FraDeCo(P)P 2012

Proofs of pointer algorithms by inductive representation of graphs

Mathieu Giorgino

Ralph Matthes Martin Strecker

Université Paris-Est Créteil

15/05/2012

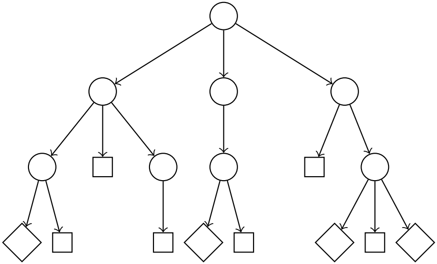
Outline

1. Context
   - Introduction
   - Approach
   - Verification of graph transformations
2. Schorr-Waite
   - Demo
   - Description
   - Verification
3. BDD
   - Description
   - Verification
4. Conclusion

First remark

Most usual data structures are tree-shaped with several kind of additional pointers:

- sharing pointers
- root pointers
- father pointers
- ...

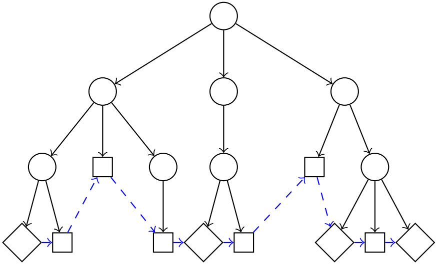


For numerous algorithms, different views on data-structures can be used for the verification

First remark

Most usual data structures are tree-shaped with several kind of additional pointers:

- sharing pointers
- root pointers
- father pointers
- ...

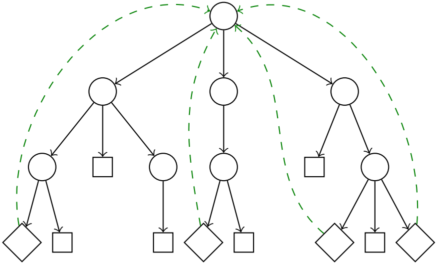


For numerous algorithms, different views on data-structures can be used for the verification

First remark

Most usual data structures are tree-shaped with several kind of additional pointers:

- sharing pointers
- root pointers
- father pointers
- ...

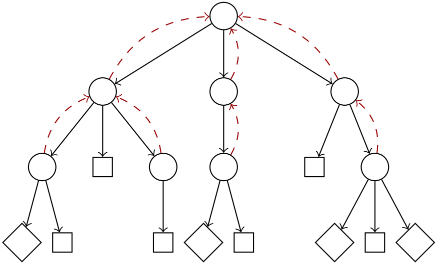


For numerous algorithms, different views on data-structures can be used for the verification

First remark

Most usual data structures are tree-shaped with several kind of additional pointers:

- sharing pointers
- root pointers
- father pointers
- ...

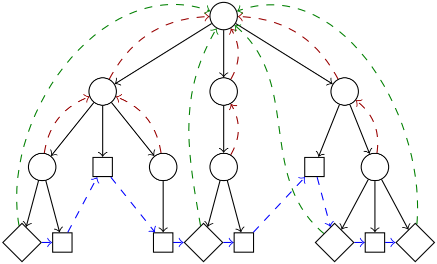


For numerous algorithms, different views on data-structures can be used for the verification

First remark

Most usual data structures are tree-shaped with several kind of additional pointers:

- sharing pointers
- root pointers
- father pointers
- ...

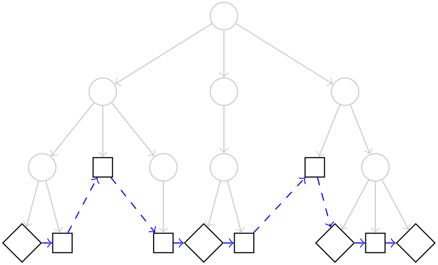


For numerous algorithms, different views on data-structures can be used for the verification

First remark

Most usual data structures are tree-shaped with several kind of additional pointers:

- sharing pointers
- root pointers
- father pointers
- ...



