Kurs

Datenbankgrundlagen und Modellierung

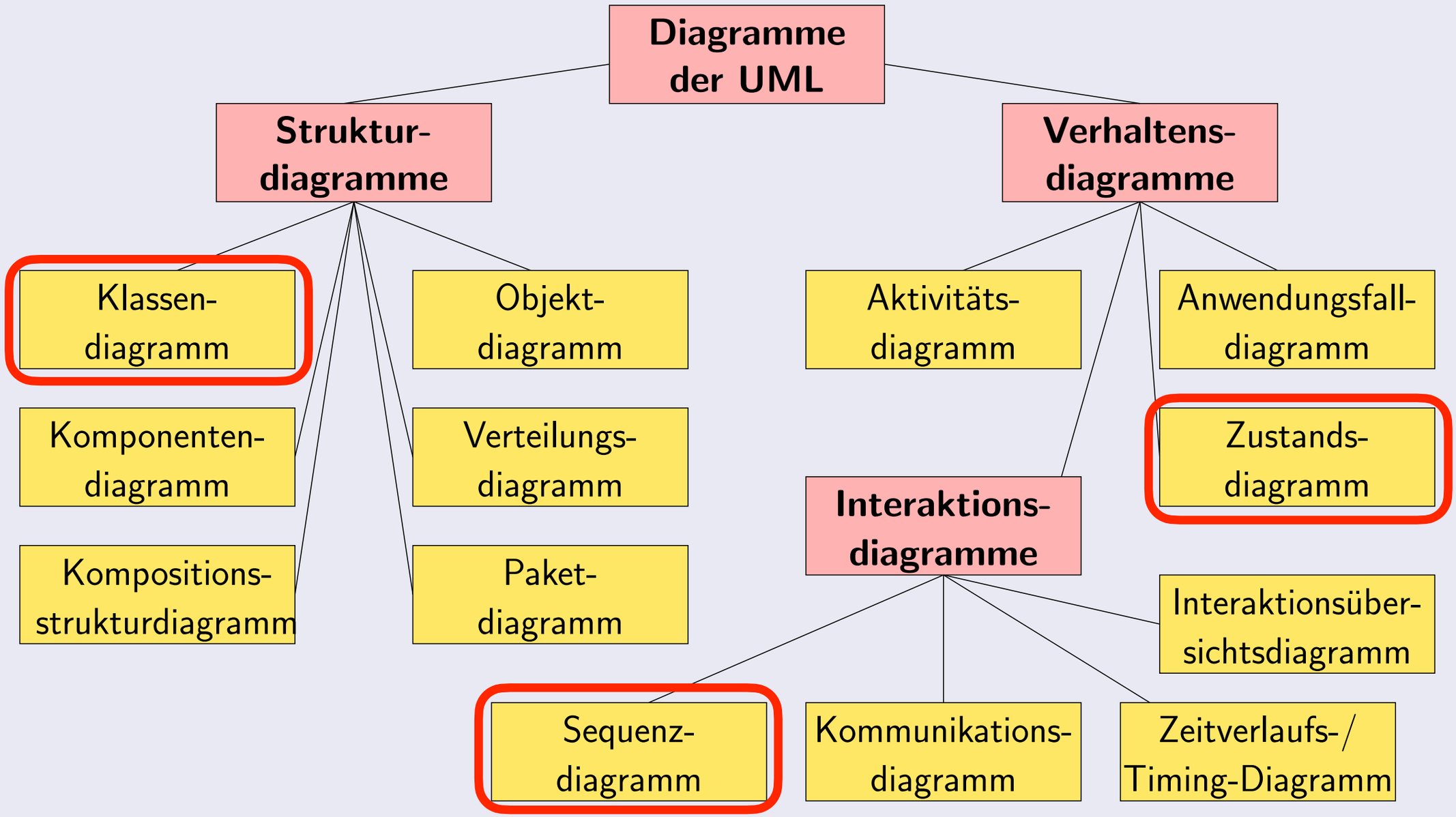
Sebastian Maneth, Universität Bremen
maneth@uni-bremen.de

Sommersemester 2023

19.6.2023
Vorlesung 8: Zustandsdiagramme
= State Charts

UML: Einführung

UML-Diagramme

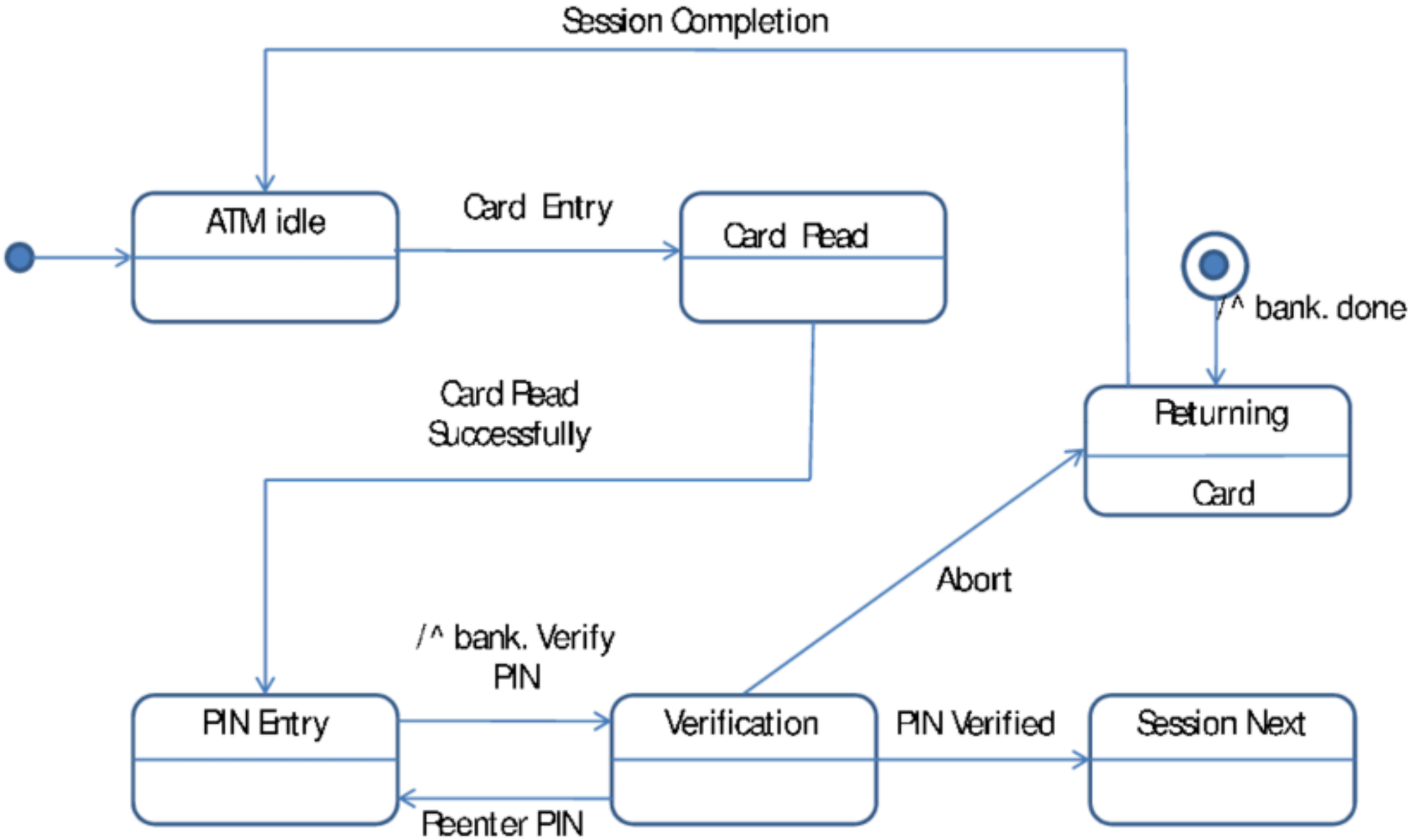


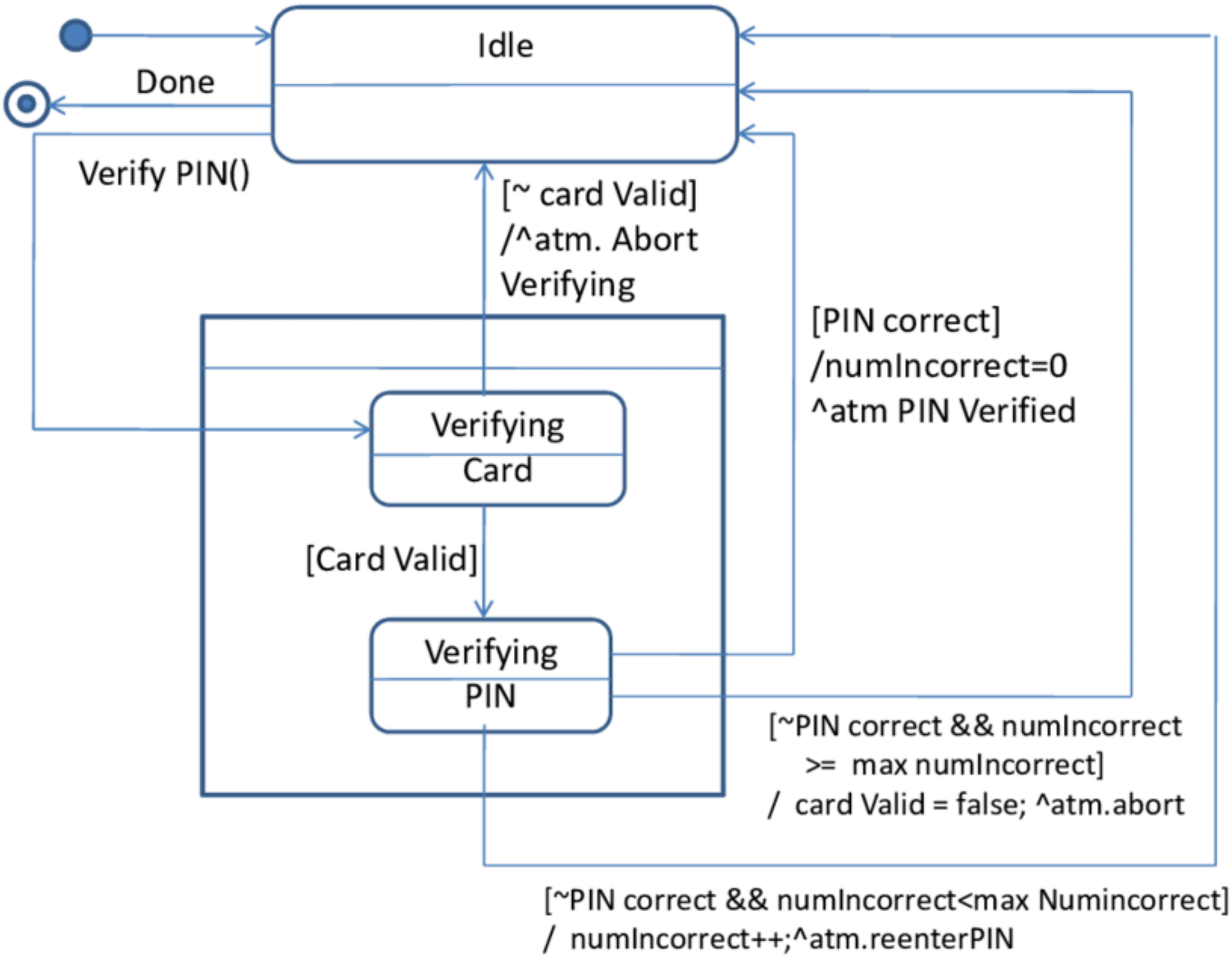
Zustandsdiagramme

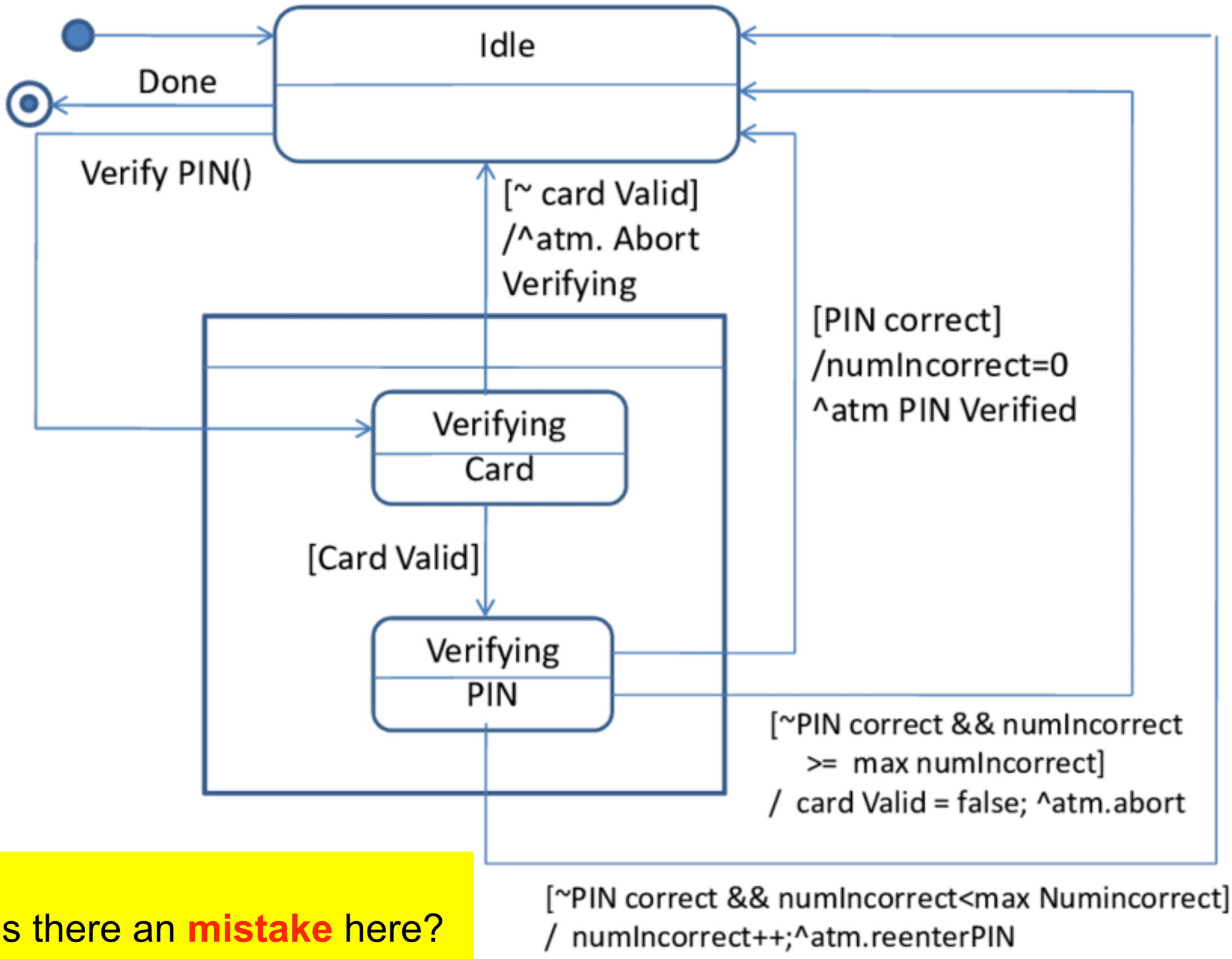
Zustandsdiagramme

Anwendungen sind die Modellierung von:

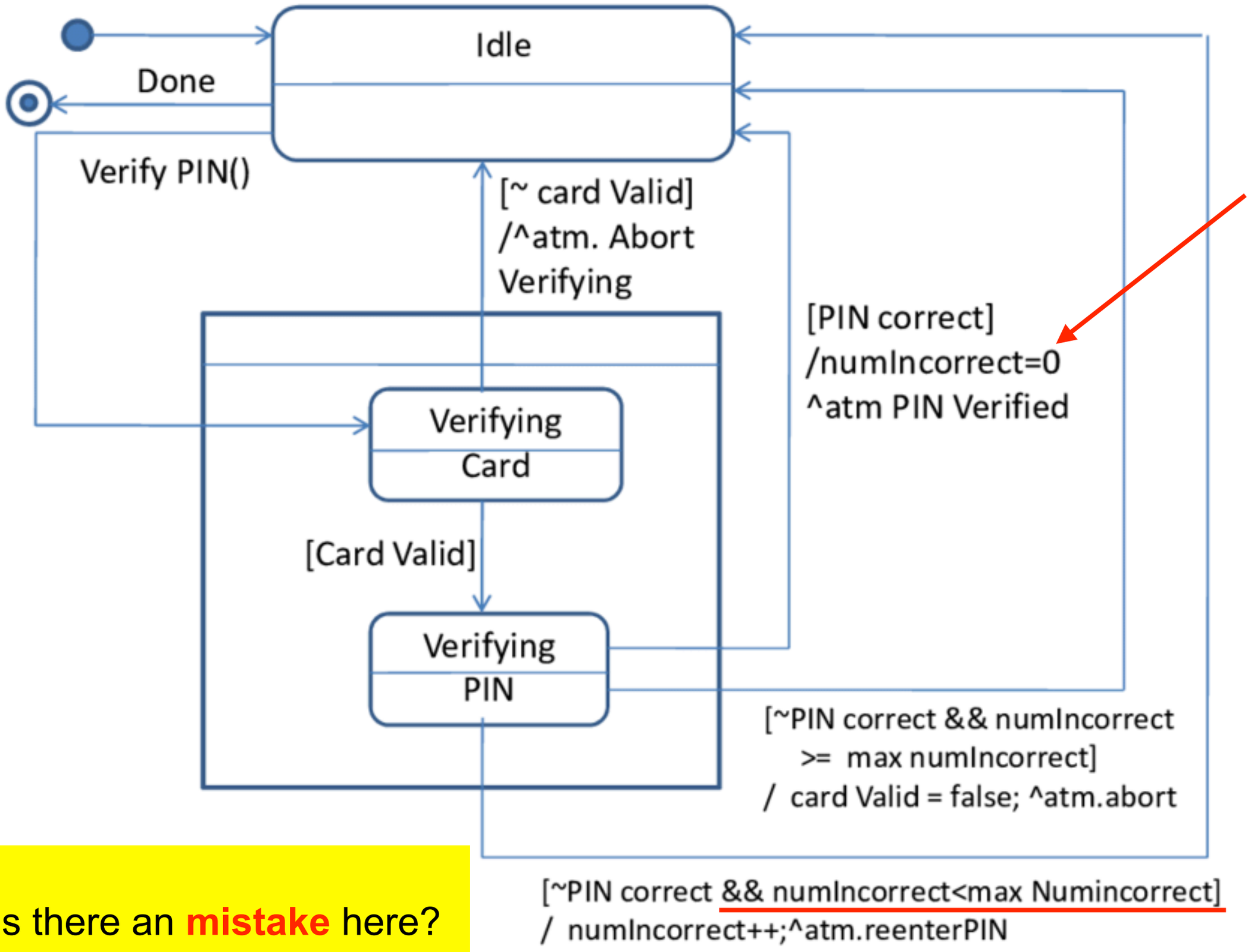
- Protokolle, Komponenten verteilter Systemen
- Benutzeroberflächen
- Eingebettete Systemen
- ...


Session Completion
ATM idle
Card Entry
Card Read
/^ bank. done
Returning
Card
Card Read
Successfully
Abort
/^ bank. Verify
PIN
PIN Entry
Verification
PIN Verified
Session Next
Reenter PIN

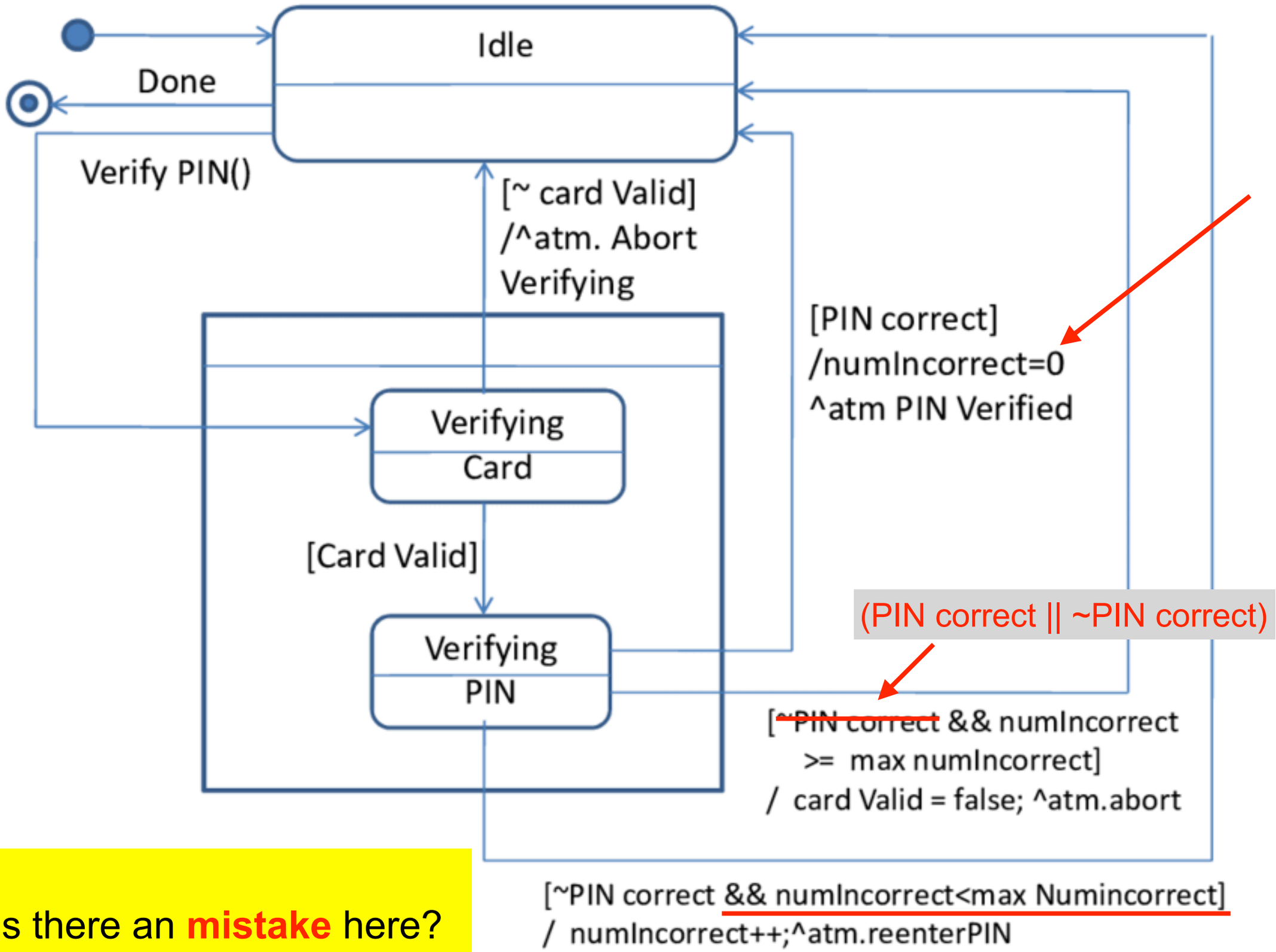

Idle
Done
Verify PIN()
[~ card Valid]
/^atm. Abort
Verifying
[PIN correct]
/numIncorrect=0
^atm PIN Verified
Verifying
Card
[Card Valid]
Verifying
PIN
[~PIN correct && numIncorrect
>= max numIncorrect]
/ card Valid = false; ^atm.abort
[~PIN correct && numIncorrect<max Numincorrect]
/ numIncorrect++;^atm.reenterPIN



Is there an **mistake** here?



Is there an **mistake** here?



Is there an **mistake** here?

Zustandsdiagramme

Eigenschaften von Zustandsdiagrammen:

- Zustände und Zustandsübergänge (Transitionen)
- Hierarchisch aufgebaute Zustände
- "Parallelschalten" von Zustandsdiagrammen durch Regionen
- Historien, um sich früher besuchte Zustände zu merken und in diese zurückzukehren

Viele dieser Eigenschaften dienen dazu, unübersichtliche Zustandsdiagramme mit vielen Zuständen und Übergängen übersichtlicher und kompakter zu gestalten.

Zustandsdiagramme

Eigenschaften von Zustandsdiagrammen:

- Zustände und Zustandsübergänge (Transitionen)
- Hierarchisch aufgebaute Zustände
- "Parallelschalten" von Zustandsdiagrammen durch Regionen
- Historien, um sich früher besuchte Zustände zu merken und in diese zurückzukehren

Viele dieser Eigenschaften dienen dazu, unübersichtliche Zustandsdiagramme mit vielen Zuständen und Übergängen übersichtlicher und kompakter zu gestalten.

Zustandsdiagramme wurden 1987 von David Harel unter dem Namen Statecharts eingeführt. Harel modellierte damit vollständig seine Armbanduhr.

Science of Computer Programming 8 (1987) 231-274
North-Holland

STATECHARTS: A VISUAL FORMALISM FOR COMPLEX SYSTEMS*

David HAREL

Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel

Communicated by A. Pnueli
Received December 1984
Revised July 1986

Abstract. We present a broad extension of the conventional formalism of state machines and state diagrams, that is relevant to the specification and design of complex discrete-event systems, such as multi-computer real-time systems, communication protocols and digital control units. Our diagrams, which we call *statecharts*, extend conventional state-transition diagrams with essentially three elements, dealing, respectively, with the notions of hierarchy, concurrency and communication. These transform the language of state diagrams into a highly structured and economical description language. Statecharts are thus compact and expressive—small diagrams can express complex behavior—as well as compositional and modular. When coupled with the capabilities of computerized graphics, statecharts enable viewing the description at different levels of detail, and make even very large specifications manageable and comprehensible. In fact, we intend to demonstrate here that statecharts counter many of the objections raised against conventional state diagrams, and thus appear to render specification by diagrams an attractive and plausible approach. Statecharts can be used either as a stand-alone behavioral description or as part of a more general design methodology that deals also with the system's other aspects, such as functional decomposition and data-flow specification. We also discuss some practical experience that was gained over the last three years in applying the statechart formalism to the specification of a particularly complex system.

1. Introduction

The literature on software and systems engineering is almost unanimous in recognizing the existence of a major problem in the specification and design of large and complex *reactive systems*. A reactive system (see [14]), in contrast with a *transformational system*, is characterized by being, to a large extent, event-driven, continuously having to react to external and internal stimuli. Examples include telephones, automobiles, communication networks, computer operating systems, missile and avionics systems, and the man-machine interface of many kinds of ordinary software. The problem is rooted in the difficulty of describing reactive behavior in ways that are clear and realistic, and at the same time formal and

* The initial part of this research was carried out while the author was consulting for the Research and Development Division of the Israel Aircraft Industries (IAI), Lod, Israel. Later stages were supported in part by grants from IAI and AD CAD, Ltd.

says that D is default among D and B, and A is the default among A and C. Of course, for zooming in and out the latter has obvious advantages. Default arrows are thus analogous to the start states of finite-state automata.

