Towards a Hybrid Transformation Language:
Implicit and Explicit Rule Scheduling
in Story Diagrams

Bart Meyers
Universiteit Antwerpen
Antwerpen, Belgium
bart.meyers@student.ua.ac.be

Pieter Van Gorp
Universiteit Antwerpen
Antwerpen, Belgium
pieter.vangorp@ua.ac.be

6th International Fujaba Days Workshop
Content of this presentation

1. Rule scheduling in transformation languages
2. Hybrid rule scheduling
3. Class2RDB transformation
4. Implementing hybrid rule scheduling
5. Special case: sanity checks
6. Conclusion
1. Rule scheduling in transformation languages

Implicit rule scheduling
- declarative, non-deterministic
- layers, priorities
- used in: AGG, AToM³

Explicit rule scheduling
- imperative, deterministic
- loops, conditionals
- used in: Fujaba, VMTS, MOLA, Progres, MoTMoT
1. Rule scheduling in transformation languages

Implicit rule scheduling
declarative, non-deterministic
layers, priorities
used in: AGG, AToM³

Equivalent languages, but different paradigm

Explicit rule scheduling
imperative, deterministic
loops, conditionals
used in: Fujaba, VMTS, MOLA, Progres, MoTMoT
1. Rule scheduling in transformation languages

**Implicit rule scheduling**
declarative, non-deterministic
layers, priorities
used in: AGG, ATOM³

**Equivalent languages, but different paradigm**

**Explicit rule scheduling**
imperative, deterministic
loops, conditionals
used in: Fujaba, VMTS, MOLA, Progres, MoTMoT

Why does a modeler need to make a choice?
2. Hybrid rule scheduling
2. Hybrid rule scheduling

Don't care about paradigms, schedule rules in the most intuitive way.
2. Hybrid rule scheduling

- Don't care about paradigms, schedule rules in the most intuitive way

Hybrid rule scheduling

- Improve structure, conciseness and readability of your transformation models
3. Class2RDB transformation

```
<table>
<thead>
<tr>
<th>Item</th>
<th>Location</th>
<th>Item</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>+name : String</td>
<td>+shelf : int</td>
<td>PK</td>
<td>name : String</td>
</tr>
<tr>
<td>+location : Location</td>
<td>+section : int</td>
<td>FK1</td>
<td>shelf : int</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>section : int</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FK1</td>
<td>shelf : int</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>section : int</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PK</td>
<td>shelf : int</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PK</td>
<td>section : int</td>
</tr>
</tbody>
</table>
```
3. Class2RDB transformation

- **No tables exist**
  - \{motmot.transprimitive\} = No tables exist

  - **success**
    - **loop**
      - Transform classes to tables
        - \{motmot.transprimitive\} = Transform classes to tables
          - **loop**
            - Data fields to columns
              - \{motmot.transprimitive\} = Data fields to columns
                - **loop**
                  - Object fields to foreign keys
                    - \{motmot.transprimitive\} = Object fields to foreign keys

  - **failure**
    - **success**
      - Transform classes to tables
        - \{motmot.transprimitive\} = Transform classes to tables
          - **loop**
            - Data fields to columns
              - \{motmot.transprimitive\} = Data fields to columns
                - **loop**
                  - Object fields to foreign keys
                    - \{motmot.transprimitive\} = Object fields to foreign keys
3. Class2RDB transformation

- No tables exist
  \{motmot.transprimitive=No tables exist\}

  - Conditional

  - Success
    - Transform classes to tables
      \{motmot.transprimitive=Transform classes to tables\}
      - Loop

  - Failure
    - Data fields to columns
      \{motmot.transprimitive=Data fields to columns\}
      - Loop
      - Loop

    - Object fields to foreign keys
      \{motmot.transprimitive=Object fields to foreign keys\}
      - Loop
3. Class2RDB transformation

- No tables exist
  \{motmot.transprimitive=No tables exist\}

- Transform classes to tables
  \{motmot.transprimitive=Transform classes to tables\}

  - Data fields to columns
    \{motmot.transprimitive=Data fields to columns\}

  - Object fields to foreign keys
    \{motmot.transprimitive=Object fields to foreign keys\}
3. Class2RDB transformation

Step 1

No tables exist
{motmot.transprimitive=No tables exist}

Step 2

Transform classes to tables
{motmot.transprimitive=Transform classes to tables}

Data fields to columns
{motmot.transprimitive=Data fields to columns}

Object fields to foreign keys
{motmot.transprimitive=Object fields to foreign keys}
3. Class2RDB transformation

- **No tables exist**
  - motmot.transprimitive=No tables exist

- **Success**
  - **Loop**
    - Transform classes to tables
      - motmot.transprimitive=Transform classes to tables

