Existential Entailment in Logical Form for Array Logic

Zhuohong Cai Wei-Ngan Chin

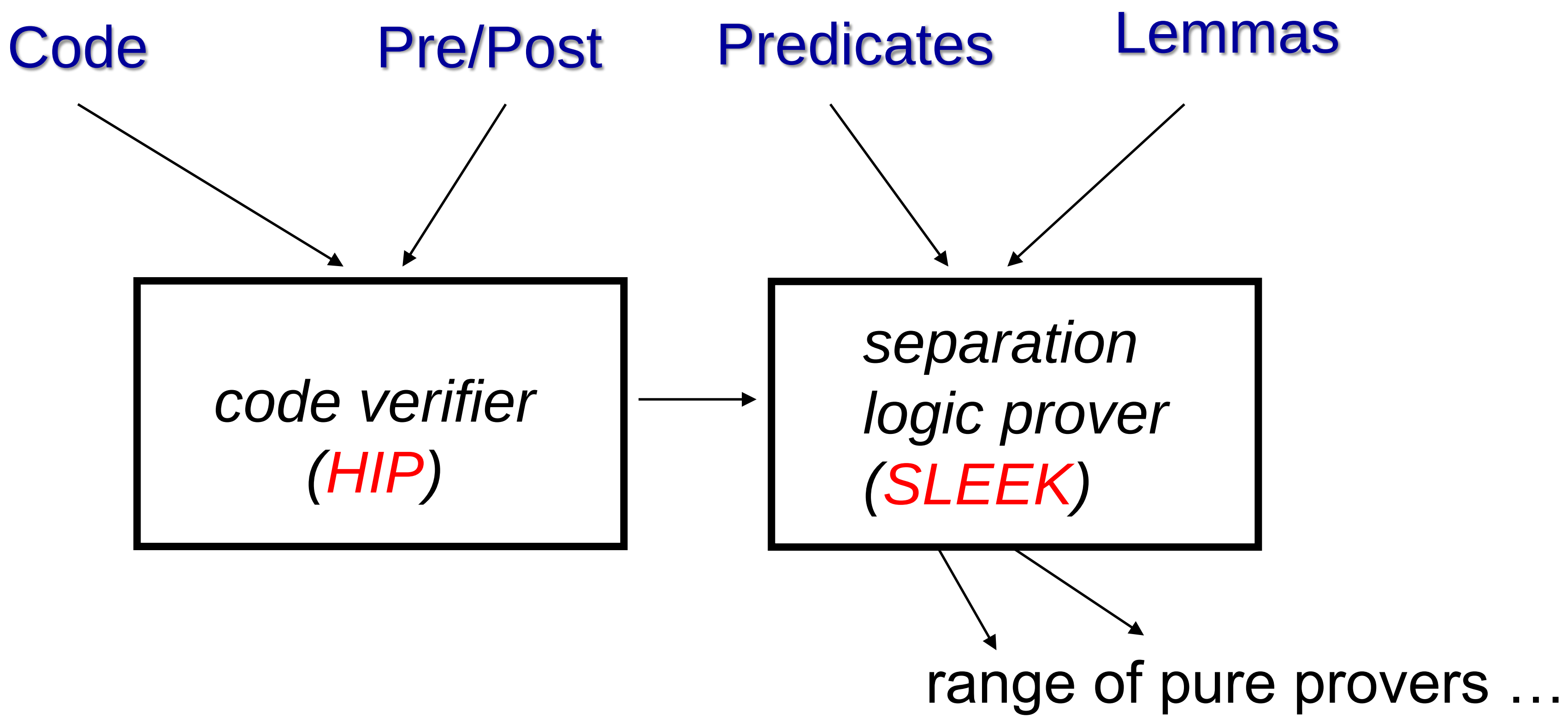
Dept of Computer Science

National University of Singapore

Demos http://loris-5.d2.comp.nus.edu.sg

Overall System

Under development since 2006 (200K+ lines of OCaml)
Currently : 4 current PhD students; 7 graduated PhD



