Tijdsautomaten

Sicco Verwer, PhD. student Computer Science aan de TU Delft
(S.E.Verwer@tudelft.nl)

Hoewel het merendeel van de artikelen die wij plaatsen in De Connectie afkomstig is van de zes universiteiten waarmee we voornamelijk samenwerken (VU, UvA, RuG, UM, RU & UU), zijn dit uiteraard niet de enige plaatsen waar onderzoek gedaan wordt naar AI. Voor het taalnummer ontvingen wij van Sicco Verwer een artikel over Tijdsautomaten. Verwer is PhD. student bij de vakgroep *Parallel and Distributed Systems* van de faculteit *Engineering, Mathematics and Computer Science* aan de Technische Universiteit Delft.

Binnen de kunstmatige intelligentie wordt al heel lang onderzoek gedaan naar taal. Vooral het probleem om computers echt taal te laten "begrijpen" is interessant. Omdat niemand precies weet hoe je het spreken van een natuurlijke taal in een computer kunt programmeren, is het probleem van het (automatisch) leren van een taal een belangrijk onderdeel hiervan. Het probleem kan als volgt worden omschreven:

Gegeven een verzameling woorden (een stuk tekst). Vind de grammatica met de grootste kans dat die deze verzameling geproduceerd heeft.

Onderzoek naar dit probleem heeft geleid tot veel machineleertechnieken die vooral toegepast worden binnen de computationele taalkunde. Wij zijn echter geïnteresseerd in het modelleren van de systemen zelf. Later zal duidelijk worden dat van deze systemen gezegd kan worden dat ze ook een soort "taal" spreken. Om deze reden kunnen de leertechnieken ook op ons probleem toegepast worden.

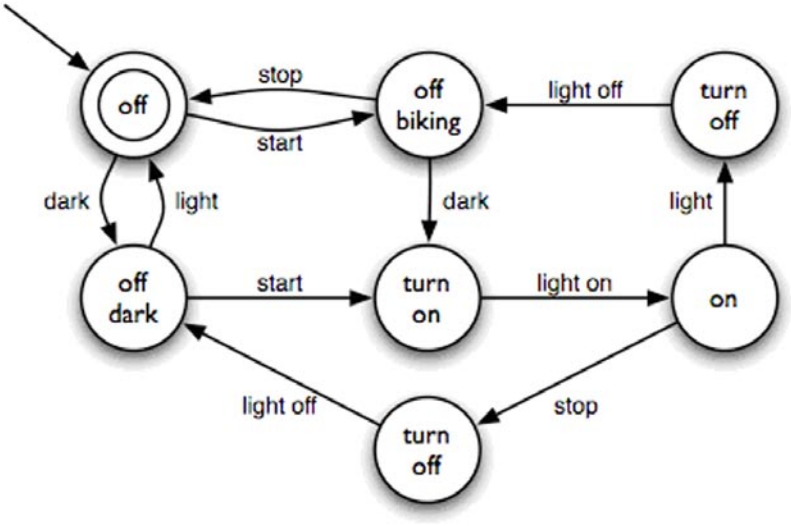
Discrete event systems kenmerken zich doordat hun werking afhangt van het continu optreden van discrete gebeurtenissen. Een veelgebruikt model voor dit soort systemen is, mede dankzij zijn inzichtelijkheid, de eindige automaat (*deterministic finite automaton*). Een eindige automaat is een gerichte graaf, waarin de knopen de verschillende toestanden van het systeem representeren en de takken de overgangen daartussen. Elke tak is voorzien van een label. Dit label geeft aan door welke gebeurtenis de toestandsovergang optreedt. Een eindige automaat bevat altijd een begintoestand en een verzameling eindtoestanden. Voor het gemodelleerde systeem geldt altijd dat het begint in de begintoestand en eindigt in één van de eindtoestanden.

Voor discrete event systems geldt eigenlijk dat ze een soort taal spreken: Wanneer de gebeurtenissen achter elkaar geplakt worden vormen ze een sequentie. Zo'n sequentie is een woord van de taal van het systeem. Die eindige automaat die precies de woorden van de taal van het systeem omschrijft (de sequentie van zo'n woord begint in de begintoestand en eindigt in een eindtoestand) is daarom een mogelijk model voor het systeem. *Figuur 1* geeft een voorbeeld van zo'n eindige automaat, als model voor een automatisch fietslicht.

Om een model te maken voor een bestaand systeem worden vaak interviews afgelegd met experts. De resultaten van deze interviews kunnen dan in logische regels, en vervolgens in een eindige automaat samengevat worden. In veel situaties is er echter niet genoeg kennis beschikbaar om een (volledig) eindige automaat voor het systeem te construeren. Vaak kunnen deze systemen echter geobserveerd worden (bijvoorbeeld met behulp van sensoren). Om in dit soort situaties toch nog een model voor zo'n systeem te maken moeten we het volgende probleem oplossen:

Gegeven een verzameling observaties van een systeem. Vind het model dat met de grootste waarschijnlijkheid deze observaties gegenereerd heeft.