Let us now introduce our running example. The Citizen Quartz Multi-Alarm III watch has a main display area and four smaller ones, a two-tone beeper, and four control buttons denoted here *a*, *b*, *c* and *d*. See Fig. 7. It can display the time (with

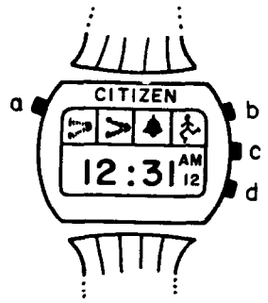


Fig. 7.

am/pm or 24 hour time modes) or the date (day of month, month, day of week), it has a chime (beeps on the hour if enabled), two independent alarms, a stopwatch (with lap and regular display modes, and a 1/100 s display), a light for illumination, a weak battery blinking indication, and a beeper test. We shall assume throughout that the main functions of these are known, and will use liberal terminology, such as 'power weakens' to denote certain events of obvious meaning, though, of course, to make things complete one would have to tie these events up with actual happenings in the physical parts of the system, or to specify them as output events produced in other, separately specified, components.

The main external events will be the depressing and releasing of buttons (e.g., event "a" denotes button a being depressed, and "$\hat{a}$" denotes it being released), and there will be certain internal ones too. The distinction is sharpened in Section 5. We remark here that while the description of the watch presented herein is intended to be as faithful to its actual workings as possible, there are some very minor differences that are not worth dwelling upon here, and that most likely will not be detected in normal use of the watch. The point is, however, that the statechart of the watch (cf. Fig. 31) was obtained by the author using the obviously inappropriate method of observation from the final product; had it been the basis for the initial specification and design of that final product, in the spirit of the gradual development presented below, the undescribed anomalies might have been avoided.

Figure 8 shows the transitions between the normal *displays* mode and the various beeping states. Here $T1$ and $T2$ stand for the respective internal time settings of the alarms, and T for the current time. Also, $P1$ abbreviates "*alarm*1 enabled $\wedge$ (*alarm*2 disabled $\vee$ $T1 \neq T2$)", and similarly for $P2$, while P abbreviates "*alarm*1

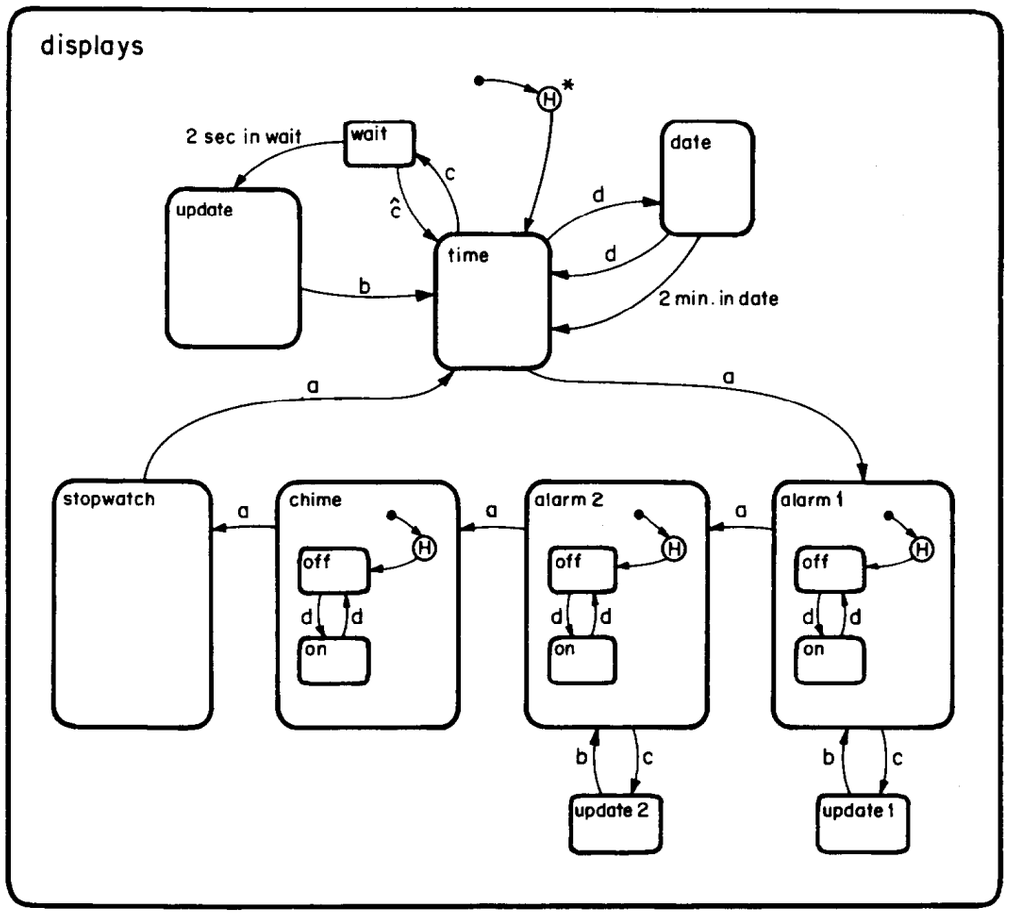

displays
2 sec in wait
wait
update
c
ĉ
time
date
d
d
2 min. in date
b
a
a
stopwatch
a
chime
a
alarm 2
a
alarm 1
off
d
d
on
b
c
update 2
update 1

Fig. 13.

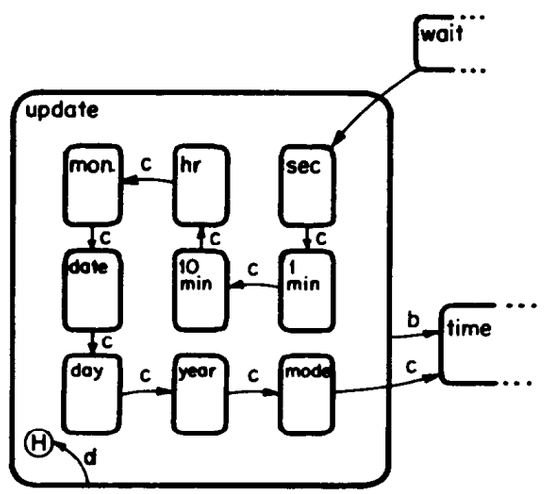

wait
update
mon.
c
hr
sec
c
c
c
date
10 min
c
1 min
c
b
time
c
day
c
year
c
mode
d

Fig. 14.

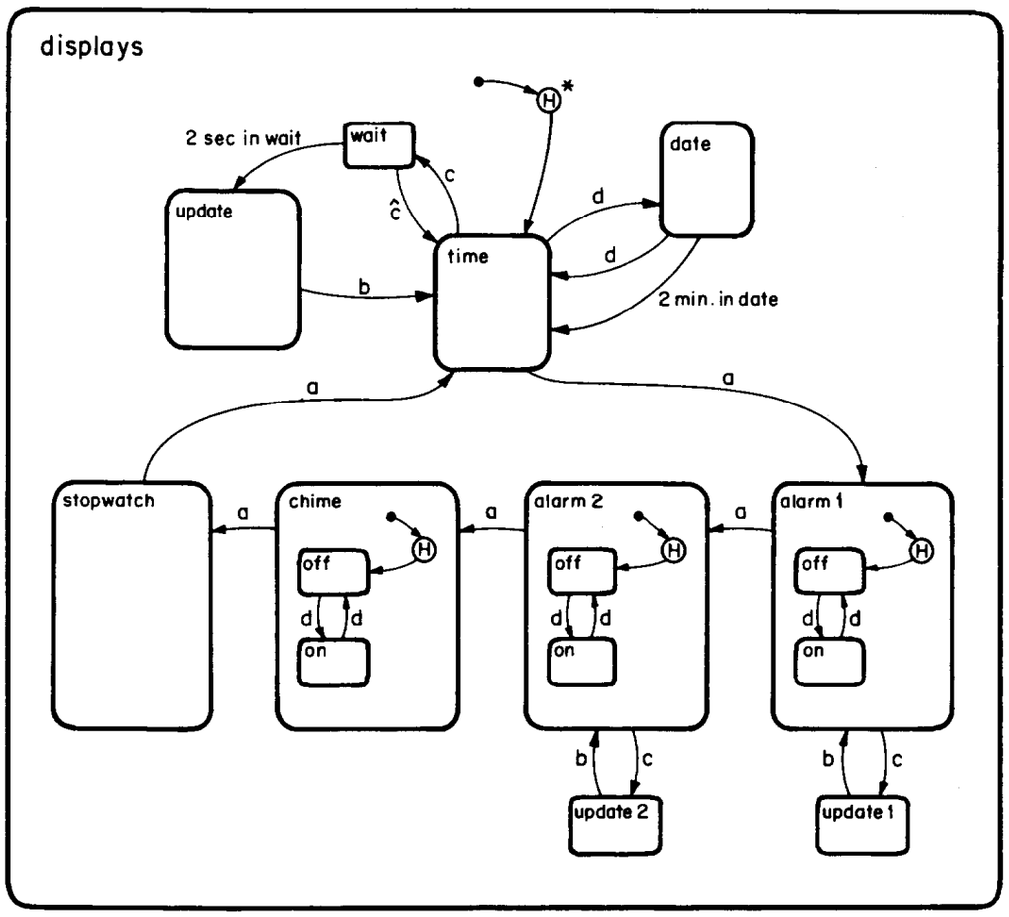


Fig. 13.

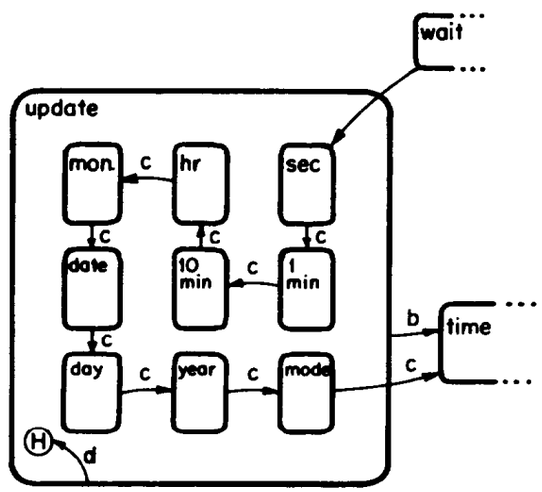


Fig. 14.

chime	*- bell*	die Glocke *Pl.:* die Glocken
chime	*- of a bell*	der Glockenton *Pl.:* die Glockentöne
chime	*- of a bell*	der Glockenschlag *Pl.:* die Glockenschläge

David Harel

Professor of Computer Science, The Weizmann Institute
Verified email at weizmann.ac.il - Homepage

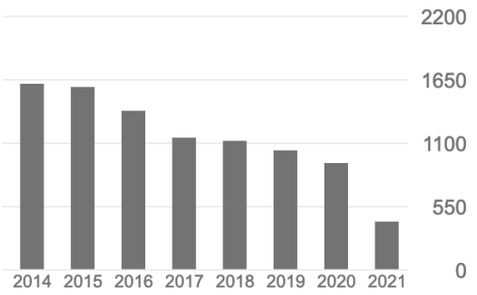
computer science systems biology

FOLLOW

TITLE	CITED BY	YEAR
Statecharts: A visual formalism for complex systems D Harel Science of computer programming 8 (3), 231-274	10651	1987
On visual formalisms D Harel Communications of the ACM 31 (5), 514-530	1975	1988
Statemate: A working environment for the development of complex reactive systems D Harel, H Lachover, A Naamad, A Pnueli, M Politi, R Sherman, ... IEEE Transactions on software engineering 16 (4), 403-414	1903	1990
Dynamic logic D Harel, D Kozen, J Tiuryn Handbook of philosophical logic, 99-217	1803	2001
The STATEMATE semantics of statecharts D Harel, A Naamad ACM Transactions on Software Engineering and Methodology (TOSEM) 5 (4), 293-333	1748	1996
On the development of reactive systems D Harel, A Pnueli Logics and models of concurrent systems, 477-498	1379	1985
Dynamic logic D Harel Handbook of philosophical logic, 497-604	1291	1984
LSCs: Breathing life into message sequence charts W Damm, D Harel Formal methods in system design 19 (1), 45-80	1242	2001
Executable object modeling with statecharts D Harel, E Gery Proceedings of IEEE 18th International Conference on Software Engineering …	1057	1996
Modeling reactive systems with statecharts: the STATEMATE approach D Harel, M Politi McGraw-Hill, Inc.	950	1998
First-order dynamic logic D Harel	893	1979
On the formal semantics of statecharts D Harel Proc of 2nd IEEE Symposium on Logic in Computer Science, 1987	804	1987

Cited by VIEW ALL

	All	Since 2016
Citations	44936	6033
h-index	77	34
i10-index	216	106



Public access VIEW ALL

1 article 11 articles

not available available

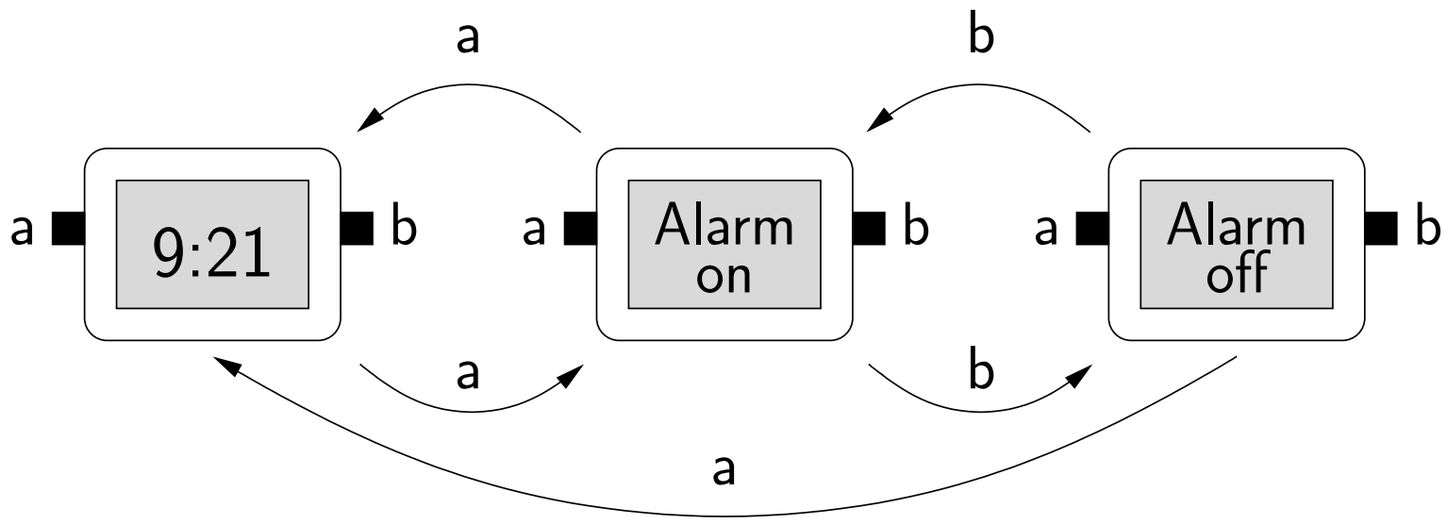
Based on funding mandates

Zustandsdiagramme

Wir lernen Zustandsdiagramme am Beispiel der Modellierung einer Armbanduhr kennen (stark vereinfacht gegenüber Harels Armbanduhr).

Die Armbanduhr hat zwei Knöpfe (a,b) und zwei Modi (Zeitanzeige, Alarmeinstellung). Zwischen diesen Modi wechselt man mit Hilfe von Knopf a.

Der Alarm kann aus (off) oder an (on) sein. Man kann zwischen den Alarmzuständen mit Hilfe von Knopf b wechseln.



Zustandsdiagramme

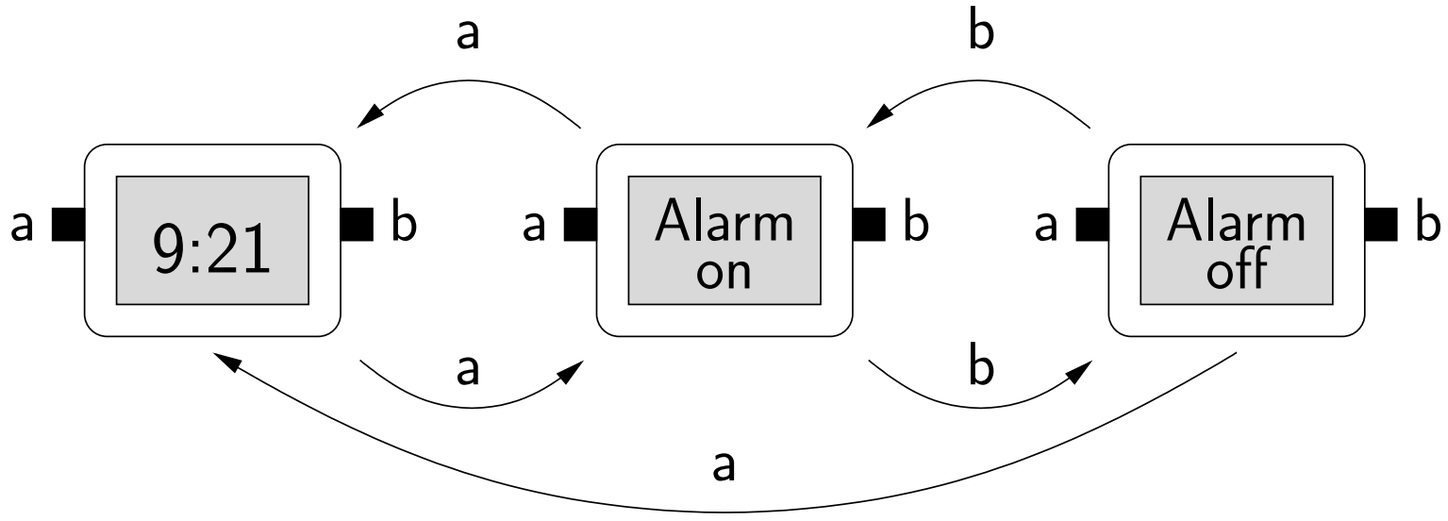
"Alarm" bedeutet, dass zu jeder vollen Stunde ein Beep ertönt.

= "Chime" before

Wir lernen Zustandsdiagramme am Be einer Armbanduhr kennen (stark vereinfacht Armbanduhr).

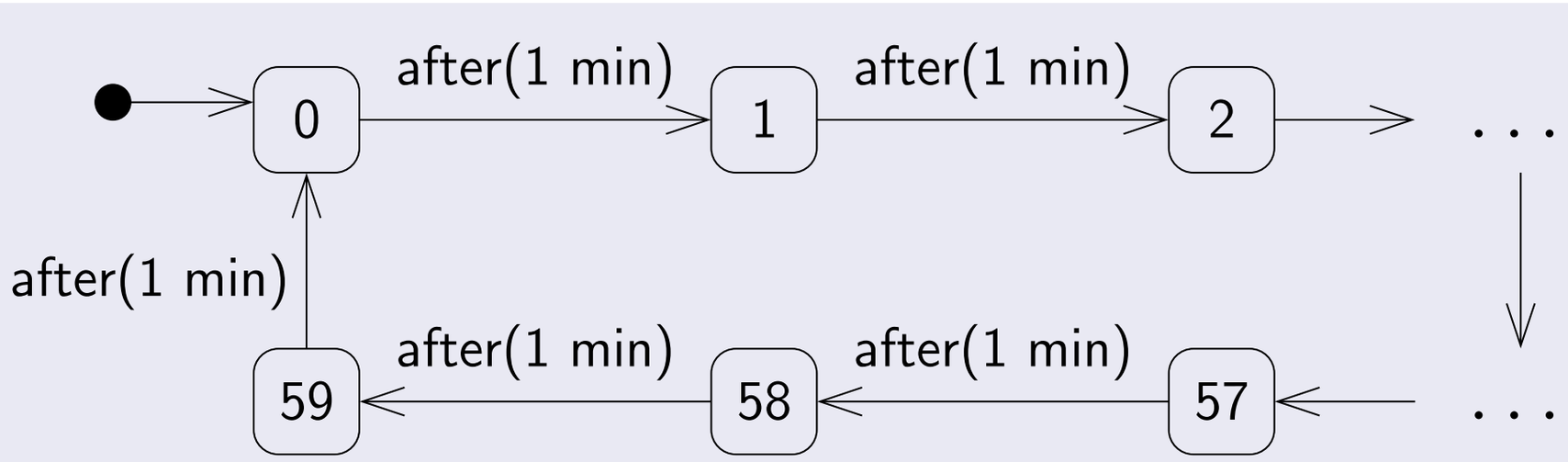
Die Armbanduhr hat zwei Knöpfe (a,b) (Zeitanzeige, Alarmeinstellung). Zwischen diesen Modi wechselt man mit Hilfe von Knopf a.

Der Alarm kann aus (off) oder an (on) sein. Man kann zwischen den Alarmzuständen mit Hilfe von Knopf b wechseln.



Zustandsdiagramme

Wir beginnen zunächst mit der Modellierung der Minutenanzeige.



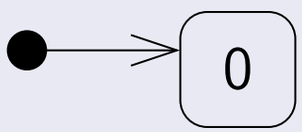
Zustandsdiagramme

Zustand

Ein Zustand eines Zustandsdiagramms wird durch ein Rechteck mit abgerundeten Ecken dargestellt.

Startzustand

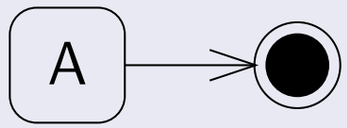
Der Startzustand wird durch einen schwarzen ausgefüllten Kreis gekennzeichnet (ähnlich wie bei Aktivitätsdiagrammen).



Zustandsdiagramme

Endzustand

Endzustände werden wie das Aktivitätsende in Aktivitätsdiagrammen gekennzeichnet.



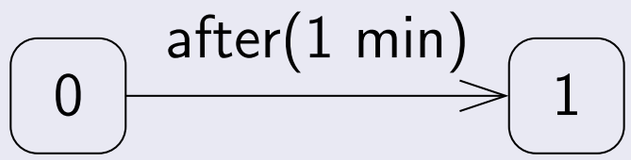
Für den Fall, dass man ein System modelliert, das nicht terminieren soll, gibt es keinen Endzustand (wie in unserem Beispiel).

Zustandsdiagramme

Transition (= Zustandsübergang)

Eine Transition ist ein Pfeil, der mit Ereignis [Bedingung]/Effekt beschriftet ist. (Bedingung und Effekt sind optional.)

- Ereignis: Signal oder Nachricht, die die entsprechende Transition auslösen.



- Bedingung: Überwachungsbedingung (auch Guard genannt).
- Effekt: Effekt, der durch die Transition ausgelöst wird .

Im obigen Beispiel (Minutenanzeige) gibt es nur Ereignisse (sogenannte time events), die die Zeitspanne spezifizieren, nach der die Transition ausgelöst wird. Es gibt aber auch andere Ereignisse, beispielsweise Methodenaufrufe.

Zustandsdiagramme

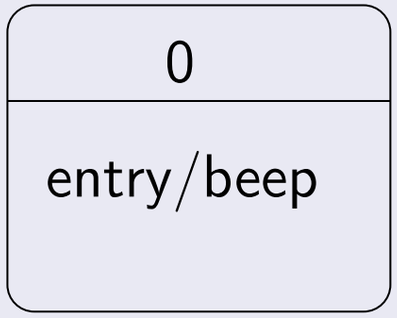
Neben den Effekten, die durch Transitionen ausgelöst werden, können in einem Zustand weitere Aktionen bei Eintritt, Verweilen oder Verlassen ausgelöst werden.

Sie haben den gleichen Aufbau wie die Beschriftung einer Transition: Ereignis [Bedingung]/Effekt. Dabei kann Ereignis unter anderem folgendes sein:

- entry: der entsprechende Effekt wird bei Eintritt in den Zustand ausgelöst.
- do: der Effekt ist eine Aktion, die nach Betreten des Zustands ausgeführt wird und die spätestens dann endet, wenn der Zustand verlassen wird
- exit: der entsprechende Effekt wird bei Verlassen des Zustands ausgelöst.

Zustandsdiagramme

Beispiel: die Uhr soll zu jeder vollen Stunden ein Signal von sich geben. Daher wird die Aktion beep bei Eintritt in den Zustand 0 ausgelöst.



Zustandsdiagramme

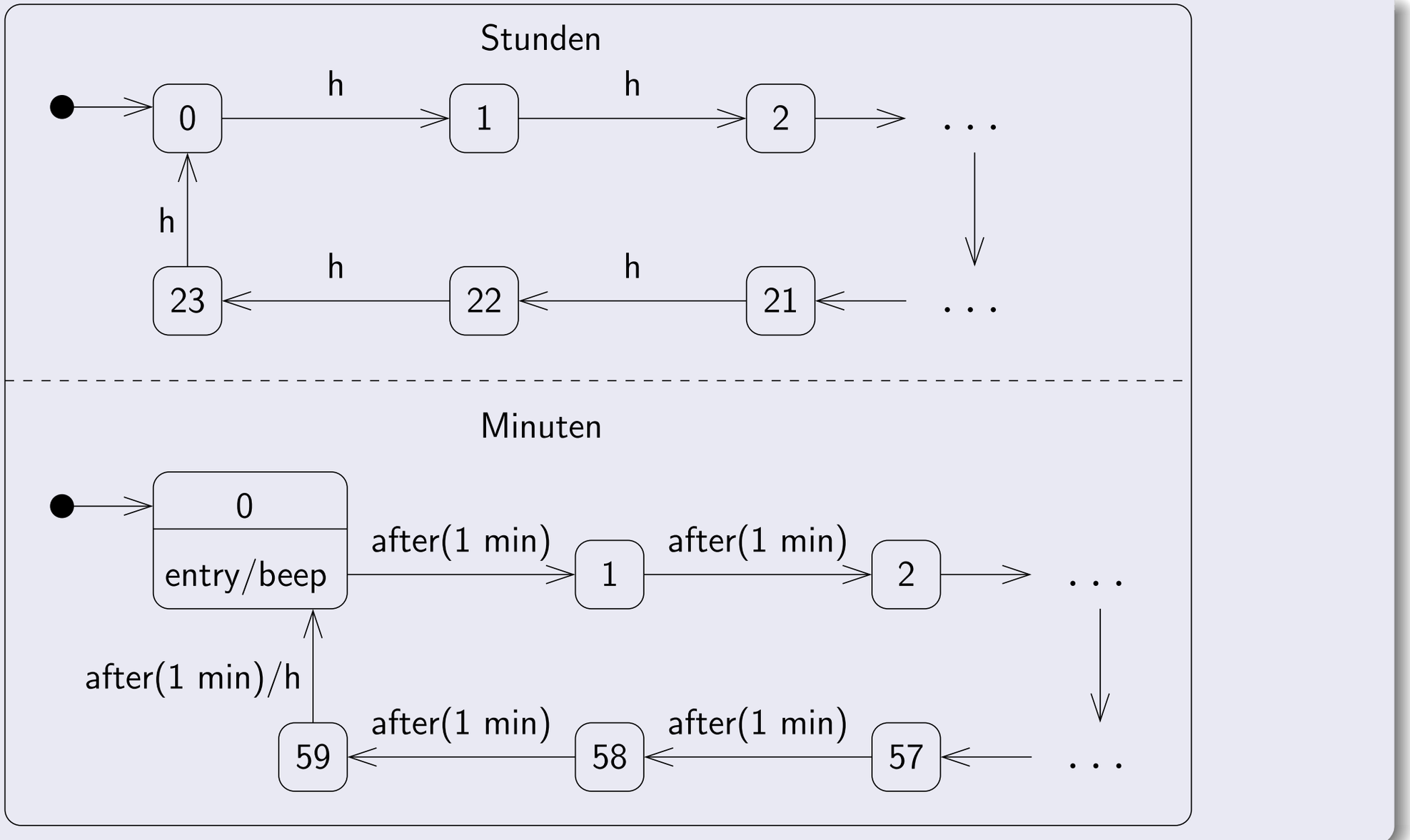
Wie sieht es mit der Stundenanzeige aus? Diese soll parallel zur Minutenanzeige laufen, d.h., die Uhr ist in zwei Zuständen gleichzeitig, ein Zustand für die Stunden, der andere für die Minuten. Diese Situation wird durch Regionen modelliert.

