

folgende Begriffe sollen definiert werden:

Visual Programming Language

Grammatik

Domain Specific Language

Schleifen

Flowchart

(Fixpunktberechnung)

Contents

1	Einleitung	3
2	Grammatikanalyse	4
2.1	Prüfungslogik	4
2.2	Datenverarbeitungs	5
2.3	Typsystem	6
3	Implementierung	7
3.1	1. Lösungsansatz	7
4	Evaluation	8
4.1	8
4.1.1	1. Lösungsansatz	8

1 Einleitung

test

2 Grammatikanalyse

2.1 Prüfungslogik

$\langle \text{ActivityModel} \rangle ::= \langle \text{Activity} \rangle^* \langle \text{ActivityConnection} \rangle$

$\langle \text{Activity} \rangle ::= \langle \text{ActivityStart} \rangle \mid \langle \text{ActivityAction} \rangle \mid \langle \text{ActivityCondition} \rangle \mid \langle \text{ActivityDisplay} \rangle$

$\langle \text{ActivityStart} \rangle ::= \epsilon$

$\langle \text{ActivityAction} \rangle ::= \langle \text{ActivityFlowCall} \rangle \mid \langle \text{ActivityPitaBuildInforRequest} \rangle \mid \langle \text{ActivityLoadExternalData} \rangle$

$\langle \text{ActivityFlowCall} \rangle ::= \text{ref}(\text{FlowTemplate}) \langle \text{ActivityPortValue} \rangle^* \langle \text{TemplateParameterValue} \rangle^* \langle \text{ValueTransformation} \rangle^*$

$\langle \text{ActivityPitaBuildInforRequest} \rangle ::= \langle \text{string abdFilename} \rangle \langle \text{string requestAlias} \rangle \langle \text{string expectedSystems} \rangle^* \langle \text{number timeout} \rangle$

$\langle \text{ActivityLoadExternalData} \rangle ::= \langle \text{Type dataType} \rangle \langle \text{string dataSource} \rangle$

$\langle \text{ActivityPortValue} \rangle ::= \langle \text{FlowPortValue} \rangle \mid \langle \text{ActivityPortRefernce} \rangle$

$\langle \text{FlowPortValue} \rangle ::= \langle \text{string} \rangle \mid \langle \text{number} \rangle \mid \langle \text{bool} \rangle \mid \langle \text{date} \rangle \mid \langle \text{FlowPortValue} \rangle^*$

$\langle \text{ActivityPortRefernce} \rangle ::= \text{ref}(\text{ActivityAction}) \langle \text{ValueTransformation} \rangle^*$

$\langle \text{ValueTransformation} \rangle ::= \langle \text{string objectReference} \rangle \mid \langle \text{number listIndex} \rangle$

$\langle \text{ActivityCondition} \rangle ::= \langle \text{ActivityBinaryCondition} \rangle \mid \langle \text{ActivityValidityCondition} \rangle$

$\langle \text{ActivityBinaryCondition} \rangle ::= \text{ref}(\text{FlowTemplate}) \langle \text{ActivityBinaryConditionOperator} \rangle \langle \text{ActivityPortValue left} \rangle \langle \text{ActivityPortValue right} \rangle$

$\langle \text{ActivityValidityCondition} \rangle ::= \langle \text{ActivityPortValue} \rangle^*$

$\langle \text{ActivityBinaryCondition} \rangle ::= '=' \mid '\neq' \mid '<' \mid '\leq' \mid '>' \mid '\geq'$

$\langle \text{ActivityDisplay} \rangle ::= \langle \text{ActivityDisplayField} \rangle^*$

$\langle \text{ActivityDisplayField} \rangle ::= \langle \text{string label} \rangle \langle \text{string color} \rangle \text{ref}(\text{ActivityAction})$

Grammatik TODO Aktivitätsmodell

2.2 Datenverarbeitungen

$\langle \text{FlowInstance} \rangle ::= \langle \text{FlowOutputPort } \lambda \text{Arguments} \rangle^* \langle \text{FlowInputPort } \lambda \text{Arguments} \rangle^*$

$\langle \text{FlowLambda} \rangle ::= \langle \text{FlowOutputPort } \lambda \text{Arguments} \rangle^* \langle \text{FlowInputPort } \lambda \text{Arguments} \rangle^*$

$\langle \text{FlowInputPort} \rangle ::= \langle \text{string name} \rangle \langle \text{Type} \rangle \langle \text{bool acceptsError} \rangle$

$\langle \text{FlowOutputPort} \rangle ::= \langle \text{string name} \rangle \langle \text{Type} \rangle \langle \text{bool producesError} \rangle$

Grammatik TODO Flow-Instanz

$\langle \text{FlowTemplate} \rangle ::= \langle \text{Flow} \rangle \langle \text{TemplateParameter} \rangle^*$

$\langle \text{Flow} \rangle ::= \langle \text{LibraryFlow} \rangle \mid \langle \text{FlowModel} \rangle$

