学 科 通 融 · 之十五 · 软件工程 × 数字保存与条约法 × 演化遗传学

哪一份算数

同一样东西存了好几份，三个互不往来的行当给出三种互相取消的处方：必须消除、必须增加且刻意不设正本、不一致才是产出。本文论证三家漏掉了同一条分界——副本本身既不危险也不安全，危险只发生在一个地方：系统预期它们一致，却没有人负责让它们一致。

王德生 + Claude · 约 1.6 万字 · 17 页 · 三种阅读方式 · 发表于2026年8月1日

软件工程有一条流传二十多年的原则：每一条知识必须有单一、无歧义、权威的表示，否则改一处会漏另一处。数字保存的做法正好相反——多存几份才丢不了，而且刻意不指定哪一份是正本，节点之间不断互相比对、按多数修复；多语种条约也是这样，各文本同等作准，歧义时按条约目的解释，不设母本。演化遗传学则说，一个基因被复制之后，多出来的那份之所以能长成新东西，**前提恰恰是没有任何机制要求两份保持一致**。三家都有几十年的证据，处方却互相拆台。本文认为三家用的不是同一个变量：**副本本身既不危险也不安全，危险只发生在一个地方——系统预期它们一致，却没有任何人负责让它们一致**。危害的量不由份数决定，由这个预期覆盖的范围决定。由此把「预期」与「机制」拆成两条独立的轴，得到一张四格表，其中最坏的一格此前没有名字：所有人心里都有一份「算数的那一份」，只是每个人心里的不是同一份——它有全部的效力，没有任何的位置。它的读数是一句一分钟就能问完的话：**如果这两份不一样，哪一份算数？**文末给一套半小时可执行的查法、十二处跨领域读数、两处反过来迫使本文让步的约束（多数重复基因终将丢失；独立编写的版本并不独立地出错）、六条判错条件，以及一条写死到 2032 年的赌注。

一、同一件事，三个相反的处方

一样东西存了好几份。这是一件极普通的事，普通到很少有人把它当成一个需要解释的对象。

但只要把三个行当的判决摆在一起，它立刻变得很不普通。

一位软件工程师会说：把它们合掉。 这个行当有一条流传了二十多年的原则，原句是：每一条知识，在一个系统之内必须有单一的、无歧义的、权威的表示。[1] 理由不是省地方——地方从来不值钱——而是改一处会漏掉另一处。一个人修好了这一份里的错，另一份还在，然后在某个谁也没预料到的场合，那个旧的错跳出来。

一位数字档案员会说：再多存几份。 这个行当有一句同样流传很广的口号，意思是"多存几份，东西才丢不了"。[2] 更要紧的是它的做法：这些副本之间不设正本，节点之间不断互相比对，谁偏离了就按多数把它修回去。它拒绝回答"哪一份是原件"这个问题，而这不是疏忽，是设计。

一位国际法律师会说：同一份文件本来就有好几份，而且每一份都算数。 多语种条约的通例是各语言文本同等作准，出现歧义时不去找一个母本，而是按条约的目的与宗旨解释。[3] 这套办法运转了很久，没有人认为它是一种妥协。

一位演化遗传学家会说：如果没有多出来的那一份，什么新东西也不会有。 一个基因被复制之后，原来那份继续干原来的活，多出来的那份才承受得起突变——因为它坏了不要紧。[4] 这个行当把绝大部分生物学上的创新记在这笔账上，而它成立的前提，恰恰是**没有任何机制要求两份保持一致**。

四句话，三种处方，互相拆台：消除、增加、放任。而且没有一家是外行——每一家都有几十年的证据。

还要补一句，四家里没有一家是在讲同一个层面上的事——这一点在第五节会成为关键。软件工程讲的是**修改时会不会漏**，数字保存讲的是**丢失时会不会灭绝**，国际法讲的是**争议时按谁**，演化遗传学讲的是**新东西从哪来**。四个问题，四条时间线：修改是日常的，丢失是罕见的，争议是偶发的，创新是极长期的。它们对同一个事实给出相反处方，一部分原因就在这里——**每一家都在为自己那条时间线上最要紧的那件事做最优安排，而这四条时间线的最优安排互不相容。**

这就是本文的起点。**同一个事实，三个成熟的行当给出不能并存的判决，说明大家用的不是同一个变量。**

二、三家各自说到了哪一步

先公允地把三家各自的强处摆出来，再说它们在哪里停下。

2.1 消除派：危害在"改一处会漏"

这一派的表述比通常的理解精确得多。它说的不是"不要复制粘贴"，而是"每一条**知识**必须有单一、无歧义、权威的表示"。差别很关键：两段代码长得一模一样，如果它们表达的是两件不同的事，那不算违反；两段代码长得完全不同，如果它们表达的是同一条规则，那才是。[1]

这一派的强处在于它抓住了一个真实的失败模式，而且这个模式可以被反复观察：一条规则改了，改的人只知道其中一处，于是系统进入一个不自洽的状态，而这个状态**在被同时用到之前不会报错**。

还有一处强处值得单列：这一派把"知识"与"表示"分开了。这在当时是一个很不容易的动作——它意味着查重不能靠比对字面，得先判断两处说的是不是同一件事。而这个判断没有算法，只能由人做。这一点在正文后面会反复用到：**这一派最深的洞见，恰恰是它自己无法自动化的那一部分。**

它的止步处也很清楚：**它把处方定死在"减少份数"上。** 它无法安置那些份数极多、却极其可靠的系统——数字保存的多副本网络、多语种条约、每一个分叉出去就不再合并的开源项目。在它的框架里，这些要么是设计错误，要么必须被重新描述成"其实只有一份"。

2.2 增加派：可靠来自互相校验

这一派的强处在于它把可靠性从"保住一份好的"改成了"让多份互相看着"。它的关键动作不是复制，是**校验**：节点定期比对，发现分歧就按多数修复。所以它的多，是有条件的多。

而它最值得注意的一步，是**主动取消正本**。多语种条约不设一个母本，是因为一旦设了，其余文本就降格为译本，缔约方的谈判成果会被悄悄地重新分配。取消正本不是无奈，是为了让权威落在**程序**上，而不落在**某一份**上。

这一派还有一个常被忽略的细节：它的副本不是静态存着的，是**主动巡检**的。没有巡检的多副本，与一堆散落的备份没有区别；口号里那句"多存几份"，隐含的完整版本是"多存几份并且不断互相比对"。本文后面把这一步单独称作机制，正是从这里来的。

它的止步处在于：**它假设一致是目标。** 凡是不以一致为目标的场合，它给不出任何指导；而下一节那一家整个建立在"不一致才有用"上。

2.3 放任派：不一致才是产出

一个基因被复制之后，两份在很长一段时间里做同一件事。这看起来是纯粹的浪费，但它做了一件别的办法做不到的事：**它把约束从其中一份上撤走了。** 原来那份继续供着必需的功能，多出来的那份可以随便变——变坏了，个体照样活；变出点别的，就是新东西。[4]

