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

Consider the decision tables shown in Figure 1.

Symbol, dass das Gegenteil ist

Reihe clash von oben nach unten, sobald man am Ende ankommt ist UND es nicht geschafft

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 ↯ 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) $(C1 \wedge \neg C2 \wedge True) \vee (\neg C1 \wedge True \wedge \neg C3)$
$\vee (True \wedge C2 \wedge True) \vee \neg(\neg C1 \wedge \neg C2)$

$\Rightarrow$ Nein $C1 = F$, $C2 = F$, $C3 = T$

C1	C2	C3	F(C1)	F(C2)	F(C3)	CA
0	0	0	0	1		
0	0	1	0	0	0	1
0	1	0	0	1		
0	1	1	0	0	1	
1	0	0	1			

1	0	1	1			
1	1	0	0	0	1	
1	1	1	0	1		

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

(b) Decision Table 2

durchgehen

geht schnelle

$c_2 \vee (\neg c_2 \wedge c_3) \vee (c_1 \wedge \neg c_2 \wedge \neg c_3) \vee (c_1 \wedge \neg c_2 \wedge \neg c_3)$

c_1	c_2	c_3	$f(r_1)$	$f(r_2)$	$f(r_3)$	$f(r_4)$
0	0	0	1	0	1	
0	0	1	0	1		
0	1	0	1			
0	1	1	1			
1	0	0	0	0	0	1
1	0	1	0	1		
1	1	0	1			
1	1	1	1			

$\Rightarrow$ 1

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

DT1: $(C_1 \wedge \neg C_2)$ $(\neg C_1 \wedge \neg C_3)$ C_2

C_1	C_2	$\neg C_1 \wedge C_3$	$\neg C_1 \wedge \neg C_2$	$\neg C_1 \wedge C_3$	C_2	$C_1 \wedge \neg C_2 \wedge \neg C_3$
			F_2		F_3	
0	0	0	0	0	1	0
0	0	1	0	0	1	0
0	1	0	0	0	1	0
0	1	1	0	0	1	0
1	0	0	0	0	1	0
1	0	1	0	0	1	0
1	1	0	0	0	1	0
1	1	1	0	0	1	0

direkt

direkt

direkt

alle Zeilen einer Klasse = 1 eine 0 ist = 1

F_1 F_2 F_3 F_4

DT2: $C_2 \vee (C_2 \wedge C_3) \vee (C_1 \wedge \neg C_2 \wedge \neg C_3) \vee (C_1 \wedge \neg C_2 \wedge C_3)$

Nein

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

(b) Decision Table 2

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

Conflicting acti

C_1	C_2	C_3	C_1 $F(r_1)$	$\neg C_2 \wedge C_3$ $F(r_2)$	$\neg C_1 \wedge \neg C_2 \wedge \neg C_3$ $F(r_3)$	$C_1 \wedge \neg C_2 \wedge \neg C_3$ $F(C_1)$
0	0	0	0	0	1	0
0	0	1	0	1	0	0
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0		1
1	0	1	0	1		1
1	1	0	1	0		1
1	1	1	1	0		0

$\Rightarrow$ det.?

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

b) Nur, dass das Gegenteil ist

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

bei subsumed

Nein, kein Wert 2 in einer Zeile

$f(r_1) \Rightarrow f(DT_2)$

Fall

~~def → keine usefy rules~~

~~– zwei Regeln in einer Zeile mit beiden aber nicht gleichen Effekt~~

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

Conflicting actions: A1 ≠ A2

(c) Decision Table 3

→ direkt nope

$c_1 \wedge c_2$ $\neg c_1$ c_2

c_1	c_2	c_3	$F_{pro}(c_1)$	$F_{pro}(c_2)$	$F_{pro}(c_3)$
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1

c_2 u. c_3 Effekte

c_1 u. c_3 effekte

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

1) 15'

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

Conflicting actions: A1 ↯ A2

(c) Decision Table 3

C1	C2	C3	C1	C2	¬C1	C2	¬(C1 ∧ C2)
			Pre(C1)	Pre(C2)	Pre(C3)		
0	0	0	0	1	0		1
0	0	1	0	1	0		1
0	1	0	0	1	1		1
0	1	1	0	1	1		1
1	0	0	0	0	0		1
1	0	1	0	0	0		1
1	1	0	1	0	1		0
1	1	1	1	0	1		0

Exercise 1 – LSC Syntax (6/20 Points)

Consider the Live Sequence Chart in Figure 1.

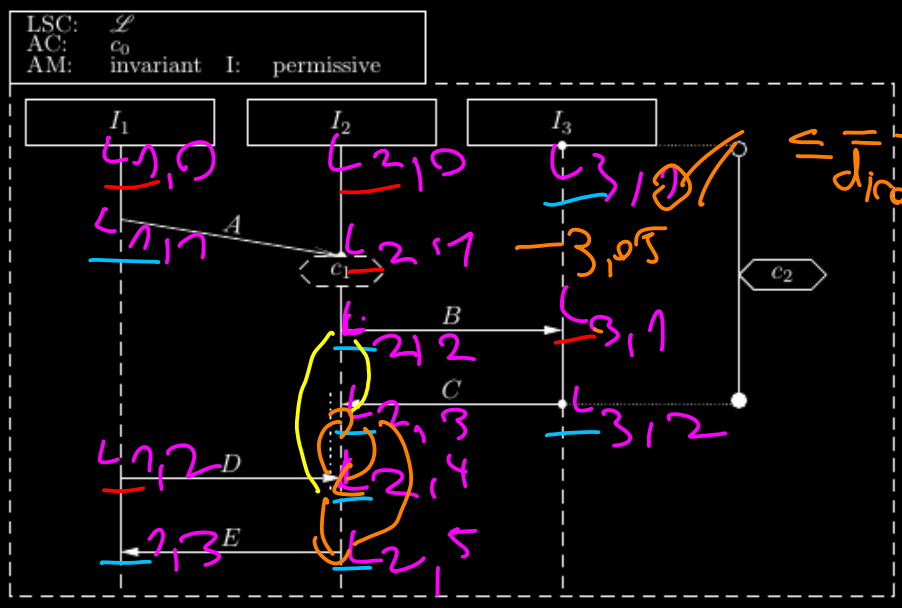


Figure 1: A Live Sequence Chart.

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)

$\preceq_{direct} = \{(10,11),(11,12),(12,13),(11,21),(20,21),(21,22),(22,23),(23,24),(24,25),\cancel{(31,32)},\cancel{(32,33)},(23,24)\}$

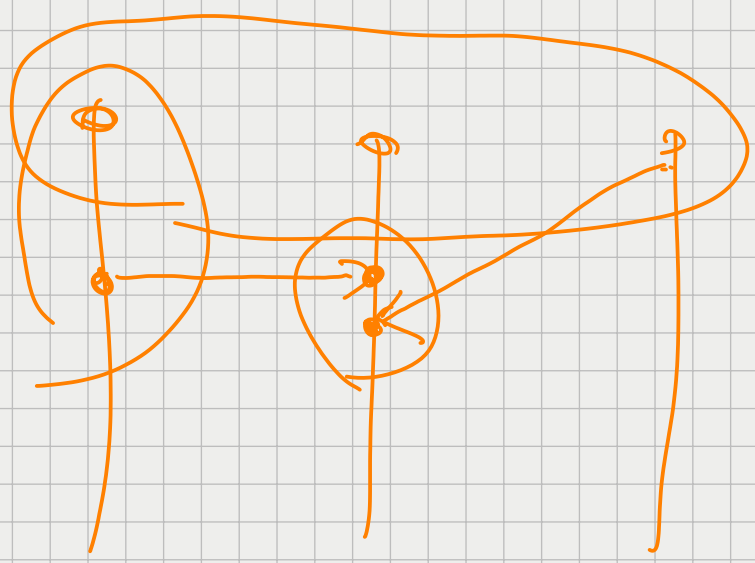
$\sim = \{(23,23),(23,25)$

$\ldots$

$Cond = \{(\{(2,3),(1,3\}$

$Msg = \{(L_{22}, B, L_{32}), \ldots\}$

$LocInv = \{(L_{31}, 0, L_2, L_{32}, 0)\}$



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 $\neg F_{!?}^{J_1,J_2}$.

2

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

$$\pi_1 = \sigma_0 \xrightarrow{\tau} \sigma_1 \xrightarrow{A_!^{I_1,I_2}} \sigma_2 \xrightarrow{A_?^{I_1,I_2}} \sigma_3 \xrightarrow{\tau} \sigma_4 \xrightarrow{\tau} \sigma_5 \xrightarrow{\tau} \cdots$$

$$\boxed{c_0} \quad \boxed{c_2} \quad \boxed{c_1 \wedge c_2} \quad \boxed{\neg c_1 \wedge c_2} \quad \boxed{c_2} \quad \boxed{c_2}$$

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)

(i)

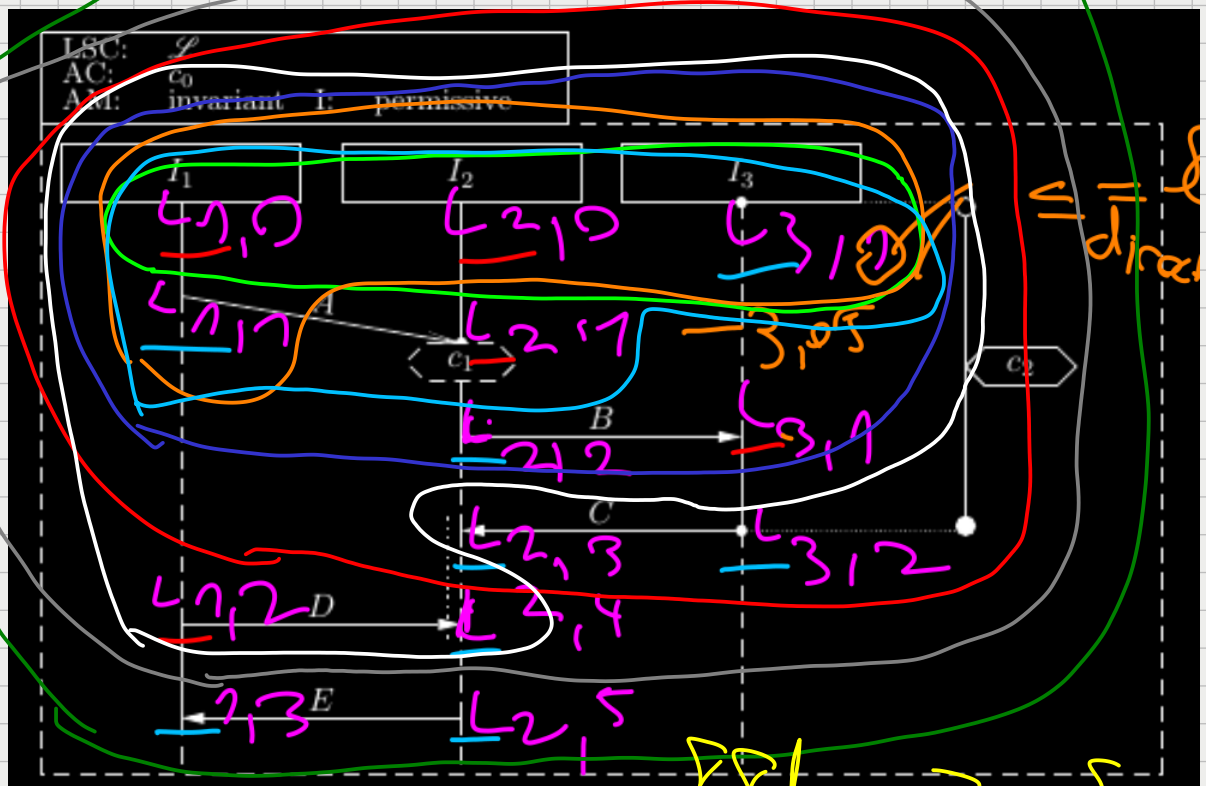


Figure 1: A Live Sequence Chart.

