基于 HR 方法的解决 Y2K 的系统方案

李 莹, 李赣生
(浙江大学计算机科学与工程系, 杭州 310027)

摘 要: 本文详细阐述了采用混合基底(Hybrid Radix)方法在解决大型主机系统上的 COBOL 程序 Y2K 问题的系统框架和具体实现策略. HR 方法避免了传统方法中必须准确找出软件系统所有与年代有关数据空间的风险, 很大程度上提高了解决传代软件系统中 Y2K 问题的速度, 降低了转化代价.

关键词: 两千年问题; 混合基底; 滑动窗口; 传代系统; 软件重建工程
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2000) 05-0084-04

System Solution Based on HR Method to Y2K Problem

LI Ying, LI Garr-sheng
(*Department of Computer Science & Engineering, Zhejiang University, Hangzhou* 310027, *China*)

Abstract: The authors specify system architecture of Y2K solutions to COBOL programs in mainframe system with a new method based on Hybrid Radix, and present some concrete implementing strategies. HR method avoids the risks in some traditional methods, which require accurately digging out all of the data concerned with years. This method to a large extent improves the speed of Y2K solutions to legacy system, and reduces the conversion expense.

Key words: Y2K; hybrid radix; sliding windowing; legacy system; software reengineering

1 引言

随着 21 世纪的来临, 两千年问题(Y2K Problem, 简称 Y2K)越来越被人们所重视, 已经成为世纪之交软件危机的一大热点. Y2K 的产生有其历史原因, 在早期的计算机系统中, 硬件价格昂贵, 存储器由于技术上的原因容量很小, 在这种的环境上支撑的软件系统就必须仔细考虑如何有效地利用相对较小的内存来完成复杂的任务. 一个典型的节省存储器空间的例子就是将四位的年代用两位来表示, 比如 1965 用 65 来表示. 这样对于年月日显示比较简洁, MM/DD/YYYY 就可以用 MM/DD/YY 来表示. 这种表示方法逐渐被人们所接受, 形成了约定, 甚至最终成为 IEEE 的一个标准. 两位表示年代的方法, 不仅仅在软件系统中被广泛采用, 而且对于硬件系统也造成了相当大的影响. 两位表示的年代 YY 缺省认为是 19YY 年, 很显然, 到了 21 世纪就会产生混乱. YY 究竟是表示 19YY 还是表示 20YY 呢, 这就是 Y2K 问题之所在.

由于 Y2K 问题涉及的方面是如此广泛, 要想彻底解决, 就成为一个十分复杂的系统工程了, 软件和硬件系统的多样性, 使得想要找到一种简单而又通用的解决方案变为不可能. 但在一些特定的传代软件系统(Software Legacy System)上, 我们可能采用软件重建工程(Software Reengineering)的方法来解决. COBOL 语言是七十年代开始流行的商业语言, 以 IBM 为代表的大型主机系统(Mainframe System)对其广泛支撑. 据统计, 存在 Y2K 问题的软件传代系统大部分以商业应用语言 COBOL 为主, 在商业领域中年份的使用是至关重要的. 本文着重讨论如何在软件重建工程领域里采用 HR 方法、使用编译技术和程序理解的自动化方法, 来解决在主机系统上 COBOL 语言中的 Y2K 问题.

2 解决 Y2K 问题的混合基底(HR)方法

目前, 主要有两种解决 Y2K 问题的方法被广泛使用, 一种是传统的两位到四位转化的方法; 一种是已经被 IBM 公司采用的滑动窗口(Sliding Windowing)的方法.

混合基底(Hybrid Radix, 简称为 HR)的方法是由 Decao Mao 于 1997 年提出的一种新的解决 Y2K 的方法, 在全球范围内申请了专利. HR 方法主要适用在前端由 CICS 所支撑的大型主机系统上. 因为大部分传代软件系统都是在这种环境下运行的, 所以 HR 方法有广泛的适用性.