后来的研究给这条加了一个更常见的版本：两份各自丢掉原来那个基因的不同部分，于是谁也不能少，两份加起来才等于原来一份。[5] 这条路径不需要任何"新功能"，仅靠退化就能把两份都保住。

这一派还有一个与前两家形成尖锐对照的性质：**它不区分"故意的不一致"与"意外的不一致"**。 一个突变是随机的，没有人打算让两份分开，也没有人打算让它们保持一致——不一致既不是目标也不是事故，它就是默认状态。前两家的整个语汇里都没有"默认状态"这一档：在它们那里，不一致要么是被追求的，要么是被容忍的，要么是被消灭的，总之总有一个态度。

这一派的止步处在于：**它的世界里没有"预期"这个东西**。 基因组里没有任何人期待两份一致，所以它从来不需要处理"以为该一样、其实不一样"这种情形——而那恰恰是前两家吵得最凶的地方。

三、三家在哪里互相取消

把三条处方并排，冲突不是笼统的"观点不同"，而是三处具体的对顶。

第一处：份数与安全的方向相反。 消除派说份数越少越安全，增加派说份数越多越安全。这看起来像同一根轴上的正面对顶，但两家说的"安全"根本不是一件事：消除派的安全是**改的时候不出错**，增加派的安全是**丢的时候不灭绝**。两种安全都要，而按任一家的处方做到极致，另一种必然崩掉——只留一份，一场火灾就没了；存一千份，改一条规则要改一千次。

第二处：正本该不该有，两家给了相反的答案，理由却在同一个位置。 消除派要一个权威表示，是为了让修改有唯一的落点。增加派要取消正本，也是为了让修改有唯一的落点——只不过那个落点是程序，不是某一份。**两家争的其实不是"要不要权威"，而是权威该落在位置上还是落在程序上**，而两家都没有把这句话说出来。

第三处：放任派把前两家的共同前提整个掀掉了。 前两家都默认副本应当一致，分歧只在怎么做到。放任派提供了一大批极其成功的反例：那里没有人期待一致，而不一致正是产出。如果一致不是必需的，那么前两家的整场争论就只覆盖了一部分场合，而它们各自都以为自己覆盖了全部。

三处合起来指向同一件事：**三家都在处理"副本"，而真正决定结局的变量不在副本上**。

四、把"预期"从"机制"里分出来

先分开两个平时总是一起出现、因而很少被分开的东西。

预期：这套系统里的人是不是认为，这几份应当是一样的。注意它是一个关于**人**的事实，不是关于副本的事实。它可以被明确写下来，也可以从来没有被说出口而人人都这么以为。

机制：有没有一个被指定的东西——一个人、一道流程、一段程序——负责让它们保持一样，并且在它们不一样的时候给出裁决。注意"被指定"三个字：一个碰巧总在做这件事的老员工，不算被指定；他哪天不做了，没有任何事情会报警。

这两件事在日常里几乎总是一起被谈论，所以很容易以为有预期就有机制、没机制就没预期。**恰恰相反：最麻烦的情形正是预期在而机制不在**。

看一个最小的例子。一家公司有一份报销标准，总部写了一版，三个分公司各自存了一份、也各自改过几处。

- 所有人都认为这三份应该是一样的——**预期在**；
- 没有任何人的职责里写着"核对这三份"——**机制不在**；

· 于是三份慢慢分开，而这件事**在有人同时用到两份之前不会暴露**。

暴露的那一天通常是一次跨分公司的调动，或者一次审计。在那之前，每一份单独看都完好，每一次报销都顺利通过，账面上没有任何异常。

这个情形有一个特征，值得写死：**它没有报警条件**。 消除派的那条原则能诊断它（三份即违反），但按那条原则去做，处方是"合成一份"，而现实里合不掉——三个分公司真的有不同的报销场景。增加派的做法能救它（互相比对、按多数修复），但没有人会想到把这套办法用在三份 Word 文档上。放任派会说这挺好（各地演化出适合自己的版本），而这句话在报销这件事上是错的，因为公司确实需要一致。

三家都够不着，是因为三家都在看副本，而这里出问题的是**预期与机制之间的那道缝**。

由此得到本文的承重判断：

> **副本本身既不危险也不安全。危险只发生在一个地方：系统预期这些副本一致，却没有任何人负责让它们一致。危害的量不由副本的数量决定，由这个预期覆盖的范围决定。**

为行文方便，下文把"预期在而机制不在"这种情形叫作**影子正本**——所有人心里都有一份"算数的那一份"，但谁也说不出它在哪儿。

五、三家为什么都看不见这一格

上一节那个报销标准的例子里，三家各自能说出一点，却都到不了那条判断。值得逐家看清它们卡在哪儿——因为卡住它们的不是疏忽，是各自方法里一个必要的设定。

消除派看不见它，是因为它的诊断单位是"份数"。 那条原则一旦发现三份，就已经判完了：违规。它不需要再问预期，也不需要再问机制——三份本身就是问题。于是它对格Ⅱ与格Ⅲ给出同一个诊断，而这两格的实际后果一个是灾难、一个是健康。**一把只量份数的尺子，量不出"份数相同而后果相反"这件事**。

更要紧的是它的处方在格Ⅱ里往往执行不了。三个分公司的报销场景真的不同，合成一份要么牺牲实际需要，要么把差异塞进一堆例外条款——而例外条款本身又是新的重复。执行不了的处方，在实践中会退化成不执行，于是格Ⅱ照旧。

增加派看不见它，是因为它的动作是"加副本"。 它的整套办法建立在一个前提上：副本越多，互相校验的能力越强。而格Ⅱ里加副本只会更糟——多一份，就多一个没人核对的分支。**它的药在这一格里是毒**。

它也不会想到把自己那套办法用在三份文档上，因为它的场景是机器与机器之间，而格Ⅱ最常发生在人与人之间。这不是它的错，是它的适用域没有被写清楚。

放任派看不见它，是因为它的世界里没有"预期"这个东西。 基因组里没有任何主体期待两份一致，所以"以为该一样、其实不一样"这种情形在那里根本不成立。它的全部工具——约束的释放、子功能的分割、退化与保留——都是关于**副本本身**的，没有一样是关于**看着副本的人**的。

于是出现一个有意思的局面：**格Ⅱ的存在，恰恰需要三家的东西合起来才能被看见**。 消除派提供了"预期一致"这一维——它是唯一把"权威表示"写进定义的一家。放任派提供了一批"不预期一致反而更好"的成功案例，否则很容易以为一致总归是对的。增加派提供了"预期一致却可以没有正本"这个反例，否则很容易以为预期一致就必须指定一份。

三样并排，唯一的出路就是把"预期"与"机制"拆成两条互不蕴含的轴。少任何一家，这两条轴都合得回去：只有消除派与增加派，会以为争的是要不要正本；只有消除派与放任派，会以为争的是要不要一致；只有增加派与放任派，会以为争的是要不要校验。

