Synthesizing Shape Predicate via Second-Order Bi-Abduction

Quang-Loc Le, Cristian Gherghina, Shengchao Qin and Wei-Ngan Chin

NUS National University of Singapore
School of Computing

Website

http://loris-7.ddns.comp.nus.edu.sg/~project/s2/beta/

Motivation

- Shape analysis is crucial for proving memory safety and is a precursor to supporting functional correctness on heap manipulating programs.
- Inferring shapes describing abstractions for data structures used by each method is challenging to automatic program verifiers.
- Shape synthesis for arbitrary data structures to support recursive procedures is even more challenging.

Main Ideas

- Grounded on abduction rather than deduction.
- Distinguish unknown pre-predicates in pre-conditions, from unknown post-predicates in post-condition; since the former may be strengthened, while the latter may be weakened.
- Support *heap guard* mechanism for more precise preconditions on heap specification.
- Normalize inferred shape predicates to facilitate the reuse of existing predicates.

Second-Order Bi-Abductive Entailment

Given an antecedent $\Delta_{\texttt{ante}}$ and a consequent $\Delta_{\texttt{conseq}}$, the second-order bi-abductive entailment checker constructs both the frame residue $\Delta_{\texttt{frame}}$ and a set of relational assumptions of the form $\mathcal{R} = \bigwedge_{i=1}^{n}(\Delta^{i}_{\texttt{lhs}} \Rightarrow \Phi^{gi}_{\texttt{rhs}})$ such that:

$$\Delta_{\texttt{ante}} \vdash \Delta_{\texttt{conseq}} \rightsquigarrow (\mathcal{R}, \Delta_{\texttt{frame}})$$

Semantics:

$$\mathcal{R} \wedge \Delta_{\texttt{ante}} \vDash \Delta_{\texttt{conseq}} * \Delta_{\texttt{frame}}$$

Example: H is an unknown predicate and $\texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{t})$ is a consequent that ensures x points to an allocated heap, as required for field access.

$$\texttt{H}(\texttt{x},\texttt{p},\texttt{t}) * \Delta \vdash \texttt{x}\mapsto\texttt{tree}(\texttt{x}_\texttt{p},\texttt{l},\texttt{r},\texttt{n}) \rightsquigarrow (\mathcal{R}_1, \Delta_1)$$

$$\mathcal{R}_1 \equiv \texttt{H}(\texttt{x},\texttt{p},\texttt{t}) \Rightarrow \texttt{x}\mapsto\texttt{tree}(\texttt{x}_\texttt{p},\texttt{l},\texttt{r},\texttt{n}) * \texttt{H}_\texttt{p}(\texttt{x}_\texttt{p},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{r}(\texttt{r},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{n}(\texttt{n},\texttt{p},\texttt{t})$$

$$\Delta_1 \equiv \Delta * \texttt{H}_\texttt{p}(\texttt{x}_\texttt{p},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{r}(\texttt{r},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{n}(\texttt{n},\texttt{p},\texttt{t})$$

Hoare Rules

$$\vdash \{\Delta_{\texttt{pre}}\}\ \texttt{e}\ \{\mathcal{R}, \Delta_{\texttt{post}}\}$$

where $\mathcal{R}$ accumulates the set of relational assumptions generated by the entailment procedure.

