Formal Characterizations of FA-based String Processors

Ernest Ketcha Ngassam[1,2,⋆], Bruce W. Watson[3], and Derrick G. Kourie[3]

[1]SAP Meraka UTD, Pretoria, South Africa
[2]School of Computing University of South Africa
Pretoria 0001
ernest.ngassam@sap.com
[3]Department of Computer Science, University of Pretoria
South Africa, Pretoria 0002
{dkourie,bwatson}@cs.up.ac.za

Abstract. We rely on the denotational semantics of algorithms to suggest an abstraction of a string recognizer. The abstraction provides a unified formalism for representing FA-based string recognizers as an instance of a parameterized function. It also forms the basis for theoretical investigations on implementations of FA-based recognizers, and represents a framework for the identification of new algorithms for further studies.

1 Preliminaries and Characterization

An acceptor (or a string recognizer) of a finite automaton is an algorithm that relies on the finite automaton's transition function in order to determine whether a string is part of the language modeled by the FA or not. An acceptor of the automaton $\mathrm{M} = (\mathcal{Q}, \mathcal{V}, \Delta, \mathcal{S}, \mathcal{F})$, where: $\mathcal{V}$ and Δ are the alphabet and transition function respectively; $\mathcal{L}(\mathrm{M}) \subseteq \mathcal{V}^*$ is the language of M; and $\mathcal{P}(X)$ is used to represent the power set of a set X, can be characterized by the following function:
$\rho : \mathcal{P}(\mathcal{Q} \times \mathcal{V} \times \mathcal{Q}) \times \mathcal{V}^* \nrightarrow \mathbb{B} = \{true, false\}$ such that for $\Delta \in \mathcal{P}(\mathcal{Q} \times \mathcal{V} \times \mathcal{Q})$, $\rho(\Delta, s) = true$ if $s \in \mathcal{L}(\mathrm{M})$ or $\rho(\Delta, s) = false$ if $s \notin \mathcal{L}(\mathrm{M})$.
In fact, ρ is the *denotational semantics* of the acceptor [4] and is a partial function, hence the mapping using the relation $\nrightarrow$. The denotational semantics indicates the "meaning" of the algorithm in functional terms, but hides details about how the algorithm that performs acceptance testing should actually work. At this level of description, the acceptor is viewed as a "black box" that receives as input a transition set and a string, and later produces a boolean as output. There are, in fact, a large number of ways in which this processing can take place, as extensively discussed and implemented in [5]. In fact, three core algorithms were discussed; namely the table-driven (TD), hardcoded (HC), and a hybrid version of the two referred to as mixed-mode (MM). Furthermore, a range of implementation strategies were investigated and implemented with their performance analyzed. Those strategies intended to optimize cache memory usage in order to improve the performance of the recognizer. The strategies discussed were Dynamic State Allocation (DSA), Allocated Virtual Caching (AVC) and State pre-Ordering (SpO). It was proven that by implementing each strategy or their combination based on any given core algorithm, the performance of the recognizer would improve [5].

⋆ Supported in part by the South African National Research Foundation (NRF).

In order to refine the generic definition of ρ above, let Δ_t denotes the transition set that is used in the TD part of an MM implementation. Similarly, let Δ_h be the transition set that is used in the TD part. Clearly, Δ_t and Δ_h must be a partition of the original transition set, Δ, i.e. $\Delta_t \cup \Delta_h = \Delta$ and $\Delta_t \cap \Delta_h = \emptyset$. Now let $\rho_{_C}$ be the function to represent the denotational semantics of a recognizer based on one of the core algorithms, TD, HC or MM. Letting $\mathcal{T} = \mathcal{P}(\mathcal{Q} \times \mathcal{V} \times \mathcal{Q})$, this function can be defined as follows: $\rho_{_C} : \mathcal{T} \times \mathcal{T} \times \mathcal{V}^* \nrightarrow \mathbb{B}$ such that $\rho_{_C}(\Delta_t, \Delta_h, s) = \rho(\Delta, s)$.

