面向测试人员的离散数学

同软件生命周期的其他活动相比，软件测试更多地依赖于数学描述和数学分析。本章和下一章将给出测试人员所需的数学知识。还是把测试人员类比成技艺师，此处介绍的各种数学方法就是工具，测试技师应该是知道如何用好这些工具的。借助这些工具，测试人员能够严谨、精确和高效地完成工作，所有这些都能提高测试工作质量。在本章标题中，“测试人员的”这个限定词非常重要。因为本章是针对那些只有初浅数学基础知识，或者已经忘了大部分数学知识的测试人员编写的，所以真正的数学家很可能会对本章中不严密的讨论有看法。已经读者很熟悉离散数学的基础知识，可直接跳到下一章学习图论。

一般情况下，离散数学更适于功能测试，而图论更适于结构测试。“离散”这个词会引发一个问题：数学上的非离散又是怎样的呢？在数学上，“离散”的反义词是“连续”，比如像在微积分中那样，但软件开发人员和测试人员很少能用到这些。离散数学包括集合论、函数、关系、命题逻辑和概率论，本章将逐一讨论这些内容。

3.1 集合论

高谈阔论了这么多严格和精确之后，才尴尬地发现，竟然找不到对集合的明确定义。这可就麻烦了，因为集合论是这两章数学内容的核心。数学家对集合的概念进行了重要的区分：朴素集合论与公理集合论。在朴素集合论中，集合被当作是一个基本概念，就像点和线这样的几何学中的基本概念一样。集合的同义词有许多，比如堆、组、束等，这样你可以领悟到其中的含义了吧。集合概念的重要意义在于，它使我们能够把若干个事物作为一组或一个整体来考察。比如，我们可能需要研究正好有 30 天的所有月份（在测试第 2 章的 NextDate 函数时就会用到这个集合）。用集合论的表述方法，这个集合写作：

$$M_1 = \{4\text{ 月}，6\text{ 月}，9\text{ 月}，11\text{ 月}\}$$

读作：M_1 是元素为 4 月、6 月、9 月和 11 月的集合。

3.1.1 集合的成员关系

集合中的各项称作集合的元素或成员，这种成员关系用符号“$\in$”来标记，例如，4 月 $\in M_1$。如果不是集合的成员，则用符号“$\notin$”来表示，比如：12 月 $\notin M_1$。

3.1.2 集合的定义方法

定义一个集合的方法有三种：简单地列举出各个元素，或者给出一个判别规则，抑或通过其他集合来构造这个集合。列举元素的方法适用于只有少量元素的集合，或元素符合某种简明形式的集合。前面定义集合 M_1 时采用的就是这种方法。可以定义 NextDate 程序中可取的年份的集合为：

$$Y = \{\ 1812，1813，1814,\cdots，2011，2012\ \}$$

在列举元素来定义集合的方式中，集合元素之间是没有顺序关系的。其原因在后面讨论

集合相等时就会弄明白。采用判别规则的方法要复杂一些，这种复杂性有利也有弊。比如可以把 NextDate 函数的允许年份集定义为：

$$Y = \{ year：1812 \leqslant year \leqslant 2012 \}$$

读作：Y 是年份的集合，要求（可以把冒号读作“要求”）年份值介于 1812 和 2012 之间（包含 1812 和 2012）。利用判别规则来定义集合时，规则必须是无歧义的。给定任何一个年份值，就可以判断这个年份是否在集合 Y 当中。

用判别规则来定义集合的优点在于无歧义，这就要求有清晰的表述。有经验的测试人员都遇到过“无法测试的需求”。究其原因，很多时候都可以归结为在判断规则上出现了歧义。比如，在前面的三角形问题中，假设定义一个集合为：

$$N = \{ t：t \text{ 为近似等边的三角形} \}$$

那可以断定边长为（500，500，501）的三角形是 N 的元素，但是对边长为（50，50，51）或（5，5，6）的三角形又该如何判断呢？

用判别规则定义集合的第二个优点在于，某些集合的元素可能很难列举出来，而我们需要使用这样的集合。比如，在佣金问题中，我们可能会对这样的集合感兴趣：

$$S = \{ sales：\text{对该销售额 } sales \text{ 来说，佣金比例应为 } 15\% \}$$

想要一一列出这个集合的每个元素不容易实现，但是对于给定的销售额，却可以很容易地使用这个判别规则来处理。

判别规则法的主要缺点是，规则在逻辑上有可能会相当的复杂，特别是需要采用谓词逻辑量词“∃”（存在）和“∀”（所有）来表述时。如果大家都能理解这种表示方法，那精确性就大有用途了。但这些包含逻辑运算符的规则往往都会把客户给搞糊涂了。判别规则法的第二个问题是自引用问题。自引用问题很有意思，但实际上对测试人员用处并不大。在判别规则指代它自己时就是自引用，就会出现死循环的问题。比如塞维利亚理发师问题（Barber of Seville）：塞维利亚的理发师是给所有人理发但不给自己理发的人。

3.1.3 空集

空集记为 ∅。空集在集合论中占有特殊地位。空集不包含任何元素，所以数学家能给出一大堆关于空集的属性，比如：

- 空集是唯一的，即不存在两个空集（我们姑且接受这种说法）；
- ∅、{∅} 和 {{∅}} 是不同的集合（我们其实并不需要用到这个性质）。

对我们有用的一点是，当定义集合的是一条永远为假的判别规则时，这个集合就是空集。比如：$\varnothing = \{ year：2012 \leqslant year \leqslant 1812 \}$。