For numerous algorithms, different views on data-structures can be used for the verification

Verification

Objective

Generate efficient and verified programs
(with pointers and/or mutable structures)

Safe code generation

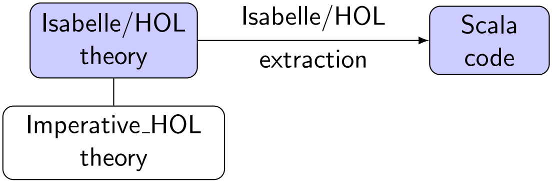
Abstract Language

- Arborescent data structure
  + pointers
- Properties verified by induction

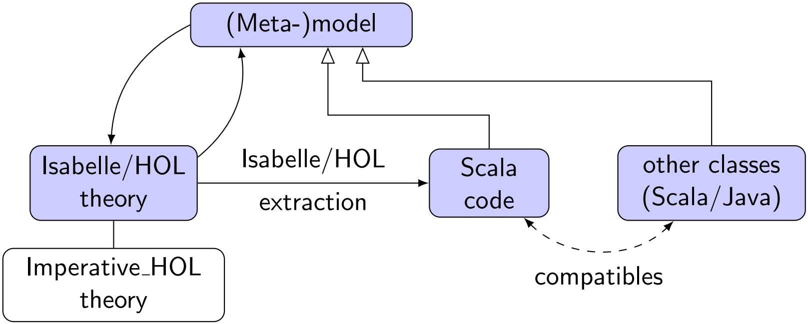
Imperative langage

- Efficient :
  - Sharing
  - Mutability
- Same properties

General frame



General frame



Related work

Graph representation

- Nodes and edges
- Coinduction
- Trees + pointers:
  - PALE [MS01]
  - Term-graph rewriting in Tom using relative positions [BB08]
  - Locally nameless encoding of lambda-terms [Cha09]
  - B+ trees functional representation with pointers [MM10]

Verification

- Proofs on raw heap and pointers
- Hoare (Separation) Logic [Rey02]

Case studies - Overview

Schorr-Waite

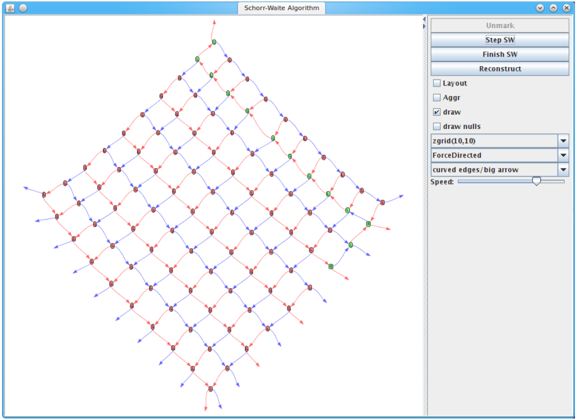
- Arbitrary rooted graph (with outgoing arity ≤ 2)
- High mutability
- Proof by simulation
- LOPSTR'2010 [GSMP10]

BDD

- Rooted acyclic graph (shared tree)
- References comparisons
- Direct proof
- TAAPSD'2010 [GS10], FoVeOOS'2011 [GS11]

Outline

Demo


Schorr-Waite Algorithm
Unmark
Step SW
Finish SW
Reconstruct
Layout
Aggr
draw
draw nulls
zgrid(10,10)
ForceDirected
curved edges/big arrow
Speed:

Algorithm features

Purpose

- Marking graphs without using more space (stack, ...)
- Traversing a tree by terminal recursivity and without stack

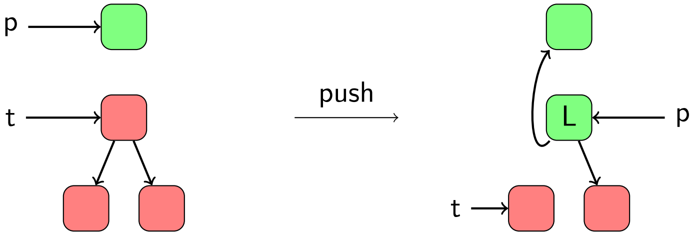
Use

Garbage collector, case study...

