## 2.4 Berechnungsmodelle

Historisch anderer Zugang in der Informatik als in der ingenieurwiss. Systemtheorie:

Modellierungsgegenstand = Berechnungsvorgang

- Bis 70er Jahre:

"Transformatorische" Systeme

Eingabe muss in endlicher Zeit in korrekte Ausgabe transformiert werden.

Bsp. Berechnung einer math. Funktion,
Lösung eines Gleichungssystems,
Sortieren eines Datensatzes,
Suchen eines Datums etc.

Fragestellung:

1. Terminiert die Berechnung? Terminierung
2. Ist das Ergebnis richtig? + partielle Korrektheit

= totale Korrektheit

Nicht terminierende Berechnungen gelten als nutzlos.

⇒ Betrachtung immer über endlichen Zeitraum.

Modelle für transformatorische Systeme:

- ~~Endl. Automaten über endl. Wörtern~~
- ~~Turing-Maschine~~
- Logiken mit Konstrukten der Art Vorzustand-Berechnungsschritt-Nachzustand $\langle x\ P\ y \rangle$ (z.B. Hoare, Floyd)

– Seit ca. 70er Jahre:

„Reaktive" Systeme

(Keine einzelne Berechnung, sondern) Ständig bereit, auf Änderungen in der Umgebung durch entspr. Ausgaben zu reagieren.

Bsp.: Server, Betriebssysteme, Steuerungen ⇒ eingebettete Systeme

Fragestellung:

1. Reagiert das System richtig? Korrektheit (Sicherheit)
2. Reagiert das System rechtzeitig? Echtzeitfähigkeit
3. Ist das System immer wieder bereit, zu reagieren? Reaktivität (Lebendigkeit)

⇒ Betrachtung über unendliche Zeiträume notwendig.

Modelle für reaktive Systeme: (Bspe)

- ~~Moore-Automaten~~
- ω-Automaten
- Kripke-Strukturen
- Temporale Logik

## 2.5 Kripke-Strukturen, Temporale Logik

Ursprung: Modale Logik, Philosophie

- Logische Aussagen sind nicht universell wahr oder falsch
- Gültigkeit hängt von der „Welt" ab, in der sie gemacht werden
- Es gibt mehrere „Welten"

Darstellung der „Welten" und Wechsel zwischen den „Welten" in Kripke-Struktur M:

Def. $M = (X, X_0, f, V, \lambda)$

$X$ : Menge von Zuständen (= „Welten")

~~$X_0 \subseteq X$ : Menge von Anfangszuständen~~

$f \subseteq X \times X$ : (links)totale Zustandsübergangsrelation

(oder $f: X \to 2^X$ )

$V$ : Menge von atomaren Formeln
(log. Aussagen)

$\lambda : X \to 2^V$ : Zustandsmarkierung

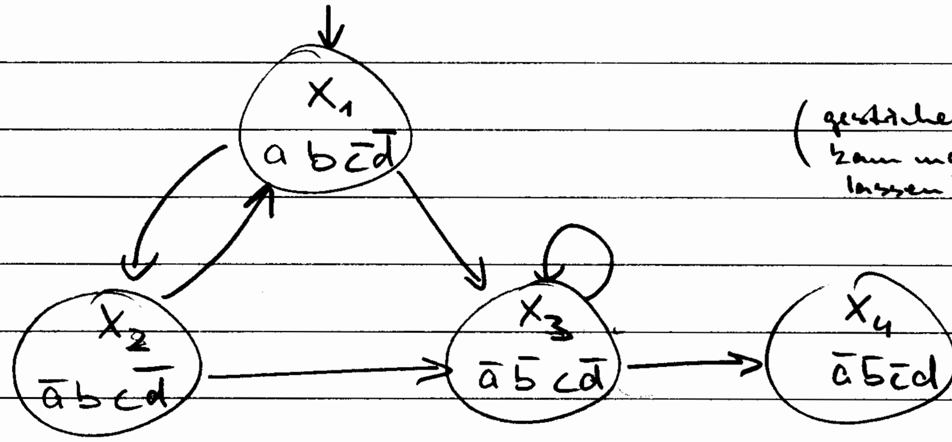
(Bedeutung: Welche Formeln sind in welchem Zustand gültig?)

