基于属性的相对约简格快速渐进式构造算法*)

曲立平　刘大昕　杨　静
(哈尔滨工程大学计算机科学与技术学院　哈尔滨 150001)

摘　要　相对约简格作为简化的概念格，在数据挖掘和知识发现等领域具有广泛的应用。相对约简格的构造在其应用过程中是一个主要问题。本文提出了采用树结构对相对约简格节点进行组织，研究了基于属性的相对约简格渐进式构造算法。相对约简格节点的树结构组织可以约束更新格节点、产生子格节点及新生格节点的子结点的搜索范围，从而可以有效地减少算法的执行时间。该算法不仅为相对约简格的构造提供了一种方法，还解决了在已构造好相对约简格的前提下，增加属性所带来的更新问题。在随机生成的数据集上进行的实验测试表明，本算法的时间性能更优越。

关键词　形式概念分析，相对约简格，渐进式算法，概念树

Attribute-based Fast Incremental Algorithm for Building Relative Reduced Concept Lattice

QU Li-Ping　LIU Da-Xin　YANG Jing
(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001)

Abstract　Relative reduced concept lattice, a simplified concept lattice, can be used widely in data mining and knowledge discovery, etc. The main difficulty with relative reduced concept lattice-based system comes from the lattice construction itself. In this paper, tree structure is employed to organize the set of concepts in relative reduced concept lattice. Based on attribute, a fast incremental algorithm is developed. The organization of concepts in tree structure can reduce the search space of update concept node, generator concept node and children of new born concept node, and consequently improve the speed of the algorithm. It provides an approach for building relative reduced concept lattice and resolves the problem of lattice update caused by appending new attributes into an existing context of the lattice. The algorithm is experimental evaluated and compared for random generated data. The results show that the algorithm performance is superior.

Keywords　Formal concept analysis, Relative reduced concept lattice, Incremental algorithm, Concept tree

1　引言

形式概念分析是由德国的 Wille 教授于 20 世纪 80 年代初提出的[1]。随着研究的深入，形式概念分析越来越多地被应用到数据挖掘、信息检索、软件工程等领域，成为处理和组织大规模数据的有效工具。概念格的构造是形式概念分析应用的前提。由于概念格自身的完备性，构造概念格的主要困难在于概念格中概念数目是形式背景大小的指数级函数，随着处理的数据量的剧增，庞大的格节点构造及其存储成为制约和影响算法效率的因素之一。研究采用新的方法和手段来构造概念格，成为概念格研究的主要内容之一。许多学者对此进行了广泛研究，提出了各种不同的构造算法(如 Bordat 算法、Ganter 算法、Norris 算法、Chein 算法[2]、Godin 算法[3]、Capineto 算法)。国内很多学者也提出了一些改进算法[4~7]。实验结果表明，与 Bordat 算法、Ganter 算法、Chein 算法比较，Godin 算法是最快的[2]。但以 Godin 算法为代表的渐进式算法在搜索产生子格节点和新生格节点的直接子节点时需要遍历所有节点，算法的时间性能可以进一步提高。同时，作为简化的概念格，相对约简格的外延存储空间可以进一步降低。本文研究采用树结构组织相对约简格节点，提出了基于属性的相对约简格快速渐进式构造算法。该算法通过缩小产生子格节点和新生格节点的直接子节点的搜索范围，达到加速相对约简格构造的作用。实验结果表明，本文提出的算法要快于著名的 Godin 算法和现有的相对约简格构造算法。

2　相对约简格的基本概念

定义 1[1]　形式背景定义为一个三元组 $K=(G,M,I)$，其中 G 为对象集，M 为属性集，I 是 G 与 M 之间的二元关系。

表 1 为一个简单的形式背景 $K=(G,M,I)$，$G=\{1,2,3,4,5\}$，$M=\{a,b,c,d,e,f,g,h,i\}$，×表示 $(g,m)\in I$。

表 1　形式背景 K

I	a	b	c	d	e	f	g	h	i
1	×		×			×		×	
2	×		×				×		×
3	×			×			×		×
4		×	×			×		×	
5		×			×		×		

