自适应自动程序设计及其在自动售货机中的应用

余世明，丁国正，刘立峰
（浙江工业大学信息工程学院，杭州　310012）

［摘要］　对程序故障的处理，通常可以使用返回值或异常来报告。但当状态模型中的程序出现故障后，使程序仍能回到出现故障处的状态或做进一步的处理，这是返回值或异常做不到的。针对自动处理的实现，提出了采用组合编码和状态冗余设计相结合的自动程序设计思想，给出了它的一个结构模型和算法。模型通过减少程序的状态切换故障（预防）和增加系统的故障恢复能力（治疗）来实现这个目标。最后，将这个设计模型部分成功地应用到自动售货机上，实际运行效果良好。

［关键词］　自适应；状态冗余设计；自动程序设计；故障恢复；自动售货机
［中图分类号］　TP311.11　［文献标识码］　A　［文章编号］　1009－1742(2007)11－0147－04

针对能自我复制的思维机器问题所提出的研究线路中，有限自动机是非常有用的类型之一[1]。MIT的McCulloch和Pitts提出了以神经元为基本单位的生物神经系统模型（McCulloch-Pitts，神经学模型），Turing提出了著名的万能机（图灵机），Von.Neumann提出了利用冗余来获得自适应能力的Neumann模型，这些模型的提出都是针对如何能制造一个机器来实现自我复制，甚至能思考[1]。这些模型虽未完全实现，但它为多状态频繁转换的大型软件工程的自适应实现提供了重要的基础。

以状态为中心的设计为故障的处理提供了基础。通常采用返回值或异常来报告程序错误，但怎样能让机器无需人工干预而自动处理这些错误，通常是办不到的[2]。利用有限自动机和冗余状态设计却可以基本实现故障的报告、修复等。如果把故障看作状态的误转换，将冗余状态设计成故障处理状态，那么故障的处理过程，就是当程序出现故障（误转换）时，通过原状态的状态信息包和原状态信息备分包的比较，来判断他们是否一致而判定出是否出现了故障，同时自动跳转到故障处理状态，故障处理状态通过对原状态的状态信息包的分析，作出相应的处理。如出现通信故障，则可以自动重连接等。因而，状态信息包的实现和冗余状态（故障处理状态）的设计是实现自适应自动程序设计的关键。

自适应自动程序设计的主要理论依据是有限自动机理论和Von.Neumann的冗余设计思想，文章提出了一种自适应自动程序设计的结构模型，它将状态切换的判别条件进行组合来减少程序的状态切换故障并对状态信息打包储存，同时采用冗余状态获取系统的故障恢复能力。

1　结构模型的建立

结构模型的建立主要由从实际问题到状态模型、状态模型的简化、状态信息包的建立、冗余状态的设计四部分组成。

1.1　从实际问题到状态模型

实际问题往往可以用一个过程来描述，这个过程就是一系列状态的有序排列，但一个状态，由于过渡条件的多样化，往往可以过渡到多个状态，如从一台自动售货机上购买饮料，如果所投金额足，则会转入正常的吐出饮料状态，如果所投金额不足，则会转到销售提醒状态。这样的模型就是典型的不确定的有限状态自动机（NFA）[3]，这些过渡条件就是状态的切换条件。因而，将实际问题用状态模型来描述，就是将问题的一系列状态有序连接，并在这些连接线上加入它们的转换条件。

［收稿日期］　2007－07－14；修回日期　2007－08－26
［基金项目］　浙江嵌入式系统与应用重大专项资助项目(2005c11011)
［作者简介］　余世明(1962－)，男，甘肃天水市人，浙江工业大学信息工程学院博士、教授、高级程序员

1.2 状态模型的简化

由于实际中的模型往往是不确定的 NFA 模型，如果直接对其进行程序的设计，将会使程序产生过多的交叉而繁杂，同时，由于过多的循环套将会延长程序的执行时间，所以有必要将它化简为一系列单向的状态过程，这样的过程就是确定的 DFA 模型[4]。

在软件工程中，不能让状态呈现不确定状态而影响接下来的程序设计，通常不直接使用 NFA 模型，而先转化为 DFA 模型。这样的转化就是将 NFA 的功能等价到 DFA，即将描述 NFA 的功能状态不重复地列出，并将各个状态的可能转化相应列出；DFA 的简化就是删除孤立的状态，合并等价的状态，但要保证 DFA 最小数量的状态是不容易实现的[5,6]。

1.3 状态信息包的建立

当前状态的状态信息包由状态的编码（上一状态和当前状态的状态信息）和状态切换条件（上一状态切换到当前状态的条件信息）组成。对于状态的编码，采用通常的连续编码，如第 i 个状态 S_i 编码为 i。状态切换条件往往是多个条件的联合，这里将采用一种新的编码方式即组合编码。

