FSML++

Carsten Hartenfels   Benjamin Haßel

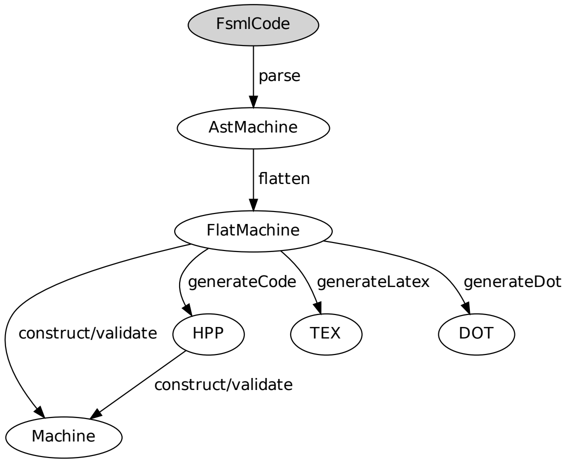
2014-01-09

Contents

1. Overview

- Language: C++
- Parser: Boost.Spirit.Qi
- Generator: Boost.Format
  - C++ header file
  - LaTeX/Ti*k*Z file
  - Graphviz/Dot file
- Testing: Google gtest

2. Program Structure



3. C++

- Intermediate-Level: No Virtual Machine or Garbage Collection
- C-Compatible
- Multi-Paradigm:
  - Procedural
  - Object-Oriented
  - Functional
  - Generic

4. Boost

- Peer-Reviewed Set of High-Quality Libraries
- 10 Libraries Adapted Into C++11
- Used by Adobe, CERN, Google etc.

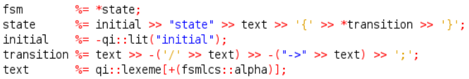
accumulators algorithm any array asio assign atomic bimap bind call_traits chrono circular_buffer compatibility compressed_pair concept config container context conversion coroutine crc date_time dynamic_bitset exception enable_if filesystem flyweight foreach format function function_types functional functional/factory functional/forward functional/hash functional/overloaded_function fusion gil geometry graph heap icl identity_type integer interprocess interval intrusive in_place_factory io iostreams iterator lambda lexical_cast local_function locale lockfree log math math/complex math/common_factor math/constants math/floating math/octonion math/special_functions math/statistical math/quaternion minmax move MPI mpl meta multi_array multi_index multiprecision numeric/conversion odeint operators optional parameter Phoenix Predef pointer polygon pool preprocessor program_options property property proto python random range ratio rational ref regex result_of scope_exit serialization signals signals2 smart_ptr statechart static_assert spirit string_algo swap system test thread timer tokenizer TR1 tribool tti tuple type_erasure type_traits typeof uBLAS units unordered utility value_initialized uuid variant wave xpressive

5. Boost.Spirit.Qi

- Recursive Descent Parser Combinator

- EBNF-Like Syntax

```
fsm : { state }∗ ;
state : initial ’state’ text ’{’ { transition }∗ ’}’ ;
initial : {’initial’}? ;
transition : text { ’/’ text }? { ’->’ text }? ’;’ ;
text : { alpha }+ ;
```



```
fsm        %= *state;
state      %= initial >> "state" >> text >> '{' >> *transition >> '}';
initial    %= -qi::lit("initial");
transition %= text >> -('/' >> text) >> -("->" >> text) >> ';';
text       %= qi::lexeme[+(fsmlcs::alpha)];
```

- DSL Embedded in C++

- No Code Generation

6. Boost.Format

- String Formatting
- Type-Safe
- Supports User-Defined Types

```
const string
generateDot(const string& name, const FlatMachine& fm)
{
    stringstream arrows;
    for (const FlatStep& fs : fm.steps)
        arrows << (format(ARROW) % fs.source % fs.target % fs.getStepText()).str();
    return (format(DOT) % name % fm.initials.at(0) % arrows.str()).str();
}
```

```
digraph %1% {
    node [shape=ellipse];
    %2% [style=filled];
%3%}
```

7. Questions and Answers

Thank You All For Listening

GitHub: `https://github.com/hartenfels/FSMLplusplus/`