六、两条轴与四个格

把上面两件事各取两值，得到四格。

	有被指定的同步机制	没有被指定的同步机制
系统预期这些副本一致	**格Ⅰ · 有主同步** 权威可以落在某一份上（主从复制、总部下发＋回执），也可以落在程序上（多副本互相比对按多数修复、多语种条约同等作准）。两种都健康，因为"不一致时怎么办"有答案。	**格Ⅱ · 影子正本** 人人默认应当一致，无人负责，且说不出不一致时哪份算数。**每一份单独看都完好，只在两份被同时用到时才暴露**。
系统不预期它们一致	**格Ⅲ · 受控分岔** 明确宣布不再合并，并有人管着分岔的边界（分叉后各自发布、方言的标准化机构、复制后两份各担一半功能而都被保住）。	**格Ⅳ · 自由分化** 没人期待一致，也没人管。产出方差极大：多数副本退化、消失，少数变成新东西。**要求一个能承受大量死亡的容量**。

四格的读数，落到能问、能查的东西上：

格Ⅰ：随便找两个人，各问一句"不一致的时候按哪份"，两人的答案相同，并且都能说出谁负责。

格Ⅱ：问同样的话，**答案分歧**——有人说按总部的，有人说按最新的，有人说按用得最多的那份，还有人愣一下说"应该都一样吧"。最后这句是最强的读数。

格Ⅲ：答案是"不需要一样"，并且能说出从哪一天起、按什么规矩分开的。

格Ⅳ：答案也是"不需要一样"，但说不出规矩，也没人管死活。

一分钟判据：拿着两份实物，问在场每个人同一句话——

> **如果这两份不一样，哪一份算数？**

这句话的好处是它不需要先认定任何人有过错，也不需要读懂内容。答案的**分歧程度**就是读数本身。

用一件小事走一遍。 一个团队里有三份"上线检查清单"：一份在共享文档里，一份印在墙上，一份在某人的笔记里。

问第一句："这三份应该一样吗？"三个人都说应该。**预期在**。

问第二句："如果不一样，按哪份？"第一个人说按共享文档，第二个人说按墙上那份（因为那是当初一起定的），第三个人愣了一下说"应该都一样吧"。**答案分歧，机制不在**。

于是可以直接判：格Ⅱ。接下来只需要再问一句就能估出损害的量级——**这三份分别是什么时候最后被改过的？** 时间差越大，已经积累的分歧越多，而这份分歧在下一次有人同时用到两份之前不会显形。

这个例子的价值在于它足够小，小到可以在一次站会上真的做完。第十七节那条最便宜的判据，用的就是这个形状。

后面几节按格展开，主戏在格Ⅱ。

七、格Ⅰ：权威可以不落在任何一份上

先立格Ⅰ，因为它给出后面三格的对照，而且它内部有一个被普遍忽略的分岔。

格Ⅰ是预期一致、且有人负责。它有两种实现，外观差得很远。

位置式：指定某一份为准，其余的向它看齐。总部下发文件、主库与从库、一份签署的正本加若干份复印件，都是这一种。它的长处是简单：任何争议一句话就能了结。它的短处是那一份出事就全出事，而且它把所有偏离都定义成违规——这一点在第十六节会变成一个具体的麻烦。

程序式：不指定任何一份，用一道程序给出权威。多副本网络按多数修复是这一种；多语种条约各文本同等作准、歧义时按条约目的解释也是这一种。[2][3]

程序式是三家争论里被漏掉最多的一格，值得多说一句。它证明了一件反直觉的事：**"预期一致"并不蕴含"必须有一个正本"**。 消除派把这两件事绑在一起了——它要"单一、无歧义、权威的表示"，而那句话在字面上就把权威放在了一个位置上。多语种条约每天都在做的事，按那条原则读，是明确违反的；而它极其可靠。

格Ⅰ内部还有一件事值得说清楚，因为它决定了这一格好不好维持。**位置式的成本集中在平时，程序式的成本集中在建时**。 指定一份为准，建起来只要一句话，但此后每一次修改都要推送、要回执、要抽查，成本按次数累加。用程序定权威，建起来很贵——要定清楚什么算多数、歧义时按什么解释、谁有资格投票——但建好之后，边际成本接近于零。

这解释了一个常见的选择偏差：**组织几乎总是选位置式，因为决策当天它更便宜**。 而那笔按次数累加的成本，会在此后的每一次修改里被悄悄地跳过一点点——跳过得多了，就滑进了第八节那一格。**位置式不是错的，它只是有一条通往格Ⅱ的滑坡，而程序式没有**。

这也是本文第一次与消除派分开的地方：**问题从来不是有几份，是"不一致时怎么办"有没有答案**。 答案可以是"按主库"，也可以是"按多数"，甚至可以是"按当初的目的重新解释"——三种都算答案。没有答案，才是问题。

八、格Ⅱ：影子正本

现在展开主戏。

格Ⅱ的定义只有两句：所有人都认为这些应当一致；没有任何人被指定去保证这件事。

它之所以是四格里最坏的一个，不是因为不一致的程度最深——格Ⅳ的分化程度深得多——而是因为它有三个性质，凑在一起使它几乎不可能被及时发现。

第一，它没有单点读数。 每一份单独看都是完好的。你可以逐份审查，逐份都合格。不一致这件事**不存在于任何一份之中**，它只存在于两份之间；而绝大多数检查是按份做的。

第二，它的暴露时点由使用决定，不由损害决定。 三份报销标准分开了三年，这三年里损害一直在积累（有人多报了，有人少报了），但暴露发生在第一次有人同时用到两份的那一天。**损害的发生与暴露之间没有固定的时间关系**，因此从暴露频率完全推不出损害规模。

第三，也是最要紧的：所有人都在按那个不存在的正本行事。 每个人心里都有一份"应该算数的那一份"，只不过每个人心里的那一份不是同一份。这就是"影子"二字的来历——它有全部的效力，没有任何的位置。

这第三点带来一个很具体的后果。**当不一致终于暴露时，处理它的通常不是规则，是嗓门**。 因为没有事先约定的裁决办法，冲突的解决落回到人际权力上：谁的部门更强势，谁的版本更"看起来正式"，谁在场。而这个解决过程不会被记录成"我们缺一个裁决机制"，它会被记录成"某某部门那次搞错了"。**于是每一次暴露都被消化成一次个人失误，而结构原封不动**。

举三个不同尺度的例子，形状完全一样。

一份被各科室改编过的临床指南。　总院发了一版，各科依自己的病人构成做了调整，都合理。所有人都认为"我们用的是同一份指南"。不一致在院内几乎不暴露——因为每个科用自己那份是自洽的。它在**跨科转诊**时暴露：同一个病人在两个科被按两套阈值处理，而两个科的医生都确信自己遵循的是指南。

一段被复制进五个项目的合同条款。起草时是同一段，五年里各自被改过。所有人都认为这是"我们公司的标准条款"。不一致在平时不暴露，因为一次只用一份。它在**争议**时暴露，而那时的裁决落回到"哪一份签在合同上"——也就是说，落回到位置式，只不过是事后的、随机的位置式。

