(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 103733590 B
(45)授权公告日 2017.05.03

(21)申请号 201280038799.2
(22)申请日 2012.06.20
(65)同一申请的已公布的文献号
申请公布号 CN 103733590 A
(43)申请公布日 2014.04.16
(30)优先权数据
13/168,450 2011.06.24 US
(85)PCT国际申请进入国家阶段日
2014.02.08
(86)PCT国际申请的申请数据
PCT/US2012/043307 2012.06.20
(87)PCT国际申请的公布数据
WO2012/177736 EN 2012.12.27
(73)专利权人 凯为公司
地址 美国加利福尼亚州
(72)发明人 R·戈亚尔 S·L·比拉
K·A·布里斯
(74)专利代理机构 北京市金杜律师事务所 11256
代理人 王茂华
(51)Int.Cl.
H04L 29/06(2006.01)
(56)对比文件
US 2008109431 A1,2008.05.08,
US 2004172234 A1,2004.09.02,
张树壮,等.一种面向网络安全检测的高性能正则表达式匹配算法.《计算机学报》.2010,第33卷(第10期),第1976-1986页.
审查员 李红玲

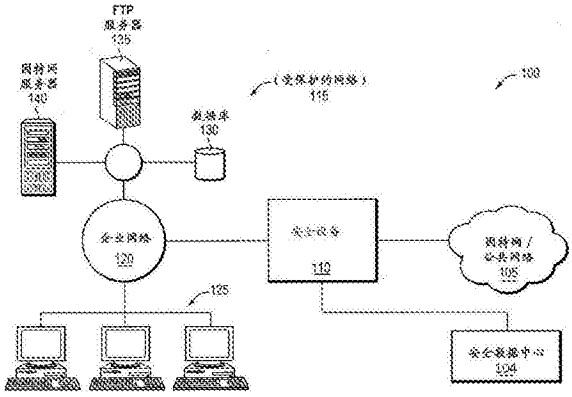
权利要求书5页 说明书18页 附图23页

(54)发明名称

用于正则表达式的编译器

(57)摘要

一种方法和相应的装置涉及将用于给定的图样集合的非确定性有限自动机(NFA)图转换成具有多个状态的确定性有限自动机(DFA)图形。DFA状态中的每一个被映射成NFA图形的一个或多个状态。计算映射到每个DFA状态的NFA的图形的一个或多个状态的哈希值。对于给定图样,DFA状态表将多个DFA状态中的每一个与NFA图形的一个或多个状态的哈希值相关。



CN 103733590 B

1.一种耦合到网络的安全设备的处理器中的方法,包括:

将用于给定的图样集合的非确定性有限自动机NFA图形转换成具有多个确定性有限自动机DFA状态的DFA图形,转换所述NFA图形包括:

将所述多个DFA状态中的每一个状态映射到所述NFA图形的一个或多个状态的对应集合;

计算映射到每个DFA状态的所述NFA图形的所述一个或多个状态的每个对应集合的相应哈希值;以及

在DFA状态表中存储所计算的每个相应哈希值,所述DFA状态表将所述多个DFA状态中的每一个状态与用于所述给定的图样的所述NFA图形的所述一个或多个状态的所述对应集合的所计算的所述相应哈希值相关。

2.根据权利要求1所述的方法,其中,所计算的每个相应哈希值是加密/完美哈希值。

3.根据权利要求1所述的方法,其中,映射包括:

确定在所述多个DFA状态内是否存在未标记DFA状态;

选择所述未标记DFA状态中的一个;

标记所述未标记DFA状态;以及

针对由所述NFA图形所识别的字母的字符,确定映射到标记的DFA状态的所述NFA图形的所述一个或多个状态到其他NFA状态的转换。

4.根据权利要求3所述的方法,其中,确定转换包括:将所述其他NFA状态和所述其他NFA状态的埃普西隆终止映射到可能的新的未标记DFA状态。

5.根据权利要求4所述的方法,其中,将可能的新的未标记DFA状态映射到所述其他NFA状态的埃普西隆终止包括:从埃普西隆高速缓存获得所述其他NFA状态的所述埃普西隆终止,并且如果在所述埃普西隆高速缓存内不存在所述其他NFA状态的所述埃普西隆终止,则计算所述其他NFA状态的所述埃普西隆终止。

6.根据权利要求5所述的方法,进一步包括:如果映射到所述可能的新的未标记DFA状态的所述其他NFA状态和所述其他NFA状态的所述埃普西隆终止没有被映射到所述多个DFA状态的现有DFA状态,则将所述可能的新的未标记DFA状态添加到所述多个DFA状态,并且将所述可能的新的未标记DFA状态存储在所述DFA状态表中。

7.根据权利要求6所述的方法,其中,添加所述可能的新的未标记DFA状态包括:

将映射到所述可能的新的未标记DFA状态的所述其他NFA状态的所述埃普西隆终止的哈希值与映射到所述多个DFA状态中的每一个状态的NFA状态的哈希值作比较。

8.根据权利要求6所述的方法,其中,将所述可能的新的未标记DFA状态添加到所述多个DFA状态进一步包括:确定映射到所述可能的新的未标记状态的所述其他NFA状态的所述埃普西隆终止是否属于最终接受NFA状态,并且如果是,则将可能的未标记状态添加为最终接受DFA状态。

9.根据权利要求6所述的方法,进一步包括:将所述可能的新的未标记状态添加为从标记的DFA状态针对由所述NFA图形所识别的所述字母的所述字符的转换。

10.根据权利要求4所述的方法,进一步包括:用计算的所述其他NFA状态和所述其他NFA状态的埃普西隆终止的哈希值到所述可能的新的未标记DFA状态的映射来替换所述其他NFA状态和所述其他NFA状态的所述埃普西隆终止到所述可能的新的未标记DFA状态的映

射。

11.根据权利要求10所述的方法，其中，替换包括删除所述其他NFA状态和所述其他NFA状态的所述埃普西隆终止。

12.根据权利要求3所述的方法，进一步包括：从所述DFA状态表中删除所述NFA图形的一个或多个状态的所述转换。

13.根据权利要求1所述的方法，其中，映射包括：

确定DFA开始状态；以及

将DFA开始状态作为未标记状态添加到包括所述多个DFA状态的数据结构。

14.根据权利要求13所述的方法，其中，确定DFA开始状态包括：

确定NFA开始状态的埃普西隆终止；

计算所述NFA开始状态的所述埃普西隆终止的哈希值；以及

将所述DFA开始状态映射到所述NFA开始状态的所述埃普西隆终止的所述哈希值。

15.根据权利要求13所述的方法，其中，映射包括：

确定在所述多个DFA状态内是否存在未标记DFA状态；

选择所述未标记DFA状态中的一个未标记DFA状态；

标记所述未标记DFA状态；以及

针对由所述NFA图形所识别的字母的字符，确定映射到标记的DFA状态的所述NFA图形的一个或多个状态到其他NFA状态的转换。

16.根据权利要求15所述的方法，其中，确定转换包括：

确定从所述标记的DFA状态针对由所述NFA图形所识别的所述字母的所述字符的转换DFA状态。

17.根据权利要求16所述的方法，其中，确定所述转换DFA状态包括：

计算所述NFA图形的所述一个或多个状态的所述转换的哈希值；以及

将计算的哈希值与埃普西隆终止EC高速缓存表中的哈希值条目作比较，所述EC高速缓存表将哈希值映射到DFA状态。

18.根据权利要求17所述的方法，进一步包括：

确定计算的哈希值是否匹配所述EC高速缓存表中的所述哈希值条目中的一个哈希值条目；以及

如果匹配存在，则将映射到匹配哈希值的所述DFA状态设置为针对由所述NFA图形所识别的所述字母的所述字符的转换DFA状态。

19.根据权利要求18所述的方法，进一步包括：

如果不存在匹配，则计算所述NFA图形的所述一个或多个状态的所述转换的埃普西隆终止；

计算所述埃普西隆终止的哈希值；

将新的条目添加到所述EC高速缓存表中，所述新的条目将新的DFA状态映射到所述埃普西隆终止的所述哈希值；以及

将所述新的DFA状态设置为针对由所述NFA图形所识别的所述字母的所述字符的所述转换DFA状态。

20.根据权利要求19所述的方法，进一步包括：

从所述DFA状态表中删除与所述标记的DFA状态相对应的所述NFA图形的所述一个或多个状态的所述转换。

21.一种耦合到网络的安全设备，所述安全设备包括：

处理器，所述处理器被配置成将用于给定的图样集合的非确定性有限自动机NFA图形转换成具有多个确定性有限自动机DFA状态的DFA图形；以及

编译器，所述编译器被配置成：

将所述多个DFA状态中的每一个状态映射到所述NFA图形的一个或多个状态的对应集合；

计算映射到每个DFA状态的所述NFA图形的所述一个或多个状态的每个对应集合的哈希值；以及

在数据存储库中的DFA状态表存储所计算的每个相应哈希值，所述DFA状态表将所述多个DFA状态中的每一个与用于给定图样的所述NFA图形的所述一个或多个状态的所述对应集合的所计算的所述相应哈希值相关。

22.根据权利要求21所述的安全设备，其中所计算的每个相应哈希值是加密/完美哈希值。

23.根据权利要求21所述的安全设备，其中，所述编译器进一步被配置成：

确定在所述多个DFA状态内是否存在未标记DFA状态；

选择所述未标记DFA状态中的一个未标记DFA状态；

标记所述未标记DFA状态；以及

确定映射到标记DFA状态的所述NFA图形的所述一个或多个状态到由所述NFA图形所识别的字母的字符的其他NFA状态的转换。

24.根据权利要求23所述的安全设备，其中，所述编译器进一步被配置成：通过将所述其他NFA状态和所述其他NFA状态的埃普西隆终止映射到可能的新的未标记DFA状态来确定转换。

25.根据权利要求24所述的安全设备，其中，所述编译器被配置成，通过下述操作将所述可能的新的未标记DFA状态映射到所述其他NFA状态的埃普西隆终止：从埃普西隆高速缓存获得所述其他NFA状态的所述埃普西隆终止，并且如果在所述埃普西隆高速缓存内不存在所述其他NFA状态的所述埃普西隆终止，则所述编译器被配置成计算所述其他NFA状态的埃普西隆终止。

26.根据权利要求25所述的安全设备，其中，所述编译器进一步被配置成，如果映射到所述可能的新的未标记DFA状态的所述其他NFA状态和所述其他NFA状态的埃普西隆终止没有映射到所述多个DFA状态的现有DFA状态，则将所述可能的新的未标记DFA状态添加到所述多个DFA状态，并且将所述可能的新的未标记DFA状态存储在所述DFA状态表中。

27.根据权利要求26所述的安全设备，其中，所述编译器被配置成，通过将映射到所述可能的新的未标记DFA状态的所述其他NFA状态的所述埃普西隆终止的哈希值与映射到所述多个DFA状态中的每一个DFA状态的NFA状态的哈希值进行比较来添加所述可能的新的未标记DFA状态。

28.根据权利要求26所述的安全设备，其中，所述编译器被配置成，通过下述操作将所述可能的未标记DFA状态添加到所述多个DFA状态：确定映射到可能的新的未标记状态的所

述其他NFA状态的所述埃普西隆终止是否属于最终接受NFA状态，并且如果是，则所述编译器被进一步配置成将可能的未标记状态添加为最终接受DFA状态。

29.根据权利要求26所述的安全设备，其中，所述编译器被进一步配置成，将可能的新的未标记状态添加为从标记的DFA状态针对由所述NFA图形所识别的所述字母的所述字符的转换。

30.根据权利要求24所述的安全设备，其中，所述编译器被进一步配置成，用计算的所述其他NFA状态和所述其他NFA状态的埃普西隆终止的哈希值到所述可能的新的未标记的DFA状态的映射来替换所述其他NFA状态和所述其他NFA状态的埃普西隆终止到所述可能的新的未标记的DFA状态的映射。

31.根据权利要求30所述的安全设备，其中，所述编译器被配置成删除所述其他NFA状态和所述其他NFA状态的所述埃普西隆终止。

32.根据权利要求23所述的安全设备，其中，所述编译器被进一步配置成，从所述DFA状态表中删除所述NFA图形的所述一个或多个状态的所述转换。

33.根据权利要求21所述的安全设备，其中，所述编译器被配置成，通过确定DFA开始状态并且将所述DFA开始状态作为未标记状态添加到包括所述多个DFA状态的数据结构来将所述多个DFA状态中的每一个DFA状态映射到所述NFA图形中的一个或多个状态。

34.根据权利要求33所述的安全设备，其中，所述编译器进一步被配置成：

确定NFA开始状态的埃普西隆终止；

计算所述NFA开始状态的所述埃普西隆终止的哈希值；以及

将所述DFA开始状态映射到所述NFA开始状态的所述埃普西隆终止的所述哈希值。

35.根据权利要求33所述的安全设备，其中，所述编译器进一步被配置成：

确定在所述多个DFA状态内是否存在未标记DFA状态；

选择所述未标记DFA状态中的一个未标记DFA状态；

标记所述未标记DFA状态；以及

确定映射到所标记的DFA状态的所述NFA图形的所述一个或多个状态到针对由所述NFA图形所识别的字母的字符的其他NFA状态的转换。

36.根据权利要求35所述的安全设备，其中，所述编译器被配置成通过确定从所述标记的DFA状态针对由所述NFA图形所识别的所述字母的所述字符的转换DFA状态来确定转换。

37.根据权利要求36所述的安全设备，其中，所述编译器进一步被配置成：

计算所述NFA图形的所述一个或多个状态的所述转换的哈希值；以及

将所计算的哈希值与埃普西隆终止EC高速缓存表中的哈希值条目作比较，所述EC高速缓存表将哈希值映射到DFA状态。

38.根据权利要求37所述的安全设备，其中，所述编译器进一步被配置成：

确定计算的哈希值是否匹配所述EC高速缓存表中的所述哈希值条目中的一个；以及

如果匹配存在，则将映射到匹配的哈希值的所述DFA状态设置为针对由所述NFA图形所识别的所述字母的所述字符的转换DFA状态。

39.根据权利要求38所述的安全设备，其中，所述编译器进一步被配置成：

如果不存在匹配，则计算所述NFA图形的所述一个或多个状态的所述转换的埃普西隆终止；

计算所述埃普西隆终止的哈希值；

将新的条目添加到所述EC高速缓存表中，所述新的条目将新的DFA状态映射到所述埃普西隆终止的所述哈希值；以及

将所述新的DFA状态设置为针对由所述NFA图形所识别的所述字母的所述字符的所述转换DFA状态。

40.根据权利要求39所述的安全设备，其中，所述编译器进一步被配置成：

从DFA状态表中删除与所述标记的DFA状态相对应的所述NFA图形的所述一个或多个状态的所述转换。

用于正则表达式的编译器

[0001] 相关申请

[0002] 本申请是2011年6月24日提交的美国申请No.13/168,450的继续并且要求其优先权。上述申请的全部教导通过引用合并于此。

技术领域

[0003] 本公开的非限制性示例一般地涉及通信网络的技术领域，并且更具体地涉及用于内容搜索的方法和装置。

背景技术

[0004] 开放系统互连(OSI)参考模型定义了用于通过传输介质进行通信的7个网络协议层(L1-L7)。上层(L4-L7)表示端对端通信并且下层(L1-L3)表示本地通信。

[0005] 联网应用感知系统需要处理、滤波和切换L3到L7网络协议层的范围，例如，L7网络协议层诸如超文本传输协议(HTTP)和简单邮件传输协议(SMTP)，以及L4网络协议层诸如传输控制协议(TCP)。除了处理网络协议层，联网应用感知系统需要通过L4-L7网络协议层来同时确保这些协议对基于访问和内容的安全性，包括防火墙、虚拟专用网(VPN)、安全套接字层(SSL)、入侵检测系统(TDS)、因特网协议安全(IPSec)、线速度的防病毒(AV)和防垃圾邮件功能。

[0006] 网络处理器可用于高吞吐量L2和L3网络协议处理，即执行分组处理以线速度转发分组。通常，通用处理器用于处理需要更多智能处理的L4-L7网络协议。虽然通用处理器可以执行计算密集型任务，但是没有足够用于处理数据使得其能够被以线速转发的性能。

[0007] 内容感知联网需要以“线速度”的对分组内容的检查。可以对内容进行分析，以确定是否存在安全漏洞或入侵。应用大量正则表示式形式的图样和规则以确保所有的安全漏洞或入侵被检测到。正则表示式是用于描述字符串的图样的紧凑型方法。由正则表示式所匹配的最简单图样是单个字符或字符串，例如，/c/或/cat/。正则表示式还包括具有特殊含义的运算符和元字符。

[0008] 通过使用元字符，正则表示式可以用于更复杂的搜索，诸如“abc.*xyz”。也就是说，在不限制“abc”和“xyz”之间的字符数目的情况下，找到字符串“abc”，之后是字符串“xyz”。另一示例是正则表示式“abc..abc.*xyz;”，也就是找到字符串“abc”，后面两个字符，然后是“abc”并且在不限制个数的字符后由串“xyz”跟随。

[0009] 入侵检测系统(IDS)应用检查所有流过网络的独立分组的内容，并且识别可能指示尝试闯入或威胁系统的可疑图样。可疑图样的一个示例可以是分组中的特定文本串，该特定文本串在100个字符以后跟随另一特定文本串。

[0010] 通常使用搜索算法来执行内容搜索以处理正则表示式，搜索算法诸如确定性有限自动机(DFA)或者非确定性有限自动机(NFA)。

发明内容

[0011] 一种方法和相应的装置，涉及将用于给定的图样集合的非确定性有限自动机(NFA)图形转换成具有多个状态的确定性有限自动机(DFA)图形。DFA状态中的每一个被映射成NFA图形的一个或多个状态。计算映射成每个DFA状态的NFA图形的一个或多个状态的哈希值。对于给定图样，DFA状态表将多个DFA状态中的每一个与NFA图形的一个或多个状态的哈希值相关。

[0012] 可以确定与由NFA图形和DFA图形所识别的字母相关联的给定图样的有效字符。基于该确定，该方法压缩NFA图形和DFA图形以识别图样，该图样仅由与NFA图形和DFA图形所识别的字母相关联的给定图样的有效字符组成。

