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

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

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

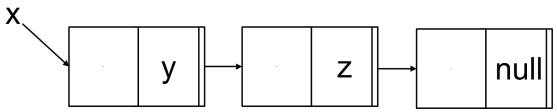
Warning: no pointer arithmetic and no magic wand

A fragment of Separation Logic

- Inductive predicate: Singly-linked list with size property

$$\texttt{pred}\ \mathit{ll\_size}(\texttt{root},n) \equiv \texttt{emp}\wedge\texttt{root}=\texttt{null}\wedge n=0 \vee \exists\, r,n_1 \cdot \texttt{root}\mapsto \mathit{node}(\_,r) * \mathit{ll\_size}(r,n_1) \wedge n=n_1+1$$

- Example:



$\mathit{ll\_size}(x,3)$

- Numerical projection

$$\mathit{ll\_size}^N(n) \equiv n=0 \vee \exists\, n_1 \cdot \mathit{ll\_size}^N(n_1)\wedge n=n_1+1$$

Satisfiability Problem

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

Challenges:

- Unbounded heaps
- Infinite numerical domain

Satisfiability Problem

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

What is decidable?

Contributions

- 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 ($\mathtt{emp}$), points-to ($\mapsto$), spatial conjunction ($*$) and Presburger Arithmetic

- Example:

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

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.

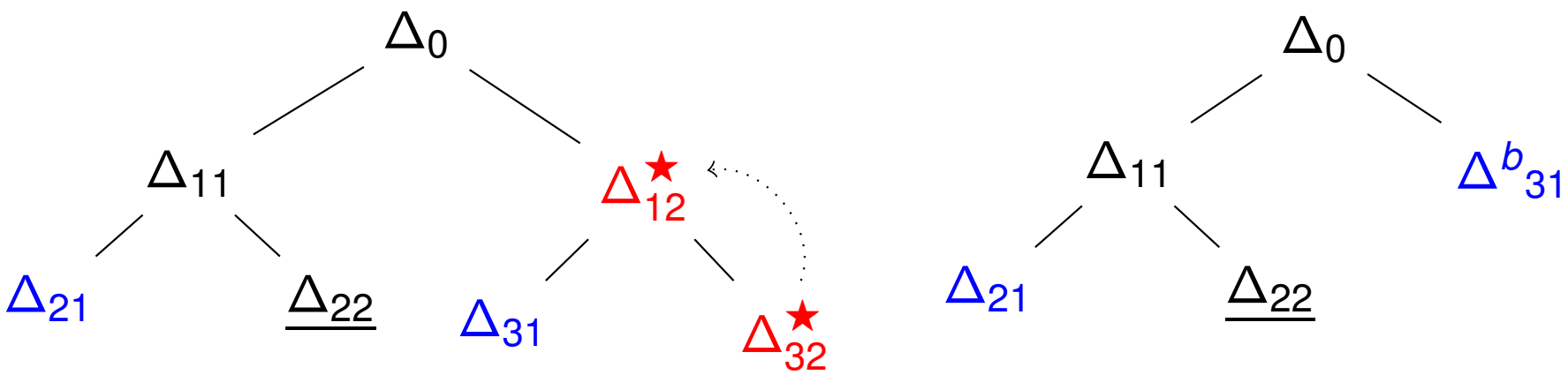
- Example: $\Delta_1 \equiv \exists r.\mathit{ll\_size}(r,n) \wedge x{=}\mathtt{null} \wedge n{=}0$
  is equi-satisfiable with
  $\Delta_2 \equiv \exists r.\mathit{ll\_size}^N(n) \wedge x{=}\mathtt{null} \wedge n{=}0$

If $\mathit{ll\_size}^N(n)$ 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

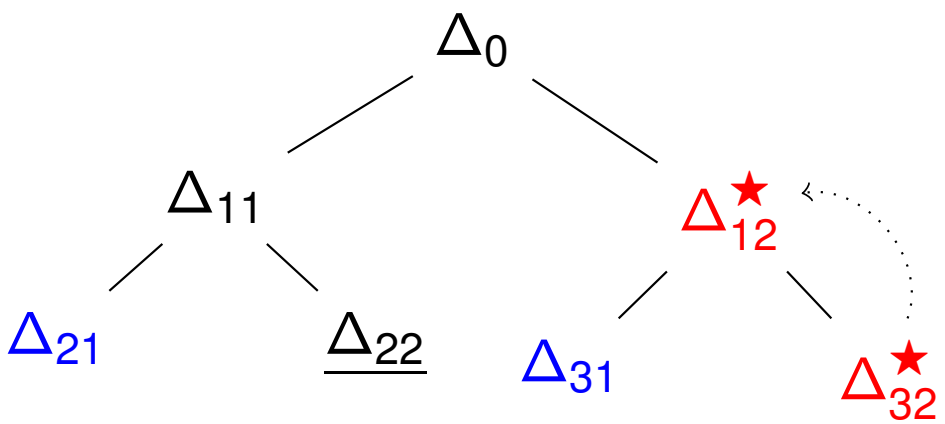


$$\mathtt{base}^{\mathcal{P}}(P(\bar{x})) \equiv \{\Delta_{21}, \Delta^{b}_{31}\}$$