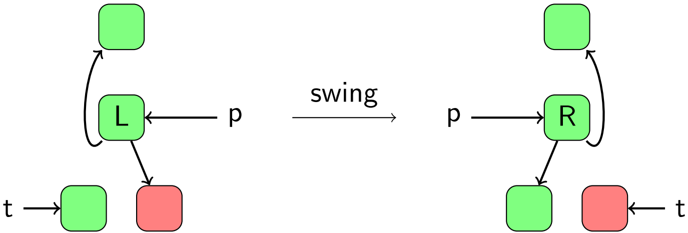
Principle

- Modification of the graph pointers to store the path to the root
- 2 variables containing the pointers :
  - t: to the current node
  - p: to the previously visited node

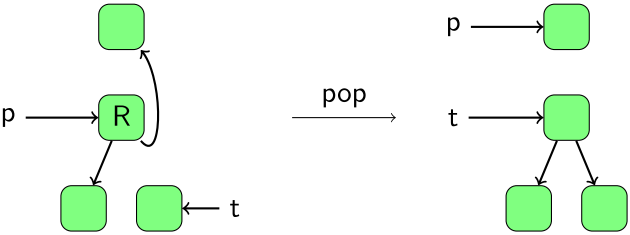
Steps : Push



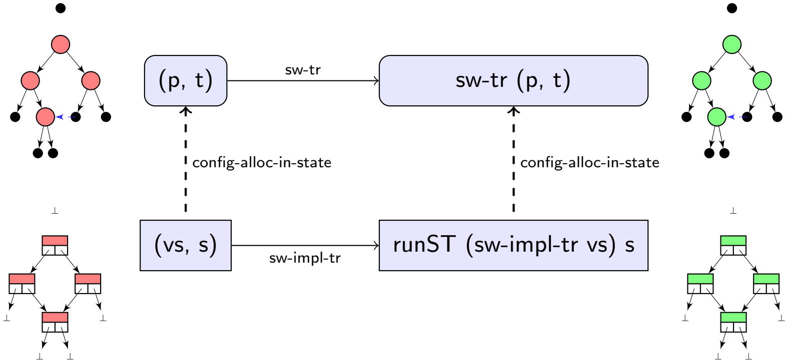
Steps : Swing



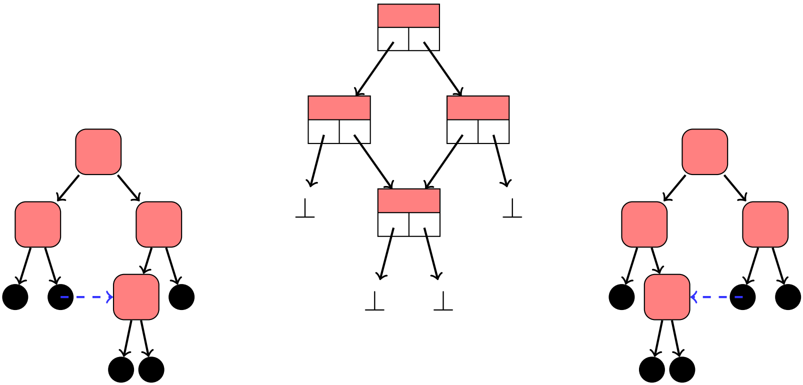
Steps : Pop



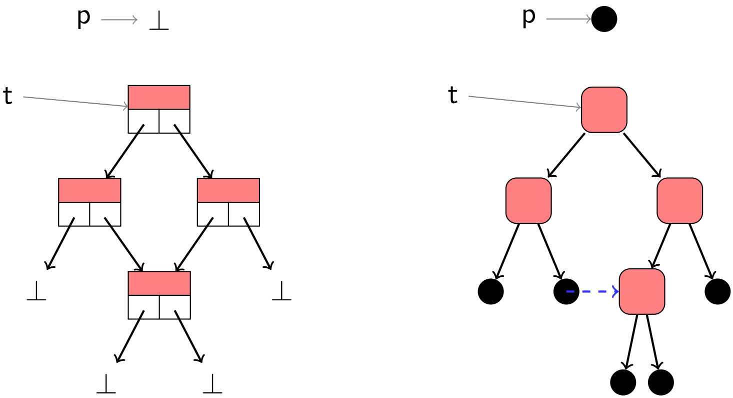
Proof by simulation



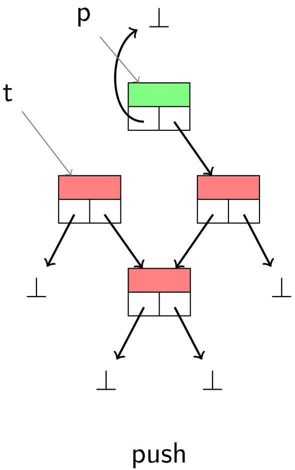
Choosing the spanning tree: Example with 2 possibility

