ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.4, April 2007, pp.808−820
DOI: 10.1360/jos180808

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

错误流模型:硬件故障的软件传播建模与分析*

杨学军, 高 珑+

(国防科学技术大学 计算机学院,湖南 长沙 410073)

Error Flow Model: Modeling and Analysis of Software Propagating Hardware Faults

YANG Xue-Jun, GAO Long+

(Computer School, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4575808, E-mail: imgaolong@yahoo.com.cn

Yang XJ, Gao L. Error flow model: Modeling and analysis of software propagating hardware faults. *Journal of Software*, 2007,18(4):808−820. http://www.jos.org.cn/1000-9825/18/808.htm

Abstract: Neither reliability models in reliability engineering nor in software reliability can be directly applied to describe the propagation of hardware errors in programs. This paper first sets up a computational data flow model, and then explains that a computational data flow graph for the program can be built, using the instruction set of URM (unlimited register machine) as an example. Upon the computational data flow model, the error flow model is set up. Errors are categorized into two kinds: Original errors and propagated errors. By analyzing the propagation rules of these two kinds of errors, 6 assumptions about error propagation are given, upon which the probabilities of errors at any time and at any place in a program can be calculated. At last, a sample of URM program is given to demonstrate the capability of the fault flow model.

Key words: SIHFT (software implemented hardware fault tolerance); reliability; computational data flow model; error flow model; error propagation

摘 要: 无论是可靠性工程还是软件可靠性中的可靠性模型,都难以描述硬件故障在程序中的传播问题.首先建立了计算数据流模型,并以无穷存储机器的指令集为例,说明可以为任意程序建立计算数据流图.在计算数据流模型的基础上,进一步建立了错误流模型.把计算过程中的错误分成物理错误和传播错误两种,通过分析这两种错误的本质和传播规律,给出了 6 条有关错误传播的规则和 2 条独立定律.根据这些规则和定律,能够计算出在程序运行过程中,任意时刻在任意位置上出现错误的概率.最后以一个简单的无穷存储机器程序为例,简要地展示了错误流模型描述硬件故障在程序中传播的能力.

关键词: 软件实现的硬件容错;可靠性;计算数据流模型;错误流模型;错误传播
中图法分类号: TP311 文献标识码: A

对于机械、电子等硬件系统的可靠性,我们可以利用传统的可靠性工程的原理和技术来解决.例如,在可靠性工程中,描述系统可靠性的模型有串联模型、并联模型、表决模型、储备模型、一般网络模型等[1].

* Supported by the National Natural Science Foundation of China, the National Science Fund for Distinguished Young Scholars of China under Grant No.60621003 (国家自然科学基金创新研究群体)

Received 2006-06-12; Accepted 2006-08-29

软件可靠性是指软件系统在特定的环境(条件)下,在给定的时间内不发生故障工作的概率[2].各种软件可靠性模型,如非齐次泊松过程(non-homogeneous Poisson process,简称 NHPP)[3,4]等,主要用来研究软件本身的错误.

软件可靠性研究的是软件本身的设计错误,通常认为软件运行在没有错误的理想硬件上.软件可靠性没有考虑在硬件存在错误的情况下,因为软件而给计算机系统带来的可靠性问题.显然,我们也不能依靠描述硬件系统可靠性的模型来描述软、硬件共同带来的计算机系统可靠性问题.

事实上,无论程序是否完全正确,如果在硬件中存在故障,运行在这些硬件上的程序都会给计算机系统最终的可靠性带来变数.不同的程序对于完全相同的硬件故障,也可能会有不同的表现:有的程序会崩溃,有的可能会正常结束但却产生错误的结果,有的程序则可能毫无影响地得到完全正确的结果.

如果计算机中的数据不进行任何计算,那么,这些数据发生错误的概率就仅仅与存储它们的硬件特性以及外界环境有关,我们可以测量和推算出任何数据发生错误的概率.然而,当数据之间由于程序中的计算而产生了联系之后,情况就完全不一样了.程序中的计算会传播错误,从而导致错误在数据与数据之间传播.大部分错误在计算的过程中被传播到计算的结果中;有些错误则被计算屏蔽掉了.对于任意程序,不论其是否达到设计目的,我们都应该能够根据程序本身判断该程序传播错误的特点.如果程序不容易传播错误,那么它对系统可靠性的影响就应该小;相反地,如果程序很容易把错误传播到计算结果中,那么这个程序对系统可靠性的负面影响就大.在未来的研究中,我们倾向于把程序传播硬件故障的这种有关计算机系统可靠性的特点总结为程序的容错能力,这将在我们随后的文章中有所体现.

本文先通过研究软件传播硬件故障的规律,力图建立一个可以描述硬件故障在程序中传播规律的错误流模型.本文第 1 节介绍相关研究.第 2 节建立计算数据流模型.第 3 节说明如何为任意程序建立计算数据流图.第 4 节在计算数据流模型上建立错误流模型.第 5 节利用错误流模型分析错误在程序中的传播规律.第 6 节进行了总结.第 7 节展望了未来的工作.

1 相关研究

1.1 瞬时故障

故障(fault)是软、硬件产生不符合其设计目的的差异;当这种差异导致计算机偏离正确的状态时,就产生了错误(error)[5],如图 1 所示.故障有不同的分类方法,大致可以分为 4 种:物理原因导致的硬件故障,软、硬件的设计失误,人为的操作故障和人为的故意攻击等[6].硬件故障又可以分为瞬时、间歇和永久故障.瞬时故障发生后大都可以自动消失.SEU(single event upset)[7]是瞬时故障的典型模式.宇宙射线中的带电粒子轰击集成电路造成瞬时充、放电,从而使存储单元的逻辑状态发生翻转[8],就产生了 SEU.在航天领域中遇到的大部分故障都可以归结为 SEU 类型的瞬时故障.瞬时故障是永久性故障数量的 100 倍以上,占故障的绝大多数[5].一般认为,瞬时故障发生的同时,错误立即发生[5].本文研究的方法主要针对瞬时故障,所以在本文中对于错误和瞬时故障不加以细致的区分.

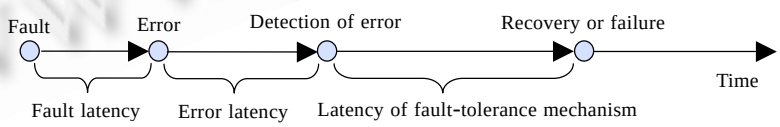

Fig.1 Fault model

图 1 故障模型

1.2 程序对硬件故障传播的影响

不会有人反对这样的命题:使用错误的数据计算得到的结果很可能也是错误的.这说明在计算过程中,由于先写后读而带来的数据相关对于错误的传播具有重要的影响:错误会被计算传播.然而错误的传播还与计算的类型有关,大部分错误会直接沿着计算传播下去;但是有些计算会屏蔽错误的传播,比如 $a=b\&0xff00$ 会屏蔽存在

于 b 中低 8 位的错误传播到 a,$c=d<e$ 会屏蔽任何满足 d 小于 e 的错误传播到 c.

1.3 软件实现的硬件容错

在软件可靠性的研究中,以设计相异性(design diversity)为理念的软件容错技术,如多版本编程(n version programming,简称 NVP)[9]和恢复块技术(recovery block)[10]等,主要目的是通过软件冗余来屏蔽存在于软件中的错误,以提高软件的可靠性.