$c_1 = \{$ $\}$

$c_2 = \{$ $\}$

$c_3 = \{$ $\}$

$c_4 = \{$ $\}$

$C_5 =$ $\{\{l_{1,2}, l_{2,1}\}\}$

$C_6 =$ $\{\{l_{2,3}, l_{3,2}\}\}$

$C_7 =$ $\{\{l_{1,3}, l_{2,5}\}\}$

$C_8 =$ $\{\}$

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

$$\pi_1 = \sigma_0 \xrightarrow{\tau} \sigma_1 \xrightarrow{A_!^{I_1,I_2}} \sigma_2 \xrightarrow{A_?^{I_1,I_2}} \sigma_3 \xrightarrow{\tau} \sigma_4 \xrightarrow{\tau} \sigma_5 \xrightarrow{\tau} \cdots$$

$$\boxed{c_0} \quad \boxed{c_2} \quad \boxed{c_1 \wedge c_2} \quad \boxed{\neg c_1 \wedge c_2} \quad \boxed{c_2} \quad \boxed{c_2}$$

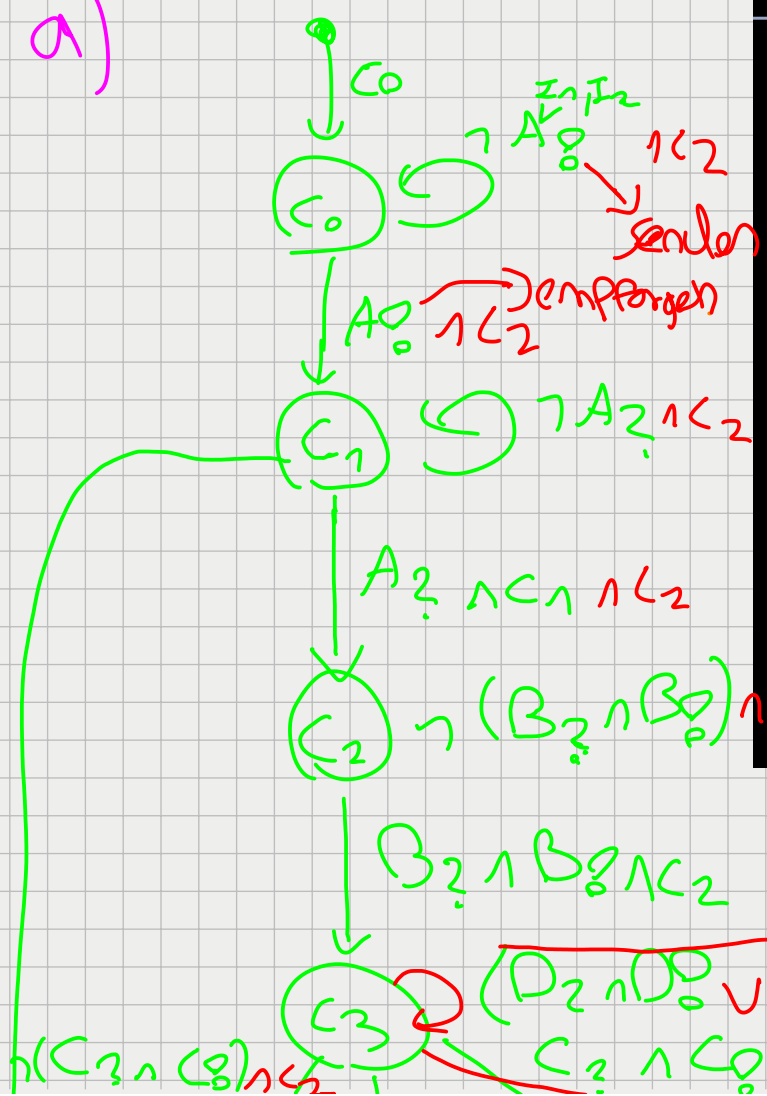
a) Does π_1 satisfy LSC $\mathscr{L}$ or does it violate the chart? (1)
Hint: Argue your result on the automaton from the previous task.

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):

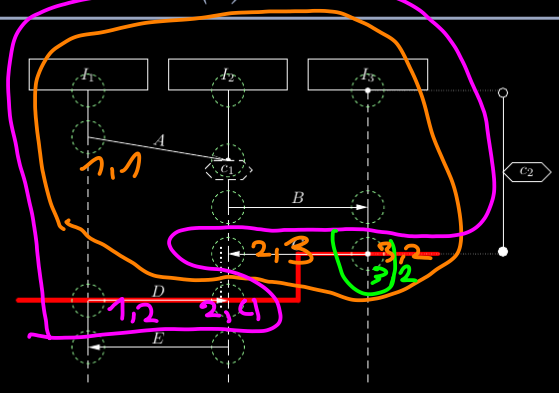
- Path violates chart.
- Path satisfies chart by taking a legal exit.
- Path (activates and) satisfies chart without taking a legal exit.

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

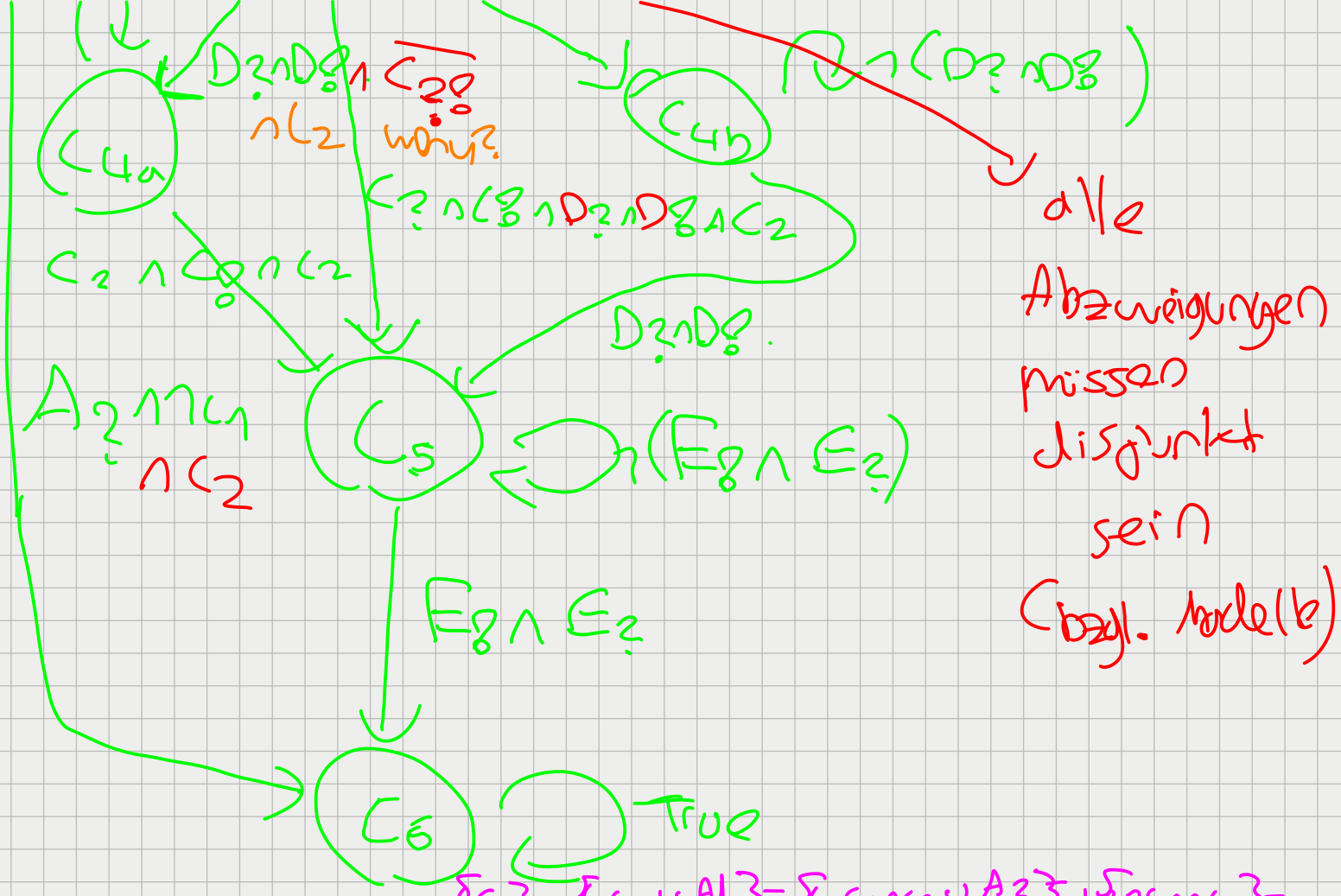
a)

