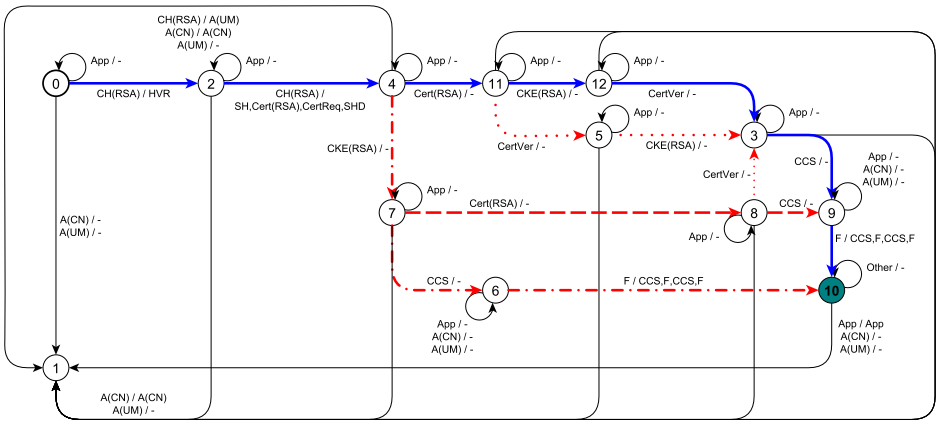
# Afstudeerproject (BSc): Visualising Differences in Finite State Machines

Begeleider: Joshua Moerman
joshua.moerman@ou.nl

**Samenvatting:** Finite state machines (FSMs) zijn modellen met toestanden en transities. Ze worden veel gebruikt om systemen te modelleren, zoals internetprotocollen, hardware, en software. In figuur 1 is een voorbeeld van zo'n FSM. In dit figuur zijn de rondjes toestanden en de pijlen transities. Transities zijn gelabeld met acties die uitgevoerd kunnen worden.




Figuur 1: Een FSM model van het DTLS protocol.

In dit project willen we verschillen tussen twee FSMs visualiseren. Het gaat hierbij om verschil in gedrag, en niet zozeer in verschil in implementatie (zo maakt het nummer van een state bijvoorbeeld niet uit voor het gedrag, maar de acties die je kan uitvoeren wel).

Een typische vorm van equivalentie van toestanden is *bisimulatie*. Twee toestanden zijn bisimulair als ze dezelfde acties toestaan en als ze vervolgens naar bisimulaire toestanden gaan. (Dit klinkt als een circulaire definitie, en dat is ook zo. Toch kunnen we dit wiskundig precies maken.)

Het is de bedoeling dat je een tool maakt die twee FSMs inleest en vervolgens de verschillen visualiseert. Het liefst in de web-browser. Je zul dus gebruik moeten maken van JavaScript, TypeScript, of een andere taal die daarnaar compileert. Daarbij mag je natuurlijk bestaande libraries gebruiken (`fsl.tools`, `Stately Studio`, `sigma.js`, `d3-force`, `vis-network`, ...).

**Keywords:** *Finite State Machines, Bisimulation, Visualisation*

# Achtergrond

## Finite State Machines

Voor dit project kun je je beperken tot deterministische FSMs. Een FSM bestaat uit:

- een eindige verzameling toestanden $Q$,
- met begintoestand $q_0 \in Q$,
- eindige verzamelingen voor input-acties $X$ en output-labels $Y$,
- een transitiefunctie $\delta \colon Q \times \Sigma \to Q$,
- een output-functie $\lambda \colon Q \times X \to Y$.

Als voorbeeld kunnen we het model van figuur 1 als FSM definiëren, neem daartoe $Q = \{0, 1, \ldots, 12\}$, $X = \{\texttt{App}, \texttt{CH(RSA)}, \texttt{A(CN)}, \ldots\}$ en $Y = \{\texttt{HVR}, \texttt{A(UM)}, \ldots\}$. De transitie van state 0 naar 2 wordt dan bijvoorbeeld gedefinieerd door $\delta(0, \texttt{CH(RSA)}) = 2$ en $\lambda(0, \texttt{CH(RSA)}) = \texttt{HVR}$.

## Bisimulatie

Een relatie $R \subseteq Q \times Q$ is een *bisimulatie* als voor alle paren van toestanden $p, q \in Q$ met $pRq$ geldt dat:

- ze dezelfde uitvoer hebben: $\lambda(p, x) = \lambda(q, x)$ voor alle $x \in X$, en
- ze een transitie hebben naar bisimulaire toestanden: $\delta(p, x) R \delta(q, x)$ voor alle $x \in X$.

Er zijn efficiënte algoritmes om bisimulaties te berekenen [Hop71; BP15].

## Visualisatie

Hoe we het beste twee modellen kunnen visualiseren, waarbij bisimulaire toestanden duidelijk worden aangegeven, is een leuk onderzoeksproject. Het zou ook nog kunnen zijn dat toestanden niet bisimulair zijn, maar toch in gedrag op elkaar lijken ("bijna bisimulair"). Ook dit zou gevisualiseerd kunnen worden.

De toepassing die we voor ogen hebben ligt in *automata learning* [Vaa17]. Dit is een techniek om van bestaande software (bijvoorbeeld implementaties van netwerkprotocollen) een FSM te leren. Hierbij leren we vaak meerdere modellen, omdat er meerdere implementaties of versies van een protocol zijn. Het zou handig zijn als we een makkelijke tool hebben om die versies te vergelijken en de verschillen en overeenkomsten te zien.

# Referenties


[BP15] Filippo Bonchi en Damien Pous. “Hacking nondeterminism with induction and coinduction”. In: *Commun. ACM* 58.2 (2015), p. 87–95.

[Hop71] John Hopcroft. “An n log n algorithm for minimizing states in a finite automaton”. In: *Theory of machines and computations*. Elsevier, 1971, p. 189–196.

[Vaa17] Frits W. Vaandrager. “Model learning”. In: *Commun. ACM* 60.2 (2017), p. 86–95.