Satisfiability Modulo Heap-Based Programs

Loc Le (SUTD) Jun Sun (SUTD) Wei-Ngan Chin (NUS)

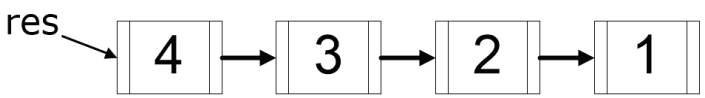
CAV 2016 - Toronto, Canada

July 21, 2016

Verifying Heap-Based Programs

Generating verification conditions for heap-based programs.

```
1 struct node {int val;node next};
2 node sll(int i){
3   if (i==0) return NULL;
4   return new node(i,sll(i-1));
5 }
6 int test(node x){
7   if (x==NULL) return 1;
8   else {
9     if (x->val <= 0) return 0;
10    else return test(x->next);
11  }
12 }
13 int main(int n) {
14  if (n<0) return 0;
15  node x = sll(n);
16  int r = test(x);//positive list?
17  if (r == 0) ERROR();
18  return 1;
19 }
```



Verification Condition:

Is ERROR() called?

Verifying Heap-Based Programs

Generating verification conditions for heap-based programs.

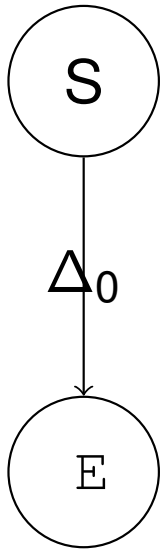
Related works:

- Constrained Horn Clauses based Verification: Bjørner *et. al.* SMT12, Gurfinkel *et. al.* CAV15 (SeaHorn)
- Large-block encoding: Beyer *et. al.* FMCAD09

Verifying Heap-Based Programs

From Verification Problem to Decision Problem.

```
1 struct node {int val;node next};
2 node sll(int i){
3   if (i==0) return NULL;
4   return new node(i,sll(i-1));
5 }
6 int test(node x){
7   if (x==NULL) return 1;
8   else {
9     if (x->val <= 0) return 0;
10    else return test(x->next);
11  }
12 }
13 int main(int n) {
14  if (n<0) return 0;
15  node x = sll(n);
16  int r = test(x);//positive list?
17  if (r == 0) ERROR();
18  return 1;
19 }
```



$$\Delta_0 \equiv sll(n,x) * test(x,r_1) \wedge n \geq 0 \wedge r_1 = 0$$

VCs are in separation logic with Inductive Predicates and arithmetic.

We need a satisfiability solver

Satisfiability Procedure for Inductive Predicates

new solver for Separation Logic (SL) with Inductive Predicates

- base fragment with empty heap ($\mathtt{emp}$), points-to ($\mapsto$), spatial conjunction ($*$) and Presburger Arithmetic

$$\begin{array}{ll} \mathtt{SAT} & \Delta_1 \equiv \mathtt{emp} \wedge x = \mathtt{null} \wedge n = 0 \\ \mathtt{UNSAT} & \Delta_2 \equiv x \mapsto \mathit{node}(n, y) * y \mapsto \mathit{node}(n-1, \mathtt{null}) \wedge x = y \end{array}$$

This base fragment is decidable (Piskac *et. al.* CAV2013)

Satisfiability Procedure for Inductive Predicates

new solver for Separation Logic with Inductive Predicates

- augmented with inductive predicates

$$\texttt{pred}\ sll(i,\texttt{root}) \equiv \texttt{emp}\wedge\texttt{root}=\texttt{null}\wedge i=0 \vee \exists\, r\cdot \texttt{root}\mapsto node(i,r) * sll(i-1,r)\wedge i>0$$

$$sll(n,x) \equiv (x=\texttt{null}\wedge n=0) \vee (z\mapsto node(n,\texttt{null})\wedge n=1) \vee \ldots$$

SAT $\quad \Delta_3 \equiv sll(n,x)\wedge n=1$ *Unfolding*

$\Delta_3 \equiv (x=\texttt{null}\wedge \underline{n=0}\wedge \underline{n=1}) \vee (x\mapsto node(n,\texttt{null})\wedge n>0\wedge n=1)$

UNSAT $\quad \Delta_4 \equiv sll(n,t)\wedge n<0$ *Unfolding*

$\Delta_4 \equiv (t=\texttt{null}\wedge \underline{n=0}\wedge \underline{n<0}) \vee \ldots$

Approaches to solving Δ_4

- Over-approximated Invariant Inference: $i\geq 0$
- Induction reasoning.