同一门课的课件被四位教师各自维护。　都认为是"这门课的课件"。不一致在期末**统考**时暴露，而暴露的形式是学生成绩差异，于是被归因为教师水平差异。

第四个例子形状相同，但它是新的，而且正在快速变多。

同一条业务规则，被四个人各自用同一套工具生成了一份说明。　四份的措辞完全不同——一份写成流程图，一份写成条款，一份写成问答，一份写成代码注释。四个人都认为自己写的是"那条规则"，也都认为四份说的是同一件事。

这一例把格Ⅱ的隐蔽性推到了新的位置。前三例里，至少还能看出这些是副本——它们长得像。这一例里**副本的外观已经不同了**，因而任何按字面查重的办法都发现不了；而按第二节那条区分，它们完全构成重复，因为它们表达的是同一条知识。

于是这一例同时满足三件事：预期一致、无人负责、且**连"这是几份"都数不出来**。第十二节会说明为什么这一类正在变多。

三例的共同结构：**预期一致（都认为是同一份东西）＋无人负责（没有任何人的职责里写着核对）＋只在合流处暴露（转诊、争议、统考）**。

最后补一句关于命名的话。本文用"影子正本"而不用"不一致"，是因为不一致只是症状。真正承重的是那个**没有位置却有效力的权威**：它使每个人都以为自己在遵循规则，使偏离在被发现之前不被当成偏离，也使冲突在被发现之后不被当成结构问题。

九、格Ⅲ：受控分岔

格Ⅲ是不预期一致，且有人管着分岔。

一个开源项目分叉出去，并明确宣布此后不再向上游合并，是这一种。一门语言有一个标准化机构，管着标准语的边界，而各地方言照旧生长，也是这一种。

生物学里有一条更精细的版本值得单列。一个基因被复制之后，最常见的**被保留下来**的路径不是其中一份长出新功能，而是两份各自丢掉原来那个基因的不同部分——这一份保住了在某个组织里表达的能力，那一份保住了另一个组织的，于是谁也不能少。[5] 从外面看，这两份既不一致，也不冗余；它们是被**选择**这道程序管着的分岔。

格Ⅲ的关键不在"分开"，在**分开这件事被说出来了**。说出来之后，三件事同时成立：不一致不再是故障；不必再为同步付出成本；而且——这一条最容易被忽略——**合流处的责任被明确了**。分叉的项目会写清楚"我们与上游不再兼容"，标准化机构会说清楚"考试按标准语"。

格Ⅱ与格Ⅲ的差别只在一句话上：**分开这件事，有没有被承认**。　承认了就是格Ⅲ，没承认就是格Ⅱ。而这两格的实际不一致程度，可以完全一样。

这也是本文与消除派最硬的分离点。消除派会把格III判为违规——那里显然有多份表达同一类知识。但格III里没有任何一次"改一处漏一处"的事故，因为没有人期待另一处会跟着改。**危害从来不在多份，在被辜负的预期**。

十、格IV：自由分化，与它要求的容量

格IV是不预期一致，也没人管。

这一格里有本文最想要的东西：新的东西。一个基因的副本可以自由变异，是因为坏了不要紧——原来那份还在供着必需的功能。这条路径被认为是生物学创新的主要来源。[4]

但这里必须做本文的第一处让步，而且这处让步来自一个本文一开始并不想要的事实。

多数副本的结局是死。 复制出来的基因，绝大部分并不会长出什么新功能；它们积累失活突变，退化成没有功能的残骸，最后被删掉。[6] 换句话说，那些被反复讲述的创新案例是**幸存者**，而幸存者背后是一大堆被清掉的失败品。

这条事实逼着本文把格IV的说法收窄成一句更谨慎的话：

> **格IV的产出方差极大，它的净收益取决于这个系统能不能承受大量死亡**。 承受不了的系统进这一格，只会得到一堆没人维护、也没人敢删的东西。

而"承受死亡"这件事有一个可查的条件：**必须允许删除**。一个组织如果每一份文档都必须留着、每一个分支都不许删、每一个项目都不能宣告失败，那么它进格IV得到的不是创新，是负担的单调增长。生物学那边之所以能靠这条路径产出新东西，是因为它有一个极其冷酷的删除机制。

这条让步顺带解释了一件常见的事：为什么很多组织在鼓励"百花齐放"之后，收获的不是创新而是混乱。它们进了格IV，却没有格IV所要求的那个容量——**它们不删东西**。

十一、机制并不总是有效

第二处让步来自可靠性工程，它打的是格 I 。

格 I 的程序式实现依赖一个假设：副本之间的错误互相独立，所以多数投票能把错的那个压下去。这个假设在 1986 年被一次实验正面检验过：二十七个程序员按同一份规格说明各自独立写出程序，然后跑一百万个测试用例。结果是每个程序单独看都极可靠，而**多个程序在同一个输入上一起出错的次数，远高于独立性假设所预测的数量**。[7]

原因不神秘，后来的讨论把它说得很清楚：规格说明是有限的，也常常有含混之处；受过相似训练、用同一种语言、读过同一批参考资料的人，会在同一个地方做出同样的误读。**独立地做，不等于独立地错**。

这条对本文的约束是具体的：格 I 里那个"有机制"不能只问有没有，还得问一句——

> **这个机制对相关失效有没有免疫力？** 也就是：这些副本是不是同源？它们是不是同一个模板、同一个团队、同一套工具生成的？

如果是，多数投票提供的保护远小于账面上的份数。这条在今天比在 1986 年更要紧，因为"同一套工具生成"这件事的普及程度已经完全不同了。

于是格 I 也不是一个可以安心待着的格。它只是把"哪一份算数"这个问题回答了；至于答案好不好，是另一层的事。本文能给的只有一条：**答案的可靠性上限，由副本之间的独立性决定，而独立性不能从"分别做的"推出来**。

十二、影子正本为什么这些年在变多

如果格 II 是最坏的一格，为什么它没有一个通行的名字？一个可能的答案是它一直存在而被当成了别的东西——每一次暴露都被记成一次个人失误。本文认为这只是一半，另一半是：**它确实在变多**。

变多的条件有三条，凑齐得越来越容易。

第一，复制的成本降到了零。 复制一份纸质文件要跑一趟，复制一份数字文件是一个动作。成本为零意味着复制不再需要理由，因而也不再留下"为什么要多一份"的记录——而那个记录，恰恰是判断预期的唯一线索。

第二，副本之间的距离变远了。 三份文件存在同一个柜子里，人会顺手对一下；存在三个人的云盘里，永远不会。**格 II 的形成不需要任何人偷懒，只需要没有人碰巧同时看到两份**。

第三，也是最新的一条：现在有大量副本是被生成出来的，而不是被复制出来的。 同一段规则被不同的人用同一套工具各写一遍，得到四份措辞不同、意思相近的表述。它们看上去不是副本——字面上完全不同——所以任何按字面查重的办法都发现不了；但它们表达的是同一条知识，因而完全符合消除派那条原则所定义的重复。**这批副本天生就在格 II 里：预期一致（大家都以为在说同一件事），无人负责，而且连"这是副本"这件事都要费劲才能看出来**。