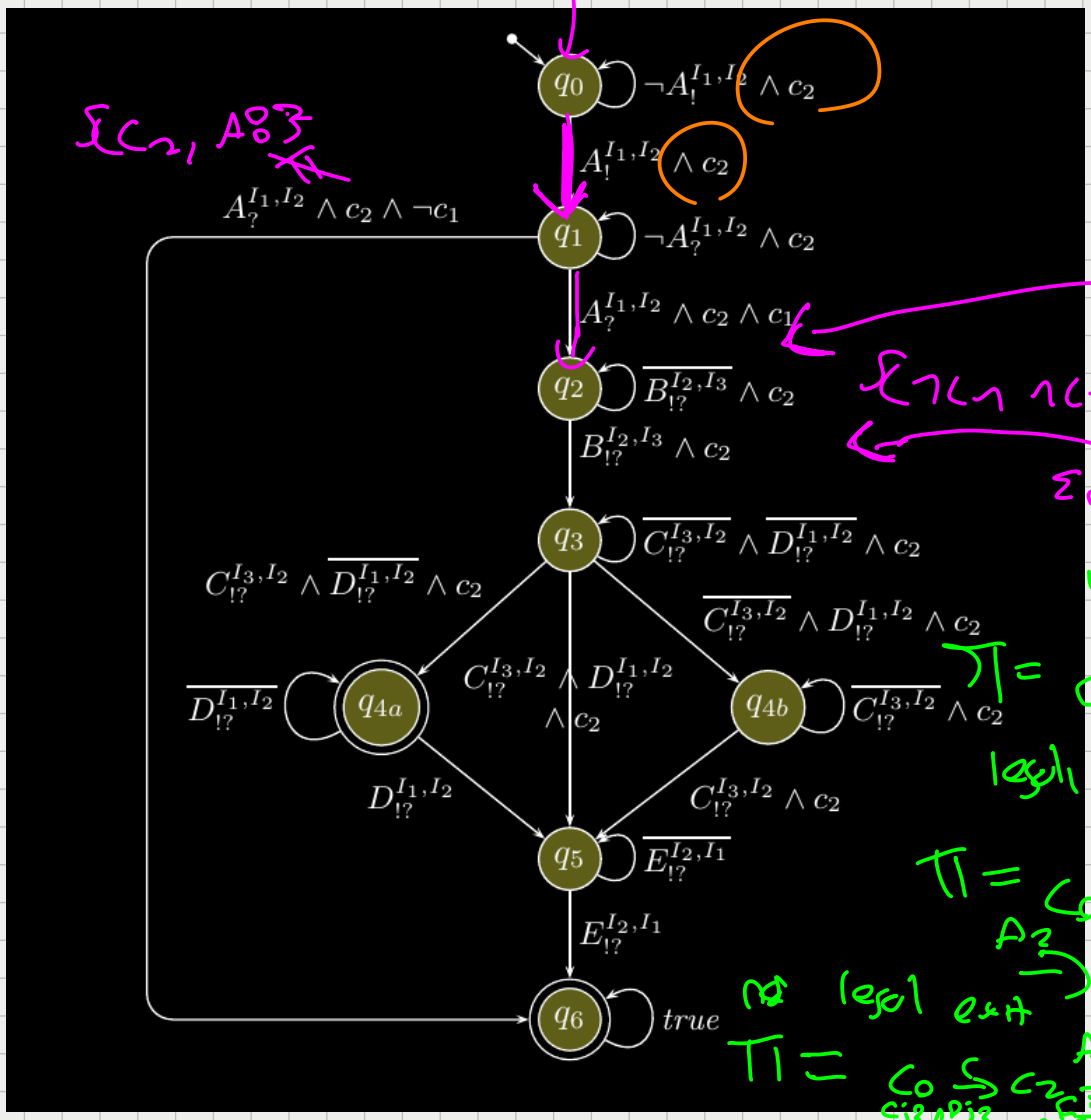


Exercise 2.(i) — LSC Semantics



- $C_0 = \{l_{1,0}, l_{2,0}, l_{3,0}\}$
- $C_1 = \{l_{1,0}, l_{2,0}, l_{3,0}, l_{1,1}\}$
- $C_2 = \{l_{1,0}, l_{2,0}, l_{3,0}, l_{1,1}, l_{2,1}\}$
- $C_3 = \{\ldots, l_{1,1}, l_{2,2}, l_{3,1}\}$
- $C_{4a} = \{\ldots, l_{1,1}, l_{2,3}, l_{3,2}\}$
- $C_{4b} = \{\ldots, l_{1,2}, l_{2,4}, l_{3,2}\}$ $C_{3,1}?$
- $C_5 = \{\ldots, l_{1,2}, l_{2,3}, l_{2,4}, l_{3,2}\}$
- $C_6 = \{\ldots, l_{1,3}, l_{2,5}, l_{3,2}\}$
- $\mathcal{F}_0 = \{l_{1,1}\}, \mathcal{F}_1 = \{l_{2,1}\}$, $\mathcal{F}_2 = \{l_{2,2}, l_{3,1}\}, \mathcal{F}_{3a} = \{l_{2,3}, l_{3,2}\}$, $\mathcal{F}_{3b} = \{l_{1,2}, l_{2,4}\}, \mathcal{F}_3 = \mathcal{F}_{3a} \cup \mathcal{F}_{3b}$, $\mathcal{F}_{4a} = \mathcal{F}_{3b}, \mathcal{F}_{4b} = \mathcal{F}_{3a}, \mathcal{F}_5 = \{l_{1,3}, l_{2,5}\}$


C_{4a}
C_{4b}
C_5
C_6
$D_?^1 \wedge D_!^2 \wedge \overline{C_?^2}$
$\wedge C_2$ wohin?
$C_?^1 \wedge C_!^2 \wedge D_?^1 \wedge D_!^2 \wedge C_2$
$C_2 \wedge C_0 \wedge C_2$
$D_?^1 \wedge D_!^2$
$A_?^1 \wedge C_1 \wedge C_2$
$\neg(E_!^2 \wedge E_?^1)$
$E_!^2 \wedge E_?^1$
True
$\neg(D_?^1 \wedge D_!^2)$
alle Abzweigungen müssen disjunkt sein (bzgl. Modell)
$\{C_0\}, \{C_2 \cup A_!^1\}, \{C_1 \wedge C_2 \cup A_?^1\}, \{\neg C_1 \wedge C_2\}, \{C_2\}, \{C_2\}$


q_0
$\neg A_!^{I_1,I_2} \wedge c_2$
$A_!^{I_1,I_2} \wedge c_2$
$\{C_2, A_!^2\}$
$A_?^{I_1,I_2} \wedge c_2 \wedge \neg c_1$
q_1
$\neg A_?^{I_1,I_2} \wedge c_2$
$A_?^{I_1,I_2} \wedge c_2 \wedge c_1$
q_2
$\overline{B_{!?}^{I_2,I_3}} \wedge c_2$
$B_{!?}^{I_2,I_3} \wedge c_2$
$\{\neg C_1 \wedge \neg C_2\} \models \{\text{alle}\}$
Ende
violate
q_3
$\overline{C_{!?}^{I_3,I_2}} \wedge \overline{D_{!?}^{I_1,I_2}} \wedge c_2$
$C_{!?}^{I_3,I_2} \wedge \overline{D_{!?}^{I_1,I_2}} \wedge c_2$
$\overline{C_{!?}^{I_3,I_2}} \wedge D_{!?}^{I_1,I_2} \wedge c_2$
$C_{!?}^{I_3,I_2} \wedge D_{!?}^{I_1,I_2} \wedge c_2$
$\overline{D_{!?}^{I_1,I_2}}$
q_{4a}
q_{4b}
$\overline{C_{!?}^{I_3,I_2}} \wedge c_2$
$D_{!?}^{I_1,I_2}$
$C_{!?}^{I_3,I_2} \wedge c_2$
q_5
$\overline{E_{!?}^{I_2,I_1}}$
$E_{!?}^{I_2,I_1}$
q_6
true
$\Pi = C_0 \xrightarrow{S} \neg C_2 \xrightarrow{A!} T$
legal exit.
$\Pi = C_0 \xrightarrow{S} C_2 \xrightarrow{A!} \neg C_1 \neg C_2$
$A_?$
...
nicht legal exit
$\Pi = C_0 \xrightarrow{S} C_2 \xrightarrow{A!} C_2 \wedge C_1 \xrightarrow{A_?} C_2 \xrightarrow{B!?} C_2$
$C_2 \wedge \neg C_1 \xrightarrow{B_!^2} \emptyset \xrightarrow{A_?^1} \ldots$

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

$$\pi_1 = \sigma_0 \xrightarrow{\tau} \sigma_1 \xrightarrow{A_!^{I_1,I_2}} \sigma_2 \xrightarrow{A_?^{I_1,I_2}} \sigma_3 \xrightarrow{\tau} \sigma_4 \xrightarrow{\tau} \sigma_5 \xrightarrow{\tau} \cdots$$

$$\boxed{c_0} \quad \boxed{c_2} \quad \boxed{c_1 \wedge c_2} \quad \boxed{\neg c_1 \wedge c_2} \quad \boxed{c_2} \quad \boxed{c_2}$$

$\{c_0\}$, $\{c_2 \cup A!\}$, $\{c_1 \cap c_2 \cup A?\}$, $\{\neg c_1 \cap c_2\}$, $\{c_2\}$, $\{c_2\}$

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)

$$\pi_{1,1} = \sigma_0 \xrightarrow{\tau} \sigma_1 \xrightarrow{\tau} \sigma_2 \xrightarrow{\tau} \sigma_3 \xrightarrow{\tau} \sigma_4 \xrightarrow{\tau} \sigma_5 \xrightarrow{\tau} \cdots$$

$$\boxed{c_0} \quad \boxed{\neg c_2} \quad \boxed{\cdot} \quad \boxed{\cdot} \quad \boxed{\cdot} \quad \boxed{\cdot}$$

not accepted: violates hot locinv

$$\pi_{1,3} = \sigma_0 \xrightarrow{\tau} \sigma_1 \xrightarrow{A_!^{I_1,I_2}} \sigma_2 \xrightarrow{A_?^{I_1,I_2}} \sigma_3 \xrightarrow{B_!^{I_1,I_2}} \sigma_4 \xrightarrow{\tau} \sigma_5 \xrightarrow{\tau} \cdots$$

$$\boxed{c_0} \quad \boxed{c_2} \quad \boxed{c_1 \wedge c_2} \quad \boxed{c_2} \quad \boxed{\cdot} \quad \boxed{\cdot}$$

not accepted, violates simultaneity

$c_2 \cap c_1$!?

$$\pi_{2,1} = \sigma_0 \xrightarrow{\tau} \sigma_1 \xrightarrow{A_!^{I_1,I_2}} \sigma_2 \xrightarrow{A_?^{I_1,I_2}} \sigma_3 \xrightarrow{\tau} \sigma_4 \xrightarrow{\tau} \sigma_5 \xrightarrow{\tau} \cdots$$

$$\boxed{c_0} \quad \boxed{c_2} \quad \boxed{\neg c_1 \wedge c_2} \quad \boxed{\cdot} \quad \boxed{\cdot} \quad \boxed{\cdot}$$

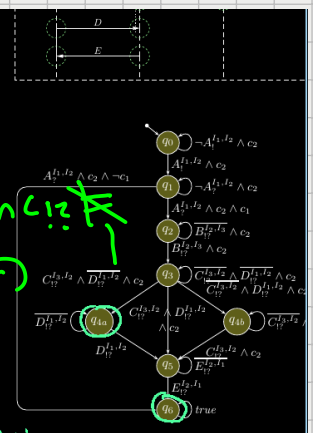
accepted: legal exit (q_4)

$$\pi_{2,2} = \sigma_0 \xrightarrow{\tau} \sigma_1 \xrightarrow{A_!^{I_1,I_2}} \sigma_2 \xrightarrow{A_?^{I_1,I_2}} \sigma_3 \xrightarrow{B_{!?}^{I_2,I_3}} \sigma_4 \xrightarrow{C_{!?}^{I_3,I_2}} \sigma_5 \xrightarrow{\tau} \cdots$$

$$\boxed{c_0} \quad \boxed{c_2} \quad \boxed{c_1 \wedge c_2} \quad \boxed{c_2} \quad \boxed{c_2} \quad \boxed{\cdot}$$