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 if
   - 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

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

$\Delta_0 \equiv \mathtt{Q}(x,y,n)$

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

Δ_0

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

Constructing Cyclic Unfolding Tree

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

$\Delta_0{\equiv}\mathtt{Q}(x,y,n)$

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

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

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

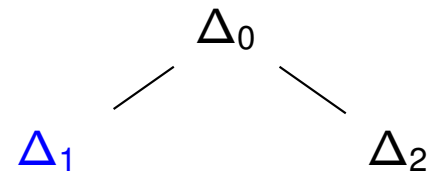


Figure : Unfolding Tree $\mathcal{T}_1$.

Constructing Cyclic Unfolding Tree

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

$\Delta_2 \equiv \exists\, x_1,y_1,n_1.y\mapsto node(x_1,y_1) * Q(x,y_1,n_1)\wedge y\neq\texttt{null}\wedge n=n_1+2$
$\Delta_3 \equiv \exists\, x_1,y_1,n_1,y_2.y\mapsto node(x_1,y_1) * x\mapsto node(\texttt{null},y_2)\wedge$
$\quad y_1=\texttt{null}\wedge x\neq\texttt{null}\wedge n_1=1\wedge y\neq\texttt{null}\wedge n=n_1+2$
$\Delta_4 \equiv \exists\, x_1,y_1,n_1,x_2,y_2,n_2.y\mapsto node(x_1,y_1)*y_1\mapsto node(x_2,y_2)*$
$\quad Q(x,y_2,n_2)\wedge y_1\neq\texttt{null}\wedge n_1=n_2+2\wedge y\neq\texttt{null}\wedge n=n_1+2$

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

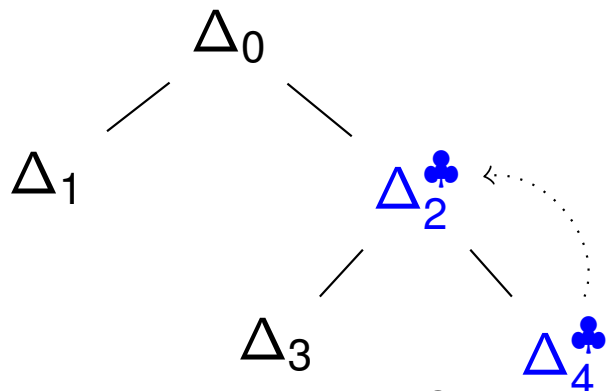


Figure : $\mathcal{T}_2^Q$.

Constructing Cyclic Unfolding Tree

Cyclic Detection

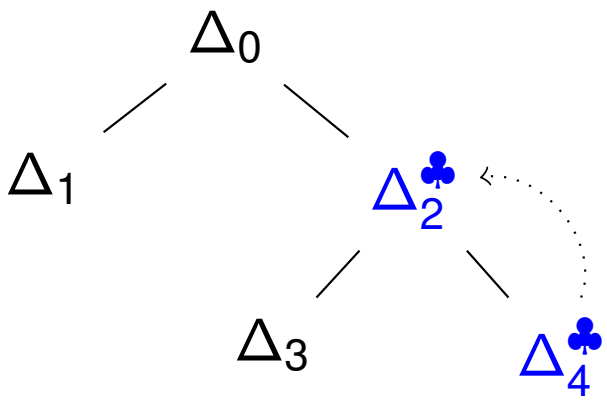
$$\Delta_2 \equiv \exists\, x_1, y_1, n_1 .\, y \mapsto node(x_1, y_1) * \mathcal{Q}(x, y_1, n_1) \wedge y \neq \mathtt{null} \wedge n = n_1 + 2$$

