

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

# Content of this presentation

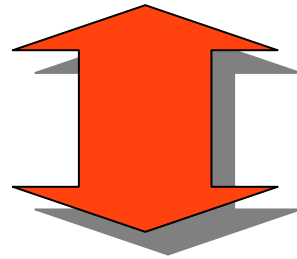
1. Rule scheduling in transformation languages
2. Hybrid rule scheduling
3. Class2RDB transformation
4. Implementing hybrid rule scheduling
5. Special case: sanity checks
6. Conclusion

# 1. Rule scheduling in transformation languages

**Implicit rule scheduling**  
declarative, non-deterministic

layers, priorities

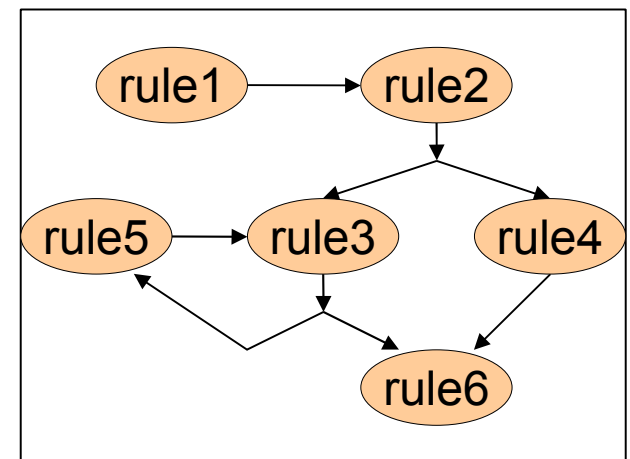
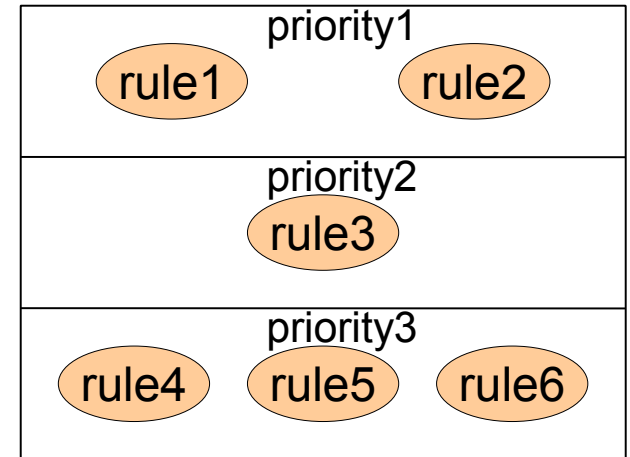
used in: AGG, AToM<sup>3</sup>



**Explicit rule scheduling**  
imperative, deterministic

loops, conditionals

used in: Fujaba, VMTS, MOLA, Progres, MoTMoT

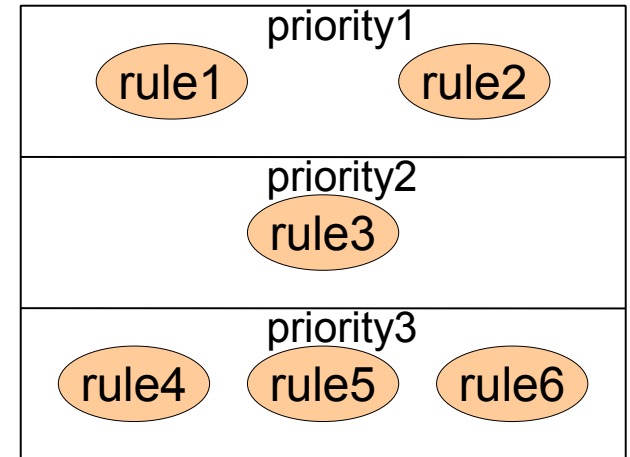


# 1. Rule scheduling in transformation languages

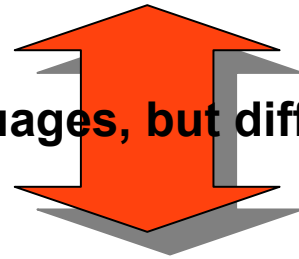
**Implicit rule scheduling**  
declarative, non-deterministic

layers, priorities

used in: AGG, AToM<sup>3</sup>



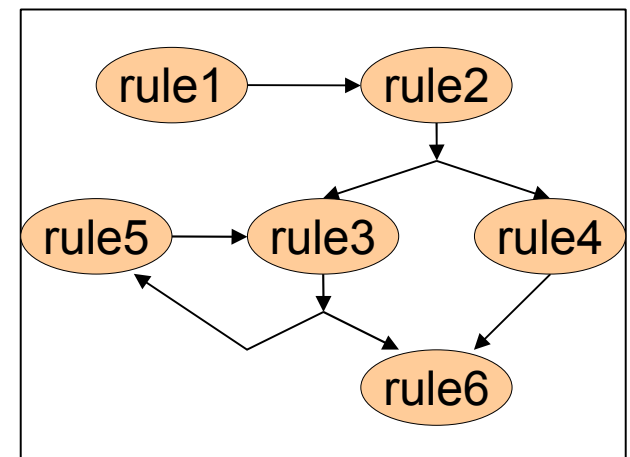
Equivalent languages, but different paradigm



**Explicit rule scheduling**  
imperative, deterministic

loops, conditionals

used in: Fujaba, VMTS, MOLA, Progres, MoTMoT

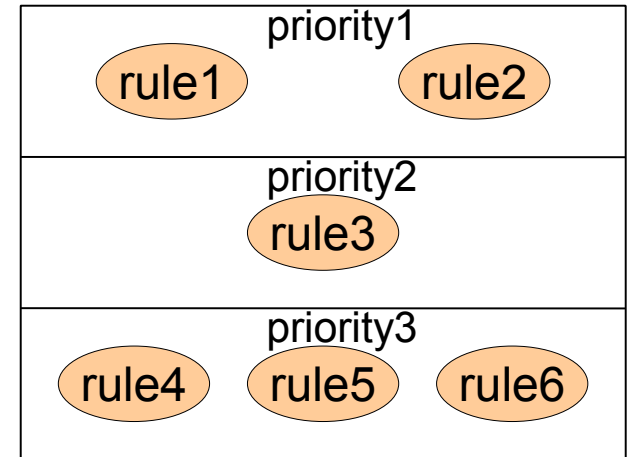


# 1. Rule scheduling in transformation languages

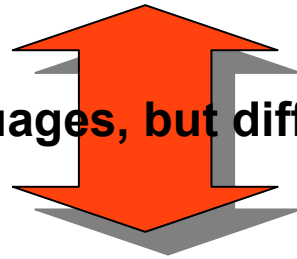
**Implicit rule scheduling**  
declarative, non-deterministic

layers, priorities

used in: AGG, AToM<sup>3</sup>



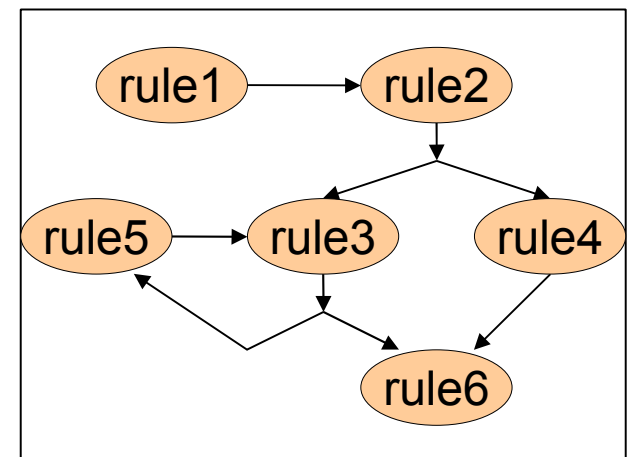
Equivalent languages, but different paradigm



**Explicit rule scheduling**  
imperative, deterministic

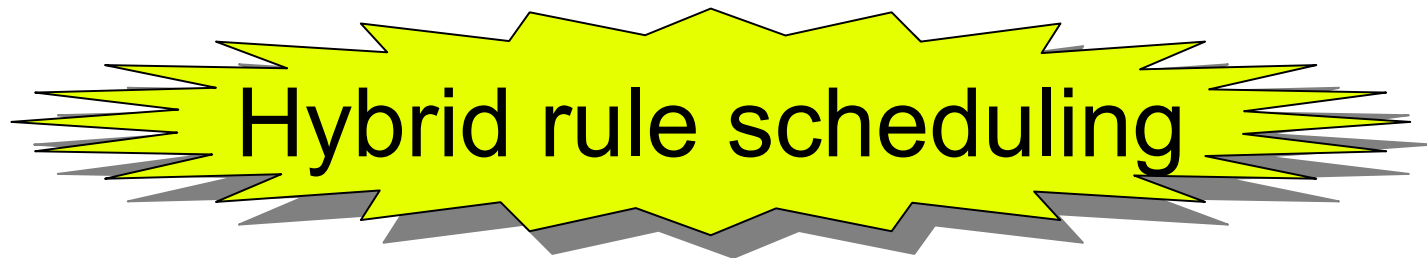
loops, conditionals

used in: Fujaba, VMTS, MOLA, Progres, MoTMoT

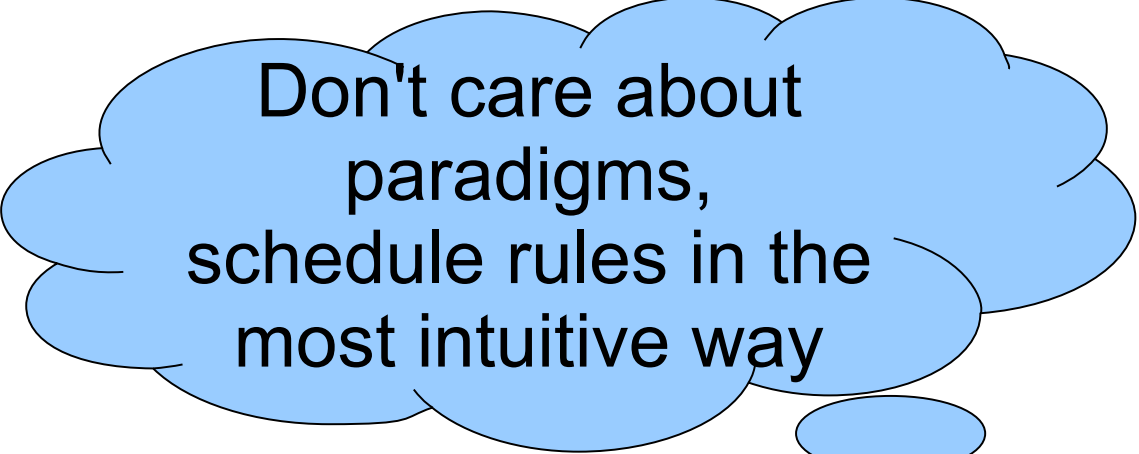


Why does a modeler *need* to make a choice?

## 2. Hybrid rule scheduling



## 2. Hybrid rule scheduling



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



**Hybrid rule scheduling**

## 2. Hybrid rule scheduling

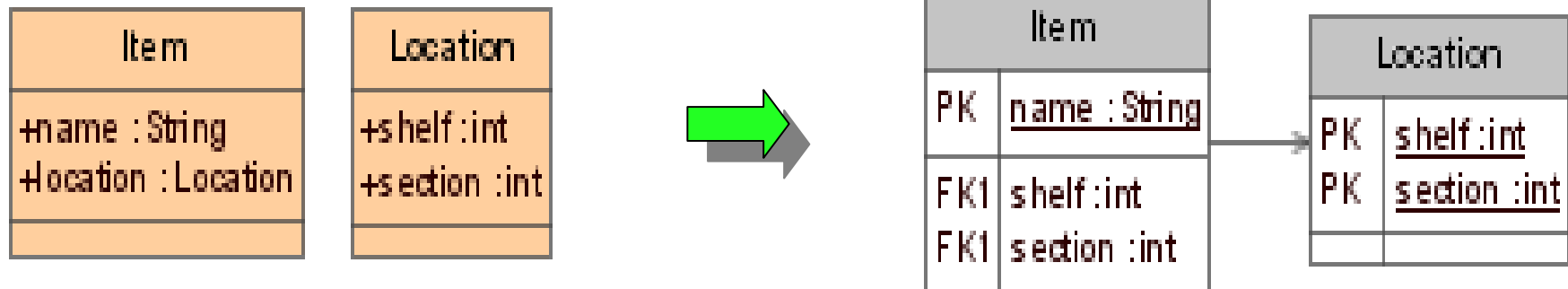
Don't care about paradigms,  
schedule rules in the  
most intuitive way

Hybrid rule scheduling

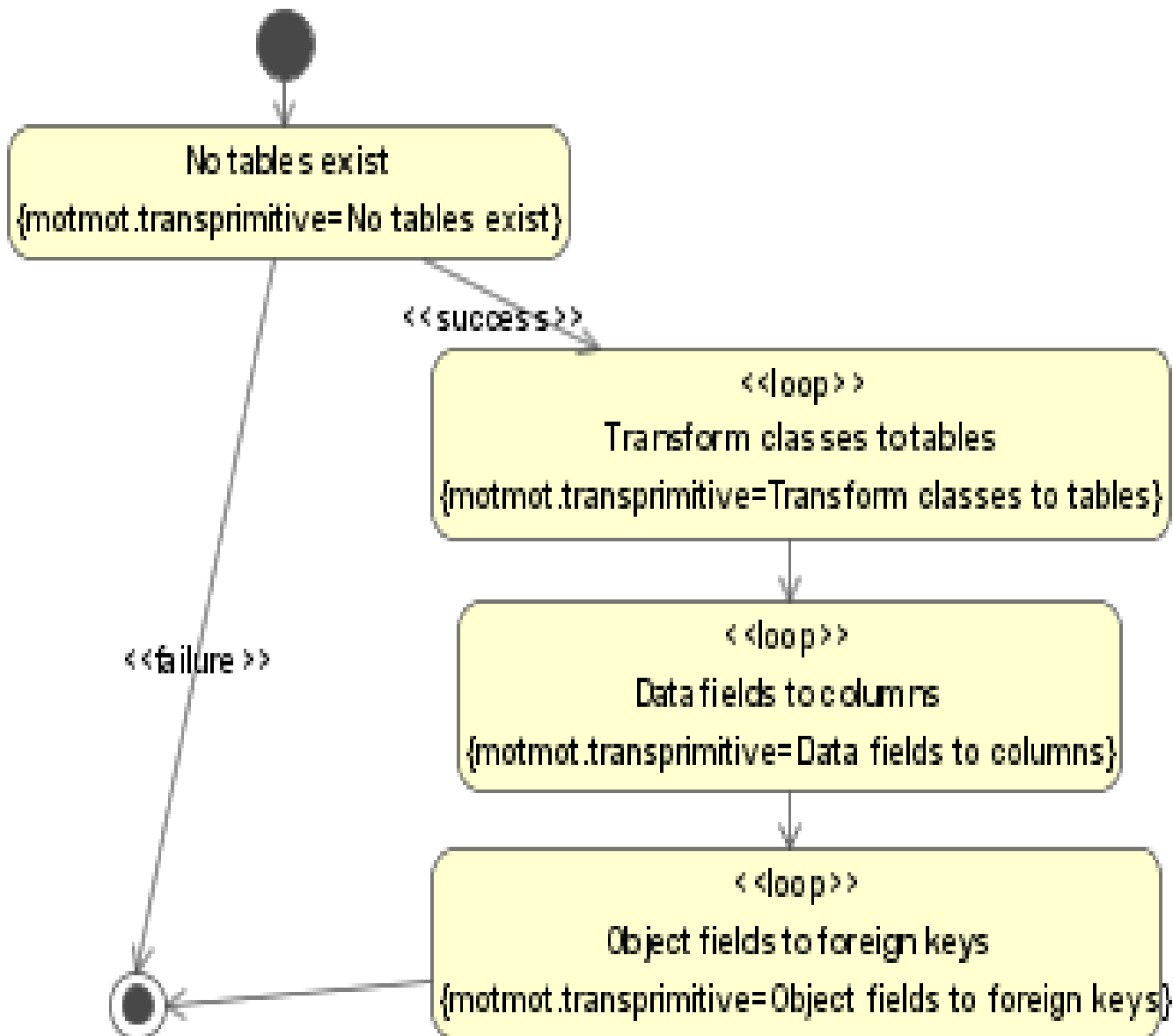
Improve structure,  
conciseness and  
readability of your  
transformation models



