doi:10.11959/j.issn.1000-436x.2015101

基于多维有限自动机的 DFA 改进算法

宫阳阳[1]，刘勤让[1]，杨镇西[1]，邵翔宇[1]，邢池强[1]，焦慧娟[2]，彭志彬[1]
（1. 国家数字交换系统工程技术研究中心，河南 郑州 450002；2. 中国石油大学 化工学院，山东 青岛 266555）

摘 要：多个正则表达式规则编译成一个 DFA(deter minister finite automata)时，会产生状态爆炸、存储急剧增加的现象。针对最严重的状态爆炸问题，从信息论的角度给出了解释，并提出多维数学模型，将冗余状态分为 0 维状态和 1 维状态，通过前者按照维度压缩，后者动态构建的方法将空间复杂度降到理论下界，并在此基础上提出多维有限自动机(MFA, multi-dimensional finite automata)。实验表明，MFA 构造时间比 XFA 略少，比 DFA、STT 冗余压缩算法和 Hybrid-FA 降低了 2~3 个数量级；存储空间比 XFA 略高，比 DFA、STT 冗余压缩算法、mDFA、Hybrid-FA 降低了 1~2 个数量级；匹配时间比 DFA、Hybrid-FA 略多，但是比 XFA 略少，比 STT 冗余压缩算法和 mDFA 降低了 1~2 个数量级。

关键词：正则表达式；DFA；有限自动机；状态爆炸
中图分类号：TP393 **文献标识码**：A

Improved DFA algorithm based on multi-dimensional finite automata

GONG Yang-yang[1], LIU Qin-rang[1], YANG Zhen-xi[1], SHAO Xiang-yu[1],
XING Chi-qiang[1], JIAO Hui-juan[2], PENG Zhi-bin[1]
(1. National Digital Switching System Engineering R&D Center, Zhengzhou 450002, China;
2. Collge of Chemical Engineering, China University of Petroleum, Qingdao 266555, China)

Abstract: Compiling multiple regular expression signatures into a combined DFA can blowup in state and storage space. Explanations from the prospective of information theory and multi-dimensional mathematical model were proposed focusing on the most serious state explosion. Redundancy states were divided into zero-dimensional ones and one-dimensional ones. The former were compressed by dimension, and the later were dynamically built. The space complexity of the model came to the theoretical lower bound by the above methods. Then the multi-dimensional finite automata (MFA) was proposed with the model. Experiments show that, the construction time taken by MFA is slightly less than XFA and is 2~3 orders of magnitude lower than DFA, STT redundancy compression algorithms and Hybrid-FA; the memory space of MFA is slightly higher than XFA, but is 1~2 orders of magnitude lower than DFA, STT redundancy compression algorithms, mDFA and Hybrid-FA; in the aspect of matching time, MFA ranks after DFA and Hybrid-FA, but ranks before XFA, and it achieves 1~2 orders of magnitude lower than that of STT redundancy compression algorithms and mDFA.

Key words: regular expression; DFA; finite automata; state explosion

收稿日期：2014-02-04；**修回日期**：2014-04-01
基金项目：国家高技术研究发展计划（“863”计划）基金资助项目（2011AA01A103, 2011AA01A101）；国家重点基础研究发展计划（“973”计划）基金资助项目（2012CB315901, 2013CB329104）；国家科技支撑计划基金资助项目（2011BAH19B01）
Foundation Items: The National High Technology Research and Development Program of China (863 Program)(2011AA01A103, 2011AA01A101); The National Basic Research Program of China (973 Progrma)(2012CB315901, 2013CB329104); The National Key Technology R&D Program (2011BAH19B01)

1　引言

近年来，以“棱镜门”为代表的各类监控和攻击事件为各国的网络安全形势敲响了警钟。在网络信息安全领域，入侵检测系统（IDS, intrusion detection system）扮演着越来越重要的角色，IDS 采用深度分组检测（DPI, deep packet inspection）进行病毒检测、入侵识别、协议分析等，网络安全专家针对各种入侵行为的特点建立一组特征模式集，当发现给定数据符合某一特征模式时会采取一定的措施。随着攻击模式的多样化，正则表达式由于其灵活高效的表达能力逐渐成为 DPI 特征模式的主流描述语言。正则表达式匹配算法采用非确定性有限自动机（NFA, non-deterministic finite automata）和确定性有限自动机（DFA, deterministic finite automata）实现[1]。相对而言，DFA 匹配速度快，适合应用在高速数据链路中。

DPI 规则集一般包含了多条规则，为了快速完成对多条规则的匹配，一般将它们编译成一个联合 DFA，然而由于正则表达式之间的相互重叠和影响，联合 DFA 可能存在状态空间爆炸问题，需要巨大的存储空间，目前硬件水平无法满足要求，因此研究者提出了多种改进算法。图 1 是联合 DFA 的构造过程，首先网络安全专家制定一系列安全规则，组成一组规则集；然后将规则集内的特征集（注：下文提到的“规则”均仅指正则表达式特征）通过 Thompson 构造法编译为 NFA；通过子集构造法生成 DFA 并进行最小化。描述 DFA 的简要记号有状态转移图（STD, state transition diagram）及状态转移表（STT, state transition table）。一是从 DFA 构造过程的角度来分类，当前算法改进主要集中在 4 个方向。Yu 等[2]提出对正则表达式规则分组（mDFA）的思想，将联合编译后不会产生状态爆炸的规则放在一组内，然后按照这个规则在这一个分组里不断添加，当达到一定的阈值后再开始创建另一个分组，这种方法极大地降低了规则之间相互作用产生的状态爆炸，然而降低了匹配效率，若分组数为 k，则匹配效率降低为原来的 $1/k$，徐乾等[3]提出一种更为合理的分组方法。二是调节 NFA 到 DFA 的转化程度，避免对可能产生状态爆炸的规则进行确定化，从而减少冗余状态[4~8]，典型算法为 Hybrid-FA[4]。这类算法当 NFA 部分频繁被激活时处理速度急速退化，因此容易受到此方面的恶意攻击。三是通过对 DFA 的二维 STD 进行分析，利用辅助变量和计数器记录自动机历史信息，从而减少冗余状态，典型算法有 History-FA[9]、XFA[10]、SFA[11]等。这类算法规则覆盖率较低，并且付出了的时间代价。四是针对 STT 进行压缩[11~17]，典型算法有 D^{2}FA[11]、State merging[12]，由于其处于整个改进流程的末端，压缩级别是线性的，因此无法从根本上解决多项式级别乃至指数级别的爆炸问题[18]。

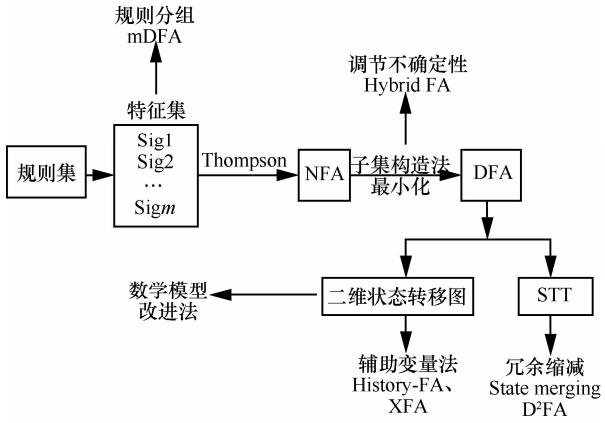


图 1　DFA 改进算法分类

从图 1 的流程来看，目前缺少数学模型方面的探索与研究。图 2 是 DFA 的二维状态转移[19]，该数学模型于 1960 年便应用在有限自动机理论中继续研究，目前几乎所有的研究分析都是建立在它的基础上。然而这种传统的二维数学模型存在不足之处是，冗余信息湮没在杂乱的二维结构中，因此本文提出多维状态转移图的数学模型，给出冗余状态的本质解释，并在此基础上进行压缩。此外，上面综述的改进算法是 NFA 和 DFA 的空间复杂度与时间复杂度的折中，如图 3 所示，目前的 DFA 改进算法的存储缩减是以牺牲部分时间复杂度换取的，还没有算法能够同时达到 DFA 的时间复杂度和 NFA 的空间复杂度下界。本文针对一类特定的规则，在多维状态转移模型的基础上设计了多维匹配算法。在模型和算法的基础上提出多维有限自动机 MFA，理论证明，MFA 同时达到了两者的下界。

