Afstudeerproject (MSc): Automata Learning

Begeleider: Joshua Moerman
joshua.moerman@ou.nl

Samenvatting: *Automata learning* is een techniek om een systeem (software of hardware) te analyseren en verifiëren, zonder dat we een precieze omschrijving hebben van het systeem. We maken hierbij gebruik van machine learning om uit input/output interactie in kaart te brengen hoe het systeem van binnen werkt.

Deze techniek is al veelvuldig gebruikt om fouten op te sporen in internetprotocollen (zoals TCP, SSH). Het toegepast bij Philips en ASML om verouderde software beter te begrijpen en vervolgens te vernieuwen. En het kan ook gebruikt worden om apparaten te "finger-printen".

Als je kiest voor dit onderwerp, werk je aan een combinatie van theorie en praktijk. Ik verwacht dus dat je kennis hebt over formele talen en automaten en dat je het leuk vindt om het een en ander te implementeren. Er zijn een aantal toepassingen te bedenken:

- NTP: Dit is een protocol om tijd te synchroniseren tussen alle computers. Dit protocol is van groot belang aangezien tijd in bijna alle communicatie tussen computer van belang is. De combinatie van automata learning en tijd is nog een open vraagstuk.
- QUIC: Dit is een nieuwe protocol als alternatief voor TCP, met toepassing voor IoT. Het zou interessant zijn om verschillende implementaties te leren en te analyseren wat de verschillen zijn.
- Misschien heb je zelf iets in je werk of interesses waarvan je een statemachine wilt leren en analyseren.

Keywords: *Automata Learning, Internet Protocols, Model Checking, State Fuzzing, Finite State Machines*

Achtergrond

In ons dagelijks leven zijn we erg afhankelijk geworden van software en hardware. Je gebruikt het dagelijks om te communiceren met anderen of om online te betalen. Het is dus erg belangrijk dat de onderliggende protocollen goed en foutloos werken. Hoe kun je bestaande software correct bewijzen?

In het algemeen kan dat helaas niet (dat is namelijk onbeslisbaar). Voor software die gemodelleerd kan worden als eindige automaat, kan dat gelukkig wel. En met automata learning kun je zo'n eindige automaat automatisch leren van een stukje software.

Een eindige automaat bestaat uit eindig veel toestanden. Je kunt met een bepaalde inputs van de ene toestand naar de andere gaan, dit gaat vaak gepaard met een bepaalde output. De toestanden, samen met de transities leggen dan het hele systeem vast. Dit klinkt simpel, maar toch is dit expressief genoeg om veel van onze internetprotocollen in uit te drukken.

Automata learning werkt als volgt (versimpeld): We veronderstellen dat we de software waarvan we een automaat willen leren beschikbaar is voor interactie (d.w.z., we kunnen het uitvoeren en inputs geven). Door slimme inputs te kiezen, komen we terecht in verschillende toestanden van de software. Maar we kunnen niet direct zien in welke toestand we zijn. Daarvoor moeten tests uitvoeren in die toestand (ook weer door middel van inputs). Door herhaaldelijk naar toestanden te gaan en te testen, "zien" we steeds meer verschillende toestanden van de software en kunnen we uiteindelijk een plaatje maken van de hele toestandsruimte.

Meer details kun je lezen in het overzichtsartikel van Frits Vaandrager [Vaa17].

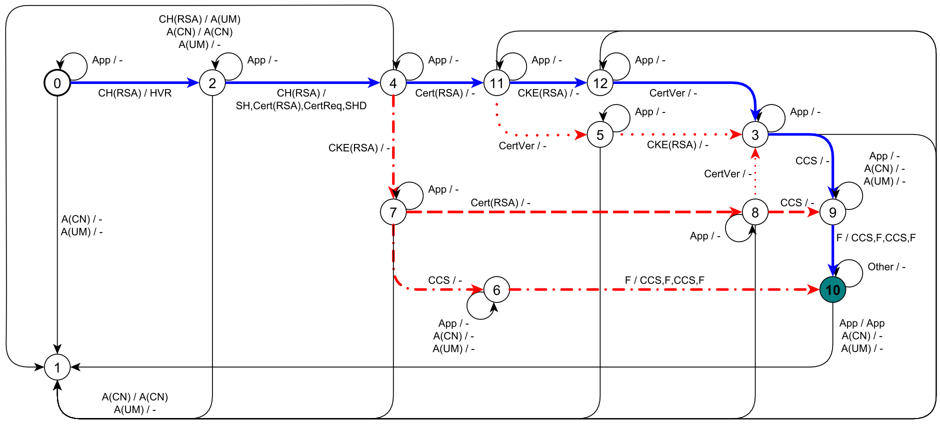
Algoritmes en software libraries

Het basisalgoritme voor automata learning is L* [Ang87], maar er zijn inmiddels efficiëntere of algemenere algoritmes gevonden. Deze algoritmes zijn te vinden in verschillende software libraries, je hoeft dit dus niet zelf te implementeren. Afhankelijk van je project kun je direct aan de slag of moet je deze pakketten uitbreiden met bepaalde functionaliteit. Hieronder staan een aantal pakketten:

- LearnLib (Java) `https://learnlib.de`
- AALpy (python) `https://github.com/DES-Lab/AALpy`
- libalf (C++) `http://libalf.informatik.rwth-aachen.de`

Toepassingen

Automata learning wordt bijvoorbeeld toegepast om modellen van (implementaties van) internet protocollen te leren. Van de volgende protocollen zijn al modellen geleerd: TLS [RP15], TCP [FJV16], SSH [Fit+17], WIFI [SCR18], VPN [DPR18], QUIC [RAR19] en DTLS [Fit+20] (zie ook Figuur 1). In veel van deze papers wordt het geleerde model tegen een specificatie gehouden om te kijken of de implementatie deugt. In enkele gevallen zijn er echte security-problemen gevonden.



