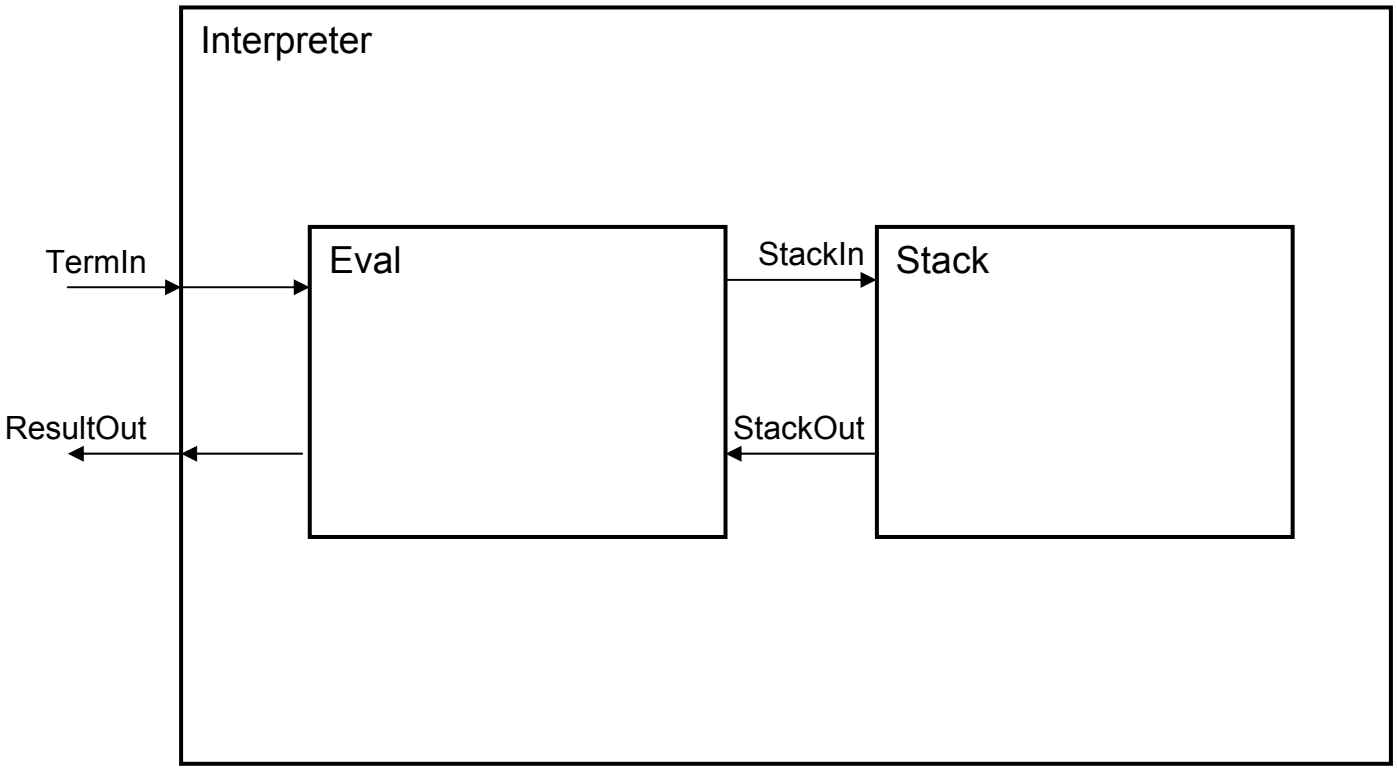
Aufgabe 1a: Architektur des Interpreters

