AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

JUNE 21 2014

A CODE GENERATION PLATFORM FOR VDM

PETER W. V. JØRGENSEN
MORTEN LARSEN
LUIS D. COUTO

12TH OVERTURE WORKSHOP
AT NEWCASTLE UNIVERSITY

AGENDA

- Motivation
- Intermediate Representations and Transformations
- The Code Generation **(CG)** Platform
- CG VDM++ to Java and C++
- Future Plans and Conclusion

MOTIVATION

- Efficient transitioning to the modeling phase
- Many popular target languages to support
- Similarity of target languages
- Some constructs are non-trivial to CG
- Reuse CG rules for other backends

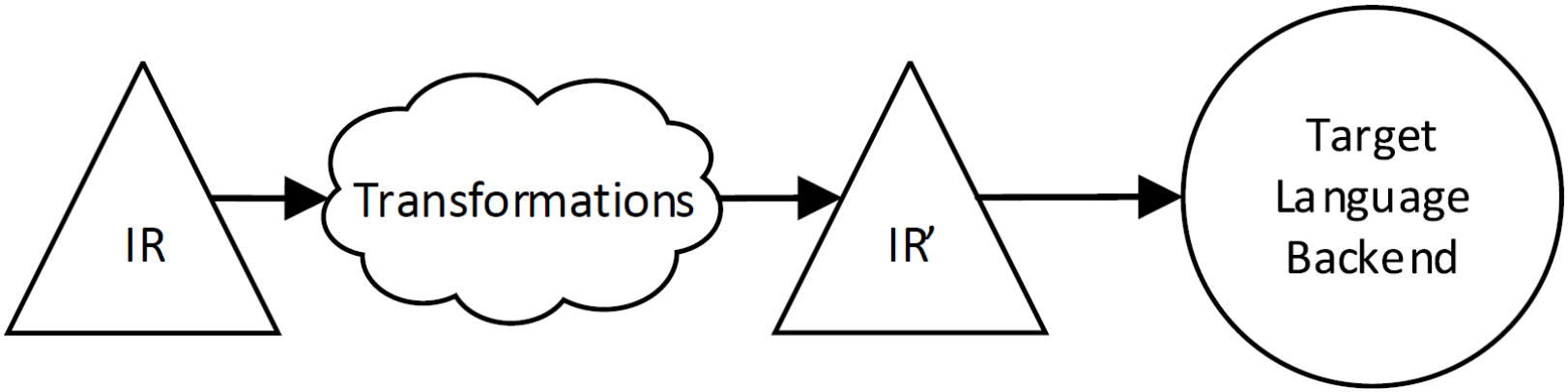
```
{x|x in set S & pred(x)}
```

INTERMEDIATE REPRESENTATION (IR)

- Represents the CG VDM model
- Independent of the source and target language
- Example: Unify VDM functions and operations
- IR constructs may not map 1-1 into the target lang.
- Idea: Transform the IR to obtain 1-1 mappings

