Data structures for Partially Ordered Sets

Prepatory work for the master thesis

June 7, 2018

Hakim BOULAHYA

supervised by
Emmanuel Filiot and Guillermo A. Pérez

Département d’Informatique
Université Libre de Bruxelles
Belgium

ULB

Contents

1. **Introduction** Subject and motivations
2. **Definitions** Poset and antichains
3. **Existing implementation**
4. **Objective** Requirements and overview

Introduction

Subject

- Implement data structures to represent partially ordered sets
- Main focus on antichain-based algorithms in automata theory

Motivations

- There exists new algorithms that uses antichains, for known pratical and theoritical problems:
  - Model checking [3]
  - Synthesis problem [4]
  - Language universality [2]
- There is a need for an efficient implementation of antichains

Partial order

A partial order $\preceq$ on a set S is a binary relation $\preceq \subseteq S \times S$ that is reflexive, transitive and antisymmetric.

Partially ordered set (or poset)

A partially ordered set is a pair $\langle S, \preceq \rangle$ where S is a set, and $\preceq$ a partial order on S.

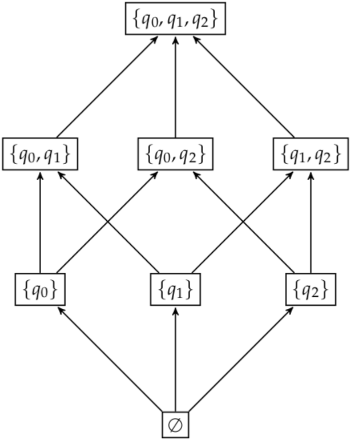
Antichains

An antichain α is a set of incomparable elements of poset $\langle S, \preceq \rangle$ w.r.t. to $\preceq$.

Interestring operations

- **Lower closure** of an antichain α on S, denoted $\downarrow\alpha$, is the set of all elements of S that are smaller or equal to an element of α
- **Appartenance** $s \in \downarrow\alpha_1$
- **Inclusion** $\downarrow\alpha_1 \subseteq \downarrow\alpha_2$
- **Union** $\downarrow\alpha_1 \cup \downarrow\alpha_2 = \downarrow\lceil\alpha_1 \cup \alpha_2\rceil$
- **Intersection** $\downarrow\alpha_1 \cap \downarrow\alpha_2$

Example



- Poset is $\langle 2^Q, \subseteq \rangle$, where $Q = \{q_0, q_1, q_2\}$
- $\alpha = \{\{q_0, q_2\}, \{q_1, q_2\}\}$ is an antichain
- Can retrieve all subset of cardinality 1 using the closure on the antichain: $\downarrow\alpha$
- $\downarrow\alpha = \{\{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}\}$

Existing implementations

Do implementation already exists ?

Yes, 2 for antichains:

1. **AaPAL**[1]: Antichains implementation in *C*
   - More of an API: user must implement comparison and intersection
2. Antichains for the Dedekind number problem [5]
   - Bitarray represention (a bit is a subset)
   - Can only be applied to natural numbers

Requirements

- `Owl` library: an automata library
- But symbolic antichain-based algorithm are missing (including antichain data structures)
- Implementation in `Java 10`

Objective

What to do ?

- Provide an API and implement it against `Owl`
- Provide some implementation for antichains depending on the universe of the sets
- Define the algorithms to test our antichains implementation against
- Study the performance of those implementations

References I

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

[2] M. De Wulf, L. Doyen, T. A. Henzinger, and J. F. Raskin.
Antichains: A New Algorithm for Checking Universality of Finite Automata.
In *Computer Aided Verification*, volume 4144, pages 17–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[3] Laurent Doyen and Jean-François Raskin.
Improved Algorithms for the Automata-Based Approach to Model-Checking.
In Orna Grumberg and Michael Huth, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4424, pages 451–465. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

References II

[4] Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin.
An Antichain Algorithm for LTL Realizability.
In *Computer Aided Verification*, volume 5643, pages 263–277.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[5] Pieter-Jan Hoedt.
Parallelizing with mpi in java to find the ninth dedekind number.
2015.

Thank you!

ULB