氏　　　名	Nguyen Trong Bach	
学位(専攻分野)	博士(情報学)	
学 位 記 番 号	総研大甲第 2496 号	
学位授与の日付	2024 年 3 月 22 日	
学位授与の要件	複合科学研究科　情報学専攻 学位規則第6条第1項該当	
学 位 論 文 題 目	Synthesizing Bidirectional Programs on Relations	
論 文 審 査 委 員	主　査	蓮尾　一郎 　情報学コース　教授
		関山　太朗 　情報学コース　准教授
		加藤　弘之 　情報学コース　助教
		胡　振江 　北京大学　計算機学院　教授
		日高　宗一郎 　法政大学　情報科学部　教授

Summary of Doctoral Thesis

Name in full: Nguyen Trong Bach

Title: Synthesizing Bidirectional Programs on Relations

Bidirectional transformations, originated from the view update problem in relational databases, are an important mechanism for synchronizing or maintaining consistency between two related information where one is specified as the source, and the other as the view. If either the source or the view is changed, the other will be changed as well to restore consistency. People apply bidirectional transformations to solve a diversity of synchronization problems such as relational view update, model-driven software development, data exchange, and serializer/deserializer.

A program that expresses a bidirectional transformation is called a bidirectional program. Essentially, a bidirectional program comprises a pair of programs – a forward program get that defines a view over a source and a backward program put that translates view updates to source updates – and provides strong guarantees about the well-behavedness of the get and the put. It is known to be challenging to develop bidirectional programs that are both well-behaved and practically useful.

Automatic synthesis of bidirectional programs is practically important to solve the synchronization problems, since manually writing such programs is non-trivial and error-prone. We focus more on solving the view update problem in relational database applications by studying the synthesis of well-behaved bidirectional programs (get, put) on relations (tables of relational database systems) from user-given examples over specified data schemas.

We surveyed existing example-based synthesis methods and found that these methods cannot address both the well-behavedness of bidirectional programs and the complexity of relations. There are three main methods of program synthesis that are most related. First, several synthesis algorithms have been developed to obtain programs written in bidirectional languages. These obtained programs are guaranteed to be well-behaved based on the well-behavedness of the underlying bidirectional languages. However, these algorithms are restricted to domains of either regular expressions or trees, and do not operate on relations. Second, relational program synthesis allows us to simultaneously generate multiple programs that satisfy a given relational specification (e.g., the well-behavedness of a bidirectional program) through example-based dual synthesis. It uses hierarchical tree automata to represent a relational version space that encodes all tuples of satisfying programs. As the space of the automata grows dramatically when dealing with complex underlying languages, it becomes challenging to handle practical relational query languages. Third, many other

general example-guided synthesis methods (e.g., PROSYNTH) possibly take examples on relations as input and independently synthesize a get and a put, but they cannot guarantee the well-behavedness of the synthesized pair. PROSYNTH is able to use a tabular input-output example to synthesize a program written in Datalog. Since a Datalog program is largely a set of rules, PROSYNTH reduces the synthesis problem to a rule-selection problem by (1) requiring the preparation of a fixed finite set of candidate rules by using *templates* and enumerations, and (2) selecting a subset of the prepared set that satisfies the given example. The reason that keeps PROSYNTH from guaranteeing the well-behavedness between two separately synthesized programs get and put is that there is no relationship between two sets of templates or candidate rules of the two programs. Naively enumerating all gets and puts rarely produces a well-behaved program.

Although many advanced synthesizers can generate unidirectional programs on relations from user-provided examples, the difficulty of synthesizing bidirectional programs from examples, especially the ones involving tables that have internal functional dependencies, still remains an unsolved issue.

In this thesis, we propose an approach to synthesizing well-behaved and practical bidirectional programs on relations from user-provided examples and data schemas with functional dependencies.

