RubySonar：一个 Ruby 静态分析器

在过去一个多月时间里，我大部分时间都在做一个 Ruby 的静态分析叫做 RubySonar。它使用与 PySonar2 类似的技术，不过针对 Ruby 的语义进行了很多调整。现在这个分析器已经能够支持 Sourcegraph 的 Ruby 代码搜索和浏览。这比起之前的效果是一个很大的进步。

在 RubySonar 的帮助下，对于很多 repo，Sourcegraph 可以搜索到比以前多几十倍甚至上百倍的符号，当然代码的使用范例也随之增加了。代码定位的准确性有很大提高，基本不会出现错位的情况了，另外还支持了局部变量的加亮，所以看起来有点像个“静态 IDE”的味道。

由于 RubySonar 比起 Sourcegraph 之前用的基于 YARD 的分析在速度上有上百倍的提高，我们现在可以处理整个 Ruby 标准库（而不只是以前的一小部分）。Ruby on Rails 的结果也有比较大的改善。另外，以前不支持的像 Homebrew 之类的独立应用，现在也可以分析了。

RubySonar 的静态分析使用跟 PySonar2 相同的跨过程，数据流+控制流分析，而且采用同样的类型推导系统，所以分析的精度是很高的。我还没有跟 Ruby 的 IDE 比较过，不过因为构架的先进性，它应该已经能处理一些现在最好的 Ruby IDE 也搞不定的事情，当然由于时间短，在细节上比起它们肯定也有不足之处。

虽然 Ruby 和 Python 看起来是差不多的语言，为了把 PySonar2 改到 Ruby 上，还是做了不少的工作的。最开头我试图让它们“重用”大部分代码，只是在不一样的地方做一些条件分支进行特殊处理。可是后来发现这样越来越复杂，越来越危险。为了照顾一个语言的特性，很容易破坏掉为另一个语言已经调试好的代码。结果最后决定把它们完全分开，其中共享的代码通过手工拷贝修改。事实证明这个决定是正确的，否则到现在我可能还在为一些莫名其妙的错误伤脑筋。这个经验告诉我，所谓的 DRY（Don’t Repeat Yourself）原则其实有它的局限性。有时候真的是宁愿拷贝粘贴代码也不要共享。

目前 RubySonar 还缺少对 native 库代码的支持，但是由于代码始终保持了简单的原则（RubySonar 只有 7000 多行代码），那些东西会比较容易加进去。感兴趣的 Ruby 用户可以看看自己的 repo 是否已经得到处理，如果没有的话可以来信告诉我，也欢迎给我指出其中存在的问题。