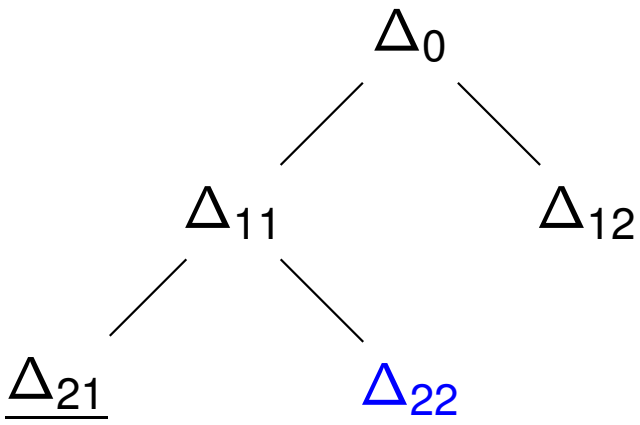
Contributions

Contribution 1: S2SAT - a generic semi-decision procedure for a logic

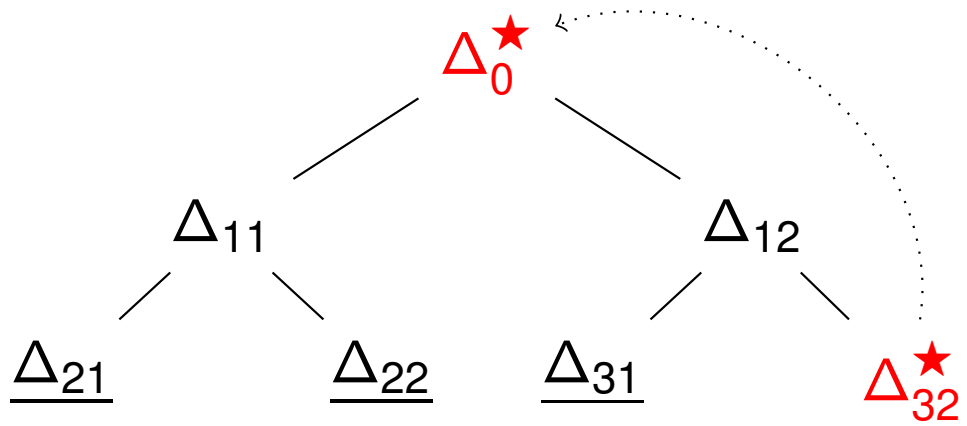
1. sound and complete base logic
2. augmented with inductive predicates

Given an input Δ_0, S2SAT

- constructs unfolding trees for Δ_0 st. $\Delta_0 \equiv \Delta_{21} \vee \Delta_{22} \vee ...$
- detects Cyclic Proof for induction proving.



SAT



UNSAT

Contributions

Contribution 2: $\texttt{S2SAT}_{\texttt{SL}}$ - an instantiation of $\texttt{S2SAT}$.

$\texttt{S2SAT}_{\texttt{SL}}$ is a semi-decision procedure for Separation Logic with General Inductive Predicates and Presburger Arithmetic.