3.1.4 集合的维恩图

有两种传统的用图解表示集合之间关系的方法：维恩图和欧拉图。这两种方法使在本文中已经被表达的概念更加形象化。我的大学数学系教授说，“数学不是图论的函数”。这种说法或许不太正确，但是图表确实很具有表达性，并且更易沟通和理解。然而现在，在讨论规定行为集合和实现行为集合时，常常像在第 1 章中那样把集合画成维恩图。在维恩图中，集合用圆圈来表示，圆圈内部的点表示该集合的元素。这样包含 30 天的月份的集合 M_1 就可以表示成图 3-1 所示的形式。

维恩图最初被约翰维恩图（一个生于 1881 年的英国逻辑学家）设计。大部分维恩图显

示两个或三个重叠的圈。(显示一个可以显示所有可能交集的五个集合的维恩图是不可能的。) 阴影部分被用于两种截然相反的方式，即：大部分情况下，阴影区域是有意义的子集，但是，阴影部分偶尔也会被表示成一个空的范围。因此，包含一个可以明确表述阴影部分意义的图例是很重要的。维恩图也可以被用在一个代表论域的矩形中。第 1 章的图 1-3 和图 1-4 展示了两个和三个集合的维恩图的例子。当这些圈叠加时，集合之间的关系就推测不出来了；同时，叠加描述了所有可能的交集。最后，没有用图解表示空集的方法。

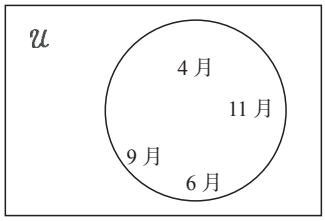

图 3-1　包含 30 天的月份的集合的维恩图

维恩图能够以一种直观的方式表示各种集合关系，但是也会有一些小问题，比如如何表示有限集合与无限集合呢？这两种集合都可以用维恩图来表示，只是对有限集合来说，不要把集内的每个点都对应集合的元素就行了。这不会有什么大问题，了解了这个限制很有用，比如在确定特定集合元素的标签的时候。

维恩图的另一个问题涉及空集：如何来表示一个集合或集合的一部分是空的呢？通常可以用阴影来表示空集部分，但有时也用阴影来强调感兴趣的部分。为了避免混淆这两种情况，更好的办法是给出图例，明确说明阴影部分的含义。

通常情况下，需要把所讨论的所有集合都看作是某个更大集合的子集，这个大集合就称为论域。比如，在第 1 章中就把所有的程序行为当作论域。一般情况下根据所给出的集合可以猜出问题的论域。在图 3-1 中，大多数人会把一年中所有月份构成的集合作为论域。测试人员应该警惕的是，随意假设论域经常会造成混乱。这是引起客户和开发人员之间误解的一个微妙的原因。

3.1.5　集合运算

集合论的大部分表达能力源自对集合的基本运算操作，如并、交和补。其他常用的运算还有相对补、对称差和笛卡儿积。下面逐一定义这些运算。在讨论每个运算时，首先研究论域 U 中两个集合 A 和 B。运算的定义中使用了谓词演算中的逻辑连接符与（$\wedge$）、或（$\vee$）、异或（$\oplus$）和非（$\sim$）。

定义

给定集合 A 和 B，各个运算的定义如下。

- A 和 B 的并集是集合 $A \cup B$：$A \cup B = \{ x : x \in A \vee x \in B \}$。
- A 和 B 的交集是集合 $A \cap B$：$A \cap B = \{ x : x \in A \wedge x \in B \}$。
- A 的补集是集合 A'：$A' = \{ x : x \notin A \}$。
- B 对 A 的相对补是集合 $A - B$：$A - B = \{ x : x \in A \wedge x \notin B \}$。
- A 和 B 的对称差是集合 $A \oplus B$：$A \oplus B = \{ x : x \in A \oplus x \in B \}$。

以上集合运算的维恩图如图 3-2 所示。

维恩图直观的表达能力对描述测试用例之间以及被测软件之间的关系十分有用。分析图 3-2 中的维恩图可以看出：

$$A \oplus B = (A \cup B) - (A \cap B)$$

事实也正是这样，此外还可以利用命题逻辑证明这个关系。

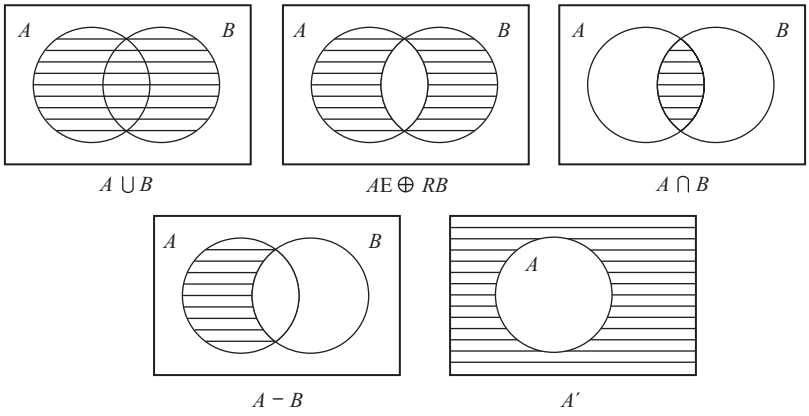

图 3-2 基本集合运算的维恩图

维恩图在软件开发的其他方面也能派上用场：与有向图相结合，就构成了状态图表示法的基础。状态图是 CASE 技术支持的最严格的规格说明方法之一，也是 IBM 公司和 OMG 集团（Object Management Group）为 UML 选定的控制表示方法。