而另外一类软件实现的硬件容错技术(software implemented hardware fault tolerance,简称 SIHFT)[11]主要使用软件的方法来屏蔽发生在硬件中的故障.这主要是针对航空航天领域中的瞬时错误.因为即使软件编写得完全没有错误,也要考虑因为宇宙射线等因素引起的硬件瞬时故障对空间计算机的影响.尤其是在高性能、低价格和低功耗的 COTS 器件上,通过软件实现的硬件容错可以在保证可靠性的同时,以较低的功耗获得较高的性价比[12].基于算法的容错(algorithm based fault tolerance,简称 ABFT)[13]通过算法中的冗余数据和冗余计算检测、定位甚至纠正硬件故障.但是,由于需要精心设计算法,所以应用范围受到限制.容错编译[14−16]由于实现方法简单、高效,不必重写代码,因而成为软件实现的硬件容错中比较活跃的一个分支.但是,由于容错编译时插入的指令数量和功能有限,所以一般的研究集中于检测错误,检测到错误后一般重新启动出现错误的任务,或者回滚到上一个检查点.

在现有的软件实现的硬件容错方法中,对于发生在硬件中的瞬时故障如何通过程序中的计算进行传播的问题,以及有关模型的研究还很少.本文提出了一种可以描述错误在程序中通过计算传播的错误流模型,并且期望在此基础上能够发展一种软、硬件协调的提高系统可靠性的方法.

2 计算数据流模型

我们把每一条指令抽象为两部分内容:原子数据和原子计算.

2.1 原子数据

定义 **1**. 原子数据是由特定的存储单元记录、在指令中不可再分的数据.

原子数据包括在指令的操作数中能够出现的各种寻址类型的数据.比如浮点寄存器 f_1 中的数据、内存地址 0*xfff*02*ef*9 中的数据等.

定义 **2**. 对于指令 i 以及由任意指令序列构成的程序 P,我们定义指令 i 中原子数据的集合 $V(i)=\{x|x$ 是指令 i 中的原子数据$\}$,程序 P 中原子数据的集合 $V(P)=\{x|x\in V(i),i\in P\}$,指令 i 的源 $S(i)=\{x|x\in V(i),$在指令 i 中 x 被读取$\}$,指令 i 的目标 $D(i)=\{x|x\in V(i),$在指令 i 中 x 被写入$\}$.

同一个存储单元中的数据,在程序执行的过程中可能被多次重写,那么它所代表的原子数据也就不一样.为了标志不同原子数据在时间上和空间上的唯一性,我们为存储单元的名称增加上标来表示某时刻在该存储单元中存放的原子数据,如 f_1^2 .我们规定,在执行指令的过程中,如果存储单元的内容被改写,那么相应原子数据的上标加 1.例如,存储单元 f_1 中的原子数据 f_1^m 被重写,则新的原子数据为 f_1^{m+1} .我们规定原子数据的初始上标为 0.为了避免歧义,在原子数据名称的两侧添加没有前缀的括号来表示该原子数据的值,如(f_1^{m+1})表示原子数据 f_1^{m+1} 的值.

2.2 原子计算

定义 **3**. 原子计算是由原子数据参与的、由一条指令完成的计算.

不同的原子计算对于传播错误的影响是不同的.某些计算能够屏蔽错误的传播,比如 $a=b\&0xff00$ 会屏蔽存在于数据 b 中低 8 位的错误传播到 a,$c=d<e$ 会屏蔽任何满足 d 小于 e 的错误传播到 c.另外,如果加法中一个加数大,一个加数小,则其和有可能最终正确;然而在除法中,如果被除数大,除数小,则其结果一定错误.实现相同逻辑功能的硬件对于传播错误的贡献是类似的.这反映了一种抽象的计算在逻辑上对于错误传播的影响,这一部分影响并不会因为硬件实现的不同而发生改变.

2.3 计算关系

定义 **4**. 对于指令 i 以及任意指令序列构成的程序 P,设 $V(i)$是指令 i 的原子数据集合,$V(P)$是程序 P 的原子数据集合,$S(i)$是指令 i 的源集合,$D(i)$是指令 i 的目标集合.那么我们定义 $V(i)$之间的计算关系为 $E(i)=\{\langle x,y\rangle|x\in S(i),y\in D(i)\}$,$V(P)$之间的计算关系为 $E(P)=\{x|x\in E(i),i\in P\}$.

指令的源到目标都存在计算关系.某条指令的目标,可能是下一条指令的源.

计算关系对于研究错误如何在程序中传播的意义在于:错误以一定的可能性沿着计算关系从源传播到目标,而不是反方向;源的错误和目标的错误以一定的可能性存在逻辑上的因果关系.

我们从源到目标画一条带有方向的弧表示计算关系,如图 2 所示.计算关系可以表示原子计算的某些特点,例如参与计算的数据个数、计算发生的方向和作用域等.图 2(a)可以看作赋值计算,图 2(b)既可以看作是乘法,也可以看作是其他二元计算.如果源或者目标处于我们的研究范围以外,则使用双横线略写原子数据,以表示我们研究领域的边界.比如,当我们只研究寄存器类型的存储单元时,在内存中的原子数据就可以使用双横线略写.图 2(c)和图 2(d)分别表示 Load 和 Store 指令.当我们不关心计算的具体种类时,可以不标出具体指令的种类.

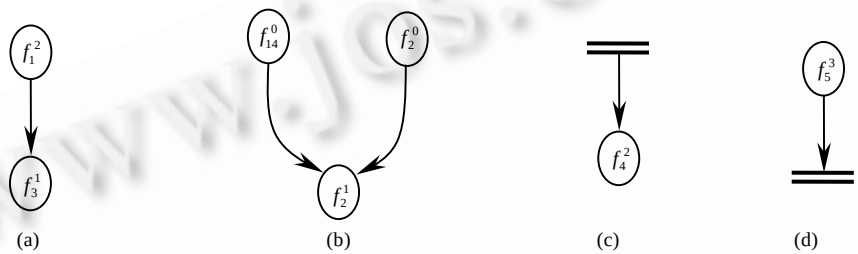

Fig.2 Atomic data and computational relation

图 2 原子数据和计算关系

2.4 执行概率

由于条件分支指令的存在,在程序 P 的某一次运行过程中,某些指令可能被执行,也可能不会被执行.因而与这些指令相关的计算关系也以指令执行的概率成立.必然执行的指令的执行概率为 1.

定义 **5**. 计算关系 E 的执行概率 B 是从 E 到[0,1]上的函数 $B:E\rightarrow[0,1]$,表示该计算关系成立的概率.

我们在表示计算关系 e 的有向边上标注该计算关系的执行概率 $B(E)$,表示该计算关系成立的概率,如图 3 所示.图中左边的程序为 SimpleScalar/PISA 指令[17],假定分支成功的概率为 85%,失败的概率为 15%.

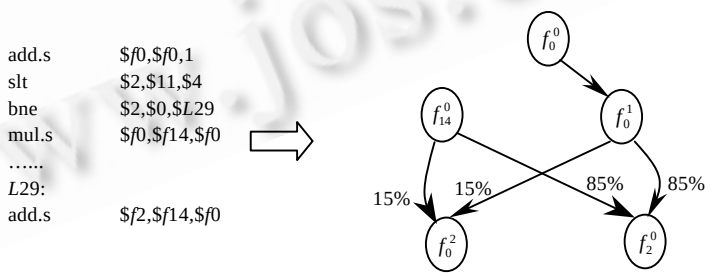

Fig.3 Conditional branch

图 3 分支指令

2.5 计算数据流模型和图

计算关系能够表示错误在计算中传播的特征.在本节中,我们以原子数据和计算关系为基本要素构建计算数据流模型.