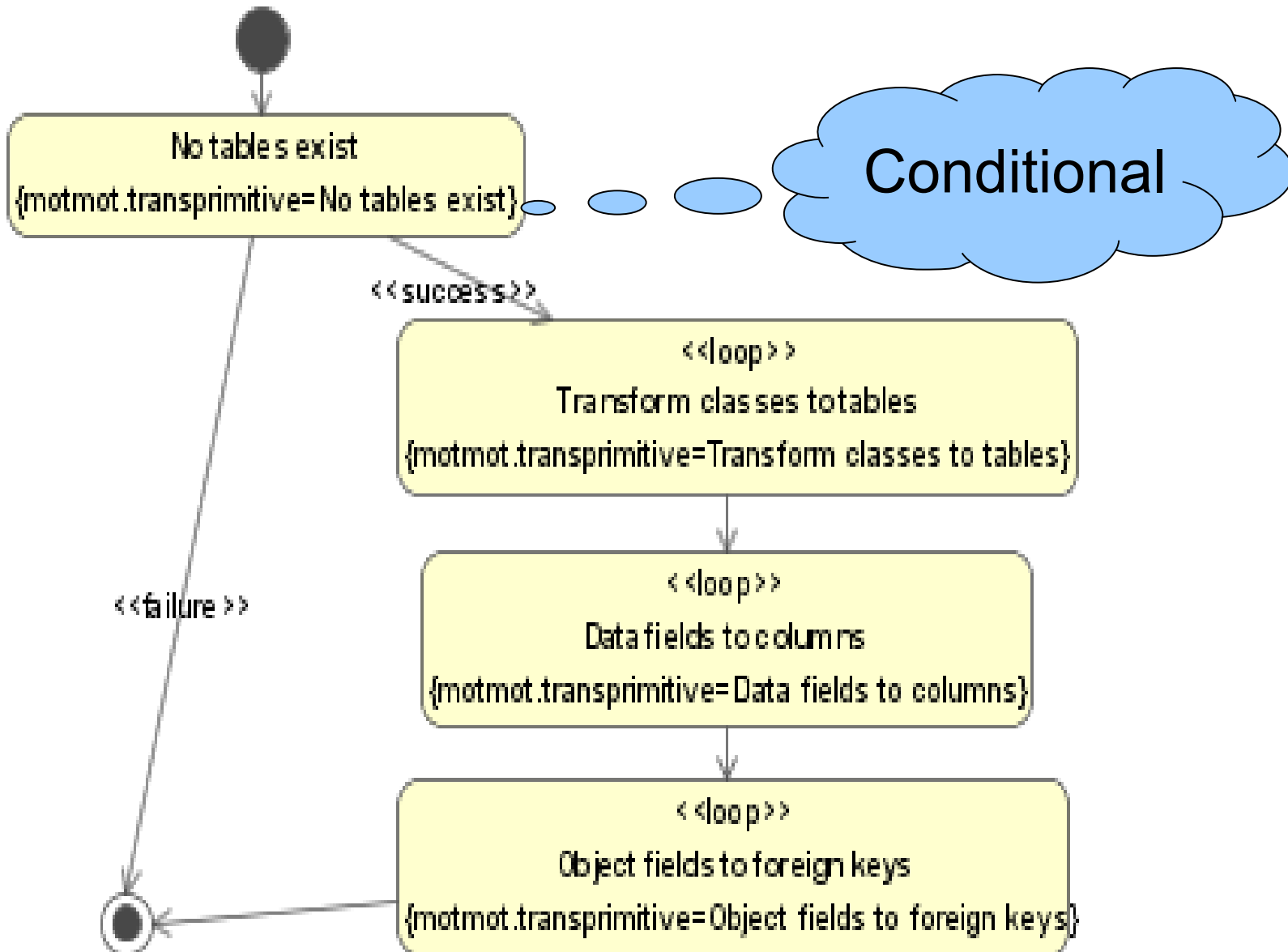
### 3. Class2RDB transformation



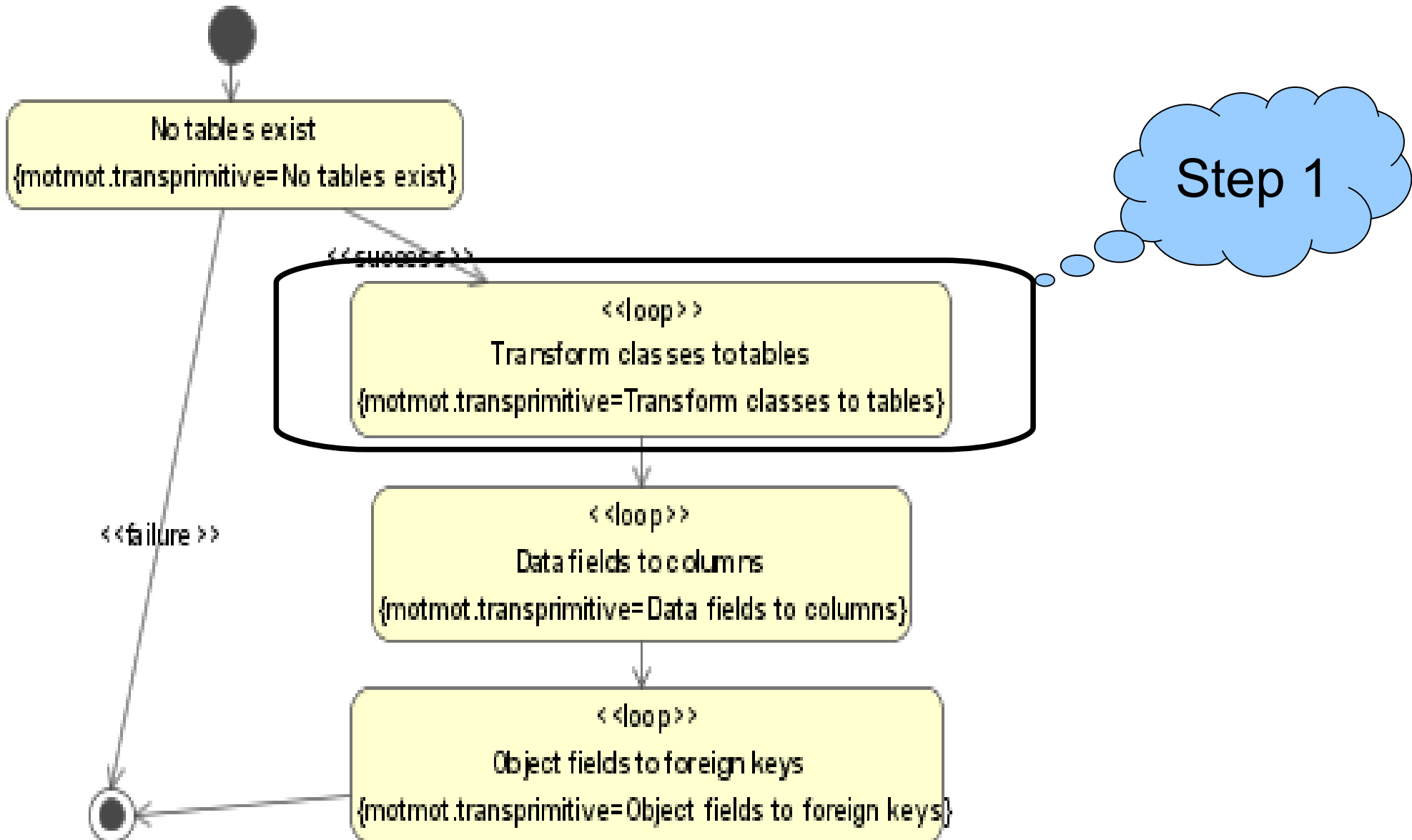
### 3. Class2RDB transformation



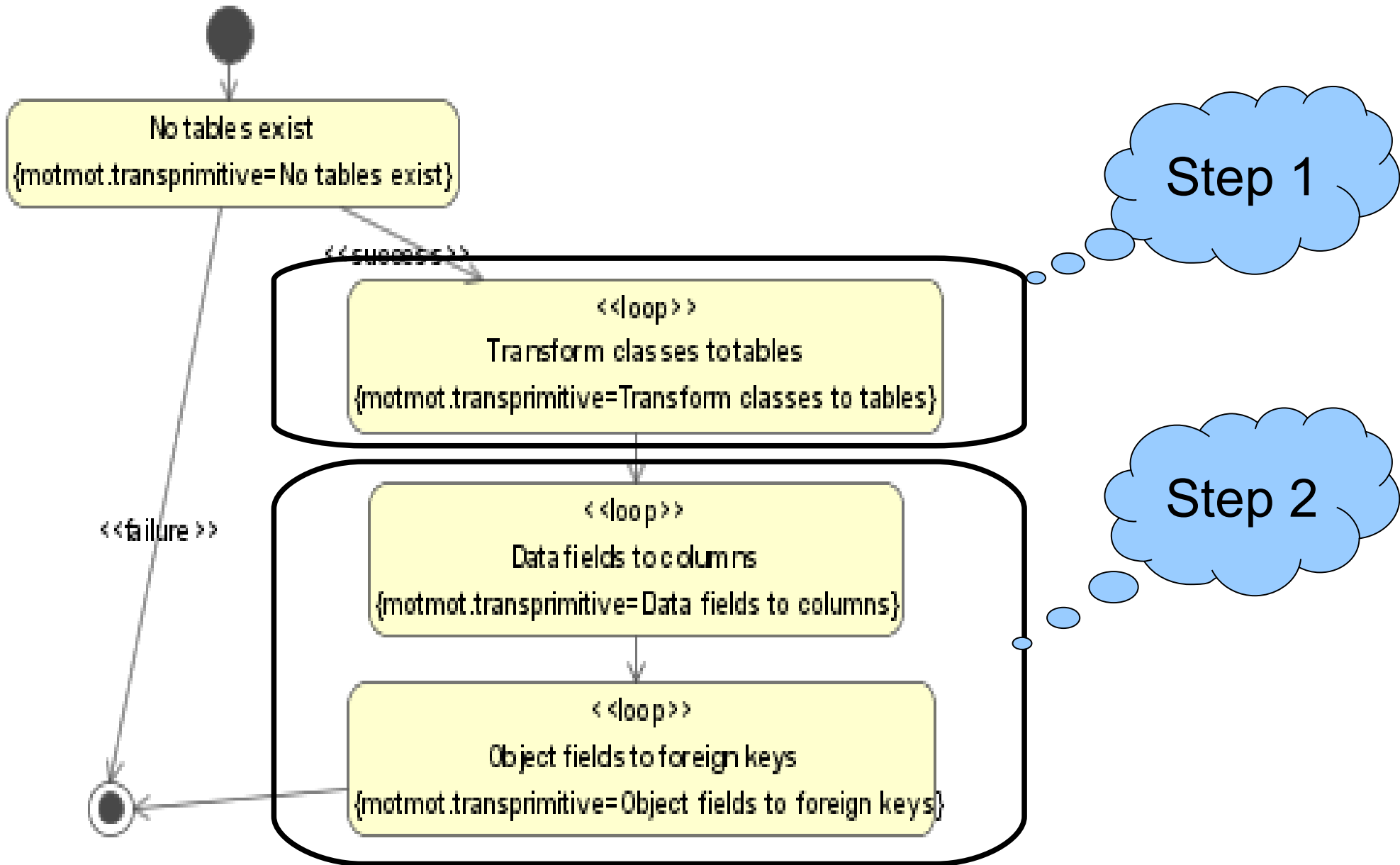
### 3. Class2RDB transformation



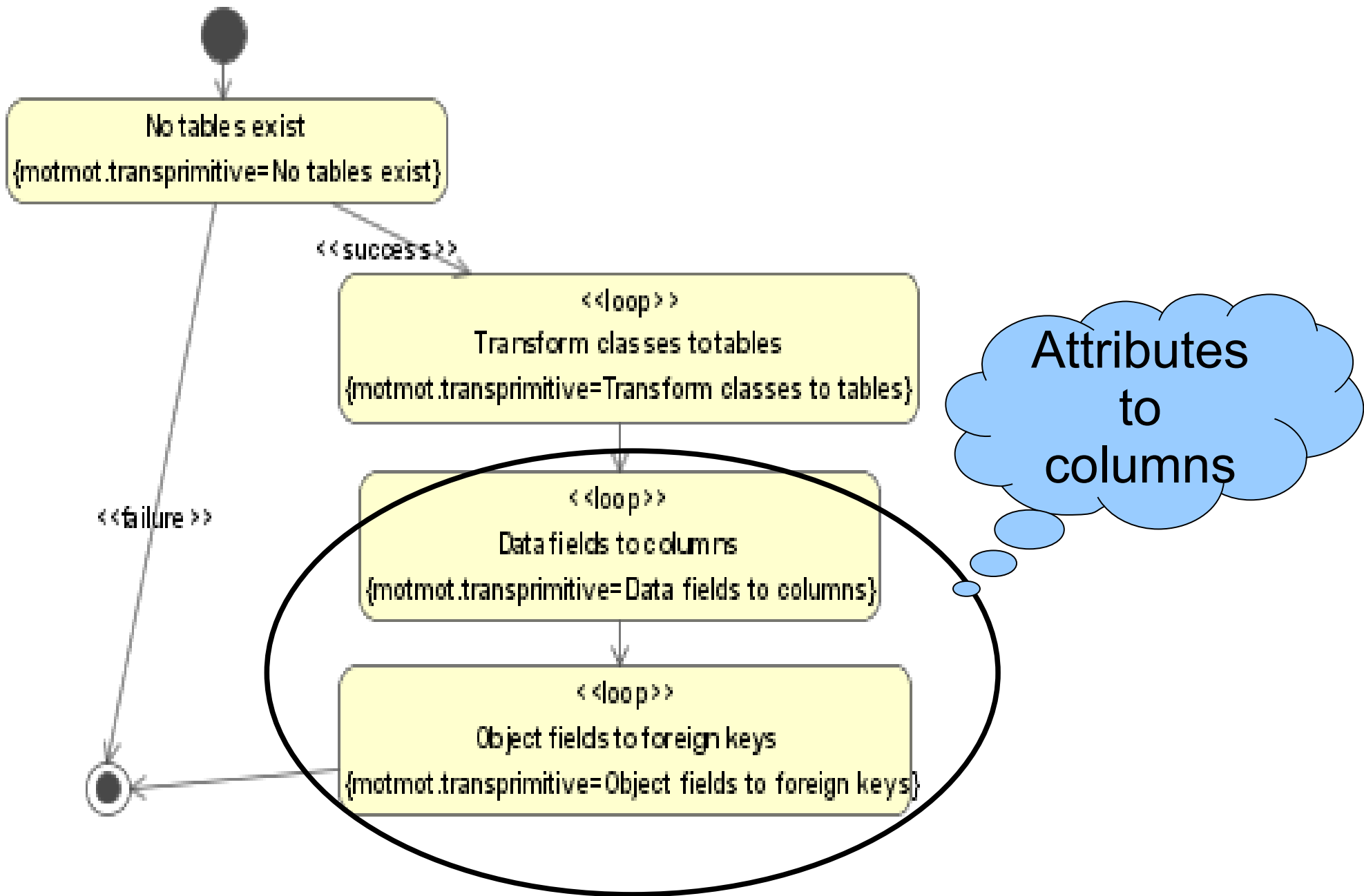
### 3. Class2RDB transformation



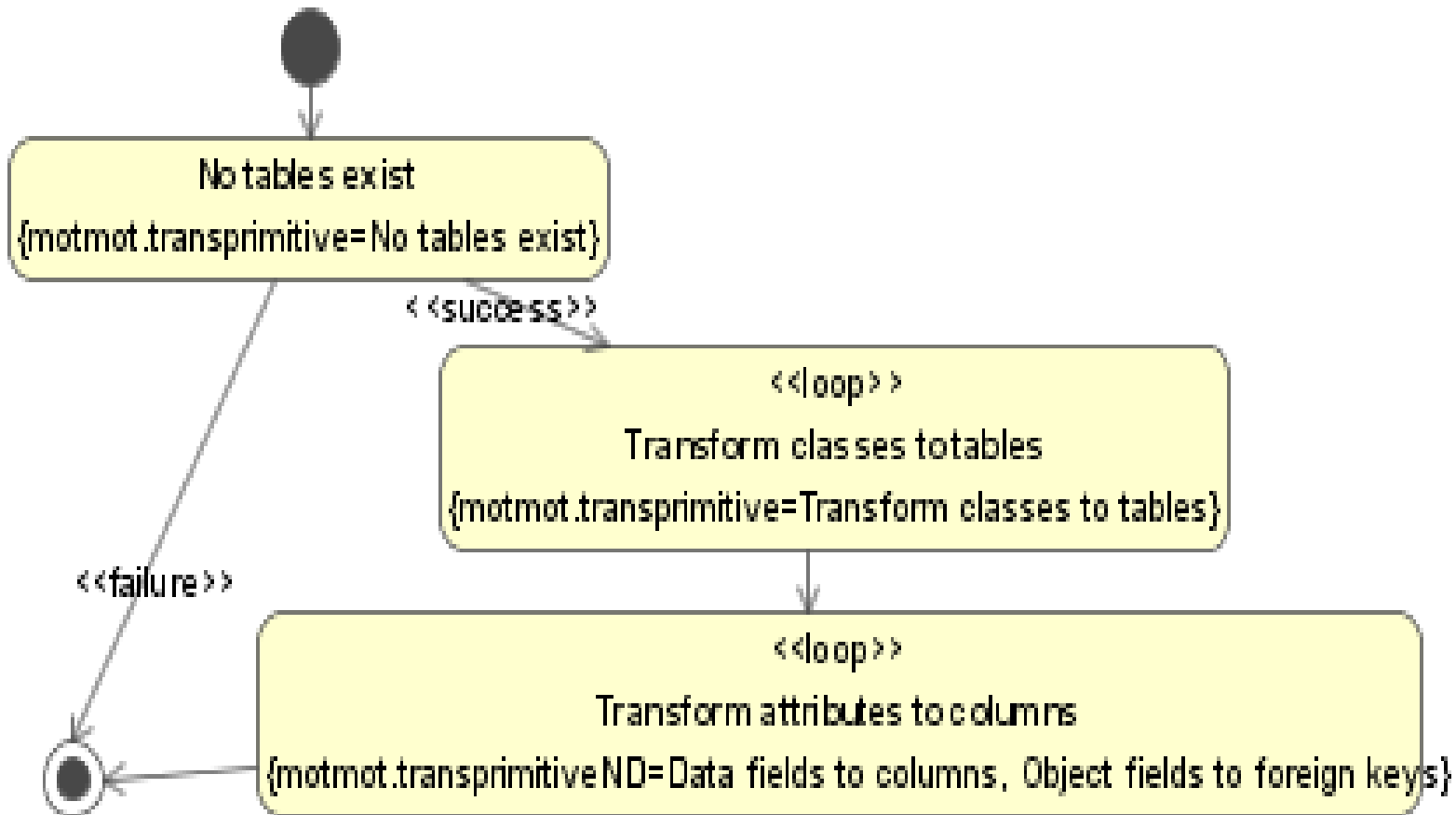
### 3. Class2RDB transformation



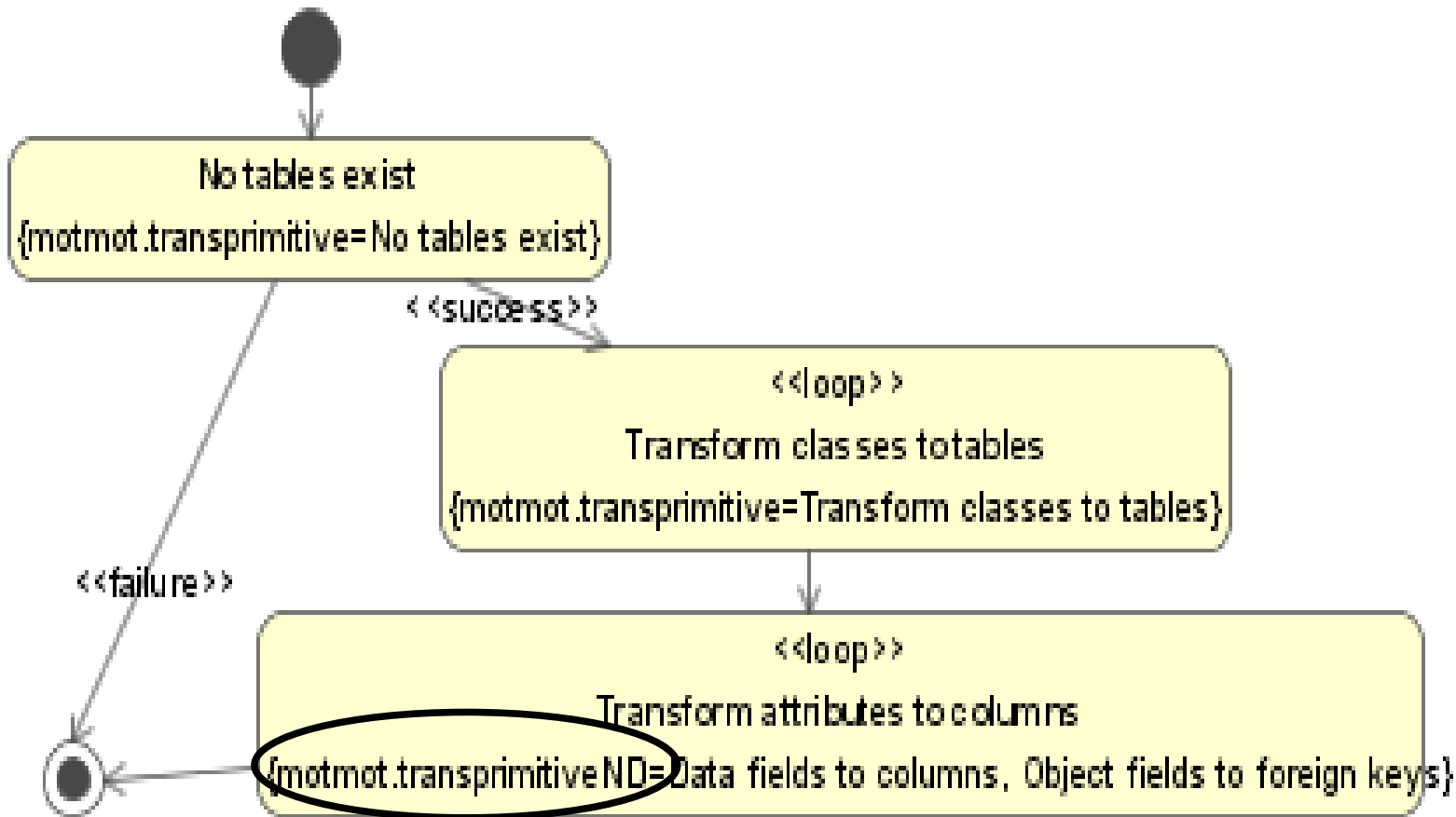
### 3. Class2RDB transformation



### 3. Class2RDB transformation



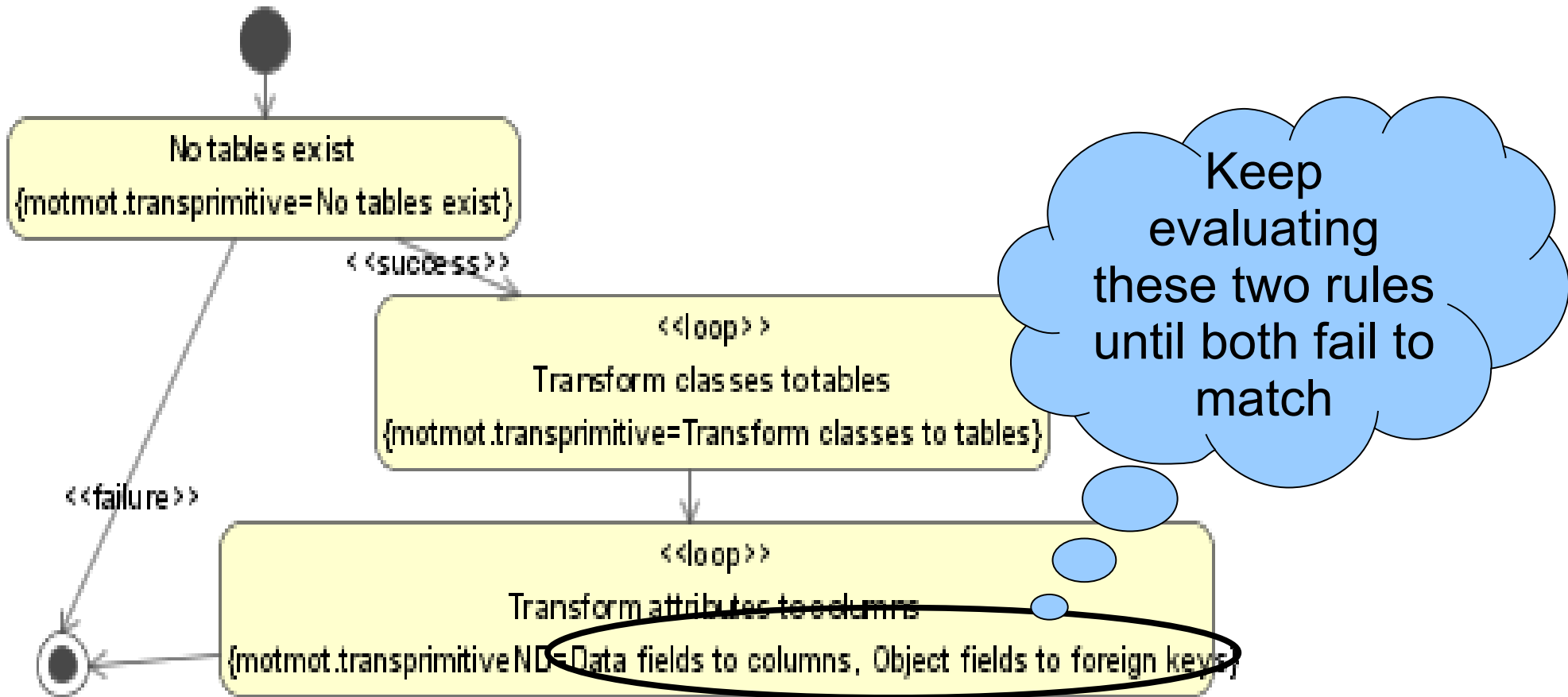
