An Overview of Cache Optimized FA-based String Processors

Ernest Ketcha Ngassam

School of Computing

University of South Africa

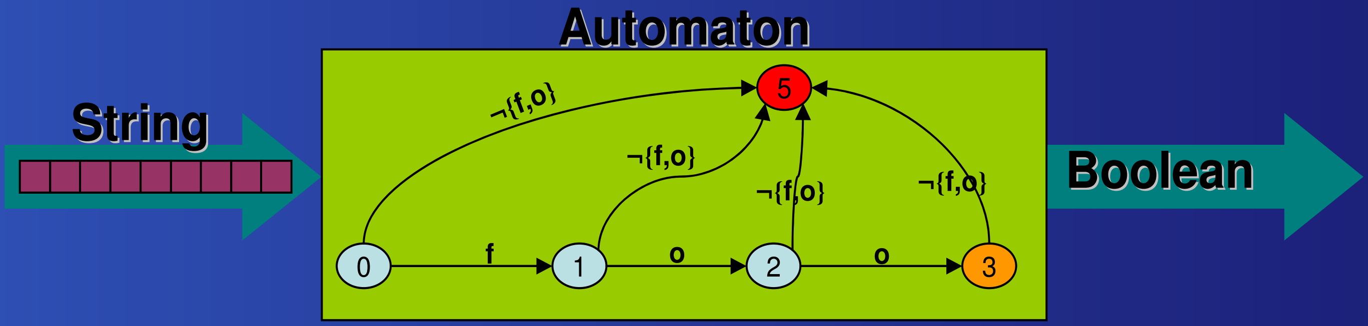
Agenda

1. Introduction
2. Cache optimized algorithms
3. Spectrum of the algorithms
4. Taxonomy and toolkit
5. Applications

Introduction: The Problem

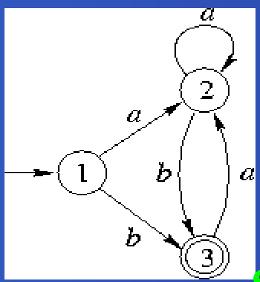
Given an automaton $\mathcal{M} = (\mathcal{V}, \mathcal{Q}, s_0, \mathcal{F}, \delta)$ and a string s

- Determine whether s is part of the language $\mathcal{L}$ modeled by $\mathcal{M}$



The table-driven (TD) algorithm

A 2D array is used to hold δ before scanning



Symbol

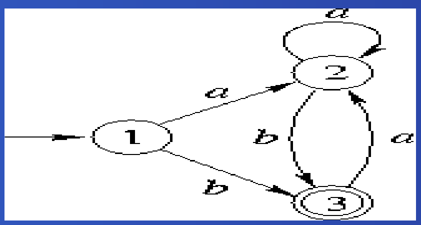
δ state	a	b
1	2	3
2	2	3
3	2	-1

```
#define isFinal(s)      ((s) < 0)
int scanner()
{  char ch;
   int currState = 1;

   while (TRUE) {
      ch = NextChar( );
      if (ch == EOF) return 0;  /* fail */
      currState = δ[currState, ch];
      if (IsFinal(currState)) {
         return 1;  /* success */
      }
   } /* while */
}
```

The hardcoded (HC) algorithm

δ forms part of the algorithm



```
#define isFinal(s)       ((s) < 0)
int scanner()
{  char ch;
   int s = 1;
   while (TRUE) {
      ch = NextChar( );
     if (ch==EOF) return 0 /*fail*/
      if (s==1)
      {
         if (ch=='a') s = 2
         else s = 3
        }
      else if (s==2)...
       ....
       if (IsFinal(s)) return 1;  /*success */
} /* while */
}
```

Limitations of TD and HC

- Performance is hampered by the memory/instruction load problem
  - As the automaton size grows, more memory required

- The random access nature of the table/states is a performance bottleneck
  - Cache misses penalties occur since states information are not well organized

