Predator and Forester: Two New Tools for Shape Analysis

Tomáš Vojnar
FIT, Brno University of Technology, Czech Republic

- Predator is a joint work with **K. Dudka** (FIT) and **P. Peringer** (FIT).
- Forester is a joint work with **P. Habermehl** (LIAFA), **L. Holík** (FIT/Uppsala), **A. Rogalewicz** (FIT) and **J. Šimáček** (FIT/VERIMAG).

Predator

Predator

❖ A new tool for shape analysis of C programs inspired by Space Invader [D. Distefano, C. Calcagno, H. Yang, and P. O'Hearn].

❖ Based on separation logic with higher-order inductive list predicates:

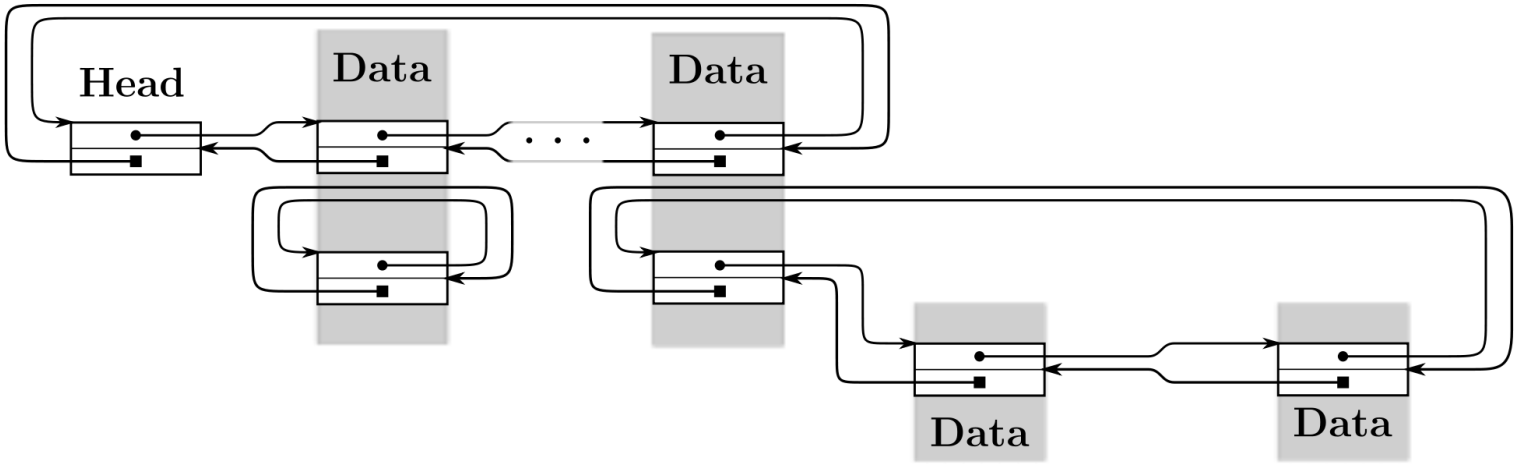
- list predicates parameterised by the shape of their elements,
- can handle nested singly- or doubly-linked lists (SLLs/DLLs) that can be cyclic and/or have various additional links (head/tail pointers, data pointers).

❖ Like Space Invader, Predator hunts for:

- basic memory safety errors: null dereferences, dangling pointer dereferences, memory leaks, double free operations,
- more complex checks can be implemented as testers written in C and attached to the analysed code.

Predator

- Compared to Space Invader:
  - DLLs supported equally well as SLLs.
  - A support for list segments of lengths 0+, 1+, 2+, and also 0 or 1.
  - A better support of pointer arithmetics, tailored for use with native Linux lists.



  - So far a very weak support of non-pointer data structures.

- Experiments with Space Invader and Predator:
  - In our test cases where both tools succeed, Invader is usually somewhat faster.
  - In multiple our test cases where Predator succeeds, Invader seems to loop, provides a false positive, and even a false negative.

