

Workshop “Trustworthy Software” 2006

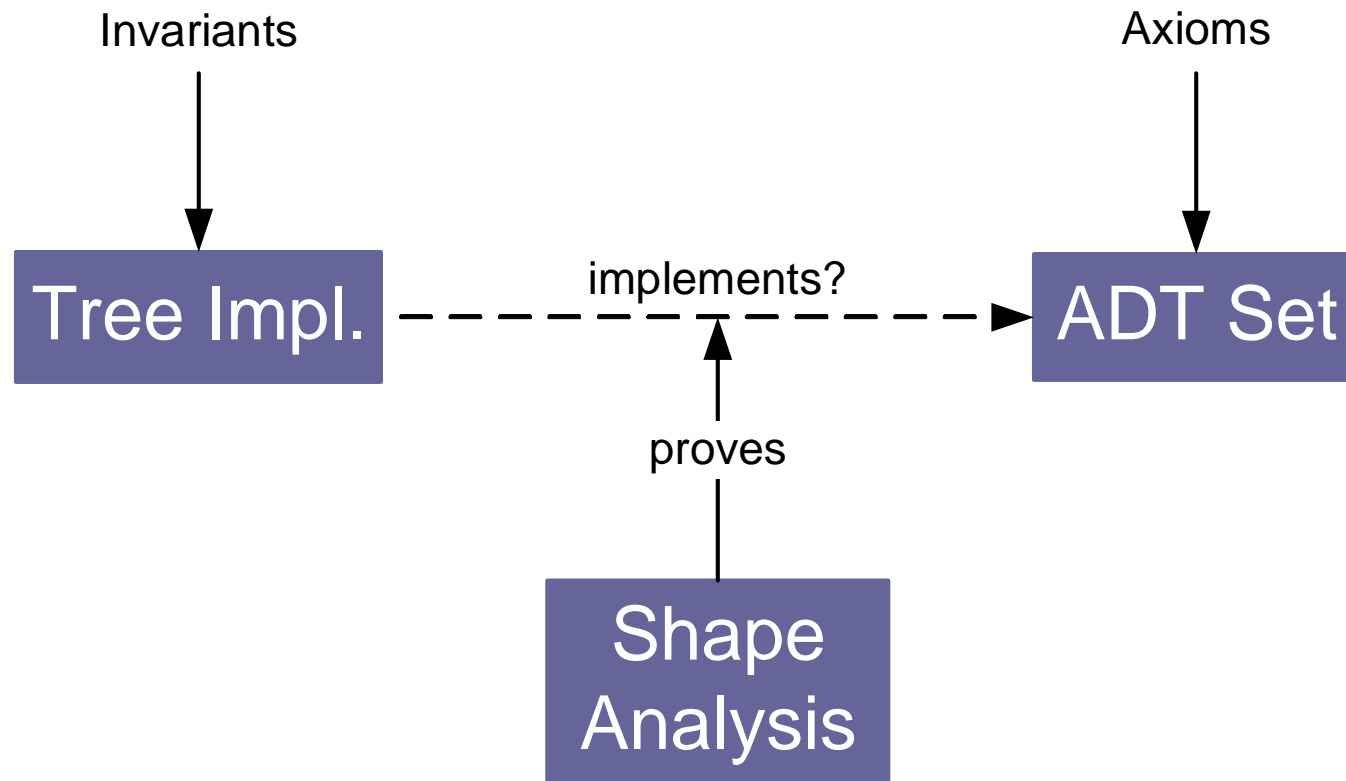
Shape Analysis of Sets

Jan Reineke

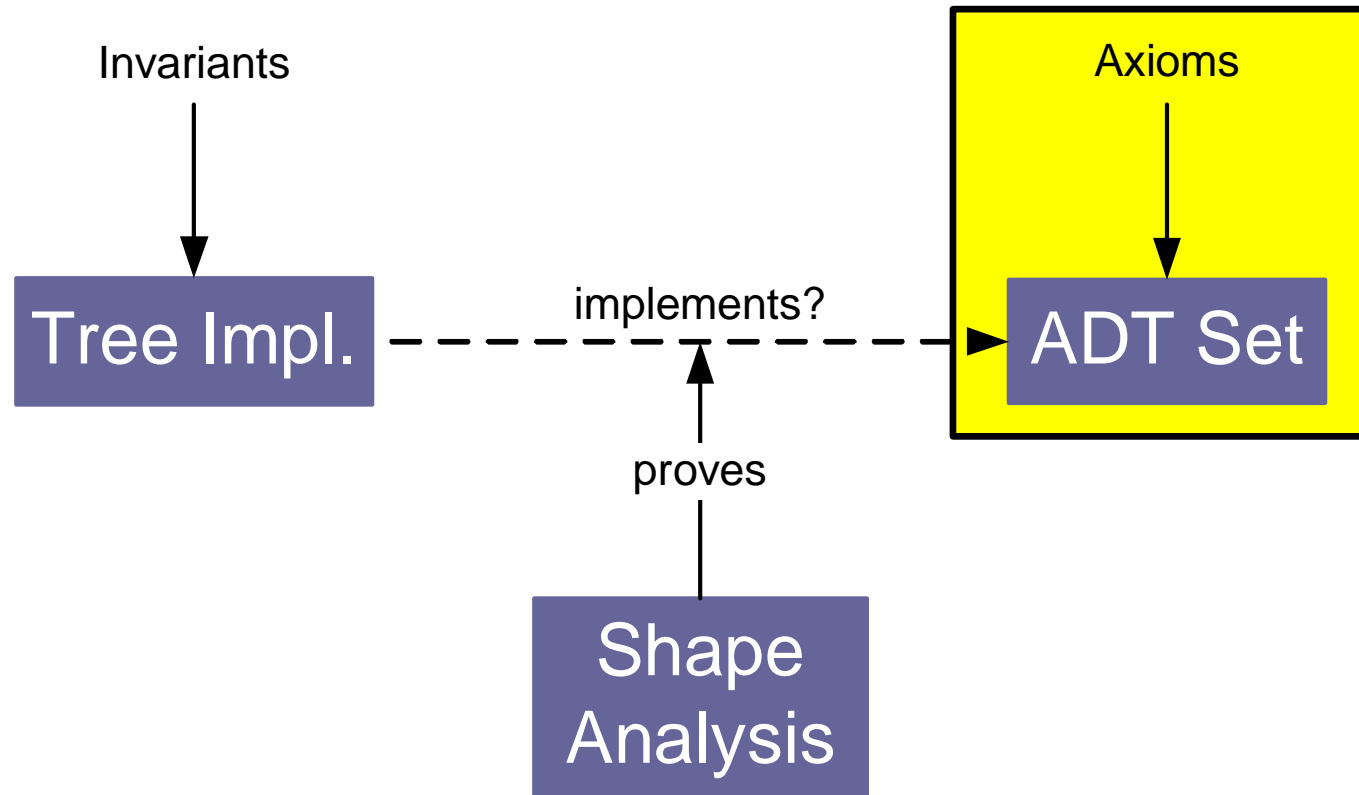
Compiler Design Lab, Saarland University