附图说明

[0013] 从如在附图中所示的本发明的示例性实施例的以下更具体的描述中，前述内容将显而易见，在附图中相同的附图标记指代所有不同视图中的相同部件。附图不一定按比例，而是着重于图示本发明的实施例。

[0014] 图1是图示系统的框图，在该系统中操作安全设备以保护专用网络。

[0015] 图2是可以由本发明使用的安全设备的框图。

[0016] 图3是示例NFA的NFA图形。

[0017] 图4是示例DFA的DFA图形。

[0018] 图5A-G是图示图形爆炸的原理的NFA和DFA图形和表。

[0019] 图6A-B是将NFA图形转换成DFA图形的方法的流程图。

[0020] 图7是用于确定NFA图形中的状态的埃普西隆终止的方法的流程图。

[0021] 图8A-C是用于将NFA图形转换成DFA图形的方法的流程图。

[0022] 图9是用于确定NFA图形中的状态的埃普西隆终止的方法的流程图。

[0023] 图9A-9B是用于将NFA图形转换成DFA图形的方法的流程图。

[0024] 图9C是用于确定用于一组NFA状态的埃普西隆终止的流程图。

[0025] 图10图示了用于使用Aho-Corasik算法来搜索例如图样“她的”、“他的”和“她”的图样匹配机制。

[0026] 图11A图示了图样匹配机制的每个状态的故障值。

[0027] 图11B图示了图样匹配机制的状态的输出函数值。

[0028] 图12图示了例如锚定的图样“帮助”和“贝壳”的状态树。

[0029] 图12A图示了图12的状态树的每个状态的故障值。

[0030] 图12B图示了图12的状态树的状态14和19的输出函数值。

[0031] 图13A图示了图12中的状态树的每个状态的故障值。

[0032] 图13B图示了图12的状态树的状态12、14、17和19的输出函数值。

具体实施方式

[0033] 在详细描述本发明的示例性实施例之前，以下马上描述其中可以实现实施例的示例安全应用以及使用DFA和NFA的典型处理，以有助于读者理解本发明的发明特征。

[0034] 正则表示式(Regex)处理在许多分组处理系统中变得普遍。Regex处理可以应用于传统的安全系统(例如，入侵防止系统(IPS)、防火墙和统一威胁管理(UTM)设备)、较新的安

全系统(例如,防恶意软件、反间谍软件、零日附加检测)、在有线/无线网络中用于计费,服务质量(QoS)和网络监控系统的新兴的协议/应用识别系统。

[0035] 正则表示式处理可以被细分为两个阶段i)将签名/图样编译成二进制数据结构,诸如DFA图形或NFA图形,以及ii)根据编译的图形处理接收到的分组。

[0036] 存储与性能的折衷要求在正则表示式处理的两个阶段均发生。分配有大的运行时间内存占用的编译器能够以较高的速度和效率来编译图样。类似地,相对于紧凑图,用于分组检查的较大的图或等效二进制数据结构可以给出更好的分组检查性能。

[0037] 而在实践中,期望编译器以尽可能少的内存占用来非常快速地编译规则。一个原因是,在网络设备仍然在运行(例如检查/转发分组)时在网络设备(例如,路由器、交换机、UTM等)的字段中更新图样。因此,需要使用在嵌入的路由器设备中的有限内存来编译规则。因为规则/图样用于防止对系统的攻击或停止被病毒感染的业务,所以需要尽可能早地应用规则/图样以便于优化系统的安全性。因此,编译器应当能够非常快地将规则编译成二进制数据结构。

[0038] 一般的方法在中央服务器上将新的图样或签名编译成图形,该中央服务器然后向路由器传送编译了的图形。然后,路由器通过使分组经过每个图形来根据接收到的图形检查到达的分组。高效的编译器需要足够的内存资源。如果编译器没有足够的资源,那么编译器性能很慢。因此,简单方法不在路由器上编译新的图样或特征,因为路由器通常没有足够的资源(即,随机存取存储器(RAM)和CPU计算)。

[0039] 本发明的实施例在路由器上将新的图样/签名编译成图形,同时保持中央服务器编译器的性能水平。

[0040] 图1是图示系统100的框图,该系统包括安全设备110、受保护的网络115和公共网络105。公共网络105可以包括不安全的广域网(WAN),诸如因特网、无线网络、局域网或者其他类型的网络。受保护的网络115可以包括安全的计算机网络,诸如办公室或数据中心的局域网。如所示,局域网可以是包括多个工作站125的企业网络120。多个工作站125可操作地耦合到数据库130、FTP(文件传输协议)服务器135和因特网服务器140。

[0041] 在系统100中,安全设备110连接到公共网络105和受保护的网络115,使得从公共网络105流动到受保护网络115的网络业务首先流动到安全设备110。安全设备110可以是独立的网络设备(例如,路由器)、另一网络设备(例如,防火墙设备)的组件、在网络设备上执行的软件模块或其它配置。通常,安全设备检查来自公共网络105的网络业务,并且确定网络业务是否包括任何计算机安全威胁。计算机安全威胁是获取对敏感信息的访问的尝试、破坏组织的操作的尝试或其他类型的攻击。示例计算机安全威胁包括计算机病毒、间谍软件、流氓软件(rootkit)、猜测密码的尝试、钓鱼邮件、与拒绝服务添加相关联的请求以及其他类型的攻击。

[0042] 计算机安全威胁可以与一个或多个符号图样相关联,该一个或多个符号图样识别计算机安全威胁,但不识别无害的数据。与计算机安全威胁相关联的符号图样在本文中被称为“威胁签名”。例如,特定的病毒可能总是包括指令序列中,当被执行时该指令序列执行恶意操作。

[0043] 如果安全设备110确定了给定的网络业务流不包括任何计算机安全威胁,则安全设备110可以将网络业务流传递到受保护的网络115。否则,如果安全设备110确定了该流包

括一个或多个计算机安全威胁，则安全设备110可以丢弃该网络业务、记录该网络业务、将该业务转发到业务分析器以用于进一步分析，和/或执行关于该网络业务的一些其他动作。以该方式，安全设备110可以防止包括计算机安全威胁的网络业务到达受保护的网络115。

[0044] 为了检测与一个或多个符号图样相关联的安全威胁，安全设备110从安全数据中心104接收在来自公共网络105的到达数据业务中要被监测的给定的图形或符号序列。一旦安全设备接收到要监测的给定图样，安全设备就针对要监测的每个给定的图样创建有限状态机。安全设备110使接收到的数据分组经过有限状态机，以确定到达的数据分组是否包括潜在的安全威胁。

[0045] 图2是可以和本发明一起使用的示例性安全设备200的高层框图。安全设备包括经由存储器总线245耦合到处理器225的存储器210以及经由输入/输出(I/O)总线250耦合到处理器的存储设备230和网络接口240。应当注意，安全设备可以包括其他设备，诸如键盘、显示单元等。网络接口240将安全设备与安全网络115、公共网络105和安全数据中心104对接，并且使得数据(例如，分组)能够在安全设备和系统100中的其他节点之间进行传送。为此，网络接口240包括常规电路，包含信号、电气和机械特征，以及交换电路，需要该交换电路以与系统100的物理介质和在该介质上运行的协议对接。

[0046] 存储器210是实施为RAM的计算机可读介质，包括诸如DRAM设备和/或闪存设备的RAM设备。存储器210包含各种软件以及由处理器225使用的数据结构，包括实施本发明的各方面的数据结构。具体地，存储器210包括操作系统215和图样匹配/编译服务220。操作系统215在功能上通过调用支持软件处理的操作和在安全设备200上执行的服务来组织安全设备200，诸如图样匹配/编译服务220。如下面将要描述的，图样匹配/编译服务220包括计算机可执行指令，用于从给定图样编译有限状态机图形和/或使到达的数据分组经过编译的图形。

[0047] 存储设备230是常规的存储设备(例如，磁盘或更可能的DRAM)，其包括图样匹配数据库(DB)235，该图样匹配数据库(DB)235是配置成保持用于从给定图样编译有限状态机的各种信息的数据结构。信息可以包括签名图样、有限状态机图形(例如，DFA图形和NFA图形)、埃普西隆终止(EC)高速缓存表以及DFA状态哈希表。

[0048] 通常，内容感知应用处理使用确定性有限自动机(DFA)或非确定性有限自动机(NFA)来识别在接收到的分组的内容中的图样。DFA和NFA都是有限状态机，即计算模型，计算模型中的每一个都包括状态集合、开始状态、输入字母(所有可能的符号集合)和转换函数。计算在启动状态中开始，并且根据转换函数而改变为新的状态。

[0049] 图样通常使用正则表示式来表述，正则表示式包括基本元素，例如，诸如A-Z、0-9的正常文本字符以及诸如*、^和|的元字符。正则表示式的基本元素是要被匹配的符号(单个字符)。这些与允许级联(+)、替换(|)和克林星号(*)元字符组合。用于级联的元字符用于从单个字符(或子串)创建多个字符匹配模式，而用于替换(|)的元字符用于创建可以匹配两个或更多个子串中任何一个的正则表示式。元字符克林星号(*)允许图样匹配任何数目，包括不出现前面字符或字符串。结合不同的运算符和单个字符允许构建复杂的表示式。例如，表示式(th(is|at)*)将匹配以下字符串：th，this，that，thisis，thisat，thatis或thatat。

[0050] 字符类结构[...]允许列出要搜索的字符的列表，例如gr[ea]y查找grey和gray。

破折号指示字符的范围，例如[A-Z]。元字符“.”匹配任何一个字符。

[0051] 对DFA或NFA状态机的输入通常是(8位)字节的串，即，字母是单个字节(一个字符或符号)。输入流中的每个字节产生从一个状态到另一状态的转换。

[0052] DFA或NFA状态机的状态和转换函数可以通过图来解释，其中图中的每个节点表示状态，并且图中的圆弧表示状态转换。状态机的当前状态由选择特定图形节点的节点识别符来表示。

[0053] 使用DFA来处理正则表示式并且找到字符的输入流中由正则表示式描述的一个或多个图样的特征在于：

[0054] 1)确定的运行时间性能：DFA的下一个状态可以从输入字符(或符号)以及DFA的当前状态来确定。换言之，每DFA状态仅存在一次状态转换。这样，DFA的运行时间性能被认为是确定的并且行为可以从输入完全预测。

[0055] 2)支持跨多个分组匹配所需要的较小的每流上下文(例如，状态或节点指针)：在搜索跨越构成流的若干分组的输入的图样中，搜索可能在一个分组处停止，并且然后在另一个分组处恢复。通常，确定要恢复搜索的状态需要跟踪、记住或以其他方式存储(例如，作为状态指针)直至搜索停止时所经历的所有状态。然而，在DFA中，为了恢复搜索，仅需要记录搜索停止时的状态。这样，可以说，DFA被特征化为需要较小的每流上下文，以支持跨多个输入分组的图样匹配，例如，以几字节的量级存储状态或节点指针。

[0056] 3)节点的数目(或图形大小)随着图样的大小呈指数增长的图形。

[0057] 相比之下，使用NFA来处理正则表示式并且找到由字符的输入流中的正则表示式描述的图样的特征在于：

[0058] 1)非确定的运行时间性能：给定输入字符(或符号)和NFA的当前状态，可能存在要转换到的多于一个NFA的下一状态。换言之，NFA的下一状态不能从NFA的输入和当前状态来确定。这样，NFA的运行时间性能被认为是不确定的，并且行为不能完全从输入预测。

[0059] 2)支持跨多个分组匹配所需要的较大的每流上下文(例如，状态或节点指针)：如前所述，图样跨多个输入分组匹配，其中搜索在一个分组处停止，然后在另一分组处恢复，需要跟踪直至在分组停止时所经历的所有状态。在NFA中，越多输入被匹配，经历的状态和需要跟踪的状态的数目就越多。这样，可以说，与DFA相比，NFA被特征化为需要较大的每流上下文，以支持跨多个输入分组的图样匹配。

[0060] 3)节点的数目(或图形大小)随着图样的大小呈线性增长的图形。

[0061] 进一步参考附图3、4和5-G来讨论上述DFA和NFA的特征。应当注意，为简单起见，对于附图中所示的所有DFA图形，没有示出到节点(状态)0的弧(状态转换)，并且也没有示出对于相同字符的到与由节点0指向的节点相同的节点的弧。

[0062] 图3示出了用于搜索图样“cavium.*networks”、“nitrox[^\r\n\t\v\s]{3}octeon”和“purevu.{5,10}videochips”的示例NFA的NFA图形300。图4示出了用于搜索相同的图样集合的示例DFA的DFA图形400。如上所述，应当注意，DFA图形400和本文中所提供的其他DFA图形是出于绘制的目的而被“简化”的。在附图中没有示出表示到DFA状态0的状态转换的到节点0的弧。在附图中也没有示出对于相同字符的到与由节点0指向的节点相同的节点的弧。

[0063] 对于相同的图样集合，图3的NFA图形300具有69个节点，表示69个状态，而图4的

DFA图形400具有931个节点(在图4中仅示出了其中的一部分),表示931个状态。如示,对于给定的一个或多个图样,DFA状态的数目可以大于NFA状态的数目,通常多了几百或几千量级的状态的。这是“图形爆炸”的示例,这是DFA的标志性特征。

[0064] 为了进一步描述“图形爆炸”的概念,考虑分别示出了对于图样“.*a[^\n]”、“.*a[^\n][^\n]”、“.*a[^\n][^\n][^\n]”的NFA图形的图5A、图5B和图5C,以及示出了对于相同图样的DFA图形的图5D、图5E和图5F。其中,在图5D、5E、5F中没有示出看到“\n”的情况下的来自每个状态的返回箭头。如图5A-5F中所示以及由图5G的表所总结的,对于一些图样,NFA可以线性增长,而DFA可以指数地增长而产生图形爆炸。

[0065] 返回图3,使用图形300表示的NFA来搜索输入流“purevuchips are video chips”中的图样,NFA处理或匹配开始于NFA的开始状态0、2、19和36,由节点305a-d表示并且简写为NFA START STATES={0,2,19,36}。关于输入流的字符“p”,NFA转换到状态37(由节点310表示)并且跟踪状态0、2和19(简写为对“p”={0,2,19,37}),并且继续如下:

[0066] 对“u”={0,2,19,38}

[0067] 对“r”={0,2,19,39}

[0068] 对“e”={0,2,19,40}

[0069] 对“v”={0,2,19,41}

[0070] 对“u”={0,2,19,42}

[0071] 对“c”={0,2,19,44}

[0072] 对“h”={0,2,19,45}

[0073] …

[0074] ...等,

[0075] 使用图4的DFA图形400表示的DFA来搜索在相同输入中的相同图样,DFA匹配开始于DFA开始状态0,由节点405表示并且简写为DFA START STATES={0}。对于输入流的字符“p”,DFA转换到状态3,由节点410表示并且简写为对“p”={3},并且继续如下:

[0076] 对“u”={6}

[0077] 对“r”={9}

[0078] 对“e”={12}

[0079] 对“v”={15}

[0080] 对“u”={18}

[0081] 对“c”={27}

[0082] 对“h”={41}

[0083] …

[0084] ...等。

[0085] 如在上面的示例中所示,在NFA中,存在至少n+1个数目的NFA状态要跟踪,其中n是要搜索的图样的数目(例如,在要搜索3个图样的情况下,存在至少4个要跟踪的状态)。与此相反,在DFA中,每个输入字符仅存在一个状态要跟踪。为了说明的目的,现在假定输入流(stream)或流(flow)“purevuchips are video chips”跨越多个分组,其中第一分组以“purevuchips”的“h”结束,并且第二分组以“purevuchips”的“i”开始。在NFA中,搜索在“h”(第一分组的结束)停止,具有四种状态要跟踪(即,状态0,2,19和45)。为了在“i”恢复搜索

(第二分组的开始)，需要记住这四个状态。相反，在DFA中，搜索在“h”(第一分组的结束)停止，具有一个状态被跟踪(即，状态41)。为了在“i”恢复搜索(第二分组的开始)，需要记住这一个状态。该示例示出，在NFA中，支持跨多个分组的匹配需要的的每流上下文是四个状态(例如，通过存储四个状态指针)，而在DFA中，每流上下文是一个状态。因此，NFA需要的每流上下文比相同图样的DFA所需要的每流上下文更大。类似地，DFA需要的每流上下文比相同图样的NFA所需要的每流上下文更小。

[0086] 对于每个非确定性有限自动机，存在等效的确定性有限自动机。这两者之间的等效以语言接受来定义。因为NFA是有限自动机，其中，对于输入符号允许0、1或多次转换，所以等效的DFA可以被构造为模拟关于特定输入符号的并行的NFA的所有移动。

[0087] 由于NFA的DFA等效模拟(并行)NFA的移动，DFA的每个状态是NFA的一个或多个状态的组合。因此，DFA的每个状态将由NFA的状态集合的某个子集来表示；并且因此，从NFA到DFA的转换通常被称为“构建”子集。因此，如果给定的NFA具有n个状态，则等效的DFA可以具有2^n数目的状态，其中初始状态对应于子集$\{q_0\}$。因此，从NFA到DFA的转换涉及找到NFA的状态集合的所有可能的子集，考虑每个子集是DFA的状态，并且然后找到对于每个输入符号从该状态的转换。

[0088] 如上所述，由于NFA的多个可能的转换，由计算机系统对NFA图形进行处理是困难的，因此NFA到DFA的转换发生。

[0089] 图6A-B是用于将NFA图形转换为DFA图形的方法600的流程图。该方法600开始于605。在该阶段，DFA状态集合“Sd”为空。在610，DFA的开始状态被确定并添加到DFA状态集合“Sd”作为未标记的状态。DFA的开始状态被确定为NFA图形的开始状态的埃普西隆终止。以下参考图7来进一步描述确定NFA状态集合的埃普西隆终止的方法。

