Fast Cone-Of-Influence Computation and Estimation in Problems with Multiple Properties

C. Loiacono, M. Palena, P. Pasini, D. Patti, S. Quer, S. Ricossa, D. Vendraminetto

joint work with

J. Baumgartner

IBM Corporation USA

This work is related to a Date 2013 Conference Poster

Outline

- Introduction
- Standard Cone Of Influence(COI) computation
- Labeled COI computation
- Using COI
- Experimental results
- Conclusions

Introduction

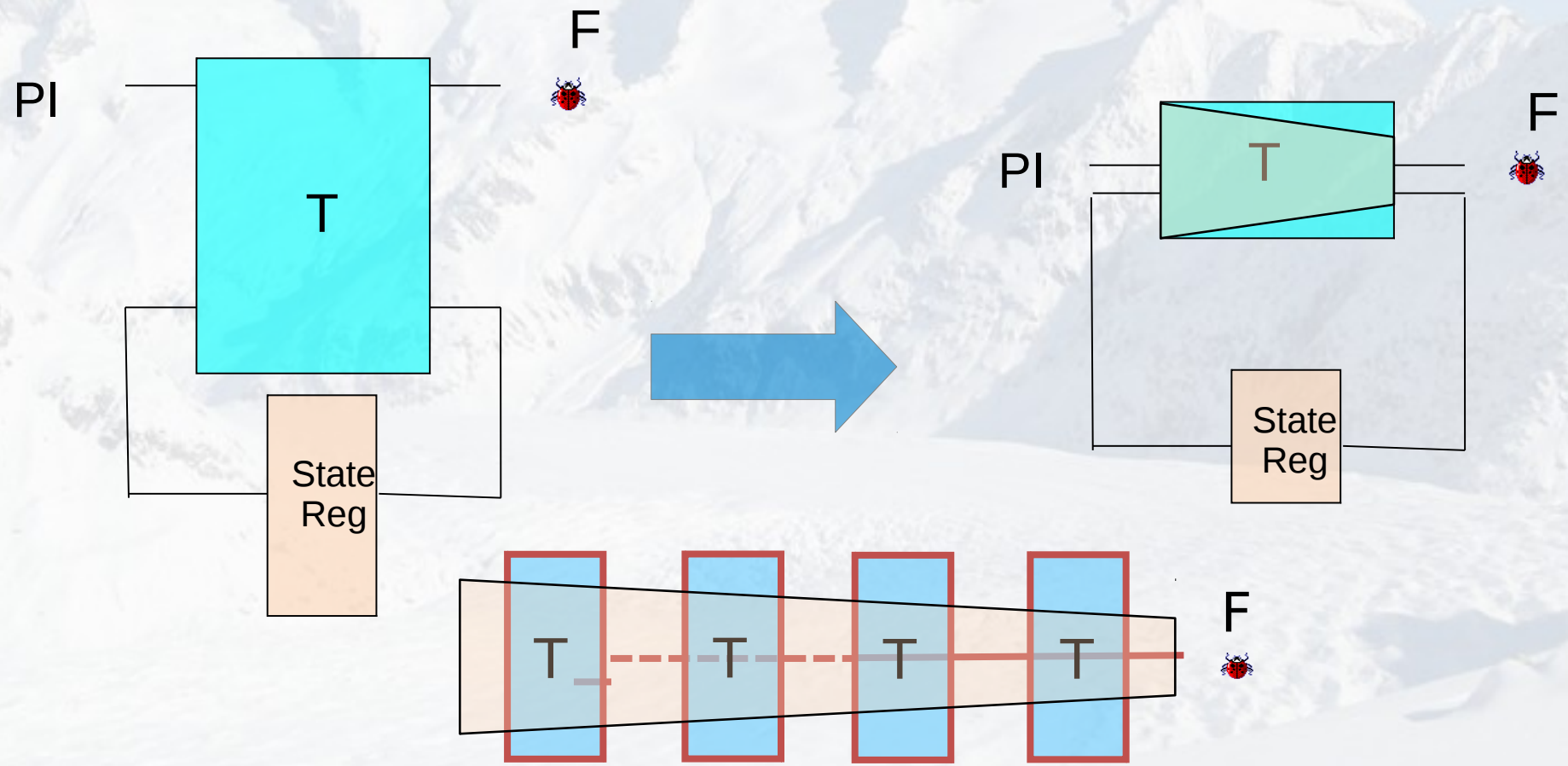
- The Cone of influence reduction(COI) is a fundamental technique to simplify designs in Hardware Model Checking
- Given a model represented as Finite State Machine (FSM)
  - specified using some state variables
  - the COI reduction simplifies the size of the model by eliminating variables not relevant to the property under verification