`reineke@cs.uni-sb.de`

The Big Picture



ADT Set



ADT Set - Algebraic Specification

- Collection of elements of a certain type
- Operations:
 - $\text{insert}(\cdot) : \text{set} \times \text{element} \rightarrow \text{set}$
 - $\text{remove}(\cdot) : \text{set} \times \text{element} \rightarrow \text{set}$
 - ...
- Predicates:
 - $\in : \text{element} \times \text{set}$
 - $\subseteq : \text{set} \times \text{set}$
 - $= : \text{set} \times \text{set}$

ADT Set Axioms (selection)

$$a \in s.\text{insert}(b) \leftrightarrow a =_{el} b \vee a \in s, \quad (1)$$

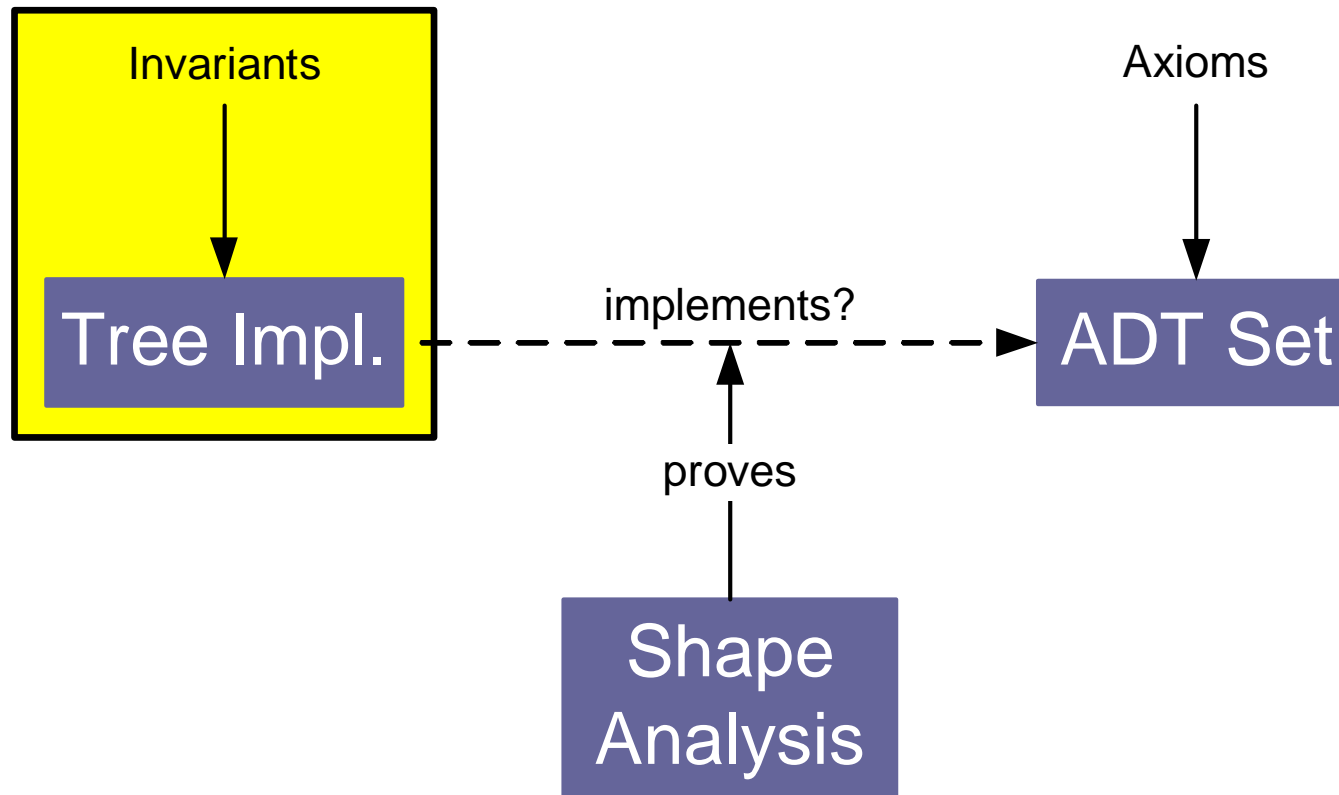
$$a \in s.\text{remove}(b) \leftrightarrow a \neq_{el} b \wedge a \in s, \quad (2) \quad \leftarrow$$

$$s \subseteq s' \leftrightarrow (\forall a. a \in s \rightarrow a \in s'), \quad (3)$$

$$s = s' \leftrightarrow (s \subseteq s' \wedge s' \subseteq s), \quad (4)$$

...

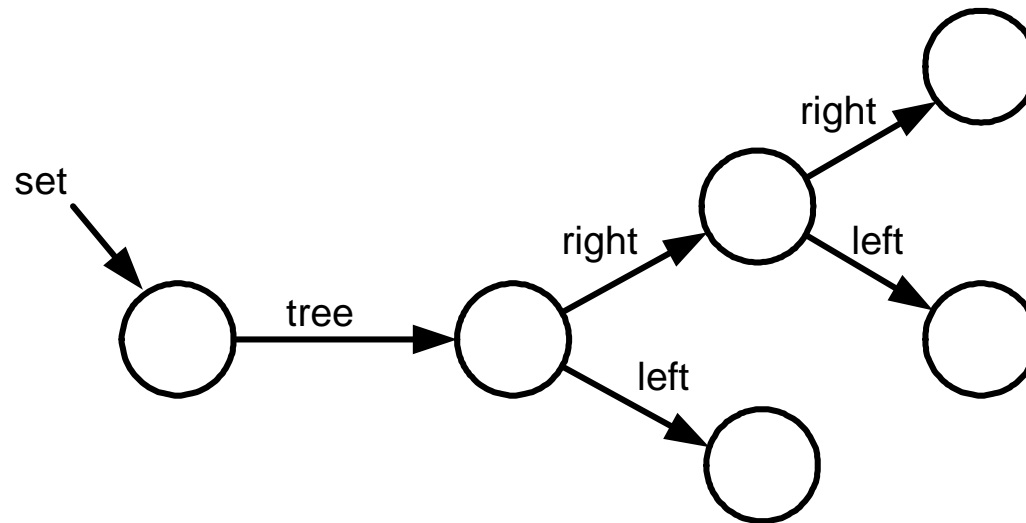
Tree-based Implementation



Structure Declarations

```
typedef struct Tree
{
    void* data;
    struct Tree* left;
    struct Tree* right;
} List;
```

```
typedef struct Set
{
    List* tree;
    int (*compare)(void*, void*);
    int size;
} Set;
```