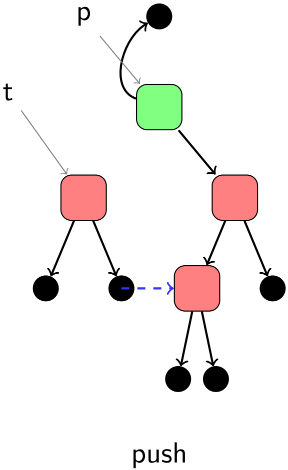


The wrong one!

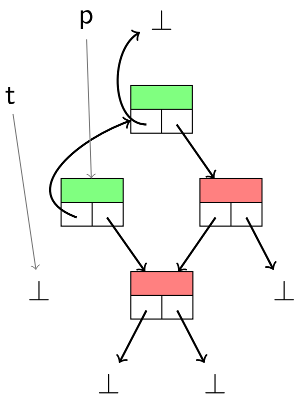


The wrong one!

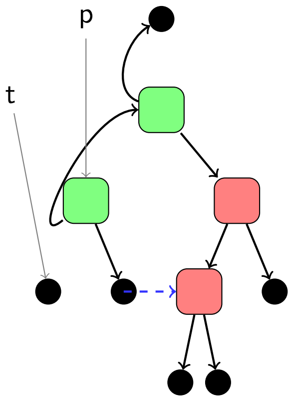




The wrong one!

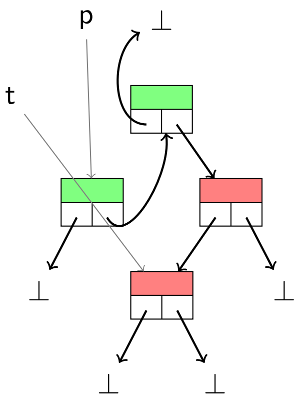


push

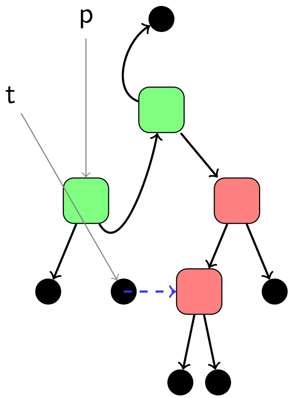


push

The wrong one!

