

國立陽明交通大學
資訊科學與工程研究所
碩士論文

Institute of Computer Science and Engineering
National Yang Ming Chiao Tung University
Master Thesis

可逆抽象機終止的形式化證明

A Formal Termination Proof for Reversible Abstract Machines

研究生：蔡文龍 (Wen-Lung Tsai)
指導教授：陳穎平 (Ying-ping Chen)

中華民國 一一三年五月
May 2024

可逆抽象機終止的形式化證明

A Formal Termination Proof for Reversible Abstract Machines

研 究 生：蔡文龍
指導教授：陳穎平

Student: Wen-Lung Tsai
Advisor: Dr. Ying-ping Chen

國立陽明交通大學
資訊科學與工程研究所
碩士論文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Science
National Yang Ming Chiao Tung University
in partial Fulfilment of the Requirements
for the Degree of
Master
in
Science

May 2024

Taiwan, Republic of China

中華民國 一一三年五月

Acknowledgement

TODO: 致謝辭

可逆抽象機終止的形式化證明

學生：蔡文龍

指導教授：陳穎平 博士

國立陽明交通大學 資訊科學與工程研究所

摘 要

在陳昭宏 (Chao-Hong Chen) 的論文《A Computational Interpretation of Compact Closed Categories: Reversible Programming with Negative and Fractional Types》中，他探討了緊湊封閉類別透過「反轉時間」與「反轉空間」的解釋，並將其形式化應用於可逆 SAT 求解器的開發上。該論文中，對可逆抽象機的終止證明尤為關鍵，涉及兩項重要定理的形式化證明。儘管這些證明顯然正確，在該論文中尚僅提供了部分的形式化證明。

我們的研究深入探討了論文中的這一陳述。首先，明確重申了陳述的具體內容：對於可逆抽象機，給予一初始狀態，並且已知該初始狀態可抵達的狀態是有限的，則該初始狀態經由可逆抽象機的推移，將會在有限的步數內終止。隨後，我們為此定理附加了一項條件並完成了相關的形式化證明，即所有狀態的個數有限的情況下，對該陳述的形式化證明。最後，我們移除了有限狀態的假設，並以類似方法完成了無限狀態下的形式化證明，從而補充並完善了論文中該部分的證明。

關鍵詞: 可逆抽象機、終止證明、Agda

陽明交大
NYCU

A Formal Termination Proof for Reversible Abstract Machines

Student: Wen-Lung Tsai

Advisor: Dr. Ying-ping Chen

Institute of Computer Science and Engineering
National Yang Ming Chiao Tung University

Abstract

In Chao-Hong Chen’s paper, “A Computational Interpretation of Compact Closed Categories: Reversible Programming with Negative and Fractional Types,” he explores an interpretation of compact closed categories through “reversing time and space,” formalizing its application in the development of a reversible SAT solver. The termination proof of a reversible abstract machine, an essential component of the paper, involved the formalization of two crucial theorems. Although these proofs are clearly correct, only partial formal proofs are provided in the paper.

Our research delves deeply into this statement from Chen’s work. Initially, we rearticulate the specifics of the statement: for a reversible abstract machine, given an initial state with a known finite number of reachable states, this initial state will terminate within a finite number of transitions through the reversible abstract machine. Subsequently, we added a condition and completed a formal proof under the assumption that all states are finite. Finally, we removed the assumption of finite states and employed similar methods to complete a formal proof under infinite states, thus complementing and enhancing the proofs in the original paper.

Keyword: Abstract Machines, Termination Proofs, Agda

陽明交大
NYCU

Contents

陽明交大
NYCU

List of Figures

陽明交大
NYCU

List of Tables

陽明交大
NYCU

Chapter 1

Introduction

1.1 Motivation

In CHAO-HONG CHEN’s study [1], three significant contributions are meticulously elaborated. Initially, the research formalizes the concepts of ”reversing time” and ”reversing space,” showcased through the compact closed categories $(N, +, 0)$ and $(N, *, 1)$ respectively; secondly, it further implements a reversible SAT solver, proving the feasibility of high-level control abstractions described in the first contribution within reversible programming languages; the third contribution focuses on proving the termination conditions for a large class of reversible abstract machines.