定义 **6**. 我们称三元组$\langle V,E,B\rangle$为计算数据流模型.其中,V 是原子数据集合,E 是 V 之间的计算关系,B 是定义

在 E 上的执行概率.

定义 **7**. 如果$\langle V,E,B\rangle$为计算数据流模型,我们就称以 V 为结点、以 E 为边的图 $G(V,E)$为计算数据流图,并且 $G(V,E)$中每一条有向边 $e\in E$ 以 $B(e)$为加权.

定义 **8**. 对于任意程序 P,如果 $V(P)$是程序 P 的原子数据集合,$E(P)$是 $V(P)$之间的计算关系,B 是定义在 $E(P)$上的执行概率,那么我们定义以 $V(P)$为结点、以 $E(P)$为边的图 $G(V(P),E(P))$为程序 P 的计算数据流图,记为 $CDF(P)$.

定义 **9**. 在计算数据流图 $G(V,E)$中,如果存在一条从结点 u 到结点 v 的路径,那么称 u 在 v 的上游,v 在 u 的下游.

3 计算数据流图的建立

考虑到每一个程序都可以表示成由基本块组成的程序流图,而在基本块内部,所有的语句都将以相同的概率执行,因而我们将以基本块为单位建立整个程序的计算数据流图.

3.1 指令的分类

我们把指令分为控制类指令和计算类指令.控制类指令是指那些能够引起指令执行顺序改变的指令,包括无条件转移指令和条件转移指令.计算类指令是指除了控制指令之外,执行算术和逻辑运算的指令.一些特殊的协处理指令也包括在其中,同时还包括一些比较复杂的函数指令,如正弦或余弦函数等.

3.2 根据程序流图建立计算数据流图

任意程序都可以表示成具有唯一首结点的程序流图[18],程序流图中每一个结点都表示一个基本块.只要我们为每一个基本块建立了计算数据流图,就可以递归地建立整个程序的计算数据流图.

我们通过在主程序 *Main*(*P*)中调用递归过程 *Build*(*G*,*s*,1,*Read*[])建立程序 P 的计算数据流图,如图 4~图 6 所示.参数 G 是程序 P 的程序流图,s 是 G 的首结点基本块编号,*Read*[]是一个传递数值的数组参数. *Build*(*G*,*s*,1,*Read*[])在遍历 G 的过程中建立程序 P 的计算数据流图.由于基本块内所有语句要么都执行,要么都不执行,所以基本块内的所有计算关系的加权值相同.数组 *Read*[]和数组 *Write*[]都以存储单元 r_i 的标号 i 作为索引,其中,*Read*[]是局部变量,*Write*[]是全局变量,如图 4 中的声明所示.算法中,*P*(*cond*)表示条件 *cond* 为真的概率.在编译程序时,我们一般可以采用静态预测的方法[19−21]预测程序中条件分支指令成功和失败的概率.*P*(*cond*)的值就等于包含条件 *cond* 的条件分支类指令成功的概率.

如果程序 P 中存在循环,也就是程序流图 G 中存在具有唯一入口的强连通子图,那么我们仅为每一个基本块建立一次计算数据流图:在 *Build*(*G*,*b*,*p*)中只为没有标记过的基本块建立计算数据流图,也就是仅遍历 G 一遍.对于循环中的错误传播问题,我们将在后面的章节中讨论如何利用循环的周期性迭代地求出错误传播的概率.

```
GLOBAL DECLARE
BEGIN
    Write[]={0,…,0}
END
```

Fig.4 Global declare

图 4 全局变量声明

```
PROCEDURE Main(P)
INPUT: Progam P;
OUTPUT: Computational data flow CDF(P) of progam P.
BEGIN
    FOR (each memory unit r_i) DO Read[i]←0
    Setup program graph G for P,s is start node of G
    Build(G,s,1,Read[])
END
```

Fig.5 Function *Main*(*P*)

图 5 *Main*(*P*)函数

```
PROCEDURE Build(G,b,p,Read[])
INPUT: Program Graph G, basic block b, Real p between 0 and 1, Real queue Read[];
OUTPUT: Computational data flow of G
BEGIN
    Mark basic block b in G
    m←start instruction number of b; n←end instruction number of b
    FOR (each instruction j from m to n) DO
    BEGIN
        SWITCH (type of j) OF
        BEGIN
            CASE computing:
                FOR (each memory unit r_u read in j) DO
                BEGIN
                    k←Read[u]
                    FOR (each memory unit r_v written in j) DO
                    BEGIN
                        l←Write[v]+1;Write[v]←l;Read[v]←l
                        draw atomic data r_v^l; draw an arc from r_u^k to r_v^l with weight p
                    END
                END
            CASE unconditional branch:
                c←b
                IF (c∈G AND c is not marked in G) THEN
                    Build(G,c,p,Read[])
            CASE conditional branch:
                c←target basic block of taken branches
                d←target basic block of not taken branches
                cond←test condition in j
                p1←p·P(cond);p2←p·(1−P(cond))
                IF (c∈G AND c is not marked in G) THEN
                    Build(G,c,p1,Read[])
                IF(d∈G AND d is not marked in G) THEN
                    Build (G,c,p2,Read[])
        END
    END
END
```

Fig.6 Function *Build*(*G*,*b*,*p*,*Read*[])

图 6 *Build*(*G*,*b*,*p*,*Read*[])函数

3.3 URM指令集

URM(unlimited register machine)是与图灵机等价的计算模型[22].URM 的存储器是一端趋于无穷的带子,如图 7 所示,存储单元记为 $R_1,R_2,\ldots$,记 R_i 中的数为 r_i.URM 指令集中包含 4 条指令,利用这 4 条指令,URM 可以计算一切图灵机可以计算的递归函数.这 4 条指令分别是:

零指令 $Z(n)$:把存储单元 R_n 的内容 r_n 变为 0,其他存储单元内的内容不变;

后继指令 $S(n)$:把存储单元 R_n 的内容 r_n 加 1,其他存储单元内的内容不变;

传送指令 $T(m,n)$:用存储单元 R_m 的内容 r_m 替换存储单元 R_n 的内容 r_n,其他存储单元内的内容不变;

转移指令 $J(m,n,q)$:如果存储单元 R_m 的内容 r_m 和存储单元 R_n 的内容 r_n 相等,执行编号为 q 的指令;否则顺序执行下一条指令.

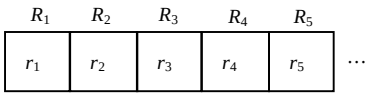

Fig.7 Registers of URM

图 7 URM 的存储单元

3.3.1 为 URM 程序建立计算数据流图

以计算 $x+y$ 的 URM 的程序为例,说明我们可以为程序建立计算数据流图.我们编制计算 $x+y$ 的 URM 程序

如图 8 所示[22].其中,x 和 y 分别已经存放在 R_1 和 R_2 中,我们不断地给初值为 0 的 R_3 加 1,同时给 R_1 加 1,直到 R_2 和 R_3 的值相等.这时,R_1 中的内容就是 $x+y$ 的值.冒号之前的数字为该指令的标号.

