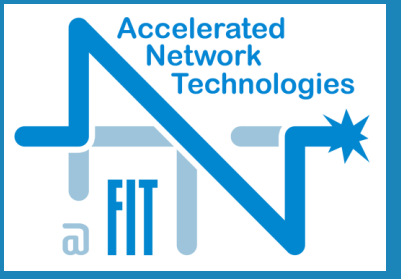
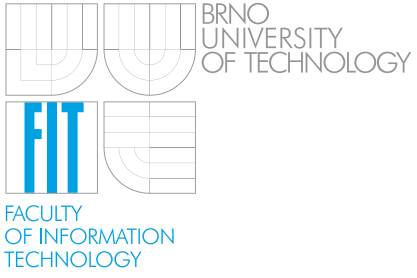
State of the art: Pattern match



Jan Kaštil

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 00 Brno, CZ
www.fit.vutbr.cz/~ikastil



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Obsah

3 **Motivace**

5 **Teoretický přístup**

9 **Vysokorychlostní vyhledávání**
- Výpočetní platformy
- Implementace NFA
- DFA
- Kombinace NFA a DFA

22 **Další zrychlování**
- Víceznakové přístupy
- Optimalizované zpracování
- Paralelní jednotky

25 **Srovnání přístupů**

26 **Závěr**

Motivace

Antiviry

- ClamAV – prohledává síťová data na výskyt virů a malware
- Pouze jednoduché vzory
- Vzory jsou dlouhé a je jich mnoho
- Prohledávají celý síťový provoz