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

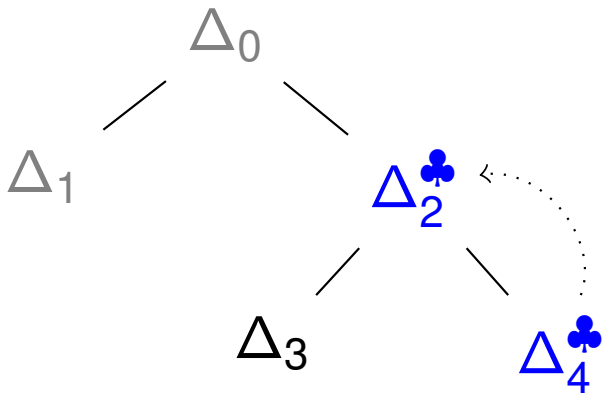
Steps

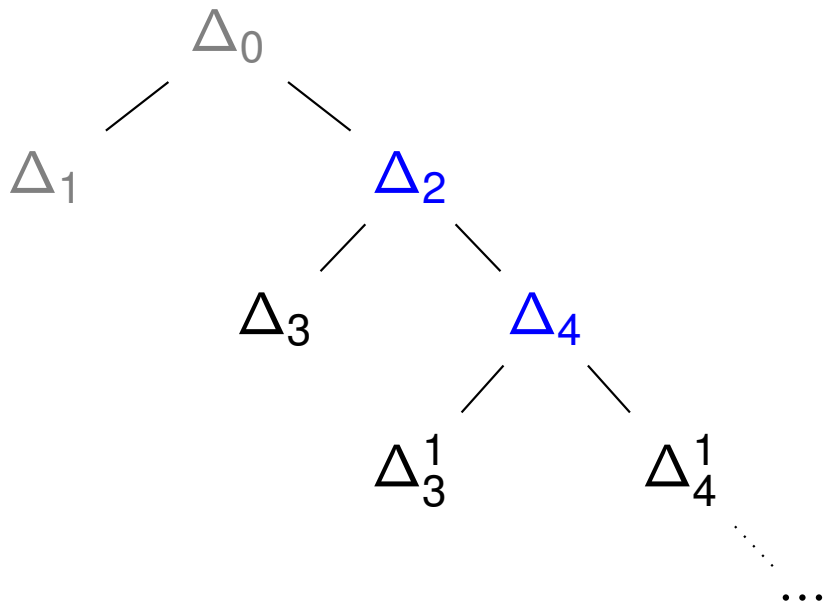
1. matching externally visible points-to predicate: $y \mapsto node(\_, \_)$
2. matching externally visible inductive predicates: $\mathcal{Q}(x, \_, \_)$
   - In general, we may need to group isomorphic inductive predicates beforehand (same predicate name and same sequence of free arguments)
3. matching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

Flattening Cyclic Unfolding Tree



Flattening Cyclic Unfolding Tree



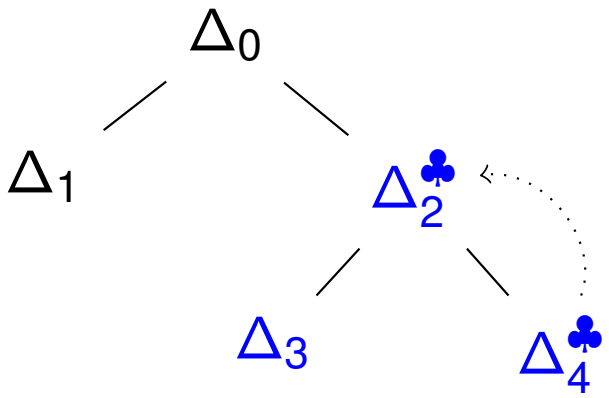


$\Delta_3^{flat} \equiv \Delta_3 \vee \Delta_3^1 \vee \ldots$

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

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

Flattening Cyclic Unfolding Tree



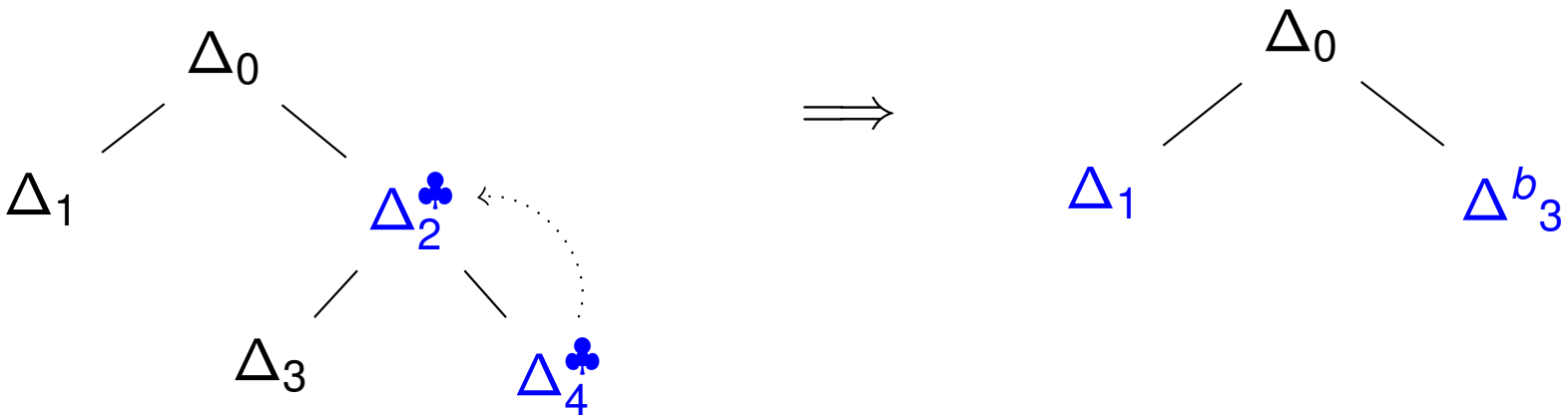
$$\mathtt{P}_{\mathtt{cyc}}(n_1) \equiv n_1 = 1 \vee \exists n_2 . n_1 = n_2 + 2 \wedge \mathtt{P}_{\mathtt{cyc}}(n_2)$$

