山东大学

毕 业 论 文(设 计)

论文（设计）题目: 程序合成基准测试数据集的建立

姓　　名 娄云飞

学　　号 201822130235

学　　院 计算机科学与技术学院

专　　业 计算机科学与技术

年　　级 2018 级

指导教师 余仲星

2020 年　5　月　17　日

目　录

目录

程序合成基准测试数据集的建立

摘　要

程序合成（Program Synthesis）是计算机研究的一个重要课题。近几年随着算法的突破以及计算机运算力的提高，该领域逐渐取得了重大突破。

但同样的由于该领域的发展历史并不算长，目前用于评价程序合成技术使用的测试集通常规模较小并且不够统一，这就造成评价体系不够完善准确，影响了不同程序合成技术的有效性的客观对比。

本课题致力于搜集并且整理程序合成研究大量文献中使用的测试集，建立一个大型的、全面的、类别清晰的程序合成算法基准测试数据集，并且提供相应的数据处理脚本程序。便于其他研究人员使用并且评判程序合成方法的有效性，以促进对程序合成技术有效性的客观评价。通过大量阅读与整理近几年该领域主要文献，整理搜集过去研究中主要使用的基准测试集以及基准数据集的优劣；并且从相关论坛爬取相关的数据进行构造更为全面的基准测试集，并提供相关的脚本程序便于使用者使用。

关键词：程序合成，基准测试集，大数据集；

ABSTRACT

Program synthesis (Program Synthesis) is an important subject in computer research. In recent years, with the breakthrough of algorithms and the improvement of computer computing power, major breakthroughs have been made in this field.

However, because the development history of this field is not long, the test machines currently used to evaluate program synthesis technology are usually small in scale and not unified enough, which results in an incomplete and accurate evaluation system, which affects the effectiveness of different program synthesis technologies. An objective comparison of sex.

This project is devoted to collecting and sorting out the test sets used in a large number of literatures on program synthesis research, establishing a large-scale, comprehensive and clear-category benchmark data set of program synthesis algorithms, and providing corresponding data processing scripts. It is convenient for other researchers to use and judge the effectiveness of program synthesis methods to facilitate objective evaluation of the effectiveness of program synthesis techniques. Through a lot of reading and sorting out the main literature in this field in recent years, sorting out and collecting the benchmark test sets mainly used in past research and the pros and cons of benchmark data sets; and crawling relevant data from relevant forums to construct a more comprehensive benchmark test set , and provide related script programs for the convenience of users.

Keywords: Program synthesis, Benchmarks, A large dataset.

第1章 绪　论

1.1 选题背景和研究意义

程序合成（Program synthesis）旨在用底层的编程语言自动搜索生成一个程序去解决我们希望解决的问题。长期以来，程序合成一直被认为是计算机科学领域内一个神圣的问题。并且在编程语言，机器学习，软件工程多个方向的研究者都已经取得了一定的进展。

程序合成涉及的任务多种多样。我们希望算法或者模型可以通过一些自然语言描述、必要断言、或者一部分的输入输出示例完成程序的搜索与自动合成。当下领域内一大问题是由于该领域的发展历史并不算长且任务众多，目前用于评价程序合成技术使用的测试集通常规模较小并且不够统一。尤其是部分单个任务相关数据比较难以搜集，研究者常常使用几十条测试数据去验证模型，对模型的准确性难以进行一个特别可靠的评估。

例如程序合成领域比赛 SyGuS-2017[1],关于字符串样例编程的测试用例只有一百余条，并不具有测试的普适性。而有关字符串样例编程的数据获取主要是依靠 excel 社区用户关于闪光填充（flashfill）功能的反馈获取，数据爬取也并不方便，直接搜集会导致获取很多的脏数据。

该课题就是要从领域内大量的文献入手，整理不同的程序合成领域目前大部分研究者正在努力攻克的任务。收集各任务所需要的训练与测试数据。建立一个尽量完整，包括大部分任务在内的基准测试集。

1.2 研究现状

长期以来，程序合成问题就被认为是计算机科学领域一个神圣的问题。在像编程语言、机器学习、人工智能等许多不同的研究方向上都已经取得了不错的进展。在建构数学的早期工作中就已经有提出通过组合较小子问题的解去构造具有证明的可解释解（或算法）的想法[2]。

之后，随着第一个自动化定理程序的开发，有相当多的基于综合性质的演绎法的开创性工作也被应用到程序合成领域[3]。这些突破性工作背后的原理主要是使用定理证明器去首先证明构造用户提供的规范，再用证明去提取相对应的逻辑程序。

另外基于通过不断转化进而生成的方式也是一个热门的进展方向。在这一方面，我们反复的转化高级别的完整规范，直到可以实现所需要的低级别的程序。

综合性的演绎方法希望构造出来一种合适的规范，可以合理的映射出目标程序意图，但在相当多的情况下，这种构造甚至不比编写代码程序简单。于是一个新的方法：基于归纳规范的新归纳合成方法得到实现，比如通过输入输出示例和演示。也有相当多的成果是基于一些输入输出示例生成学习受限的框架，比如 Shaw 等人的成果：从单个输入输出示例学习的 lisp 程序框架[4]。

Pygmalion 也第一个实现了依据一些演示系统实现目标程序生成；该系统从程序的一组具体执行中推断出递归程序。在基于遗传算法等启发式算法引导方向也出现了很多开创性工作。这些方法受到达尔文进化论的启发，通过一代一代的迭代随机的程序段，筛选合适的程序，进行最终程序的生成。

但是程序合成依然是一个具有挑战性的难题。难点主要有两点：一是程序空间太过于复杂很难去解决，二是用户的意图种类太多而且形式不统一。

一般来说（对于图灵完备的语言和任意的约束），程序合成几乎是不能被判定的，所以几乎全部的成功的程序合成二点方法都是在程序空间上面执行一些搜索。这种搜索本身就是一个十分困难的组合问题。任何一种编程语言中的程序数量都会随着程序的规模大小（简单说就是行数多少）成指数级别的快速增长。这就导致稍微复杂一点的程序，我们都要面临特别大的空间去进行搜索，几乎是不能完成的。所以早期程序合成的方式都是侧重于演绎和转化方法。这些方法可以维护一个确保正确的代码重写规则指数增长的定理树。这两种方法都可以满足生成的程序保证提供的约束。

在过去的一段时间里，程序合成的再一次得到注意也带来许多技术和算法上

的更新。首先由于摩尔定律的进展，研究者被允许在更快的运算速度和更大的内存空间内部进行搜索。基于约束求解的方法的不断更新也使得许多旧的思路得到实现与应用。其次，基于程序空间枚举的许多新方法，比如随机搜索和自适应的自顶而底的搜索等等方法，也让程序合成应用可以在难以通过定理和重写规则形式化方式表达的难题在新的领域内部得到实现。