Dekomposition in stromverarbeitende Funktionen



Aufgabe 1b: Schnittstellenspezifikation

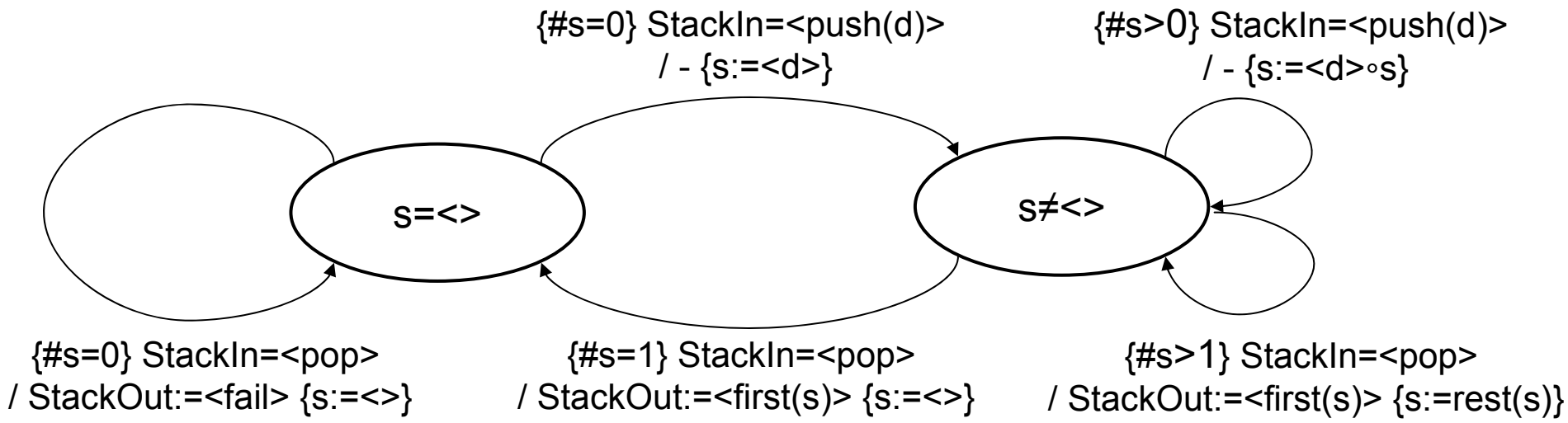
Eingaben

- TermIn = Int∪{+,-,*,/} ∪{◊}
- StackIn = {push(d): d∈Data}∪{pop}

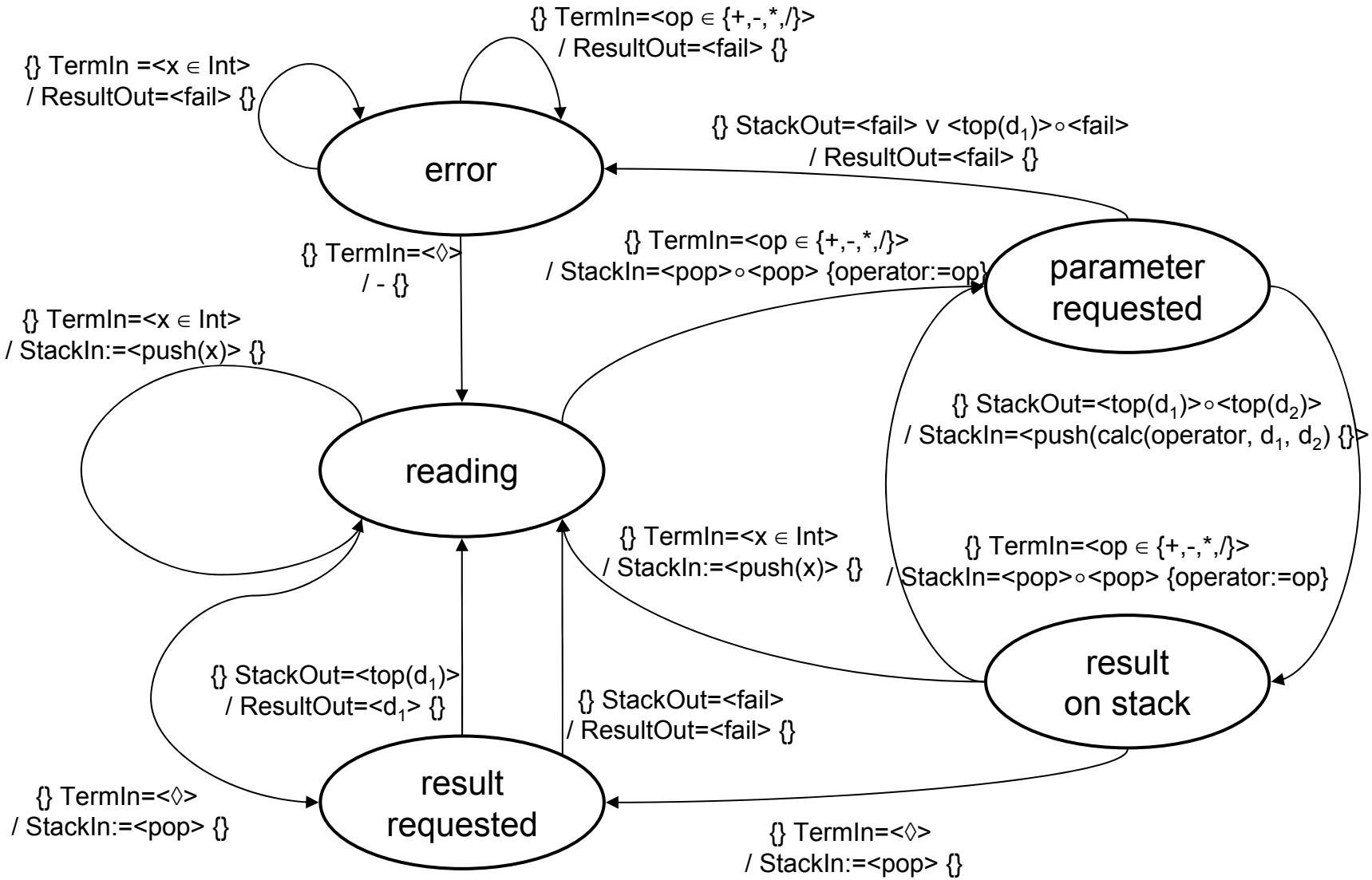
Ausgaben

- ResultOut = Int∪{fail}
- StackOut = {top(d): d∈Data}∪{fail}

Aufgabe 1c: Komponente „Stack“



Aufgabe 1c: Komponente „Eval“



Aufgabe 1d: Verhalten in Focus Notation

Stack

(1) StackOut=Stack(StackIn) ∧ StackIn=<pop> ∘ StackIn´⇒
StackOut=<fail> ∘ StackOut´ ∧ StackOut´ = Stack(StackIn´)

(2) StackOut=Stack(StackIn) ∧ <push(d_1), …,push(d_n), pop> ∘ StackIn´⇒
StackOut=<top(d_n)> ∘ StackOut´ ∧ StackOut´ = Stack(<push(d_1), …,
push(d_{n-1})> ∘ StackIn´)

Aufgabe 1d: Verhalten in Focus Notation

Eval

Interner Zustand:

$\sigma \in$ {reading, parameter requested, result on stack, result requested, error}

operator $\in$ {+,-,*,/}

Eval(TermIn, StackOut) $\triangleq$ EvalState(TermIn, StackOut, „reading“, „+“)

(3) (ResultOut, StackIn)=
EvalState(TermIn, StackOut, σ, operator) ∧ σ =reading ∧ TermIn=<x∈Int> ∘ TermIn´⇒ StackIn=<push(x)> ∘ StackIn´ ∧ σ =reading ∧ (ResultOut, StackIn´) = EvalState(TermIn´, StackOut, σ, operator)

(4) (ResultOut, StackIn)=
