COMPILER COMPILER SYSTEM

Summary

- **Compiler compiler system**: *a compiler creating compilers.*

How?

- Having a source grammar definition language specification is used as an input to the compiler compiler to generate a target parser with additional code to be compiled and linked into an executable program.
- That program is ready to compile source code and also it is ready to generate syntax controlled binary, as well as it is ready to decompile binary to original definition.
- Basically, syntax is preserved by the Parsing Model and all semantics processing can be done using Syntax-Controlled Binary API.
- Parsing Model with Binary and Runtime is a new level of automation.

Background

- The present invention relates to information technology.
- More specifically, the generation of source code from formal descriptions that can be compiled and linked, creating an executable program.
- Compiler compilers as parser and tokenizer generators have been implemented since the late 1960's.
- A developer who implements a compiler based on a conventional compiler compiler technology, is responsible for writing his own code, manually.
- An ideal compiler compiler is supposed to take an input grammar specification and generate source code in an automatic mode without any further manual procedures.

Our Compiler Compiler vs. Traditional Compiler Compiler

	Traditional	C++ CCS
Grammar	LALR or LL(1)	LL(1)
Parsing Results	Parsing Tree	Parsing Model
Grammar definition is separated from parsing results	Yes	No
Compilation Phases	Manual steps manipulating the parsing tree	Syntax-Controlled Runtime and Binary are formally connected by the parsing model. There are formal operations transforming Runtime to Binary and vise versa

Advantages Over The Traditional Compiler Compiler System

- C++ CCS generates C++ code for the target parser and additional code that could be compiled and linked into an executable program, which is a target compiler with built in operations compiling source code into Binary and decompiling Binary back into source definition.
- This is important because it is done completely in an automated mode unlike traditional compiler compiler systems.
- Syntax-Controlled Binary is a multi platform protocol with a provided API for implementing subsequent compilation phases in accordance with Parsing Model

What Is So Special?

- Our compiler compiler system, with a design paradigm different from traditional compiler compiler systems, is defined by the Patent (last slide).
- First, instead of having a parsing tree, compiler compiler runtime and binary are designed according to the compiler compiler parsing model.
- Second, any semantics processing is totally separated from the syntax processing.
- Third, the whole compilation process is defined as syntax processing and semantics processing followed by syntax processing performed under compiler compiler management supervision.
- Fourth, syntax processing has two phases: building the compiler compiler runtime, and converting the compiler compiler runtime into compiler compiler binary with available options to convert back the compiler compiler binary to compiler compiler runtime.
- Fifth, compiler compiler runtime and binary syntax-controlled APIs are defined in terms of syntax.
- Sixth, there are formal methods de-compiling the compiler compiler runtime and/or binary into original program text accordingly to the syntax.
- Seventh, compiler compiler runtime and binary with their syntax-controlled APIs serve as a multiplatform for obfuscation, security, binary files processing, and program-to-program communication.

Parsing Model

- Parsing Model considers the parsing results to be represented in a form of entities such as:
  - Context - an entity representing parsing results dedicated for compiled elements under a single scope having common collection of Name instances.
  - Name - an entity representing language names: identifiers and reserved key words.
  - Symbol - represents grammar terminal or non-terminal symbol.
  - Rule - represents the grammar rule to be processed by parser with related parsing results.

- Their relationships:
  - All parsing results are maintained in a form of collection of Context instances.
  - There are two relationships between Context and Name: namesByName and namesByNumber providing Name look up by name and by number.
  - In other words, within a given Context, each Name is identified by name and by number.
  - Each Context maintains a collection of Symbol instances.
  - Each Symbol maintains a collection of Rule instances.

C++ CCS Parsing Model Implementations

- Syntax- Controlled Runtime
  - Optimized to be used by a parser manipulating with Syntax-Controlled Runtime Parsing Model entities and their relationships having full set of read/write operations.
- Syntax- Controlled Binary
  - Optimized to be used by any other application in a read-only mode with Syntax-Controlled Binary Parsing Model entities and their relationships.

Obfuscation

- A method that transforms data in a way that the original data elements and their relationships become modified to hide their original content, i.e., *to be obfuscated*
- The key component for obfuscation is to provide an efficient algorithm for removing the introduced new items and new relationships to completely restore the original content, i.e., *providing de-obfuscation*

NP-Complete Tasks for Grammar Recognition and Obfuscation

- It is known that grammar recognition tasks are NP-complete.
- In simple words, NP- complete means that the task would run almost forever.
- Having Syntax- Controlled Binary, the task of the grammar recognition is NP- complete.
- Having Syntax- Controlled Runtime, obfuscated one way or another, with subsequent formal conversion into Syntax- Controlled Binary would make grammar recognition tasks even harder.
- Implementing obfuscation/de-obfuscation algorithms for Syntax- Controlled Runtime and Binary is an advanced, patent protected way of transmitting data with built-in security features and content management.

US Patent 8,464, 232

- Aleksandr F Urakhchin. *Compiler compiler system with syntax-controlled runtime and binary application programming interfaces.* June 11 2013. US Patent 8,464, 232.
- https://www.google.com/patents/US8464232