- **Failure**
  - **Loop**
    - Data fields to columns
      - motmot.transprimitive=Data fields to columns
  - **Loop**
    - Object fields to foreign keys
      - motmot.transprimitive=Object fields to foreign keys

Attributes to columns
3. Class2RDB transformation

- **No tables exist**
  - `motmot.transprimitive=No tables exist`

- **Success**
  - **Loop**
    - **Transform classes to tables**
      - `motmot.transprimitive=Transform classes to tables`

- **Failure**
  - **Loop**
    - **Transform attributes to columns**
      - `motmot.transprimitiveND=Data fields to columns, Object fields to foreign keys`
3. Class2RDB transformation

New language construct to support implicit rule scheduling
3. Class2RDB transformation

Keep evaluating these two rules until both fail to match.
3. Class2RDB transformation

Equivalent to the `transprimitiveND` state
3. Class2RDB transformation

A rule is chosen non-deterministically through random numbers.
3. Class2RDB transformation

One of these rules is evaluated
One of these rules is evaluated
3. Class2RDB transformation

If the rule didn't match
3. Class2RDB transformation

- No tables exist
  - No tables exist

- Transform classes to tables
  - Transform classes to tables

- Initialize

- Choose pattern id

- Data fields to columns
  - Data fields to columns

- Object fields to foreign key
  - Object fields to foreign keys

- Clear ignore list

- No match, ignore pattern

- End
If all rules are ignored, then continue to the final state.
3. Class2RDB transformation

Otherwise, choose a new rule for evaluation.
3. Class2RDB transformation

Otherwise, choose a new rule for evaluation
If the rule matched
3. Class2RDB transformation

Clear the list of ignored rules for the next iteration
3. Class2RDB transformation

Class2RDB using plain story diagrams
4. Implementing hybrid rule scheduling

Class2RDB transformation model using *plain story diagrams*
4. Implementing hybrid rule scheduling

MoTMoT

Class2RDB transformation model using plain story diagrams
How can this transformation model be created automatically?
4. Implementing hybrid rule scheduling

Class2RDB transformation model using \textit{transprimitiveND}

... from the transformation model using the new language construct

Class2RDB transformation model using \textit{plain story diagrams}
4. Implementing hybrid rule scheduling

Class2RDB transformation model using \textit{transprimitiveND}

MoTMoT

Higher order transformation

Class2RDB transformation model using \textit{plain story diagrams}

Platform independence
4. Implementing hybrid rule scheduling

- Match SDM Profile metadata
  - \{motmot.transprimitive=Match profile metadata\}

- For all transprimitiveND states
  - \{motmot.transprimitive=For all transprimitiveND states\}

- For all referenced patterns
  - \{motmot.transprimitive=For all referenced patterns in ND state\}

- Add perform pattern state
  - \{motmot.transprimitive=Add perform pattern state\}

- Remove transprimitiveND tag definition
  - \{motmot.transprimitive=Remove transprimitiveND tag definition\}

- Remove prototype perform pattern
  - \{motmot.transprimitive=Remove prototype perform pattern\}

- Remove nondeterministic state
  - \{motmot.transprimitive=Remove nondeterministic state\}

- Create flattened transformation prototype

- Set number of patterns
  - \{motmot.transprimitive=Set number of patterns\}

- Lookup perform pattern state
  - \{motmot.transprimitive=Lookup perform pattern state\}

- Lookup referenced pattern
  - \{motmot.transprimitive=Lookup referenced pattern\}
4. Implementing hybrid rule scheduling

Transform all non-deterministic states one by one
These three states could be modeled as one transprimitiveND state!
4. Implementing hybrid rule scheduling

These three states could be modeled as one transprimitiveND state!

We could bootstrap this transformation
5. Special case: sanity checks

First, a lot of “sanity” checks have to be passed.
5. Special case: sanity checks

\[
\text{motmot.transprimitiveND} = \text{sanity check 1, sanity check 2, sanity check 3, ..., sanity check n}
\]

\[
\text{motmot.transprimitive} = \text{actual transformation}
\]
5. Special case: sanity checks

If one of the rules failed to match
If all of them matched
5. Special case: sanity checks

- Improved structure, readability and conciseness
6. Conclusion

Hybrid rule scheduling
6. Conclusion

Syntax

Hybrid rule scheduling
6. Conclusion

- Hybrid rule scheduling
- Informal semantics
- Syntax
6. Conclusion

Hybrid rule scheduling

Syntax

Informal semantics

Tool support
6. Conclusion

Hybrid rule scheduling

- Syntax
- Informal semantics
- Tool support
- Improve structure, readability and conciseness of models
6. Conclusion

Hybrid rule scheduling

- Informal semantics
- Syntax
- Tool support
- Improve structure, readability and conciseness of models
- Higher order transformations for platform independence
6. Conclusion

Discussion

Higher order transformations for platform independence?

Hybrid transformation language?

True non-determinism or an imposed choice?