~~accepted: ends in q_{4a} (accepting state)~~

$\cap D_{!?}$



$$\pi_{2,3} = \sigma_0 \xrightarrow{\tau} \sigma_1 \xrightarrow{A_!^{I_1,I_2}} \sigma_2 \xrightarrow{A_?^{I_1,I_2}} \sigma_3 \xrightarrow{B_{!?}^{I_2,I_3}} \sigma_4 \xrightarrow{C_{!?}^{I_3,I_2}} \sigma_5 \xrightarrow{D_{!?}^{I_1,I_2}} \sigma_6 \xrightarrow{E_{!?}^{I_2,I_1}} \sigma_7 \xrightarrow{\tau} \cdots$$

$$\boxed{c_0} \quad \boxed{c_2} \quad \boxed{c_1 \wedge c_2} \quad \boxed{c_2} \quad \boxed{c_2} \quad \boxed{\cdot} \quad \boxed{\cdot} \quad \boxed{\cdot}$$

accepted ends in q_6 (accepting state)

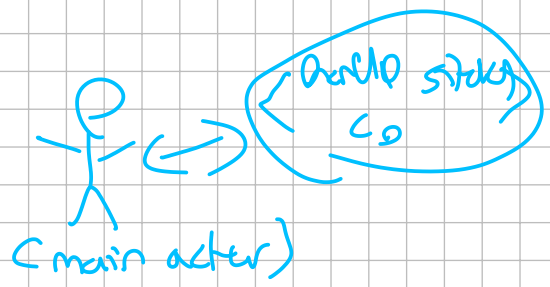
$\cap D_{!?}$

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.

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)

Figure 2: Example Use Case.

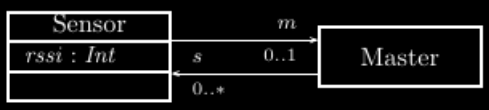


Figure 3: WFAS masters and slaves.

(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)

(i) $\cancel{\mathscr{D}(M) = \mathbb{N} \times \mathscr{C}^{M^S}}$

$\cancel{\mathscr{D}(S) = \mathbb{N} \times \mathscr{C}^{S^S}}$

$\varphi = (\{Int\}, \{S, M\}, \{rssi : Int,$

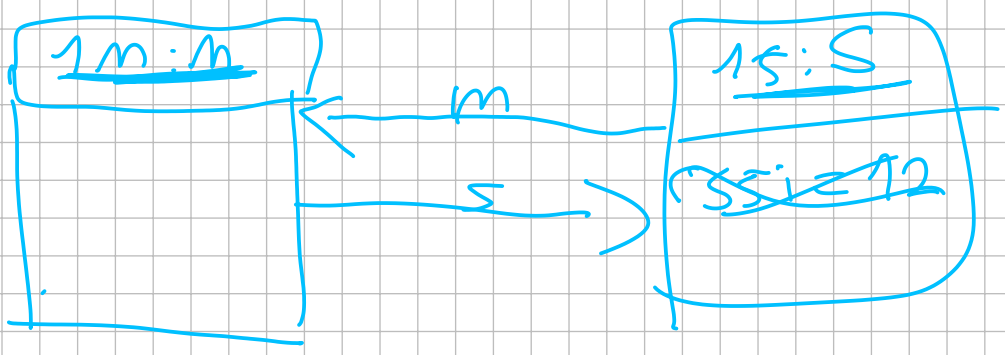
$m : M_{0..1}, s : S_{0..*}\}, \{S \mapsto \{m, rssi\}\}$

$M \mapsto \{\}\}, \emptyset, \{S \mapsto \emptyset, M \mapsto \emptyset\})$

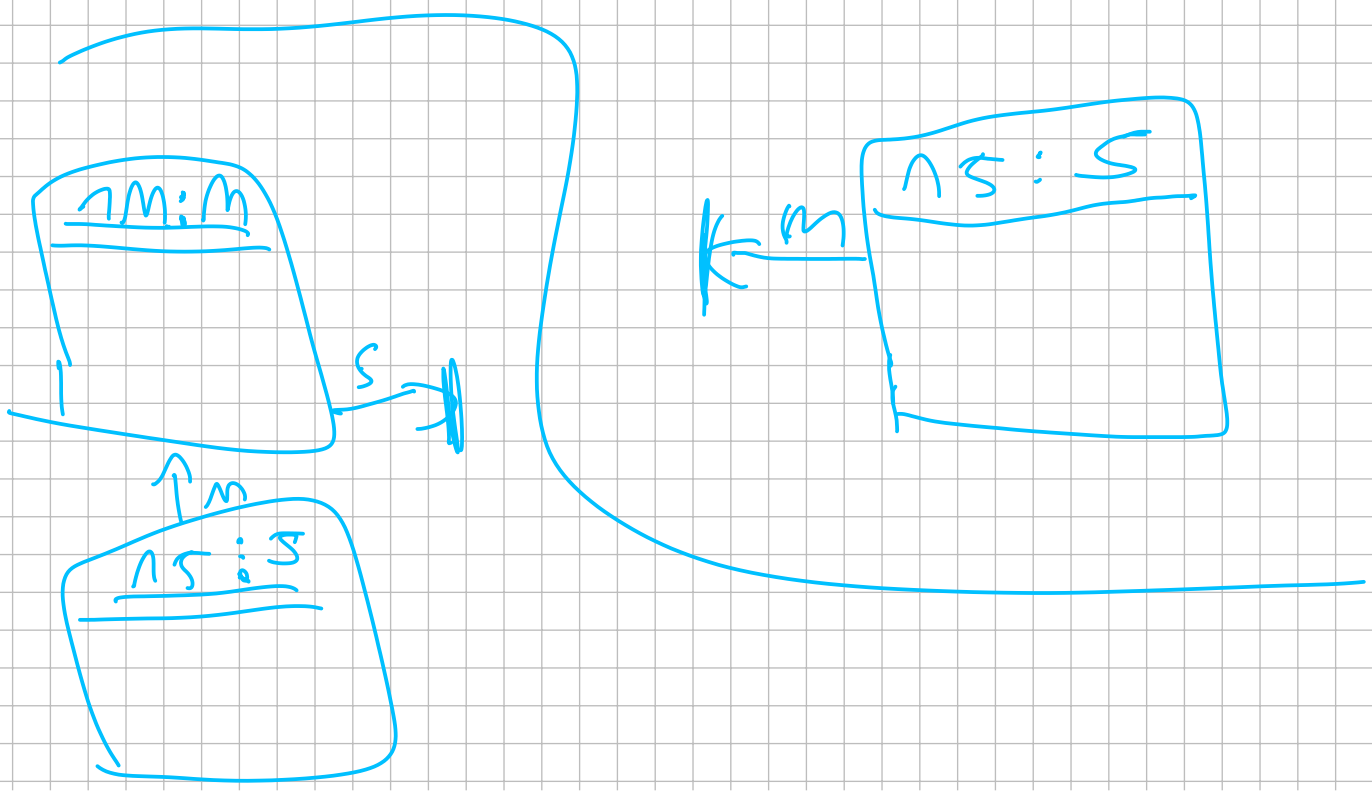
(ii) $D(Int) = \mathbb{Z}$

$D(M) = \mathbb{Z} \times \{M\}$

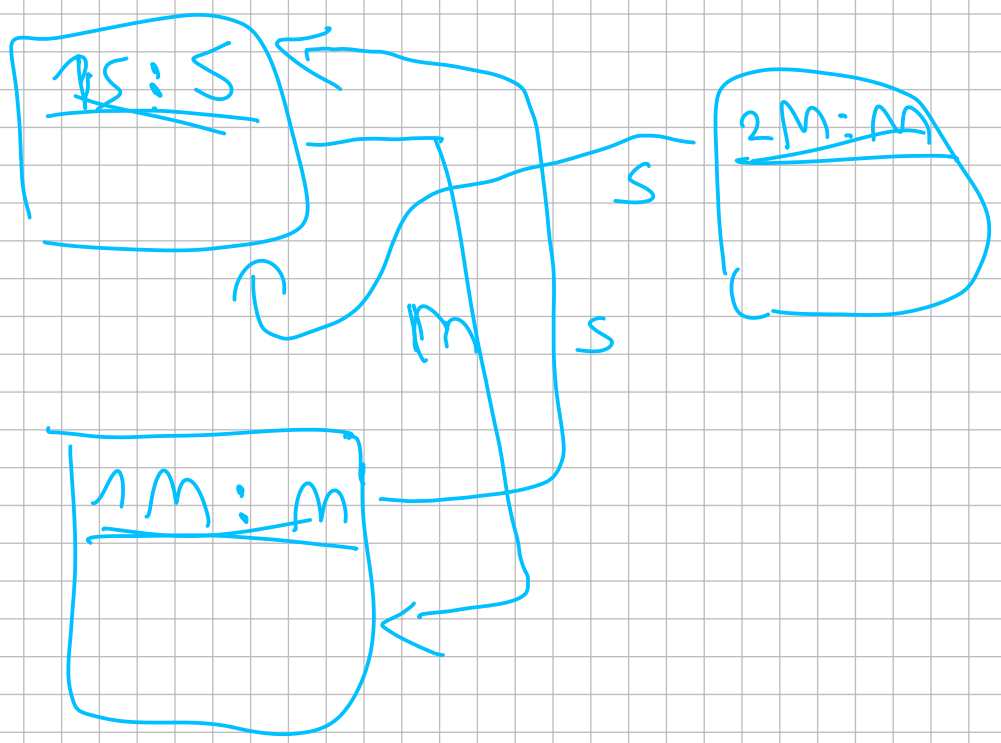
$D(S) = \mathbb{Z} \times \{S\}$



not allowed:



- 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)



$\sigma = \{ 1S \mapsto \{ rssi \mapsto 2 \}, m \mapsto \{ 1M \} \}$
$1M \mapsto \{ s \mapsto \{ 1S \} \}$
$2M \mapsto \{ s \mapsto \{ 1S \} \}$

Exercise 1 – Evaluating OCL Formulae (5/20 Points)

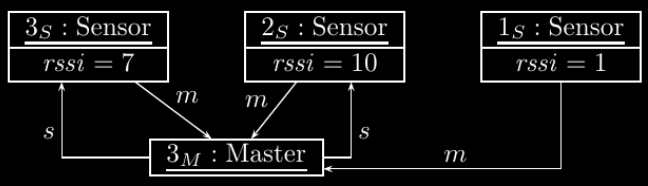


Figure 1: Complete object diagram.