Region

Regionen unterteilen ein Zustandsdiagramm in zwei Bereiche, die parallel zueinander ausgeführt werden.

Zustandsdiagramme

So sieht das modifizierte Zustandsdiagramm für die Uhr jetzt aus:



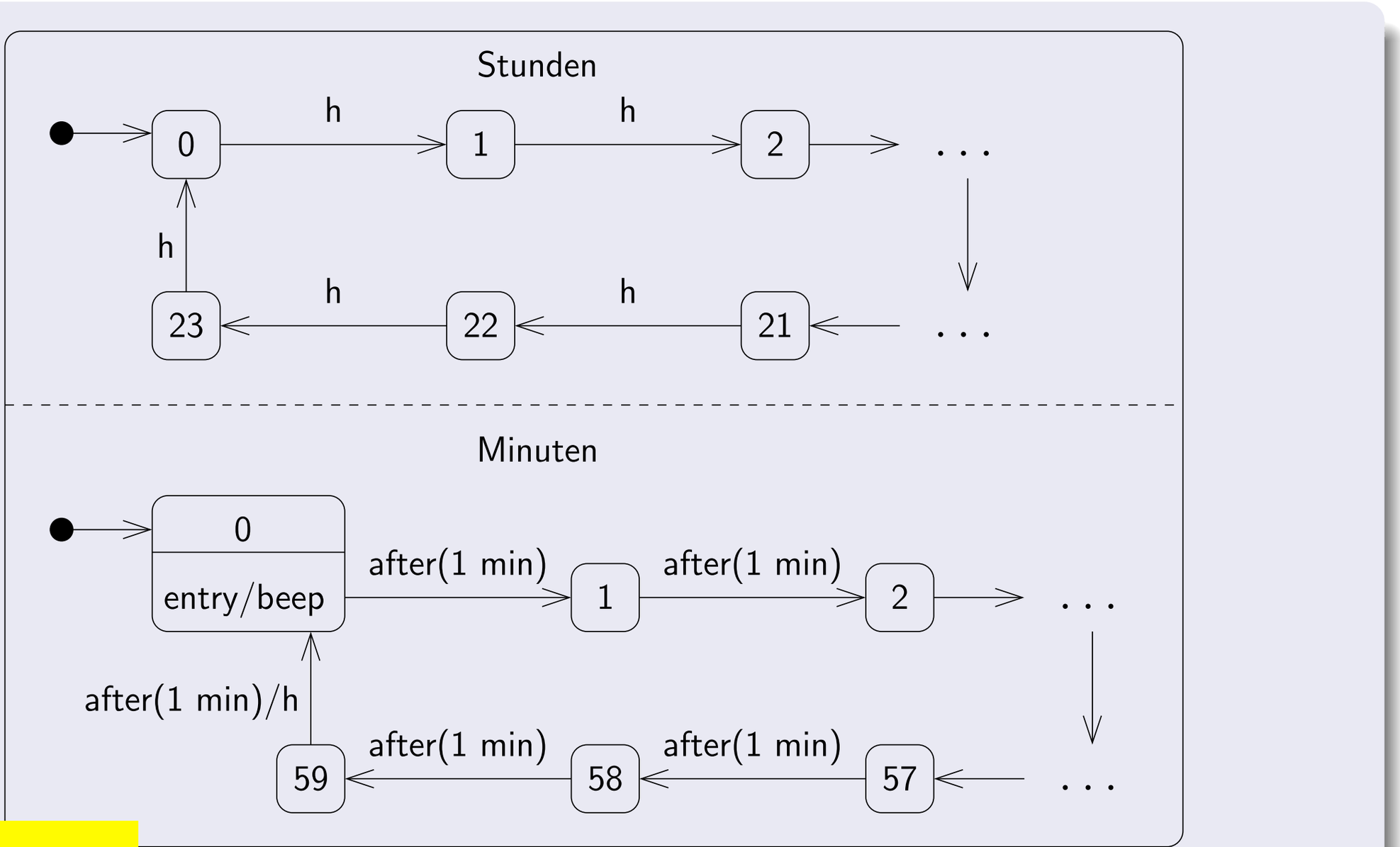
Zustandsdiagramme

So sieht das modifizierte Zustandsdiagramm für die Uhr jetzt aus:

24 Zustände

60 Zustände

84 Zustände (in total)



Zustandsdiagramme

Wie sieht es mit der Stundenanzeige aus? Diese soll parallel zur Minutenanzeige laufen, d.h., die Uhr ist in zwei Zuständen gleichzeitig, ein Zustand für die Stunden, der andere für die Minuten. Diese Situation wird durch Regionen modelliert.

Region

Regionen unterteilen ein Zustandsdiagramm in zwei Bereiche, die parallel zueinander ausgeführt werden.

Question construct an equivalent state diagram (finite state automaton) without regions (only one region).
— How many states does it have?

Zustandsdiagramme

Wie sieht es mit der Stundenanzeige aus? Diese soll parallel zur Minutenanzeige laufen, d.h., die Uhr ist in zwei Zuständen gleichzeitig, ein Zustand für die Stunden, der andere für die Minuten. Diese Situation wird durch Regionen modelliert.

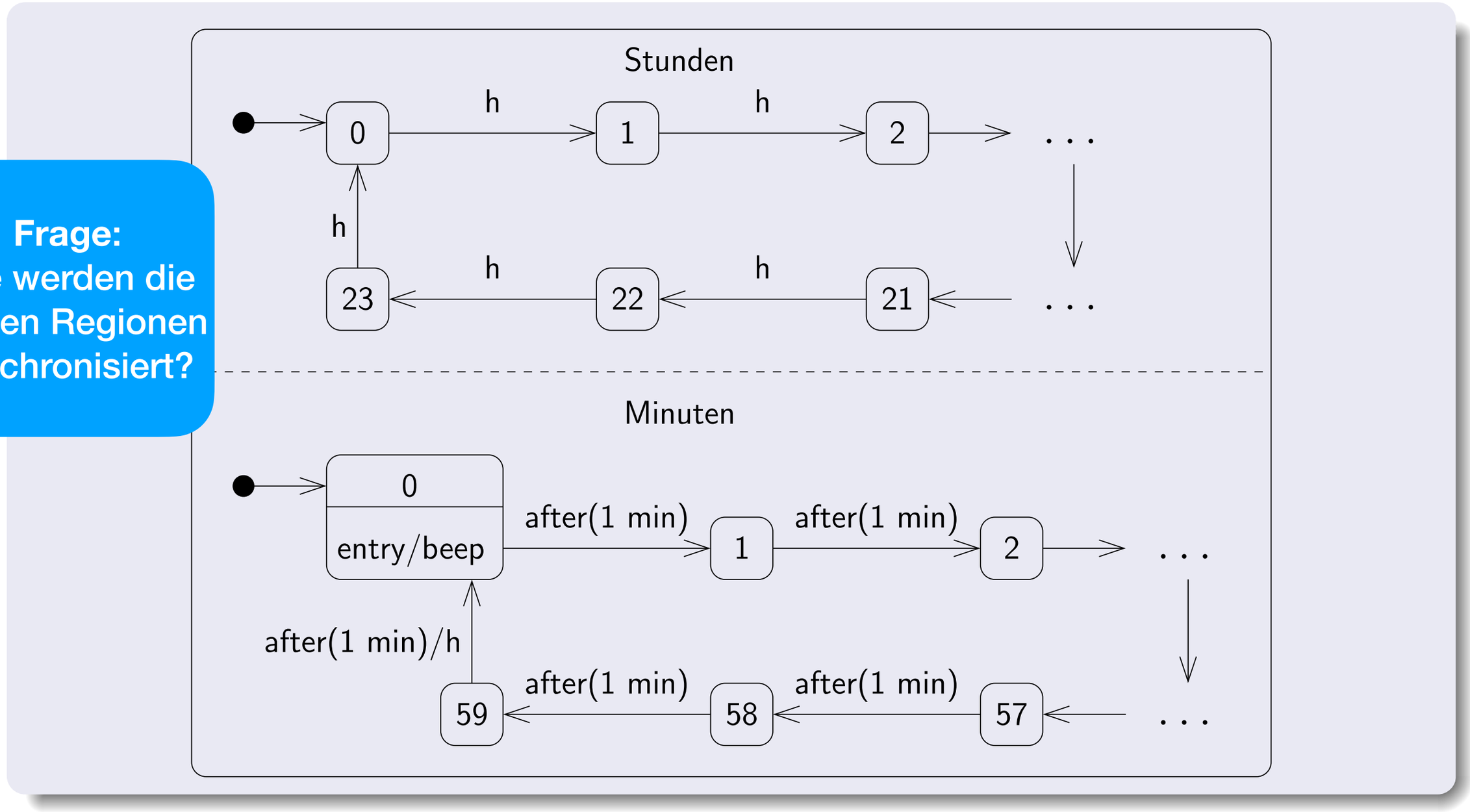
Region

Regionen unterteilen ein Zustandsdiagramm in zwei Bereiche, die parallel zueinander ausgeführt werden.

Dies erlaubt uns, insgesamt nur $24 + 60 = 84$ Zustände, anstatt $24 \cdot 60 = 1440$ Zustände zu zeichnen.

Zustandsdiagramme

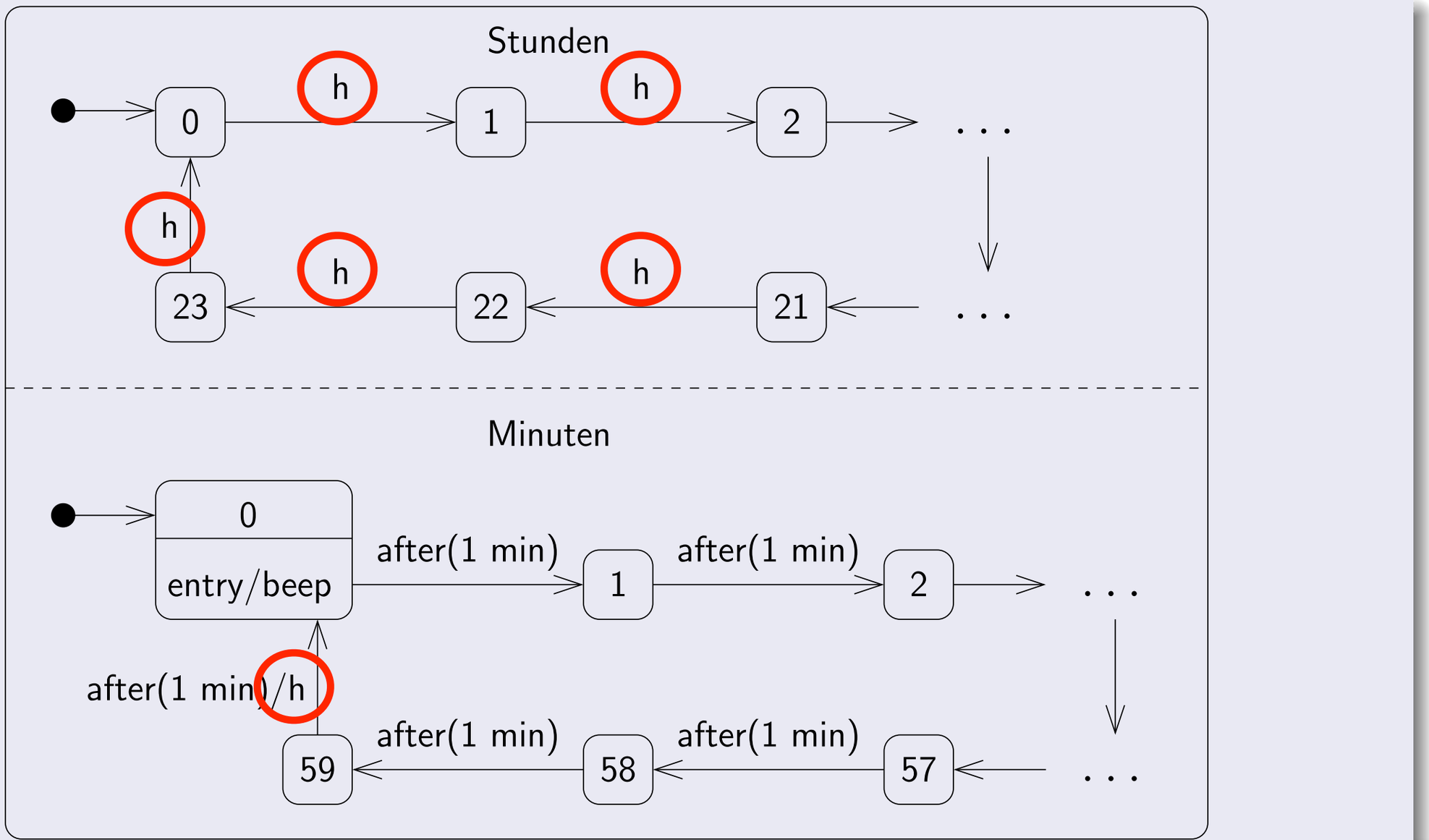
So sieht das modifizierte Zustandsdiagramm für die Uhr jetzt aus:



Frage:
Wie werden die beiden Regionen synchronisiert?

Zustandsdiagramme

So sieht das modifizierte Zustandsdiagramm für die Uhr jetzt aus:



Zustandsdiagramme

Bemerkungen:

- Es gibt jetzt zwei Startzustände, die beide zu Beginn betreten werden (Uhrzeit: 0:00).
- Da Stunden- und Minutenanzeige nicht vollkommen unabhängig voneinander arbeiten, ist eine Synchronisation eingebaut. Die Transition in den Minutenzustand 0 löst einen Effekt h (h für "hour") aus.

Dieser Effekt ist dann ein Trigger, der das entsprechende Ereignis triggert und den Übergang in den nächsten Stundenzustand verursacht. Die beiden Transitionen werden synchronisiert und finden gleichzeitig statt.

Zustandsdiagramme

Bemerkungen:

- Transitionen verbrauchen im allgemeinen keine Zeit (im Gegensatz zum Aufenthalt in Zuständen).

Zustandsdiagramme

Bemerkungen:

- Transitionen verbrauchen im allgemeinen keine Zeit (im Gegensatz zum Aufenthalt in Zuständen).
- Neben Triggern, die direkt ausgeführt werden, gibt es auch Events, die zunächst in einer Event Queue (= Warteschlange) abgelegt werden. Diese Warteschlange wird schrittweise abgearbeitet, wobei die entsprechenden Ereignisse ausgelöst werden. Damit kann asynchrone Kommunikation beschrieben werden.

Zustandsdiagramme

Bemerkungen:

- Transitionen verbrauchen im allgemeinen keine Zeit (im Gegensatz zum Aufenthalt in Zuständen).
- Neben Triggern, die direkt ausgeführt werden, gibt es auch Events, die zunächst in einer Event Queue (= Warteschlange) abgelegt werden. Diese Warteschlange wird schrittweise abgearbeitet, wobei die entsprechenden Ereignisse ausgelöst werden. Damit kann asynchrone Kommunikation beschrieben werden.
- Effekte können direkte Kommunikation bedeuten (beispielsweise Methodenaufrufe), können aber auch sogenannte Broadcasts (= Rundrufe) sein. Diese sind überall im Zustandsdiagramm sichtbar. (Die ursprüngliche Statecharts-Semantik von Harel verwendete nur Broadcasts.)

Zustandsdiagramme

Nun soll noch die Möglichkeit hinzugefügt werden, das Alarmsignal zur vollen Stunden aus- und wieder einzuschalten. Dazu führen wir weitere Zustände (Alarm on, off) ein, in die man durch Drücken von *a* gelangt.

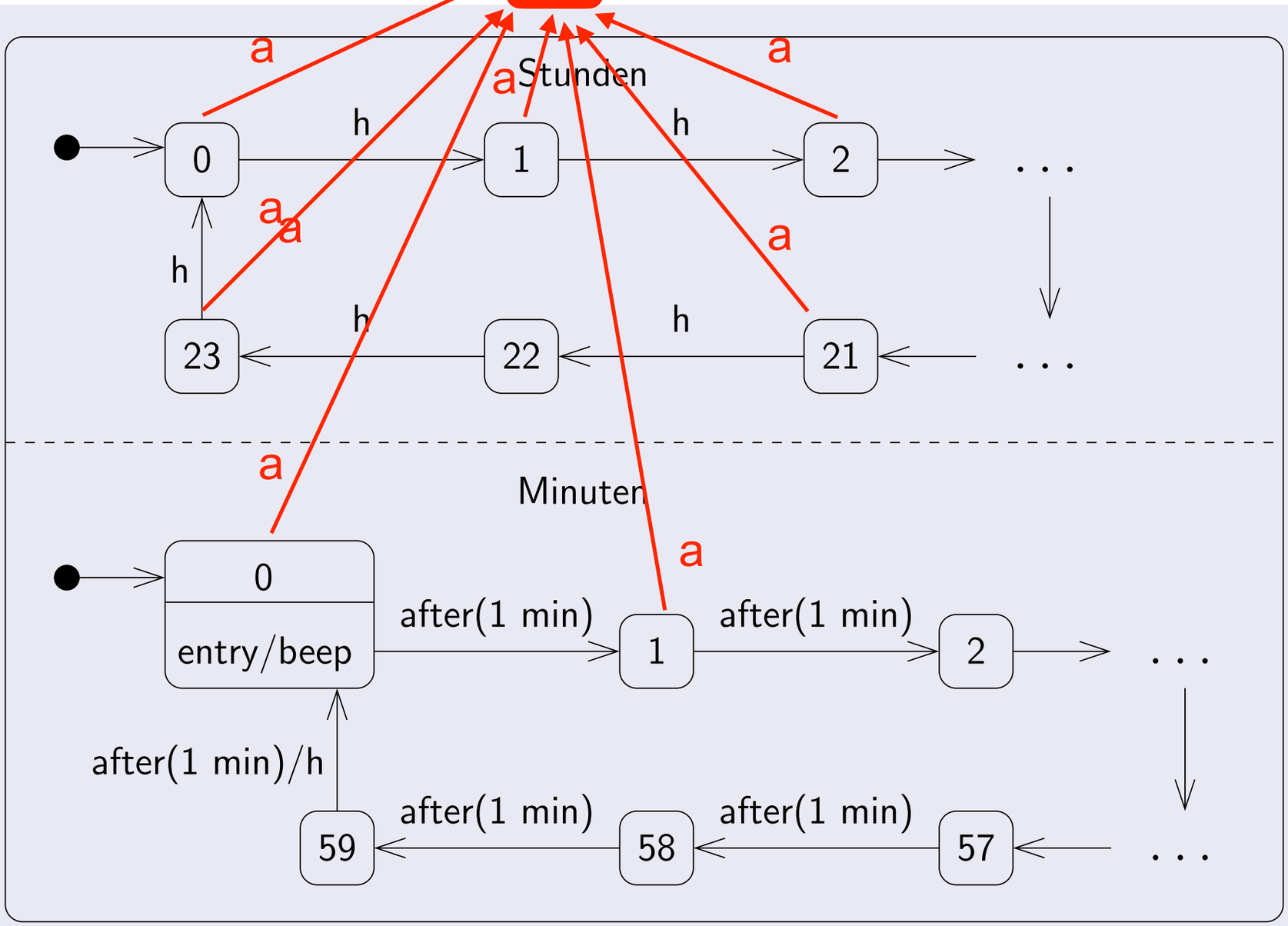
Zustandsdiagramme

Nun soll noch die Möglichkeit hinzugefügt werden, das Alarmsignal zur vollen Stunden aus- und wieder einzuschalten. Dazu führen wir weitere Zustände (Alarm on, off) ein, in die man durch Drücken von *a* gelangt.

Problem: wir brauchen mindestens 84 mit *a* beschriftete Transitionen, die aus den Stunden-/Minutenzuständen ausgehen! Das sind ziemlich viele ...

Zustandsdiagramme

So sieht das modifizierte Zustandsdiagramm für die Uhr jetzt aus:



Zustandsdiagramme

Nun soll noch die Möglichkeit hinzugefügt werden, das Alarmsignal zur vollen Stunden aus- und wieder einzuschalten. Dazu führen wir weitere Zustände (Alarm on, off) ein, in die man durch Drücken von *a* gelangt.

Problem: wir brauchen mindestens 84 mit *a* beschriftete Transitionen, die aus den Stunden-/Minutenzuständen ausgehen! Das sind ziemlich viele ...

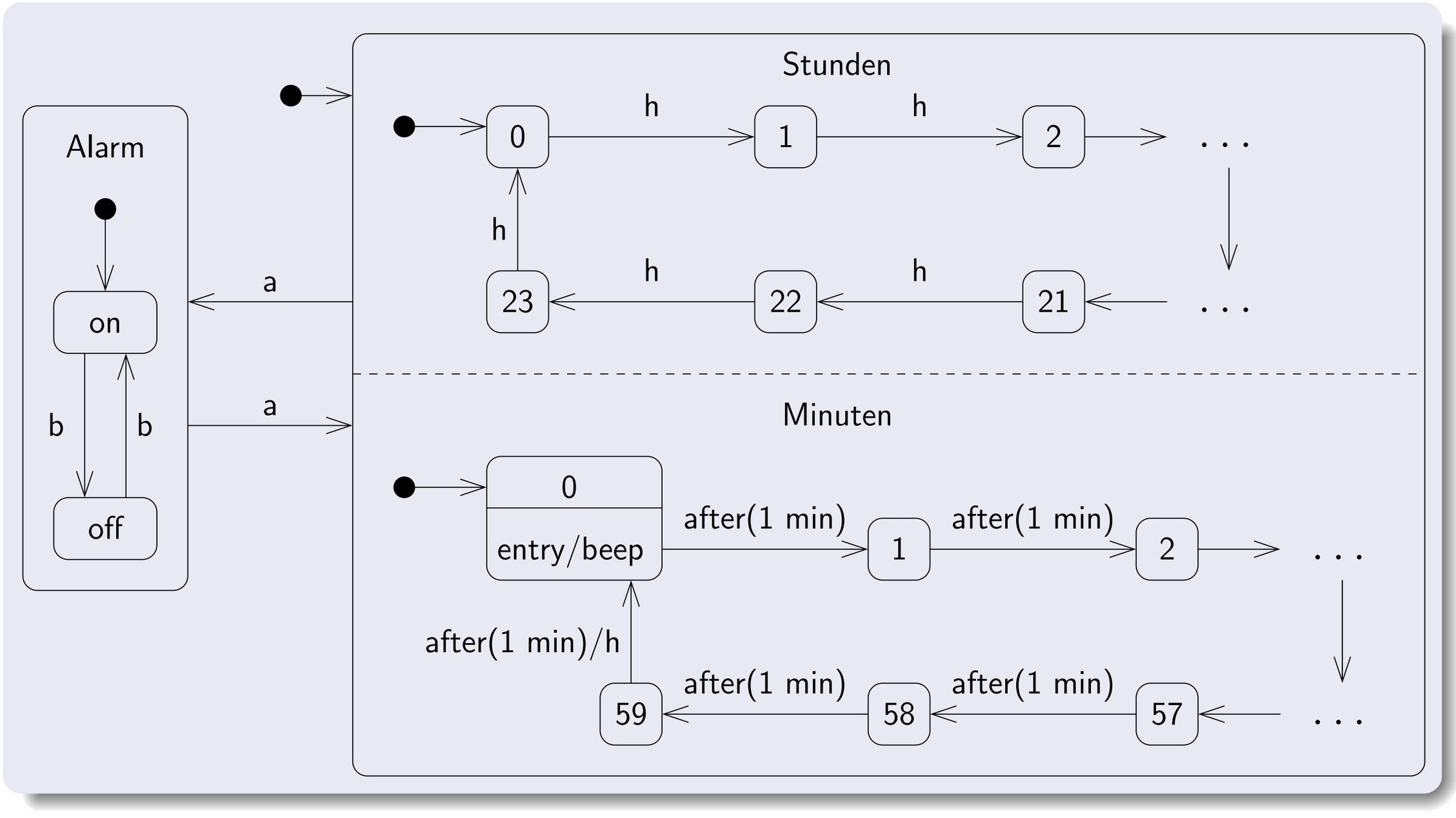
Dafür bieten Zustandsdiagramme folgende Lösung:

Zusammengesetzte Zustände

Zusammengesetzte Zustände dienen dazu, um Hierarchien von Zuständen zu modellieren und damit ein- und ausgehende Transitionen zusammenzufassen.