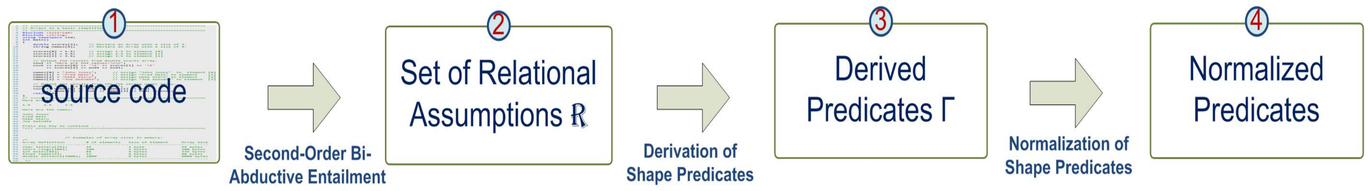
Derivation of Shape Predicates

function PRED_SYN($\mathcal{R}$)
  $\Gamma \leftarrow \emptyset$
  $\mathcal{R} \leftarrow$ exhaustively apply [Syn-Base] on $\mathcal{R}$
  $\mathcal{R}_{pre}, \mathcal{R}_{post} \leftarrow$ *sort-group*($\mathcal{R}$)
  while $\mathcal{R}_{pre} \neq \emptyset$ **do**
    $\texttt{U}^{\texttt{pre}}, \sigma \leftarrow$ pick unknown and select related assumptions in $\mathcal{R}_{pre}$
    $\texttt{U}^{\texttt{pre}}_{\texttt{def}} \leftarrow$ apply [Syn-Case], [Syn-Group-Pre] and [Syn-Pre-Def] on σ
    $\mathcal{R}_{pre}, \mathcal{R}_{post} \leftarrow$ inline $\texttt{U}^{\texttt{pre}}_{\texttt{def}}$ in $(\mathcal{R}_{pre} \setminus \sigma)$ and $\mathcal{R}_{post}$
    discharge $\texttt{U}^{\texttt{pre}}$ obligations
    $\Gamma \leftarrow \Gamma \cup \{\texttt{U}^{\texttt{pre}}_{\texttt{def}}\}$
  end while
  while $\mathcal{R}_{post} \neq \emptyset$ **do**
    $\texttt{U}^{\texttt{post}}, \sigma \leftarrow$ pick unknown and select related assumptions in $\mathcal{R}_{post}$
    $\texttt{U}^{\texttt{post}}_{\texttt{def}} \leftarrow$ apply [Syn-Group-Post], [Syn-Post-Def] on σ
    discharge $\texttt{U}^{\texttt{post}}$ obligations
    $\mathcal{R}_{post} \leftarrow \mathcal{R}_{post} \setminus \sigma$ $\Gamma \leftarrow \Gamma \cup \{\texttt{U}^{\texttt{post}}_{\texttt{def}}\}$
  end while
  return Γ
end function

Normalization of Shape Predicates

function PRED_NORM(Γ)
  $\Gamma_1 \leftarrow$ process-dangling-and-unused-preds Γ
  $\Gamma_2 \leftarrow$ eliminate-useless-parameters Γ_1
  $\Gamma_3 \leftarrow$ perform-predicate splitting on Γ_2
  return reuse-predicates Γ_3
end function

The Proposed Method

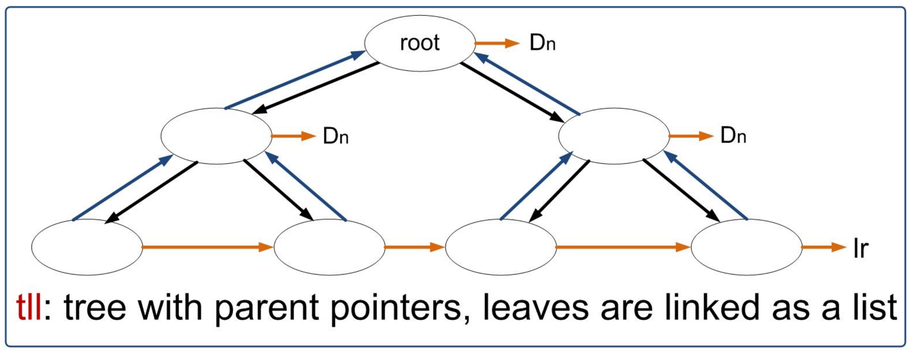


Motivating Example

struct tree { struct tree* parent; struct tree* l; struct tree* r; struct tree* next}
requires H(x,p,t) ensures G(x,p,res,t)

(1)

```
struct tree* tll(struct tree* x, struct tree* p, struct tree* t)
{
//S1(PRE) : H(x,p,t)
  x->parent = p;
//S2(BIND) : x↦tree(p,l,r,n)*Hl(l,p,t)*Hr(r,p,t)*Hn(n,p,t)*Hp(xp,p,t)
  if (x->r=NULL) {
//S3(THEN) : x↦tree(p,l,r,n)*Hl(l,p,t)*Hr(r,p,t)*Hn(n,p,t)∧r=NULL
    x->next=t; return x; }
//S6(POST1) : x↦tree(p,l,r,t)*Hl(l,p,t)*Hr(r,p,t)∧r=NULL∧res=x
  else{
//S4(ELSE) : x↦tree(p,l,r,n)*Hl(l,p,t)*Hr(r,p,t)*Hn(n,p,t)∧r≠NULL
    struct tree* lm;
//(REC−PRE−1) :
    lm = tll(x->r, x, t);
//S5(REC−PRE−2) : x↦tree(p,l,r,n)*Hl(l,p,t)*G(r,x,lm,t)*Hn(n,p,t)∧r≠NULL
    return tll(x->l, x, lm); }
//S7(POST2) : x↦tree(p,l,r,n)*G(l,x,res,lm)*G(r,x,lm,t)*Hn(n,p,t)∧r≠NULL
}
```



tll: tree with parent pointers, leaves are linked as a list

(2)