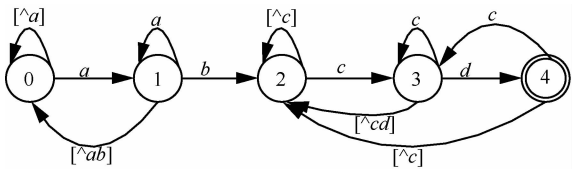


图 2　/.*ab.*cd/的 DFA 状态转移

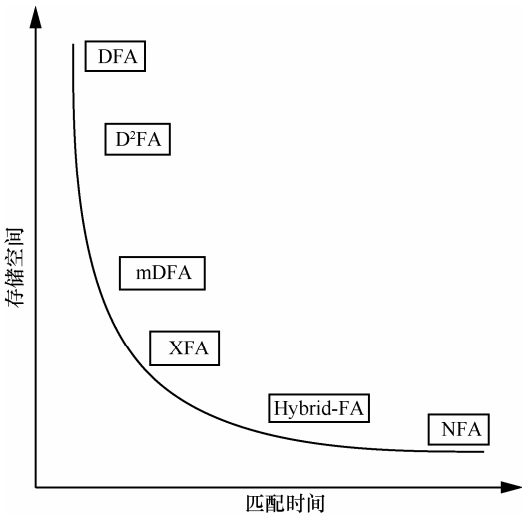


图 3 DFA 改进算法复杂度示意

针对上面的 2 个问题，本文从解决一类特定规则相互混合产生的爆炸问题出发，提出新型有限自动机 MFA。该类规则由确定字符串及克林闭分组“.*”线性叠加而成，这类规则产生的爆炸问题也是目前所发现的所有状态爆炸问题中最为严重的[18]。

2 多维立方体相关理论

本节首先介绍图论中 m 维立方体的概念以及相关性质。

定义 1 设有序实数$(x_1, x_2, \cdots, x_m)$表示 m 维空间中的一个点，则 m 维空间即为集合 $V_m=\{(x_1, x_2, \cdots, x_m) |$其中 x_i 属于 R，$i= 1, 2, \cdots, m\}$。

若$(x_1, x_2, \cdots, x_m)$中 m 个坐标均为常数，则表示 m 维空间中的一个点，可称为 0 维面（或点）；若$(x_1,x_2,\cdots,x_m)$中 $m-1$ 个坐标为常数，只有一个坐标为变量，则表示 m 维空间中的一条直线，可称为一维面（或边）；若$(x_1,x_2,\cdots, x_m)$中 $m-2$ 个坐标均为常数，另有 2 个坐标为变量，则表示 m 维空间中平行于坐标平面的一个平面，可称为二维面；若$(x_1,x_2,\cdots,x_m)$中 $m-n$ ($n\leqslant m$)个坐标均为常数，另有 m 个坐标为变量，这些点的集合可称为 n 维面。

定义 2 $V_m=\{(x_1,x_2,\cdots,x_m) \mid 0\leqslant x_i\leqslant 1,2,\cdots,m\}$ 称为 m 维立方体。

m 维面为 x_i 在[0,1]内有 m 个坐标任意取值，而有$(m-n)$个坐标取常数(0 或 1)的点组成，因而个数有

$$m_n = 2^{m-n}C_n^m \tag{1}$$

当 n=0 时，m 维立方体 0 维面的数目为

$$m_0 = 2^m \tag{2}$$

当 n=1 时，m 维立方体一维面的数目为

$$m_1 = 2^{m-1}m \tag{3}$$

定义 3 已知图 $G=(V, E)$，若 G 满足

$V(G)=\{ x_1x_2\cdots x_m: x_1\in \{0, 1\}, i=1, 2,\cdots, m\}$

并且若 $u=x_1x_2\cdots x_m$, $v= y_1y_2\cdots y_m\in V(G)$, 有

$$uv\in E(G) \Leftrightarrow \sum_{i=1}^{m}|x_i - y_i| = 1$$

则称图 G 为 m 维立方体，并记作 $Q_m=(V(Q_m), E(Q_m))$。

图 4 中分别是 Q_1、Q_2、Q_3、Q_4，其中 Q_1、Q_2、Q_3 在三维世界可以找到对应实物，而四维以上的立方体只能依靠想象。值得注意的是纸张是二维的，因此超过二维的图像都是无法在纸面上真实展示的。

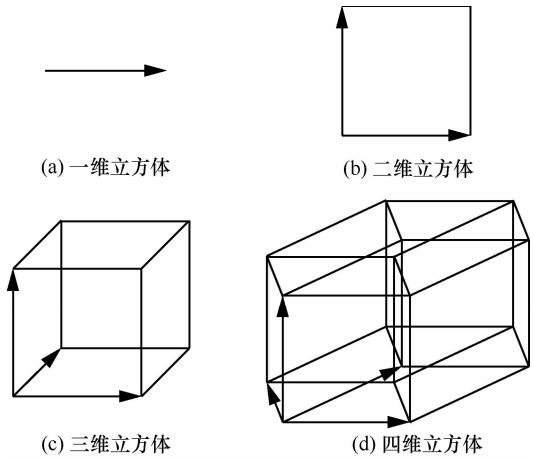


图 4 多维立方体示意

在代数中定义的 m 维面数目，式(1)~式(3)适用于图论中定义的 m 维立方体（如定义 3）。

定理 Q_m 是 m-正则简单图。

各个顶点的度为 m，即每个顶点与 m 条边相连。

3 多维状态转移模型

本节首先介绍一类最典型也是最严重的状态爆炸现象，然后从信息论的角度进行冗余分析，给出多维状态转移图（MSTD, multi-dimensional state transition diagram），最后在此基础上给出冗余解决方案。

3.1 DFA 状态爆炸问题

DFA 状态爆炸的一种典型情况是，多条正则表达式编译成一个联合 DFA 时，由于规则间的相互作用，且为了保证 DFA 的状态确定性，联合 DFA 需要记录所有状态可能的组合情况，从而产生状态爆炸现象。其中最为严重的一种由克林闭分组“.*”

造成。一般而言，如果有 k 个正则表达式，每个表达式中出现 x 次“.*”，则需要 $O((x+1)^k)$个状态构建 DFA，在这种情况下即使某个规则增加一个“.*”也会使状态数目加倍[18]。如表 1 所示，在常见的 IDS 规则集中含有大量这样的规则，例如 2013 年 12 月发布的 snort-snapshot-2 946[20]中“.*”出现了 3 064 次，共有 1 291 条规则；在 2009 年 5 月发布的 L7-filter protocol[21]中含有“.*”的规则共有 47 个。制定如表 2 所示的规则集，采用 BECCHI M 提供的软件 RegEx Processor[22]分别生成 DFA 和 NFA，状态数统计如表 3 所示，定义膨胀率 P = DFA 状态数/NFA 状态数，可以发现 P 几乎随规则数目呈指数级别增长。通过以上的分析可知，将这些规则编译成联合 DFA 需要巨大的空间，因此传统的 DFA 针对于“.*”类规则无法直接应用在实际中。

表 1　snort 和 l7-filter 规则中“.*”数目统计

规则集	snort-snapshot-2 946	L7-filter protocol
规则数目	29 822	114
正则表达式	24 119	114
“.*”出现次数	3 064	62
含“.*”规则数	1291	47
比例	4.3%	41.2%