对于 $A\subseteq G$，定义 $A'=\{m \mid m\in M, \forall g\in A, gIm\}$。对于 $B\subseteq M$，定义 $B'=\{g \mid g\in G, \forall m\in B, gIm\}$。

定义 2[1]　设 (G,M,I) 为形式背景，如果一个二元组 $C=(A,B)$ 满足 $A'=B$ 且 $B'=A$，称 C 是一个概念。其中，A

*)国家自然科学基金资助项目(60673131)、黑龙江省自然科学基金项目(F-0304)资助。曲立平　副教授，博士研究生，主要研究方向为数据挖掘、软件工程；刘大昕　教授，博士生导师，主要研究方向为数据库与知识库；杨　静　教授，博士生导师，主要研究方向为数据挖掘、数据库。

称为概念的外延，记为 $extent(C)$，B 称为概念的内涵，记为 $intent(C)$。

定义 3[1] $C_1=(A_1,B_1)$ 和 $C_2=(A_2,B_2)$ 是形式背景 (G,M,I) 上的任意两个概念，定义二元关系 $\leqslant$：

$$C_2\leqslant C_1\Leftrightarrow B_1\subseteq B_2\Leftrightarrow A_2\subseteq A_1$$

称 C_1 是 C_2 的父概念，C_2 是 C_1 的子概念。如果不存在另外一个概念 C_3，使得 $C_2\leqslant C_3\leqslant C_1$，称 C_1 是 C_2 的直接父概念，C_2 是 C_1 的直接子概念，记为 $C_2<C_1$。

定义 4[1] 形式背景 K 中所有概念和偏序关系 $\leqslant$ 构成了一个完全格，称为概念格，记为 $L(K)$。

定义 5[7] 不能由父概念的内涵确定的内涵称为初始内涵。不能由子概念的外延确定的外延称为初始外延。

定义 6 概念格 $L(K)$ 中，每个概念的外延只保留初始外延，内涵只保留初始内涵，这样形成的概念格称为相对约简格，记为 $RE(K)$。

相对约简格的概念反映了较其父概念的内涵增量和较其子概念的外延增量，但不会出现信息丢失。图 1 给出了表 1 形式背景所对应的相对约简格。

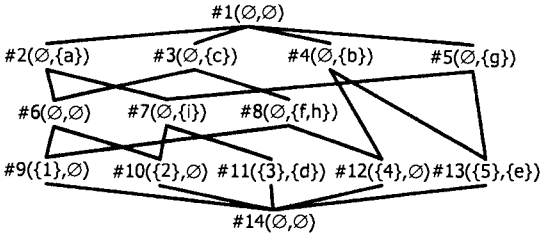

图 1 形式背景 K 对应的相对约简格

3 基于属性的相对约简格渐进式构造算法的基本思想

基于属性的相对约简格渐进式构造就是在给定原始形式背景 $K=(G,M,I)$ 所对应的初始相对约简格 $RE(K)$ 以及新增属性 m^* 的情况下，求解形式背景 $K^*=(G,M\cup\{m^*\},I\cup I^*)$ 所对应的相对约简格 $RE(K^*)$，其中，$I^*\subseteq G\times\{m^*\}$。对于 $RE(K)$ 中的每个格节点，根据它和新增属性 m^* 的对象集 $\{m^*\}'$ 之间的关系，可以定义它们的不同类型。对于 $RE(K)$ 中的概念 C，设其在形式背景 K 下的真正外延和真正内涵分别用 $a\_extent(C)$ 和 $a\_intent(C)$ 表示。记 $C_m=(A_m,B_m)$，其中 $A_m=\{m^*\}'$，$B_m=\{m^*\}$。

定义 7 如果 $RE(K)$ 中某个格节点 C 满足 $a\_extent(C)\subseteq\{m^*\}'$，称 C 为更新格节点。

对于更新格节点 C，执行 $A_m=A_m\cap extent(C)$ 操作。如果 C 满足 $a\_extent(C)=\{m^*\}'$，在 $RE(K^*)$ 中 C 被更新为 $(extent(C),intent(C)\cup\{m^*\})$。