### 3. Class2RDB transformation



New language construct  
to support implicit rule  
scheduling

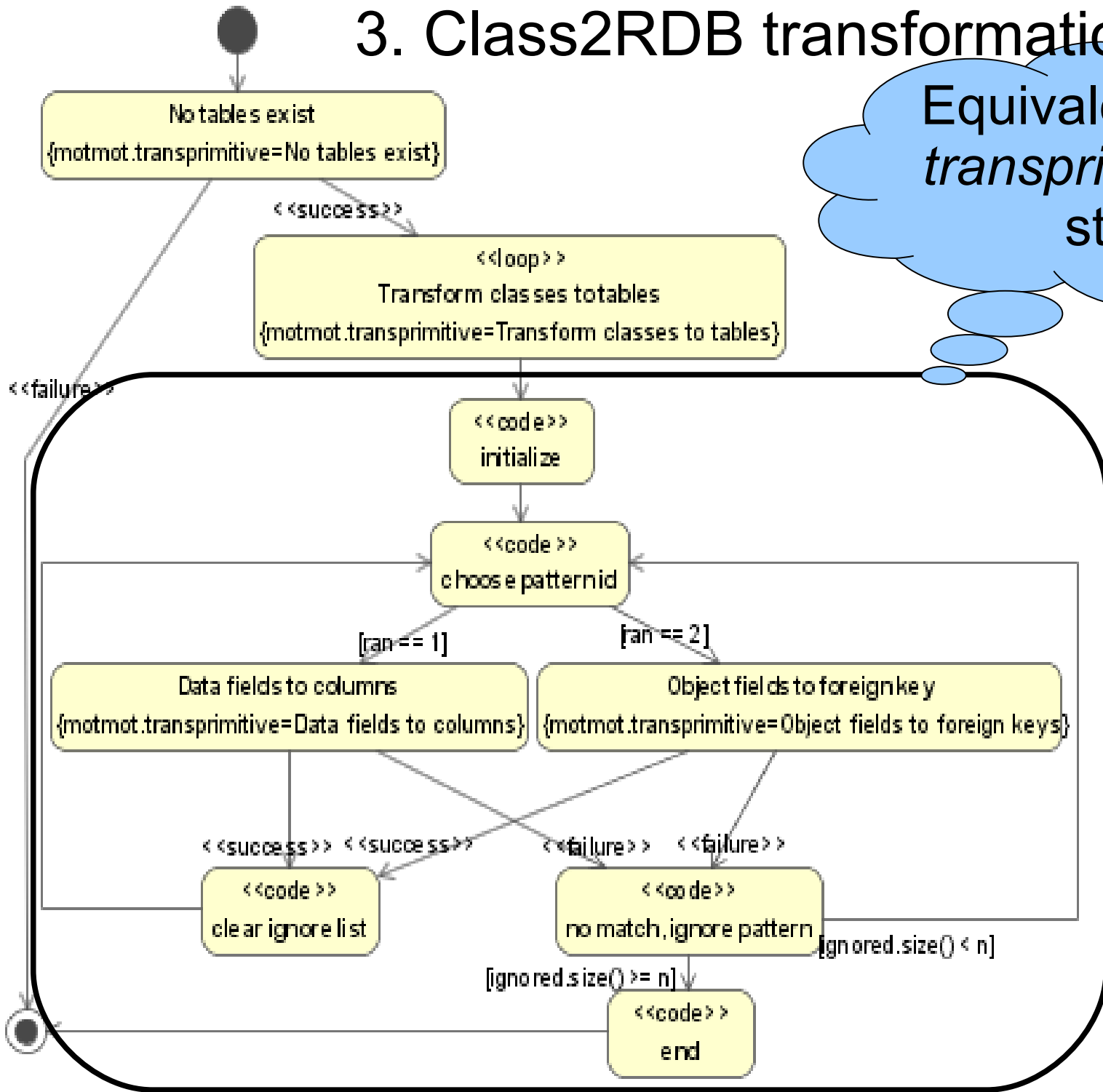


### 3. Class2RDB transformation

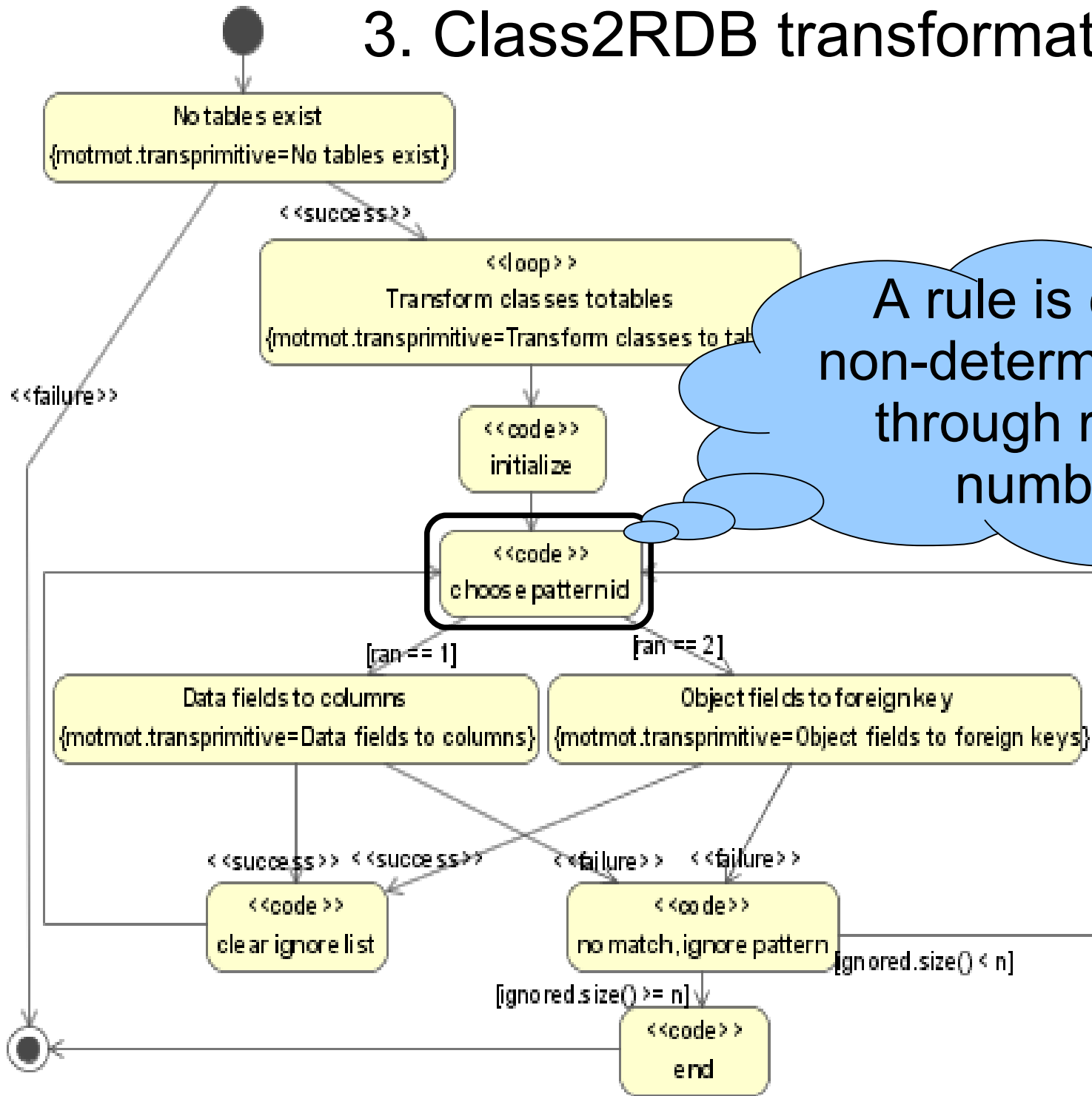


### 3. Class2RDB transformation

Equivalent to the *transprimitiveND* state

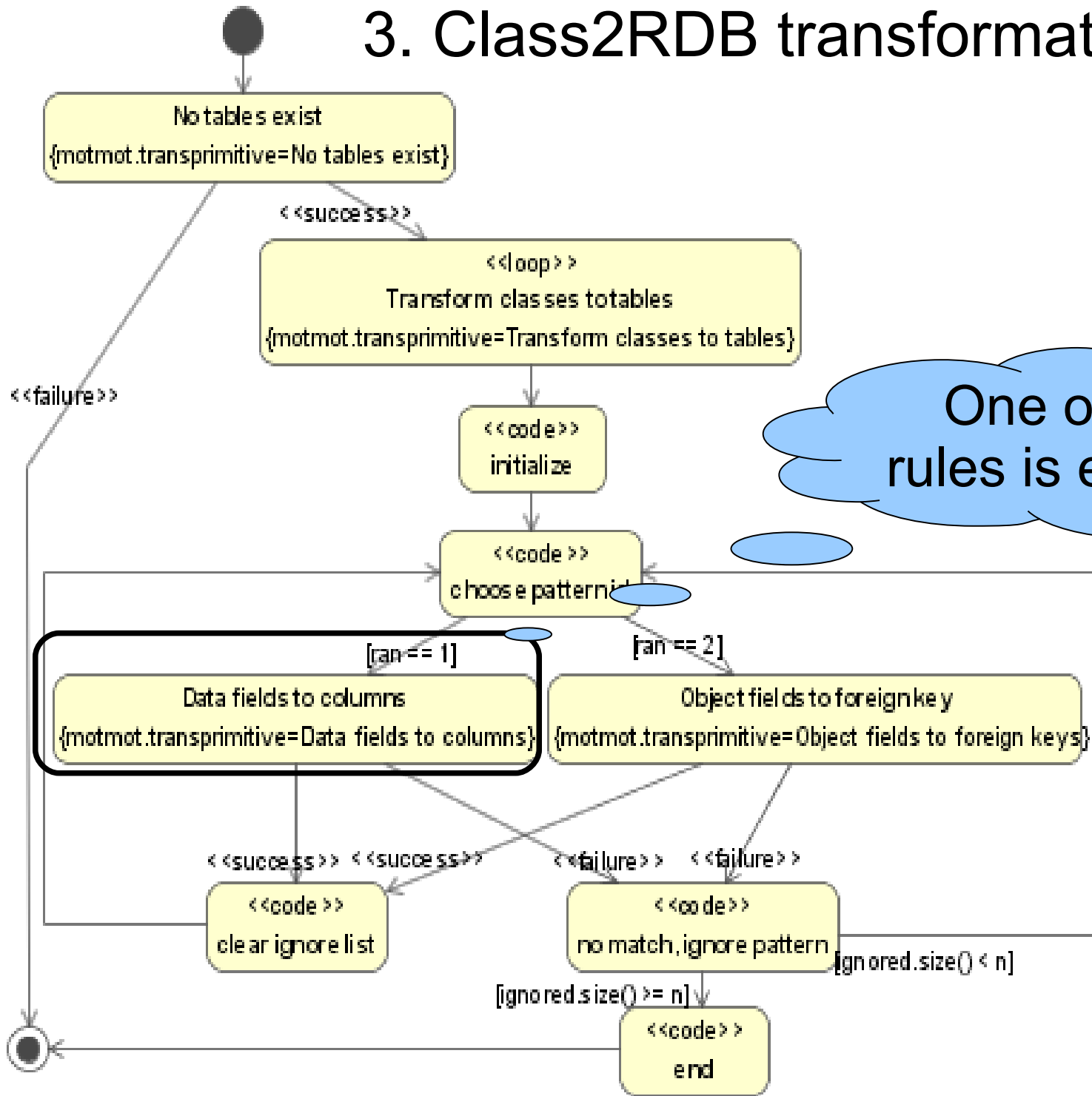


### 3. Class2RDB transformation



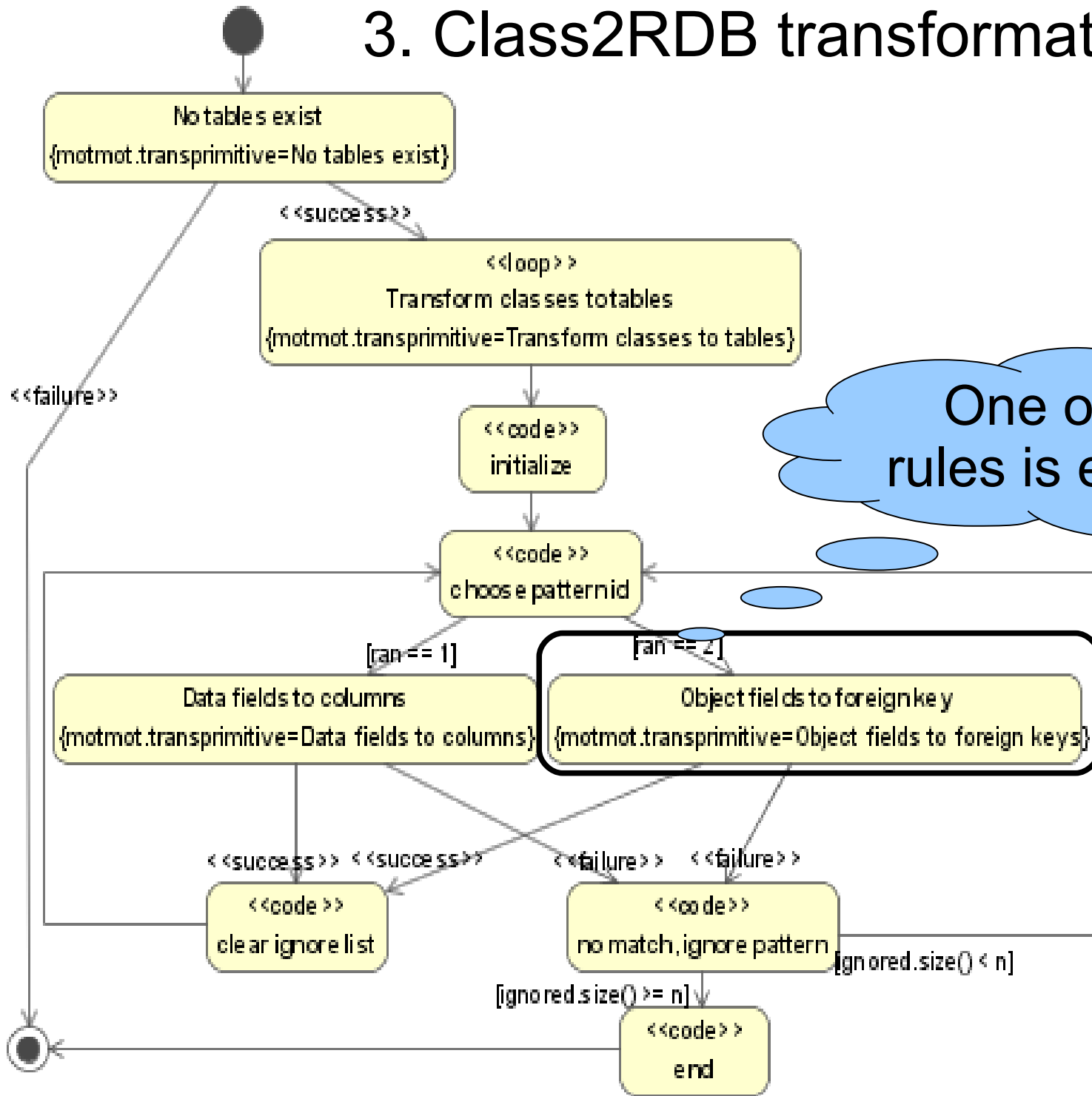
A rule is chosen non-deterministically through random numbers

