
Education

- Jan 2013 – Dec 2015 **B.Sc. Mathematics**, *Université Laval*, Québec.
Sep 2011 – May 2012 **B.Sc. Software Engineering (not completed)**, *Université Laval*, Québec.

Experience

- Jun 2016 – present **Software Development Engineer**, A9.com, Palo Alto.
Member of the search infrastructure team powering Amazon's search engine.
- 2014 – present **Development of [Hana](#)**, a [Boost](#) library for metaprogramming in C++14.
Design and implementation of a library to manipulate heterogeneous sequences at compile-time and at runtime. The library introduces a new paradigm for expressing meta-computations allowing a very high level of expressiveness with little to no performance penalty.
- Dec 2014 – Jun 2016 **C++ consulting**, (finance, embedded systems).
Development of C++ libraries to retain a high level of abstraction in applications where both performance and correctness matter. Also some refactoring of existing systems to add new features and/or improve performance.
- 2014 – 2015 **GSoC student with Boost**, [Google Summer of Code](#).
Work on [Boost.Hana](#) during the summers of 2014 and 2015 as part of the Google Summer of Code program. I also received a grant from the [Boost](#) Steering Committee to continue working during the winter of 2015, which had never been done before for a GSoC student.
- Sep 2012 – Dec 2012 **Software Developer**, [Coveo Solutions](#), Québec.
Work on a MIME parser in C++. Resigned to pursue a degree in mathematics.
- May 2012 – Aug 2012 **Intern**, [Coveo Solutions](#), Québec.
 - Conception and implementation of a deadlock detection system for internal use
 - Presentations on C++ techniques and idioms to co-workers:
 - The [Boost.ConceptCheck](#) library and associated template metaprogramming techniques
 - C++11 rvalue references

Talks

- 2017 **Metaprogramming in C++14 and beyond** ([slides](#)/[video](#)), [ACCU 2017](#), Bristol.
- 2016 **Closing keynote on metaprogramming** ([slides](#)/[video](#)), [Meeting C++](#), Berlin.
Was voted the best presentation by attendees
- 2016 **Metaprogramming for the brave** ([slides](#)/[video](#)), [C++Now](#), Aspen.
- 2016 **Metaprogramming for dummies** ([slides](#)/[video](#)), [C++Now](#), Aspen.
- 2016 **Podcast about Boost.Hana** ([audio](#)), [CppCast](#).
- 2015 **Metaprogramming: a paradigm shift** ([slides](#)/[video](#)), [CppCon](#), Seattle.
- 2015 **Metaprogramming: a paradigm shift** ([slides](#)/[video](#)), [C++Now](#), Aspen.
Awards for the best presentation and the most inspiring presentation
- 2014 **Metaprogramming in C++14** ([french only slides](#)), [OpenCode XXII](#), Québec.
- 2014 **Hana: Expressive metaprogramming** ([slides](#)/[video](#)), [CppCon](#), Seattle.
- 2014 **Towards painless metaprogramming** ([slides](#)/[video](#)), [C++Now](#), Aspen.
- 2013 **A system for resource deadlock prevention** ([slides](#)/[video](#)), [C++Now](#), Aspen.
- 2013 **Deadlock detection with d2** ([slides](#)), [OpenCode XII](#), Québec.
- 2013 **Concept based overloading in C++** ([slides](#)), [OpenCode IX](#), Québec.

Personal Projects

Metabench

A simple framework for doing compile-time benchmarks

Implemented a self-contained CMake module to perform compile-time benchmarks of C++ metaprograms. Such benchmarks are very useful when writing a metaprogramming library, where the performance of the library must be measured in terms of compilation time. The module works by having the user write [ERB](#) templates that are then used to generate C++ programs. The C++ programs are compiled and various metrics such as compilation time, link time and executable size are gathered. The module generates HTML5 charts to easily visualize the metrics.

mpl11

Conception and implementation of a C++11 replacement for the Boost.MPL

Reimplemented the functionality of the Boost.MPL library using new template metaprogramming techniques made possible by C++11. Redesigned the API of the library using ideas from Haskell to make it more powerful, easier to use and to extend.

d2

Conception and implementation of a deadlock detection system in C++

Detects deadlocks that would have happened under different thread scheduling conditions by performing intrusive dynamic analysis on a non-deadlocking run of a program. Additionally, provides statistics about lock and thread usage.

joy

Implementation of a preprocessor metaprogramming library

Implemented associative sequences and other utilities for preprocessor metaprogramming on top of the [Chaos preprocessor library](#).

nstl

Conception and implementation of a generic algorithm library in pure C

Implemented a basic name mangling system and “preprocessor-based classes” using PMP techniques. Using these facilities, implemented a subset of the C++ standard library algorithms. The result is a collection of generic algorithms instantiable and usable from pure C without sacrificing type safety or performance by using traditional techniques like pointers to void.

duck

Implementation of a minimal concept-based overloading library

Implemented a subset of Boost.ConceptCheck’s concepts as metafunctions, which allows overloading based on the modeling of a concept by a type.

cisp

Implementation of a minimalist object system with the preprocessor

Created a system to manipulate complex preprocessor objects using associative sequences imbued with object semantics.

nstl-lang

Implementation of a translator for a toy language in Python

Implemented basic parsing, semantic analysis and code generation to C.

Contributions to other projects

- Contribution of the [hawick_circuits](#) algorithm to Boost.Graph
- Occasional patches to Boost (Spirit, Graph, Archive, MPL and others)
- Active on the [Boost.Dev](#) mailing list
- CMake port of the [FastPFor](#) integer compression library’s build system
- Too frequent bug reports against the [Clang](#) and [GCC](#) compilers.