

A Decidable Fragment in Separation Logic with Inductive Predicates and Arithmetic

Quang Loc Le (TU) Makoto Tatsuta (NII)
Jun Sun (SUTD) Wei-Ngan Chin (NUS)

Computer Aided Verification, 29th International Conference
Heidelberg Germany

July 28, 2017

A fragment of Separation Logic

Formula	$\Phi ::= \Delta \mid \Phi_1 \vee \Phi_2$	$\Delta ::= \exists \bar{v}. (\kappa \wedge \pi)$
Spatial formula	$\kappa ::= \text{emp} \mid x \mapsto c(v_i) \mid P(\bar{v}) \mid \kappa_1 * \kappa_2$	
Pure formula	$\pi ::= \pi_1 \wedge \pi_2 \mid \alpha \mid \phi$	

- α : Pointer (Dis)Equalities
- ϕ : Presburger arithmetic
- P : inductive predicate. Predicate Definition: $P(\bar{t}) \equiv \Phi$

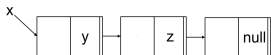
Warning: no pointer arithmetic and no magic wand

A fragment of Separation Logic

- Inductive predicate: Singly-linked list with size property

$$\text{pred } ll_size(\text{root}, n) \equiv \text{emp} \wedge \text{root} = \text{null} \wedge n = 0 \\ \vee \exists r, n_1. \text{root} \mapsto \text{node}(_, r) * ll_size(r, n_1) \wedge n = n_1 + 1$$

- Example:



$ll_size(x, 3)$

- Numerical projection

$$ll_size^N(n) \equiv n = 0 \\ \vee \exists n_1. ll_size^N(n_1) \wedge n = n_1 + 1$$

Satisfiability Problem

- Input: A formula Δ in the fragment
- Question: Is Δ satisfiable?

Challenges:

- Unbounded heaps
- Infinite numerical domain

The satisfiability problem is undecidable
by simulating Peano arithmetic (Tatsuta *et. al.* - APLAS 2016).

What is decidable?

- Decidable Fragment:

A subfragment which is decidable and more expressive than all fragments which have been shown to be decidable previously.

- Decision Procedure: Base Computation

Compute for each inductive predicate a **finite representation** that precisely characterises its satisfiability.

Decidable Fragment

Finite Representation: Base Formula (without inductive predicates)

- Combining empty heap (emp), points-to (\mapsto), spatial conjunction ($*$) and Presburger Arithmetic
- Example:

SAT $\Delta_1 \equiv \text{emp} \wedge x = \text{null} \wedge n = 0$

UNSAT $\Delta_2 \equiv x \mapsto \text{node}(n, y) * y \mapsto \text{node}(n-1, \text{null}) \wedge x = y$

The fragment of base formulas is decidable

(Piskac, Wies and Zufferey - CAV 2013, Navarro and Rybalchenko
- APLAS 2013)

Foundation of Base Computation

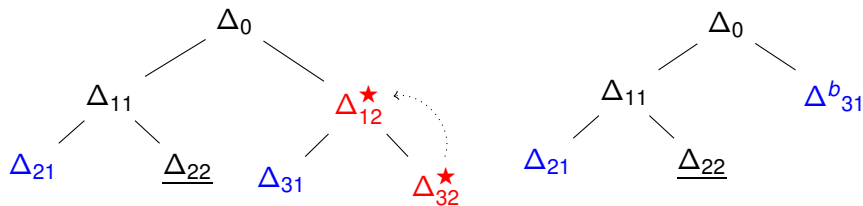
- For each formula, eliminating existentially quantified pointer-typed variables produces an **equi-satisfiable** formula.
- Example: $\Delta_1 \equiv \exists r. ll_size(r, n) \wedge x = \text{null} \wedge n = 0$
is equi-satisfiable with
 $\Delta_2 \equiv \exists r. ll_size^N(n) \wedge x = \text{null} \wedge n = 0$

If $ll_size^N(n)$ can be computed as a Presburger formula, then Δ_2 can be reduced into a base formula and thus is decidable.

