$$\sigma = \{u_1 \mapsto \{length \mapsto 27, n \mapsto u_2\}, u_2 \mapsto \{length \mapsto 13, n \mapsto \emptyset\}\}$$

$$F := \forall\, self : Segment \bullet length(self) > length(n(self))$$

$I[[F]](\sigma, \emptyset)$

$= I[[\forall\, self : Segment \bullet length(self) > length(n(self))]](\sigma, \emptyset) = \{u_1, u_2\}$

$\beta_1 = \{self \to u_1\}$ $\beta_2 \{self \to u_2\}$

For β_1:

$I[[length(self) > length(n(self))]](\sigma, \beta_1)$

$= (I[[length(self)]](\sigma, \beta_1)) \; I[[length(n(self))]](\sigma, \beta_1)$

(σ, β_1)

$= (\sigma(I[[self]](\sigma, \beta_1))(length)) \; \sigma(I[[n(self)]](\sigma, \beta_1))(length)$

① $\beta(self) = u_1$ $\beta(self) = u_1$ ②

$= (\sigma(u_1)(length)) \; \sigma(\sigma(I[[self]](\sigma, \beta_1))(n))(length)$

$= (27, 13) = 27 > 13$

$= True$

Fall β_2:

③ $\mathcal{I}[\![\text{length}(n(\text{self}))]\!](\sigma,\beta_2)$

$= \mathcal{I}[\![\text{length}]\!](\mathcal{I}[\![n(\text{self})]\!](\sigma,\beta_2))(\sigma,\beta_2)$

③ $= \sigma(\mathcal{I}[\![\text{self}]\!](\sigma,\beta_2))(\text{length}\ (*))$

length : Int
$n : C_{0,1}$
$\text{self} \mapsto U_2$
$\sigma(U_2)(n) = \emptyset$

① $= \sigma(\beta_2(\text{self})(\sigma,\beta_2))(n)$, $*$

① $= (13, *)$ $*$ $\mathcal{I}[\![\text{length}(n(\text{self}))]\!](\sigma,\beta_2)$ (R3) (R4)

$= (13, \bot)$, da $\mathcal{I}[\![n(\text{self})]\!] \notin \text{dom}(\sigma)$

und da $\sigma(\mathcal{I}[\![\text{self}]\!](\sigma,\beta))(n) \neq \{u'\}$

- $\mathcal{I}[\![v(F)]\!](\sigma,\beta) = \begin{cases} \underline{u'} & \text{, if } \mathcal{I}[\![F]\!](\sigma,\beta) \in \text{dom}(\sigma) \text{ and } \sigma(\mathcal{I}[\![F]\!](\sigma,\beta))(v) = \{\underline{u'}\} \\ \bot & \text{, otherwise} \end{cases}$ (if $v : C_{0,1}$)

$= (13, \bot)$, da $\mathcal{I}[\![\bot]\!](\sigma,\beta_2) \overset{①}{=} \mathcal{P}(\bot)$

$= \bot$, da $\bot$ und ③ $\emptyset \in \text{dom}(\sigma) = \phi$

$\exists 13 \forall n \in 3$

dh. True und $\bot$

$\Rightarrow \bot$

Exercise 3 – Verification with PD Calculus (10/20 Points)

Consider the program `multiply` shown in Figure 2. It is supposed to implement multiplication of two non-negative integers by successive addition as specified by the given pre- and post-conditions. The operands are y and x and the result is stored in the variable r.

$\{y \geq 0 \wedge x \geq 0\}$

```
r = 0;
i = 0;
while (i < x) do
  r = r + y;
  i = i + 1;
od
```

$\{r = y \cdot x\}$

Figure 2: Program `multiply`.

The goal of this exercise is to use the proof system PD to show that `multiply` is partially correct. We approach this goal in three steps:

(i) Give a *loop invariant* for the while loop of which you expect that it will enable you to prove the correctness of the program. Explain how you came up with the loop invariant and argue why you expect it to help with a correctness proof for the program. (3)

Hint: If you do not have a good idea for a loop invariant, do not hesitate to ask your tutor to propose one so that you can continue with the other tasks.

(ii) Apply the rules of the proof system PD to *derive a proof* that your invariant is indeed a loop invariant (in other words: establish the premise of Rule 5).

Specify which rules or axioms you use at every proof step. (3)

- $\mathcal{I}[\![v(F)]\!](\sigma,\beta) = \begin{cases} u' & \text{, if } \mathcal{I}[\![F]\!](\sigma,\beta) \in \text{dom}(\sigma) \text{ and } \sigma(\mathcal{I}[\![F]\!](\sigma,\beta))(v) = \{u'\} \\ \bot & \text{, otherwise} \end{cases}$ (if $v : C_{0,1}$)

I: $r = i \cdot y \wedge i \leq x$

i)

$\{y \geq 0 \wedge x \geq 0\}\ S_0$ while (B) do S_1 od $\{r = y \cdot x\}$ R3

$\{r = i \cdot y \wedge i \leq x \wedge i \geq x\} \rightarrow \{r = x \cdot y\}$

$\{y \geq 0 \wedge x \geq 0\}$

$\{0 = 0 \wedge x \geq 0\}$ A2

$r = 0$

$\{r = 0 \cdot y \wedge 0 \geq 0\}$ A2

$i = 0$

$\{r = i \cdot y \wedge i \leq x\}$

while $i < x$ do

$\{r = i \cdot y \wedge i \leq x \wedge i < x\}$

$\{r + y = y \cdot i + y \wedge i + 1 \leq x\}$ A2

$r = r + y$

$\{r = (i+1) \cdot y \wedge i + 1 \leq x\}$ A2

$i = i + 1$

$\{r = i \cdot y \wedge i \leq x\}$

od

$\{r = i \cdot y \wedge i \leq x \wedge i \geq x\}$

$\{r = x \cdot y\}$ A3

$\{true\}\ y := x; y = ((x-1)\cdot x + y) + z; \textbf{while}\ z\ \textbf{do}\ skip\ \{y = x^2\}$

$\{true\}\ y := x; y = ((x-1)\cdot x + y) + z; \textbf{while}\ z\ \textbf{do}\ skip\ \{y = x^2\}$

$y^2 = x^2$

R3

$\{true\}\ y = x;\ y = ((x-1)\cdot x + y) + z \{r\},\ \{r\}\ \text{while } z \text{ do } skip\ \{y = x^2\}$

R3

$\{true\}\ y = x\ \{r'\},\ \{r'\}\ y = ((x-1)\cdot x + y) + z \{r\}$

R6

$\{true\} \to \{x^2 = (x-1)\cdot x + x + z - z\}$ $y = x$ $\{x^2 = (x-1)\cdot x + y + z - z\}$

genau

$\{x^2 = \{x-1\}\cdot x + y + z - z\}$ $y = ((x-1)\cdot x + y) + z$ $\{x^2 = y - z\}$

$x^2 = x^2$

$\Leftrightarrow x^2 = (x-1)\cdot x + x + z - z$

Rechte Seite vor dem Einsetzen, linke Seite nach dem Einsetzen

OPER: Man kann beliebige Variable(n) von einem Typ in $\Rightarrow$ Richtung ersetzen (alle, 3, 2 oder nur eine)

$\{r\} \to \{p_1\},\ \{p_1\}$ while B do $skip$ $\{q_1\}$, $\{q_1\} \Rightarrow \{y = x^2\}$

R5

R6

$\{\ln B \Rightarrow \{\}\}\ \{B\}\ skip\ \{\}, \{\} \to \{\}$

A1

$\ln z \neq 0 \to 1$

$x^2 = y - z \wedge z = 0$

$\Rightarrow x^2 = y$

$\{X = \{z + z\}\ Y := z;\ \{X = y + z\}$

$\{true\}$

$\{x^2 = x^2 \Leftrightarrow x^2 - x + x = x^2\}$

$y^2 = f^2$

$\{x^2 - x + y = x^2 \Leftrightarrow (x-1)\cdot x + y + z = x^2 - z\}$

$y = (x-1)\cdot x + y + z$

$\{y = x^2 - z\}$

while $d\sigma \supset$

$\{y = x^2 - z \wedge z \neq 0\}$

$\{y = x^2 - z\}$

st:$\cap$

$\{y = x^2 - z\}$

od

$\{y = x^2 - z \wedge z = 0\}$

$\{y = x^2\}$

A2 RG R3 R3

A2

A1 RG

R5

R6

⇒ immer p und q von unten teil der Regel

Exercise 1 – Testing & Coverage Measures (8/20 Points)

```
1  int convert(char[] str) throws Exception {
2      if (str.length > 6)
3          throw new Exception("Length exceeded");
4      int number = 0;
5      int digit;
6      int i = 0;
7      if (str.length > 0 && str[0] == '-')
8          i = 1;
9      while (i < str.length){
10         digit = str[i] - '0';
11         if (digit <= 0 || digit > 9)
12             throw new Exception("Invalid character");
13         number = number * 10 + digit;
14         i = i + 1;
15     }
16     if (str.length > 0 && str[0] == '-')
17         number = -number;
18     if (number > 32767 || number < -32768)
19         throw new Exception("Range exceeded");
20     return number;
21 }
```

Figure 1: Function `convert`.