假设一个两位的年代变量为 X_1X_2, 如果高位 X_1 用十六进制表示, 而低位仍然采用十进制, 那么 X_1X_2 所能表示的范围就扩大到 0~159, 这是 HR 方法的核心思想. 那么, 对于最终的用户来说, 如何能够接受高位用 16 进制的表示呢? 本文定义 6 个新字符' 0、' 1、' 2、' 3、' 4、' 5, 分别与十六进制中的 A、B、C、D、E、F 对应. 当年代大于 1999 时高位就用这 6 个新的字符来扩充, 比如,' 18 就表示 2018 年. 这种表示方法比较直观, 实践证明一般用户可以接受. 使用 HR 方法以后, 两位变

收稿日期: 1999-07-09; 修订日期: 1999-11-10

量所能表示的空间从 1900~ 1999 扩展到 1900~ 2059.

两位的十进制变量在 IBM 大型主机系统上占有两个字节, 物理空间分配如下图所示.

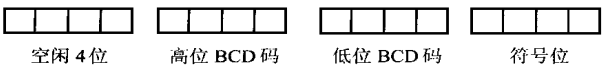

图 1　两位十进制变量 BCD 码的物理表示

从上图可以看出, 一位十进制数字的 BCD 码占有四个 bit, 实际物理空间所能表示的范围是 0~ 15, 比 0~ 9 大, 我们就可以有效地利用这些剩余空间来实现低位十进制, 而高位十六进制的 HR 方法.

混合基底的两位数字在进行算术运算时, 低位仍然采用十进制算术运算, 而高位则采用十六进制算术运算, 比如, ' 18 + 25= ' 43, 对于年代变量实际上就表示, 2018+ 25= 2043. 可以注意到, 非年代两位变量在源和目的都是两位的算术运算中, 采用了 HR 方法的运算规则以后, 与原来没有语义上的区别. 因为非年代两位变量经过算术运算以后, 不会产生与年代有关的相似的溢出, 否则, 与非年代变量有关的代码的语义本来就存在错误, 而不是由 HR 方法所造成. 因此, 对于所有两位数据空间进行 HR 变换并不影响程序原有的语义, 同时在一定程度上修正了 Y2K 问题. HR 方法要求找出的数据集合是可能与年代有关的数据, 可以包容非日期的数字. 这一特征与其它方法相比有无可比拟的优越性, 减少了原来需要通过分析源代码准确找出所有与年代有关的数据空间的代价. HR 方法的一个显而易见的缺点是, 只能将 Y2K 问题延迟到 2059 年, 到了 2060 年就又要面临新的问题. 但事实上, 在长达 60 年的时间里, 对这些传代系统有足够的时间彻底重构到新的系统上, 甚至到那时这些系统可能已经被遗弃, 本文下面着重论述如何采用 HR 方法解决 Y2K 问题, 给出在实现上的各种具体问题的解决策略.

3　用 HR 方法解决 Y2K 的几个重要问题

3.1　基本实现

下面针对 COBOL 语言讨论具体的实现细节. 在 COBOL 中, 两位的数字变量是用" PICTURE 99" 来描述的, 为了实现 HR 方法, 对于需要进行 HR 变化的两位变量(虽然, 理论上对于所有的两位变量都可以进行 HR 变换, 而不影响程序语义, 但是为了提高变换后程序的运行效率, 尽量保证不因为 HR 的引入而使原有的程序效率显著降低. 我们采用一些优化策略, 对那些肯定不是与年代有关的变量不给予 HR 变换. 但是要注意到, 剩余进行 HR 变换的变量的集合是真正与年代有关变量的超集), 对它使用 REDEFINE 重新定义类型, 同时还给它分配一个影子变量(Shadow Var) . 重新定义类型的主要目的是使代码的 I/O 层在 HR 变换前后保持一致, 不需要因为 HR 而改变标准 I/O 的界面和存储介质 I/O 的实际空间. 分配影子变量的主要目的是, 使得 HR 方法在 COBOL 中能够显式实现, 而不需要调用其它类型的语言, 并且使得变换后的代码易于优化, 尽量少调用 HR 变换子程序.

对于声明部分的处理举例如下:

原始代码: 10 A PIC 99

目标代码: 10 A PIC 99

10 A_ X REDEFINE A PIC XX

77 A_ SHADOW PIC 999

对于过程部分的引用点, 要进行相应的变化,

原始代码: ADD 10 TO A

目标代码: CALL HR99TO999 A A_ SHADOW

ADD 10 TO A_ SHADOW

CALL HR999TOXX A_ SHADOW A_ X

在上面的简单例子中, HR99TO999 和 HR999TOXX 是用 COBOL 语言实现的两个 HR 库函数, 用于 HR 变量和影子变量之间的数据转化. 所谓 HR 变量是指存储 HR 类型数据的, 需要对与其相关运算进行 HR 变换的变量. 影子变量是指与 HR 变量成对出现的, 并且始终与 HR 变量的语义在引用点前后保持同步变量. 在我们实现的 HR 支撑库中包含了一整套对各种不同长度变量进行处理的库函数.

