计算机科学1999Vol 26№ 11

图回路算法的演绎综合

Deductive Synthesis of Graphic Circuit Algorithms

黄明和　　刘润杰

（江西师范大学计算机科学系　南昌330027）（安阳市邮电局　安阳455000）

Abstract This paper begins with a common and explicit specification of graphic circuit problem. By appropriate controlling constraint conditions and using program transformation techniques, we formally derive a class of graphic circuit algorithms, effectively realize the reusability of equations and deductive steps, and reveal the common characteristics, inherent characteristics and relations between the algorithms.

Keywords Circuit, Algorithm, Synthesis, Program transformation

1. 引言

演绎综合是形式化开发算法(程序)的重要方法之一，用这种方法开发算法或程序不仅使人信服所得的算法(程序)确实能够完成给定的任务，而且通过揭示导致最后算法(程序)的设计决策来清楚表达算法(程序)怎样完成给定任务，改变开发过程中的设计决策和适当控制约束条件和决策的时机，我们能得到求解同类问题的一个算法族，这有利于揭示算法之间的内在联系，使得已经完成的工作得到尽可能的重用，本文将这种方法应用到有向图的回路这一典型图算法上，并对一般回路，简单回路、欧拉回路、哈密尔顿回路算法进行了演绎综合，以展示这种方法的功能。本文综合的基础是一个小的程序变换规则集，基本集合演绎法则集和与问题有关的若干记号和表示。

2. 变换法则、记号和表示

为了描述和推导算法，我们需要如下基本程序变换法则、集合演绎法则、有关定义和记号。

2.1 基本程序变换法则

(1)展开(unfolding)　设$E<==E_1$ $F<==F_1$是方程，如果在F_1中出现E的实例，则可用对应的E_1的实例去替换它。

(2)折迭(folding)　设$E<==E_1$ $F<==F_1$是方程，如果在F_1中出现E_1的实例，则可用对应的E的实例去替换它。

(3)实例化(instantiating)　引进对应方程的一个实例。

2.2 基本集合演绎法则

RS1　$\{f(x)|x\in\Phi\wedge p(x)\}<==\Phi$

RS2　$x\in A\wedge x\neq s<==x\in A-\{s\}$

RS3　$x\in A\wedge x\in B<==x\in A\cap B$

RS4　$\{x|x\in\{s\}\wedge p(x)\}<==\begin{cases}\{s\} & \text{若 } p(s)\\ \Phi & \text{其它}\end{cases}$

RS5　$\{f(x)|x\in S_1\cup S_2\wedge p(x)\}<==\{f(x)|x\in S_1\wedge p(x)\}\cup\{f(x)|x\in S_2\wedge p(x)\}$

RS6　$x\in\{s\}<==x=s$

2.3 有关记号

设G(E,V)表示一个无自环的有向图，E是其边集，V是其顶点集；

NO1　设P为图G的一条路径，D(P)表示P上结点组成的集合，VD(P)表示P上边组成的集合，则：

$$\text{Sample}(p)=\begin{cases}\text{True} & \text{若 p 上无相同边}\\ \text{False} & \text{否则}\end{cases}$$

$$\text{Sample2}(p)=\begin{cases}\text{True} & \text{若 p 上无相同点}\\ \text{False} & \text{否则}\end{cases}$$

NO2　设$S_1=\langle k_1,k_2,\cdots,k_i\rangle$ $S_2=\langle l_1,l_2,\cdots,l_j\rangle$是两个序列，则：

$S_1+S_2=\langle k_1,\cdots,k_i,l_1,\cdots,l_j\rangle$

$S_1+\{M_i|i=1,2,\cdots k,M_i\text{是序列}\}=\{S_1+M_i|i=1,2,\cdots,k\}$

$S_1+\Phi=\Phi+S_1=\Phi$

NO3　设$X\subseteq V$，$Y\subseteq E$，$f(n,Y)=\{x|(n,x)\in Y\}$，$g(n,X)=\{x|(n,x)\in E\wedge x\in X\}$