Consider the Java function `convert` shown in Figure 1. The function is supposed to convert the (up to 6-digit) string representation (decimal, base 10) of a 16-bit number to the represented integer. The following exceptional cases should be considered, i.e. corresponding exceptions should be thrown:

- The input string `str` has strictly more than 6 characters.
- The input string has at most 6 characters, and one of them is not a digit, i.e. from {'0', ..., '9'} (the first character may in addition be a minus ('-')).
- The input string has at most 6 characters, all of them digits (possibly a '-' in front) and the denoted integer value is outside the range $[-32768, 32767]$.

If none of the exceptional cases applies, let $c_0, \ldots, c_{n-1}$, $n \geq 0$, be the first, second, ..., n-th character in `str` (from left to right). The expected return value is $\sum_{i=0}^{n-1} c_i \cdot 10^{n-i-1}$ if the first character c_0 is not '-' and $-\sum_{i=1}^{n-1} c_i \cdot 10^{n-i-1}$ if the first character c_0 is '-'. (Note that, unlike other implementations of string-to-integer conversions, this one should return 0 for the empty string and the string "-".)

(i) Give an *unsuccessful* test suite for `convert` that achieves 100% *statement coverage* and 100% *branch coverage*. (6)

Hint: Make sure that your presentation easily and strongly convinces your tutor about your claims on the test cases and the coverage measures.

(ii) Modify your test suite from Task (i) such that the test suite is still *unsuccessful* and still achieves 100% *statement coverage*, but now achieves *strictly less* than 100% *branch coverage*. (1)

(iii) Is `convert` correct wrt. the (functional) specification detailed above? Argue your claim. (1)

Exercise 1 – Evaluating OCL Formulae (5/

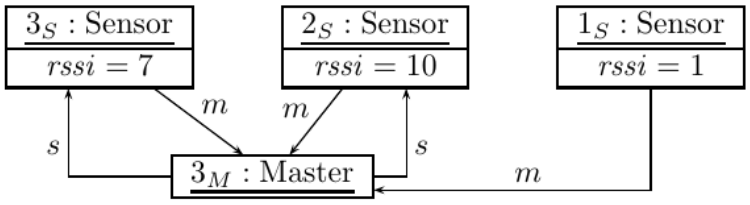


Figure 1: Complete object diagram.

Consider the system state σ_1 given by the complete Object Diagram in Figure 1.

(i) To which value does the Proto-OCL formula

$$F := \forall\, self \in allInstances_{\text{Master}} \bullet \forall\, n \in s(self) \bullet rssi(n) > 1$$

$I[\![F]\!](\sigma_1, \emptyset)$

$= I[\![allInstances_M]\!](\sigma_1, \emptyset) = \{3M\}$

$\beta_1 = \{self \mapsto 3M\}$

Fall 3M:

$I[\![\forall n \in s(self) : rssi(n) > 1]\!](\sigma_1, \beta_1)$

$= I[\![\forall n \in s(self)]\!](\sigma_1, \beta_1)$

$= \delta(I[\![self]\!](\sigma_1, \beta_1))(S)$

$= \delta(3M)$ $\beta(self) = 3M$

$= \{3S_1, 2S\}$

$\beta_1 = \{self \mapsto 3M, a \mapsto 3S\}, \beta_2 = \{a \mapsto 2S\}$

Fall 3S:

$I[\![>(rssi(n), 1)]\!](\sigma_1, \beta_1)$

$= >(I(rssi(n))(\sigma_1, \beta_1), I[\![1]\!](\sigma_1, \beta_1))$

$= >(\delta(I[\![n]\!](\sigma_1, \beta_1))(rssi), 1^I)$ (Def. 3, Def. 5)

$= I[\![n]\!](\sigma_1, \beta_1) = 3S$

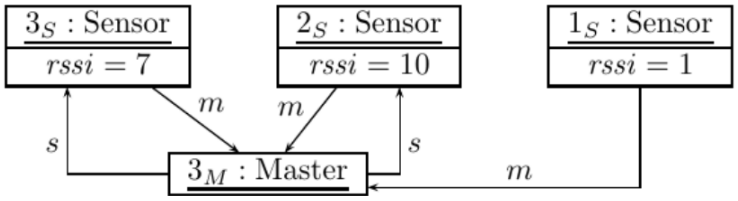
$= >(\delta(3S)(rssi), 1^I)$

$= >(7^I, 1^I) = True$

(ii) Provide system states σ_2 and σ_3 such that for each truth value (*true*, *false*, $\perp$), there is an $i \in \{1, 2, 3\}$ such that $\mathcal{I}[\![F]\!](\sigma_i, \emptyset)$ evaluates to this truth value. Argue your claim. (2)

Hint: As we refer to the same OCL formula as in Task (i), you may use the detailed proof that you provided for Task (i) to guide your argument here.

$$F := \forall\, self \in allInstances_{\text{Master}} \bullet \forall\, n \in s(self) \bullet rssi(n) > 1$$



Exercise 2 – Computation Graph / Transition Graph (6/20 Points)

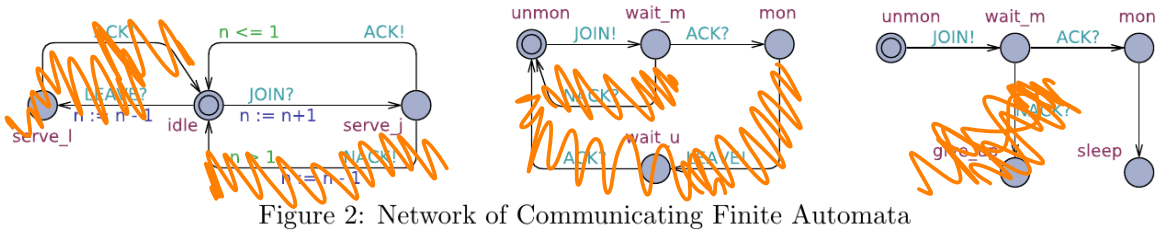
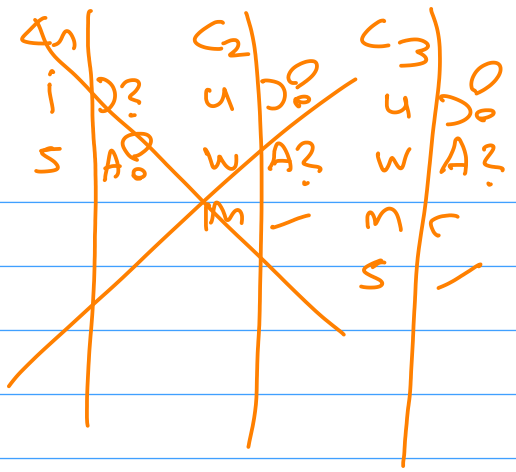


Figure 2: Network of Communicating Finite Automata

Provide the reachable part of the transition graph of the CFA model shown in Figure 2. (6)

Hint: Make sure to clearly indicate the initial configuration(s). And you may want to introduce abbreviations for location names if this increases readability of your computation graph.

D:	i u *	s w * n+1
	i * u	s * w n+1
A:	s w * n≤1	i m *
	s * w n≤1	i * m
r:	* * m	* * s



<(i, u, w), n=0>

<(s, w, u), n=1>

A

<(i, m, u), n=1>

D

<(s, m, w), n=2>

<(s, u, w), n=1>

A

<(i, u, m), n=1>

r

<(i, w, s), n=1>

n4:30

D

<(s, w, m), n=2>

r

<(s, w, s), n=2>

7 Communicating Finite Automata

Communicating Finite Automata

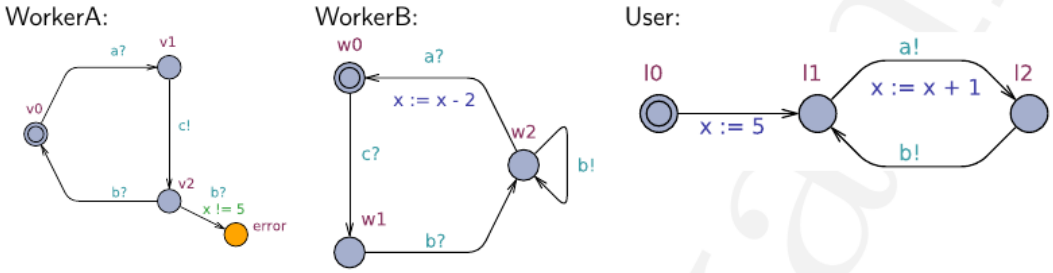


Figure 8: Network of Communicating Finite Automata (x is a global variable).
Abbildung 8: Netzwerk Kommunizierender Endlicher Automaten (x ist eine globale Variable).

Problem 7.1 (Computation Paths)

Aufgabe 7.1 (Berechnungspfade)

Consider the network $\mathcal{N}$ of communicating finite automata given in Figure 8.