Elk van deze observaties is een sequentie van gebeurtenissen die door het systeem geproduceerd is. Zo kun je het leren van een



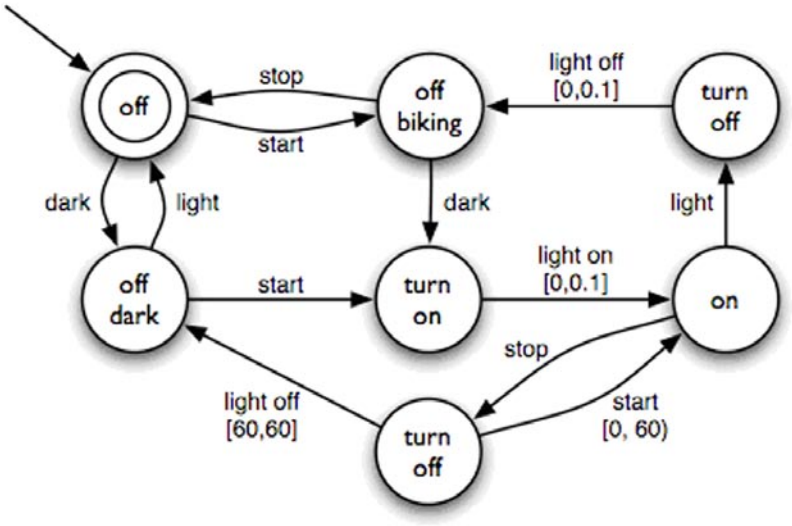
Figuur 1. *Een eindig automaat voor het aansturen van een fietslicht. Het alfabet van deze automaat is de verzameling* `{start, stop, dark, light, light_on, light_off}`. *De automaat begint in de* `off` *toestand (aangegeven door de pijl die vanuit het niets komt) en eindigt ook in de* `off` *toestand (aangegeven door de dubbele rand). Deze automaat kan gebruikt worden om een fietslicht aan te sturen met behulp van de stuursignalen* `light_on` *en* `light_off`.

model, voor een systeem, dus zien als het leren van een taal.

In de praktijk zal de data van de observaties echter meer gegevens bevatten dan slechts de namen van de gebeurtenissen. Het systeem zal namelijk tevens de tijdstippen waarop deze gebeurtenissen plaatsvonden bevatten. Een simpele eindige automaat is ontoereikend als deze tijdstippen beslissen of een sequentie al dan niet tot de taal van het systeem behoort. Bij het fietslicht kan bijvoorbeeld de tijd tussen een `stop` en een `light_off` gebeurtenis belangrijk zijn: slimme fietslichten gaan pas uit nadat er een bepaalde tijd niets meer is gebeurd na een `stop`. Dit is slim omdat de fietser in dat geval ook bij een stoplicht nog steeds zichtbaar is voor de overige weggebruikers.

Gelukkig bestaan er automaten die wel gebruik maken van de genoemde tijdstippen. Dit zijn de zogenaamde tijdsautomaten (*timed automata*). Tijdsautomaten worden veel gebruikt bij het modelleren van real-time systemen. De toestandsovergangen van deze automaten bevatten naast een label ook nog een tijdsgrens. Deze grens is een restrictie op de tijdstippen waarop de toestandsovergang kan optreden. Wanneer er niet wordt voldaan aan de restrictie, dan gebeurt er niets (of er treedt een andere toestandsovergang op). Omdat het probleem van het leren van tijdsautomaten erg ingewikkeld is (het is nog niet zeker of het wel mogelijk is) gebruiken wij een versimpelde vorm van tijdsautomaten. In deze vorm is het alleen mogelijk om een restrictie te leggen op het tijdstip van een gebeurtenis, ten opzichte van de gebeurtenis daarvoor. Met deze tijdsautomaten kunnen we systemen modelleren waarvoor de tijdstippen van de gebeurtenissen van belang zijn. In *figuur 2* zie je bijvoorbeeld een model van een tijdsautomaat voor een slim fietslicht.

Het probleem van het leren van deze versimpelde vorm van tijdsautomaten is op te lossen met een door ons ontwikkeld algoritme. Dit algoritme is een aanpassing van het standaardalgoritme om eindige automaten te laten leren (state merging). Er wordt momenteel onderzoek gedaan naar hoe dit algoritme presteert ten opzichte van andere taalleermethoden. De eerste resultaten van dit onderzoek geven aan dat het probleem van het leren van zo'n tijdsautomaat al snel erg moeilijk wordt, maar dat de gevonden automaten erg hoog scoren qua correctheid. Voor zover wij weten is ons algoritme het (tot nu toe) enige algoritme dat tijdsautomaten kan leren. ø



Figuur 2. *Een tijdsautomaat voor een slim fietslicht. Een stop voor een stoplicht moet niet langer duren dan 1 minuut (60 tijdseenheden). Dit wordt gemodelleerd door een tijdsinterval van 0 tot 60, waarbinnen de automaat van de* `turn_off` *toestand weer naar de on toestand overgaat bij het optreden van een* `start` *gebeurtenis.*

Referenties:

• *Sicco Verwer (http://www.st.ewi.tudelft.nl/~sicco/), Mathijs de Weerdt (http://www.st.ewi.tudelft.nl/~mathijs/) and dr. C. Witteveen (http://www.pds.twi.tudelft.nl/People/C_Witteveen.html), Identifying an automaton model for timed data (http://www.st.ewi.tudelft.nl/~sicco/publications/verwer06benelearn.pdf). Proceedings of the Annual Machine Learning Conference of Belgium and the Netherlands, 2006.*

• *R. Alur, Timed Automata (http://www.cis.upenn.edu/~alur/Nato97.ps.gz). 11th International Conference on Computer-Aided Verification, LNCS 1633, pp. 8-22, Springer-Verlag, 1999.*

• *Lang, K., B. Pearlmutter and R. Price, Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. Proceedings of the Fourth International Colloquium on Grammatical Inference, 1998.*