然而即使现代合成技术已经可以成功合成相当大的实际代码片段，但这些成果还是很难应用到实际的工程里面去。例如，Phothilimthana 的文章中所使用的方法[5]，已经是最先进的超优化技术(给定函数的较短实现的合成器)，能够实现搜索大小为 1079 的程序空间。但相比之下，发现 MD5 哈希函数的专业实现搜索大小为 105943 大小的程序空间。可以看出仍然有很大的差距。新的算法的进步，去促进大型程序的空间探索。这仍然是程序合成方向的一个正在努力的方向。

1.3 论文组织结构

论文的正文部分一共有四章。

其中第二章是论述收集并整理数据、构建基准测试集的指导思路。从数据来源到数据整理进行的必要操作进行介绍。

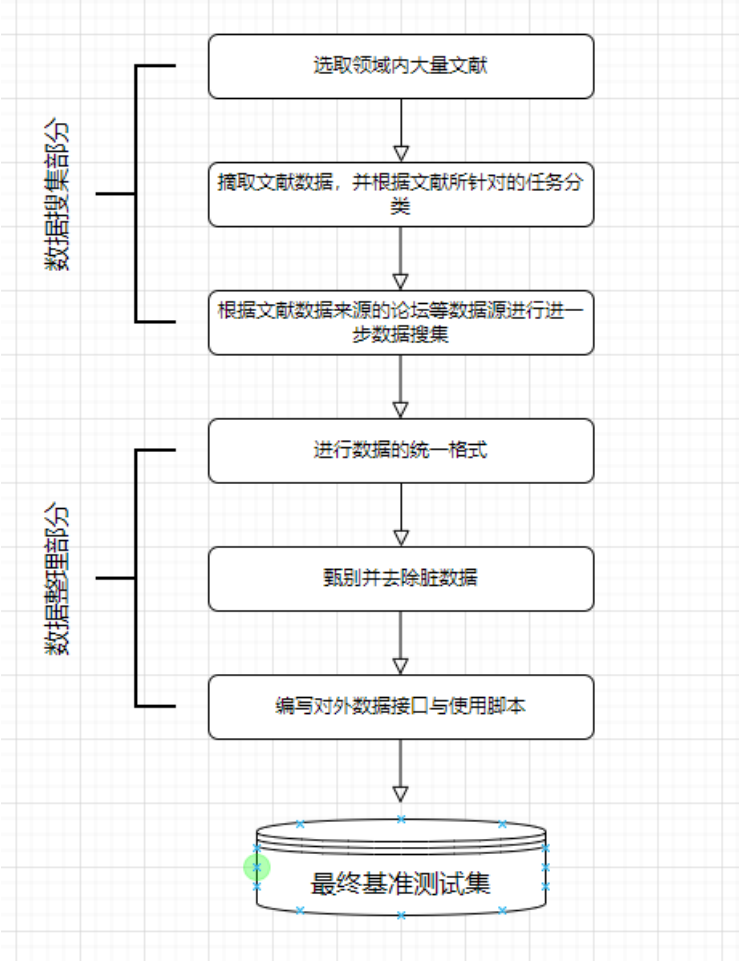
第三章主要介绍了基准测试集本身的内容：包括基准测试集涵盖哪些方向的哪些具体任务、各任务的目标与示例输入输出、以及各任务数据的组织与存储形式、各任务的基准测试集的特点与优点、各任务的建议评价方式等问题。最后总结了一下整个基准测试集的结构与整体内容统计。

第四章主要比较了本课题中构造的基准测试集与其他基准测试集的区别与优劣，阐述了本课题主要的应用领域与应用价值。

第五章总结了本课题的全部工作，并且展望了下一步还能继续开展的延申工作。

第2章 数据集构建

为了搜集并整理程序合成领域大部分的任务类型以及目前科研工作者正在使用的基准测试集。我们首先搜集程序合成方向近几年主要的有影响力的学术论文与期刊文章并进行阅读与整理。



图表 6-1 数据集整理流程

2.1 文献的搜集与整理

针对于程序合成这一领域。我们选取了在该领域内发表成果较多的会议与期刊进行筛选文献。由于该课题最近成果比较多，为了选取比较有代表性的。我们只针对比较有影响力的会议论文进行搜集。具体会议如下表：

表格 1-1 选取论文涉及范围

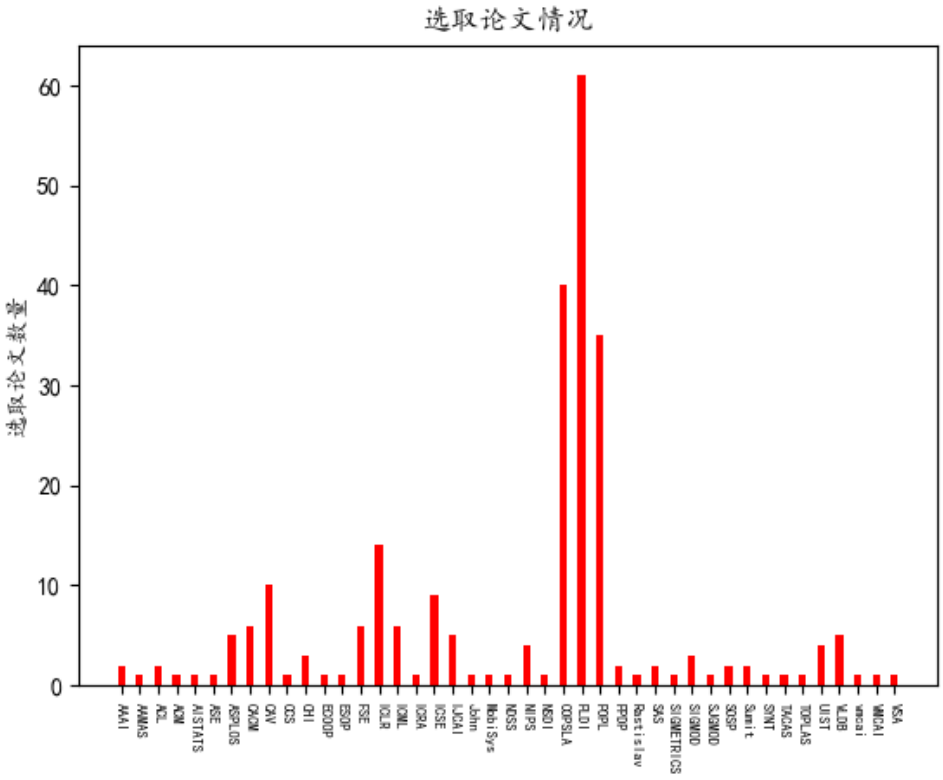
编程语言领域会议	PLDI：Programming Language Design and Implementation
	POPL： Principles of Programming Languages
	OOPLSA ： Object-Oriented Programming, Systems, Languages & Applications
	SAS: International Static Analysis Symposium
	ASPLOS：International Conference on Architectural Support for Programming Languages and Operating Systems
	ECOOP: European Conference on Object-Oriented Programming
	ESOP: European Symposium on Programming
	PPDP： Principles and Practice of Declarative Programming
计算机体系结构领域会议	
人工智能领域会议	AAAI：Association for the Advancement of Artificial Intelligence
	AAAMAS：International Conference on Autonomous Agents and Multiagent Systems
	AISTATS：Artificial Intelligence and Statistics
	IJCAI：International Joint Conference on Artificial Intelligence
自然语言处理会议	ACL：Association for Computational Linguistics
人机交互期刊与会议	ACM Transactions on Computer-Human Interaction
	CHI：Conference on Human Factors in Computing Systems
软件工程领域会议	ASE：International Conference on Automated Software Engineering
	FSE：Foundations of Software Engineering
	CAV ： International Conference on Computer-Aided Verification
	ICSE：International Conference on Software Engineering
ACM 其他期刊	CACM：Communications of the ACM
机器学习方向	LCLR ： International Conference on Learning Representations
	NIPS：Conference and Workshop on Neural Information Processing Systems
	ICML：International Conference on Machine Learning
机器人领域会议	ICRA：IEEE International Conference on Robotics and Automation
	USIT: The ACM Symposium on User Interface Software and Technology
数据库领域会议	VLDB：International Conference on Very Large Data Bases
	SIGMOD: the Association for Computing Machinery's

