Reasoning about Pointer-based Data Structures

Scott McPeak
CS 294-3
10/21/04

What's so hard about pointers?

- Indirect access begs aliasing question
- Consider axiom for memory:
  - $i \neq j \Rightarrow sel(upd(M, i, v), j) = sel(M, j)$

need to be able to conclude *disequality*

otherwise, every write invalidates all previous info

i.e., strong update

Traditional Alias Analysis

- Each allocation site is an "object"
- Analysis finds, for each variable, the set of objects it might point to
  - conservative dataflow analysis
- This is too imprecise
  - cells of a data structure are typically conflated
  - each struct field is (usually) a single "variable"

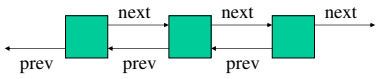
Language of Pointer Equalities

- Claim: we can adequately characterize data structures with sets of axioms of the form

$$\forall\ p.\ pABC = pRST$$

- A, B, etc. are unary function symbols
- pA means A(p); think p->A or p.A
- any # of symbols on each side (in principle)

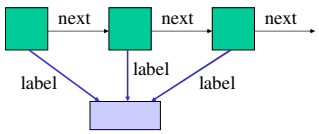
Example: Doubly-linked List



$$\forall\ p.\ p.next.prev = p$$

says that "next" is *injective*, as "prev" is its inverse

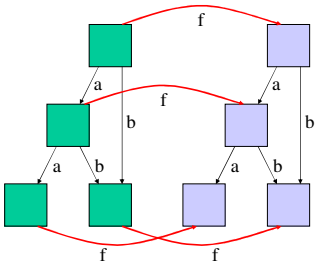
Example: Labeled List



$$\forall\ p.\ p.next.label = p.label$$

the label is *transitively* spread throughout the structure

Example: Grid



$\forall$ p. p.a.f = p.f.a
$\forall$ p. p.b.f = p.f.b

"f" is an isomorphism

7

Problem: Undecidable Theories

- The language includes axiom sets with undecidable theories (consequences)
- Consider a rule such as $\forall$ p. pABC = pRST
- Creates equivalence classes of strings:

xyzABCijk

||

xyzRSTijk

8

Turing Machine Encoding

Use following function symbols:

A,B	tape symbols
S_i	r/w head loc, state i
E	tape end (expandable)
C	accepting consumer

Example tape, in state i

$xAABABS_iBBABAE$

symbol being read

In state 4, when reading a B, write A, move left, go to 7.

$\forall$ p. $pAS_4B = pS_7AA$

In state 6, when reading a B, accept.

$\forall$ p. $pS_6B = pC$

Expand tape (adding A's)

$\forall$ p. pE = pAE

Collapse to just "C"

$\forall$ p. pCA = pC

9

To Prove a Theory Decidable

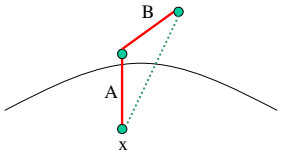
- Give matching rules (i.e. when to instantiate the axioms)
- Show matching/completion terminates
- Show all equalities among a given finite set of terms are discovered
  - EDAG expansion argument

10

Example: Injectivity

$\forall$ p. $\underbrace{pA}$B = p

match: pA



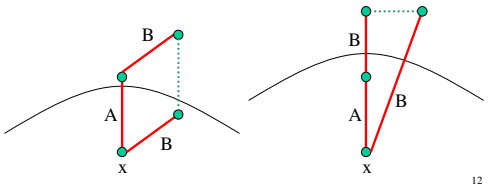
11

Example: Transitivity

match: pAB

$\forall$ p. $\overbrace{\underbrace{pA}B}$ = $\underbrace{pB}$

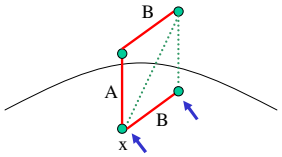
match: pA and pB



12

Counterexample: Inj+Trans

$$\left.\begin{array}{l}\forall\ p.\ pAB = p \\ \forall\ p.\ pAB = pB\end{array}\right\}\ \forall\ p.\ pB = p$$