- Weak in processing large strings based on large automata

Cache optimized algorithms

Dynamic State Allocation (DSA)

- Predefined portion of memory used for recognition
- A state out the predefined memory is first copied-in before recognition
    - Can be unbounded (no restriction on the size of the predefined memory)
    - Can be bounded (number of states to be allocated restricted)
- Efficient for recognizing large strings

Cache optimized algorithms

State pre-Ordering (SpO)

- States are reordered based on some previous processing history
- New states positions are kept in an auxiliary array
  - The array enables access to states' information
- Frequently accessed states are grouped together
  - Improves spatial and temporal locality of reference

Cache optimized algorithms

Allocated Virtual Caching (AVC)

- A portion of memory occupied by states is dedicated for acceptance testing
- The dedicated memory behaves as a cache (virtual)
    - Access policy: Direct and associative mapping ...
- Frequently accessed states are grouped together
    - Improves spatial and temporal locality of reference
- Efficient for large string visiting a limited set of states

Recognizer’s Denotational Semantics

Given an automaton $M(\mathcal{Q}, \mathcal{V}, \Delta, \mathcal{F}, s_0)$;
ρ is the string recognizer that maps (Δ, s) to a boolean

We assume the transition set Δ *is split into two disjoints subsets* Δ_t *and* Δ_h*:*

- Δ_t (resp. Δ_h) is the transition set for TD (resp. HC)

The following relationship holds $\forall s \in \mathcal{V}^*$:

$$\rho(\Delta, s) = \rho_{_C}(\Delta_t, \Delta_h, s) = \rho_{_C}(\Delta_t, \emptyset, s) = \rho_{_C}(\emptyset, \Delta_h, s)$$

- $\rho(\Delta, s) = \rho_{_C}(\Delta_t, \Delta_h, s)$ is the mixed-mode (MM) algorithm

Characterization of a Recognizer

A string recognizer of an FA can now be defined as an algorithm that maps its input string, its transition sets, and its associated strategy variables to a boolean:
Assume $\mathcal{T} = \mathcal{P}(\mathcal{Q} \times \mathcal{V} \times \mathcal{Q})$ *(transition relation)*

D_t *and* D_h *are natural numbers associated to DSA*
P_t *and* P_h *are boolean values associated to SpO*
V_t *and* V_h *are natural numbers associated to AVC*

Characterization of a Recognizer

$$\rho_N : \mathcal{T} \times \mathcal{T} \times \mathbb{N} \times \mathbb{N} \times \mathbb{B} \times \mathbb{B} \times \mathbb{N} \times \mathbb{N} \times \mathcal{V}^* \rightarrow \mathbb{B}$$

$$\rho_N(\Delta_t, \Delta_h, D_t, D_h, P_t, P_h, V_t, V_h, s) = \begin{cases} \text{T} & \text{if } s \in \mathcal{L}(\mathcal{M}) \\ \text{F} & \text{if } s \notin \mathcal{L}(\mathcal{M}) \end{cases}$$

Such that

$$\rho_N(\Delta_t, \Delta_h, D_t, D_h, P_t, P_h, V_t, V_h, s) = \rho(\Delta, s)$$

Characterization of a Recognizer

Formalisms with special values assigned to parameters:

- TD formalism: $\rho_{_N}(\Delta_t, \emptyset, D_t, 0, P_t, \mathsf{F}, V_t, 0, s)$
- HC formalism: $\rho_{_N}(\emptyset, \Delta_h, 0, D_h, \mathsf{F}, P_h, 0, V_h, s)$
- MM formalism: $\rho_{_N}(\Delta_t, \Delta_h, D_t, D_h, P_t, P_h, V_t, V_h, s)$