Specifically, the paper presents two combinatorial proofs in Chapters 5 and 7, regarding the property that a class of reversible abstract machines will inevitably terminate for initial states with a finite number of reachable states. This proof, although intuitively obvious —that is, on a path of unique states, the number of reachable states decreases as the state progresses, until no further states can be reached, or a termination state is achieved. However, the original paper only partially formalized this proof, failing to fully substantiate this concept.

Our contribution starts from constructing a similar class of reversible abstract machines. Begin with a given initial state, demonstrating that when all possible states of a reversible abstract machine are finite, it will ultimately reach a termination state. Subsequently, we modified the constraints to align with the objective of ”finite reachable states” mentioned in the original paper, thereby accomplishing a more generalized proof.

This paper is accompanied by approximately 400 lines of Agda code, serving to complement the proofs of the two theorems mentioned in the original paper as being incomplete.

1.2 Research Objectives

1.3 Road Map

TODO Chapter ?? introduces thesis.cls document class. Chapter ?? introduces section ordering. Chapter ?? and ?? explain how to load citation, figures and tables (not completed).

陽明交大
NYCU

Chapter 2

Background Knowledge

In this chapter, we introduce the target language used in the research: Agda. As a dependently typed functional programming language, we provide a foundational explanation of its usage.

2.1 Agda

Agda is a dependently typed functional programming language. Writing in the Agda language is akin to organizing a mathematical proof. It begins with self-evident datatypes, proposes assumptions to be proven, and provides detailed evidence for their validity thereafter. The following is an example of a datatype concerning natural numbers.

```
data  $\mathbb{N}$  : Set where
  zero  :  $\mathbb{N}$ 
  suc   :  $\mathbb{N} \rightarrow \mathbb{N}$ 
```

This definition captures the first two axioms corresponding to Peano's axioms, and the other axioms within Peano's axioms can also be easily proven. The function `suc` can be considered as a mapping operation. We can describe this operation in natural language as follows: "Given a natural number n , `suc(n)` is also a natural number." Such a function from \mathbb{N} to \mathbb{N} will be utilized in Chapter TODO, where it will constitute a critical part of the proof process.

And the following demonstrates two examples of the proof process:

```
axioms-3 :  $\forall [n] (\text{suc } n \equiv \text{zero})$ 
axioms-3 ()
```

Here, we take the third Peano axiom as an example, which is to prove that there does not exist a natural number such that $\text{suc}(n)$ is zero. In Agda, a proof of non-existence typically begins by assuming the existence of such an entity, and then deriving a contradiction (denoted as \perp). In this case, the proof process is merely an empty parenthesis. When Agda attempts to find n among the two possible kinds of natural numbers (zero and $\text{suc}(n)$), it immediately encounters an obvious contradiction. Therefore, it concludes that no further cases need to be proven.

```
axioms-5 :  $\forall \{f : \mathbb{N} \rightarrow \text{Set}\}$ 
   $\rightarrow f\ 0$ 
   $\rightarrow (\forall n \rightarrow f\ n \rightarrow f\ (\text{suc}\ n))$ 
   $\rightarrow (\forall n \rightarrow f\ n)$ 
```

This statement corresponds to the fifth Peano axiom. Agda uses the right arrow to sequentially assign the necessary conditions, with the conclusion appearing to the right of the last right arrow. The fifth Peano axiom describes that for any function f , if:

1. $f(0)$ holds true,
2. $f(n)$ being true implies that $f(\text{suc}(n))$ is also true

then for all natural numbers n , $f(n)$ holds true.

```
axioms-5 f0 fn-sucn zero = f0
axioms-5 f0 fn-sucn (suc n)
  with axioms-5 f0 fn-sucn n
... | fn = fn-sucn n fn
```

This illustrates the proof of the fifth Peano axiom. This straightforward proof highlights a few commonly used proof techniques in Agda:

1. **Pattern Matching on Variables:** The example splits the natural number n into two cases: zero and $\text{suc}\ n$, for separate discussion. This technique allows for detailed examination of different scenarios directly related to the structure of natural numbers.