(i) It is claimed that $\mathcal{N} \models$ `E<>` WorkerA.error holds.
Prove this claim by providing an appropriate computation path.
Es wird behauptet, daß $\mathcal{N} \models$ `E<>` WorkerA.error gilt.
Beweisen Sie diese Behauptung, indem Sie einen entsprechenden Berechnungspfad angeben.

Correction:

- 1 point for each correct configuration and transition (in shortest path)

(ii) It is claimed that $\mathcal{N} \models$ `A[]` WorkerA.error $\vee$ `not deadlock` does not hold.
Prove this claim by providing an appropriate computation path.
Es wird behauptet, daß $\mathcal{N} \models$ `A[]` WorkerA.error $\vee$ `not deadlock` nicht gilt.
Beweisen Sie diese Behauptung, indem Sie einen entsprechenden Berechnungspfad angeben.

Correction:

- $\frac{3}{13}$ points for each correct configuration and transition (in shortest path)

(iii) It is claimed that $\mathcal{N} \models$ `E[] not` WorkerA.error holds for an infinite computation path.
Prove this claim by providing an appropriate computation path.
Es wird behauptet, daß $\mathcal{N} \models$ `E[] not` WorkerA.error für einen unendlichen Berechnungspfad gilt.
Beweisen Sie diese Behauptung, indem Sie einen entsprechenden Berechnungspfad angeben.

Correction:

- $\frac{2}{13}$ points for each correct configuration and transition (in shortest path)
- 1 point for indicating the loop

(i)

a: $v_0 * l_1$ | $v_1 * l_2$ $x+1$

$* w_2 l_1$ | $* w_0 l_2$ $x+1-2$

b: $v_2 * l_2$ | $v_p * l_1$

$* w_1 l_2$ | $* w_2 l_1$

$v_2 * l_2$ $x \neq 5$ | $e * l_2$

$v_2 w_2 *$ | $v_0 w_2 *$

$v_2 w_2 *$ $x \neq 5$ | $e w_2 *$

c: $v_1 w_0 *$ | $v_2 w_1 *$

c: $* * l_0$ | $* * l_1$ $x=5$

(j)

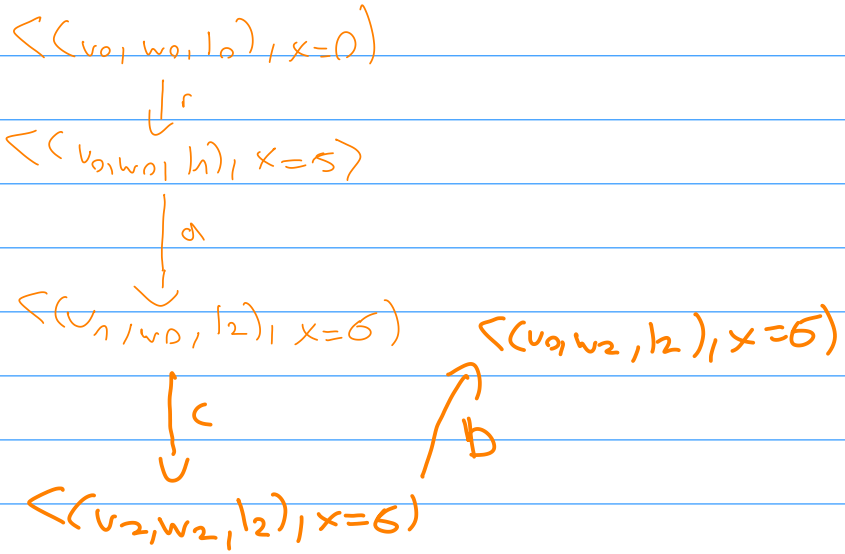
$N \vdash E \langle\rangle$ writeA,error

$$\langle (v_0, w_0, l_0), x=0 \rangle$$
$$\downarrow r$$
$$\langle (v_0, w_0, l_1), x=5 \rangle$$
$$\downarrow a$$
$$\langle (v_1, w_0, l_2), x=6 \rangle$$
$$\downarrow c$$
$$\langle (v_2, w_1, l_2), x=6 \rangle$$
$$\downarrow$$
$$\langle (e, w_2, l_2), x=6 \rangle$$

(ii) $N \not\models A[]$ worker.error ✓ not deadlock

$N \not\models E<>$ not worker.error ∧ deadlock

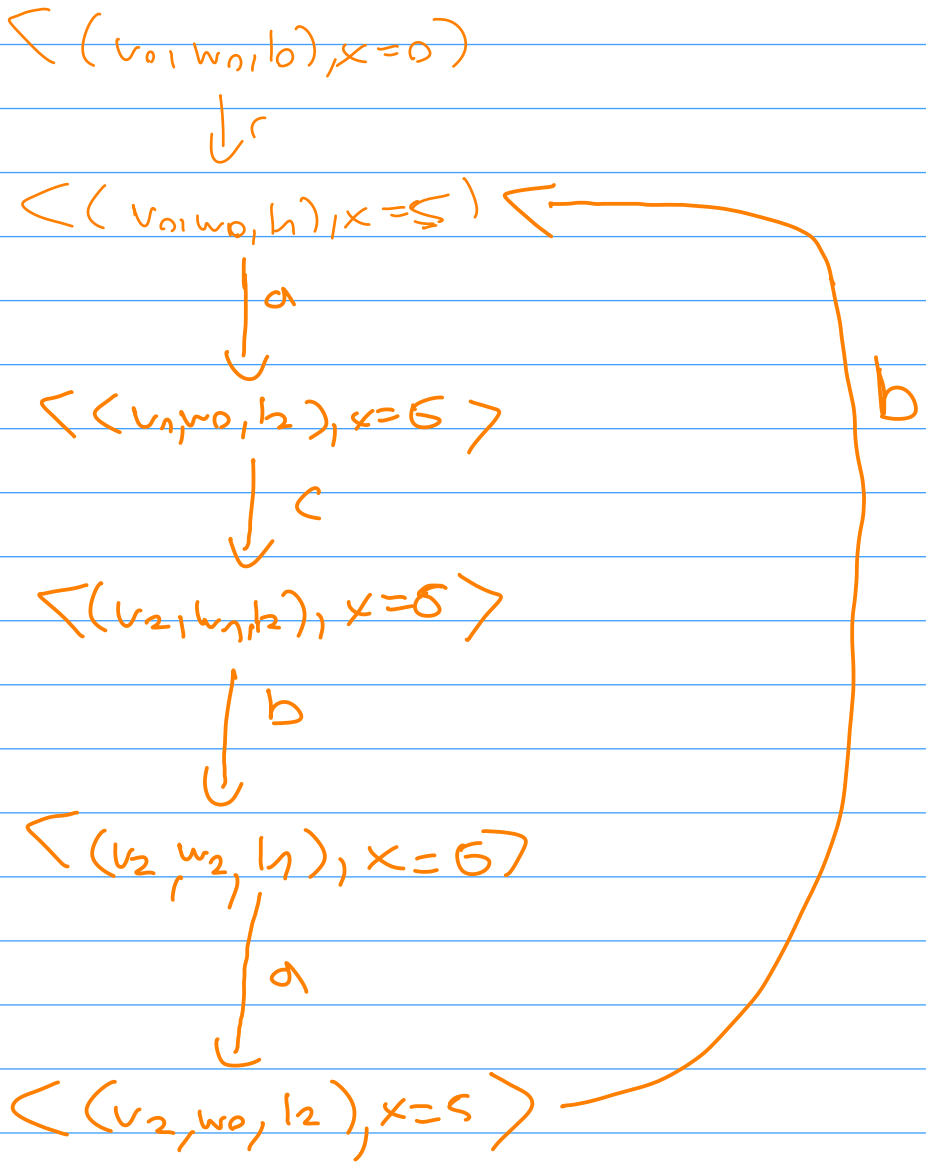
suche Transition die einen in einen Zustand (post condition) bringt, der bei bestimmten * keine Transition hat (Precondition)



(iii) It is claimed that $\mathcal{N} \models$ E[] not WorkerA.error holds for an infinite computation path. Prove this claim by providing an appropriate computation path.

Es wird behauptet, daß $\mathcal{N} \models$ E[] not WorkerA.error für einen unendlichen Berechnungspfad gilt. Beweisen Sie diese Behauptung, indem Sie einen entsprechenden Berechnungspfad angeben.

$N \models EC$ not worker error



15:15

6 OCL

OCL

$$\sigma = \{u_1 \mapsto \{length \mapsto 27, n \mapsto u_2\}, u_2 \mapsto \{length \mapsto 13, n \mapsto \emptyset\}\}$$

Figure 6: System state.
Abbildung 6: Systemzustand.

Problem 6.1 (Objectdiagrams)

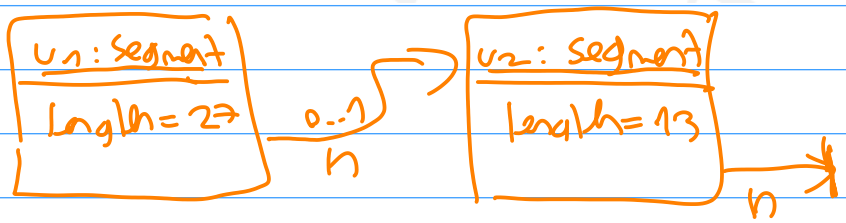
Aufgabe 6.1 (Objektdiagramme)

Provide an object diagram which completely represents the system state shown in Figure 6.
Geben Sie ein Objektdiagramm an, das den in Abbildung 6 gegebenen Systemzustand vollständig darstellt.