3. 算法的演绎综合

首先我们给出其顶层规范描述如下：

$cycle<==\{P|P=\langle n_0,n_1,\cdots,n_i\rangle\wedge n_0=n_i\wedge\forall i,0\leqslant i<j\ n_{i+1}\in f(n_i,E)\}$

问题的描述是非常清楚的，它所表示的即是G中

从n_s出发的所有回路的集合。当所给的图有回路时，这是一个无穷集，（在同一回路上走不同的次数将得到不同的回路）由此可见，求解图的毫无限制的回路算法是没有意义的。这样得到的算法一般都不会终止，（无回路图除外）我们这里仅利用这个最朴素的回路定义求解其它回路算法。

3.1 一般回路的算法综合

所谓一般回路即回路上下允许有相同的边，由回路的顶层定义，我们很容易写出一般回路的方程定义如下：

1）$cycle1 <==\{P|P\in cycle \wedge sample1(P)\}$

$<==\{P|P=(n_s,n_1,\cdots,n_t)\wedge n_0=n_t\wedge\forall\ i\ 0\leqslant i<t\ \ n_{i+1}\in f(n_i,E)\wedge sample1(P)\}$ 按顶层描述展开

稍加变形有：

$<==\{P|P=(n_0,n_1,\cdots,n_t)\wedge n_0=n_t\wedge\forall\ i,0\leqslant i<t\ \ n_{i+1}\in f(n_i,E)\wedge n_t\in f(n_0,E)\wedge VD(P)\subseteq E\wedge sample1(P)\}$

为了导出递归，我们引进更一般的方程。

2）$path1(n,n_t,N,K)<==\{P|P=(n,n_1,\cdots,n_j)\wedge\forall\ i\ 1\leqslant i<j\ \ n_{i+1}\in f(n_i,K)\wedge n_j\in N\wedge VD(P)\subseteq K\wedge sample1(P)\}$

易见，方程1是方程2的实例，亦即有：

3）$cycle1<==path1(n_0,n_0,f(n_0,E),E)$ 实例化方程2

下面我们来推导方程2，也就是要设法把方程2写成递归的形式。令$N=N_1\cup N_2$，且$N_1\cap N_2=\Phi$，则有：

4）$path1(n,n_t,N_1\cup N_2,K)<==\{P|P=(n,n_1,\cdots,n_j)\wedge\forall\ i\ 1\leqslant i<j\ \ n_{i+1}\in f(n_i,K)\wedge n_j\in N_1\cup N_2\wedge VD(P)\subseteq K\wedge sample1(P)\}$ 实例化方程2

$<==\{P|P=(n,n_1,\cdots,n_j)\wedge\forall\ i\ 1\leqslant i<j\ \ n_{i+1}\in f(n_i,K)\wedge n_j\in N_1\wedge VD(P)\subseteq K\wedge sample1(P)\}\cup\{P|P=(n,n_1,\cdots,n_j)\wedge\forall\ i\ 1\leqslant i<j\ \ n_{i+1}\in f(n_i,K)\wedge n_j\in N_2\wedge VD(P)\subseteq K\wedge sample1(P)\}$ 根据RS5

$<==path1(n,n_t,N_1,K)\cup path1(n,n_t,N_2,K)$ 用方程2折迭

必须考虑的三种基本情况是：

5）$path(n,n_t,\{v\},K)<==\{P|P=(n,n_1,\cdots,n_j)\wedge\forall\ i\ 1\leqslant i<j\ \ n_{i+1}\in f(n_i,K)\wedge n_j\in\{v\}\wedge VD(P)\subseteq K\wedge sample1(P)\}$ 实例化方程2

$<==\{P|P=(n,n_1,\cdots,n_j)\wedge\forall\ i\ 1\leqslant i<j\ \ n_{i+1}\in f(n_i,K)\wedge n_j=v\wedge VD(P)\subseteq K\wedge sample1(P)$ 根据RS4

稍加变形有：

$<==\{P|P\in(n)+(v,m_1,\cdots,m_k)\wedge\forall\ i\ 1\leqslant i<k\ \ m_{i+1}\in f(m_i,K-\{(n,v)\})\wedge m_k=n_t\wedge m_1\in f(v,K-\{(n,v)\})\wedge VD(P)\subseteq K\wedge sample1(P)\}$