We start by synthesizing a get and decomposing it into a set of simple and atomic get_a whose corresponding put_a exists. With user-given functional dependencies and the set of atomic get_a, we forward-propagate functional dependencies from the a source through intermediate relations to the view, so that the functional dependencies imposed on all relations can be clearly determined. We also compute atomic examples corresponding to (get_a, put_a). Then, the synthesis of (get, put) could be reduced into sub-synthesis of (get_a, put_a). To solve each sub-synthesis task, we design well-behaved templates for put_a given get_a. These templates encode not only the existing minimal-effect atomic view update strategies but also the extra constraints and effects of functional dependencies, following the knowledge in the database community. With the set of well-designed templates, we adopt the modern example-and-template-based synthesizer PROSYNTH to find put_a, and then combine all results $(get_a, put_a)s$ to form the final bidirectional program (get, put).

We have implemented our approach in two prototypes, SynthBX and SynthBP, which are developed using two different template designs. On a benchmark suite of 56 tasks from three sources, SynthBX successfully synthesizes well-behaved bidirectional programs for 52 tasks, with an average synthesis time of 19 seconds per task and within 3 seconds each for 37 of them, which shows practical usefulness of our approach. SynthBX is limited in supporting functional dependencies. SynthBP is a more advanced prototype with supporting templates for functional dependencies. It can automatically solve 37 out of 38 practical benchmarks. The overheads for handling functional dependencies are not too significant when the number of functional dependencies is reasonable.

（別紙１ Separate Sheet 1）

Results of the doctoral thesis defense

博士論文審査結果

氏　名 (Name in Full)　Nguyen Trong Bach

論文題目 (Title)　Synthesizing Bidirectional Programs on Relations

本学位論文は、「Synthesizing Bidirectional Programs on Relations （リレーション上の双方向プログラムの合成）」と題し、英語で記述されており、全6章から構成されている。

第１章は序論である。研究の背景、先行研究の課題、研究目的、主要な貢献など、論文全体の構成を述べている。

第２章は基礎知識の紹介である。関係データベースとその問い合わせ言語である Datalog の基礎知識、ビュー更新問題と双方向変換の基礎知識、および事例ベース(example-based)プログラム合成手法に関する研究について議論している。

第３章は提案手法の概要である。双方向プログラムの合成を3つのステップに分ける。まず、与えられた事例から順方向プログラム （クエリ） を（既存手法で）合成する。次に、順方向プログラムをアトミッククエリに分解する。最後に、各アトミッククエリに対して、適切なテンプレートを構築し逆方向プログラムを事例から合成することによって、最終双方向プログラムを構成する。

第４章では、クエリからアトミッククエリに分解するアルゴリズムを提案し、事例と関数従属性の情報をアトミッククエリに伝搬する手法を与えている。

第5章では、アトミッククエリに対して、その逆方向プログラムを事例から合成するためのテンプレートを構築する方法を与えている。得られた双方向プログラムが往復性質(roundtrip property)を満たすことを示し、構成手法の実現と評価を議論している。

第6章では、第5章の手法を関数従属性に対処するために拡張する方法を議論している。

第7章は論文のまとめと今後の課題である。

公開発表会では博士論文の章立てに従って発表が行われ、その後に行われた論文審査会及び口述試験では、審査員からの質疑に対して適切に回答がなされた。 質疑応答後に審査委員会を開催し、審査委員で議論を行った。審査委員会では、出願者の博士研究が新規的であり充分な水準を持っていることが評価された。

以上を要するに本学位論文は、リレーション上で往復性質を満たす双方向プログラムの合成という未解決問題に対して、事例から分割統治法で自動的に合成する方法を示したものであり、理論だけでなく実践的な観点からも本研究の貢献が大きいもので、学術的価値が大きい。また、本学位論文の成果は、学術雑誌論文2件、フルペーパー査読付き国際会議論文2件として発表され、

社会的な評価も得ている。以上の理由により、審査委員会は、本学位論文が学位の授与に値すると判断した。