Code-generating VDM for Embedded Devices

Victor Bandur **Peter W. V. Tran-Jørgensen**
Miran Hasanagić Kenneth Lausdahl

AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

15th Overture workshop
Newcastle, UK – September 15

Agenda

Introduction

Translation

Conclusion and future plans

Agenda

Why another code-generator?

- Existing VDM code-generators
  - Suitable for resource-rich hardware platforms
  - Target Java, C#, Smalltalk and C++ etc.
- Resource-constrained microcontrollers
  - Limited processing power and memory
  - Often only have C compilers available

Why another code-generator?

- Existing VDM code-generators
  - Suitable for resource-rich hardware platforms
  - Target Java, C#, Smalltalk and C++ etc.
- Resource-constrained microcontrollers
  - Limited processing power and memory
  - Often only have C compilers available

VDM2C context

- Developed in INTO-CPS to support:
  - Implementation of VDM-RT models in C
  - FMI-based co-simulation of VDM-RT models
- Translation assesment
  - Validated through comprehensive testing
  - Industrial INTO-CPS pilot/case studies

VDM2C context

- Developed in INTO-CPS to support:
    - Implementation of VDM-RT models in C
    - FMI-based co-simulation of VDM-RT models
- Translation assesment
    - Validated through comprehensive testing
    - Industrial INTO-CPS pilot/case studies

Agenda

Translating VDM to C

- VDM2C feature highlights
  - *Runtime* implements VDM types/operators
  - `TVP` stores type information
  - User-guided garbage collection
  - `VdmModelFeatures.h` to limit runtime size
  - OO features handled using VTables
  - Supports distribution (VDM-RT)
- Limitations
  - No pattern matching
  - Limited support for concurrency

Translating VDM to C

- VDM2C feature highlights
  - *Runtime* implements VDM types/operators
  - `TVP` stores type information
  - User-guided garbage collection
  - `VdmModelFeatures.h` to limit runtime size
  - OO features handled using VTables
  - Supports distribution (VDM-RT)
- Limitations
  - No pattern matching
  - Limited support for concurrency

Translating VDM to C

- VDM2C feature highlights
  - *Runtime* implements VDM types/operators
  - `TVP` stores type information
  - User-guided garbage collection
  - `VdmModelFeatures.h` to limit runtime size
  - OO features handled using VTables
  - Supports distribution (VDM-RT)
- Limitations
  - No pattern matching
  - Limited support for concurrency

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

```
static  TVP _Z2opE2XCN(ACLASS this, TVP x){
 if ( toBool(isNat(x)) )
  return CALL_FUNC_PTR(A, A, this,
                               CLASS_A__Z1gEV);
 else
  return CALL_FUNC_PTR(A, A, this,
                                   CLASS_A__Z1hEV);
}
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

```
static  TVP _Z2opE2XCN(ACLASS this, TVP x){
 if ( toBool(isNat(x)) )
  return CALL_FUNC_PTR(A, A, this,
                           CLASS_A__Z1gEV);
 else
  return CALL_FUNC_PTR(A, A, this,
                               CLASS_A__Z1hEV);
}
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

```
static  TVP _Z2opE2XCN(ACLASS this, TVP x){
 if ( toBool(isNat(x)) )
  return CALL_FUNC_PTR(A, A, this,
                              CLASS_A__Z1gEV);
 else
  return CALL_FUNC_PTR(A, A, this,
                                  CLASS_A__Z1hEV);
}
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

```
static  TVP _Z2opE2XCN(ACLASS this, TVP x){
 if ( toBool(isNat(x)) )
  return CALL_FUNC_PTR(A, A, this,
                              CLASS_A__Z1gEV);
 else
  return CALL_FUNC_PTR(A, A, this,
                                  CLASS_A__Z1hEV);
}
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

```
static  TVP _Z2opE2XCN(ACLASS this, TVP x){
 if ( toBool(isNat(x)) )
  return CALL_FUNC_PTR(A, A, this,
                                CLASS_A__Z1gEV);
 else
  return CALL_FUNC_PTR(A, A, this,
                                    CLASS_A__Z1hEV);
}
```

```
TVP a_instance = _Z1AEV(NULL);
TVP arg = newInt(42)
TVP res = CALL_FUNC(A, A, a_instance,
                          CLASS_A__Z2opE2XCN,
                          arg)
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