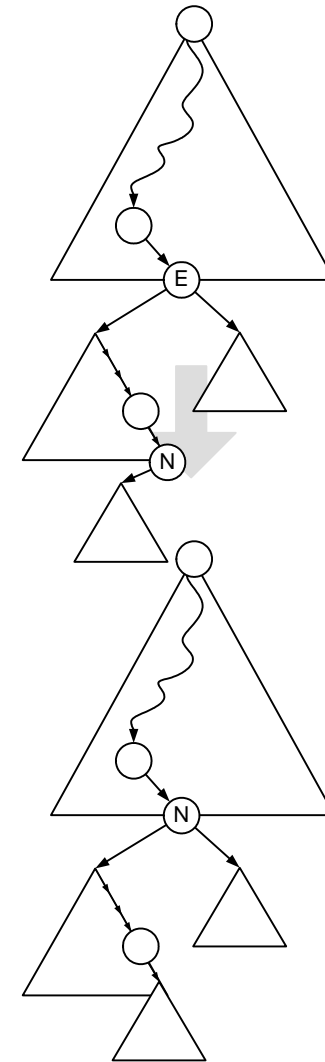
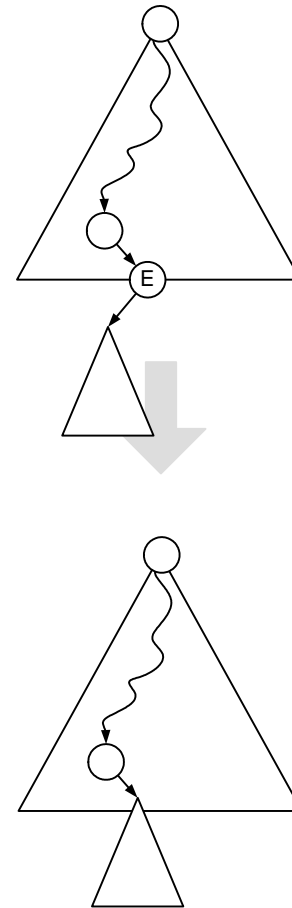
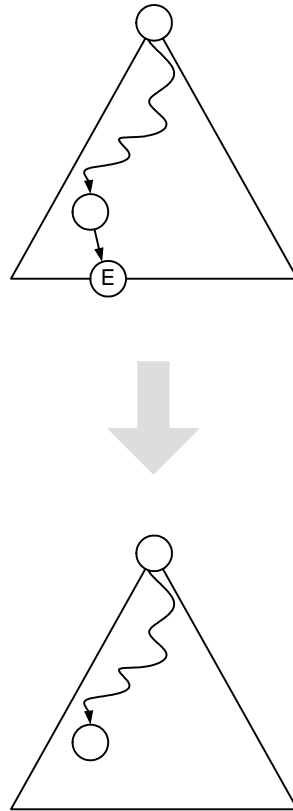
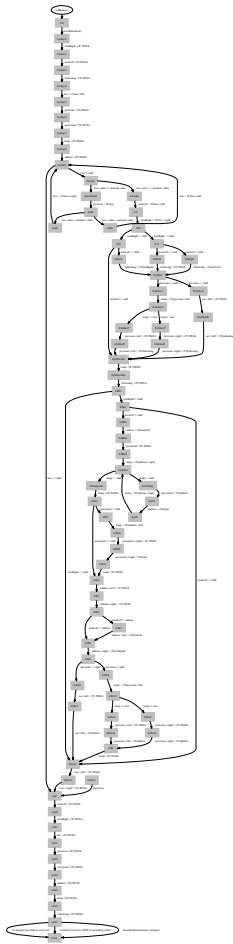


Data Structure Invariants

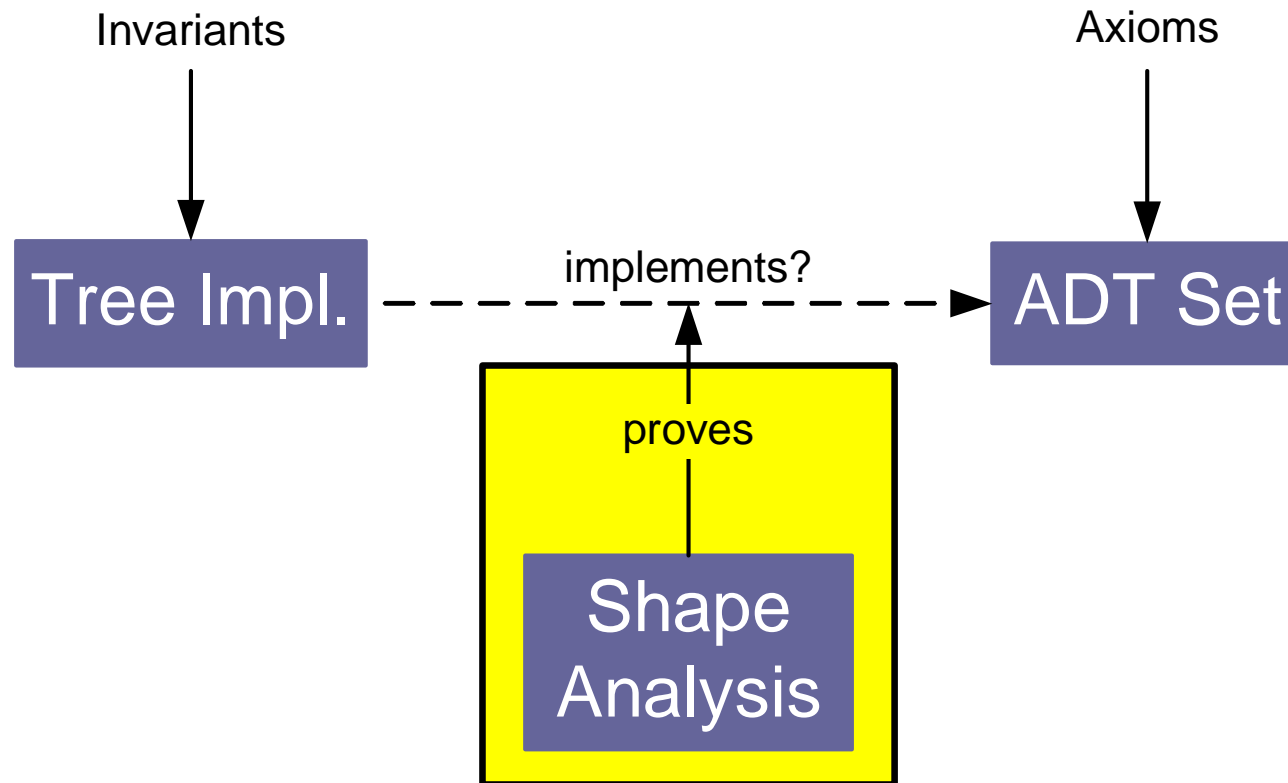
- The tree is in fact a tree ;-)
i.e. every node is reachable from *set* and pointed to from exactly one other node
- The tree is ordered:
left descendants are smaller, right descendants larger