两个集合的笛卡儿积运算（也叫叉积）要复杂一些，它基于有序偶对的概念。有序偶对是两个元素的组合，这两个元素出现的顺序至关重要。无序偶对和有序偶对通常的表示法为：

- 无序偶对记为（a，b）
- 有序偶对记为 $<a, b>$

二者的差别在于，对于 $a \neq b$，有（a，b）=（b，a），但是 $<a, b> \neq <b, a>$。这个差别对于第 4 章中的内容十分重要。可以看出，普通图和有向图之间的根本差别恰恰就是无序偶对和有序偶对之间的差别。

定义

两个集合 A 和 B 的笛卡儿积定义为：

$$A \times B = \{ <x, y> : x \in A \wedge y \in B \}$$

维恩图不能表示笛卡儿积，所以这里需要举一个简单的例子来说明一下。设集合 $A = \{1, 2, 3\}$，$B = \{w, x, y, z\}$，则 A 和 B 的笛卡儿积为：

$$A \times B = \{ <1, w>, <1, x>, <1, y>, <1, z>, <2, w>, <2, x>, <2, y>, <2, z>, <3, w>, <3, x>, <3, y>, <3, z> \}$$

笛卡儿积同算术运算直接相关。集合 A 的势定义为 A 中元素的个数，记为 $|A|$。（有的人喜欢用 Card(A) 来表示。）给定集合 A 和 B，有 $|A \times B| = |A| \times |B|$。在第 5 章中研究功能测试时，将利用笛卡儿积来描述多输入程序的测试用例。集合笛卡儿积的势的乘法特征意味着这种测试方法会产生大量的测试用例。

3.1.6 集合关系

可以利用集合运算从现有集合构造出有趣的新集合。在进行此类运算时，常常需要了解新集合与已有集合之间的关系。给定两个集合 A 和 B，定义三种基本的集合关系如下：

定义

- 当且仅当 $a \in A \Rightarrow a \in B$ 时，称 A 是 B 的子集，记为 $A \subseteq B$；

- 当且仅当 $A \subseteq B \wedge B - A \neq \varnothing$ 时，称 A 是 B 的真子集，记为 $A \subset B$；
- 当且仅当 $A \subseteq B \wedge B \subseteq A$ 时，称 A 和 B 相等，记为 $A = B$。

用文字来叙述就是，如果 A 的每个元素也是 B 的元素，则 A 是 B 的子集。要成为 B 的真子集，A 必须首先是 B 的子集，而且 B 中必须包含不属于 A 的元素。最后，如果 A 与 B 互为子集，则 A 和 B 相等。

3.1.7　集合划分

集合的划分是一种非常特殊的情况，对软件测试人员来说至关重要。日常生活中有很多类似集合划分的情况。比如，把一个办公区域分割成多个独立办公室，这是空间上的划分；把一个州分成若干个行政区，这是行政区域的划分等。从这两个例子中可以看出，“划分”的含义是把一个整体分割成若干小的部分，并保证所有的事物都会出现在某个部分中，没有遗漏。集合划分的正式定义如下。

定义

给定集合 A 和一组 A 的子集 A_1，A_2，…，A_n，当且仅当 $A_1 \cup A_2 \cup \cdots \cup A_n = A$，且 $i \neq j \Rightarrow A_i \cap A_j = \varnothing$ 时，称这些子集是 A 的一个划分。

由于划分实际上是一组子集的集合，因此也经常把单个子集称为划分的元素。

对于测试人员来说，集合划分定义中的两部分内容都十分重要：第一部分保证了集合 A 的每个元素都会出现在某个子集中，第二部分保证了 A 中没有任何元素会同时出现在两个子集中。

这一点在行政分区的例子中表现得非常好：每个行政区都设有立法委员，不存在设两个立法委员的区。拼图游戏是另一个很好的划分实例。实际上，集合划分的维恩图就常常画成类似拼图的样子，如图 3-3 所示。

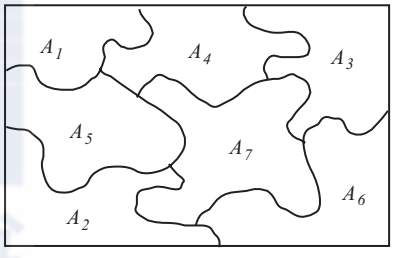

图 3-3　集合划分的维恩图

集合划分的概念对测试人员非常有用，因为划分定义的这两个性质保证了对测试的两个重要要求：完备性（每个事物都在某个确定地方）和无冗余性。对功能测试来说，其固有的缺点就是缺漏和冗余这两个问题：有些东西始终测试不到，而有些又被反复地测试。如何找到恰当的划分是功能测试的一大难点。比如，在三角形问题中，论域是所有正整数三元组的集合。（这实际上是正整数集与其自身的三次笛卡儿积。）对此论域可以有 3 种划分方法：

（1）划分成：三角形和非三角形；

（2）划分成：等边三角形、等腰三角形、一般三角形和非三角形；

（3）划分成：等边三角形、等腰三角形、一般三角形、直角三角形和非三角形。

乍看起来这三种划分方法好像都是可行的，但是仔细一研究就会发现最后一种划分形式有一个问题。一般三角形和直角三角形这两类并不是不相交的（比如，边长为 3、4、5 的三角形既是一般三角形也是直角三角形）。