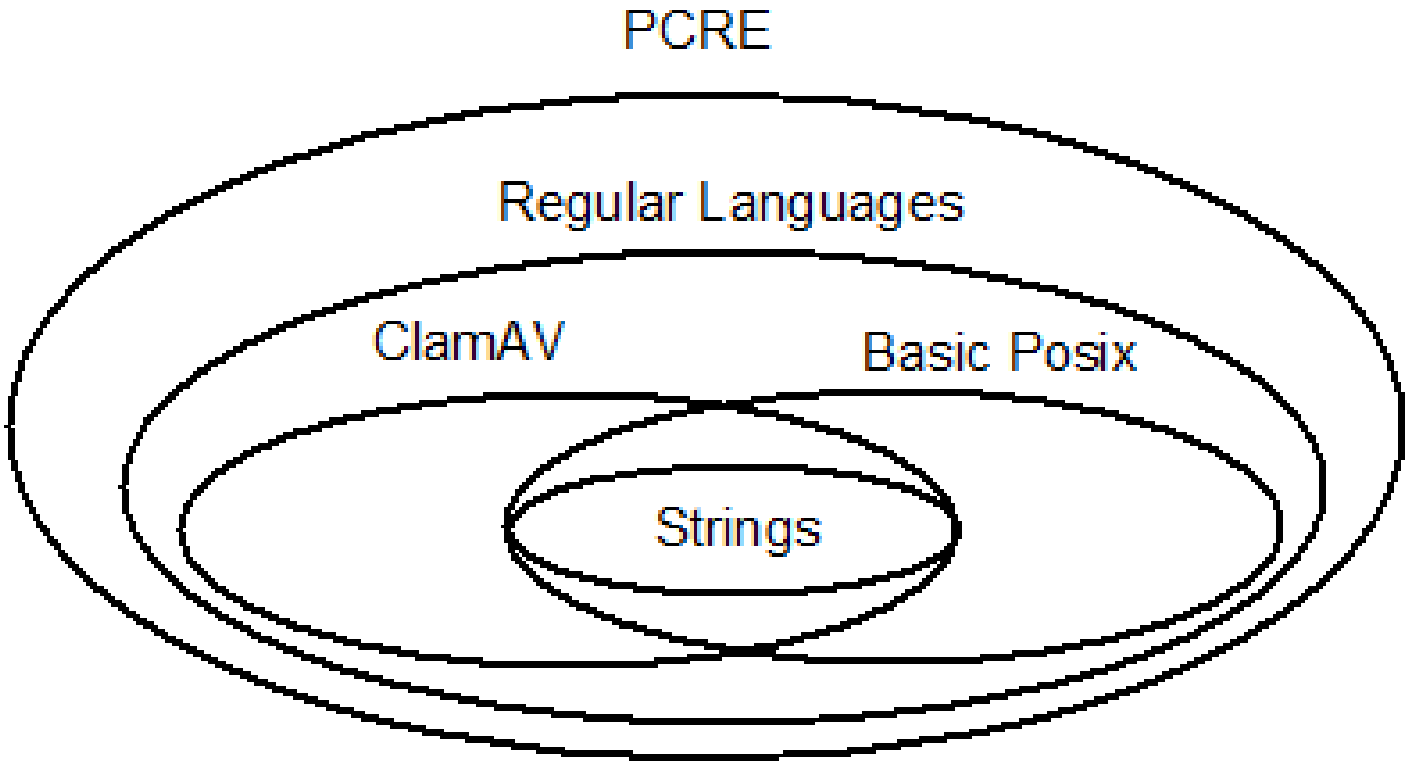
IDS – Intrussion Detection Systémy

- IDS založené na signaturách vyhledávají vzory v payloadu
- IDS založené na zranitelnostech využívají vzory pro parsování protokolů
- Několik tisíc komplikovaných vzorů
- Údajně stačí prvních pár KB z každého toku

Rozpoznávání protokolů

- Maximálně stovky jednoduchých vzorů
- Pár stovek bytů z každého protokolu

Používané jazyky



Síly jazyků

- PCRE pouze Snort
- Většina vzorů popisuje regulární jazyky
- Zápisi regulárních výrazů zjednodušeny
  - Třídy znaků
  - Počty opakování

Definice regulárního výrazu

Let Σ be an alphabet. The regular expressions over Σ and the languages that these expressions denote are defined recursively as follows:

1. $\emptyset$ is a regular expression denoting the empty set.
2. ϵ is a regular expression denoting $\{\epsilon\}$
3. a, where $a \in \Sigma$, is a regular expression denoting $\{a\}$
4. If r and s are regular expressions denoting the languages R and S, respectively, then
   (a) $(r \bullet s)$ is a regular expression denoting RS
   (b) $(r + s)$ is a regular expression denoting $R \cup S$
   (c) (r^*) is a regular expression denoting R^*.

Regulární jazyky

- Popsané regulárními výrazy a nebo konečnými automaty
- V praxi se používají mnohá rozšíření
  - Třídy znaků, Iterace s konkrétní hodnotou
  - Rozšířené regulární výrazy zahrnují i ty, co zvyšují sílu jazyka

Nedeterministický konečný automat (NFA)

Nedeterministický konečný automat je pětice $M = (Q, \Sigma, \delta, q_0, F)$, kde

- Q je konečná množina stavů
- Σ značí vstupní abecedu
- $\delta : Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$ je přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $F \subset Q$ je množina koncových stavů

Deterministický konečný automat (DFA)

Deterministický konečný automat je pětice $M = (Q, \Sigma, \delta, q_0, F)$, kde

- Q je konečná množina stavů
- Σ značí vstupní abecedu
- $\delta : Q \times \Sigma \rightarrow Q$ je přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $F \subset Q$ je množina koncových stavů

Konstrukce NFA

McNaughton a Yamada – 1960

- Vychází z definice regulárního výrazu
- Sestrojí automaty pro každý použitý symbol abecedy
- Sestrojí automaty pro každou konkatenaci, alternaci a iteraci

Thompsonův Automat – Ken Thompson – 1968

- Překladač RV do vyhledávacího programu
- Podobný přístup jako McNaughton a Yamada

Glushkovův Automat – Glushkov – 1961

- Každému symbolu abacedy přiřadí jeden stav
- Vygeneruje množiny first, follow, last, kde
  - First – je množina prvních stavů
  - Last – množina koncových stavů
  - Follow – množina následníků daného stavu
- Vždy m+1 stavů pro RV s m symboly a až m^2 přechodů

Vlastnosti vyhledávání regulárních výrazů

Vlastnosti Thompsonova automatu

- Počet stavů v rozmezí od $m + 1$ do $2m$, kde m je délka regulárního výrazu bez operátorů
- Pouze ϵ přechody mohou jít zpět
- Až m^2 přechodů po odstranění ϵ přechodů

Vlastnosti Glushkovova automatu

- Počet stavů vždy $m + 1$ a až m^2 přechodů
- Neobsahuje ϵ přechody
- Všechny přechody vedoucí do jednoho stavu mají stejný symbol

Konstrukce malých automatů – 2001

- Algoritmus konstrukce NFA s maximálně $m(\log m)^2$ přechody
- V roce 2006 bylo prokázáno, že tato hranice je v limitě přesná