表 2　　测试规则集

编号	规则
1	/.*ab.*cd/
2	/.*ef.*gh/
3	/.*ij.*kl/
4	/.*mn.*op/
5	/.*qr.*st/
6	/.*uv.*wx/
7	/.*yz.*12/
8	/.*34.*56/

表 3　仿真得到的 NFA 与 DFA 状态数目统计

规则数目	状态数目		P
	NFA	DFA	
1	7	5	n/a
2	12	16	1.33
3	17	44	2.58
4	22	112	5.09
5	27	272	10.1
6	32	640	20.0
7	37	1 472	39.8
8	42	3 328	79.2

首先分析/.*S_i.*S_i'/类型规则联合编译产生的状态爆炸问题，其中 S_i 和 S_i'为精确字符串。作为例子，取 S_1=*ab*，S_1'=*cd*，S_2=*ef*，S_2'=*gh*。如图 5 所示，当规则 Sig1: /.*ab.*cd/与 Sig2: /.*ef.*gh/联合编译时，由于状态 V、X 表示匹配任意字符，当和 Sig1 的状态进行联合编译时，所有状态都复制了一份。状态 P、Q、R、S、T 复制为 PV、QV、RV、SV、TV 以及 PX、QX、RX、SX、TX；同理对于 Sig2，状态 V、W、X、Y、Z 复制为 PV、PW、PX、PY、PZ 以及 RV、RW、RX、RY、RZ。

正则表达式与 NFA、DFA 对应了相同的语言[1]，从信息论的角度分析，两者含有的信息量是相等的，为何 DFA 的状态数比 NFA 多？Sig1 和 Sig2 对应的 NFA 状态为 5，由 Thompson 构造法的性质可知联合编译后的 NFA 状态不会超过 10，但是联合 DFA 却有 16 个状态。实际上，QX、SX、TX 仅仅是 QV、SV、TV 的简单重复，没有增加任何有用的信息量。同样，RW、RY、RZ 仅仅是 PW、PY、PZ 的简单重复。若将这些状态去掉不会减少 DFA 的任何有用信息量。

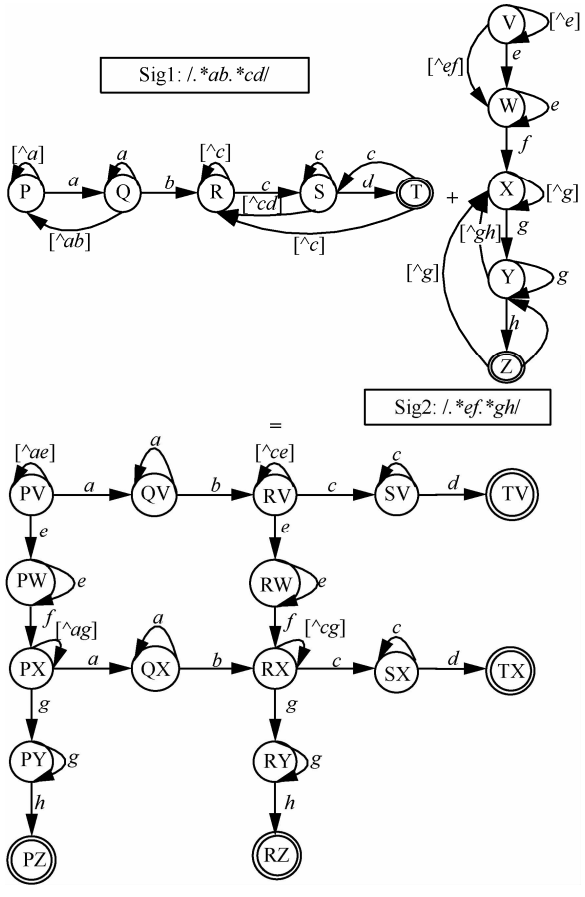


图 5　/.*ab.*cd/和/.*ef.*gh/编译为联合 DFA

Sig1 和 Sig2 中/.*/包含的信息量，其对应的语言为任意字符串，因此不确定度最大（注：这里的不确定度指概率，和自动机 NFA 的不确定性是 2 个不同的概念），因此含有的信息量也最大。而/*a*/对应的语言，在最短匹配的原则下为 *a*，不确定度最小，因此信息量也是最小的。Sig1 含有的“.*”分别对应状态 P 和 V，当遇到字符 *a* 时，/.*/的优先级小于/*a*/，因此状态进行了跳转，优先朝着信息量减小的地方前进。当编译成联合 DFA 时，每个规则含有的“.*”对应的状态会增加指向其他规则的迁移边，与其他规则的确定字符相连，优先朝着确定字符串对应的状态前进。联合 DFA 的 4 个顶点 PV、RV、PX、RX 都会与 Sig1 和 Sig2 的确定字符对应的状态相连，也就是说每个顶点都会指向其他规则跳转的迁移边。将所有状态连线，4 个交点的坐标分别计为 00、01、10、11，则对应了二维立方体的数学模型，且每个顶点的度为 2。因此图 5 的状态转移图对应了一个二维立方体(2-正则简单图)。每个规则的“.*”对应的状态为二维立方体的 0 维面，即顶点，确定字符对应的状态位于一维面上，即顶点相连的边上。实际上，*m* 维立方体的模型(*m*>2)适用于 *m* 个规则联合编译的情形。

3.2 多维状态转移图 MSTD

本节采用 $Q_m = (V(Q_m), E(Q_m))$作为状态转移图继续处理这类特殊的规则，将在这个模型下更加清晰地发现冗余状态并进行缩减。

设规则类型为：Sig*i*：/.*S_i.*S_i'/ (*i*=1, 2,…, *m*），“.*”对应的状态称为 0 维状态，其他状态为一维状态，每个规则分配 2 个顶点坐标 0 和 1，对应前后 2 个 0 维状态，*m* 维模型的顶点坐标由 *m* 个规则的“.*”对应的状态坐标决定，每个规则的一维状态分布在由 0 维状态交叉构成的边上。

图 6 为规则集：{/.*ab.*cd/、/.*ef.*gh/、/.*ij.*kl/}对应的三维状态转移图。例如顶点 110，分别对应的 Sig1 的第 1 个 0 维状态，Sig2 的第 1 个 0 维状态，Sig3 的第 0 个 0 维状态，3 个规则的 0 维状态构成顶点 110。

按照这个模型计算状态数目，作为该模型合理性的佐证。

由式(2)和式(3)可知，边数 $L = m2^{m-1}$，顶点数 $V = m2^m$，设每条边的一维状态数为 n_1，则多维立方体上的格点数为

$$T = n_1 L + V \tag{4}$$

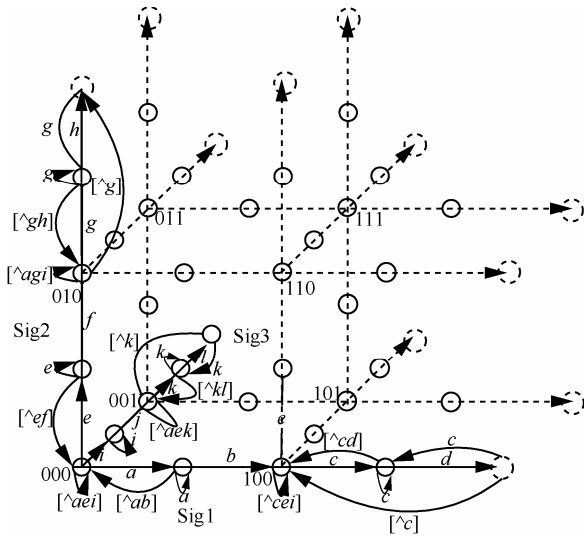


图 6 三维状态转移 3STD

因此 *m* 个规则编译成联合 DFA 对应的状态数目为

$$S = T = n_1 m 2^{m-1} + 2^m \tag{5}$$

取表 2 所示的规则集，则 n_1=3，$m = 1,2,\cdots,7,8$，按照式(5)计算得到的规则数如表 4 所示，表 4 和表 3 的对比可知式(5)的计算结果与实验结果是一致的。