Omega, MONA, Isabelle, Coq, SMT, Redlog, MiniSAT, Mathematica

Research Directions

- Expressive Specification (POPL08,FM11)
- Inference (APLAS13,CAV14)
- Termination (ICFEM14,PLDI15)
- Concurrency (ATVA13,PEPM15,IECCS15)
- Lemmas (CAV08,POPL18)
- Algorithms (CAV16,FM16)+today

Sound and Decision Procedures

Algorithms

- Satisfiability
  - Over-Approximation (UNSAT)
  - Under-Approximation (SAT)
  - Decision Procedure (SAT & UNSAT)
- Entailment
  - Classical, Frame Inference, Bi-Abduction
  - Sound Entailment with Maybe & Must error
  - Decision Procedure (for some fragment)

Scope of Interests

- Decision Procedure
  - Array Logic with existential quantifiers
  - Bi-Abductive Entailment
  - Array with Universal Properties
- Today's Presentation
  - Entailment of Array Logic with Frame Inference
  - Lazy Case Splitting

Example of Verification

```
int* foo(int *a)
{
   int t = random(6, 8);
   int *tmp = a + t;
   *tmp = 42;
   return tmp;
}
```

- Ensure memory safety
- Use pre-post specification

$$\exists r, c \cdot \ \texttt{requires}\ (r \mapsto c) \land 6 \leqslant r - a \leqslant 8$$
$$\texttt{ensures}\ (r \mapsto 42) \land res = r$$

Array Specification Logic

Array Logic

$$
\begin{array}{rrcl}
\text{Term} & \tau & ::= & v \mid c \mid -\tau \mid \tau_1 + \tau_2 \mid \tau_1 - \tau_2 \mid c\tau \\
\text{Pure formula} & \pi & ::= & \mathtt{true} \mid \mathtt{false} \mid \tau_1 < \tau_2 \mid \tau_1 \leqslant \tau_2 \mid \tau_1 > \tau_2 \mid \tau_1 \geqslant \tau_2 \mid \tau_1 = \tau_2 \mid \tau_1 \neq \tau_2 \mid \\
& & & \neg\pi \mid \pi_1 \wedge \pi_2 \mid \pi_1 \vee \pi_2 \mid \pi_1 \rightarrow \pi_2 \mid \exists v.\,\pi \mid \forall v.\,\pi \\
\text{Spatial Atom} & \alpha & ::= & \mathtt{emp} \mid \tau_1 \mapsto \tau_2 \mid \mathtt{Aseg}(\tau_1, \tau_2) \mid \mathtt{Aseg}^+(\tau_1, \tau_2) \\
\text{Heap formula} & \kappa & ::= & \alpha \mid \kappa_1 * \kappa_2 \mid \kappa_1 \circledast \kappa_2 \\
\text{ASL formula} & \Theta & ::= & \pi \wedge \kappa \mid \exists v.\ \Theta
\end{array}
$$

- $a+10 \rightarrow 3$

  array elem at index 10 is 3

- $Aseg(x,y)$

  possibly empty array segment from x to (y-1)

- $Aseg^+(x,x+3)$

  non-empty array segment from x to (x+2)

Array Segment Definition

Possibly empty array segment

$Aseg(x,y) == emp \ \& \ x=y$

$\vee \ x \to \_ * Aseg(x+1,y) \ \& \ x<y$

inv $x \leq y$

Non-empty array segment

$Aseg^{+}(x,y) == x \to \_ * Aseg(x+1,y)$

inv $0 < x < y$

Two Recent Works

4. Brotherston, J., Gorogiannis, N., Kanovich, M.: Biabduction (and related problems) in array separation logic. In: CADE-26 (2017)

Negation of entailment converted into propositional formula
Limitation (i) hard to support frame inference
(ii) does not support existential content
Aseg(x,x+1) |- ex v. x $\rightarrow$ v

13. Kimura, D., Tatsuta, M.: Decision procedure for entailment of symbolic heaps with arrays. In: APLAS 2017, Suzhou, China (to appear) (2017)