13

Example: Grid

$$\forall\ p.\ pAB = pBA$$

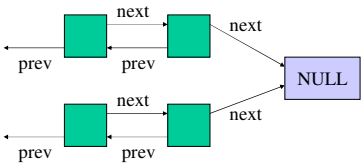
Is it complete? Exercise! Either:
- find a counterexample, or
- convince yourself it is complete

14

Extension to Conditionals

- Usually, we need special cases:

$$\forall\ p.\ pA \neq 0 \Rightarrow pAB = p$$



15

Matching with Conditionals

- In general:

$$\forall p. \quad \bigvee_i p\bar{A}_i = p\bar{B}_i \quad \vee \quad \bigvee_i p\bar{C}_i \neq p\bar{D}_i$$

- Claim: Matching rule is intersection of disjunct matching rules
- Proof: If a disjunct's rule is not satisfied, it could be true but irrelevant

16

A Short Proof

invariant $\forall p. \quad p\texttt{->next} \neq 0 \Rightarrow p\texttt{->next->prev} = p$

premise $h \neq 0 \wedge h\texttt{->prev} = 0$

¬ goal $h = h\texttt{->next}$

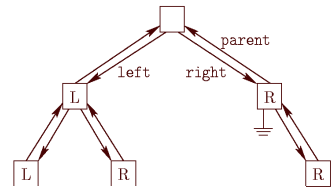
instantiate $h\texttt{->next} = 0 \ \vee \ h\texttt{->next->prev} = h$

split cases

$h = 0$ (contradiction)	$h\texttt{->prev} = h$
	$0 = h$ (contradiction)

17

Example: Binary Tree



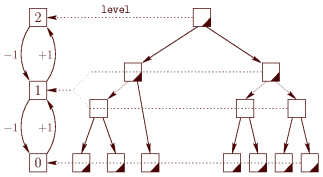
$\texttt{L} \neq \texttt{R}$

$\forall p. \ \ p \neq 0 \ \wedge \ p.\texttt{left} \neq 0 \implies p.\texttt{left.parent} = p \ \wedge \ p.\texttt{left.kind} = \texttt{L}$

$\forall p. \ \ p \neq 0 \ \wedge \ p.\texttt{left} \neq 0 \implies p.\texttt{left.parent} = p \ \wedge \ p.\texttt{right.kind} = \texttt{R}$

18

Example: Red-black Tree



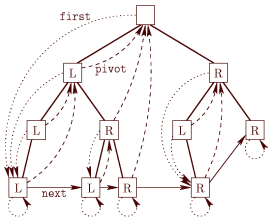
$\forall p.\ \ p \neq 0 \ \wedge\ p.\texttt{child} = 0 \implies p.\texttt{level} = 0$

$\forall p.\ \ p \neq 0 \ \wedge\ p.\texttt{child} \neq 0 \ \wedge\ p.\texttt{child.color} = \texttt{Red}$
$\implies p.\texttt{child.level} = p.\texttt{level}$

$\forall p.\ \ p \neq 0 \ \wedge\ p.\texttt{child} \neq 0 \ \wedge\ p.\texttt{child.color} = \texttt{Black}$
$\implies p.\texttt{child.level} + 1 = p.\texttt{level}$

19

Example: Linked Leaves



$\forall p.\ \ p \neq 0 \ \wedge\ p.\texttt{kind} = \texttt{R}$
$\implies p.\texttt{pivot} = p.\texttt{parent.pivot}$

$\forall p.\ \ p \neq 0 \ \wedge\ p.\texttt{kind} = \texttt{L} \ \wedge\ p.\texttt{parent.right} = 0$
$\implies p.\texttt{pivot} = p.\texttt{parent.pivot}$

