN implements a graph representation of the generate Petri net.

[1] The Petri nets usually have a large number of nodes so that the graphical representation of a 'real world' process would not be suitable to process, read or understand. That is why the Dot output shall be seen as a means to debug small patterns.

4 Petri Net Patterns

In version 1.0 of BPEL2oWFN the following Petri net patterns are implemented:

4.1 Petri net semantics from [Sta05]

The Petri net semantics for BPEL4WS from Christian Stahl (Humboldt-Universität zu Berlin) published in [Sta04].

4.1.1 Overview

Feature complete semantics covering both positive control flow with event handling and negative control flow (fault and compensation handling).

4.1.2 Limitations of the semantics

- Only one instance of a BPEL process can be transformed into a Petri net.
- The semantics abstracts from the connection of a BPEL process to its partner processes. The interface of a BPEL process is transformed into a set of message channels, i.e. places in the Petri net.
- In our Petri net patterns we model data, but we abstract from the definition of the functions which edit the data. Furthermore, we did not specify the transition guards and so we did not specify which circumstances are necessary that a specific fault can occur.
- Every activity is limited to one correlation set (except the synchronous invoke which is limited to two correlation sets).

4.1.3 Changes and Modulation

We tried to stick as close to the Petri net patterns of [Sta05] as possible. However, the implemented patterns in the pattern database sometimes differ to the given patterns due to discovery of bugs or implementation decisions. In this subsection we sum up these changes to help you understand the generated Petri net model.

- **Faults**. At most one error can occur in the positive control flow of each scope or process. Yet this confines the possible runs of the process it is only a little change of the semantics, since — according to the specification — only the first fault is handled anyway. While further faults occurring before the positive control flow is stopped are ignored in the original semantics of [Sta05] (in fact, the faults are collected on place '`fault_in`' and then consumed by a reset arc) they are prevented in the implemented semantics. In our model, exactly the *first* occurring fault is handled, whereas in [Sta05] one fault is chosen non-deterministically.

  Furthermore, all '`failed`' places of the activities were removed. In the original Petri net semantics, all faults of a scope were collected on the '`fault_in`' of the stop-pattern and then classified as being the first fault of the scope, a following fault, a fault from the fault handler, a fault from the compensation handler, or a fault from a child scope. In our implementation, new places ('`fh_fault_in`' and '`ch_fault_in`') were introduced and each activity throws its faults to the "correct" place automatically.

  To ensure that at most one error can occur (i.e. at most one token is produced on any fault place) the fault places are guarded by state places: To throw a fault from an activity enclosed in a scope, the state of that scope has to be '`Active`'. The first thrown fault changes the state to '`!Active`' thus preventing more faults to occur. The places '`fh_fault_in`' and '`ch_fault_in`', resp. are guarded by '`!FHFaulted`'/'`FHFaulted`' and '`!CHFaulted`'/'`CHFaulted`', resp.

  Moreover, the generated Petri nets have less nodes than those generated by BPEL2PN [SHS05] since an unfolding of the reset arcs in not necessary any more.

- **1-safety**. The new modeling of the fault management yields to 1-safe Petri nets (i.e. any reachable state of the Petri net model puts at most one token on each place of the net).

  Beside performance (e.g. only 1 bit is needed to store the marking of a place) and compatibility issues (e.g. 1-safety is a prerequisite to use the Model Checking Kit [MCK]), features not supported by the Petri net semantics can be discovered since the generated net will most likely violate 1-safety when an unsupported BPEL feature is used. If, for example, a scope is enclosed in a while loop (which would model instantiation which is not supported by the Petri net semantics [Limitations of the semantics], page 9), the state places of that scope would not be 1-safe.
- **Assign activity**. All copy branches of an assign activity are modeled in a single pattern (i.e. Fig. 6 and Fig. 7 are merged). Furthermore, when an error (outside that activity) occurs, an active assign-activity is not stopped until all copy branches have finished. This is described in [ACD+03] as:

  > The assign activities are sufficiently short-lived that they are allowed to complete rather than being interrupted when termination is forced.

  This change fixes a bug in the Petri net semantics.
- **Event handlers**. There is one pattern for both alarm and message event handlers (i.e. Fig. 29 and Fig. 30 are merged). When no event handler is specified, an "implicit" event handler is installed which is just a stub and does not change the semantics.
- **Deadlocks**. A transition named '`1.internal.finishloop`' was added to livelock the process upon completion. This leads to deadlock-free Petri nets in case of processes with "reasonable" control flow and helps to find unwanted deadlocks occurring due wrong modeling. If, for example, the links of a process model are cyclic the generated Petri net will deadlock.

