(Pre)Thesis draft

Hakim Boulahya

May 7, 2018

Contents

Todo list

Chapter 1

Introduction

1.1 Motivation

Automata theory is used in various field in Computer Science, especially in Computer-Aided Verification.

Problems such as synthesis [FJR09] or the universality problem [DWRLH06] has shown to be PSPACE-complete [**?**, DWRLH06].

More efficient algorithms to resolve those problems have been implemented using antichain based-algorithms. Antichains are data structures that allow to represent a partial order sets, in a more compact way. .

cite the original papers of those problems

Ref for complexity of univers. pbl

Talk about the problems, complexity and alternative (Safra vs antichain)

1.2 Objective

The goal of this thesis is to provide an efficient implementation of data structures used in those algorithms, especially antichain. We will mainly focus on implementing antichain-related data structure and provide a library to be used in different tool such as Acacia+, Owl.

The first, and main, objective is to implement an Antichain object in Java to be used in Owl. Then if possible used this implementation for other tool such as Acacia+, by either providing bindings or other.

As we mainly focus on effiency, it could be interesting to use a C implementation and provide binding to a Java class.

This small paragraph is an open discussion

1.3 Related work

AaPAL library is a that was implemented in the context of Bohy's PhD thesis to provide an antichain library and be able to implement the antichains based algorithm for the synthesis problem.

Java already provide built-in implementation for Set.

Includes limitation of Java built-in and different possible solution for antichains found on stack overflow

Chapter 2

Data Structures

In this section, we will provide formal definitions of the data structures that we will implement. We recall the notion of binary relations and important propreties of such relations. We then define partially ordered set, totally order set and closed set. Finally we give a formal definition for antichains.

The definitions and examples for this section are based on [Boh14b].

2.1 Binary relations

A binary relation for an arbitrary set S is a set of pair $R \subseteq S \times S$. There are five important properties: reflexitivity, transitivity, symmetry, antisymmetry and total.

A relation R on S is said to be:

- Reflexive: iff $\forall s \in S$ it holds that $(s, s) \in R$
- Transitive: iff $\forall s_1, s_2, s_3 \in S$, if $(s_1, s_2) \in R$ and $(s_2, s_3) \in R$ then it holds that $(s_1, s_3) \in R$
- Symmetric: iff $(s_1, s_2) \in R$ then $(s_2, s_1) \in R$.
- Antisymmetric: iff $(s_1, s_2) \in R$ and $(s_2, s_1) \in R$ then $s_1 = s_2$
- Total: iff $\forall s_1, s_2 \in S$ then $(s_1, s_2) \in R$ or $(s_2, s_1) \in R$

is this Total def correct ?

Orders A *partial order* is a binary relation that is *reflexive*, *transitive* and *antisymmetric*. We note a partial order relation by R. We note $s_1 R s_2$ to show the belonging of a binary relation to a partial order, which is equivalent to $(s_1, s_2) \in R$. A *total order* is a partial order that is *total*.

2.2 Partially ordered set

An arbitrary set S associated with a partial order $\preceq$ is called a *partially ordered set* or *poset*. It is denoted by the pair $\langle S, \preceq \rangle$.

Comparable Let $s_1, s_2 \in S$ and $\langle S, \preceq \rangle$ a poset. The two elementes s_1 and s_2 are called *comparable* if either $s_1 \preceq s_2$ or $s_2 \preceq s_1$. If neither of those two comparaisons are correct, then s_1 and s_2 are called *uncomparable*.

give definition of (un)comparable using partial order definition

In this thesis we are more interested in partial ordered sets as total order sets can be easily implemented as lists.

Is this total order affirmation correct ?

Ordered sets Partial/Order sets
Lattice (semi upper and lower)
Closed sets
Antichain
Includes propreties to implement

2.3 Antichains and pseudo-antichains

Chapter 3

Implementation

3.1 Summary of objectives

The main focus of the thesis is to be able to provide an efficient implementation of antichains and pseudo-antichains in `Java`. The first step is to provide an interface for the different operations that can be applied to antichains. We then give a description of the implementation. Antichains provide a way to represent in a compact way partially ordered set that are closed. Pseudo-antichains are an extension of antichains and provide a compact way to represent partially ordered sets. Pseudo-antichains does not specifically require closed set.

3.2 Existing implementation

Everything below is a fast/draft notes

3.3 Motivation and objective

For the moment, the implementation to represent the data strutures in Acacia+ is specifically designed for a specific set. The idea is to propose a new library implementation to provide an API that will implement important data structures that are used in syntesis algorithms.

The objective of the thesis is to provide an efficient library to represent antichain data structures and implement different operation. An final goal is to be able to use this library within Aaron/Acacia.

* Impl. datastrcture antichain in Java * Theoritacal context: synthesis, unverversality and automata theory known problem * Practical context: Owl, Acacia+ and AaPAL

* Why interesting: More efficient than CTL symbolic with BDD

Related Work

* Java Built-in IMPL * AaPAL * Stackoverflow * Acacia ?

3.4 Notions

Model checking and synthesis

Model checking is the process to verify a software model with specifications.
Synthesis is the process to derive the system from specifications.
[**?**]
[FJR09]

Data structures

Binary Decision Diagram

Antichain BDD vs antichain
Antichain vs pseudo-antichain

3.5 Questions

* Maastricht library ? [BBFR] [Boh14a] * How to references ? * What to impl (about operations) ?
* Is implementing LTL Rea. or Universality a final goal of the thesis ?

Bibliography

[BBFR] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin. Acacia+ website `http://lit2.ulb.ac.be/acaciaplus`.

[Boh14a] Aaron Bohy. Aapal website `http://lit2.ulb.ac.be/aapal`, 2014.

[Boh14b] Aaron Bohy. *Antichain based algorithms for the synthesis of reactive systems*. PhD thesis, Université de Mons, 2014.

[DWRLH06] Martin De Wulf, Jean-François Raskin, Doyen Laurent, and Thomas A. Henzinger. Antichains: A new algorithm for checking universality of finite automata. 2006.

[FJR09] Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. An antichain algorithm for ltl realizability. 2009.