上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

本科生毕业设计（论文）中期检查报告

论文题目：基于数据流分析的 R1CS 语言等价性验证及范式生成

学生姓名：施宸昊

学生学号：519021910434

专　　业：软件工程

指导教师：李国强

学院(系)：电子信息与电气工程学院

教务处制表

填表说明

1. 请每位学生根据学校及院（系）检查的要求认真进行自查，及时发现课题研究过程中存在的问题，分析原因，并提出解决思路和措施，明确下一阶段任务。
2. 每位学生应根据项目实施情况认真、实事求是填写。填写字体请用宋体小四号，并用 A4 纸打印，于左侧装订成册。
3. 毕业设计（论文）中期检查报告总字数应满足本院（系）要求。
4. 该表填写完毕后，须请指导教师审核，并签署意见。
5. 《上海交通大学本科生毕业设计（论文）中期检查报告》将作为答辩资格审查的主要材料之一。
6. 本表格不够可自行扩页。

课题进展情况：

零知识证明在现代社会中越来越体现出他的重要性，越来越多的加密社区正在寻求通过零知识证明技术来解决一些区块链的最大难题：隐私安全和可扩展性问题。但是在零知识证明的底层工具链中由 Circom 到 R1CS 约束这一步转换存在着很多局限，首要问题就是 R1CS 的可合并性较差，A 与 B 合并后所生成的 R1CS 与 A 和 B 独立生成的 R1CS 在形式上毫无关联，而这与 R1CS 本身表达能力局限性有关外，根本原因便是程序本身可以生成多个等价的 R1CS 约束，所以我们需要在 R1CS 约束中提出 R1CS 约束的范式，使得对于不同的 R1CS 约束，我们可以较容易地判断其等价性和正确性。这对我们验证程序的等价性以及正确性，包括后续更加深入研究 R1CS 的可合并性方面都将大有裨益。

目前已经通过 Circom 文档以及 github 上的 circom 编译器仓库，熟悉 Circom 语言，以及当前主流 Circom 编译器生成 R1CS 的具体过程。通过构造等价的 Circom 程序，由 npm 中的 snarkjs 框架，生成等价的 R1CS 约束，总结其在形式上不同之处的产生规律。并且设计出了表达数据流图的数据结构，完成了数据流图的生成。还在数据流图的基础上实现了瓦片的选取算法，成功将数据流图划分成多个瓦片。然后再对数据流图进行进一步的抽象，制定在数据流图上生成 R1CS 的规则与过程并生成 R1CS 范式，使 R1CS 约束与具体的程序实现细节解耦，基本完成了 R1CS 的范式生成与代码撰写。

由于国内外相关研究较少，对于实验没有可靠的 benchmark，所以后续还需要整理并拓展编写代码的过程中时使用的测试集，完善论文中的实验集。

课题研究已取得的阶段性成果：

根据 R1CS 约束组的特性设计了数据流图的结构与生成算法，并成功生成了数据流图。将每一个约束先按照 a • b = c 转换为算式，将原 R1CS 中每一个约束，都化成这样的算式，再将其结合在一起，就会得到一个以 DAG 形式存储的含有公共子式的算式树。在这个过程中经历了两次尝试，先是将操作符和变量的 node 类型分开，使用两种类型的 node 进行图的构建。 但是这样就导致当 R1CS 约束组尝试合并两个约束得到等价约束组时，建立的数据流图不同，同时也会导致 DAG 的结构相当复杂，算法设计相对来说也会比较困难。所以最后选择对于操作符和变量使用一种 node 来存储，这样一来每一个操作符本身也是一个中间变量。同时，使用同一种 node 来构造数据流图还有一个好处，那就是减少了 R1CS 约束组中不同约束之间合并或者拆分时，所增加或减少的中间变量对数据流图整体的影响。但此时由等价 R1CS 约束组生成的数据流图仍然存在不同之处那就是在线性的约束中，他们的加法执行顺序不同。这也是因为在数据流图生成时，算法必然需要一个顺序去先把两个变量相加，再将其他的变量相加。在范式生成的开始阶段我们没有足够的信息去判断相加的顺序。不过相对于整个图来说，这只是一些微小的变化， 所以在这一步我尝试了直接使用 pagerank 算法计算图中每个节点的权重。根据相关论文中的说法，pagerank 算法对图中局部性的变化不太敏感的。最终经过统计发现，对于同一个节点，其权重大概会在上下百分之十波动，在大部分情况下，按权重排序的节点序列并不会发生变化。但是一旦节点的权重发生变化，那么范式的生成就将直接宣告失败。因此需要后续对数据流图进行进一步的抽象。

