Predator Hunting Party

Michal Kotoun    Petr Peringer    Tomáš Vojnar
Veronika Šoková

FIT, Brno University of Technology, Czech Republic
Predator: An Overview

- Focuses on **shape analysis** of **low-level system code**.
Predator: An Overview

- Focuses on **shape analysis** of low-level system code.
- Uses **symbolic memory graphs** (SMGs) to encode sets of heap configurations with various kinds of (nested) lists:

```plaintext
DLS 2+ level 0

head
next
prev

(hfo,fst)
(pfo,ptr)

DLS 2+ level 0

(nfo,ptr)
(hfo,lst)

DLS 0+ level 1

(hfo2,all)
```

- Uses efficient graph-based algorithms to implement all needed operations: join, abstraction, entailment, ...
- Looks for memory safety errors: invalid dereferences, double frees, buffer overruns, memory leaks, ...
- Implemented as an open source GCC plug-in.
Predator: An Overview

- Focuses on **shape analysis** of low-level system code.
- Uses **symbolic memory graphs** (SMGs) to encode sets of heap configurations with various kinds of (nested) lists:

  \[
  \begin{align*}
  &\text{DLS 2+ level 0} \\
  &\text{head} \quad (h_{fo}, fst) \quad (h_{fo}, lst) \\
  &\text{next} \quad (h_{fo}, fst) \quad (n_{fo}, ptr) \\
  &\text{prev} \quad (p_{fo}, ptr) \\
  &DLS 0+ level 1 \\
  &\text{DLS 2+ level 0} \\
  &h_{fo} \quad n_{fo} \quad p_{fo} \quad (h_{fo}, lst) \\
  &h_{fo} \\
  &h_{fo}, all
  \end{align*}
  \]

- Uses efficient **graph-based algorithms** to implement all needed operations: join, abstraction, entailment, ...

looks for memory safety errors: invalid dereferences, double frees, buffer overruns, memory leaks, ...

implemented as an open source GCC plug-in.
Predator: An Overview

- Focuses on **shape analysis** of low-level system code.
- Uses **symbolic memory graphs** (SMGs) to encode sets of heap configurations with various kinds of (nested) lists:

  - DLS 2+ level 0
    - head
    - next
    - prev
    - (hfo, fst)
    - (pfo, ptr)
  - DLS 2+ level 0
    - (hfo2, all)
  - DLS 2+ level 0
    - (hfo, list)
  - DLS 2+ level 0
    - (nfo, ptr)
  - DLS 2+ level 0
    - (hfo, all)

- Uses efficient **graph-based algorithms** to implement all needed operations: join, abstraction, entailment, ...

- Looks for **memory safety errors**: invalid dereferences, double frees, buffer overruns, memory leaks, ...
Predator: An Overview

- Focuses on **shape analysis** of low-level system code.
- Uses **symbolic memory graphs (SMGs)** to encode sets of heap configurations with various kinds of **(nested) lists**:
  - \((h_{fo},fst)\)
  - \((h_{fo},lst)\)
  - \((n_{fo},ptr)\)
  - \((p_{fo},ptr)\)

- Uses efficient **graph-based algorithms** to implement all needed operations: **join**, **abstraction**, **entailment**, ...
- Looks for **memory safety errors**: invalid dereferences, double frees, buffer overruns, memory leaks, ...
- Implemented as an open source **GCC plug-in**.
Four instances of Predator running in parallel:

- One Predator Verifier: the original sound Predator, its verdict "correct" $\Rightarrow$ final verdict "correct" + witness.
- Two Predator Depth-First (DFS) Hunters: no heap abstraction, join up to isomorphism, sampled intervals bounded DFS: 200/900 GIMPLE instructions, one of them says "error" $\Rightarrow$ final verdict "error" + witness.
- One Predator Breadth-First (BFS) Hunter: no heap abstraction, join up to isomorphism, timeout-bounded BFS, its verdict within the time limit: "correct" $\Rightarrow$ final verdict "correct" + witness, "error" $\Rightarrow$ final verdict "error" + witness.

Otherwise the answer is "unknown".
Four instances of Predator running in parallel:

One **Predator Verifier**:
- the original sound Predator,
- its verdict "correct" ⇒ final verdict "correct" + witness.