Decidable Fragment: Base Computation

Given an inductive predicate $P(\bar{x}) \equiv \Phi$,

- 1 Construct a cyclic unfolding tree for $\Delta_0 \equiv P(\bar{x})$
- 2 Flatten the tree into a disjunctive set of base formulas

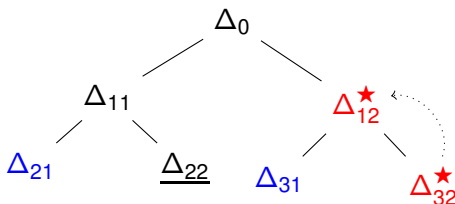


$$\text{base}^{\mathcal{P}}(P(\bar{x})) \equiv \{\Delta_{21}, \Delta^{b}_{31}\}$$

Constructing Cyclic Unfolding Tree

Given an inductive predicate $P(\bar{x}) \equiv \Phi$, construct a **unfolding tree** for $\Delta_0 \equiv P(\bar{x})$ through iterations of actions:

- 1 Choose a (open) leaf, close it
 - it can be reduced into a **base** formula.
 - a base formula
 - a formula in which pointer-typed parameters of every inductive predicates are existentially quantified.
 - its over-approximation is unsat.
 - can be linked back to form a **circular** path.
- 2 Otherwise, unfold it.



Constructing Cyclic Unfolding Tree

$$\text{pred } Q(x, y, n) \equiv \exists y_1. x \mapsto \text{node}(\text{null}, y_1) \wedge y = \text{null} \wedge x \neq \text{null} \wedge n = 1 \\ \vee \exists x_1, y_1, n_1. y \mapsto \text{node}(x_1, y_1) * Q(x, y_1, n_1) \wedge y \neq \text{null} \wedge n = n_1 + 2;$$

$$\Delta_0 \equiv Q(x, y, n)$$

- 1 Base Detection. None
- 2 Over-Approximation. $\pi_0 \equiv \text{true}$.
Not UNSAT
- 3 Cyclic Detection. None

Δ_0

Figure : Unfolding Tree \mathcal{T}_0 .

Constructing Cyclic Unfolding Tree

$$\text{pred } Q(x, y, n) \equiv \exists y_1. x \mapsto \text{node}(\text{null}, y_1) \wedge y = \text{null} \wedge x \neq \text{null} \wedge n = 1 \\ \vee \exists x_1, y_1, n_1. y \mapsto \text{node}(x_1, y_1) * Q(x, y_1, n_1) \wedge y \neq \text{null} \wedge n = n_1 + 2;$$

$$\Delta_0 \equiv Q(x, y, n)$$

$$\Delta_1 \equiv \exists y_1. x \mapsto \text{node}(\text{null}, y_1) \wedge y = \text{null} \wedge x \neq \text{null} \wedge n = 1$$

$$\Delta_2 \equiv \exists x_1, y_1, n_1. y \mapsto \text{node}(x_1, y_1) * Q(x, y_1, n_1) \wedge y \neq \text{null} \wedge n = n_1 + 2$$

- 1 Base Detection. Δ_1
- 2 Over-Approximation.
 $\pi_2 \equiv \exists x_1, y_1, n_1. y \mapsto \text{node}(x_1, y_1) \wedge \text{true} \\ \wedge y \neq \text{null} \wedge n = n_1 + 2. \text{ Not UNSAT}$
- 3 Cyclic Detection. None

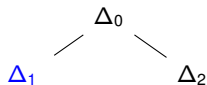


Figure : Unfolding Tree \mathcal{T}_1 .

