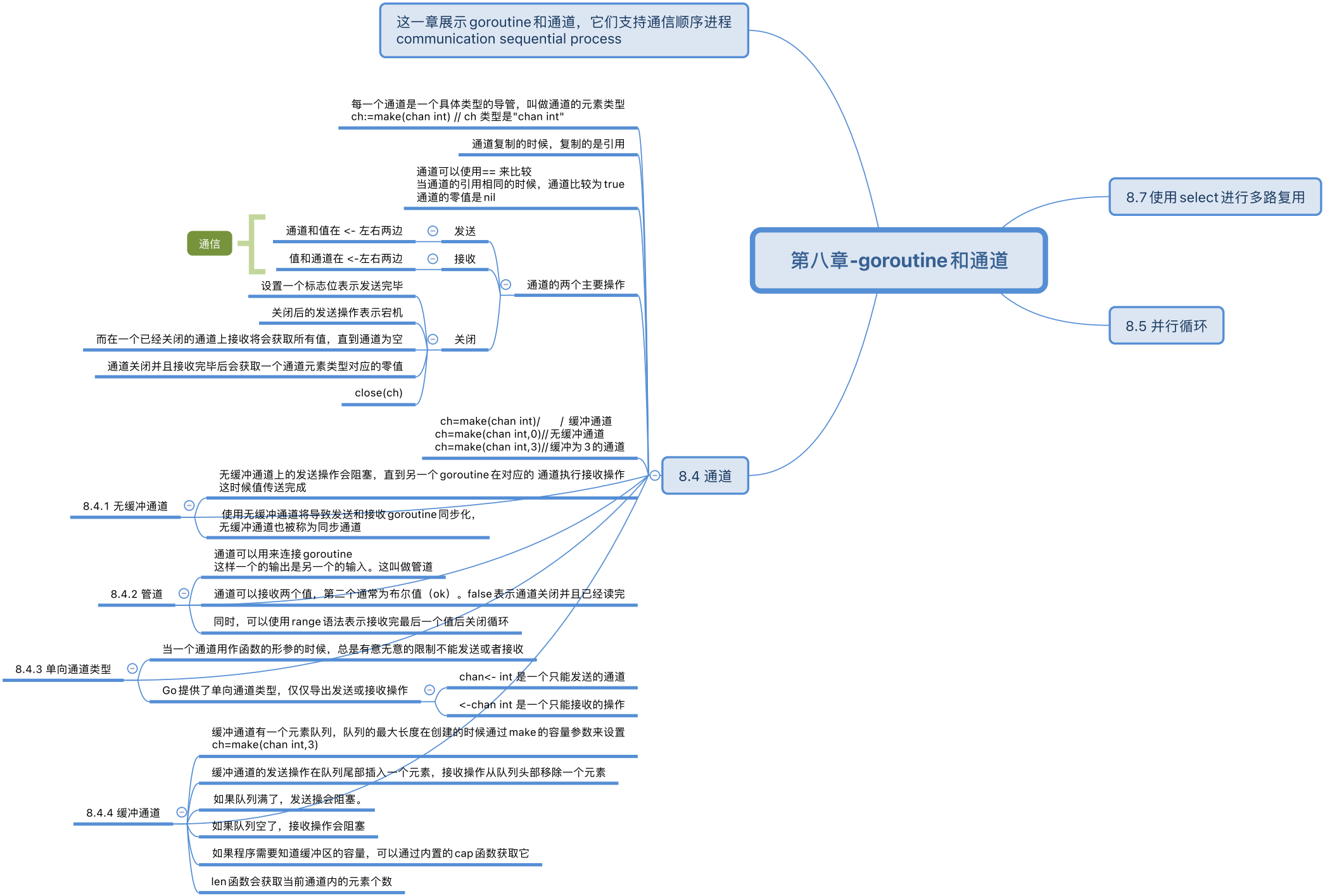
这一章展示 goroutine 和通道，它们支持通信顺序进程 communication sequential process
第八章-goroutine 和通道
8.7 使用 select 进行多路复用
8.5 并行循环
8.4 通道
每一个通道是一个具体类型的导管，叫做通道的元素类型
ch:=make(chan int) // ch 类型是"chan int"
通道复制的时候，复制的是引用
通道可以使用== 来比较
当通道的引用相同的时候，通道比较为 true
通道的零值是 nil
通道的两个主要操作
通信
通道和值在 <- 左右两边
发送
值和通道在 <-左右两边
接收
关闭
设置一个标志位表示发送完毕
关闭后的发送操作表示宕机
而在一个已经关闭的通道上接收将会获取所有值，直到通道为空
通道关闭并且接收完毕后会获取一个通道元素类型对应的零值
close(ch)
ch=make(chan int)/ / 缓冲通道
ch=make(chan int,0)//无缓冲通道
ch=make(chan int,3)//缓冲为 3 的通道
8.4.1 无缓冲通道
无缓冲通道上的发送操作会阻塞，直到另一个 goroutine 在对应的 通道执行接收操作
这时候值传送完成
使用无缓冲通道将导致发送和接收 goroutine 同步化，
无缓冲通道也被称为同步通道
8.4.2 管道
通道可以用来连接 goroutine
这样一个的输出是另一个的输入。这叫做管道
通道可以接收两个值，第二个通常为布尔值（ok）。false 表示通道关闭并且已经读完
同时，可以使用 range 语法表示接收完最后一个值后关闭循环
8.4.3 单向通道类型
当一个通道用作函数的形参的时候，总是有意无意的限制不能发送或者接收
Go 提供了单向通道类型，仅仅导出发送或接收操作
chan<- int 是一个只能发送的通道
<-chan int 是一个只能接收的操作
8.4.4 缓冲通道
缓冲通道有一个元素队列，队列的最大长度在创建的时候通过 make 的容量参数来设置
ch=make(chan int,3)
缓冲通道的发送操作在队列尾部插入一个元素，接收操作从队列头部移除一个元素
如果队列满了，发送操会阻塞。
如果队列空了，接收操作会阻塞
如果程序需要知道缓冲区的容量，可以通过内置的 cap 函数获取它
len 函数会获取当前通道内的元素个数