这里插一句反面的话，免得这三条被读成一部退步史。这三个条件同时也在让格 III 变得更容易：复制便宜、分叉便宜，于是"明确分开、各走各的"这条路的门槛也降低了。开源生态里那些声明不再合并的分叉，二十年前几乎不可能——那要靠邮寄磁带。**同一批条件既在生产格 II，也在生产格 III；决定落到哪一格的，仍然是有没有人把那句话说出来**。

第三条同时把上一节那个约束推到了更坏的位置：这批副本不但同源，而且同源得极其彻底。**它们最像的地方，正是它们一起错的地方**。

这三条还有一个合起来才看得清的后果：**格 II 的形成不再需要任何决定**。 从前多做一份东西是要下决心的——要花钱，要占地方，要有人搬。决心留下痕迹，痕迹里就带着"为什么要多这一份"，而那句话本身就回答了预期问题。现在多一份不需要决心，于是也不留痕迹，于是预期这件事从来没有被表述过——它只是被默认。

一个从未被表述过的预期，是无法被撤销的。 你没法开会宣布取消一件从来没有人正式说过的事；在场的人会觉得莫名其妙。这就是格 II 难处理的深层原因：它的两个成因里，"机制不在"可以靠配人解决，而"预期在"因为从未被说出口，连解除的对象都找不到。

十三、几个容易被当成同一件事的说法

下面逐个处理，每一个都落到具体的判别场景上。

最像的是消除派那条原则本身。"每一条知识必须有单一、无歧义、权威的表示"（Hunt & Thomas, 1999）——它已经站在门口了，因为它是唯一一条把"权威"写进正文的。分离线在**处方**上：那条原则的处方是减少份数；本文说份数与危害无关，要改的是预期或机制。

判别场景：一个有五百份副本、明确宣布不预期一致的生态（分叉后各自发布的开源项目群）。按那条原则判，这是极端违规，事故率应当极高；按本文判，这是格Ⅲ，事故率应当低于同一生态里那些声称保持同步却没有同步流程的项目。两个预测在公开的缺陷追踪数据上会分开。

第二个是"唯一真相源"这一族，在信息系统与主数据管理里是标准做法：为每一类实体指定一个权威系统，其余系统只读。它比上一条更具操作性，也更接近格 I 的位置式。

分离线：这一族默认"指定"本身就解决了问题。本文说指定只完成了一半——**指定之后如果没有回执与核对，它就变成第十八节要讲的那种更坏的形态**。 判别场景：统计主数据项目的失败原因。这一族预测失败主要来自技术集成；本文预测失败主要来自**没有人被授权裁决冲突**——即指定了权威系统，却没规定业务方不服时找谁。

第三个来自分布式系统。 那里有一条广为人知的结论：在网络可能分区的条件下，一致性与可用性不可兼得。这与本文第三节讲的"两种安全"形状相似。

分离线：那条结论是关于**读写操作**的形式化结果，它假设"什么算一致"已经定义好了。本文处理的正是它假设掉的那一层——一致是不是被期待、由谁来裁决。判别场景：一个技术上完全一致的系统（所有节点同一份数据），照样可以处在格Ⅱ——只要那份数据在系统之外还有几份 Excel 副本，而人人以为它们一样。技术一致性与本文说的一致预期，可以取相反的值。

第四个来自社会科学：共同知识。 那一族关心的是"我知道你知道我知道"这种递归结构。影子正本看起来像共同知识的失败：大家以为存在共识，其实没有。

分离线：共同知识的框架关心**信念的层级**；本文关心**职责的空缺**。判别场景：把所有人召集起来，当众宣布"这三份是不一样的"。共同知识的框架预测问题就此解决（信念对齐了）；本文预测**不解决**——除非同时指定谁负责，否则三个月后它们会再次分开。这个实验很便宜，而且大多数组织其实做过，只是没记录结果。

第五个也来自社会科学：未言明的相互预期。 劳动关系研究里有一族概念处理雇员与组织之间没写进合同、却被双方默认的义务。影子正本里那个"应该一样"的预期，形式上确实属于这一类。

分离线：那一族的预期是**关于对方会做什么**；本文的预期是**关于哪一份文本算数**，而后者可以被一句话问出来，前者不能。判别场景：问卷的可回答率。问"你觉得公司应当为你做什么"，答案发散且难以核对；问"这两份不一样时哪份算数"，答案是有限的几项，可以直接统计分歧度。

第六个来自法律与档案：正本与副本的区分。 这套区分极其成熟：正本有原始签名，副本只是复制品，效力不同。

分离线：那套区分预设了**正本一定存在**，问题只是如何辨认与保管。本文的第二格里正本**根本不存在**——所有人以为有，但从来没有任何一份被赋予过那个地位。判别场景：追问"那份正本在哪儿"。法律框架预测追问总能得到一个答案（在某个保险柜里、在某个登记处）；本文预测在格Ⅱ里追问会得到互相矛盾的答案，而且没有人觉得这是件怪事。

第七个是复制与原作的那一族讨论。 那一族讲的是复制取消了原作在特定时空里的独一无二性。

分离线：那一族关心的是原作的**光环**是否消失；本文关心的是原件的**裁决权**如何处置。这两件事可以取相反的值：一份带数字签名的权威文本，光环早已荡然无存，裁决权却清清楚楚。判别场景：给一批人看两份内容相同、其中一份有正式签署的文本，问哪份"更真"、再问哪份"算数"。那一族预测两个问题的答案高度相关；本文预测在数字场景里两者可以分离——多数人会说"都一样真"，但"算数"的那份指认得毫不含糊。

第八个是本文必须处理的一个近处的说法：有一类分析处理的是**原物够不着**时各领域如何应对——手稿失传、原始状态无法观测——并论证不同领域的处置办法造出了不同的替代物。本文与它形状相近而问题不同：那里的原物是**够不着的**，问题是缺口怎么填；本文的所有副本都**在手边**，问题是它们之间谁说了算。两

者在同一个案例上会分岔：一份人人都能打开、只是各自改过的文档，在那套分析里不构成问题（原物并没有丢），在本文这里正是最坏的一格。

第九个来自版本控制。 现代的版本控制系统提供了一个几乎完美的格 I 实现：任何两条分支的差异都可以被机械地检出，合并时冲突会被强制摆到人面前，要求当场裁决。它比本文说的任何机制都硬——它不给人"没注意到"的余地。

分离线在**覆盖范围**上：那套机制只对进了库的东西有效。而现实中的格 II，绝大多数发生在库外——邮件附件里的那份、打印出来贴在墙上的那份、某人本地改过没提交的那份、按同一条规则各自写出的四份说明。判别场景：统计一个团队的不一致事故，看有多少发生在版本控制覆盖的对象上。那套机制的框架预测事故集中在合并冲突处理不当；本文预测**绝大多数事故发生在库外**，且与库内的规范程度无关。这条在任何一个用版本控制的团队里都能查。

