VDM2Dafny:

An Automated Translation Tool for VDM-SL to Dafny

Integrated Master's dissertation project presentation

Adam Winstanley – Leo Freitas

Aims and Objectives

- Establish translation environment from Dafny to VDM.
    - Part of a master’s dissertation within 2 months time constraints.
- Cover relevant subset of VDM-SL for translation to Dafny
    - Targets parts of VDM-SL which overlap with Dafny implementations.
    - Dafny VDM preamble library.
- The approach to translation aims to be extensible and modifiable.

Background: Languages

- Both target languages are:
  - Specification languages.
  - Formally defined.
- Possible to extend current translations with the object-oriented features of VDM++, as Dafny supports this paradigm.
- Dafny has:
  - Native support of the .NET platform with compilation to .dll libraries
  - Powerful static program verifiers (e.g. SAT/SMT solvers).
  - Natively supported translation capabilities to other languages.

Background: Translation Tools

- Existing translation tools:
  - VDM2UML
  - VDM2C
  - VDM2Java
  - **VDM2Isa**
- Translation through as an VDMJ compiler plugin

- Works within tools for development in VDM.

Why Dafny?

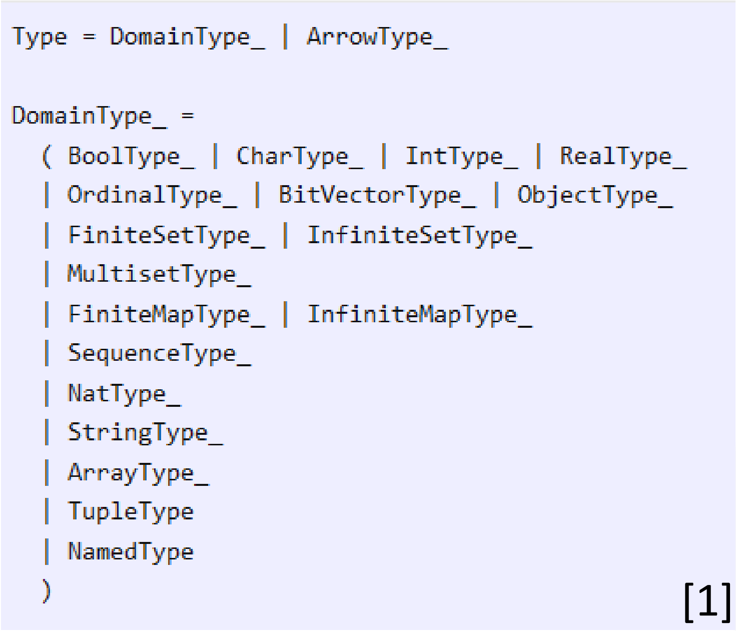
- Built-in compatibility with the .NET platform.
- Automatically compiled C# libraries, which extends the number of compatible modern languages that VDM can access.
- Similarities between the languages. There are a few missing features in Dafny.
- Additional features in Dafny:
  - Built on top of the Boogie platform.
  - User declared lemmas for improved proof automation.
  - Natively supports classes in specification.

Required Technologies

- The tool operates as a plugin for the VDMJ compiler.
- VDM2Isa from the VDMToolkit was used as a basis for:
  - Hooking into VDMJ.
  - Structure of the project.
  - Command registration code.
- String templates are used to produce translations.

Grammar Rules

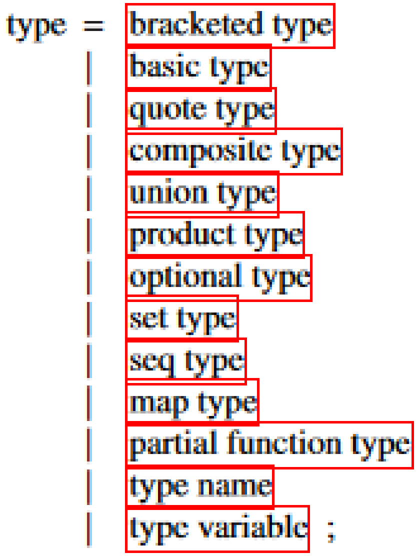
- Formal rules by which a language is defined.
- Generally use Backus-Naur Form notation to describe the syntax.
- These rules are available for both Dafny and VDM.
- Translation strategies were devised with these rules in mind.



```
Type = DomainType_ | ArrowType_

DomainType_ =
  ( BoolType_ | CharType_ | IntType_ | RealType_
  | OrdinalType_ | BitVectorType_ | ObjectType_
  | FiniteSetType_ | InfiniteSetType_
  | MultisetType_
  | FiniteMapType_ | InfiniteMapType_
  | SequenceType_
  | NatType_
  | StringType_
  | ArrayType_
  | TupleType
  | NamedType
  )
```

[1]



```
type = bracketed type
     | basic type
     | quote type
     | composite type
     | union type
     | product type
     | optional type
     | set type
     | seq type
     | map type
     | partial function type
     | type name
     | type variable ;
```