2. Mathematical Induction on Natural Numbers: Theorems involving natural numbers often leverage induction. In Agda, we typically prove a case for n to infer the case for $\text{suc } n$. Agda ensures that there is a corresponding proof for the base case (usually zero), which acts as the termination condition for a comprehensive proof.
3. Simplifying Variables Using 'with': In the example, the proof for $f(n)$ is obtained using axiom-5, and it is named fn . This approach helps to avoid verbose variable expressions, streamlining the proof.

2.2 Reversible Abstract Machine

The Reversible Abstract Machine, abbreviated as RevMachine, is defined as follows.

field

State : Set ℓ

$_ \mapsto _ : \text{Rel State } \ell$

deterministic : $\forall \{st\ st_1\ st_2\} \rightarrow st \mapsto st_1 \rightarrow st \mapsto st_2 \rightarrow st_1 \equiv st_2$

deterministic_{rev} : $\forall \{st\ st_1\ st_2\} \rightarrow st_1 \mapsto st \rightarrow st_2 \mapsto st \rightarrow st_1 \equiv st_2$

"State" is the set of all states.

" \mapsto " is used to record state transitions, for example, $st_0 \mapsto st_1$ indicates that st_0 transitions to st_1 .

"Deterministic" refers to forward determinism, meaning that identical states will transition to the same next state.

"deterministic_{rev}", on the other hand, is the opposite of "deterministic", indicating that for each state, all possible transitions to it are the same.

Based on the definitions above, given a RevMachine and one of its states, we can determine an invariant trace composed of states.

Chapter 3

Narrow Reversible Machine Termination

3.1 Formulate Statement

First of all, 先定義以下名詞：“initial state” means that in the machine, there is no other state reaching it. “the termination of machine” means we reach a state in the machine, and there is no next state can be reached. The last state also be called the “stuck state” When we said “the machine will terminate from an initial state” means that starting from initial state, 經過 machine 有限次的推移，we will reach an stuck state.

We have a reversible abstract machine, and 指定 one of the state as initial state. It's known that there are finite number of reachable states from initial state. With the forward deterministic of reversible machine, we could make the assumption that the initial state will terminate from the initial state.

Here is our 完整的 statement: “Given a reversible abstract machine, it will inevitably terminate from any initial state with a finite number of reachable states.” We call it “Broad RevTerminate Principle”.

In the statement, the number of total state in the machine may be infinite. For the informal logic proof, we can easily ignore those unreachable states. However, it's 相對困難 for the formal proof. Therefore, we make an 較狹義的 statement as below: “Given a reversible abstract machine with a finite number of total states, it will inevitably terminate from any initial state.” We call it “Narrow RevTerminate Principle”.

In the statement, we could say “we have walked through all the states, so it should be terminate.” in both informal and formal proof easily. We will make the 更詳細的證明 in the next section.

3.2 Informal Logic Proof

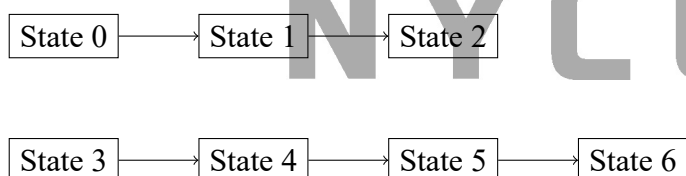
3.2.1 No Repeat

“No Repeat Principle” is formally proved in the paper “A Computational Interpretation of Compact Closed Categories: Reversible Programming with Negative and Fractional Types.” Here is the definition of the principle: Given a reversible abstract machine and an initial state, given two different natural numbers m and n , and the initial state reach st_m and st_n by walk through m and n steps. Then st_m should not equal to st_n .

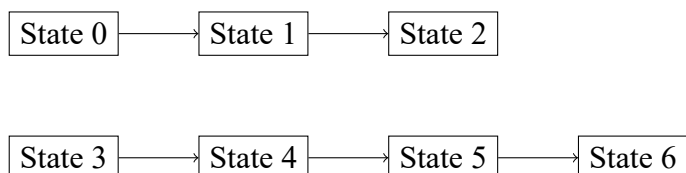
3.2.2 狹義的 statement

重述一次 the statement we have to proof: “Given a reversible abstract machine with a finite number of total states, it will inevitably terminate from any initial state.”

