Tillståndsmaskiner

Linus Karlsson

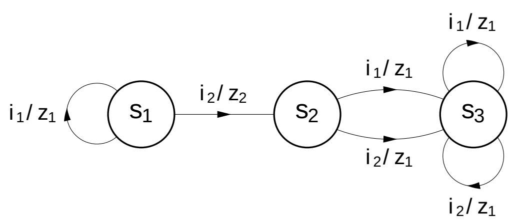
Matematisk definition av tillståndsmaskiner

Förutom att beskriva tillståndsmaskiner i grafisk form med hjälp av tillståndsgrafer ser man ibland i kursboken en mer matematisk beskrivning av tillståndsmaskiner på formatet $\mathcal{M} = (\mathcal{I}, \mathcal{S}, \mathcal{Z}, \delta, \lambda)$.

Den grafiska representationen är väldigt praktisk för att snabbt få en överblick, men den är inte särskilt formell eller praktisk vid exempelvis programmering av algoritmer. Den matematiska definitionen är dessutom bra att känna igen eftersom den dyker upp på flertalet platser i boken. Framförallt är den bra att känna igen senare i kursen när algoritmer för att förenkla tillståndsmaskiner behandlas.

Praktiskt exempel

Utgå från grafen i figur 1. Vi ska nu se hur denna kan uttryckas mer formellt.



Figur 1: En tillståndsmaskin (från uppgift 2.1)

Frågan är nu[1]: vad *är* en tillståndsmaskin? Vad *består* den av? Försöker vi bryta ner grafen i figur 1 ser vi att den har ett antal delar:

1. Till och börja med har vi ett antal olika möjliga insignaler – i det här fallet i_1 och i_2. Vår mängd $\mathcal{I}$ med insignaler är alltså $\mathcal{I} = \{i_1, i_2\}$.
2. Vi har en mängd med samtliga av grafens tillstånd: $\mathcal{S} = \{s_1, s_2, s_3\}$.
3. En mängd med samtliga möjliga utsignaler: $\mathcal{Z} = \{z_1, z_2\}$.

Än så länge har vår nedbrytning varit förhållandevis rättfram. Men hur beskriver vi grafens kanter[2] som går mellan grafens olika tillstånd?

En kant går från ett tillstånd till ett tillstånd, och har en insignal och en utsignal. Om vi vet i vilket tillstånd vi befinner oss, och vilken insignal vi har, kan vi i vår graf enkelt se vilket som är nästa tillstånd. Notera att detta är precis en matematisk funktion! Vi vet två saker: nuvarande tillstånd, samt insignal. Det gör att vi kan beräkna (följa i grafen) och se vilket som är nästa tillstånd.

[1] Detta är nästan en existentiell fråga.

[2] kanter, bågar, edges. Kärt barn har många namn.

Formaliserat kallar vi denna nästa-tillstånd-funktion för δ, och den kan definieras som[3]:

$$\delta : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$$

På motsvarande sätt kan vi konstruera en funktion λ för utsignalen: $\lambda : \mathcal{S} \times \mathcal{I} \to \mathcal{Z}$. Detta eftersom att om vi vet nuvarande tillstånd och nuvarande insignal, kan vi bestämma utsignalen.

Vi kan skriva dessa funktioner i tabellform[4]:

4.

δ	i_1	i_2
s_1	s_1	s_2
s_2	s_3	s_3
s_3	s_3	s_3

5.

λ	i_1	i_2
s_1	z_1	z_2
s_2	z_1	z_1
s_3	z_1	z_1

Slutligen kan vi alltså säga att vår formella definition av tillståndsmaskinen är kombinationen av de fem egenskaperna ovan:

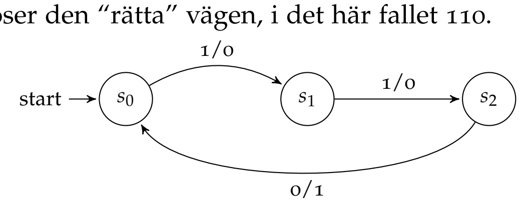
$$\mathcal{M} = (\mathcal{I}, \mathcal{S}, \mathcal{Z}, \delta, \lambda)$$

Tips vid konstruktion av tillståndsmaskiner

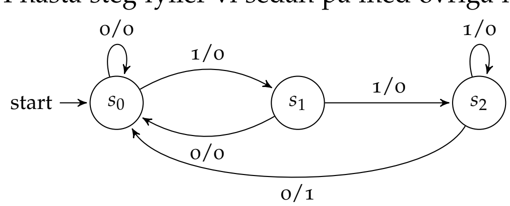
Det finns tyvärr inget recept som alltid fungerar när man ska gå från en problemställning och konstruera en tillståndsgraf som löser problemet. Det krävs alltså en del övning för att lära sig konstruera. Det finns dock en del tips som kan vara bra för en viss typ av problem. Nedan följer ett exempel:

Problem: Konstruera en detektor som detekterar om de tre senaste insignalerna är 110. När så är fallet ska utsignalen vara 1. I alla andra fall ska utsignalen vara 0.

Lösningsförslag: När vi har problem av typen "kolla på de x senaste insignalerna" är det ofta ett bra taktik att först konstruera en graf som löser den "rätta" vägen, i det här fallet 110.



I nästa steg fyller vi sedan på med övriga fall:



Hur vet vi att vi är klara? Kom ihåg uppgift 2.2 från kursboken – antalet kanter som lämnar ett tillstånd ska alltid vara lika med antalet möjliga insignaler! Annars har vår tillståndsmaskin inte ett fullständigt definierat beteende.

[3] Om ni inte känner igen notationen: en funktion f som tar två argument av typerna A respektive B, och som returnerar ett resultat av typen C kan skrivas som: $f : A \times B \to C$.

[4] Läs t.ex. tabell 4 så här: Om vi befinner oss i tillstånd s_1 och har insignalen i_2 kommer nästa tillstånd att vara s_2