上面仅仅谈到两位年代有关变量的处理, 在实际的源代码中还存在多位的年代有关变量, 比如, 包含生日信息的身份 ID 变量, 就是一个多位年代有关变量, 对于这种变量的处理可以采用相似的方法. 在无论几位的年代有关变量的实际存储空间中, 最高端半个字节的空间总是空闲的, 用这前半个空闲的字节标识出变量中表示年代的两位(各占半个字节) 相对于最左端的偏移量. 例如, 对于两位年代变量, 前半个字节是零; 对于 MMDDYY, 我们在最高的四位上标识出 4, 表示这个数据变量的年代位置是在左起第四个位置上(从零开始) . 这种在与年代有关变量的原有空间中表示出年代位置的方法, 给 HR 方法的实现提供了便利. 由用户标定的年代信息(注意, 尽管所有两位的年代有关变量可以全部采用 HR 变换, 不需要用户干预, 但是对于大部分多于两位的年代有关变量, 使用 HR 方法时, 仍然需要用户主动给出年代信息. 由于采用了数据流分析技术, 已经使得这种人工干预的操作风险尽可能地降低) 可以直接存储在原有的数据空间中, 也就是说, 不需要另外增加描述年代位置的空间. 当然, 在 COBOL 中还存在 PACKED、BINARY 等复杂的数据类型, 针对这些情况我们采用特殊的方法进行处理, 这里不再赘述.

总之, 对数字表示能力的扩展和在原有空间上标识出年代位置两个策略, 使得自动化转换传代系统的 HR 方法得以简单实现.

3.2　标准 I/O

在解决 Y2K 问题中, 不仅要使软件系统内部年份处理正确, 而且还要转化处理相应的软件用户界面等交互接口. 在两位到四位转化的方法中, 用户界面必须要重写; 在滑动窗口的方法中, 用户界面可保持不变, 但是在标准 I/O 层两位年份数字与原来的含义已经不同了, 一般要求用户清楚知道当前软件系统的轴心年. 比如, 82 表示 1982, 而 12 就可能表示 2012 了; 在 HR 方法中, 用户界面也不需要改变, 但用户要接受新的表示 2000 年以后年代的两位表示方法. 这里引入新字符, ' 0、' 1、' 2、' 3、' 4、' 5.

3.3　桥接(Bridging)

对于在全球范围内存在的 Y2K 问题,有不同的解决方法.采用不同方法解决 Y2K 问题的各个系统之间必然存在数据交换需求,这就引发新问题出现,即如何保证使用不同方法解决 Y2K 的系统之间仍然保证正确的数据交换.如在自动柜员机(ATM)上以滑动窗口改造的年份数据格式送到以 HR 方法构造的系统上处理前就须进行转化.因此,对于 HR 方法而言,存在与另外两种方法的接口问题:①与两位到四位变换方法的桥接,②与滑动窗口方法的桥接.

一般情况下,对于软件系统的桥接操作是针对某特定设备进行的.该设备可能是文件,也可能是通讯接口.所以在完成分析和转化软件源代码之前,就需要由用户来指定哪些设备要进行哪类桥接,这样在工具转化系统时,针对相应设备的输入输出进行特定的变换.

3.4 与数据库接口(Database Interface)

在大型传代软件系统中,使用数据库来管理海量数据.在 COBOL 程序与数据库之间存在着接口问题.考虑到:

(1) HR 数据在数据库中能否正确存储和运算

在某些情况下(比如,索引不能正确建立),HR 类型数据的出现可能导致必须进行数据库系统重构(如,早期版本的 UNISYS).但是在一般的主机系统上,IBM 的操作系统都不存在此问题.作为一项系统工程而言,在使用 HR 方法进行系统转化之前,先要对其数据库系统测试,评估其支持 HR 的能力,以此来判断使用 HR 方法的代价.