[0090] 在615，确定DFA状态集合“Sd”是否包括未标记的DFA状态。如果该DFA状态集合“Sd”的未标记的DFA状态存在，则在620，未标记的状态“S”被选择并标记。在625，由NFA图形所识别的语言“A”的字母(例如，文字)被选择。在步骤630，DFA状态“S”的NFA状态“s”被选择。此外，步骤630之前，用于保持NFA状态集合的数据结构“St”被设置为“空”。在635，转换函数“TTn＝(s,a)”被使用字母“a”应用于NFA状态“s”。如果接收到“a”的输入，转换函数确定从NFA状态“s”到达的所有NFA状态。然后，确定的NFA的状态被添加到该数据结构“St”。在644，确定DFA状态“s”是否包括其他的NFA状态。如果是，则该方法重复步骤630和635，直到DFA状态“s”的所有NFA状态“s”已经被处理。如果已经处理了所有的NFA状态，则该方法在步骤640继续。在640，在数据结构“St”中的所有NFA状态“s”的埃普西隆终止被确定并且被添加到数据结构“St”。

[0091] 在步骤645，将数据结构“St”与所有现有的DFA状态“s”作比较，以确定DFA状态“S”是否已经包括数据结构“St”中的所有NFA状态“s”。当前的方法存储与数据结构中的每个DFA状态“S”相关联的每个NFA状态“s”。为了确定数据结构“St”中的NFA状态是否已经与DFA状态“S”相关联，数据结构“St”的“每个NFA状态“s”必须与每个DFA状态“S”每个NFA状态“s”作比较。因此，这样的比较需要大量的时间和内存。

[0092] 以下表1说明了将DFA状态数与NFA状态集合相关联的示例DFA状态表。NFA状态集合可以针对每个DFA状态数被存储在数据结构中，如上所述。

[0093] 表1

[0094]

DFA状态数	NFA状态集合
0	{0,1,2,3,4}
1	{0,5,6,7}
2	{8,9,2,3}
…	…
…	…

[0095] 例如，根据数据结构的实现(包含DFA状态及其相应的NFA状态集合)，以下参考表2来获得在步骤645的操作的运行时间。表2列出了示例数据结构实现的存储和维护成本。表2的注释列提供了每个示例数据结构的描述。对于每个数据结构，假定存在“N”个DFA的状态，并且进一步假设每个DFA状态平均表示’M’个NFA状态。

[0096] 表2

[0097]

运行时间	存储	数据结构的维护（NFA状态的插入）	注释
$O(N*M^2)$	O(N*M)	O（1）	NFA状态的每个集合（与DFA状态相关联）被存储为包含NFA状态数的线性阵列。
O(N*M)	O(N*NFA状态的最大数目/8)	O（1）	每个DFA状态具有大小等于系统中的NFA状态的最大数目的位图。单独位标识每个NFA状态。如果DFA状态具有该NFA状态，则相应位被打开
O(N*logM)	O(N*k*M)，其中k是树指针等的恒定成本。	O（logM）	NFA状态的每个集合（与DFA状态相关联）被存储为包含NFA状态数的树。

[0098] 继续图6A-B，如果DFA状态“s”已经包括数据结构“St”的所有NFA状态“s”，则该方法移动到步骤695.如果没有，则该方法移动到步骤685.在685，数据结构“St”中的NFA状态“s”被添加到DFA状态集合“Sd”作为新的DFA状态“D”。在690，确定NFA状态“s”中的任何一个是否属于NFA图形的最终状态集合。如果是，则新的DFA状态被确定为NFA图形的最终接受状

态。在695，具有输入“s”的从标记的DFA状态“S”的转换被设置为新的DFA状态“D”。

[0099] 图7是方法700的流程图，该方法700用于确定对于NFA图形的任何给定的NFA状态“s”的埃普西隆终止。方法700开始于步骤705。在710，埃普西隆终止函数接收用于处理的NFA状态“s”。在715，NFA的状态“s”没有被标记，并且被添加到未标记的NFA状态集合“s”。在720，从未标记的NFA状态“S”中选择NFA状态“s”。在725，确定从选择的NFA状态“s”的埃普西隆转换。在步骤730，确定是否存在任何埃普西隆转换。如果没有，则在740，所选择的NFA状态’s’被添加在埃普西隆终止集合中。如果存在转换，那么在步骤745，从选择的NFA状态’s’的所有确定的NFA转换被添加到集合“S”作为未标记的NFA状态。步骤750确定在NFA状态“s”中是否存在任何未标记的NFA状态。如果有，则方法从步骤720继续。如果没有，该方法在步骤755结束。

[0100] 用于以上参考的方法的示例伪代码#1(开始于图6A的步骤615)如下：

[0101] 1.对于每个未标记的DFA状态“d”(列-1——在DFA状态表中，上述表1)

[0102] 1.对于字母集合中的每个字母“a”

[0103] 1.集合S＝{}

[0104] 2.对于“d”的每个NFA状态“n”(列-2——在DFA状态表，上述表1)

[0105] 1.如果“n”具有对“a”的到“m”弧的外出弧

[0106] 1.S＝S U{m}

[0107] 3.SE＝ECLOSURE(S)

[0108] 4.将发现指定为“假”

[0109] 5.对于DFA状态表中的每个DFA状态，上述表1

[0110] 1.使“p”是对应于DFA状态“f”的NFA状态集合

[0111] 2.如果集合“Se”和“p”相等

[0112] 1.将发现指定为“真”

[0113] 2.去往1.6

[0114] 6.如果发现为“真”

[0115] 1.设置转换(“d”，“a”)＝“f”

[0116] 7.其他

[0117] 1.添加新的DFA状态“f”到DFA状态表，表1以“Se”作为COL.2中的NFA状态的集合

[0118] 2.设置转换(“d”，“a”)＝“f”

[0119] 2.将“d”设置为“标记的”

[0120] 上述参考方法的缺点如下：i)伪代码的步骤1.1.3，表示方法700和600，总是计算埃普西隆终止(ECLOSURE())，因为在存储器中没有存储埃普西隆终止的历史；ii)步骤1.1.5.2由于集合等效测试而非常耗时。由该集合等效测试所消耗的时间取决于要比较的集合中的元素的数目(即，在DFA状态中的NFA状态数(如表1中示出)；以及iii)表1中的条目不能被删除因为条目对于执行集合等效测试的步骤1.1.5.2时需要的，由此需要大量的内存资源。

[0121] 在本发明的示例实施例中，NFA图形被转换为等效DFA图形。

[0122] 图8A-C是按照本发明的示例实施例的用于将NFA图形转换为NFA图形的方法800的流程图。该方法800开始于805。在该阶段，DFA状态的集合“Sd”为空。在810，DFA的开始状态

被确定并被添加到DFA状态的集合“Sd”作为未标记的状态。DFA的开始状态被确定为NFA图形的开始状态的埃普西隆终止。以下参考图9来进一步描述按照本发明的示例实施例的确定NFA状态的埃普西隆终止的方法。在815，与DFA开始状态相关联的NFA状态的加密/完美哈希值被计算并存储表中，该表使DFA状态与关联于DFA状态的NFA状态的哈希值相关，如以下表4中所示。在820，NFA开始状态的埃普西隆终止被存储在在表5中进一步所示的埃普西隆高速缓存中。埃普西隆高速缓存被输入NFA状态的集合的哈希所检索，并且所存储的数据是计算的输入NFA状态的埃普西隆终止。

[0123] 加密哈希/完美哈希函数是确定性过程，该过程需要取得任意数据块，并且返回固定大小的位串，该位串是加密的哈希值。示例加密哈希函数包括，例如，消息摘要算法(MD5)或安全哈希算法(SHA1/SHA2)。在较大摘要(例如，对于MD5为128b)的情况下，冲突的机会不太可能。然而，“完整性检查”可以离线进行，以验证不存在冲突(具有相同哈希值的不同数据集合)，使得如果发生冲突，则可以校正图形。

[0124] 在825，确定DFA状态集合“Sd”是否包括未标记的DFA状态。如果不包括，该方法在步骤895结束。如果该DFA状态集合“Sd”的未标记的DFA状态存在，则在830，未标记的状态“S”被选择并且标记。在835，由NFA图形识别的语言“A”的字母(例如，文字)被选择。此外，用于保持NFA状态集合的数据结构“St”被设置为“空”。在步骤840，选择与DFA状态“S”相关联的NFA状态“s”。在850，使用字母“a”输入值，将转换函数“TTn＝(s,a)”应用于NFA状态“s”。如果接收到“a”的输入，转换函数确定从NFA的状态”s”到达的所有NFA状态。在855，确定的NFA的状态然后被存储在数据结构“St”中。在860，确定DFA状态“S”是否包括额外的关联的NFA状态。如果包括，则该方法在步骤850和855重复，直到DFA状态“S”的所有NFA状态的“s”已经被处理。如果所有的NFA状态已经被处理，则该方法在步骤865继续。在865，数据结构“St”中的所有NFA状态”s”的埃普西隆终止按照图9被确定并且被添加到数据结构“Se”。

[0125] 在步骤870，将数据结构“Se”与所有存在的DFA状态“S”作比较，以确定DFA状态“S”是否将所有的NFA状态“s”包括在数据结构“Se”中。如以上参考方法600的步骤645所述，一般的方法将与每个DFA状态“S”相关联的NFA状态“s”的集合存储在数据结构中。为了确定数据结构“Se”中的NFA状态“s”是否已经与DFA状态“S”相关联，必须针对每个DFA状态“S”来将数据结构“Se”的每个NFA状态集合“s”与每个NFA状态集合“s”作比较。因此，这样的比较需要大量的时间和内存，如表2中所示。

[0126] 在本实施例中，数据结构“Se”中的NFA状态的加密/完美哈希值被计算并且然后与表作比较，该表将DFA状态数与其一个或多个NFA状态的对应集合的哈希值相关。如果匹配哈希值存在，那么，在步骤870，确定与数据结构“Se”中的NFA状态相关联的DFA状态是否已经存在，并且该方法移动到步骤890。在890，具有字母“a”的输入的从DFA状态“S”的转换被设置为与匹配哈希值相关联的现有DFA状态。该方法移动到步骤845，其中作出关于另一字母“a”是否存在于语言‘A’中的确定，如果存在，则该方法从步骤835重复。如果不存在，则该方法移动到步骤847。在步骤847，该方法删除NFA状态数集合，并且将状态集合添加为标记的。然后，该方法在步骤825继续。

[0127] 以下参考表3来获得在步骤870处的操作的运行时间。表3列出了根据本发明的示例实施例的哈希匹配的储存和维护成本。表3的注释列提供了哈希匹配的描述。

[0128] 表3

[0129]

运行时间	存储	数据结构的维护（NFA状态的插入）	注释
O（N）	O（N）	O（N）	NFA状态的（与DFA状态相关联）每个集合用其等效加密哈希来替代。

[0130] 如以下所示的表4是DFA表，如上所述，该表将DFA状态数与其一个或多个NFA状态的对应集合的哈希值相关。

[0131] 表4

[0132]

DFA状态数	DFA状态哈希	NFA状态数集合	标记(M)/未标记(U)
0	81237912891273	删除	M
1	09237504823405	删除	M
2	23894729379237	{4,5,6,2,0,1}	U
3	89345798731278	{4,2,3,7,1,8}	U
…	…	…	

[0133] 继续图8，如果在步骤870不存在匹配哈希值，则该方法移动到步骤875。在875，数据结构“Se”中的NFA状态“s”被添加到DFA状态集合“Sd”作为新的未标记的DFA状态(例如，“D”)。此外，NFA状态“s”的加密/完美哈希值被计算，并且该新的DFA状态被映射到上述表4中的哈希值。在880，确定NFA状态“s”中的任何一个是否属于NFA图形的最终状态集合。如果是，则新的DFA状态被确定为NFA图形的最终接受状态。在885，具有输入“a”从标记的DFA状态“S”的转换被确定为新的DFA状态“D”。

[0134] 图9是方法900的流程图，该方法用于按照本发明的示例实施例确定NFA图形中的状态集合的埃普西隆终止。方法900开始于步骤905。在910，埃普西隆终止计算器接收到NFA状态的集合。在915，将NFA状态的集合与高速缓存内的条目作比较，以确定高速缓存是否包含匹配的NFA状态的集合。

[0135] 高速缓存内的每个条目可以被存储为哈希值，该哈希值表示映射到NFA状态的输入集合的埃普西隆终止的NFA状态集合。计算器根据接收到的NFA状态的集合计算哈希值，并且确定EC高速缓存是否具有与NFA状态的集合相关联的匹配哈希值条目。

[0136] 如以下所示，表5是埃普西隆终止高速缓存表，如上所述，该表将NFA状态的集合映射到其埃普西隆终止。

[0137] 表5

[0138]

EC输入集合哈希	EC输出集合
78346782346782	{3,4,1,2,7}
89237489237492	{8,3,2,5,19}
…	…

[0139] 继续图9，如果发现匹配，则在920返回埃普西隆终止。然而，如果没有发现匹配，则计算接收到的NFA状态的集合的埃普西隆终止。如以上参考图7描述的方法700所述来计算埃普西隆终止。一旦计算了埃普西隆终止，在930，埃普西隆终止被存储作为埃普西隆高速缓存中的新的条目。在935，返回计算的埃普西隆终止。

[0140] 方法900通过消除冗余处理而允许埃普西隆终止的高效处理。例如，方法900仅当还没有计算埃普西隆终止时计算NFA状态集合的埃普西隆终止。这消除了对于一个NFA集合进行多于一次的埃普西隆终止的处理的需要。参考以上参考图6A-B描述的方法600，该方法可以多于一次地计算任何给定NFA节点的埃普西隆终止。然而，通过将先前计算的埃普西隆终止存储在高速缓存中，方法900消除了不必要的数据处理的需要。

[0141] 以上所参考的方法的示例伪代码#2(开始于图8A的步骤825)如下：

[0142] 1.对于每个未标记DFA状态“d”(列-1—在DFA状态表中，上述表4)

[0143] 1.对于字母集合中的每个字母“a”

[0144] 1.设置S＝{}

[0145] 2.对于“d”的每个NFA装置“n”(列-3—在DFA状态表中，上述表4)

[0146] 1.如果“n”具有对于“a”的至“m”的外出弧

[0147] 1.S＝S U{m}

[0148] 3.获取集合“S”的ECLOSURE“Se”，如以下给出

[0149] 1.计算集合“S”的哈希“Hi”

[0150] 2.对于EC高速缓存表中的每个条目“e”，上述表5

[0151] 1.使“He”是条目“e”处的哈希值(列-1—在上述EC高速缓存表中)

[0152] 2.如果“Hi”和“He”是相同的

[0153] 1.Se＝EC输出集合(e)，即，列-2—在上述EC高速缓存表中

[0154] 2.前进到1.1.4

[0155] 3.Se＝ECLOSURE(S)

[0156] 4.以字段“Hi”和“Se”将新的条目添加到上述EC高速缓存表

[0157] 4.将发现指定为“假”

[0158] 5.计算集合“Se”的哈希“q”

[0159] 6.对于DFA状态表中的每个DFA状态“f”，上述表4

[0160] 1.使得“p”是DFA状态“f”的NFA状态的哈希

[0161] 2.如果“p”和“q”相同

[0162] 1.将发现指定为“真”

[0163] 2.前进到1.1.7

[0164] 7.如果发现为“真”

[0165] 1.设置转换(“d”，“a”)＝“f”

[0166] 8.其他

[0167] 1.将新的DFA状态“f”添加到DFA状态表，具有字段“q”和“Se”的表4

[0168] 2.设置转换(“d”，“a”)＝“f”

[0169] 2.从DFA状态表、表4中删除DFA状态“d”的NFA状态数的集合，并且将“d”设置为标记的。

[0170] 上面所参考的方法的优点如下：i)如果已经计算过，则步骤1.1.3避免计算ECLOSURE()；ii)步骤1.1.6.2是哈希值的比较，其大小是恒定的，并且花费固定时间量用于比较，与集合等效测试相比更好。时间量是O(1)，如以上表3中所示；以及iii)因为在进行处理之后删除DFA状态的NFA状态集合，所以节省编译器的大量内存占用，如表4所示。

[0171] 另一优化在于，EC_CACHE还可以存储直接DFA状态数，而不是对应于(NFA状态的)埃普西隆终止集合的NFA状态集合，使得完全不需要步骤870。例如，如果在EC_CACHE()中存在命中，则不需要搜索等效的DFA节点。如以下所示，表6是示例EC_CACHE表，其存储对应于(NFA状态的)埃普西隆终止集合的直接DFA状态数。

[0172] 表6

[0173]

EC 输入集合哈希	DFA 状态数

[0174]

78346782346782	13
89237489237492	14
…	

[0175] 因此，步骤870的处理表变为：

[0176] 表7

[0177]

运行时间	存储	数据结构的维护（NFA 状态的插入）	注释
O(1)或零处理	O(N)	O(1)	在 EC_CACHE 中命中的情况下。因为 EC_CACHE 包含对应的 DFA 数。

[0178] 图9A-B是方法901的流程图，按照本发明的示例实施例，该方法用于将NFA图形转换成DFA图形。方法901开始于902。在该阶段，DFA状态集合“Sd”为空。在903，DFA的开始状态被确定并且被添加到DFA状态集合“Sd”作为未标记的状态。DFA的开始状态被确定为NFA图形的开始状态的埃普西隆终止。以下参考图9C来进一步描述按照本发明的示例实施例的确定NFA状态的埃普西隆终止的方法。在904，与DFA开始状态相关联的NFA状态的加密/完美哈希值被计算，并且在906，DFA开始状态和加密哈希的映射被存储在表中，该表将DFA状态与关联于该DFA状态的NFA状态的哈希值相关，如上述表6中所示。

[0179] 在907，确定DFA状态集合“Sd”是否包括未标记的DFA状态。如果不包括，则该方法在步骤908结束。如DFA状态集合“Sd”的未标记的DFA状态存在，则在909，未标记的状态“d”被选择和标记。在911，由NFA图形所识别的语言’A’的字母(如文字)被选择。此外，用于保持

NFA状态集合的数据结构的“S”被设置为“空”。在步骤913，与DFA状态“d”相关联的NFA状态’n’被选择。在914，转换函数“TTn＝(s,a)”被应用于NFA状态“n”，使用字母“a”输入值。如果接收到“a”的输入，则该转换函数确定从NFA状态“n”到达的所有NFA状态。在916，然后将确定的NFA状态存储在数据结构“S”中。在917，确定DFA状态’d’是否包括额外的相关联的NFA状态。如果是，则该方法在步骤913重复，直到DFA状态“d”的所有NFA状态’n’已经被处理。如果所有的NFA状态已经被处理，则该方法在步骤918继续。在918，根据字母“a”的从DFA状态“d”的转换被按照图9C确定。在步骤919，具有字母“a”的输入的从DFA状态“d”的转换状态“f”被设置并存储在DFA状态表中。在921，从数据结构“S”中删除存储的NFA转换状态的集合。