	Relational Assumption	Remark	Entailment
(1).	$\texttt{H}(\texttt{x},\texttt{p},\texttt{t}) \Rightarrow \texttt{x}\mapsto\texttt{tree}(\texttt{x}_\texttt{p},\texttt{l},\texttt{r},\texttt{n}) * \texttt{H}_\texttt{p}(\texttt{x}_\texttt{p},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{r}(\texttt{r},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{n}(\texttt{n},\texttt{p},\texttt{t})$	BIND	S1 ⊢ $\texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n})$
(2).	$\texttt{H}_\texttt{r}(\texttt{r},\texttt{p},\texttt{t}) \wedge \texttt{r}\neq\texttt{NULL} \Rightarrow \texttt{H}(\texttt{r},\texttt{x},\texttt{t})\ @\ \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n})$	PRE−REC−1	S4 ⊢ $\texttt{H}(\texttt{r},\texttt{p},\texttt{t})$
(3).	$\texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) \Rightarrow \texttt{H}(\texttt{l},\texttt{x},\texttt{lm})\ @\ (\texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n}) \wedge \texttt{r}\neq\texttt{NULL})$	PRE−REC−2	S5 ⊢ $\texttt{H}(\texttt{l},\texttt{p},\texttt{lm})$
(4).	$\texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{r}(\texttt{r},\texttt{p},\texttt{t}) * \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{t}) \wedge \texttt{r}=\texttt{NULL} \wedge \texttt{res}=\texttt{x} \Rightarrow \texttt{G}(\texttt{x},\texttt{p},\texttt{res},\texttt{t})$	POST−1	S6 ⊢ $\texttt{G}(\texttt{x},\texttt{p},\texttt{res},\texttt{t})$
(5).	$\texttt{H}_\texttt{n}(\texttt{n},\texttt{p},\texttt{t}) * \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n}) * \texttt{G}(\texttt{r},\texttt{x},\texttt{lm},\texttt{t}) * \texttt{G}(\texttt{l},\texttt{x},\texttt{res},\texttt{lm}) \wedge \texttt{r}\neq\texttt{NULL} \Rightarrow \texttt{G}(\texttt{x},\texttt{p},\texttt{res},\texttt{t})$	POST−2	S7 ⊢ $\texttt{G}(\texttt{x},\texttt{p},\texttt{res},\texttt{t})$

(3)

Pre-Predicate Derivation

Apply [Syn-Base] on assumption (4)

(4a) $\texttt{res}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{t}) * \texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) \wedge \texttt{r}=\texttt{NULL} \wedge \texttt{res}=\texttt{x} \Rightarrow \texttt{G}(\texttt{x},\texttt{p},\texttt{res},\texttt{t})$
(4b) $\texttt{H}_\texttt{r}(\texttt{r},\texttt{p},\texttt{t}) \wedge \texttt{r}=\texttt{NULL} \Rightarrow \texttt{emp}$
(4c) $\texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) \Rightarrow \texttt{H}^{\texttt{f}}_\texttt{1}(\texttt{l},\texttt{p},\texttt{t})\ @\ (\texttt{res}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{t}) \wedge \texttt{r}=\texttt{NULL})$
(4d) $\texttt{H}^{\texttt{f}}_\texttt{1}(\texttt{l},\texttt{p},\texttt{t}) \Rightarrow \top$

Apply [Syn-Case], [Syn-Group-Pre], [Syn-Pre-Def]

Synthesized Predicates	Assumptions
$\texttt{H}(\texttt{x},\texttt{p},\texttt{t}) \equiv \texttt{x}\mapsto\texttt{tree}(\texttt{x}_\texttt{p},\texttt{l},\texttt{r},\texttt{n}) * \texttt{H}_\texttt{p}(\texttt{x}_\texttt{p},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{r}(\texttt{r},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{n}(\texttt{n},\texttt{p},\texttt{t})$	(1)
$\texttt{H}_\texttt{r}(\texttt{r},\texttt{p},\texttt{t}) \equiv \texttt{H}(\texttt{r},\texttt{x},\texttt{t}) \wedge \texttt{r}\neq\texttt{NULL}\ @\ \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n}) \vee \texttt{emp} \wedge \texttt{r}=\texttt{NULL}$	(2), (4b)
$\texttt{H}_\texttt{l}(\texttt{l},\texttt{p},\texttt{t}) \equiv \texttt{H}(\texttt{l},\texttt{x},\texttt{lm})\ @\ \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n}) \wedge \texttt{r}\neq\texttt{NULL}$ $\vee\ \texttt{H}^{\texttt{f}}_\texttt{1}(\texttt{l},\texttt{p},\texttt{t})\ @\ \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n}) \wedge \texttt{r}=\texttt{NULL}$	(3), (4c)
$\texttt{H}^{\texttt{f}}_\texttt{1}(\texttt{l},\texttt{p},\texttt{t}) \equiv \top$	(4d)