这九个里，最近的仍是第一个。 那条原则是唯一一条把"权威"这个词写进定义的，也是唯一一条同时管到了知识与表示两层的。它和本文之间只差一个动作：它把"权威"和"单一"焊在了一起。一旦把这两个词拆开——权威要有，单一不必——它的全部反例（多副本网络、多语种条约、受控分岔）就各归其位了。

本文接的正是它焊住的那个地方。所以准确的说法不是本文推翻了它，而是：它在"改一处会漏"这个问题上是对的，而它把处方定得比它的诊断窄。

十四、在别的行当里应该看到什么

一条判断如果只能在提出它的那几个例子上成立，它就还没离开那几个例子。下面十二处，每一条都是可以查的。

医疗。 同一份指南被各科室改编。预测：不一致的暴露**集中在转诊与会诊**，院内单科使用时几乎不暴露；且不良事件的归因记录里，这类事件会被高比例记成"沟通问题"而非"文件不一致"。查法：调阅不良事件报告，看"指南版本"是否曾作为一个字段出现过。

法律实务。 同一段条款被复制进多份合同并各自演化。预测：争议中该条款的解释分歧率，与该事务所是否设有条款库管理员**强相关**，与条款被复制的次数**不相关**。这条可以在一家规模足够的事务所内部做回溯。

教育。 同一门课的课件由多位教师各自维护。预测：跨班成绩差异在**统考**科目上显著、在各班自主命题的科目上不显著；且教师本人普遍认为自己用的是"同一份课件"。

药品说明。 同一药物在不同国家的说明书各自更新。预测：不一致在跨境医疗与药品代购时暴露；且更新滞后最长的那一版，恰恰是被最多人当作"权威版本"引用的那一版——因为它最老、最容易被搜到。

统计口径。 同一个业务指标在多个报表系统里各有实现。预测：口径分歧的暴露发生在**跨部门汇报**的场合，而不是日常运营；且分歧被发现后的第一反应是"以哪个系统为准"这句话本身第一次被问出来。

工程图纸。 同一零件的图纸存在设计、工艺、供应商三处。预测：不一致在**首件检验**时暴露，且返工成本与副本数无关、与是否设有更改通知回执强相关。

语言。 有标准化机构的语言与没有的语言。预测：前者的方言分化照样发生（格III），但在正式场合的裁决是明确的；后者在需要统一书写时会出现典型的格 II 症状——每一方都认为自己写的是"正确的"。

宗教文本。 抄本传统里那些设有逐字计数与核对制度的（例如把每卷的字数、中间字都记下来备查），与没有的。预测：前者的异文率显著低，而且异文的**分布形态**不同——前者集中在计数不覆盖的层面（读音、断句），后者散布在全文。

开源软件。 分叉后明确宣布不再合并的项目（格III），与把上游代码直接复制进自己仓库、又声称会跟进上游的项目（格II）。预测：后者的安全漏洞修复滞后期显著更长，因为没有人被指定去跟进；且滞后期与项目活跃度无关。

主数据管理。 预测：这类项目的失败复盘里，"没有人被授权裁决业务口径冲突"出现的频率高于任何技术原因。这条可以在公开的失败案例复盘中统计。

分布式系统之外的数据。 预测：一个技术上强一致的系统，其周边的电子表格副本数量与该系统的"权威性宣称强度"**正相关**——越是被宣布为唯一真相源，私下的影子副本越多，因为业务方的例外需求无处安放。这条反直觉，也最容易做。

生物学（反向）。 预测：在基因组里，被保留下来的重复基因中，通过各自丢掉不同部分而互补保住的那一类，其比例应当高于长出全新功能的那一类。这条已有文献支持，本文借用它作为格III在自然界的存在证据，不算独立证据——明写。

十二处里，教育与统计口径那两条最便宜；"权威性宣称越强、影子副本越多"那条最硬，因为它的方向与常识相反。**如果那一条查下来是负相关，本文关于格 I 位置式实现的整段判断就要重写。**

十五、怎么在半小时里查完一个部门

这一节交出一份操作办法，因为一条判断如果只能被同意、不能被执行，它离一个说法就不远了。

第一步，列副本，不看内容（十分钟）。 挑一件这个部门真的在用的东西——一份标准、一份清单、一段条款、一个指标口径。然后问在场的人：这东西你手上有吗，存在哪儿。把所有位置记下来，不打开，不比对。这一步只求数量与位置，不求差异；打开比对是后面的事，先打开会让所有人立刻开始辩论谁对，而那正是要避免的。

第二步，各问两句（十分钟）。 对每个持有者分别问，不要当众问——当众问会收敛到一个答案，而分歧本身才是读数。

第一句："这几份应该是一样的吗？" 第二句："如果不一样，哪一份算数？"

记原话，不要归纳。 "按共享盘那份"和"按最新的那份"看起来接近，其实是两个完全不同的答案：前者指向一个位置，后者指向一个属性，而这两个在某次修改之后会指向不同的文件。

第三步，看最后修改时间（五分钟）。 不比对内容，只看每一份最后一次被改是什么时候。时间跨度就是已积累分歧的粗略上界。这一步之所以放在这里，是因为它便宜到几乎没有成本，而它给出的量级估计通常比逐行比对更早派上用场。

第四步，判格（五分钟）。

- 第一句多数答"应该一样"，第二句答案一致且能指出谁负责 → 格 I 。到此结束。
- 第一句多数答"应该一样"，第二句答案分歧，或出现"应该都一样吧"这种反问 → **格II**。
- 第一句多数答"不用一样"，并能说出从什么时候、按什么规矩分开 → 格III。
- 第一句多数答"不用一样"，说不出规矩，且这些东西谁也不删 → 格IV，且缺容量。

判成格II之后，先别急着解决。 按第十七节那条硬规矩，只补一半比不补更糟。在配上人和核对动作之前，能做的最有用的一件事是把第二步的原话贴出来——让所有人看到彼此的答案不一样。这一步不解决问题（第十三节里已经说明，光对齐信念不解决问题），但它把一个从未被表述过的预期第一次变成了可以被讨论的东西，而第十二节说过，从未被表述的预期连撤销的对象都找不到。

最后一句提醒：**这套办法查得出格，估不出损害**。 分歧度告诉你这是哪一格，告诉不了你已经付出了多少代价——因为代价的暴露由使用决定，而使用的分布这套办法没有采集。要估损害，得另外去看合流处：转诊、审计、统考、争议、跨部门汇报。

十六、这条判断管不到的地方

第一，"一致"在有些领域不可定义。 两份手工制品、两次演出、两个译本，说它们"一致"没有意义。这类场合本文的两条轴都失效。

第二，预期是分布的，不是二值的。 一个组织里往往有一半人预期一致、一半人不预期。本文按"多数是否预期"简化，边缘案例会失灵，而边缘案例并不罕见。

第三，"机制"没有分级。 一个每年核对一次的流程与一个实时同步的程序，在本文这张表里落在同一格。这是一个真实的粗糙之处：格Ⅰ内部的差别可能比格Ⅰ与格Ⅱ的差别还大。

