Prof. Dr. Burkhard Stiller, Universität Zürich, Binzmühlestrasse 14, CH-8050 Zürich
Telefon: +41 44 635 6710, Fax: +41 44 635 6809, stiller@ifi.uzh.ch
Daniel Dönni, Telefon: +41 44 635 4375, doenni@ifi.uzh.ch
Dr. Corinna Schmitt, Telefon: +41 44 635 7585, schmitt@ifi.uzh.ch

Übungen zu Informatik 1

Technische Grundlagen der Informatik - Übung 9

Ausgabedatum: 11. November 2013
Besprechung: Übungsstunden in der Woche 47 (18.11. - 22.11.2013)

1) Normal- und Minimalformen

1.1) Definieren Sie die Begriffe Implikant, Minterm, Implikat sowie Maxterm.

Implikant:
Minterm:
Implikat:

Maxterm:

2) Schaltungen und Schaltnetze

2.1) Zeichnen Sie die Schaltsymbole für die unten angegebenen Verknüpfungen (IEEE Standard 91-1984).

AND	OR	NOT	NAND	NOR	XOR

2.2) Ergänzen Sie unten stehende Schaltungen so, dass eine UND- und eine ODER-Schaltung entsteht.

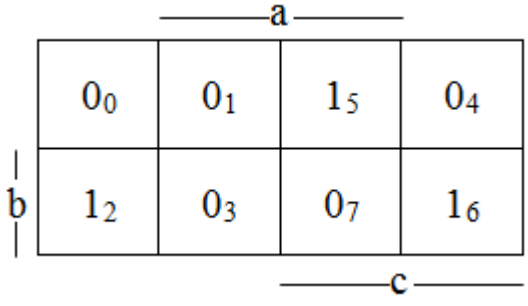
UND-Schaltung

ODER-Schaltung

2.3) Zeichnen Sie ein Schaltnetz für den Boolschen Ausdruck $y = (\bar{a} \vee b) \wedge (\overline{\bar{a} \wedge c})$ ohne den Ausdruck zu verkürzen bzw. umformen. Benutzen Sie die Gatter aus Aufgabe 1.1.

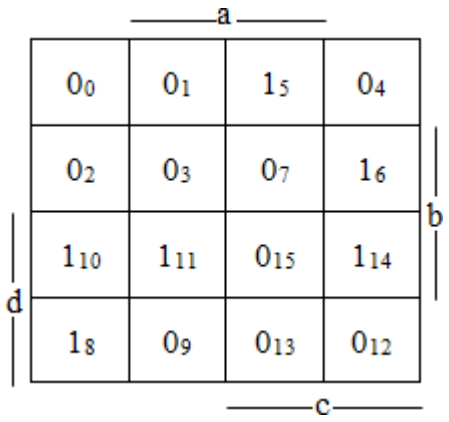
3) KV-Diagramme

3.1) Geben ist folgendes KV-Diagramm. Kreuzen Sie jeweils die richtige Antwort zu folgenden Aussagen an und begründen Sie ihre Antwort.



	Aussage	Richtig	Falsch
a)	Der Maxterm bei Index 4 entspricht dem Ausdruck $\bar{a} \vee b \vee \bar{c}$.		
b)	Benachbarte Felder eines KV-Diagramms unterscheiden sich nur in einer Eingangsvariablen.		
c)	Das KV-Diagramm enthält vier Primimplikate.		
d)	Das KV-Diagramm beschreibt nur eine konjunktive Minimalform, die dem Ausdruck $(\bar{a} \vee \bar{b}) \wedge (a \vee b) \wedge (b \vee c)$ entspricht.		

3.2) Lesen Sie alle Minterme aus dem untenstehendem KV-Diagramm heraus. Bestimmen Sie anschliessend alle Primimplikanten sowie die disjunktive Minimalform bestehend aus möglichst wenigen Primimplikanten. (Hinweis: Auf der Webseite http://ti.itec.uka.de/KVD/index.html finden Sie ein Applet, mit welchem Sie den Umgang mit KV-Diagrammen und die Berechnung der versch. Minimalformen üben können.)