(3)

Correction:

- 0.5 points for each correctly represented piece of information
- max. -1 points for syntax mistakes



Problem 6.2 (Proto-OCL Proof)

Aufgabe 6.2 (Proto-OCL Beweis)

$$F := \forall\, self : Segment \bullet length(self) > length(n(self))$$

Figure 7: Proto-OCL formula.
Abbildung 7: Proto-OCL Formel.

It is claimed that the Proto-OCL formula F shown in Figure 7 evaluates to $\perp$ in the system state from Figure 6, i.e. that $\mathcal{I}[\![F]\!](\sigma, \emptyset) = \perp$.
Prove this claim using the semantics of Proto-OCL.
Es wird behauptet, daß die in Abbildung 7 gegebene Proto-OCL Formel F in dem durch das Objektdiagramm in Abbildung 6 beschriebenen System State zu $\perp$ ausgewertet wird, d.h. daß $\mathcal{I}[\![F]\!](\sigma, \emptyset) = \perp$.
Beweisen Sie diese Behauptung unter Verwendung der Proto-OCL Semantik.

(5)

I(F)(σ, ∅)

= I(ALL Segment)(σ, ∅) = {u1, u2}

= β1 = {self ↦ u1}, β2 = {self ↦ u2}

Fall u1:

I(∀(length(self), length(n(self)))(σ, β1)

= ... I(length(self), length(n(self)))(σ, β1)

= (σ(u1)(length), σ(u2)(length))

= (27, 13) = true

Fall u2:

$I[\![G(\text{length}(\text{self}), \text{length}(n(\text{self})))]\!](\sigma_1, \beta_2)$

$= > (13, I[\![\text{length}(n(\text{self}))]\!](\sigma_1, \beta_2)) = \bot$, da

$= > (13, \bot) = \bot$, da $I[\![n(\text{self})]\!](\sigma_1, \beta_2) = \bot \notin \text{dom}(\sigma_1)$,

da $\sigma(I[\![\text{self}]\!](\sigma_1, \beta_2))(n) = \emptyset \neq \{u_1\}$

Fallunterscheidung

$\Rightarrow$ Insg: $\bot$

Problem 6.3 (Proto-OCL Examples)

Aufgabe 6.3 (Proto-OCL Beispiele)

Provide two system states σ_1 and σ_2 such that $\mathcal{I}[\![F]\!](\sigma_1, \emptyset) = \mathit{true}$ and $\mathcal{I}[\![F]\!](\sigma_2, \emptyset) = \mathit{false}$.

Geben Sie zwei System States σ_1 und σ_2 an so, daß $\mathcal{I}[\![F]\!](\sigma_1, \emptyset) = \mathit{true}$ und $\mathcal{I}[\![F]\!](\sigma_2, \emptyset) = \mathit{false}$ gilt.

(4)

Correction:

- σ_1: (1 point, if correct and notation right)
- σ_2: (3 points, if correct and notation right)
- can be given as object diagram (say: complete, otherwise possibly fewer points), or in system state notation

$\sigma_1 = \{\}$ oder $\emptyset$?

~~$\sigma_1 = \{u_1 \mapsto \{\text{length} \mapsto 1, n \mapsto \{u_2\}\}$~~
~~$u_2 \mapsto \{\text{length} \mapsto 2, n \mapsto \{u_3\}\}$~~
~~$u_3 \mapsto \{\text{length} \mapsto 3, n \mapsto \{u_2\}\}$~~

$\sigma_2 = \{u_1 \mapsto \{\text{length} \mapsto 1, n \mapsto \{u_2\}\}$
$u_2 \mapsto \{\text{length} \mapsto 2, n \mapsto \{u_1\}\}$

Problem 6.4 (Proto-OCL Formalisation)

Aufgabe 6.4 (Proto-OCL Formalisierung)

Formalise the following requirements using Proto-OCL formulae:

(i) The value of the basic type attribute of each object of class *Segment* should be strictly bigger than 10.

(ii) The value of the *length* attribute of two different *Segment* instances to which a *Train* instance refers by an *s*-link differs by at most 1.

Formalisieren Sie die folgenden Anforderungen mit Hilfe von Proto-OCL Formeln:

(i) Der Wert des Basistyp-Attributs jedes Objektes der Klasse *Segment* soll streng größer als 10 sein.

(ii) Die Werte des *length* Attributs zweier verschiedener *Segment*-Instanzen, auf die eine *Train*-Instanz mit einem *s*-Link verweist, unterscheiden sich um höchstens 1.

(3)

Correction:

- 0.5 points for each element marked in the reference solution
- 0.125 points for each element marked in the reference solution

basic math

(i) $\forall i \in \text{AllInst}_{\text{Segment}} \bullet \text{length}(i) > 10$

(ii) $\forall i \in \text{AllInst}_{\text{Train}} \bullet \forall j \in s(i) \bullet \forall k \in s(i) \bullet |\text{length}(j) - \text{length}(k)| \leq 1$

15:00

4 Live Sequence Charts

Live Sequence Charts

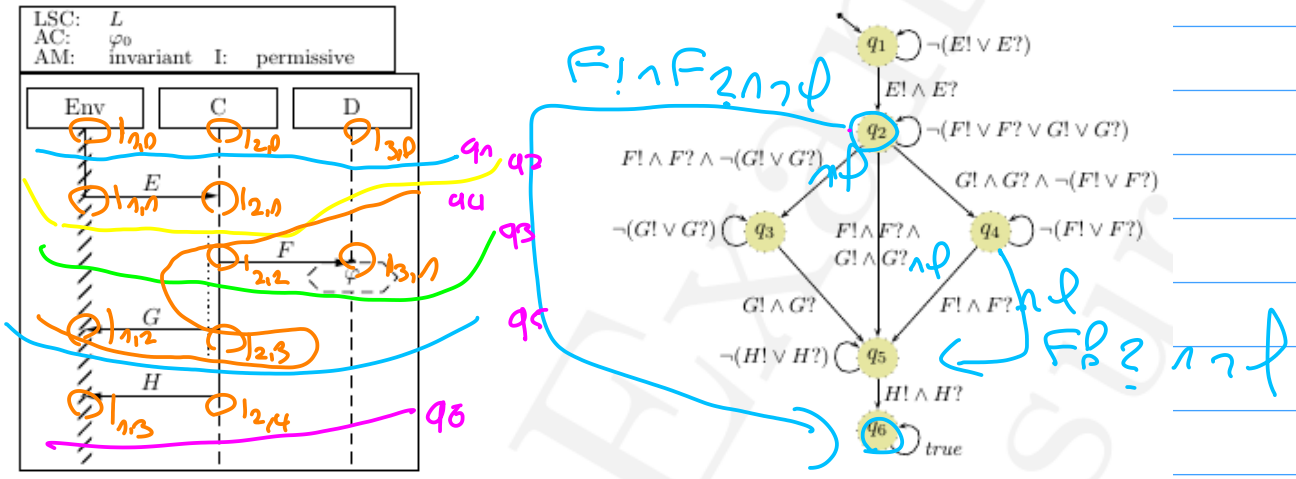


Figure 4: Live Sequence Chart and its TBA.
Live Sequence Chart und zugehöriger TBA.

Problem 4.1 (Locations and Cut Temperatures)

Aufgabe 4.1 (Locations und Cut Temperaturen)

(i) What are the locations of the LSC L given in Figure 4 on page 6.
Clearly mark them in the LSC in Figure 4.

Geben Sie die Locations der in Abbildung 4 auf Seite 6 gegebenen LSC an.
Markieren Sie die Locations eindeutig in der LSC in Abbildung 4.

(3)

Correction:

- $\frac{3}{11}$ points per correct location

(ii) According to the LSC semantics, which states of the TBA shown in Figure 4 need to be accepting and which not?
Mark the states in the TBA in Figure 4 with single or double outlines accordingly.

Welche Zustände des in Abbildung 4 gezeigten TBA sind entsprechend der LSC Semantik akzeptierend und welche nicht?
Markieren Sie die Zustände im TBA in Abbildung 4 entsprechend mit einfachem oder doppeltem Rand.

(2)

Correction:

- $\frac{1}{3}$ point for each correct mark

Problem 4.2 (Computation Paths)

Aufgabe 4.2 (Berechnungspfade)

(i) Provide a computation path which trivially satisfies L.

(ii) Provide a computation path which non-trivially satisfies L.

(iii) Provide a computation path which violates L.

You may disregard the condition for this task.

(i) Geben Sie einen Berechnungspfad an, der L trivial erfüllt.

(ii) Geben Sie einen Berechnungspfad an, der L nicht-trivial erfüllt.

(iii) Geben Sie einen Berechnungspfad an, der L verletzt.

Die Bedingung können Sie für diese Aufgabe ignorieren.