Výpočetní platformy

Procesor

- Paralelizace na úrovni jader a částečně vláken
- Síťové procesory jsou drahé, ale mají řádově desítky rychlých jader
- Běžné procesory jsou levné, mají do desítky jader
- Hierarchije cache pamětí

FPGA

- Cenou téměř srovnatelné se síťovými procesory
- Nízkoúrovňový návrh, vysoká míra paralelizmu
- Nízké frekvence a spotřeba

Grafické procesory

- Hodně nových přístupů v poslední době
- Vysoká míra paralelizmu na úrovni vláken
- Vysoká hodinová frekvence

Implementace NFA na procesoru

Thomsonův přístup

- Nutná znalost strojového jazyka IBM 7094 a algolu-60
- Základní algoritmus předpokládá korektní RV v polské notaci
- Pro každý primitivní RV sestavý kód který jej přijímá a tyto kódy pak spojuje pomocí pointerů

Navarro – 2002 až 2004

- Vychází z Glushkovova automatu
- V registru si uchovává 1 bitovou representaci ke každému stavu (32 bit = 32 stavů)
- Logický součet množin Follow lze předpočítat do tabulky T: $2^{m+1} \rightarrow 2^{m+1}$
- Pro snížení paměťové náročnosti lze tabulku horizontálně dělit
- Lze považovat za DFA

Implementace NFA v FPGA

Floyd a Ullman

- Jedna z prvních method (1982)– ještě pro obvody PLA
- Vychází z klasické konstrukce NFA z RV
  - Spojování jednoduchých automatů do složitějších ϵ přechody
- Stavy automatu jsou implementovány Latch, přechody pak pomocí propojů a logických hradel
- Řeší rozmístění automatů na čipu, aby dosáhli optimálního využití

Sidhu a Prasanna – FCCM2001

- První praktická implementace NFA v rekonfigurovatelné logice (2001)
- Stejný princip jako v Floyd a Ullman, ale do FPGA
- Diskutují možnost samorekonfigurace pro zrychlení změny vzorů
- Použito pro prohledávání textových souborů

Clarkův přístup

- Rozšiřuje automat o předřazený dekodér –FPL2003
  - jednoduší routování a méně komparátorů
- Umožňuje zpracovávat více znaků v jednom hodinovém cyklu - FCCM2004
- Dosáhli propustnosti téměř 100Gbps
- Nedostupné zdrojové kódy

Znovu Prasanna – ANCS2008

- Zavádí REME, což je jednotka pro hledání dané množiny RV
- Princip vyhledávání shodný s předchozím slajdem
- Nový algoritmus pro zvýšení počtu zpracovaných symbolů
- Předřazený dekodér znaků společný pro více REME
- Zdrojové kódy pro generování REME jsou dostupné

Implementace NFA v FPGA

Tsern-Huei –Transaction on Computers (2006 – 2009)

- Založeno na Glushkovově automatu
- Rozšířená Shift-OR architektura
- Podpora zpracování více znaků
- Popisují algoritmus možné redukce počtu stavů
- Propustnost 4Gbps na ML310 pro jeden RV
- Pro zvolené RV mají zřejmě menší spotřebu logiky než Prasanna a Sidhu, ale používají BlockRAM
- K určení následujících stavů používají OR a AND nad bitovými vektory

Ming Cong – ICCSA 2010

- Navrhují rozšíření pipeline procesoru o jednotku podporující NFA
- Implementace rozšiřující jednotky v FPGA

Implementace NFA v FPGA

Hao Wang – FPGA 2010

- Základním blokem je CCR – Character Class with Constrain Repetition
- Každý CCR blok si pamatuje informaci o své aktivitě
- Aktivní CCR může aktivovat navazující CCR
- Propojení a konfigurace CCR může být změněna pomocí částečné dynamické rekonfigurace
- Každé CCR obsahuje dva čítače pro podporu contrain repetition
- Snadné rozšíření pro podporu backreferencí
- 250Mhz na Virtex5

Problémy k řešení

- Neoptimálnost Thomsonova algoritmu
  - Vytvořený automat má zbytečné stavy a přechody
  - Možno hledat různé redukce – teorie automatů
- Hledání nových mapování
  - Současné mapování pochází z roku 1982

Akcelerace NFA v GPU

SIMT architektury

- Singe Instruction Multiple Threads
- Vlákna mají stejný program, ale pracují nad rozdílnými daty
- Paměť je připojena extrémně rychlou sběrnicí
  - stále je ale bottleneck