```
static  TVP _Z2opE2XCN(ACLASS this, TVP x){
 if ( toBool(isNat(x)) )
  return CALL_FUNC_PTR(A, A, this,
                           CLASS_A__Z1gEV);
 else
  return CALL_FUNC_PTR(A, A, this,
                               CLASS_A__Z1hEV);
}
```

```
TVP a_instance = _Z1AEV(NULL);
TVP arg = newInt(42)
TVP res = CALL_FUNC(A, A, a_instance,
                          CLASS_A__Z2opE2XCN,
                          arg)
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

```
static  TVP _Z2opE2XCN(ACLASS this, TVP x){
 if ( toBool(isNat(x)) )
  return CALL_FUNC_PTR(A, A, this,
                          CLASS_A__Z1gEV);
 else
  return CALL_FUNC_PTR(A, A, this,
                              CLASS_A__Z1hEV);
}
```

```
TVP a_instance = _Z1AEV(NULL);
TVP arg = newInt(42)
TVP res = CALL_FUNC(A, A, a_instance,
                          CLASS_A__Z2opE2XCN,
                          arg)
```

Translation example

```
class A

operations

op:nat|char ==> bool
op (x) ==
  if is_nat(x)
  then
    g()
  else
    h();
...
end A
```

```
static  TVP _Z2opE2XCN(ACLASS this, TVP x){
 if ( toBool(isNat(x)) )
  return CALL_FUNC_PTR(A, A, this,
                            CLASS_A__Z1gEV);
 else
  return CALL_FUNC_PTR(A, A, this,
                                CLASS_A__Z1hEV);
}
```

```
TVP a_instance = _Z1AEV(NULL);
TVP arg = newInt(42)
TVP res = CALL_FUNC(A, A, a_instance,
                          CLASS_A__Z2opE2XCN,
                          arg)
```

Type information

```
#define TVP struct TypedValue*

struct TypedValue {
  vdmtype type;
  TypedValueType value;
  ...
};
```

Type information

```
#define TVP struct TypedValue*

struct TypedValue {
  vdmtype type;
  TypedValueType value;
  ...
};
```

Type information

```
#define TVP struct TypedValue*

struct TypedValue {
  vdmtype type;
  TypedValueType value;
  ...
};
```

Distribution: remote calls

```
#define DIST_CALL(sTy, bTy, obj, supID ,nrArgs ,
    funID, args...)
     ((obj->type==VDM_CLASS) ?
     CALL_FUNC(sTy, bTy, obj, funID, ## args) :
     send_bus(obj->value.intVal, funID, supID,
          nrArgs, ## args))
```

Distribution: remote calls

```
#define DIST_CALL(sTy, bTy, obj, supID ,nrArgs ,
    funID, args...)
     ((obj->type==VDM_CLASS) ?
     CALL_FUNC(sTy, bTy, obj, funID, ## args) :
     send_bus(obj->value.intVal, funID, supID,
         nrArgs, ## args))
```

Distribution: remote calls

```
#define DIST_CALL(sTy, bTy, obj, supID ,nrArgs ,
    funID, args...)
     ((obj->type==VDM_CLASS) ?
     CALL_FUNC(sTy, bTy, obj, funID, ## args) :
     send_bus(obj->value.intVal, funID, supID,
          nrArgs, ## args))
```

Distribution: remote calls

```
#define DIST_CALL(sTy, bTy, obj, supID ,nrArgs ,
    funID, args...)
     ((obj->type==VDM_CLASS) ?
     CALL_FUNC(sTy, bTy, obj, funID, ## args) :
     send_bus(obj->value.intVal, funID, supID,
          nrArgs, ## args))
```

Agenda

Conclusion and future plans

- VDM-to-C translation for embedded devices
  - Uses garbage-collection
  - Type information is captured using `TVP`
  - Supports OO and distribution (VDM-RT)
- Future plans
  - Extending VDM coveage
  - Compare to other generators

Conclusion and future plans

- VDM-to-C translation for embedded devices
  - Uses garbage-collection
  - Type information is captured using `TVP`
  - Supports OO and distribution (VDM-RT)
- Future plans
  - Extending VDM coveage
  - Compare to other generators

Thank you

Find us on Github:

`https://github.com/overturetool/vdm2c`