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

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

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

evaluate for σ_1? Prove your claim, i.e., compute $\mathcal{I}[\![F]\!](\sigma_1, \emptyset)$. (3)

(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.

$\sigma = \{ 3S \mapsto \{ rssi \mapsto 7, m \mapsto 3M \},$
$2S \mapsto \{ rssi \mapsto 10, m \mapsto 3M \},$
$1S \mapsto \{ rssi \mapsto 1, m \mapsto 3M \},$

$3M \mapsto \{ s \mapsto \{ 3s, 2s \} \}$

a) $\Leftarrow \forall self \in allInstances_{Master} \bullet \forall n \in s(self) \bullet rssi(n) > 1$

$I[[F]](\sigma_1, \emptyset) = True$

- $I[[allInstances_{Master}]](\sigma, \emptyset) = \{ 3M \}$
- $\beta_1 = \{ self \mapsto 3M \}$
- $I[[s(self)]](\sigma, \beta_1) = \{ 3s, 2s \}$
- $\beta_1 = \{ self \mapsto 3M, n \mapsto 3s \}$, $\beta_2 = \{ self \mapsto 3M, n \mapsto 2s \}$

$= I[[(rssi(n) > 1)]]$

$= I[[>(I[[rssi(n)]], I[[1]])]]$

$\overset{1)}{=} I[[rssi(n)]] = I[[n]](\sigma, \beta_1)(rssi) = \{ 7 \}$

$\overset{2)}{=} I[[rssi(n)]] = I[[n]](\sigma, \beta_2)(rssi) = \{ 10 \}$

$\overset{3)}{=} I[[1]] = 1_I$

$\overset{1}{=} I[[>([7, 1_I])]] = True$

$\overset{2}{=} I[[>([10, 1_I])]] = True$

$\Rightarrow I[[F]](\sigma, \emptyset) = True$ ✓

b) $\sigma_3 = \{ 3s \to \{ rssi \mapsto \{ 1 \}, m \mapsto 3M \}, \ldots \}$

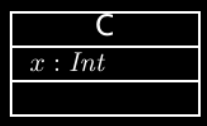
$\delta_2 = \{ \ldots, 3M \mapsto \{ s \mapsto \{ 2s \}, 2s \to \{ rssi \mapsto \emptyset \}, m \mapsto 3M \} \}$ $(3s)$

$= \perp$, da $I[[rssi(\emptyset)]](\sigma, \beta_1) \notin dom(\sigma)$

$\beta_1 = \{ self \mapsto 3M, n \mapsto 2s \}$

$\emptyset \in dom(\sigma)$?

σ : $\boxed{\begin{array}{c} 1_C : \mathsf{C} \\ \hline x = 13 \end{array}}$



$\sigma = \{1_C \mapsto \{x \mapsto 13\}\}$

$$\underbrace{\forall c \in allInstances_C \bullet x(c) \neq 27}$$

- Recall **prefix notation**: $\forall c \in allInstances_C \bullet \neq(x(c), 27)$

  Note: $\neq$ is a binary function symbol, 27 is a 0-ary function symbol.

$F = \forall c \in allInstances_C \bullet x(c) \neq 27$

$\mathcal{I}[\![F]\!](\sigma, \emptyset)$

$- \mathcal{I}[\![allInstances_C]\!](\sigma, \emptyset) = \{1_C\}$

$- \beta_1 = \{c \mapsto 1_C\}$

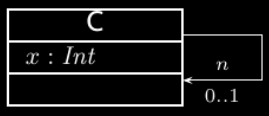
$= \neq(\mathcal{I}[\![x(c)]\!](\sigma,\beta_1), \mathcal{I}[\![27]\!])$

$- \mathcal{I}[\![c]\!](\sigma,\beta_1)(x) = \{13\}$

$- \mathcal{I}[\![27]\!] = \{27_I\}$

$= \neq(13, 27) = True$

σ : $\boxed{\begin{array}{c} 1_C : \mathsf{C} \\ \hline x = 13 \end{array}}$ $\xrightarrow{n}$



$\sigma = \{1_C \mapsto \{x \mapsto 13, n \mapsto \{\}\}\}$

$$\forall c \in allInstances_C \bullet x(n(c)) \neq 27$$

- Similar to the previous slide, we need the value of

$$\mathcal{I}[\![x(n(c))]\!](\sigma, \beta), \beta = \{c \mapsto 1_C\}$$

$\beta = \{1_C \mapsto \{x \mapsto 13, n \mapsto \emptyset\}\}$

$\mathcal{I}[\![F]\!](\sigma, \emptyset)$

$- \mathcal{I}[\![allInstances_C]\!] = \{1_C\}$

$- \beta_1 = \{c \mapsto 1_C\}$

$= \neq(\mathcal{I}[\![x(n(c))]\!](\sigma,\beta_1), \mathcal{I}[\![27]\!](\sigma,\beta_1))$

$- \mathcal{I}[\![c]\!](\sigma,\beta_1)(n)(x) = \bot$, since $\mathcal{I}[\![c]\!](\sigma,\beta_1) \overset{(*)}{=} \bot$ since $\mathcal{I}[\![x(c)]\!](\sigma,\beta_1) \notin dom(\sigma)$ and

?

$- \mathcal{I}[\![27]\!] = 27_I$

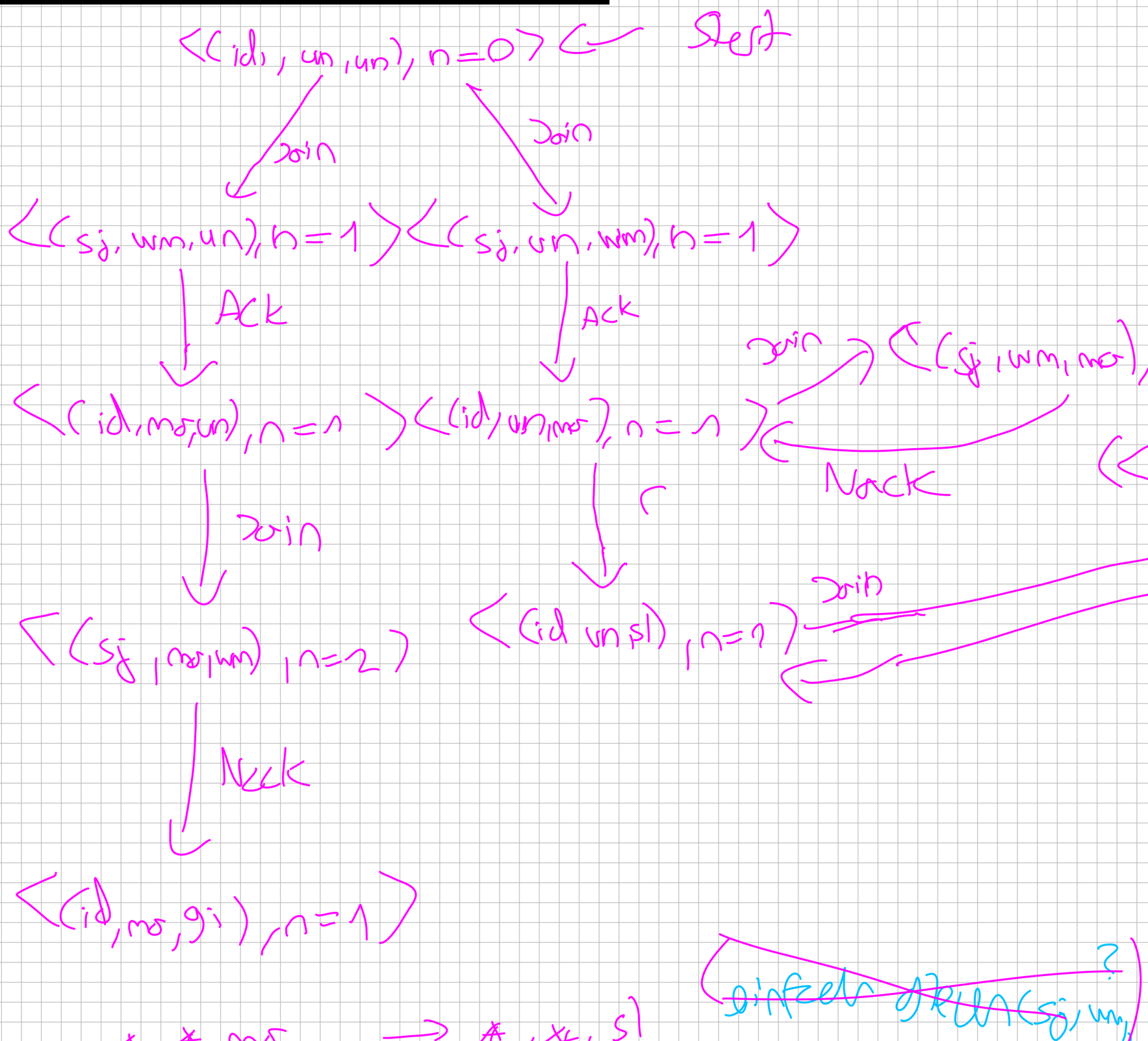
$= \neq(\bot, 27_I) = \bot$

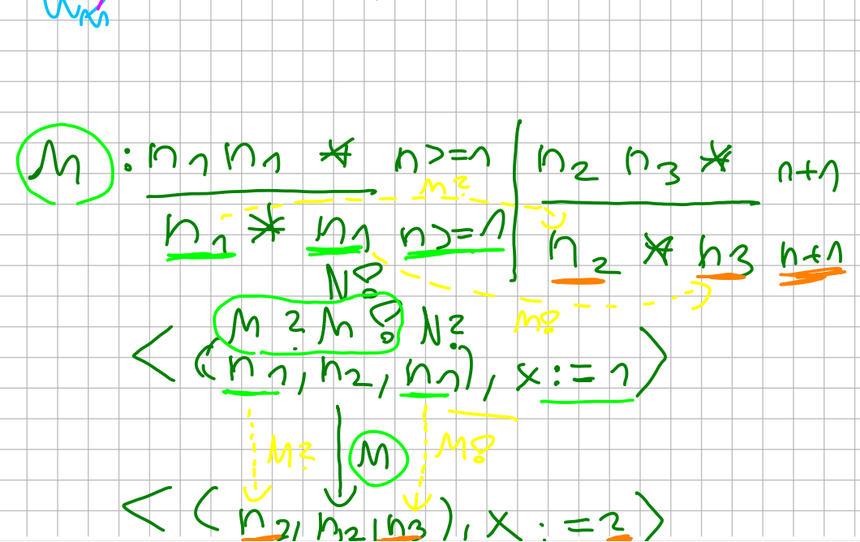
Exercise 2 – Computation Graph / Transition Graph (6/20 Points)

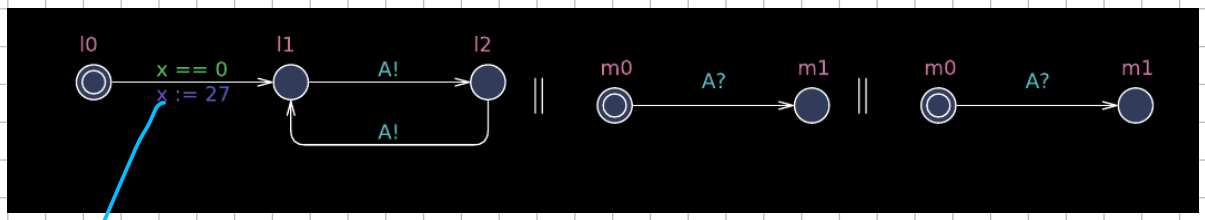
Figure 2: Network of Communicating Finite Automata

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.







es ändern sich IMMER nur 2 Spalten

ändert nur 1 Spalte i** etc.