# 3. Class2RDB transformation



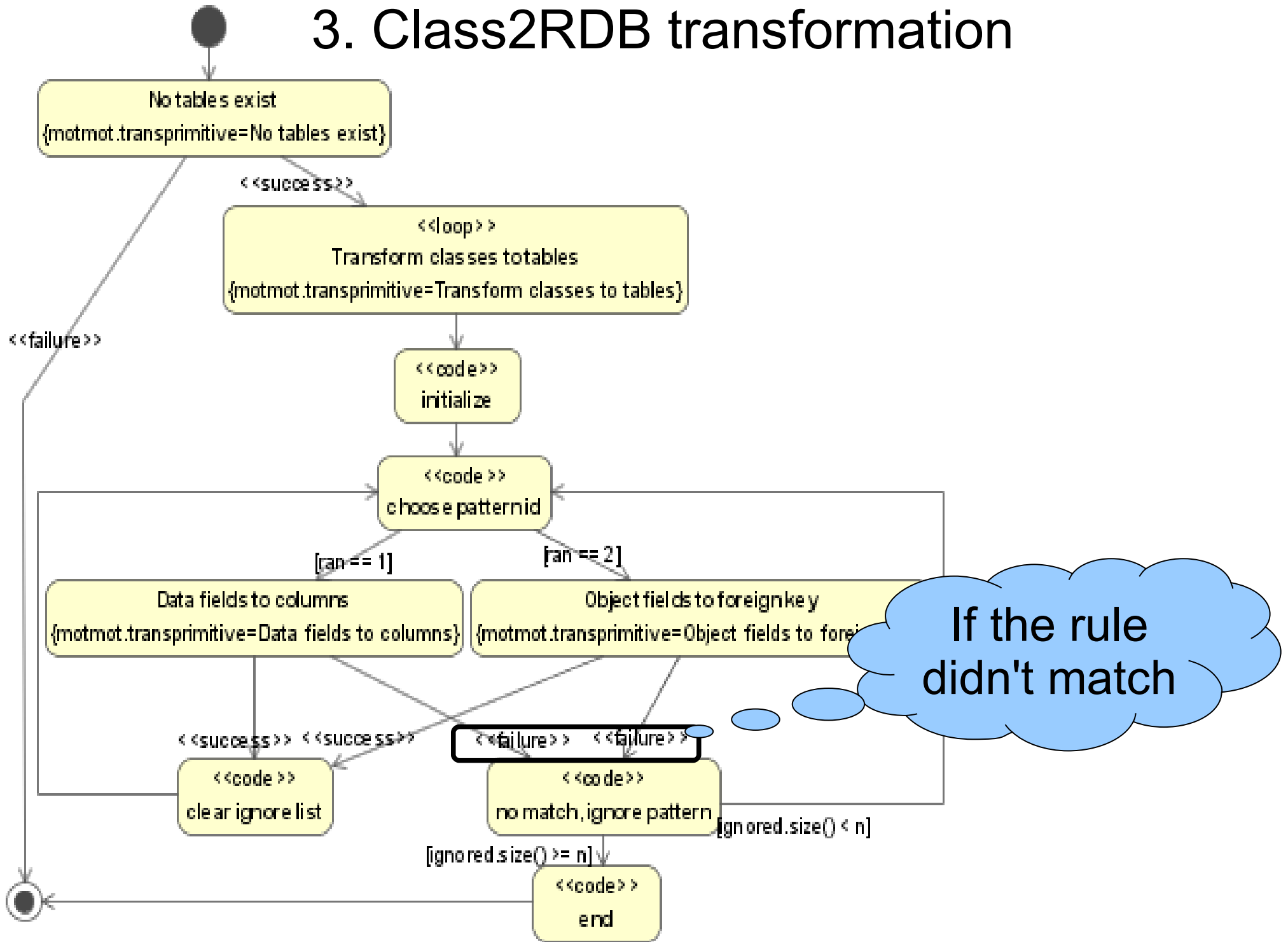
One of these rules is evaluated

# 3. Class2RDB transformation

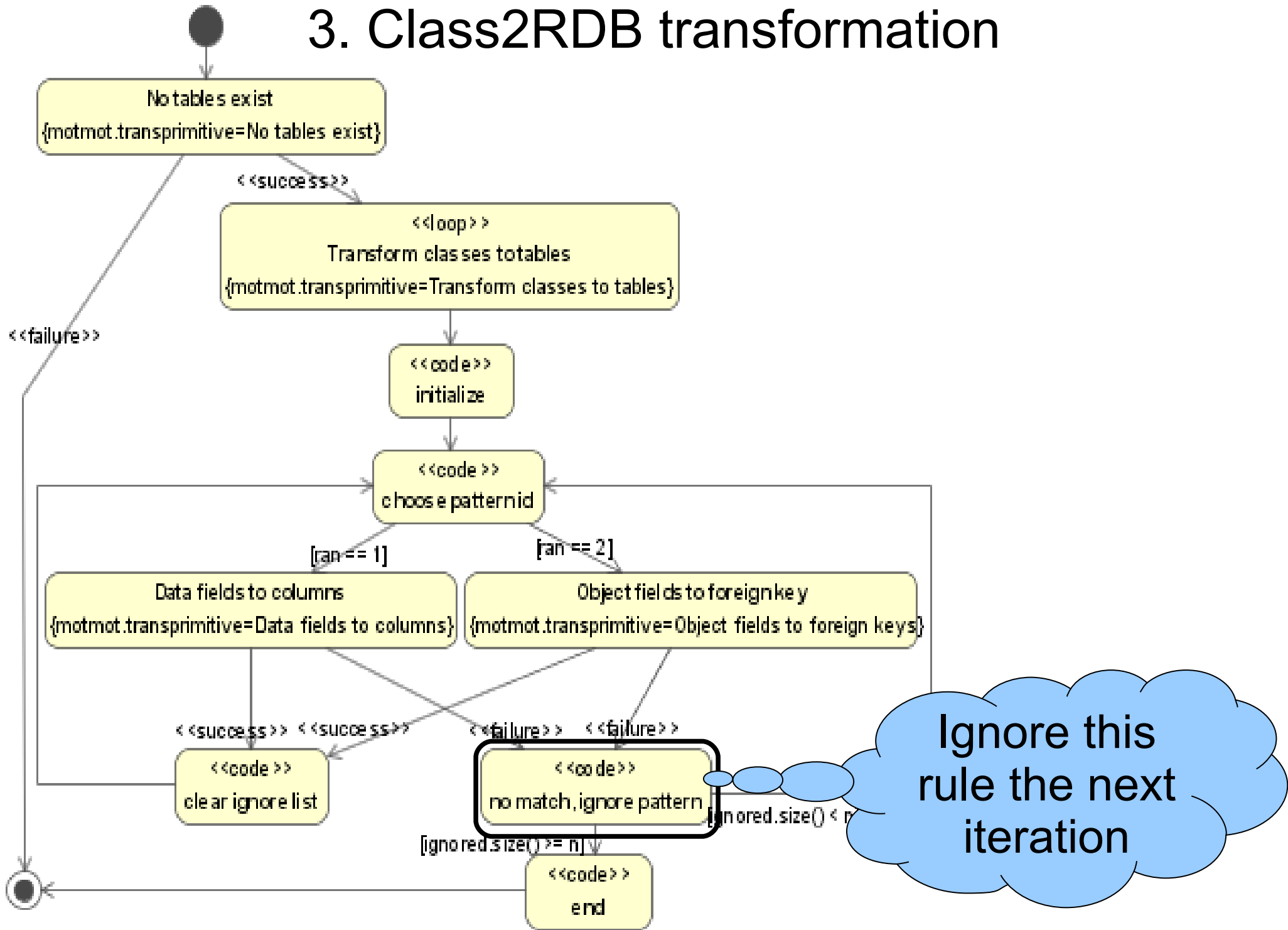


One of these rules is evaluated

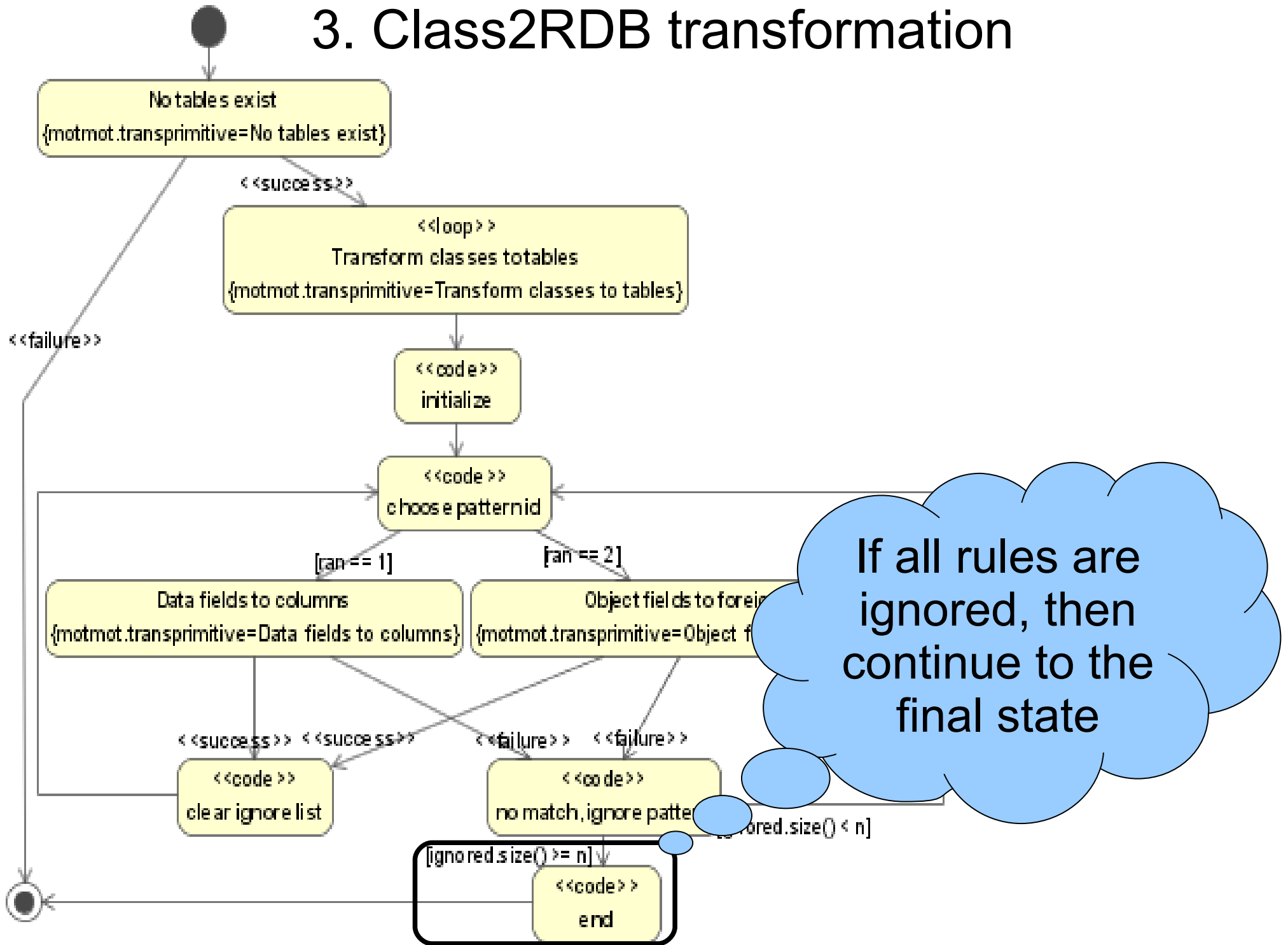
# 3. Class2RDB transformation



### 3. Class2RDB transformation

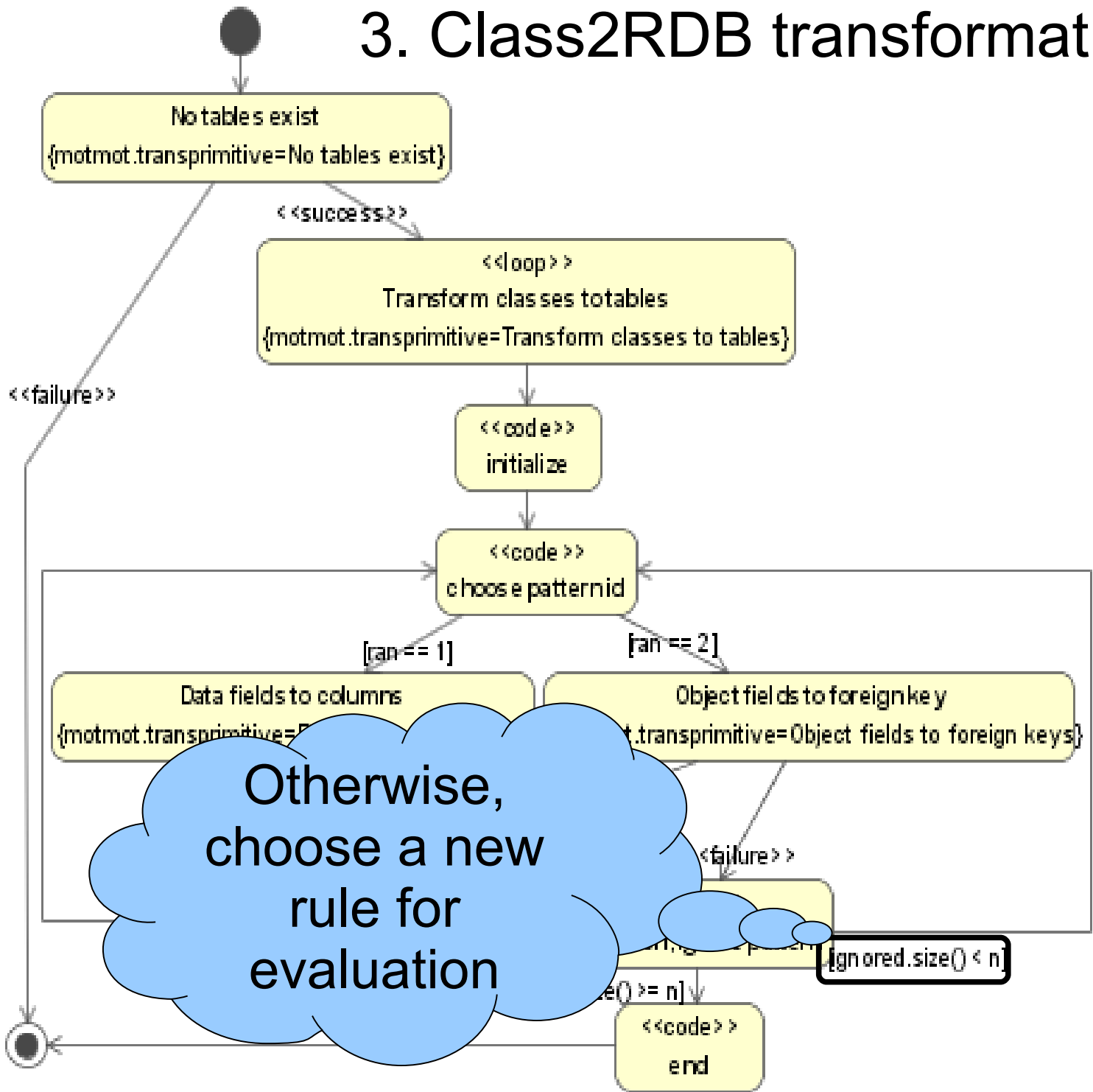


### 3. Class2RDB transformation

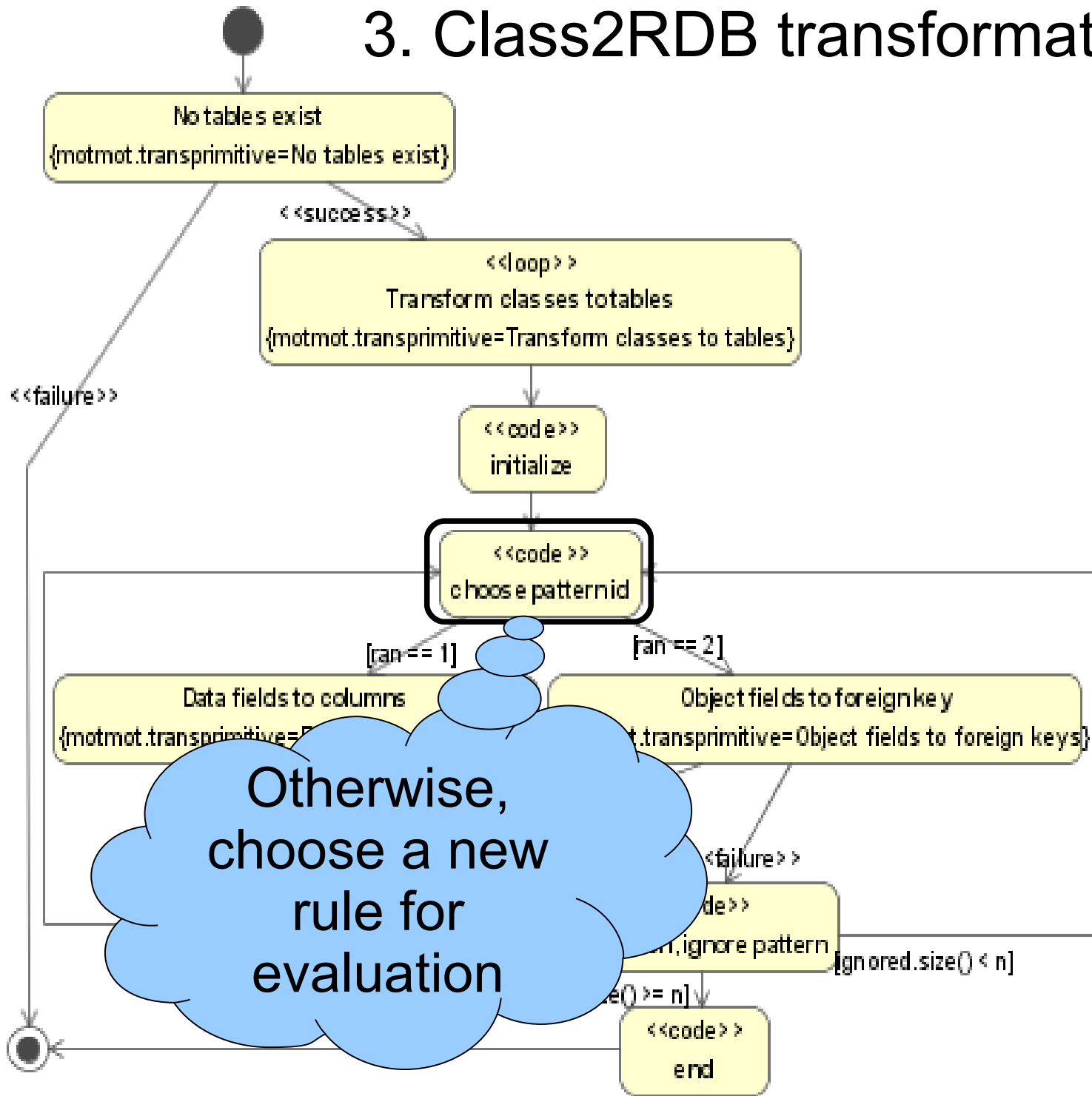




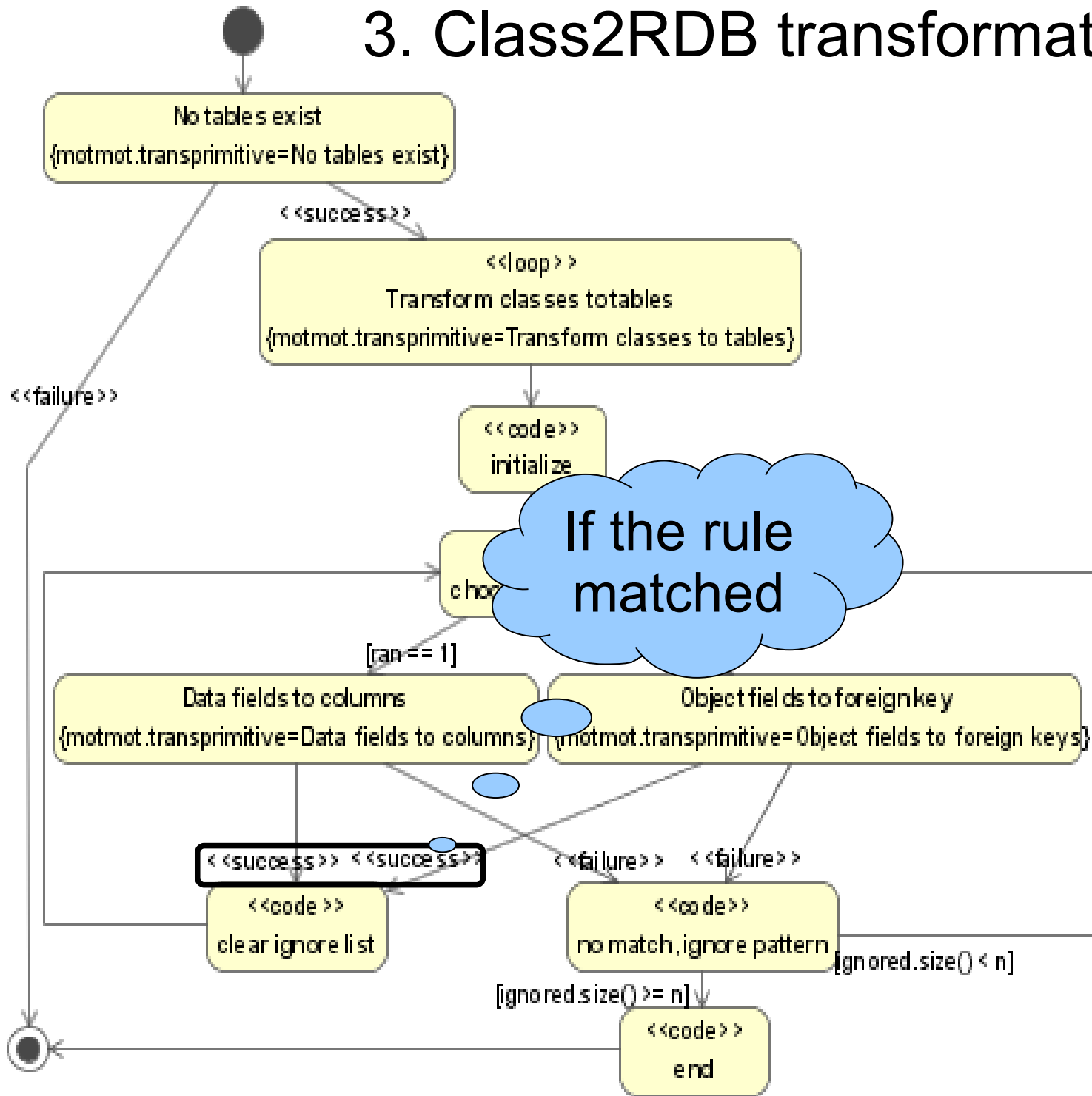
# 3. Class2RDB transformation



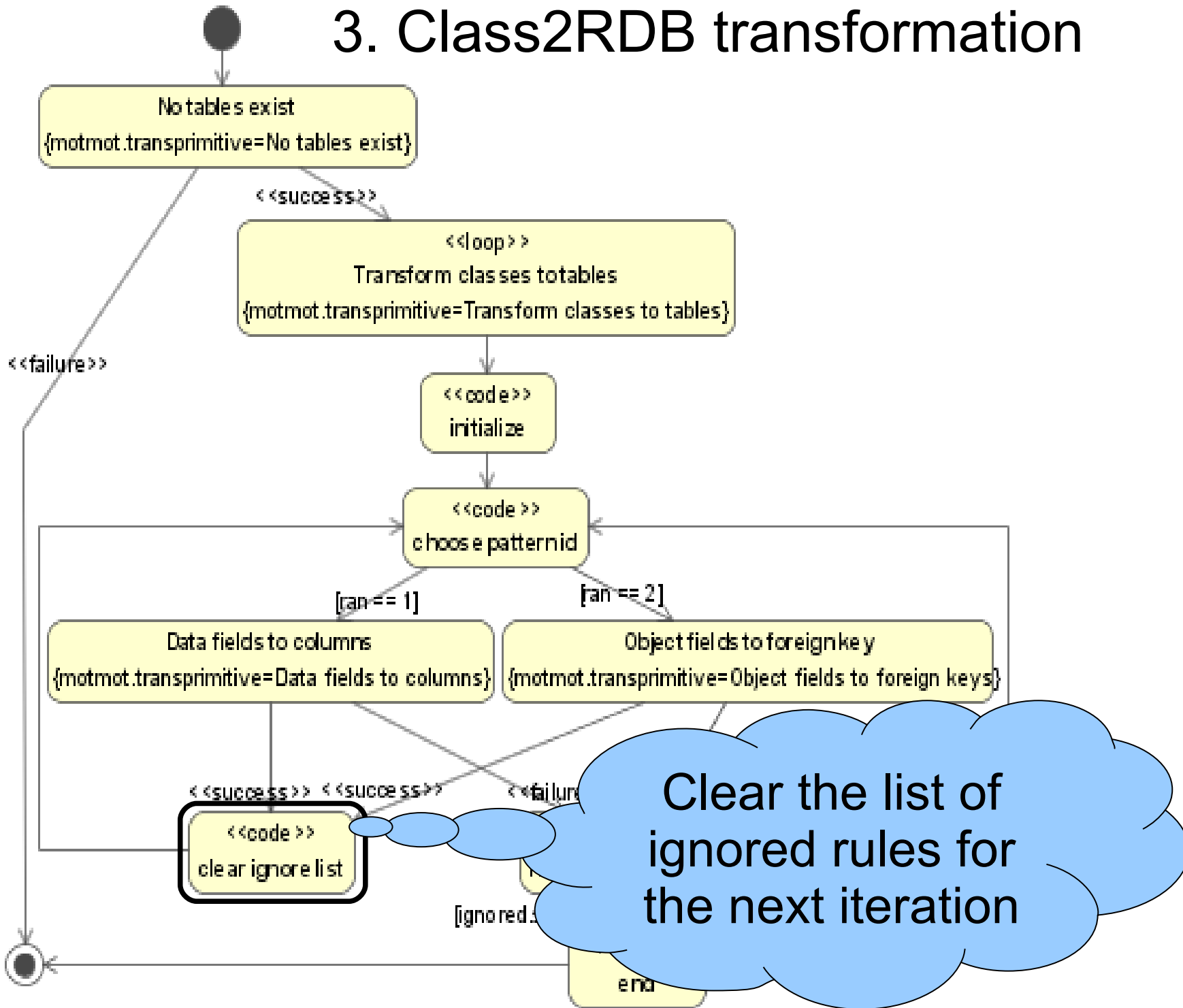
