R1.O1 On the transition function

For a Non-Deterministic Finite Automaton (NFA), the transition function $\delta: Z \times \Sigma \rightarrow 2^{Z}$ takes a state from the set of states Z and a character from the alphabet Σ to transition the NFA from one state to a set of follow-up states from the power set of Z. Therefore, an NFA can be in several states simultaneously, allowing a single character to cause transitions in multiple states. For instance, using the character C, the NFA shown in our running example Fig. 2 transitions from state z1 to z1 and from z2 to z1.

R1.O1 On the Thompson construction

The Thompson construction is an algorithm to convert a Regular Expression (RegEx) into an equivalent NFA. It is recursive and works as follows:

Base cases: For a RegEx that consists of a single character, an NFA is created with two states: a non-accepting initial state and an accepting state; we connect the initial-to-final states by a transition labeled with that character. For ε, the NFA also has two states, but they are connected by an ε-transition, which requires no input to move from the initial to the accepting state, i.e., this NFA accepts the empty string.

Concatenation A.B: To concatenate RegEx A with RegEx B, we create a new NFA where A's accepting state is merged with B's initial state.

Union A|B: We create a new NFA with an initial and an accepting state for the union operator. Epsilon transitions are added from the new initial state to the initial states of A and B, and from the accepting states of A and B to the new accepting state.

*Kleene Star A**: We create a new NFA with an initial and accepting state for the Kleene star operator. Epsilon transitions are added to connect the new initial state to both the initial state of A and the new accepting state. Additionally, another epsilon transition connects the accepting state of A back to its initial state.

These steps are recursively applied to construct an NFA for a regular expression, building it from NFAs of simpler sub-expressions.

R2.O1.1 On incorporating stricter skipping policies into our model.

Under the Skip-Till Any Match (STAM) policy, non-deterministic actions are allowed on relevant events (suitable for transitions), enabling a partial match to be duplicated, with one instance transitioning and the other discarding the event to remain in its current state. Skip-Till Next Match (STNM) is a stringent policy that permits skipping irrelevant events (unsuitable for transitions) but requires using relevant events without duplicating the instance. The most restrictive policy is strict contiguity, demanding that events be contiguous in the stream to constitute a match, thus disallowing any skipping.

Based on skip-till next match, we would like to sketch how a stricter policy can be incorporated into our model. Since the main difference between STAM and STNM is that STNM does not duplicate partial matches upon transition, we have to reflect this within the

count rules. The arrival of an event triggers partial matches in a state that can use the event to transition. Consequently, the count of partial matches in the follow-up state(s) increases by the number from the originating states. However, under the STNM policy, as no duplicates are created, the count in the originating states decreases by the number of partial matches that transitioned using the event.

Applying STNM to our running example in Fig. 2, when A1 occurs, it increases #z1 by 1. Event B2 causes all partial matches in z1 to transition to z2, increasing #z2 by 1. However, at the same time, #z1 is reduced by 1 (from 1 to 0) because an instance is not duplicated upon transition.

References

Finance

Kia Teymourian, Malte Rohde, and Adrian Paschke. 2012. **Knowledge-based processing of complex stock market events**. In 15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings, Elke A. Rundensteiner, Volker Markl, Ioana Manolescu, Sihem Amer-Yahia, Felix Naumann, and Ismail Ari (Eds.). ACM, 594–597. https://doi.org/10.1145/2247596.2247674

Algorithmic trading

Olga Poppe, Chuan Lei, Elke A. Rundensteiner, and David Maier. 2017. **GRETA: graph-based real-time event trend aggregation**. Proceedings of the VLDB Endowment 11, 1 (Sept. 2017), 80–92. https://doi.org/10.14778/3151113.3151120

Urban transportation

Alexander Artikis, Matthias Weidlich, François Schnitzler, Ioannis Boutsis, Thomas Liebig, Nico Piatkowski, Christian Bockermann, Katharina Morik, Vana Kalogeraki, Jakub Marecek, Avigdor Gal, Shie Mannor, Dimitrios Gunopulos, and Dermot Kinane. 2014. **Heterogeneous Stream Processing and Crowdsourcing for Urban Traffic Management**. In Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014, Sihem Amer-Yahia, Vassilis Christophides, Anastasios Kementsietsidis, Minos N. Garofalakis, Stratos Idreos, and Vincent Leroy (Eds.). OpenProceedings.org, 712–723. https://doi.org/10.5441/002/edbt.2014.77

Electrical grids

Raul Castro Fernandez, Matthias Weidlich, Peter R. Pietzuch, and Avigdor Gal. 2014. **Scalable stateful stream processing for smart grids**. In The 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, Mumbai, India, May 26-29, 2014, Umesh Bellur and Ravi Kothari (Eds.). ACM, 276–281. https://doi.org/10.1145/2611286.2611326