$\forall p.\ \ p \neq 0 \ \wedge\ p.\texttt{kind} = \texttt{L} \ \wedge\ p.\texttt{parent.right} \neq 0$
$\implies p.\texttt{pivot} = p.\texttt{parent}$

20

Reasoning About Pointer-Based Data Structures (Part 2)

Scott McPeak
CS 294-3
10/26/2004

21

Outline

- First half: survey of approaches
  - Axiomatize transitive closure (Nelson)
  - Use graph grammars (Shape Types)
  - Abstract interpretation (TVLA)
  - Proving via semantic transformation (PALE)
- Second half: How PALE works

22

General Themes

- Need a language to *describe* data structures

- Need a reasoning engine for that language
- Key question: What is done with transitive closure?

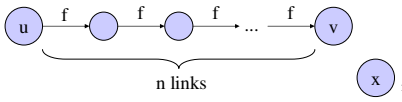
23

Nelson's Reachability (POPL 83)

- Everything revolves around one predicate:

$$\boxed{u \xrightarrow[x]{f} v}$$

$$\overset{\text{def}}{\iff} \quad \exists n.\ \ f^n(u) = v \ \wedge \ (\forall m < n.\ \ f^m(u) \neq x)$$



24

Derived Axioms

A1. $u \xrightarrow[x]{f} v \iff (u = v \vee (u \neq x \wedge f(u) \xrightarrow[x]{f} v))$

A2. $u \xrightarrow[x]{f} v \ \wedge \ v \xrightarrow[x]{f} w \implies u \xrightarrow[x]{f} w$

A3. $u \xrightarrow[x]{f} v \implies u \xrightarrow{f} v$

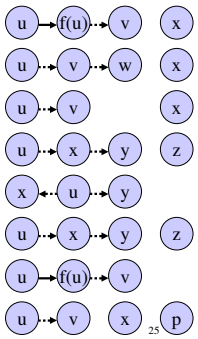
A4. $u \xrightarrow[y]{f} x \ \wedge \ u \xrightarrow[z]{f} y \implies u \xrightarrow[z]{f} x$

A5. $(u \xrightarrow{f} x \vee u \xrightarrow{f} y) \implies (u \xrightarrow[y]{f} x \vee u \xrightarrow[x]{f} y)$

A6. $u \xrightarrow[y]{f} x \ \wedge \ u \xrightarrow[z]{f} y \implies x \xrightarrow[z]{f} y$

A7. $f(u) \xrightarrow{f} v \iff f(u) \xrightarrow[u]{f} v$

A8. $u \xrightarrow[x,p]{f[p \mapsto q]} v \iff u \xrightarrow[x,p]{f} v$



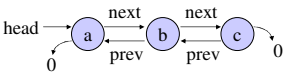
Complete?

- Sufficient to verify Union-find, where equivalence classes are circular lists
- Otherwise, little is known
- My experience: even if complete, hard to use automatically

Shape Types [Fradet/Métayer 97]

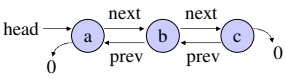
- Heap is set of objects, relations
  - unary relations: program variables
  - binary relations: pointers among objects
- Graph grammars constrain legal heaps
  - generative rules for building heaps

Example: Doubly-linked List



head = { a }
next = { (a,b), (b,c), (c,0) }
prev = { (a,0), (b,a), (c,b) }

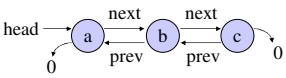
Example: Doubly-linked List



head = { a }
next = { (a,b), (b,c), (c,0) }
prev = { (a,0), (b,a), (c,b) }

Doubly → head x, prev x 0, Expand x
Expand x → next x y, prev y x, Expand y
Expand x → next x 0