♣ **Specialized notations:**

- TD: $\rho_{_N}(\Delta_t, \emptyset, D_t, 0, P_t, \mathsf{F}, V_t, 0, s) = \rho_t(\Delta_t, D_t, P_t, V_t, s)$
- HC: $\rho_{_N}(\emptyset, \Delta_h, 0, D_h, \mathsf{F}, P_h, 0, V_h, s) = \rho_h(\Delta_h, D_h, P_h, V_h, s)$
- MM: $\rho_{_M}(\Delta_t, \Delta_h, D_t, D_h, P_t, P_h, V_t, V_h, s)$

Spectrum of the Algorithms (TD/HC)

- DSA Strategy: The variable $D_{t/h} \in \{0, d_{t/h}, |\mathcal{Q}_{t/h}|\}$; $d_{t/h} < |\mathcal{Q}_{t/h}|$
- SpO Strategy: The variable $P_{t/h} \in \{\texttt{T},\texttt{F}\}$
- AVC Strategy: The variable $V_{t/h} \in \{0, v_t\}$; $v_{t/h} < |\mathcal{Q}_{t/h}|$
- Resulting in $3 \times 2 \times 2 = 12$ different algorithms

Spectrum of the Algorithms (TD/HC)

Combination	Active strategy	TD/HC Name
$(0, \mathsf{F}, 0)$	None (core TD/HC)	t/h
$(d_{t/h}, \mathsf{F}, 0)$	bounded DSA	t/h_{b1}
$(n, \mathsf{F}, 0)$	unbounded DSA	t/h_{u1}
$(0, \mathsf{T}, 0)$	SpO	t/h_2
$(0, \mathsf{F}, v_{t/h})$	AVC	t/h_3
$(0, \mathsf{T}, v_{t/h})$	SpO and AVC	t/h_{23}
$(d_{t/h}, \mathsf{T}, 0)$	bounded DSA and SpO	t/h_{b12}
$(d_{t/h}, \mathsf{T}, v_{t/h})$	bounded DSA, SpO and AVC	t/h_{b123}
$(d_{t/h}, \mathsf{F}, v_{t/h})$	bounded DSA and AVC	t/h_{b13}
$(n, \mathsf{T}, 0)$	unbounded DSA and SpO	t/h_{u12}
$(n, \mathsf{T}, v_{t/h})$	unbounded DSA, SpO and AVC	t/h_{u123}
$(n, \mathsf{F}, v_{t/h})$	unbounded DSA and AVC	t/h_{u13}

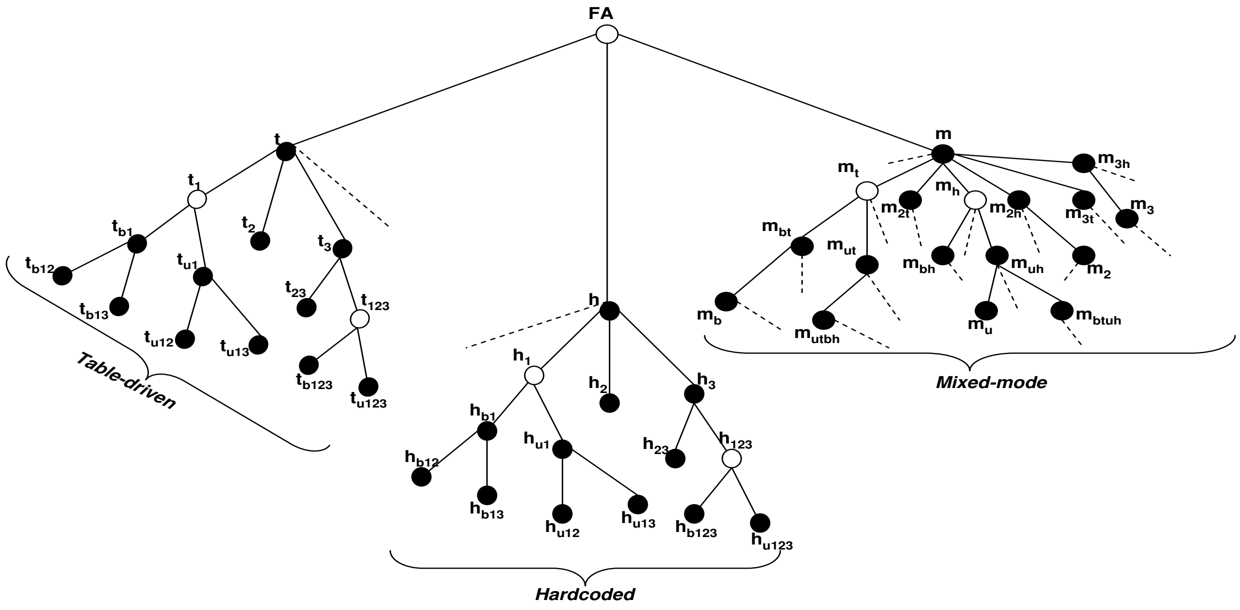
Spectrum of the Algorithms (MM)