[0180] 图9C是方法922的流程图，根据本发明的示例实施例，该方法用于确定NFA图形中的状态集合的埃普西隆终止。该方法922开始于步骤923。在924，埃普西隆终止计算器接收NFA状态集合并且针对接收到的NFA状态集合来计算哈希值“Hi”。在926，将哈希值“Hi”与在埃普西隆高速缓存内的条目作比较，以确定该高速缓存是否包含匹配。

[0181] 高速缓存中的每个条目可以被存储为哈希值，该哈希值表示映射到DFA状态的NFA状态的集合。计算器根据接收到的NFA状态集合计算哈希值，并且确定EC高速缓存是否具有与NFA状态集合相关联的匹配哈希值条目，该NFA状态集合与DFA状态相关。

[0182] 如果找到匹配，则在933，返回映射到高速缓存表的匹配哈希值的DFA状态“f”。然而，如果没有找到匹配，则在步骤928，计算接收到的NFA状态的集合的埃普西隆终止。可以如以上参考图7描述的方法700如上所述计算埃普西隆终止。一旦计算了埃普西隆终止，在929，埃普西隆终止的加密哈希被计算并作为新条目被存储在埃普西隆高速缓存中。在931，对应于NFA状态集合的哈希值的新DFA状态“f”被映射到EC高速缓存表中的哈希值。在932，返回新DFA状态“f”。

[0183] 上述所参考的方法的示例伪代码#3如下：

[0184] 1.对于每个未标记DFA状态“d”(列-1—在DFA状态表、上述表4中)

[0185] 1.对于字母集合中的每个字母“a”

[0186] 1.设置S＝{}

[0187] 2.对于“d”的每个NFA状态“n”(列-3—在DFA状态表、上述表4中)

[0188] 1.如果“n”具有对于“a”的至“m”的外出弧

[0189] 1.S＝S U{M}

[0190] 3.获取DFA状态“d”的关于字母“a”的转换DFA状态“f”如下

[0191] 1.计算集合“S”的哈希“Hi”

[0192] 2.对于EC高速缓存表、上述表6中的每个条目“e”，

[0193] 1.使“He”是条目“e”处的哈希值(列-1——上述EC高速缓存表6中)

[0194] 2.如果“Hi”和“He”是相同的

[0195] 1.向条目“e”的DFA状态数指派“f”，即，列-2—在上述EC高速缓存表中

[0196] 2.前进到1.1.5

[0197] 3.Se＝ECLOSURE(S)

[0198] 4.计算集合“Se”的哈希“q”

[0199] 5.将发现指定为“假”

[0200] 6.对于上述DFA状态表中的每个DFA状态“g”

[0201] 1.使得“p”是DFA状态“g”的NFA状态的哈希

[0202] 2.如果“p”和“q”相同

[0203] 1.将发现指定为“真”并且将“g”指派给为“f”

[0204] 2.前进到1.1.3.7

[0205] 7.如果发现为“真”

[0206] 1.前进到1.1.4

[0207] 8.使用“q”和“Se”将新的未标记DFA状态“f”添加到DFA状态表

[0208] 4.使用哈希“Hi”和DFA状态数“f”将新的条目添加在EC EACHE表中

[0209] 5.设置转换(“d”,“a”)=“f”

[0210] 2.从DFA状态表、上述表4中删除DFA状态“d”的NFA状态数的集合,并且将状态设置为“标记的”。

[0211] EC_CACHE的大小根据允许的运行时间内存占用是可配置的并且是有限的。如果运行时间内存占用是有限的,则需要有替换策略。例如,替换策略可能保留NFA状态集合的最近最少使用的埃普西隆终止(EC)或者完全没有替换。在后一种情况下,EC_CACHE仅保持固定数目的NFA状态集合的预定EC。后一种情况已被发现是非常有用的。

[0212] 上面所参考的方法的优点如下:i)如果已经计算过,则步骤1.1.3.2避免计算ECLOSURE();ii)如果可能(先前算法、伪代码#2中的步骤1.1.6.2),给定其ECLOSURE(),将DFA节点数目而不是ECLOSURE集合存储在EC高速缓存表中避免搜索DFA节点;以及iii)因为在表4中,进行处理之后删除DFA状态的NFA状态集合,所以节省编译器的大量内存占用。

[0213] 如上文所述,通常使用搜索算法,诸如,确定性有限自动机(DFA)或者非确定性有限自动机(NFA)来执行内容搜索以处理正则表示式。可以实现的另一类型的串搜索算法是Aho-Corasik算法。

[0214] Aho-Corasik算法可以被用于从文本串集合中创建有限状态图样机,该文本串集合可以被用于处理单次通过中的输入净荷。例如,给定串集合,Aho-Corasik算法创建用于处理任何的任意文本字串的有限图样匹配机,该任意文本字串可以经由通信网络中的分组接收。创建的图样匹配机的行为是由三个函数来控制:i)前往函数“g”,ii)故障函数“f”,以及iii)输出函数“输出”。

[0215] 图10示出了用于使用Aho-Corasik算法来搜索图样“hers”、“his”和“she”的图样匹配机1000。去往函数’g’将由状态和输入符号组成的对映射到状态或消息“故障”。故障函数“f”将状态映射到状态,并且每当前往函数“g”报告“故障”时进行查询。输出函数“输出”将一组关键字(可能为空)与每个状态相关联。

[0216] 开始状态是状态0(由节点1005表示)。在任何给定的状态下,如果前往函数“g(s,a)=t”(“s”是有限机的当前状态,“a”是输入值,并且“t”是转换状态),则图样匹配机1000进入状态“t”,并且输入严格的下一个符号变成当前输入符号。例如,参考图10,如果处于状态0(节点1005)并且接收到’s’输入值,则图样匹配机1000将转换为状态3(节点1015)。

[0217] 然而,如果前往函数“g(s,a)=故障”并且故障函数“f(s)=“s””,那么该图样匹配机以“s”作为当前状态并且以输入字母“a”作为当前输入符号而重复该循环。图11A示出了对于图样匹配机1000的每个状态的现有技术故障值。图11B示出了对于状态2、5、7和9的输

出函数值。

[0218] 例如，参考图10，任意输入串“ushers”由处理机处理如下：

[0219] 处理或匹配在开始状态0(由节点1005表示)开始。对于输入流的字符“u”，处理机保持处于状态0(节点1005)。对于输入流的字符“s”，处理机转换到状态3(节点1015)，并如下继续：

[0220] 对“h”＝{4}

[0221] 对“e”＝{5}

[0222] 对“r”＝{8}

[0223] 对“s”＝{9}

[0224] 在状态4(节点1020)，因为前往函数g(4，“e”)＝5，并且处理机1000进入状态5，关键字“she”和“he”在文本串“ushers”中位置4的末尾被匹配，并且输出函数发出输出(5)(如在图11B中看到的)。在状态5对于输入符号“r”，该机1000进行两次状态转换。因为g(5,r)＝故障，机1000进入状态2＝f(5)。此外，因为g(2,r)＝8，所以机1000进入状态8并且前进到下一输入符号。

[0225] 已经描述了其中可以实现本发明的示例实施例的示例安全应用和使用Aho-Corasik机1000的典型处理，以下马上具体描述本发明的示例实施例。

[0226] 如上所述，Aho-Corasik机1000在任意输入串的每个位置处检测关键字或图样的出现。在某些情况下，给定图样或关键字仅在输入串的特定区域或位置中被找到才是有意义的。例如，HTTP协议请求解析器仅在关键字“Get”在请求的开始时发生才对其感兴趣，并且此后不对任何其他“Get”感兴趣。这样的图样可以被称作锚定图样。

[0227] 本发明的实施例使得能够创建识别非锚定图样和锚定图样两者的Aho-Corasik图样匹配机。如上所述，图样匹配机1000使用Aho-Corasik算法来识别未锚定的图样“hers”、“his”和“she”。通过修改Aho-Corasik算法，图样匹配机可以被创建为识别额外的锚定图样“help”和“shell”。

[0228] 给定图样集合，锚定的图样必须与非锚定的图样进行区分。锚定的图样可以具有附加的宏，其指定该图样被锚定。例如，“{@0}”可以被添加到图样的开始以指定图样是锚定图样。因此，给定图样列表“he,she,his,hers，{@0}help和{@0}shell”，编译器能够识别关键字“help”和“shell”是锚定图样。

[0229] 一旦编译器接收到关键字/图样列表时，编译器也能够将非锚定图样从锚定图样中区分。然后，编译器使用如上所述的去往函数’g’，创建针对所有锚定图样的独立状态树和针对非锚定图样的独立状态树(图10中所示的机1000)。图12图示了针对锚定图样“help”和“shell”的状态树1200。图12A图示了按照现有技术针对状态树1200的每个状态的故障值(即，假设这些图样被作为非锚定图样编译)。图12B图示了状态树1200的状态14和状态19的输出函数值。

[0230] 一旦创建了针对锚定图样和非锚定图样的状态树，编译器计算两个状态数的故障函数“f”。对于表示非锚定图样的状态树，编译器实现如以上参考图10和11A描述的故障函数。按照本发明的示例实施例，锚定图样的故障函数按照如下规则被构建：

[0231] a)锚定树的根节点的故障被设置为等于非锚定状态树的根节点。因此，如果没有锚定图样被匹配，则跟踪非锚定图样。例如，状态10(节点1205)的故障“f”被设置为等于开

始状态0(图10中的节点1005)。

[0232] b)一旦确定了锚定树的开始状态的故障,锚定树的每个状态的故障“f”被确定,使得如图13A所示,非锚定关键字与锚定关键字的部分匹配也被使用前往函数“g”而跟踪。

[0233] 锚定图样的输出函数被单独计算,但是看起来保持与非锚定图样的重叠,如图13B中所示。

[0234] 此后,按照本发明的示例实施例,锚定的状态树的根节点被设置为最终状态树的根节点(锚定及非锚定状态树的组合)。

[0235] 现在,锚定状态树和非锚定状态树已经有效地合并成单个状态树。

[0236] 例如,参考图10、图11A-B、图12和图13A-B,任意输入的串“ushers”由处理机处理如下:

[0237] 处理或匹配在开始状态10(由节点1205表示)开始。对于输入流的字符“u”,该机按照图13A中所示的故障函数转换到状态0(图10中的节点1005),并且在节点0(1005)处再次处理“u”,并且机器停留在节点“0”(1005)处。对于输入流的字符“s”,该机转换到状态3(节点1015),并如下继续:

[0238] 对“h”={4}

[0239] 对“e”={5}

[0240] 对“r”={8}

[0241] 对“s”={9}

[0242] 在状态4(节点1020),因为去往函数g(4,“e”)=5,而该机1000进入状态5,关键字“she”和“he”在文本串“ushers”中位置4的末尾被匹配,并且输出函数发出输出(5)(如在图11B中看到的)。在状态5对于输入符号“r”,该机1000转换到状态8。在状态8对于输入符号“s”,机1000转换到状态9,关键字“hers”被匹配并且输出函数发出输出(9)(如在图11B中看到的)。

[0243] 在另一示例中,参考图10、图11A-B、图12和图13A-B,任意输入串“shell”由处理机处理如下:

[0244] 处理或匹配在开始状态10(由节点1205表示)开始。

[0245] 对输入流的字符“s”,对状态15(图12)的机器转换如下继续:

[0246] 对“h”={16}

[0247] 对“e”={17}

[0248] 对“l”={18}

[0249] 对“l”={19}

[0250] 在状态16(图12)中,因为去往函数g(16,’e’)=17,并且机1200进入状态17,关键字“she”和“he”在文本串“shell”的位置三的末尾被匹配,并且输出函数发出输出(17)(如在图13B中看到的)。在状态17中对于输入符号“l”,机1200转换到状态18。在状态18对于输入符号“l”,机1200转换到状态19,关键字“shell”被作为锚定图样匹配,并且输出函数发出输出(19)(如在图13B中看到的)。

[0251] 如上所述,对DFA或NFA状态机的输入通常是(8位)字节的串,即,字母是单个字节(一个字符或符号)。因此,整个字母集合的大小可以是256。此外,在输入流中的每个字节产生了从一个状态到另一状态的转换。然而,没有多少图样、串或正则表示式使用整个字母集

合。大多数图样使用字母集合的小子集，其在以下可以被简称为“有效字符”集合。例如，可以只有可打印ASCII字母(例如，包括a-z、A-Z、0-9以及一些符号)被使用。

[0252] 本发明实施例压缩NFA和NFA图形以仅识别“有效字符”集合。按照本发明的一个示例实施例，伪代码#1、#2和#3在每个伪代码的步骤1.1期间仅处理“有效字符”集合中的字母。

[0253] 尽管本发明已经参考其示例实施例被具体地示出和描述，但是本领域技术人员可以理解，在不背离由所附权利要求所涵盖的本发明的范围和精神的情况下，可以在其中进行形式和细节上的各种改变。

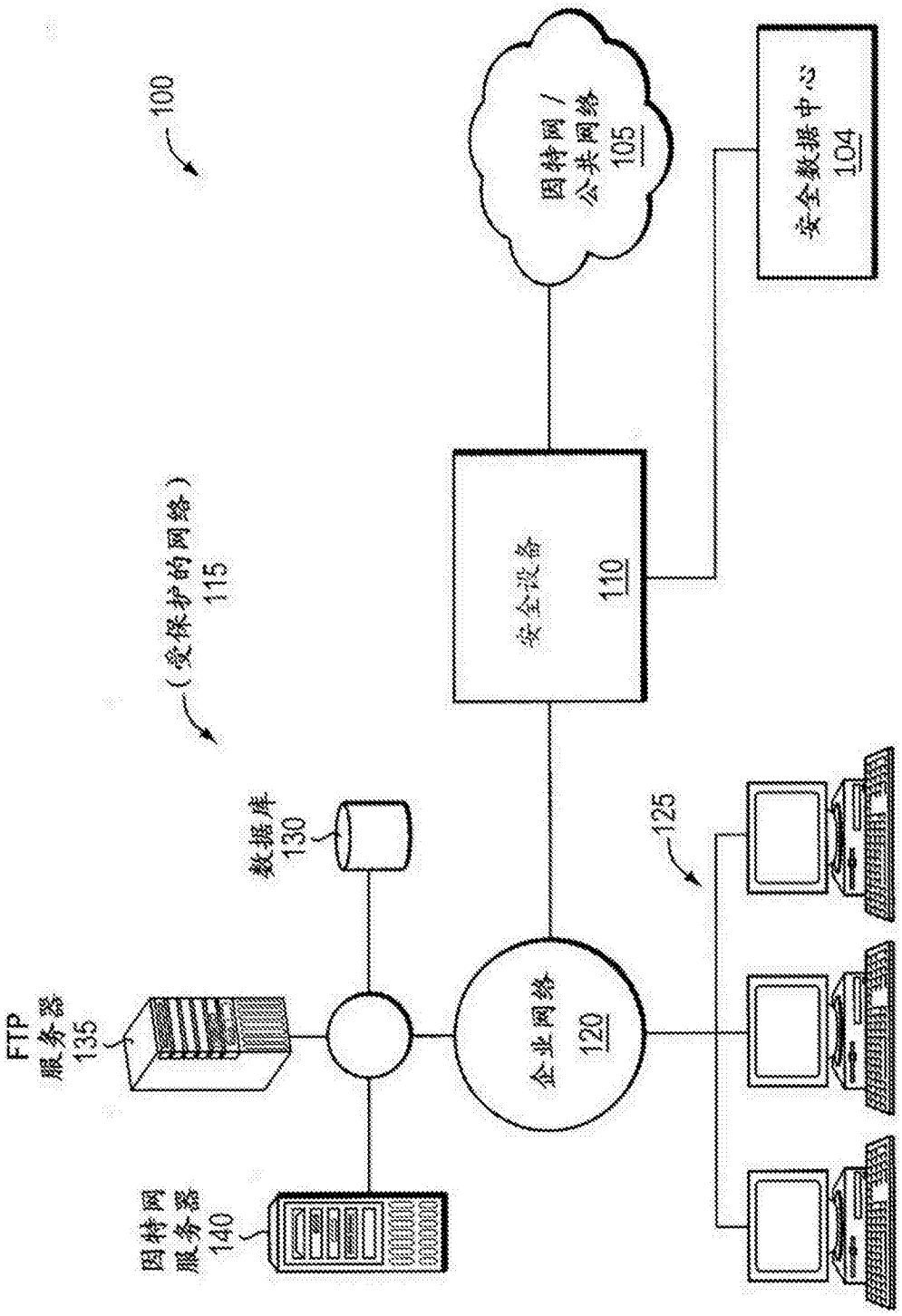

100
因特网 / 公共网络 105
安全数据中心 104
安全设备 110
(受保护的网络) 115
数据库 130
125
FTP 服务器 135
企业网络 120
因特网服务器 140