$\langle (l_0, m_0, m_0), \{x \to 0\} \rangle$ Regist?

$\downarrow r$

A: $\frac{i1\ m0\ *}{i1\ *\ m0} \Rightarrow \frac{i2\ m1\ *}{i2\ *\ m1}$

$\langle (l_1, m_0, m_0), \{x \to 27\} \rangle$ $\frac{i2\ m0\ *}{i2\ *\ m0}$ $\frac{i1\ m1\ *}{i1\ *\ m1}$

r: A $\frac{l0\ *\ *}{x==0} \Rightarrow \frac{i1\ **}{x=27}$

A

$\langle (l_2, m_1, m_0), x=27 \rangle$ $\langle (l_2, m_0, m_1), x=27 \rangle$

A A

$\langle (l_1, m_1, m_1), x=27 \rangle$

Exercise 3 – Basic Behavioural Model Analysis **(7/20 Points)**

The file `sensormaster.xml` includes a behavioural model of some aspects of the sensor/master system from Exercise Sheet 4. Note that the model in `sensormaster.xml` has similarities to the one shown in Figure 2 but is not identical in different aspects.

(i) Use the UPPAAL[1] simulator to create one computation path which shows that the model is

[1] See Appendix A for instructions.

able to exhibit the simple scenario that one sensor joins a master and then leaves again. (1)

Hint: Include a screenshot showing the final state of the computation path and the sequence diagram into your submitted document; and submit the `.xtr` *file of the simulation as well so that your tutor can reproduce your result.*

(ii) a) Recall the requirement that masters only need to monitor a bounded number of sensors. Use the verifier to check whether it is possible for some master to monitor (as indicated by the value of its variable `n`) the allowed maximum number of sensors (which is given by the global constant (or parameter) `max_sns`).

What is the result? How should this result be interpreted? (2)

Hint: That is, what can we conclude from the result of the above check about the modelled design idea?

b) Use the verifier to check whether it holds that for each master, the value of `n` never exceeds the assumed maximum number of sensors. Some queries have been proposed for the check and are included in `sensormaster.xml`.

What are the results? How should these results be interpreted? (2)

Hint: Also submit a copy of `sensormaster.xml` *with your 1.(ii).a) query filled in; make sure that your tutor understands which of the many files you submit is to be considered for which task.*

(iii) The author of the model has expressed doubts about the model's correctness. One invariant that needs to hold for the model to be correct (according to the requirements department) is that if the value of variable `m` in any sensor is a proper master's id, then the value of `n` of this master must not be 0.

Explain how the outcome of checking the corresponding query in `sensormaster.xml` (unfortunately) confirms these doubts. (1)

(iv) Explain the reason for the incorrectness of the model according to Task (iii) and suggest an as small-as-possible change that preserves the outcomes of the Tasks (i) and (ii), but changes the outcome of the previous check. (1)

Hint: Also submit the changed model in a separate file. If you do not spot any reasons for the incorrectness, ask your tutor for clues (together with a clear description of the state of your investigation and your hypotheses so far).

```
int convert(char[] str) throws Exception {
  if (str.length > 6)
    throw new Exception("Length exceeded");
  int number = 0;
  int digit;
  int i = 0;
  if (str.length > 0 && str[0] == '-')
    i = 1;
  while (i < str.length){
    digit = str[i] - '0';
    if (digit <= 0 || digit > 9)
      throw new Exception("Invalid character");
    number = number * 10 + digit;
    i = i + 1;
  }
  if (str.length > 0 && str[0] == '-')
    number =  -number;
  if (number > 32767 || number < -32768)
    throw new Exception("Range exceeded");
  return number;
}
```

In	Soll	i_2^t	s_3	i_2^f	s_4	i_7^t	s_8	i_7^f	i_9^t	s_{10}	i_{11}^t	s_{12}	i_{11}^f	s_{13}	i_9^f	i_{16}^t	s_{17}	i_{16}^f	i_{18}^t	s_{19}	i_{18}^f	s_{20}
"0000000"	E:kein	X		X																		
"-1.30"	E:Int			X	X	X	X		X	X	X	X	X	X								
"-44444"	E:Reg			X	X	X	X		X	X			X	X	X	X	X		X	X		
"27"	27			X	X			X	X	X			X	X	X			X			X	X

(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)

Exercise 2 – Exhaustive Testing (2/20 Points)

(i) What is the number of test cases required to *exhaustively* test `convert` from Exercise 1 for only the strings of length six? Assume that a character in a string has size 16-bit. (1)

Extra exercise without points: How many test cases are needed for all strings of length up to and including 7?

(ii) If we can conduct around 1000 tests per second (say, 1024, for simplicity), how many seconds (hours? days? years?) would it then at least take to exhaustively test the program on only the strings of length six? (1)

Extra exercise without points: How long for all strings of length up to and including 7?

(i) $\cancel{2^{16}}$ $(2^{16})^6 = 2^{6 \cdot 16} = 2^{96}$

(ii) $\frac{\quad}{1000 \cdot 60 \cdots \cdot 12}$

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`.

$A \cup B$

A=1 | A | B | T | T | T | T | = 25%

A=0 B=0 | A | B | T | T | T | T | = 75%

A=0 B=1 | = 100%

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)

```java
int multiply( int x, int y ) {
    int r = 0;
    int i = 0;
    while (i < x) {
        r = r + y;
        i = i + 1;
    }
    return r;
}
```

Figure 3: Function `multiply` in Java.

(iii) Apply the rules of the proof system PD to *derive a proof* that `multiply` is *partially correct*. Specify which rules or axioms you use at every proof step. (2)

(iv) Assume that you have implemented `multiply` in Java as shown in Figure 3 and you want to make sure that:

- Using your implementation outside its specification ($y \geq 0$ and $x \geq 0$) cannot go unnoticed.
- Changes to your implementation that do not respect your loop invariant do not go unnoticed.

How can Java Assertions[1] help towards these goals? Change the program accordingly, explain the change, and prepare a demonstration that the change is principally effective (against an unsupported use and against an undesired change). (2)

Hint: You can use, modify, compile, and run the file `Multiply.java` to demonstrate effectiveness; report your observations and submit all files that your tutor needs to reproduce your observations.

(i)

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

(ii)

$i \geq x \Rightarrow i \cdot y = r \wedge i \leq x \wedge \neg(i < x)$

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

$i \cdot y = r \wedge i < x$

$i \cdot y = r$

$\Leftrightarrow i \cdot y \cancel{+y} = r \cancel{+y}$

$\Leftrightarrow (i+1) \cdot y = r + y$

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

$r = r + y$

$(i+1) y = r \wedge (i+1) \leq x$

A2: $i < x$

$\Leftrightarrow i+1 \leq x$

$i = i + 1$

$i \cdot y = r \wedge i \leq x$ } A2 → R3

$y \geq 0 \wedge x \geq 0$

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

$r = 0$

$0 \cdot y = r \wedge 0 \leq x$ } A2

$i = 0$

$i \cdot y = r \wedge i \leq x$ } R3

(iv) Assume that you have implemented `multiply` in Java as shown in Figure 3 and you want to make sure that:

- Using your implementation outside its specification ($y \geq 0$ and $x \geq 0$) cannot go unnoticed.
- Changes to your implementation that do not respect your loop invariant do not go unnoticed.

How can Java Assertions[1] help towards these goals? Change the program accordingly, explain the change, and prepare a demonstration that the change is principally effective (against an unsupported use and against an undesired change). (2)

Hint: You can use, modify, compile, and run the file `Multiply.java` to demonstrate effectiveness; report your observations and submit all files that your tutor needs to reproduce your observations.

(iv)

```
int multiply( int x, int y ) {
    int r = 0;
    int i = 0;
    while (i < x) {
        r = r + y;
        i = i + 1;
    }
    return r;
}
```

assert $y \geq 0 \wedge x \geq 0$

assert $i \cdot y = r$

assert $x \cdot y = r$

Figure 3: Function `multiply` in Java.

Exercise 4 – Verification with VCC (10 Bonus)

(i) We implemented the program `multiply` from Exercise 3 as a C function (see file `multiply.c`). Assume that `multiply` is used in a larger program where it is only called with values for y and x between 0 and 15, i.e., in the considered larger program, all callers guarantee the pre-condition $\{0 \leq x \leq 15 \wedge 0 \leq y \leq 15\}$.

Annotate the function with pre- and post-condition, and a loop invariant (and whatever else you consider necessary) using the VCC syntax for annotations.

Use the VCC tool[2] to see whether it is able to *verify* that the annotated function is correct wrt. pre- and post-condition. Which outcome did you expect (assuming that Exercise 3.(iii) has a solution)? (4 Bonus)

(ii) For the values $N = 15, 150, 1500, 15000$,

a) how many test cases are needed to exhaustively test `multiply` wrt. the pre-condition $\{0 \leq y \leq N \wedge 0 \leq x \leq N\}$? (1 Bonus)

Hint: We need a term to compute the number of test cases for any given N, and (at least the orders of magnitude of the) particular numbers for the values of N given above.

b) Use VCC to verify the function with the new pre-conditions for the four values of N from above and provide the reported verification time.

Is this measurement plausible? Why (not)? (2 Bonus)

(iii) Assume that we were unsure about our proof from Exercise 3.(iii) and would like to confirm the result using VCC.

Modify the C function such that it in particular considers the original pre-condition (cf. Figure 2), and try to verify it using VCC. Describe what you expect to be the outcome of this experiment and what the results of the verification attempt with VCC were. Interpret the output of VCC. (2 Bonus)

(iv) We have observed that with testing, it is easy to provide an *unsuccessful* test suite for a program which is *not correct* wrt. its specification. How is it with VCC? (1 Bonus)

Hint: As an experiment, change the C function such that the function is not *correct wrt. the specification anymore. Describe your modification (in how far does it cause a violation of the specification?) and the behaviour of VCC when applied to the modified function.*

To enable your tutor to reproduce your results, all code artefacts of this exercise (with appropriate comments or explanations, if necessary) should be submitted as separate text files, i.e., not embedded in the PDF. Make sure that it is adequately easy for your tutor to know which artefact to consider for which tasks and give clear instructions on how to reproduce.