Apply [Syn-Inline]

$\texttt{H}(\texttt{x},\texttt{p},\texttt{t}) \equiv \texttt{x}\mapsto\texttt{tree}(\texttt{x}_\texttt{p},\texttt{l},\texttt{r},\texttt{x}_\texttt{n}) * \texttt{H}(\texttt{l},\texttt{x},\texttt{lm}) * \texttt{H}(\texttt{r},\texttt{x},\texttt{t}) * \texttt{H}_\texttt{p}(\texttt{x}_\texttt{p},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{n}(\texttt{x}_\texttt{n},\texttt{p},\texttt{t}) \wedge \texttt{r}\neq\texttt{NULL}$
$\vee\ \texttt{x}\mapsto\texttt{tree}(\texttt{x}_\texttt{p},\texttt{l},\texttt{r},\texttt{x}_\texttt{n}) * \texttt{H}^{\texttt{f}}_\texttt{1}(\texttt{l},\texttt{x},\texttt{t}) * \texttt{H}_\texttt{p}(\texttt{x}_\texttt{p},\texttt{p},\texttt{t}) * \texttt{H}_\texttt{n}(\texttt{x}_\texttt{n},\texttt{p},\texttt{t}) \wedge \texttt{r}=\texttt{NULL}$

Post-Predicate Derivation

Apply [Syn-Group-Post] and [Syn-Post-Def]

Synthesized Predicates	Assumptions
$\texttt{G}(\texttt{x},\texttt{p},\texttt{res},\texttt{t}) \equiv \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{t}) * \texttt{H}_\texttt{1}(\texttt{l},\texttt{x}_\texttt{h},\texttt{t}_\texttt{h}) \wedge \texttt{res}=\texttt{x} \wedge \texttt{r}=\texttt{NULL}$ $\vee\ \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n}) * \texttt{G}(\texttt{r},\texttt{x},\texttt{l}_\texttt{m},\texttt{t}) * \texttt{G}(\texttt{l},\texttt{x},\texttt{res},\texttt{l}_\texttt{m}) * \texttt{H}_\texttt{n}(\texttt{n},\texttt{x}_\texttt{h},\texttt{t}_\texttt{h}) \wedge \texttt{r}\neq\texttt{NULL}$	(4a), (5)

Apply [Syn-Inline]

$\texttt{G}(\texttt{x},\texttt{p},\texttt{res},\texttt{t}) \equiv \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{t}) * \texttt{H}^{\texttt{f}}_\texttt{1}(\texttt{l},\texttt{x}_\texttt{h},\texttt{t}_\texttt{h}) \wedge \texttt{res}=\texttt{x} \wedge \texttt{r}=\texttt{NULL}$
$\vee\ \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\texttt{n}) * \texttt{G}(\texttt{r},\texttt{x},\texttt{l}_\texttt{m},\texttt{t}) * \texttt{G}(\texttt{l},\texttt{x},\texttt{res},\texttt{l}_\texttt{m}) * \texttt{H}_\texttt{n}(\texttt{n},\texttt{x}_\texttt{h},\texttt{t}_\texttt{h}) \wedge \texttt{r}\neq\texttt{NULL}$

(4)

Detecting and Eliminating Dangling Predicates

$\texttt{H}(\texttt{x},\texttt{p},\texttt{t}) \equiv \texttt{x}\mapsto\texttt{tree}(\mathcal{D}_\texttt{p},\mathcal{D}_\texttt{l},\texttt{r},\mathcal{D}_\texttt{n}) \wedge \texttt{r}=\texttt{NULL}$
$\vee\ \texttt{x}\mapsto\texttt{tree}(\mathcal{D}_\texttt{p},\texttt{l},\texttt{r},\mathcal{D}_\texttt{n}) * \texttt{H}(\texttt{l},\texttt{x},\texttt{lm}) * \texttt{H}(\texttt{r},\texttt{x},\texttt{t}) \wedge \texttt{r}\neq\texttt{NULL}$