EvalState(TermIn, StackOut, σ, operator) ∧ σ =reading ∧ TermIn=<op∈{+,-,*,/}> ∘ TermIn´ ⇒ StackIn=<pop, pop> ∘ StackIn´ ∧ σ =parameter requested ∧ operator=op ∧ (ResultOut, StackIn´) = EvalState(TermIn´, StackOut, σ, operator)

(5) (ResultOut, StackIn)=
EvalState(TermIn, StackOut, σ, operator) ∧ σ =parameter requested ∧ StackOut=<top(d_1), top(d_2)> ∘ StackOut´⇒ StackIn=<push(calc(operator, d_1, d_2))> ∘ StackIn´ ∧ σ =result on stack ∧ (ResultOut, StackIn´) = EvalState(TermIn, StackOut´, σ, operator)

(6) … weitere Transitionen analog

Aufgabe 2a: Modularität von Focus Ausdrücken

Systemverhalten (logische Sicht)

- Das System ist durch eine Konjunktion der Einzelausdrücke gegeben

Interpreter (TermIn) = Eval(TermIn, StackOut) ∧ Stack(StackIn)

Systemverhalten (operationelle Sicht)

- Das Systemverhalten ergibt sich als kleinster Fixpunkt des rekursiven Gleichungssystems
  - (ResultOut, StackIn) = Eval(TermIn, StackOut)
  - StackOut = Stack(StackIn)

[Bro+93] Manfred Broy ++: The Design of distributed Systems – An introduction to FOCUS, Technical Report, TUM-I9202, Technische Universität München, 1993.

Aufgabe 2b, c: Nachweis der Korrektheit

„Conformance Test“

Ausrechnen ob definierter Wert erzielt wird

Beispiel:

- 4 2 3 + 5 - * 1 + = 1

Zu Zeigen:

<1>=Interpreter(<4, 2, 3, +, 5, -, *, 1, +>)