Uses sorted array segments + generate condition on validity
Limitation (i) classical entailment
(ii) complex rule for disjunction
(iii) no existential size of arrays
Aseg(x,x+1) |- ex v. Aseg(a,a+v) & v>0

Contributions

Our Improvements

- No restriction on existential quantifier
- Use of both *Aseg* and *Aseg*$^+$
- Infers weakest pre-condition to entailment
- Lazy case-splitting
  - supported using both $\kappa_1 * \kappa_2 \mid \kappa_1 \circledast \kappa_2$

Semantics

$$
\begin{array}{lll}
s,h \models \pi & \texttt{iff } s \models \pi \text{ and } dom(h) = \varnothing \\
s,h \models \texttt{emp} & \texttt{iff } dom(h) = \varnothing \\
s,h \models \tau_1 \mapsto \tau_2 & \texttt{iff } dom(h) = \{s(\tau_1)\} \text{ and } h(s(\tau_1)) = s(\tau_2) \\
s,h \models \texttt{Aseg}(\tau_1,\tau_2) & \texttt{iff } s(\tau_1)=s(\tau_2) \text{ and } dom(h) = \varnothing \text{ or} \\
 & \qquad 0<s(\tau_1)<s(\tau_2) \text{ and } dom(h) = \{s(\tau_1),\ldots,s(\tau_2)-1\} \\
s,h \models \texttt{Aseg}^+(\tau_1,\tau_2) & \texttt{iff } 0<s(\tau_1)<s(\tau_2) \text{ and } dom(h) = \{s(\tau_1),\ldots,s(\tau_2)-1\} \\
s,h \models \kappa_1 * \kappa_2 & \texttt{iff } \exists h_1,h_2.\ h_1 \# h_2 \text{ and } h = h_1 \circ h_2 \text{ and } s,h_1 \models \kappa_1 \text{ and } s,h_2 \models \kappa_2 \\
s,h \models \kappa_1 \circledast \kappa_2 & \texttt{iff } \exists h_1,h_2.\ h_1 \# h_2 \text{ and } h = h_1 \circ h_2 \text{ and } s \models dom(h_1) \prec dom(h_2) \\
 & \qquad \text{and } s,h_1 \models \kappa_1 \text{ and } s,h_2 \models \kappa_2 \\
s,h \models \kappa \wedge \pi & \texttt{iff } s \models \pi \text{ and } s,h \models \kappa \\
s,h \models \exists v.\ \Theta & \texttt{iff } \exists l \in \texttt{Val}.\ s[v \mapsto l],h \models \Theta
\end{array}
$$

Array Logic to Pure Logic

- Array Logic → Pure Logic
- Important for satisfiability + entailment.
- Disjoint Heap:

$$\mathtt{inv}(\kappa[*]) = \mathtt{disj}(\kappa[*]).$$

- Sorted Heap:

$$\mathtt{inv}(\kappa[\circledast]) = \mathtt{disj}(\kappa[\circledast]) \wedge \mathtt{Sorted}(\kappa[\circledast]).$$

Examples

- $inv\ (\ Arr^{+}(a,b) * Arr^{+}(x,y)\ )$

  $(0<a<b \wedge 0<x<y) \wedge (b \leq x \vee y \leq a)$

- $inv\ (\ Arr^{+}(a,b) \circledast Arr^{+}(x,y)\ )$

  $(0<a<b \wedge 0<x<y) \wedge b \leq x$

Invariants of individual predicates

Definitions

- Disjoint Heap:

$$\mathtt{inv}(\kappa[*]) = \mathtt{disj}(\kappa[*]).$$

- Sorted Heap:

$$\mathtt{inv}(\kappa[\circledast]) = \mathtt{disj}(\kappa[\circledast]) \wedge \mathtt{Sorted}(\kappa[\circledast]).$$