3.1.8　集合恒等

把集合运算和集合关系结合在一起，能推导出一系列重要的集合恒等关系，利用这些恒等关系可以大大简化复杂的集合表达式。数学系的学生通常是要亲自推导出这些恒等关系，

而这里只是简单地列出这些等式（见表 3-1），并偶尔用一下。

3.2 函数

函数的概念对软件开发和软件测试都至关重要。在功能分解的整个方法体系当中，就使用了函数的数学概念。不严格地讲，函数把多个集合的元素关联起来了。比如，在 NextDate 函数中，给定日期的函数是其下一天的日期；在三角形问题中，三个输入整数的函数是以此为边长的三角形的种类；而在佣金问题中，销售人员的佣金是销售额的函数，销售额又是枪机、枪托和枪管销售量的函数；SATM 系统中的函数要更复杂一些，这自然也会使测试变得更加复杂。

表 3-1 集合的恒等关系

定律名称	表达式
同一律	$A \cup \varnothing = A$ $A \cap U = A$
支配律	$A \cup U = U$ $A \cap \varnothing = \varnothing$
幂等律	$A \cup A = A$ $A \cap A = A$
自反律	$(A')' = A$
交换律	$A \cup B = B \cup A$ $A \cap B = B \cap A$
结合律	$A \cup (B \cup C) = (A \cup B) \cup C$ $A \cap (B \cap C) = (A \cap B) \cap C$
分配律	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
德摩根定律	$(A \cup B)' = A' \cap B'$ $(A \cap B)' = A' \cup B'$

任何程序都可以看作是把输出同输入关联起来的函数。在函数的数学表达形式中，所有的输入是函数的定义域，所有输出构成了函数的值域。

定义

给定集合 A 和 B，函数 f 是 $A \times B$ 的一个子集，对于 a_i、$a_j \in A$，b_i、$b_j \in B$ 和 $f(a_i) = b_i$，$f(a_j) = b_j$，则有：$b_i \neq b_j \Rightarrow a_i \neq a_j$。

像这样的形式化定义有些过于简洁了，所以要进一步仔细研究一下。函数 f 的输入是集合 A 的元素，输出是集合 B 的元素。在以上定义中，函数 f “表现良好”，其含义是 A 的任何元素永远都不会与 B 的多个元素相对应。（如果出现了这种一个对应多个的情况，那怎么来测试这样的函数呢？这是非确定性的一个示例。）

3.2.1 定义域与值域

在前面给出的函数定义中，集合 A 是函数 f 的定义域，集合 B 是值域。由于从输入到输出的对应关系呈现出一种“天然”的顺序，因此不难看出函数 f 实际上是一个有序偶对的集合，第一项来自定义域，第二项来自值域。以下是两种常见的函数表示法：

$$f: A \rightarrow B$$

$$f \subseteq A \times B$$

在这个函数定义中没有对集合 A 和 B 做出任何约束，所以可以有 $A = B$，A 或 B 也可以是其他集合的笛卡儿积。

3.2.2 函数类型

函数还可以进一步通过具体映射来加以描述。在下面的定义中，从函数 f：$A \rightarrow B$ 出发，定义集合为：

$$f(A) = \{ b_i \in B：\text{对于 } a_i \in A，\text{则有 } b_i = f(a_i) \}$$

这个集合有时也称为集合 A 在函数 f 下的像。

定义

- 当且仅当 $f(A) = B$，f 是 A 到 B 上的函数。

- 当且仅当 $f(A) \subset B$（注意：这里是 B 的真子集），f 是 A 到 B 中的函数。
- 当且仅当对于任何 a_i，$a_j \in A$ 且 $a_i \neq a_j \Rightarrow f(a_i) \neq f(a_j)$，$f$ 是 A 到 B 的一对一映射函数。
- 当且仅当存在 a_i，$a_j \in A$ 且 $a_i \neq a_j$ 可以使得 $f(a_i)=f(a_j)$，f 是 A 到 B 的多对一映射函数。

用自然语言来解释上面的定义就是：如果 f 是 A 到 B 上的函数，则可以断定 B 的每个元素都会与 A 的某个元素相对应。如果 f 是 A 到 B 中的函数，则 B 中至少有一个元素不与 A 的元素对应。一对一映射函数保证了函数对应的唯一性：定义域中的不同元素不能对应于值域中的同一元素。（可以看出，这是前面提及的函数“表现良好”的反情况。）如果函数不是一对一映射的，就是多对一映射的，即多个定义域元素可以映射到相同的值域元素上。从这些定义上看，所谓要求函数“表现良好”就是要防止出现一对多映射的情况。了解关系型数据库的测试人员一定会看出：所有的可能性对关系来说都是适用的（一对一、一对多、多对一和多对多）。

再回头研究这些测试实例，假设取 A、B 和 C 为 3 个 NextDate 程序的日期集合，有

$$A = \{\text{日期：1812 年 1 月 1 日} \leqslant \text{日期} \leqslant \text{2012 年 12 月 31 日}\}$$
$$B = \{\text{日期：1812 年 1 月 2 日} \leqslant \text{日期} \leqslant \text{2013 年 1 月 1 日}\}$$
$$C = A \cup B$$

那么，函数 NextDate：$A \rightarrow B$ 就是 A 到 B 上的一对一映射，而函数 NextDate：$A \rightarrow C$ 则是 A 到 C 中的一对一映射。