[2]

[1]: *Dafny Reference Manual*. Available at: https://dafny.org/latest/DafnyRef/DafnyRef#g-type

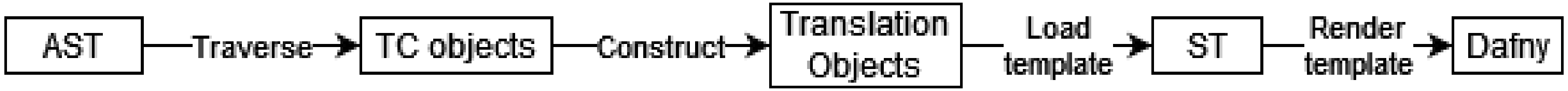
[2]: *VDM Language Manual 10.* Available at: http://lausdahl.github.io/overturetool.github.io/files/VDM10_lang_man.pdf

String Templates

- Originally a part of Antlr.
- Separated as a standalone tool.
- Allows for various useful expressions in a template to simplify writing translations.
- Allows for a grammar-inspired translation strategy.
- Works as a simple markup language with very limited expressions which are rendered through the Java package.

Translating VDM-SL: Overview

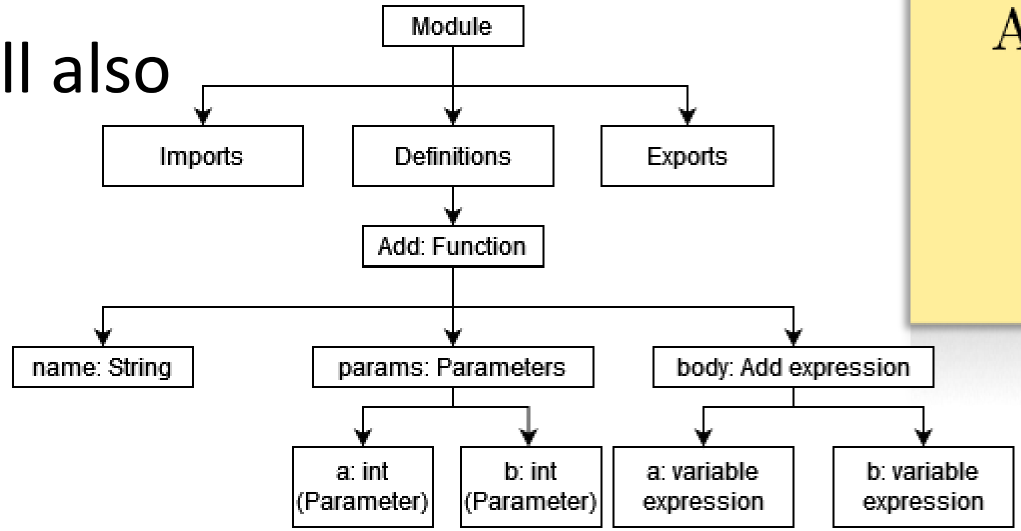
- Use of the VDMJ compiler classes to produce an AST structure.
- Type checker objects are used to produce translation objects.
- Translation objects handle template loading and rendering to string.



Translating VDM-SL: Visualising translation

- Each module produces an individual tree structure.
- Each node has a *translate()* method to produce the Dafny code.
- Calling the *translate()* method will also translate all the children of the node.

```
module MyModule
definitions
functions
Add: int * int -> int
Add(a, b) == a + b
end Test
```