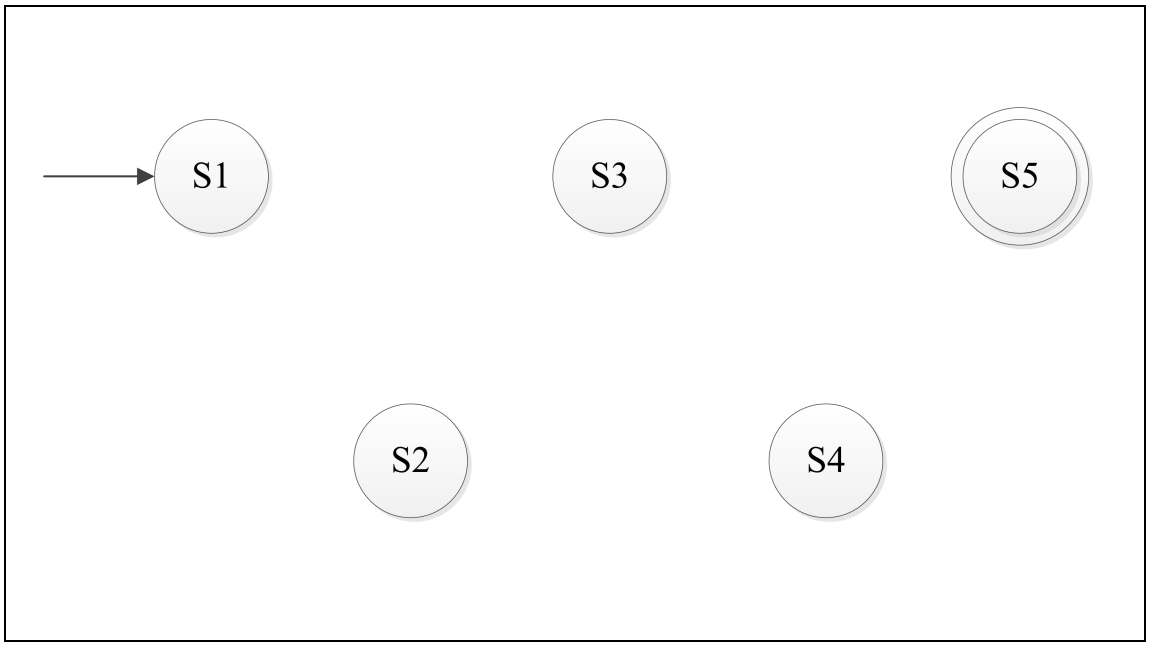
4) Automatentheorie

4.1) Es seien die Ausgabefunktionen a = ω(z) und a = ω(z,e) gegeben. Erklären Sie anhand dieser Funktionen in *maximal zwei* Sätzen was der Unterschied zwischen einem Mealy- und einem Moore-Automaten ist.

4.2) Modellieren Sie einen Mealy-Automaten mit dem folgenden Verhalten:

- Als Eingabe kann der Automat entweder “a” oder “b” verarbeiten.
- Als Ausgabe kann der Automat entweder “0” oder “1” produzieren.
- Bevor die Sequenz “abba” eingegeben ist, gibt der Automat immer “0” aus.
- Nachdem die Sequenz “abba” eingegeben ist, gibt der Automat immer “1” aus.