对 NextDate 函数来说，多对一映射是没有意义的；但对三角形问题来说，很容易发现它是多对一的。如果函数是一对一映射的上函数，比如前面讨论的 NextDate：$A \rightarrow B$，那么定义域中的每个元素都恰好与值域中的某一个元素相对应；反之，值域中的每个元素也恰好与定义域中的一个元素相对应。在这种情况下，就可以找到函数的逆函数，把值域一对一地映射回定义域（参见第 2 章习题中的 YesterDate 函数）。

这些性质对测试来说每个都很重要。中函数与上函数之间的差异，意味着在功能测试中基于定义域的测试和基于值域的测试的不同；对一对一映射函数来说，会需要比多对一映射函数更多的测试工作。

3.2.3　函数复合

假设有若干个集合和函数，其中一个函数的值域恰好是另一个函数的定义域。

$$f: A \rightarrow B$$
$$g: B \rightarrow C$$
$$h: C \rightarrow D$$

在这种情况下，就可以进行函数复合。为此，把各个定义域和值域中的元素记为 $a \in A$，$b \in B$，$c \in C$，$d \in D$，并设 $f(a) = b$，$g(b) = c$ 和 $h(c) = d$。这样函数 h、g 和 f 的复合即为：

$$\begin{aligned} h \circ g \circ f(a) &= h(g(f(a))) \\ &= h(g(b)) \\ &= h(c) \\ &= d \end{aligned}$$

在软件开发中，函数复合是十分常见，很自然地存在于过程和子过程的定义中。佣金问题中就有这样一个例子：

$$f_1(\text{枪机，枪托，枪管}) = \text{销售额}$$

$$f_2(\text{销售额}) = \text{佣金}$$

于是有：

$$f_2(f_1(\text{枪机，枪托，枪管})) = \text{佣金}$$

对测试人员来说，多次复合的一系列函数可能会产生很多问题，特别是在函数复合过程中前一个函数的值域恰好是下一个函数定义域的真子集时。有一种特殊的函数复合情况，可以在某些方面帮助测试人员。回想曾经讨论过的，一对一映射的上函数必定会存在逆函数。逆函数肯定存在，而且是唯一的（数学家可以严格证明这个结论）。如果 f 是从 A 到 B 上的一对一映射函数，其唯一的逆函数记作 f^{-1}。结果对于 a ∈ A，b ∈ B，总会有 $f^{-1} \circ f(a) = a$ 和 $f \circ f^{-1}(b) = b$。NextDate 函数和 YesterDate 函数就是这样的逆函数。函数可逆性给测试人员提供的帮助是，利用给定函数的逆函数可以实现“交叉验证”，从而提高构造功能测试用例的速度。

3.3 关系

函数是关系的特例。函数和关系都是笛卡儿积的子集，但对于函数来说，有“表现良好”的约束，以保证一个定义域元素不会对应于多个值域元素。在通常使用时，人们认同这种约束：在说到某个事物是其他事物的“函数”时，实际上是说有某种确定的关系存在。并不是所有的关系都严格地成为函数。比如，病人集合和医生集合之间的对应关系，这就不是函数，一个病人可能接受多个医生的治疗，一个医生又会诊治多个病人，显然这是一种多对多的映射。

3.3.1 集合之间的关系

定义

给定集合 A 和 B，其关系 R 是笛卡儿积 $A \times B$ 的一个子集。

有两种常见的关系表达方法。如果想在整体上表述关系，通常仅记为 $R \subseteq A \times B$；而对于具体元素 $a_i \in A$，$b_i \in B$，它们之间的关系记为 $a_i\ R\ b_i$。很多数学著作中都忽略了对关系的详细论述，但关系对我们来说更重要，因为它是数据建模和面向对象分析的基础。

接下来要介绍一个广泛使用的概念——势。回想前面对集合势的定义，是指集合中元素的个数。因为关系也是集合，因此你可能会认为关系的势是指集合 $R \subseteq A \times B$ 中有序偶对的个数。但实际上关系的势并不是这样定义的。

定义

给定集合 A、B 和关系 $R \subseteq A \times B$，关系 R 的势为：

- 当且仅当 R 是 A 到 B 的一对一映射函数，关系具有一对一势；
- 当且仅当 R 是 A 到 B 的多对一映射函数，关系具有多对一势；
- 当且仅当至少存在一个元素 $a \in A$ 同时出现在 R 中的两个有序偶对中，即有 $<a, b_i> \in R$ 和 $<a, b_j> \in R$，关系具有一对多势；
- 当且仅当至少存在一个元素 $a \in A$ 同时出现在 R 中的两个有序偶对中，即有 $<a, b_i> \in R$ 和 $<a, b_j> \in R$，而且至少存在一个元素 $b \in B$ 同时出现在 R 中的两个有序偶

对中，即有 $<a_i, b> \in R$ 和 $<a_j, b> \in R$，关系具有多对多势。

类似于映射到值域的上函数和中函数之间的差别，对关系也存在同样的参与的概念。

定义

给定集合 A、B 和关系 $R \subseteq A \times B$，关系 R 的参与定义为：

- 当且仅当 A 的每个元素都出现在 R 的某个有序偶对中，关系是全参与关系；
- 当且仅当 A 中存在元素不出现在 R 的任何有序偶对中，关系是部分参与关系；
- 当且仅当 B 的每个元素都出现在 R 的某个有序偶对中，关系是上参与关系；
- 当且仅当 B 中存在元素不出现在 R 的任何有序偶对中，关系是中参与关系。

