Iorina Documentation

Release v0.1

Heinz Riener

Oct 26, 2018

Contents

CHAPTER 1

Introduction

1.1 Synopsis

The C++ library *lorina* implements parsers for simple file formats used in logic synthesis and formal verification. A callback mechanism in form of customizable visitor classes allows users to react on parsed primitives and parse errors.

The library consists of several independent parsers. Each parser has its own header file that contains a *reader function* `read_<format>` and a *reader visitor* `<format>_reader`, where `<format>` has to be substituted with the format to be parsed. The reader visitor class `<format>_reader` provides a set of default implementations of virtual `on_<primitive>` methods called by the parser. Each callback method corresponds to an *event point* defined by the implementation of the parsing algorithm, e.g., the completion of the parsing of the format's header information, or a certain input or gate definition.

The `read_<format>` function can either be used to open and parse a file provided by name (`const std::string& filename`) or to parse tokens from an input stream (`std::istream& in`):

```
/* read file */
const std::string filename = ...;
read_<format>( filename, <format>_reader() );

/* read istream */
std::ifstream in( filename.c_str(), std::ifstream::in );
read_<format>( in, <format>_reader() );
```

1.2 Customization

The default behavior of any reader visitor can be customized by deriving from a reader visitor and overriding its virtual callback methods. In the following code snippet, the class `print_bench_input_decl` derives from `bench_reader` and customizes the behavior of the `on_input` event point:

```
class print_bench_input_decl : public bench_reader
{
public:
  void on_input( const std::string& name ) const override
  {
    std::cout << "INPUT: " << name << std::endl;
  }
};
```

1.3 Diagnostics

To reduce code and executable size, error handling is implemented via return codes and a lightweight callback visitor mechanism (instead of C++ exceptions). This allows users to embed lorina into their projects even when C++ exceptions are globally disabled.

All reader functions `read_<format>` either return `return_code::success` if parsing has been successful or otherwise `return_code::parse_error`. The *diagnostic engine* additionally allows users to react on parse errors. Diagnostics are automatically activated by providing a pointer to a `diagnostic_engine` as a third parameter to a reader function:

```
diagnostic_engine diag;
return_code result = read_<format>( filename, <format>_reader(), &diag );
```

The *parse error* event points and the corresponding *error messages* are specified by the respective parsing algorithm. The default behavior of the diagnostic engine can be customized (similarly to the reader visitors) by deriving from *diagnostic_engine* and overriding its virtual callback method `emit`.

CHAPTER 2

Installation

lorina is a header-only C++ library. Just add the include directory of lorina to your include directories, and you can integrate lorina into your source files using

```
#include <lorina/lorina.hpp>
```

2.1 Building examples

Examples are enabled by default. In order to build them, run the following commands from the base directory of lorina:

```
mkdir build
cd build
cmake ..
make
```

2.2 Building tests

In order to run the tests, you need to enable them in CMake:

```
mkdir build
cd build
cmake -DLORINA_TEST=ON ..
make
./test/run_tests
```

CHAPTER 3

Change Log

3.1 v0.3 (Not yet released)

3.2 v0.2 (October 18, 2018)

- Reader visitor for a simplistic version of structural Verilog (*verilog_reader*)
- Implementation of a pretty-printer for Verilog: (*verilog_pretty_printer*)
- Writer visitor for assembling strings to PLA (*pla_writer*)
- Support for parsing AIGER 1.9 (*aiger_reader*, *ascii_aiger_pretty_printer*)

3.3 v0.1 (April 27, 2018)

- Reader visitors for simple file formats commonly used in logic synthesis and formal verifiation (*aiger_reader*, *bench_reader*, *blif_reader*, *pla_reader*)
- Implementations of reader visitors for pretty-printing (*ascii_aiger_pretty_printer*, *bench_pretty_priter*, *blif_pretty_printer*, *pla_pretty_printer*)
- Diagnostic infrastructure (*diagnostic_builder‘*, *diagnostic_engine*, *silent_diagnostic_engine*)
- Utilities for generic topological re-sorting (*call_in_topological_order*)

CHAPTER 4

Acknowledgments

The implementation of `aiger_reader` was highly inspired by the function `read_aiger_file` in pyaig by Baruch Sterin. I thank Mathias Soeken, Gianluca Martino, Marcel Walter, Jin Hee Kim for discussions, code constribtuions, and bug reports.

CHAPTER 5

Parsers

5.1 AIGER format

Header: `lorina/aiger.hpp`

5.1.1 AIGER parser

The header `<lorina/aiger.hpp>` implements methods to parse the AIGER format (see http://fmv.jku.at/aiger/).

The class *`lorina::aiger_reader`* provides the following public member functions.