⇒ both are formalized by instrumentation predicates

Removing an Element



Shape Analysis



Proving Compliance to ADT Axioms

Problem 1

- Problem 1: Axioms relate different routines of the set implementation: \in and `remove`
 $a \in s \text{ .remove } (b) \leftrightarrow a \neq_{el} b \wedge a \in s$
- Solution:
 - Represent \in predicate by *isElement* instrumentation predicate
 - Prove the equivalence of \in -implementation and *isElement*
 - Prove compliance of `remove`-implementation to axiom in terms of *isElement*

Proving Compliance to ADT Axioms

Problem 2

- Problem 2: Axioms relate state of predicates before and after execution:

$$a \in \text{s.remove}(b) \leftrightarrow a \neq_{el} b \wedge a \in s$$

- Solution: Remember old element relation *isElementOld*
 - Predicate is fixed before invocation of method
 - Allows to compare new and old values of element property
 - Primed vs. Unprimed versions

So what do our analyses really prove?

Axiom: $a \in s.\text{remove}(b) \leftrightarrow a \neq_{el} b \wedge a \in s$

1. $isElement(a, s) \leftrightarrow a \in s$

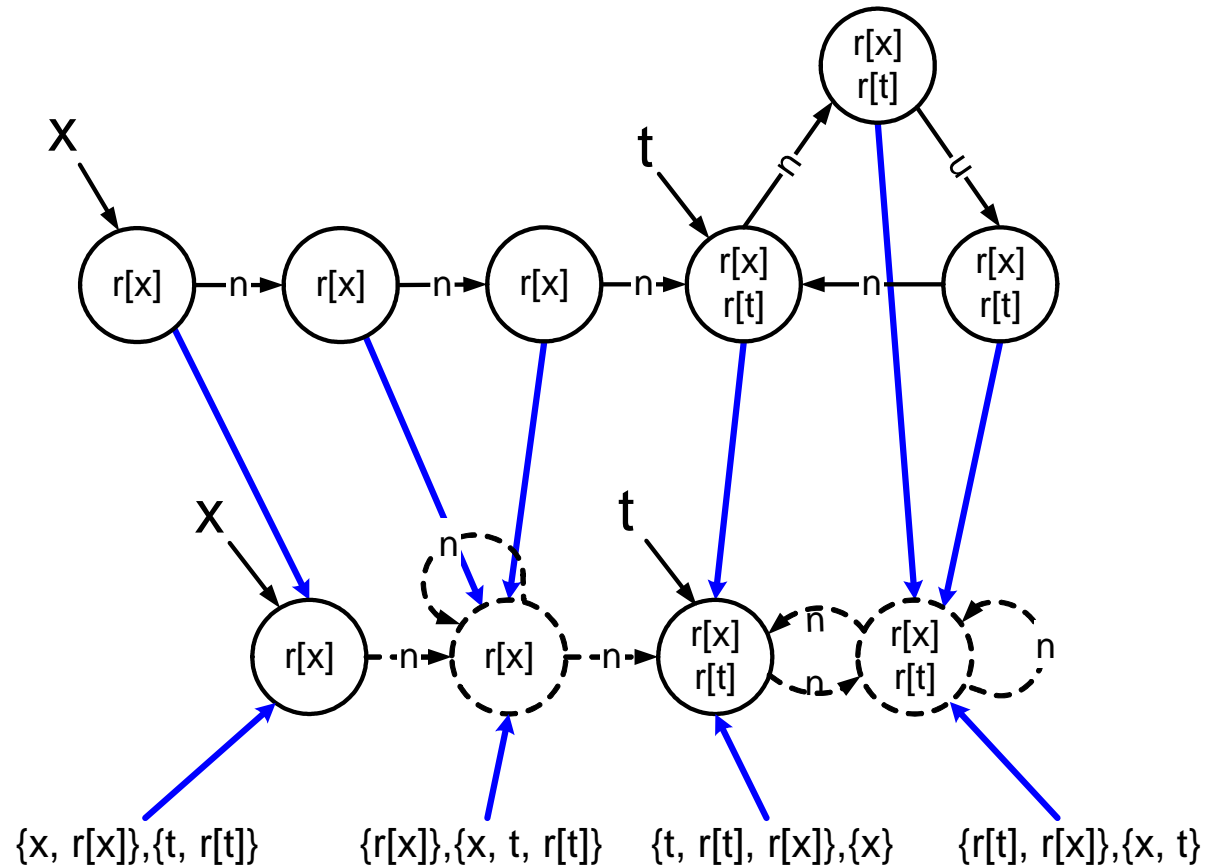
2. After executing $s.\text{remove}(b)$ we check
 $isElement(a, s) \leftrightarrow a \neq_{el} b \wedge isElementOld(a, s)$

Recent Developments in Shape Analysis

- very precise Shape Analysis algorithms
→ able to prove partial correctness of programs: bubble-sort, insertion-sort, etc.
(LARSW00)
- instantiations of a Parametric Shape Analysis Framework of (SRW02) that use logical structures to represent states
- has been implemented in a tool called TVLA (= Three-Valued-Logic Analyzer)

Canonical Abstraction

- Collapse individuals that agree on unary predicates.



- At most $3^{|U|}$ abstract individuals.