表 4 计算得到的状态数目

m	DFA 状态数
1	5
2	16
3	44
4	112
5	272
6	640
7	1 472
8	3 328

由式(5)可知，“.*”产生的状态爆炸是指数级别的，目前硬件水平无法满足指数规模的存储。假设存储一个状态需要 1 KB，当 30 个规则联合编译，所需要的存储就达到了 1 TB，而一般的规则集都有数百乃至上千个规则，因此必须进行冗余缩减。

在三维模型下，能够发现冗余，而传统的 DFA 算法的状态转移模型被限制在二维的纸面上，若将图 6 中用虚线表示的冗余状态以及状态迁移边均被映射到了二维纸面上，将显得杂乱无章，因此难以从本质上去除所有冗余。而二维状态转移图不很容易分辨的状态冗余在三维情况下一览无遗。

下面介绍基于 m 维立方体的冗余状态缩减方法。如图 7 所示，由式(4)可知，状态冗余主要由两部分构成，一维状态产生的状态冗余和 0 维状态所构建的结构框架，前者的规模为 $O(m2^m)$，后者的规模是 $O(2^m)$。根据分析可知，一维冗余没有任何有用信息量，因此，仅需要记录坐标轴的状态信息即可，与坐标轴平行的状态均投影到坐标轴上，这相当于将冗余按照维度进行了压缩，压缩后规模为 $O(m)$。0 维状态作为整个模型的框架并不存在冗余，但是基于多维数学模型的 DFA 和基于二维数学模型的 DFA 都具有状态确定性和唯一性，即在任何一个时刻都只有一个状态被激活。在多维数学模型下，该状态一定位于某条边上，该边上仅有 2 个顶点，因此保存 2 个顶点的信息可确定状态位置。随着状态发生跳转，顶点信息需要动态更新，而内存中始终只有 2 个顶点的信息，每个顶点需要 $2m$ bit，因此存储规模为 $O(2m)=O(m)$。此外，0 维状态信息和一维状态信息都是从每个规则的 DFA 转移信息中读取的，还需要存储 $O(m)$个 DFA。因此，多维立方体结构模型状态空间复杂度为 $O(3m)=O(m)$。

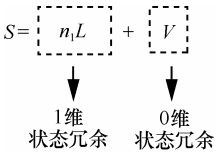


图 7　式(4)状态数目剖析

4　多维状态转移算法 MSTA

本节以三维状态转移算法的设计为例，在 MSTD 的基础上给出多维状态转移算法(MSTA, multi- dimensional state transition algorithm)的初步实现。

传统的 DFA 算法实现主要靠 STT 模拟状态转移函数实现状态转移，而对于 MSTA，不仅需要 STT，还需要辅助变量来实现与 DFA 等价的状态转移，具体结构如图 8 所示。

1) 预处理阶段。生成每个规则对应的 DFA 状态转移表 STT，静态存储在内存中，并找到对应的 0 维状态及对应的非默认转移字符。例如对于 Sig1 的状态 2，非默认转移字符为 c，辅助变量占 6 bit。currentNode 和 nextNode 共同用于确定激活状态所在边。通过两者的异或操作获取当前 DFA 指针 DFA_Pointer。

2) 初始化阶段。激活状态位于原点，为 0 维状态，辅助变量为 6’b 000 000，非默认跳转分别为 aei，DFA 指针为 NULL。

3) 字符匹配阶段。

step1　当激活状态在 0 维状态处迁移时，例如当前激活状态坐标为 101，获取非默认跳转字符分别为 cek，则当遇到字符 cek 则会跳转到一维状态，遇到其他字符不会发生跳转，并更新 nextNode 节点坐标以及跳转信息，该跳转信息从每个 DFA 的 STT 中读取，重新计算 DFA 指针，准备接受下一个字符。

step2　当激活状态为一维状态时，在当前 DFA 上进行跳转，每当跳转回 0 维状态时，进行 step1。

在采用二维状态转移图的 DFA 算法中，每接受一个字符需要进行 1 次跳转，而本算法的状态转移可能需要 2 次跳转，并且需要进行 1 次数据更新，这是本算法付出的时间代价，然而每个步骤的算法复杂都是 $O(1)$，因此并没有改变 DFA 的算法复杂度。

5　模型及算法修正

前面针对一类特殊规则，介绍了基于多维状态转移图 MSTD 的多维状态转移算法 MSTA，两者一

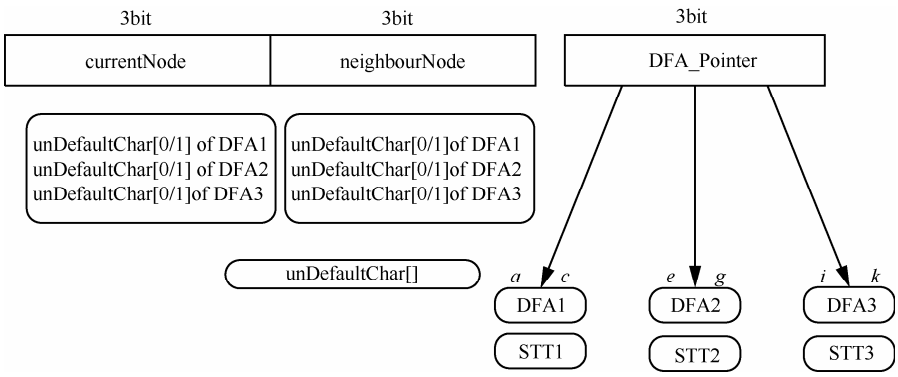


图 8　3STA 的存储结构

起构成多维有限自动机(MFA, multi-dimensional finite automata)。MFA 适用规则集类型为{/.*S_i.*S_i'/(i=1, 2, m) }。但是该规则集对称性强，在实际中基本上没有此类规则集，因此必须对模型和算法进行修正，扩展 MFA 应用范围。

5.1 字符串交叠

前面为了表述的方便，每个规则的字符串之间没有任何的交叠.交叠指的是“.*”后的字符存在相同的情况，例如对于规则：/.*abc.*def/与/.*abg.*hij/，按照 MSTA 算法，激活状态位于 00 时，当接收字符 a 时，DFA 指针为不定态，无法进行后续的跳转，记这样的字符为交叠字符。DFA 的处理方法是直接将 2 个状态合并成一个状态，但这种方法不适合 MFA。因为这会造成 MSTD 的对称性破坏，增加自动机预处理时间。为了保持结构的对称性，同时又能达到同样的效果，需要在匹配时构建一个缓存区动态建立一组等价状态同时表示 2 个规则。具体的处理算法如下。

当出现非默认跳转字符相同时，DFA_pointer = NULL，用下一个非默认跳转字符替换当前 DFA 的非默认跳转字符，直到当前读取的字符与交叠字符不同。

如图 9 所示，该方法不会对 MFA 的处理效率带来影响。在 0 维状态处，每读取一个字符，仍然只需要进行一次状态迁移，不发生在边上，是跳转到交叠字符对应状态的连线的中央状态，而每个中央状态就是暂时的 0 维状态，即说 0 维状态发生了转移，直到满足初始 0 维状态的要求。不存在交叠字符。这个过程类似于 NFA 的字符驱动特性，不同的输入字符造成的中央状态链的长度也不同。