Here is the graph of the machine we expected. It seems monotonous. The initial state will 不重複地經過 all the other states and then terminate.



This is the other case for the machine with finite states. In the graph, the initial state also walk through finite state and terminate. However, not all the states are traversed.

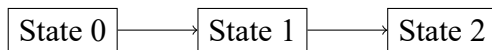


Assume number of all states is N . That is, we have st_0, st_1, \dots , and st_{N-1} . Assume st_0 is the initial state, we can check whether st_0 has next state. Once current state doesn't have next state, the machine will terminate. Therefore, we just have to consider the states starting from st_0 always have its next state.

當 trace 推移了 N 次, it should be like the graph below:



這裡，我們不特別提及已遍歷的前 n 個 state 應完全不重複，即使顯而易見地他們重複時會引發矛盾。我們將重點關注於 the whole trace with length of $N+1$. The reversible machine have only N states, but there are $N+1$ state we have traversed. According to pigeonhole principle, we can find two same states in the trace.



With No-Repeat principle, we know it's impossible for an initial state, walk through two different steps but reach same state. 到這邊，我們成功找出矛盾。The contradiction shows the length of trace is always less or equal than N , and then terminate.

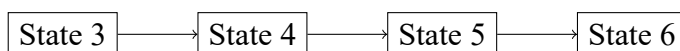
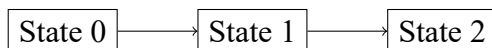
3.2.3 廣義的 statement

重述一次 the statement we have to proof: "Given a reversible abstract machine, it will inevitably terminate from any initial state with a finite number of reachable states."

Similar to Narrow RevTerminate Principle, we assume st_0 is the

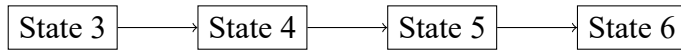
initial state, and 關注於 *there are always states after each state we walk through. The Broad RevTermi*

st_0 be able to reach. Assume the number of finite states st_0 be able to reach is N . 我們與先前一樣走 N 步以讓 the length of trace be $N+1$.



There are $N+1$ state we have traversed. And now, there are only N states st_0 be able to

reachable. According to pigeonhole principle, we can find two same states in the trace.



With No-Repeat principle, we know it's impossible for an initial state, walk through two different steps but reach same state. 到這邊，我們成功找出矛盾。The contradiction shows the length of trace is always less or equal than N , and then terminate.

陽明交大
NYCU

Chapter 4

Broad Reversible Machine Termination

4.1 Formulate Statement

The definition of broad reversible machine shows below:

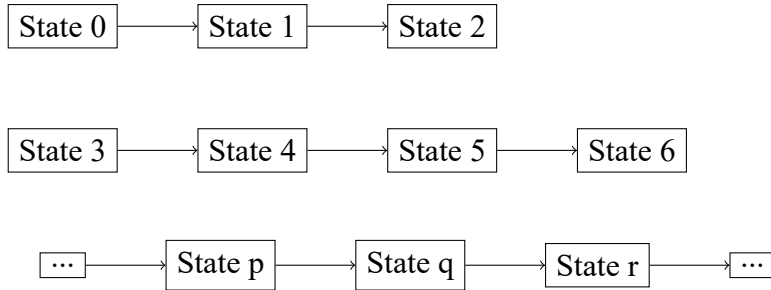
Finite-Reachable-State-Termination : $\forall \{N \ sto\}$

$\rightarrow (St-Fin : \exists [m] \exists [st_m] (sto \mapsto [m] st_m) \text{ Fin } N)$

$\rightarrow \text{is-initial } sto$

$\rightarrow \exists [st_n] (sto \mapsto^* st_n \wedge \text{is-stuck } st_n)$

It is similar to the narrow one, except for the Bijection. In the Statement of narrow reversible machine termination, we consider the number of all states is finite. On the contrary, the Statement of broad reversible machine termination firstly given an st_0 , and 限定了 the number of "reachable states from st_0 " is finite. In this restriction, the number of states could be infinite. Some initial states in machine may also not be terminated unless we know the initial state can just reach finite states.