Constructing Cyclic Unfolding Tree

$$\text{pred } Q(x, y, n) \equiv \exists y_1. x \mapsto \text{node}(\text{null}, y_1) \wedge y = \text{null} \wedge x \neq \text{null} \wedge n = 1 \\ \vee \exists x_1, y_1, n_1. y \mapsto \text{node}(x_1, y_1) * Q(x, y_1, n_1) \wedge y \neq \text{null} \wedge n = n_1 + 2;$$

$$\Delta_2 \equiv \exists x_1, y_1, n_1. y \mapsto \text{node}(x_1, y_1) * Q(x, y_1, n_1) \wedge y \neq \text{null} \wedge n = n_1 + 2$$

$$\Delta_3 \equiv \exists x_1, y_1, n_1, y_2. y \mapsto \text{node}(x_1, y_1) * x \mapsto \text{node}(\text{null}, y_2) \wedge \\ y_1 = \text{null} \wedge x \neq \text{null} \wedge n_1 = 1 \wedge y \neq \text{null} \wedge n = n_1 + 2$$

$$\Delta_4 \equiv \exists x_1, y_1, n_1, x_2, y_2, n_2. y \mapsto \text{node}(x_1, y_1) * y_1 \mapsto \text{node}(x_2, y_2) * \\ Q(x, y_2, n_2) \wedge y_1 \neq \text{null} \wedge n_1 = n_2 + 2 \wedge y \neq \text{null} \wedge n = n_1 + 2$$

- 1 Base Detection. Δ_3
- 2 Over-Approximation. $\pi_4 \equiv \dots$
Not UNSAT
- 3 Cyclic Detection. Yes

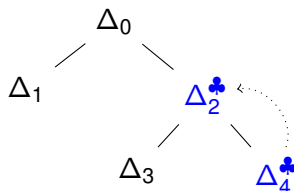


Figure : \mathcal{T}_2^Q .

Cyclic Detection

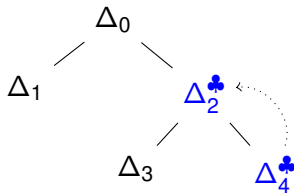
$$\Delta_2 \equiv \exists x_1, y_1, n_1. y_1 \mapsto \text{node}(x_1, y_1) * Q(x, y_1, n_1) \wedge y \neq \text{null} \wedge n = n_1 + 2$$

$$\Delta_4 \equiv \exists x_1, y_1, n_1, x_2, y_2, n_2. y_1 \mapsto \text{node}(x_1, y_1) * y_1 \mapsto \text{node}(x_2, y_2) * \\ Q(x, y_2, n_2) \wedge y_1 \neq \text{null} \wedge n_1 = n_2 + 2 \wedge y \neq \text{null} \wedge n = n_1 + 2$$

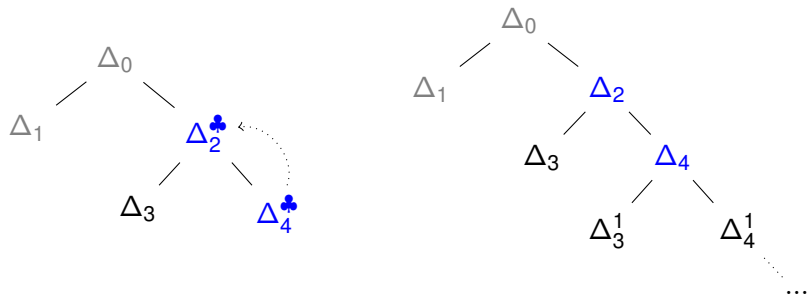
Steps

- 1 matching externally visible points-to predicate: $y \mapsto \text{node}(-, -)$
- 2 matching externally visible inductive predicates: $Q(x, -, -)$
 - In general, we may need to group isomorphic inductive predicates beforehand (same predicate name and same sequence of free arguments)
- 3 matching externally visible (dis)equalities over pointers: $y \neq \text{null}$

Flattening Cyclic Unfolding Tree



Flattening Cyclic Unfolding Tree

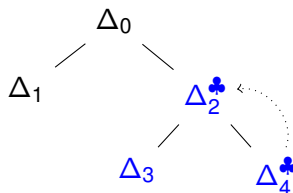


$$\Delta_3^{flat} \equiv \Delta_3 \vee \Delta_3^1 \vee \dots$$

$$\Delta_3 \equiv \exists x_1, y_1, n_1, y_2. (y_1 \mapsto \text{node}(x_1, y_1) * x_1 \mapsto \text{node}(\text{null}, y_2) \wedge x_1 \neq \text{null} \wedge y_1 \neq \text{null} \wedge n = n_1 + 1) \wedge (y_1 = \text{null} \wedge n_1 = 1)$$

