Esterel-Konstrukt: if Anweisung in Bezug auf Signale

- Durch Verwendung der if- Anweisung kann auch die Existenz eines Signals geprüft werden.
- **Syntax**:

```
if Signal-Name then
    Body-1
else
    Body-2
```

- **Semantik**: Bei Start dieser Anweisung wird geprüft, ob das Signal `Signal-Name` verfügbar ist. Ist es verfügbar, so wird der Code von `Body-1` ausgeführt, anderenfalls von Body-2. Innerhalb der Anweisung `if` kann auch entweder der `then Body-1` oder der `else Body-2` -Teil weggelassen werden.

Fragen zur letzten Vorlesung

- Wieso wird Rekursion in Esterel nicht unterstützt?
- Wie funktioniert die Umsetzung des Await-Statements?

```
module Temperature:
input IN1,IN2;
output OUT1,OUT2;
    loop
       await IN1; emit OUT1;
    end loop;
    ||
    loop
       await IN2; emit OUT2;
    end loop;
```

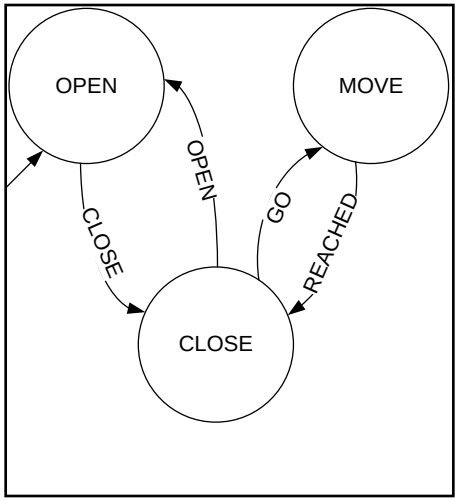
Exkurs: Automaten zur Modellierung von reaktiven Systemen

Automaten im Kontext von reaktiven Systemen

- Durch graphische Darstellung (z.B. Automaten) kann die Verständlichkeit des Codes stark verbessert werden.
- Ein reaktives System kann durch die zyklische Ausführung folgender Schritte beschrieben werden:
  1. Lesen der Eingangssignale
  2. Berechnung der Reaktionen
  3. Auslösen der Ausgangssignale
- Die zyklische Ausführung kann im Automatenmodell wie folgt interpretiert werden:
  1. Lesen der Eingabe im aktuellen Zustand
  2. Berechnen der Zustandsübergangsfunktion und ggfs. Zustandswechsel
  3. Erzeugung von Ausgangssignalen (abhängig von alten Zustand und gelesenem Eingangssignal)
- Die Synchronitätshypothese bedeutet im Bezug auf den Automaten der im Vergleich zur Zeit für Änderungen der Umgebung eine vernachlässigbare Zeit für die Berechnung der Zustandsübergange und Ausgabefunktion.
- Pro Runde wird im Allgemeinen (1 Ausnahme) genau ein Zustandsübergang berechnet.

Endliche Automaten

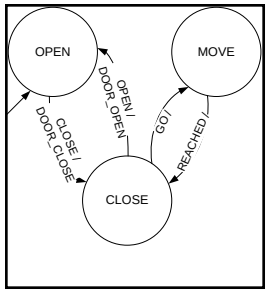
- Ein klassischer endlicher Automat (Q, $\sum$, δ, S, F) ist eine endliche Menge von Zuständen und Zustandsübergängen mit:
  - endlicher Menge von Zuständen Q
  - endliches Eingabealphabet $\sum$
  - Übergangsfunktion δ : Q x $\sum \rightarrow$ Q
  - Endlicher Menge von Startzuständen S
  - Menge von Endzuständen F $\subseteq$ Q



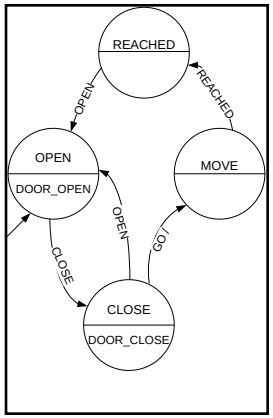
- Um die Verständlichkeit zu verbessern, werden nur deterministische Automaten modelliert, also |S|=1 und $\delta(q,\alpha)=q' \wedge \delta(q,\alpha)=q'' \Rightarrow p'=q''$
- Problem bei der klassischen Definition von Automaten: Ausgaben können nicht modelliert werden.