图 9 中灰色的方框和实箭头表示计算 $x+y$ 的 URM 程序的程序流图.基本块 1 包含指令 1,基本块 2 包含指令 2~指令 4.图 10 表示由图 9 中的程序流图生成的计算数据流图.每条有向边的加权值为 $P(r_2\neq r_3)=1-P(r_2=r_3)$.我们可以按照文献[19–21]中的编译时算法估计 $P(r_2\neq r_3)$的值.

我们仅表示了一次循环迭代的数据流图.有关多次循环的情况将在后面的章节中继续讨论.

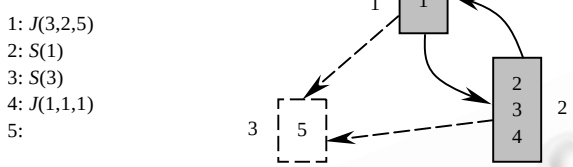

Fig.8 URM program of $x+y$

图 8 计算 $x+y$ 的 URM 程序

Fig.9 Program graph

图 9 程序流图

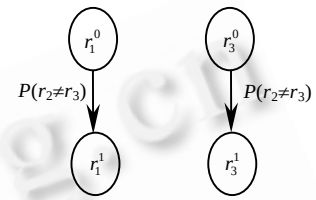

Fig.10 Computational data graph

图 10 计算数据流图

4 错误流模型

定义 **10**. 如果三元组$\langle V,E,B\rangle$为计算数据流模型,我们称四元组$\langle V,E,F,\hat{f}\rangle$为错误流模型.其中,$F$ 是从原子数据集合 V 到[0,1]上的函数 $F:V\to[0,1]$,表示原子数据发生错误的概率,$\hat{f}$ 是从计算关系集合 E 到[0,1]上的函数 $\hat{f}:E\to[0,1]$,表示计算关系传播错误的概率.

如果 $f(e)$表示当计算关系 e 成立时计算关系传播错误的概率,而 $B(e)$是表示计算关系 e 成立的执行概率,根据条件概率的含义,$\hat{f}(e)=f(e)\cdot B(e)$.后面我们将会介绍 $f(e)$在不同类型的计算关系中的定义和形式.

定义 **11**. 如果$\langle V,E,F,\hat{f}\rangle$为错误流模型,我们就称以 V 为结点、以 E 为边的图 $G(V,E)$为错误流图,并且 $G(V,E)$中每一个结点 $v\in V$ 以 $F(v)$为加权,每一条有向边 $e\in E$ 以 $\hat{f}(e)$ 为加权.

定义 **12**. 对于任意程序 P,如果 $V(P)$是程序 P 的原子数据集合,$E(P)$是 $V(P)$之间的计算关系,B 是定义在 $E(P)$上的执行概率,那么我们定义以 $V(P)$为结点、以 $E(P)$为边的图 $G(V(P),E(P),F(V(P)),\hat{f}(E(P)))$ 为程序 P 的错误流图,记为 $EF(P)$.

5 基于错误流图的错误传播分析

我们已经能够建立任意程序 P 的错误流图,下面我们将研究错误如何在程序中传播.

5.1 错误的分类

我们把导致一个存储单元错误的原因分为两种:物理错误和传播错误.

物理错误是由于物理原因诱发的错误,比如由于高能带电粒子的轰击,使存储单元以及附近电路发生瞬时充、放电,从而导致存储单元保持的逻辑状态发生变化.传播错误是由于在计算中使用了错误的数据,从而导致的计算结果错误.这两种错误可能叠加.叠加的效果可能会保持错误,也可能最终使错误消失.比如传播错误使原子数据的某一位从 1 变成 0,而物理错误又使得该位从 0 翻转回 1,这时,叠加的效果就是消除错误.

物理错误的概率可以通过仪器精确测量出来,而计算关系传播错误的概率是与程序中计算关系密切相关的.当我们研究其中一种错误的特性时,必须排除另一种错误的影响.

5.2 错误传播的规则和定律

在计算存在的情况下,错误之间强烈地受到因果律的支配.这里,我们给出 6 条错误在错误流图中传播的规则和 2 条独立定律,利用这些规则和定律,我们就能够定量地研究原子数据发生错误的概率.

5.2.1 传播律

传播律. 原子数据的传播错误仅来源于上游原子数据的错误.

这是显然的,在错误的传播中我们不能颠倒因果.上游错误是下游错误的原因,下游错误是上游错误的结果,而不可能是相反的.由于我们的错误流图中是没有环路的,因而传播错误只能从错误流图的上游获得,而不可能来源于错误流图的下游.

5.2.2 组成律

组成律. 原子数据的错误仅来源于物理错误或者传播错误.

按照物理错误和传播错误的划分方法,原子数据的错误只能是这两种错误,我们不可能再找出第 3 种.

5.2.3 本源律

本源律. 原子数据的传播错误归根结底来源于上游原子数据的物理错误.

错误归根结底起源于物理错误.如果没有任何物理错误,那么也不可能存在任何传播错误.传播错误只是物理错误在传播过程中的一种表现形式.

5.2.4 叠加律

叠加律 **1**. 如果一个原子数据仅发生物理错误而没有传播错误,那么该原子数据最终一定错误.

叠加律 **2**. 如果一个原子数据仅发生传播错误而没有物理错误,那么该原子数据最终一定错误.

叠加律 **3**. 如果一个原子数据同时发生物理错误和传播错误,则该原子数据既可能错误,也可能正确.

第 3 种情况是因为错误的叠加可能会导致错误消失.

5.2.5 独立定律

独立定律 **1**. 原子数据是否发生物理错误与该原子数据是否发生传播错误是无关的.

证明:如图 11 所示,原子数据发生传播错误的事件用深色小圆表示,原子数据发生物理错误的事件用浅色小正方形表示,虚线椭圆为任意原子数据发生错误的事件.定义关于任意事件 A 的二值随机变量 $X=\begin{cases}1, & A\text{发生}\\ 0, & A\text{不发生}\end{cases}$.设 X_1 到 X_n 是关于上游原子数据物理错误的二值随机变量,X_0 和 Y_0 分别是关于目标 v 的物理错误和传播错误的二值随机变量.根据本源律,Y_0 由 $X_1,\ldots,X_n$ 决定,所以 Y_0 是 $X_1,\ldots,X_n$ 的函数 $Y_0=\xi(X_1,\ldots,X_n)$,ξ是某一函数.又因为物理错误 $X_0,X_1,\ldots,X_n$ 为各自独立发生的物理错误事件,根据概率论中有关相互独立的随机变量的任意函数与其中任何一个随机变量独立的结论,有 X_0 和 $Y_0=\xi(X_1,\ldots,X_n)$之间独立.亦即独立定律 1 得证. □

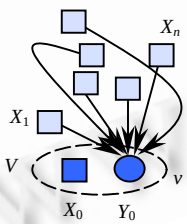

Fig.11 Independence principle 1

图 11 独立定律 1

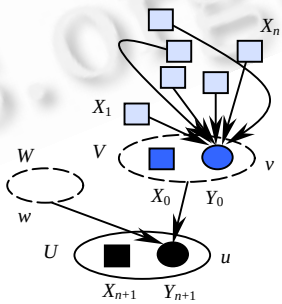

Fig.12 Independence principle 2

图 12 独立定律 2

独立定律 **2**. 原子数据的物理错误是否发生和位于该原子数据上游的原子数据的错误是否发生是无关的.