# 3. Class2RDB transformation



# 3. Class2RDB transformation

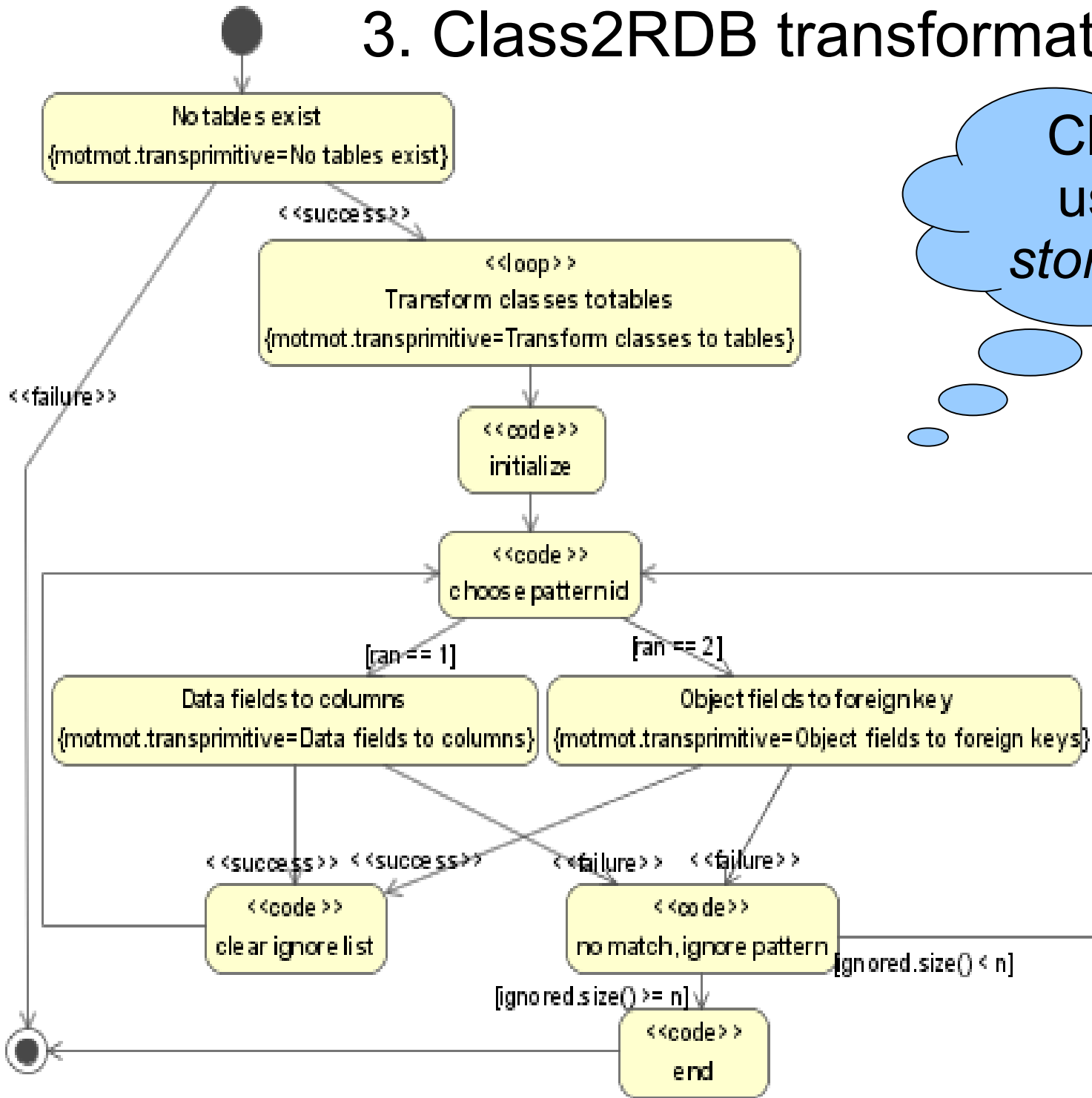


# 3. Class2RDB transformation

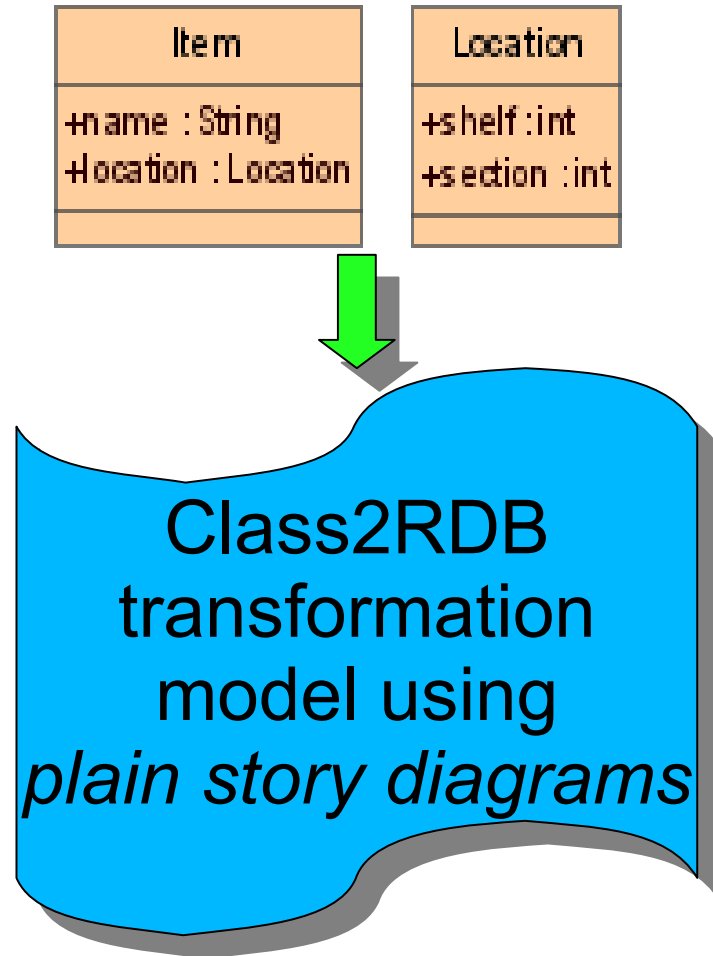


### 3. Class2RDB transformation

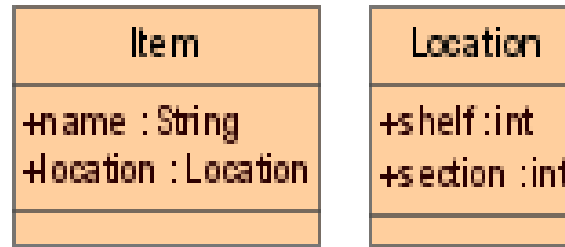
Class2RDB  
using *plain*  
*story diagrams*



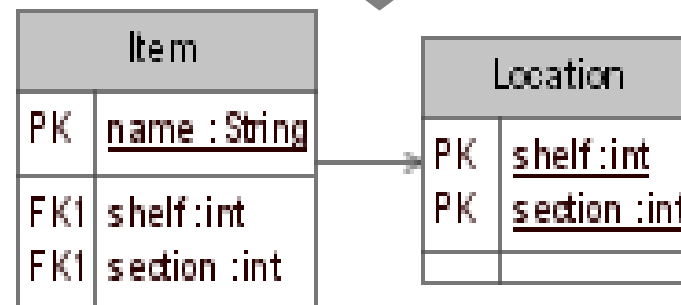
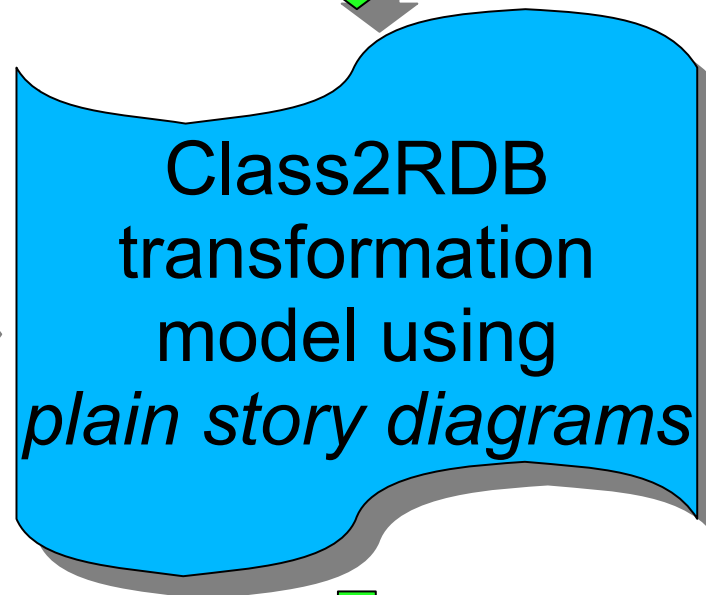
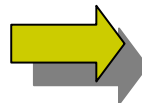
## 4. Implementing hybrid rule scheduling



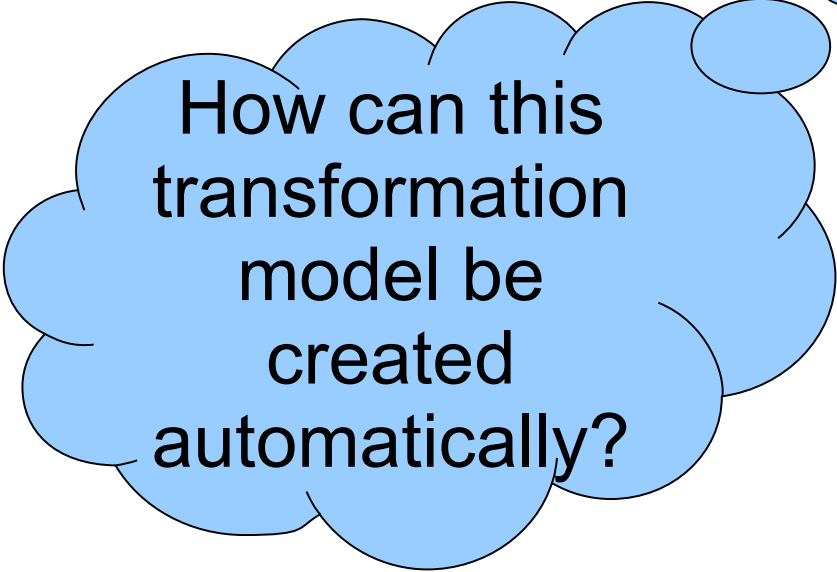
## 4. Implementing hybrid rule scheduling