注意到，此时若有$v=n_t$，则我们已求到一条满足条件的路，此路应该收集。同时从v出发还可能走出更大的满足条件的路的其他部分。由此我们有：

$<==\{P|P\in(n)+(\{v\}\cup P_1)|P_1=\{P'|P'=(v,m_1,\cdots,m_k)\wedge\forall\ i\ 1\leqslant i<k\ \ m_{i+1}\in f(m_i,K-\{(n,v)\})\wedge m_k=n_t\wedge m_1\in f(v,K-\{(n,v)\})\wedge VD(P')\subseteq K-\{(n,v)\}\wedge sample1(P')\}$ 若$v=n_t$

$<==\{P|P\in(n)+P_1\ \ P_1=\{P'|P'=(v,m_1,\cdots,m_k)\wedge\forall\ i\ 1\leqslant i<k\ \ m_{i+1}\in f(m_i,K-\{(n,v)\})\wedge m_k=n_t\wedge m_1\in f(v,K-\{(n,v)\})\wedge VD(P')\subseteq K-\{(n,v)\}\wedge sample1(P')\}$ 其它

$<==\{P|P\in(n_t+(\{v\}\cup P_1))\}$ 若$v=n_t$

$<==\{P|P\in(n)+P_1\}$ 其它

这里$P_1=path1(v,n_t,f(v,K-\{(n,v)\}),K-\{(n,v)\})$ 用方程2折迭

6）$path1(n,n_t,\Phi,K)<==\Phi$ 根据方程2的定义和RS1

7）$path1(n,n_t,N,\Phi)<==\Phi$ 根据方程2的定义和RS1

由此我们得到求一般回路算法H1.

$H1_1$ $cycle1<==paht1(n_s,n_s,f(n_0,E),E)$

$H1_2$ $path1(n,n_t,N_1\cup N_2,K)<==path1(n,n_t,N_1,K)\cup path1(n,n_t,N_2,K)$

$H1_3$ $path1(n,n_t,\{v\},K)<==\{P|P\in(n)+(\{v\}\cup P_1)\}$ 若$v=n_t$

$<==\{P|P\in(n)+P_1\}$ 其它

这里$P_1=path1(v,n_t,f(v,K-\{(n,v)\}),K-\{(n,v)\})$

$H1_4$ $path1(n,n_t,\Phi,K)<==\Phi$

$H1_5$ $path1(n,n_t,N,\Phi)<==\Phi$

该算法将求出图G中所有边下相同的回路，f函数的引入将大大提高算法效率，单点情况的分析避免了可能出现的小回路被大回路淹没的情况，确保算法求出所有符合要求的回路。

3.2 简单回路的算法综合

所谓简单回路，即除头尾外，中间不允许有相同点的回路，我们容易写出其形式化描述如下：

8）$cycle2<==\{P|P=(n_0,n_1,\cdots,n_t,n_{t+1})\wedge P\in cycle\wedge\forall\ i,k\ i\neq k\ 0\leqslant i,k\leqslant t\ n_i\neq n_k\}$，亦可等价地写成：

$<==\{P|P=(n_0,n_1,\cdots,n_t)\wedge\forall\ i\ 0\leqslant i<t\ \ n_{i+1}\in f(n_i,E)\wedge n_0\in f(n_t,E)\wedge sampe2(p)\ +(n_0)$

稍加变形又有：

$<==\{P|P=(n_0,n_1,\cdots,n_t)\wedge\forall\ i\ 1\leqslant i<t\ \ n_{i+1}\in f(n_i,E)\wedge n_1\in g(n_0,V)\wedge D(P)\subseteq V\wedge n_0\in f(n_t,E)\wedge sample2(p)\}-(n_0)$

为了求出粗斜体部分的推导，我们需要引入更一般的方程。

9）$path2(n,n',N,K)<==\{P|P=(n,n_1,\cdots,n_t)$

∧∀ i 1≤i<j $n_{i+1}\in f(n_i,E)\wedge n_i\in N\wedge n\in f(n_i,E)$ ∧D(P)⊆K∧sample2(P)