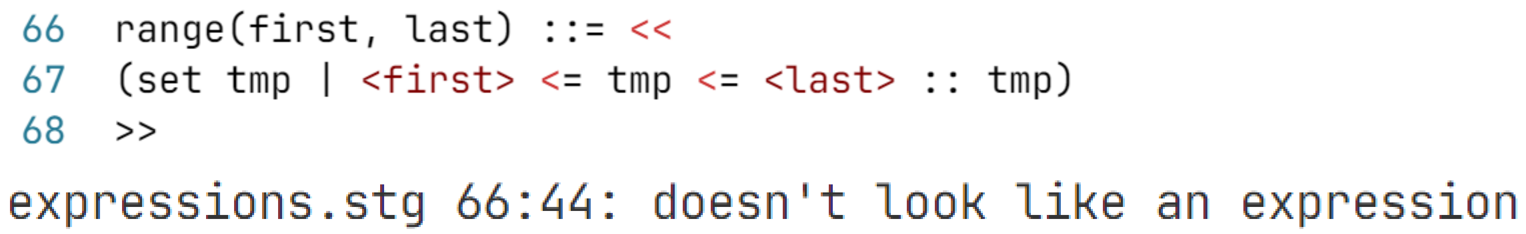


Only non-null fields are shown

Intro
Background
Approach
Evaluation

Translating VDM-SL: Problems with ST

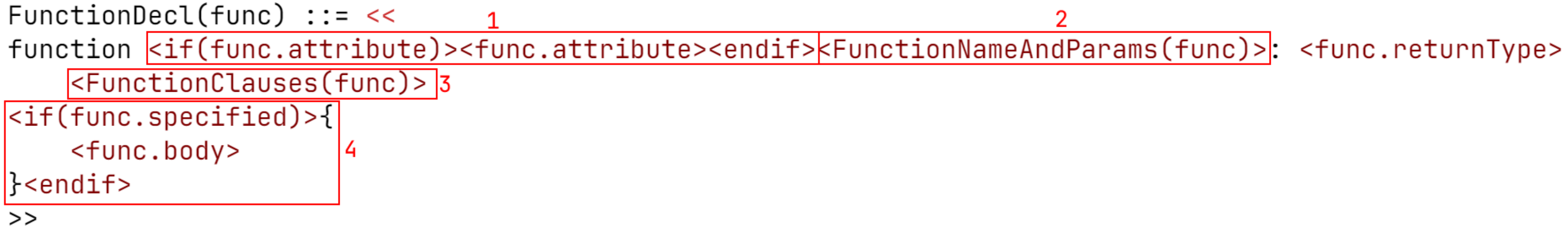
- Certain translations do not require the use of a template to translate.
- Using templates for every simple translation would considerably bloat the project.
  - This is where the template would be incredibly simple, I.E. adding quotes to a string literal.
  - Using reusable Java code for these avoids unnecessary loading and unloading of templates.
- ST can be poor at explaining errors.



```
66  range(first, last) ::= <<
67  (set tmp | <first> <= tmp <= <last> :: tmp)
68  >>
```

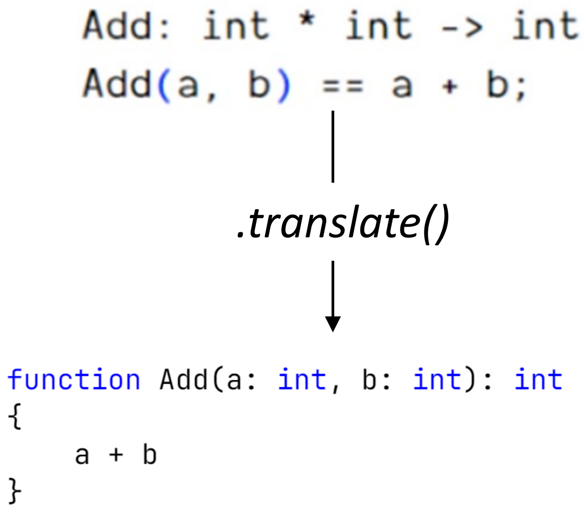
```
expressions.stg 66:44: doesn't look like an expression
```

Translating VDM-SL: Building up Dafny code



```
FunctionDecl(func) ::= <<                1                                    2
function <if(func.attribute)><func.attribute><endif><FunctionNameAndParams(func)>: <func.returnType>
    <FunctionClauses(func)> 3
<if(func.specified)>{
    <func.body>           4
}<endif>
>>
```

1. Checks and adds any required Dafny attributes to the function.
2. Calls a separate template to handle the function’s name and parameters.
3. Calls a separate template to handle the function’s requires, ensures, and decreases clauses.
4. Checks if the function has been specified, and adds the body expression if it has.



```
Add: int * int -> int
Add(a, b) == a + b;
```

.translate()

```
function Add(a: int, b: int): int
{
    a + b
}
```

Production of the Helper Module

- Implements nat1, seq1, set1, and optional types into Dafny.
- Provides function implementations for:
  - Map overrides/unions.
  - Domain/range restriction/exclusion.
  - Distributed sequence concatenation.
  - Function iteration and compositions.
  - Exists1, and Iota functions.
- Lemmas to aid in type ordering proofs.
  - Written into Dafny from the VDM language manual.

Demonstration of the tool

What hasn’t been done

- State definitions.
  - Attempts have been made, but a global program state is difficult in Dafny.
- Iota expressions.
  - Possible to translate but is difficult for the static provers to discharge and can fail when in use.
- Assignments, cases expressions.
  - Some patterns are not supported in Dafny assignments/match expressions.
- Field/make expressions for union types.
  - The current point of translation loses some important context for translating these properly to Dafny with the current strategies.
- Sequence comprehension.
  - Impossible to translate due to a difference in how VDM-SL and Dafny handle sequence comprehension.
- Error statements, non-determinism.
  - Impossible to translate, these are not included in Dafny by design.

Known Issues

- Language incompatibilities.
  - Differences in union type construction cause issues in fully automated translation.
  - Some important context is lost for some translations currently.
- Lacking proof obligations.
  - Could be resolved by automatically adding lemmas to manually discharge.
- Automated casting of types.
  - This is handled in some cases but is problematic when comparing custom number types.

What could have been done better?

- Use of VDMJ class mapping would have been better than traversing the AST manually.
- Implementation of missing language features.
- Automated production of lemmas to discharge in Dafny.
- Extensions to VDM++ and VDM-RT.