图1

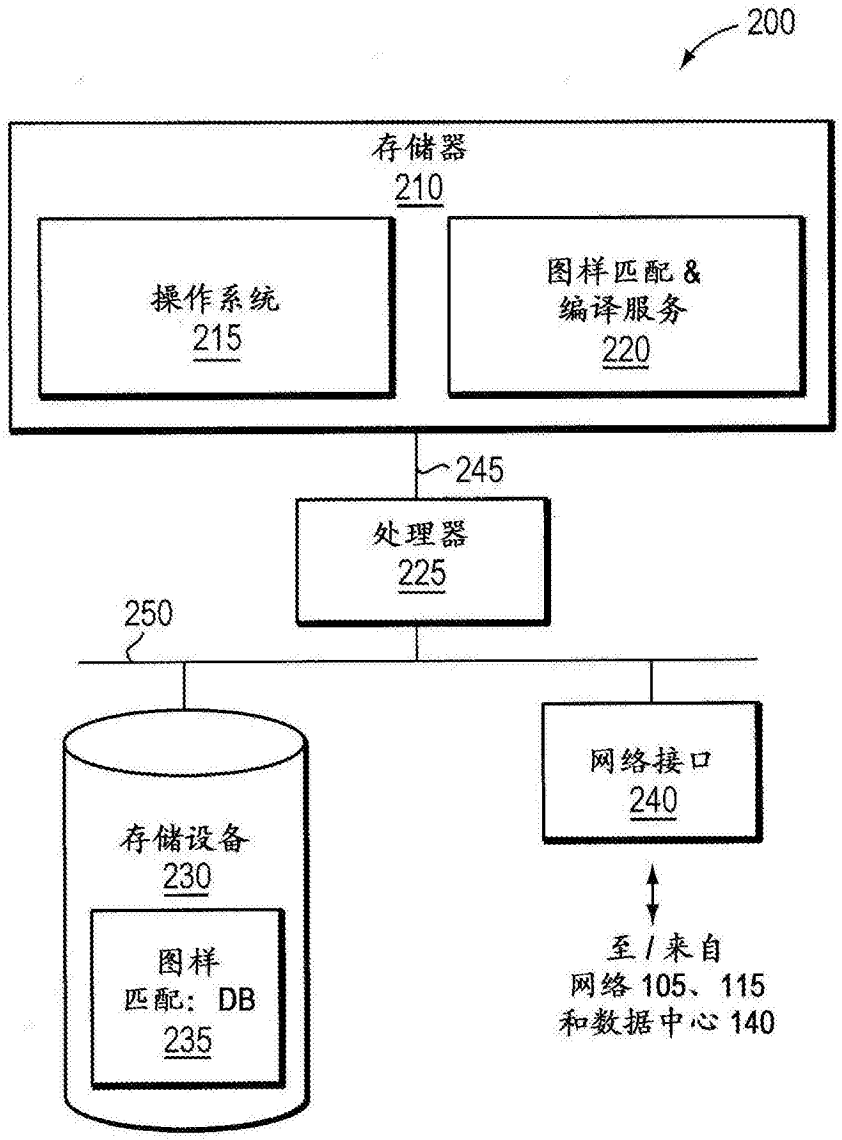

200
存储器
210
操作系统
215
图样匹配 &
编译服务
220
245
处理器
225
250
存储设备
230
图样
匹配：DB
235
网络接口
240
至 / 来自
网络 105、115
和数据中心 140

