Disproving Inductive Entailments in Separation Logic via Base Pair Approximation

James Brotherston[1] Nikos Gorogiannis[2]

[1]UCL

[2]Middlesex University

TABLEAUX'15, Wroclaw, 23 Sept 2015

Disproof, in general

- Disproof is the problem of showing that an entailment $A \vdash B$ (in some undecidable logic) is not valid.

Disproof, in general

- Disproof is the problem of showing that an entailment $A \vdash B$ (in some undecidable logic) is not valid.
- Application in proof search: backtrack from invalid subgoals.

Disproof, in general

- Disproof is the problem of showing that an entailment $A \vdash B$ (in some undecidable logic) is not valid.
- Application in proof search: backtrack from invalid subgoals.
- Application in lemma speculation and automated theory exploration: filter out invalid "lemmas".

Disproof, in general

- Disproof is the problem of showing that an entailment $A \vdash B$ (in some undecidable logic) is not valid.
- Application in proof search: backtrack from invalid subgoals.
- Application in lemma speculation and automated theory exploration: filter out invalid "lemmas".
- Precision usually costs.

Disproof, in general

- Disproof is the problem of showing that an entailment $A \vdash B$ (in some undecidable logic) is not valid.
- Application in proof search: backtrack from invalid subgoals.
- Application in lemma speculation and automated theory exploration: filter out invalid "lemmas".
- Precision usually costs.
- Our setting: symbolic-heap separation logic with inductive definitions, widely used in program verification.

Symbolic-heap separation logic

- Terms t are either variables $x, y, z \ldots$ or the constant nil.

Symbolic-heap separation logic

- Terms t are either variables $x, y, z \ldots$ or the constant nil.
- Spatial formulas F and pure formulas π given by:

$$F ::= \mathsf{emp} \mid x \mapsto \mathbf{t} \mid P\mathbf{t} \mid F * F \qquad \pi ::= t = t \mid t \neq t$$

  (where P a predicate symbol, $\mathbf{t}$ a tuple of terms).
- $\mapsto$ ("points-to") denotes an individual pointer to a record in the heap.
- $*$ ("separating conjunction") demarks domain-disjoint heaps.

Symbolic-heap separation logic

- Terms t are either variables $x, y, z \ldots$ or the constant nil.
- Spatial formulas F and pure formulas π given by:

$$F ::= \mathsf{emp} \mid x \mapsto \mathbf{t} \mid P\mathbf{t} \mid F * F \qquad \pi ::= t = t \mid t \neq t$$

  (where P a predicate symbol, $\mathbf{t}$ a tuple of terms).
- $\mapsto$ ("points-to") denotes an individual pointer to a record in the heap.
- $*$ ("separating conjunction") demarks domain-disjoint heaps.
- Symbolic heaps A given by $\exists \mathbf{x}.\ \Pi : F$, for Π a set of pure formulas.

Inductive definitions in separation logic

- Inductive predicates defined by a set of rules of form:

$$A \Rightarrow P\mathbf{t}$$

(We typically suppress the existential quantifiers in A.)

Inductive definitions in separation logic

- Inductive predicates defined by a set of rules of form:

$$A \Rightarrow P\mathbf{t}$$

  (We typically suppress the existential quantifiers in A.)

- E.g., linked list segments with root x and tail element y given by:

$$\begin{array}{rcl} \mathsf{emp} & \Rightarrow & \mathsf{ls}\, x\, x \\ x \neq \mathsf{nil} : x \mapsto z * \mathsf{ls}\, z\, y & \Rightarrow & \mathsf{ls}\, x\, y \end{array}$$

Inductive definitions in separation logic

- Inductive predicates defined by a set of rules of form:

$$A \Rightarrow P\mathbf{t}$$

(We typically suppress the existential quantifiers in A.)