Introduction

- We address the frameworks where repeated COI computations are required for
  - multiple properties
  - Internal model nodes

  to avoid potentially quadratic slow down
- We wish to use COI information as property and/or variable scoring heuristics in various Model Checking algorithms

COI Reduction

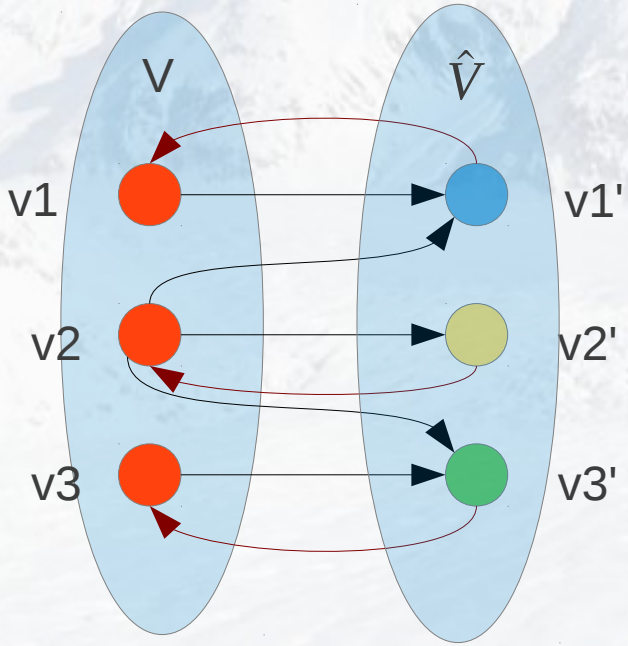


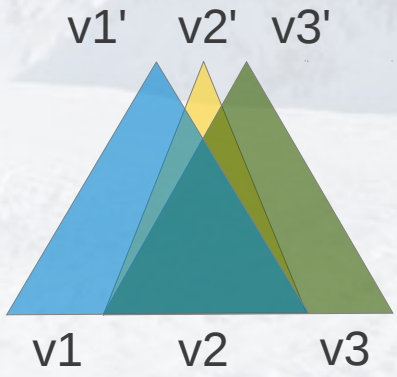
Standard COI computation

- The standard algorithm for computing COI(V) works on the variable dependency graph a bipartite graph where:
- V(present state) and V'(next state) variables are the nodes
- all (vj,vi') edges represent a dependency of the next state variable vi' upon present state variable vj

Standard COI computation

- The algorithm basically implements a backward traversal of the graph starting from all variables of $\hat{V}$
- The final COI is the subset of the reached V nodes