Function	Description
`on_header(m, i, l, o, a)`	Callback method for parsed header
`on_header(m, i, l, o, a, b, c, j, f)`	Callback method for parsed header
`on_input(index, lit)`	Callback method for parsed input
`on_output(index, lit)`	Callback method for parsed output
`on_latch(index, next, reset)`	Callback method for parsed latch
`on_and(index, left_lit, right_lit)`	Callback method for parsed AND gate
`on_input_name(index, name)`	Callback method for parsed input name
`on_latch_name(index, name)`	Callback method for parsed latch name
`on_output_name(index, name)`	Callback method for parsed output name
`on_bad_state_name(index, name)`	Callback method for a parsed name of a bad state property
`on_constraint_name(index, name)`	Callback method for a parsed name of an invariant constraint
`on_fairness_name(index, name)`	Callback method for a parsed name of a fairness constraint
`on_comment(comment)`	Callback method for parsed comment

The following reader functions are available.

Function	Description
read_ascii_aiger	Reader function for ASCII AIGER format.
read_ascii_aiger	Reader function for ASCII AIGER format.
read_aiger	Reader function for binary AIGER format.
read_aiger	Reader function for binary AIGER format.

return_code `lorina::`**`read_ascii_aiger`**(**`const`** std::string &*filename*, **`const`** *aiger_reader* &*reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for ASCII AIGER format.

Reads ASCII AIGER format from a file and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `filename`: Name of the file
- `reader`: An AIGER reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

return_code `lorina::`**`read_ascii_aiger`**(std::istream &*in*, **`const`** *aiger_reader* &*reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for ASCII AIGER format.

Reads ASCII AIGER format from a stream and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `in`: Input stream
- `reader`: An AIGER reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

return_code `lorina::`**`read_aiger`**(**`const`** std::string &*filename*, **`const`** *aiger_reader* &*reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for binary AIGER format.

Reads binary AIGER format from a file and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `filename`: Name of the file
- `reader`: An AIGER reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

return_code `lorina::read_aiger`(std::istream *&in*, **const** *aiger_reader* *&reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for binary AIGER format.

Reads binary AIGER format from a stream and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `in`: Input stream
- `reader`: An AIGER reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

class aiger_reader

A reader visitor for the binary AIGER format.

Callbacks for the AIGER format.

Subclassed by lorina::ascii_aiger_pretty_printer

5.2 BENCH format

Header: `lorina/bench.hpp`

5.2.1 BENCH parser

The header <lorina/bench.hpp> implements methods to parse the BENCH format.

The class *lorina::bench_reader* provides the following public member functions.

Function	Description
`on_input(name)`	Callback method for parsed input
`on_output(name)`	Callback method for parsed output
`on_gate(inputs, output, type)`	Callback method for parsed gate
`on_assign(input, output)`	Callback method for parsed gate assignment

The following reader functions are available.

Function	Description
read_bench	Reader function for the BENCH format.
read_bench	Reader function for BENCH format.

return_code `lorina::read_bench`(std::istream *&in*, **const** *bench_reader* *&reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for the BENCH format.

Reads BENCH format from a stream and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `in`: Input stream
- `reader`: A BENCH reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

return_code `lorina::read_bench`(**const** std::string &*filename*, **const** *bench_reader* &*reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for BENCH format.

Reads BENCH format from a file and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `filename`: Name of the file
- `reader`: A BENCH reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

class bench_reader

A reader visitor for the BENCH format.

Callbacks for the BENCH format.

Subclassed by lorina::bench_pretty_printer

5.3 BLIF format

Header: `lorina/blif.hpp`

5.3.1 BLIF parser

The header <lorina/blif.hpp> implements methods to parse the BLIF format.

The class *lorina::blif_reader* provides the following public member functions.

Function	Description
`on_model(model_name)`	Callback method for parsed model
`on_input(name)`	Callback method for parsed input
`on_gate(inputs, output, cover)`	Callback method for parsed gate
`on_end()`	Callback method for parsed end
`on_comment(comment)`	Callback method for parsed comment

The following reader functions are available.

Function	Description
read_blif	Reader function for the BLIF format.
read_blif	Reader function for BLIF format.

return_code `lorina::`**`read_blif`**(std::istream *&in*, **`const`** *blif_reader &reader*, *diagnostic_engine *diag* = nullptr)

Reader function for the BLIF format.

Reads BLIF format from a stream and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `in`: Input stream
- `reader`: A BLIF reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

return_code `lorina::`**`read_blif`**(**`const`** std::string *&filename*, **`const`** *blif_reader &reader*, *diagnostic_engine *diag* = nullptr)

Reader function for BLIF format.

Reads binary BLIF format from a file and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `filename`: Name of the file
- `reader`: A BLIF reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

`class blif_reader`

A reader visitor for the BLIF format.

Callbacks for the BLIF (Berkeley Logic Interchange Format) format.

Subclassed by lorina::blif_pretty_printer

5.4 PLA format

Header: `lorina/pla.hpp`

5.4.1 PLA parser

The header `<lorina/pla.hpp>` implements methods to parse the PLA format.

The class *`lorina::pla_reader`* provides the following public member functions.