Disjointness

$$
\begin{aligned}
\mathtt{disj}(\alpha) &= \mathtt{true} \\
\mathtt{disj}(\alpha_{\mathtt{k}} * \circledast_{\mathtt{i}=1}^{\mathtt{m}} \alpha_{\mathtt{i}}) &= \mathtt{disj}(\circledast_{\mathtt{i}=1}^{\mathtt{m}} \alpha_{\mathtt{i}}) \wedge \bigwedge_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{disj}'(\alpha_{\mathtt{k}}, \alpha_{\mathtt{i}}) \\
\mathtt{disj}'(\mathtt{Aseg}^{+}(\mathtt{a},\mathtt{b}), \mathtt{Aseg}^{+}(\mathtt{a}',\mathtt{b}')) &= a<b \wedge a'<b' \wedge (b \leqslant a' \vee b' \leqslant a) \\
\mathtt{disj}'(\mathtt{Aseg}^{+}(\mathtt{a},\mathtt{b}), \mathtt{Aseg}(\mathtt{a}',\mathtt{b}')) &= \begin{array}{l} (a'=b' \wedge a<b) \\ \vee\, a'<b' \wedge \mathtt{disj}'(\mathtt{Aseg}^{+}(\mathtt{a},\mathtt{b}), \mathtt{Aseg}^{+}(\mathtt{a}',\mathtt{b}')) \end{array} \\
\mathtt{disj}'(\mathtt{Aseg}(\mathtt{a},\mathtt{b}), \mathtt{Aseg}^{+}(\mathtt{a}',\mathtt{b}')) &= \mathtt{disj}'(\mathtt{Aseg}^{+}(\mathtt{a}',\mathtt{b}'), \mathtt{Aseg}(\mathtt{a},\mathtt{b})) \\
\mathtt{disj}'(\mathtt{Aseg}(\mathtt{a},\mathtt{b}), \mathtt{Aseg}(\mathtt{a}',\mathtt{b}')) &= \begin{array}{l} a=b \wedge a'=b' \\ \vee\, a<b \wedge \mathtt{disj}'(\mathtt{Aseg}^{+}(\mathtt{a},\mathtt{b}), \mathtt{Aseg}(\mathtt{a}',\mathtt{b}')) \\ \vee\, a'<b' \wedge \mathtt{disj}'(\mathtt{Aseg}(\mathtt{a},\mathtt{b}), \mathtt{Aseg}^{+}(\mathtt{a}',\mathtt{b}')) \end{array} \\
\mathtt{disj}'(\mathtt{Aseg}^{+}(\mathtt{a},\mathtt{b}), \mathtt{x} \mapsto \mathtt{v}) &= a<b \wedge (x<a \vee b \leqslant x) \\
\mathtt{disj}'(\mathtt{x} \mapsto \mathtt{v}, \mathtt{Aseg}^{+}(\mathtt{a},\mathtt{b})) &= \mathtt{disj}'(\mathtt{Aseg}^{+}(\mathtt{a},\mathtt{b}), \mathtt{x} \mapsto \mathtt{v}) \\
\mathtt{disj}'(\mathtt{Aseg}(\mathtt{a},\mathtt{b}), \mathtt{x} \mapsto \mathtt{v}) &= \mathtt{disj}'(\mathtt{Aseg}^{+}(\mathtt{a},\mathtt{b}), \mathtt{x} \mapsto \mathtt{v}) \vee a=b \\
\mathtt{disj}'(\mathtt{x} \mapsto \mathtt{v}, \mathtt{Aseg}(\mathtt{a},\mathtt{b})) &= \mathtt{disj}'(\mathtt{Aseg}(\mathtt{a},\mathtt{b}), \mathtt{x} \mapsto \mathtt{v}) \\
\mathtt{disj}'(\mathtt{x} \mapsto \mathtt{v}, \mathtt{x}' \mapsto \mathtt{v}') &= x \neq x'
\end{aligned}
$$