	Special Interest Group on Management of Data

其中选取会议与期刊的原则就是，选取在领域内比较有影响力、学术声誉优秀的文献，来确保我们搜集数据的质量。

针对表格 1-1 中的期刊与会议。我们选取其中在 2005-2020 年期间发表并属于程序合成方向的文章，作为我们研究的主体。过程中我们总计选取了 253 篇论文，具体的论文目录可以参见附录。

至此，我们可以认为我们基本获取了在程序合成领域近几年大部分科研工作者的研究方向。并开始根据研究者对数据的要求以及已有的基准测试集的数据与与数据来源构建一个更为完整的基准测试集。



图表 1-2 各期刊与会议选取论文情况

2.2 文献阅读

该部分是对我们在文献的搜集与整理中搜集的文献进行阅读与原始数据获取。在取得了大量的学术文献之后。我们有理由认为我们可以从中获得近几年研

究者所着重聚焦的学术热点。并根据学术热点进行两项工作（1）对当前该领域任务进行整理与分类（2）对每一个主要的任务着手建立可靠的基准测试集。

所以针对文献的阅读方式，我们主要需要阅读文章的引论（introduce）部分、基准测试集（benchmark）部分、实验（experiment）部分、以及结论（conclusion）部分。总结作者进行的基本任务，并根据作者所使用的基准测试集以及作者使用的测试方式。总结该任务所需的数据，归纳到我们本地的数据当中。

经过大量文献的阅读。将程序合成领域内的问题总结为主要的四大部分进行讨论：

（1） 依据示例进行编程。主要是依靠一些输入与输出示例自行在领域特定语言（domain specific language 以下简称为 DSL）中进行检索生成对应的底层语言进行问题的自行建模。其中最主要也是研究最多的问题就是类似于 excel 社区闪光填充（flashfill）问题，依据部分数据整理的样例来实现大量数据的数据整理。另外还有同时依据一些输入输出为示例以及自然语言描述进行数学问题程序建构的问题[6]。

（2） 依据代码的部分信息或者原始代码进行代码的优化。代码的自动优化一直也是编译问题以及编程语言问题领域内一个很大的课题。代码编译后的自动优化依靠传统的去常量等优化措施在当前的编译器下已经有了不错的成果，虽然尽管依然存在一定的缺陷：比如编译优化都是要依据上下文，对于比较庞大工程的代码优化会有困难。程序合成领域内的编译优化依然是借用一些理论方式完成多个维度的代码优化工作。这里经过搜集的主要有如下的课题：a. C++的编译优化 b. 代码的安全性优化（依据部分代码构建更为安全的代码块）c. ISA 指令集代码优化 d. GPU 编译器的优化。其中数据搜集主要针对前两个课题。

（3） 依据自然语言进行编程。主要是依靠一些包含一定语义的自然语言生成自然语言语义中想要生成的代码工作。这一部分工作既属于自然语言处理领域也属于程序合成领域。这一部分的任务有两个特点： i. 由于自然语言语义比较复杂，目前大部分的自然语言处理工作都是使用机器学习和深度学习相关的方法进行实现。所以这一部分对数据需求量也比较大，需要大数量级的数据才能进行比较好的训练与验证工作；ii. 由于程序语言只依

靠相关语义并不容易实现程序自动的合成，目前大部分的工作还是一些简单语言的生成工作。该部分经过搜集有如下的课题：a. 依靠自然语言生成数据库查询语言 sql；b. 依靠自然语言生成 python 语言 django 框架内数据库操作程序；c. 依靠自然语言生成 geoqueries 语言[7]。d. 依靠自然语言生成 atis 语言[8]；e.依靠自然语言生成正则表达式；

（4） 其他类型的合成任务。目前收集到两个任务：将图片格式的图表转化为 latex 格式的内容[9]与依靠故事板信息进行程序生成。

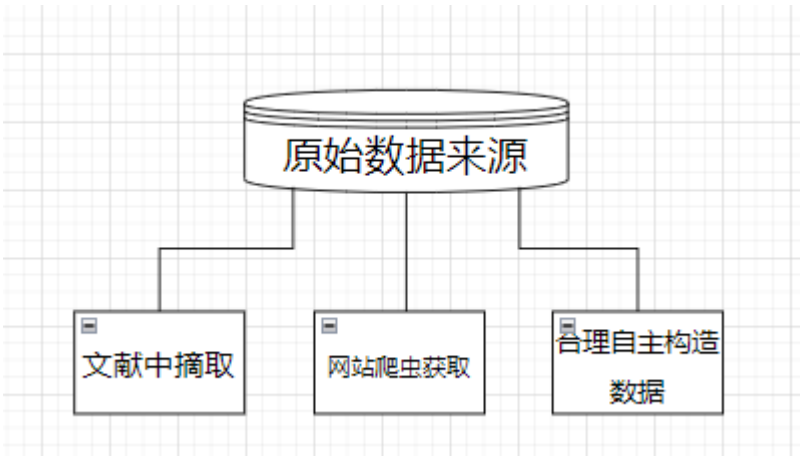
2.3 原始数据获取

在获取到文献的主要任务之后，下一步的工作就是获取原始的数据。

一方面是文献中所使用的实验数据，这一部分数据是科研者对其模型进行基准测试而已经经过处理与验证后的数据，可以认为是比较可靠的数据，我们会收录到我们的数据集中。同时做这一部分时要注意，在归纳数据的同时要了解数据的特点与格式，并且记录下来作为下一步是否使用和是否需要进行处理的参考。