图2

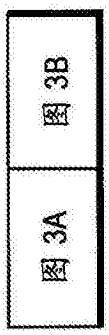


图 3B
图 3A

图3

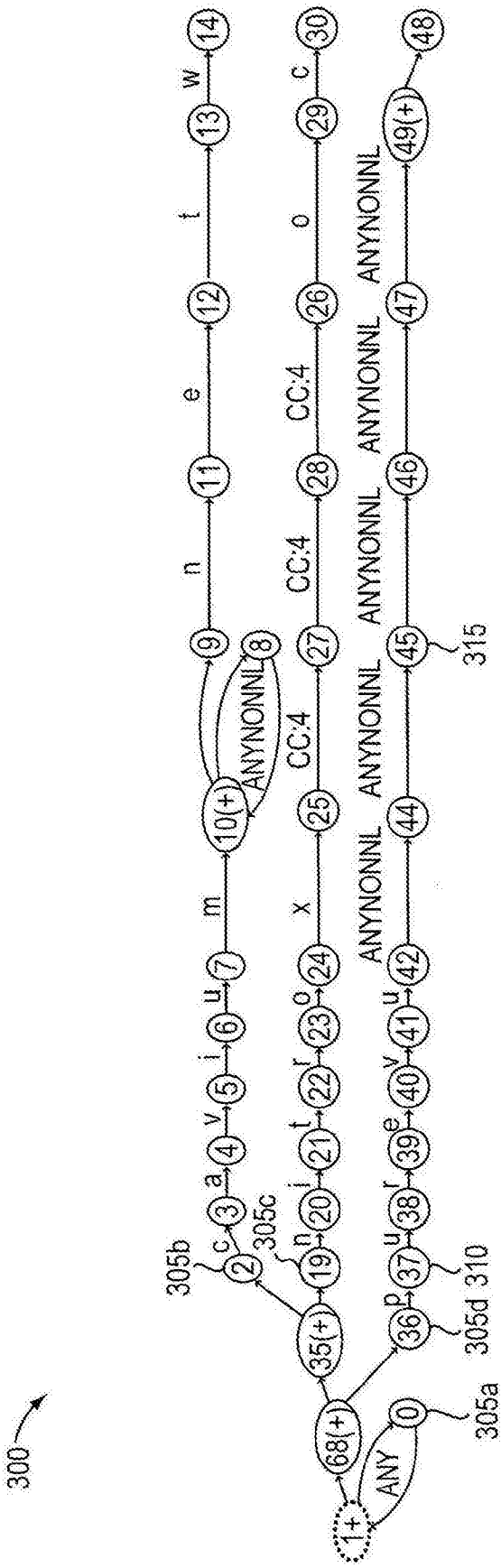


图3A

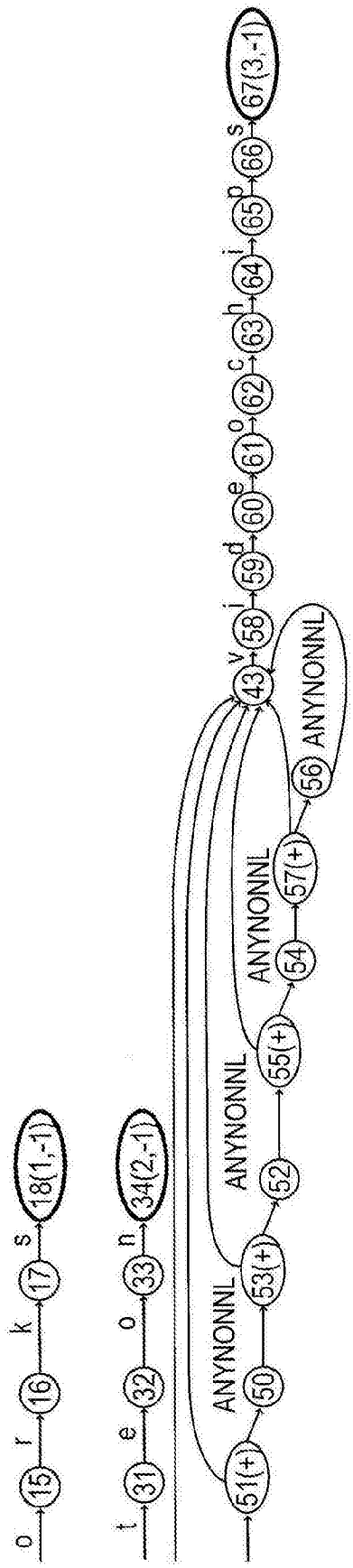


图3B

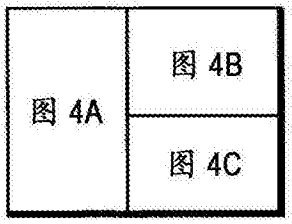


图4

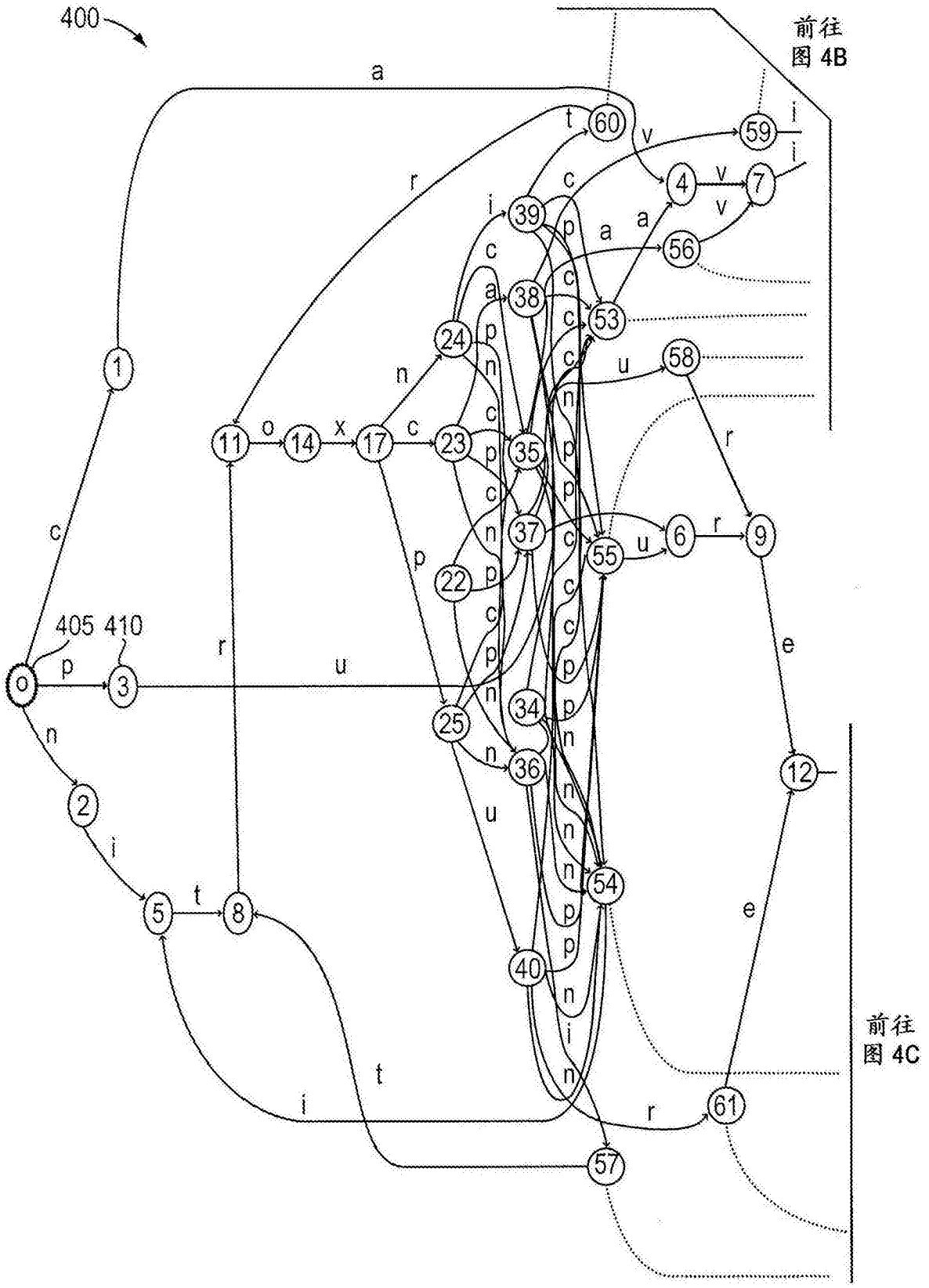


图4A

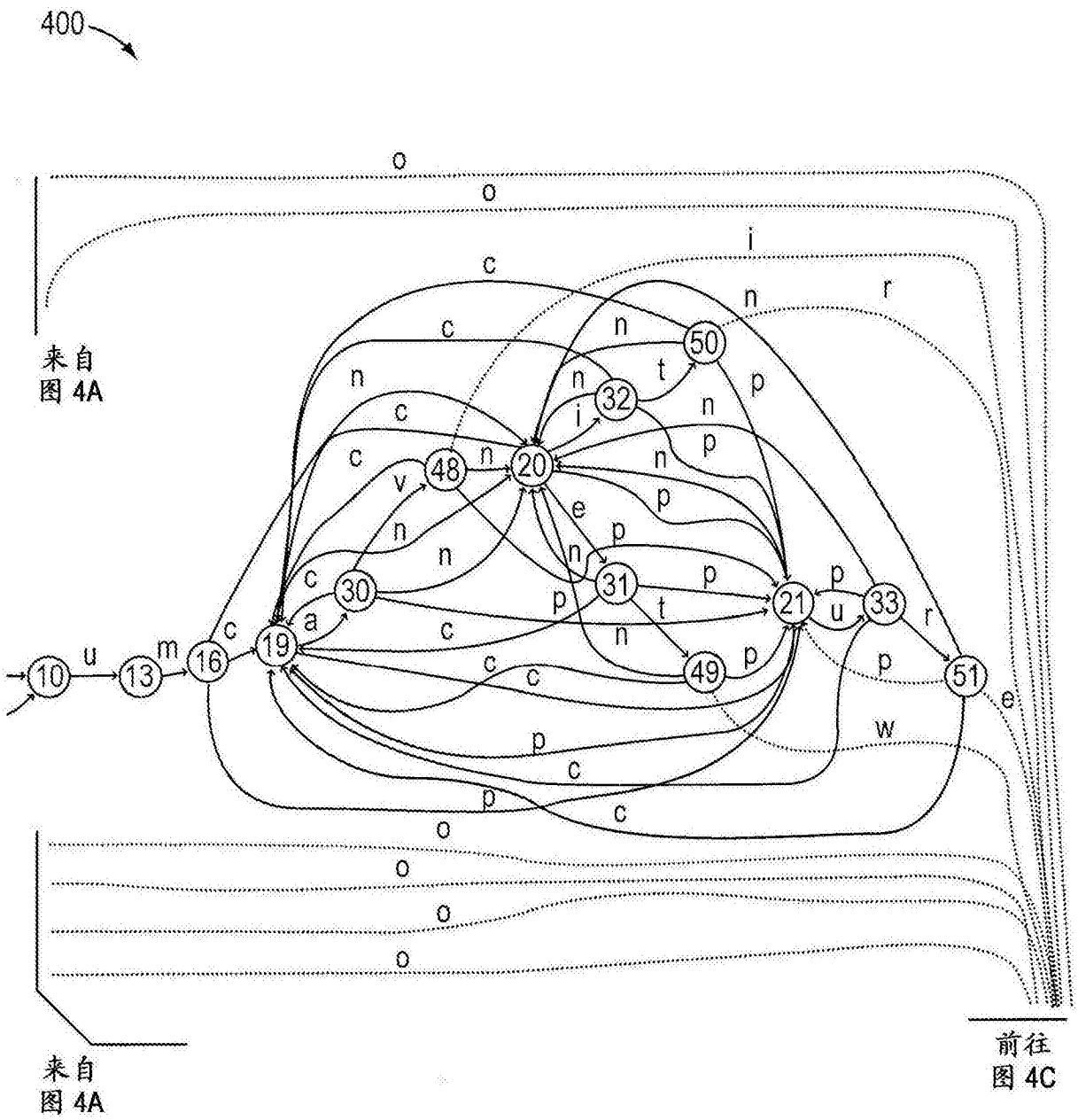


图4B

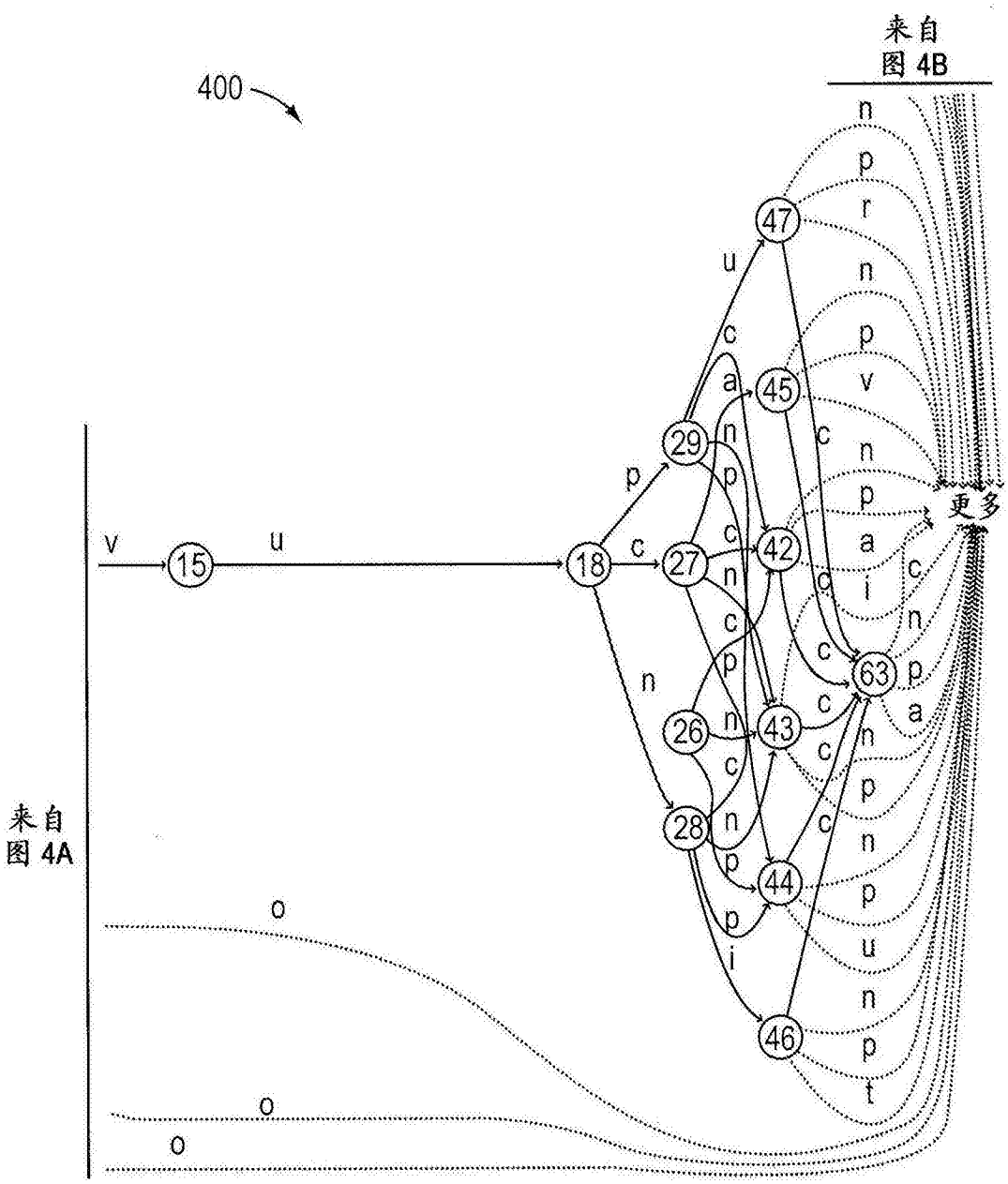

400
来自
图 4B
来自
图 4A
更多
v
u
15
18
c
p
n
o
o
o
29
27
26
28
42
43
44
45
47
46
63

图4C

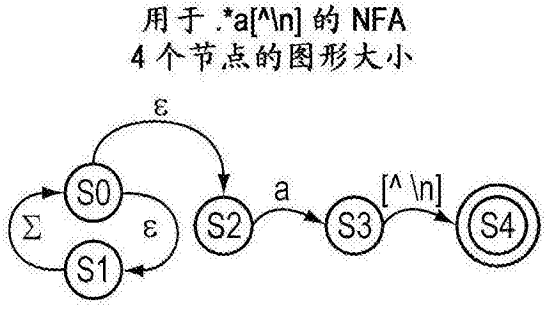

用于 .*a[^\n] 的 NFA
4 个节点的图形大小
ε
S0
Σ
ε
S1
S2
a
S3
[^ \n]
S4

图5A

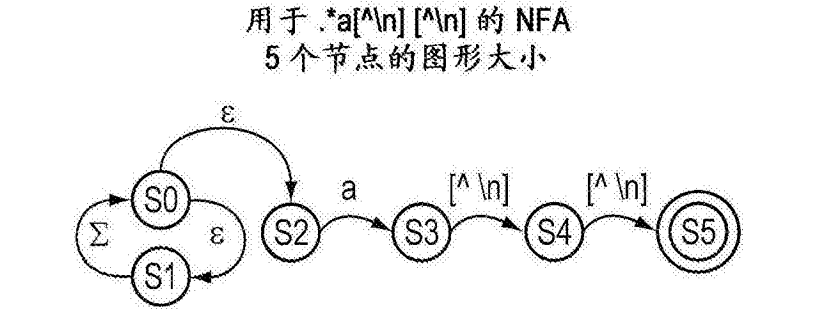

用于 .*a[^\n] [^\n] 的 NFA
5 个节点的图形大小
ε
S0
Σ
ε
S1
S2
a
S3
[^ \n]
S4
[^ \n]
S5

图5B

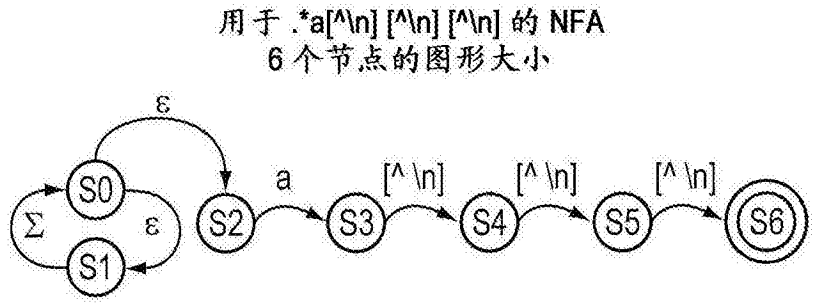

用于 .*a[^\n] [^\n] [^\n] 的 NFA
6 个节点的图形大小
ε
S0
Σ
ε
S1
S2
a
S3
[^ \n]
S4
[^ \n]
S5
[^ \n]
S6

图5C

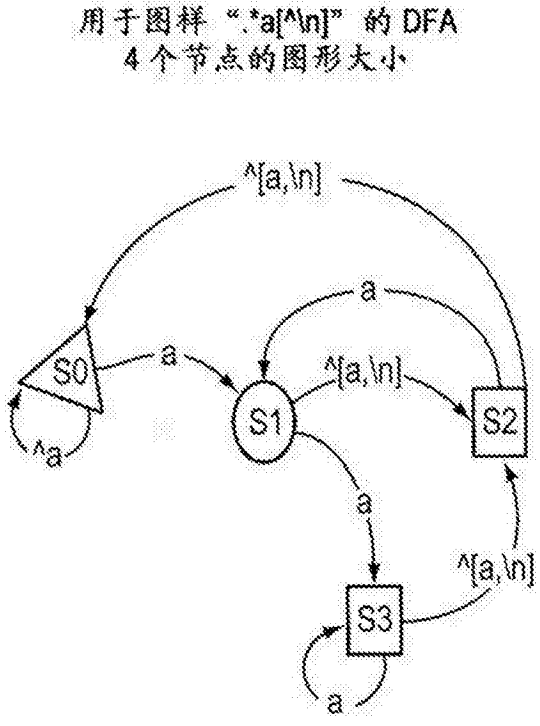

用于图样 ".*a[^\n]" 的 DFA
4 个节点的图形大小
^[a,\n]
a
S0
a
^a
S1
^[a,\n]
S2
a
^[a,\n]
S3
a

图5D

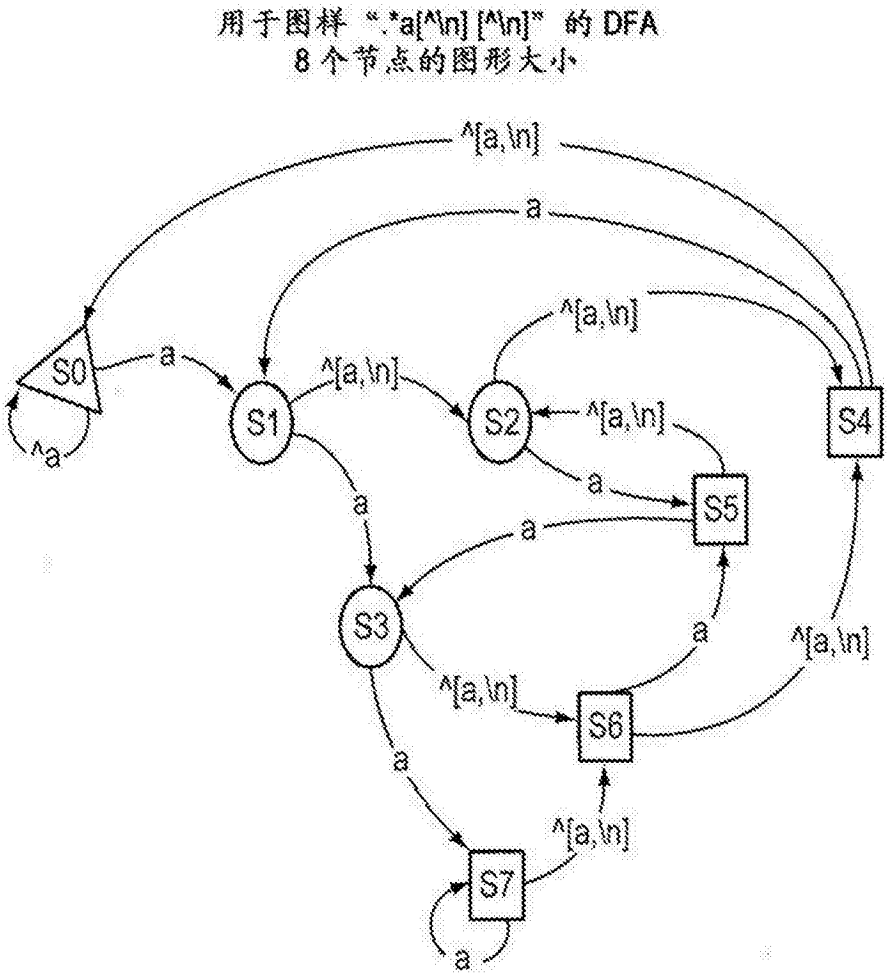

用于图样 ".*a[^\n][^\n]" 的 DFA
8 个节点的图形大小
^[a,\n]
a
S0
a
^a
S1
^[a,\n]
S2
^[a,\n]
^[a,\n]
S4
a
S5
a
a
S3
a
^[a,\n]
^[a,\n]
S6
a
^[a,\n]
S7
a

图5E

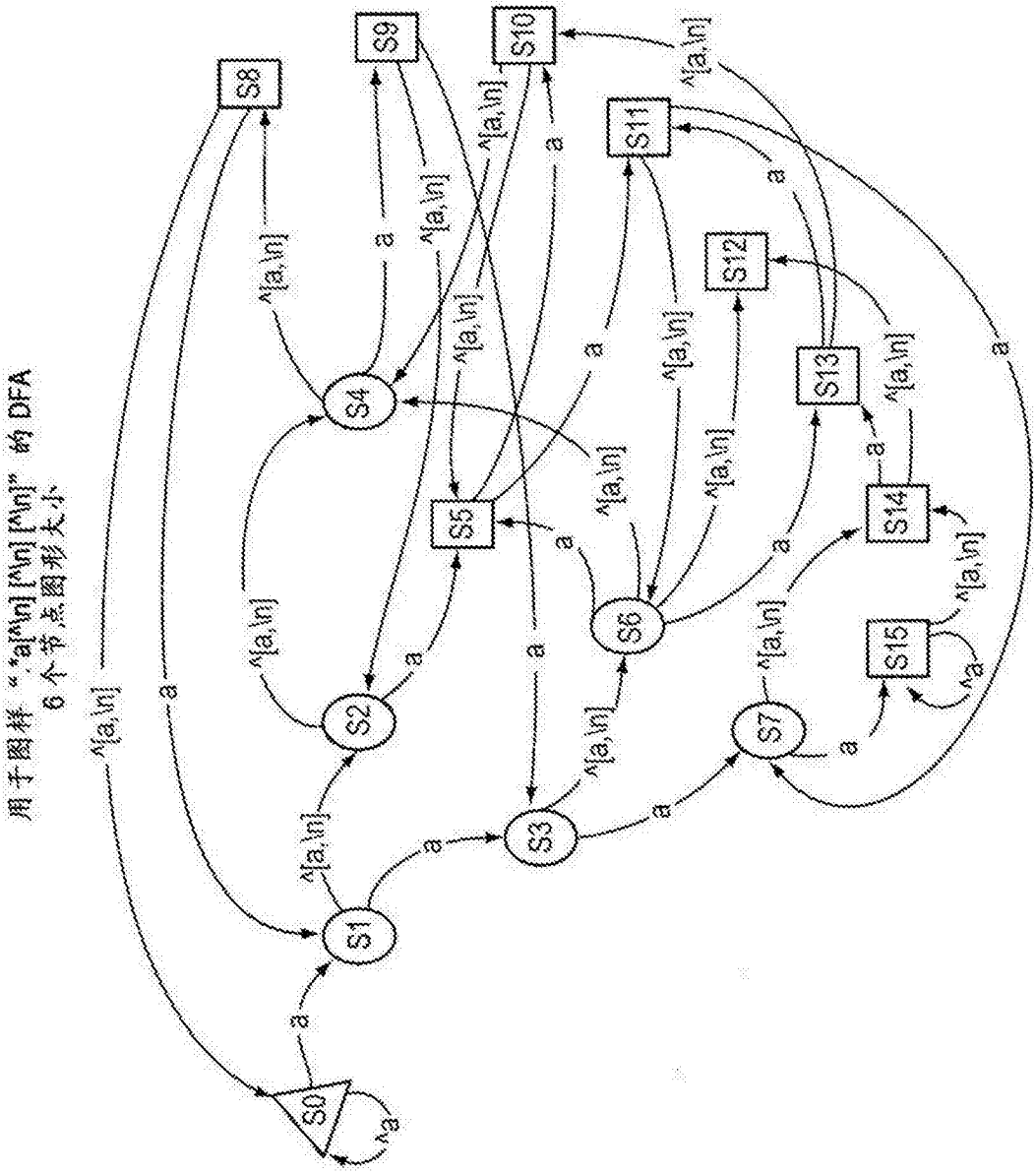

用于图样 “.*a[^\n][^\n]” 的 DFA
6个节点图形大小
S0
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
a
^a
^[a,\n]

图5F

图样	NFA 节点的数目	DFA 节点的数目
.*a[^\n]	4	4
.*a[^\n][^\n]	5	8
.*a[^\n][^\n][^\n]	6	16
.*a[^\n][^\n][^\n][^\n]	7	32
.*a[^\n][^\n][^\n][^\n][^\n]	8	64
•	•	•
•	•	•
•	•	•
$.*a[^\n]_1...[^\n]_n$	n+3	2^n

图5G

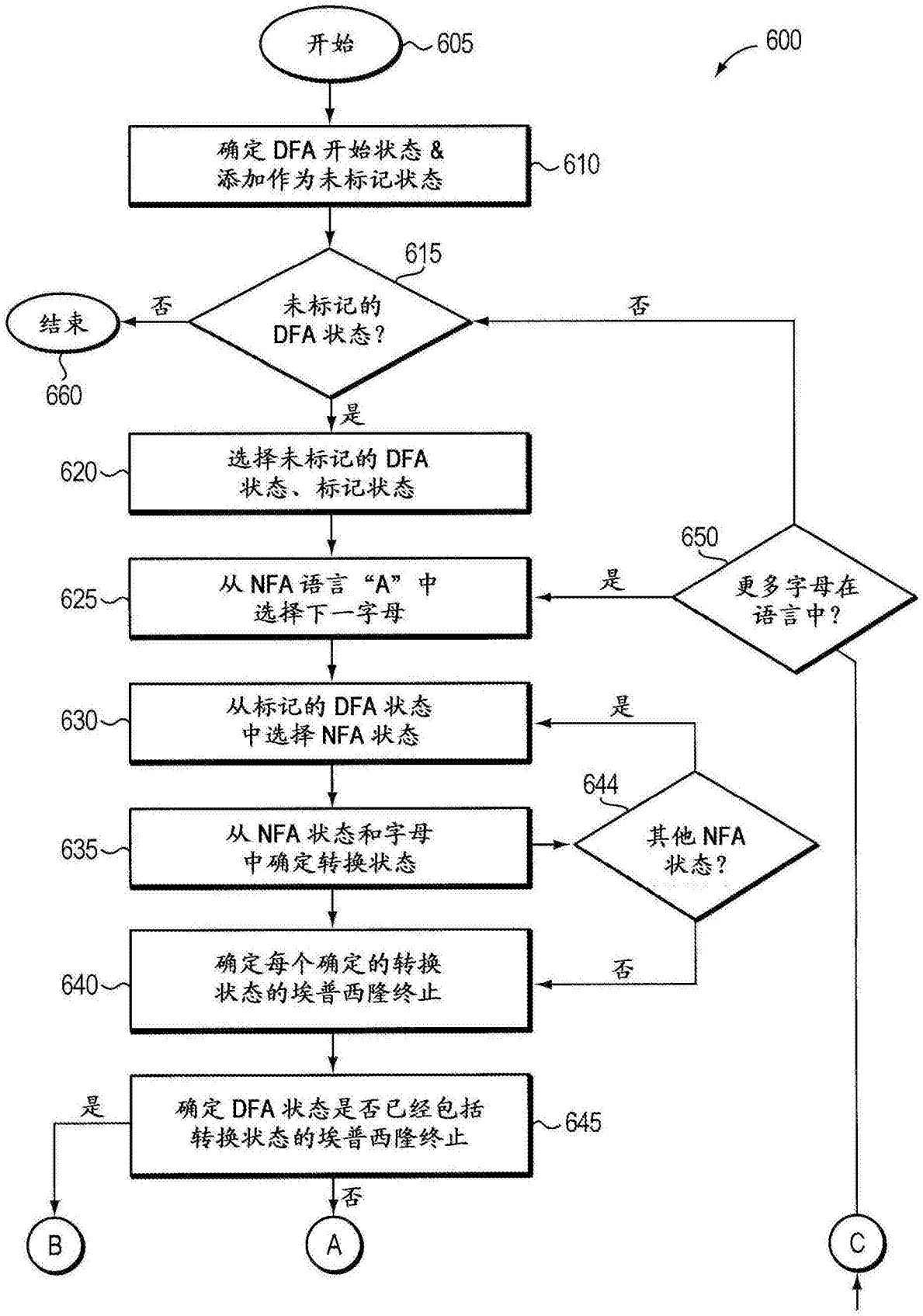

开始
605
600
确定 DFA 开始状态 &
添加作为未标记状态
610
615
未标记的
DFA 状态?
否
结束
660
是
620
选择未标记的 DFA
状态、标记状态
650
更多字母在
语言中?
是
否
625
从 NFA 语言 "A" 中
选择下一字母
630
从标记的 DFA 状态
中选择 NFA 状态
是
644
其他 NFA
状态?
635
从 NFA 状态和字母
中确定转换状态
640
确定每个确定的转换
状态的埃普西隆终止
否
是
确定 DFA 状态是否已经包括
转换状态的埃普西隆终止
645
否
B
A
C

图6A

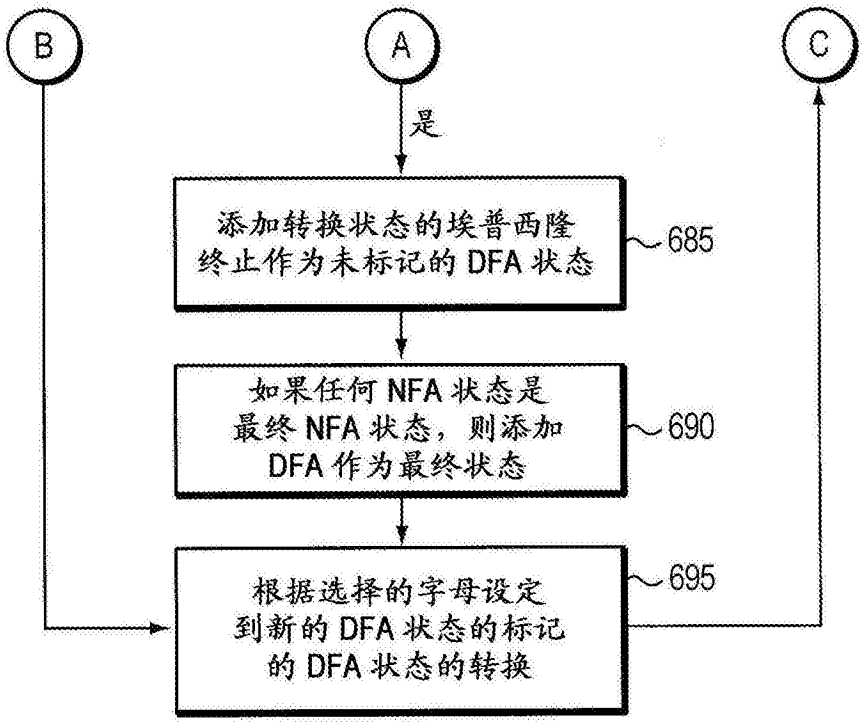

B
A
C
是
添加转换状态的埃普西隆
终止作为未标记的 DFA 状态
685
如果任何 NFA 状态是
最终 NFA 状态，则添加
DFA 作为最终状态
690
根据选择的字母设定
到新的 DFA 状态的标记
的 DFA 状态的转换
695