程序设计中，状态间的切换条件是多个判别变量的组合，需要逐一判别（用循环套等待判别条件成立），如果把各个判别变量逐一分开来判别，一旦出现程序的误跳转，程序将很难获得发生故障处的状态信息，当然就不能恢复到正常运行状态了。针对这样的问题，将状态信息和所有判别的条件变量排序组合到一个组合变量中（即状态信息包），这样就可以在出现误跳转的情况下，通过查看状态信息包来获取发生故障处的状态信息，为故障的处理做好信息准备。状态信息包的建立具体如下：

对于任意的一个动态系统，假设它存在 n 个状态，且记为 $\{S_1, S_2, \cdots, S_n\}$，并假设从状态 S_i 转移到状态 S_j 的切换条件变量共有 m 个即 $C_{ij}^1\ C_{ij}^2 \cdots\ C_{ij}^m$，将它们合记为 $C_{ij} = ijC_{ij}^1\ C_{ij}^2 \cdots\ C_{ij}^m$（$i, jC_{ij}^m$ 的有序排列）其中 C_{ij}^k（$k=1,2,\cdots,m$；$i,j=1,2,\cdots,n$）为各个判别变量的参数标志，并规定 C_{ij} 的值（下标 ij 和 C_{ij}^m 分别单独处理）为从首至尾以 4 个分条件的参数标志（最后一组不足 4 时在此组前面补 0）为一组的值的定序排列，类似 BCD 码的记录方式，当某一条件 C_{ij}^k 有效时，令 $C_{ij}^k=1$；无效时，令 $C_{ij}^k=0$。采用组合方式，除了能起到状态信息包的基本作用外，还有如下意义：

1）所有转移条件实现了形式化，对于编程人员的编程来说，编程将只面对条件参数 C_{ij}，而不再需要寻找具体的条件变量。

2）可以保证状态切换时出现死机的可能性更小。因为每个判别的循环套会判别所有必须的条件而不只是依次判别各个条件变量，这样就可以即时改变组合条件中的已经满足条件，避免因某一条件变量未满足而出现死循环而死机。

3）可以根据具体的切换情况为各个判别变量加权。如把此次切换最重要的变量排到组合变量的前面，将相对不重要的排到组合变量的后面，判别时用循环套依次判别它们，并在判别的循环套中加入一些强行退出循环套的程序（如时间限制等）。这样，当出现故障时，可将系统强行切换到故障处理状态。

1.4 冗余状态设计

冗余设计最早主要是用在数字逻辑电路设计中[7]。在程序的结构设计上，也可以借鉴它的冗余设计思想，不加入冗余电路，而是加入冗余状态，冗余状态是用来处理故障状态的，也称为故障处理状态，当程序出现如 1.3 节所述的故障时，通过分析状态信息包，做相应的故障处理。

故障的类型通常有[8]：针对通信故障的临时链路功能状态，针对处理资源故障的缓存功能状态，针对存储资源故障的状态备份功能状态和针对处理资源故障使状态功能失效的补偿功能状态。通过分析信息包虽能获得一定的故障信息，但故障状态的具体类型仍很难完全的确定，所以在不能确定的地方要出现这四种故障状态都要报告的状况，将它们进行轮循处理。这些手段只是在出现故障时使用，不会对程序的正常运行产生负面影响。

2 相关命题及其证明

命题 1　任何 NFA 均可以等价于 DFA[4]。

命题 2　（编码识别）对于任意序列 $a=ijxxx\cdots$，$b=klxxx\cdots$（i,k 为正整数，j,l 为非负整数，x 为任意非负整数），只依靠前有限位完全区分 a 和 b 的编码不可以直接用二进制方式，但可以取前有限位如 ij 和 kl，对各数单独采用二进制编码或类 BCD 方式编码可以区分，且用类 BCD 编码更好（类 BCD 编码指的是像 BCD 码一样将序列拆分成一段一段来编码）。

证明：设 a 和 b 的编码相同，取 i 为 p 位，j 为 q

位，取 k 为 q 位，l 为 p 位，显然 a 和 b 被区分了。反过来，就可以用它们组成序列 a 和 b，说明只依靠前有限位完全区分 a 和 b 的编码不可以直接用二进制方式。