In the example above, we know the initial state 0 has only 2 reachable states (state 1 and state 2), therefore, it will terminate after limited steps. 類似的情況，state 3 can also terminate by 3 steps. 與 narrow reversible termination statement 不同的是，或許 State 的總數是無限

的。而事實上，the third trace shows 未知數量的 states. 不過即使 State 的總數可能無限，we still can 把握前兩個 trace can terminate.

4.2 informal Logic Proof

We also use Mathematical Induction to proof the statement. Here is the 流程

1. given an initial state st_0
2. if there is no step from st_0 , then it terminate; otherwise, we reach st_1
3. keep 推進 the trace, until reach st_n .
4. When we reach st_n , we have traversed all n reachable states.
5. if st_n not terminate, consider the next state of st_n , the piegonhole principle says that there are two states will be same, but there are in different steps from st_0 obviously.
6. The statement will occurs a contradiction with No-repeat principle.

Similar to the narrow reversible termination, however, the bijection 從 “State to index” 變成了 “State to reachable state index”. In fact, in informal logic proof, 除了 the bijection 被替換了, all processes are the same.

4.3 Formal Logic Proof

Base on the informal logic proof, we continue 討論 the descriptions of the formal one. Same as the narrow reversible termination proof, we have the following steps:

1. Starting from initial state, try looking for its next state.
2. If the next state exists, keep looking for 再下一個 state

3. Upon reaching the n-th state, proof that it's impossible for the existence of next states with no-repeat principle.

4.3.1 Countdown rules

We add countdown rule to make agda terminate.

Finite-Reachable-State-Termination-CountDown : $\forall \{N\} \{sto\}$

$\rightarrow (St-Fin : \exists [m] \exists [st_m] (sto \mapsto [m] st_m) \text{ Fin } N)$
 $\rightarrow (has-next : \forall (st : \text{State}) \rightarrow \text{Dec } (\exists [st'] (st \mapsto st')))$
 $\rightarrow \text{is-initial } sto$
 $\rightarrow \forall cd\ m \quad st_m \rightarrow cd + m \equiv N \rightarrow sto \mapsto [m] st_m$
 $\rightarrow \exists [st_n] (sto \mapsto^* st_n \times \text{is-stuck } st_n)$

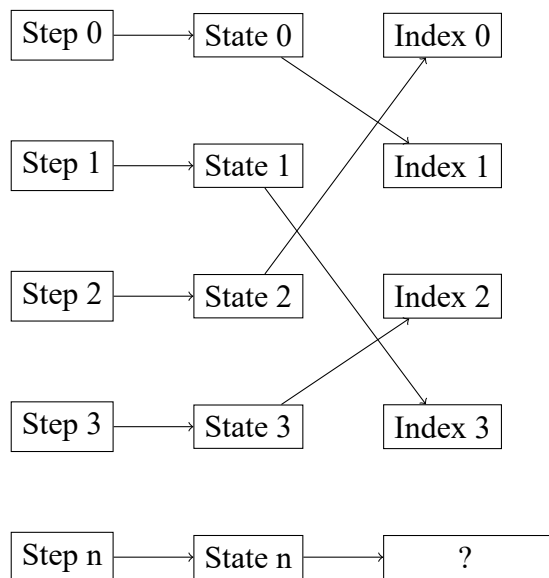
Our target Principle can be regarded as the special case when Countdown is N, 也就是剛開始的時候。With countdown rule, we could check if each state we through has next state. Once it doesn't have next state, it will terminate immediately. Otherwise, we will get N-th state.

4.3.2 At n-th state

The proof of 「當抵達 n-th state 會終止」 can be simply 分為 the two steps below :

1. 假定 n+1-th state 存在，with Pigeonhole principle，可以在 n 個 state 映射到 n + 1 個 state，而找到重複經過的兩個相同 state
2. No-repeat 告訴我們找到的兩個 state 不該是相同的，因為他們的 step 不同，得到矛盾。

Here is the mapping graph to explain the steps:



The number of reachable state is only n . That is, the Step n should be mapping to 重複的 index. And we'll 藉由 the bijection to find out state n is also 重複 with one of previous states. It's a contradiction with No-repeat principle.

陽明交大
NYCU

Chapter 5

Conclusion

陽明交大
NYCU

Appendix A

附錄標題

A.1 Testing

陽明交大
NYCU