证明:如图 12 所示,设 V 是关于上游某个原子数据 v 错误的二值随机变量,X_0 和 Y_0 是关于 v 的物理错误和传播错误的二值随机变量.U 是关于目标 u 错误的二值随机变量,X_{n+1} 和 Y_{n+1} 是关于目标 u 物理错误和传播错误的二值随机变量,根据组成律和本源律,可得 $V=\zeta(X_0,Y_0)=\zeta(X_0,\xi(X_1,\ldots,X_n))=\varsigma(X_0,X_1,\ldots,X_n)$,其中,$\xi$,$\zeta$和$\varsigma$都是某一函数.又因为物理错误 $X_0,X_1,\ldots,X_n,X_{n+1}$ 为各自独立发生的物理错误事件,类似独立定律 1 中的证明,所以 X_{n+1} 和

$V=\varsigma(X_0,X_1,\ldots,X_n)$之间独立.所以独立定律 2 得证. □

同理,如果设 W 是关于另一个上游原子数据 w 错误的二值随机变量,那么 X_{n+1} 和 $W,W\cdot V,(1-W)\cdot V$ 以及 $W\cdot(1-V)$都独立.我们注意到,与随即变量 $1-W$ 和 W 相关的事件互为补事件,我们会用到这些结论.

5.3 错误传播概率的计算

我们把错误事件化:取一次原子计算的过程为一次实验,而把原子数据的错误看作事件.错误事件包括源错误和目标错误.源和目标的错误之间通过计算关系产生因果关系.

源 u 的错误通过计算关系 e 传播到目标 v 的概率,就是计算关系 e 传播错误的概率 $f(e)$.当然,此时我们要避免目标 v 发生物理错误.因为目标 v 的物理错误与源 u 的错误无关,也就不应该算作计算关系 e 传播的错误.我们将要通过源错误的概率 $F(u)$和计算关系传播错误的概率 $f(e)$计算出目标的错误概率 $F(v)$.先计算当计算关系一定存在时的情况,也就是执行概率 $B(e)=1$ 时的情况,随后讨论 $B(e)\neq1$ 的情况.

在进行计算之前,我们先证明两个引理.假定本文中所有条件概率中形如 $X|Y$ 的表达式中,$Y\neq\varnothing$.

引理 **1**. $P(X|Y)=0$ 的充要条件是 $XY=\varnothing$.

证明:$P(X|Y)=P(XY)/P(Y)$,

充分性:$\therefore XY=\varnothing\Rightarrow P(XY)=0\Rightarrow P(X|Y)=0$;必要性:$P(X|Y)=0\Rightarrow P(XY)=0\Rightarrow XY=\varnothing$. □

引理 **2**. $P(X|Y)=1$ 的充要条件是 $Y\subseteq X$.

证明: $P(X\mid Y)=1\Leftrightarrow P(\overline{X}\mid Y)=0\Leftrightarrow \overline{X}Y=\varnothing$,

充分性: $\therefore Y\subseteq X\Rightarrow \overline{X}\subseteq\overline{Y}\Rightarrow \overline{X}Y\subseteq\varnothing\Rightarrow \overline{X}Y=\varnothing$;

必要性: $\therefore \forall y\in Y, \overline{X}Y=\varnothing\Rightarrow y\notin\overline{X}\Rightarrow y\in X\Rightarrow Y\subseteq X$. □

5.3.1 入度为 1 的结点

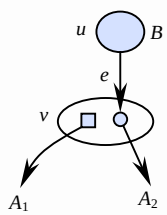

Fig.13 Node with one arc

图 13 一元计算

错误流图中入度为 1 的结点,对应于只有一个参数的原子计算,如图 13 所示.我们设定以下事件:

事件 B:u 错误,$F(u)=P(B)$;

事件 A:v 错误,$F(v)=P(A)$;

事件 A_1:v 物理错误,$F_1(v)=P(A_1)$,$F_1(v)$表示 v 物理错误的概率;

事件 A_2:v 传播错误,$F_2(v)=P(A_2)$,$F_2(v)$表示 v 传播错误的概率;

函数 $f(e)$:表示计算关系 e 在成立的情况下传播错误的概率.

那么,根据前面设立的规则和定律,我们有:

传播律:$A_2\subseteq B$,或者 $P(B|A_2)=1$(引理 2);

组成律:$A\subseteq A_1+A_2$,或者 $P(A_1+A_2|A)=1$(引理 2);

叠加律 1: $A_1\overline{A}_2\subseteq A$,或者 $P(A\mid A_1\overline{A}_2)=1$(引理 2);

叠加律 2: $\overline{A}_1A_2\subseteq A$,或者 $P(A\mid\overline{A}_1A_2)=1$(引理 2);

独立定律 1:$P(A_1A_2)=P(A_1)P(A_2)$;

独立定律 2:$P(A_1B)=P(A_1)P(B)$.

定理 **1**. $P(A_1)=P(A\mid\overline{A}_2)$.

证明:先证 $A_1=A\overline{A}_2+A_1A_2$,

$\because A_1\overline{A}_2\subseteq A$(叠加律1) $\because A_1\overline{A}_2\subseteq\overline{A}_2\therefore A_1\overline{A}_2\subseteq A\overline{A}_2\therefore A_1=A_1\overline{A}_2+A_1A_2\subseteq A\overline{A}_2+A_1A_2\because A\subseteq A_1+A_2$(组成律)

$\therefore A\overline{A}_2\subseteq(A_1+A_2)\overline{A}_2=A_1\overline{A}_2\therefore A\overline{A}_2+A_1A_2\subseteq A_1\overline{A}_2+A_1A_2=A_1\therefore A_1=A\overline{A}_2+A_1A_2$.

再证 $P(A_1)=P(A\mid\overline{A}_2)$,

$\because A\overline{A}_2 \cap A_1A_2 = \varnothing \therefore P(A_1) = P(A\overline{A}_2 + A_1A_2) = P(A\overline{A}_2) + P(A_1A_2) \because P(A_1A_2) = P(A_1)P(A_2)$(独立定律1)
$\therefore P(A_1) = P(A\overline{A}_2 + A_1A_2) = P(A\overline{A}_2) + P(A_1)P(A_2)$,
$\therefore P(A\overline{A}_2) = P(A_1) - P(A_1)P(A_2) = P(A_1)(1 - P(A_2)) = P(A_1)P(\overline{A}_2)$,

$$\therefore P(A_1) = \frac{P(A\overline{A}_2)}{P(\overline{A}_2)} = P(A \mid \overline{A}_2).$$ □

定理 **2**. $P(A_2) = P(A \mid \overline{A}_1)$.

证明:类似定理 1,由于 A_1 和 A_2 的对称性,先证 $A_2 = A\overline{A}_1 + A_1A_2$,再证 $P(A_2) = P(A \mid \overline{A}_1)$. □

定义 **13**. 我们定义计算关系 e 传播错误的概率为 $f(e) = P(A \mid \overline{A}_1B)$.

定理 **3**. $f(e) = P(A_2 \mid \overline{A}_1B)$.

证明:类似定理 1 中的证明,易证 $A_2 = A\overline{A}_1 + A_1A_2$.