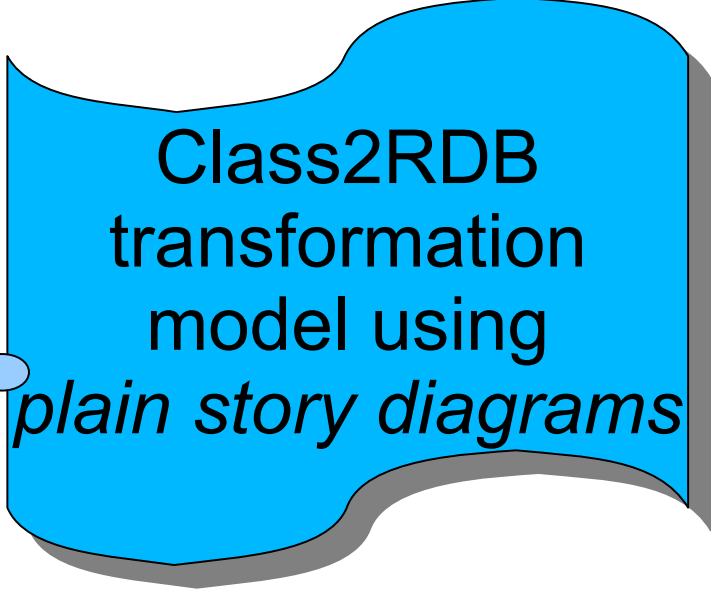
MoTMoT



## 4. Implementing hybrid rule scheduling



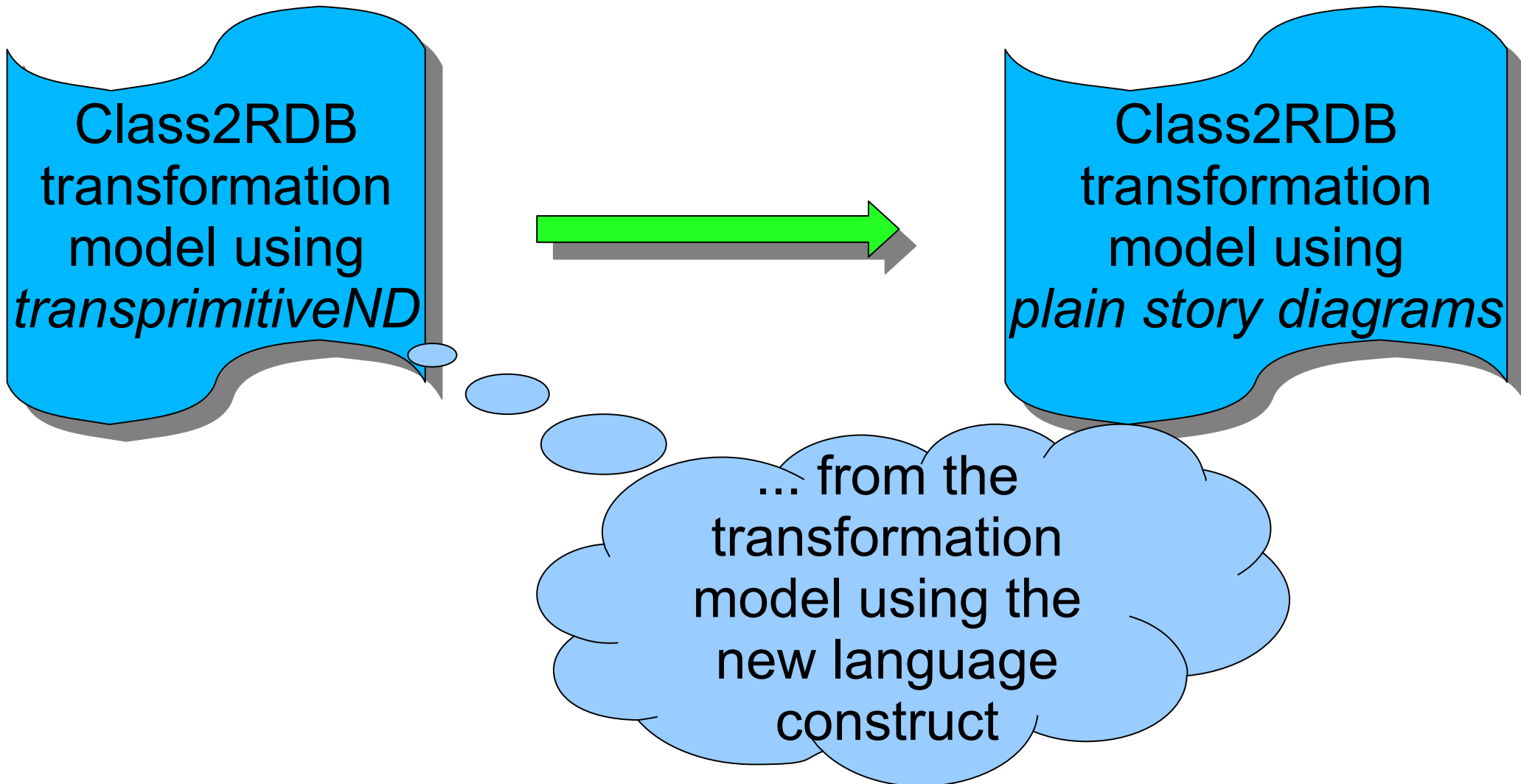
How can this transformation model be created automatically?



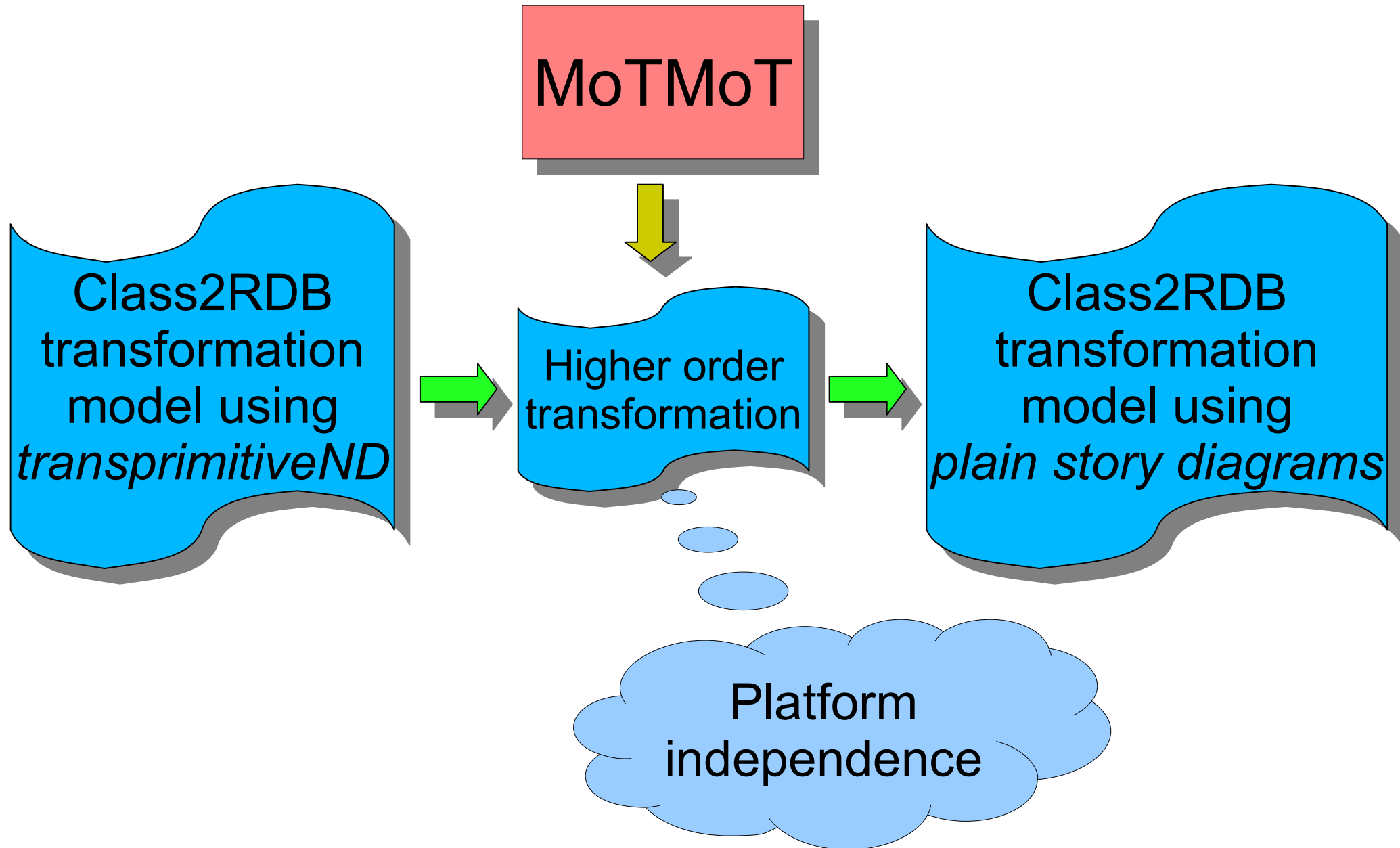
Class2RDB transformation model using *plain story diagrams*



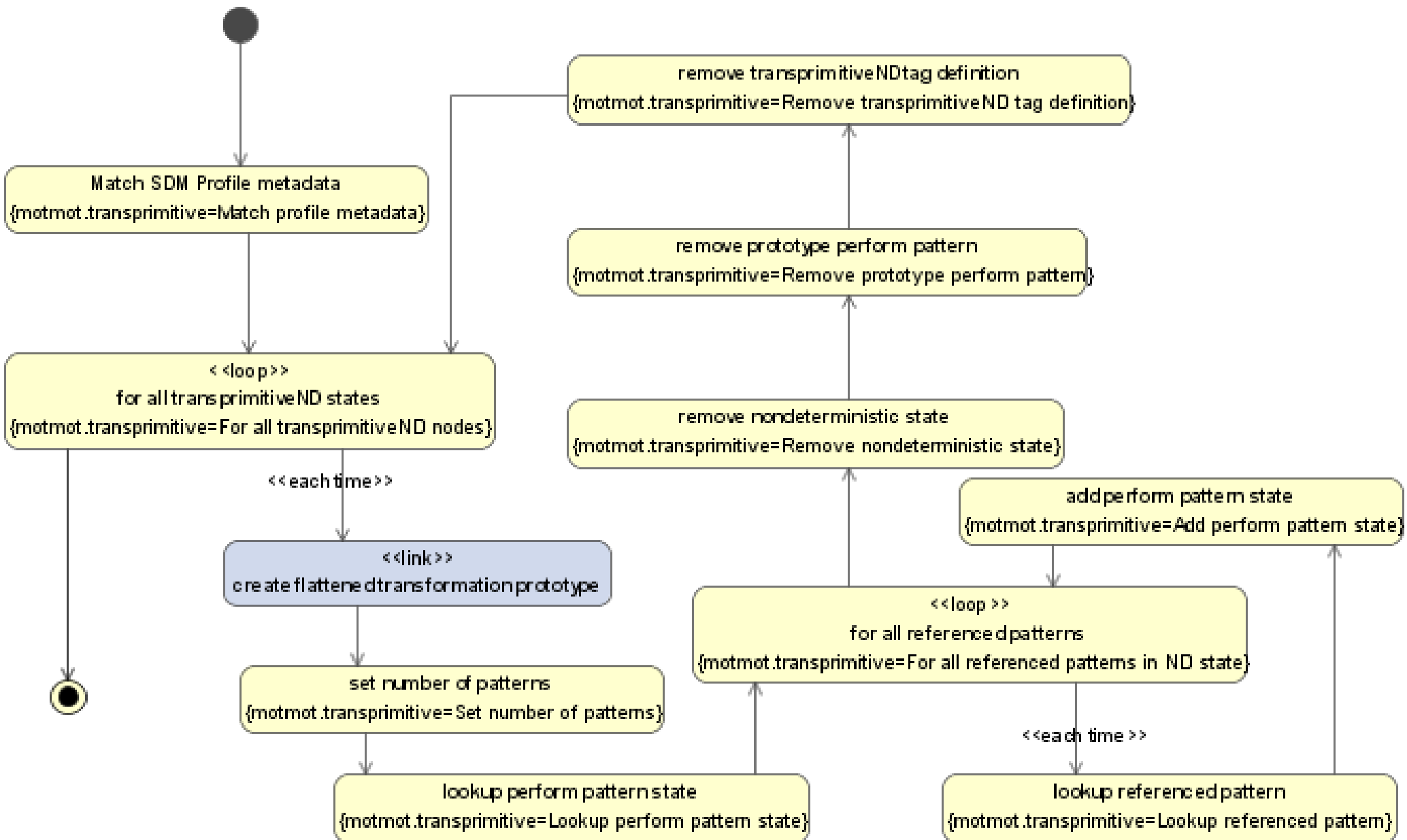
## 4. Implementing hybrid rule scheduling



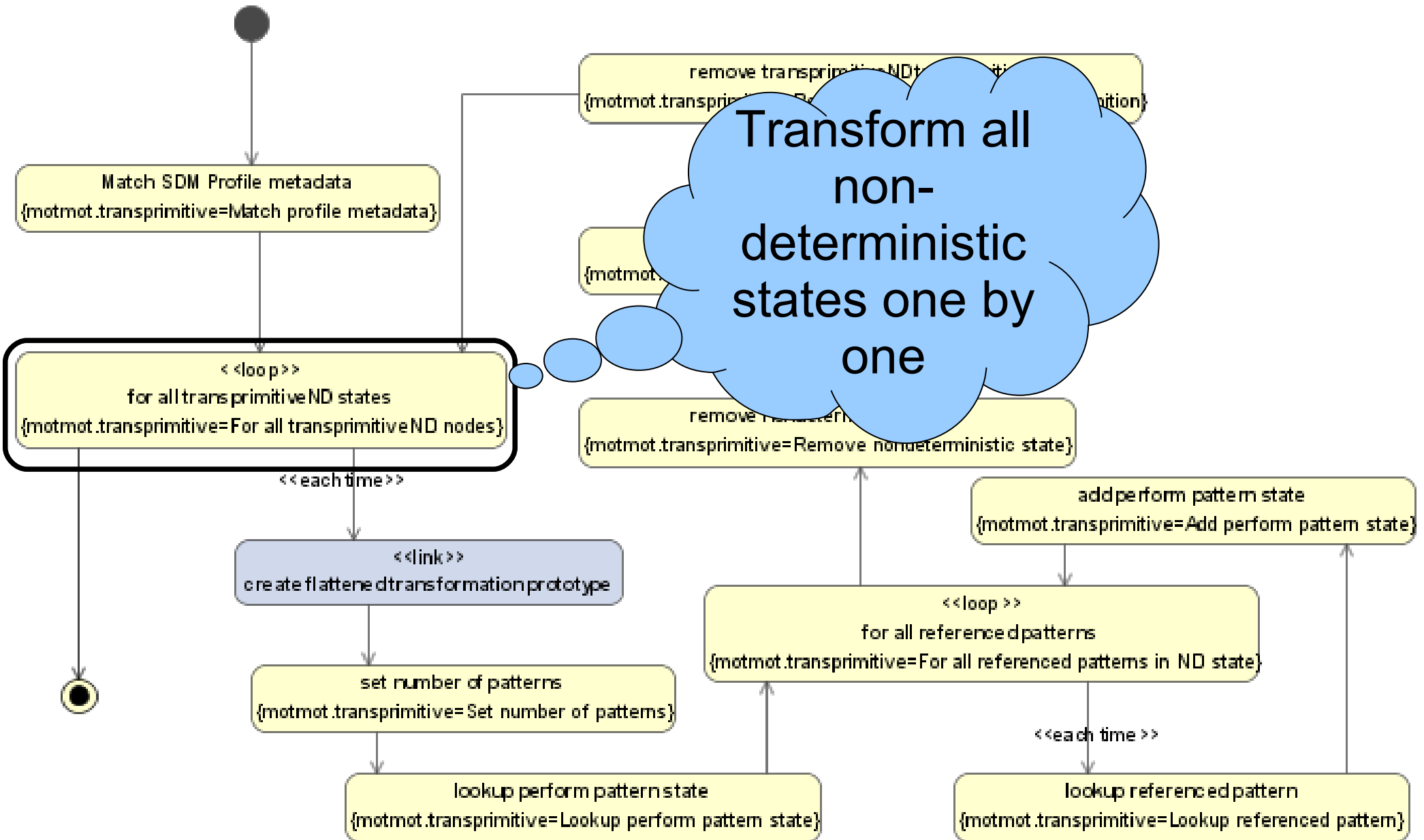
## 4. Implementing hybrid rule scheduling



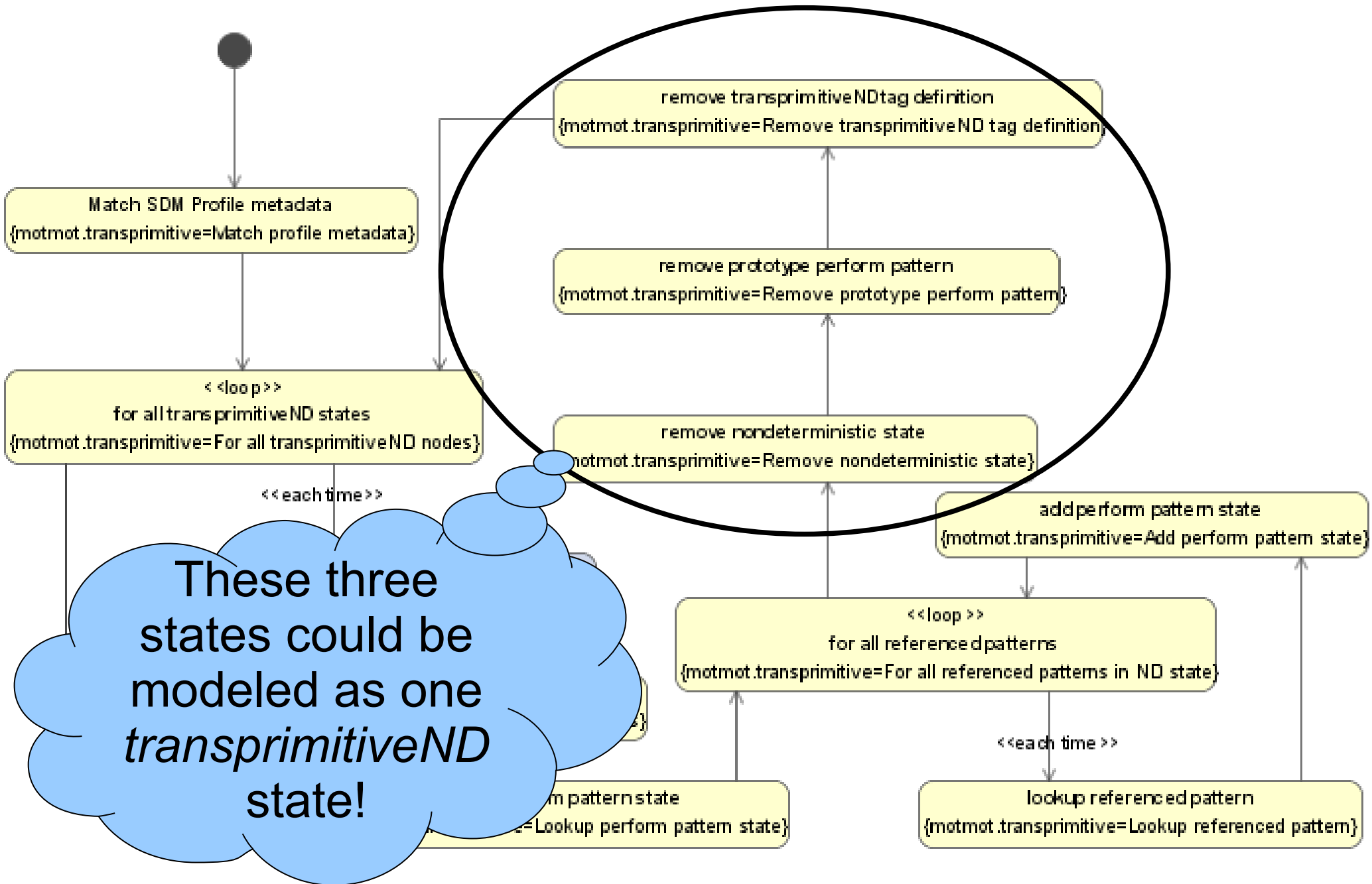
# 4. Implementing hybrid rule scheduling