(i)

Exercise 2 – Cyclomatic Complexity (5/10 Points)

Consider the *cyclomatic complexity* or *McCabe* metric.

(i) Construct the control flow graph (CFG) for the method `quickSort()` of the class `MyQuickSort` in the file accompanying this exercise sheet and calculate the value of its cyclomatic complexity as shown in the example of Fig. 1. (4)

(ii) In the example, we introduced additional auxiliary nodes to the control flow graph that serve as junction points for control paths; see, e.g., the circular node below node number 5. Another possibility of constructing the CFG would be to directly connect the nodes representing program locations. In our example, there would be a direct edge from node 5 to node 6 and from node 8 to node 6.

Does this choice of CFG construction alter the value of the cyclomatic complexity metric? Justify your answer. (1)

Program:

```
1  void insertionSort(int[] array) {
2    for (int i = 2; i < array.length; i++) {
3      tmp = array[i];
4      array[0] = tmp;
5      int j = i;
6      while (j > 0 && tmp < array[j-1]) {
7        array[j] = array[j-1];
8        j--;
9      }
10     array[j] = tmp;
11   }
12 }
```

The cyclomatic complexity is defined for graph G corresponding to the program as

$$v(G) = e - n + p.$$

Number of edges: $e = 11$
Number of nodes: $n = 6 + 2 + 2 = 10$ (nodes are marked with the corresponding line numbers)
External connections: $p = 2$

$v(G) = 11 - 10 + 2 = 3$

Corresponding graph G

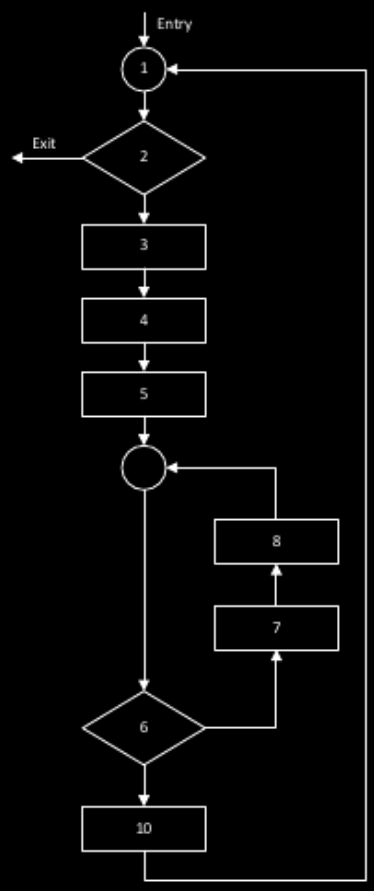


Figure 1: Example of the calculation of cyclomatic complexity

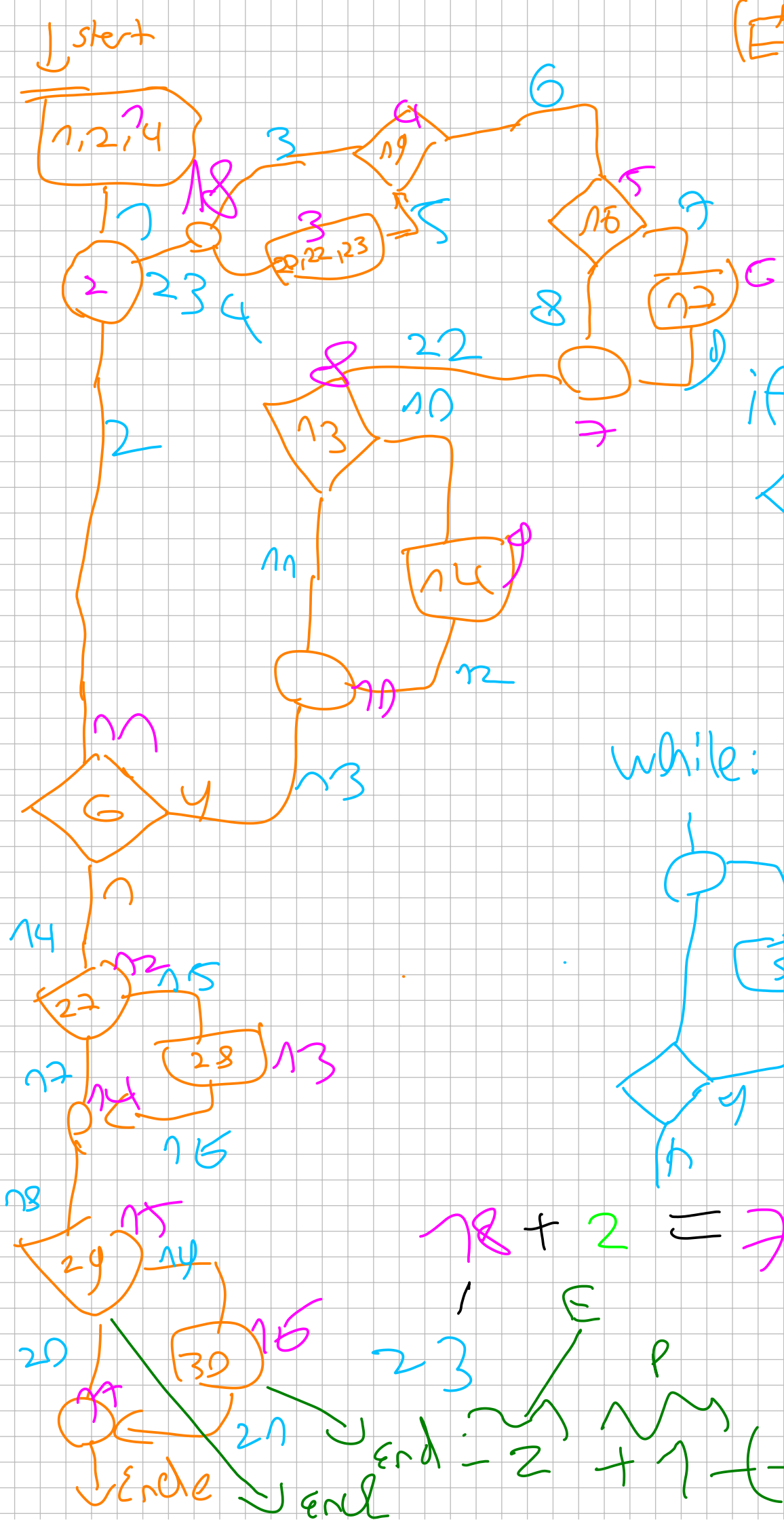
Nein denn

$v(G) = |E| - |V| + p$

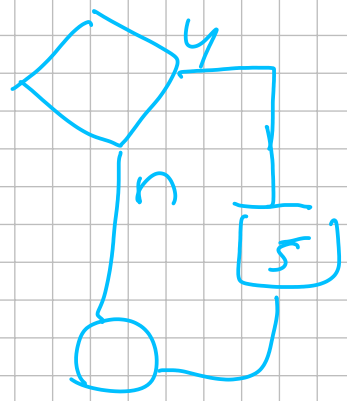
$= (|E'| - 1) - (-1 + |V'|) + p$

$= |E'| - 1 + 1 + |V'| + p =$

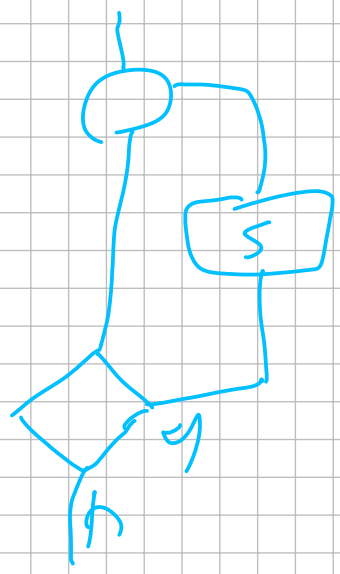
$|E| - |V| + P$



if:



while:

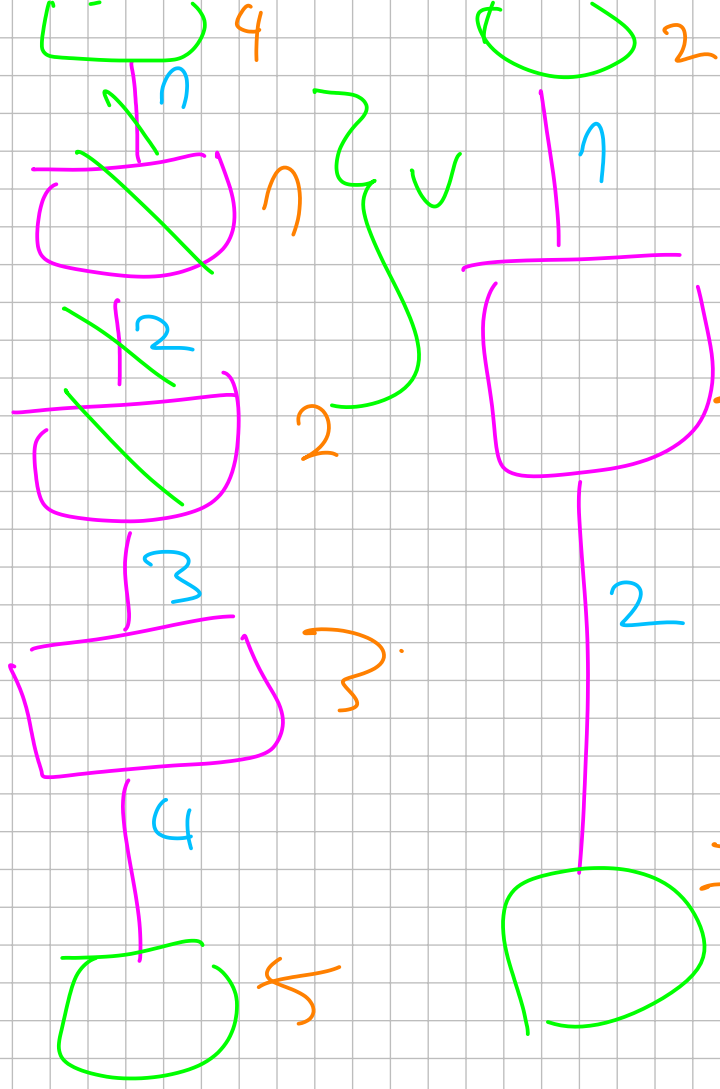


$18 + 2 = 7$

$1 \quad E \quad V$

$23 \quad P$

$end - 2 + 1 - (-1) = 0$



$$|E| - |V| + p = |E'| - |V'| + p$$

$$|E| - |V| + p$$

$$= (|E'| - v) - (|V'| - v) + p$$

$$= |E'| + |V'| + p$$

$4 - 5 = 1$

$2 - 3 = 1$

Exercise 1 – Cost Estimation **(7/20 Points)**

This exercise is about estimating the effort for the development of a video game as specified by the requirements given in Appendix A.