易见方程8粗斜体部分恰是方程9的实例，亦即有：

10)cycle2<==path2($n_2,n_0,g(n_2,V),V$)+(n_0)

下面的任务是推导方程9，

类似前面的方法，令$N=N_1\cup N_2$，且$N_1\cap N_2=\Phi$，则有：

11)path2$(n,n',N_1\cup N_2,K)$<==={P|P=⟨n,n_1,…,n_j⟩∧∀ i 1≤i<j $n_{i+1}\in f(n_i,E)\wedge n_i\in N_1\cup N_2\wedge n'\in f(n_j,E)\wedge D(P)\subseteq K\wedge$sample2(P)} 实例化方程9

<==={P|P=⟨$n_2,n_1,\cdots,n_j$⟩∧∀ i 1≤i<j $n_{i+1}\in f(n_i,E)\wedge n_i\in N_1\wedge n'\in f(n_j,E)\wedge D(P)\subseteq K\wedge$sample2(P)}∪{P|P=⟨n,$n_1,\cdots,n_j$⟩∧∀ i 1≤i<j $n_{i+1}\in f(n_i,E)\wedge n_i\in N_2\wedge n'\in f(n_j,E)\wedge D(P)\subseteq K\wedge$sample2(P)}

根据 RS5

<==path2$(n,n',N_2,K)\cup$path2(n,n',N_2,K)

用方程9折迭

同理，我们必须考虑以下基本情况：

12)path2$(n,n',\{v\},K)$<==={P|P=⟨n,$n_1,\cdots,n_j$⟩∧∀ i 1≤i<j $n_{i+1}\in f(n_i,E)\wedge n_i\in\{v\}\wedge n'\in f(n_j,E)\wedge D(P)\subseteq K\wedge$sample2(P)} 实例化方程9

注意到若$n'\in f(v,E)$，则 v 是 path2中一条符合条件的路的尾点。同时从 v 出发，还可寻求其它符合path2中条件的路，由此就有：

<==={P|P∈(n)+({(v)}∪P_1) P_2={P'|P'=⟨v,$m_2,\cdots,m_k$⟩∧∀ i 1≤i<k $m_{i+1}\in f(m_i,E)\wedge m_1\in g(v,K-\{n\})\wedge m_k=n_1\wedge n'\in f(n_1,E)\wedge D(P')\subseteq K-\{n\}\wedge$sample2(P')} 若$n'\in f(v,E)$

<==={P|P∈(n)−P_1 P_2={P'|P'=⟨v,$m_2,\cdots,m_k$⟩∧∀ i 1≤i<k $m_{i+1}\in f(m_i,E)\wedge m_1\in g(v,K-\{n\})\wedge m_k=n_1\wedge n'\in f(n_1,E)\wedge D(P')\subseteq K-\{n\}\wedge$sample2(P')} 其它

<==={P|P∈(n)+({(v)}∪P_1) P_2=path2$(v,n',g(v,K-\{n\}),K-\{n\})$} 若$n'\in f(v,E)$

<==={P|P∈(n)+P_2 P_1=path2$(v,n',g(v,K-\{n\}),K-\{n\})$} 其它

用方程9折迭

13)path2(n,n',Φ,K)<==Φ 根据方程9和 RS1

14)path2(n,n',N,Φ)<==Φ 根据方程9和 RS1

这样，我们得到求解图 G 中简单回路的算法 H2：

$H2_1$ cycle2<==path2$(n_0,n_0g(n_0,V),V)$+(n_0)

$H2_2$ path2$(n,n',N_1\cup N_2,K)$<==path2$(n,n',N_1,K)\cup$path2(n,n',N_2,K)

$H2_3$ path2$(n,n',\{v\},K)$<==={P|P∈(n)+({(v)}∪P_1)∧P_1=path2$(v,n',g(v,K-\{n\}),k-\{n\})$} 若$n'\in f(v,E)$