图6B

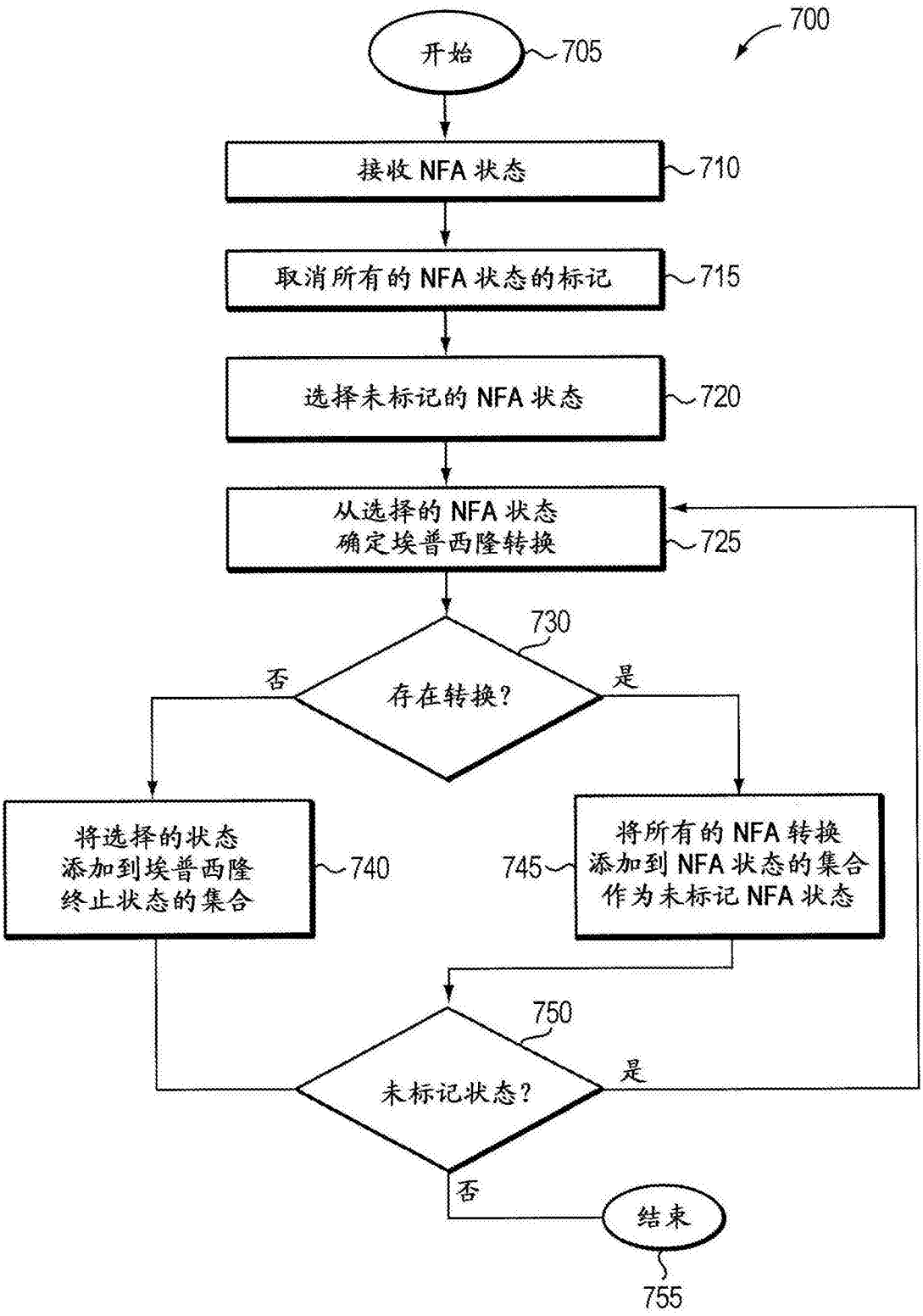

700
开始
705
接收 NFA 状态
710
取消所有的 NFA 状态的标记
715
选择未标记的 NFA 状态
720
从选择的 NFA 状态
确定埃普西隆转换
725
730
存在转换?
否
是
将选择的状态
添加到埃普西隆
终止状态的集合
740
745
将所有的 NFA 转换
添加到 NFA 状态的集合
作为未标记 NFA 状态
750
未标记状态?
是
否
结束
755

图7

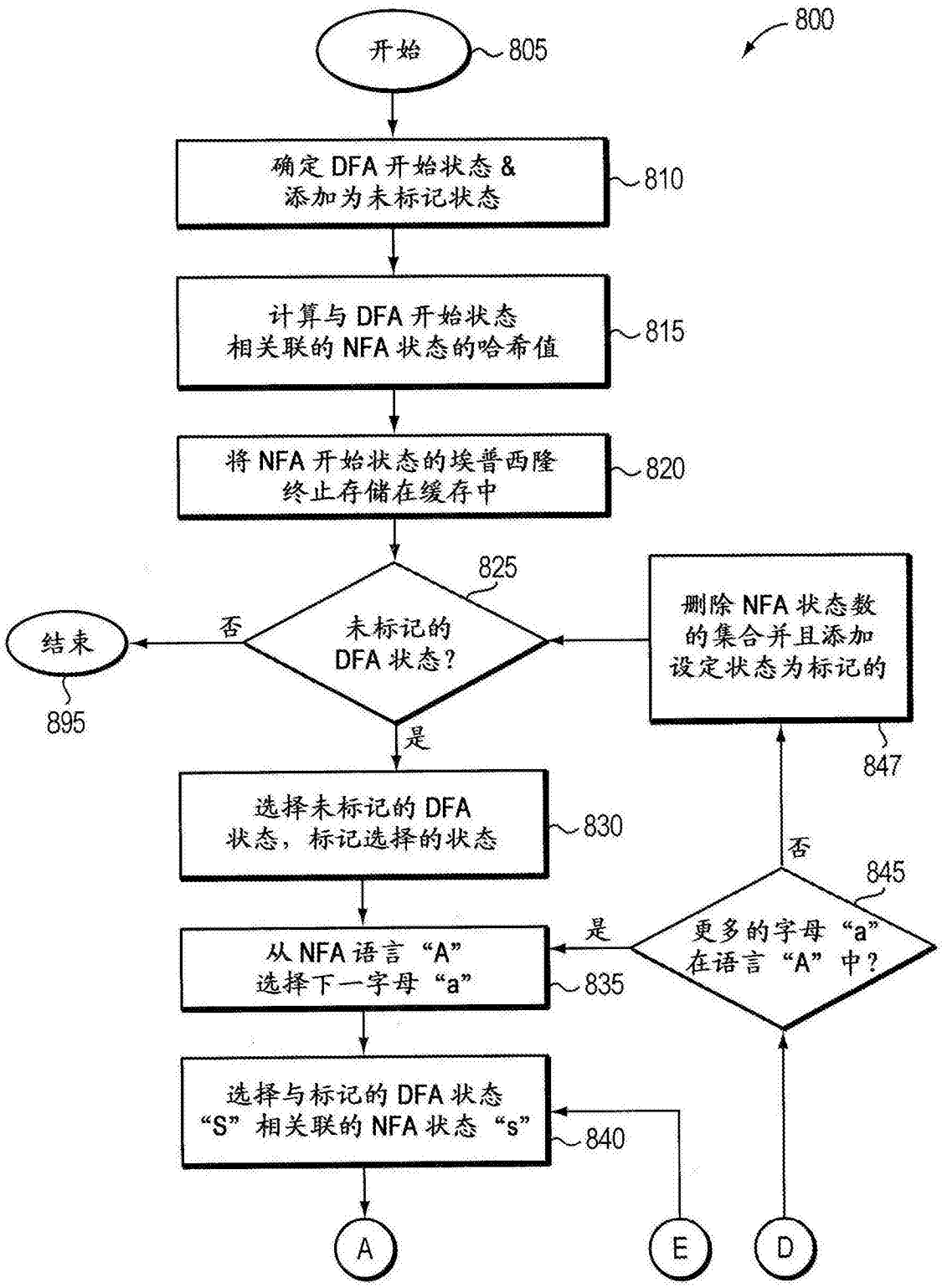

800
开始
805
确定 DFA 开始状态 &
添加为未标记状态
810
计算与 DFA 开始状态
相关联的 NFA 状态的哈希值
815
将 NFA 开始状态的埃普西隆
终止存储在缓存中
820
825
未标记的
DFA 状态?
否
结束
895
是
删除 NFA 状态数
的集合并且添加
设定状态为标记的
847
选择未标记的 DFA
状态，标记选择的状态
830
否
845
更多的字母 “a”
在语言 “A” 中?
是
从 NFA 语言 “A”
选择下一字母 “a”
835
选择与标记的 DFA 状态
“S” 相关联的 NFA 状态 “s”
840
A
E
D

图8A

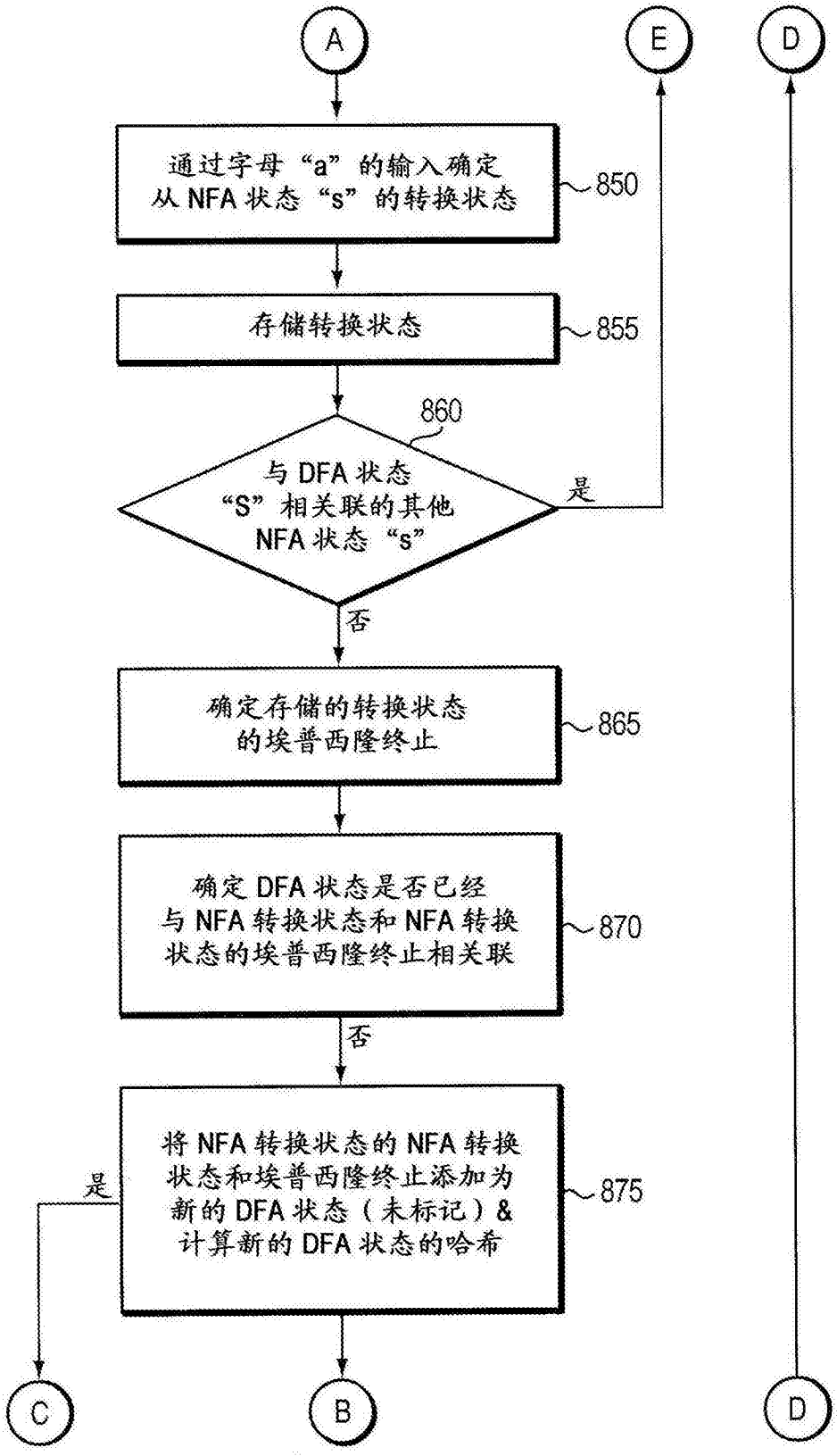

A
E
D
通过字母“a”的输入确定
从NFA状态“s”的转换状态
850
存储转换状态
855
860
与DFA状态
“S”相关联的其他
NFA状态“s”
是
否
确定存储的转换状态
的埃普西隆终止
865
确定DFA状态是否已经
与NFA转换状态和NFA转换
状态的埃普西隆终止相关联
870
否
将NFA转换状态的NFA转换
状态和埃普西隆终止添加为
新的DFA状态（未标记）&
计算新的DFA状态的哈希
875
是
C
B
D

图8B

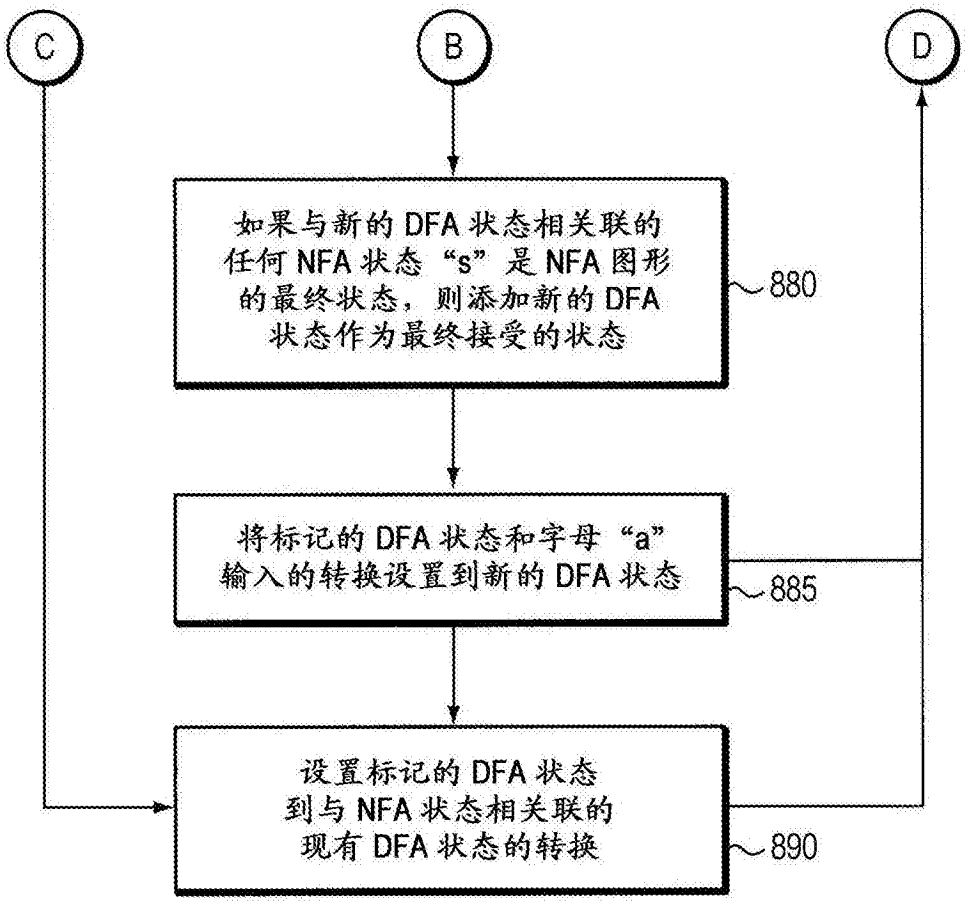

C
B
D
如果与新的 DFA 状态相关联的任何 NFA 状态“s”是 NFA 图形的最终状态，则添加新的 DFA 状态作为最终接受的状态
880
将标记的 DFA 状态和字母“a”输入的转换设置到新的 DFA 状态
885
设置标记的 DFA 状态到与 NFA 状态相关联的现有 DFA 状态的转换
890