另一方面有的文献中并没有提供数据分享的文章可以通过作者提供的数据源去进行数据的下载与整理。

第三部分是收纳一些文献中并没有使用但是可以通过一些方式获取的合理的基准测试集。其中一部分是通过阅读文章的任务之后通过检索 github 代码仓库进行选取，比如一些代码的优化工作。我们摘取一定量的 GitHub 项目中代码，按照星数量进行排序，选取排位靠前的一部分代码进行摘取，作为样本作为原始的数据后期等待处理；另一种方式是选取一些合理的基准测试方式对任务进行测试。比如合理的随机生成测试数据进行测试。



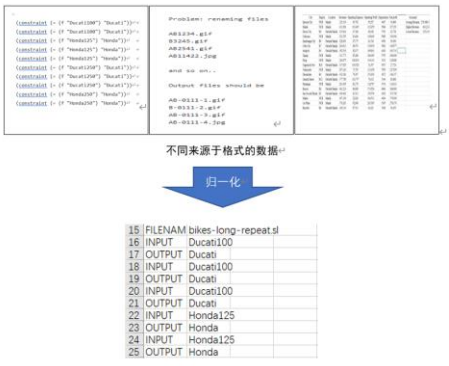
图表 1-3

通过以上的三种方式针对每一项对应的任务都尽可能多的搜集足够的原始数据，先进行初步的去重（因为有可能多个文献用同一个来源的数据，也有可能某些文献的数据和我们在网上爬取的数据重复），去除脏数据的操作。再进行下一步的数据整理，基准测试集的构建。

2.4 数据整理与处理

针对不同来源的数据，主要的处理手法操作有：

（1） 统一格式，因为不同数据源的数据会有不同的数据格式与数据存储形式，第一步就是将全部的数据格式进行统一。主要工作包括：解析文件格式并提取数据，统一数据结构，用统一格式进行转存。



图表 1-4 统一数据结构示例

（2） 编写数据说明样式，方便使用者能清楚的了解数据的含义、格式等信息，方便使用者使用。针对每一项任务，我们对数据整理之后都编写完整的数据格式说明，在第 3 章中对应的每一部分都可以看到这些清

晰的数据说明，在数据仓库中的每一个对应任务数据的目录下也可以查看到具体的 readme 文件。

（3） 编写相应的脚本文件，方便使用者直接进行一些数据的预处理、读入与统计等工作。例如针对编译优化问题，我们提供了相对应的 shell 脚本文件辅助检测编译后程序的运行占用内存与时间。用于直接检测编译优化模型的有效性。

（4） 进行更为具体的数据分类工作。针对一个特定任务的数据，我们会评估现有的数据集和可以爬取的数据部分的数据特点。如测试样例的规模，多样性等等方面。进行更精细化的分类，并用标签进行标注。如用文件名去标注测试用例的属性，如长用例还是短用例，或者在数据中加一个标签标注数据的输入输出数据类型等。并在说明文件中说明这些问题，辅助使用者在使用的同时也能根据数据的属性进一步评估自己的模型或者算法。在第 3 章中也能具体从各个例子看到我们这一部分工作的内容。

第3章 数据表示与数据特点

3.1 示例编程（program by example）部分

3.1.1 文本数据整理

文本数据处理任务，主要对应于 excel 的闪光填充（flashfill）工作。依据一部分的示例信息来完成剩余信息的处理，用于一些数据整理的工作。

表格 1-2 任务示例

John Doyle	John D.
Matt Walters	Matt W.
Jody Foster	Jody F.
Angela Lindsay	
Maria Schulte	

如表格 1-2 任务示例中一样，依靠前三行的数据示例进行代码的自动合成，在领域特殊语言中搜索可以解决问题的题解，从而进行任务的完成工作。

该部分原始数据主要来源于：

（1） 相关论坛的数据爬取。但由于多数文献从 excel 论坛爬取数据，因为 excel 论坛分享的相关案例都是用户在实际操作中的诉求，相对来说只是提供一个思路，有合理数据的条目相对来说不多。需要人工的筛选，在这里由于时间与精力的限制获取的数据不多。

（2） 程序合成领域内比赛 SyGus[错误!未定义书签。]是该领域内比较权威的比赛。在 17-19 年之间都收录了部分数据整理相关的数据。经过去除重复和整理格式（为了实现与其他来源数据格式的统一）。获取了一定部分的数据。

（3） 部分文献中所给了一些案例示例，以及一些文献中使用并且分享出来的数据。经过去重（因为一些文献都引用某篇文献中数据可能会导致重复）与格式整理。获得了一部分数据。

综上我，我们一共获取了 318 条测试用例。数据格式采用 csv 存储。存储每一列形式如表格 1-3 所示。

表格 1-3 数据格式

文件名	Input 列	Output 列	划分列
一组数据，表示该文件数据类型。	多组输入	多组期待输出	一组数据，表示训练与测试的划分比例，可以通过定义修改

该数据集的特点有：

（1） 数据量较大部分当前文献中采用的基准测试集更大，作为基准测试集来说可信度更高。

（2） 数据的类型参考比赛**错误!未定义书签。**的样式用文件名标注单条数据特点，并且搜集了多组文献中特别提及的某一些模型暂时无法解决的样例，作为特殊数据提高对模型合理的考验性。作者可以依靠文件名来判断模型的瓶颈所在，并提出下一步合理的解决方案。

（3） 提供数据标记脚本文件，方便研究者自己通过自己定义的方式进行划分训练集和验证集，方便用户使用。

3.1.2 同时依靠示例输入输出与自然语言生成目标程序

同时依靠少量的输入输出与自然语言描述来生成示例程序的任务。这一部分任务由于一般任务的复杂程度比文本数据整理部分的更加复杂，所能依据的示例也比较少，同时解决问题的方式也不同。所以方法上一般是结合深度学习和程序综合领域的方法，设计丰富的领域特定语言（DSL），进行设计高效的搜索算法。所以对数据量的要求比较高。

本部分的数据来源主要来自于搜集已有数据集 AlgoLisp[10]和 NAPS[11]两个数据集。进行统一格式；两个数据集都是专注于解决该任务的比较优秀的数据集。