Function	Description
`on_number_of_inputs(number_of_inputs)`	Callback method for parsed number of inputs
`on_number_of_outputs(number_of_outputs)`	Callback method for parsed number of outputs
`on_number_of_terms(number_of_terms)`	Callback method for parsed number of terms
`on_keyword(keyword, value)`	Callback method for parsed keyword-value pair
`on_end()`	Callback method for parsed end
`on_term(term, out)`	Callback method for parsed terms

The class `lorina::pla_writer` provides the following public member functions.

Function	Description
`on_number_of_inputs(number_of_inputs)`	Callback method for writing number of inputs
`on_number_of_outputs(number_of_outputs)`	Callback method for writing number of outputs
`on_number_of_terms(number_of_terms)`	Callback method for writing number of terms
`on_keyword(keyword, value)`	Callback method for writing keyword-value pair
`on_end()`	Callback method for writing end
`on_term(term, out)`	Callback method for writing terms

The following reader functions are available.

Function	Description
read_pla	Reader function for the PLA format.
read_pla	Reader function for PLA format.

return_code `lorina::`**`read_pla`**(std::istream &*in*, **`const`** *pla_reader* &*reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for the PLA format.

Reads PLA format from a stream and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `in`: Input stream
- `reader`: A PLA reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

return_code `lorina::`**`read_pla`**(**`const`** std::string &*filename*, **`const`** *pla_reader* &*reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for PLA format.

Reads PLA format from a file and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `filename`: Name of the file

- `reader`: A PLA reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

class pla_reader

A reader visitor for the PLA format.

Callbacks for reading the PLA format.

Subclassed by lorina::pla_pretty_printer

5.5 VERILOG format

Header: `lorina/verilog.hpp`

5.5.1 VERILOG parser

The header `<lorina/verilog.hpp>` implements methods to parse a very simplistic version of the VERILOG format.

The class *lorina::verilog_reader* provides the following public member functions.

Function	Description
`on_module_header(module_name, inouts)`	Callback method for parsed module header
`on_inputs(inputs)`	Callback method for parsed input declarations
`on_outputs(outputs)`	Callback method for parsed output declarations
`on_wires(wires)`	Callback method for parsed wire declarations
`on_assign(lhs, rhs)`	Callback method for parsed immediate assignment
`on_and(lhs, op1, op2)`	Callback method for parsed AND assignment (with 2 operands)
`on_or(lhs, op1, op2)`	Callback method for parsed OR assignment (with 2 operands)
`on_xor(lhs, op1, op2)`	Callback method for parsed XOR assignment (with 2 operands)
`on_and3(lhs, op1, op2, op3)`	Callback method for parsed AND assignment (with 3 operands)
`on_or3(lhs, op1, op2, op3)`	Callback method for parsed OR assignment (with 3 operands)
`on_xor3(lhs, op1, op2, op3)`	Callback method for parsed XOR assignment (with 3 operands)
`on_maj3(lhs, op1, op2, op3)`	Callback method for parsed MAJ3 assignment
`on_endmodule`	Callback method for parsed end of module

The following reader functions are available.

Function	Description
read_verilog	Reader function for VERILOG format.
read_verilog	Reader function for VERILOG format.

return_code `lorina::read_verilog`(**const** std::string &*filename*, **const** *verilog_reader* &*reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for VERILOG format.

Reads a simplistic VERILOG format from a file and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `filename`: Name of the file
- `reader`: A VERILOG reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

return_code `lorina::read_verilog`(std::istream &*in*, **const** *verilog_reader* &*reader*, *diagnostic_engine* **diag* = nullptr)

Reader function for VERILOG format.

Reads a simplistic VERILOG format from a stream and invokes a callback method for each parsed primitive and each detected parse error.

Return Success if parsing have been successful, or parse error if parsing have failed

Parameters

- `in`: Input stream
- `reader`: A VERILOG reader with callback methods invoked for parsed primitives
- `diag`: An optional diagnostic engine with callback methods for parse errors

class verilog_reader

A reader visitor for a simplistic VERILOG format.

Callbacks for the VERILOG format.

Subclassed by lorina::verilog_pretty_printer

CHAPTER 6

Diagnostics

The header <lorina/diagnostics.hpp> implements a mechanism to react on parse errors.

The *diagnostic engine* (class diagnostic_engine) is used to emit diagnostics when a parsing algorithm encounters mistakes. The possible error messages are specified by the implementation of the respective parsing algorithm.

The *diagnostic builder* (class diagnostic_builder) is a lightweight temporary container for diagnostic information. A builder is constructed when the method report of the diagnostic engine is called. After construction, the diagnostic algorithm may add additional information while the temporary is alive. When the temporary is destructed, the diagnostic is issued by invoking the method emit of the diagnostic engine.

Header: lorina/diagnostics.hpp

class diagnostic_builder
A builder for diagnostics.

An object that encapsulates a diagnostic. The diagnostic may take additional parameters and is finally issued at the end of its life time.

class diagnostic_engine
A diagnostic engine.

Subclassed by *lorina::silent_diagnostic_engine*

class silent_diagnostic_engine : **public** lorina::*diagnostic_engine*

CHAPTER 7

Indices and tables

Index

L