第四，本文不处理谁有权设定预期。 "这些应当一致"这句话由谁说了算，是一个权力问题，本文完全没碰。而在很多真实场合，格Ⅱ的形成正是因为有人有权提出预期、却无权配置人手。

第五，安全关键领域不适用格Ⅳ。 在飞机、核电、药物这类场合，即使不预期一致也不能容忍自由分化，因为"承受死亡"这个条件在道义上不成立。

第五之二，本文假设"哪一份算数"这个问题有意义。 在一些场合它没有意义——两份都算数且必须同时满足（例如两个监管机构各自的要求），此时不一致不是待裁决的冲突，而是一个真实的两难。本文的判据在这类场合会把健康的两难误判成格Ⅱ。

第六，本文给不出"预期覆盖范围"的度量。 承重判断里那句"危害的量由预期覆盖的范围决定"，本文只能定性地说，量不出来。这是本文最想要而没有的东西。

十七、怎么判它错

六条，读数各不相同。

判据一（低成本、正向读数）。 在同一个组织里选若干组持有多份同类文件的人，各问两句："这些应该一样吗？"和"如果不一样，哪份算数？"记录第二问的**分歧度**与该组的历史不一致事故率。本文预测：事故率与分歧度显著相关，**与副本数量不相关**。若事故率与副本数量强相关而与分歧度无关，本文的核心（数量无关、预期有关）失败。这条只要一份问卷和一份工单记录。

判据二（打反噬预言）。 找一组处在格Ⅱ的对象，只做一件事：正式宣布"以某一份为准"，不配任何核对流程。本文预测事故率**不降反升**，且暴露时点后移。若事故率单调下降，第十六节整节作废。

判据三（打格Ⅳ）。 在不预期一致、也无人管的系统里，比较允许删除与不允许删除的两类。本文预测后者的维护成本单调上升而新产出接近于零。若两类无差异，"容量"这个条件不成立。

判据四（打格Ⅰ）。 对采用多数裁决的系统，比较副本同源与不同源两组的相关失效率。本文预测同源组显著更高。**这条已有 1986 年那次实验支持，属借来的证据，不计入本文的独立检验**。

判据五（打暴露时点那条读数）。 格Ⅱ的不一致应当**只在合流处**暴露。若在单点使用中也高频暴露，那么"没有单点读数"这条性质错了，第七节的三条性质要重写。

判据六见末节的赌注：它有截止日期，到期不兑现即作废。

另外，本文主动交出两个填不上的洞：**预期覆盖范围量不出来**（第十六节第六条），以及**本文说不清预期是怎么形成的**——为什么有些副本一出生就带着"应该一样"的预期，有些不带。本文只能观察它在不在，说不出它从哪来。

十八、按它设计干预会怎样出事

本文的处方最容易被砍掉一半，而砍掉一半比不做更糟。

格Ⅱ的病因是两条：预期在、机制不在。因此出路只有两条——**要么把机制补上，要么把预期撤掉**（明确宣布这些从此不必一致，并说清合流处怎么办）。两条都行，但必须整条做完。

现实里最省事的做法是发一份通知：从即日起以某某版本为准。这一步是**只补了预期的正式性，没补机制**。它的后果不是中性的，是负的：

- 从此所有偏离都成了违规。而各分公司真实存在的差异并没有消失——报销场景确实不同、病人构成确实不同。
- 于是偏离不再上报，转入地下：私下维护一份"实际在用的版本"，对上声称用的是标准版。
- **暴露时点被推得更远**，因为现在多了一层掩盖；而暴露时的损害更大，因为积累得更久。
- 更麻烦的是，这套通知会让组织觉得问题已经解决了——账面上有了一份权威文件，检查时人人都答得出"以总部版为准"。**格Ⅱ因此变得更难被发现，而不是更容易**。

所以本文把处方写成一条硬的：**"指定正本"这个动作，如果不同时指定一个人和一次核对，就不要做**。 什么都不做，至少还留着一个诚实的读数——问起来大家答案不一致。发了通知之后，那个读数就没了。

第三种出事方式来自这条判断本身的方便。 一旦"要么补机制、要么撤预期"这句话被接受，撤预期会显得比补机制便宜得多——补机制要人，撤预期只要一句话。于是可以预见一种应对：把大量本该一致的东西宣布为"不必一致"，从而在账面上把格Ⅱ变成格Ⅲ。

这个动作在形式上完全符合本文的处方，实质上是把风险从台面转到台下。本文能给的鉴别办法只有一条，而且它落在合流处：**格Ⅲ必须交代合流处怎么办**。 分叉的项目要说清与上游不再兼容，标准化机构要说清考试按哪一套。**一次不交代合流处的"宣布不必一致"，不是格Ⅲ，是给格Ⅱ换了个名字**。 判据很简单：问一句"那么两边碰上的时候按什么"，答不出来就是换名字。

第四种出事方式是滥用。"影子正本"这个说法很好用，因为它以"找不到那份权威"为特征，所以任何一次配合失败都可以被这么描述，而且无法被反驳。本文能给的唯一防线是那句一分钟判据必须**真的去问，并记下每个人的原话**：如果所有人答案一致，这个词就不适用。这条防线不牢，但它要求提出主张的人交出一次可核对的记录，而不只是一个说法。

十九、这篇文章自己在哪一格

按前面的规矩，一条判断要问它对自己适不适用。**本文处在第二格，而且已经因此出过一次事**。

支撑本文这条判断的那套做法，被写在好几个地方：一份工作规程、一份供检索用的条目库、一份用来自查的清单，以及若干篇正文里对同一条判断的复述。这些是同一条知识的多份表示。**我预期它们一致；没有任何人被指定去核对它们**。

后果是具体的，而且是可查的。就在写作本文之前不久，同一套规程被两个互不知情的执行过程各自读了一遍，各自按同样的约束选源、各自撞出同一条判断、各自写成一篇近两万字的文章——两篇的节标题逐节相

同。直到发布前的例行检查，才发现其中一篇已经在几个小时前被发出去了。**没有任何机制在这两条线之间报警，因为从来没有人被指定去看它们是不是同一件事。**

这个例子有一个让本文尴尬的性质：它同时也是本文的证据。第八节说格 II 的暴露由使用决定、不由损害决定——那两条线的重复从一开始就存在，而暴露发生在第一次要把它们同时放上去的那一刻。第八节说暴露之后通常被消化成一次个人失误——第一反应确实是"这次没查"，而不是"我们缺一个机制"。

所以本文对自己的处置，按第十七节那条硬规矩：不发通知。要么给那份规程配一个核对的动作，要么明确宣布各条线不必一致、并说清合流处（发布前）怎么办。在两者之一被真的做出来之前，本文不宣称自己已经离开第二格。