$\therefore P(A_2 \mid \overline{A}_1B) = P(A\overline{A}_1 + A_1A_2 \mid \overline{A}_1B). \because A\overline{A}_1 \cap A_1A_2 = \varnothing \therefore P(A_2 \mid \overline{A}_1B) = P(A\overline{A}_1 \mid \overline{A}_1B) + P(A_1A_2 \mid \overline{A}_1B)$.
$\because A_1A_2 \cap \overline{A}_1B = \varnothing \therefore P(A_1A_2 \mid \overline{A}_1B) = 0$(引理1) $\therefore P(A_2 \mid \overline{A}_1B) = P(A\overline{A}_1 \mid \overline{A}_1B)$.
$\because f(e) = P(A \mid \overline{A}_1B) = P(AA_1 + A\overline{A}_1 \mid \overline{A}_1B) = P(AA_1 \mid \overline{A}_1B) + P(A\overline{A}_1 \mid \overline{A}_1B) \because AA_1 \cap \overline{A}_1B = \varnothing$,
$\therefore P(AA_1 \mid \overline{A}_1B) = 0$(引理1) $\therefore f(e) = P(A \mid \overline{A}_1B) = P(A\overline{A}_1 \mid \overline{A}_1B) \therefore f(e) = P(A_2 \mid \overline{A}_1B)$. □

定理 **4**. $P(A_2)=f(e)\cdot P(B)$.

证明: $P(A_2) = P(A \mid \overline{A}_1) = \frac{P(A\overline{A}_1)}{P(\overline{A}_1)} = \frac{P(A\overline{A}_1(B + \overline{B}))}{P(\overline{A}_1)} = \frac{P(A\overline{A}_1B) + P(A\overline{A}_1\overline{B})}{P(\overline{A}_1)}. \because A_2 \subseteq B$(传播律) $\therefore \overline{B} \subseteq \overline{A}_2$,

$\therefore A\overline{A}_1\overline{B} \subseteq A\overline{A}_1\overline{A}_2. \because A \subseteq A_1 + A_2$(组成律) $\therefore \overline{A}_1\overline{A} \subseteq \overline{A} \therefore A\overline{A}_1\overline{A}_2 \subseteq A\overline{A} = \varnothing \therefore A\overline{A}_1\overline{B} = \varnothing \therefore P(A\overline{A}_1\overline{B}) = 0$,

$$\therefore P(A_2) = \frac{P(A\overline{A}_1B)}{P(\overline{A}_1)} = \frac{P(A \mid \overline{A}_1B)P(\overline{A}_1B)}{P(\overline{A}_1)}.$$

$\because P(\overline{A}_1B) = P(\overline{A}_1) \cdot P(B)$(独立定律2) $\therefore P(A_2) = P(A \mid \overline{A}_1B)P(B) = f(e) \cdot P(B)$. □

定理 **5**. 令 $P(A_1) = a, P(B) = b, P(\overline{A} \mid A_1A_2) = \beta$,那么 $P(A)=a+f(e)b-(1+\beta)f(e)ab$.

证明: $F(v) = P(A) = P(A(A_1 + \overline{A}_1)) = P(AA_1(A_2 + \overline{A}_2)) + P(A\overline{A}_1) = P(AA_1A_2) + P(AA_1\overline{A}_2) + P(A \mid \overline{A}_1)P(\overline{A}_1) =$

$P(A \mid A_1A_2)P(A_1A_2) + P(A \mid A_1\overline{A}_2)P(A_1\overline{A}_2) + P(A_2)P(\overline{A}_1)$.

$\because P(A \mid A_1\overline{A}_2) = 1$(叠加律1),设$P(\overline{A} \mid A_1A_2) = \beta \therefore P(A) = (1-\beta)P(A_1A_2) + P(A_1\overline{A}_2) + P(A_2)P(\overline{A}_1)$.
$\because P(A_1A_2) = P(A_1)P(A_2)$(独立定律1) $\therefore P(A) = P(A_1) + P(A_2) - (1+\beta)P(A_1)P(A_2)$.

设$P(B) = b, P(A_1) = a$,且$P(A_2) = f(e) \cdot P(B) \therefore P(A) = a + f(e)b - (1+\beta)f(e)ab$. □

在定理 5 中,a 是可以测量的物理错误概率,b 是源的错误概率. $\beta = P(\overline{A} \mid A_1A_2)$ 表示当同时存在物理错误以及传播错误时,目标 v 中居然最终没有发生错误的概率.这是一个小概率事件.它的发生至少需要同时满足以下 3 个条件:A_1 和 A_2 错误的模式和位置一模一样;A_2 应该先于 A_1 发生;两者叠加的效果是消除错误.定理 5 说明,我们可以根据某些模型参数,定量地计算出错误流图中参与计算的原子数据的错误概率.

5.3.2 入度为 2 的结点

错误流图中入度为 2 的结点,对应于有两个参数的原子计算,如图 14 所示.我们设定以下事件:

事件 C:w 错误,$F(w)=P(C)$;

事件 B:u 错误,$F(u)=P(B)$;

事件 A:v 错误,$F(v)=P(A)$;

事件 A_1:v 物理错误,$F_1(v)=P(A_1)$,$F_1(v)$表示 v 物理错误的概率;

事件 A_2:v 传播错误,$F_2(v)=P(A_2)$,$F_2(v)$表示 v 传播错误的概率;

函数 $f(e_1)$:表示计算关系 e_1 在成立的情况下传播错误的概率.

函数 $f(e_2)$:表示计算关系 e_2 在成立的情况下传播错误的概率.

那么,根据前面设立的规则和定律,我们有:

传播律:$A_2 \subseteq B+C$,或者 $P(B+C|A_2)=1$(引理 2);

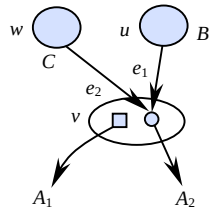

Fig.14 Node with two arcs

图 14 二元计算

组成律:$A\subseteq A_1+A_2$,或者 $P(A_1+A_2|A)=1$(引理 2);

叠加律 1: $A_1\overline{A}_2\subseteq A$,或者 $P(A\mid A_1\overline{A}_2)=1$ (引理 2);

叠加律 2: $\overline{A}_1A_2\subseteq A$,或者 $P(A\mid \overline{A}_1A_2)=1$ (引理 2);

独立定律 1:$P(A_1A_2)=P(A_1)P(A_2)$;

独立定律 2:$P(A_1B)=P(A_1)P(B)$,$P(A_1C)=P(A_1)P(C)$,$P(A_1BC)=P(A_1)P(BC)$, $P(A_1\overline{B}C)=P(A_1)P(\overline{B}C)$, $P(A_1B\overline{C})=P(A_1)P(B\overline{C})$.

定理 **6**. $P(A_1)=P(A\mid\overline{A}_2)$, $P(A_2)=P(A\mid\overline{A}_1)$.

证明:同定理 1 和定理 2. □

定义 **14**. 我们定义计算关系 e_1 传播错误的概率为 $f(e_1)=P(A\mid\overline{A}_1B\overline{C})$,计算关系 e_2 传播错误的概率为 $f(e_2)=P(A\mid\overline{A}_1\overline{B}C)$.

定理 **8**. $f(e_1)=P(A_2\mid\overline{A}_1B\overline{C})$, $f(e_2)=P(A_2\mid\overline{A}_1\overline{B}C)$.

