Size Does Matter: Two Certified Abstractions for Disproving Entailment between Separation Logic Formulas

François Bobot*  Clément Hurlin‡  Alexander Summers‡

* INRIA Saclay – Île-de-France, France
‡ INRIA Sophia Antipolis – Méditerranée, France
Twente University, The Netherlands

‡‡ Imperial College London

January 8th 2009
Motivation

- Disprove entailment between formulas

\[ \rightarrow \text{I.e. to prove } A \not\models B \]

- \( A \) and \( B \) are *separation logic* formulas.
Motivation

- Disprove entailment between formulas
  - I.e. to prove $A \not\models B$
- $A$ and $B$ are separation logic formulas.

Technique:
- By *discriminating models* of $A$ and $B
Separation Logic: $a \xrightarrow{\pi} v$

$a \xrightarrow{\pi} v$ (called “points-to predicate”) has a dual meaning:
- Address $a$ contains value $v$.
- Permission $\pi$ to access address $a$.

$\pi$ is a fraction in $(0, 1]$:
- 1 is the permission to write access a location.
- Any $0 < \pi < 1$ is the permission to read-only access a location.
Separation Logic: ⋆

$A \star B$ is the *separating conjunction*:

- Permissions to access heap $A$ and heap $B$
- $A \star A$ does not imply $A$ (*no weakening*).
- But $A$ does not imply $A \star A$ (*no copying*).
- $\star$ *separates* permissions.
Separation Logic: ★

$A \star B$ is the *separating conjunction*:

- Permissions to access heap $A$ and heap $B$
- $A \star A$ does not imply $A$ (no weakening).
- But $A$ does not imply $A \star A$ (no copying).
- ★ *separates* permissions.

Last item means:

- $a \neq b$: $a \xrightarrow{1} \_ \star b \xrightarrow{1} \_ \quad \checkmark$
- But: $a \xrightarrow{1} \_ \star a \xrightarrow{\pi} \_ \quad \times$
Separation Logic: ★

Two axioms:

\[ a \overset{\pi}{\leftrightarrow} v \Rightarrow a \overset{\pi}{\leftrightarrow} v \star a \overset{\pi}{\leftrightarrow} v \quad \text{(Split)} \]

\[ a \overset{\pi}{\leftrightarrow} v \star a \overset{\pi}{\leftrightarrow} v \Rightarrow a \overset{\pi}{\leftrightarrow} v \quad \text{(Merge)} \]
Separation Logic: \( \Rightarrow \)

\( A \Rightarrow B \) is the *linear implication* (or “baguette magique”):

- Reads “consume A yielding B” or “trade A and receive B”
- \( A \Rightarrow (A \Rightarrow B) \) implies B
Semantics: $\mathcal{M} \models A$

- Models $\mathcal{M}$ are lists of couples of an address and a permission.
- An example model is $(245, \frac{1}{2}) :: (246, 1) :: (245, \frac{1}{3}) :: []$. 
Semantics: $\mathcal{M} \models A$

- Models $\mathcal{M}$ are lists of couples of an address and a permission.
- An example model is $(245, \frac{1}{2}) :: (246, 1) :: (245, \frac{1}{3}) :: []$.

$$\mathcal{M} \models a \leftarrow \pi \quad \text{iff} \quad \mathcal{M} = (a, \pi) :: []$$

$$\mathcal{M} \models A \star B \quad \text{iff} \quad \exists \mathcal{M}_A, \mathcal{M}_B, \mathcal{M} = \mathcal{M}_A \uplus \mathcal{M}_B, \text{ and }$$

$$\mathcal{M}_A \models A \text{ and } \mathcal{M}_B \models B$$

$$\mathcal{M} \models A \rightarrow B \quad \text{iff} \quad \forall \mathcal{M}_A, \mathcal{M}_A \models A \text{ and } \mathcal{M}_A \cap \mathcal{M} = \emptyset$$

implies $\mathcal{M}_A \uplus \mathcal{M} \models A \star B$
Semantics: $\mathcal{M} \models A$

$\mathcal{M} \models a \xrightarrow{\pi} \_ \iff \mathcal{M} = (a, \pi)$

$\mathcal{M} \models A \star B \iff \exists \mathcal{M}_A, \mathcal{M}_B, \mathcal{M} = \mathcal{M}_A \uplus \mathcal{M}_B$, and $\mathcal{M}_A \models A$ and $\mathcal{M}_B \models B$

