

# **Project Work**

## **Translation of Java-Embedded Database Queries**

---

by

Kaichuan Wen

Information and Media Technologies

20729335

supervised by

Prof. Dr. Möller

Dr. Garcia

STS, TUHH

# Agenda

---

- Introduction
- Roadmap
- LINQ Grammar
- CST Metamodel
- Transformation
- Next Step: Plug-in for Java Compiler
- Conclusions

# Introduction

---

- Language Integrated Query (LINQ)
  - Data access syntax in source code
  - Unified syntax over different data sources
  - Compile-time type checking and inferring
  - Only supported in .NET languages at the moment
  
- Goal: supporting LINQ in Java

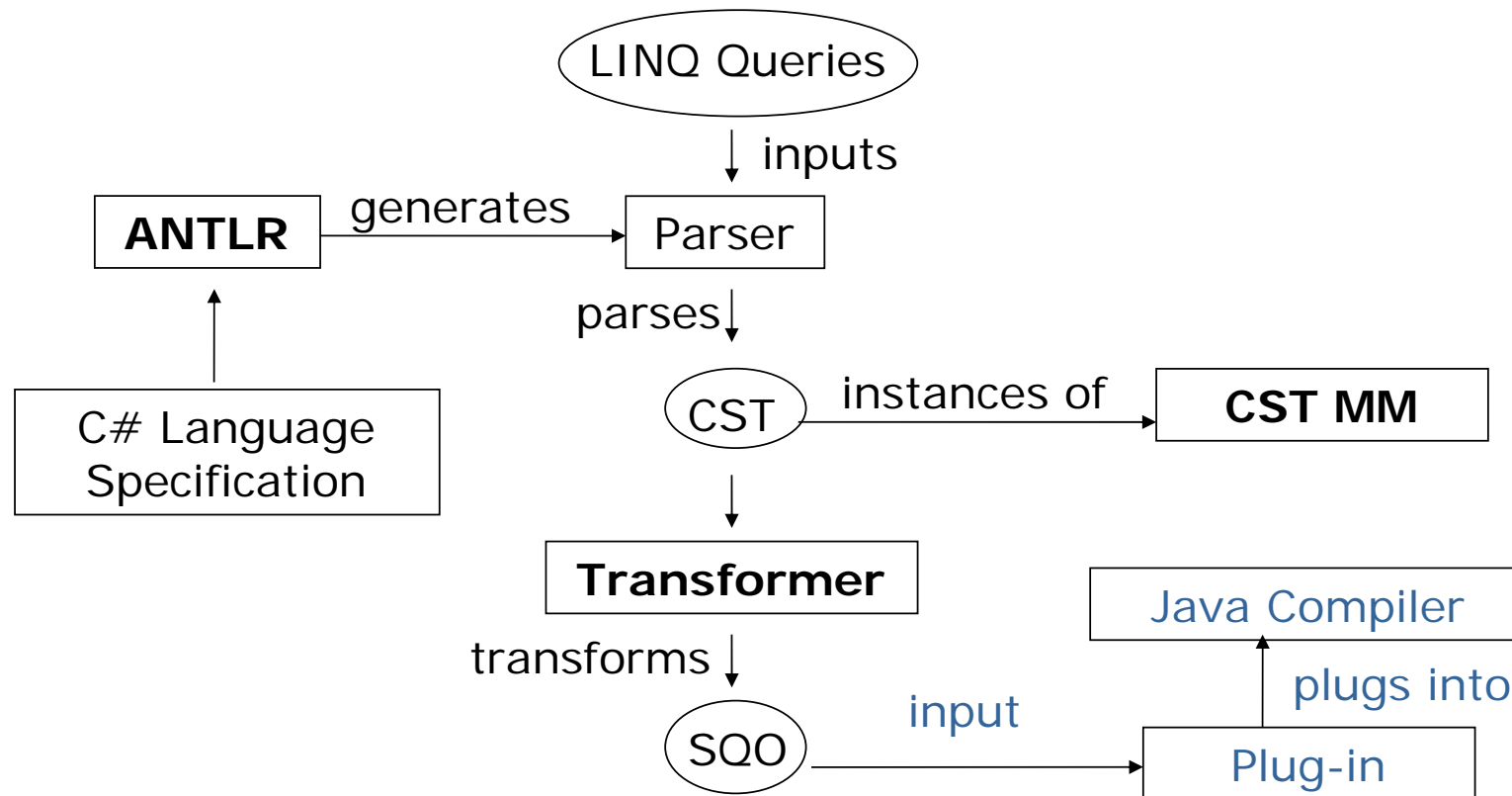
# Introduction

---

- Abbreviations
  - ANTLR – **AN**other **T**ool for **L**anguage **R**ecognition  
Tool to generate parser for LINQ syntax
  - CST MM — Metamodel for Concrete Syntax Trees
  - SQO — Standard Query Operations

# Roadmap

---



# LINQ Grammar

---

- Two ways to formulate queries:
  - Textual query expression
  - SQO (chain of method invocations)
- Textual query expression
  - High level language, natural, flexible
  - Example:

```
from c in Customers where c.age > 20
```

```
from o in Orders where c.customerID = o.customerID
```

```
select new {c.name, o.itemName }
```

# LINQ Grammar

---

- SQO (series of method invocations)
  - Low level language
  - Allows mapping to different DB providers
  - Example:

Customers.**Where**(c => c.age >20)

.**SelectMany**(c => Orders, ( c, o ) => **new** { c = c, o = o } )

.**Where**(TransparentID\_0 =>  
(TransparentID\_0.c .customerID )=(TransparentID\_0.o.customerID) )

.**Select**(TransparentID\_0=>new{ TransparentID\_0.c.name ,  
TransparentID\_0.o.itemName } )

# LINQ Grammar

---

- Production rules in ANTLR for each grammar element
- ANTLR parser matches input tokens
- When a series of tokens is matched, corresponding *semantic action* is taken
  - Semantic action: ideal place to build CST

# CST Metamodel

---

- CST Metamodel
  - Classes representing LINQ grammar elements and expressions
  - Associations and multiplicities in between
- CST
  - Tree representing particular LINQ query
  - Constructed within ANTLR semantic actions by calls to EMF factory methods
  - Visitor needed to walk those CSTs

# Transformation

---

- Textual query expression → SQO
  - High level language → low level language
  - Guaranteed to succeed
- 18 transformation rules apply
  - Cover all possibilities
  - Repeatedly and recursively apply to sub trees
- Gradual reduction of clauses

# Transformation

---

- Cloning-visitors for tree manipulation
  - Copy sub trees “as is”
  - No cross-referencing between original and clone
  - One sub class for each transformation rule
- Resolution of transparent identifiers
  - “Scope”: no duplicate variable names allowed in LINQ for readability and unambiguity
  - “Fresh name”

# Transformation

---

- Final result: only series of method calls
  - These methods have fix signatures
  - Can be implemented by different DB providers
  - In terms of .NET Framework: implementations of interface *IQueryable<T>*

# Next Step: Plug-in for Java Compiler

---

- Annotation
  - Metadata in source code
- JSR 269: *Pluggable Annotation Processing API*
  - Custom annotation processors plug-in to compiler
  - Query processors add here
- Other issues concerning compiler plug-in:
  - type checking
  - type inference
  - ...

# Conclusions

---

- Similar techniques applicable to other query languages
  - Common steps:
    - Syntax recognition (with ANTLR or other tools)
    - CST Metamodel
    - Java Compiler plug-in
  - Others are language-specific tasks
    - In our project: Transformation into SQO

---

**Thank you for your attention!**

**Your questions are welcome.**