(i) $\neg C_0$

$\{C_0 \to \text{Fail}\} \hookrightarrow \{C_0 \to \text{Fail}\} \hookrightarrow \cdots$

(ii)

$\{C_0\} \hookrightarrow \{\cdot\} \xrightarrow{F!?} \{\varphi\}$

$\xrightarrow{F!?\&G!?} \{\cdot\} \xrightarrow{H!?} \{\cdot\} \hookrightarrow \cdots$

(iii)

$\{C_0\} \hookrightarrow \{\cdot\} \xrightarrow{F!?} \{\cdot\}$

Problem 4.3 (Cold Conditions)

Aufgabe 4.3 (Kalte Bedingungen)

(i) The condition φ in LSC L has not been considered in the TBA shown in Figure 4. Complete the TBA such that the condition is properly represented.

Die Bedingung φ in LSC L wurde bei der Konstruktion des in Abbildung 4 gezeigten TBA nicht berücksichtigt. Vervollständigen Sie den TBA derart, daß die Bedingung korrekt repräsentiert wird.

(1)

Correction:

- $\frac{1}{7}$ point for each correct new transition and label (modification)

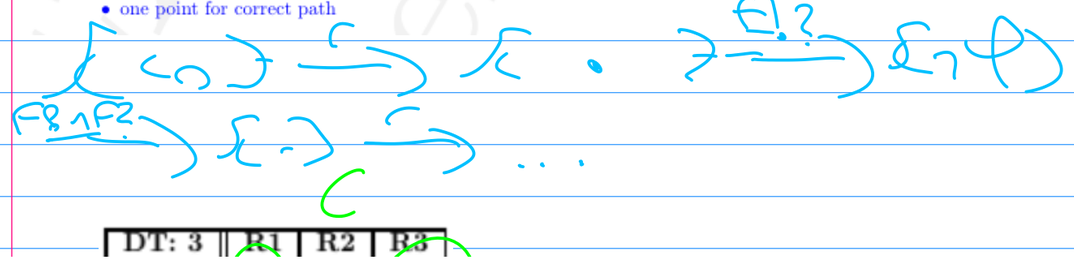
(ii) Provide a computation path which satisfies L by taking a legal exit.

Geben Sie einen Berechnungspfad an, der L erfüllt, indem ein legaler Ausstieg genommen wird.

(1)

Correction:

- one point for correct path



DT: 3 ‖ R1 | R2 | R3

Exercise 1 – LSC Syntax (6/20 Points)

Consider the Live Sequence Chart in Figure 1.

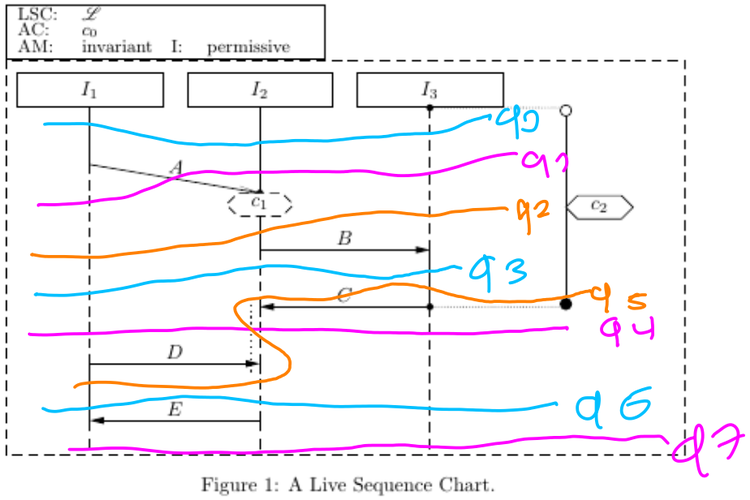


Figure 1: A Live Sequence Chart.

Consider the **abstract syntax** of the chart in Figure 1.

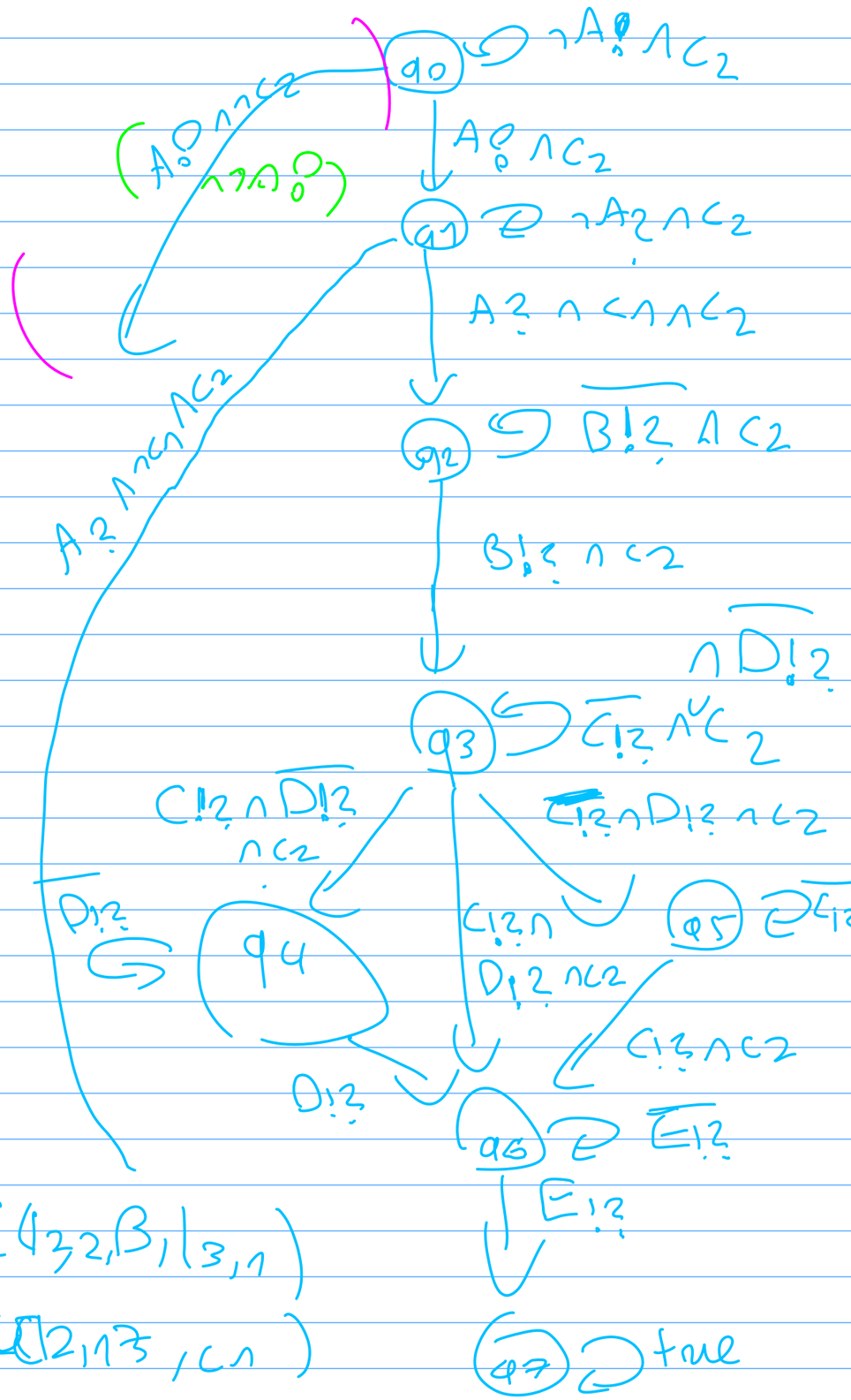
(i) Provide the set of locations $\mathcal{L}$ (including their temperature). Make sure to clearly indicate where in the chart which element of $\mathcal{L}$ occurs. (2)

Hint: Note that all locations in a coregion obtain the same temperature from the relevant instance line segment adjacent to the whole coregion.

(ii) Give the partial order relation $\preceq$ for the locations on instance lines I_1 and I_2. (2)

Hint: Only direct predecessors/successors need to be given, the full relation is then the transitive, reflexive closure of those.

(iii) Give one example element each from the simultaneity relation $\sim$, the set of messages Msg, the set of local invariants LocInv, and the set of conditions Cond (the latter three including their temperature). (2)



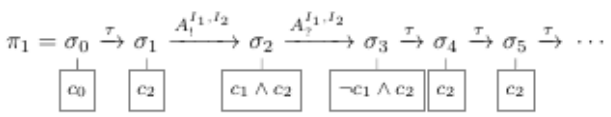
Exercise 2 – LSC Semantics (7/20 Points)

(i) Construct the Büchi automaton for the body of the chart in Figure 1.