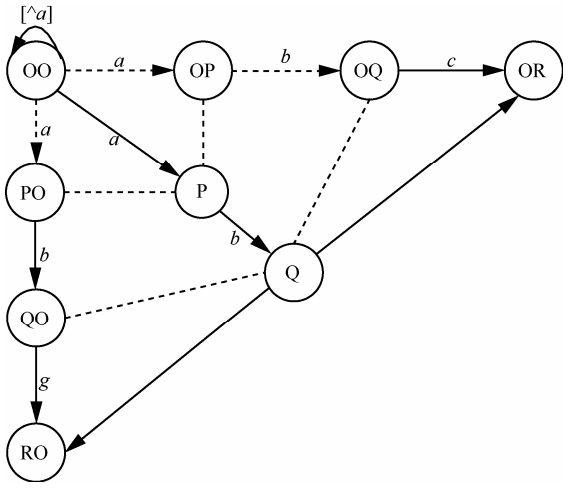


图 9 /.*abc.*def/与/.*abg.*hij/的 2 STD

5.2 克林闭分组数目修正

本节解决“.*”数目不是固定的 2 种情况，设规则类型为/.*S_i.*S_{i1}.*S_{i2}⋯.*S_{in}/。需要的修正的地方如下。

1) 每当一个规则增加一个.*S_{ij}，将在与其他规则正交的方向上增加一个多维立方体。如图 10 所示规则集{/.*ab.*cd.*ef/、/.*gh.jk/、/.*lm.*op/}的三维状态转移图，在和/.*gh.jk/和/.*lm.*op/正交的方向上，增加了一个三维立方体。

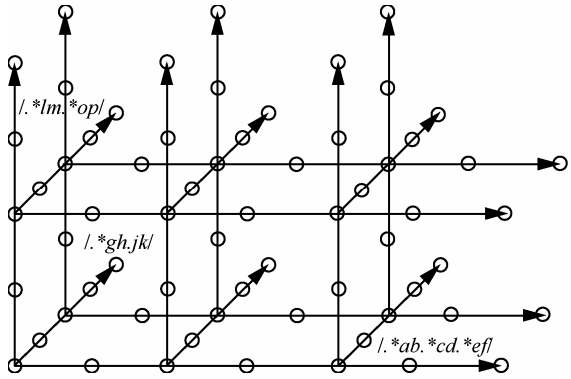


图 10 {/.*ab.*cd.*ef/、/.*gh.jk/、/.*lm.*op/}的 3STD

2) 对 currentNode 与 nextNode 的位数进行扩展，由于一般情况下一个规则的“.*”数目不会超过 10 个，因此在软件实现中均采用 1 byte 分别存储“.*”的坐标。

6 多维有限自动机 MFA

前面非形式化介绍了多维状态转移图和相应算法，本节形式化地提出多维有限自动机 MFA，并给出复杂度分析。

1) 规则集：规则集数目为 m，规则类型为 $/A_1A_2\cdots A_{nj}/$，其中，$nj\in N^+$,对于$j=1,2,\cdots,m$。$A_{ij}=.*S_{ij}$，对于每个 j 有$ij\in\{1,2,\cdots,nj\}$，且S_{ij}为精确字符串。

2) 数学模型：MSTD 为 $MG_m=(V(MG_m),E(MG_m))$其中MG_m满足

$V(MG_m)=\{x_1x_2\cdots x_m : x_j\in\{0,1,\cdots,nj-1\}, j=1,2,\cdots,m\}$，$nj$ 为第 j 个正则表达式含有的“.*”数目。

下面是多维状态转移图 MSTD 的递归定义。

① 基础

1 阶 MSTD：当$nj=2, j=1,2,\cdots,m$时，$V(MG_m)=\{x_1x_2\cdots x_m : x_j\in\{0,1\}, j=1,2,\cdots,m\}$，并且要求若$u=x_1x_2\cdots x_m$, $v=y_1y_2\cdots y_m\in V(MG_m)$, 有

$uv \in E(MG_m) \Leftrightarrow \sum_{i=1}^{m} |x_i - y_i| = 1$，即 1 阶 MSTD 为 m 维立方体。

② 归纳

记 $V(MG_m) = \{x_1 x_2 \cdots x_m : x_j \in \{0,1,\cdots,nj-1\}, j = 1,2,\cdots,m\}$ 为 $\prod_{j=1}^{m}(nj-1)$ 阶 MSTD，若存在 i 满足 $x_i \in \{0,1,\cdots,ni\}$，则需要在 $\prod_{j=1}^{m}(nj-1)$ 阶 MSTD 的基础上，多增加的一个顶点与其他边相连，从而满足度为 m，即在与每条边正交的方向上扩展出一个 m 维立方体，此时为 $\prod_{j\neq i}^{m}(nj-1)ni$ 阶 MSTD。

3) 状态转移

构造虚拟的 MSTD，状态分为 2 类，0 维状态和一维状态。0 维状态位于 0 维面上，由每个 DFA 的“.*”对应的状态构成，一维状态位于一维面上，由每个 DFA 的精确字符对应的状态构成。0 维状态是所有状态之间的中转站。0 维状态只跳转到与之相连的 m 个一维状态，由非默认迁移字符决定。当存在默认迁移字符相同时，构造中央状态，先不确定迁移的一维状态所在边，而是在临时构建的中央状态链上进行状态转移。一维状态的跳转规则分为 2 种情况，当跳到一维状态时不做处理；当跳回 0 维状态时，需要在相同的字符下按照 0 维状态跳转规则进行状态跳转。激活状态的位置由相邻的 2 个 0 维状态的坐标确定。

采用 MSTD 的 MFA 状态跳转与采用 STD 的 DFA 的状态跳转是等价的。MSTD 并没有改变 DFA 的状态迁移起点和终点，唯一的区别是为了保证结构的对称性将顶点作为中转站，因此状态间的跳转可能需要 2 跳。这种 2 跳的情况发生在不同维度之间的状态转移之间，即发生在不同的规则相互连接的过程中。

4) 复杂度分析

分析 MFA 的最差时间复杂度，发生在两条不同边之间一维状态跳转的过程中，需要进行的步骤依次如下。

step1　通过顶点坐标运算获得 DFA 指针。

step2　读取相应的 DFA 的 STT 进行跳转，回到 0 维状态。

step3　读取当前顶点坐标及对应的非默认跳转字符。

step4　当前字符为非默认跳转字符，进行跳转。

step5　更新顶点坐标及对应的非默认跳转字符。

step1~step4 的复杂度都是 $O(1)$，下面说明 step5 的复杂度也是 $O(1)$。由于每个顶点的度均为 m，因此状态之间的跳转仅发生在两条相邻边上，两条边的位置由 3 个顶点决定，由第 2 节定义 1 可知，顶点坐标相互之间最多有 2 个不同，因此对顶点信息更新所需的时间为常数，step5 的复杂度也是 $O(1)$。

综上所述，可知 MFA 的时间复杂度为 $O(1)$，与 DFA 的时间复杂度相同，因此达到了有限自动机的时间复杂度下界。

下面分析空间复杂度，证明有限自动机的空间复杂度下界为 NFA 的空间复杂度 $O(m)$。

假设有 $m>1$ 个正则表达式联合编译，第 i 个正则表达式($i = 1,2,\cdots,m$)有 N_i 个字符，那么将 m 个正则表达式编译成一个 NFA 对应的状态数为

$$S = \sum_{i=1}^{m} N_i \tag{6}$$

NFA 通过子集构造法生成 DFA，因此 NFA 的状态数目对应编译为 DFA 的状态数目的理论下界。m 个 /.* S_1.* S_2 ….* S_{nj} / 类型的规则合并编译，每个子串 S_{ij} 对应的长度为 L_{ij}，则合并编译得到的 DFA 的状态空间理论下界为

$$S_{\text{lmt}} = \sum_{j=1}^{m}\left(\sum_{ij=1}^{nj}\left(L_{ij}\right) + nj\right) \tag{7}$$