Figuur 1: Een model geleerd van een DTLS-implementatie. Analyse van dit model heeft aangetoond dat er zwakheden (vulnerabilities) in de implementatie zijn, aangegeven met rode transities.

Automata learning wordt ook toegepast in embedded systemen, waar de software dus echt black-box is. Enkele voorbeelden hiervan zijn: Betaalkaarten [ARP13], E-reader voor online betalen [Cha+14], Printer driver van Océ [Sme+15] en Bluetooth apparaten (IoT) [PA21]. Door het onderzoek met de e-readers, is er een fout ontdekt en later ook gefixed door de bank. In het onderzoek met bluetooth wordt gekeken naar het fingerprinten van apparaten: Kunnen we achterhalen met welk apparaat we verbonden zijn door bepaalde inputs te sturen? (Het antwoord is "ja", ondanks dat bluetooth in principe een vast protocol heeft.)

Een andere toepassing van automata learning is het vernieuwen van bestaande software. Hierbij moet eerst de oude software worden begrepen. Dit is bijvoorbeeld toegepast bij Philips Healthcare [SHV16] en ASML [Asl+20].

Referenties

Hieronder staan alle genoemde referenties, je hoeft ze uiteraard niet allemaal te lezen. Mocht je dit onderwerp interessant vinden, raad ik je wel aan het overzichtsartikel van Frits Vaandrager [Vaa17] te lezen.

[Ang87] Dana Angluin. "Learning Regular Sets from Queries and Counterexamples". In: *Inf. Comput.* 75.2 (1987), p. 87–106.

[ARP13] Fides Aarts, Joeri de Ruiter en Erik Poll. "Formal Models of Bank Cards for Free". In: *ICST Workshops*. IEEE Computer Society, 2013, p. 461–468.

[Asl+20] Kousar Aslam e.a. "Interface protocol inference to aid understanding legacy software components". In: *Softw. Syst. Model.* 19.6 (2020), p. 1519–1540.

[Cha+14] Georg Chalupar e.a. “Automated Reverse Engineering using Lego®”. In: *WOOT*. USENIX Association, 2014.

[DPR18] Lesly-Ann Daniel, Erik Poll en Joeri de Ruiter. “Inferring OpenVPN State Machines Using Protocol State Fuzzing”. In: *EuroS&P Workshops*. IEEE, 2018, p. 11–19.

[Fit+17] Paul Fiterau-Brostean e.a. “Model learning and model checking of SSH implementations”. In: *SPIN*. ACM, 2017, p. 142–151.

[Fit+20] Paul Fiterau-Brostean e.a. “Analysis of DTLS Implementations Using Protocol State Fuzzing”. In: *USENIX Security Symposium*. USENIX Association, 2020, p. 2523–2540.

[FJV16] Paul Fiterau-Brostean, Ramon Janssen en Frits W. Vaandrager. “Combining Model Learning and Model Checking to Analyze TCP Implementations”. In: *CAV (2)*. Deel 9780. Lecture Notes in Computer Science. Springer, 2016, p. 454–471.

[PA21] Andrea Pferscher en Bernhard K. Aichernig. “Fingerprinting Bluetooth Low Energy Devices via Active Automata Learning”. In: *FM*. Deel 13047. Lecture Notes in Computer Science. Springer, 2021, p. 524–542.

[RAR19] Abdullah Rasool, Greg Alpár en Joeri de Ruiter. “State machine inference of QUIC”. In: *CoRR* abs/1903.04384 (2019).

[RP15] Joeri de Ruiter en Erik Poll. “Protocol State Fuzzing of TLS Implementations”. In: *USENIX Security Symposium*. USENIX Association, 2015, p. 193–206.

[SCR18] Chris McMahon Stone, Tom Chothia en Joeri de Ruiter. “Extending Automated Protocol State Learning for the 802.11 4-Way Handshake”. In: *ESORICS (1)*. Deel 11098. Lecture Notes in Computer Science. Springer, 2018, p. 325–345.

[SHV16] Mathijs Schuts, Jozef Hooman en Frits W. Vaandrager. “Refactoring of Legacy Software Using Model Learning and Equivalence Checking: An Industrial Experience Report”. In: *IFM*. Deel 9681. Lecture Notes in Computer Science. Springer, 2016, p. 311–325.

[Sme+15] Wouter Smeenk e.a. “Applying Automata Learning to Embedded Control Software”. In: *ICFEM*. Deel 9407. Lecture Notes in Computer Science. Springer, 2015, p. 67–83.

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