Show the steps of your construction by writing down the cuts that need to be considered (incl. their temperature), their direct successor relation via fired-sets, and point out which state in the automaton each cut corresponds to. For the three smallest cuts, note down all locations that the cut comprises, and for all other cuts only note down their front locations. (4)

Hint: For instantaneous messages F from J_1 to J_2, you may use the shorthand notations $F_{!?}^{J_1,J_2}$ and $\overline{F_{!?}^{J_1,J_2}}$ for $F_{!}^{J_1,J_2} \wedge F_{?}^{J_1,J_2}$ and $\neg(F_{!}^{J_1,J_2} \vee F_{?}^{J_1,J_2})$, respectively. Note that $\overline{F_{!?}^{J_1,J_2}}$ is not equivalent to $\neg F_{!?}^{J_1,J_2}$.

(ii) Consider the following computation path π_1:[1]



a) Does π_1 satisfy LSC $\mathscr{L}$ or does it violate the chart? (1)
Hint: Argue your result on the automaton from the previous task.

b) For each of the following three cases, provide a computation path that is an example for the case. Use the example to explain the concept covered by the case in your own words (for example, for the case in the middle, explain the concept of satisfaction without legal exit and how this concept is visible in your example):

- Path violates chart.
- Path satisfies chart by taking a legal exit.
- Path (activates and) satisfies chart without taking a legal exit.

You may re-use path π_1 from the previous task if it matches one of the three cases. (2)

$\{n \geq 0\}$

$\{n \geq 0 \land 0 = n \cdot \frac{1}{2}((0-1)\cdot 0) - \frac{1}{6}\cdot(0-1)\cdot 0\cdot(2\cdot 0-1) \land 0 \leq n+1 \land 0 = 0\}$

(A2)

$r := 0;$

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((0-1)\cdot 0) - \frac{1}{6}\cdot(0-1)\cdot 0\cdot(2\cdot 0-1) \land 0 \leq n+1 \land 0 = 0\}$

$i := 0;$

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land i = 0\}$

if $n > 0$ **then**

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land i = 0 \land n > 0\}$

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land n > 0\}$

while $i \leq n$ **do**

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land i \leq n\}$

$\{n \geq 0\}$

$\{n \geq 0 \land 0 = n \cdot \frac{1}{2}((0-1)\cdot 0) - \frac{1}{6}\cdot(0-1)\cdot 0\cdot(2\cdot 0-1) \land 0 \leq n+1 \land 0 = 0\}$

(A2)

$r := 0;$

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((0-1)\cdot 0) - \frac{1}{6}\cdot(0-1)\cdot 0\cdot(2\cdot 0-1) \land 0 \leq n+1 \land 0 = 0\}$

$i := 0;$

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land i = 0\}$

if $n > 0$ **then**

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land i = 0 \land n > 0\}$

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land n > 0\}$

while $i \leq n$ **do**

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land i \leq n\}$

$\{n \geq 0 \land r + i\cdot(n-i) = n \cdot \frac{1}{2}(i\cdot(i+1)) - \frac{1}{6}\cdot i\cdot(i+1)\cdot(2\cdot i+1) \land (i+1) \leq n+1\}$

$\{n \geq 0 \land r + i\cdot(n-i) = n \cdot \frac{1}{2}(((i+1)-1)\cdot(i+1)) - \frac{1}{6}\cdot((i+1)-1)\cdot(i+1)\cdot(2\cdot(i+1)-1) \land (i+1) \leq n+1\}$

R6 R5

$r := r + i\cdot(n-i);$

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i+1)-1)\cdot((i+1)-1)\cdot(i+1)) - \frac{1}{6}\cdot((i+1)-1)\cdot(i+1)\cdot(2\cdot(i+1)-1) \land (i+1) \leq n+1\}$ R3

$i := i + 1;$

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1\}$

R4

do

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land \neg(i \leq n)\}$

$\{n = i+1 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1)\}$

$\{r = n \cdot \frac{1}{2}(n\cdot(n+1)) - \frac{1}{6}\cdot n\cdot(n+1)\cdot(2\cdot n+1)\}$

else

$\{n \geq 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1) \land i \leq n+1 \land i = 0 \land n \leq 0\}$

$\{n = 0 \land i = 0 \land r = n \cdot \frac{1}{2}((i-1)\cdot i) - \frac{1}{6}\cdot(i-1)\cdot i\cdot(2\cdot i-1)\}$

$\{n = 0 \land r = n \cdot \frac{1}{2}((n+1)\cdot n) - \frac{1}{6}\cdot n\cdot(n+1)\cdot(2\cdot n+1)\}$

skip A1

$\{n = 0 \land r = n \cdot \frac{1}{2}((n+1)\cdot n) - \frac{1}{2}\cdot n\cdot(n+1)\cdot(4\cdot n+1)\}$

$\{r = n \cdot \frac{1}{2}((n+1)\cdot n) - \frac{1}{6}\cdot n\cdot(n+1)\cdot(2\cdot n+1)\}$

fi

$\{r = n \cdot \frac{1}{2}((n+1)\cdot n) - \frac{1}{6}\cdot n\cdot(n+1)\cdot(2\cdot n+1)\}$

$\{r = \sum_{j=0}^{n} j\cdot(n-j)\}$ (*)

Figure 10: Incomplete proof sketch for function SOMESUM.
Abbildung 10: Unvollständige Beweisskizze für Funktion SOMESUM.

Problem 8.2 (Prove with PD)

Aufgabe 8.2 (Beweisen mit PD)

$$\{true\}\ y := x; y = ((x-1)\cdot x + y) + z; \textbf{while } z \textbf{ do } skip\ \{y = x^2\}$$

Figure 9: A simple program.
Abbildung 9: Ein einfaches Programm.

Prove that the Hoare triple given in Figure 9 holds for partial correctness using the PD calculus.

Beweisen Sie mit Hilfe des PD-Kalküls, daß das in Abbildung 9 angegebene Hoare-Triple im Sinne von partieller Korrektheit gilt.

(3)

17:30

5 Class Diagrams

Klassendiagramme

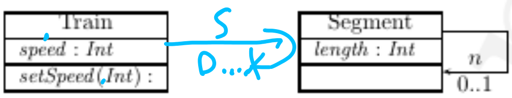


Figure 5: Class diagram.
Abbildung 5: Klassendiagramm.

S : Segment*

Problem 5.1 (Signature)

Aufgabe 5.1 (Signatur)

Provide the (minimal) signature described by the class diagram in Figure 5.

Geben Sie die (kleinste) Signatur an, die vom Klassendiagramm in Abbildung 5 beschrieben wird.

(4)

Correction:

- 0.25 points for each correctly rewritten element
- max. -1 for notation errors

{ {Int}

{Train, S}

{(speed : Int, length : Int)}

{T ↦ {speed}, S ↦ {length}}

{setSpeed : Int → ∅}

{Tr ↦ setM, Se → ∅}

Problem 5.2 (Associations)

Aufgabe 5.2 (Assoziationen)

Assume we add $s : Segment_*$ to V and s to $atr(Train)$ to the signature of the class diagram shown in Figure 5. Extend the class diagram in Figure 5 such that this additional modelling element is properly *graphically* represented, i.e. do not just add s to the attributes compartment.

Nehmen Sie an, in der Signatur des in Abbildung 5 gezeigten Klassendiagramms wird $s : Segment_*$ zu V und s zu $atr(Train)$ hinzugeügt.
Erweitern Sie das Klassendiagramm in Abbildung 5 derart, daß dieses zusätzliche Modellelement korrekt *graphisch* repräsentiert wird, d.h. s soll nicht einfach dem Bereich für Attribute hinzugefügt werden.

(1)

Correction:

- 0.25 points each for: line between correct things, arrow at correct side, labels at correct sides

18:00

Exercise 3 – Use Cases and Use Case Diagrams (0/20 Points + 2 Bonus)

Assume that the table in Figure 2 is provided together with LSC $\mathscr{L}$ as a proper use case. What would be the use case diagram for this use case? Explain. (2 Bonus)

Exercise 4 – Class and Object Diagrams (7/20 Points)

Recall the Wireless Fire Alarm System (WFAS) from the lectures on requirements. A cornerstone of the design is the idea that for each sensor,[2] there is a so-called master to monitor the sensor and take notifications (like indications of smoke or high temperatures). If the system is operational (not in maintenance mode), each sensor must have a link to exactly one master, and the sensor-master links must be reciprocal, i.e. the master's link includes all sensors who have a link to this master.
Figure 3 shows an abstract model of this design idea.

(i) Provide the **abstract syntax** of the diagram. (2)

[1] The notation with boxes below states (or configurations) indicates which atomic propositions we assume to be satisfied in the labelled state. For example, σ_0 is any state in which c_0 is satisfied (while we do not care about the satisfaction of c_1 and c_2). State σ_1 is any state in which condition c_1 is satisfied and c_2 not (and about c_0 we do not care).

[2] which keeps track of a received signal strength indicator (RSSI)

name	handle situation c_0
goal	get system back into safe mode
precondition	system in situation c_0
postcondition	safe mode reached (indicated by message D), optional: user confirmation E acknowledged with F
actors	user (main actor, instance line I_1 in $\mathscr{L}$)
open questions	none
normal case	see LSC $\mathscr{L}$ (without legal exit)
exception case	see LSC $\mathscr{L}$ (legal exit) see LSC $\mathscr{L}$ (cold cut)

Figure 2: Example Use Case.