(2) HR 数据在 COBOL 主程序与数据库之间能否正确通讯

在 IBM 主机系统上的 COBOL 语言与数据库之间的操作是通过嵌入式.因此在处理与数据库的接口之前要先对嵌入式的数据库描述预编译.这样才能给 COBOL 主程序足够的信息,对数据库的接口进行必要的转换操作.

4 基于 HR 方法的 Y2K 解决方案的系统框架

4.1 解决 Y2K 的工程策略

对传代软件系统解决 Y2K 问题属于软件重建工程的范畴,是当前软件重建工程领域的一个典型.在解决 Y2K 问题过程中,我们可以总结并积累软件重建工程的经验和发展许多新的方法.Y2K 问题的转化策略按照时间顺序大致如下:

(a)评估应用系统,确定影响的程度

(b)根据对应用系统的研究执行工程计划

(c)定义出明确的转化范围

(d)使用工具辩识产生问题的具体区域

(e)使用逆向工程工具找要修改的代码,并修改之

(f)改善转化方法

(g)开发测试例程

(h)测试典型的工程应用

(i)使转化过程尽可能的自动化,最小程度地降低风险

在以上的步骤中,如何使转化过程尽可能地自动化和最大程度地降低风险是解决两千年问题的关键所在.而如何自动化这一转化过程是和对软件系统的程序理解(Program Understanding)密切相关,不可分割的.

4.2 系统构架

由于 PC 的迅猛发展,现在大多数主机系统的前端都是采用仿真终端,灵活性很强,这是 HR 方法实现的必要条件.我们在 Window98 下开发了解决 Y2K 问题的系统工具及其支撑环境.其中支撑环境包括两个部分,用于 HR 输入的键盘和各种 HR 数字的字体库.

工作流程如图 2,主要包括五个部分.

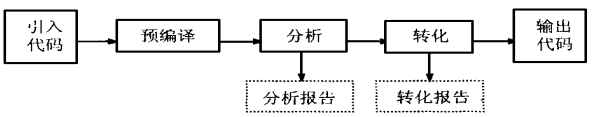

图 2 工作流程

(1)引入代码(Import Codes)

这一阶段要把传代软件系统的源代码引入到工具中,然后对其分类,划分各种不同类型的源代码文件,如 CBL(Main Program)、MAP、DMS、DDL、DDS 等等.分类的过程是通过分析文件,搜索不同类型文件的特征代码进行的,尽量避免人工干预.

(2)预编译(Precompile)

对分类好的某些辅助类型的文件(比如,MAP,DMS,DDL 等等),要在 COBOL 源码分析和转化之前,进行预处理,从中抽取对解决 Y2K 有关的全部信息,并且把这些有效信息以中间代码的形式分类保存在不同的文件中.

(3)分析(Analyze)

对预编译好的中间代码和主程序代码,进行 Y2K 分析,产生出许多辅助信息并保存在数据库中.同时产生一个详细的分析报告,其内容包括,警告信息、需要转化代码的统计信息,转化代价评估等等.这个报告要提交给用户评定.然后由用户决定是否要真正进行以 HR 为基础的系统转化.

(4)转化(Convert)

如果用户接受评估报告,那么就可以开始进行代码转化了.在转化之前,可以先浏览分析阶段产生的辅助信息,对其静态调整.这一步骤是关键,因为它涉及到转化产生的代码的效率和正确性.

有如下四个方面需要用户干预:

® 四位年代(CCYY,表示 Century & Year)

在某些特定的情况下,原有的软件系统中有一些年份变量可能已经用四位的数据空间表示出来了,所以对于这些变量就需要特殊处理.因为这类变量具有代表性,而且出现的可能性很大,所以要对其进行优化处理.在工具提供的可视化界面上,用户可以明确指定哪些变量是表示四位年代的,给以后的转化阶段提供了准确的信息.