定义 8 如果 $RE(K)$ 中某个格节点 C 满足：(1)在 $RE(K)$ 中不存在某个格节点 C_1 满足 $a\_extent(C_1)=a\_extent(C)\cap\{m^*\}'$；(2)对于 $RE(K)$ 中任意满足 $C_2\leqslant C$ 的节点 C_2，都有 $a\_extent(C_2)\cap\{m^*\}'\neq a\_extent(C)\cap\{m^*\}'$，称 C 为产生子格节点。

定义 9 如果 $RE(K)$ 中某个格节点 C 既不是更新格节点，也不是产生子格节点，称 C 为不变格节点。

定义 10 如果 $RE(K^*)$ 中某个格节点 C 满足 $\neg\exists C_1\in RE(K)(a\_extent(C)=a\_extent(C_1))$，称 C 为新生格节点。

对于产生子格节点 C，令 $A_\cap=extent(C)\cap A_m$，执行 $extent(C)=extent(C)\rightarrow A_\cap$ 和 $A_m=A_m-A_\cap$ 操作。产生子格节点 C 与新生格节点 C_{new} 一一对应。如果 $\{m^*\}'\subseteq a\_extent(C)$，则 $C_{new}=(A_\cap,B_m)$，否则 $C_{new}=(A_\cap,\varnothing)$。

显然，求解 $RE(K^*)$ 的关键是找出 $RE(K)$ 中所有更新格节点、产生子格节点和生成 $RE(K^*)$ 中所有新生格节点，以及维护新生格节点与其直接父节点和直接子节点之间的边的关系。

4 基于属性的相对约简格快速渐进式构造算法

4.1 相对约简格节点的树结构组织

参考文[6]中索引树组织方式，本文用概念树对相对约简格节点的集合进行组织。在概念树中，每条边表示一个对象，从根节点到概念的路径上对象的集合为该概念在形式背景下的真正外延。对于表 1 的形式背景 K，假定 G 上的全排序为 $1<2<3<4<5$，则图 1 相对约简格中所有概念对应的概念树如图 2 所示。

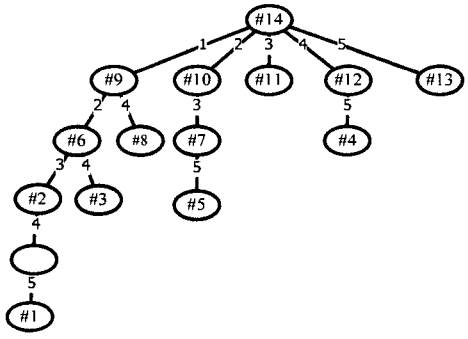

图 2 形式背景 K 对应的概念树

定义 11[6] 对于概念树节点 T_1 和 T_2，如果 T_1 是 T_2 的父节点，且 T_1 到 T_2 的边是 g，称 T_2 是 T_1 的第 g 个子节点，记 $T_2=T_1.child[g]$。

定义 12[6] 对于概念树节点 T，如果存在某个相对约简格节点 C 满足 $a\_extent(C)=\varphi(T)$，称 T 为有效树节点，记 $T.lattice=C$。否则，称 T 为无效树节点，记 $T.lattice=$ NULL。其中，$\varphi(T)$ 为根节点 $root$ 到 T 的路径上边的集合。

为了建立组织相对约简格中格节点的概念树，首先将概念树初始化为只包含一个有效树节点的树，该有效树节点对应真正外延为全部对象，而真正内涵为空的相对约简格节点，然后将概念格中格节点相应的概念树节点使用 *SearchForInsert* 逐个地插入到概念树中。函数如下：

```
FUNCTION SearchForInsert(root, a_extent(C))
BEGIN
  T = root;
  FOR g∈a_extent(C)按照升序排列 DO
    IF (T.child[g] == NULL) THEN
      创建一个新的概念树节点 New,并置它的所有子节点为
      NULL;
      New.lattice = NULL;
      T.child[g] = New;
    ENDIF
    Node= Node.child[g];
  ENDFOR
  return T;
END
```

按层遍历概念树可以定义概念树节点的访问顺序，这是一个全序关序，记为 $<_L$。$T_1<_L T_2$ 表示 T_1 在 T_2 之前被访问。

定理 1 相对约简格节点 C_1、C_2 对应的概念树节点分别为 T_1、T_2，若 $a\_extent(C_1)\subset a\_extent(C_2)$，则 $T_1<_L T_2$ 和 C_1