$$\Delta_3^1 \equiv \exists x_1, y_1, n_1, x_2, y_2, n_2, y_3. (y_1 \mapsto \text{node}(x_1, y_1) * x_1 \mapsto \text{node}(\text{null}, y_3) \wedge x_1 \neq \text{null} \wedge y_1 \neq \text{null} \wedge n = n_1 + 1) * (y_1 \mapsto \text{node}(x_2, y_2) \wedge y_2 = \text{null} \wedge \underline{n_1 = n_2 + 2} \wedge n_2 = 1)$$

Flattening Cyclic Unfolding Tree



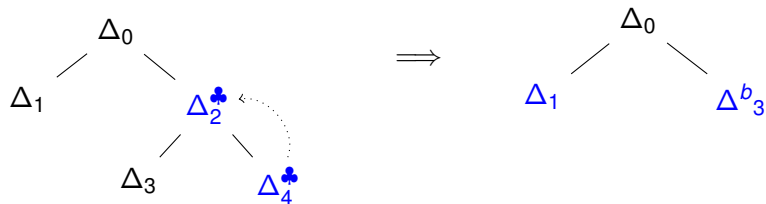
$$P_{\text{cyc}}(n_1) \equiv n_1 = 1 \vee \exists n_2. n_1 = n_2 + 2 \wedge P_{\text{cyc}}(n_2)$$

$$P_{\text{cyc}}(n_1) \equiv \exists k. n_1 = 2k + 1 \wedge k \geq 0$$

Δ^b_3 is equi-satisfiable to Δ_3^{flat} :

$$\Delta^b_3 \equiv \exists x_1, y_1, x_2, y_2, n_1. (y_1 \mapsto \text{node}(x_1, y_1) * x_1 \mapsto \text{node}(\text{null}, y_2) \wedge x \neq \text{null} \wedge y \neq \text{null} \wedge n = n_1 + 1) \wedge (\exists k. n_1 = 2k + 1 \wedge k \geq 0)$$

Flattening Cyclic Unfolding Tree



$$\text{base}^{\mathcal{P}}(Q(x, y, n)) \equiv \{\Delta_1, \Delta^b_3\}$$

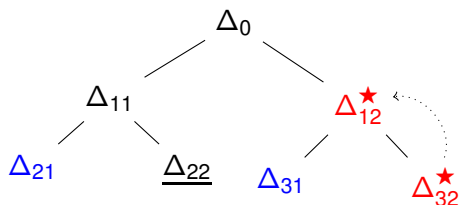
- An inductive predicate is in the proposed decidable fragment if all
 - numerical projections of base leaves; and
 - P_{cyc} predicatesare Presburger-definable (i.e., can be computed as Presburger formulas).
- Some systems of arithmetic inductive predicates are Presburger-definable:
 - DPI (Tatsuta *et. al.* - APLAS 2016)
 - periodic sets (Bozga *et. al.* - CAV 2010)

- Correctness, Termination, Complexity
 - Correctness of Cyclic Proofs (Le *et. al* - CAV 2016)
- Implementation and Evaluation
 - Based on HIP/SLEEK/S2 (Chin *et. al*. SCP 2012)
 - Equi-satisfiable Bases (sll, dll, even lists, ..)
 - Over-approximated/Under-approximated Bases (avl, rb, ..)

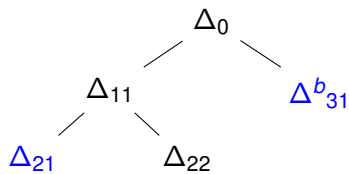
Conclusion

- A decision procedure for an extensible decidable fragment in separation logic including general inductive predicates and arithmetic
- Base Computation:

Construct Unfolding Tree



Flatten Unfolding Tree



$$\text{base}^{\mathcal{P}}(\mathcal{P}(\bar{v})) \equiv \{\Delta_{21}, \Delta_{31}^b\}$$