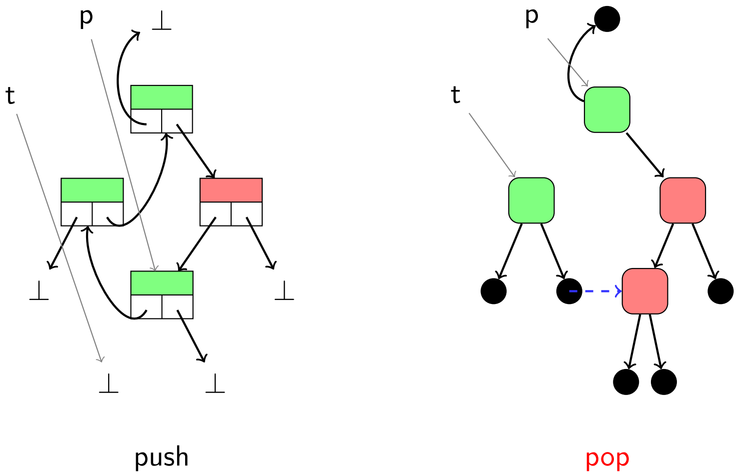


swing

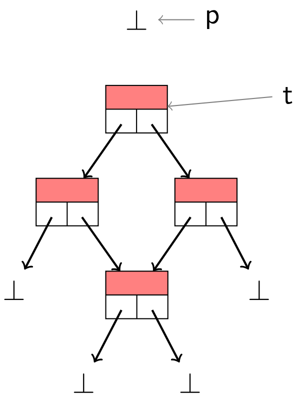


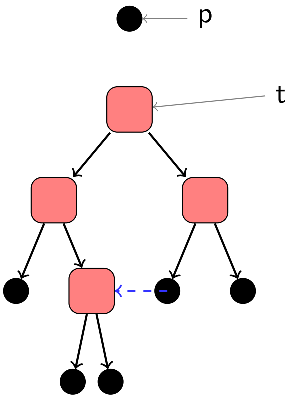
swing

The wrong one!

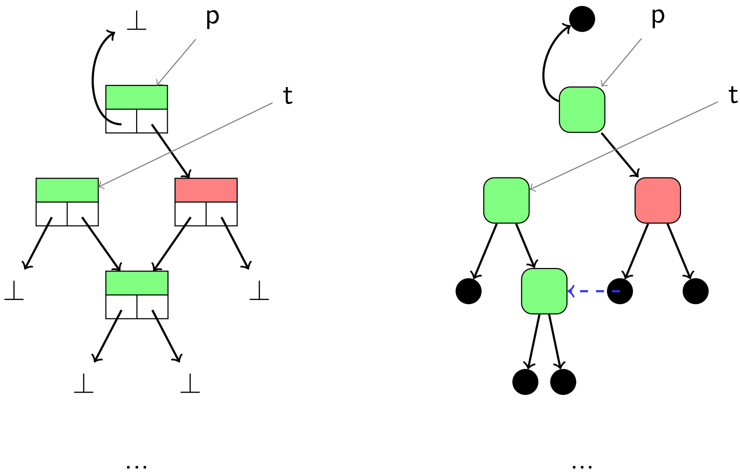


The good one!

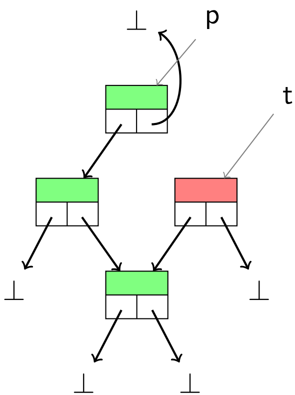




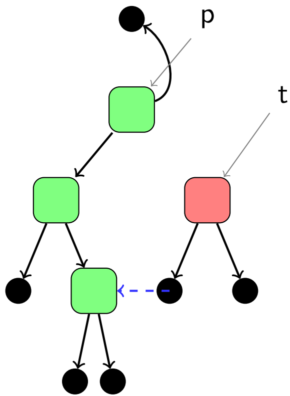
The good one!



The good one!

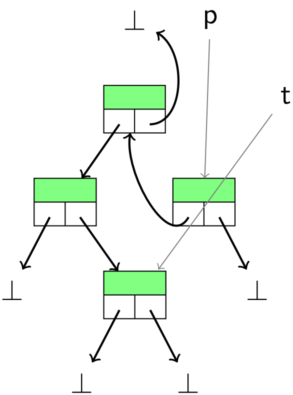


swing

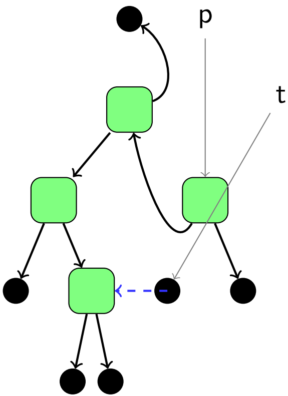


swing

The good one!

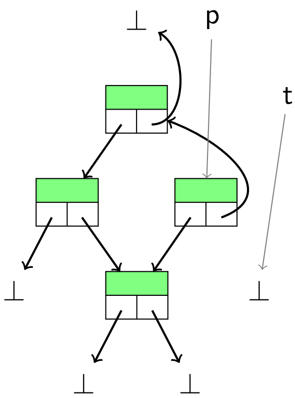


push

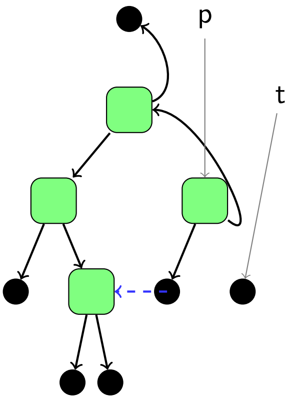


push

The good one!