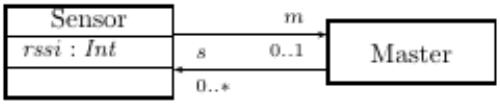


Figure 3: WFAS masters and slaves.

(ii) Clarify the intended usage of the data-structure as described in the introductory text to this exercise; use at least three *own* (i.e., not from the lecture, not from an exercise) non-trivial system states (over the Class Diagram from Figure 3 and structure $\mathscr{D}$ with $\mathscr{D}(Int) = \mathbb{Z}$ and $\mathscr{D}(\mathscr{C})$ of your choice) shown as Object Diagrams.

- One system state that models a typical configuration in operational mode. (2)
- One system state that is not allowed in operational mode. (1)
- One system state that models a corner-case configuration, i.e., which may be (dis)allowed by the wording above but which seems questionable from the WFAS context. (1)

Hint: Note that the task asks for 'clarification', i.e., a set of system states alone is not a solution to this task.

(iii) For one of your Object Diagrams from Task (ii), which includes at least one Sensor-object, provide the underlying system state in function notation. (1)

Exercise 2 – Analysis of Decision Tables (7/20 Points)

Consider the decision tables shown in Figure 1.

DT: 1	R1	R2	R3
C1	×	–	*
C2	–	*	×
C3	*	–	*
A1	×	–	–
A2	–	×	–
$\neg[\neg C1 \wedge \neg C2]$			

(a) Decision Table 1

DT: 2	R1	R2	R3	R4
C1	*	*	–	×
C2	×	–	–	–
C3	*	×	–	–
A1	×	–	–	×
A2	×	–	×	–

(b) Decision Table 2

DT: 3	R1	R2	R3
C1	×	–	*
C2	×	*	×
C3	*	*	*
A1	×	–	–
A2	–	–	×
$\neg[C1 \wedge C2]$			

Conflicting actions: A1 $\not\perp$ A2

(c) Decision Table 3

Figure 1: Decision Tables for Exercise 1

(i) Are decision tables "DT1" and "DT2" *complete*? Is "DT1" relative complete? (3)

(ii) Are decision tables "DT1" and "DT2" *deterministic*? (2)

(iii) Do decision tables "DT2" and "DT3" have *useless rules*? (1)

(iv) Is decision table "DT3" *consistent* with respect to *conflicting actions* in the collecting semantics? (1)

Justify your answers with proofs or counterexamples.

Hint: One possible method for showing properties of a decision table is by creating a truth table *with all possible combinations of conditions and determining which rules and actions are applied to each combination. Another possible method is by deriving the formulas represented by the table and using the rules of propositional logic to show their validity, satisfiability, etc.*

(i)

DT: 1	R1	R2	R3
C1	×	–	*
C2	–	*	×
C3	*	–	*
A1	×	–	–
A2	–	×	–
$\neg[\neg C1 \wedge \neg C2]$			

(a) Decision Table 1

DT: 2	R1	R2	R3	R4
C1	*	*	–	×
C2	×	–	–	–
C3	*	×	–	–
A1	×	–	–	×
A2	×	–	×	–

(b) Decision Table 2

C1 C2 C3 übrig

1 0

0 0 1 0

C1 C2 3 übrig

1 1

1 0 1 1

Nein FSC

C1↦1 C2↦1 0 0 1 1

C3↦T

2 Spalten

1 0 * 0 * 1 * 1 *

* 1 * * 0 1 0 0 0 1 0 0

C1	C2	C3	R1	R2	R3	V	R1	R2	R3	R4	V
0	0	0		1	0	1			1		
1	0	0			1	1		1			1
1	0	1			1	1	1				1
1	0	1			1	1	1			1	1
1	1	0	1	1		1		1			1
1	1	1	1		1	1					1

DT: 1	R1	R2	R3
C1	×	–	*
C2	–	*	×
C3	*	–	*
A1	×	–	–
A2	–	×	–
$\neg[\neg C1 \wedge \neg C2]$			

(a) Decision Table 1

DT: 2	R1	R2	R3	R4
C1	*	*	–	×
C2	×	–	–	–
C3	*	×	–	–
A1	×	–	–	×
A2	×	–	×	–

(b) Decision Table 2

CA1	CA2	CA3
X	X	–
X	–	X
*	*	*

C1 C2 C3 Dorf

1 1 2

1 0 2

0 0 1 0

Nein für C1 ↦ False
C2 ↦ False
C3 ↦ True

c)ii)

DT: 1	R1	R2	R3
C1	×	–	*
C2	–	*	×
C3	*	–	*
A1	×	–	–
A2	–	×	–
$\neg[\neg C1 \wedge \neg C2]$			

(a) Decision Table 1

DT: 2	R1	R2	R3	R4
C1	*	*	–	×
C2	×	–	–	–
C3	*	×	–	–
A1	×	–	–	×
A2	×	–	×	–

(b) Decision Table 2

C1	C2	C3	R1 (1 0 *)	R2 (1 * 1)	R3 (* 1 *)	wrongly = not true
0	0	0		0	0	1
0	0	1				1
0	1	0			1	1
0	1	1			1	1
1	0	0	1			1
1	0	1	1	1		0
1	1	0			1	1
1	1	1		1	1	0

→ Nein
right
det-

(iii) Do decision tables "DT2" and "DT3" have *useless rules*?

(iii)

DT: 2	R1	R2	R3	R4
C1	*	*	–	×
C2	×	–	–	–
C3	*	×	–	–
A1	×	–	–	×
A2	×	–	×	–

(b) Decision Table 2

DT: 3	R1	R2	R3
C1	×	–	*
C2	×	*	×
C3	*	*	*
A1	×	–	–
A2	–	–	×
	$\neg[C1 \wedge C2]$		

Conflicting actions: A1, A2

(c) Decision Table 3

DT: 3	R1	R2	R3
C1	×	–	*
C2	×	*	×
C3	*	*	*
A1	×	–	–
A2	×	–	×
¬[C1 ∧ C2]			

Conflicting actions: A1 ≠ A2

(c) Decision Table 3

(iv) Is decision table "DT3" *consistent* with respect to *conflicting* mantics?

$\{(R_1, R_3)\} \subset$ Conflict with η → nicht konsistent

$\Leftrightarrow$

$\{(R_i, R_j, \ldots) \mid R_i \vee R_j \vee \ldots\}$ Konflikt mit η

c_1	c_2	c_3	$F_{pre}(R_1)$	$F_{pre}(R_2)$	$F_{pre}(R_3)$	$F_{pre}(R_1) \wedge F_{pre}(R_3)$
0	0	0	0	1	0	0
0	0	1	0	1	0	0
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	1	1
1	1	1	1	0	1	1

→ R_1 und R_2 sind gleichzeitig aktiv und sind somit in Konflikt mit einander

$\Rightarrow$ inkonsistent

c_1	c_2	c_3	$F_{pre}(R_1)$	$\bigwedge$	$\bigwedge$
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	1	1	1

$\Rightarrow$ konsistent

3 Decision Tables

Entscheidungstabellen

T: registration		r_1	r_2	r_3	r_4
c_1	already registered	×	-	-	×
c_2	$n < n_{\max}$	*	×	-	*
c_3	delayed payments	*	-	-	×
a_1	message: "already registered"	×	-	-	-
a_2	message: "fully booked"	-	-	×	-
a_3	message: "payment reminder"	-	-	-	×
a_4	register applicant	-	×	-	-

Figure 3: Decision table.

Abbildung 3: Entscheidungstabelle.

Problem 3.1 (Analysing Decision Tables)

Aufgabe 3.1 (Analyse von Entscheidungstabellen)

Consider the decision table for a registration procedure as shown in Figure 3. In some cases, if applicants receive a message anyway, that message should be combined with a payment reminder if there are delayed payments. The maximum number of registrations is $n_{\max}$, the current number of registrations n.

Betrachten Sie die Entscheidungstabelle für eine Registrierungsprozedur in Abbildung 3. In einigen Fällen, in denen Antragsteller eh eine Nachricht bekommen, soll diese Nachricht mit einer Zahlungserinnerung kombiniert werden, falls Zahlungen ausstehen. Maximal können $n_{\max}$ Registrierungen vorgenommen werden, n bezeichnet die aktuelle Anzahl an Registrierungen.

(i) Is T as given in Figure 3 complete?
Prove your claim by giving a counter-example or a proof of completeness, respectively.

Ist T in Abbildung 3 vollständig?
Beweisen Sie Ihre Behauptung durch ein Gegenbeispiel bzw. einen Beweis der Vollständigkeit.

(2)

Correction:

- $\frac{2}{N}$ points for the correct answer, and for each of the N parts of the argumentation (example or proof)

(ii) Is T as given in Figure 3 deterministic?
Prove your claim by giving a counter-example or a proof of completeness, respectively.

Ist T in Abbildung 3 deterministisch?
Beweisen Sie Ihre Behauptung durch ein Gegenbeispiel bzw. einen Beweis der Vollständigkeit.