用自然语言来叙述就是：如果关系适用于 A 的每个元素，则为全参与；如果关系不能适用于 A 的所有元素，则为部分参与。全参与和部分参与之间差别的另一种表述方式是强制参与和可选参与。类似地，如果关系适用于 B 的每个元素，则为上参与；如果关系不能适用于 B 的所有元素，则为中参与。值得特别讨论的是，上参与和中参与之间的差别同全参与和部分参与之间出奇地相似。从关系型数据库理论的角度来看，是没有理由呈现出这种平行性的；实际上更有理由应该刻意避免出现这种情况。数据建模在本质上是陈述性的，而过程建模在本质上却是强制性的。术语上的平行性暗示着关系必须具有方向性，但实际上并不需要这种方向性。之所以给人造成这种方向性的感觉，部分原因可能是源于笛卡儿积事实上是有序偶对构成的，自然排列出了第一元素和第二元素。

至此仅讨论了两个集合之间的关系。将关系推广到 3 个或更多的集合上，要比推广笛卡儿积复杂得多。比如，假设有 3 个集合 A、B、C 和一个关系 $R \subseteq A \times B \times C$，那么应该把关系严格地定义在三个元素上，还是定义在一个元素和一个有序偶对上呢（这会有 3 种可能性）？这个思路还需要适用于势和参与的定义。对参与来说情况比较简单，但势是具有二值性的。（比如，设想一下 A 到 B 是一对一映射关系，而 A 到 C 是多对一映射关系的情况。）在第 1 章中研究规定行为、实现行为和已测试行为时，曾经讨论过这种三向关系。我们希望在测试用例和规格说明 – 程序实现偶对之间能够存在某种形式的全参与关系，在后面研究功能测试和结构测试时还要继续讨论这个问题。

测试人员需要留心关系的定义，因为关系定义同被测软件的性质直接相关。例如，上参与和中参与之间的差别就直接源自所谓的基于输出的功能测试；而强制参与和可选参与之间的差别则是异常处理的基础，对测试人员也非常有用。

3.3.2　单个集合上的关系

排序关系和等价关系这两种重要的数学关系非常有用，它们都定义在单个集合上，分别侧重关系的某些具体属性。

给定集合 A，$R \subseteq A \times A$ 是定义在 A 上的关系，有 $<a, a>$、$<a, b>$、$<b, a>$、$<b, c>$、$<a, c> \in R$。关系 R 具有 4 种特殊性质。

定义

关系 $R \subseteq A \times A$ 是：

- 当且仅当对所有 $a \in A$，均有 $<a, a> \in R$，关系 R 是自反关系。
- 当且仅当 $<a, b> \in R \Rightarrow <b, a> \in R$，关系 R 是对称关系。
- 当且仅当 $<a, b>$、$<b, a> \in R \Rightarrow a = b$，关系 R 是反对称关系。
- 当且仅当 $<a, b>$、$<b, c> \in R \Rightarrow <a, c> \in R$，关系 R 是传递关系。

家庭关系可以很好地说明这些性质。思考以下这些关系，确定哪些性质适用于各个关系：兄弟关系、同胞关系、祖先关系。下面来定义排序关系和等价关系这两个重要关系。

定义

如果关系 $R \subseteq A \times A$ 是自反、反对称和传递的，则 R 是排序关系。

排序关系具有一定的方向性，一些常见的排序关系有：年长于、≥、⇒ 和祖先。（自反性常常会产生一些不合常理的话，比如说“不比……年轻”和“不是……的后代”。）排序关系在软件中是很常见的：在数据访问技术、散列码、树型结构和数组中都广泛使用了排序关系。

集会的幂集是这个集合所有子集的集合。集合 A 的幂集记作 $P(A)$。子集关系 $\subseteq$ 是 $P(A)$ 上排序关系，因为它具有自反性（任何集合都是它本身的子集），反对称性（见集合相等的定义）和传递性。

定义

如果关系 $R \subseteq A \times A$ 是自反、对称和可传递的，则 R 是等价关系。

在数学中存在着大量的等价关系，相等和同余就是两个例子。等价关系和集合划分密切相关。比如，给定集合 B 的某个划分 B_1，B_2，…，B_n，如果两个元素 b_1 和 b_2 出现在同一划分中，则称 b_1 和 b_2 是相关的（即 $b_1 R\ b_2$）。这个关系具有自反性（任何元素都会出现在自己的划分中）、对称性（如果 b_1 和 b_2 出现在某个划分中，那么 b_2 和 b_1 也在这个划分中）和传递性（若 b_1 和 b_2 在同一个划分中，而 b_2 和 b_3 也在同一个划分中，则 b_1 和 b_3 在同一个划分中）。通过划分来定义的关系被称为由划分所导出的等价关系。其逆过程也同样成立：从集合上定义的等价关系出发，可以根据彼此相关的元素定义子集。这就形成了一个划分，称为由等价关系导出的划分。这个划分中的集合称为等价类。综上得出的结论是：划分和等价关系可以互换的。这个概念对测试人员来说非常重要。如前所述，划分的两个重要属性是完备性和无冗余性。应用到软件测试领域中，测试人员可以利用这两个性质明确地定义对待测试软件的测试程度。除此之外，假设等价类的各个元素都具有相似的行为，就可以只测试每个等价类中的一个元素，从而大大提高测试效率。