Automaten mit Ausgaben

- Mealy- und Moore-Automaten unterstützen die Ausgabe von Signalen
- Die Ausgaben von Mealy-Automaten $(Q,\sum,\Omega, \delta, \lambda, S,F)$ sind dabei an die Übergänge gebunden mit
  - Q, $\sum$, δ, S, F wie bei klassischem Automat
  - Ausgabealphabet Ω
  - Ausgabefunktion λ: Q x $\sum \rightarrow \Omega$



- Bei Moore-Automaten $(Q,\sum,\Omega, \delta, \lambda, S,F)$ ist die Ausgabe dagegen von den Zuständen abhängig:
  - Q, $\sum$, Ω, δ, S, F wie bei Mealy-Automat
  - Ausgabefunktion λ: Q $\rightarrow \Omega$
- Moore- und Mealy-Automat sind gleich mächtig: sie können ineinander konvertiert werden.



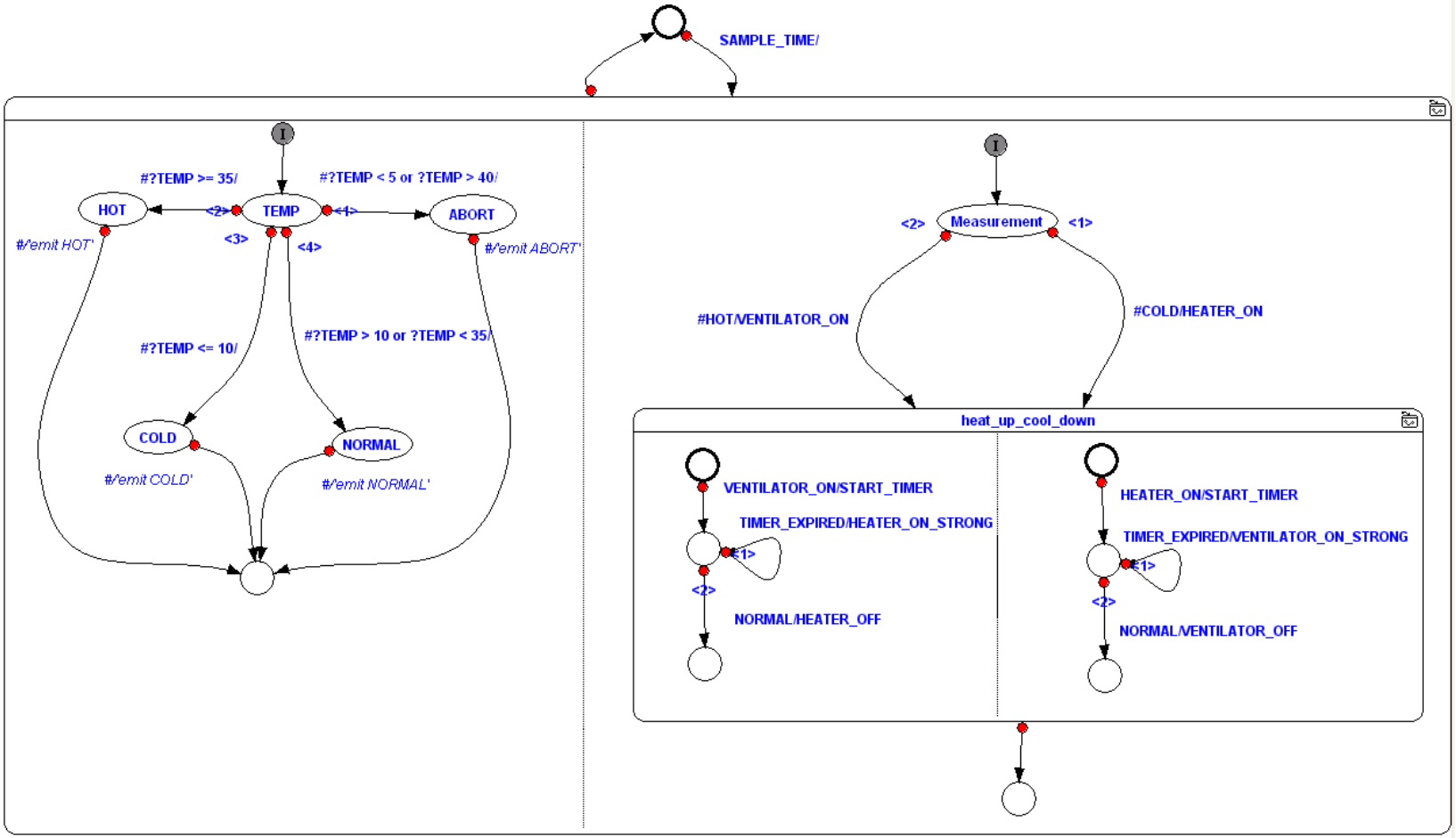
Harel-Automaten / Statecharts

- Heutiger Standard zur Beschreibung von reaktiven Systemen sind die von David Harel 1987 vorgeschlagenen Statecharts.
- Statecharts zeichnen sich durch folgende Eigenschaften aus:
  - Vereinigung der Eigenschaften von Mealy- und Moore-Automaten
    - Ausgaben in Abhängigkeit von Zustandsübergängen möglich
    - Durchführung von Ausgaben beim Erreichen eines Zustands (**`onEntry`**) oder Verlassen eines Zustandes (**`onExit`**)
  - Zur Erhöhung der Lesbarkeit: Hierarchische Strukturierung von Teilautomaten möglich inkl. Gedächtnisfunktionalität
  - Darstellung paralleler Abläufe durch parallele Teilautomaten.
  - Verknüpfung von Zuständen mit Aktionen: Befehle **`do`** (zeitlich begrenzte Aktivität), **`throughout`** (zeitlich unbegrenzte Aktivität)
  - Einführung spontaner bzw. überwachter (guarded) Übergänge