- DSA Strategy: The variable $D_t \in \{0, d_t, |\mathcal{Q}_t|\}$, $d_t < |\mathcal{Q}_t|$; $D_h \in \{0, d_h, |\mathcal{Q}_h|\}$, $d_h < |\mathcal{Q}_h|$
- SpO Strategy: The variable $P_t \in \{\texttt{T}, \texttt{F}\}$; $P_h \in \{\texttt{T}, \texttt{F}\}$
- AVC Strategy: The variable $V_t \in \{0, v_t\}$, $v_t < |\mathcal{Q}_t|$; $V_h \in \{0, v_h\}$, $v_h < |\mathcal{Q}_h|$
- Resulting in $3 \times 3 \times 2 \times 2 \times 2 \times 2 = 144$ different algorithms

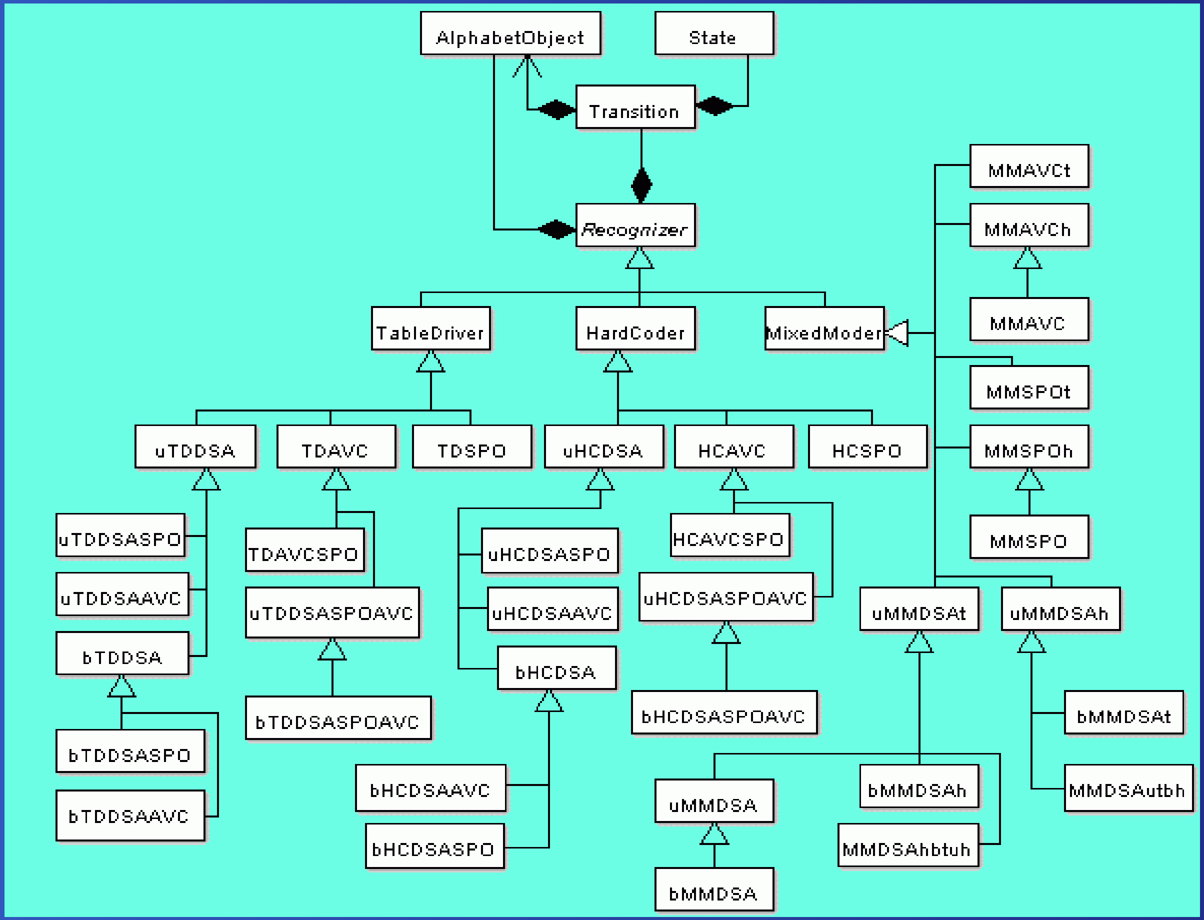
Spectrum of the Algorithms (MM)