$\texttt{S2SAT}_{\texttt{SL}}$: a solver for SL with Inductive Predicates

$\texttt{S2SAT}_{\texttt{SL}}$ is an instantiation of $\texttt{S2SAT}$.

```
1 struct node {int val;node next};
2 node sll(int i){
3   if (i==0) return NULL;
4   return new node(i,sll(i-1));
5 }
6 int test(node x){
7   if (x==NULL) return 1;
8   else {
9     if (x->val <= 0) return 0;
10    else return test(x->next);
11  }
12 }
13 int main(int n) {
14  if (n<0) return 0;
15  node x = sll(n);
16  int r = test(x);//positive list?
17  if (r == 0) ERROR();
18  return 1;
19 }
```

$$\texttt{pred}\ \mathit{sll}(i,\texttt{res}) \equiv \texttt{emp}\wedge\texttt{res}=\texttt{null}\wedge i=0 \vee \exists r\cdot \texttt{res}\mapsto \mathit{node}(i,r)*\mathit{sll}(i-1,r)\wedge i\neq 0$$

$$\overline{\mathit{inv}}: i\geq 0;$$

$$\texttt{pred}\ \mathit{test}(x,\texttt{res}) \equiv \texttt{emp}\wedge x=\texttt{null}\wedge\texttt{res}=1 \vee x\neq\texttt{null}\wedge x.\mathit{val}\leq 0\wedge\texttt{res}=0 \vee \mathit{test}(x.\mathit{next},\texttt{res})\wedge x\neq\texttt{null}\wedge x.\mathit{val}>0;$$

$$\overline{\mathit{inv}}: 0\leq\texttt{res}\leq 1;$$

$$\Delta_0 \equiv \mathit{sll}(n,x)*\mathit{test}(x,r_1)\wedge n\geq 0\wedge r_1=0$$

$\texttt{S2SAT}_{\texttt{SL}}$: a solver for SL with Inductive Predicates

$$\texttt{pred}\ sll(i,\texttt{res}) \equiv \texttt{emp}\wedge\texttt{res}=\texttt{null}\wedge i=0 \vee \exists r\cdot \texttt{res}\mapsto node(i,r)*sll(i-1,r)\wedge i\neq 0 \quad \overline{inv}: i\geq 0;$$

$$\texttt{pred}\ test(x,\texttt{res}) \equiv \texttt{emp}\wedge x=\texttt{null}\wedge\texttt{res}=1 \vee x\neq\texttt{null}\wedge x.val\leq 0\wedge\texttt{res}=0 \vee test(x.next,\texttt{res})\wedge x\neq\texttt{null}\wedge x.val>0 \quad \overline{inv}: 0\leq\texttt{res}\leq 1;$$

$$\Delta_0 \equiv sll(n,x)*test(x,r_1)\wedge n\geq 0\wedge r_1=0$$

1. Under-approximation. None
2. Over-Approximation. $\pi_0 \equiv n\geq 0\wedge 0\leq r_1\leq 1\wedge n\geq 0\wedge r_1=0$. Not $\texttt{UNSAT}$
3. Cyclic Detection. None

Δ_0

Figure : Unfolding Tree $\mathcal{T}_0$.

$\texttt{S2SAT}_{\texttt{SL}}$: a solver for SL with Inductive Predicates

$\texttt{pred}\ sll(i,\texttt{res}) \equiv \texttt{emp}\wedge\texttt{res}=\texttt{null}\wedge i=0$
$\vee\ \exists r\cdot\ \texttt{res}\mapsto node(i,r)*sll(i-1,r)\wedge i\neq 0$ $\overline{inv}$: $i\geq 0$;
$\texttt{pred}\ test(x,\texttt{res}) \equiv \ldots$ $\overline{inv}$: $0\leq\texttt{res}\leq 1$;
$\Delta_0 \equiv sll(n,x)*test(x,r_1)\wedge n\geq 0\wedge r_1=0$
$\Delta_{11} \equiv test(x,r_1)\wedge n\geq 0\wedge r_1=0\wedge x=\texttt{null}\wedge n=0$
$\Delta_{12} \equiv x\mapsto node(n,x_1)*sll(n-1,x_1)*test(x,r_1)\wedge n\geq 0\wedge r_1=0\wedge n\neq 0$

