# CODE GENERATION OF VDM++ CONCURRENCY

Georgios Kanakis, Peter Gorm Larsen, Peter W. V. Tran-Jørgensen

AARHUS
UNIVERSITET
SCIENCE AND TECHNOLOGY

# AGENDA

Introduction

Code Generation Platform

Generating concurrency

Mapping Synchronization

Conclusion

Future Vision of Overture

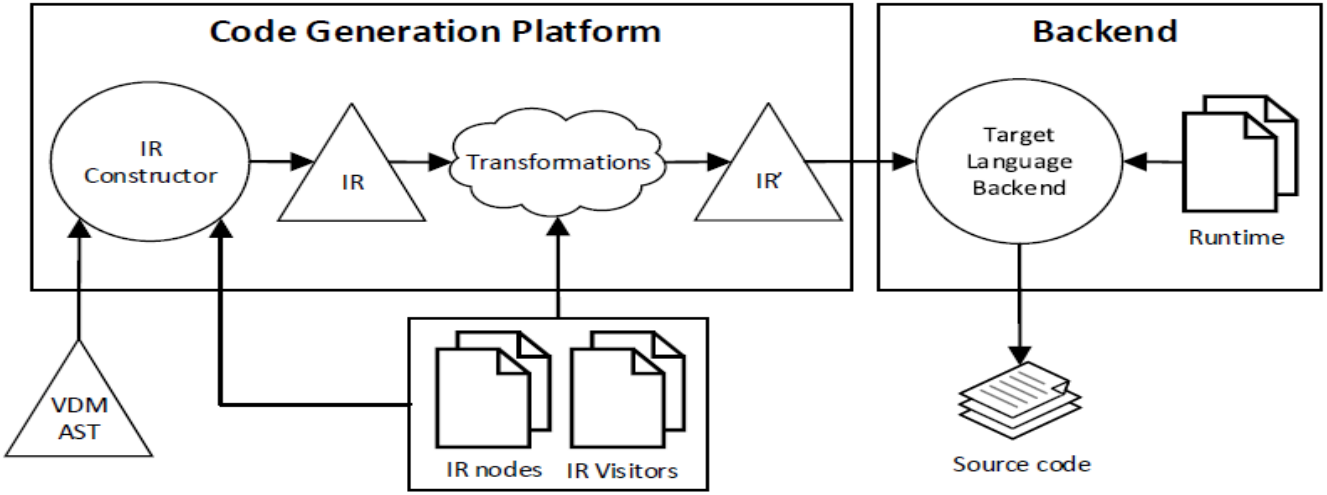
Q&A

# MOTIVATION

- Expanding the Code Generation Platform
- Concurrency is becoming more and more important

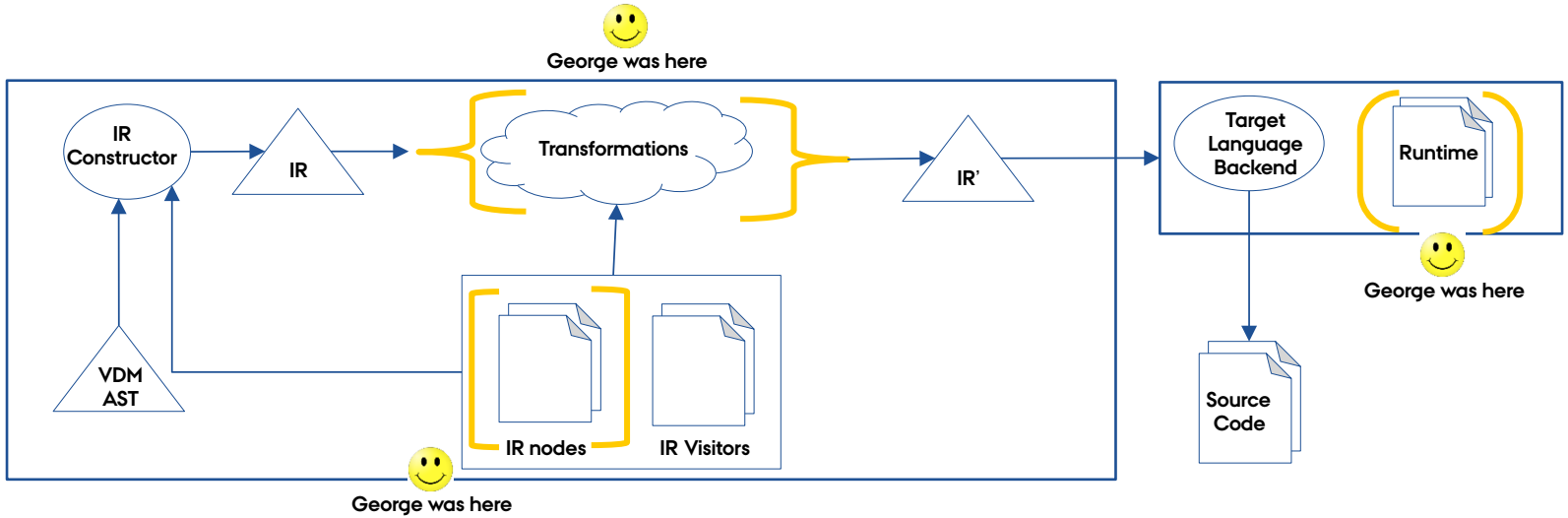
# CODE GENERATION PLATFORM

## CG Platform Overview



# CODE GENERATION PLATFORM

## CG Platform Extension



# VDM++ VS JAVA CONCURRENCY

**VDM++ Concurrency**

- Threads
- History counters
- Permission Predicates
- Mutex

≠

**Java Concurrency**

- Threads
- Monitors
- Locks

# GENERATING CONCURRENCY

The Main Solution

- Simple thread generation
- Managing history counters
- Mechanism for evaluating permission predicates