MM formalism	Active strategies	MM Name
$(0, 0, F, F, 0, 0, s)$	None	m
$(0, d_h, F, F, 0, 0, s)$	bounded DSA on HC	m_{bh}
$(0, \|\mathcal{Q}_h\|, F, F, 0, 0, s)$	unbounded DSA on HC	m_{uh}
$(d_t, 0, F, F, 0, 0, s)$	bounded DSA on TD	m_{bt}
$(d_t, d_h, F, F, 0, 0, s)$	bounded DSA on TD and HC	m_b
$(d_t, \|\mathcal{Q}_h\|, F, F, 0, 0, s)$	bounded TD-DSA and unbounded HC-DSA	m_{btuh}
$(\|\mathcal{Q}_t\|, 0, F, F, 0, 0, s)$	unbounded DSA on TD	m_{ut}
$(\|\mathcal{Q}_t\|, d_h, F, F, 0, 0, s)$	unbounded TD-DSA and bounded HC-DSA	m_{utbh}
$(\|\mathcal{Q}_t\|, \|\mathcal{Q}_h\|, F, F, 0, 0, s)$	unbounded DSA on TD and HC	m_u
$(0, 0, T, T, 0, 0, s)$	SPO on TD and HC	m_2
$(0, 0, T, F, 0, 0, s)$	SPO on TD	m_{2t}
$(0, 0, F, T, 0, 0, s)$	SPO on HC	m_{2h}

A taxonomy graph



The Toolkit: An extract class-diagram



Applications: NLP

1. *Efficient Dictionary representation*
    - Very fast implementation based on FAs
2. *Morphological Analysis*
    - FA-based Morphotactics
    - FA-based Segmentation (Approximate PM)
    - FA-based N-gram analysis (?)
3. *Acoustic Analysis*
    - Related to Morphology(but now Phonetic)
4. *Part-Of-Speech Tagging*

Applications: Genomic/DNA Analysis

1. *Efficient for **short** DNA representation*
    - Only 4 alphabet symbols ($ACGT$)
2. *Challenge in general genomic*
    - DNA sequence very long (Text unknown in advance)
    - Pattern to investigate unknown (for Virus scanning)
3. Current MSc exercise
    - FA-based DNA Analysis

Applications

Network Intrusion Detection

1. *Efficient for pattern matching*
    - Predefined alphabet symbol set
    - known frequently used symbols (SpO?)

Questions?