Kripke-Struktur = markiertes Transitionssystem

Bsp.: $X = \{x_1, x_2, x_3, x_4\}$, $X_0 = \{x_1\}$

$V = \{a, b, c, d\}$

$f$ und $\lambda$ dargestellt im Zustandsgraph:




(gestrichene Formeln kann man weglassen)

Für uns temporale Interpretation wichtig:

(„Welten" wechseln mit der Zeit.)

$\Rightarrow$ Gültigkeit von logischen Aussagen ändert sich mit der Zeit.

Bsp.: Borussia Dortmund ist Deutscher Meister.

Übliche Interpretation:

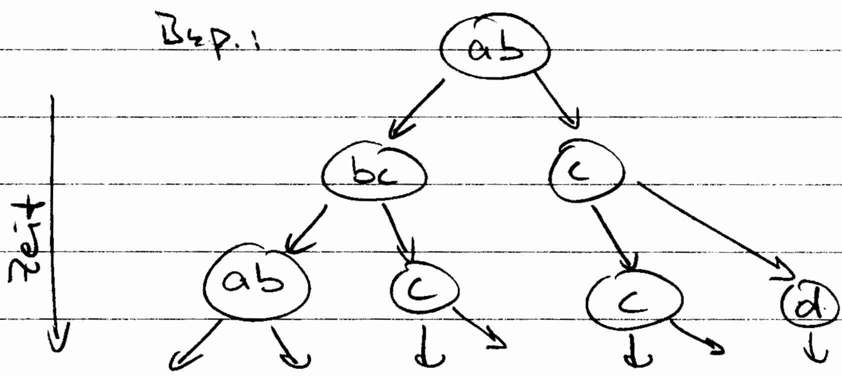
Wechsel von Zuständen = Berechnungsschritt

$\lambda$ ≙ Werte der Programmvariablen zwischen den Berechnungsschritten

„Entfaltung" der Kripke-Struktur über der Zeit

⇒ Computation Tree

Bsp.:




⋮ (unendliche Tiefe) ?

⇒ Menge von Pfaden:

ab → bc → ab → bc → ...

⋮

ab → bc → c → c → ...

⋮

ab → c → c → c → ...

⋮

In jedem Knoten eines Pfades kann man nun <u>Aussagen über die Zukunft</u> machen.

→ <u>Temporale Operatoren</u>

$\Box c$

1. Bsp.: (ab) → (bc) → (c) → (c) → (c) → ...

falsch, wahr, wahr, wahr, wahr ...

„Ab hier gilt immer c."

Engl.: Globally c

Always c

Symbol: G c oder $\Box$ c („Box")

$\Diamond a$

2. Bsp.: (ab) → (bc) → (ab) → (bc) → ...

wahr, wahr, wahr, hängt vom Restpfad ab

„Irgendwann wird mal a gelten."

Engl.: Future a

Sometimes a

Symbol: F a oder $\Diamond$ a („Diamond")

$\bigcirc c$

3. Bsp. (ab) → (bc) → (ab) → ...

wahr, falsch, hängt vom Restpfad ab

„Im nächsten Zustand gilt c"

Engl.: Next c

Symbol: X c oder $\bigcirc$ c

$c \,\mathcal{U}\, d$ falsch wahr wahr wahr falsch

4. Bsp.: $ab \rightarrow c \rightarrow c \rightarrow c \rightarrow d \rightarrow \ldots$

↯

„c gilt solange, bis d gilt."

Engl.: c until d

Symbol: $c \,\mathcal{U}\, d$