$\mathcal{M} \models A \rightarrow B \iff \forall \mathcal{M}_A, \mathcal{M}_A \models A$ and $\mathcal{M}_A \cap \mathcal{M} = \emptyset$ implies $\mathcal{M}_A \uplus \mathcal{M} \models A \star B$

$\mathcal{M} \models A \land B \iff \mathcal{M} \models A$ and $\mathcal{M} \models B$

$\mathcal{M} \models A \lor B \iff \mathcal{M} \models A$ or $\mathcal{M} \models B$
Disproving Technique

Soundness of the proof system:

\[
A \vdash B \text{ implies } (\forall M, M \models A \rightarrow M \models B)
\]
Disproving Technique

Soundness of the proof system:

\[ A \vdash B \text{ implies } (\forall M, M \models A \rightarrow M \models B) \]

Contraposition:

\[ (\exists M, M \models A \land \neg M \models B) \text{ implies } A \not\vdash B \]

Goal of this work:

- Take \( A \) and \( B \) and prove that \( A \not\vdash B \)
- By discriminating models of \( A \) and \( B \)
Disproving Technique

Contraposition:

\[(\exists M, M \models A \land \neg M \models B) \text{ implies } A \not\models B\]

Objective:

Find \( M \) such that \( M \models A \) and \( \neg M \models B \)
Disproving Technique

Objective:

Find $\mathcal{M}$ such that $\mathcal{M} \models A$ and $\neg \mathcal{M} \models B$

To do that:

- We compute bounds on the size of models.
- $\max : \text{Formula} \rightarrow \mathbb{S}$
  ($\mathbb{S}$ is the set of sizes)
- $\min : \text{Formula} \rightarrow \mathbb{S}$
- $\text{size} : \text{Model} \rightarrow \mathbb{S}$

Properties of $\max$ and $\min$:

$\forall \mathcal{M}, \mathcal{M} \models A$ implies $\min(A) \leq \text{size}(\mathcal{M}) \leq \max(A)$
Disproving Technique

\[(\exists \mathcal{M}, \overline{\mathcal{M}} \models A \land \neg \mathcal{M} \models B) \implies A \not\models B\]

\[\forall \mathcal{M}, \overline{\mathcal{M}} \models A \implies \min(A) \leq \text{size}(\mathcal{M}) \leq \max(A)\]

\[\downarrow\]

\[\max(A) < \min(B) \implies A \not\models B\]
Disproving Technique

\((\exists \mathcal{M}, \mathcal{M} \models A \land \neg \mathcal{M} \models B) \implies A \not\models B\)

\(\forall \mathcal{M}, \mathcal{M} \models A \implies \min(A) \leq \text{size}(\mathcal{M}) \leq \max(A)\)

\[\downarrow\]

\(\max(A) < \min(B) \implies A \not\models B\)
Defining size (1)

- \( \text{size}(\mathcal{M}) \triangleq \text{sum of } \mathcal{M}'s \text{ permissions} \)
- \( \text{size}: \text{Model} \to \mathbb{Q} \)
Defining size (1)

- \( \text{size}(\mathcal{M}) \overset{\Delta}{=} \text{sum of } \mathcal{M} \text{'s permissions} \)
- \( \text{size}: \text{Model} \rightarrow \mathbb{Q} \)

\[
\text{size}((245, \frac{1}{2}) :: (246, 1) :: (245, \frac{1}{3}) :: []) = \frac{1}{2} + 1 + \frac{1}{3} = \frac{11}{6}
\]
Defining max/min (1)

\[
\begin{align*}
\max(\_ \overset{\pi}{\to} \_) &= \pi \\
\max(A \star B) &= \max(A) + \_ Q \max(B) \\
\min(\_ \overset{\pi}{\to} \_) &= \pi \\
\min(A \star B) &= \min(A) + \_ Q \min(B)
\end{align*}
\]

\[
\begin{align*}
\mathcal{M} \models a \overset{\pi}{\to} \_ \text{ iff } \mathcal{M} = (a, \pi) \\
\mathcal{M} \models A \star B \text{ iff } \exists \mathcal{M}_A, \mathcal{M}_B, \mathcal{M} = \mathcal{M}_A \uplus \mathcal{M}_B, \mathcal{M}_A \models A \text{ and } \mathcal{M}_B \models B
\end{align*}
\]
Defining max/min (1)