$<_L C_2$。

证明：略。

4.2 算法原理

算法的主要思想是按$<_L$对$RE(K)$中格节点进行遍历，当访问每个格节点时，识别其类型并执行相应的操作。

定理2 $RE(K)$中任意格节点C是产生子格节点当且仅当$\neg(a_extent(C)\subseteq\{m^*\}')$且$\neg\exists C_1\in$ BEFORE$(C)(a_extent(C_1)=a_extent(C)\cap\{m^*\}')$。其中，BEFORE$(C)=\{C_x \mid C_x<_L C,\ C_x\in RE(K)\}(\{C_{new}\mid C_{new}$是$C_x$产生的新生格节点$\wedge\ (C_x<_L C),\ C_x\in RE(K)\}$。

证明：必要性。C是产生子格节点，故C不是更新格节点，即$\neg(a_extent(C)\subseteq\{m^*\}')$。由定义8可知，$\neg\exists C_1\in\{C_x\mid C_x<_L C, C_x\in RE(K)\}(a_extent(C_1)=a_extent(C)\cap\{m^*\}')$，且$\neg\exists C_1\in\{C_{new}\mid C_{new}$是$C_x$产生的新生格节点$\wedge(C_x<_L C)\wedge(C_x\leqslant C), C_x\in RE(K)\}(a_extent(C_1)=a_extent(C)(\{m^*\}')$。设$C_2$是$C_x$产生的新生格节点，且$(C_x<_L C)\wedge\neg(C_x\leqslant C)$，假设$a_extent(C_2)=a_extent(C)\cap\{m^*\}'$，有$a_extent(C_x)\ a_extent(C)\cap\{m^*\}'=a_extent(C)\cap\{m^*\}'$，由格的完备性可知，$\exists C_3\in RE(K)\wedge(C_3\leqslant C)\wedge(a_extent(C_3)=a_extent(C_x)\cap a_extent(C))$，与$C$是产生子格节点矛盾，假设不成立。必要性成立。

充分性。对于任意$C_1\in RE(K)$且$C<_L C_1$，假设$a_extent(C_1)=a_extent(C)\cap\{m^*\}'$，则$a_extent(C_1)\subset a_extent(C)$，由定理1可知，$C_1<_L C$，矛盾，所以对于任意$C_1\in RE(K)$，有$a_extent(C_1)\neq a_extent(C)\cap\{m^*\}'$。对于任意$C_2\leqslant C$，有$a_extent(C_2)\subset a_extent(C)$，由定理1可知，$C_2<_L C$，所以$a_extent(C_2)\cap\{m^*\}'\neq a_extent(C)\cap\{m^*\}'$。充分性成立。

由定理2可知，判定C是否为产生子格节点时，可以忽略C之后访问的格节点及其生成的新生格节点。*SearchForInsert*可以识别产生子格节点。

定理3 T_1、T_2为概念树的树节点，且$T_2=T_1.child[g]$，若T_1为有效树节点且$g>\max(\{m^*\}')$，则以T_2为根的子概念树中所有有效树节点对应的相对约简格节点均为不变格节点。

证明：设C_1是有效树节点T_1对应的格节点，C_x是以T_2为根的子概念树中有效树节点T_x对应的格节点，则有$C_1<_L C_x, a_extent(C_1)\subset a_extent(C_x), g\in a_extent(C_x)$。$g>\max(\{m^*\}')$，故$\neg(a_extent(C_x)\subseteq\{m^*\}')$，即$C_x$不是更新格节点。$a\_extent(C_x)\cap\{m^*\}'=a\_extent(C_1)\cap\{m^*\}'$，且$C_1<_L C_x$，由定理2可知，$C_x$不是产生子格节点。

由定理3可知，搜索更新格节点和产生子格节点时，可以忽略有效树节点T的大于$\max(\{m^*\}')$分支下的所有树节点对应的格节点。

定理4 如果$C\in RE(K)$，且$a_extent(C)=\{m^*\}'$，对于任意$C_x\in RE(K)$，若$C<_L C_x$，则C_x必为不变格节点。