Two **Predator Depth-First (DFS) Hunters**:
- no heap abstraction, join up to isomorphism,
- sampled intervals bounded DFS: 200/900 GIMPLE instructions,
- one of them says "error" ⇒ final verdict "error" + witness.

One **Predator Breadth-First (BFS) Hunter**:
- no heap abstraction, join up to isomorphism,
- timeout-bounded BFS,
- its verdict within the time limit: "correct" ⇒ final verdict "correct" + witness,
- "error" ⇒ final verdict "error" + witness.

Otherwise the answer is "unknown".
Four instances of Predator running in parallel:

- One **Predator Verifier**:
  - the original sound Predator,
  - its verdict "correct" ⇒ final verdict "correct" + witness.

- Two **Predator Depth-First (DFS) Hunters**:
  - no heap abstraction, join up to isomorphism, **sampled intervals**
  - **bounded DFS**: 200/900 GIMPLE instructions,
  - one of them says "error" ⇒ final verdict "error" + witness.

Otherwise the answer is "unknown."
Predator Hunting Party

Four instances of Predator running in parallel:

- One Predator Verifier:
  - the original sound Predator, 
  - its verdict “correct” ⇒ final verdict “correct” + witness.

- Two Predator Depth-First (DFS) Hunters:
  - no heap abstraction, join up to isomorphism, sampled intervals 
  - bounded DFS: 200/900 GIMPLE instructions, 
  - one of them says “error” ⇒ final verdict “error” + witness.

- One Predator Breadth-First (BFS) Hunter:
  - no heap abstraction, join up to isomorphism, timeout-bounded BFS,
  - its verdict within the time limit: “correct” ⇒ final verdict “correct” + witness, “error” ⇒ final verdict “error” + witness.
Predator Hunting Party

Four instances of Predator running in parallel:

- One **Predator Verifier**:
  - the original sound Predator,
  - its verdict “correct” ⇒ final verdict “correct” + witness.

- Two **Predator Depth-First (DFS) Hunters**:
  - no heap abstraction, join up to isomorphism, **sampled intervals**
  - **bounded DFS**: 200/900 GIMPLE instructions,
  - one of them says “error” ⇒ final verdict “error” + witness.

- One **Predator Breadth-First (BFS) Hunter**:
  - no heap abstraction, join up to isomorphism,
  - **timeout-bounded BFS**,
  - its verdict **within the time limit**:
    - “correct” ⇒ final verdict “correct” + witness,
    - “error” ⇒ final verdict “error” + witness.

- Otherwise the answer is “unknown”.
Competition Results

- **MemSafety**: silver.
  - best results among (in principle) sound tools especially for heap and linked lists.
Competition Results

- **MemSafety**: silver.
  - best results among (in principle) sound tools especially for heap and linked lists.

- **Errors reported by Predator Hunters only**:
  - many false alarms suppressed (all but 4 in ReachSafety),
  - some false alarms remain: abstraction of non-pointer data for unbounded data structures.

Soundness preserved:
- Only Predator Verifier can claim infinite-state programs correct.
- Some finite-state programs proved correct by the BFS Hunter.

What about letting Hunters verify?
- Correctly and quickly "verified" all cases with trees and skip-lists.
- Great to harvest points!

But: This is (in principle) unsound! Hence, not taken.
Competition Results

- **MemSafety**: silver.
  - best results among (in principle) sound tools especially for heap and linked lists.

- **Errors reported by Predator Hunters only**:
  - many false alarms suppressed (all but 4 in ReachSafety),
  - some false alarms remain: abstraction of non-pointer data for unbounded data structures.

- **Soundness preserved**:
  - Only Predator Verifier can claim infinite-state programs correct.
  - Some finite-state programs proved correct by the BFS Hunter.
Competition Results

- **MemSafety**: silver.
  - best results among (in principle) sound tools especially for heap and linked lists.

- **Errors reported by Predator Hunters only**:
  - many false alarms suppressed (all but 4 in ReachSafety),
  - some false alarms remain: abstraction of non-pointer data for unbounded data structures.

- **Soundness preserved**:
  - Only Predator Verifier can claim infinite-state programs correct.
  - Some finite-state programs proved correct by the BFS Hunter.

- **What about letting Hunters verify?**
  - Correctly and quickly “verified” all cases with trees and skip-lists.
  - Great to harvest points!
  - But: This is (in principle) unsound! Hence, not taken.