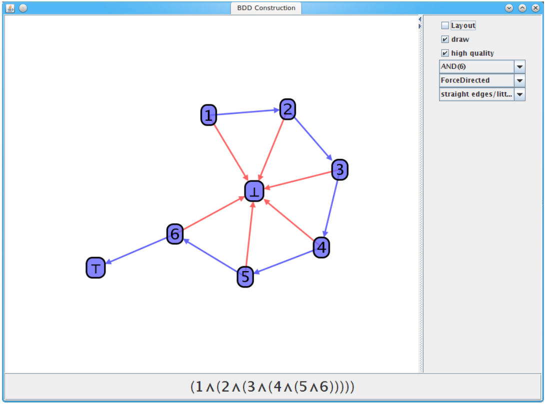
swing



swing

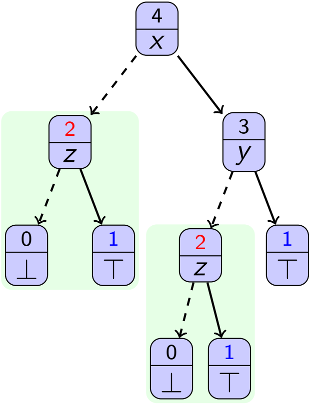
Outline

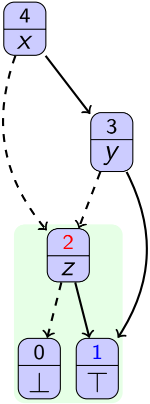
Demo


BDD Construction
Layout
draw
high quality
AND(6)
ForceDirected
straight edges/litt...
1
2
3
⊥
4
5
6
⊤
(1∧(2∧(3∧(4∧(5∧6)))))

Sharing

We add references to represent sharing. They also allow to add mutable content in the nodes.



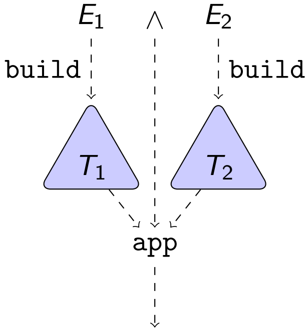


0	nbrefs $=$ 0 …
1	nbrefs $=$ 0 …
2	nbrefs $=$ 0 …
3	nbrefs $=$ 0 …
4	nbrefs $=$ 1 …
5	nbrefs $=$ - …

⋮

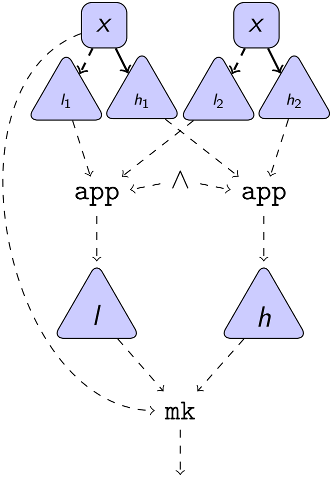
build

```
primrec build ::
    'v expr ⇒ (bool, 'v) rtree Heap
where
  build (Var i) = do{
   cf ← constLeaf False;
   ct ← constLeaf True;
   mk i cf ct
  }
| build (Const b) = (constLeaf b)
| build (BExpr bop e1 e2) = do{
   n1 ← build e1;
   n2 ← build e2;
   app bop (n1, n2)
  }
```



app

```
function app ::
  (bool ⇒ bool ⇒ bool)
  ⇒ ((bool, ′v) rtree ∗ (bool, ′v) rtree)
  ⇒ (bool, ′v) rtree Heap
where
  app bop (n1, n2) = do {
    if tpair is-leaf (n1, n2) then
      constLeaf (bop (leaf-val n1) (leaf-val n2))
    else do {
       let ((l1, h1), (l2, h2)) =
         select split-lh dup (n1, n2);
       l ← app bop (l1, l2);
       h ← app bop (h1, h2);
       mk (varOfLev (min-level (n1, n2))) l h
    }
  }
```