对于一般的规则，规则的子串 S_{ij} 长度 L_{ij} 及“.*”数目 nj 与规则数目相比都认为是常数，因此，m 个 /.* S_1.* S_2 ….* S_{nj} / 类型的规则合并编译生成的 DFA 状态数目的理论下界，即自动机状态空间的理论下界

$$S_{\text{lmt}} = O(m) \tag{8}$$

由于每个状态处的跳转信息均是常数级别，式(8)也是有穷自动机存储空间的复杂度下界。

通过第 3.2 节和第 5 节的介绍，MSTD 保存的状态仅限于坐标轴上的状态，以及 2 个动态状态，因此状态空间复杂度为 $O(m)$；此外，MFA 仅需要存储 m 个 DFA 信息，动态存储 $2m$ byte 的顶点信息以及 1 个 m bit 的 DFA 指针，常数级别的中央状态跳转信息。因此 MFA 的存储复杂度为 $O(m)$。

MFA 与经典的自动机算法的复杂度对比如表 5 所示，MFA 的时间复杂度与 DFA 相当，空间复杂度与 NFA 相当。MFA 所付出的代价是常数级别的

访存次数、计算量以及存储空间。

表 5 有限自动机算法复杂度对比

算法	时间复杂度	状态/存储空间复杂度
NFA	$O(m)$	$O(m)$
DFA	$O(1)$	$\geqslant O(m)$
MFA	$O(1)/O(c)$	$O(m)/O(cm)$

7 仿真实验与分析

仿真实验分为两部分，通过对比 MFA 与 DFA、NFA 的存储状况，验证 MFA 的存储复杂度达到了有限自动机的理论下界；从 snort、L7-filter、dotstar0.9 中选取部分规则集，与目前主流 DFA 改进算法 State merging、D^2FA、Hybrid-FA、mDFA、XFA 等的预处理时间、存储空间以及匹配速率进行对比。

实验采用的计算机环境为 Virtual Machine 8.0，虚拟机配置操作系统为 Ubuntu 10.04，内存 8 GB、双 Intel Pentium CPU，主频 2.53 GHz，硬盘 20 GB；编程环境为 gcc4.2.4，主程序采用 RegEx Processor，内含 DFA、NFA、State merging、D^2FA、Hybrid-FA 等算法，采用 C++补充实现了 mDFA、XFA 以及本文的 MFA 算法。

仿真 1 MFA 状态空间及存储复杂度

MFA 与传统 DFA 及 NFA 进行仿真，由于 RegEx Processor 含有 DFA 规则自动分组算法以及默认的 STT 压缩算法（State merging+D^2FA），因此对源码进行了改编并重新编译，使其可以产生唯一并且不含任何压缩的算法。规则集类似于表 2，在不同的规则数量下，分别生成 DFA、NFA 以及 MFA，从而得到三者的状态数以及存储空间。如表 6 所示，存储空间主要由状态个数、迁移边条数、辅助变量等构成。由 MFA 的存储统计一列可以看出，MFA 所需要付出的辅助变量存储占总存储的比例非常低。

由图 11(a)和图 11(b)所示，在对数坐标下 DFA 的状态空间和存储空间的增长随状态数目呈线性增加，即两者随状态数目指数增长。由图 11(c)和图 11(d)看出，在普通坐标下，MFA 和 DFA 的状态空间和存储空间随状态数目线性增长，两者的复杂度均为 $O(m)$，达到了有穷自动机的空间复杂度下界，验证了表 5 的结论。

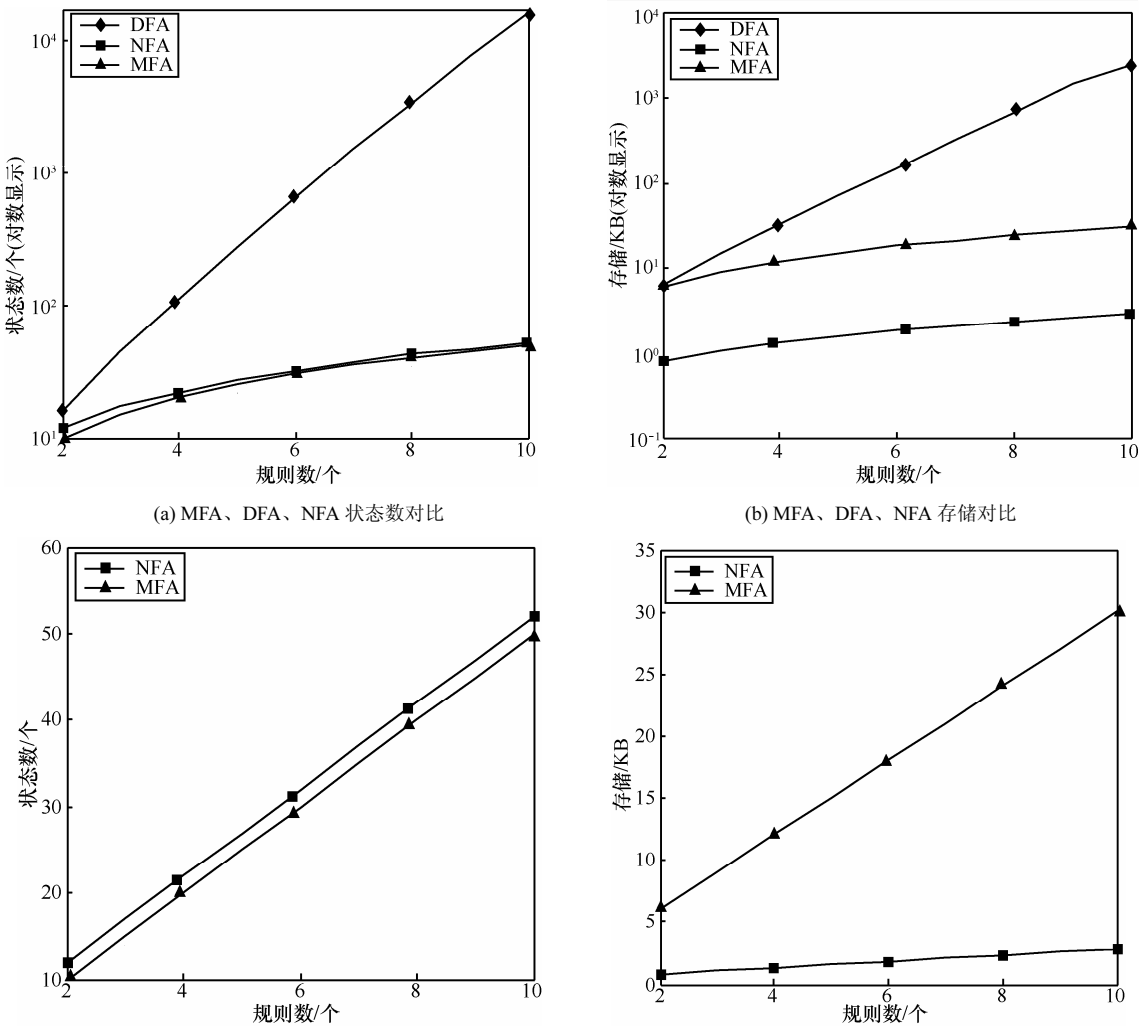
由图 11(a)和图 11(b)看出，在这类特殊规则集下 MFA 对 DFA 的空间压缩率相当大，并且随着规则数的增加，压缩率不断提高。由表 6 看出，当规则数达到 7 时，存储压缩率就已经达到了 90%以上，随着规则数目的增加，压缩率无限接近 100%。成功解决 DFA 产生的状态爆炸问题。

仿真 2 MFA 与其他 DFA 改进算法的对比

测试规则集从 snort、l7-filter、dotstar0.9 中选取了部分满足 MFA 的规则共 189 条，测试数据由 RegEx Processor 的 traffic trace generator 生成。与图 1 所示的目前 DFA 改进算法各方向的经典算法进行比较，分别是基于 STT 冗余压缩的 state merging 和 D^2FA（以下简称 STT），基于辅助变量的 XFA，基于规则分组方向的 mDFA，基于调节确定度方向的 Hybrid-FA。尝试构建全部规则生成一个联合 DFA，但是由于规则集过大，20 GB 的存储空间无法满足条件。为了缩短仿真时间，经过不断地测试最终选取其中 80 条规则进行实验，得到的实验数据如表 7 所示。