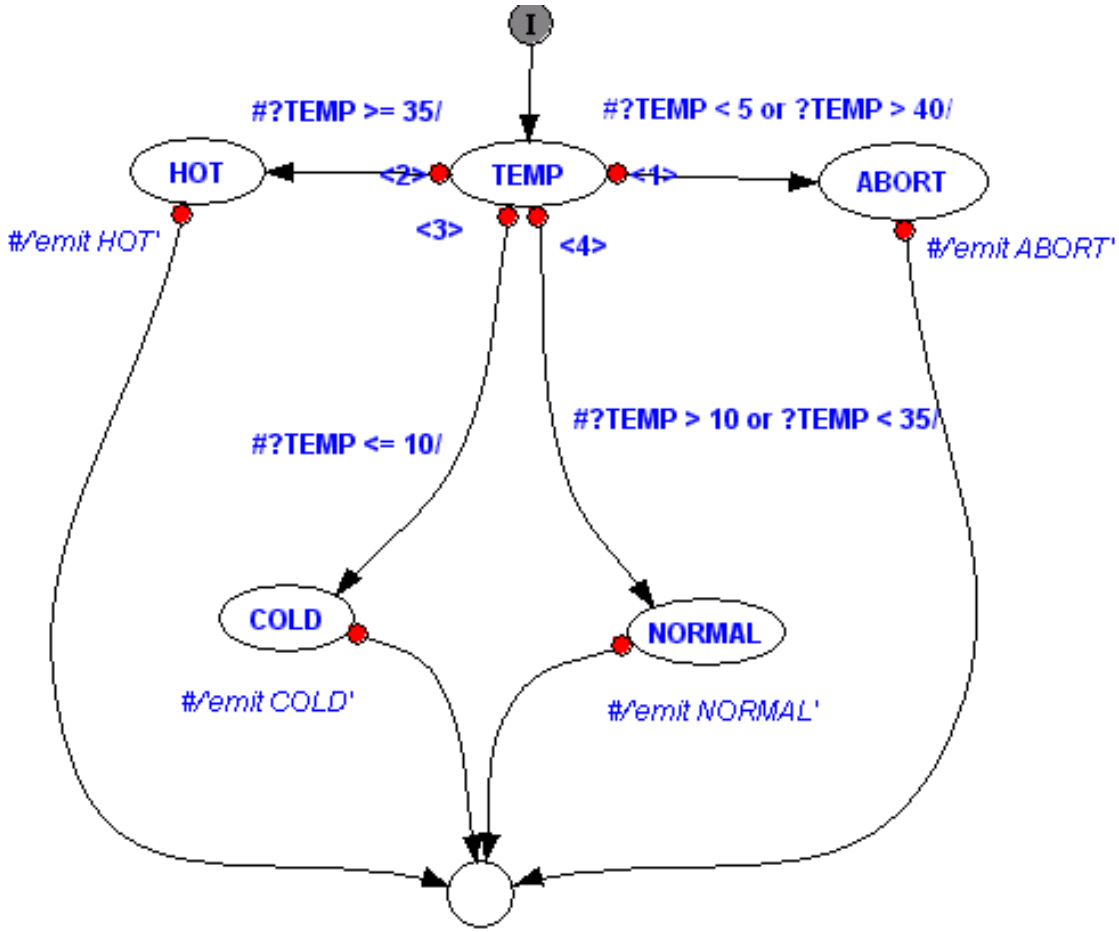
Safe State Machine

- Esterel benutzt eine eigene Klasse von Automaten, die den Statecharts sehr ähnlich sind:
  - Vereinigung der Eigenschaften von Mealy- und Moore-Automaten
    - Ausgaben in Abhängigkeit von Zustandsübergängen möglich
    - Durchführung von Ausgaben beim Erreichen eines Zustands (**`onExit`**) oder Verlassen eines Zustandes (**`onEntry`**)
  - Zur Erhöhung der Lesbarkeit: Hierarchische Strukturierung von Teilautomaten möglich inkl. Gedächtnisfunktionalität
  - Darstellung paralleler Abläufe durch parallele Teilautomaten.
  - Verknüpfung von Zuständen mit Aktionen: Befehle **`do`** (zeitlich begrenzte Aktivität), **`throughout`** (zeitlich unbegrenzte Aktivität)
  - Einführung spontaner bzw. überwachter (guarded) Übergänge
  - Zusätzliche Esterel abhängige Konstrukte (z.B. `pre` Operator)