证明:类似于定理 3 中的证明,先证 $A_2=A\overline{A}_1+A_1A_2$,再证 $f(e_1)=P(A_2\mid\overline{A}_1B\overline{C})$, $f(e_2)=P(A_2\mid\overline{A}_1\overline{B}C)$. □

定理 **9**. 令 $P(\overline{A}\mid\overline{A}_1BC)=\gamma$,那么 $P(A_2)=f(e_1)\cdot P(B\overline{C})+f(e_2)\cdot P(\overline{B}C)+(1-\gamma)\cdot P(BC)$.

证明: $P(A_2)=P(A\mid\overline{A}_1)=\dfrac{P(A\overline{A}_1)}{P(\overline{A}_1)}$.

$\because P(A\overline{A}_1)=P(A\overline{A}_1(B+\overline{B})(C+\overline{C}))=P(A\overline{A}_1BC+A\overline{A}_1B\overline{C}+A\overline{A}_1\overline{B}C+A\overline{A}_1\overline{B}\overline{C})$

$=P(A\overline{A}_1BC)+P(A\overline{A}_1B\overline{C})+P(A\overline{A}_1\overline{B}C)+P(A\overline{A}_1\overline{B}\overline{C})$,

$\because A_2\subseteq B+C$(传播律)$\therefore \overline{B}\overline{C}\subseteq\overline{A}_2\because A\subseteq A_1+A_2$(组成律)$\therefore \overline{A}_1\overline{A}_2\subseteq\overline{A}$,

$\therefore A\overline{A}_1\overline{B}\overline{C}\subseteq A\overline{A}_1\overline{A}_2\subseteq A\overline{A}=\varnothing\therefore A\overline{A}_1\overline{B}\overline{C}=\varnothing\therefore P(A\overline{A}_1\overline{B}\overline{C})=0$.

$\because P(A_1B\overline{C})=P(A_1)P(B\overline{C})$(独立定律2)$\therefore P(A\overline{A}_1B\overline{C})=P(A\mid\overline{A}_1B\overline{C})P(\overline{A}_1)P(B\overline{C})$.

同理, $P(A\overline{A}_1\overline{B}C)=P(A\mid\overline{A}_1\overline{B}C)P(\overline{A}_1)P(\overline{B}C)$(独立定律2),

$P(A\overline{A}_1BC)=P(A\mid\overline{A}_1BC)P(\overline{A}_1)P(BC)$(独立定律2),

$$\therefore P(A_2)=\frac{P(A\overline{A}_1\overline{B}C)+P(A\overline{A}_1B\overline{C})+P(A\overline{A}_1\overline{B}C)+P(A\overline{A}_1BC)}{P(\overline{A}_1)}$$

$$=P(A\mid\overline{A}_1B\overline{C})P(B\overline{C})+P(A\mid\overline{A}_1\overline{B}C)P(\overline{B}C)+P(A\mid\overline{A}_1BC)P(BC).$$

设 $P(\overline{A}\mid\overline{A}_1BC)=\gamma$, 且 $P(A\mid\overline{A}_1B\overline{C})=f(e_1)$, $P(A\mid\overline{A}_1\overline{B}C)=f(e_2)$,

$\therefore P(A_2)=f(e_1)P(B\overline{C})+f(e_2)P(\overline{B}C)+(1-\gamma)P(BC)$. □

定理 **10**. 令 $P(B|C)=b_1$, $P(B\mid\overline{C})=b_2$, $P(C)=c$, $P(A_1)=a$, $P(\overline{A}\mid\overline{A}_1BC)=\gamma$,那么,

$$P(A)=a+f(e_1)b_2+f(e_2)c-(1+\beta)f(e_1)ab_2-(1+\beta)f(e_2)ac+(1-f(e_2)-\gamma)b_1c-$$
$$f(e_1)b_2c+(1+\beta)f(e_1)ab_2c-(1+\beta)(1-f(e_2)-\gamma)ab_1c.$$

证明:$\because P(A)=P(A_1)+P(A_2)-(1+\beta)P(A_1)P(A_2)$,

$\because P(A_2)=f(e_1)P(B\overline{C})+f(e_2)P(\overline{B}C)+(1-\gamma)P(BC)$,

设 $P(B\mid C)=b_1$, $P(B\mid\overline{C})=b_2$, $P(C)=c$, $P(A_1)=a$,

$\therefore P(B\overline{C})=b_2(1-c)\therefore P(\overline{B}C)=(1-b_1)c\therefore P(BC)=b_1c\therefore P(A_2)=f(e_1)b_2-f(e_1)b_2c+(1+f(e_2)-\gamma)b_1c+f(e_2)c$,

$\therefore P(A)=a+f(e_1)b_2+f(e_2)c-(1+\beta)f(e_1)ab_2-(1+\beta)f(e_2)ac+(1-f(e_2)-\gamma)b_1c-f(e_1)b_2c+$

$(1+\beta)f(e_1)ab_2c-(1+\beta)(1-f(e_2)-\gamma)ab_1c$. □

定理 10 中,b_1 和 b_2 反映 B 和 C 之间的相关性. $\gamma=P(\overline{A}\mid\overline{A}_1BC)$ 表示当目标不存在物理错误,但是两个源都存在错误的情况下,目标 v 居然没有错误的概率.这也是一个小概率事件. $\beta=P(\overline{A}\mid A_1A_2)$ 与上一节中的定义一样,仍然表示当同时存在物理错误以及传播错误的时候,目标 v 居然最终没有发生错误的概率.

如果 B 和 C 之间无关,那么 $b_1=P(B|C)=P(B)$, $b_2=P(B\mid\overline{C})=P(B)$,设 $P(B)=b$,则有

$$P(A)=a+f(e_1)b+f(e_2)c-(1+\beta)f(e_1)ab-(1+\beta)f(e_2)ac+$$
$$(1-f(e_2)-f(e_1)-\gamma)bc+(1+\beta)(f(e_1)+f(e_2)+\gamma-1)abc.$$

如果 B 和 C 之间无关并且 $f(e_1)=f(e_2)=1$,则有

$$P(A)=a+b+c-(1+\beta)ab-(1+\beta)ac-(1+\gamma)bc+(1+\beta)(1+\gamma)abc.$$

定理 10 说明,在二元计算中我们同样可以根据某些模型参数,定量地计算出错误流图中参与计算的原子数据的错误概率.三元及更高元计算很少见,错误传播分析暂略.

5.3.3 边界和执行概率

入度为 0 和出度为 0 的结点是错误流图的边界.在边界结点上错误的传播终止.入度为 0 的结点仅有物理错误而没有从其他结点传播过来的传播错误;出度为 0 的结点不再把错误传播给其他结点.当计算关系以一定概率存在的情况下,应该使用 $\hat{f}(e)=f(e)\cdot B(e)$ 来代替 $f(e)$进行计算;当 $B(e)=1$ 时,即以上两小节中叙述的情况,此时 $\hat{f}(e)=f(e)$.

上面 3 小节论述了如何根据上游原子数据的错误概率、可测量的物理错误概率以及若干参数,计算出下游原子数据的错误概率.这种概率体现了计算关系对于错误传播的影响:程序中的计算改变了错误的时空分布.根据程序中的计算对于错误传播的影响,我们就可以根据需要优化程序的计算结构,减小程序中传播的错误对程序的运行所造成的危害.