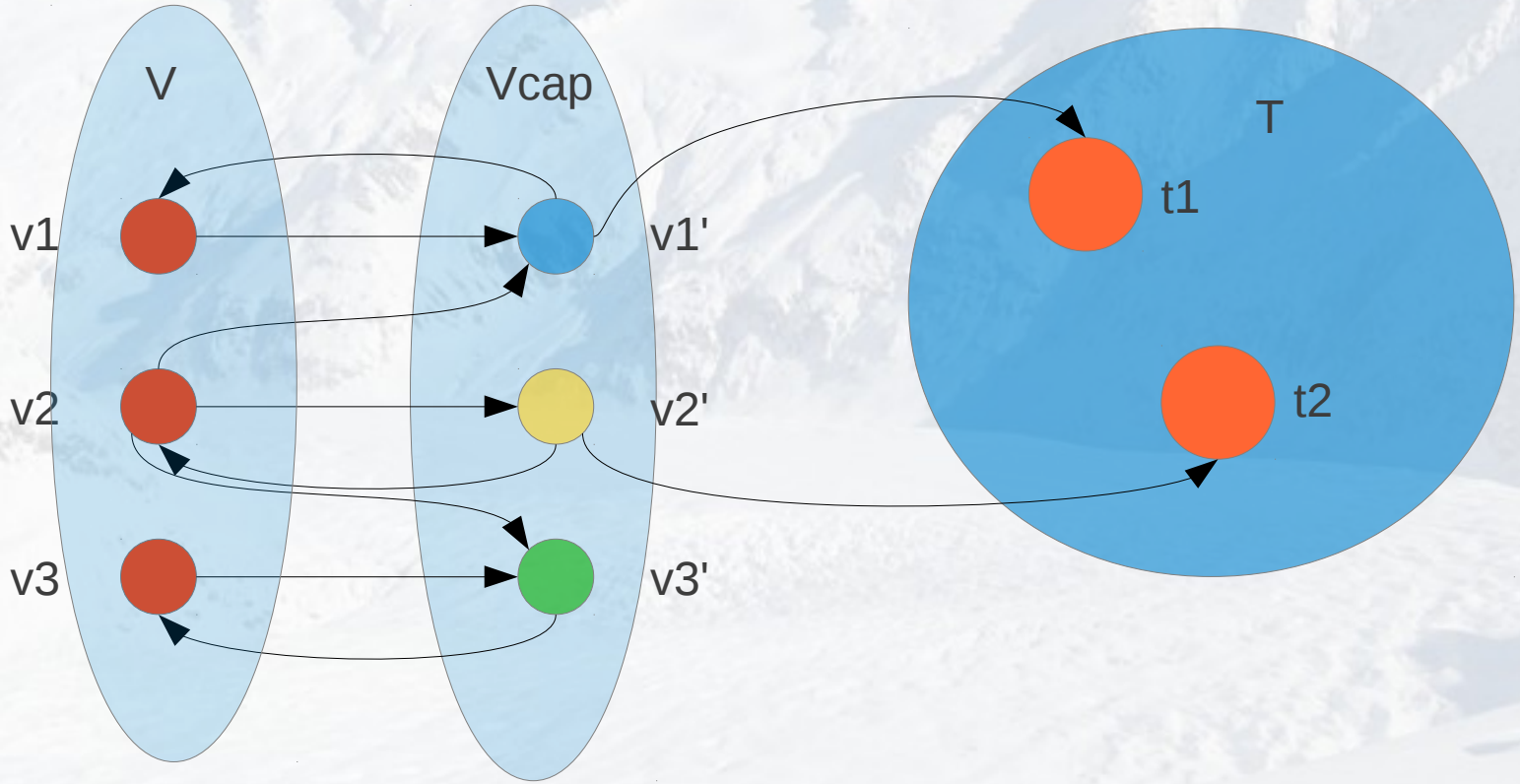
Standard COI computation

- Though computing each COI has a linear-time solution this process may become computationally expensive when repeated COI evaluations are necessary
- For example, when multiple properties need to be verified, and each requires an independent COI computation
- The base algorithm described would need to be applied repeatedly. This entails obvious overhead when multiple properties have overlapping COIs due to the sub-graph re-traversal
- The overall COI computation process may degrade to requiring quadratic resources

Labeled COI computation

- Our approach follows the graph labeling approach, in which graph nodes are assigned labels such that after labeling , the mutual reachability between nodes can be decided by inspecting labels alone.
- We associate to all nodes a visited flag and a bit array encoding → Bitmap where i-th bit correlates to the i-th present state variable vi
- Our bitmaps thus have one bit per state variable.