3.4 命题逻辑

我们已经在不知不觉中使用命题逻辑了。如果你还没弄明白前面所使用的命题逻辑定义，这也没有什么大不了的。集合论和命题逻辑之间的关系就类似于鸡和蛋的关系——很难说清楚应该先讨论哪一个。就像前面把集合作为基本术语不进行严格定义就直接使用一样，这里把命题也作为基本术语。命题是或者为真或者为假的一种陈述。真或假称为命题的真值。此外，命题还应该是无歧义的：对于给定的命题，应该总能判断出为真还是为假。“数学很难”这个陈述就不能作为命题，因为其中有模糊性。命题还有时间性和空间性。比如，“下雨了”这句话有些时间为真，有些时候为假，而且对同一时间处于不同地点的两个人来说，还可能对一个人为真，对另一个为假。

通常用小写字母来表示命题，如 p、q 和 r。命题逻辑具有与集合论非常类似的运算、表达式和恒等属性（命题逻辑和集合论实际上是同构的）。

3.4.1 逻辑运算符

逻辑运算符（又叫逻辑连接符或逻辑操作符）是通过它们对命题的真值所起的作用来定

义的。这很简单，因为命题只有两个取值：T（真）和 F（假）。也可以用同样的方式来定义算术运算符 (事实上这是孩子们思考算术的方法)，但这样产生的真值表（见表 3-2）会特别庞大。3 个基本逻辑运算符为与（∧）、或（∨）和非（~），有时也称为合取运算、析取运算和非运算。非运算是唯一的一元逻辑运算符（只有一个操作数），其他都是二元运算符。这些，以及其他的一些逻辑运算符都被真值表定义。

合取运算和析取运算在日常生活中很常见：只有所有参与运算的对象都为真时，合取结果才为真；而至少有一个对象为真，析取结果就为真。非运算的结果则一目了然。另外两个常用运算符是异或运算（⊕）和条件运算（→），其定义如表 3-3 所示。

表 3-2　逻辑运算符的真值表

p	q	$p \wedge q$	$p \vee q$	$\neg p$
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

表 3-3　异或运算和条件运算

p	q	$p \oplus q$	$p \rightarrow q$
T	T	F	T
T	F	T	F
F	T	T	T
F	F	F	T

对异或运算来说，仅当其中一个命题为真时，其结果为真；而对析取运算（也就是或）来说，当两个命题都为真时也为真。条件连接符通常来讲要更困难一些。简单情况下可以只把它当作一种定义，但是由于其他连接符都能很好地用自然语言来叙述，自然期望对条件也能如此。简单地说，条件同演绎过程是密切相关的：在有效的三段论推理中，可以说“如果前提条件成立，则结论成立”，这样条件语句将是重言式。

3.4.2　逻辑表达式

用逻辑运算符来构建逻辑表达式，同使用算术运算符来构建代数表达式是一模一样的。可以按照传统的括号内优先的原则来定义运算顺序，也可以直接定义运算符的优先顺序（如非运算优先级最高，合取运算次之，析取运算最低）。给定逻辑表达式，根据括号所确定的顺序总能一步步“构建”出真值表来。例如，对表达式 $\sim((p \rightarrow q) \wedge (q \rightarrow p))$ 就可以构造出表 3-4 所示的真值表。

表 3-4　$\sim((p \rightarrow q) \wedge (q \rightarrow p))$ 的真值表

p	q	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$	$\sim(p \rightarrow q) \wedge (q \rightarrow p)$
T	T	T	T	T	F
T	F	F	T	F	T
F	T	T	F	F	T
F	F	T	T	T	F

3.4.3　逻辑等价

算术相等和集合恒等的概念在命题逻辑中也有类似的情况。可以看出，表达式 $\sim((p \rightarrow q) \wedge (q \rightarrow p))$ 的真值表和表达式 $p \oplus q$ 是一样的，这就意味着不管基本命题 p 和 q 被赋予什么样的真值，这两个表达式的真值永远都相同。逻辑等价的定义有很多形式，这里采用一种最简单的定义。

定义

对两个命题 p 和 q，当且仅当其真值表相同时，p 和 q 是逻辑等价的（记为 $p \Leftrightarrow q$）。

顺便解释一下，这里的“当且仅当”条件有时也称为双态条件。因此，当且仅当 q 为真时，p 为真，实际上是指 $(p \rightarrow q) \wedge (q \rightarrow p)$，记作 $p \Leftrightarrow q$。

定义

永远为真的命题是重言式，永远为假的命题是矛盾式。

命题要成为重言式或矛盾式，必须包含至少一个连接符以及两个或多个基本命题。有时把重言式记作 T 命题，把矛盾式记作 F 命题。表 3-5 中给出一些可以直接同集合论类比的命题定律。

表 3-5 命题定律

定律名称	表达式
同一律	$p \wedge T \Leftrightarrow p$ $p \vee F \Leftrightarrow p$
支配律	$p \vee T \Leftrightarrow T$ $p \wedge F \Leftrightarrow F$
幂等律	$p \wedge p \Leftrightarrow p$ $p \vee p \Leftrightarrow p$
自反律	$\sim(\sim p) \Leftrightarrow p$
交换律	$p \wedge q \Leftrightarrow q \wedge p$ $p \vee q \Leftrightarrow q \vee p$
结合律	$p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$ $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
分配律	$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$ $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
德摩根定律	$\sim(p \wedge q) \Leftrightarrow \sim p \vee \sim q$ $\sim(p \vee q) \Leftrightarrow \sim p \wedge \sim q$

3.5 概率论

在研究软件测试的过程中有两个地方会用到概率论：一个是在研究语句的某个具体执行路径的可能性时，二是在将其推广为业界流行的性能分析的概念时（见第 14 章）。由于使用不多，这里仅介绍一些基础知识。