证明：反证法。对于任意$C_x\in RE(K)$，$C<_L C_x$，假设C_x为更新格节点，则$a_extent(C_x)\subset\{m^*\}'$。由于$a_extent(C)=\{m^*\}'$，故$a_extent(C_x)\subset a_extent(C)$。由定理1可知，$C_x<_L C$，与已知矛盾。假设$C_x$为产生子格节点，则$RE(K)$中不存在$C_1$满足$a_extent(C_1)=a_extent(C_x)\cap\{m^*\}'=a_extent(C_x)\cap a_extent(C)$，与格的完备性矛盾。

由定理4可知，搜索更新格节点和产生子格节点时，若存在$C\in RE(K)$，使得$a_extent(C)=\{m^*\}'$，可以忽略C之后访问的所有格节点。

定理5 C为$RE(K)$中产生子格节点，其对应的新生格节点为C_{new}，$Child$为C_{new}的直接子格节点集合，(1)对于任意$C_c\in Child, m^*\in a_intent(C_c)$。(2)对于任意$C_{md}\in RE(K)$，若$C_{md}$为更新格节点，$RE(K^*)$中该节点被更新为$C_{md'}$，且$C<_L C_{md}$，则$C_{md'}\notin Child$。(3)对于任意$C_{nb}\in RE(K^*)$，若$C_{nb}$为新生格节点，其对应的产生子格节点为$C_{ge}$，且$C<_L C_{ge}$，则$C_{nb}\notin Child$。(4)对于任意$C_{c1}, C_{c2}\in Child$，必有$a_extent(C_{c1})\not\subset a_extent(C_{c2})$且$a_extent(C_{c2})\not\subset a_extent(C_{c1})$。

证明：由$C_c\in Child$，有$a_intent(C_{new})\subset a_intent(C_c)$，故$m^*\in a_intent(C_c)$。(1)成立。假设$C_{md'}\in Child$，有$a_extent(C_{md'})\subset a_extent(C_{new})$，由于$a_extent(C_{new})\subset a_extent(C)$，则$a_extent(C_{md})\subset a_extent(C)$，由定理1可知，$C_{md}<_L C$，与$C<_L C_{md}$矛盾。(2)成立。假设$C_{nb}\in Child$，有$a_extent(C_{nb})\subset a_extent(C_{new})$，则$a_extent(C_{ge})\cap a_extent(C)\cap\{m^*\}'=a_extent(C_{ge})\cap\{m^*\}'$，由格的完备性可知，$C_{ge}$不是产生子格节点，与已知矛盾。(3)成立。由$C_{c2}\in Child$，有$a_extent(C_{c2})\subset a_extent(C_{new})$。假设$a_extent(C_{c1})\subset a_extent(C_{c2})$成立，有$a_extent(C_{c1})\subset a_extent(C_{c2})\subset a_extent(C_{new})$，与$C_{c1}$是$C_{new}$的直接子格节点矛盾。同理可证$a_extent(C_{c2})\not\subset a_extent(C_{c1})$。(4)成立。