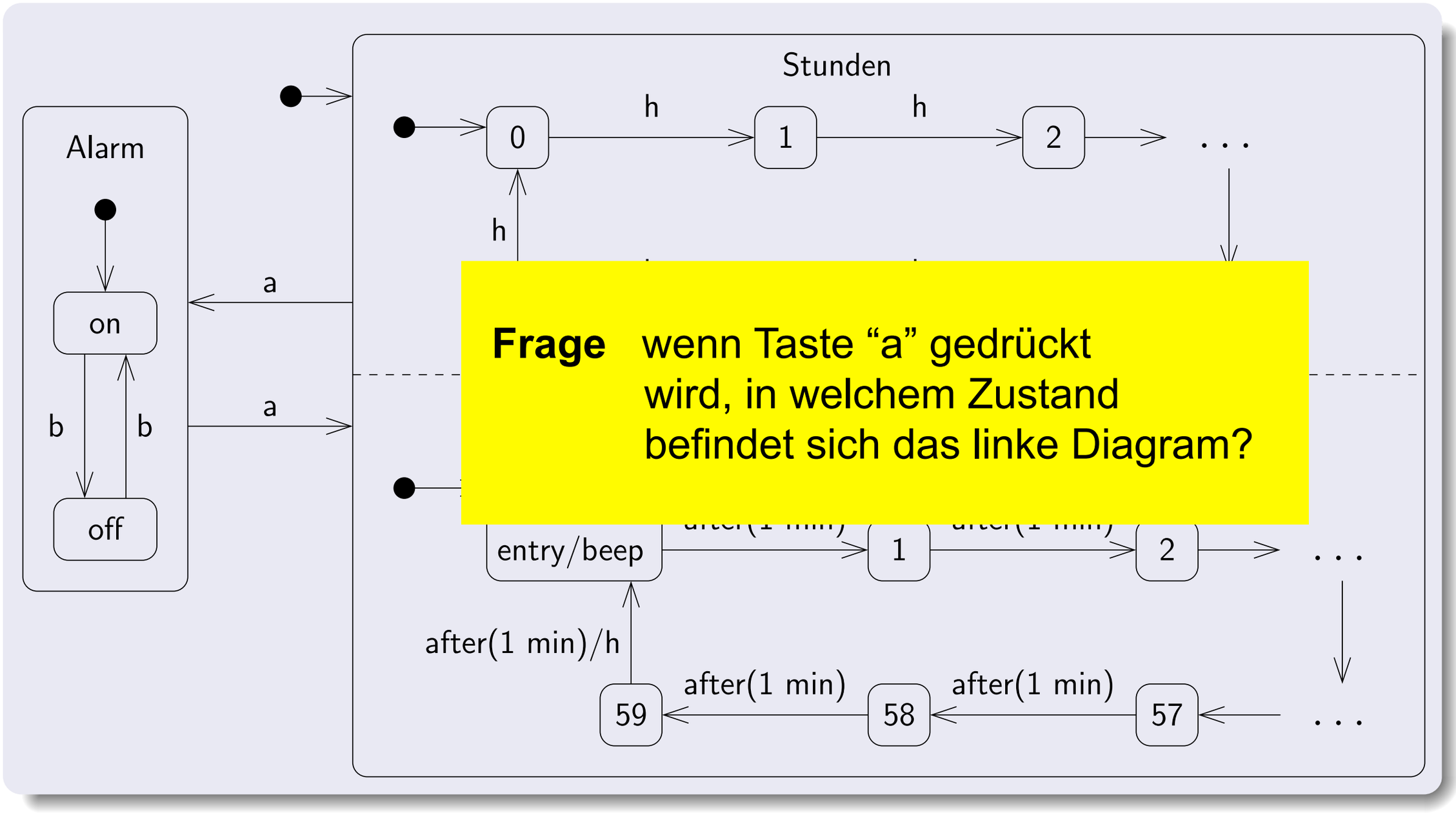
Zustandsdiagramme

Damit ergibt sich folgendes Diagramm:



Zustandsdiagramme

Damit ergibt sich folgendes Diagramm:



Frage wenn Taste “a” gedrückt wird, in welchem Zustand befindet sich das linke Diagram?

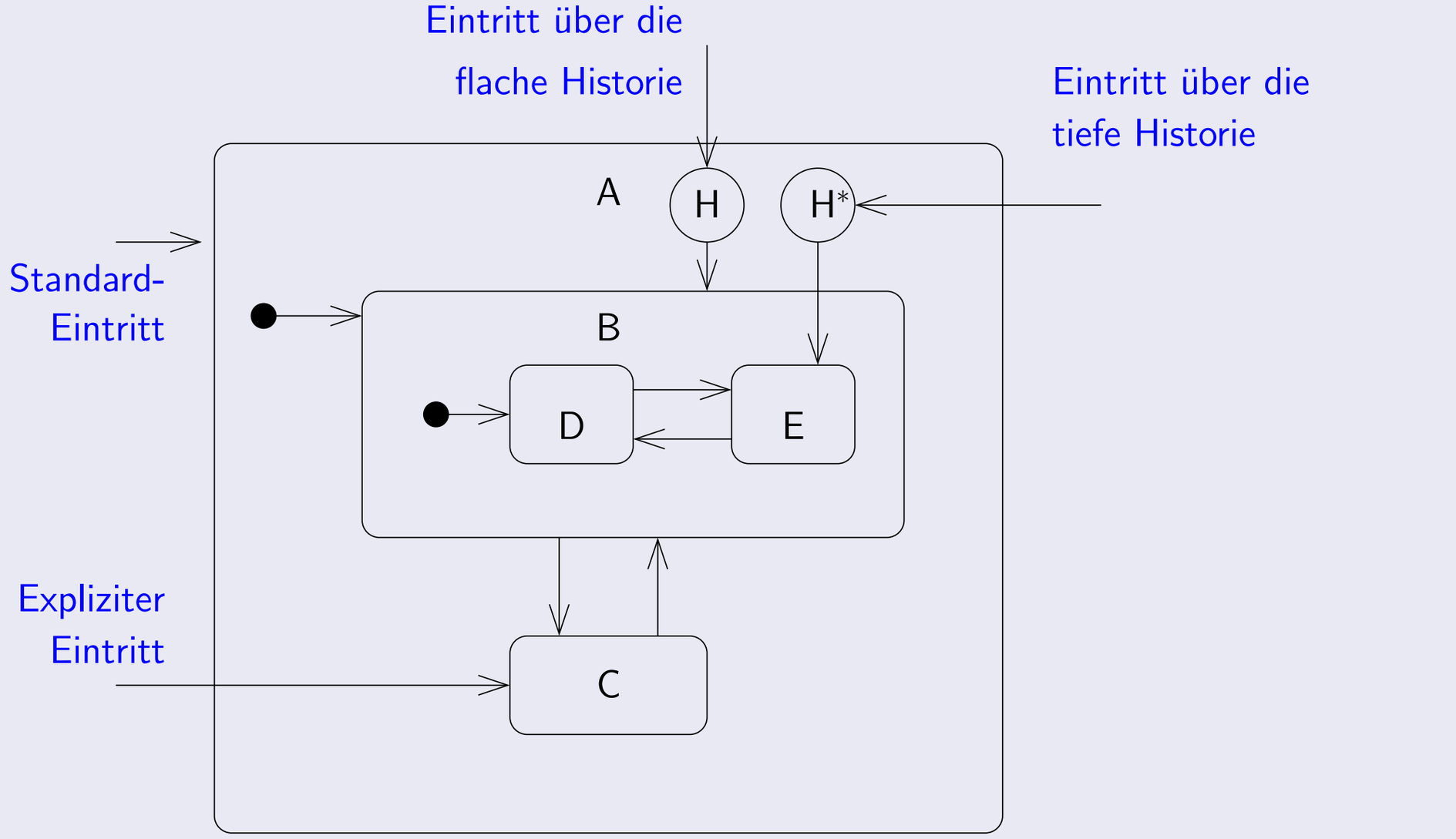
Zustandsdiagramme

Um eine gewisse Flexibilität bei der Modellierung zu haben, werden verschiedene Eintritts- und Austrittsmöglichkeiten bereitgestellt.

Zustandsdiagramme

Um eine gewisse Flexibilität bei der Modellierung zu haben, werden verschiedene Eintritts- und Austrittsmöglichkeiten bereitgestellt.

Eintrittsmöglichkeiten in einen Zustand (graphisch)



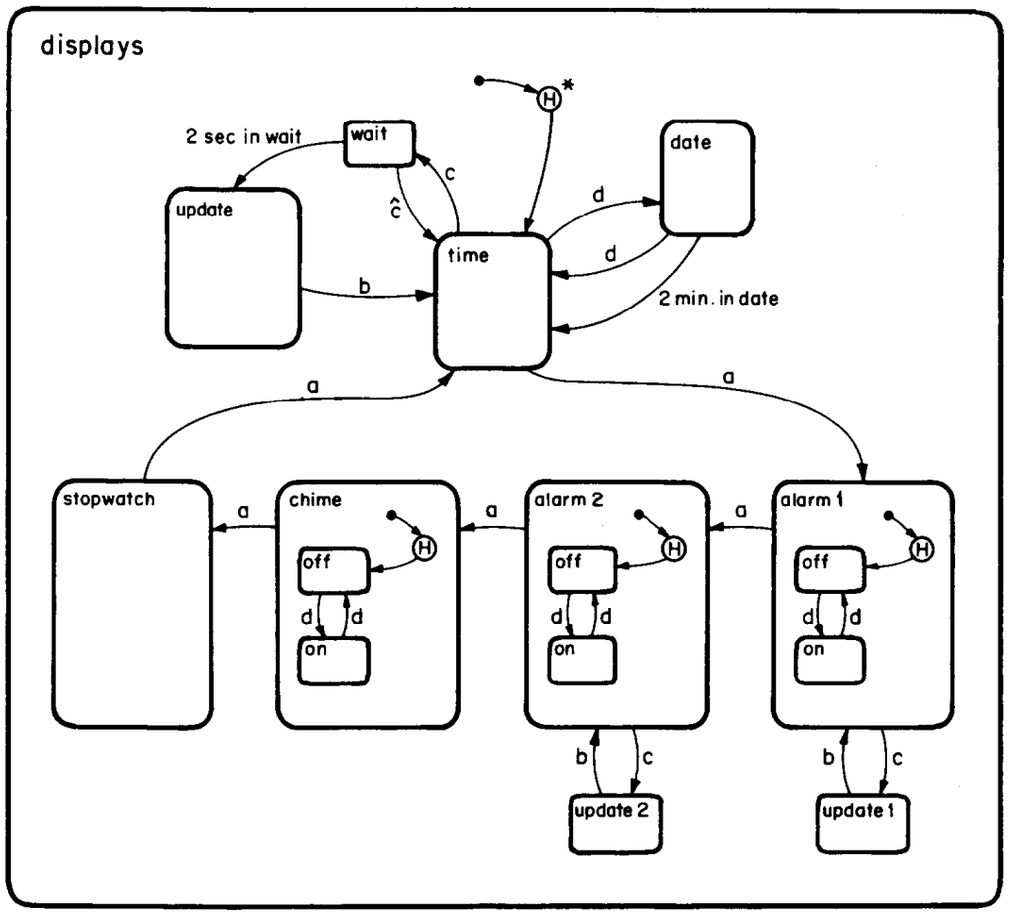

displays
2 sec in wait
wait
update
c
ĉ
time
date
d
d
2 min. in date
b
a
a
stopwatch
a
chime
a
alarm 2
a
alarm 1
off
d
d
on
b
c
update 2
update 1

Fig. 13.

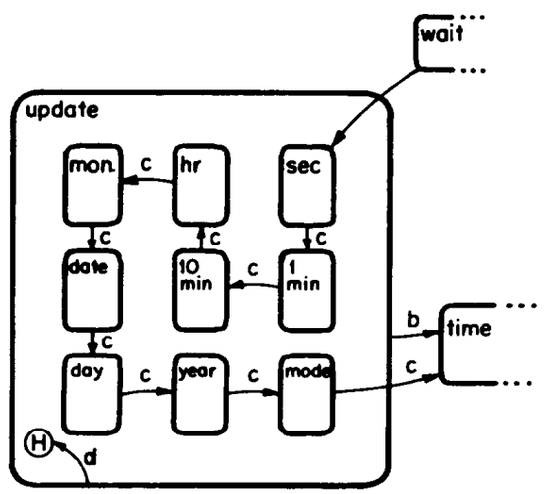
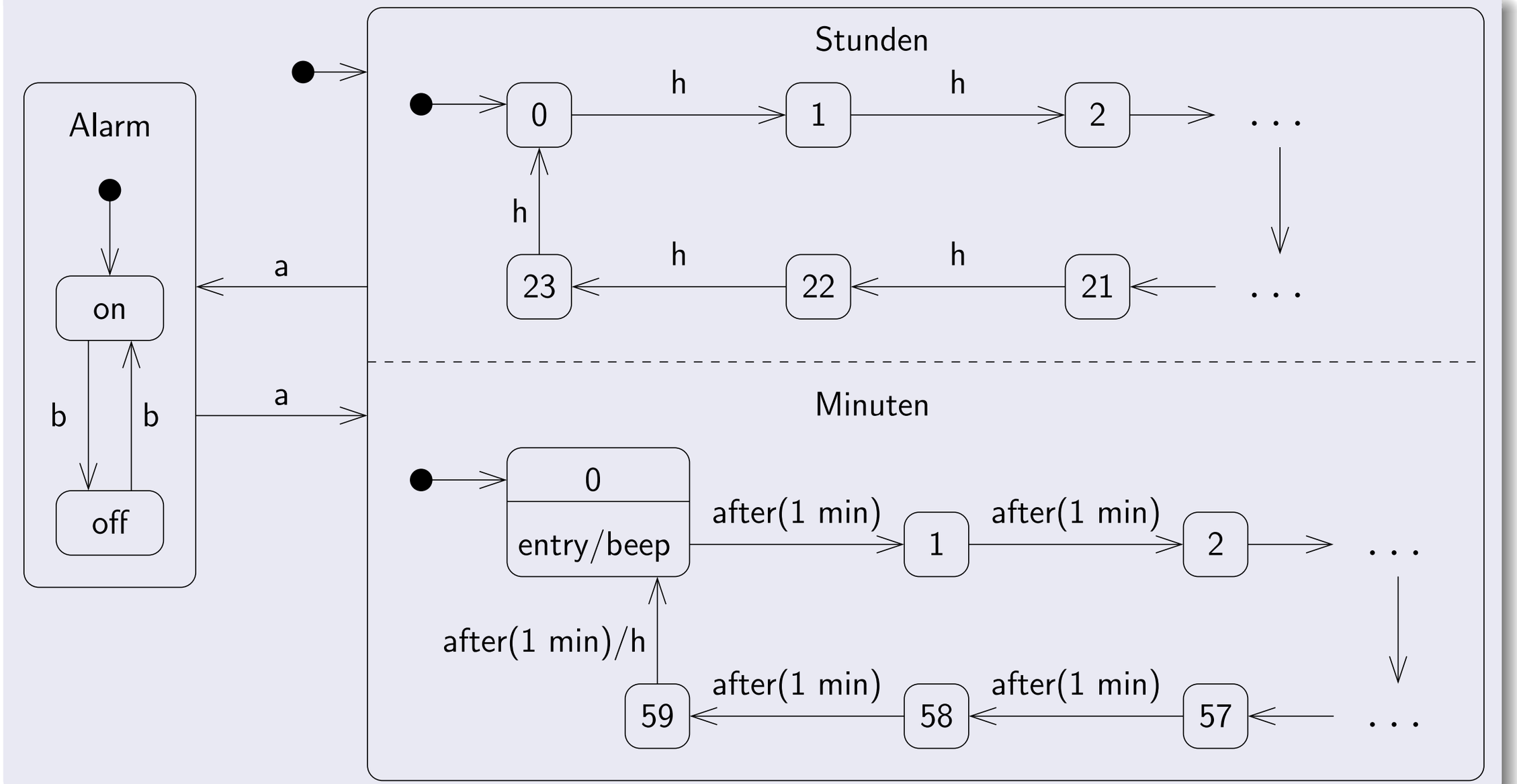

wait
update
mon.
c
hr
sec
c
c
c
date
10 min
c
1 min
c
b
time
c
day
c
year
c
mode
d

Fig. 14.

Zustandsdiagramme

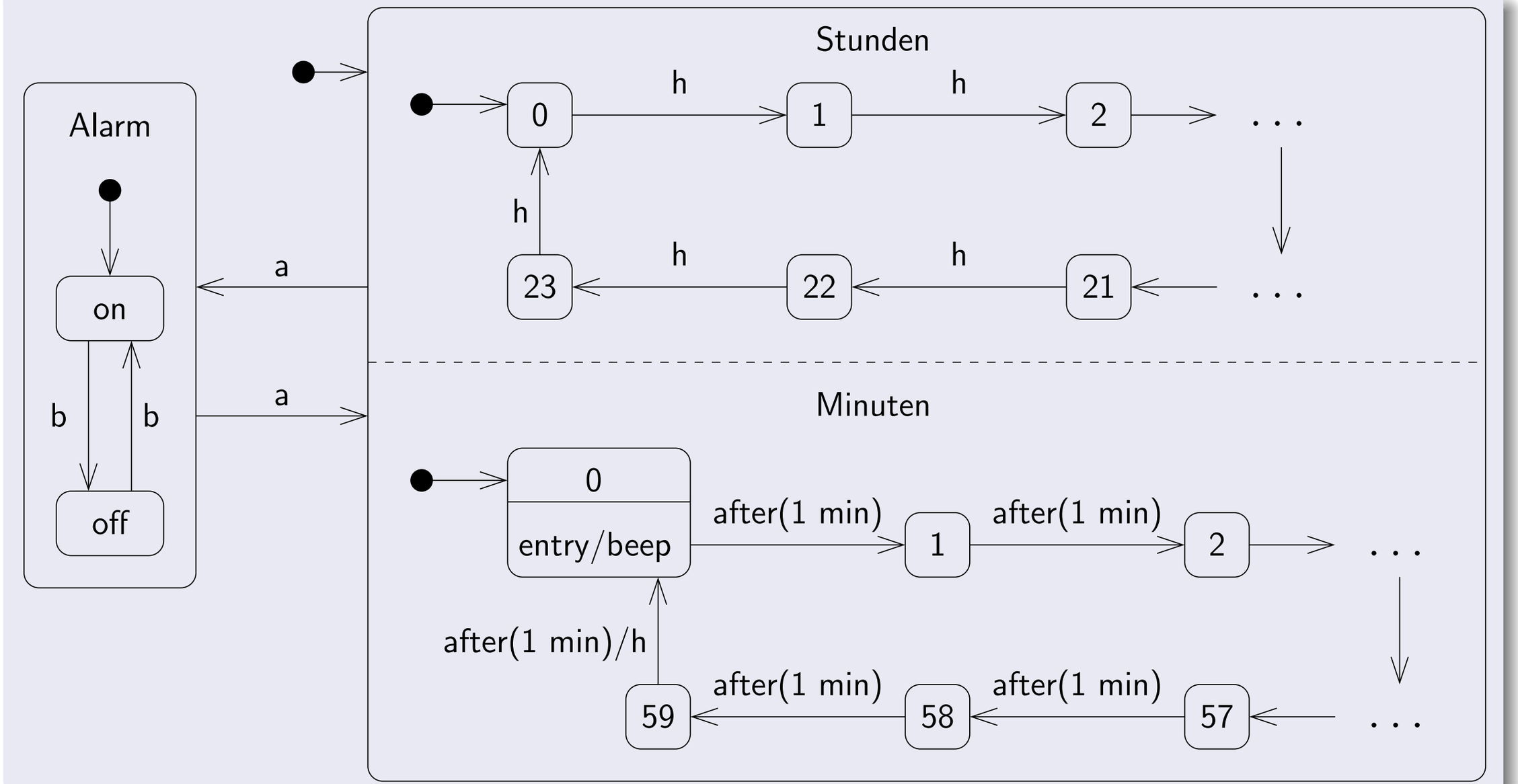
Beschreibung der Eintrittsmöglichkeiten:

- Standard-Eintritt: dabei wird der Startzustand des zusammengesetzten Zustands angesprungen.
  (Fortsetzung bei B, was schließlich zu einer Fortsetzung bei D führt)
- Expliziter Eintritt: es wird bei dem explizit angegebenen Folgezustand fortgesetzt.
  (Fortsetzung bei C.)

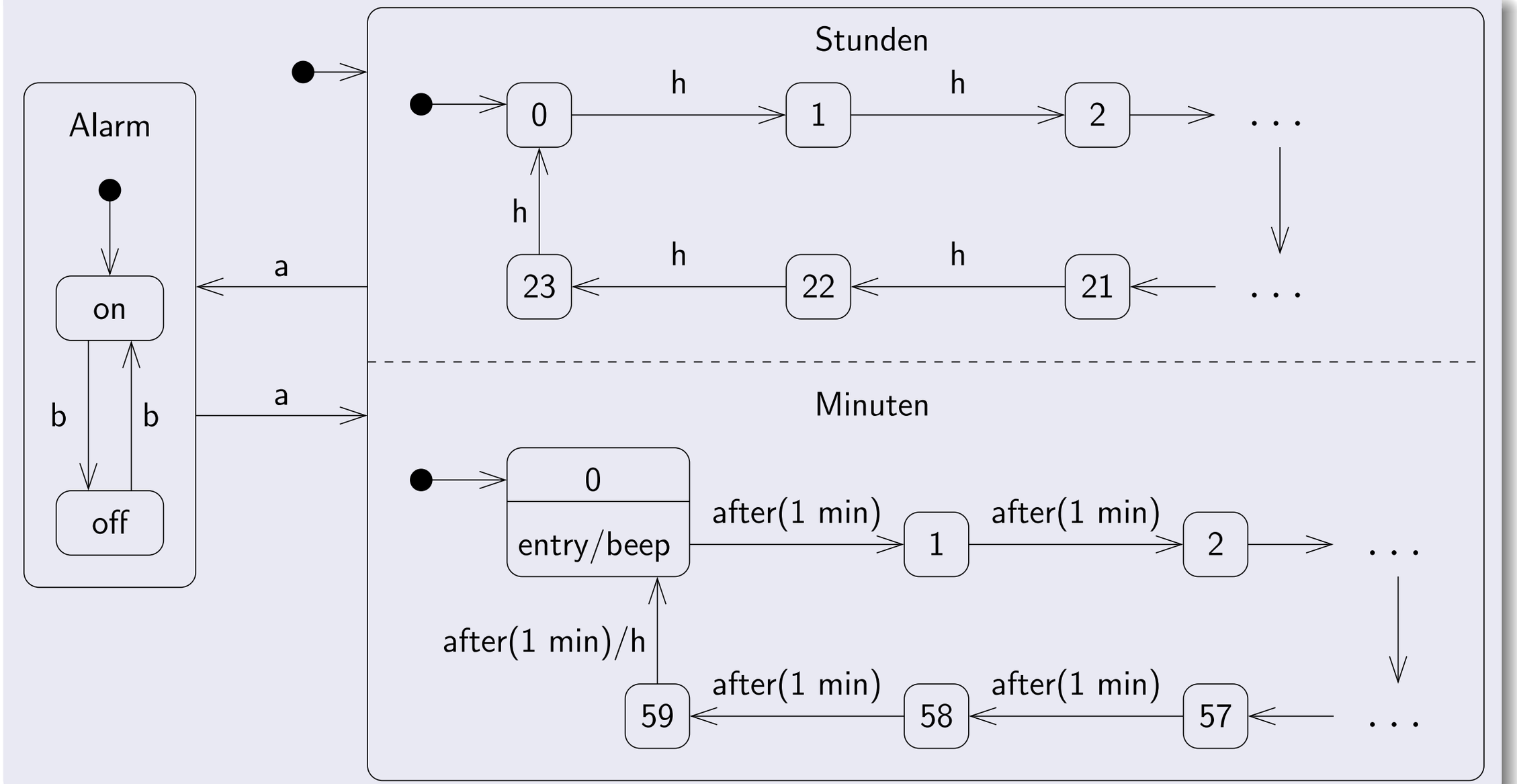


No marking: Standard-Eintritt is used

What does that mean wrt using the watch?



If we press “a”, then the Alarm is always “on”.
To turn the alarm “off”, we need to press “a” followed by “b”.



It would be more intuitive if:

— pressing “a” shows the current state (“on” / “off”)
— then use “b” to toggle between “on” and “off”

Zustandsdiagramme

Beschreibung der Eintrittsmöglichkeiten (Fortsetzung):

- Eintritt über die tiefe Historie: Wurde der zusammengesetzte Zustand bereits früher besucht, so wird der letzte vor dem Verlassen des Zustands aktive Unterzustand *der tiefstmöglichen Ebene* betreten.

  (Falls also der zusammengesetzte Zustand das letzte Mal von E aus verlassen wurde, so wird jetzt wieder bei E fortgesetzt.)

  Falls man noch niemals zuvor diesen zusammengesetzten Zustand betreten hat, so wird der Zustand betreten, der mit der Kante gekennzeichnet ist, die aus dem H^*-Knoten ausgeht.

Zustandsdiagramme

Beschreibung der Eintrittsmöglichkeiten (Fortsetzung):

- Eintritt über die flache Historie: Wurde der zusammengesetzte Zustand bereits früher besucht, so wird der letzte vor dem Verlassen des Zustands aktive Unterzustand *der obersten Ebene* betreten.

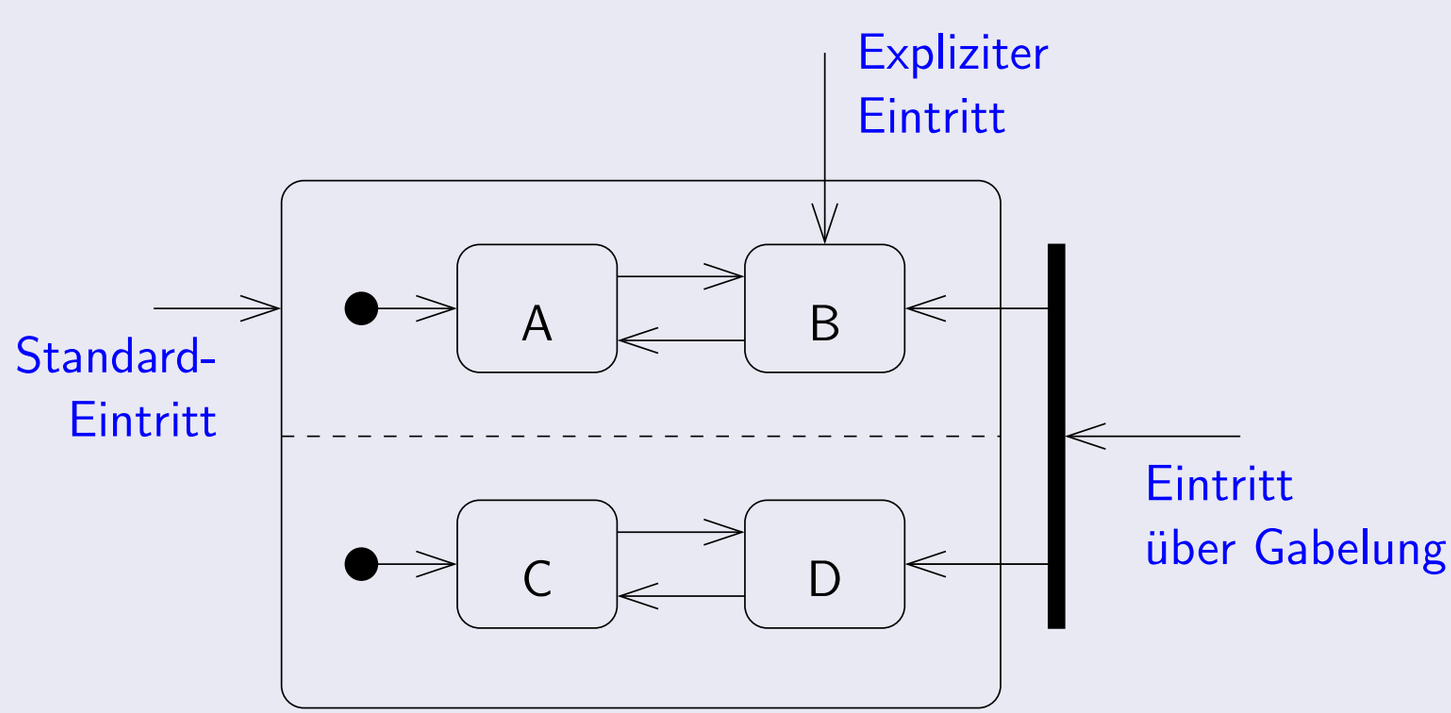
  (Falls also der zusammengesetzte Zustand das letzte Mal von E aus verlassen wurde, so wird jetzt bei B fortgesetzt, was letztendlich zu einer Fortsetzung bei D führt.)