In order to obtain a generic formalism that takes into consideration the various foregoing implementation strategies, appropriate parameters are necessary. In fact, DSA and AVC would each require two variables D_t and D_h (resp. V_t and V_h) which are natural numbers that reflect the extent to which the TD part (resp. HC part) of the recognizer will be table-driven (resp. hardcoded). Similarly, for the SpO strategy, two boolean parameters P_t and P_h are required. They indicate whether the TD part (resp. the HC part) of the algorithm requires pre-ordering.

By putting all the strategies together, we may now characterize a recognizer as a new function, $\rho_{_{CDPV}}$, that is parameterised by: its transition sets Δ_t and Δ_h; all its strategies (D_t, D_h, P_t, P_h, V_t and V_h); and its input string s. Therefore, once the transitions sets have been specified, given an arbitrary input string, one may choose to implement $\rho_{_{CDPV}}$ using any of the strategies or some combination of them, as long as the necessary conditions on strategies are respected. A recognizer's denotational semantics is now formally expressed in general as follows:
$\rho_{_{CDPV}} : \mathcal{T} \times \mathcal{T} \times \mathbb{N} \times \mathbb{N} \times \mathbb{B} \times \mathbb{B} \times \mathbb{N} \times \mathbb{N} \times \mathcal{V}^* \nrightarrow \mathbb{B}$ such that

$$\text{if } \begin{cases} (\Delta_t \cup \Delta_h = \Delta) \wedge (\Delta_t \cap \Delta_h = \emptyset) \\ (0 \leq D_t \leq |\mathcal{Q}_t|) \wedge (0 \leq D_h \leq |\mathcal{Q}_h|) \\ (P_t \in \mathbb{B}) \wedge (P_h \in \mathbb{B}) \\ (0 \leq V_t < |\mathcal{Q}_t|) \wedge (0 \leq V_h < |\mathcal{Q}_h|) \end{cases}$$

$$\text{then } \rho_{_{CDPV}}(\Delta_t, \Delta_h D_t, D_h, P_t, P_h, V_t, V_h, s) = \rho(\Delta, s)$$

The recognizer defined as such shows that, the strategies used depend on the nature of the transition set itself. Arguments subscripted with $_t$ are associated to the TD algorithm, whereas those subscripted with $_h$ are associated to the HC algorithm. The high-level formalisms associated with each type of algorithm may be used in obtaining of new algorithms using appropriate instances of their associated strategy variables.

2 Conclusion and Future Work

This paper suggested a formal characterization of FA-based string processor taking into consideration a range of strategies that could be explored for leveraging the performance of the recognizer at run-time. Such characterization could be explored theoretically in order to determine appropriate properties of each strategic variables employed. The suggested formalism relied only on investigations conducted in [5] for cache optimization strategies. As a matter of future work, we consider exploring other aspects such computing platform (OS) and appropriate computational medium. It also worth mentioning that many of the algorithms formalized in this paper based on implementation strategies have proven to be more efficient that their core counterparts. Work on further investigation on other algorithms is still ongoing and results presenting their performance can be found in [1–3, 5].

References

1. E. N. Ketcha, D. G. Kourie, and B. W. Watson: *Reordering finite automatata states for fast string recognition*, in In Proceeding of the Prague Stringology Conference, Prague, Czech Republic, August 2005, Czech Technical University.

2. E. N. Ketcha, D. G. Kourie, and B. W. Watson: *A taxonomy of dfa-based string processors*, in In Proceeding of the SAICSIT Conference, Gordon's Bay, South Africa, October 2006, ACM, pp. 111–121.

3. E. N. Ketcha, B. W. Watson, and D. G. Kourie: *On implementations and performances of table-driven fa-based string processors*, in In Proceeding of the Prague Stringology Conference, Prague, Czech Republic, August 2006, Czech Technical University.

4. B. Meyer: *Introduction to the Theory of Programming Languages*, Prentice Halh, c.a.r hoare series ed., 1990.

5. E. K. Ngassam: *Towards Cache Optimization in Finite Automata Implementations*, PhD thesis, Department of Computer Science, Pretoria, South Africa, November 2006.