  In future versions of BPEL2oWFN these found deadlocks shall be mapped back into the BPEL code to highlight the "unreasonable" activities (i.e. a cycle-closing link).
- **Unfoldings**. Due to the abstraction (high-level to low-level) of the patterns some places were unfolded: the place '`compScope`' of Fig. 42–44. usually holding a token with a name of a scope is unfolded to '`compScope.`*`scopename`*' and only merged with the '`ch_in`'-place of that respective scope. In all other cases the places are "converted" to low-level places so the generated model completely abstracts from data.
- **Link semantics**. The generated Petri net model always generates '`negLink`' places for structured activities independently of the presence of links. Anyway, the semantics is not changed since the resulting subnets are dead in this case.
- **Correlation sets**. Correlation sets are not implemented and are simply ignored during parsing.

5 Petri Net related Functions

Currently implemented Petri net-specific functions:

5.1 Structural Simplification

- If two transitions t_1 and t_2 have the same preset and postset, one of them can be removed.
- If a transition has a singleton preset and postset, the transition can be removed (sequence) and the preset and postset can be merged.

These structural reduction rules are implemented in the command-line option '`--simplify`', see Chapter 2 [Invoking BPEL2oWFN], page 3).

5.2 Abstractions

- To obtain a place/transition Petri net from an open workflow net the communication places are removed. (Implemented in command-line option '`--lola`'.)
- To improve readability of the implemented patterns all transition guards, arc inscriptions and arc types were adopted from [Sta05]. To "convert" these patterns to a low-level Petri net, all transition guards and arc inscriptions were removed (decisions are now made non-deterministically) and read arcs are "unfolded" to loops. Due to a new fault management (see [Changes and Modulation], page 9) the semantics does not contain any reset arcs.

  These abstractions are implemented in the command-line option '`--low-level`'.

5.3 Markings

The following places are initially marked to ensure a deadlock-free model of processes with "reasonable" control-flow (e.g. with an acyclic link structure):

- the initial place of the process ('`1.internal.initial`'),
- the variable places ('`variable.`*`variablename`*'), and
- the clock ('`1.internal.clock`').

All interleavings should end in a livelock-marking which covers '`1.internal.final`' and in which only transition '`1.internal.finishloop`' can fire. All livelock-markings differ in the marking of the state places of the process and its child scopes.

Deadlocks indicate modeling mistakes (e.g. a cyclic link structure). Non-1-safe markings indicate the usage of an unsupported feature of the Petri net semantics (e.g. instantiation, see [Limitations of the semantics], page 9).

6 Limitations and Bugs

6.1 Limitations

The current version of BPEL2oWFN underlies several limitations:

- The pattern database consists of only one collection of patterns (i.e. the semantics of [Sta05], see *Petri Net Patterns*).

6.2 Known Bugs

As this is the first public version of BPEL2oWFN the translation from a BPEL process to an open workflow net might be unstable or incorrect in some few scenarios:

- **Problem:** The original semantics of [Sta05] was created to support executable BPEL processes. Therefore the translation of abstract BPEL processes (business protocols) might throw an exception or even crash.

  Solution: Each communicating activity (i.e. `invoke`, `receive`, `reply`) should be defined with (input/output) variables.
- **Problem:** The implementation of the structural reduction rules might crash when applied to BPEL processes that created Petri nets with a large number of nodes.

  Solution: Avoid using the '`-s`' mode in this case.
- **Problem:** The parser of BPEL2oWFN is not capable of skipping XML elements originating other namespaces than '`bpws`'. Processes using these elements are rejected with a syntax error message.

  Solution: Try removing or commenting these elements.

6.3 Reporting Bugs

If you find a bug in BPEL2oWFN, please first check that it is not a known bug listed in 'Known Bugs'. Otherwise please send us an electronic mail to `nlohmann@informatik.hu-berlin.de`. Include the version number which you can find by running '`bpel2owfn --version`'. Also include in your message the input BPEL process and the output that the program produced. We will try to answer your mail within a week.

If you have other questions, comments or suggestions about BPEL2oWFN, contact us via electronic mail to `nlohmann@informatik.hu-berlin.de`.