How to make analyses precise

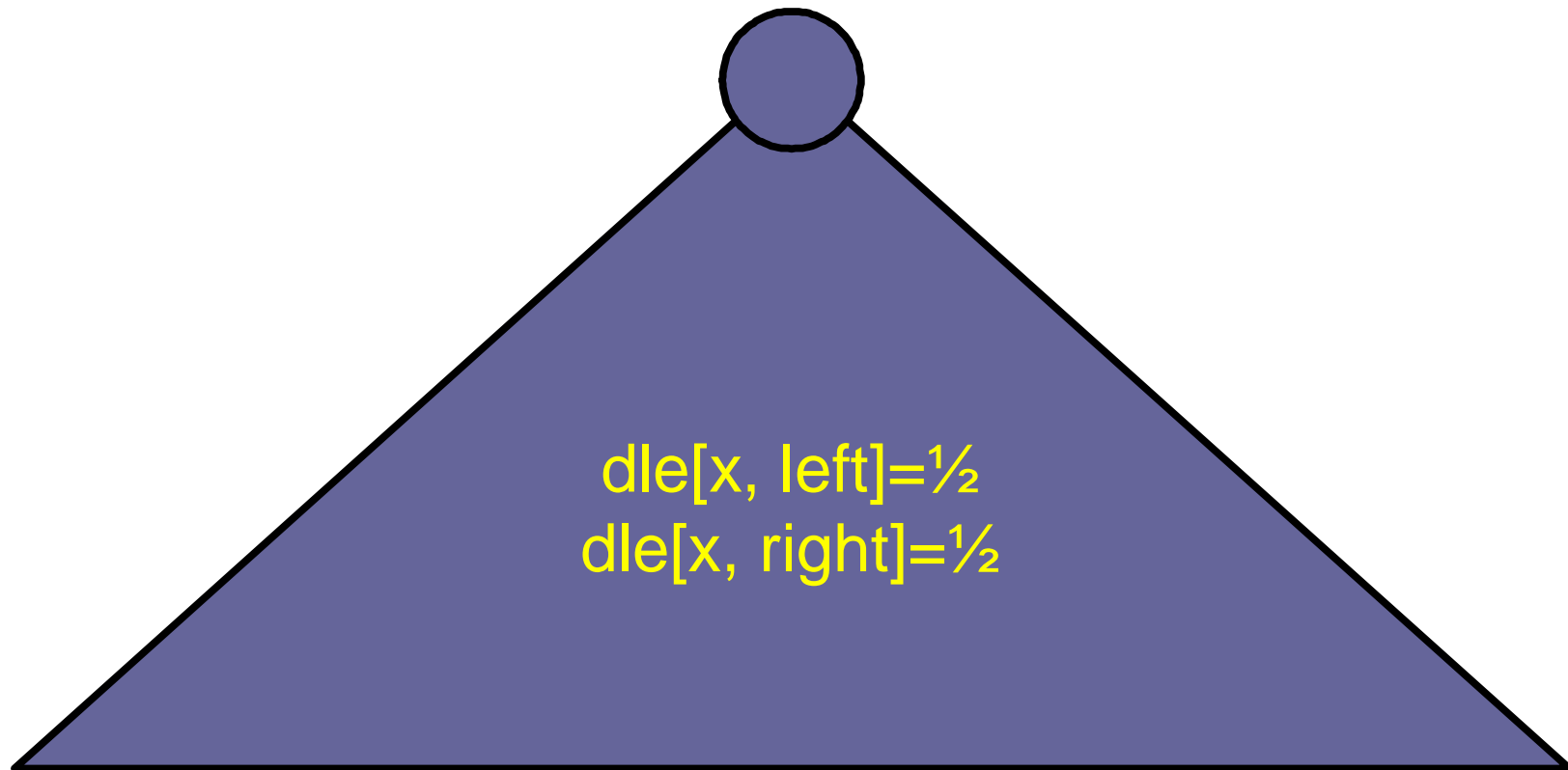
Key Predicates

- Model *data*-field indirectly
dle-predicate - “data less or equal”
stores value of *compare*-function
- Capture *dle*-relation with nodes pointed to by variables: $dle[variable, left]$ - and $dle[variable, right]$ -predicate family
- Keep precise reachability information through:
 $downStar[left], downStar[right]$

Predicates - $dle(\text{var}, \text{left})$ / $dle(\text{var}, \text{right})$

$$dle[x, \text{left}](v) = \exists v_1. (x(v_1) \wedge dle(v, v_1) \wedge \neg dle(v_1, v))$$

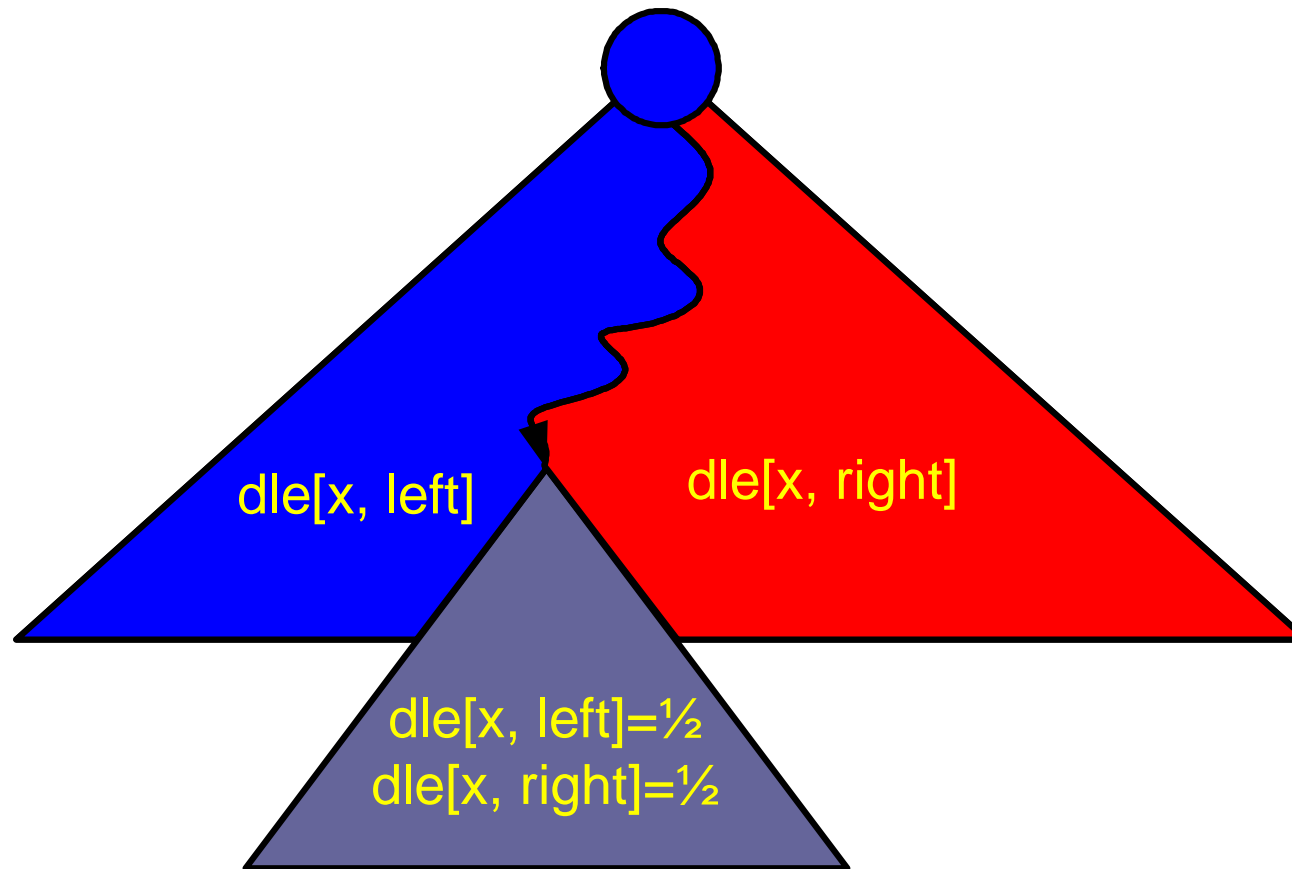
$$dle[x, \text{right}](v) = \exists v_1. (x(v_1) \wedge \neg dle(v, v_1) \wedge dle(v_1, v))$$