$\langle \text{LibraryFlow} \rangle ::= \epsilon$

$\langle \text{TemplateParameter} \rangle ::= \text{'String'} \mid \text{'Number'} \mid \text{'Bool'} \mid \langle \text{TemplateParameterList} \rangle$

$\langle \text{TemplateParameterList} \rangle ::= \langle \text{TemplateParameter} \rangle$

Grammatik TODO Flow-Template

$\langle \text{FlowModel} \rangle ::= \langle \text{FlowInstance} \rangle \langle \text{FlowNode} \rangle^* \langle \text{FlowConnection} \rangle^*$

$\langle \text{FlowNode} \rangle ::= \langle \text{FlowNodeOutput} \rangle \mid \langle \text{FlowNodeInput} \rangle \mid \langle \text{FlowNodeLambda} \rangle \mid \langle \text{FlowNodeFlowCall} \rangle$

$\langle \text{FlowNodeOutput} \rangle ::= \text{ref}(\text{FlowOutputPort}) \langle \text{FlowPortValue} \rangle$

$\langle \text{FlowNodeLambda} \rangle ::= \text{ref}(\text{FlowLambda}) \langle \text{FlowPortValue} \rangle^*$

$\langle \text{FlowNodeFlowCall} \rangle ::= \text{ref}(\text{FlowTemplate}) \langle \text{FlowPortValue} \rangle^* \langle \text{TemplateParameterValue} \rangle^*$

$\langle \text{FlowConnection} \rangle ::= \text{ref}(\text{FlowOutputPort source}) \text{ref}(\text{FlowOutputPort target})$

$\langle \text{FlowConnection} \rangle ::= \text{ref}(\text{FlowOutputPort source}) \text{ref}(\text{FlowOutputPort target})$

$\langle \text{TemplateParameterValue} \rangle ::= \langle \text{string} \rangle \mid \langle \text{number} \rangle \mid \langle \text{bool} \rangle \mid \langle \text{TemplateParameterValueList} \rangle$

$\langle \text{TemplateParameterValueList} \rangle ::= \langle \text{TemplateParameterValue} \rangle^*$

Grammatik TODO Flow-Modell

2.3 Typsystem

$\langle Type \rangle ::= \langle TypePrimitive \rangle \mid \langle TypeOptional \rangle \mid \langle TypeList \rangle \mid \langle TypeObject \rangle$

$\langle TypePrimitive \rangle ::= \text{'String'} \mid \text{'Number'} \mid \text{'Bool'} \mid \text{'Data'} \mid \text{'PtiaResponse'}$

$\langle TypeOptional \rangle ::= \langle Type \rangle \text{'?'}$

$\langle TypeList \rangle ::= \langle Type \rangle \text{'[]'}$

$\langle TypeObject \rangle ::= \text{'{' } (\langle string\ key \rangle \text{' : ' } \langle Type \rangle)^* \text{'}'}$

$\langle TypeGeneric \rangle ::= \text{'$'} \langle string\ genericName \rangle$

$\langle TypeReference \rangle ::= \text{ref}(\text{Type})$

Grammatik TODO Typ-Defintion mit generischen und Referenz-Typen

3 Implementierung

3.1 1. Lösungsansatz

Der Benutzer gibt von vorneherein eine Zahl a an, welche die maximale Anzahl von Schleifendurchläufe beschränkt. Für die Zahl muss dafür folgendes gelten TODO. Die Idee des Ansatzes ist es, den Schleifenkörper nicht Iterativ oder Rekursiv ausführen, sondern a -mal auszurollen. Dafür wird der Schleifenkörper und die nachfolgenden Anweisungen a -mal kopiert. Die Schleife wird dadurch nicht dynamisch ausgeführt, sondern statisch in den Code implementiert. Dadurch entstehen $a+1$ Graphen. Jeder dieser Graphen repräsentiert eine ursprüngliche Iteration. Dabei werden die einzelnen Graphen mit ihren direkten Nachbarn verbunden. Da die aktuell zugrunde liegende Implementierung deterministisch ist und aktuell nur auf die gleiche Eingabe zugegriffen werden kann, muss ein Mechanismus implementiert werden, welcher den aktuellen Sensorwert ausliest und diesen an die nachfolgenden Anweisungen weitergibt. Dieser Vorgang muss für jede neu eingefügte Verbindung wiederholt werden. Dieser Ansatz wird auch von Ye et al. im Konferenz-Paper "Loop Unrolling Based on SLP and Register Pressure Awareness" beschrieben.

Auf die Vor- und Nachteile der Implementierung wird im Kapitel 4 eingegangen.

4 Evaluation

4.1

4.1.1 1. Lösungsansatz

+einfach zu implementieren +keine Endlosschleife +keine Zyklen +weniger Sprünge
+möglicher Performance gewinn -größerer Codeumfang -höherer verbraucht an
ressourcen zB Speicher -möglicherweise ineffizient, wenn der faktor zu groß
gewählt wird -schlechtere Lesbarkeit - wenn bereits nach 3 durchlaufen fest-
steht, dass das gewünschte ergebniss nicht mehr erreicht werden kann werden
trotzdem die restlichen schritte ausgeführt