表 6 MFA 和 NFA、DFA 的状态数和存储仿真数据

规则数目	NFA		DFA		MFA		DFA vs MFA	
	状态数	存储/KB	状态数	存储/KB	状态数(动态状态数)	存储(辅助变量)/KB	状态压缩率/%	存储压缩率/%
2	12	0.779	16	6.455	10(2)	6.077(0.014)	37.50	5.86
3	17	1.040	44	14.667	15(2)	9.039(0.021)	65.91	38.37
4	22	1.301	112	31.997	20(2)	12.047(0.028)	82.14	62.35
5	27	1.562	272	69.841	25(2)	15.032(0.035)	90.81	78.48
6	32	1.823	640	149.419	30(2)	18.030(0.042)	95.31	87.93
7	37	2.084	1 472	324.355	35(2)	21.063(0.049)	97.62	93.51
8	42	2.345	3 328	684.086	40(2)	24.052(0.056)	98.80	96.48
9	47	2.606	7 424	1 480.794	45(2)	27.029(0.063)	99.39	98.17
10	52	2.867	15 872	2 560.307	50(2)	30.057(0.070)	99.68	99.05



(a) MFA、DFA、NFA 状态数对比　　(b) MFA、DFA、NFA 存储对比

(c) MFA、NFA 状态数对比　　(d) MFA、NFA 存储对比

图 11　理想规则集下 MFA 和 DFA、NFA 状态与存储对比曲线

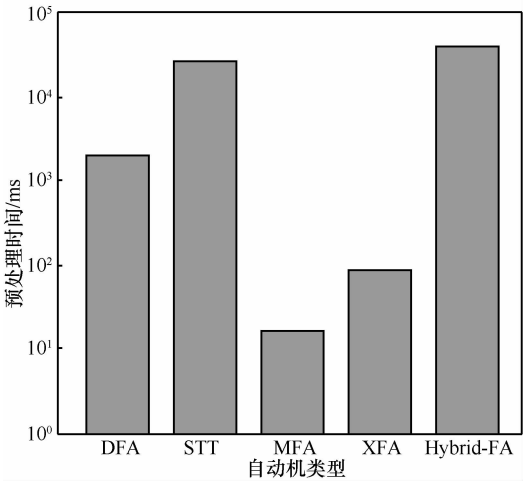
表 7　　**DFA 改进算法的预处理时间、存储空间、匹配时间实验数据**

自动机类型	预处理时间	存储	自动机数目	匹配时间/(s·GB^{-1})	是否支持全部规则集
DFA	35 min	16.1 GB	n/a	13.8	Y
STT(state merging+D2FA)	7.5 h	789 MB	n/a	220.8	Y
MFA	16 s	10.4 MB	n/a	57.9	N
XFA	90 s	4.64 MB	n/a	80.1	N
Hybrid-FA	11.5 h	7.35 GB	n/a	19.4	Y
mDFA	n/a	47.1 MB	6	231	Y
	n/a	26.4 MB	14	541	
	n/a	20.4 MB	37	1 429	
	n/a	9.3 MB	56	2 163	
	n/a	7.0 MB	72	2 781	

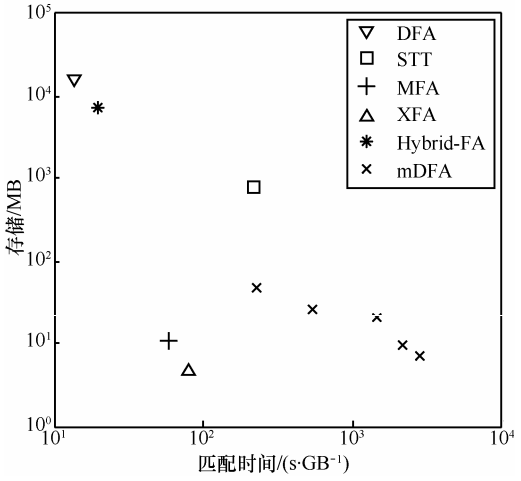
在预处理时间方面，如图 12(a)所示，MFA 与 XFA 预处理时间最短，在同一个数量级上，与 DFA、STT 和 Hybrid-FA 相比，预处理时间降低了 2~3 个数量级，大大提高了自动机构造效率和实时更新能力。在字符匹配阶段，如表 7 所示，存储空间最低的为 XFA(4.64 MB)，其次是 MFA(10.4 MB)，与 DFA 及

STT、mDFA、Hybrid-FA 相比，降低了 2~3 个数量级；匹配时间最少的是 Hybrid-FA(19.4 s)，其次是 MFA(57.9 s)，与 DFA 的匹配时间相当，与 STT、mDFA 相比，降低了 1~2 个数量级。由图 12(b)所示，DFA、STT 和 Hybrid-FA 匹配效率较高，但是存储空间过大；而 mDFA 存储消耗小，但是匹配效率低；MFA 与 XFA 的综合性能最佳，达到了存储空间和匹配效率最好的折中效果。

由图 12(b)还可以发现，Hybrid-FA 的综合性能很差，这与一般情况相符，因此又进行了补充实验。从测试规则集表 2 中随机选取其中的 6 个规则，即满足 MFA 条件的规则，又从 snort24 规则集中随机选出 6 个规则，构造对应 DFA 及 Hybrid-FA，实验结果如表 8 所示，对于 MFA 规则，Hybrid-FA 所有的 Tail-NFA 均没有被激活，因此 Hybrid-FA 对 MFA 很好解决的规则几乎没有作用。



(a) DFA 改进算法预处理时间对比



(b) DFA 改进算法存储 vs 匹配时间

图 12 DFA 与其他改进算法对比

表 8 HFA 对 MFA 规则没有效果

规则集	DFA	Hybrid-FA			
	状态数	head size	tail size	tails	border size
MFA 规则	15 634	11 704	0	0	0
snort24	23 346	12 314	42	1	108

实际上 MFA 和 XFA 有一个相同的缺点，覆盖规则范围较窄，而其他算法适用于所有的正则表达式规则。这就提供了一个新思路，为不同类型的规则设计不同的有限自动机。设计不同的规则模板，分别进行处理，从而达到更好的压缩效果。这种思路类似于 ASIC 与 CPU 的关系，CPU 通用性强，但是针对于特定应用 ASIC 可以表现出更好的性能。本文的规则模板主要为精确字符串和克林闭分组“.*”的线性组合，MFA 可以取得非常好的效果，为了扩展规则覆盖率必须设计新的规则模板并在此基础上扩展 MFA。扩展方法以第 6 节所描述的规则类型 $/A_1A_2\cdots A_{nj}/$，$A_{ij}=.*S_{ij}$ 为基础，扩展 A_{ij} 的类型，从而修正模型及对应算法。设计原则为将正则语言的基本操作与 A_{ij} 联系起来，连接操作“”、或操作“|”对应到 S_{ij}，闭分组操作“*”及其变形在“.*”的基础上进行改进，具体如下。

1) S_{ij} 不仅有连接操作“”，而且还包含“|”操作，那么需要在 S_{ij} 对应的边上增加相应的状态以及迁移边。

2) 克林闭分组“.*”变为“.[*m-n*]”、“[^*n*]*”、“[^*n**s*][*m-n*]”等在规则集中同样很常见的子串，那么对应的模型和算法均需要改进，在交点处增加计数器记录访问次数从而匹配“[*m-n*]”，此外需要增加针对“*n*”、“*s*”等字符的状态迁移边，从而对“[^*n*]”、“[^*n**s*]”等进行特殊处理。