里面涵盖了多种类型输入和输出的数据。

总计一共约十六万条数据，足够满足该任务的训练与验证工作。

表格 1-4 任务示例

文本描述	"given", "an", "integer", "array", "var0", ",", "let", "var1", "be", "the", "size", "of", "var0", ".", "create", "an", "array", "var2", "identical", "to", "var0", ".", "sort", "var2", ".", "initialize", "var4", "to", "0", ".", "set", "var5", "to", "1", ",", "so", "long", "as", "var5", "is", "<", "=", "var1", ",", "do", "the", "following", "in", "a", "loop", ":", "set", "var6", "to", "var5", ".", "initialize", "var7", "to", "var1", "-", "1", ",", "do", "the", "following", "in", "a", "loop", "so", "long", "as", "var6", "is", "more", "than", "0", ":", "add", "var2", "[", "var7", "]", "to", "var4", ",", "and", "decrement", "var7", "and", "var6", ".", "(", "end", "of", "inner", "loop", ")", "set", "var5", "to", "var5", "*", "4", ".", "(", "end", "of", "outer", "loop", ")", "return", "var4", "."
输入输出	{"input": [[963662765, 272656295, 383441522, 477665112]], "output": 3061088459}, {"input": [[1, 2, 3, 4]], "output": 14}, {"input": [[7, 9, 6, 9]], "output": 40}}…（等多组输入输出示例）
期待结果	能解决多组输入输出且满足文本描述的程序。

这方面整合多个数据集的难点在于，不同数据集对于领域特定语言（domain specific language）的定义不同，对于语法树的定义也不同。考虑到使用者可能会选择不同的形式，我们的处理办法是依然保持各个数据集之间的独立。然后再组织一个合并的验证集只包含文本描述与输入与输出的数据集用于整体模型的检测。

由于数据量比较大，为了更加易读最终的数据格式采用 json 格式存储。具体数据格式如表格 1-5 数据格式所表示相同。

表格 1-5 数据格式

名称	对应值的含义
return_type	程序返回值的数据类型
text	问题的文本描述，已分词
tests	数组形式，多组输入输出示例

Input_type	输入数据的数据类型

考虑到有些课题要专门对一些特定数据类型的输入和输出的任务进行单独的分类与讨论。例如只讨论浮点数类型输入和输出的问题。我们特意对数据的输入与输出类型进行标记，并且可以通过 json 脚本进行直接进行筛选，便于使用者进行某一类型数据的统计，其中标签中对数据类型分类的命名方式同 sql 语言对数据类型的分类。

另外有一部分模型不需要自然语言描述仅仅通过输入输出的示例进行训练，本数据集的单个输入输出示例足够多，也能满足测试任务。

该数据集的特点有：

（1） 数据量足够大，能满足模型的训练与验证工作。

（2） 结合了主流的一些已经使用的框架，可以参考之前的一些工作并作为 baseline 进行验证自己模型的有效性。

（3） 对数据进行了精细化的分类，可以通过标签中的输入类型与返回类型进行特定输入输出类型（如，浮点数输入与输出计算问题）来提取相应的数据。进行特定问题的分类。

3.2 依据自然语言自动生成代码部分

3.2.1 自然语言转类数据库查询语句

自然语言与类数据库搜索语言之间的转化，主要的处理任务是纯依据一些自然语言的语义去构建相对应的代码。因为解释自然语言语义单纯用搜索的方式难以完成，所以与同时依靠示例输入输出与自然语言一样一般都是通过一些深度学习的方式去进行模型的构建与训练因为解释自然语言语义单纯用搜索的方式难以完成。

首先是类数据库查询语言，我们分为逻辑查询语言和高级查询语言两种。其中逻辑查询语言以[7]和[8]中定义的为例子进行整理数据。高级查询语言则以 wikisql 和 django 为例子进行数据的整理。

表格 1-6 任务示例

语言类型	示例
geoqueries	Input：which state has the most rivers running through it? Output：(argmax $0 (state:t $0) (count $1 (and (river:t $1) (loc:t $1 $0))))
atis	Input：allflights from dallas before 10am Output：(lambda $0 e (and (flight $0) (from $0 dallas:ci) (< (departure time $0) 1000:ti)))
Wikisql	Table schema: ||Pianist||Conductor||Record Company||Year of Recording||Formatk Input：What record company did conductor Mikhail Snitko record for after 1996? Output：SELECT Record Company WHERE (Year of Recording > 1996) AND (Conductor = Mikhail Snitko)
Django	Input：if length of bits is lesser than integer 3 or second element of bits is not equal to string’as’ Output：if len(bits) < 3 or bits[1] !=’as’：

数据源为多篇论文数据的整合，统一格式，统一用 json 格式存储。存储格式如表格 1-7 数据格式表示。其中 wikisql 的数据库表格式如表格 1-8 wikisql 表数据格式所表示的。

表格 1-7 数据格式

名称	对应值的含义
Questions	自然语言问题描述
Answer	不同语言的解答

表格 1-8 wikisql 表数据格式

名称	对应值的含义
Name	表名称
Header	表各元素名
Rows	多组数据，其中一组表示一行数据
Page_title	主键
Type	表各元素类型

其中各组数据规模如下

表格 1-9 数据规模

geoqueries	训练集：600 条 测试集：280 条
atis	训练集：4435 条 测试集：448 条
Wikisql	训练集：56355 条 测试集：15878 条
Django	训练集：15967 条 测试集：1801 条

该数据集的特点是：

（1） 针对同一类型的任务，选取多个类似任务并且构建相似的数据集，便于验证使用者方法的可推广性。同时针对逻辑查询语言与高级查询语言选择其中两个案例进行测试，测试覆盖率高。

（2） 针对比较接近实用的高级查询语言的基准测试集。数据规模足够大，可以划分训练集测试集验证集来进行实验，有一定的实用性与说服力。

（3） 针对 sql 语言的查询，同时给出数据库表格式与初始的存储数据，可以直接通过 sql 直接操作实验证明结论的正确性。

3.2.2 自然语言转正则表达式

从自然语言生成正则表达式任务依靠自然语言的语义作为约束，自动化的生成满足语义任务的代码。一般也是通过深度学习与程序合成相结合的方式去完成任务的求解，同时也有一些传统方式的解决方案。

表格 1-10 任务示例

示例编号	示例内容
1	Input： A list of five strings. A first string consist a single digit. Second string consist 3 'h'. Followed by a 2 lowercase optional. Fourth string consist 3+ capital. Last one is followed by a J Output： concat(<num>,concat(repeat(<h>,3),concat(optional(repeat(<low>,2)),concat(repeatatleast(<cap>,3),<J>)))
2	Input： The pattern starts with a digit followed by 3 h's. After that comes an optional 2 lowercase letters followed by a required 3 plus capital letters. The string ends with a capital J. Output： concat(<num>,concat(repeat(<h>,3),concat(optional(repeat(<low>,2)),concat(repeatatleast(<cap>,3),<J>))))

通过对以往文献的阅读，这是一个相对热点的研究任务，对应的数据集也有很多，如 KB13[12]和 NL-TURK[13]。以及比较新的 STRUCTUREDREGEX[14]。