Example: Doubly-linked List

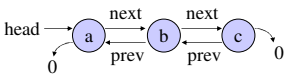


head = { a }
next = { (a,b), (b,c), (c,0) }
prev = { (a,0), (b,a), (c,b) }

Doubly → head x, prev x 0, Expand x
Expand x → next x y, prev y x, Expand y
Expand x → next x 0

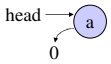
head = { }
next = { }
prev = { }
Doubly

Example: Doubly-linked List



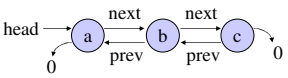
head = { a }
next = { (a,b), (b,c), (c,0) }
prev = { (a,0), (b,a), (c,b) }

Doubly → head x, prev x 0, Expand x
Expand x → next x y, prev y x, Expand y
Expand x → next x 0



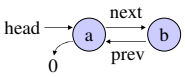
head = { a }
next = { }
prev = { (a,0) }
Expand a

Example: Doubly-linked List



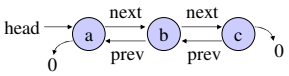
head = { a }
next = { (a,b), (b,c), (c,0) }
prev = { (a,0), (b,a), (c,b) }

Doubly → head x, prev x 0, Expand x
Expand x → next x y, prev y x, Expand y
Expand x → next x 0



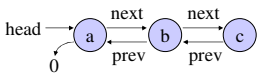
head = { a }
next = { (a,b) }
prev = { (a,0), (b,a) }
Expand b

Example: Doubly-linked List



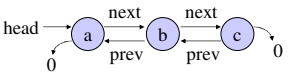
head = { a }
next = { (a,b), (b,c), (c,0) }
prev = { (a,0), (b,a), (c,b) }

Doubly → head x, prev x 0, Expand x
Expand x → next x y, prev y x, Expand y
Expand x → next x 0



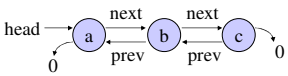
head = { a }
next = { (a,b), **(b,c)** }
prev = { (a,0), (b,a), **(c,b)** }
Expand c

Example: Doubly-linked List



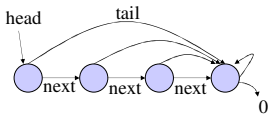
head = { a }
next = { (a,b), (b,c), (c,0) }
prev = { (a,0), (b,a), (c,b) }

Doubly → head x, prev x 0, Expand x
Expand x → next x y, prev y x, Expand y
Expand x → next x 0



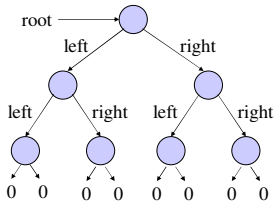
head = { a }
next = { (a,b), (b,c), **(c,0)** }
prev = { (a,0), (b,a), (c,b) }

Example: Tail-linked List



TailList → head x, tail x z, Expand x z
Expand x z → next x y, tail y z, Expand y z
Expand x z → next x z, tail z z

Example: Binary Tree



BinTree → root x, Expand x
Expand x → left x y, right x z, Expand y, Expand z
Expand x → left x 0, right x 0

Decision Problem

- Assuming the heap conformed to a given graph grammar, and a change is applied to that heap, does the new heap also conform?
  - Undecidable in general
- Authors suggest restricting the grammars to obtain decidability
  - Their claim: restrictions ok in practice

37

TVLA [Sagiv et al. 02]

- As in Shape Types, heap is objects/relations
- *Sets* of heaps denoted by letting binary relations have value 1/2, "unknown"
  - Formulas simply evaluated (conservatively)
- Shape characterized by additional "instrumentation predicates"
  - Evolution across updates still an issue

38

PALE [Møller/Schwartzbach 01]

- Data structures described via "graph types"
  - Certain fields for a backbone tree
  - Other fields' values determined by the backbone, with routing expressions
- Graph types translated into WS*k*S, weak second-order logic of *k* successors
- WS*k*S formulas decided by reduction to automata