- E.g., linked list segments with root x and tail element y given by:

$$\begin{array}{rcl} \mathsf{emp} & \Rightarrow & \mathsf{ls}\,x\,x \\ x \neq \mathsf{nil} : x \mapsto z * \mathsf{ls}\,z\,y & \Rightarrow & \mathsf{ls}\,x\,y \end{array}$$

- E.g., binary trees with root x given by:

$$\begin{array}{rcl} x = \mathsf{nil} : \mathsf{emp} & \Rightarrow & \mathsf{bt}\,x \\ x \neq \mathsf{nil} : x \mapsto (y,z) * \mathsf{bt}\,y * \mathsf{bt}\,z & \Rightarrow & \mathsf{bt}\,x \end{array}$$

Semantics

- Models are stacks $s : \mathsf{Var} \to \mathsf{Val}$ paired with heaps $h : \mathsf{Loc} \rightharpoonup_{\text{fin}} \mathsf{Val}$. $\circ$ is union of domain-disjoint heaps; e is the empty heap; nil is a non-allocable value.

Semantics

- Models are stacks $s : \mathsf{Var} \to \mathsf{Val}$ paired with heaps $h : \mathsf{Loc} \rightharpoonup_{\text{fin}} \mathsf{Val}$. $\circ$ is union of domain-disjoint heaps; e is the empty heap; nil is a non-allocable value.
- Forcing relation $s, h \models A$ given by

$$
\begin{array}{lcl}
s, h \models_\Phi t_1 = (\neq) t_2 & \Leftrightarrow & s(t_1) = (\neq) s(t_2) \\
s, h \models_\Phi \mathsf{emp} & \Leftrightarrow & h = e \\
s, h \models_\Phi x \mapsto \mathbf{t} & \Leftrightarrow & \mathrm{dom}(h) = \{s(x)\} \text{ and } h(s(x)) = s(\mathbf{t}) \\
s, h \models_\Phi P_i \mathbf{t} & \Leftrightarrow & (s(\mathbf{t}), h) \in [\![P_i]\!]^\Phi \\
s, h \models_\Phi F_1 * F_2 & \Leftrightarrow & \exists h_1, h_2.\ h = h_1 \circ h_2 \text{ and } s, h_1 \models_\Phi F_1 \\
 & & \text{and } s, h_2 \models_\Phi F_2 \\
s, h \models_\Phi \exists \mathbf{z}.\ \Pi : F & \Leftrightarrow & \exists \mathbf{v} \in \mathsf{Val}^{|\mathbf{z}|}.\ s[\mathbf{z} \mapsto \mathbf{v}], h \models_\Phi \pi \text{ for all} \\
 & & \pi \in \Pi \text{ and } s[\mathbf{z} \mapsto \mathbf{v}], h \models_\Phi F
\end{array}
$$

Disproof in our logic

- Entailment is here undecidable [Antoupoulos et al., FOSSACS'14], although satisfiability is decidable [Brotherston et al., CSL-LICS'14].

Disproof in our logic

- Entailment is here undecidable [Antoupoulos et al., FOSSACS'14], although satisfiability is decidable [Brotherston et al., CSL-LICS'14].
- To disprove $A \vdash B$, we need a *countermodel* (s, h) s.t. $s, h \models_\Phi A$ but $s, h \not\models_\Phi B$.

Disproof in our logic

- Entailment is here undecidable [Antoupoulos et al., FOSSACS'14], although satisfiability is decidable [Brotherston et al., CSL-LICS'14].
- To disprove $A \vdash B$, we need a *countermodel* (s, h) s.t. $s, h \models_\Phi A$ but $s, h \not\models_\Phi B$.
- Model checking has only very recently been shown decidable, in fact EXPTIME-complete [Brotherston et al., submitted, 2015].

Disproof in our logic

- Entailment is here undecidable [Antoupoulos et al., FOSSACS'14], although satisfiability is decidable [Brotherston et al., CSL-LICS'14].
- To disprove $A \vdash B$, we need a *countermodel* (s, h) s.t. $s, h \models_\Phi A$ but $s, h \not\models_\Phi B$.
- Model checking has only very recently been shown decidable, in fact EXPTIME-complete [Brotherston et al., submitted, 2015].
- Enumerating and checking all possible counter-models is complete, but complicated and, I suspect, ridiculously expensive.

Base pairs [Brotherston et al., CSL-LICS'14]

- For any symbolic heap A, we can compute an overapproximation, $base^{\Phi}(A)$.