(2)

Correction:

- $\frac{2}{N}$ points for the correct answer, and for each of the N parts of the argumentation (example or proof)

(iii) The customer in addition stated that payment reminders should be combined with "other messages with a positive connotation", i.e. being already registered or having been successfully registered, but not with other messages. Is this policy ensured by T in the collecting semantics? Explain!

Der Kunde hat zusätzlich geäußert, daß Zahlungserinnerungen mit "anderen Nachrichten mit positiver Konnotation", also bereits registriert sein oder erfolgreich registriert worden sein, kombiniert werden sollen, aber nicht mit anderen Nachrichten. Wird dieser Grundsatz von T in der Sammelsemantik garantiert? Erläutern Sie!

(1)

- max. 1 point for correct answer and explanation

(iii) The customer in addition stated that payment reminders should be combined with "other messages with a positive connotation", i.e. being already registered or having been successfully registered, but not with other messages. Is this policy ensured by T in the collecting semantics? Explain!

Der Kunde hat zusätzlich geäußert, daß Zahlungserinnerungen mit "anderen Nachrichten mit positiver Konnotation", also bereits registriert sein oder erfolgreich registriert worden sein, kombiniert werden sollen, aber nicht mit anderen Nachrichten. Wird dieser Grundsatz von T in der Sammelsemantik garantiert? Erläutern Sie!

(1)

Problem 3.2 (Writing Decision Tables)

Aufgabe 3.2 (Entscheidungstabellen Erstellen)

Gathering requirements for the software of a new heating controller yielded the following statements by the customer:

- If the room temperature is below target temperature (condition c_1), shift to the next higher heating level (action a_1).
- If the room temperature is above the target temperature (condition c_2), shift to the next lower heating level (action a_2).
- When the "power heating" button is pushed (condition c_3), and the room temperature is not above the target temperature, shift to the next higher heating level.
- If room is neither above nor below the target temperature, and if "power heating" is not pressed, do not change heating level.

Eine erste Anforderungsanalyse für das Programm eines neuen Steuerungsgeräts einer Heizungsanlage hat die folgenden Aussagen des Kunden ergeben:

- Wenn die Raumtemperatur unterhalb der Zieltemperatur liegt (Bedingung c_1), schalte in die nächsthöhere Heizstufe (Aktion a_1).
- Wenn die Raumtemperatur oberhalb der Zieltemperatur liegt (Bedingung c_2), schalte in die nächstniedrigere Heizstufe (Aktion a_2).
- Wenn der "schnell aufheizen"-Knopf gedrückt wird (Bedingung c_3) und die Raumtemperatur nicht oberhalb der Zieltemperatur liegt, schalte in die nächsthöhere Heizstufe.
- Wenn die Raumtemperatur weder oberhalb noch unterhalb der Zieltemperatur liegt und der "schnell aufheizen"-Knopf nicht gedrückt ist, ändere die Heizstufe nicht.

(i) Provide a decision table which formalises these requirements.

Formalisieren Sie die Anforderungen mit Hilfe einer Enscheidungstabelle. (4)

Correction:

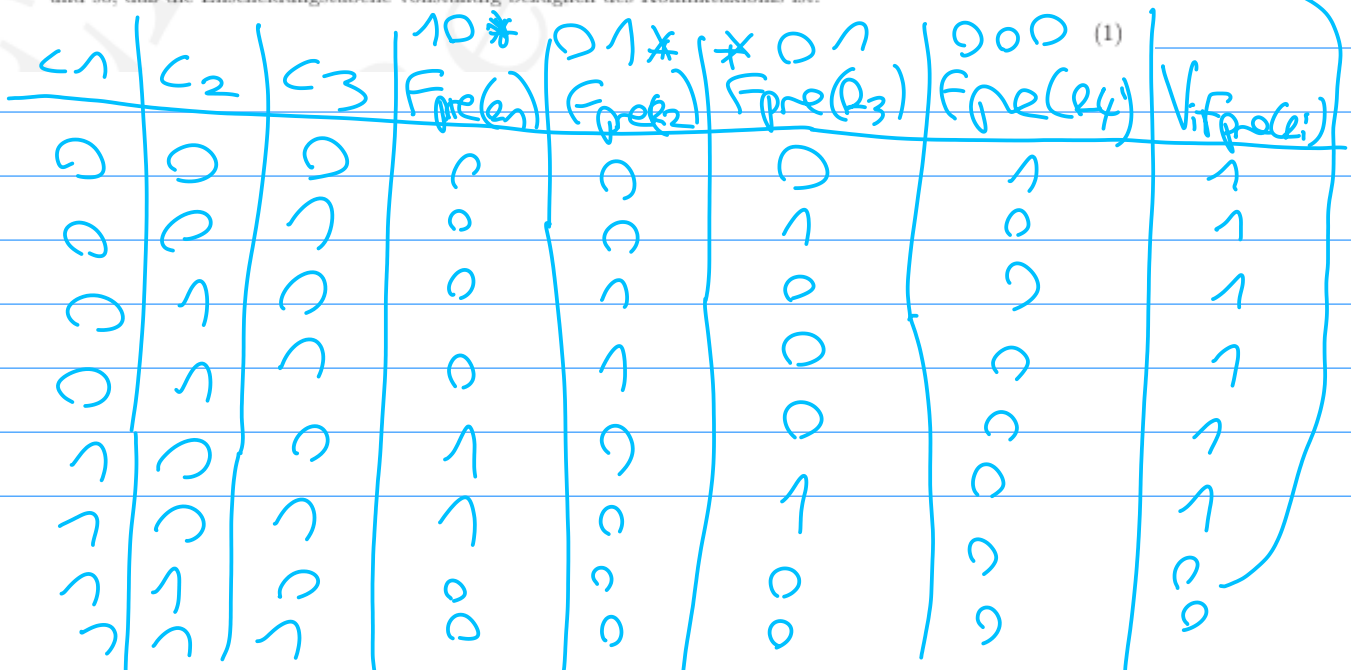
- one point per correct rule

(ii) Analyse the requirements for completeness by checking the resulting decision table for completeness.

If the decision table is complete, provide a proof, if not, propose an environment assumption in the form of a conflict axiom (which is consistent with all existing rules in the table) such that the decision table is complete with respect to the conflict axiom.

Analysieren Sie die Anforderungen auf Vollständigkeit indem Sie die resultierende Entscheidungstabelle auf Vollständigkeit überprüfen.

Wenn die Enscheidungstabelle vollständig ist, geben Sie einen Beweis an; wenn nicht, schlagen Sie eine Umgebungsannahme in Form eines Konfliktaxioms vor, das mit allen in der Tabelle vorhandenen Regeln konsistent ist und so, daß die Enscheidungstabelle vollständig bezüglich des Konfliktaxioms ist. (1)



$$\text{Conf. Axiom} = (C_1 \cap C_2)$$

2 Software Metrics

Software Metriken

Problem 2.1 (Lines of Code)

Aufgabe 2.1 (Anzahl Codezeilen)

```
1  public int convert(char[] str) throws Exception {
2    if (str.length > 6)
3      throw new Exception("Length exceeded");
4
5    int number = 0;
6    int digit;
7    int i = 0;
8
9    // none of the early submitters found this bug! :-)
10   //
11   if (str[0] == '-')
12     i = 1;
13
14   while (i < str.length){
15     digit = str[i] - '0';
16     if (digit < 0 || digit > 9)
17         throw new Exception("Invalid character");
18     number = number * 10 + digit;
19     i = i + 1;
20   }
21
22   if (str[0] == '-')
23     number = -number;
24
25   if (number > 32767 || number < -32768)
26     throw new Exception("Range exceeded");
27
28   return number;
29 }
```

Figure 2: Function `convert`.

Abbildung 2: Funktion `convert`.

Consider the function `convert` shown in Figure 2.

(i) Which lines (line numbers) contribute to the program size of `convert`?

(ii) What is the net program size of `convert`?
Which lines *do not* contribute to the net program size of `convert`?

(iii) What is the code size of `convert`?
Which lines *do not* contribute to the code size of `convert`?

Abbildung 2 zeigt die Funktion `convert`.

(i) Welche Zeilen(nummern) werden für die Programmgröße von `convert` gezählt?

(ii) Was ist die Netto-Programmgröße von `convert`?
Welche Zeilen(nummern) werden für die Netto-Programmgröße von `convert` *nicht* gezählt?

(iii) Was ist die Codegröße von `convert`?
Welche Zeilen(nummern) werden für die Codegröße von `convert` *nicht* gezählt?

Exercise 3 – Creating Process Models (5/20 Points)

Working on software engineering tasks is an activity and the creation of each solution to a software engineering task does have a process. So it should be possible to model these processes.

(i) Create a process model of the way of working that is employed by your team between the issue of an exercise sheet of this course and your final submisison (or make up a plausible way of working, if you do not want to disclose the one that your team actually follows). Use the basic process model building blocks (activity, role, etc.) and briefly describe the activity, role, etc. that they represent. (3)

(ii) If a team is looking for a way of working that promises that every team member gets about the same learning effect, would you recomend the approach that you have modelled in Task (ii)? Why (not)? Discuss advantages and disadvantages on the basis of the process model. (2)