Außerdem: Eintritt über einen Eintrittspunkt (wird hier nicht behandelt).

Zustandsdiagramme

Wenn ein Zustand in mehrere Regionen unterteilt ist, so ergeben sich bei den Eintrittsmöglichkeiten einige Besonderheiten.

Eintrittsmöglichkeiten bei Regionen (graphisch)



Zustandsdiagramme

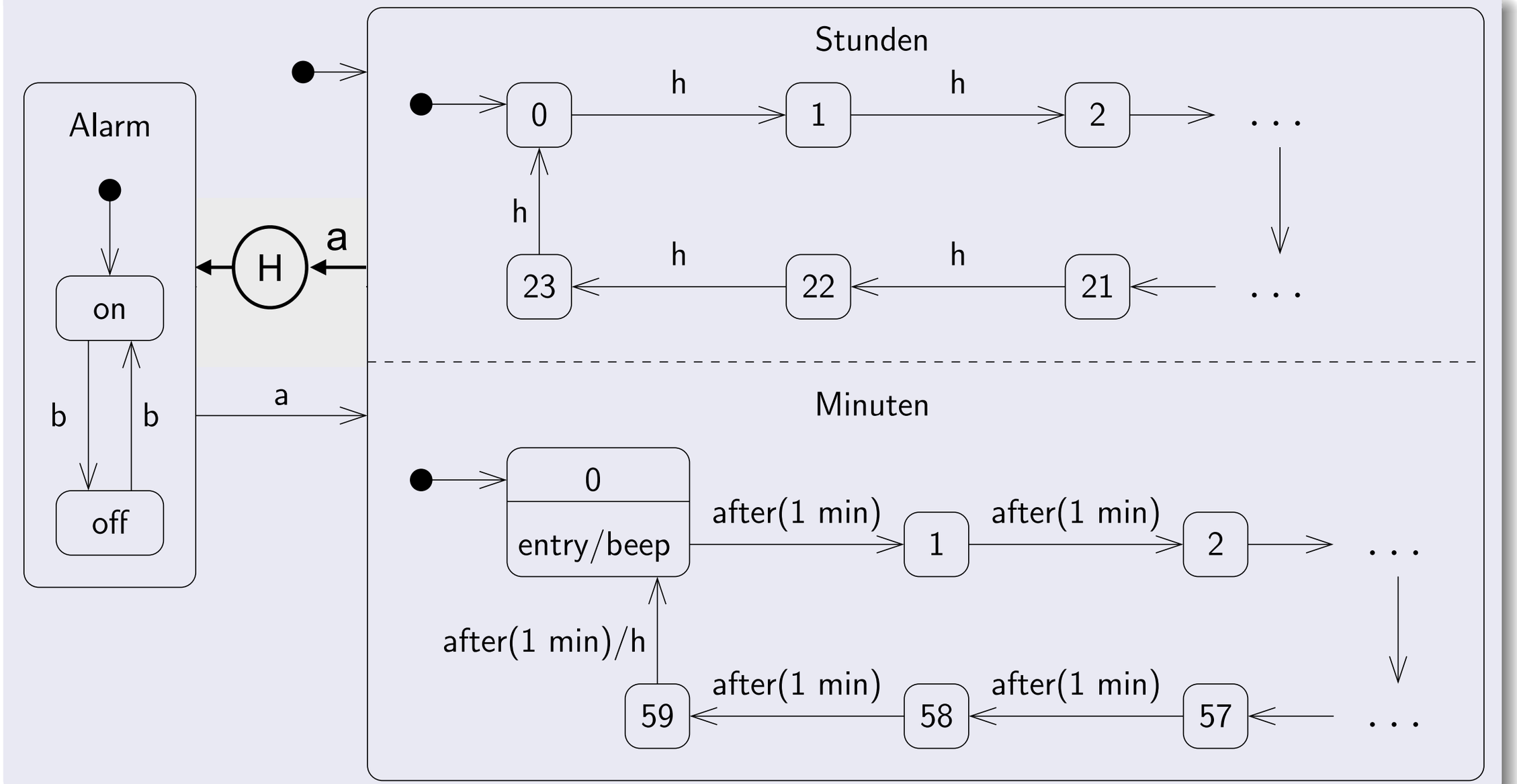
Beschreibung der Eintrittsmöglichkeiten bei Regionen:

- Standard-Eintritt: dabei werden die jeweiligen Startzustände der Regionen angesprungen.

  (Fortsetzung bei A und C)
- Expliziter Eintritt: ein Zustand einer Region wird direkt angesprungen. In der anderen Region wird bei dem entsprechenden Startzustand fortgesetzt.

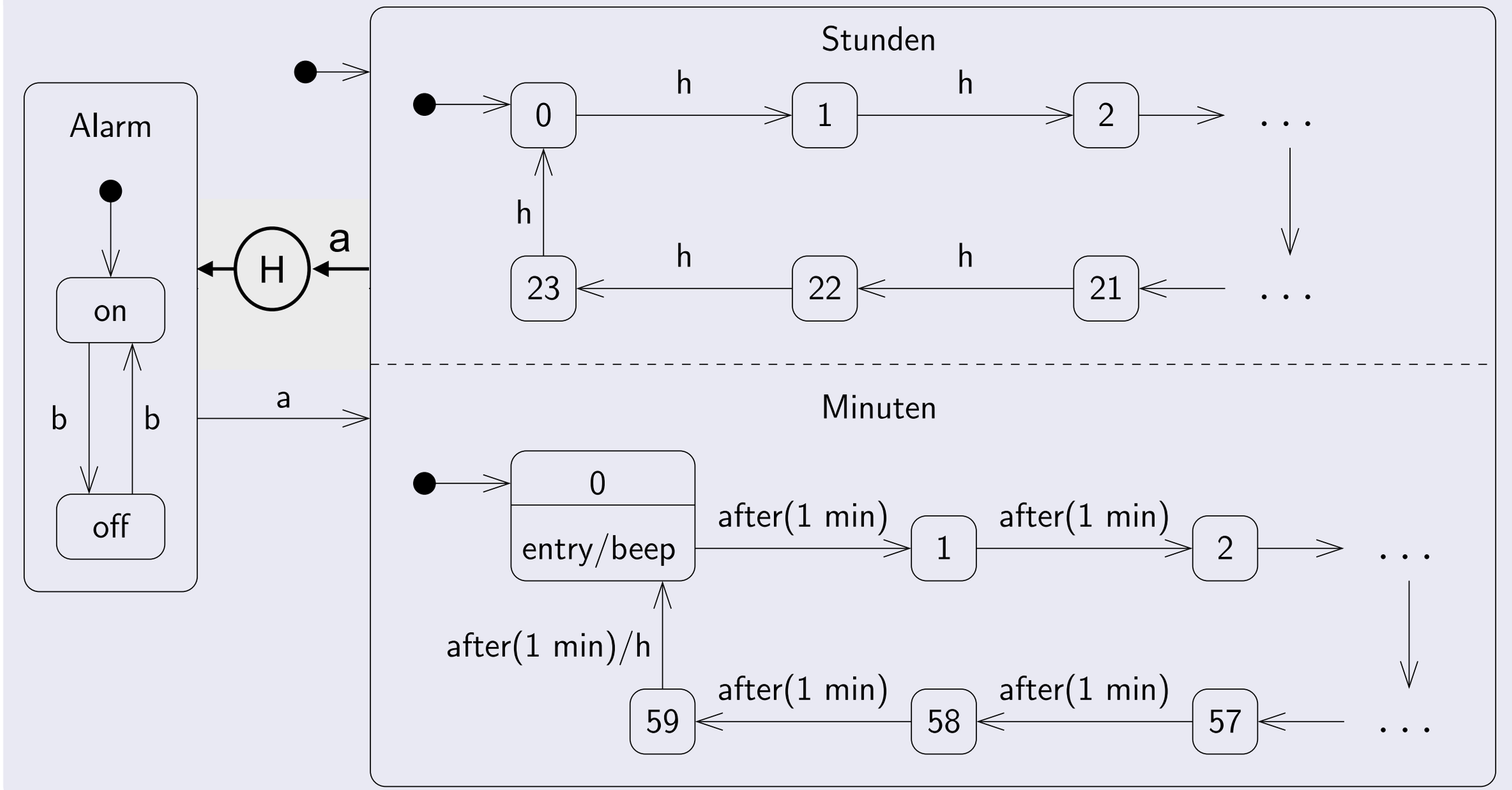
  (Fortsetzung bei B und C.)
- Eintritt über Gabelung: ähnlich wie bei Aktivitätsdiagrammen wird eine Gabelung eingesetzt, um die beiden anzuspringenden Zustände in den Regionen zu kennzeichnen.

  (Fortsetzung bei B und D.)



It would be more intuitive if:

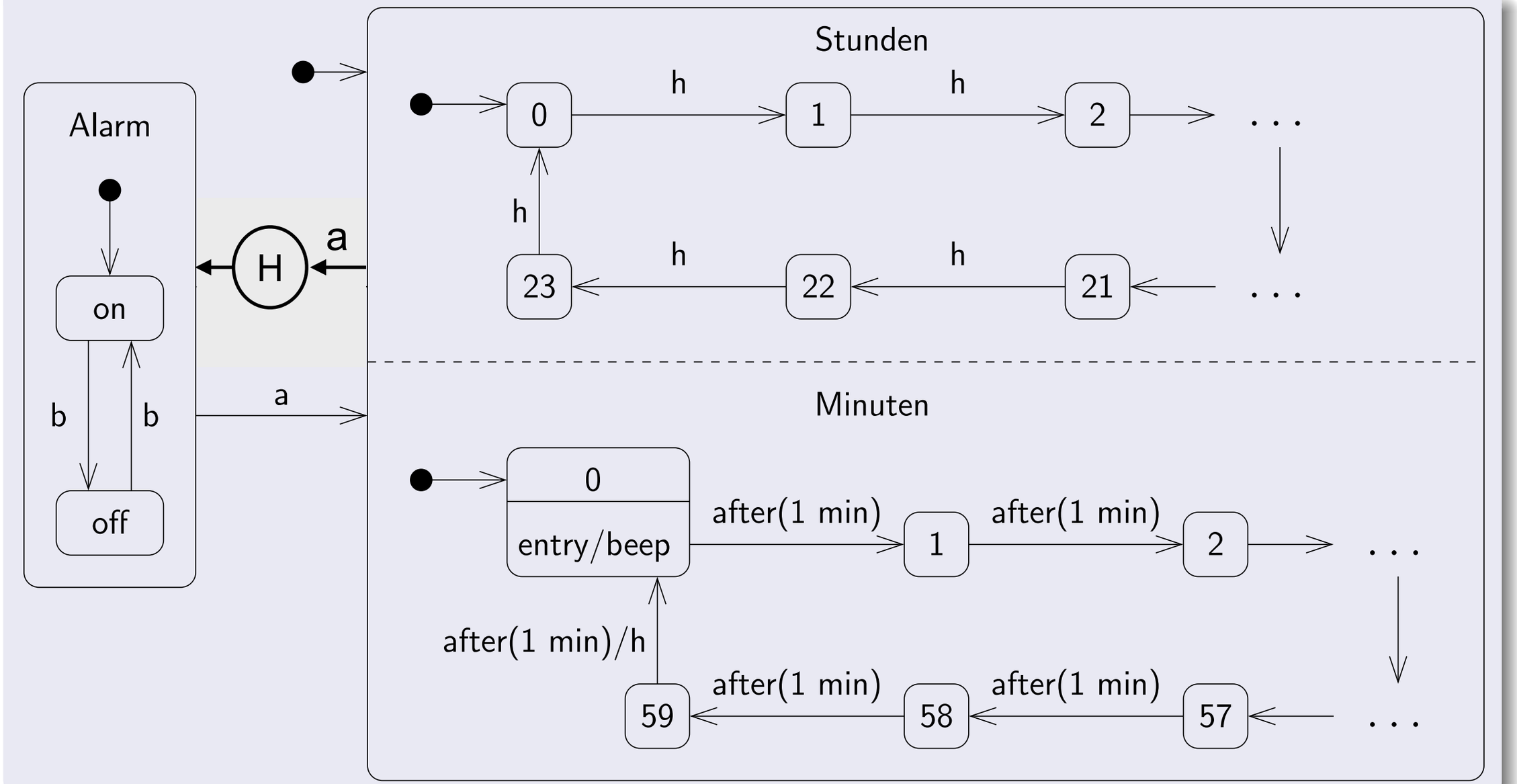
— pressing “a” show us the current state (“on” / “off”)
— then use “b” to toggle between “on” and “off”



It would be more intuitive if:

— pressing “a” show us the current state (“on” / “off”)
— then use “b” to toggle between “on” and “off”

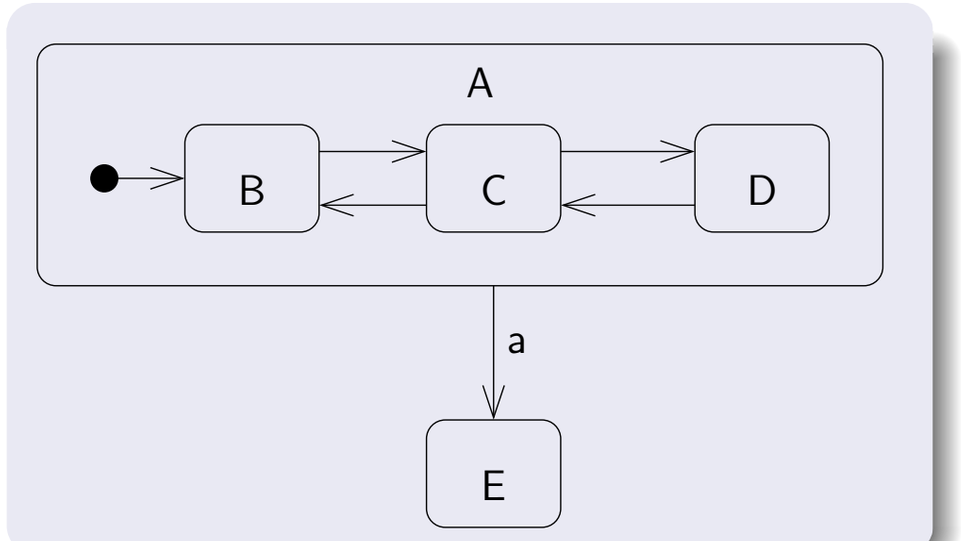
Which “history”,
when “a” is pressed
for the very first time?



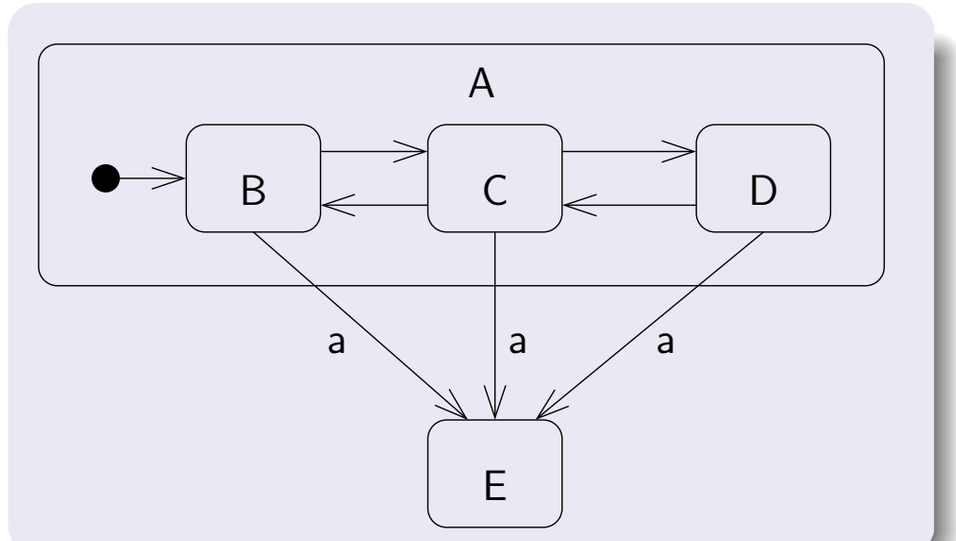
Question: construct an equivalent diagram without the small "Alarm-diagram" on the left.
— How many states does it have?

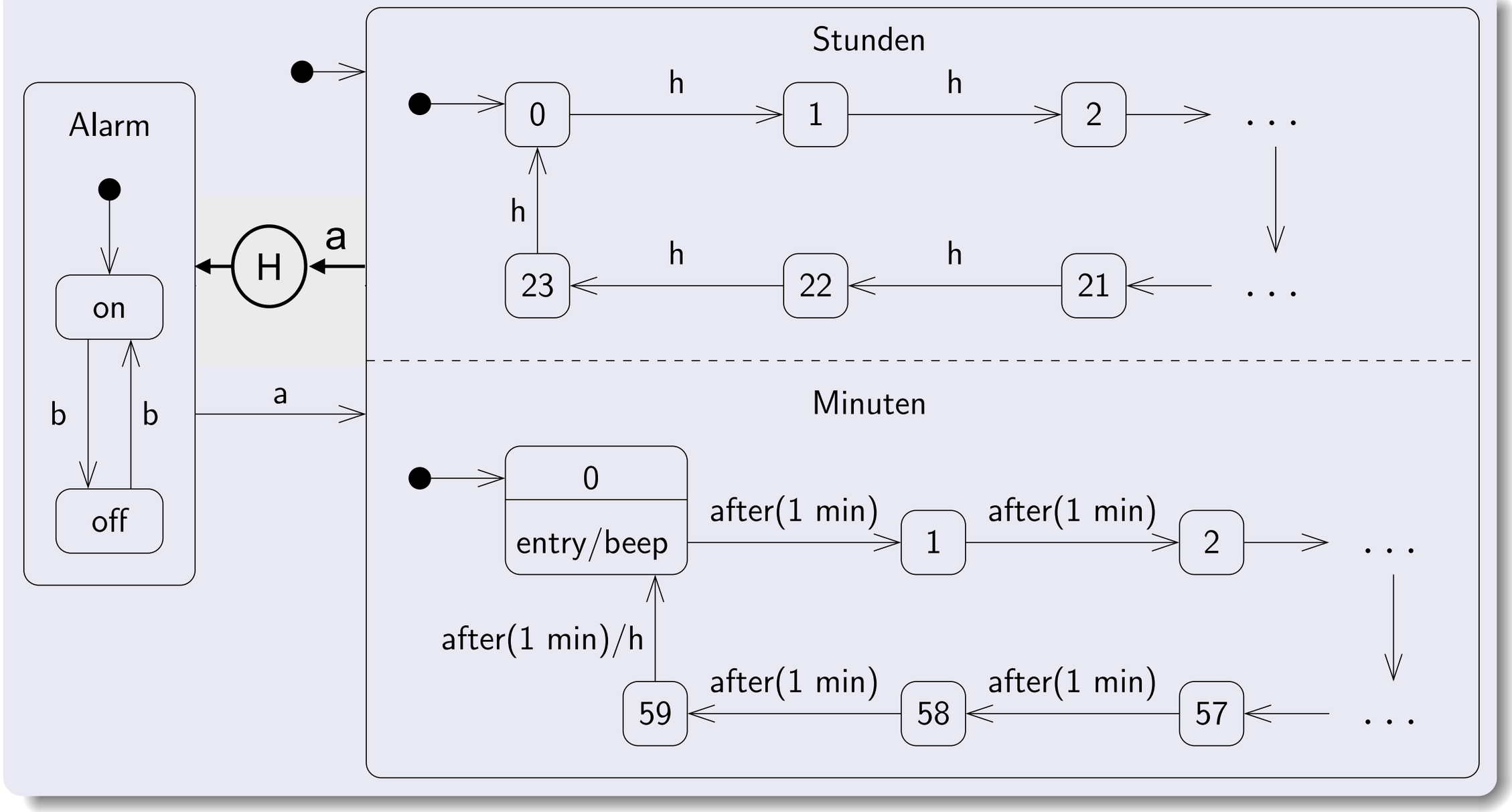
Zustandsdiagramme

Beispiel zu Austrittsmöglichkeiten:



entspricht

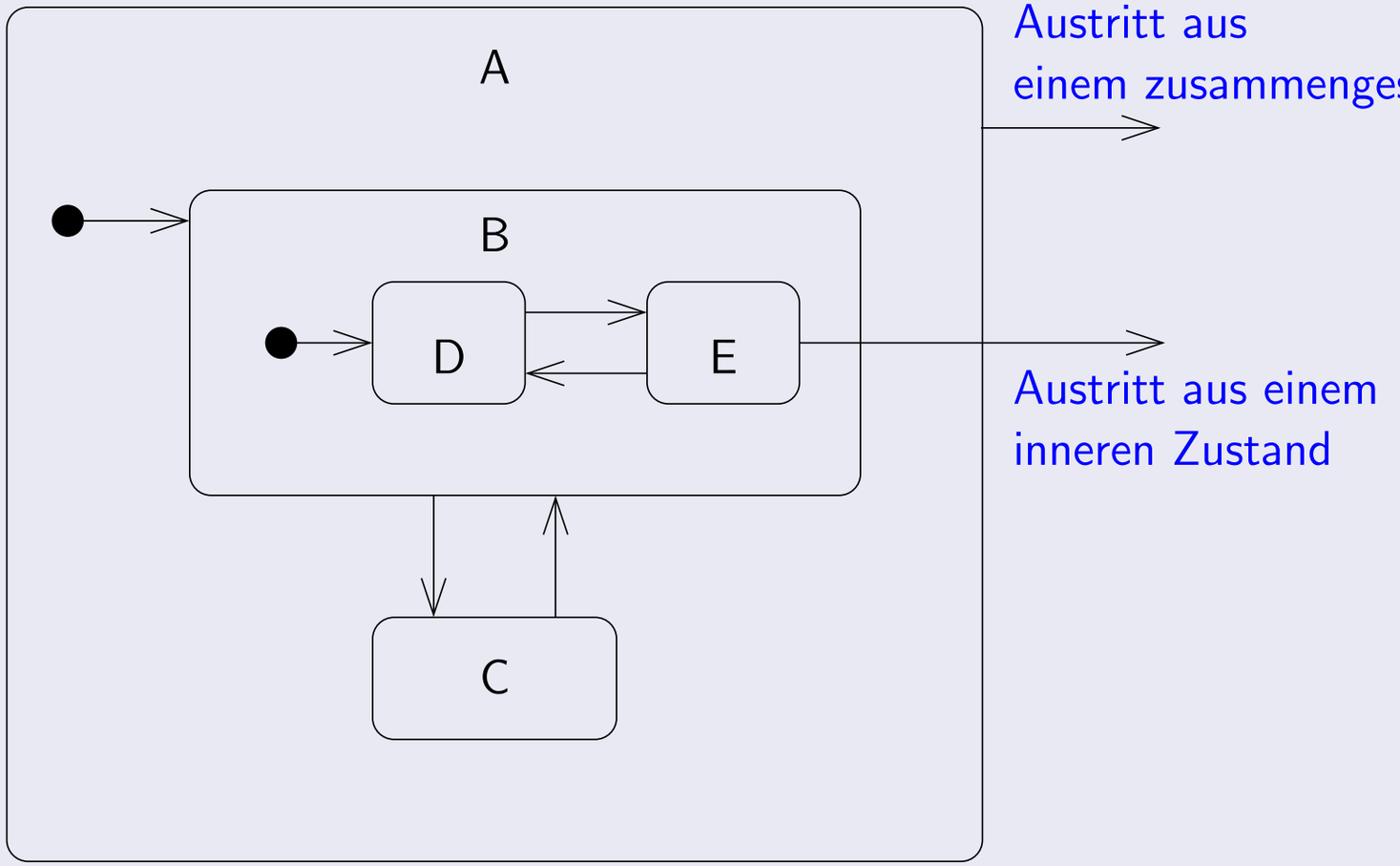




Question.
Transform the above automaton into an ordinary finite state automaton?
How many states does this automaton have?