\[
\begin{align*}
\text{max}(\_ \mapsto \_ \pi) &= \pi \\
\text{max}(A \star B) &= \text{max}(A) + \mathbb{Q} \text{max}(B) \\
\text{max}(A \rightarrow B) &= \text{max}(B) - \mathbb{Q} \text{min}(A)
\end{align*}
\]

\[
\begin{align*}
\text{min}(\_ \mapsto \_ \pi) &= \pi \\
\text{min}(A \star B) &= \text{min}(A) + \mathbb{Q} \text{min}(B) \\
\text{min}(A \rightarrow B) &= \text{min}(B) - \mathbb{Q} \text{max}(A)
\end{align*}
\]

\[
\begin{align*}
\mathcal{M} \models o \mapsto \_ & \iff \mathcal{M} = (o, \pi) \\
\mathcal{M} \models A \star B & \iff \exists \mathcal{M}_A, \mathcal{M}_B, \mathcal{M} = \mathcal{M}_A \uplus \mathcal{M}_B, \mathcal{M}_A \models A \text{ and } \mathcal{M}_B \models B \\
\mathcal{M} \models A \rightarrow B & \iff \forall \mathcal{M}_A, \mathcal{M}_A \models A \text{ and } \mathcal{M}_A \cap \mathcal{M} = \emptyset \\
& \quad \text{implies } \mathcal{M}_A \uplus \mathcal{M} \models A \star B
\end{align*}
\]
Defining max/min (1)

\[
\begin{align*}
\max(A \land B) &= \min_\mathbb{Q}(\max(A), \max(B)) & \min(A \land B) &= \max_\mathbb{Q}(\min(A), \min(B)) \\
\max(A \lor B) &= \max_\mathbb{Q}(\max(A), \max(B)) & \min(A \lor B) &= \min_\mathbb{Q}(\min(A), \min(B))
\end{align*}
\]

\[
\begin{align*}
\mathcal{M} \models A \land B & \text{ iff } \mathcal{M} \models A \text{ and } \mathcal{M} \models B \\
\mathcal{M} \models A \lor B & \text{ iff } \mathcal{M} \models A \text{ or } \mathcal{M} \models B
\end{align*}
\]
Demo
Demo

\[
0 \overset{\frac{1}{2}}{\rightarrow} _* 0 \overset{\frac{1}{4}}{\rightarrow} _* 1 \overset{?}{\rightarrow} 0 \overset{1}{\rightarrow} _
\]

\[
0 \overset{\frac{1}{2}}{\rightarrow} _* 0 \overset{\frac{1}{4}}{\rightarrow} _* 2 \overset{\frac{1}{4}}{\rightarrow} _* 3 \overset{1}{\rightarrow} _* 1 \overset{?}{\rightarrow} ((0 \overset{\frac{1}{2}}{\rightarrow} _* 1 \overset{\frac{1}{2}}{\rightarrow} _) \wedge (1 \overset{\frac{1}{2}}{\rightarrow} _* 0 \overset{1}{\rightarrow} _)) \overset{3}{\rightarrow} _
\]
Refinement and Extension

Previously:

- Whole heap abstraction
  \[
  \text{size}((245, \frac{1}{2}) :: (246, 1) :: (245, \frac{1}{3}) :: []) = \frac{1}{2} + 1 + \frac{1}{3} = \frac{11}{6}
  \]
- Information on different addresses is lost.
Refinement and Extension

Previously:

- Whole heap abstraction

  \[
  \text{size}((245, \frac{1}{2}) :: (246, 1) :: (245, \frac{1}{3}) :: []) = \frac{1}{2} + 1 + \frac{1}{3} = \frac{11}{6}
  \]

  Information on different addresses is lost.

Next slides:

- Per address abstraction.
- Pure formulas

  Semantics of pure formulas is permission-independent.
Per Address Abstraction

Previously:

- \( \text{max} : \text{Formula} \rightarrow \mathbb{Q} \)
- \( \text{min} : \text{Formula} \rightarrow \mathbb{Q} \)
- \( \text{max}(A) < \text{min}(B) \) where \(<\) is on \(\mathbb{Q}\).