不过对于不同数据集的观察与了解，其中 KB13 数据集的数据量比较小，一共只有 814 个正则数据对，一是不太适合作为深度学习的训练集，而是仅仅作为基准测试集也缺乏一定的普适性。而 NL-TURK 虽然在数据量上实现了扩大，但是单个测试样例比较简单，通过观察几乎全部的正则表达式都是比较初级的简单的正则表达式。同样存在测试上的不完善性，不够具有说服力。

相比于以上两个数据集，STRUCTUREDREGEX 很好的解决了以上的两个问题。所以数据集的构建原则就是以 STRUCTUREDREGEX 为数据基础，并且结合三个数据的数据（主要将 NL-TURK 和 KB13 作为辅助测试集工具），进行统一格式。

表格 1-11 数据格式

列名	列内容
description	任务的自然语言描述
regex	正则表达式
pos_examples	正向例子
neg_examples	负向例子

该数据集的特点是：

（1） 数据量足够大，且包括三个不同的数据集在内，可以大量的覆盖从简单到

难的多种类型的测试样例。

（2） 包括大量的比较复杂的测试样例。经过观察，同 stackflow 上用户提交的问题比较类似。

（3） 数据通知包括对每一条测试样例二点正向例子和负向例子。有利于用于检测一些伪正则表达式等机器难于发现的错误，如区分大小写等现实问题。为每一条样例都生成六个正向例子和负向例子，这也符合大部分 stackflow 用户提供的样例个数。

3.3 代码优化部分

3.3.1 编译优化

这里主要论述 C++语言编译优化的工作，任务主要是为了使优化后的代码能有更优秀的性能，尽可能的去除无用变量，冗余计算等等无意义的功耗实现编译的进一步优化。

考虑到当前 C++常见的编译器 gcc 已经经过多代的发展已经在传统的编译代码优化上有非常出色的表现。在诸多文献中也常常用 gcc 的编译作为基准测试（baseline）来测定新编译模型的性能指标。所以我们仍然决定以 gcc 的编译效果作为基准测试。

基准测试集主要的测定指标确定为运行内存与运行时间，所以我们为在 Linux 系统上测定指定可执行程序的运行内存与运行时间提供了便于收集和统计的 shell 脚本文件。测试的方式就是分别用新的模型与 gcc 编译器对源代码进行编译以及编译之后的优化，再通过一定的统计指标来判断模型的有效性。

在这里我们提供实时检测项目运行时检测占用内存情况与运行时长的脚本，用来作为指标判断代码优化程度的指标。代码段 1 为示例脚本程序，在基准测试集内部有完整脚本以及使用方式。

代码段 1 检测程序占用内存脚本

```
#!/bin/sh
source /etc/profile

#define variable

psUser=$1
psProcess=$2
pid= `ps -ef | egrep ${psProcess} | egrep ${psUser} |  egrep -v
"grep|vi|tail" | sed -n 1p | awk '{print $2}'`
echo ${pid}
if [ -z ${pid} ];then
    echo "The process does not exist."
    exit 1
fi

MemUsage=`ps -p ${pid} -o vsz |egrep -v VSZ`
 (( ${MemUsage} /= 1000))
echo ${MemUsage}

if [ ${MemUsage} -ge 1600 ];
    then
        echo “The usage of memory is larger than 1.6G”
    else
        echo “The usage of memory is ok”
fi
```

测试的代码选定为 openssl 代码库。也可以支持使用者随意选定任意的大型 C++源代码库。

3.3.2 代码安全性优化

该部分任务主要是为了实现不安全的代码段自动添加必要的保护代码段的工作。

不安全的代码段主要是指由于编程者的疏忽与不注意，导致代码会可能出现数组越界、死循环、试图改变静态数据等等会导致代码运行崩溃的情况[15]。

表格 1-12 任务示例

示例代码	错误处
1.if (optind == argc) 2. ok = nl_file ("-"); 3.else 4. for (optind++) 5. ok &= nl_file (argv[optind]);	第四行的for循环没有循环结束的标识，会导致死循环

我们搜集到了 10 个对应的 cpp 项目，其中缺乏安全性的代码处都使用_SYN 进行标识。并提供一个脚本文件帮助使用者统计经过模型处理过之后的 cpp 项目中有多少处的不安全代码没有经过处理。使用者也可以自行检索哪处的不安全代码没有得到处理。

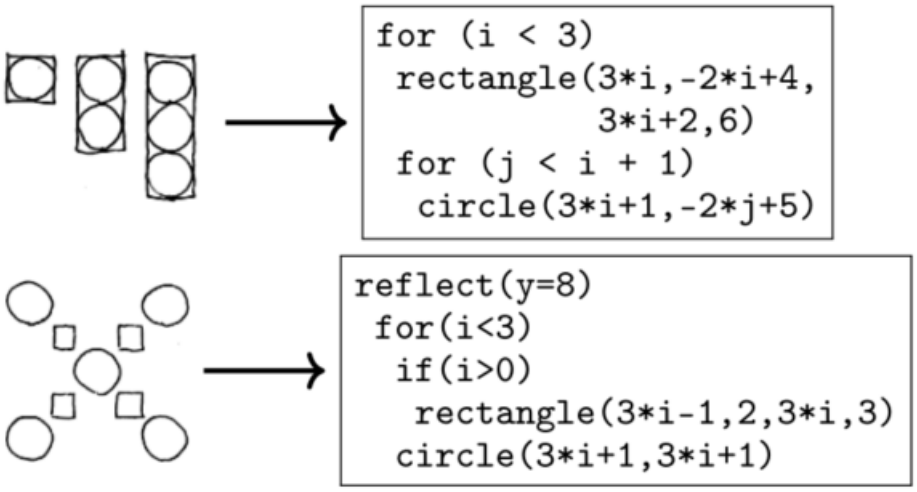
该数据集的特点是：

涉及的不安全漏洞的类型较多，且通过不同的项目来进行集中表现，对于模型的测试比较准确灵活。

3.4 其他合成任务

3.4.1 手绘图形转 latex 代码段

这是一个偏向于图像识别与图像处理和程序合成领域结合的任务，任务描述为将一些手绘的表格转化为 latex 代码段进行表示[16]。任务示例如图表 1-5 任务示例。



图表 1-5 任务示例

该任务的的难点一是在于约束主要靠图片格式，识别起来需要借助 cv 的工具相对来说比较困难，我们在这里推荐文章[10]中的方法与预训练模型基础上进行进一步的研究和参考。

二是在于我们对训练好的模型下一阶段得出的结果很难有一个工具可以进行比对。目前只能人工去进行测试。

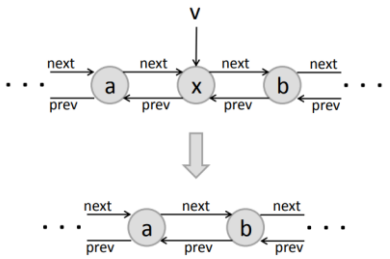
数据我们采取[15]中提供的测试集，该数据集包含一百个手绘的图表。

该数据集的特点为：

使用者要通过模型生成对应的 latex 文件之后，再人工识别得到的结果与原结果是否相符合。尽管这种依靠人眼检测的方式并不算高效，但由于原数据的手绘格式，没有太好的方式。