# THREADS

Threads

- VDM thread maps easily to Java.
- VDM thread has a slightly different life cycle

# HISTORY COUNTERS

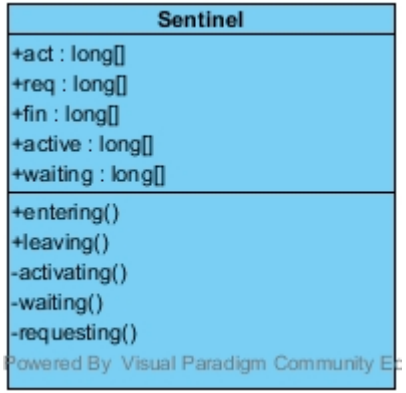
Managing History Counters

- Sentinel Class
  - Keeps history counters
  - Manages history counters
  - Forces evaluation of permission predicates

# HISTORY COUNTERS

Mapping History Counters

- Are kept as arrays in the Sentinel class
- Each method is represented with one index of the array
- Overloaded operations are given only one index



# HISTORY COUNTERS

VDM++

```
class test_class_pptypes

instance variables
x:int:=0;

operations
public foo : () ==> int
foo () == return 1;

private bar : () ==> bool
bar () == return false;

end test_class_pp
```

Java

```
public class test_class_pp implements EvaluatePP {
 public Number foo() {
   sentinel.entering(sentinel.foo);
   try {
     return 1L;
   } finally {
     sentinel.leaving(sentinel.foo);}
 }
 private Boolean bar() {
   sentinel.entering(sentinel.bar);
   try {
     return false;
   } finally {
     sentinel.leaving(sentinel.bar);}
 }
 public static class test_class_pp_sentinel
 extends Sentinel {
   public static final int foo = 0;
   public static final int bar = 1;
   public test_class_pp_sentinel(final valuatePP,instance)
   {
      init(instance, function_sum); }
 }
}
```

# SENTINEL.ENTERING

- Increments relevant history counters
- Also requests the evaluation of permission predicates

# PERMISSION PREDICATES

Evaluating Permission Predicates

- `EvaluatePP` interface
  - Contains only one method `evaluatePP(...)`
  - Generated for all user defined classes

# PERMISSION PREDICATES

Mapping Permission Predicates

- Permission Predicates become `evaluatePP` method
- Each permission predicate becomes a branch in the `evaluatePP`

# PERMISSION PREDICATES

VDM++

```
class test_class_pptypes

instance variables
   x:int:=0;
operations
 public foo : () ==> int
 foo () == return 1;

 private bar : () ==> bool
 bar () == return false;

sync
 per foo => #act(foo) < 3;
 per bar => x < 5;

end test_class_pp
```

Java

```
public class test_class_pp implements EvaluatePP {
 private volatile Number x = 0L;
 public volatile Sentinel sentinel;
 public Number foo() {
…}
 private Boolean bar() {
…}
 public Boolean evaluatePP(final Number fnr) {
    if (Utils.equals(fnr, 0L)) {
          return (sentinel.act[sentinel.foo] < 3L)
    } else if (Utils.equals(fnr, 1L)) {
          return (x.longValue() < 5L)
    }
    else {
            return true;
    }    }
```

# MUTEX

```
mutex(foo, bar);
```

Transformed

```
per  foo => #active(bar) = 0;
```

```
per bar => #active(foo) = 0;
```

# EVALUATION

- Tested on examples:
  - POP3
  - Concurrent Factorial
  - Shared Variable Buffers

- **It is in the Tool!**

# CONCLUSION – FUTURE WORK

- Code generation of VDM ++ to Java concurrency
  - Permission predicates, history counters, threads…
- Validated on various models

- In the future…
  - Code generate concurrency with static operations
  - Investigate portability of concurrency codegen transformations

# FUTURE OF OVERTURE

- 1 Year
  - Clear its features, providing the most useful
- 5 Years
  - Expand its developer community and its user base
  - Be part of important projects in the industry
- 10 years
  - Being a well established industry tool

# Q&A

## QUESTIONS?

***ΛΕΓΕ ΕΙΔΩΣ – ΝΕΩΤΕΡΟΝ ΔΙΔΑΣΚΕ – ΣΟΦΙΑΝ ΖΗΤΕΙ***

Teach what you know – Teach the younger – Seek wisdom

Δελφικά Παραγγελματα - Delphic Maxims

AARHUS
UNIVERSITET