Predicates - $dle(\text{var}, \text{left})$ / $dle(\text{var}, \text{right})$

$$dle[x, \text{left}](v) = \exists v_1. (x(v_1) \wedge dle(v, v_1) \wedge \neg dle(v_1, v))$$

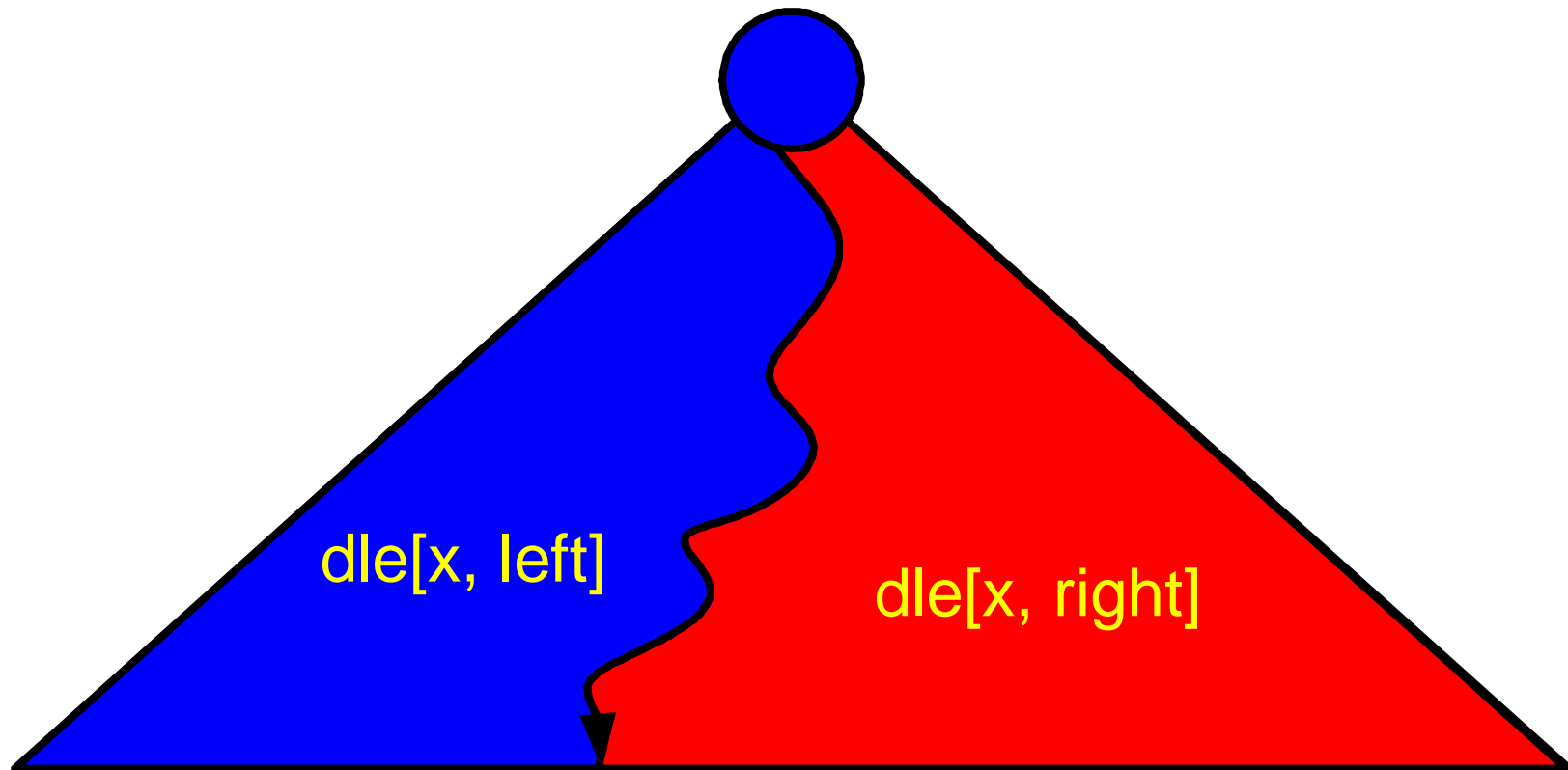
$$dle[x, \text{right}](v) = \exists v_1. (x(v_1) \wedge \neg dle(v, v_1) \wedge dle(v_1, v))$$



Predicates - $dle(\text{var}, \text{left})$ / $dle(\text{var}, \text{right})$

$$dle[x, \text{left}](v) = \exists v_1. (x(v_1) \wedge dle(v, v_1) \wedge \neg dle(v_1, v))$$

$$dle[x, \text{right}](v) = \exists v_1. (x(v_1) \wedge \neg dle(v, v_1) \wedge dle(v_1, v))$$