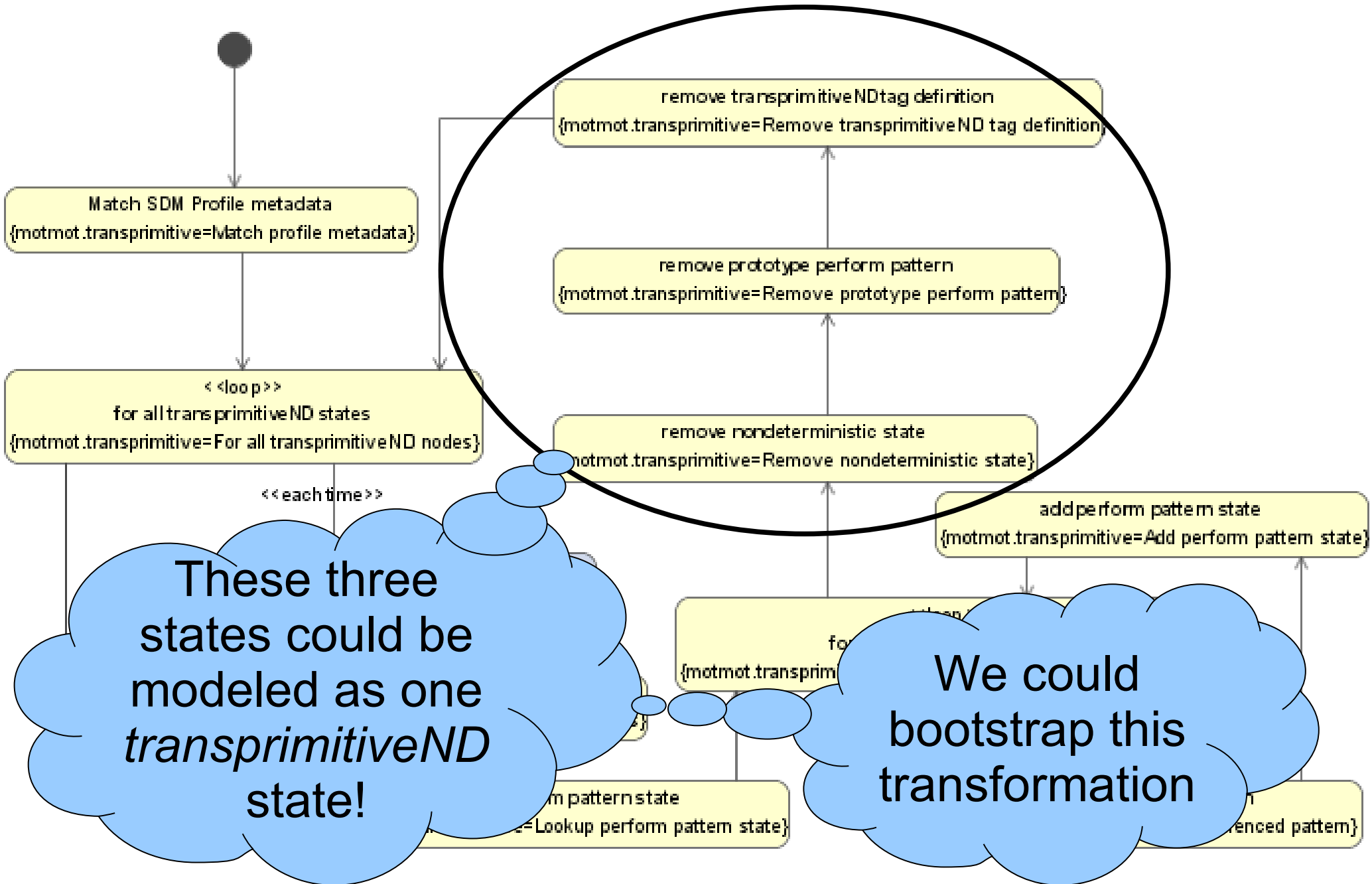
# 4. Implementing hybrid rule scheduling



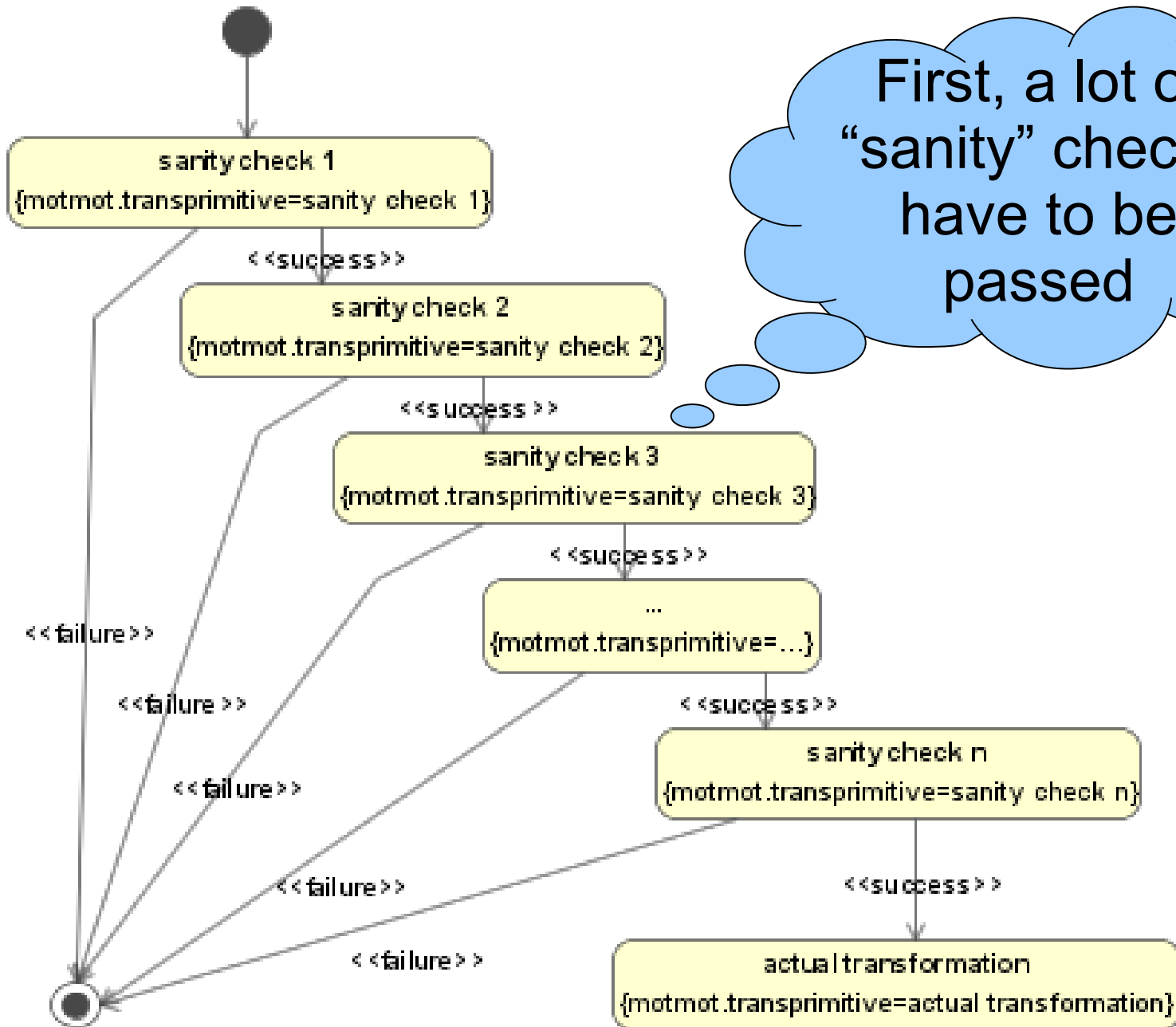
# 4. Implementing hybrid rule scheduling



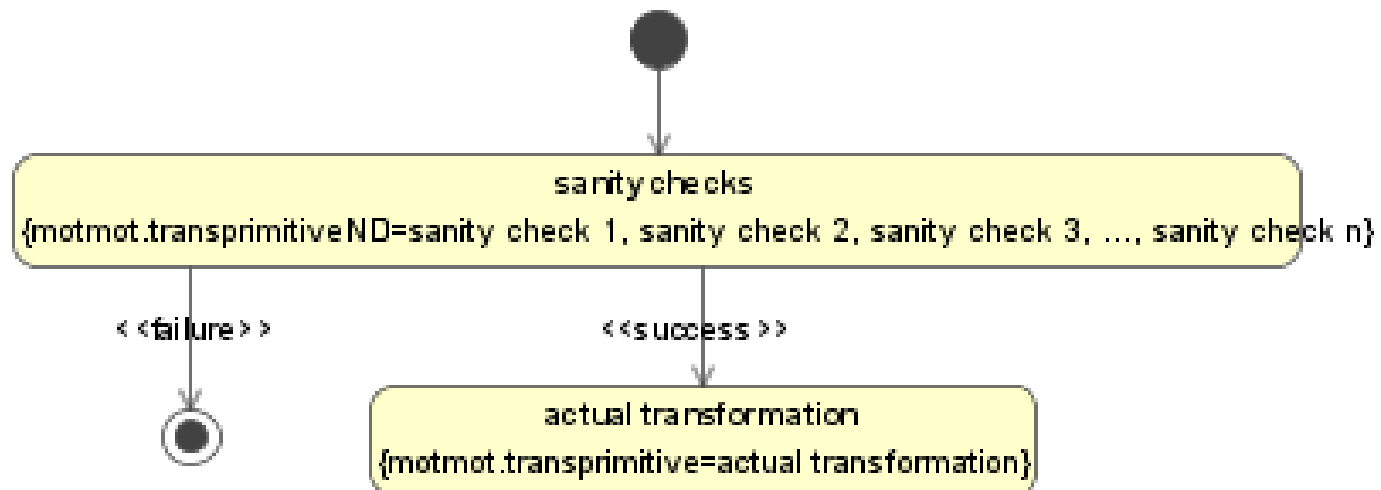
# 4. Implementing hybrid rule scheduling