® 优化(Optimize)

通过分析源代码,在一定程度上的语义理解,此工具可以找出那些由操作系统的时间环境变量传递得到的数据空间的集合,在此集合内的所有变量被认为是必须要进行变换的,是不能被优化的.另外,此工具还提供给用户一个人工干预的手段,对于那些具有明显名字特征的,可以人工判断出肯定不是年代有关的变量,可以加上将要被优化的标志.在以后的转化

过程中, 这些变量就不会进行代码转化了, 被系统忽略. 注意, 这种操作与挑选年代有关的两位变量是完全不同的. 因为在剔除了认为肯定不是年代有关的变量以后, 剩于变量的集合是与年代有关的所有变量集合的超集. 因此, 使用 HR 方法, 不会导致转化后系统语义严重出错, 在很大程度上降低了风险.

® 数据流(Data Flow) 分析

在"优化"中出现的所有变量都是两位的, 在实际的软件系统中与年份有关的变量不仅仅限于两位. 大体有三种情况: 其一, 包含两位年代变量的整体数据空间的使用(比如, 在一个记录结构中有一个两位的年代变量, 而使用整个访问记录); 其二, 某一特定类型的包含年份信息的数据变量是整体使用的, 但是通过一种特别的操作(比如, 乘以 0. 0001), 可能将年份信息传递给其它的两位变量. 其三, 一些系统环境变量也是和年份有关的. 从这些变量开始传递出来的所有的变量都可能是与年份有关的. 所以在通过数据流的分析之后, 可以形成一个与年份有关的变量的传递闭包. 对其中的每一个闭包都可以有用户指定是否对其优化. 在此图中用户可以清楚地看到闭包中变量之间的传递关系, 提供了给用户判断的依据.

® 桥接(Bridging)

Bridging 操作是根据不同的设备和文件进行的.

在对这些辅助信息经过调整以后, 就可以开始自动的转化过程. 这个使用 HR 方法的工具对于用户策略是面向数据的, 而不是面向过程的. 因此减少了用户分析过程语义的负担. 工具本身会把用户的辅助调整信息和源代码做为输入, 使用以编译技术为基础的程序理解, 分析产生最终转化结果. 这一过程是高度自动化的, 极大地减轻了用户转换过程的负担, 是其它方法所无法比拟的.

(5) 输出代码(Export Codes)

在对传代系统应用软件的源代码转化以后, 需要输出转化结果, 对于其他的辅助文件如, MAP, DLL, DMS, 一般情况下不需要对其转化. 但是在某些特殊的系统环境(如 UNISYS) 下, 可能需要人工干预, 对 DMS 进行系统重构.

5　小结

本文阐述了如何使用 HR 方法解决 Y2K 问题, 以及在解决 Y2K 问题中应该考虑的一些重要问题. 我们实现的工具有两个版本, 一个是在 Linux 下, 一个是在 Window98 下. 经过对两个系统转化的实际应用, 证明了这种方法的可行性, 同时显示出了这种方法的高效性. 这种方法给用户提供了面向数据的修改手段, 而真正的转化过程是由机器来完成的, 因此极大的减少了改造以后系统出错的概率. 在我们的实现中除了采用传统的编译技术以外, 还使用了人工智能中的一些方法, 使系统在给定的规则下能够更准确的找出不能被优化的数据空间, 给用户提供更加可靠的依据. 我们的工作具有时间的紧迫性, 在某些方面可能还需要更进一步的研究. Y2K 解决工具的实现对于我们在软件重建工程领域中探索新方法、新思路提供了有力的实践, 对于软件传代系统的程序理解领域我们还需要做更多的工作.

致谢　本文得到了美国 Stepwise Solutions 公司的支持, 他们为本系统开发的成功提供了有力的赞助, 特此表示感谢.

参考文献

[1] Mao Decao. Two-digit Hybrid radix year numbers for year 2000 and beyond. United States Patent 5668989, Sept. 16, 1997

[2] 毛德超、李赣生. 一个解决两千年问题的新方法. 电子学报, 1999, 2: 139~141

[3] Hausi A. Müller. Understanding Software Systems Using Reverse Engineering Technologies Research and Practice. the 18th International Conference on Software Engineering, 1996

李　莹　1973 年生, 博士生. 1994 年毕业于浙江大学计算机科学与工程系. 1997 年获浙江大学计算机硕士学位. 主要研究方向为软件工程、编译技术、软件自动化等.

李赣生　1940 年生, 浙江大学计算机科学与工程系教授, 博士生导师. 1963 年毕业于武汉大学数学系. 主要领域: 编译技术, 程序设计、语言设计与实现, 形式化方法, 软件自动化, 软件工程等.