Base pairs [Brotherston et al., CSL-LICS'14]

- For any symbolic heap A, we can compute an overapproximation, $base^{\Phi}(A)$. Each "base pair" records, for each possible way of constructing a model of A,
    1. the variables in $FV(A)$ that must be allocated, and
    2. the (dis)equalities over $FV(A) \cup \{\mathsf{nil}\}$ that must hold.

Base pairs [Brotherston et al., CSL-LICS'14]

- For any symbolic heap A, we can compute an overapproximation, $base^{\Phi}(A)$. Each "base pair" records, for each possible way of constructing a model of A,
  1. the variables in $FV(A)$ that must be allocated, and
  2. the (dis)equalities over $FV(A) \cup \{\mathsf{nil}\}$ that must hold.

- E.g., recall linked list segment predicate `ls`:

$$
\begin{array}{rcl}
\mathsf{emp} & \Rightarrow & \mathsf{ls}\,x\,x \\
x \neq \mathsf{nil} : x \mapsto z * \mathsf{ls}\,z\,y & \Rightarrow & \mathsf{ls}\,x\,y
\end{array}
$$

Base pairs [Brotherston et al., CSL-LICS'14]

- For any symbolic heap A, we can compute an overapproximation, $base^{\Phi}(A)$. Each "base pair" records, for each possible way of constructing a model of A,
  1. the variables in $FV(A)$ that must be allocated, and
  2. the (dis)equalities over $FV(A) \cup \{\mathsf{nil}\}$ that must hold.

- E.g., recall linked list segment predicate `ls`:

$$\begin{array}{rcl} \mathsf{emp} & \Rightarrow & \mathsf{ls}\, x\, x \\ x \neq \mathsf{nil} : x \mapsto z * \mathsf{ls}\, z\, y & \Rightarrow & \mathsf{ls}\, x\, y \end{array}$$

  We obtain two base pairs:

$$\begin{array}{rcl} base^{\Phi}(\mathsf{ls}\, x\, y) & = & \{(\emptyset, \{x = y\}), \\ & & (\{x\}, \{x \neq \mathsf{nil}\})\} \end{array}$$

Connecting base pairs and models

- Base pairs are formally related to models as follows.

Connecting base pairs and models

- Base pairs are formally related to models as follows.

Lemma (1)

Given $(V, \Pi) \in base^{\Phi}(A)$*, a stack* s *s.t.* $s \models \Pi$*, and finite set* $W \subset \mathsf{Loc} \setminus s(V)$*, then* $\exists h.\ s, h \models_{\Phi} A$ *and* $W \cap \mathrm{dom}(h) = \emptyset$*.*

Connecting base pairs and models

- Base pairs are formally related to models as follows.

Lemma (1)

Given $(V, \Pi) \in \mathit{base}^{\Phi}(A)$*, a stack* s *s.t.* $s \models \Pi$*, and finite set* $W \subset \mathsf{Loc} \setminus s(V)$*, then* $\exists h.\ s, h \models_{\Phi} A$ *and* $W \cap \mathrm{dom}(h) = \emptyset$*.*

Lemma (2)

If $s, h \models_{\Phi} B$*, there is a base pair* $(V, \Pi) \in \mathit{base}^{\Phi}(B)$ *such that* $s(V) \subseteq \mathrm{dom}(h)$ *and* $s \models \Pi$*.*

Connecting base pairs and models

- Base pairs are formally related to models as follows.

Lemma (1)

Given $(V, \Pi) \in \mathit{base}^{\Phi}(A)$*, a stack* s *s.t.* $s \models \Pi$*, and finite set* $W \subset \mathsf{Loc} \setminus s(V)$*, then* $\exists h.\ s, h \models_{\Phi} A$ *and* $W \cap \mathrm{dom}(h) = \emptyset$*.*

Lemma (2)

If $s, h \models_{\Phi} B$*, there is a base pair* $(V, \Pi) \in \mathit{base}^{\Phi}(B)$ *such that* $s(V) \subseteq \mathrm{dom}(h)$ *and* $s \models \Pi$*.*

- Consequently, we can use Lemma 1 to construct a model of A and then Lemma 2 to show it cannot be a model of B.

Disproof "game"

Game (1)

- *Given* $A \vdash B$. *a move by Player 1 is a choice of:*
  - *a base pair* $(X, \Pi) \in base^{\Phi}(A)$;
  - *a stack* s *such that* $s \models \Pi$; *and*
  - *a finite set* $W \subset \mathsf{Loc} \setminus s(X)$.