Now:

- \( \text{max} : \text{Formula} \rightarrow \text{Model} \)
- \( \text{min} : \text{Formula} \rightarrow \text{Model} \)
- \( \text{max}(A) < \text{min}(B) \) where \(<\) is on \text{Model}.\)
Defining max/min (2)

Previously:
\[
\begin{align*}
\max(_{\pi \mapsto _{}}) &= \pi \\
\max(A \star B) &= \max(A) + \mathbb{Q}\max(B) \\
\min(_{\pi \mapsto _{}}) &= \pi \\
\min(A \star B) &= \min(A) + \mathbb{Q}\min(B)
\end{align*}
\]

Now:
\[
\begin{align*}
\max(a \mapsto _{\pi}) &= (a, \pi) :: [] \\
\max(A \star B) &= \max(A) @ \max(B) \\
\min(a \mapsto _{\pi}) &= (a, \pi) :: [] \\
\min(A \star B) &= \min(A) @ \min(B)
\end{align*}
\]
Defining max/min (2)

Previously:
\[
\begin{align*}
\max(A \land B) &= \min_Q (\max(A), \max(B)) & \min(A \land B) &= \max_Q (\min(A), \min(B)) \\
\max(A \lor B) &= \max_Q (\max(A), \max(B)) & \min(A \lor B) &= \min_Q (\min(A), \min(B))
\end{align*}
\]

Now:
\[
\begin{align*}
\max(A \land B) &= \min_M (\max(A), \max(B)) & \min(A \land B) &= \max_M (\min(A), \min(B)) \\
\max(A \lor B) &= \max_M (\max(A), \max(B)) & \min(A \lor B) &= \min_M (\min(A), \min(B))
\end{align*}
\]

- \(\max_M\): Per address maximum
- \(\min_M\): Per address minimum
Defining max/min (2)

\[
\begin{align*}
\max(A \land B) &= \min_M (\max(A), \max(B)) \\
\min(A \land B) &= \max_M (\min(A), \min(B)) \\
\max(A \lor B) &= \max_M (\max(A), \max(B)) \\
\min(A \lor B) &= \min_M (\min(A), \min(B))
\end{align*}
\]

- \(\max_M\): Per address maximum
- \(\min_M\): Per address minimum

\[
\max( (245, \frac{1}{2}) :: (245, \frac{1}{2}) :: [] , (245, \frac{1}{2}) :: (246, 1) :: [] ) = (245, \frac{1}{2}) :: (245, \frac{1}{2}) :: (246, 1) :: []
\]
Pure Formulas

Pure formulas include:

- **Address comparison:** $a = a'$, $a \neq a'$.
  - With arithmetic: $a + a' = b$.

- ...
Pure Formulas

Pure formulas include:

- Address comparison: \( a = a', a \neq a' \).
- With arithmetic: \( a + a' = b \).
- …

Semantics of a pure formula \( A^p \):

\[ \mathcal{M} \models A^p \iff \text{oracle}(A^p) \]

- No size constraint on \( \mathcal{M} \)
Pure Formulas

\[ \mathcal{M} \models A^p \iff \text{oracle}(A^p) \]

- No size constraint on \( \mathcal{M} \)
- We add \( \top \) in max/min’s range.
- \( \max(A) = \top \): A’s models cannot be max-bounded.

\[
\max(A^p) = \top \quad \min(A^p) = \emptyset
\]
A^p$ a subformula of $B$ does not imply $\max(B) = \top$ (see case $\land$).

$$\max(A \star B) = \begin{cases} 
\top & \text{iff } A = \top \text{ or } B = \top \\
\max(A) \land \max(B) & \text{otherwise}
\end{cases}$$

$$\max(A \land B) = \begin{cases} 
\top & \text{iff } A = \top \text{ and } B = \top \\
\max(A) & \text{if } B = \top \\
\max(B) & \text{if } A = \top \\
\min_{\mathcal{M}}(\max(A), \max(B)) & \text{otherwise}
\end{cases}$$
Conclusion

- Lightweight method for disproving entailment for an undecidable fragment of separation logic
- Two different abstractions of different precision
- Certified with Coq
Conclusion

- Lightweight method for disproving entailment for an undecidable fragment of separation logic
- Two different abstractions of different precision
- Certified with Coq

- Deal with fractional permissions (this talk)
- Deal with counting permissions (work in progress)
Future Work

1. Unified model of permissions (fractional + counting)
2. Intuitionistic flavor of separation logic
3. Extend the mechanical proof to quantifiers
4. Abstraction mechanisms (Parkinson’s abstract predicates)