1. Under-approximation. None
2. Over-Approximation.
   $\pi_{11} \equiv 0\leq r_1\leq 1\wedge n\geq 0\wedge r_1=0\wedge x=\texttt{null}\wedge n=0$.
   Not $\texttt{UNSAT}$
   $\pi_{12} \equiv x\neq\texttt{null}\wedge n-1\geq 0\wedge 0\leq r_1\leq 1\wedge n\geq 0\wedge$
   $r_1=0\wedge n\neq 0$. Not $\texttt{UNSAT}$
3. Cyclic Detection. None

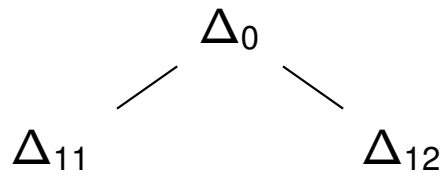


Figure : Unfolding Tree $\mathcal{T}_1$.

S2SAT$_{SL}$: a solver for SL with Inductive Predicates

$$\texttt{pred}\ \mathit{test}(x,\texttt{res}) \equiv \texttt{emp}\wedge x{=}\texttt{null}\wedge\texttt{res}{=}1 \vee x{\neq}\texttt{null}\wedge x.\mathit{val}{\leq}0\wedge x{=}0 \vee \mathit{test}(x.\mathit{next},\texttt{res})\wedge x{\neq}\texttt{null}\wedge x.\mathit{val}{>}0\ \ \overline{\mathit{inv}}{:}\ 0{\leq}\texttt{res}{\leq}1;$$

$$\Delta_0{\equiv}\mathit{sll}(n,x){*}\mathit{test}(x,r_1)\wedge n{\geq}0\wedge r_1{=}0$$
$$\Delta_{11}{\equiv}\mathit{test}(x,r_1)\wedge n{\geq}0\wedge r_1{=}0\wedge x{=}\texttt{null}\wedge n{=}0$$
$$\Delta_{12}{\equiv}x{\mapsto}\mathit{node}(n,x_1){*}\mathit{sll}(n{-}1,x_1){*}\mathit{test}(x,r_1)\wedge n{\geq}0\wedge r_1{=}0\wedge n{\neq}0$$

Context-sensitive Unfolding using *branch* invariant:

1. $\mathit{inv}_{br_1}{\equiv}x{=}\texttt{null}\wedge r_1{=}1$. Contradict with $r_1{=}0$ in Δ_{11}.
2. $\mathit{inv}_{br_2}{\equiv}x{\neq}\texttt{null}\wedge r_1{=}0$
3. $\mathit{inv}_{br_3}{\equiv}x{\neq}\texttt{null}\wedge 0{\leq}r_1{\leq}1$

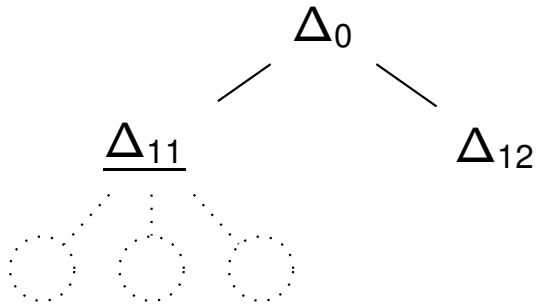


Figure : Unfolding Tree $\mathcal{T}_2$.

S2SAT$_{SL}$: a solver for SL with Inductive Predicates

pred $sll(i,\mathtt{res}) \equiv \ldots \overline{inv}: i \geq 0;$
pred $test(x,\mathtt{res}) \equiv \ldots \overline{inv}: 0 \leq \mathtt{res} \leq 1;$

$\Delta_0 \equiv sll(n,x) * test(x,r_1) \wedge n \geq 0 \wedge r_1 = 0$
$\Delta_{32} \equiv x \mapsto node(n,x_1) * ll(n-1,x_1) \wedge x \neq \mathtt{null} \wedge \underline{n<0} \wedge r=0 \wedge \underline{n>0} \wedge r_1 = 0$
$\Delta_{33} \equiv x \mapsto node(n,x_1) * sll(n-1,x_1) * test(x_1,r_1) \wedge x \neq \mathtt{null} \wedge n>0 \wedge r_1 = 0$

1. Under-approximation. None
2. Over-Approximation. Δ_{32} is UNSAT.
   $\pi_{33} \equiv x \neq \mathtt{null} \wedge n-1 \geq 0 \wedge 0 \leq r_1 \leq 1 \wedge$
   $x \neq \mathtt{null} \wedge n>0 \wedge r_1 = 0$. Not UNSAT
3. Cyclic Detection. Δ_{33} is linked back to Δ_0.