Přístupy

- Implementační zaměření
- Různé úrovně paralelizmu
  - Paketová – každé vlákno zpracovává jiný paket
  - Výrazová – každé vlákno implementuje svůj regulární výraz
  - Přechodová – každé vlákno má svých několik přechodů
- Vysoká propustnost
  - GrAVity: 20Gbps na ClamAV vzorech – RAID 2010
  - iNFAnt: 3x lepší než HFA s menšími paměťovými nároky – SIGCOM2010
  - Nepublikovaná latence jednoho paketu
- Většina GPU používá DFA

Delayed DFA

- Dragon book – bible překladačníků
- Zavedení "defaultních"přechodů pro snížení velikosti přechodové tabulky
  - Nepřijímá žádný symbol
  - Provede se pouze pokud nelze provést přechod akceptující symbol
  - Pokud dva stavy spojené "defaultním"přechodem mají stejnou přechodovou funkci, můžeme přechody z prvního z nich odmazat
- Sailesh Kumar – SIGCOM 2006
- Libovolně dlouhé sekvence defaultních přechodů
- Nalezení vhodné sekvence přechodů je NP–Hard
  - Použití Kruskalova algoritmu

Zlepšení DDFA – Becchi

- Delayed přechody mohou vést pouze zpět
  - Maximálně 2 dotazy do paměti na znak
  - Jednodušší hledání optimálních přechodů – lepší výsledky

Content Adressing DDFA – Sailesh Kumar ANCS 2006

- Umožňuje odstranit zbytečné dotazy do paměti u DDFA
- Stavy nejsou označovány čísly, ale řetězci (Content label)
  - Umožňují zjistit jaké přechody vedou z daného stavu
  - Umožňují zjistit kam vede defaultní přechod z daného stavu
- Adresa následujícího stavu v paměti se určí hash funkcí
  - V principu se jedná o perfektní hashování
  - Místo hledání hash funkce upravují hodnotu klíčů
- Hledání optimálního CDDFA není triviální
  - Snaha o co nejmenší DDFA
  - Stavy s hodně vstupními hranami mají mít krátké názvy

Ficara – SIGCOM 2008

- Místo defaultních přechodů zavádí cache
- Předpokládá, že přechody aktivního stavu jsou uloženy v malé paměti na čipu (cache), ale celá přechodová tabulka se nachází mimo čip
- Místo stažení přechodové funkce se updatuje cache

Sailesh Kumar – ANCS 2007

- Identifikuje tři základní problémy konečného automatu
  - Insomnia – zbytečně velký a rychlý automat
  - Amnesia – stavová informace je uložena jen ve stavu
  - Acalculia – automat neumí počítat
- Rozšíření konečného automatu o pomocnou paměť
  - Nezvyšuje popisnou sílu jazyka
  - Zvyšuje kompaktnost representace

History based FA (H-FA)

- Přidána sada flagů
- Přechody mohou nastavovat nebo resetova flagy
- Každý flag efektivně zdvojnásobí množinu stavů
- Přechody mohou být podmíněny hodnotou flagu

History based Counting FA (Hc-FA)

- Přidává čítač pro řešení constrained repetition

Randy Smith – Extended Finite Automaton (XFA)

- SIGCOM 2008, Oakland 2008, ICISS 2008 (invited paper)
- Klasický DFA rozšířený o pomocnou paměť
- Přechody mohou do paměti zapisovat nebo číst
- Stejná myšlenka jako Hc-FA, ale formálněji uchopena

Asynchronous Paralel Finite Automata – Yang Li 2009

- Určený pro clustery počítačů
- Vychází z pozorování, že většina počítání v FA je čekání na jeden nebo několik znaků
- Jednotka je rozdělena do dvou asynchroních částí
  - Vyhledávací část
  - Počítací část
- Počítací část se pohybuje o n znaků před vyhledávací
- Počítací část ví, jak daleko je hledaný znak
- Pokud vyhledávací část vstoupí do čekání, podívá se na vzdálenost a hned ví, zda může pokračovat

Brodie – 2006

- Využívá RLE kompresi přechodové tabulky spolu s víceznakovým zpracováním
- Ke každému symbolu abecedy náleží sekvence následujících stavů kódovaná RLE
- Počáteční a koncová pozice sekvence se zjisí pomocí nepřímé adresace