Zustandsdiagramme

Austrittsmöglichkeiten aus einem Zustand (graphisch)



Zustandsdiagramme

Beschreibung der Austrittsmöglichkeiten:

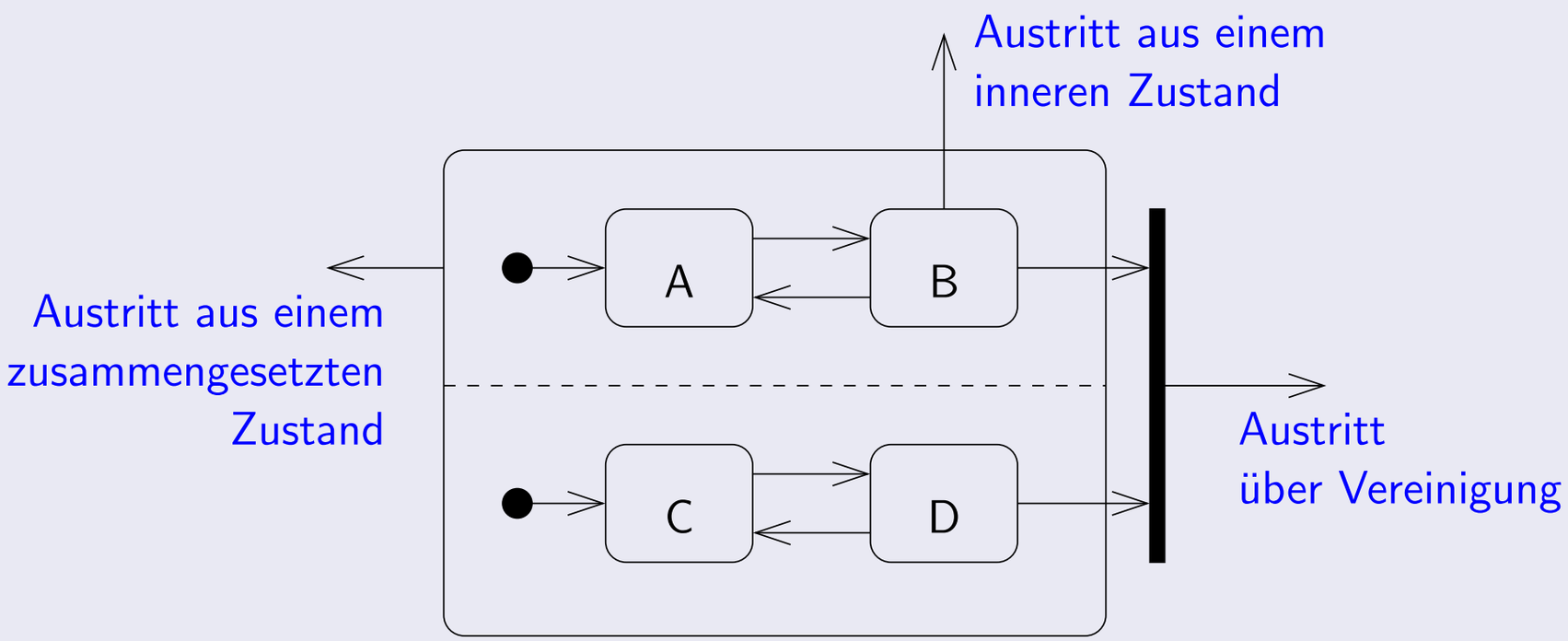
- Austritt aus einem zusammengesetzten Zustand: sobald das mit der Transition assoziierte Ereignis stattfindet, wird jeder beliebige (Unter-)Zustand von A verlassen.
- Austritt aus einem inneren Zustand: die Transition wird nur genommen, wenn man sich gerade im entsprechenden Zustand (hier: Zustand E) befindet (und das entsprechende Ereignis stattfindet).

Außerdem: Austritt über einen Austrittspunkt, über einen Endzustand oder Terminator (wird hier nicht behandelt).

Zustandsdiagramme

Auch für Austrittsmöglichkeiten müssen wir untersuchen, was sich bei Regionen ändert.

Austrittsmöglichkeiten bei Regionen (graphisch)



Zustandsdiagramme

Beschreibung der Austrittsmöglichkeiten bei Regionen:

- Austritt aus einem zusammengesetzten Zustand: dabei wird der zusammengesetzte Zustand verlassen, egal in welchen Unterzuständen man sich gerade befindet.
- Austritt aus einem inneren Zustand: der zusammengesetzte Zustand wird nur verlassen, wenn man sich in der jeweiligen Region in dem Zustand befindet, der durch den Pfeil verlassen wird. In den anderen Regionen kann man sich in beliebigen Zuständen befinden.

  (Austritt nur, wenn man sich in der ersten Region in B befindet.)

Zustandsdiagramme

Beschreibung der Austrittsmöglichkeiten bei Regionen (Fortsetzung):

- Austritt über Vereinigung: ähnlich wie bei Aktivitätsdiagrammen wird eine Vereinigung eingesetzt, um mehrere Regionen zusammenzuführen. Der zusammengesetzte Zustand kann nur verlassen werden, wenn man sich in den Zuständen befindet, von denen Pfeile in die Vereinigung hineinführen.

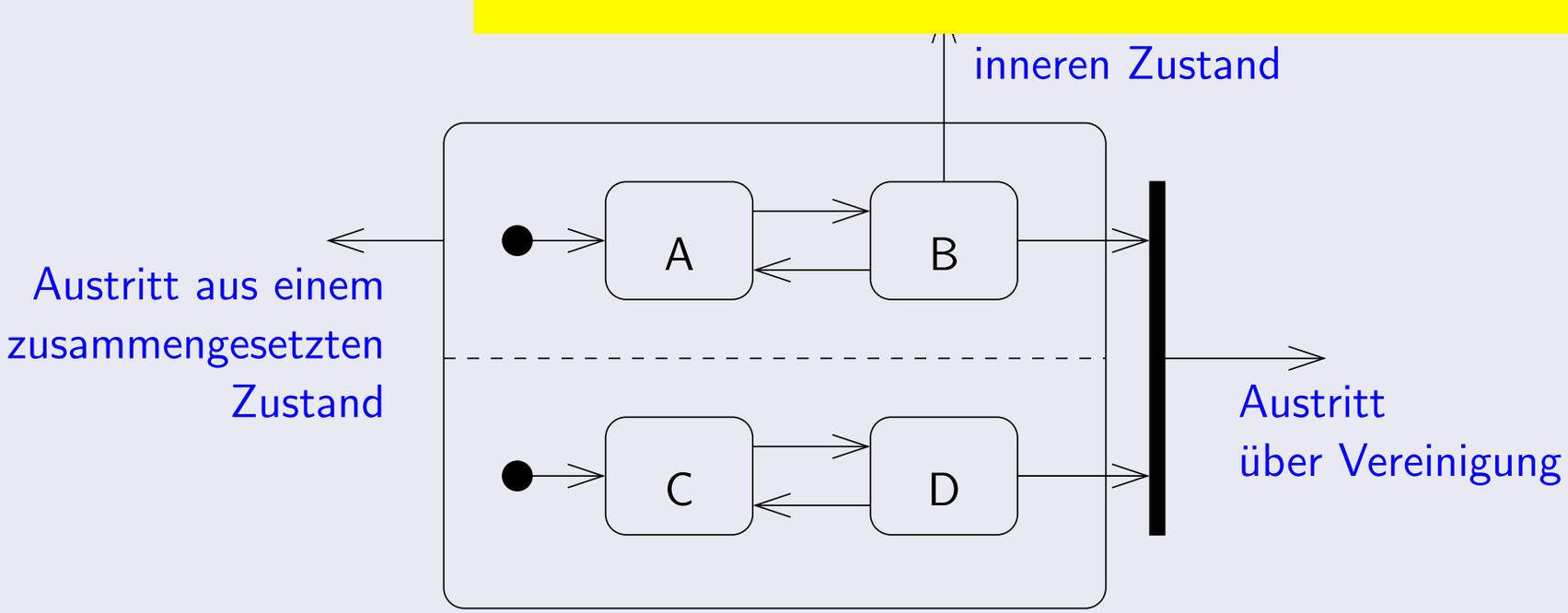
  (Austritt nur, wenn man sich in B und D befindet.)

Zustandsdiagramme

Auch für Austrittsmögl
bei Regionen ändert.

Frage wenn beide Austrittstransitionen
mit dem selben Label (event) markiert sind,
was passiert dann?

Austrittsmöglichkeiten

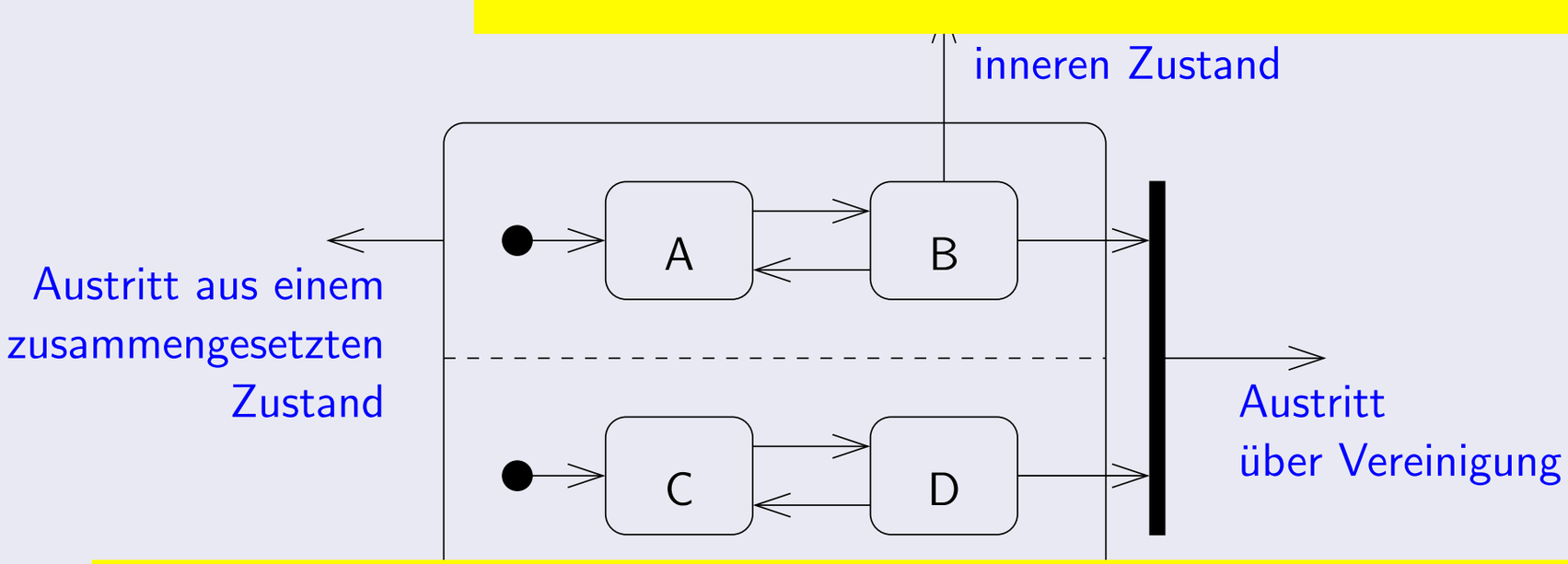


Zustandsdiagramme

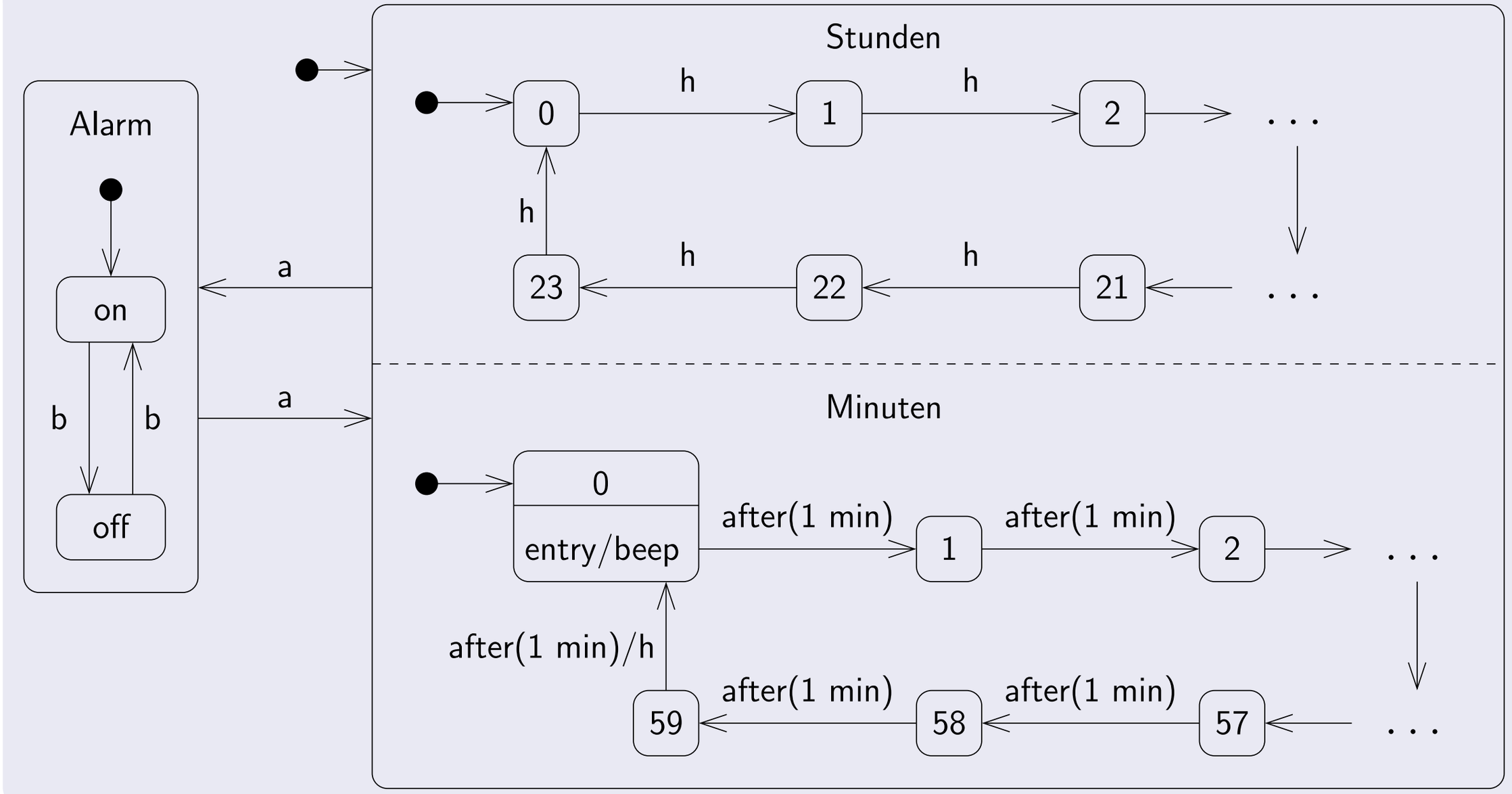
Auch für Austrittsmögl
bei Regionen ändert.

Frage wenn beide Austrittstransitionen
mit dem selben Label (event) markiert sind,
was passiert dann?

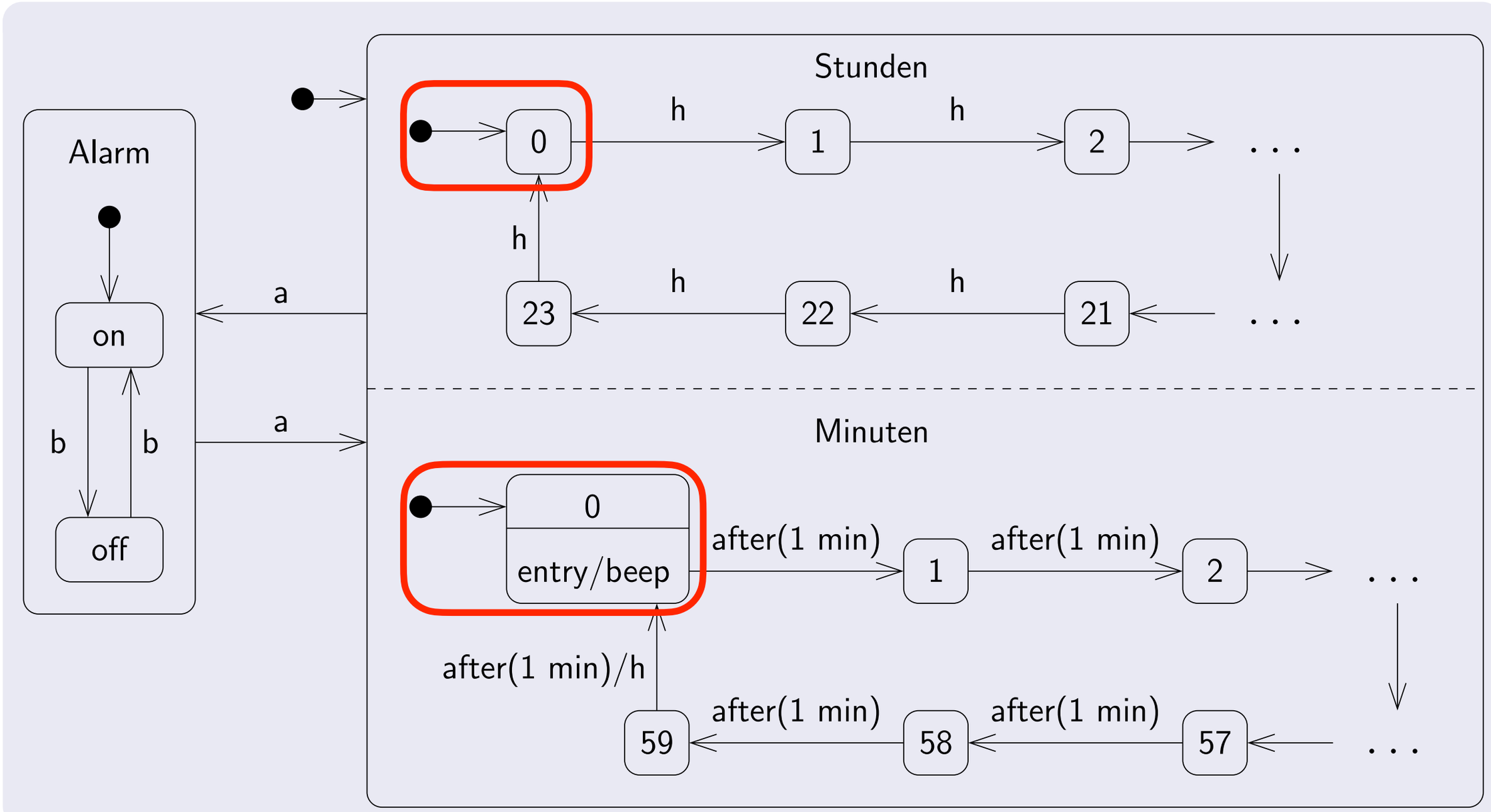
Austrittsmöglichkeiten



Frage läßt Harel nur deterministische Diagramme zu?



Frage: was passiert, wenn wir aus der Alarmeinstellung per a-Druck zurück in die Zeitanzeige kommen?



Frage: was passiert, wenn wir aus der Alarmeinstellung per a-Druck zurueck in die Zeitanzeige kommen?

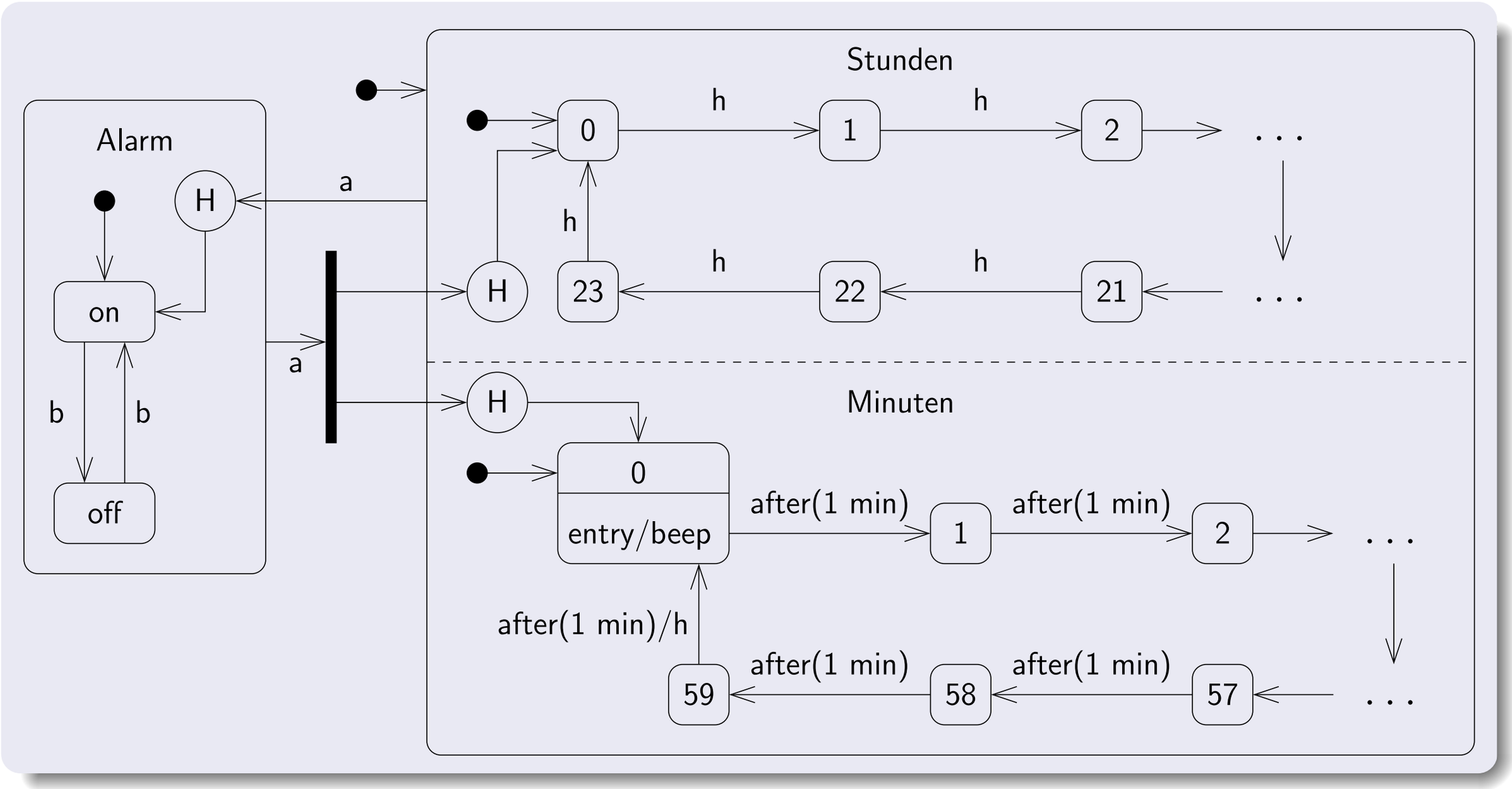
Standard Eintritt: Also wird die Zeit auf 0:00 Uhr gestellt!

Zustandsdiagramme

Es gibt ein weiteres Problem: wenn man aus der Alarmeinstellung zurückkehrt, ist die Zeit auf 0:00 zurückgesetzt!

Lösung: Verwendung des Eintritts über die (flache) Historie.
Da man im Fall der Zeitanzeige in zwei Regionen gleichzeitig eintreten muss, verwenden wir eine Gabelung.

Zustandsdiagramme

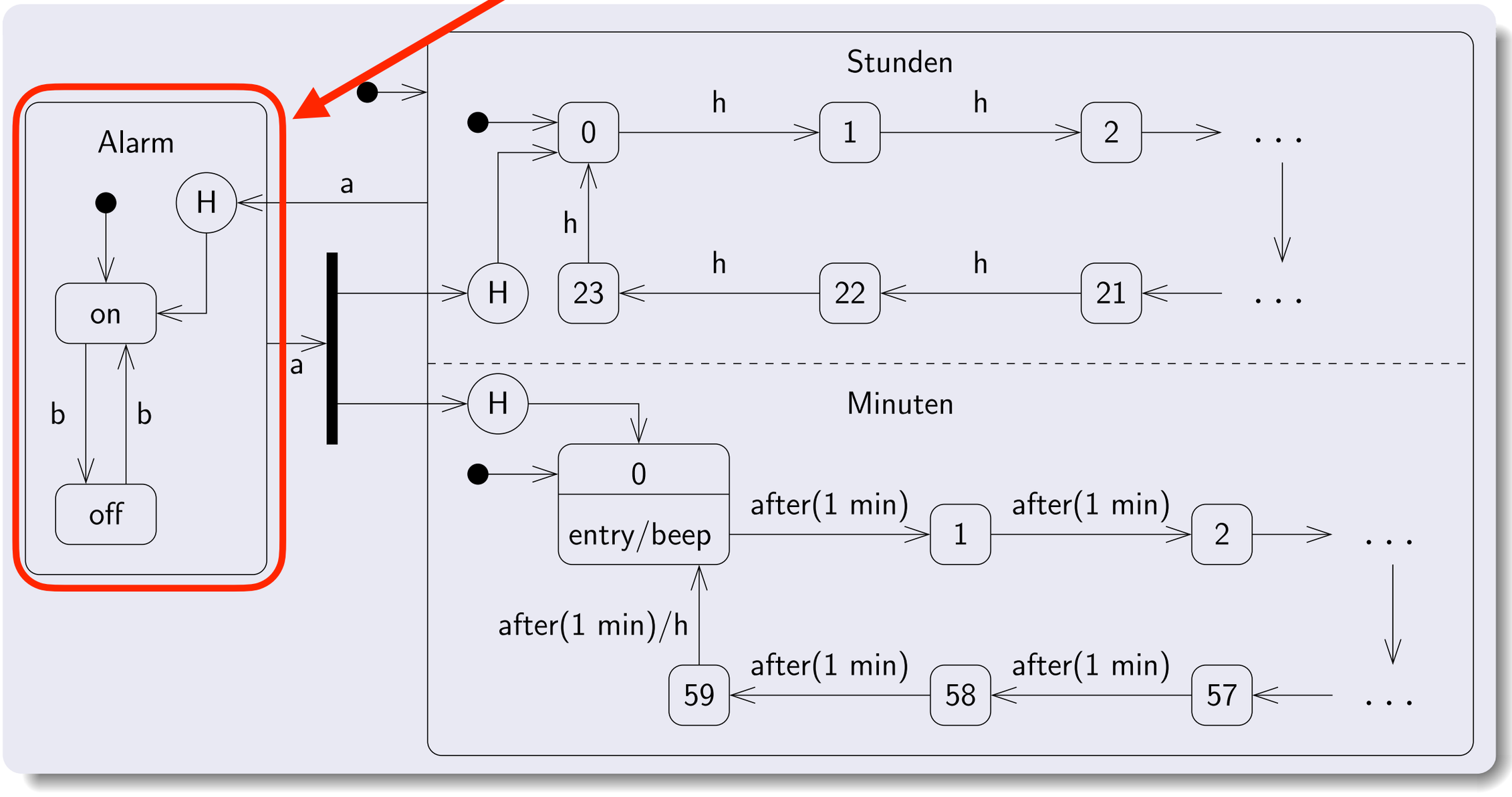


Zustandsdiagramme

Weiteres Problem: wenn man längere Zeit im Alarm-Zustand verbringt, so wird in dieser Zeit die Minuten-/Stundenanzeige nicht entsprechend aktualisiert. Das Time Event (after(1 min)) bezieht sich nur auf die Zeit, die seit dem Eintritt in den entsprechenden Zustand vergangen ist.