Labeled COI computation



Labeled COI computation

- Initially all visited flag are set to false, and all nodes except present state are labeled with a 0 bitmap(Bmp)
- V nodes are labeled with a one-hot encoding of their variable index:

Bmp(vi)= OneHot(i)

OneHot(0)=..00001

OneHot(1)=..00010

Labeled COI computation

- For each target ti we perform a backward depth-first traversal of unvisited nodes.
- “backward” refers to the direction followed for edges in the dependency graph
- Bitmaps are propagated in the forward direction
- For each node, set the visited flag to true, and we recur for all adjacent fan in nodes

Labeled COI computation

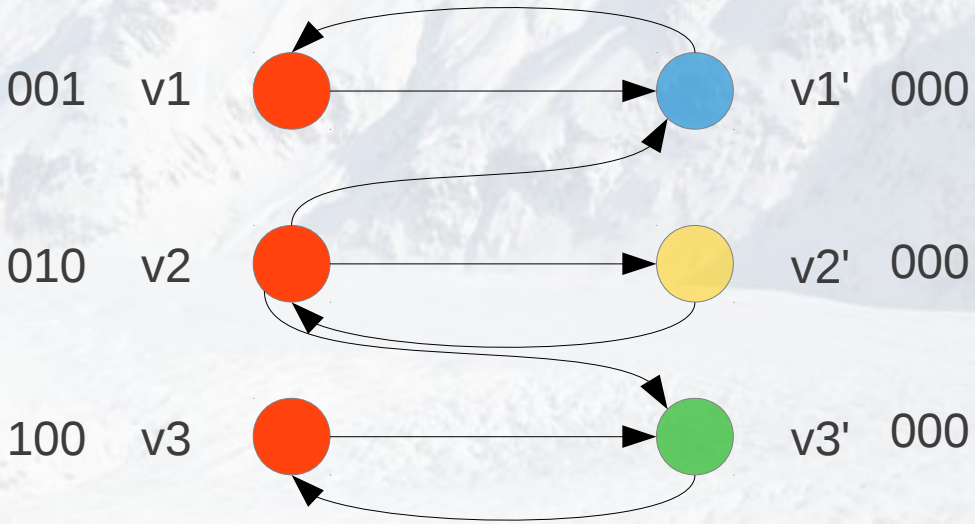
- Whenever node nj is reached by node ni the label of nj is bitwise Ored with the label of ni:

$$Bmp(nj)=Bmp(nj)|Bmp(ni)$$

- The topological order followed by the DFV guarantees that labels fully represent COI dependencies