由定理5可知，搜索新生格节点的直接子格节点时，只需搜索已经产生的更新格节点和新生格节点，同时忽略直接子格节点的所有子节点。

4.3 算法描述

根据上述定理，本节给出基于属性的相对约简格快速渐进式构造算法(简称RRCL_A)，算法描述如下：

输入：形式背景(G,M,I)对应的相对约简格$RE(K)$，相应的概念树*root*，新增属性m^*；

输出：形式背景$(G,M\cup\{m^*\},I\cup I^*)$对应的相对约简格$RE(K^*)$，相应的概念树*root*；

```
BEGIN
  RE(K*)= RE(K);
  t.node=root, t.extern=∅, t 入队列;
  A_m={m*}';
  WHILE 队列非空 DO
    队头节点出队列并置于 p 中;
    Node=p.node, ext= p.extern;
    IF Node 为有效树节点 THEN
      G_field={g|g∈G∧ g≤ max({m*}') };
    ELSE
      G_field=G;
    FOR 每个 g∈ G_field 按升序排列 DO
      SubNode=Node.child[g];
      IF (SubNode ! = NULL) THEN
        t.node=SubNode, t.extern=ext∪g, t 入队列;
    IF ext⊆{m*}' THEN
      A_m= A_m−extent(Node.lattice);
      q.con=Node.lattice, q.ext=ext, q 加入 V[|ext|];
      IF ext = {m*}' THEN
        将 m* 加入到 Node.lattice 的内涵集;
        exit;
    ELSE
      C=Node.lattice;
      A_css=ext∩{m*}';
      Index=SearchForInsert(root, A_css);
      IF Index.lattice==NULL THEN
        A_∩= extent(C)∩{m*}';
        IF {m*}'⊆ext THEN
          C_add=(A_∩, {m*});
        ELSE
          C_add=(A_∩, ∅);
        extent(C) = extent(C)−A_∩, A_m= A_m −A_∩;
        创建新节点 C_new=C_add, 并加入 RE(K*);
        q.con=C_new, q.ext=A_css, q 加入 V[|A_css|];
        Index.lattice =C_new;
        在 C 和 Cnew之间增加一条边;
        i= |A_css|−1;
        Child=∅;
        WHILE i≥ 0 DO
          FOR 每个 q∈V[i] DO
            IF q.ext⊆ A_css THEN
              tag=True;
```

```
        FOR 每个 r∈ Child DO
          IF q. ext ⊆ r. ext THEN
            tag=FALSE;
            break;
          IF tag THEN
            IF q. con 和 C 间存在边 THEN 删除 q. con 和 C 之
              间的边;
            C_new 和 q. con 间增加一条边;
            Child=Child ∪{q};
      i=i-1;
    IF {m*}'⊆ext THEN
      exit;
END
```

5 实验结果及其分析

在 windows XP 下用 Java 编程实现了 RRCL_A 算法，在 P4 2.8G 的计算机上对随机生成的数据集进行了实验。对象数目为 200，属性数目为 100，关系概率为 30%和 40%，生成了 2 个形式背景。实验将属性分成 10 组，每组 10 个属性，分别采用 Godin 算法[3]、CLIF_A 算法[4]、RRCL_A 和文[7]算法生成概念格，2 个形式背景的实验结果分别如图 3 和图 4 所示。

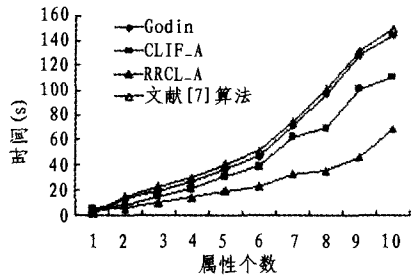

图 3 测试结果(|G|=200，|M|=100，关系概率为 30%)

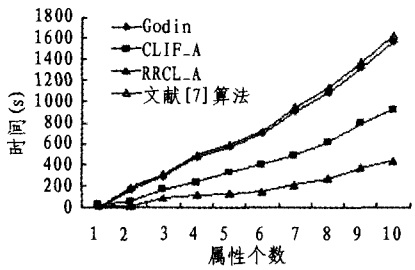

图 4 测试结果(|G|=200，|M|=100，关系概率为 40%)

由图 3 和图 4 可以看出：随着属性个数的增加，RRCL_A 算法和 CLIF_A 算法在时间上比 Godin 算法、文[7]算法更优。作为基于属性的算法，RRCL_A 算法优于 CLIF_A 算法，且随着关系概率的增大，RRCL_A 算法的优势体现得愈加明显。这种现象是由于较高的关系概率使得具备一个属性的对象数目增大，从而使得每个格节点的子节点数目增多，被 RRCL_A 算法忽略的节点个数也随之增多。从概念格存储角度，RRCL_A 算法最小，文[7]算法其次，CLIF_A 算法和 Godin 算法最差，这是由于 RRCL_A 算法只存储概念的初始内涵和初始外延，文[7]算法存储概念的初始内涵，而 CLIF_A 算法和 Godin 算法存储概念的真正内涵和真正外延。从算法的空间耗费角度，RRCL_A 算法需建立概念树，导致 RRCL_A 算法空间耗费最高。但相比概念格存储空间和构造时间上的节省，概念树的空间耗费可以忽略。