Disproof "game"

Game (1)

- *Given* $A \vdash B$. *a move by Player 1 is a choice of:*
  - *a base pair* $(X, \Pi) \in base^{\Phi}(A)$*;*
  - *a stack* s *such that* $s \models \Pi$*; and*
  - *a finite set* $W \subset \mathsf{Loc} \setminus s(X)$.
- *A response by Player 2 is a base pair* $(Y, \Theta) \in base^{\Phi}(B)$ *such that* $s \models \Theta$ *and* $W \cap s(Y) = \emptyset$.

Disproof "game"

Game (1)

- *Given* $A \vdash B$. *a move by Player 1 is a choice of:*
  - *a base pair* $(X, \Pi) \in \mathit{base}^{\Phi}(A)$;
  - *a stack* s *such that* $s \models \Pi$; *and*
  - *a finite set* $W \subset \mathsf{Loc} \setminus s(X)$.
- *A response by Player 2 is a base pair* $(Y, \Theta) \in \mathit{base}^{\Phi}(B)$ *such that* $s \models \Theta$ *and* $W \cap s(Y) = \emptyset$.
- *A move is winning if there is no possible response.*

Disproof "game"

Game (1)

- *Given* $A \vdash B$. *a move by Player 1 is a choice of:*
    - *a base pair* $(X, \Pi) \in base^{\Phi}(A)$*;*
    - *a stack* s *such that* $s \models \Pi$*; and*
    - *a finite set* $W \subset \mathsf{Loc} \setminus s(X)$.
- *A response by Player 2 is a base pair* $(Y, \Theta) \in base^{\Phi}(B)$ *such that* $s \models \Theta$ *and* $W \cap s(Y) = \emptyset$.
- *A move is winning if there is no possible response.*

Proposition

If Player 1 has a winning move for $A \vdash B$ *then it is invalid.*

Refined disproof "game"

Game (2)

- *Given* $A \vdash B$*, a move by Player 1 is a choice of:*
  - *a base pair* $(X, \Pi) \in base^{\Phi}(A)$*, and*
  - *a partition* σ *of* $FV(A) \cup FV(B) \cup \{\mathsf{nil}\}$ *s.t.* $\sigma \models \Pi$*.*

Refined disproof "game"

Game (2)

- *Given* $A \vdash B$, *a* *move* *by Player 1 is a choice of:*
  - *a base pair* $(X, \Pi) \in base^{\Phi}(A)$, *and*
  - *a* *partition* σ *of* $FV(A) \cup FV(B) \cup \{\mathsf{nil}\}$ *s.t.* $\sigma \models \Pi$.
- *A* *response* *by Player 2 is a base pair* $(Y, \Theta) \in base^{\Phi}(B)$ *such that* $\sigma \models \Theta$ *and* $\forall y \in Y \setminus X.\ \exists x \in X.\ y \equiv_{\sigma} x$.

Refined disproof "game"

Game (2)

- *Given $A \vdash B$, a move by Player 1 is a choice of:*
  - *a base pair $(X, \Pi) \in base^{\Phi}(A)$, and*
  - *a partition σ of $FV(A) \cup FV(B) \cup \{\mathsf{nil}\}$ s.t. $\sigma \models \Pi$.*
- *A response by Player 2 is a base pair $(Y, \Theta) \in base^{\Phi}(B)$ such that $\sigma \models \Theta$ and $\forall y \in Y \setminus X.\ \exists x \in X.\ y \equiv_{\sigma} x$.*
- *A winning move is (still) a move with no response.*

Refined disproof "game"

Game (2)

- *Given $A \vdash B$, a move by Player 1 is a choice of:*
  - *a base pair $(X, \Pi) \in base^{\Phi}(A)$, and*
  - *a partition σ of $FV(A) \cup FV(B) \cup \{\mathsf{nil}\}$ s.t. $\sigma \models \Pi$.*
- *A response by Player 2 is a base pair $(Y, \Theta) \in base^{\Phi}(B)$ such that $\sigma \models \Theta$ and $\forall y \in Y \setminus X.\ \exists x \in X.\ y \equiv_{\sigma} x$.*
- *A winning move is (still) a move with no response.*

Theorem

Games 1 and 2 are equivalent, and decidable.