two predicates

Sortedness

last address encountered

$$
\begin{aligned}
\mathtt{Order}(r, \mathtt{emp}) &\equiv \mathtt{true} \\
\mathtt{Order}(r, \mathtt{emp} \circledast \kappa) &\equiv \mathtt{Order}(r, \kappa) \\
\mathtt{Order}(r, \mathtt{Aseg}(a, b) \circledast \kappa) &\equiv (a{=}b \wedge \mathtt{Order}(r, \kappa)) \vee (r{<}a{<}b \wedge \mathtt{Order}(b{-}1, \kappa)) \\
\mathtt{Order}(r, \mathtt{Aseg}^{+}(a, b) \circledast \kappa) &\equiv r{<}a{<}b \wedge \mathtt{Order}(b{-}1, \kappa) \\
\mathtt{Order}(r, x {\mapsto} \_ \circledast \kappa) &\equiv r{<}x \wedge \mathtt{Order}(x, \kappa) \\
\mathtt{Sorted}(\kappa[\circledast]) &\equiv \mathtt{Order}(0, \kappa \circledast \mathtt{emp})
\end{aligned}
$$

Existential Entailment

Existential Entailment

$$\Theta_L \vdash \exists V.\ \Theta_R \iff \forall s, h.\ s, h \models \Theta_L \rightarrow s, h \models \exists V.\ \Theta_R$$

Deriving Pre-Conditions

existential variables
for consequent

$$\Theta_L \vdash^V \Theta_R \rightsquigarrow f$$

sound & weakest
pre-condition

Base Case Scenarios

$$\frac{[\mathbf{Base\!-\!Case\!-\!1}]}{\pi \wedge \mathtt{emp} \vdash^{V} \pi' \wedge \mathtt{emp} \rightsquigarrow \pi \rightarrow \exists V.\pi'}$$

weakest pre-condition

$$\frac{\begin{gathered}[\mathbf{Base\!-\!Case\!-\!2}]\\ \kappa_1 = \mathtt{emp} \wedge \kappa_2 \neq \mathtt{emp} \;\; or \;\; \kappa_2 = \mathtt{emp} \wedge \kappa_1 \neq \mathtt{emp}\end{gathered}}{\pi \wedge \kappa_1 \vdash^{V} \pi' \wedge \kappa_2 \rightsquigarrow \pi \rightarrow \mathtt{false}}$$

contradiction
in entailment

Structural Rules

$$\frac{\Theta_1 \vdash^V \Theta' \rightsquigarrow f_1 \quad \Theta_2 \vdash^V \Theta' \rightsquigarrow f_2}{\Theta_1 \vee \Theta_2 \vdash^V \Theta' \rightsquigarrow f_1 \wedge f_2}\ [\mathbf{LHS}\ \vee]$$

$$\frac{\Theta \vdash^V \Theta_1 \rightsquigarrow f_1 \quad \Theta \vdash^V \Theta_2 \rightsquigarrow f_2}{\Theta \vdash^V \Theta_1 \vee \Theta_2 \rightsquigarrow f_1 \vee f_2}\ [\mathbf{RHS}\ \vee]$$

completeness depends on
deriving weakest pre-conditions

Structural Rules

$$\frac{[a/x]\Theta_L \vdash^V \Theta_R \rightsquigarrow [a/x]f}{\exists x{\cdot}\Theta_L \vdash^V \Theta_R \rightsquigarrow \forall x.\ f}\ [\mathbf{LHS}\ \exists]$$

$$\frac{\Theta_L \vdash^{V \cup \{x\}} \Theta_R \rightsquigarrow f}{\Theta_L \vdash^V \exists x.\Theta_R \rightsquigarrow f}\ [\mathbf{RHS}\ \exists]$$

Predicate Definitions

$$\mathtt{Aseg}(a,b) \triangleq a{=}b \wedge \mathtt{emp} \;\vee\; \mathtt{Aseg}^{+}(a,b) \wedge a{<}b$$
$$\mathtt{Aseg}^{+}(a,b) \triangleq \exists\, v, c.\; c{=}a{+}1 \wedge a{\mapsto}v \circledast \mathtt{Aseg}(c,b)$$

