UNIVERSITÉ LIBRE DE BRUXELLES

DÉPARTMENT D'INFORMATIQUE

MEMO-F-403
PREPARATORY WORK FOR THE MASTER THESIS

Thesis - draft #1

Author:
Hakim BOULAHYA

Supervisor:
Emmanuel FILIOT

May 6, 2018

Contents

1 Introduction

1.1 Motivation

Automata theory is used in various field in Computer Science, especially in Computer-Aided Verification. Multiple algorithms has been theorized to provide more efficient way to resolve those problem .

Those different algorithms used a data structure called antichain. An antichain is a subset of a set that allow to represent it, in a more compact way.

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.

Talk about unversality and LTL real.

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.

2 Preliminaries

In this section, we will provide formal definitions of the data structures that we will implement. We recall the notion of set, partially ordered set and total order set. We then give a definition for antichains.

The definitions and examples for this section are based on ...

insert ref of examples

2.1 Formal definitions

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

A relation R on S is said to be:

- Reflexive: iff for all $s \in S$ it holds that $(s, s) \in R$
- Transitive: iff for all $s_1, s_2, s_3 \in S$, if $(s1, s2) \in R$ and $(s2, s3) \in R$ then it holds that $(s1, s3) \in R$

- 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:

total definition

Order A *partial order* is a binary relation that is reflexive, transitive and antisymmetric. We note a partial order relation by ... To note the appartenance of a binary relation to a partial order, we note s1 .. s2 which is equivalent to (s1, s2) in ... A total order is a partial order that is total.

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

give definition of (un)comparable using partial order definition

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

Is this total order affirmation correct ?

2.2 Existing implementation

Everything below is a fast/draft notes

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

3.1 Related Work

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

4 Notions

4.1 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.
[Boh14b]
[FJR09]

4.2 Data structures

Binary Decision Diagram

Antichain BDD vs antichain
Antichain vs pseudo-antichain

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 ?

References

[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.

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