# 5. Special case: sanity checks

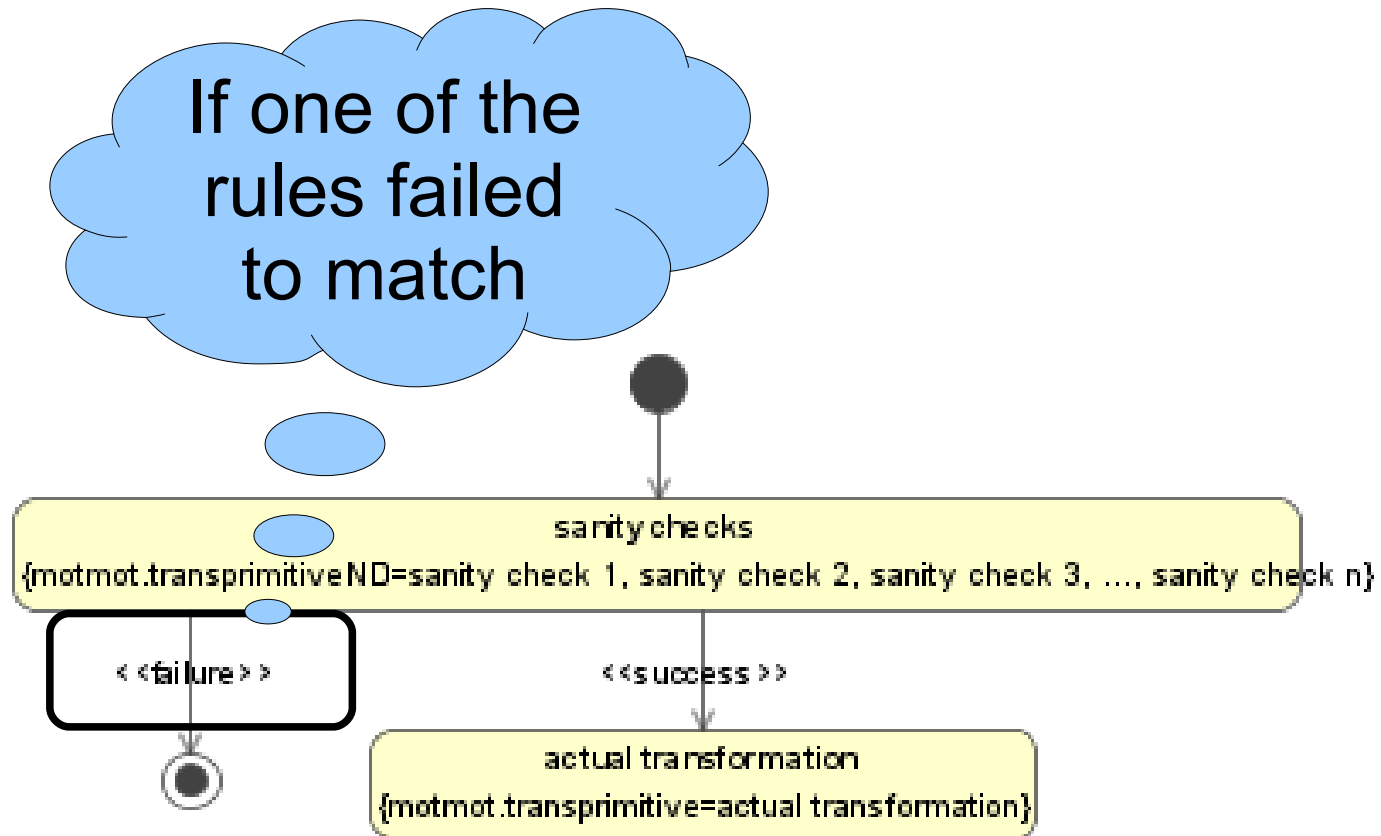


## 5. Special case: sanity checks

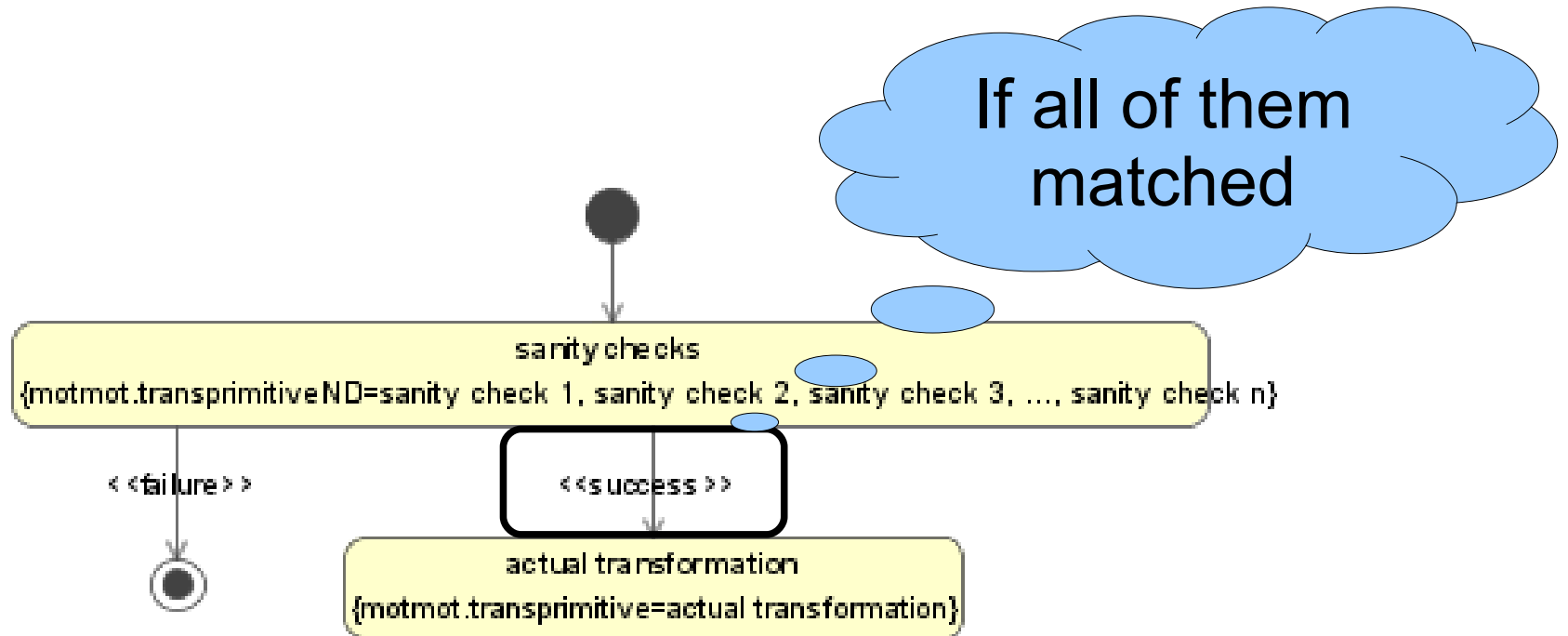




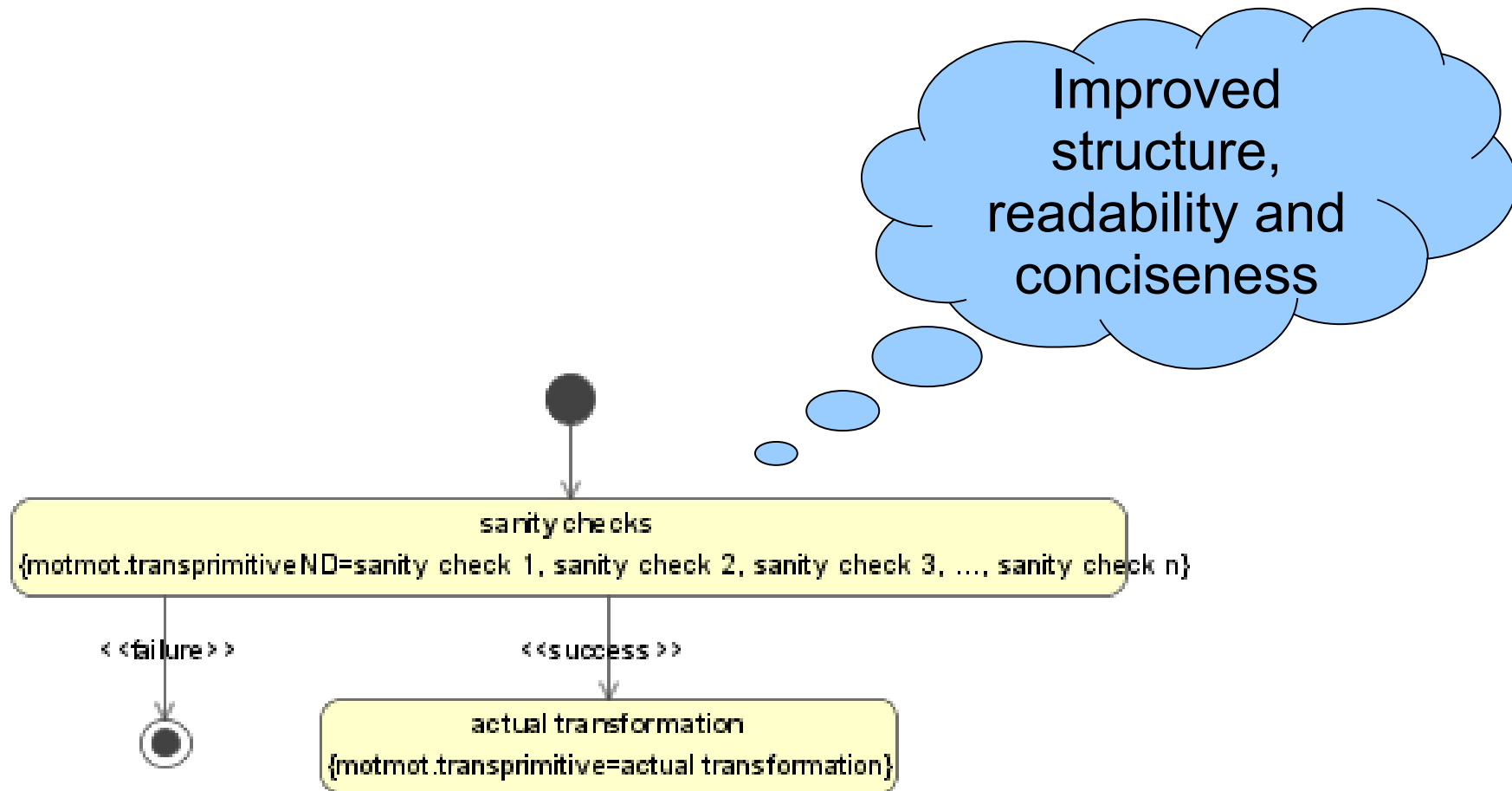
## 5. Special case: sanity checks



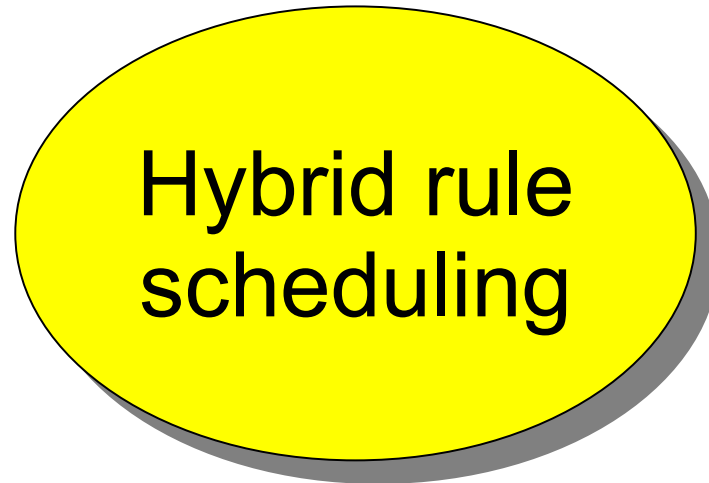
## 5. Special case: sanity checks



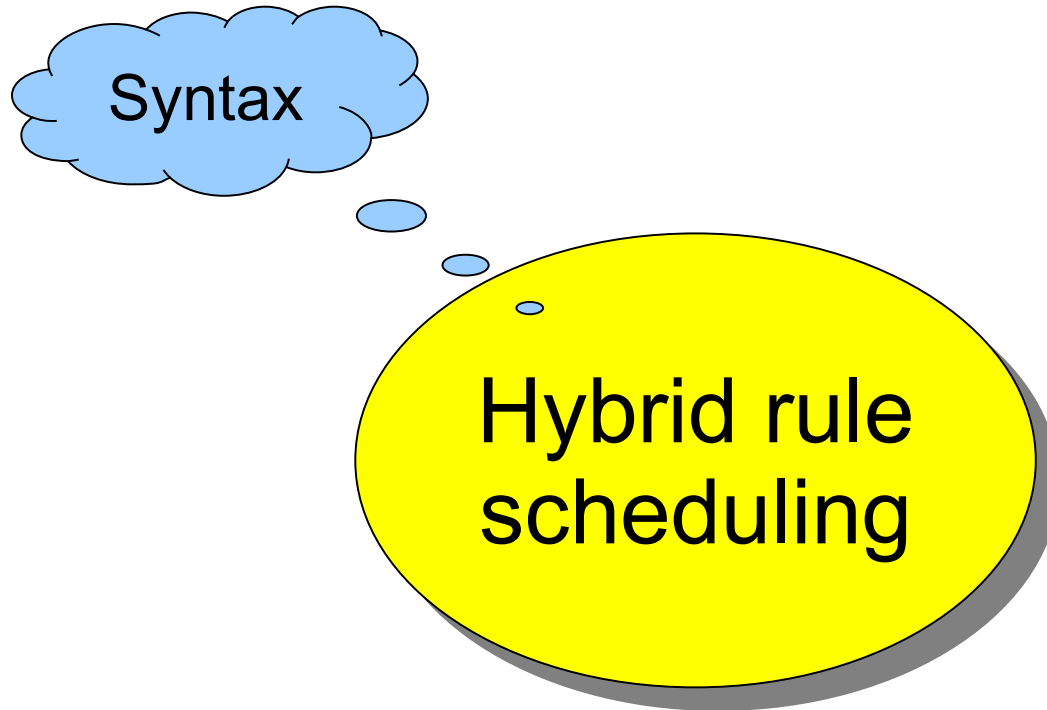
## 5. Special case: sanity checks



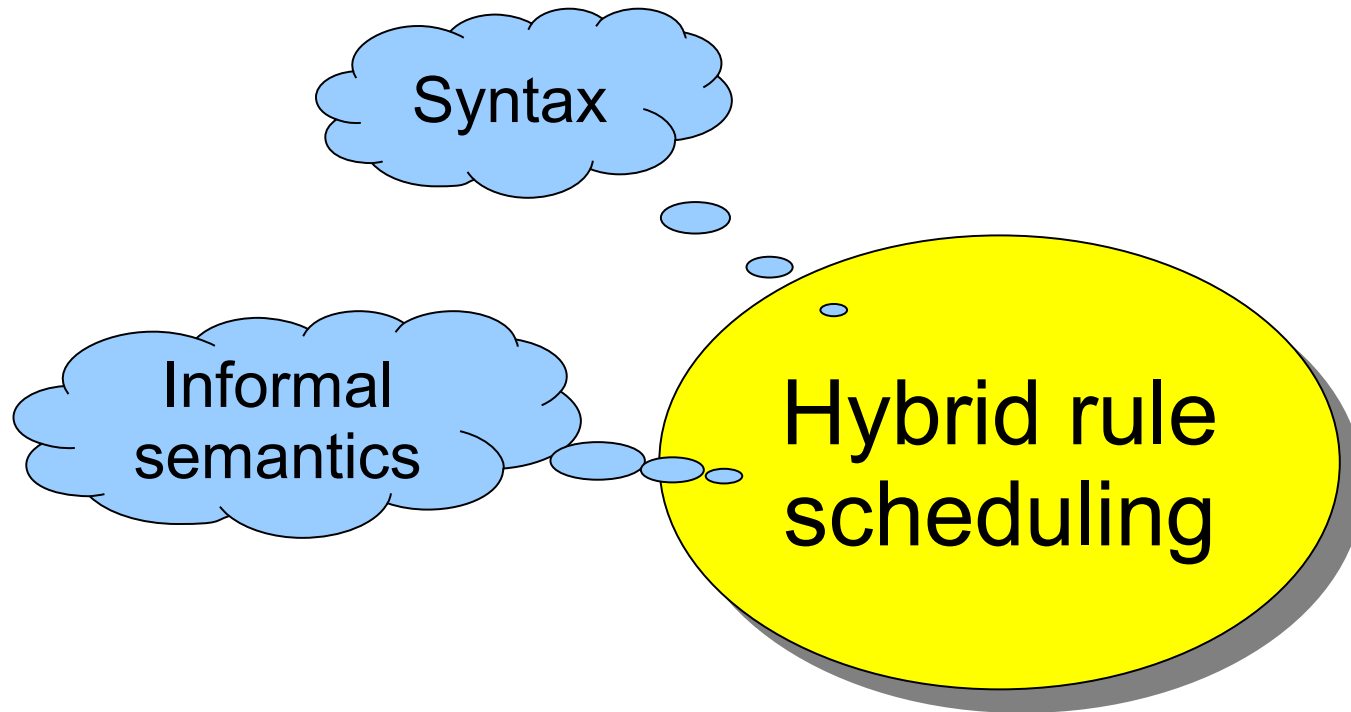
## 6. Conclusion



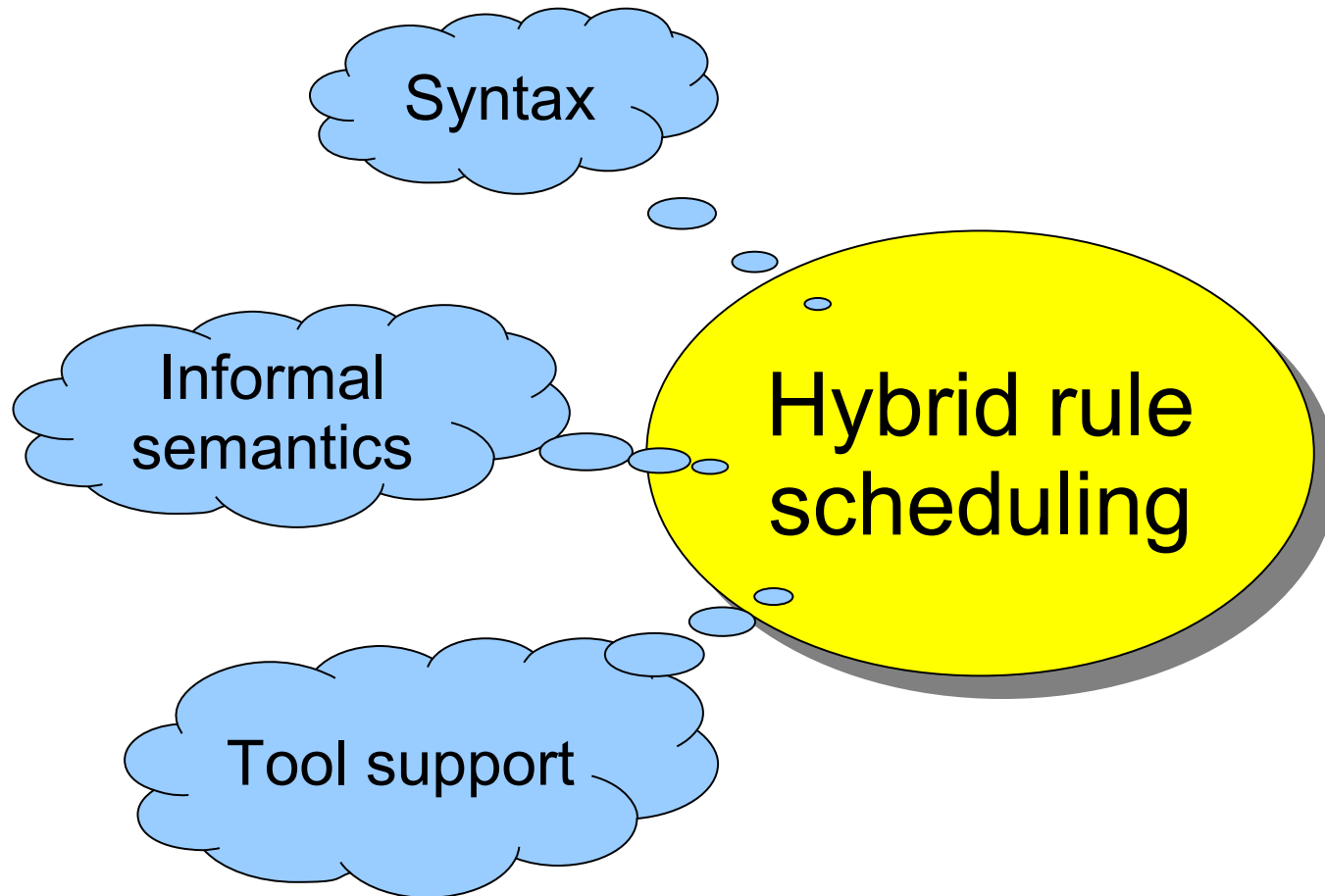
## 6. Conclusion



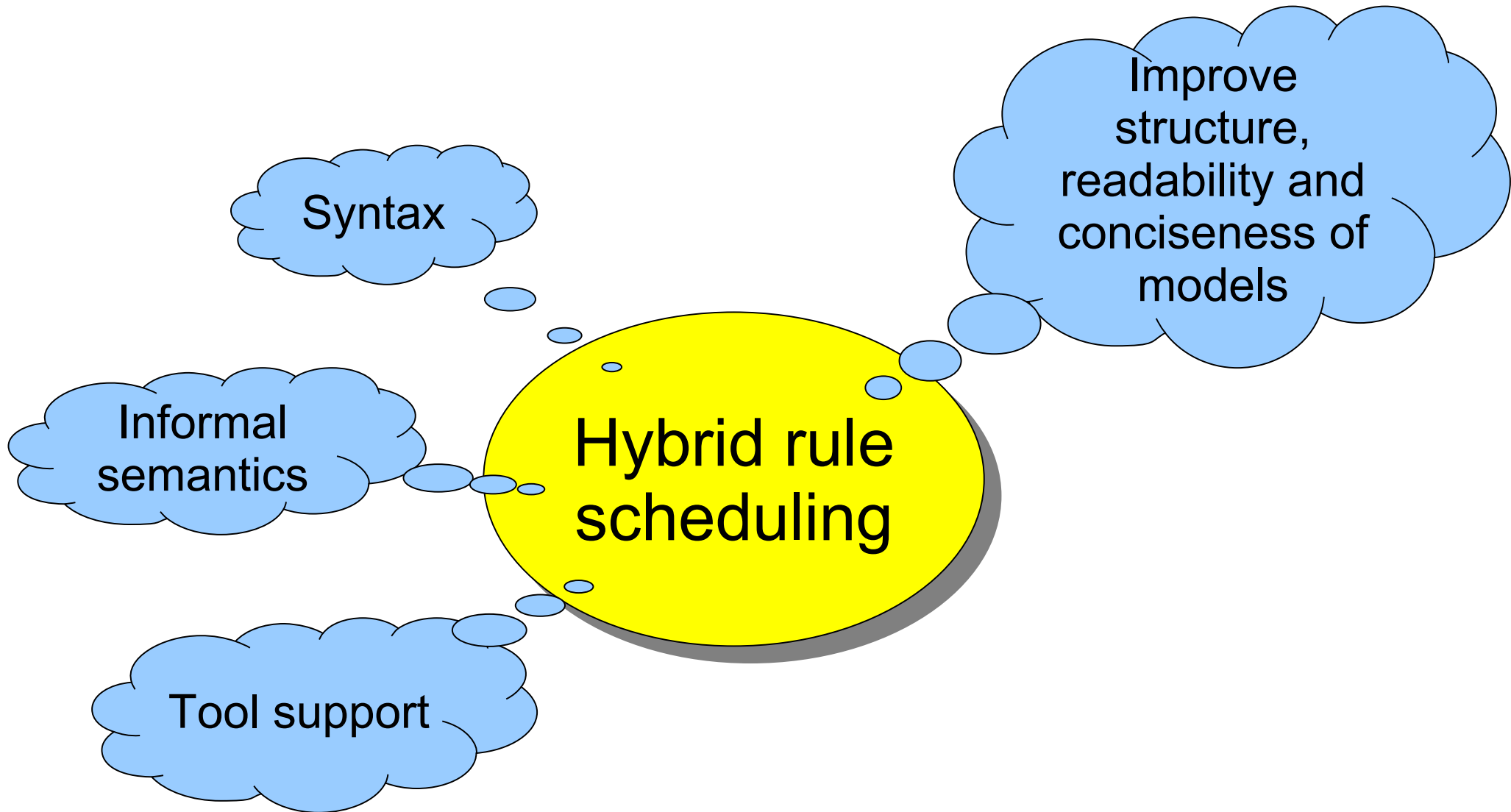
## 6. Conclusion



## 6. Conclusion

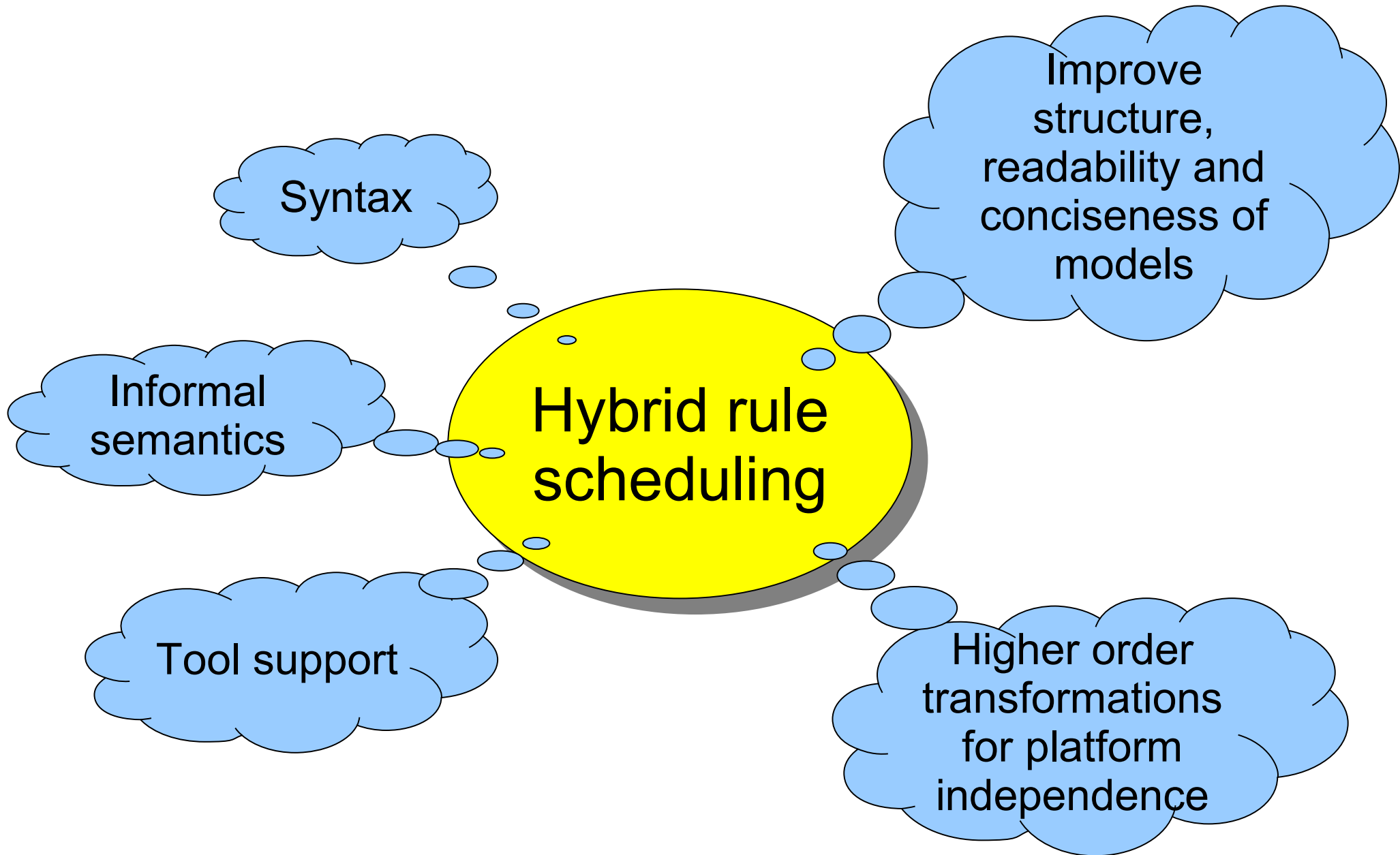


## 6. Conclusion





## 6. Conclusion



## 6. Conclusion

Higher order  
transformations  
for platform  
independence?

Hybrid  
transformation  
language?

Discussion

True  
non-determinism  
or an imposed  
choice?