to support unfolding of predicates

Ordered Heap Entailment

$$[\textbf{Match } \texttt{Aseg}^{+} \textbf{ vs } \mapsto]$$

$$\frac{\begin{array}{c} fresh\ u,c \\ c{=}a{+}1 \wedge \pi \wedge a{\mapsto}u \circledast \texttt{Aseg}(c,b) \circledast \kappa \vdash^{V} \pi' \wedge x{\mapsto}v \circledast \kappa' \rightsquigarrow f \end{array}}{\pi \wedge \texttt{Aseg}^{+}(a,b) \circledast \kappa \vdash^{V} \pi' \wedge x{\mapsto}v \circledast \kappa' \rightsquigarrow \forall\, u,c{\cdot} f}$$

unfolding followed by
LHS ∃

Ordered Heap Entailment

$$\textbf{[Elim LHS } \mathtt{Aseg}\textbf{]}$$

$$\frac{a{=}b \wedge \pi \wedge \kappa \vdash^{V} \Theta \rightsquigarrow f_1 \quad a{<}b \wedge \pi \wedge \mathtt{Aseg}^{+}(a,b) \circledast \kappa \vdash^{V} \Theta \rightsquigarrow f_2}{\pi \wedge \mathtt{Aseg}(a,b) \circledast \kappa \vdash^{V} \Theta \rightsquigarrow f_1 \wedge f_2}$$

Unfolding Aseg followed by LHS $\vee$

Ordered Heap Entailment

$$\textbf{[Elim RHS } \texttt{Aseg}\textbf{]}$$

$$\frac{\Theta \vdash^{V} a{=}b \wedge \pi \wedge \kappa \rightsquigarrow f_1 \quad \Theta \vdash^{V} a{<}b \wedge \pi \wedge \texttt{Aseg}^{+}(a,b) \circledast \kappa \rightsquigarrow f_2}{\Theta \vdash^{V} \pi \wedge \texttt{Aseg}(a,b) \circledast \kappa \rightsquigarrow f_1 \vee f_2}$$

Unfolding Aseg followed by
RHS $\vee$

Structural Case Analysis

Case-Analysis Structural Rule

$$\textbf{[Case Analysis]}$$

$$\frac{U = V \cap vars(\pi) \quad \pi \wedge \Theta_L \vdash^{V-U} \Theta_R \rightsquigarrow f_1 \quad \neg\pi \wedge \Theta_L \vdash^{V-U} \Theta_R \rightsquigarrow f_2}{\Theta_L \vdash^{V} \Theta_R \rightsquigarrow \exists U.(f_1 \wedge f_2)}$$

arbitrary π leads to
just *sound* pre-condition

critical π leads to
just *weakest* pre-condition

Aligning Leftmost Predicate

[Align $\mathtt{Aseg}^+$]

$$\frac{U = \{a'\} \cap V \quad a = a' \wedge \pi \wedge \mathtt{Aseg}^+(a,b) \circledast \kappa \vdash^{V-U} \pi' \wedge \mathtt{Aseg}^+(a,b') \circledast \kappa' \leadsto f}{\pi \wedge \mathtt{Aseg}^+(a,b) \circledast \kappa \vdash^{V} \pi' \wedge \mathtt{Aseg}^+(a',b') \circledast \kappa' \leadsto \exists U \cdot (f \wedge (\pi \rightarrow a = a'))}$$

[Match $\mapsto$]

$$\frac{U = \{x', v'\} \cap V \quad x = x' \wedge v = v' \wedge \pi \wedge \kappa \vdash^{V-U} \pi' \wedge \kappa' \leadsto f}{\pi \wedge x \mapsto v \circledast \kappa \vdash^{V} \pi' \wedge x' \mapsto v' \circledast \kappa' \leadsto \exists U \cdot (f \wedge (\pi \rightarrow (x = x' \wedge v = v')))}$$

critical case analysis
a=a' $\vee$ a≠a'

Splitting Lemma

$$a < k \leqslant b \wedge \mathtt{Aseg}^+(a,b) \Leftrightarrow \mathtt{Aseg}^+(a,k) \circledast \mathtt{Aseg}(k,b)$$