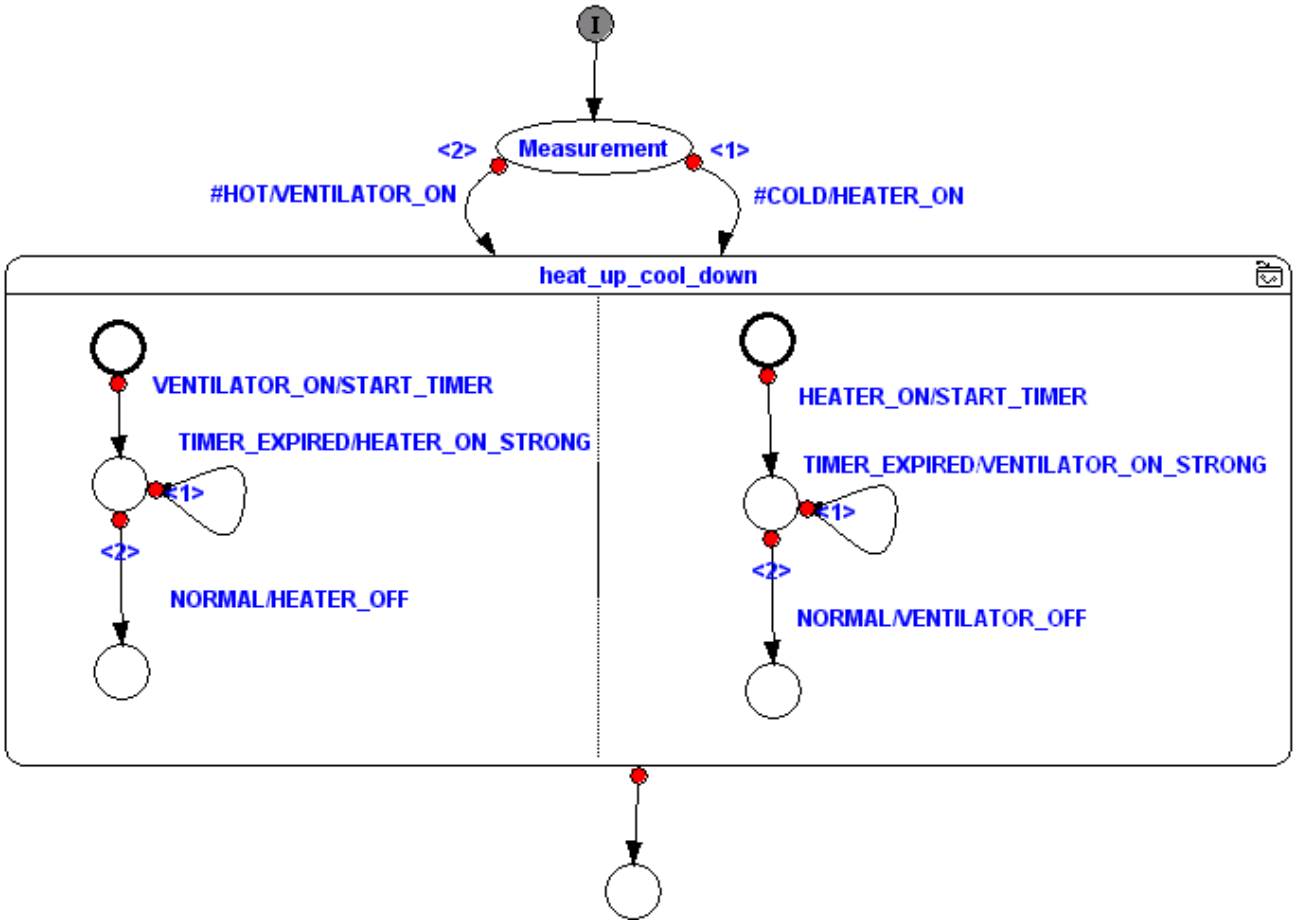
Beispiel als Automat



Beispiel als Automat – Teil 1



Beispiel als Automat – Teil 2



Modellierung von Echtzeitsystemen

Reaktive Systeme - Klausurfragen

Werkzeuge: SCADE, Esterel Studio

Fragen zur Vorlesung

- Was ist der Vorteil von ereignisgesteuerten Applikationen gegenüber zeitgesteuerten?

Zeitgesteuerte Applikationen	Ereignisgesteuerte Applikationen
Zeitlicher Systemablauf wird zur Übersetzungszeit festgelegt.	Ausführungen werden durch das Eintreten von Ereignissen angestoßen.
Präzise und globale Uhr erforderlich (inkl. Uhrensynchronisation)	Garantierte Antwortzeiten sind erforderlich.
Einzelberechnungen werden in einem jeweils reservierten Zeitslot durchgeführt. Ableitung der max. Laufzeit notwendig. (worst case execution time)	Das Scheduling erfolgt dynamisch. Keine Aussage über zeitlichen Ablauf zur Übersetzungszeit möglich.
VORTEIL: Statisches Scheduling möglich. Vorhersagbares deterministisches Verhalten.	**VORTEIL:** ???

Auszug aus Klausur WS 06/07

a) Vervollständigen Sie folgende Testfälle, so dass das Modul xxx diese Testfälle erfüllt:

1. T1=({D},___),(___,{F})
2. T2=({D},___),({D},___)
3. T3=(___,{E}),(___,{F}),(___,{})
4. T4=(_____),(___,{N})
5. T5=(_____),(___,{E}),(___,{E})

Zur Erinnerung: ({A},{B}),({C},{D}) bedeutet: im ersten Moment erfolgt das Ereignis A als Eingabe, die Reaktion des Moduls ist B, im zweiten Moment erfolgt das Ereignis C als Eingabe mit der Reaktion D.

```
module xxx:
    input U, D;
    output E,F,N;
    var V=1;
    loop
        await
          case U do
                    if(?V>0)
                              V:=V+1;
                    else
                              V:=V+1;
                              emit F;
          case D do
                    if(?V>0) then
                              V:=V-1;
                              emit E;
                    else
                              emit N;
        end await;
    end loop;
```