图8C

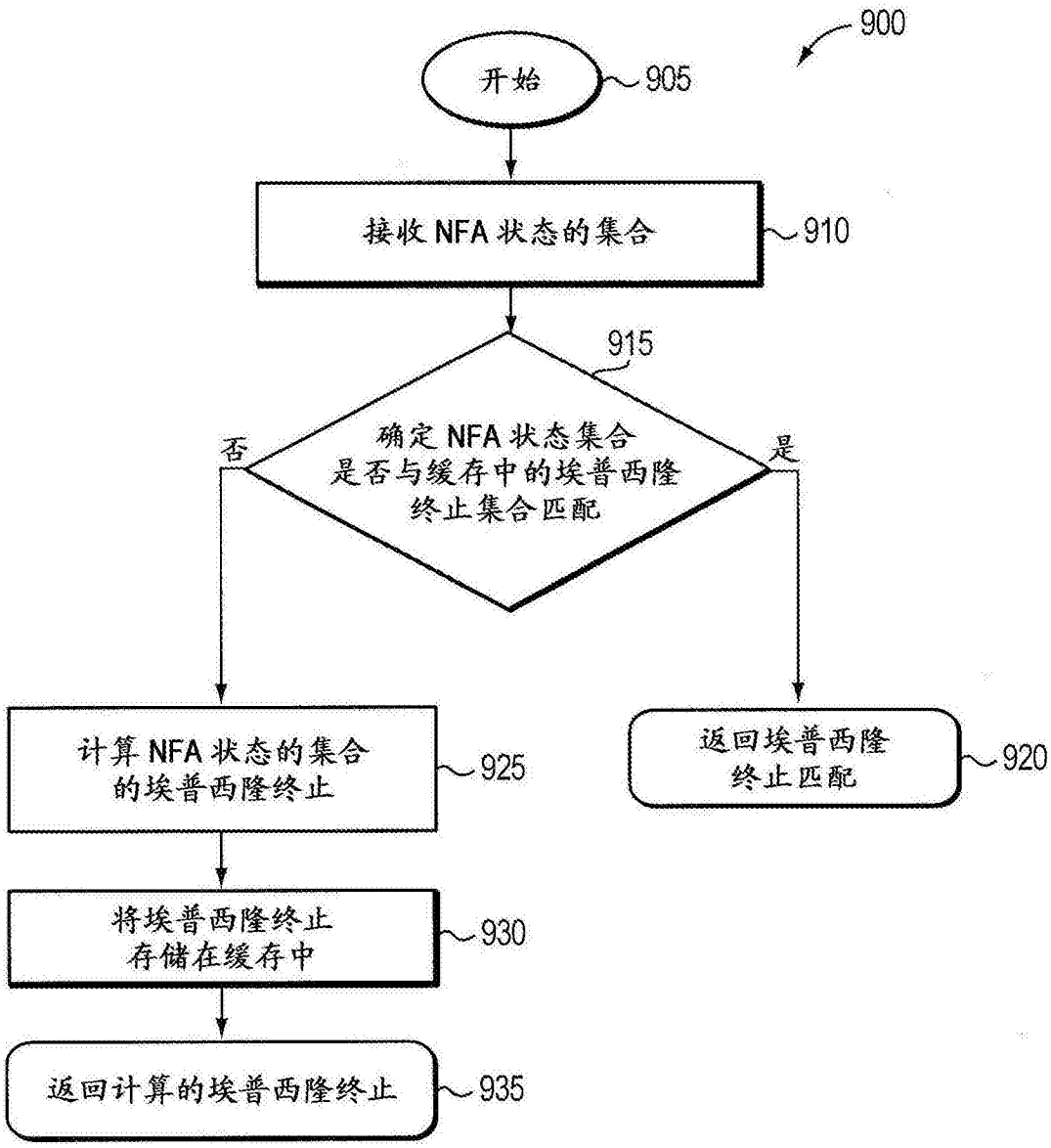

900
开始
905
接收 NFA 状态的集合
910
915
确定 NFA 状态集合
是否与缓存中的埃普西隆
终止集合匹配
否
是
计算 NFA 状态的集合
的埃普西隆终止
925
返回埃普西隆
终止匹配
920
将埃普西隆终止
存储在缓存中
930
返回计算的埃普西隆终止
935

图9

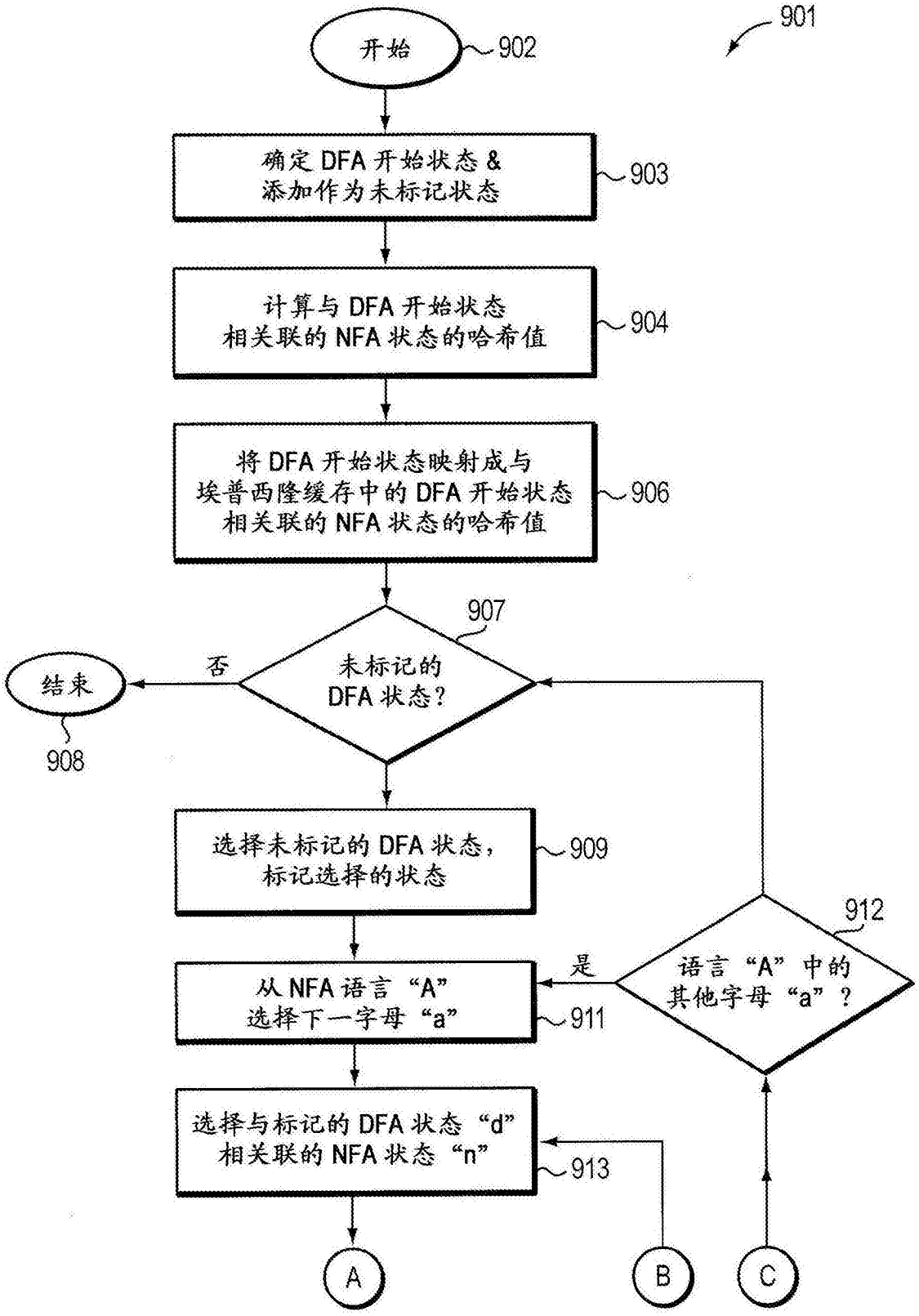

901
开始
902
确定 DFA 开始状态 &
添加作为未标记状态
903
计算与 DFA 开始状态
相关联的 NFA 状态的哈希值
904
将 DFA 开始状态映射成与
埃普西隆缓存中的 DFA 开始状态
相关联的 NFA 状态的哈希值
906
907
未标记的
DFA 状态?
否
结束
908
选择未标记的 DFA 状态,
标记选择的状态
909
912
语言 “A” 中的
其他字母 “a” ?
是
从 NFA 语言 “A”
选择下一字母 “a”
911
选择与标记的 DFA 状态 “d”
相关联的 NFA 状态 “n”
913
A
B
C

图9A

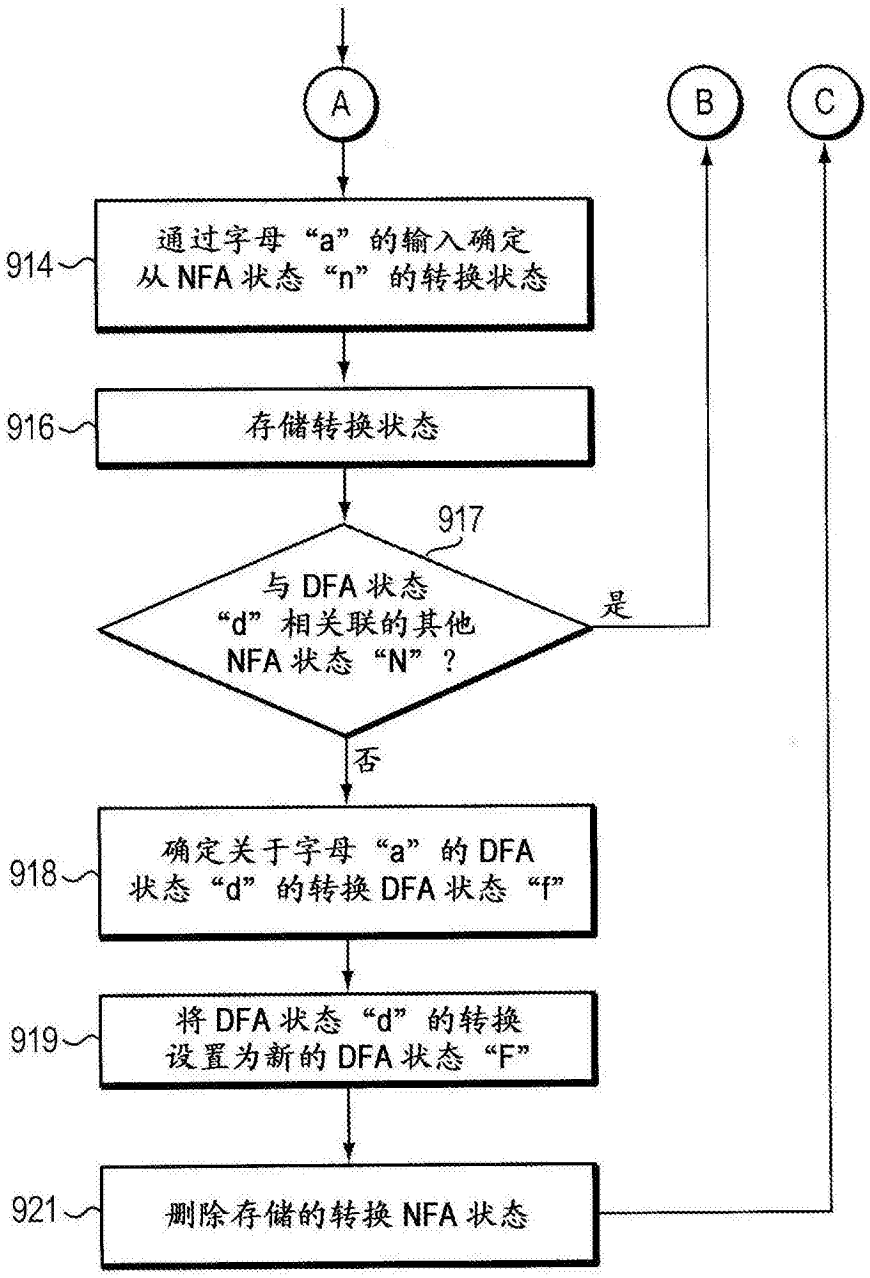

A
B
C
914
通过字母“a”的输入确定
从NFA状态“n”的转换状态
916
存储转换状态
917
与DFA状态
“d”相关联的其他
NFA状态“N”？
是
否
918
确定关于字母“a”的DFA
状态“d”的转换DFA状态“f”
919
将DFA状态“d”的转换
设置为新的DFA状态“F”
921
删除存储的转换NFA状态

图9B

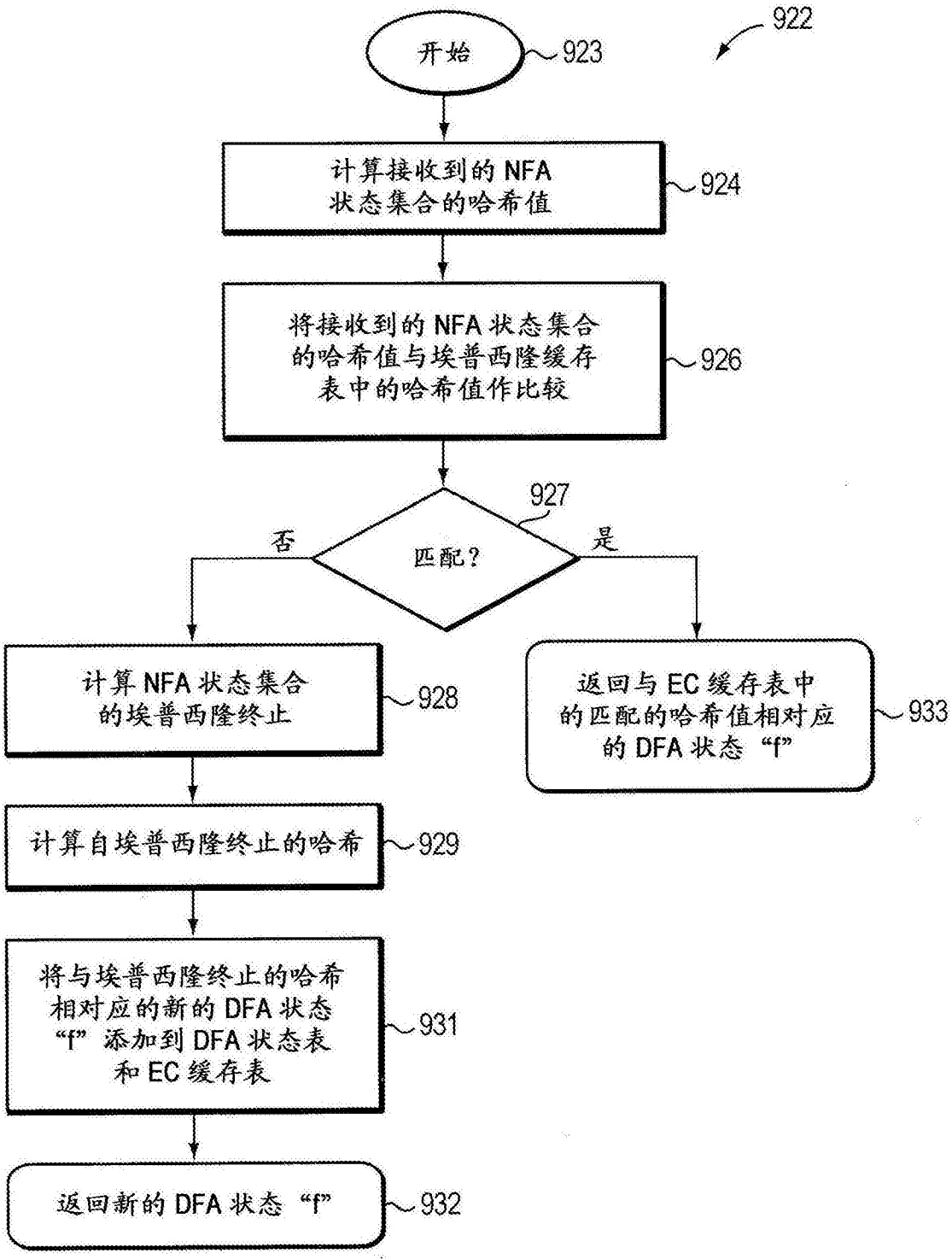

922
开始
923
计算接收到的 NFA 状态集合的哈希值
924
将接收到的 NFA 状态集合的哈希值与埃普西隆缓存表中的哈希值作比较
926
927
匹配？
否
是
计算 NFA 状态集合的埃普西隆终止
928
返回与 EC 缓存表中的匹配的哈希值相对应的 DFA 状态 “f”
933
计算自埃普西隆终止的哈希
929
将与埃普西隆终止的哈希相对应的新的 DFA 状态 “f” 添加到 DFA 状态表和 EC 缓存表
931
返回新的 DFA 状态 “f”
932

图9C

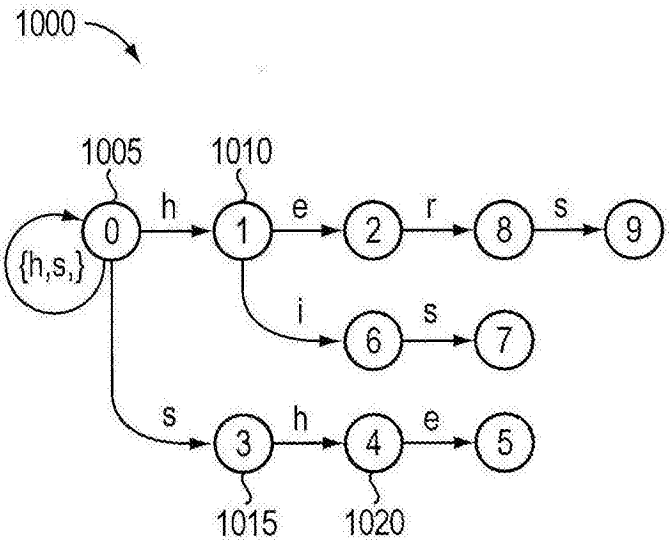


图10

i	1	2	3	4	5	6	7	8	9
f(i)	0	0	0	1	2	0	3	0	3

故障函数

图11A

i	output (i)
2	{he}
5	{she, he}
7	{his}
9	{hers}

输出函数

图11B

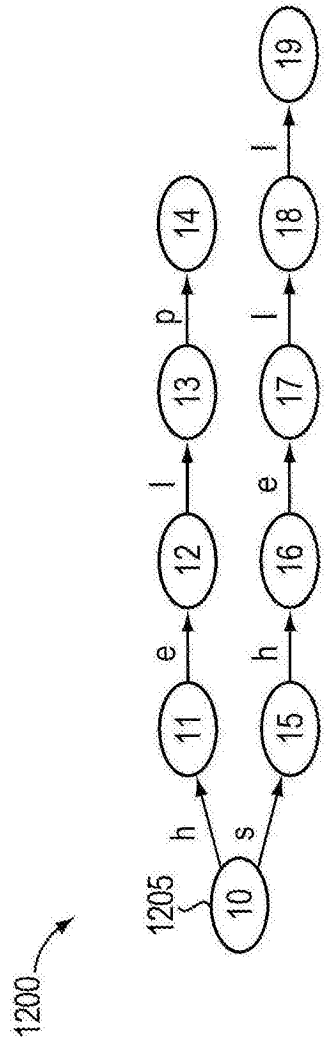


图12

i	10	11	12	13	14	15	16	17	18	19
f(i)	0	0	0	0	0	0	1	2	3	0

故障函数

图12A

i	output (i)
14	{help}
19	{shell}

输出函数

图12B

i	10	11	12	13	14	15	16	17	18	19
f(i)	0	1	2	0	0	3	4	5	0	0

故障函数

图13A

i	output (i)
12	{he}
14	{ {@0} help}
17	{he, she}
19	{ {@0} shell }

输出函数

图13B