与讨论集合论和命题逻辑一样，这里也从基本概念（事件的概率）入手。下面是经典教科书所给出的概率定义（Rosen，1991）：

事件 E 是有限样本空间 S 的子集，S 由发生的可能性相同的结果构成，则事件 E 的概率为 $p(\mathrm{E}) = |\mathrm{E}| / |\mathrm{S}|$。

这个概率定义基于这样的思想：每次实验只产生一个结果，样本空间是所有可能结果的集合，事件是结果的集合。这是个循环的定义：什么叫可能性相等的结果？这里假设所有结果具有相等的概率，那么概率这个概念就由它自身来定义了。

两个世纪之前，法国数学家拉普拉斯曾给出一个合理而实用的概率定义。值得强调的是，事件发生的概率是事件期望的发生方式数量除以全部可能的发生方式数量（期望和不期望发生方式的总和）。拉普拉斯的概率定义在研究从袋子中取各色彩球的概率时非常有效（概率论学者通常研究彩球问题，可能其中有些特殊的教益），但却很难推广到无法枚举所有可能性的情况。

这里我们试图运用在集合论和命题逻辑中“练就的本领”来给出一个更好的概率定义。作为测试人员，关心的是要发生的事情。把这些事情称为事件，把所有事件的集合作为论域。然后给出事件的命题，使命题指向论域中的元素。给定论域 U 和某个 U 中元素的命题 p，给出如下定义。

定义

给定命题 p，p 的真集 T 是论域 U 中所有使 p 为真的元素的集合，记为 $T(p)$。

命题或者为真或者为假，所以 p 将论域划分为两个集合，即 $T(p)$ 和 $(T(p))'$，有 $T(p) \cup (T(p))' = U$。注意：$(T(p))'$ 和 $T(\sim p)$ 是相同的。利用真集的概念，可以很容易地在集合论、命题逻辑和概率论之间建立起明确的对应关系。

定义

命题 p 为真的概率记作 $Pr(p)$，有：$Pr(p) = |T(p)| / |U|$。

有了这个定义，拉普拉斯概率定义中“期望的发生方式数量”就是真集 $T(p)$ 的势，发生方式的总量就是论域的势。这样会产生另一个推论：鉴于重言式的真集是论域，矛盾式的真集是空集，因此 $\varnothing$ 和 U 的概率就分别为 0 和 1。

在 NextDate 问题中可以找到许多很好的例子。比如，考察月份变量 m 和命题 $p(m)$：

$$p(m)\text{:}m \text{ 是一个有 30 天的月份}$$

论域是集合 U = { 1 月，2 月，…，12 月 }，则 $p(m)$ 的真集是集合

$$T(p(m)) = \{ 4 \text{ 月}，6 \text{ 月}，9 \text{ 月}，11 \text{ 月} \}$$

那么给定月份 m 有 30 天的概率就是

$$Pr(p(m)) = |T(p(m))| / |U| = 4/12$$

论域的作用相当微妙，这也是在软件测试中使用概率论的一个技巧——要选择正确的论域。假设要问月份是 2 月的概率。答案很简单是 1/12。但是，如果要知道恰好有 29 天的月份的概率，那就不那么容易了。首先需要定义一个论域，包括闰年和平年。利用同余算术，建立一个连续 4 年中所有月份的论域，比如取 1991 年、1992 年、1993 年和 1994 年。在这个论域中共有 48 个月份，所以 29 天月份的概率是 1/48。另一种可能的计算方法是使用 NextDate 函数全部取值范围的两个世纪作为论域，其中 1900 年不是闰年。这样就会得到一个稍微小一点的概率。可见，选择正确的论域非常重要。结论是：避免“转变论域”，这甚至比选择论域还要重要。

下面给出一些概率的性质，这里不加证明就直接拿来使用了。对于给定的论域，命题 p 和 q，以及真集 $T(p)$ 和 $T(q)$，有：

$$Pr(\sim p) = 1 - Pr(p)$$

$$Pr(p \wedge q) = Pr(p) \times Pr(q)$$

$$Pr(p \vee q) = Pr(p) + Pr(q) - Pr(p \wedge q)$$

上述性质同集合恒等和命题逻辑等价的各个公式结合在一起，提供了概率表达式的强大计算能力。

3.6　习题

表 3-6　集合运算符和逻辑运算符的关系

运算操作	在命题逻辑中	在集合论中
析取	或	并
合取	与	交
非	非	补
蕴涵	条件	子集
	异或	对称差

1. 集合运算符和命题逻辑中的逻辑运算符是紧密相连的（同构的），如表 3-6 所示。
   a. 用语言来表述 $A \oplus B$。
   b. 用语言来表述 $(A \cup B) - (A \cap B)$。
   c. 试证明 A ⊕ B 和 $(A \cup B) - (A \cap B)$ 是相等的。
   d. $A \oplus B = (A - B) \cup (A - B)$ 成立吗？
   e. 在上面的表格中，应该给空格中的运算操作起什么名字？
2. 在美国很多地方，房产税的征收是要根据不同的对象来区别对待的，例如学校地区、防火地区、城镇等。试讨论对某个州来说，这些征税对象能否构成一个划分？美国的 50 个州是否构成了一个划分？（对华盛顿哥伦比亚特区应该如何处理？）
3. 在所有人集合上，兄弟关系是等价关系吗？同胞关系呢？

3.7　参考文献

Rosen, K.H., *Discrete Mathematics and Its Applications*, McGraw-Hill, New York, 1991.