8 结束语

基于 DFA 的正则表达式匹配算法被广泛应用于网络信息安全 IDS 中，然而当多条规则联合编译时存在状态爆炸问题。为了解决该问题，研究者提出了诸多改进算法，然而缺乏数学模型方面的研究，并且没有达到有限自动机复杂度的下界。针对状态爆炸最典型也是最严重的一类问题，本文从信息论的角度重新分析，并提出多维状态转移图 MSTD，在此数学模型的基础上进行了冗余状态压缩，并给出多维状态转移算法 MSTA，两者构成了一种新型的有限自动机

MFA。理论证明，MFA 同时达到 NFA 的空间复杂度下界 $O(m)$和 DFA 的时间复杂度下界 $O(1)$，MFA 付出的代价是常数级别的访存次数、计算量以及存储空间，表明 MFA 是适合处理“.*”相互作用产生状态爆炸问题的有限自动机。实验表明，MFA 对 DFA 的空间压缩率几乎可以达到 100%。与经典的 DFA 改进算法相比，MFA 的预处理时间最短，与 STT 冗余压缩算法、Hybrid-FA 相比降低了 2~3 个数量级；存储空间与 DFA 及 STT 冗余压缩算法、mDFA、Hybrid-FA 相比，降低了 2~3 个数量级；匹配时间与 DFA 相当，与 STT 冗余压缩算法、mDFA 相比，降低了 1~2 个数量级。MFA 的综合性能在所有的 DFA 改进算法中是最好的。

MFA 缺点是规则覆盖率低，因此需要设计更多的规则模板，针对不同的模板扩展 MFA。此外 DFA 改进算法也有各自的使用范围，针对不同的规则设计并选择合适的算法。

参考文献：

[1] HOPCROFT J E, ULLMANJ D. Introduction to Automata Theory, Languages and Computation 2nd Edition[M]. US: Addison Wesley, 2001.

[2] YU F, CHEN Z F, DIAO Y L, *et al*. Fast and memory-efficient regular expression matching for deep packet inspection[A]. Proceedings of the IEEE/ACM Symposium on Architectures for Networking and Communications Systems[C]. San Jose, Canada, 2006. 93-102.

[3] 徐乾, 鄂跃鹏, 葛敬国等. 深度包检测中一种高效的正则表达式压缩算法[J]. 软件学报, 2009, 20(8): 2214-2226.
XU Q, E Y P, GE J G, *et al*. Efficient regular expression compression algorithm for deep packet inspection[J]. Journal of Software, 2009, 20(8): 2214-2226.

[4] BECCHI M, CROWLEY P. A hybrid finite automaton for practical deep packet inspection [A].Proceedings of the 2007 ACM CoNEXT conference[C]. New York, USA, 2007. 1.

[5] 张树壮, 罗浩, 方滨兴等. 一种面向网络安全检测的高性能正则表达式匹配算法[J]. 计算机学报, 2010,33(10): 1976-1986.
ZHANG S Z, LUO H, FANG B X, *et al*. An efficient regular expression matching algorithm for network security inspection[J]. Chinese Journal of Computers, 2010, 33(10): 1976-1986.

[6] 乔登科, 王卿, 柳厅文等. 基于状态分组的高效 i-DFA 构造技术[J]. 通信学报, 2013,34(8): 102-109.
QIAO D K, WANG Q, LIU T W, *et al*. Efficient i-DFA construction algorithm based on state grouping[J]. Journal on Communications, 2013, 34(8): 102-109.

[7] 贺炜, 郭云飞, 扈红超. 基于状态约束的大规模正则表达式匹配算法[J]. 通信学报, 2013, 34(10): 183-190.
HE W, GUO Y F, HU H C. States constrain-based algorithm for large scale regular expression matching[J]. Journal on Communications, 2013, 34(10): 183-190.

[8] YANG Y H E, PRASANNA V K. Space-time tradeoff in regular expression matching with semi-deterministic finite automata[A]. INFOCOM, 2011 Proceedings IEEE[C]. Shanghai, China, 2011. 1853-1861.

[9] KUMAR S, CHANDRASEKARAN B, TURNER J, *et al*. Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia[A]. Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems[C]. Orlando, USA, 2007. 155-164.

[10] SMITH R, ESTAN C, JHA S, *et al*. Deflating the big bang: fast and scalable deep packet inspection with extended finite automata [A].SIGCOMM '08 Proceedings of the ACM SIGCOMM 2008 conference on Data communication [C]. Seattle, USA, 2008. 207-218.

[11] 张大方, 张洁坤, 黄昆. 一种基于智能有限自动机的正则表达式匹配算法[J].电子学报, 2012,40(8):1617-1623.
ZHANG D F, ZHANG J K, HUANG K. A regular expression matching algorithm with smart finite automaton [J].Acta Electronica Sinica, 2012, 40(8):1617-1623.

[12] KUMAR S, DHARMAPURIKAR S, YU F, *et al*. Algorithms to accelerate multiple regular expressions matching for deep packet inspection[A]. Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications[C]. Pisa, Italy, 2006. 339-350.

[13] BECCHI M, CADAMBI S. Memory-efficient regular expression search using state merging[A]. INFOCOM 2007, the 26th IEEE International Conference on Computer Communications[C]. Anchorage, USA, 2007. 1064-1072.

[14] FICARA D, GIORDANO S, PROCISSI G, *et al*. An improved DFA for fast regular expression matching[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(5): 29-40.

[15] FICARA D, DI PIETRO A, GIORDANO S, *et al*. Differential encoding of DFA for fast regular expression matching[J]. IEEE/ACM Transactions on Networking, 2011, 19(3): 683-694.

[16] QI Y, WANG K, FONG J, *et al*. Feacan: front-end acceleration for content-aware network processing [A]. INFOCOM, 2011 Proceedings IEEE[C]. Shanghai, China, 2011.2114-2122.

[17] LIU T, YANG Y, LIU Y, *et al*. An efficient regular expressions compression algorithm from a new perspective[A]. INFOCOM2011 Proceedings IEEE[C]. Shanghai, China, 2011. 2129-2137.

[18] 张树壮, 罗浩, 方滨兴. 面向网络安全的正则表达式匹配技术[J]. 软件学报, 2011, 22(8): 1838-1853.
ZHANG S Z, LUO H, FANG B X. Regular expressions matching for network security [J]. Journal of Software, 2011, 22(8): 1838-1853.

[19] MCNAUGHTON R, YAMADA H. Regular expressions and state graphs for automata [J]. IRE Transactions on Electronic Computers, 1960, (1): 39-47.

[20] Snortrules-snapshot-2956[EB/OL].http://www.snort.org/snort-rules/?, 2014.

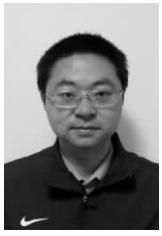
[21] L7-protocols-2009-05-28[EB/OL].http://l7-filter.clearfoundation.com/downloads/start, 2014.

[22] Regular expression processor[EB/OL]. http://regex.wustl.edu/index.php/Main_Page, 2014.

作者简介：

宫阳阳（1988-），男，河南濮阳人，国家数字交换系统工程技术研究中心硕士生，主要研究方向为宽带信息网络及系统级芯片设计。

刘勤让（1975-），男，河南睢县人，博士，国家数字交换系统工程技术研究中心研究员、硕士生导师，主要研究方向为宽带信息网络及系统级芯片设计。



杨镇西（1971-），男，山东郓城人，博士，国家数字交换系统工程技术研究中心副研究员，主要研究方向为系统级芯片设计。

邵翔宇（1992-），男，河南清丰人，国家数字交换系统工程技术研究中心硕士生，主要研究方向为宽带信息网络及系统级芯片设计。

邢池强（1989-），男，河南邢台人，国家数字交换系统工程技术研究中心硕士生，主要研究方向为宽带信息网络。

焦慧娟（1989-），女，山东菏泽人，中国石油大学硕士生，主要研究方向为有限元分析。

彭志彬（1988-），男，河南濮阳人，国家数字交换系统工程技术研究中心硕士生，主要研究方向为模式识别及数据挖掘。