还有一点得说明。本文的可信度不能靠"作者做到了"来担保——作者显然没有做到。它只能押在第十四节那十二处读数与第十七节那六条判据上：若那些查下来是错的，作者做没做到都不重要；若查下来是对的，作者没做到也不构成减损。

二十、一个到期的赌注

到 2032 年 12 月 31 日，押的是这样一件事。

在软件工程或知识管理的主流规范里，出现一条被广泛采用的实践，其形式是**"允许重复，但必须声明是否预期一致，以及不一致时按谁"**——也就是把处方从"消除重复"改成"标注预期"。它可以长成一个字段、一条模板里的必填项、一份检查表上的一行，形式不重要。

如果到期没有出现，那不算数——没出现可以有很多原因。

真正押的是反面：如果它出现了，被广泛采用了，而相关的不一致事故率没有变化——也就是说，标注了预期而事故照旧——那么本文的承重判断（危害由预期而非数量决定）就被推翻了，本判断作废。

还要说明为什么押的是这个而不是别的。本文完全可以押一条更容易赢的：比如押"影子正本"这个说法会流行起来。但一个说法流行与否，跟它对不对没有关系，而且流行这件事本文自己也能推动，押它等于押自己。**押规范的形态就没有这个毛病**——它要求一个本文影响不到的、由许多人的实践共同决定的东西发生改变，而且改变的方向是可辨认的：从"消除重复"改成"标注预期"。

我倾向于认为它会出现。但这句话不值钱：押注的价值在于它有截止日期，不在于押注的人有多少信心。

注释

[1] 该原则的原句为"Every piece of knowledge must have a single, unambiguous, authoritative representation within a system"，出自 Andrew Hunt 与 David Thomas 的《The Pragmatic Programmer》（1999）。本文只取其原始表述，不取后来被简化成"不要复制粘贴"的通俗版本——两者的差别在正文 2.1 节已说明。

[2] 多副本互相校验、按多数修复、且不设正本的做法，本文取自数字保存领域的多副本保存网络实践。具体实现细节各系统不同，本文只用其结构。

[3] 多语种条约各文本同等作准、歧义时依条约目的与宗旨解释，见《维也纳条约法公约》第 33 条。**引用前请按官方文本核校条款序号与措辞**，本文只用其结构，不引原文。

[4] 复制之后一份继续承担原功能、另一份因而得以自由变异，这一路径通常追溯到 Susumu Ohno 的《Evolution by Gene Duplication》（1970）。

[5] 两份各自丢掉不同子功能因而都被保住的路径（复制—退化—互补），见 Force et al.（1999）。

[6] 重复基因的多数最终丢失、退化为假基因，见 Lynch & Conery（2000）及此后的综述。本文以此作为对格IV的约束。

[7] 二十七个独立编写的版本在同一输入上共同失效的次数远超独立性预测，见 Knight & Leveson（1986）。后续讨论把原因归于规格说明的含混与编写者共享的训练背景。

[8] 本文所引各家的年份与表述按公开检索结果录入；页码须按所用版本核校。文中的核心检验**均未执行**，交付的是命题、判据与方案。

参考文献

Force, A., Lynch, M., Pickett, F. B., Amores, A., Yan, Y. L., & Postlethwait, J. (1999). Preservation of duplicate genes by complementary, degenerative mutations. *Genetics*, 151(4), 1531–1545.

Hunt, A., & Thomas, D. (1999). *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley.

Knight, J. C., & Leveson, N. G. (1986). An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1), 96–109.

Lynch, M., & Conery, J. S. (2000). The evolutionary fate and consequences of duplicate genes. *Science*, 290(5494), 1151–1155.

Ohno, S. (1970). *Evolution by Gene Duplication*. Springer-Verlag.

Reich, V., & Rosenthal, D. S. H. (2001). LOCKSS: A permanent web publishing and access system. *D-Lib Magazine*, 7(6).

United Nations. (1969). *Vienna Convention on the Law of Treaties*, Article 33.

字数：正文约 1.67 万汉字 · 版本 v1.0 · 2026-08-01 · 作者 王德生 ＋ Claude

人机分工声明：本文的三家源材料均为站外既有文献（见注释与参考文献）；两条轴的设定、四格表、一分钟判据、以及第十九节的自我定位由本次写作产出。全文由机器一次写成，人类确定选题与验收标准。第十九节所述的那次重复事件属实，且尚未被处置。

附：与本栏另外两篇的关系

本栏另有两篇与本篇形状相近而问题不同，正文里已各划一刀。《我们说的「本来的样子」，是我们造出来的》处理的是**原物够不着**时各领域如何应对，问题是缺口怎么填；本篇的所有副本都在手边，问题是它们之间谁说了算——一份人人都能打开、只是各自改过的文档，在那一篇里不构成问题，在本篇这里正是最坏的一格。《痕迹能不能被复制，决定了这门学科相信什么》问的是痕迹的可复制性如何塑造一门学科的默认判断，自变量在**证据的物理性质**上；本篇的自变量在**看着副本的人有没有预期**上，两者正交。

这一篇撞的三家，与可点开核对的出处

本篇的三家源全在站外，分属软件工程、数字保存与条约法、演化遗传学——三家谁也不读谁，却在同一个事实上给出不能并存的判决。下面三条给出可直接点开核对的原始出处。正文第二节逐一交代三家各自说到

了哪一步，第三节把三处对顶写死，第五节说明它们各自为什么看不见本文那一格，第十三节再与九种最容易被混为一谈的说法逐条分界——**包括本文承认最可能把它吸收掉的那一个**。

软件工程 · 不要重复原则（Hunt & Thomas, 1999）
每一条知识，在一个系统之内必须有单一的、无歧义的、权威的表示
它说的不是「不要复制粘贴」——两段代码长得一样却表达不同的事，不算违反；长得完全不同却表达同一条规则，才是。**与本文的分离线**：这一家的处方是减少份数；本文说份数与危害无关，要改的是预期或机制。一个有五百份副本、明确宣布不预期一致的生态，按它判是极端违规，按本文判是健康。
数字保存与条约法 · 多副本互校，不设正本
多存几份并且不断互相比对，按多数修复；多语种条约各文本同等作准，歧义时依条约目的解释
它的关键动作不是复制，是校验；而它主动取消正本，是为了让权威落在**程序**上而不落在**某一份**上。**与本文的分离线**：这一家假设一致总是目标，因而在「不预期一致」的场合给不出任何指导；而在本文那最坏的一格里，加副本只会更糟——它的药在那里是毒。
演化遗传学 · 基因重复（Ohno 1970；Force et al. 1999）
复制之后一份继续供着原功能，另一份因而承受得起突变；或两份各丢掉不同部分而都被保住
这条路径被认为是生物学创新的主要来源，而它成立的前提是没有任何机制要求两份一致。**与本文的分离线**：这一家的世界里没有「预期」这个变量——基因组里没有人期待两份一样，所以「以为该一样、其实不一样」这种情形在那里根本不成立，而那正是前两家吵得最凶的地方。