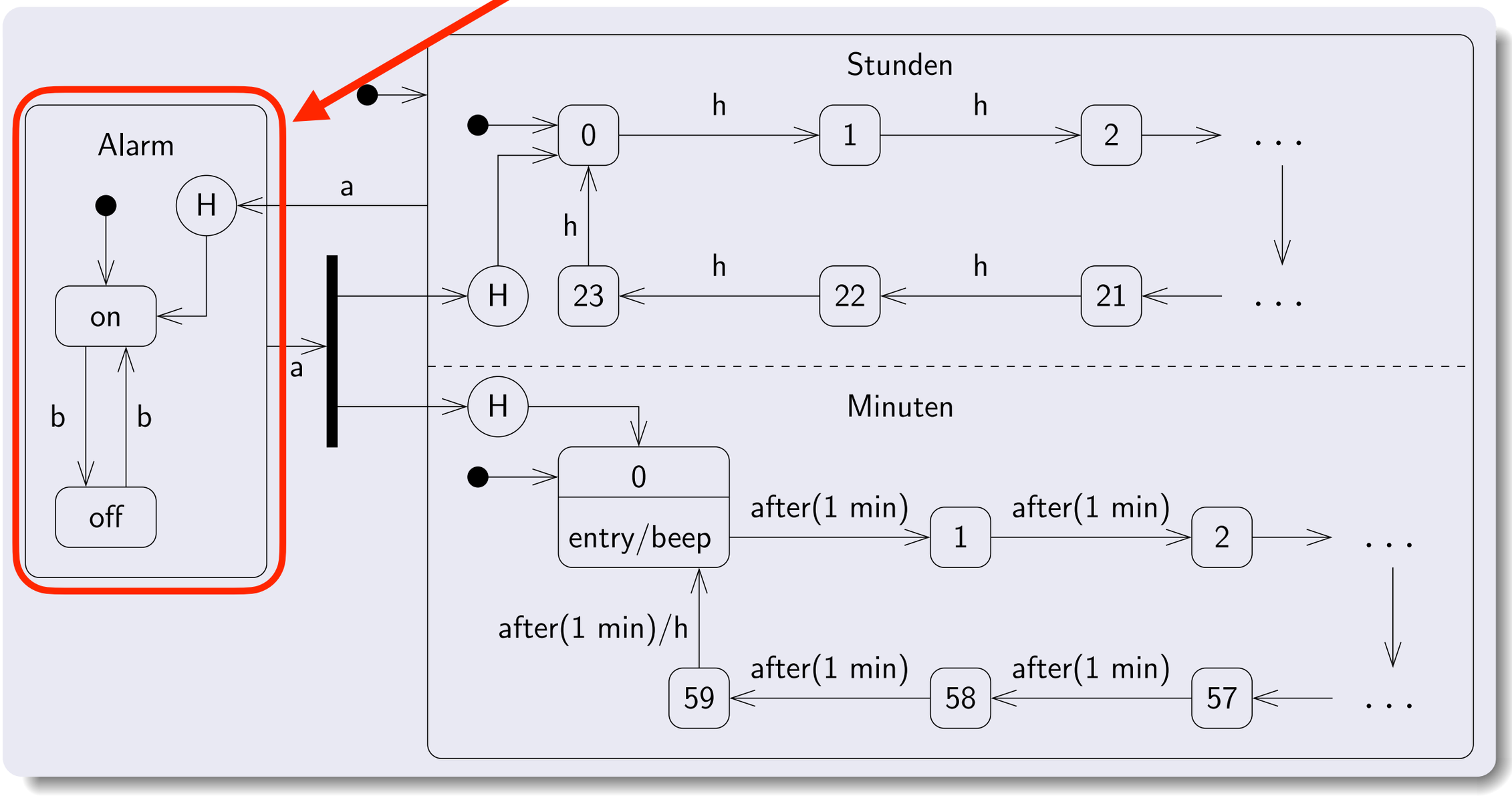
Zustandsdiagramme

die Zeit die wir hier verbringen, “verlieren wir”,
d.h. die Uhr geht dann “nach”

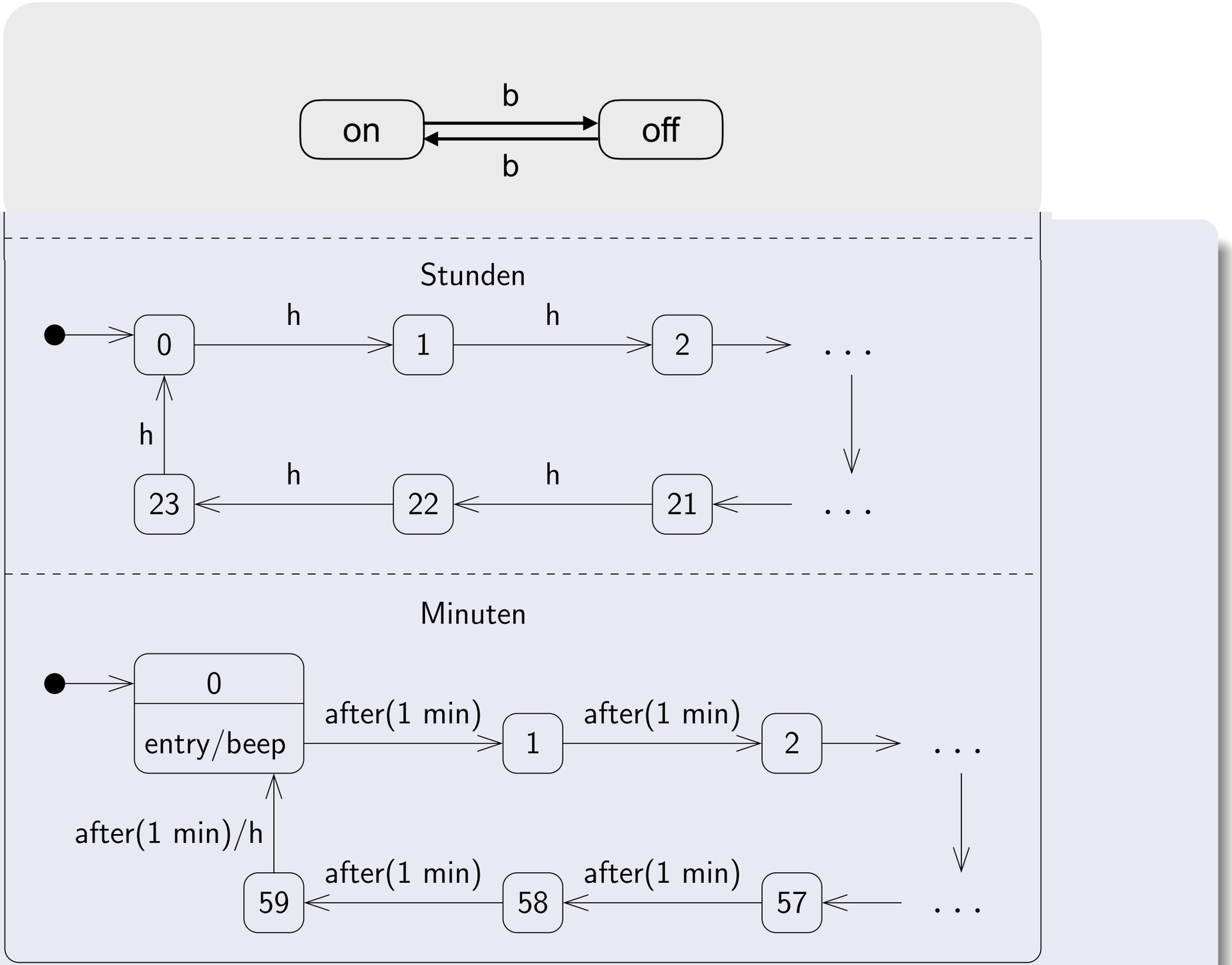


Wie kann man diese Problem lösen?

die Zeit die wir hier verbringen, “verlieren wir”,
d.h. die Uhr geht dann “nach”



All three automata must run in parallel at all times!



Zustandsdiagramme

Was fehlt noch?

Zustandsdiagramme

Was fehlt noch?

$\rightsquigarrow$ Beim Wechseln zwischen den Alarm-Zuständen (on,off) muss ein Flag (genannt al) gesetzt werden, um damit den beep-Effekt zu kontrollieren.

Dieses Flag muss mit Hilfe einer Bedingung im Minutenzustand 0 abgefragt werden.

Zustandsdiagramme

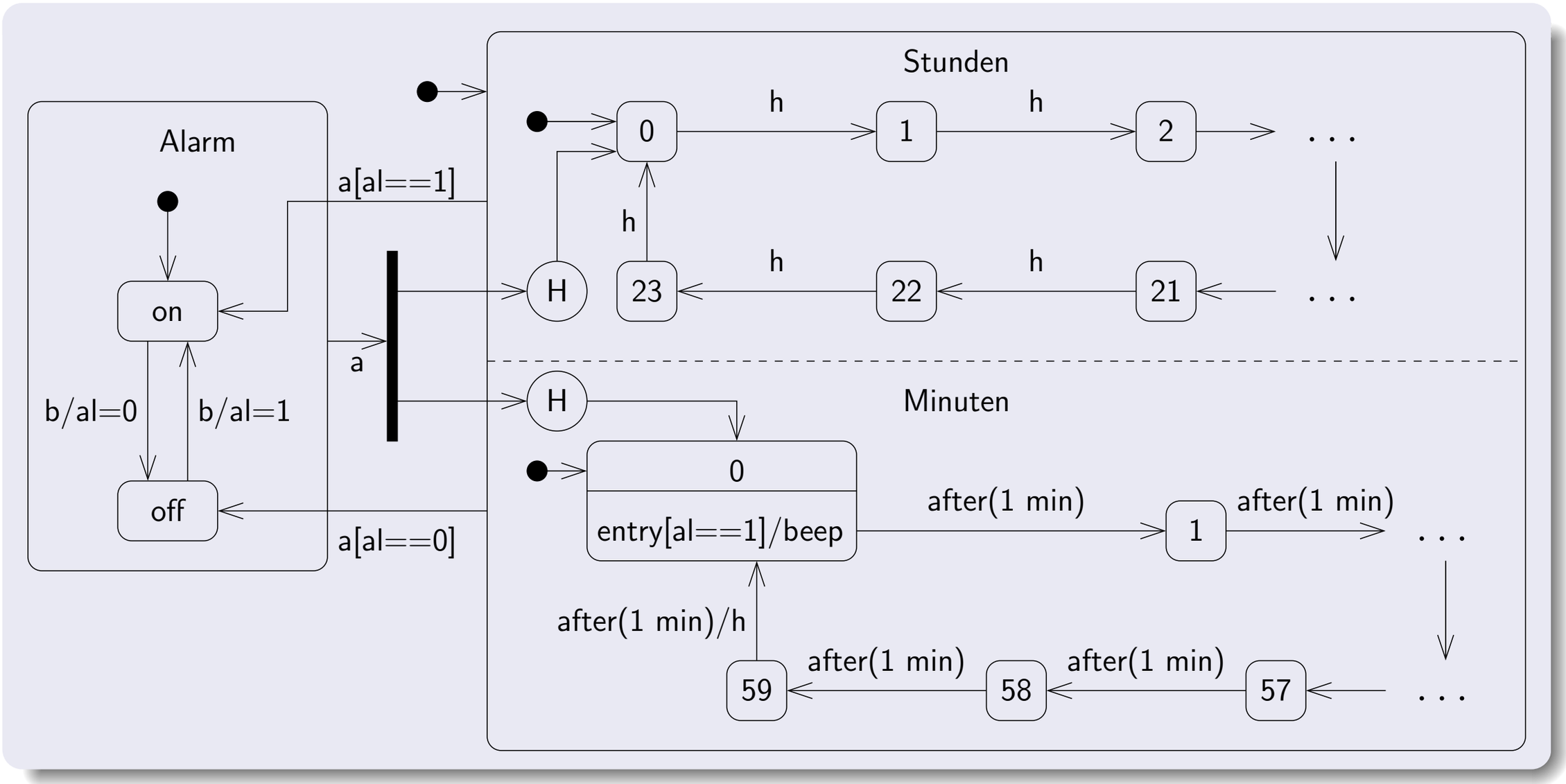
Was fehlt noch?

$\rightsquigarrow$ Beim Wechseln zwischen den Alarm-Zuständen (on,off) muss ein Flag (genannt al) gesetzt werden, um damit den beep-Effekt zu kontrollieren.

Dieses Flag muss mit Hilfe einer Bedingung im Minutenzustand 0 abgefragt werden.

Außerdem betreten wir den Zustand Alarm nun nicht mehr über die flache Historie, sondern fragen mit Hilfe von Bedingungen ab, wie al belegt ist. (Damit ist die Markierung des Anfangszustands eigentlich überflüssig geworden.)

Zustandsdiagramme



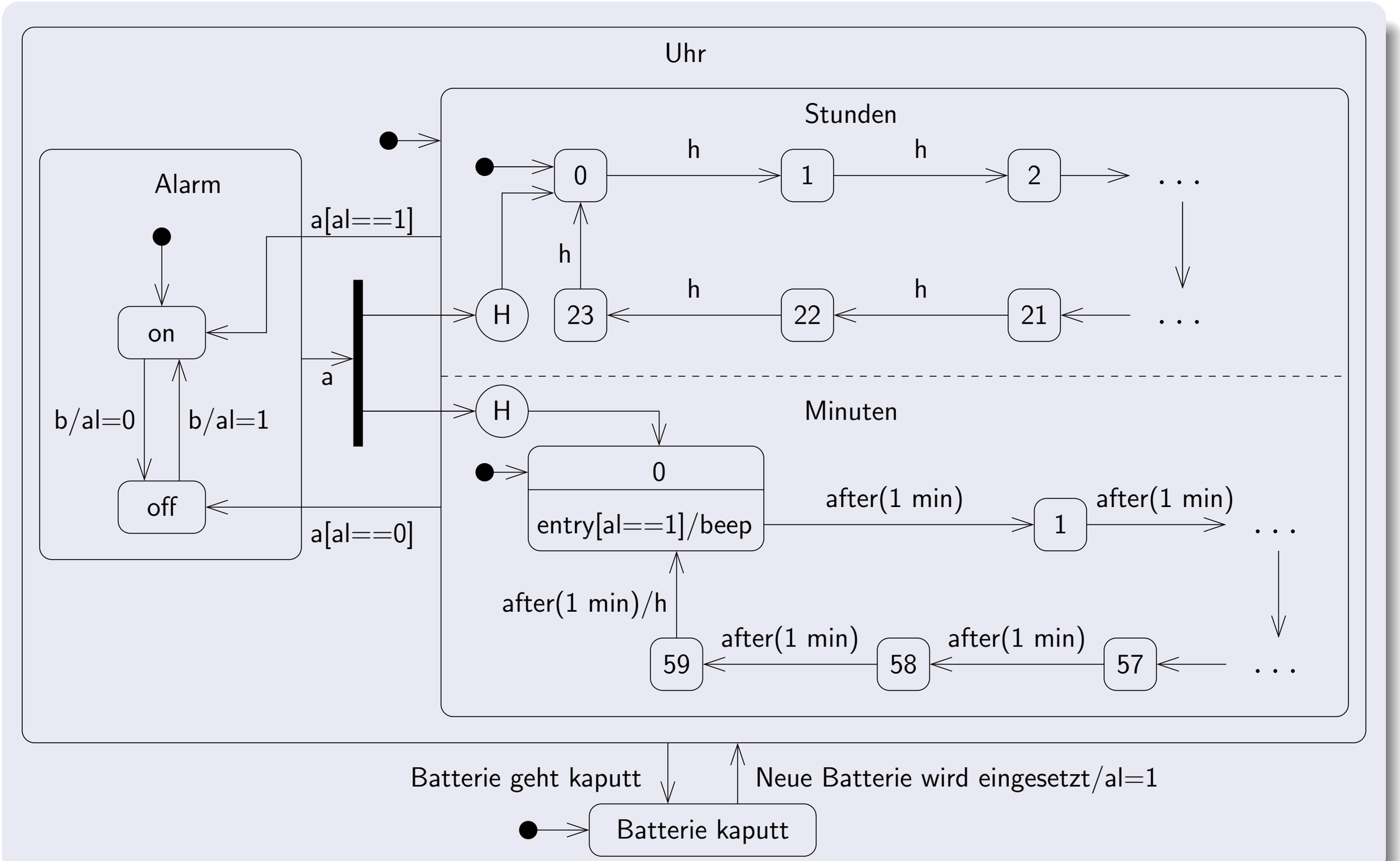
Zustandsdiagramme

Wir modellieren, dass die Batterie der Uhr kaputtgehen kann und gewechselt werden muss.

In diesem Fall will man den zusammengesetzten Zustand nicht über die flache oder tiefe Historie betreten! Es wird also hier tatsächlich die Zeit auf 0:00 zurückgesetzt.

Außerdem setzen wir das Flag al beim Einsetzen der Batterie zurück auf den Anfangswert 1.

Zustandsdiagramme



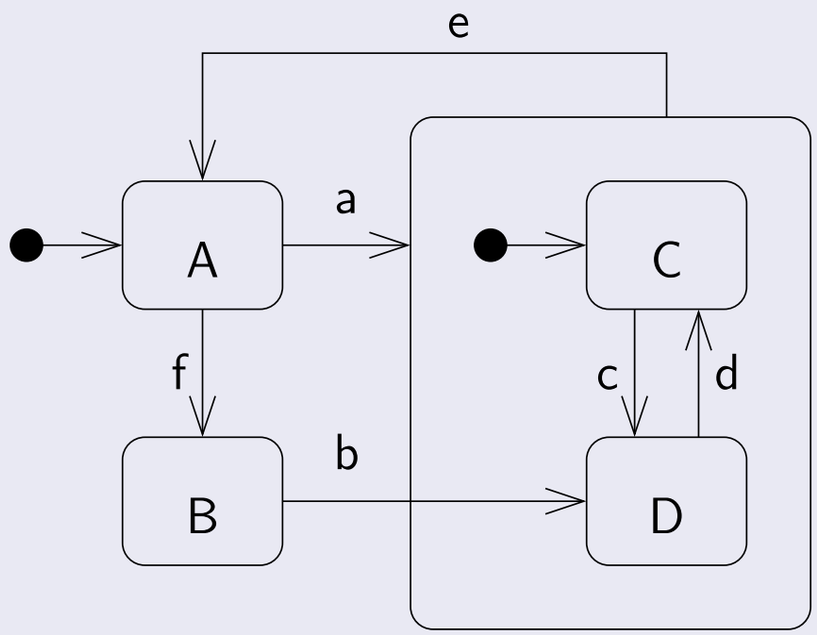
Zustandsdiagramme

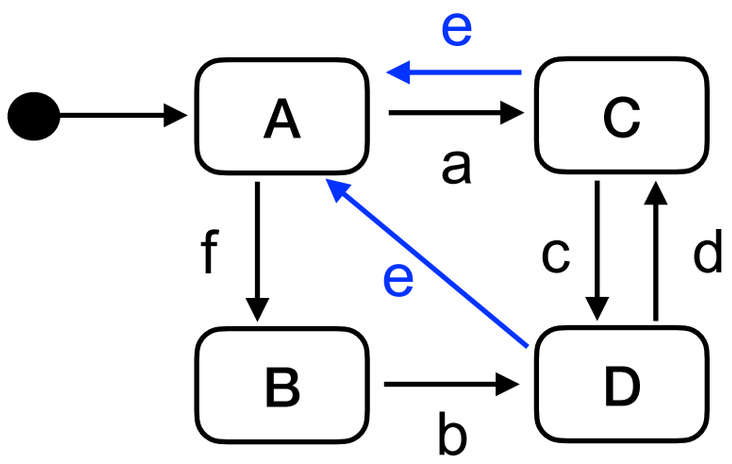
Ein großer Teil der Modellierungsmöglichkeiten von Zustandsautomaten dient dazu, Zustandsdiagramme kompakt und übersichtlich zu notieren.

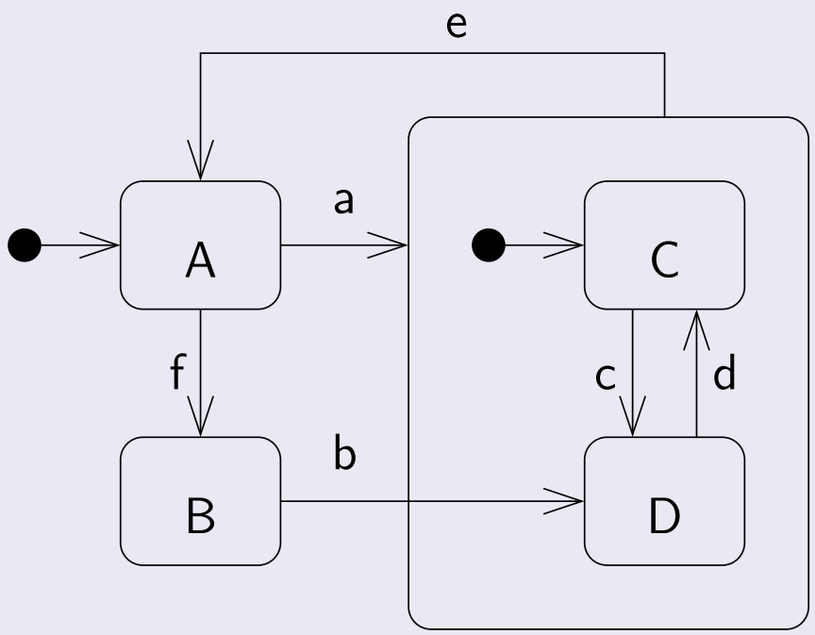
Oft kann man Zustandsdiagramme “flachklopfen” und zusammengesetzte Zustände auflösen, wodurch man äquivalente Zustandsdiagramme erhält, die die gleichen Übergänge erlauben. Dabei erhält man jedoch im allgemeinen mehr Zustände und/oder Transitionen.

Wir sehen uns einige Beispiele an . . .

Construct an equivalent finite-state automaton (without regions)

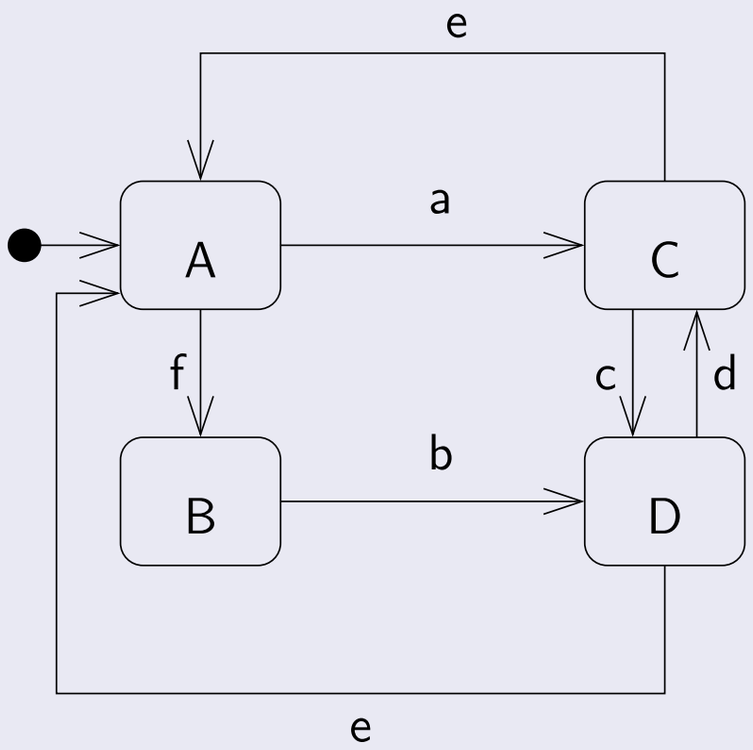



e
A
C
a
f
e
c
d
B
D
b


e
A
a
C
f
c
d
B
b
D

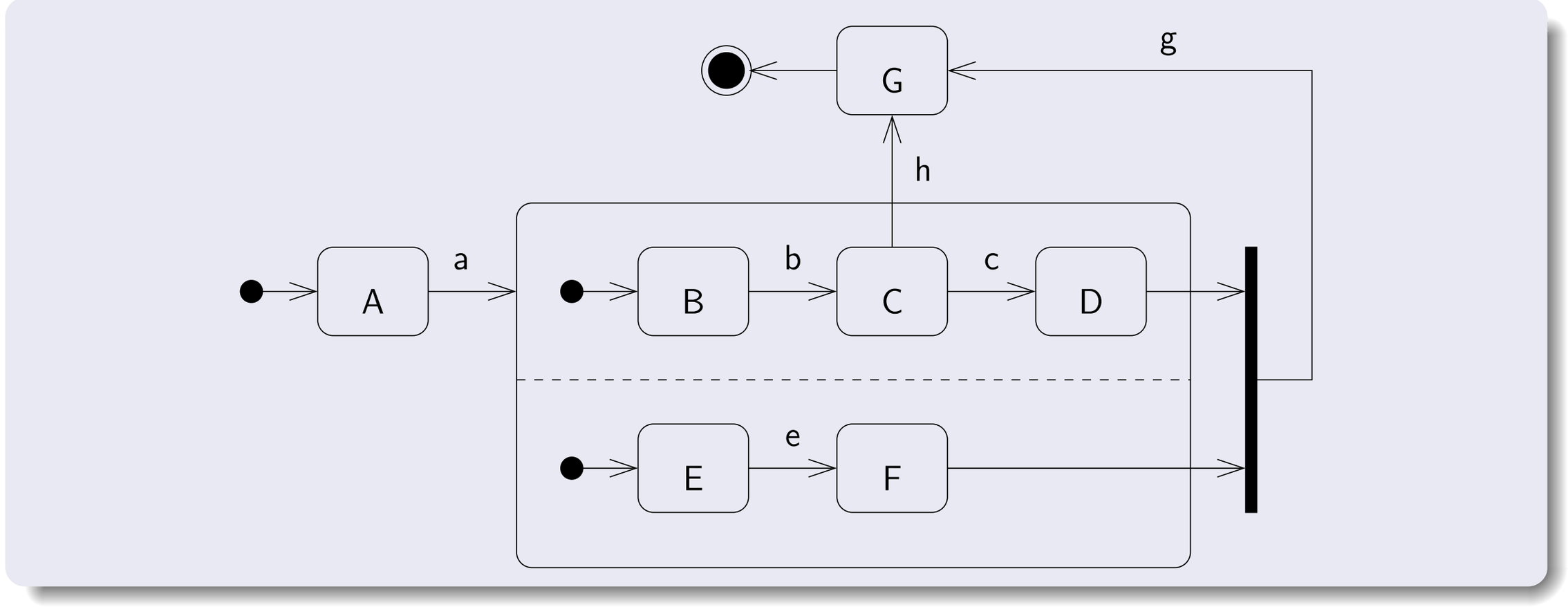
Zustandsdiagramme

Lösung zu Beispiel 1:

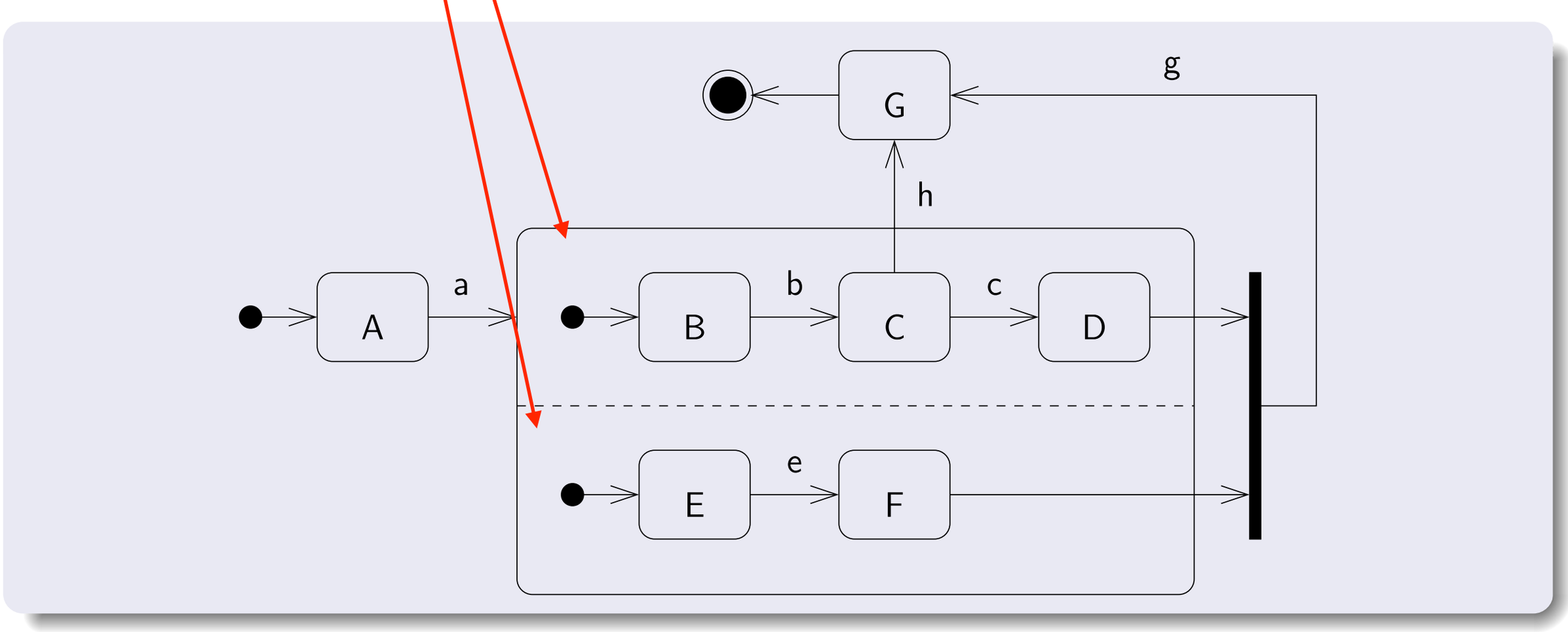


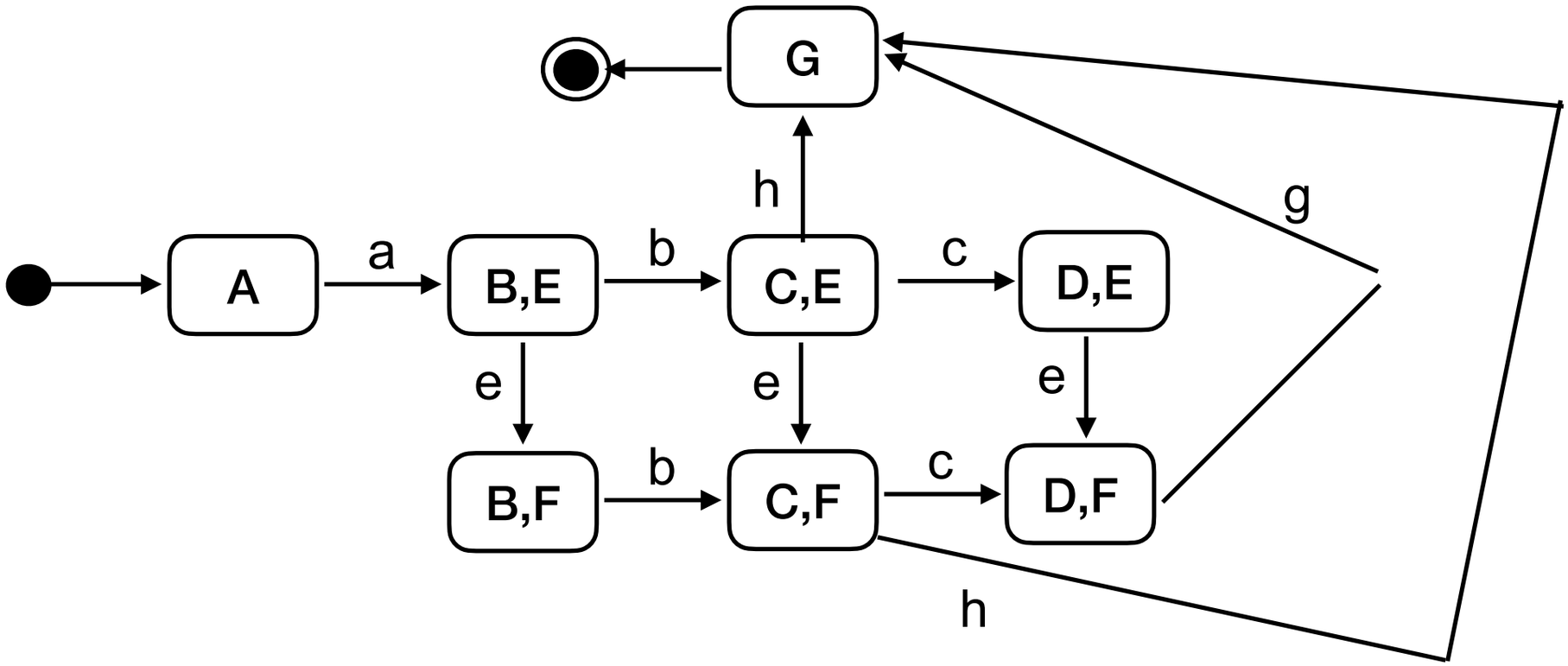
Idee: die Transition, die von dem zusammengesetzten Zustand wegführt, durch mehrere Transitionen ersetzen, die von den inneren Zuständen ausgehen.

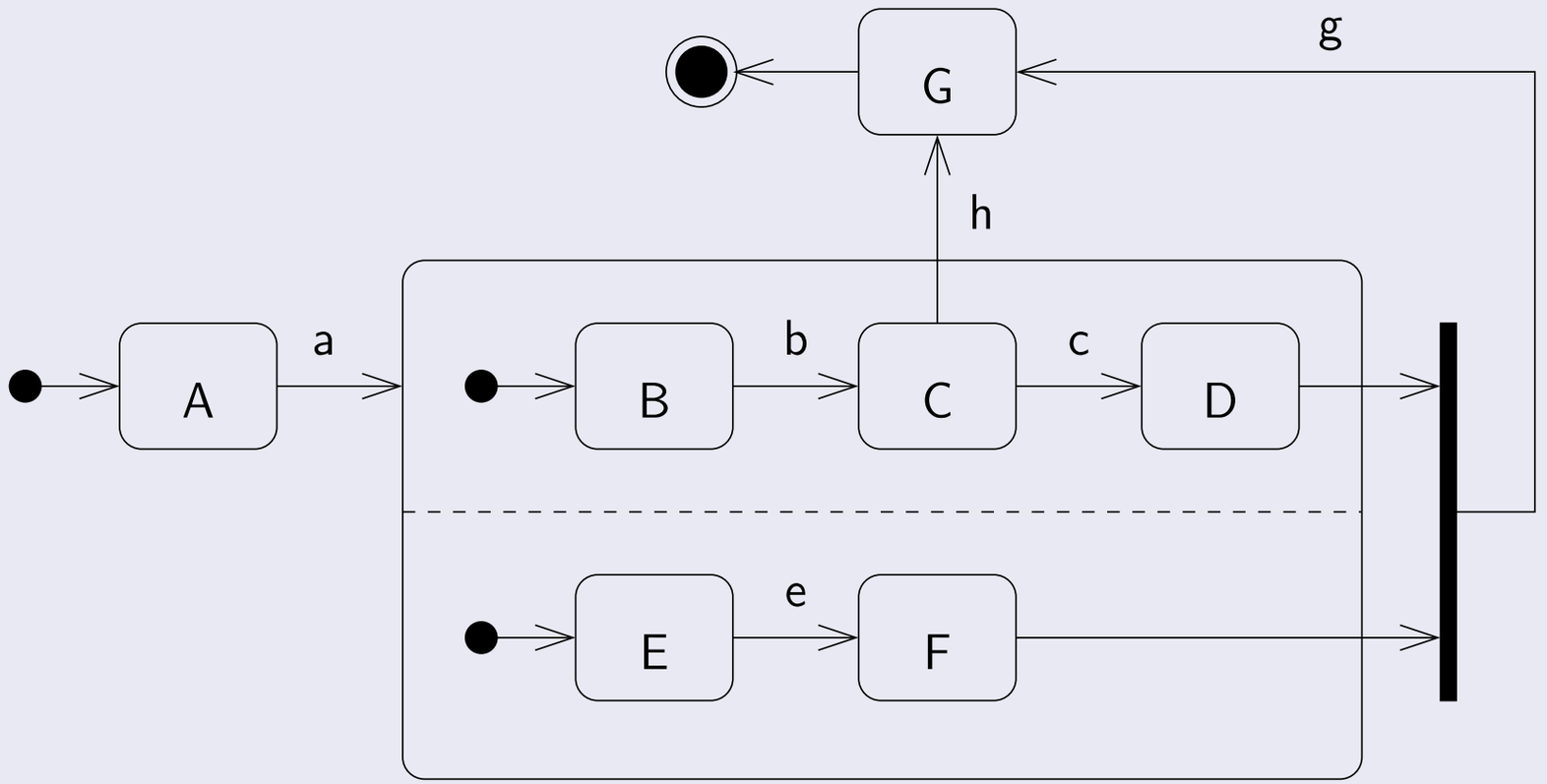
Construct an equivalent finite-state automaton (without regions)



Build a “product autmaton” from the two “region automata”.

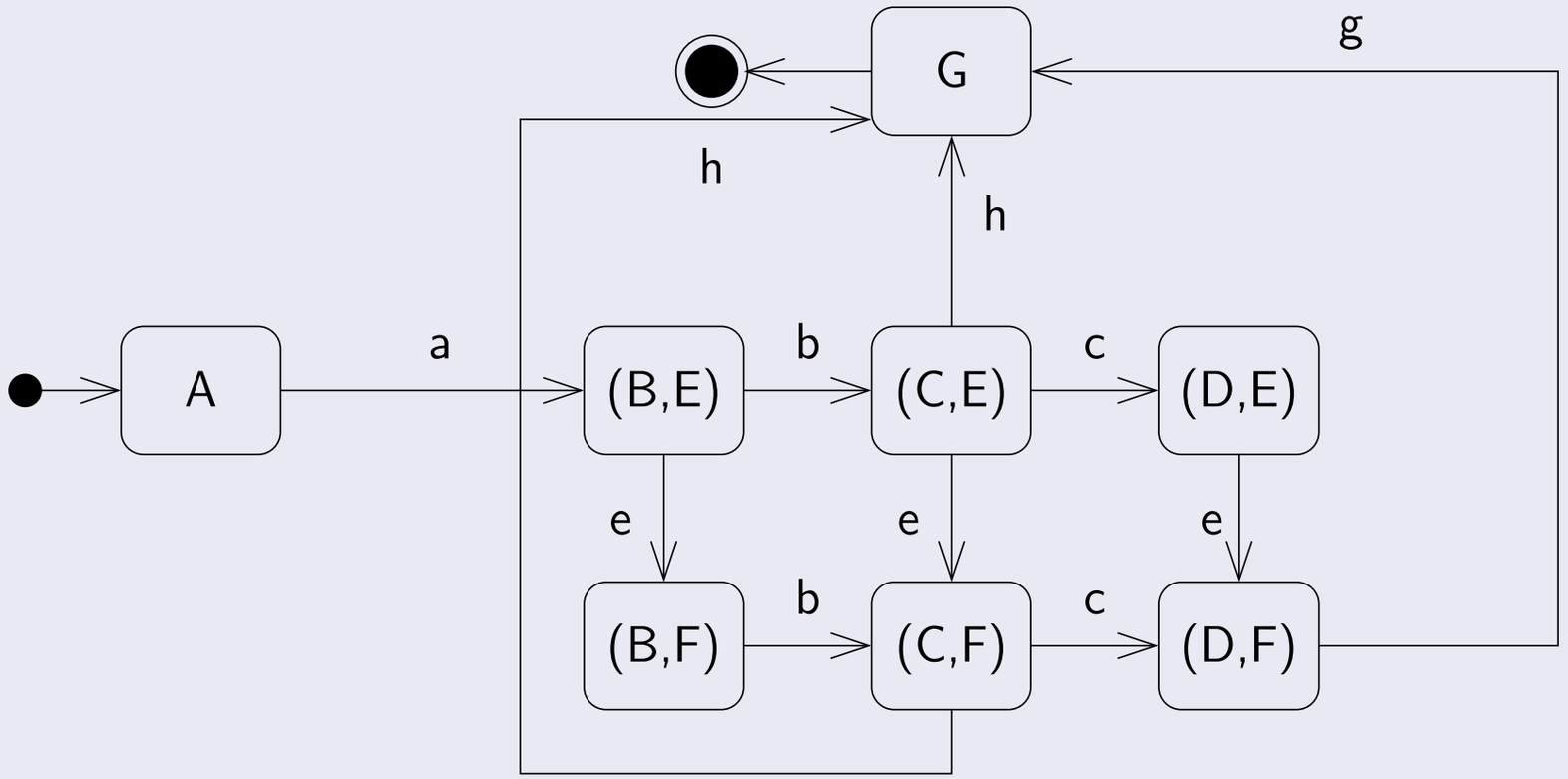



G
h
g
A
a
B,E
b
C,E
c
D,E
e
e
e
B,F
b
C,F
c
D,F
h


G
g
h
A
a
B
b
C
c
D
E
e
F

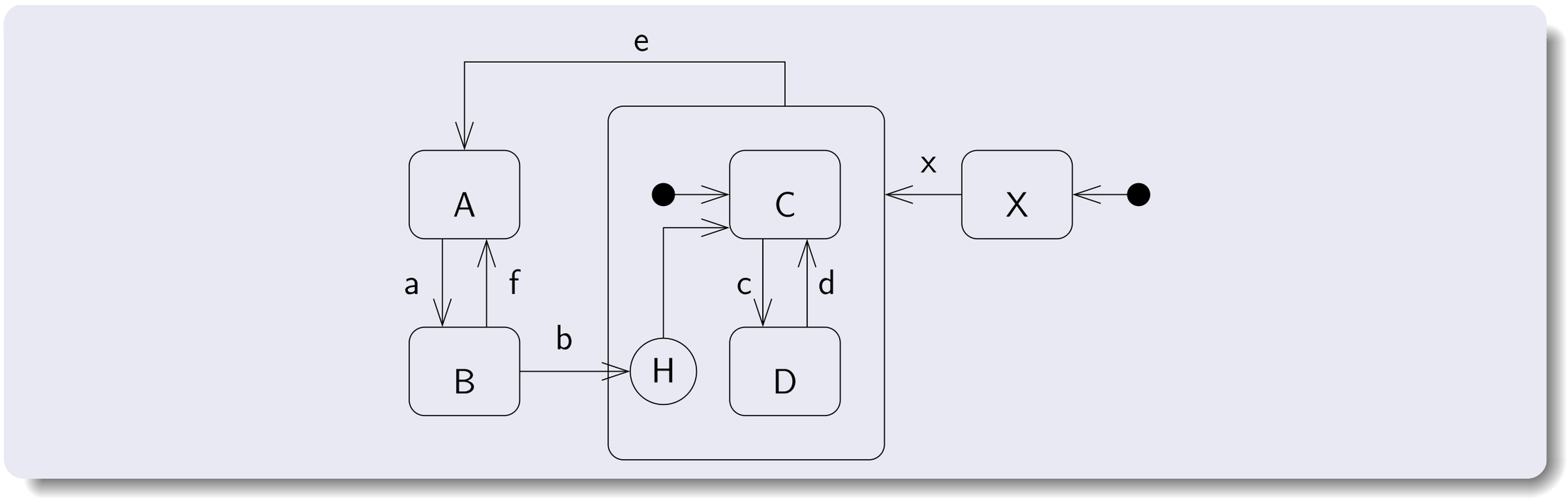
Zustandsdiagramme

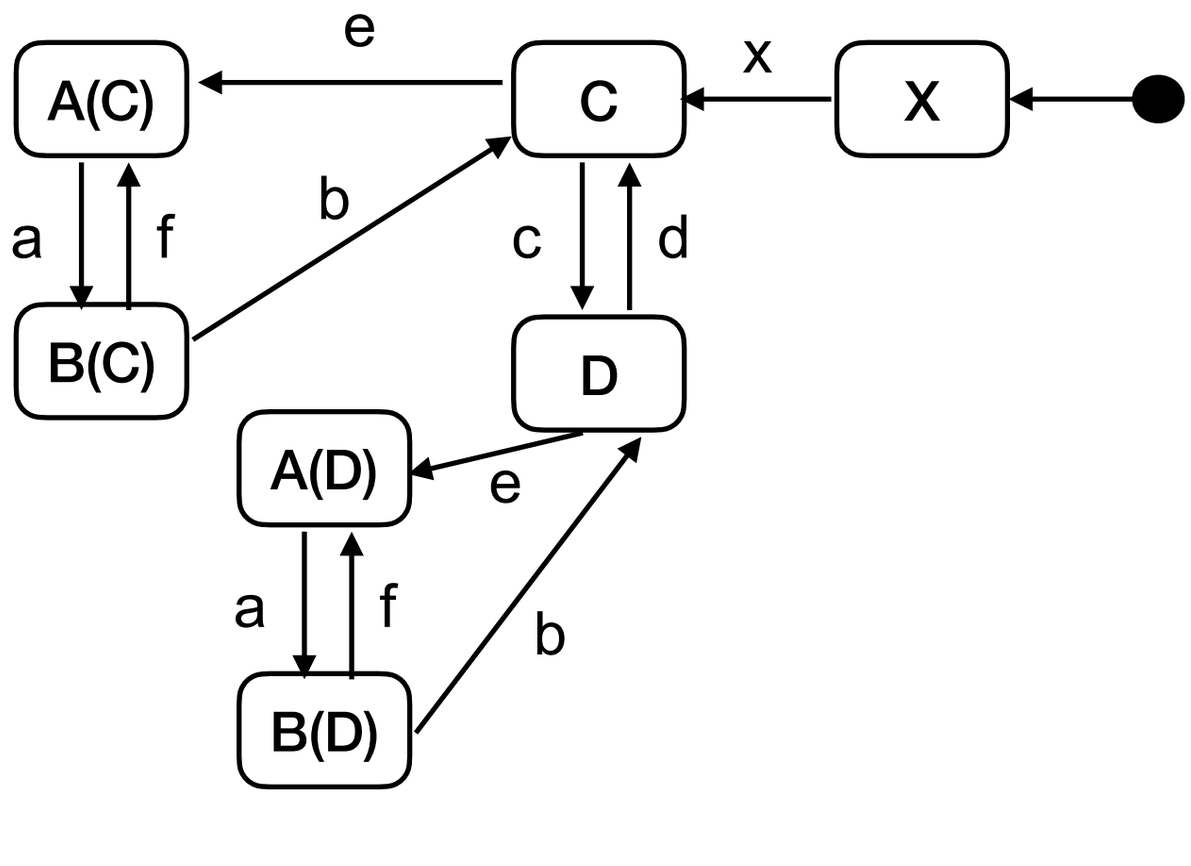
Lösung zu Beispiel 2:

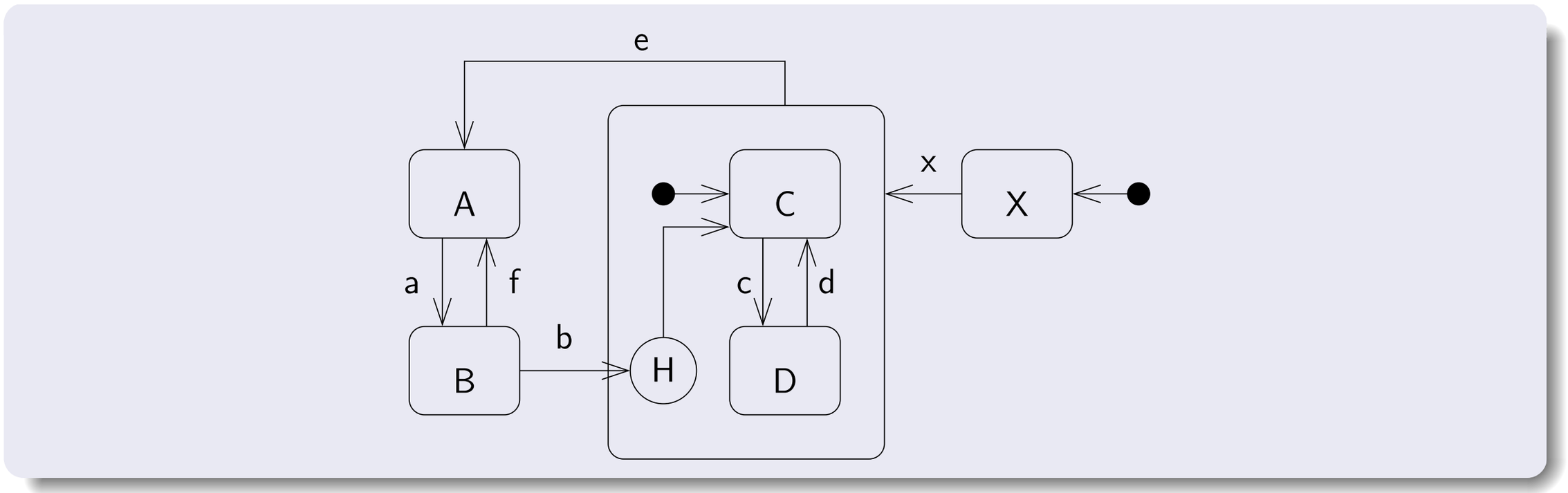


Idee: Kreuzprodukt der Zustandsmengen der Regionen bilden.

Construct an equivalent finite-state automaton (without regions)

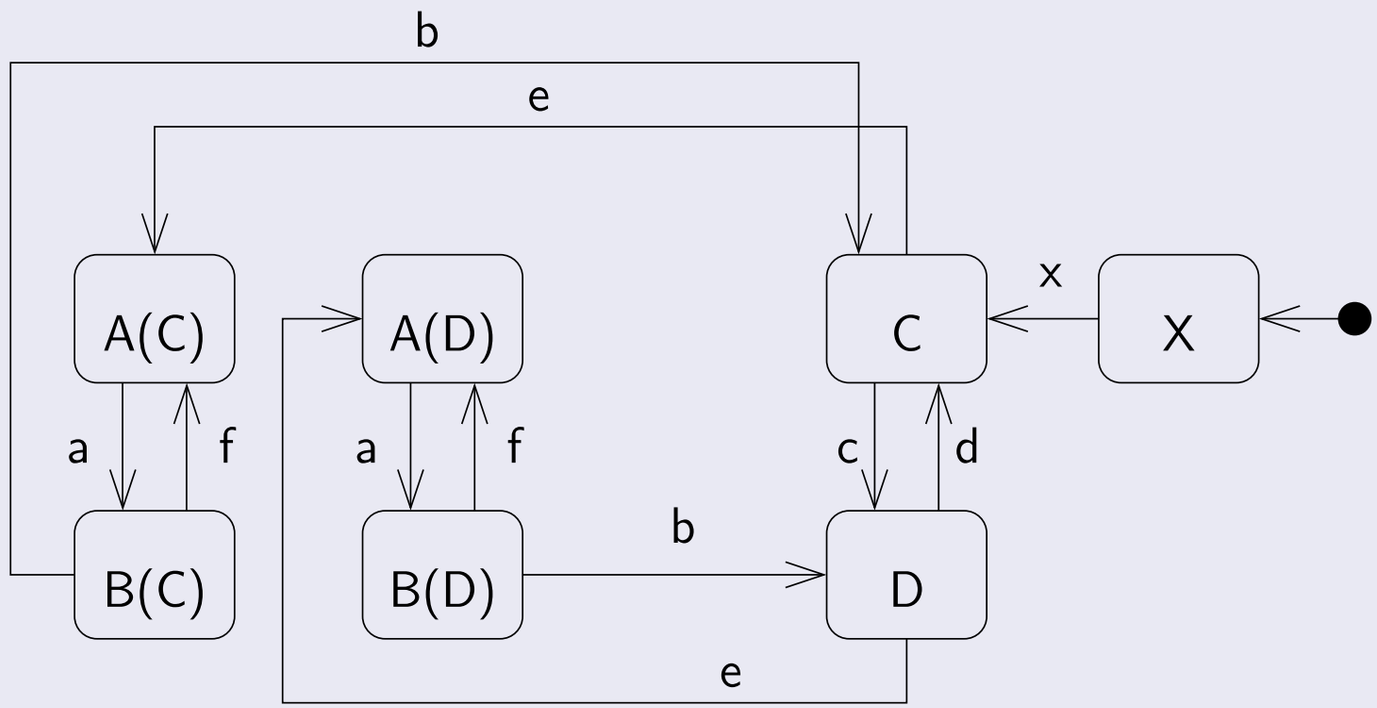



e
A(C)
C
x
X
a
f
b
c
d
B(C)
D
A(D)
e
a
f
b
B(D)


e
A
C
x
X
a
f
c
d
b
B
H
D

Zustandsdiagramme

Lösung zu Beispiel 3:



Idee: in den äußeren Zuständen muss man sich merken, aus welchem Zustand man den zusammengesetzten Zustand verlassen hat. Dies führt zu zusätzlichen Zuständen.

Zustandsdiagramme

Weitere Elemente von Zustandsdiagrammen:

- Unterscheidung zwischen verschiedenen Arten von Ereignissen: call event, signal event, change event, time event, any receive event
- Verzögern und Ignorieren von Ereignissen
- Entscheidungen und Kreuzungen
- Rahmen und Wiederverwendung von Zustandsdiagrammen

Außerdem: Generalisierung und Spezialisierung von Zustandsdiagrammen

Zustandsdiagramme

Zusammenfassung:

Nach Harel werden Zustandsdiagramme bzw. deren Eigenschaften durch folgende “Formel” beschrieben:

UML-Zustandsdiagramme =
Zustandsübergangsdiagramme + Tiefe + Orthogonalität
(+ Broadcast-Kommunikation)

Dabei steht “Orthogonalität” für die Parallelität, die durch Regionen erreicht werden kann.

Kurs

Datenbankgru
und Mod

S
ität Bremen
emen.de

ersemester 2023

19.6.2023

Vorlesung 8: Zustandsdiagramme
= State Charts