Rozdíly mezi řetězci a regulárními výrazy

- Řetězce neobsahují smyčky a třídy znaků (alternace)
  - Řetězce lze implementovat pomocí stromových algoritmů (Aho-Corasick)
- Smyčky v regulárních výrazech jsou příliš krátké
- HEXA (ICNP 2007)
  - Hashovací struktura pro průchod stromem
  - 3 až 5 krát menší paměťové nároky než Aho-Corasick
  - 2 krát menší paměťové nároky než treebitmap
  - Žádné zlepšení na Regulárních výrazech

Kombinace NFA a DFA

Sailesh Kumar – ANCS 2007

- Většina provozu aktivuje pouze prvních několik stavů automatu
- Tyto často aktivní stavy v DFA, zbytek v NFA
  - NFA je malé, ale pomalé
  - Nutno detekovat DOS útoky

Kořenek – ANCS 2010

- Rozdělení automatu mezi DFA a NFA pro snížení paměťové náročnosti

SangKyun Yun – FPL2010

- Hledají v NFA sousedící skupiny stavů, kde pouze dva mohou být aktivní
- Lze rozšířit na k součastbně aktivních, ale nebude stačit 1LUT-registr

Multistrides

Základní princip

- Automat akceptuje více znaků v rámci jednoho přechodu
- Až exponenciální nárust velikosti abecedy automatu
- Využití tříd znaků pro kompresy abecedy

Brodie – 2006

- Vytvoří veškeré použité kombinace symbolů
- Pak hledají třídy ekvivalence

Becchi – ANCS2008

- Vytvoří třídy znaků na jednoznakové abecedě
- Rekurzivně spojuje symboly (1,2,4,8,...)

Norio Yamagaki – FPL2008

- Stejný přístup jako Becchi, ale aplikovaný na NFAa FPGA
- Rychleji konstrukce automatu než u Clarka

Optimalizované zpracování

Preprocesing

- Většina provozu na síti není útok
- Provoz se zpracovává filtrem a regulární výraz se hledá jen v podezřelých paketech
  - Chybující vyhledání
  - Hledání řetězců či podřetězců
  - Přeskakování znaků

Jen vyžadovaný provoz

- Většina důležitých informací se nachází z počátku síťového toku či při změně jeho směru
- Limmer – CCS 2010
  - První 2KB komunikace je dostatečné pro detekci 98% útoků
  - Změna směru toku ukazuje na další komunikaci
  - V rámci jednoho TCP spojení změřili až devět komunikací
- Cascarano – SAC 2010
  - Prvních 256 bytů stačí pro přesnou detekci protokolu v UDP
  - Prvních několik paketů v komunikaci stačí pro detekci TCP

Paralelní jednotky

Problémy paralelizace

- Vstupní data se musí bufferovat – nesouvisí s hledáním
- Každá jednotka musí mít přístup ke své paměti – různé optimalizace
- Vhodné pro zpracování per paket kde se neuchovává stav jednotky v rámci toku
- Při použití per flow nemusí dostačovat propustnost pro rychlé toky

Jedna jednotka pro více toků

- ICC 2010 – Junchen Jiang
- DFA má více aktivních stavů, pro každý flow jeden
- Přechodová tabulka je rozdělena do nezávislých pamětí
- Dosáhli zrychlení $0,5k$ až $0,7k$, kde k je počet modulů

Sdílení paměti mezi jednotkami

- SIGCOM 2006 – Sailesh Kumar
- Více DDFA zdílí mnoho paměťových jednotek

Srovnáváme hrušky s jabkama

Co lze srovnávat?

- Rychlost
- Spotřebu zdrojů
- Spotřebu paměti
- Spolehlivost vyhledání

Nad čím srovnávat

- Různé množiny pravidel
- Různé datové sady
- Zdrojové kódy nedostupné a nebo pouze na požádání

Pro co srovnávat

- Každá úloha má jiné požadavky
- Často existuje vhodnější řešení než vyhledávání regulárních výrazů

Závěr

Využitelnost pattern match

- Každé využití klade své vlastní nároky
- Velká mezera mezi výzkumem a implementací

Nezkoumáme už vyzkoumané?

- Nezohledňujeme výsledky teoretického výzkumu
- Stále se objevují nové přístupy
- Mnoho bílých míst

Několik základních přístupů

- NFA versus DFA
- Mnoho různých implementací
- Snaha využít specifické vlastnosti síťového provozu

Otázky ????