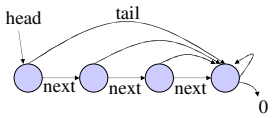
39

Graph Types

- Backbone tree
  - simply identify the fields and roots involved; it is implicit that they must form a tree
- Routing expressions
  - constrain non-tree pointers by (in essence) writing a little navigation program
  - actual notation resembles regular expressions

40

Example: Tail-linked List



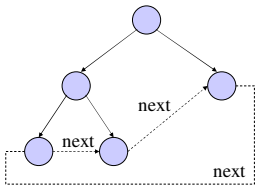
Tree fields: head, next

Routing expression constraint for tail:

$\forall\, p \neq 0.\quad p\ <\ \downarrow next^{*}\,.\,[\lambda x.\ x.next=null]\ >\ p.tail$

41

Example: Tree w/ Linked Leaves



^ = [λx. isroot(x)]
$ = [λx. isleaf(x)]

$\forall\, p \neq 0.$

$p\ <\ \uparrow right^{*}\,.\,(\uparrow left\ \downarrow right \mid \wedge\,)\,.\,\downarrow left^{*}\,.\,\$\ >\ p.next$

42

break

43

WS1S

Weak monadic second order logic of one successor

$\neg\phi$
$\phi_1 \wedge \phi_2$
$\exists X_i.\ \phi$

$X_i \subseteq X_j$
$X_i = X_j \backslash X_k$
$X_i = X_j + 1$

44

WS1S

Weak monadic second order theory of one successor

$\Psi : \{X_i\} \rightarrow \{M \mid M \text{ finite} \subseteq \mathbb{N}\}$

$$
\begin{array}{lcl}
\Psi \models \neg\phi & \iff & \Psi \not\models \phi \\
\Psi \models \phi_1 \wedge \phi_2 & \iff & \Psi \models \phi_1 \text{ and } \Psi \models \phi_2 \\
\Psi \models \exists X_i.\ \phi & \iff & \exists \text{ finite } M \subseteq \mathbb{N}. \\
 & & \quad \Psi[X_i \mapsto M] \models \phi \\
\Psi \models X_i \subseteq X_j & \iff & \Psi(X_i) \subseteq \Psi(X_j) \\
\Psi \models X_i = X_j \backslash X_k & \iff & \Psi(X_i) = \Psi(X_j) \backslash \Psi(X_k) \\
\Psi \models X_i = X_j + 1 & \iff & \Psi(X_i) = \{k+1 \mid k \in \Psi(X_j)\}
\end{array}
$$

45

Definable Concepts in WS1S

- isEmpty(X) $\Leftrightarrow \forall$ Y. X $\subseteq$ Y
- isEqual(X,Y) $\Leftrightarrow$ X $\subseteq$ Y $\wedge$ Y $\subseteq$ X
- isSingleton(X) $\Leftrightarrow$
  X $\neq \emptyset$ $\wedge$ $\forall$ Y. Y $\subseteq$ X $\Rightarrow$ Y=X $\vee$ Y=$\emptyset$
- isZero(X) $\Leftrightarrow$
  isSingleton(X) $\wedge$ $\neg \exists$ Y. X = Y+1

46

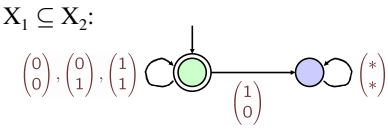
Interpretation Encoding

- Idea: Use bit strings to encode finite sets
  - e.g. 010110 = { 1, 3, 4 }
- Encode several sets with bit vector strings
  - e.g. $\binom{0}{0}\binom{1}{0}\binom{0}{1}\binom{1}{1}\binom{0}{0}$
    means X_1 = { 1, 3 } and X_2 = { 2, 3}
- Note: Not a canonical encoding (trailing 0s)

