Improving the Understandability of Formal Specifications

An Experience Report

Felix Kossak, Atif Mashkoor, Verena Geist, Christa Illibauer

+43 7236 3343 811
felix.kossak@scch.at
www.scch.at
Contents

- Motivation
- Specific Suggestions
- Conclusion
Motivation

- Rigorous methods improve software quality
- But they are hard to sell
  - Stakeholders of a specification: domain experts, managers, lawyers, developers, ...
  - A specification is part of a contract
  - The first project phase is slower
  - It takes longer until something “can be seen”

- Key Problems:
  - Notation
  - Style
Suggestions

- Notation: Abstract State Machines
- Style:
  - Top-down
  - Identifiers
  - Bracketing
  - Keywords
  - Structure of expressions
  - Set expressions
- Flexibility
**Identifiers**

```plaintext
rule ConsumeOneToken(in, i) =
     choose t in toksInSFForInst(in, i) do
     remove t from toksInSF(in)

rule ConsumeOneToken(incomingSequenceFlow, instance) =
     choose token in tokensInSequenceFlowForInstance(
         incomingSequenceFlow, instance) do
     remove token from tokensInSequenceFlow(
         incomingSequenceFlow)
```
rule FlowNodeBehaviour(flowNode) =
    if eventCondition(flowNode)
    and controlCondition(flowNode)
    and dataCondition(flowNode)
    and resourceCondition(flowNode) then
        DataOperation(flowNode)
        ControlOperation(flowNode)
        EventOperation(flowNode)
        ResourceOperation(flowNode)
rule FlowNodeBehaviour(flowNode) =
    if eventCondition(flowNode)
        and controlCondition(flowNode)
        and dataCondition(flowNode)
        and resourceCondition(flowNode) then
        parallelblock
            DataOperation(flowNode)
            ControlOperation(flowNode)
            EventOperation(flowNode)
            ResourceOperation(flowNode)
        endparallelblock
    endif
Bracketing (3)

```plaintext
rule FlowNodeBehaviour(flowNode) =
  if eventCondition(flowNode)
    and controlCondition(flowNode)
    and dataCondition(flowNode)
    and resourceCondition(flowNode) then
  do in parallel
    DataOperation(flowNode)
    ControlOperation(flowNode)
    EventOperation(flowNode)
    ResourceOperation(flowNode)
```
Keywords and Structure

- `forall` token `do` `instanceOfToken(token) := instance`
  
  `forall` token `instanceOfToken(token) = instance`

  `foreach` token `do` ...
  `foreach` token `holds` ...

  `forsome` token `holds` ...

- `do` `completionQuantity(flowNode)` `times` `ProduceToken(...)`
\{ \text{node} \mid \text{node} \in \text{eventGateTargetNodes}(\ldots) \text{ and } \ldots \}\} \\

The set containing each \text{node} \ for which holds \text{node is in eventGateTargetNodes}(\ldots) \text{ and } \ldots
Set Expressions (2)

\[
\{ \text{instanceOfToken}(\text{token}) \mid
\begin{array}{l}
\text{forsome} \text{ sequenceFlow} \in \text{incomingSequenceFlows}(\text{flowNode}) \text{ holds} \\
\text{token} \in \text{sequenceFlow}
\end{array}
\}
\]

\[
\{ \text{instance} \mid
\begin{array}{l}
\text{forsome} \text{ token} \text{ holds} \\
\text{forsome} \text{ sequenceFlow} \in \text{incomingSequenceFlows}(\text{flowNode}) \text{ holds} \\
\text{token} \in \text{sequenceFlow} \text{ and} \\
\text{instanceOfToken}(\text{token}) = \text{instance}
\end{array}
\}
\]
Rigorous methods can and should be used in all kinds of software projects
Rigorous methods can and should be made more generally understandable
Rigorous methods can be introduced “gently”