3.4.2 依靠故事板推理生成目标程序

故事板（storyboard）是一种表达数据结构的形式[17]。可以理解为一种数据结构可视化的形式。该任务就是打算用故事板（storyboard）对数据结构的声明与定义，自动的生成目标程序。




```
void dllRemove(Node v){
    v.n.p = v.p;
    v.p.n = v.n;
}
```

图表 1-6 示例输入与输出（左为输入右为输出）

考虑到图形化的描述并不好用数据表示，我们根据文章[17]中的描述 storyboard 的方式进行整理数据。即用类似 C++指针链接的方式抽象的表达数据结构的存储关系。输入输出是用来描述操作前后的数据结构的前后结构变化，从而作为约束生成目标程序。

数据样例如下：

表格 3-13 样例输入输出

样例一	input father -> floc, z -> zloc, zloc.f -> floc, xn -> xnloc, xnloc.f -> floc, xnloc.m -> anodes:r, xnloc.nm -> anodes:r, floc.f -> floc.f output father -> floc, z -> zloc, xn -> xnloc, xnloc.f -> floc, xnloc.m -> anodes:r, zloc.f -> floc, xnloc.nf -> zloc, xnloc.nm -> anodes:r
样例二	input father -> floc, z -> zloc, xn -> xnloc, zloc.f -> floc, xnloc.f -> anodes:r, xnloc.m -> floc, xnloc.nf -> anodes:r output father -> floc, z -> zloc, xn -> xnloc, xnloc.f -> anodes:r, xnloc.m -> floc, zloc.f -> floc, xnloc.nm -> zloc, xnloc.nf -> anodes:r

一共整理出 37 种表示不同数据结构操作的形式。

该数据集的特点是：

（1） 每组数据都有很好的现实描述。描述具体做了哪些数据结构上的操作，便于研究者观察模型表现不好的原因。

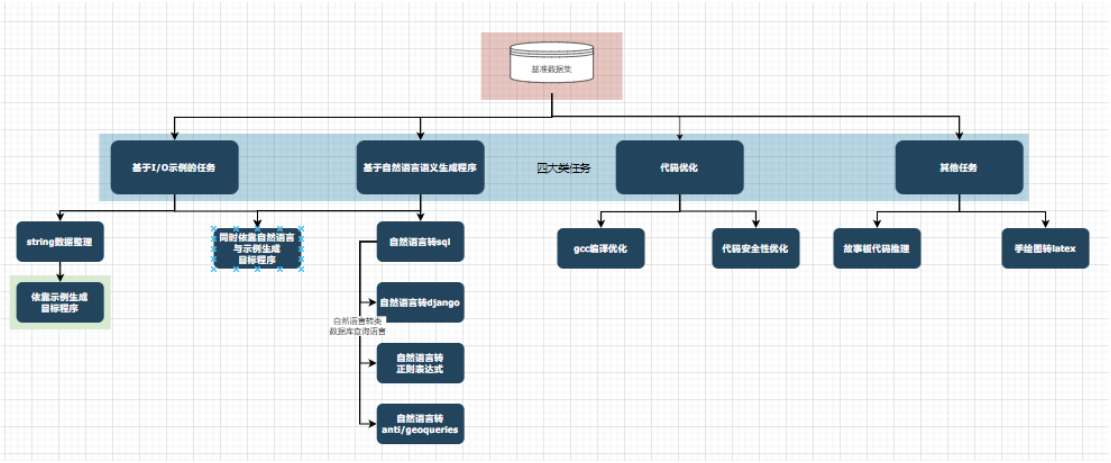
（2） 数据涵盖了基本的数据结构，尤其是链表和树的增删改查功能，对初级的数据结构功能的覆盖性比较强。

（3） 数据的局限性在于还没有复杂数据结构，如红黑树、AVL 树的测试样例，这同样是 storyboard 描述性的瓶颈与相关转化模型的瓶颈。

（4） 检测工具依靠 storyboard 官网的数据对比工具进行检测。

3.5 整体数据集总结

整个基准测试集一共收录了四大类一共十一个具体的任务，其中有些任务可以进行再细化分类，依靠具体的数据标签进行提取。



图表 1-7 收录任务类型统计

具体各任务的数据收录情况可见表格 3-14 具体各任务情况统计。

表格 3-14 具体各任务情况统计

任务描述	任务输入输出描述	数据量
3.1.1 文本数据整理	输入文本示例，输出目标程序与整理之后文本	318 个示例
3.1.2 示例输入输出结合自然语言生成程序	输入自然语言与数据输入，输出目标程序与数据输出	162978 条示例
3.2.1 自然语言转 geoqueries	输入自然语言，输出 geoqueries 程序	880 条示例
3.2.1 自然语言转 atis	输入自然语言，输出 atis 程序	4883 条示例
3.2.1 自然语言转 wikisql	输入自然语言与数据库表，输出 wikisql 程序	72233 条示例
3.2.1 自然语言转 django	输入自然语言，输出 django 程序	17768 条示例
3.2.2 自然语言转正则表达式	输入自然语言，输出 geoqueries 程序	6522 条示例
3.3.1 gcc 编译优	输入 c++项目。输出编译优化后	Openssl 项目；并提供一项工

化	的汇编代码	具，可选取任何 c++工程
3.3.2 代码安全性优化	输入 c++项目。输出优化后的 c++代码	10 个带数据标记的 c++工程
3.4.1 手绘图形转 latex	输入手绘图形，输出 latex 代码	100 个示例
3.4.2 storyboard 转目标程序	输入 storyboard 描述，输出目标程序	37 个示例

对于全部的数据集，在本章各部分都具体的描述了数据的来源、存储格式、提供的使用脚本等等情况。

第4章 与其他程序合成基准测试集的比较

在整体的基准测试集构建基本完毕之后，我们也考察了程序合成领域近几年其他的基准测试集进行比较，来得到我们基准测试集的优越性。在这里着重以本文收集为数据源之一的[12]中 KB13 与[13]中的 NL-TURK，以及近几年 GECCO[18]，和 progRES_CR[19]。

我们从数据集大小，是否涉及多任务，主要涵盖领域，数据种类是否单一，是否提供了评价机制与合理的使用脚本等角度进行比较几个基准测试集的优劣。具体可见表格 1-15 各数据集之间的指标比较。

表格 1-15 各数据集之间的指标比较

	本数据集	KB13	NL-TURK	GECCO	progRES_CR
涉及多领域	√	×	×	×	×
涵盖多任务	√	×	×	√	√
数据集大小	大	小	大	大	大
数据形式	丰富	丰富	单一	丰富	丰富
提供了评价机制与合理的使用脚本	是	否	否	是	否