记 i 有 n 位(十进制)且可以用最少 x 位二进制数表示，j 可以用最少 y 位二进制数表示，序列 ij 可以用最少 z 位二进制数表示，那么有 $i\leqslant 2^x$，$j\leqslant 2^y$，$ij\leqslant 2^z$ 即 $10^n i+j\leqslant 2^z$，令 $\Delta=x+y-z$，由上面的假设可知，$\Delta\leqslant 0$ 等价于 $\log 2[i*j-(10^n i+j)]\leqslant 0$ 即 $(i-1)(j-10^n)\leqslant 10^n+1$，这个不等式是显然成立的，因为 $j-10^n<0$，左边恒为负数，这样就说明了将序列 ij 当作 i 和 j 来编码比当作 ij(即 10^n*i+j)编码位数更少，即用类 BCD 编码更好，证毕。

由命题 2，容易得到如下推论：

推论　对于任意的一个序列 $a=a_1\ a_2\ a_3\cdots$，$a_i(i=1,2,\cdots)$为非负整数，对序列中的各个 a_i 分别编码是最优编码(编码位数最少)。

3　主要算法实现

针对模型的结构(从实际问题到状态模型、状态模型的简化、状态信息包的建立和冗余状态的设计)，模型算法分为两部来实现，一是预备工作，建立状态模型的 NFA，将 NFA 转化为 DFA 并化简；二是具体的算法，建立状态切换条件表和故障状态的替换策略。

3.1　状态切换条件表的建立

依据 1.3 节中所述的方法，可以建立各个状态间切换的条件表，如表 1 所示。

表 1　状态切换条件

Table 1　Condition of state switch

Stc	swc	Stc	swc	Stc	……	Stc	swc	Stc
S_1	C_{12}	S_2	C_{23}	S_3	……	S_i	C_{ii+1}	S_{i+1}
						S_i	C_{ij}	S_j

注：表中 $S_i(i=1,2,\cdots,n)$可以根据不同条件转移到多个其他状态，每一行就是一个 DFA 流程(无分支的流程)，这个表是用于指导编程的。Ste 为 State code 的缩写，swc 为 switch condition 的缩写。

3.2 故障状态的替换

将各个状态(除了初始状态)的第一句程序设为从上一状态切换到这个状态的切换条件(从表 1 中获得的预设值)与实际切换条件(可能出现误跳转即在只有部分条件满足后就发生了跳转)比较它们是否相等。如果程序未出现误跳转，显然会相等；反之，就会不相等。

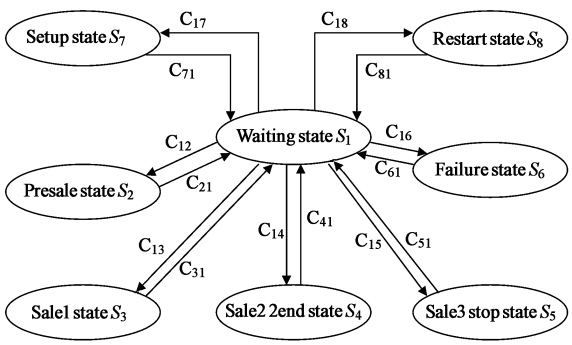
当它们不相等时，就引发一次切换，将程序从当前状态切换到冗余状态即故障处理状态，并分析状态信息包来获取故障的原因，并做相应的处理。

4　应用实例

以动售货机的主程序状态转换为例，在程序设计中，共有 8 种状态，系统处在其中一种状态中，并且状态间的切换由状态控制中心来实现。

4.1　自动售货机主程序设计

其状态模型的建立主要包括：NFA 建立、NFA 的转化及 DFA 的化简和状态切换条件表的建立。系统有 8 个状态$\{S_1, S_2, \cdots, S_8\}$，状态 S_i 编码为 i，各存在转换的状态间的切换条件及状态信息用状态信息包 $C_{ij}(i,j=1,2,\cdots,8)$来存储，就可以得到图 1 所示的 NFA 模型。



注：waiting state S_1－等待销售状态，presale state S_2－预销售状态，sale state S_3－销售状态，sale end state S_4－销售结束状态，sale stop state S_5－销售严重故障状态，failure state S_6－销售非严重故障状态，setup state S_7－设置状态，restart state S_8－重起状态。

图 1　自动售货机主程序状态转换

Fig.1　The state switch of main program in vending machine

将 NFA 模型转化为相应的 DFA，可得系统的 DFA，DFA 由成功销售、成功设置和故障(包括系统故障和部分功能性故障)组成，它们的状态转换流程分别如下：

成功销售——restart state $S_8 \xrightarrow{C_{81}}$ waiting state $S_1 \xrightarrow{C_{12}}$ presale state $S_2 \xrightarrow{C_{21}}$ waiting state $S_1 \xrightarrow{C_{13}}$ sale state $S_3 \xrightarrow{C_{31}}$ waiting state $S_1 \xrightarrow{C_{14}}$ sale end state $S_4 \xrightarrow{C_{41}}$ waiting state S_1