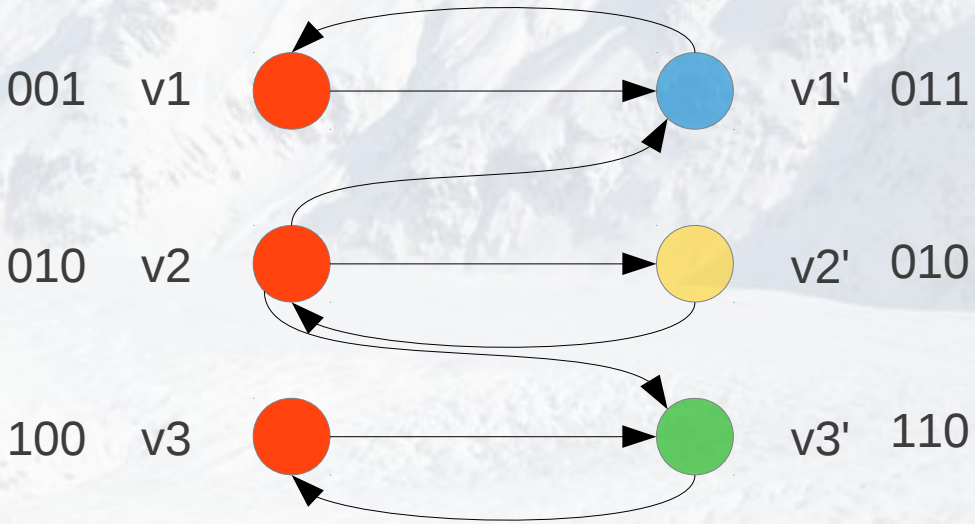
Labeled COI computation

Initializations

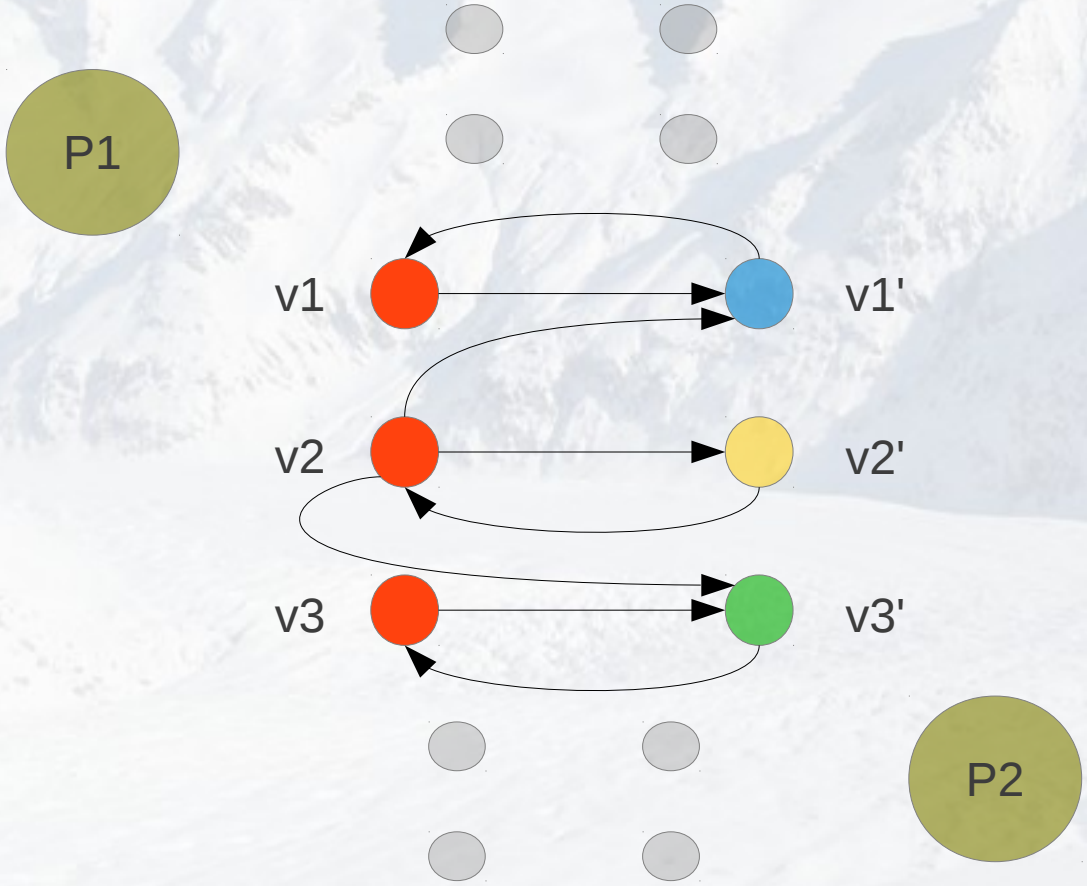


Labeled COI computation

Final result



Labeled COI computation



Strong Connected Component reduction

- Many circuits comprise one or more SCCs within each node may reach each other node
- SCC can be identified using Tarian's linear time algorithm
- Each SCC can be collapsed into a single representative node
- SCC can be used to avoid loops in the dependency graph

Using COI

Sorting and Grouping/Clustering Properties

- A first application of multiple COIs is the verification of multiple properties of the same model
- Whenever the number of properties is high COIs can be exploited for:
  - Sorting properties based on COI size
  - Grouping/clustering two or more properties into a single verification problem

Using COI

Sorting and grouping/clustering variables

- Another potential application for multiple COI analysis is to augment existing algorithms that statically and/or dynamically sort/group state variables, in BDD and SAT-based model checking
- We propose to compute COIs of individual state variables and consider COI statistics as a base for
  - Exploit heuristics for variable sorting
  - Partitioned transition relation management
  - Partitioned image computation