Aseg^+ Lemma Splitting

Matching Aseg+ Predicates

$$\frac{[\textbf{Match } \texttt{Aseg}^+] \quad U = V \cap \{b'\} \quad a < b' \wedge b < b' \wedge \pi \wedge \kappa \vdash^{V-U} \pi' \wedge \texttt{Aseg}^+(b, b') \circledast \kappa' \leadsto f_1 \quad a < b' \leqslant b \wedge \pi \wedge \texttt{Aseg}(b', b) \circledast \kappa \vdash^{V-U} \pi' \wedge \kappa' \leadsto f_2}{\pi \wedge \texttt{Aseg}^+(a, b) \circledast \kappa \vdash^{V} \pi' \wedge \texttt{Aseg}^+(a, b') \circledast \kappa' \leadsto \exists U \cdot (f_1 \wedge f_2 \wedge (\pi \wedge a \geqslant b' \rightarrow \texttt{false}))}$$

critical case analysis
a<b’ ∧ b<b’ ∨
a<b’ ∧ b≥ b’ ∨
a≥b’

Lazily Sorted Entailment

$$\frac{\pi_1 \wedge (\bigwedge_{i=1 \wedge i \neq k}^{m} \texttt{Order}(\alpha_k, \alpha_i)) \wedge (\alpha_k \circledast *_{i=1 \wedge i \neq k}^{m} \alpha_i) \vdash^V \Theta_R \rightsquigarrow f_k, \;\; k = 1...m}{\pi_1 \wedge *_{i=1}^{m} \alpha_i \vdash^V \Theta_R \rightsquigarrow \bigwedge_{i=1}^{m} f_i} \;[\textbf{LHS}-\textbf{LazySort}]$$

$$\frac{\Theta_L \vdash^V \pi_1 \wedge (\bigwedge_{i=1 \wedge i \neq k}^{m} \texttt{Order}(\alpha_k, \alpha_i)) \wedge (\alpha_k \circledast *_{i=1 \wedge i \neq k}^{m} \alpha_i) \rightsquigarrow f_k, \;\; k = 1...m}{\Theta_L \vdash^V \pi_1 \wedge *_{i=1}^{m} \alpha_i \rightsquigarrow \bigvee_{i=1}^{m} f_i} \;[\textbf{RHS}-\textbf{LazySort}]$$

Experiments

Comparing with [K+T APLAS17]

Base				
	<0.1s	<1s	<10s	<300s
K&T	111	120	120	120
Lazy	120	120	120	120

SingleFrame-2				
	<0.1s	<1s	<10s	<300s
K&T	101	118	120	120
Partial order	120	120	120	120

SingleNFrame-2				
	<0.1s	<1s	<10s	<300s
K&T	79	119	120	120
Partial order	119	120	120	120

Multi				
	<0.1s	<1s	<10s	<300s
K&T	57	108	120	120
Lazy	116	118	120	120

SingleFrame-3				
	<0.1s	<1s	<10s	<300s
K&T	81	107	118	120
Lazy	120	120	120	120

SingleNFrame-3				
	<0.1s	<1s	<10s	<300s
K&T	34	102	119	120
Partial order	113	119	120	120

Lazy vs Eager Splitting

Program	Lazy	Eager
add_last-alloca_true-valid-memsafety.c	3.15	3.64
add_last_unsafe_false-valid-deref.c	3.20	3.51
array01-alloca_true-valid-memsafety.c	1.02	1.14
array02-alloca_true-valid-memsafety.c	1.06	1.30
array03-alloca_true-valid-memsafety.c	2.16	2.90
bubblesort-alloca_true-valid-memsafety.c	3.90	4.34
bubblesort_unsafe_false-valid-deref.c	1.27	2.08
count_down-alloca_true-valid-memsafety.c	4.95	6.78
count_down_unsafe_false-valid-deref.c	1.06	2.02

Conclusion

- Unrestricted existential entailment
- Both Aseg and $Aseg^{+}$
- Simpler disjunction rule
- Critical Case Analysis
- Lazy case-splitting
- Support for Frame and Bi-Abduction