$$\mathtt{P}_{\mathtt{cyc}}(n_1) \equiv \exists k . n_1 = 2k + 1 \wedge k \geq 0$$

$\Delta^b{}_3$ is equi-satisfiable to Δ_3^{flat}:

$$\Delta^b{}_3 \equiv \exists\, x_1, y_1, x_2, y_2, n_1 . (y \mapsto node(x_1, y_1) * x \mapsto node(\mathtt{null}, y_2) \wedge x \neq \mathtt{null} \wedge y \neq \mathtt{null} \wedge n = n_1 + 2) \wedge (\exists k . n_1 = 2k + 1 \wedge k \geq 0)$$

Flattening Cyclic Unfolding Tree



$\mathtt{base}^{\mathcal{P}}(\mathtt{Q}(\mathtt{x},\mathtt{y},\mathtt{n}))\equiv\{\Delta_1, \Delta^b{}_3\}$

Proposed Decidable Fragment

- An inductive predicate is in the proposed decidable fragment if all
  - numerical projections of base leaves; and
  - P_{cyc} predicates

  are 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)

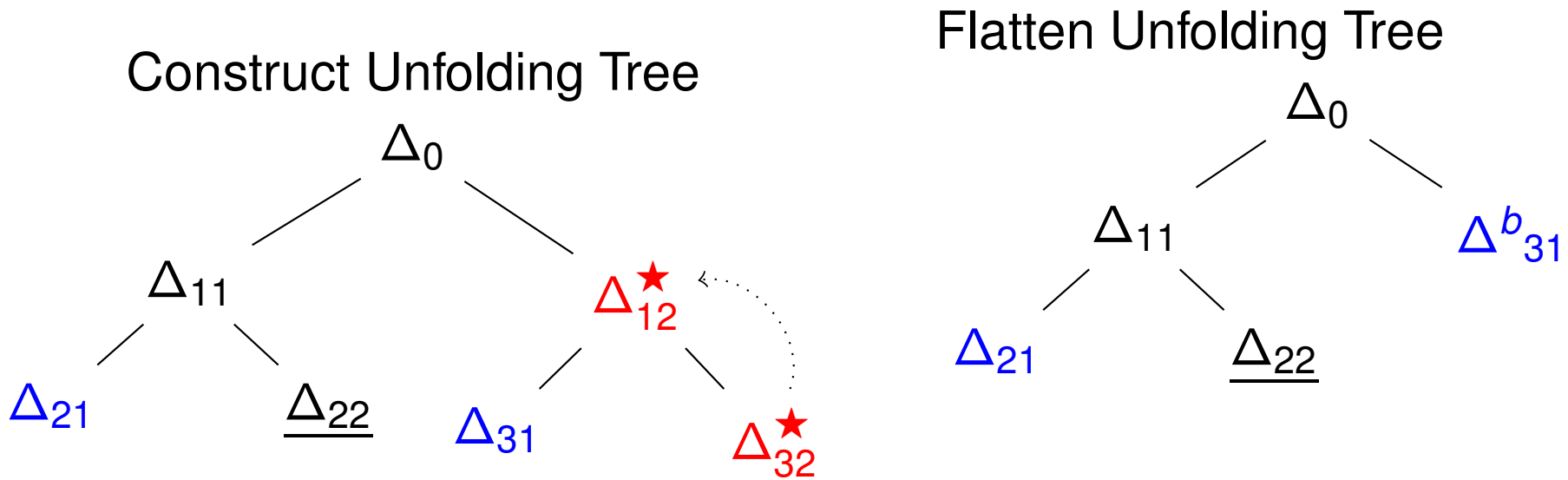
More in the paper

- 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:



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