Memoization and Garbage Collection

Memoization

Records previous computations results to reuse them

- little change in functions and proofs

Garbage collection

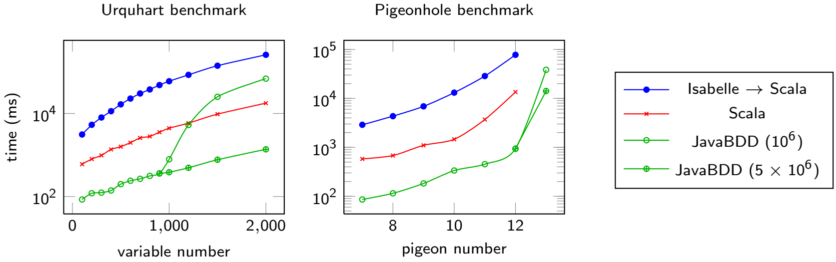
Removes no more used BDDs from the maximal sharing table

- several reference counter mutations in functions and proofs
- weakening of an invariant

Main theorem & benchmarks

Equivalent expressions construct the same BDD

lemma build-correct: $[\![\forall t \in \text{trees s1} \cup \text{trees s2. robdd-refs t};$
$\text{wf-heap s1}; \ \text{effect (build e1) s1 s1}' \text{ t1};$
$\text{wf-heap s2}; \ \text{effect (build e2) s2 s2}' \text{ t2}]\!]$
$\Longrightarrow \text{(interp-expr e1 = interp-expr e2)} \longleftrightarrow \text{struct-equal (t1, t2)}$



Outline

Conclusion

Approach

Representing graphs as (trees + pointers) to verify transformations

Schorr-Waite

- Arbitrary rooted graph (with outgoing arity ≤ 2)
- High mutability / Proof by simulation

BDD

- Rooted acyclic graph (shared tree)
- References comparisons / Direct proof

Conclusion

Approach

Representing graphs as (trees + pointers) to verify transformations

Schorr-Waite

- Arbitrary rooted graph (with outgoing arity ≤ 2)
- High mutability / Proof by simulation

BDD

- Rooted acyclic graph (shared tree)
- References comparisons / Direct proof

Thanks for your attention

Emilie Balland and Paul Brauner.
Term-graph rewriting in Tom using relative positions.
In Ian Mackie, editor, *4th International Workshop on Computing with Terms and Graphs TERMGRAPH 2007 ENTCS*, volume 203 of *ENTCS*, pages 3–17, Braga Portugal, 2008. ELSEVIER.

Arthur Charguéraud.
The locally nameless representation.
Unpublished.
http://arthur.chargueraud.org/research/2009/ln/, July 2009.

Mathieu Giorgino and Martin Strecker.
Bdds verified in a proof assistant (preliminary report).
In A.V. Anisimov and M.S. Nikitchenko, editors, *Proc. TAAPSD*, Univ. Taras Shevchenko, Kiev, 2010.

Mathieu Giorgino and Martin Strecker.

Towards the verification of efficient bdd algorithms.
In *Formal Verification of Object-Oriented Software (FoVeOOS)*, 2011.

Mathieu Giorgino, Martin Strecker, Ralph Matthes, and Marc Pantel.
Verification of the Schorr-Waite algorithm – From trees to graphs.
In *Logic-Based Program Synthesis and Transformation (LOPSTR)*, 2010.
http://www.irit.fr/~Mathieu.Giorgino/Publications/GiSt2010SchorrWaite.html.

J. Gregory Malecha and Greg Morrisett.
Mechanized verification with sharing.
In *ICTAC*, pages 245–259, 2010.

Anders Møller and Michael I. Schwartzbach.
The pointer assertion logic engine.

In *Proc. ACM PLDI*, pages 221–231, 2001.

John C. Reynolds.
Separation logic: A logic for shared mutable data structures.
In *LICS '02: Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, pages 55–74, Washington, DC, USA, 2002. IEEE Computer Society.