成功设置——restart state $S_8 \xrightarrow{1C_{81}}$ waiting state $S_1 \xrightarrow{C_{17}}$ setup state $S_7 \xrightarrow{C_{71}}$ waiting state S_1

故障——restart state $S_8 \xrightarrow{C_{81}}$ waiting state $S_1 \xrightarrow{C_{16}}$ failure state $S_6 \xrightarrow{C_{61}}$ waiting state S_1

或 restart state $S_8 \xrightarrow{C_{81}}$ waiting state $S_1 \xrightarrow{C_{15}}$ sale stop state $S_5 \xrightarrow{C_{51}}$ waiting state S_1

按照 3.1 节的方式，获得 $C_{ij}(i,j=1,2,\cdots,8)$的值，如表 2 所示。

表 **2** 自动售货机快速状态转移条件

Table 2 Condition of quick state switch in vending machine

Stc	S_8	S_1	S_2	S_3
Swc	C_{81} (81FFFH)	C_{12} (121H)	C_{23} (2311H)	C_{34} (3411H)
Stc	S_4	S_1	S_7	
Swc	C_{41} (411H)	C_{17} (173H)	C_{71} (710H)	
Stc	S_4	S_1	S_6	
Swc	C_{41} (411H)	C_{16} (16C18H)	C_{61} (610H)	
Stc	S_4	S_1	S_5	
Swc	C_{41} (411H)	C_{15} (153E7H)	C_{15} (510H)	

注：其中 $C_{23}=C_{21}C_{13}$，$C_{34}=C_{31}C_{14}$

4.2 故障状态替换

为了应对系统可能出现的故障，加入故障处理状态 S_5 和 S_6。在程序正常运行下成功销售或成功设置，当前状态在满足切换条件后，系统实现如下切换：重启状态 S_8→待机状态 S_1→设置状态 S_7→待机状态 S_1 或重启状态 S_8→待机状态 S_1→设置状态 S_7→待机状态 S_1。若在待机状态 S_1 检测出故障即 $C_{15}\neq$153E7H 或 $C_{16}\neq$16C18H，就将当前状态切换到故障处理状态 S_5 或 S_6。自动售货机中的故障包括螺旋电机故障、升降电机故障或霍尔传感器故障、销售超时、温度传感器采样故障、硬币机初始化超时、纸币机初始化超时。

初始化超时、CAN 通信故障、零钱不足、照明灯开关时间设置有误等(均来自实践)，显然这些故障有轻有重，设置了对轻故障和重故障区别处理的两个处理状态 S_5 和 S_6，S_5 处理严重故障，它包括螺旋电机故障、升降电机故障或霍尔传感器故障、硬币机初始化超时、纸币机初始化超时、CAN 通信故障、零钱不足等。S_6 处理轻微故障，它包括温度传感器采样故障、照明灯开关时间设置有误等。由于这些故障的标志变量存储在状态信息包中并且位置确定，就可以从中获取故障信息，并转入相应的故障处理程序中。

5 结论

自适应自动程序设计思想及其结构模型，是对返回值和异常报告程序出错的进一步探索和发展，它仅针对状态模型，其他模型是否可行尚待探索；此模型已初步应用(仅对故障做了报告和对部分故障做了修复)到自动售货机中，运行效果良好，准产品已交付企业。对故障的分类自动自修复，在很多领域都是非常重要的，有一定实用价值。

参考文献

[1] Kline M 著.现代世界中的数学[M]. 齐民友译. 上海：上海教育出版社，2001.448～464

[2] Richter J 著.Microsoft.NET 框架程序设计(修订版)[M]. 李建忠译. 北京：清华大学出版社，2003

[3] 刘晓东，等. 编译程序设计算法[M]. 成都：四川大学出版社，2006

[4] 徐凤生. 离散数学及其应用[M]. 北京：机械工业出版社，2006

[5] Malcher A. Minimizing finite automata is computationally hard[J]. Theoretical Computer Science, 2004,327(3-2):375～390

[6] Karakostas G, Lipton R J, Viglas A. On the complexity of intersecting finite state automata and NL versus NP [J]. Theoretical Computer Science, 2003,302(1-3):257～274

[7] 王 巍，高德远. 有限自动机设计策略[J]. 计算机工程与应用，1999,(7):54～55

[8] 黄燕芳，张玉清. 可生存性控制系统的有限自动机的设计[J]. 微电子学与计算机，2006,23(10):7～9

(下转第 181 页)