<==={P|P∈(n)+P_2∧P_1=path2$(v,g(v,K-\{n\}),$ $K-\{n'\})$} 其它

$H2_4$ path2(n,n',Φ,K)<==Φ

$H2_5$ path2(n,n',N,Φ)<==Φ

应该看到，由于 f 函数和 g 函数的引入，以及算法的具体做法，上面算法中的判别条件：VD(P)⊆K，D(P)⊆K，sample1(P)，sample2(P)等只有说明意义，算法中的具体做法已经保证了它们的成立，由此在算法的程序实现时可以去掉，这无疑将进一步提高算法效率。

3.3 欧拉回路的算法综合

所谓欧拉回路是我们前述的一般回路的一种特殊情况，它要求 E 中的所有边在回路中出现一次。这种回路，如果存在的话，在我们的 H1算法中已经求出来了。为了给出专门求欧拉回路的算法，我们现在只需对算法 H1的推导过程作简单的修正，绝大多数推导步将得到完全的重用。下面我们仅给出说明和修正部分，然后直接给出算法，欧拉回路的形式说明如下：

15)Ecycle<==={P|P∈cycle∧VD(P)=E}

类似地，path1将变成：

16)path3(n,n_1,N,K)<==={P|P=⟨n,$n_1,\cdots,n_i$⟩∧∀ i 1≤i<j $n_{i+1}\in f(n_i,K)\wedge n_1\in N\wedge$VD(P)=K∧sample1(P)}

如此有单点情况变化如下：

17)path3$(n_i,n_j,\{v\},K)$<==={P|P=⟨n⟩+⟨v,$m_1,\cdots,m_k$⟩∧∀ i i≤i<k $m_{i+1}\in f(m_i,K-\{(n,v)\})\wedge m_k=n_j\wedge m_1\in f(v,K-\{(n,v)\})\wedge$VD(P)=K∧sample1(P)} 实例化方程16并根据 RS4和 NO3

此时应注意的是：如果 $v=n_j$，并且 $K-\{(n,v)\}=\Phi$，则一条符合条件的路已经产生，并且从 v 出发，不可能再走下去。由此对上式可继续推导出：

<==={P|P=⟨n,v⟩}若 $v=n_j$ and $K-\{(n,v)\}=\Phi$

<==={P|P∈(n)−P_1∧P_2=path3$(v,n_j,f(v,k-\{(n,v)\},k-\{(n,v)\})$} 其它

根据上述分析和用方程16折迭

最后我们给出求欧拉回路的算法 H3如下：

$H3_1$ Ecycle<==path3$(n_0,n_0,f(n_0,E),E)$

$H3_2$ path3$(n,n_i,N_1\cup N_2,K)$<==path3$(n,n_j,N_1,K)\cup$path3(n,n_i,N_2,K)

$H3_3$ path3$(n,n_i,\{v\},K)$<==={P|P=⟨n,v⟩}

若 $v=n_j$ and $K-\{(n,v)\}=\Phi$

<==={P|P∈(n)+P_2∧P_2=path3$(v,n_j,f(v,K-\{(n,v)\}),K-\{(n,v)\})$} 其它

$H3_4$ path3(n,n_j,Φ,K)<==Φ

$H3_5$ path2(n,n_j,N,Φ)<==Φ

3.4 哈密尔顿回路算法的综合

所谓 H 回路是图 G 中所有点在回路中出现一次

且仅出现一次的回路。这恰好又是我们算法H2的特例，也就是说，如果G中有H回路的话，在我们的H2算法中已经求出来了，我们可以从H2算法的推导中轻松地得到求H回路的算法，完全同理，下面我们仅给出求H回路算法的说明和修正部分，然后直接给出求H回路的算法。

(18)Hcycle<==={P|P∈cycle2∧D(P)=V}

类似地，path2将变成

(19)path4(n,n',N,K)<==={P|P=(n,n_1,…,n_i)∧∀i(1≤i<j n_{i+1}∈f(n_i,E)∧n_i∈N∧D(P)=K∧n'∈f(n_j,E)∧sample2(P).