5.4 URM程序示例

对于第 3.3.1 节中计算 $x+y$ 的 URM 程序例子,如果后继指令 $S(n)$的源和目标之间的计算关系 e_s传播错误的概率为 $f(e_s)$,而由于分支指令的存在计算关系 e_s 成立的执行概率是 $B(e_s)=P(r_2\neq r_3)$,那么 $\hat{f}(e_s)=f(e_s)\cdot B(e_s)$,令 $a_1=F_1(r_1^1)$ 是 r_1^1 物理错误概率,根据定理 5 有 $F(r_1^1)=a_1+\hat{f}(e_s)\cdot F(r_1^0)-(1+\beta)\hat{f}(e_s)a_1\cdot F(r_1^0)$.

如果令函数 $\zeta(x)=a_1+\hat{f}(e_s)\cdot x-(1+\beta)\hat{f}(e_s)a_1\cdot x$,那么 $F(r_1^1)=\zeta(F(r_1^0))$.

我们记 $r_1^{(i,1)}$ 和 $r_1^{(i,0)}$ 为第 i 次循环中的原子数据 r_1^1 和 r_1^0.由于循环的迭代关系,我们知道 $r_1^{(i,1)}=r_1^{(i+1,0)}$,即本次循环的 r_1^1 就是下次循环中的 r_1^0.又根据 $F(r_1^1)=\zeta(F(r_1^0))$,我们容易得到,

$$F(r_1^{(i,1)})=\zeta(F(r_1^{(i,0)}))=\zeta(F(r_1^{(i-1,1)}))=\zeta(\zeta(F(r_1^{(i-1,0)})))=\ldots=\zeta^{i+1}(F(r_1^{(0,0)})).$$

对于 r_3^1 和 r_3^0 的分析类似.可以看出,只要我们知道了循环迭代之间原子数据之间的关系以及循环迭代内原子数据的错误概率之间的关系,我们就可以递归地求出任意一次循环迭代中原子数据的错误概率.

6 结 论

我们已经能够根据程序流图为任意程序建立计算数据流,并在计算数据流的基础上进一步建立错误流图.在错误流分析过程中,我们可以通过 6 条规则和 2 条独立定律计算出任意存储单元在任意计算后发生错误的概率.利用错误流模型,我们能够对错误在任意程序中的传播进行细致的量化研究.

7 未来的工作和展望

不同的计算关系有着不同的错误传播概率 $f(e)$,模型中的参数β和γ都是很小的概率,为了确定这些模型参数,我们还需要做进一步的工作.$f(e)$是与特定计算相关的待定参数.我们可以通过两种方法确定$f(e)$的值:通过实验的方法,或者通过进一步建模分析.

利用错误流模型,我们可以定量研究硬件故障通过程序传播的情况,为我们分析和设计容错算法提供依据.根据程序的错误流分析对程序进行等价变换,可能得到更不容易传播错误的程序.我们相信,对于程序传播错误的研究将有助于编写和编译出更可靠的程序.

References:

[1] Liu P. Reliability Engineering Principles. Revised ed., Beijing: Measurements Press, 2002 (in Chinese).

[2] Xu RZ, Xie M, Zheng RJ. Software Reliability Models and Applications. Beijing: Tsinghua University Press, 1994 (in Chinese).

[3] Tian J. Integrating time domain and input domain analyses of software reliability using tree-based models. IEEE Trans. on Software Engineering, 1995,21(12):945−958.

[4] Huang CY, Lyu MR. A unified scheme of some nonhomogenous poisson process models for software reliability estimation. IEEE Trans. on Software Engineering, 2003,29(3):261−269.

[5] Clark JA, Pradhan DK. Fault injection: A method for validating computer-system dependability. IEEE Computer, 1995,28(6): 47−56.

[6] Avizienis A. Toward systematic design of fault-tolerant systems. IEEE Computer, 1997,30(4):51−58.

[7] Cheynet P, Nicolescu B, Velazco R, Rebaudengo M, Reorda MS, Violante M. Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors. IEEE Trans. on Nuclear Science, 2000,47(6):2231−2236.

[8] Ziegler JF. IBM experiments in soft fails in computer electronics (1978-1994). IBM Journal of Research and Development, 1996,40(1):3−18.

[9] Avizeinis A. The *n*-version approach to fault-tolerant software. IEEE Trans. on Software Engineering, 1985,SE-11(12):1491−1501.

[10] Randell B. System structure for software fault tolerance. IEEE Trans. on Software Engineering, 1975,SE-1(2):220−223.

[11] Oh N. Software implemented hardware fault tolerance [Ph.D. Thesis]. Stanford: Stanford University, 2000.

[12] Gerke RD, Shapiro AA. Use of commercial off-the-shelf (COTS) for space applications. In: Proc. of the Aerospace Conf. IEEE Computer Society, 2003. 230.

[13] Huang KH, Abraham JA. Algorithm-Based fault tolerance for matrix operations. IEEE Trans. on Computers, 1984,33(6):518−528.

[14] Maurizio R, Matteo SR, Massimo V, Marco T. A source-to-source compiler for generating dependable software. In: Proc. of the 1st IEEE Int'l Workshop on Source Code Analysis and Manipulation. Florence: IEEE Computer Society, 2001. 33−42. http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/scam/2001/1387/00/1387toc.xml&DOI=10.1109/SCAM.2001.972664

[15] Oh N, Shirvani PP, McCluskey EJ. Error detection by duplicated instructions in super-scalar processors. IEEE Trans. on Reliability, 2002,51(1):63−75.

[16] Oh N, Mitra S, McCluskey EJ. ED^{4}I: Error detection by diverse data and duplicated instructions. IEEE Trans. on Computers, 2002, 51(2):180−199.

[17] Burger DC, Austin TM. The SimpleScalar tool set, version 2.0. ACM SIGARCH Computer Architecture News, 1997,25(3):13−25.

[18] Chen HW, Qian JH, Sun YQ. Principles of Compilers. 2nd ed., Beijing: Press of Defense Industry, 1999 (in Chinese).

[19] Cliff Y, Michael DS. Static correlated branch prediction. ACM Trans. on Programming Languages and Systems, 1999,21(5): 1028−1075.

[20] Wu Y, Larus JR. Static branch frequency and program profile analysis. In: Proc. of the 27th Annual Int'l Symp. on Microarchitecture. New York: ACM Press, 1994. 1−11. http://portal.acm.org/citation.cfm?id=192725&dl=ACM&coll=portal

[21] Jason RC, Patterson DA. Accurate static branch prediction by value range propagation. In: Proc. of the ACM SIGPLAN 1995 Conf. on Programming Language Design and Implementation. New York: ACM Press, 1995. 67−78. http://portal.acm.org/citation.cfm?id=223428.207117

[22] Yang DP, Li AS. Computing Theories. Beijing: Science Press, 1999 (in Chinese).

附中文参考文献:

[1] 刘品.可靠性工程基础.修订版.北京:计量出版社,2002.

[2] 徐仁佐,谢旻,郑人杰.软件可靠性模型及应用.北京:清华大学出版社,1994.

[18] 陈火旺,钱家骅,孙永强.编译原理.第 2 版.北京:国防工业出版社,1999.

[22] 杨东屏,李昂生.可计算性理论.北京:科学出版社,1999.

杨学军(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为并行体系结构,并行编译,并行操作系统.

高珑(1978－),男,山东胶州人,博士生,主要研究领域为操作系统,编译器,嵌入式系统.