47

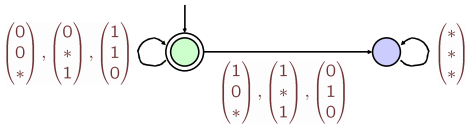
Evaluate Using Automata

- Given a formula ϕ, want to create an automaton M(ϕ) s.t.
  $\Psi \models \phi \quad \Leftrightarrow \quad$ M(ϕ) recognizes Ψ_{string}
- $X_1 \subseteq X_2$:



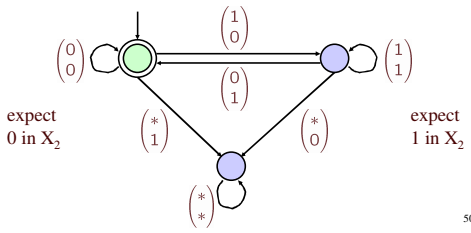
48

$X_1 = X_2 \setminus X_3$



$X_2 = X_1 + 1$

example input: $\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}$



Propositional Connectives

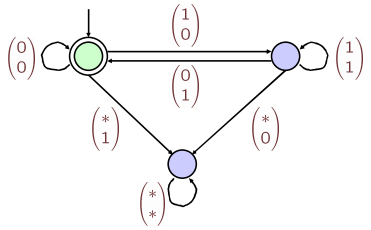
- $\phi = \neg\, \theta$
  - Let M be automaton for θ
  - Make sure M is complete, deterministic
  - Simply invert accepting/non-accepting states
- $\phi = \phi_1 \wedge \phi_2$
  - Construct product automaton
  - Accept wherever both do

$\exists\, X_i.\ \phi$

- Let M be automaton for ϕ
- Change all ith vector components to *
  - i.e. remove sensitivity to X_i
  - result is a nondeterministic automaton
- Let state s be accepting if it can *reach* an accepting state via $\begin{pmatrix}0\\0\end{pmatrix}$ transitions

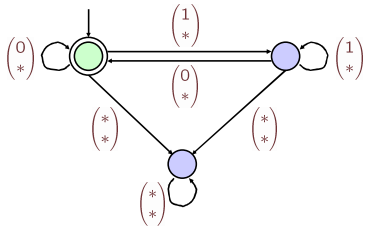
$\exists\, X_2.\ X_2 = X_1 + 1$

Original: $X_2 = X_1 + 1$



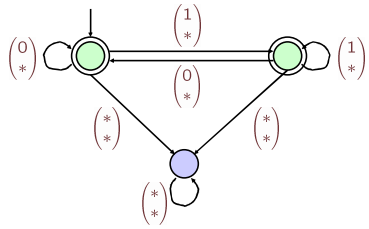
$\exists\, X_2.\ X_2 = X_1 + 1$

Change references to X_2 to *



$\exists X_2.\ X_2 = X_1 + 1$

Accept if can reach accepting via $\binom{0}{0}$ transitions



55

Automaton-Logic Connection

- Conversion to automata give effective procedure for deciding formulas:
  - Convert formula to automaton
  - ϕ is valid iff $M(\phi)$ can only reach accepting states
  - ϕ is satisfiable iff $M(\phi)$ has a reachable accepting state

56

Is this magic?

- Original semantics

$$\Psi \models \exists X_i.\ \phi \iff \exists \text{ finite } M \subseteq \mathbb{N}.\ \Psi[X_i \mapsto M] \models \phi$$

- Not obviously decidable---so how did the automaton do it?
- Automaton is a measure of formula complexity; compare to counting "+1"s

57

Complexity

- Consider formula: $\underbrace{\exists \neg \exists \neg \ldots \exists \neg}_{n} \phi$
- Each use of a quantifier introduces nondeterminism
- Each use of negation requires determinization (powerset construction)
- Cost: $O(\left.2^{2^{2^{\cdot^{\cdot^{\cdot^{2^c}}}}}}\right\}^n)$, $c = |M(\phi)|$