在数据流图的基础上实现了瓦片选取算法。实现了 R1CS 范式中各个约束的选取方式。这里我主要将瓦片分成三个类型：

1. Quadratic：x * y = z 的瓦片，其中 x，y，z 均为变量。
2. MullLinear：根节点由其两个父亲相乘得到的瓦片，两个父亲中至少有一个为常数，比如 (5 * x) * 7 = z。
3. AddLinear：根节点由其两个父亲相加得到的瓦片，比如 $x^3 + 5 + x = z$。

其中 AddLinear 和 MullLinear 瓦片本质上都由线性约束产生，都是 线性瓦片，但是由于在算法处理上逻辑完全不同，所以在此将其分为两个种类讨论。在瓦片选择时，我们将上一个步骤中的数据流图分割成上述三种类型，这样选取有几个考虑方面：

1. 将约束合并步骤暂时搁置，待后续步骤获取树中的更多信息后再进行。
2. 产生未合并的范式后，如有产生合并范式的需求，只需在未合并的范式上应用固定算法即可，相对简单。
3. 瓦片选取的算法实现较为简单。

前面我们提到过，等价的 R1CS 约束组所生成的数据流图其不同之处在于在处理线性瓦片时，节点之间相加的顺序不同，但是在一个线性约束中所相加的节点看成一个集合的话，他们其实是等价的。也就是说，相加顺序的不同，在瓦片选取的过程中仅仅意味着各节点遍历顺序的不同，如果将选取好的线性瓦片视为其相加节点与其系数的乘积的集合，那么等价 R1CS 约束组产生的数据流图中所选出的线性瓦片之间显然是相同的。

在数据流图的基础上，以瓦片的选取为参考，对数据流图进行再一次的抽

象，进一步消除了各个等价 R1CS 约束组在数据流图层面的不同。对线性瓦片进行进一步的抽象，将他在数据流图中用一个抽象出的新节点代替。也就是对自身以外的节点，屏蔽线性瓦片内部相加的顺序，让外部节点到线性瓦片中具体节点的联系，变成到这个节点具体所属的瓦片的联系。而在这一抽象后的数据流图中，边的类型有以下几种：

1. 非线性瓦片抽象节点到非线性瓦片抽象节点：两个顶点在抽象前的数据流图中便已经存在。与抽象前的数据流图保持一致。
2. 非线性瓦片抽象节点到线性瓦片抽象节点：当且仅当抽象节点所代表的线性瓦片中存在非抽象节点时存在。
3. 线性瓦片抽象节点到线性瓦片抽象节点：当且仅当两个抽象节点所代表的线性瓦片存在公有的非抽象节点时存在。

参考相关论文，利用改进后的 Weighted PageRank 算法，计算出所选出的各个瓦片的权重。在对数据流图进行进一步的抽象之后，整个图看起来简单多了，节点和边的数量也变少了很多。在具体的约束组中比较容易出现一些对称的情形，进而会让一些节点在算法中得到相同的权重。让后续的约束的排序变得比较困难。所以参考了一些文献中的做法，使用了 Weighted PageRank 算法。 使用线性瓦片中系数归一化后的方差来作为数据流图中边的权重。

将各个瓦片各自按次序生成 R1CS 的范式。 至此，约束组中约束的划分与约束的排序已经确定，在二次约束中出现过的变量的排序也已经排序完毕。因此在这个步骤中还需要对线性瓦片中新出现的变量的排序进行调整。替每一个在线性瓦片中新加入的变量计算权重，其权重的计算方式为除去本身线性瓦片以外的其他所有线性瓦片中，自己的系数和线性瓦片权重的乘积的绝对值之和。然后便可以对新引入的变量进行排序，首先比较变量的权重，如果一致，再对本身在线性瓦片中的系数进行排序。在线性瓦片中引入的新变量，其在其他 Linear 瓦片中出现的情况某种程度上反映了其在整个约束组中的重要程度。同时如果某些新变量只在本身的约束中出现，他们的权重都将为 0。 并且他们的排序只会对自身的线性瓦片所产生的约束产生影响，并不会改变其他约束的顺序，所以只需将他们按照系数降序排序即可。

存在的问题及解决思路：

1. 缺少数据集

解决思路：按照之前所整理出的等价 R1CS 约束组产生原因，结合编程实现过程中所使用的测试集，自行整理数据集并设计实验进行设计。

下一阶段的工作计划和研究内容：

数据集整理与论文撰写同步进行。

指导教师意见：

完成的非常好，已经开始撰写学术论文，预计四月底投稿。

指导教师签名：

2023 年 4 月 7 日

学院（系）意见：

同意

审 查 结 果： √ 同 意 □ 不 同 意

学院（系）负责人签名：

2023 年 4 月 7 日