可见，有一部分经典的基准测试集如 KB13 主要是由于早期提出问题的概念后为了研究问题的评价机制提出的实验性的基准测试集。随着学科发展已经无法提供一些前沿方法的测试工作。还有一些测试集如 NL-TURK、progRES_CR 更多的是提供一种自动数据整理与生成的方式。由于生成算法的约束可能会导致出现 NL-TURK 数据集一样的数据形式过于单一。且只能适应单一任务。其余大部分优秀的基准测试集也大部分都是和 GECCO、progRES_CR 一样主要着眼于一项单一的任务。

所以本课题相较于以上以及其他基准测试集优点主要有：

（1） 涵盖的任务多，分类精细；各个小类之间也通过一些标签标记、文件名标记等方式进行更细化的分类。

（2） 数据量大，且涵盖的数据类型丰富，数据的覆盖性强。

（3） 有明确的数据存储方式说明。便于使用者使用；适当的提供了一些数据处理脚本，适当的提供了一些指标评价的建议。有利于帮助使用者直接上手。

比起有些如 progRES_CR 数据集更多的是表达一种数据构造的思路与基准测试的优先方式。本课题做的更多的是归纳，统计当前已有的成果，并进行整理与分类。所以本课题构造的数据集更多的价值不是创造新的基准测试思路，而是实用性的角度为使用者提供测试的方式方法。

第5章 结论

本文中，我们描述了一个比较完整的、全面的、大型的程序合成领域基准测试集。基准测试集涵盖了四个大的部分 11 个具体的任务类型；

针对每一个具体任务都提出了较为合理且有说服力的基准测试集以及测试方案；提供了基准测试集项目的数据格式说明与使用方式、并且提供了必要的、方便使用者使用的脚本文件。便于使用者在研究过程中直接应用我们的数据进行新模型或者算法的有效性检验。

第6章 结束语

6.1 本文总结

本文主要着手的工作主要在于构建一个比较完整的、全面的、大型的程序合成领域基准测试集。我们从过去几年间程序合成领域主要的学术成果与应用成果收集任务与数据。对任务进行细致的分类。并且收集整理了每个任务所需要的基准测试数据。最终通过数据整理方式整理了一个大型的基准测试数据集，并对我们整理的整个数据集进行细致的描述。

6.2 工作展望

基准测试集的构建是一个需要不断更新且远大的目标；本课题的研究主要聚焦于希望构建一个整个程序合成领域的比较大比较全的基准测试集。其中“大”和“全”都是比较相对的词语；当前的基准测试集涵盖的任务还不算特别全面，尤其是该领域发展迅速，与自然语言处理与计算机视觉等领域也有相联系的课题与成果在不断发展，新的任务也在不断涌出。

另一方面，我们单个任务的基准测试集也还有很大的进步空间。如文本整理任务，目前的数据量还不是特别充足。下一步面对更新颖的模型，我们还需要统计更多的数据对其进行更深入的测试。

下一步的工作中，我们还要继续加大对新的成果的搜集力度，进一步对已有的任务进行进一步的数据搜集，不断的维护整个基准测试数据集。对其不断更新，保证其保持自己的数据意义。

致　谢

近几年来，计算机领域的工程师总会自嘲为码农以去表达对于许多代码与工程工作事实上只是简单重复劳动这一尴尬的事实。从这个角度来讲，program synthesis 的确是一个神圣的、将会极大解放劳动力的工作。在短短接触课题的几个月内也能感受到领域内做出突出贡献的工作者们难得的科学情怀与探索精神。十分感谢领域内的前辈，特别是余仲星老师的指导。在过程中我虽然依然力有不及所做工作还有更大努力空间，但在老师的指导下还是获益匪浅。

另外也十分感谢父母在求学过程中毫无保留的支持，无论是经济还是心理建设上。十分感谢四年来同窗与朋友的陪伴，母校和学院以及各位恩师四年来的培养，以及党和国家对教育事业的支持。建校之初章程中便写道，公家设立学堂是为了“为天下储人才，为国家图富强。”言语终归轻描淡写，希望今后的学习与工作之中可以践行此句，用行动体现自己感恩之情。

参考文献

[1] https://sygus.org

[2] A. N. Kolmogorov. Zur deutung der intuitionistischen logik. Math.Zeitschr, 35:58–365, 1932.

[3] C. Cordell Green. Application of theorem proving to problem solving.In IJCAI, pages 219–240, 1969.

[4] David E. Shaw, William R. Swartout, and C. Cordell Green. Inferring LISP programs from examples. In Proceedings of the 4th International Joint Conference on Artificial Intelligence Volume 1, pages 260–267.Morgan Kaufmann Publishers Inc., 1975.

[5] Phitchaya Mangpo Phothilimthana, Aditya Thakur, Rastislav Bodík, and Dinakar Dhurjati. Scaling up superoptimization. In Proceedings of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 297–310, 2016.

[6] Rishabh Singh, Armando Solar-Lezama, Synthesizing Data-structure Manipulations from Storyboards. In *FSE* 2011.

[7] Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, pages 658–666, Edinburgh, Scotland.

[8] Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1512–1523, Edinburgh, Scotland.

[9] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, Joshua B. Tenenbaum. Learning to Infer Graphics Programs from Hand-Drawn Images. In *ICLR* 2018.

[10] Polosukhin I, Skidanov A. Neural program search: Solving programming tasks from description and examples[J]. arXiv preprint arXiv:1802.04335, 2018.

[11] Zavershynskyi M, Skidanov A, Polosukhin I. Naps: Natural program synthesis dataset[J]. arXiv preprint arXiv:1807.03168, 2018.

[12] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic

Parsing on Freebase from Question-Answer Pairs. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

[13] Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural Generation of Regular Expressions from Natural Language with Minimal Domain Knowledge. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

[14] Ye X, Chen Q, Dillig I, et al. Benchmarking multimodal regex synthesis with complex structures[J]. arXiv preprint arXiv:2005.00663, 2020.

[15]Dillig, Thomas, Isil Dillig, and Swarat Chaudhuri. "Optimal guard synthesis for memory safety." *International Conference on Computer Aided Verification*. Springer, Cham, 2014.

[16] Ellis K, Ritchie D, Solar-Lezama A, et al. Learning to infer graphics programs from hand-drawn images[J]. Advances in neural information processing systems, 2018, 31.

[17] Singh R, Solar-Lezama A. Synthesizing data structure manipulations from storyboards[C]//Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. 2011: 289-299.

[18] Helmuth T, Spector L. General program synthesis benchmark suite[C]//Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. 2015: 1039-1046

[19] Alet F, Lopez-Contreras J, Koppel J, et al. A large-scale benchmark for few-shot program induction and synthesis[C]//International Conference on Machine Learning. PMLR, 2021: 175-186.