In the following, assume that the development is conducted by a team of 6–7 people, all of them computer science students with typical 3rd-semester programming knowledge. The project is part of the students' study plan, and each team is supported by a tutor. The final result (game and source code) is assessed in an expert review, but there are no plans to distribute the created software to users, i.e., there is no need for later maintenance releases.

(i) If each team member contributes ca. 180 hours of work in total, what is the overall development effort in *person-months* (PM)? (1)

Hint: Submission are only comprehensible if the used definition of person-month is provided.

(ii) In the tutorial on this exercise sheet, we want to apply the Delphi method to estimate the size of the video game code given the requirements.

To prepare the tutorial, state your team's estimation of delivered $KLOC_{pars}$. (1)

Hint: Recall that an essential ingredient of a Delphi estimation is a thorough explanation of the reasons for the stated estimation.

$$\frac{180 \cdot 6}{152} = 7$$

(iii) COCOMO, intermediate level, considers effort coefficients based on so-called *cost drivers* (cf. Table 2 on page 6; a short explanation of the cost drivers can be found in Appendix B).

Determine ratings for all cost drivers for the game project.

Explain your rating for the cost driver 'Required Development Schedule' and for two other cost drivers: One where the factor determined by your rating is closest to the lowest factor and one where the factor determined by your rating is closest to the highest factor. (3)

(iv) Categorize the considered video game according to the COCOMO 81 software project types as indicated in Table 1 and briefly explain how you chose the project type.

Then calculate the estimated effort E in person-months by using the COCOMO formula

$$E = a \cdot (KLOC)^b \cdot EAF$$

where a and b are the coefficients specified by the project type in Table 1, $KLOC$ is your estimation from Task (ii), and EAF is the *effort adjustment factor*, calculated as the multiplication of the factors for all cost drivers (see Table 2) according to your categorization.

How big is the difference between the planned effort according to Task (i) and the result of the COCOMO calculation? What are possible reasons for the deviation? (2)

Exercise 2 – Process Model Application (8/20 Points + 5 Bonus)

Assume you are the manager of a company that develops radio communication technology for fire alarms. Your company is hired by an external client to develop the software of a new radio module for fire detectors. Once developed, the module needs to be certified to comply with its corresponding industry standard, and you are also required to provide technical documentation.

(i) Consider the activities in Figure 1 (together with their respective artefacts and responsibilities). Those are the established activities in your company.

Provide a *graphical process model* for delivering a finished radio module by using those building blocks. (5)

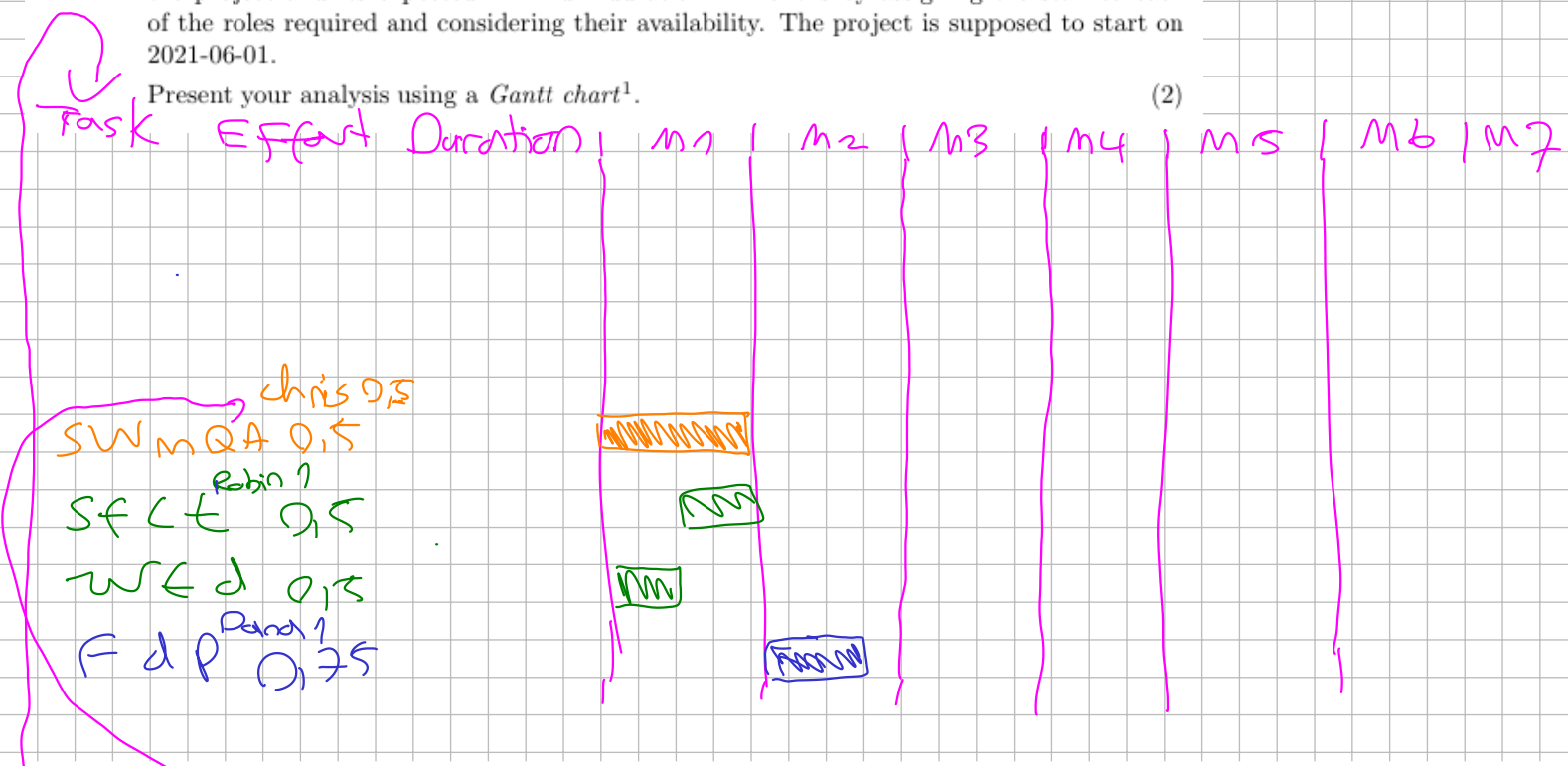
(ii) Your company has the following staff available, with their corresponding qualifications for roles:

- Alex: Requirements analyst, Test manager
- Dana: Product manager
- Robin: Certification manager, Technical writer
- Chris: Software developer, Software tester, System tester

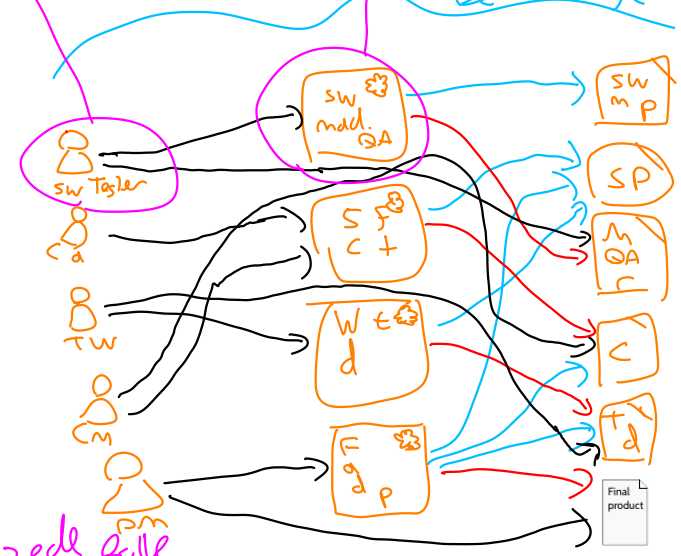
All of them have full-time availability (1 PM per calender month), except for Robin, who works only part-time (0.5 PM per month).

Consider the effort estimates indicated in Figure 1 and analyze the expected total effort of the project and its expected minimum duration in months by assigning the staff to each of the roles required and considering their availability. The project is supposed to start on 2021-06-01.

Present your analysis using a *Gantt chart*[1]. (2)

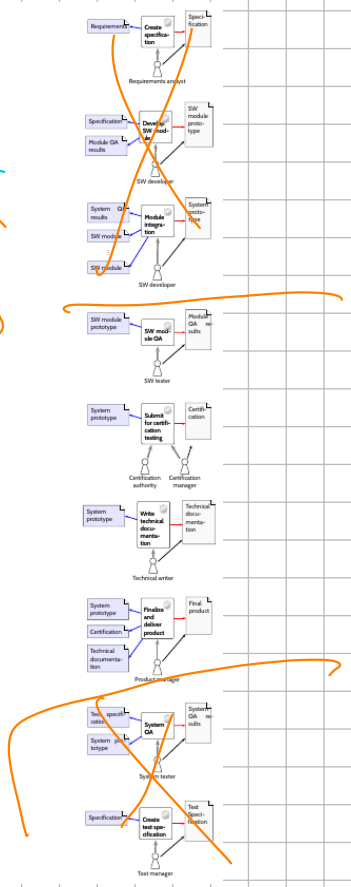


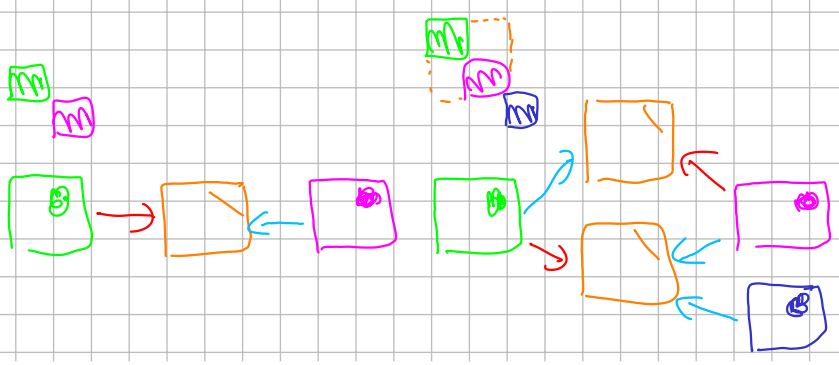
Exercise 2.(i) – Process Model



jede Rolle nur für eine Person/Id oder aus einem Pool wählt
=> mehrfache Rollen

mehrfache Aktivity
=> mehrfach reinmalen, da man verschiedene Inputs und Outputs nicht mischen soll



Task	Effort	Duration	M1	M2	M3	M4	M5	M6	M7
Cr SP	Alex 1	0,5							
Cr Te		0,25							
Dev.	chris 1	1,5							
Rad. Mod		0,5							
Mod Int.		0,5							
SW m QA		0,5							
S f C t	Robin 0,5	0,5							
W t d		0,5							
F d p	Pascal 1	0,75							

2,25 PM