An example

- Consider $\mathsf{bt}\,x \vdash \mathsf{ls}\,x\,y$ (invalid).

An example

- Consider $\mathsf{bt}\,x \vdash \mathsf{ls}\,x\,y$ (invalid).
- We have base pair approximations:

$$
\begin{array}{rcl}
base^{\Phi}(\mathsf{bt}\,x) & = & \{(\emptyset, \{x = \mathsf{nil}\}), (\{x\}, \{x \neq \mathsf{nil}\})\} \\
base^{\Phi}(\mathsf{ls}\,x\,y) & = & \{(\emptyset, \{x = y\}), (\{x\}, \{x \neq \mathsf{nil}\})\}
\end{array}
$$

An example

- Consider $\mathsf{bt}\,x \vdash \mathsf{ls}\,x\,y$ (invalid).
- We have base pair approximations:

$$\begin{aligned} base^{\Phi}(\mathsf{bt}\,x) &= \{(\emptyset, \{x = \mathsf{nil}\}), (\{x\}, \{x \neq \mathsf{nil}\})\} \\ base^{\Phi}(\mathsf{ls}\,x\,y) &= \{(\emptyset, \{x = y\}), (\{x\}, \{x \neq \mathsf{nil}\})\} \end{aligned}$$

- Winning move: choose base pair $(\emptyset, \{x = \mathsf{nil}\})$ and any partition σ s.t. $x \equiv_\sigma \mathsf{nil}$ and $x \not\equiv_\sigma y$.

An example

- Consider $\mathsf{bt}\,x \vdash \mathsf{ls}\,x\,y$ (invalid).
- We have base pair approximations:

$$\begin{array}{rcl} base^{\Phi}(\mathsf{bt}\,x) & = & \{(\emptyset, \{x = \mathsf{nil}\}), (\{x\}, \{x \neq \mathsf{nil}\})\} \\ base^{\Phi}(\mathsf{ls}\,x\,y) & = & \{(\emptyset, \{x = y\}), (\{x\}, \{x \neq \mathsf{nil}\})\} \end{array}$$

- Winning move: choose base pair $(\emptyset, \{x = \mathsf{nil}\})$ and any partition σ s.t. $x \equiv_\sigma \mathsf{nil}$ and $x \not\equiv_\sigma y$.
- Now consider $\mathsf{ls}\,x\,y \vdash \mathsf{bt}\,x$ (also invalid).

An example

- Consider $\mathsf{bt}\,x \vdash \mathsf{ls}\,x\,y$ (invalid).
- We have base pair approximations:

$$\begin{array}{rcl} base^{\Phi}(\mathsf{bt}\,x) & = & \{(\emptyset, \{x = \mathsf{nil}\}), (\{x\}, \{x \neq \mathsf{nil}\})\} \\ base^{\Phi}(\mathsf{ls}\,x\,y) & = & \{(\emptyset, \{x = y\}), (\{x\}, \{x \neq \mathsf{nil}\})\} \end{array}$$

- Winning move: choose base pair $(\emptyset, \{x = \mathsf{nil}\})$ and any partition σ s.t. $x \equiv_\sigma \mathsf{nil}$ and $x \not\equiv_\sigma y$.
- Now consider $\mathsf{ls}\,x\,y \vdash \mathsf{bt}\,x$ (also invalid).
- Winning move: choose base pair $(\emptyset, \{x = y\})$ and any partition σ s.t. $x \equiv_\sigma y$ and $x \not\equiv_\sigma \mathsf{nil}$.

Limitations

- Our method is terminating and therefore incomplete.

Limitations

- Our method is terminating and therefore incomplete.
- Most importantly, our base pair overapproximations are essentially projections onto the free variables of entailments.

Limitations

- Our method is terminating and therefore incomplete.
- Most importantly, our base pair overapproximations are essentially projections onto the free variables of entailments.
- E.g., the entailment $x \mapsto \mathsf{nil} \vdash \mathsf{emp}$ is invalid, while $x \mapsto \mathsf{nil} \vdash \exists y.\ y \mapsto \mathsf{nil}$ is valid but, since neither RHS has any free variables,

$$base^{\Phi}(\mathsf{emp}) = base^{\Phi}(\exists y.\ y \mapsto \mathsf{nil}) = \{(\emptyset, \emptyset)\}$$

so we can't distinguish the two entailments.