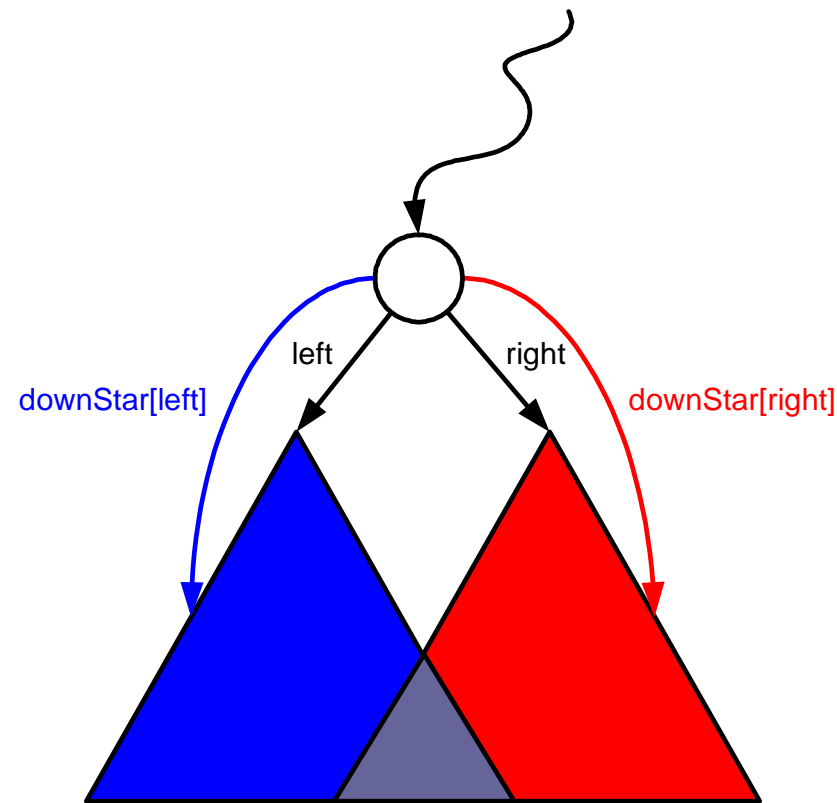


Predicates - downStar(left) / downStar(right)

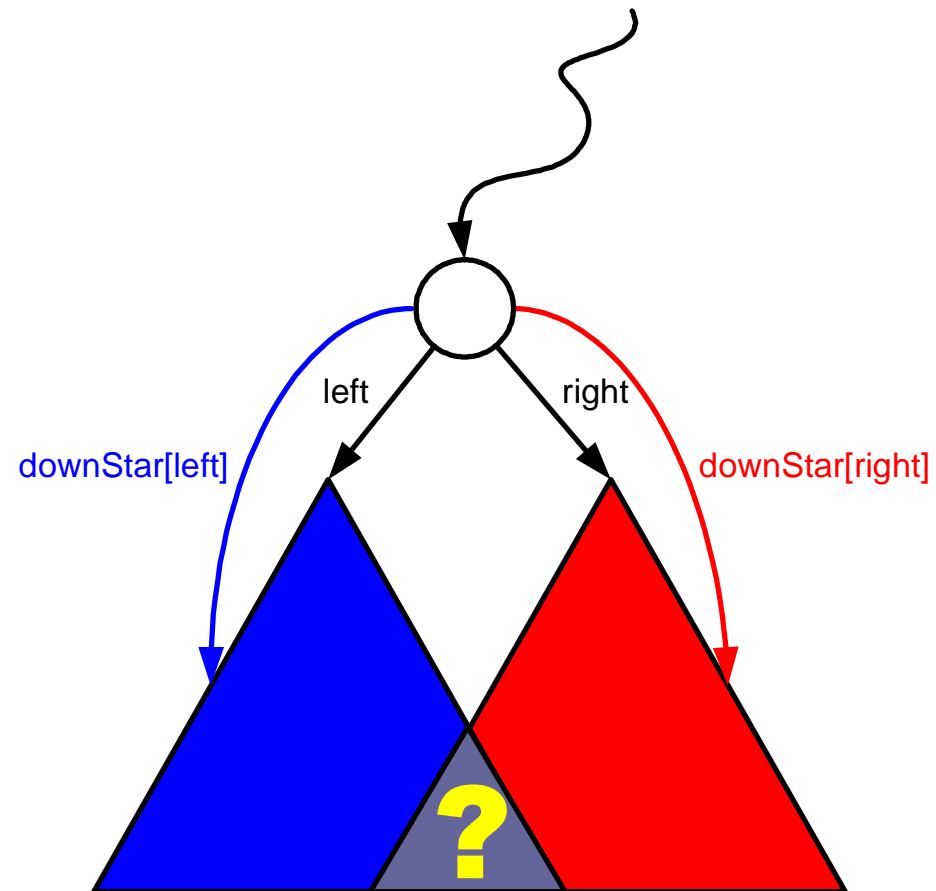
$$\text{down}(v_1, v_2) = \text{left}(v_1, v_2) \vee \text{right}(v_1, v_2)$$

$$\text{downStar}[\text{left}](v_1, v_2) = \exists v. \text{left}(v_1, v) \wedge \text{down}^*(v, v_2)$$

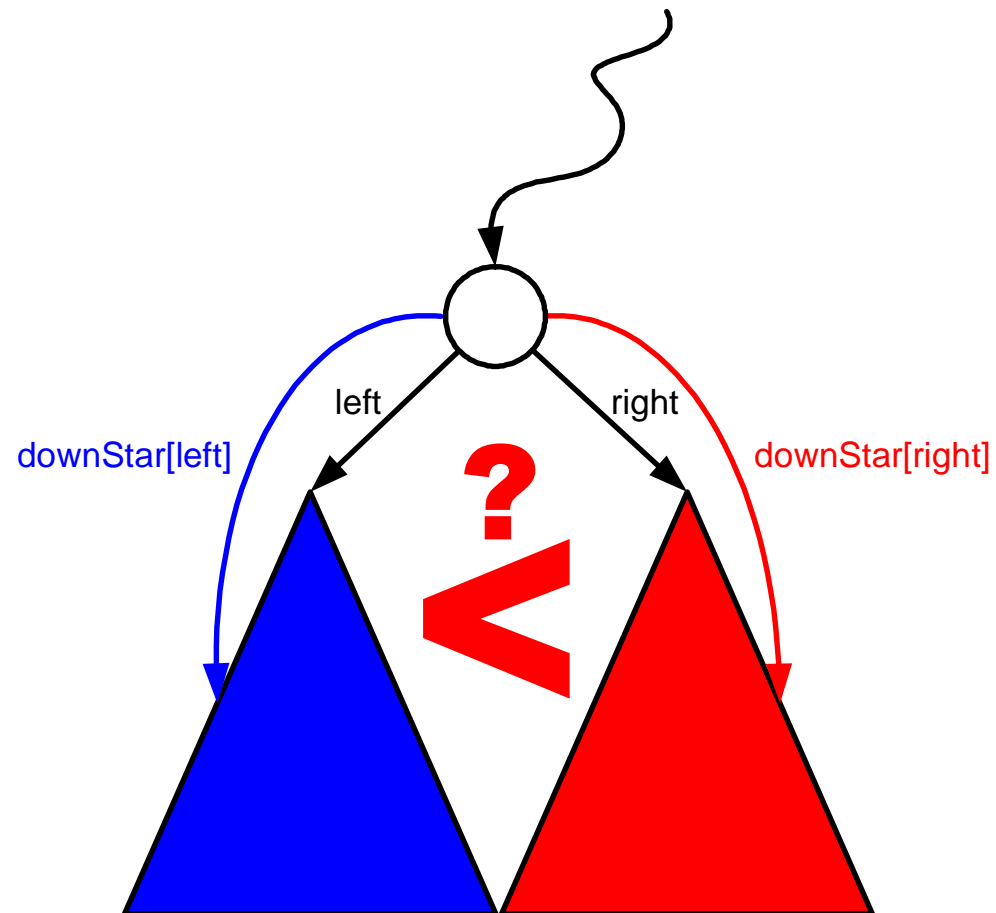
$$\text{downStar}[\text{right}](v_1, v_2) = \exists v. \text{right}(v_1, v) \wedge \text{down}^*(v, v_2)$$