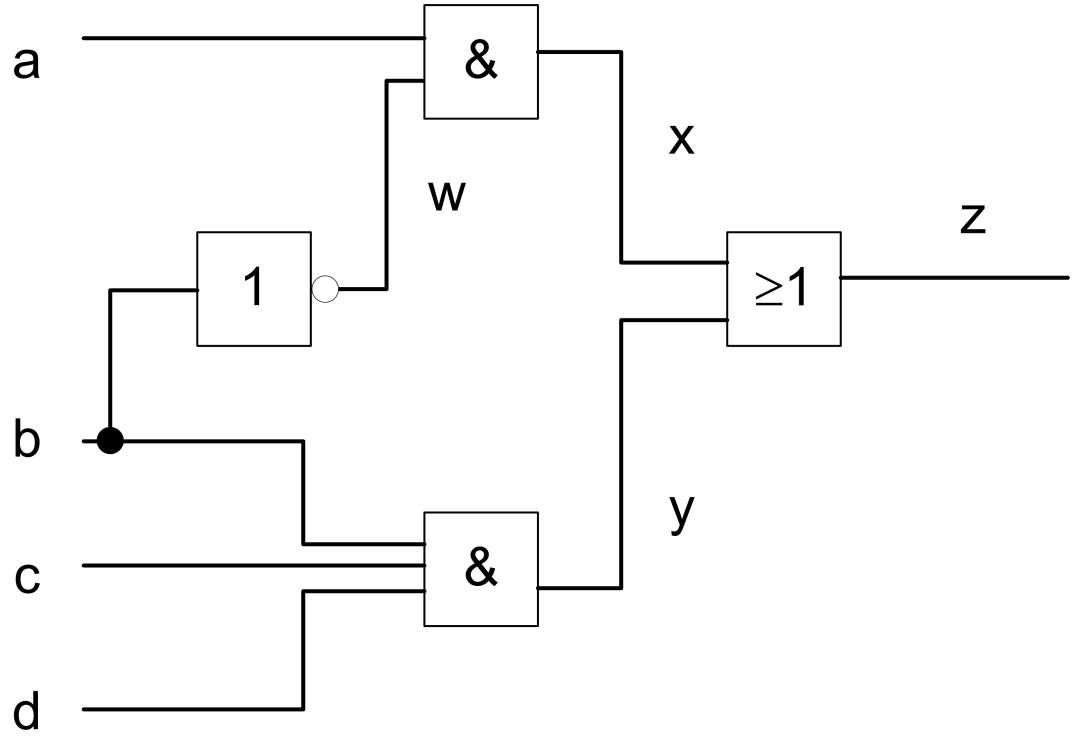
Vervollständigen Sie das nachfolgende Diagramm eines Mealy-Automaten mit dem Startzustand “S1”, dem Endzustand “S5” und den weiteren Zuständen “S2”, “S3” und “S4” so, dass der Automat das beschriebene Verhalten aufweist.



Platz für die Berechnung:

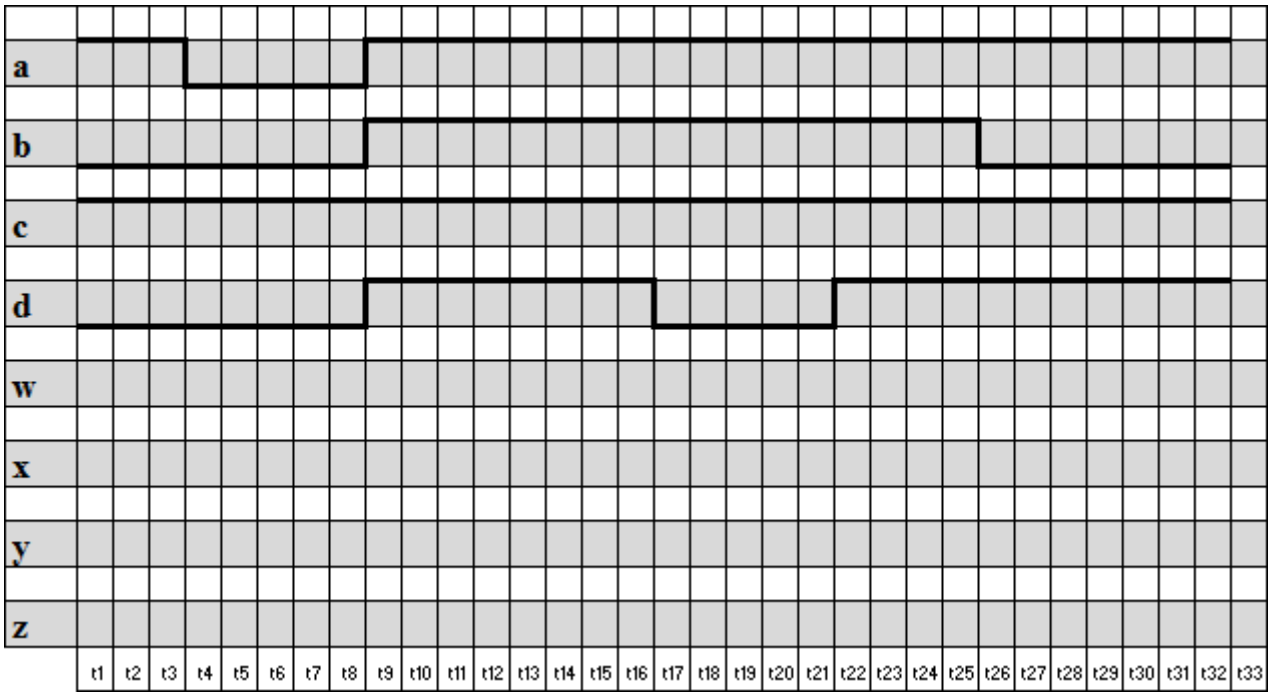
5) Totzeitmodell / Hasard

Die folgende Zeichnung stellt eine elektrische Schaltung dar. Jedes Gatter hat eine Verzögerung von t.



5.1) Wozu dient das Totzeitmodell?

5.2) Vervollständigen Sie das folgende Zeitdiagramm entsprechend zur obigen Schaltung:



5.3) Fällt Ihnen im Zeitdiagramm bezüglich dem Stichwort Hasard etwas auf? Kommentieren Sie Ihren Befund in *maximal einem Satz*.

5.4) Erklären Sie was ein Hasardfehler ist in *maximal drei Sätzen*.

5.5) Zeichnen Sie je ein Beispiel eines dynamischen und eines statischen Hasards.

Dynamischer Hasard:	Statischer Hasard:

6) Automatentheorie / Programmierung in Java

Implementieren Sie den Mealy-Automaten aus Aufgabe 4.2 in Java. Gehen Sie dabei wie folgt vor:

1. Schreiben Sie eine Klasse `AbstractState`, welche eine abstrakte Methode `nextState(char c)` besitzt. Implementieren Sie anschliessend für jeden State des Automaten eine Subklasse von `AbstractState`, die beim Aufruf von `nextState(char c)` den dem Input entsprechenden neuen State zurück gibt. Verwenden Sie innerhalb der Methode zur Abwechslung anstatt eines `if-else` Statements ein `switch-case` Statement, um die möglichen Input-Characters zu unterscheiden. Eine Beschreibung von `switch-case` finden Sie in Savitch Kapitel 3.3. Für einen ungültigen Input soll einfach der existierende State zurückgegeben werden (verwenden Sie hierfür das `default` Keyword im `switch-case` Kontrollfluss). Jeder State sollte ausserdem eine Methode haben, die den Output des States ausgibt.
2. Schreiben Sie eine Kommandozeilenapplikation, welche die Eingaben des Benutzers verarbeitet. Es soll möglich sein, Sequenzen an Buchstaben einzugeben, um dann zu sehen, welche States dabei durchlaufen werden. Die Eingabe einer leeren Sequenz soll das Programm beenden. Hier finden Sie eine Beispiel, wie es aussehen könnte, wenn man die Applikation verwendet (wobei die State Nummern hier versteckt wurden und durch X ersetzt wurden):

```
$ java MealyMachine
StateX: 0
ab
StateX: 0
StateX: 0
abba
StateX: 0
StateX: 0
StateX: 0
StateX: 1
bab
StateX: 1
StateX: 1
StateX: 1
```