Experimental evaluation (1)

- We generated entailments of the form $P\mathbf{x} \vdash Q\mathbf{y}$, where
  - P and Q are inductive predicates taken from pre-existing benchmarks in SL-COMP competition (63 predicates total);
  - $\mathbf{x}$ is a tuple of distinct variables;
  - all variables in $\mathbf{y}$ appear in $\mathbf{x}$.

Experimental evaluation (1)

- We generated entailments of the form $P\mathbf{x} \vdash Q\mathbf{y}$, where
  - P and Q are inductive predicates taken from pre-existing benchmarks in SL-COMP competition (63 predicates total);
  - $\mathbf{x}$ is a tuple of distinct variables;
  - all variables in $\mathbf{y}$ appear in $\mathbf{x}$.
- This is typical of automated theory exploration. We get 818988 entailments; most will be invalid.

Experimental evaluation (1)

- We generated entailments of the form $P\mathbf{x} \vdash Q\mathbf{y}$, where
  - P and Q are inductive predicates taken from pre-existing benchmarks in SL-COMP competition (63 predicates total);
  - $\mathbf{x}$ is a tuple of distinct variables;
  - all variables in $\mathbf{y}$ appear in $\mathbf{x}$.
- This is typical of automated theory exploration. We get 818988 entailments; most will be invalid.
- Our technique disproves $> 97\%$ of the entailments in the test set, taking at most 30ms for each.

Experimental evaluation (1)

- We generated entailments of the form $P\mathbf{x} \vdash Q\mathbf{y}$, where
  - P and Q are inductive predicates taken from pre-existing benchmarks in SL-COMP competition (63 predicates total);
  - $\mathbf{x}$ is a tuple of distinct variables;
  - all variables in $\mathbf{y}$ appear in $\mathbf{x}$.
- This is typical of automated theory exploration. We get 818988 entailments; most will be invalid.
- Our technique disproves $> 97\%$ of the entailments in the test set, taking at most 30ms for each.
- Of the remainder, we could prove about 250 valid.

Experimental evaluation (2)

- SLL test suite (from SL-COMP competition) considers entailments over acyclic list segments only:

$$\begin{array}{rcl} \mathsf{emp} & \Rightarrow & \mathsf{als}\, x\, x \\ x \neq \mathsf{nil}, x \neq y : x \mapsto z * \mathsf{als}\, z\, y & \Rightarrow & \mathsf{als}\, x\, y \end{array}$$

Experimental evaluation (2)

- SLL test suite (from SL-COMP competition) considers entailments over acyclic list segments only:

$$\begin{array}{rcl} \mathsf{emp} & \Rightarrow & \mathsf{als}\,x\,x \\ x \neq \mathsf{nil}, x \neq y : x \mapsto z * \mathsf{als}\,z\,y & \Rightarrow & \mathsf{als}\,x\,y \end{array}$$

- Here, of 120 invalid sequents, we disprove only about 24%.

Experimental evaluation (2)

- SLL test suite (from SL-COMP competition) considers entailments over acyclic list segments only:

$$\begin{array}{rcl} \mathsf{emp} & \Rightarrow & \mathsf{als}\,x\,x \\ x \neq \mathsf{nil}, x \neq y : x \mapsto z * \mathsf{als}\,z\,y & \Rightarrow & \mathsf{als}\,x\,y \end{array}$$

- Here, of 120 invalid sequents, we disprove only about 24%.
- So we do (much) better in some situations than others.
- In fact this sub-fragment is polynomially decidable anyway.

Conclusions / future work

- We give a method for entailment disproof in separation logic with user-defined inductive predicates.
- Our method is incomplete, but terminating, and pretty cheeeap.
- Therefore, potentially useful for proof search and automated theory exploration.
- Future work: develop more precise disproving techniques (e.g., by direct countermodel generation).

Thanks for listening!

Try our techniques within the Cyclist distribution:

`github.com/ngorogiannis/cyclist`