6.4 Contact Person

Niels Lohmann

Humboldt-Universität zu Berlin
Institut für Informatik
Unter den Linden 6
10099 Berlin, Germany

Homepage	`http://www.informatik.hu-berlin.de/top/mitarbeiter/lohmann`
E-mail	`nlohmann@informatik.hu-berlin.de`
Phone	(+49) (30) 2093-3070
Fax	(+49) (30) 2093-3067

7 Future Work

For future releases of BPEL2oWFN the following features are planned:

- **Flexible model generation**. Originally, BPEL2oWFN was designed to support flexible model generation, i.e. finding the most abstract (thus smallest) model capable to check a given property. With static analysis we hope to generate the smallest Petri net possible to check properties like deadlock-freedom or controllability.
- **More Petri net patterns**. Being feature-complete the Petri nets generated from the Petri net semantics of [Sta05] consist of a large number of nodes and might be to large to be analyzed by model checkers. The elaboration of different petri net patterns specialized for certain purposes (e.g. omitting the negative control flow) does not only help to find smaller models, but is also a prerequisite for flexible model generation.
- **More output file formats**. The currently supported output file formats limit the analysis of the generated Petri net to the model checker LoLA. The support of more output file formats such as the low-level PEP notation allows the analysis of the generated models with the large number of both explicit and symbolic model checking tools embedded in the Model Checking Kit [MCK].
- **Detailed info-files**. The generated info-files currently just list the nodes of the generated net. To help the retranslation of Petri net-specific properties to the input process the generated files have to be more detailed.
- **Support for WS-BPEL**. The specification of WS-BPEL (Web Service Business Process Execution Language) version 2.0 is in its final phase. As soon as the standardization is completed, WS-BPEL can be supported by BPEL2oWFN by overworking the grammar and adding appropriate patterns to the pattern database.

Appendix A References

[ACD+03] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. **Business Process Execution Language for Web Services, Version 1.1**. Technical report, BEA Systems, IBM, Microsoft, SAP, Siebel Systems. May 2003.

[SHS05] Sebastian Hinz, Karsten Schmidt, and Christian Stahl. **Transforming BPEL to Petri Nets**. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Proceedings of the Third International Conference on Business Process Management (BPM 2005)*, pp. 220-235, Springer-Verlag (LNCS 3649), September 2005.[1]

[LoLA] Karsten Schmidt: **LoLA: A Low Level Analyser**. Manual.[2]

[Mar03] Axel Martens. **Verteilte Geschäftsprozesse – Modellierung und Verifikation mit Hilfe von Web Services** *(in German)*. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, 2003.

[MCK] Javier Esparza, Claus Schröter, and Stefan Schwoon. **Model-Checking Kit**.[3]

[MRS05] Peter Massuthe, Wolfgang Reisig, and Karsten Schmidt. **An Operating Guideline Approach to the SOA**. Proceedings of the 2nd South-East European Workshop on Formal Methods 2005 (SEEFM05), Ohrid, Republic of Macedonia, 2005.[4]

[Sch00] Karsten Schmidt: **LoLA: A Low Level Analyser**. In: Mogens Nielsen, and Dan Simpson, editors: *Application and Theory of Petri Nets, 21st International Conference (ICATPN 2000)*, pp. 465-474, Springer-Verlag (LNCS 1825), June 2000.

[Sta05] Christian Stahl. **A Petri Net Semantics for BPEL**. Informatik-Berichte 188, Humboldt-Universität zu Berlin, July 2005.[5]

[Wei04] Daniela Weinberg. **Analyse der Bedienbarkeit**. Diplomarbeit, Humboldt-Universität zu Berlin, October 2004.[6]

---

[1] Tool available at `http://www.informatik.hu-berlin.de/top/bpel2pn`

[2] `http://www.informatik.hu-berlin.de/top/lola/doku.ps`

[3] `http://www.fmi.uni-stuttgart.de/szs/tools/mckit`

[4] `http://www.informatik.hu-berlin.de/top/download/publications/MassutheReisigSchmidt-OGApproach.ps`

[5] `http://www.informatik.hu-berlin.de/Institut/struktur/systemanalyse/preprint/stahl188.pdf`

[6] `http://www.informatik.hu-berlin.de/top/download/publications/weinberg04.pdf`

Appendix B GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program," below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification.") Each licensee is addressed as "you."

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version," you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 20yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type 'show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type 'show c'
for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program 'Gnomovision'
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.