单点情况变化如下：

20)path1(n,n',{v},K)<==={P|P=(n)-(v,m_1,…,m_k)∧∀i(1≤i<k m_{i+1}∈f(m_i,E)∧m_k=n_i∧m_i∈g(v,k-{n})∧n'∈f(n_i,E)∧D(P)=K∧sample2(P)}　　实例化方程19并根据RS4和NO3

注意到此时若n'∈f(v,E)，且K-{n}={v}，则一条符合条件的路已经求出，并且从v出发不可能再走下去。由此上面方程可继续推导为：

<==={P|P=(n,v)}

若n'∈f(v,E)and K-{n}={v}

<==={P|P∈(n)-P_1∧P_1=path4(v,n',g(v,K-{n}),K-{n})}　　其它

最后有求解H回路的算法H4如下：

$H4_1$ Hcycle<==path4(n_0,n_t,g(n_0,V),V)+(n_0)

$H4_2$ path4(n,n',N_1∪N_2,K)<==path4(n,n',N_1,K)∪path4(n,n',N_2,K)

$H4_3$ path4(n,n',{v},K)<==={P|P=(n,v)}

若n'∈f(v,E)and K-{n}={v}

<==={P|P∈(n)+P_1∧P_1=path4(v,n',g(v,K-{n}),K-{n})}　　其它

$H4_4$ path4(n,n',Φ,K)<==Φ

$H4_5$ path4(n,n',N,Φ)<==Φ

结束语　算法综合是提高软件可靠性，可维护性，可重用性，最终实现软件自动生成的重要手段之一。本文旨在利用演绎和重用技术寻求一条高效的形式化开发同类问题串型算法族的有效途径，将算法综合转向并行化和部件化，以更大提高算法软件的开发效率和执行效率是我们正在研究的新课题。

参考文献

1 Darlington J. A synthesis of several sorting algorithms. Acta Informatic, 1978(11):1～30

2 Manna Z, Waldinger R. Synthesis: Dream ⇒ Program. IEEE Transsation on Software engineering, 1979, SE-5(4) 291～328

3 黄明和，邓少敏. 程序变换综述. 计算机与现代化，1991.1

4 许卓群，张乃孝，杨冬青，唐世渭. 数据结构. 高等教育出版社，1987

5 黄明和 树遍历算法的演绎综合. 计算机工程与科学，1997.1

(上接第71页)

2　4　8　4　2

或者：　　X　5　3

2　4　5　4　2

2　3　2

有抖动算法不能够建立颜色对应表，因为在原图像中同一种颜色的点经过分色处理后的颜色未必一样(因为抖动处理时，被分散的色差将改变点的颜色)，所以只能够动态地生成对应颜色。其分色算法描述如下：

(1)读入原图像的一点的颜色，在被选中的颜色表中选择与之色差最小的颜色，在其相应的分色图像中置该颜色的标志，在其它的分色图像中置空白(背景)色。

(2)求得处理之后该点的颜色与原始图像中该点的色差，然后根据抖动算法将色差分散，改写相应点的颜色(原始图像中该点的颜色并不能改变)。

(3)重复进行(1)、(2)的处理，直至原图像中所有的点均被处理。

以上介绍了两种专色分色处理的算法

5. 分色的色点省略

在图像中往往有一些颜色的权值很小，但其颜色却很特别，即与其它颜色的色差很大，这些颜色在确定分色数的归并中就很可能被保留，并致使其它权值大的颜色被归并掉，使得处理后得到的图像失真严重；或者在确定归并色差的归并中，使分色数过大，为了解决这个问题，可以将一定权值以下的颜色先归并成相近的颜色，再进行专色分色处理。为此要引进"色点省略比率"这个系数，该系数定义如下：

色点省略比率＝要省略颜色的最大权值/总权值

用户定义了色点省略比率之后，即可得到被省略颜色的最大权值(因为总权值即为图像总点数)，小于该权值的颜色被归并为与之最为接近的颜色，归并算法同3.1.2中介绍的算法。通过省略权值过小的颜色，能够有效保留主要颜色，使专色分色处理获得较好的结果。