$\texttt{G}(\texttt{x},\texttt{p},\texttt{res},\texttt{t}) \equiv \texttt{x}\mapsto\texttt{tree}(\texttt{p},\mathcal{D}_\texttt{l},\texttt{r},\texttt{t}) \wedge \texttt{res}=\texttt{x} \wedge \texttt{r}=\texttt{NULL}$
$\vee\ \texttt{x}\mapsto\texttt{tree}(\texttt{p},\texttt{l},\texttt{r},\mathcal{D}_\texttt{n}) * \texttt{G}(\texttt{l},\texttt{x},\texttt{res},\texttt{l}_\texttt{m}) * \texttt{G}(\texttt{r},\texttt{x},\texttt{l}_\texttt{m},\texttt{t}) \wedge \texttt{r}\neq\texttt{NULL}$

Eliminating Useless Parameters

$\texttt{H}(\texttt{x},\texttt{p},\texttt{t}) \equiv \texttt{H}_\texttt{f}(\texttt{x})$
$\texttt{H}_\texttt{f}(\texttt{x}) \equiv \texttt{x}\mapsto\texttt{tree}(\mathcal{D}_\texttt{p},\mathcal{D}_\texttt{l},\texttt{r},\mathcal{D}_\texttt{n}) \wedge \texttt{r}=\texttt{NULL}$
$\vee\ \texttt{x}\mapsto\texttt{tree}(\mathcal{D}_\texttt{p},\texttt{l},\texttt{r},\mathcal{D}_\texttt{n}) * \texttt{H}_\texttt{f}(\texttt{l}) * \texttt{H}_\texttt{f}(\texttt{r}) \wedge \texttt{r}\neq\texttt{NULL}$

Implementation and Experiments

Example	w/o norm. size	w/o norm. Syn.	w/ norm. size	w/ norm. Syn.	Veri.
SLL (delete)	9	0.23	5	0.23	0.2
SLL (reverse)	20	0.2	12	0.23	0.18
SLL (insert)	13	0.21	11	0.21	0.21
SLL (setTail)	7	0.18	2	0.19	0.18
SLL (get-last)	20	0.24	15	0.24	0.22
SLL-sorted (check)	8	0.26	2	0.27	0.23
SLL (bubblesort)	13	0.26	9	0.29	0.31
SLL (insertsort)	15	0.26	11	0.26	0.25
SLL (zip)	12	0.31	2	0.31	0.36
SLL-zip-leq	10	0.3	2	0.3	0.32
SLL + head (check)	9	0.23	2	0.25	0.2
SLL + tail (check)	9	0.25	2	0.26	0.23
skip-list$_2$ (check)	11	0.3	6	0.29	0.27
skip-list$_3$ (check)	17	0.45	3	0.46	0.45
SLL of 0/1 SLLs	8	0.24	8	0.24	0.25
CSLL (check)	8	0.23	2	0.2	0.22

Example	w/o norm. size	w/o norm. Syn.	w/ norm. size	w/ norm. Syn.	Veri.
CSLL (traverse)	8	0.23	2	0.24	0.33
CSLL of CSLLs (check)	18	0.31	2	0.3	0.31
SLL2DLL	8	0.19	2	0.18	0.19
DLL (check)	8	0.2	2	0.18	0.2
DLL (append)	23	0.18	8	0.16	0.19
CDLL (check)	9	0.25	7	0.24	0.23
CDLL of 5CSLLs	29	0.69	20	0.74	118
CDLL of CSLLs$_2$	33	0.44	22	0.41	0.51
tree (search)	11	0.25	2	0.24	0.24
tree-parent (traverse)	14	0.22	14	0.22	0.27
rose-tree (check)	9	0.22	7	0.24	0.23
swl (traverse)	19	0.39	17	0.34	1
mcf (check)	28	0.31	28	0.31	0.26
tll (traverse)	18	0.19	14	0.22	0.23
tll (check)	39	0.21	32	0.21	0.29
tll-parent (check)	19	0.24	13	0.26	0.29

Implementation:
- CIL infrastructure for C programs.
- Omega and Z3 to discharge numerical proof obligations

Experimental results:
- synthesis time < 1s (Syn. column)
- normalization (w/ norm column) reduces 40% (288/482) the size of synthesized predicates with an overhead of 1% time.

More experiment on Glib library:

	LOC	#Proc	#Loop	#√	Syn. (second)
gslist.c	863	44	19	59	3.45
glist.c	957	29	19	43	6.41
gtree.c	1334	36	14	43	5.26
gnode.c	1131	37	25	55	9.17

- gtree: balanced binary tree
- gnode: N-ary tree

infer specifications that guarantee memory safety for 90% of procedures & loops.