结论 本文提出了一种基于属性的相对约简格快速渐进式构造算法。该算法利用树结构组织格节点，通过概念树的按层遍历实现对格节点的访问。算法通过缩小产生子格节点判定时所需搜索的格节点的范围、缩小更新格节点和产生子格节点的搜索范围以及缩小新生格节点的直接子节点的搜索范围，达到了加速相对约简格构造的作用。实验结果表明，本文提出的算法优于 Godin 算法和现有的相对约简格构造算法。

参 考 文 献

1 Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations. Berlin Heidelberg: Springer-Verlag, 1999
2 胡可云，陆玉昌，石纯一. 概念格及其应用进展. 清华大学学报(自然科学版)，2000，40(9)：77～81
3 Godin R, M issaoui R, A laoui H. Incremental concept formation algorithms based on Galois (concept) lattices. Computational Intelligence, 1995, 11(2): 246～267
4 李云，刘宗田，陈峻. 基于属性的概念格渐进式生成算法. 小型微型计算机系统，2004，25(10)：1768～1771
5 张凯，胡运发，王瑜. 基于互关联后继树的概念格构造算法. 计算机研究与发展，2004，41(9)：1493～1499
6 谢志鹏，刘宗田. 概念格的快速渐进式构造算法. 计算机学报，2002，25(5)：490～496
7 张意德，宋简全，赵文兵，等. 相对约简格及其构造. 计算机工程与应用，2002，38(6)：196～197

(上接第 123 页)

现移位表的值为 0(如果它是某些模式的后缀)，因此，不得不分别检查具有这种后缀的所有模式，看它们是否与文本匹配。为了加速这个处理，引入了另外一张表格，叫做前缀表。当发现移位表哈希值为 0，并需要利用哈希表确定是否有匹配的时候，可以先检查前缀表中的值与文本中相应的前缀(通过向左移动 $m-B'$ 个字符，B' 是前缀表中字符块的大小)是否相符，过滤掉大量的模式。但是，只有当移位表的值经常为 0，也就是规则条数很多，且具有很高的冲突可能性的时候，使用前缀表才有意义。因此，在具体的应用中，是否需要前缀表要根据具体应用环境而定。

(2)扫描阶段

算法的主循环有以下几个步骤：

1)根据文本中当前的 B 个字符，计算哈希值；

2)检查 SHIFT[h]的值，如果该值大于 0，移动文本，转向 1；否则转向 3；

3)计算文本前缀的哈希值，(从第 m 个字符开始，到当前位置的左侧)，记为 text Prefix；

4)检查每个 p，其中 HASH[h]$\leqslant p\leqslant$HASH[$h+l$]，PREFIX[p]=test_prefix 是否成立。如果它们相等，使用真正的模式(由 PAT_PIONT[p]得到)对文本直接进行检查。

结论 本文主要讨论了基于 Snort 的网络入侵检测的问题，Snort 是一个开放源代码的网络入侵检测系统，这就使得我们可以对入侵检测的方法进行改进，本文主要从规则优化和匹配规则优化两个角度讨论了对 Snort 安全性能提高的方法。本文创新点在于将基于 Snort 的网络入侵检测系统进行了研究，并对入侵检测的算法进行了改进，使得检测的有效率更高。

参 考 文 献

1 INSKI A C~. A synchronization, Algorithm for processes with dynamic priorities in computer networks with node failures[J]. Information Processing Letters, 1989, 32(3): 129～136
2 INSKIA Two Algorithms for Mutual Exclusion in. Real-time Distributed Computer Systems[J]. Journal of Parallel and Distributed Computing, 1990, 9(1): 77～82
3 CASWELLBrian, BEALEJay, FOSTERJamesC, et al. Snort2.0 intrusion detection[M]. 北京：国防工业出版社，2004
4 唐正军. 黑客入侵防护系统源代码分析[M]. 北京：机械工业出版社，2002
5 孙振龙，等. 基于数据挖掘技术的 snort 入侵检测系统的研究[M]，微计算机信息，2006，22(11-3)