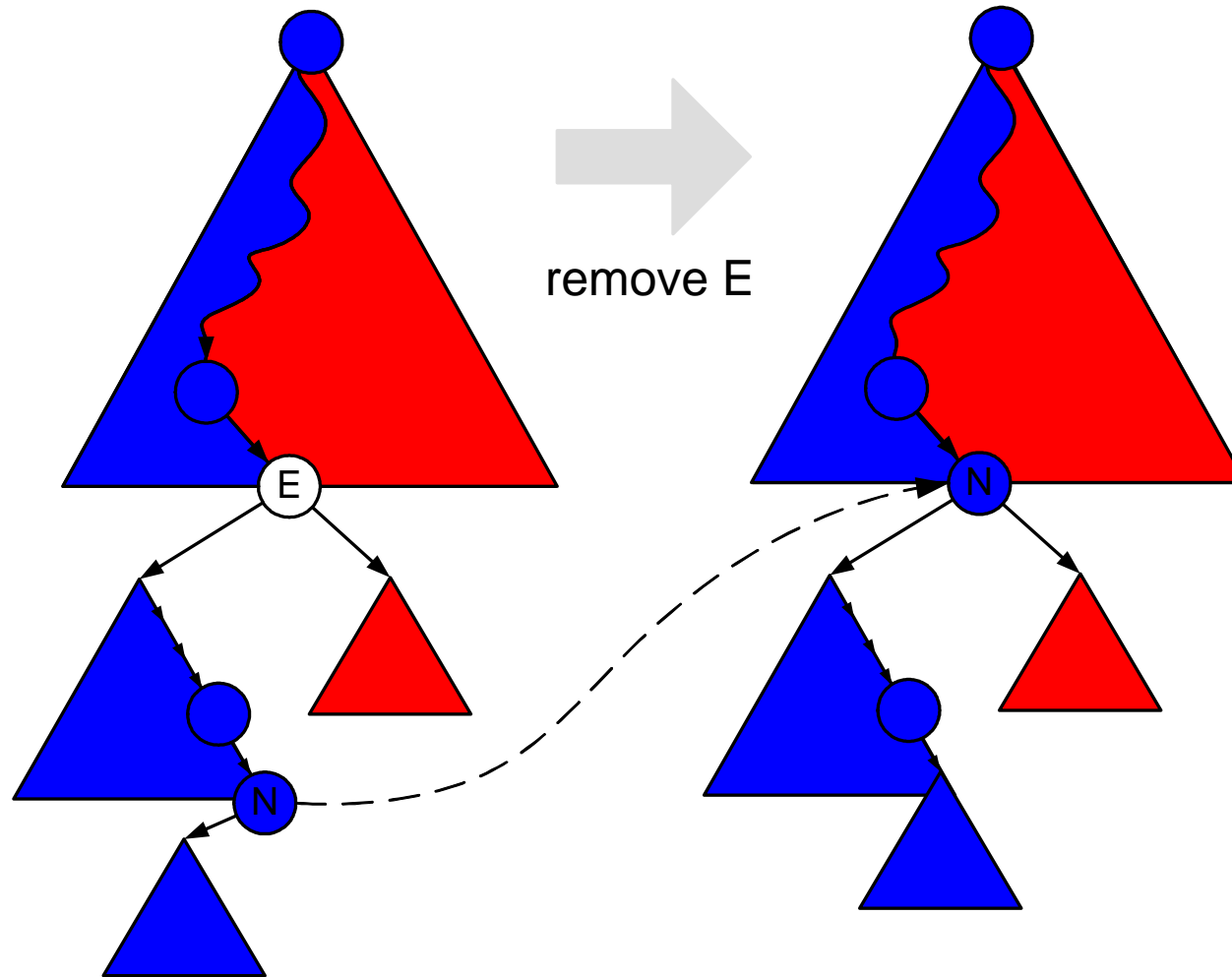
Data Structure Invariants - Is it a tree?



Data Structure Invariants - Is it ordered?



Combined Effect of Predicates



Summary

- Successfully analyzed complex heap-manipulating routines!
- From Axiom to Analysis:
 - Coupled different analyses by instrumentation predicates ($isElement(a, s) \leftrightarrow a \in s$)
 - Remembered old state of predicate to compare it with new state ($isElementOld$)
- Tailoring the abstraction specifically to the data structure was the key: Keeping important ordering and reachability information precise
→ one abstraction for all methods, no loop invariants

The Abstract Data Type Set ✓
A Tree-based Set Implementation ✓
Shape Analysis of the Implementation ✓

Thanks for your attention!

References

- (EM85) Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specification I*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1985.
- (LARSW00) Tal Lev-Ami, Thomas Reps, Mooly Sagiv, and Reinhard Wilhelm. Putting static analysis to work for verification: A case study. In *ISSTA '00: Proceedings of the 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 26–38, New York, NY, USA, 2000. ACM Press.
- (LEW97) Jacques Loeckx, Hans-Dieter Ehrich, and Markus Wolf. *Specification of abstract data types*. John Wiley & Sons, Inc., New York, NY, USA, 1997.
- (Rei05) Jan Reineke. Shape Analysis of Sets. Master's thesis, Saarland University, Saarbrücken, Germany, June 2005.

(SRW02) Mooly Sagiv, Thomas Reps, and Reinhard Wilhelm. Parametric shape analysis via 3-valued logic. *ACM Trans. Program. Lang. Syst.*, 24(3):217–298, 2002.