Automated Migration of Programs between Languages, through Annotated Grammars

Kim Mens
Diego Ordóñez Camacho

Outline

• Context of our research problem
• Annotated grammars and language translation
• Our case study
• Future work
• Conclusions
Overview

- Spacecraft mission control systems domain.
- There is a multitude of operations languages in this particular domain.
- New specialized operations design tools.
- Migrate existent procedures (programs) from its original language to the tool’s internal language.
- A specific translator is needed for every couple of languages.
- We want to automate as much as possible the construction of such translators.
- Thanks to the use of annotated grammars.
Operations Languages

- Small programming languages
  - to write procedures
  - to provide instructions to satellites

- Syntactically different but semantically equivalent

- Control flow-driven and imperative

```plaintext
LOCAL T1, T2
POINT SOL_TP1, SOL_TP2
T1 = IS_WARN(SOL_TP1)
T2 = IS_WARN(SOL_TP2)
IF (T1 || T2) THEN
    START sol_monitor()
ENDIF
```

A STOL procedure
Annotated Grammars

“DO” “WHILE”“(“ Expression “)” StatementList CommentStmt “ENDDO”

-> DoWhile

"<DecisionStep><BooleanResult>"
MoisExpression
"</BooleanResult><WHILE><REPEAT>"
StepList
MoisComment
"</REPEAT></WHILE></DecisionStep>”

-> WHILE
Annotated Grammars

“DO” “WHILE” “(“ Expression “)” StatementList CommentStmt “ENDDO”

-> DoWhile {cons(“While”)}

"<DecisionStep><BooleanResult>" MoisExpression "</BooleanResult><WHILE><REPEAT>" StepList MoisComment "</REPEAT></WHILE></DecisionStep>”

-> WHILE {cons(“While”)}
Annotated Grammars

“DO” “WHILE” “(
cond: Expression
“)”
StatementList
CommentStmt
“ENDDO”

-> DoWhile {cons(“While”)}

"<DecisionStep><BooleanResult>"
cond: MoisExpression
"</BooleanResult><WHILE><REPEAT>
StepList
MoisComment
"</REPEAT></WHILE></DecisionStep>”

-> WHILE {cons(“While”)}
Annotated Grammars

“DO” “WHILE” “(
cond: Expression
“)”
block: StatementList
CommentStmt
“ENDDO”

-> DoWhile {cons(“While”)}

"<DecisionStep><BooleanResult>
cond: MoisExpression
"</BooleanResult><WHILE><REPEAT>
block: StepList
MoisComment
"</REPEAT></WHILE></DecisionStep>”

-> WHILE {cons(“While”)}
Annotated Grammars

"DO" "WHILE" "(

\textbf{cond}: \text{Expression}

"")"

\textbf{block}: \text{StatementList}

\textbf{comm}: \text{CommentStmt}

"ENDDO"

\begin{verbatim}
-> DoWhile {cons("While")}
\end{verbatim}

"<DecisionStep><BooleanResult>"

\textbf{cond}: MoisExpression

"</BooleanResult><WHILE><REPEAT>"

\textbf{block}: StepList

\textbf{comm}: MoisComment

"</REPEAT></WHILE></DecisionStep>"

\begin{verbatim}
-> WHILE {cons("While")}
\end{verbatim}
Language Translation

Source Grammar

Annotations

Target Grammar

"DO WHILE (" 
Expres sion 
")" 
Do While 
StatementLi st 
Comm Stmt 
"ENDDO"

While 
cond block comm

"<DecisionStep> 
<BooleanResult>" 
Mois Expres sion 
"</BooleanResult> 
<WHILE> 
<REPEAT>" 
Step List 
Mois Com ment 
"</REPEAT> 
</WHILE> 
</DecisionStep>"
Generating Program Transformers

APPAREIL meta-tool

Source Language Specification → Partial Program Transformer

Target Language Specification → Additional Transf. Rules

Program in Source Language → Program Transformer

Program Transformer + Additional Transf. Rules → Program in Target Language

meta meta level (tool generator)

meta level (tool builder)

base level (tool user)
Case Study: from Stol to MOIS

- Stol: 150 productions
- MOIS: 62 productions
- Direct mapping: 109 Stol productions
- Mismatches: 41 Stol productions
- 59 different interventions to solve those mismatches

<table>
<thead>
<tr>
<th>Category</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving nodes to different places in the tree</td>
<td>7</td>
</tr>
<tr>
<td>Replacing partially/completely a node’s contents</td>
<td>29</td>
</tr>
<tr>
<td>Mapping the type of a node to another one</td>
<td>8</td>
</tr>
<tr>
<td>Creating additional nodes</td>
<td>13</td>
</tr>
<tr>
<td>Removing nodes from the parse tree</td>
<td>3</td>
</tr>
</tbody>
</table>
A Mismatch example

A Stol procedure:
DECLARE var1, var2, var3
Instruction1
DECLARE VAR4
Instruction2

An equivalent MOIS procedure:
<Procedure>
  <Header>
    <Variable>var1</Variable>
    <Variable>var2</Variable>
    <Variable>var3</Variable>
    <Variable>var4</Variable>
  </Header>
  <Body>
    <Step>Instruction1</Step>
    <Step>Instruction2</Step>
  </Body>
</Procedure>
Transforming a mismatch

Diagram showing the transformation process with nodes labeled as follows:

- Procedure
  - Declaration
    - var1
    - var2
    - var3
  - Instruction
    - Instruction1
    - Declaration
      - var4
    - Instruction2
  - Instruction1
  - Declaration
  - Instruction2
Transforming a mismatch
A Program Transformations Library

- move node(s):
  - void moveAfter(String nodePath, String destPath)
  - void moveBefore(String nodePath, String destPath)
  - void moveInside(String nodePath, String destPath)

- create node(s):
  - Node createAfter(Node ref, String tag, Object content)
  - Node createBefore(Node ref, String tag, Object content)
  - Node createInside(Node ref, String tag, Object content)

- transforms the type of node(s):
  - Node mapTo(Node node, Document gramm, String production)

- replace node(s)’s content:
  - void replace(String nodePath, Object content)

- ...

...
Future work

- We believe that the definition of transformations can be further simplified by wrapping the library under a dedicated language to define the transformations:

  - MOVE path1 (AFTER | BEFORE | INSIDE) path2
    - MOVE "//Declaration" BEFORE "/*/Procedure/Header"
  - path MAPTO production
  - MOVE path1 (AFTER | BEFORE | INSIDE) path2 MAPTO production
  - REPLACE path WITH nodeexpression
  - CREATE nodeexpression (AFTER | BEFORE | INSIDE) path
  - ...
Conclusions

• Annotated grammars can provide a partial generation of a language translator.
• Mismatches need to be solved by manually adding transformations to the translator.
• In our case study, the different kinds of transformations were consolidated into a library.
• We believe that further automation could be achieved by defining a dedicated transformation language backed up by our library.
Thanks for your attention !!!