58

WS2S

- Two successor functions instead of one

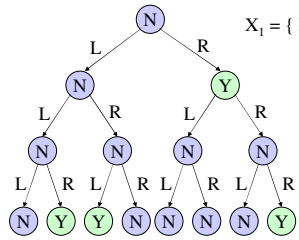
$$\Psi : \{X_i\} \to \{M \mid M \text{ finite} \subseteq \{L, R\}^*\}$$

$$\Psi \models X_i = X_j.L \iff \Psi(X_i) = \{k.L \mid k \in \Psi(X_j)\}$$
$$\Psi \models X_i = X_j.R \iff \Psi(X_i) = \{k.R \mid k \in \Psi(X_j)\}$$

59

Interpretation Encoding (WS2S)

- Idea: represent sets of strings using trees



60

Evaluate With Tree Automata

- Bottom-up tree automaton:
  - Finite set of states Q
  - Accepting set of states $Q_F \subseteq Q$
  - Transition relation, elements of form
    $f(q_1, q_2, ..., q_n) \to q$
    where f is a tree constructor of arity n
- Example:

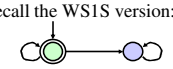
nil → 1	N(1,2) → 3 (accept)
N(1,1) → 1	N(2,2) → 3 (accept)
Y(1,1) → 2	else → 4

61

$X_1 \subseteq X_2$ (WS2S)

nil	→ 1	$\binom{Y}{N}$(_,_)	→ 2
$\binom{N}{N}$(1,1)	→ 1	$\binom{*}{*}$(2,_)	→ 2
$\binom{N}{Y}$(1,1)	→ 1	$\binom{*}{*}$(_,2)	→ 2 (rejecting)
$\binom{Y}{Y}$(1,1)	→ 1 (accepting)		

recall the WS1S version:



62

Remaining Cases

- Negation: determinize, invert
- Conjunction: product, intersection
- Set difference: easy, like subset
- Successor application: easy, like +1
- Existential
  - Nondeterminize the quantified track
  - Any state reachable via $\binom{N}{N}$ prefix is initial

63

Graph Types → WS*k*S

- Backbone: p.left corresponds to {p}.L
- Routing expressions: X is p.left* iff
  - $p \in X$
  - $y \in X \ \wedge \ y \neq 0 \ \Rightarrow \ y.\text{left} \in X$

  } contains left*
  - $\forall$ Y. ...(above)..Y.. $\Rightarrow$ $X \subseteq Y$

  } smallest such

(the above is actually just a guess)

64

Integration into Nelson-Oppen?

- Can test for unsatisfiability? Yes.
- Can emit implied equalities? Only(?) by making separate queries $H \Rightarrow x = y$
- Non-convex theory; can emit case splits? I do not see how.
- Decision algorithm is powerful but opaque

65

Conclusion

- Data structure description language:
  - powerful for auxilliaries; has transitive closure
  - weak for backbone: must be a tree
- Decision algorithm:
  - entirely based on semantics (no proof system)
  - clever use of automata
  - potentially very bad performance
  - unclear how to integrate to share results

66

References

- Nelson Reachability
  - Greg Nelson
    Verifying reachability invariants of linked structures
    POPL 83
- Shape Types
  - Pascal Fradet, Daniel Le Métayer
    Shape Types
    POPL 97
- TVLA
  - Shmuel Sagiv, Thomas W. Reps, Reinhard Wilhelm
    Parametric shape analysis via 3-valued logic
    TOPLAS 2002

67

References (part 2)

- PALE
  - Anders Møller, Michael Schwartzbach
    The Pointer Assertion Logic Engine
    PLDI 01
  - Nils Klarlund, Michael Schwartzbach
    Graph Types
    POPL 93
  - Nils Klarlund, Anders Møller, Michael Schwartzbach
    MONA Implementation Secrets
    IJFCS 00 (I got my WSkS material from here)
  - Nils Klarlund
    Mona and Fido: The Logic-Automaton Connection in Practice
    CSL 97 (Another WSkS treatment, a little less clear)

68