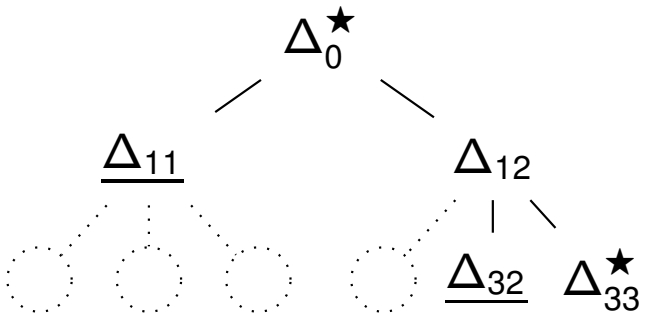


Figure : Unfolding Tree $\mathcal{T}_3$.

S2SAT$_{SL}$: a solver for SL with Inductive Predicates

More in the paper

1. A decidable fragment s.t. S2SAT$_{SL}$ always terminates.
2. Over-approximated Invariant Inference

Implementation and Evaluation

Implemented solver $\texttt{S2SAT}_{\texttt{SL}}$ and verifier $\texttt{S2}_{\texttt{td}}$ based on HIP/SLEEK (Chin *et. al SCP12*).

- 102 recursive/loop programs taken from SV-COMP2016
- points: +2 for s$\surd$, +1 for e$\surd$, 0 unk, -16 for s$\times$, and -32 for e$\times$

Tool	#s$\surd$	#e$\surd$	#unk	#s$\times$	#e$\times$	points	mins
ESBMC	38	40	21	0	3	20	53
UAutomizer	17	23	62	0	0	57	23
SeaHorn	48	45	5	4	0	77	26
CBMC	33	39	29	1	0	89	90
Smack-Corral	33	37	28	0	0	103	105
$\texttt{S2}_{\texttt{td}}$	41	45	16	0	0	127	25

$\texttt{S2}_{\texttt{td}}$ performed well over recursive/loop programs.

Implementation and Evaluation

More experimental results in the paper

- $\texttt{S2SAT}_{\texttt{SL}}$ vs. SLSAT (Brotherston *et. al.* LiSC2014). $\texttt{S2SAT}_{\texttt{SL}}$ is more effective and efficient than SLSAT.

- $\texttt{S2SAT}_{\texttt{SL}}$ on a set of satisfiability problems over complex data structures. $\texttt{S2SAT}_{\texttt{SL}}$ is expressive.

Conclusion

1. S2SAT, a generic satisfiability procedure for a logic
   - sound and complete base logic
   - augmented with inductive predicates
2. S2SAT_{SL}, a semi-decision procedure for Separation Logic with general Inductive Predicates and Presburger Arithmetic.
3. Experimental Evaluation for S2SAT_{SL} and S2_{td}.

SAT Procedure for Inductive Predicates

Building a new SAT solver for Presburger with Inductive Predicates

1. Presburger arithmetic is decidable [1].
2. augmented with inductive predicates

$$\begin{array}{l}\texttt{pred}\ \mathit{add}(n,m,\texttt{res})\ \equiv\ \texttt{res}{=}m \wedge n{=}0\\ \qquad\quad \vee\ \exists r_1 \cdot\ \mathit{add}(n{-}1,m,r_1) \wedge n{>}0 \wedge \texttt{res}{=}r_1{+}1\\ \overline{\mathit{inv}}\colon n{\geq}0;\end{array}$$

[1] *H. B. Enderton, A Mathematical Introduction to Logic, Second Edition, Academic Press, 2000.*