TRANSFORMATIONS

› Change the IR to make it easier to CG

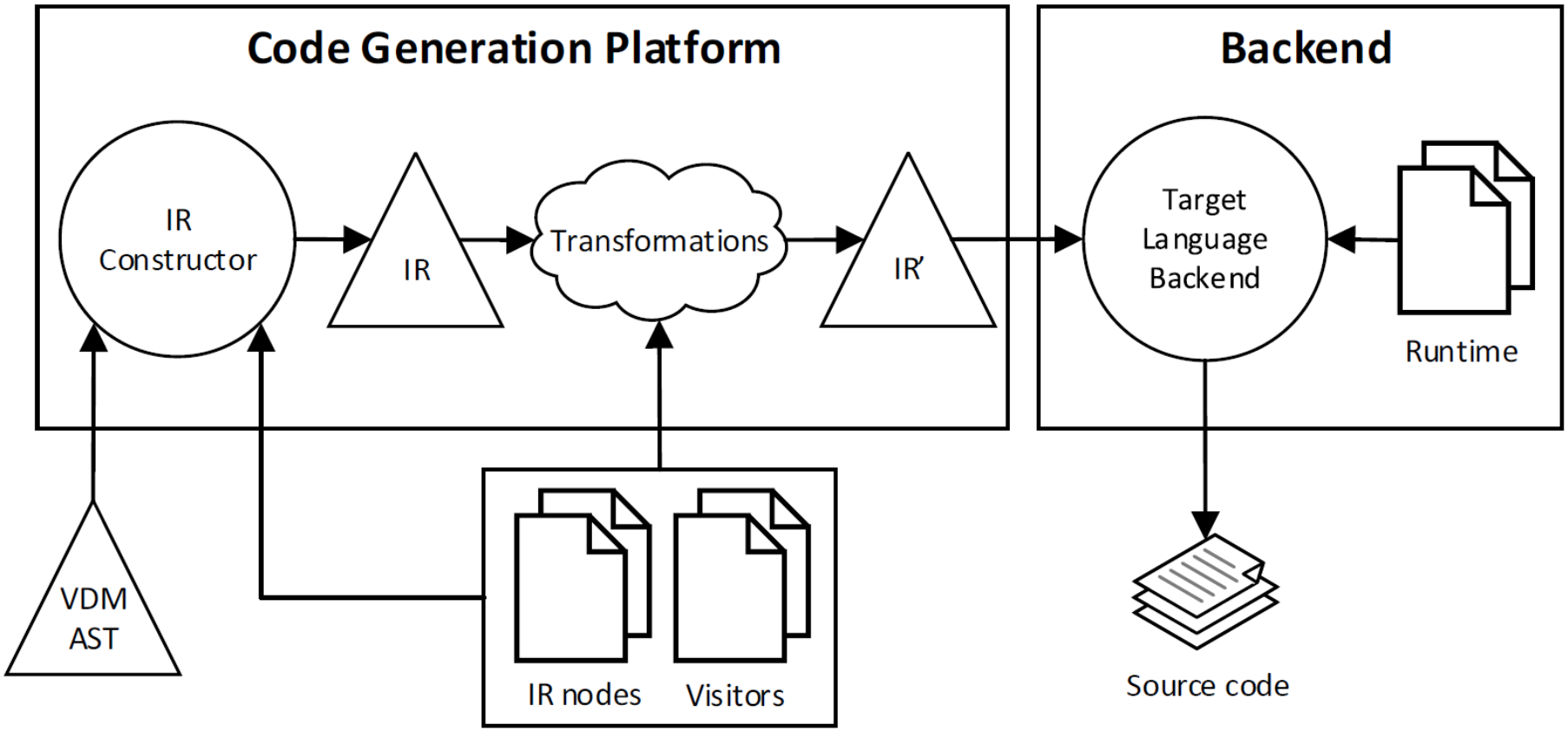


TRANSFORMATIONS

Properties of a transformation

- Transparent to the modeler
- Works for all backends (reuse)
- Simplifies the backend implementation
- Different backends use different transformations

THE CODE GENERATION PLATFORM



EXAMPLE

- The set comprehension is a functional construct
- How does it map into an imperative language?

```
public f : () -> set of nat
f () ==
let a = {x | x in set S & pred(x)}
in
  g(a,a);
```

VDM++ TO JAVA

› Computing a set comprehension in Java

```
public static VDMSet f() {
    VDMSet setCompResult_1 = SetUtil.set();
    VDMSet set_1 = S.clone();
    for (Iterator iterator_1 = set_1.iterator(); iterator_1.hasNext();) {
        Number x = ((Number) iterator_1.next());
        if (pred(x)) {
            setCompResult_1 = SetUtil.union( setCompResult_1, SetUtil.set(x));
        }
    }
    VDMSet a = setCompResult_1;
    return g(a, a);
}
```

VDM++ TO JAVA

› Obtaining the effect of copy-by-value semantics

VDM:

```
public op : () ==> nat
op () ==
(
 dcl v1 : Vector2D := mk_Vector2D(1,2);
 dcl v2 : Vector2D := v1;
 v1.x := 2;
 return v2.x;
)
```

Java:

```
public Number op()
{
  Vector2D v1 = new Vector2D(1L, 2L);
  Vector2D v2 = v1.clone();
  v1.x = 2L;
  return v2.x;
}
```

VDM++ TO C++

- Reuse of transformations
- Memory management is handled manually
- Object references are CG as shared pointers
- Records are CG as classes

FUTURE PLANS

- Adding support for new (and different) languages
- Atomic (finer-grained) transformations
- Student projects: Concurrency & RT aspects

CONCLUSION

A Code Generation Platform:

- Simplifies the backend implementation
- Easier support for multiple target languages
- Transformations at the IR level promote reuse
- A platform requires maintenance