Using COI

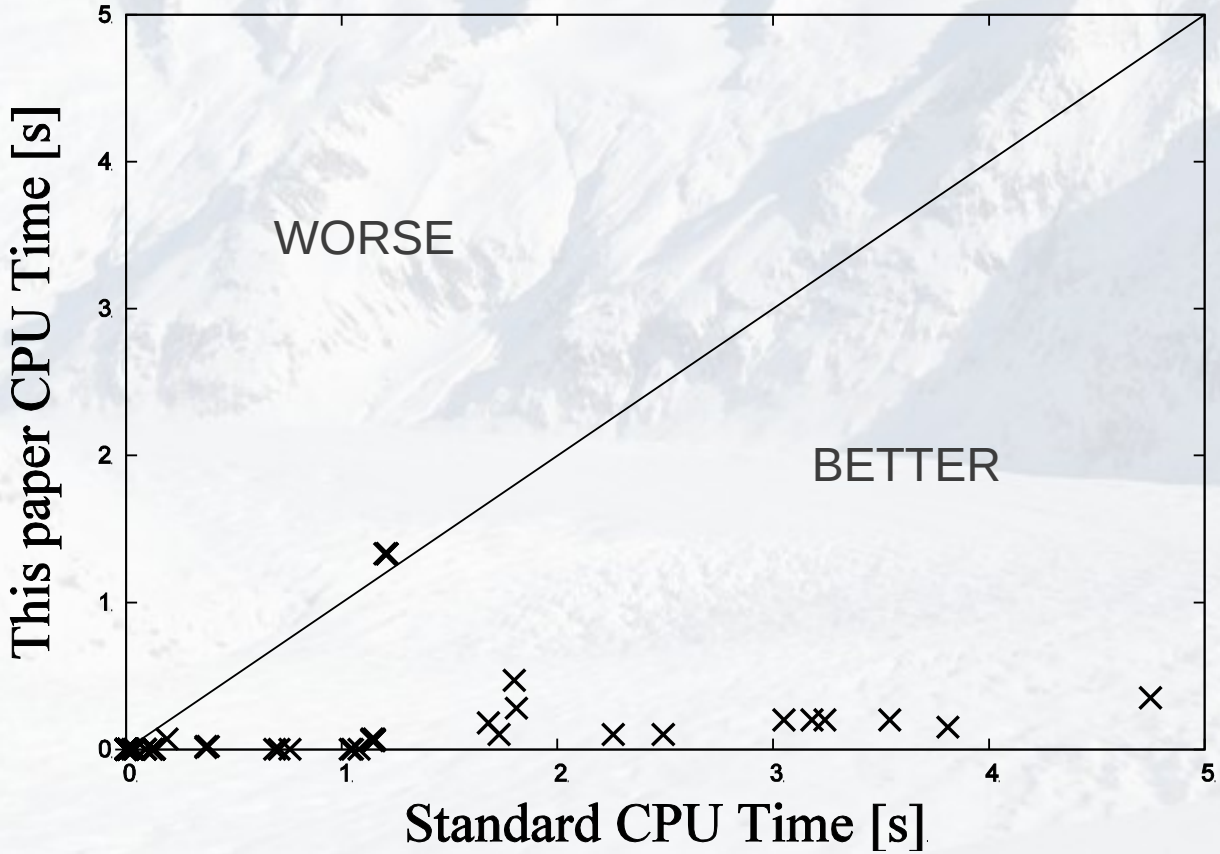
Partitioned transition relation management using COI (On going work)

- We are investigating the question of how to perform partitioning in reachability based verification using COI informations
- Clustering algorithms
  - K-means
  - Hierarchical
    - Single
    - Complete
    - Average

Experimental Results

- We run experiments on the multi-property suite of the HWCC'11
- HWCC'11 consists of 24 benchmarks a some of them with more than 1000 properties
- Our prototype ran an Intel i7 860470/2010 Workstation with 8 MB cache memory, a clock speed of 2.8 G Hz, 4 cores 8 threads, 8 GB of main memory DDR III 1333, and hosting a Ubuntu 12.04 LTS Linux distribution

Experimental Results



Conclusions

- The work introduces new techniques for a fast computation, estimation , and application of the COI of multiple properties
- In order to avoid multiple repeated traversals of the same some sub-graph our algorithm is based on graph node labeling, and it performs a single visit of the variable dependency graph
- It costs is linear in time but quadratic in memory

Fast Cone-Of-Influence Computation and Estimation in Problems with Multiple Properties

C. Loiacono, M. Palena, P. Pasini, D. Patti, S. Quer, S. Ricossa, D. Vendraminetto

Joint work with

J. Baumgartner

IBM Corporation USA

This work is related to a Poster published on Date 2013 Conference