Forester

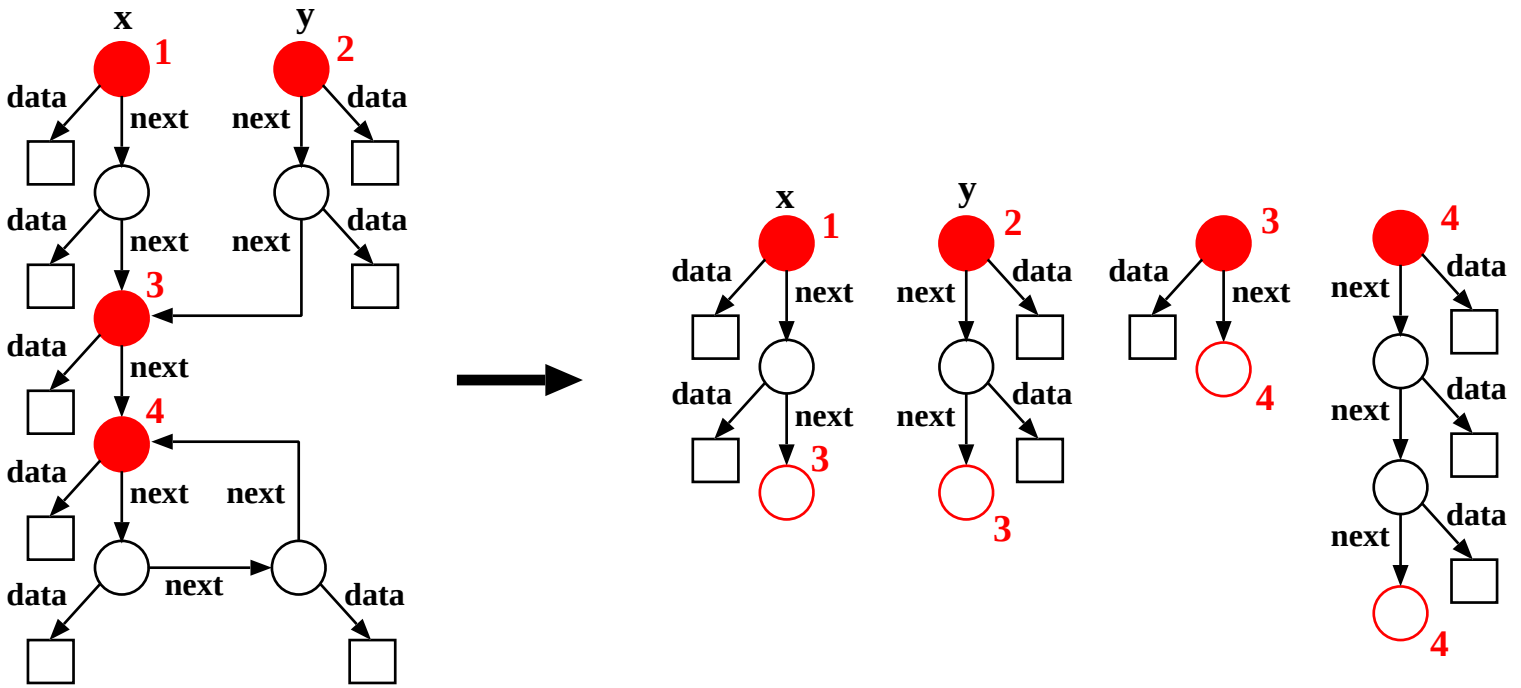
Forester

- A new tool for shape analysis of C programs based on tree automata.
- Looks for the same kind of errors as Predator.
- Unlike previous automata-based approaches,
  - does not use a single monolithic automaton to encode sets of memory configurations,
  - instead several automata representing parts of memory configurations separated like in separation logic are used.
- Uses abstract regular tree model checking on non-deterministic tree automata to over-approximate sets of reachable memory configurations.

Forest Automata

❖ Heaps are split at the so called cut-points into tree components whose leaves may refer back to the roots.

- Hence, heaps are represented by tuples of trees, i.e., forests.
- Tree automata used to represent sets of tree components.
- Sets of forests represented by tuples of tree automata, i.e., forest automata.

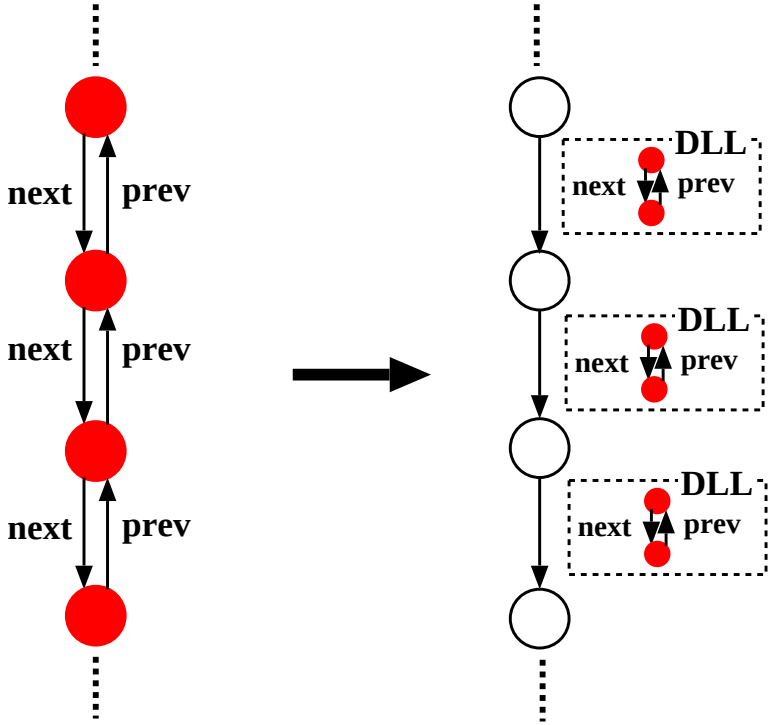


❖ Can be put into a canonical form: inclusion is decidable.

Hierarchical Forest Automata

❖ Sets of structures with unboundedly many cut-points (e.g., DLLs) are represented in a hierarchical way.

- The alphabet of forest automata can contain nested forest automata.
- If a nested automaton “hides” a bounded number of cut-points, its use on a loop in the higher-level automaton hides unboundedly many cut-points.



❖ Forester currently uses pre-defined nested automata for common data structures.

Evaluation

❖ Experiments with Forester, Invader, and ARTMC (our older tool based on abstract regular tree model checking with a monolithic heap encoding).

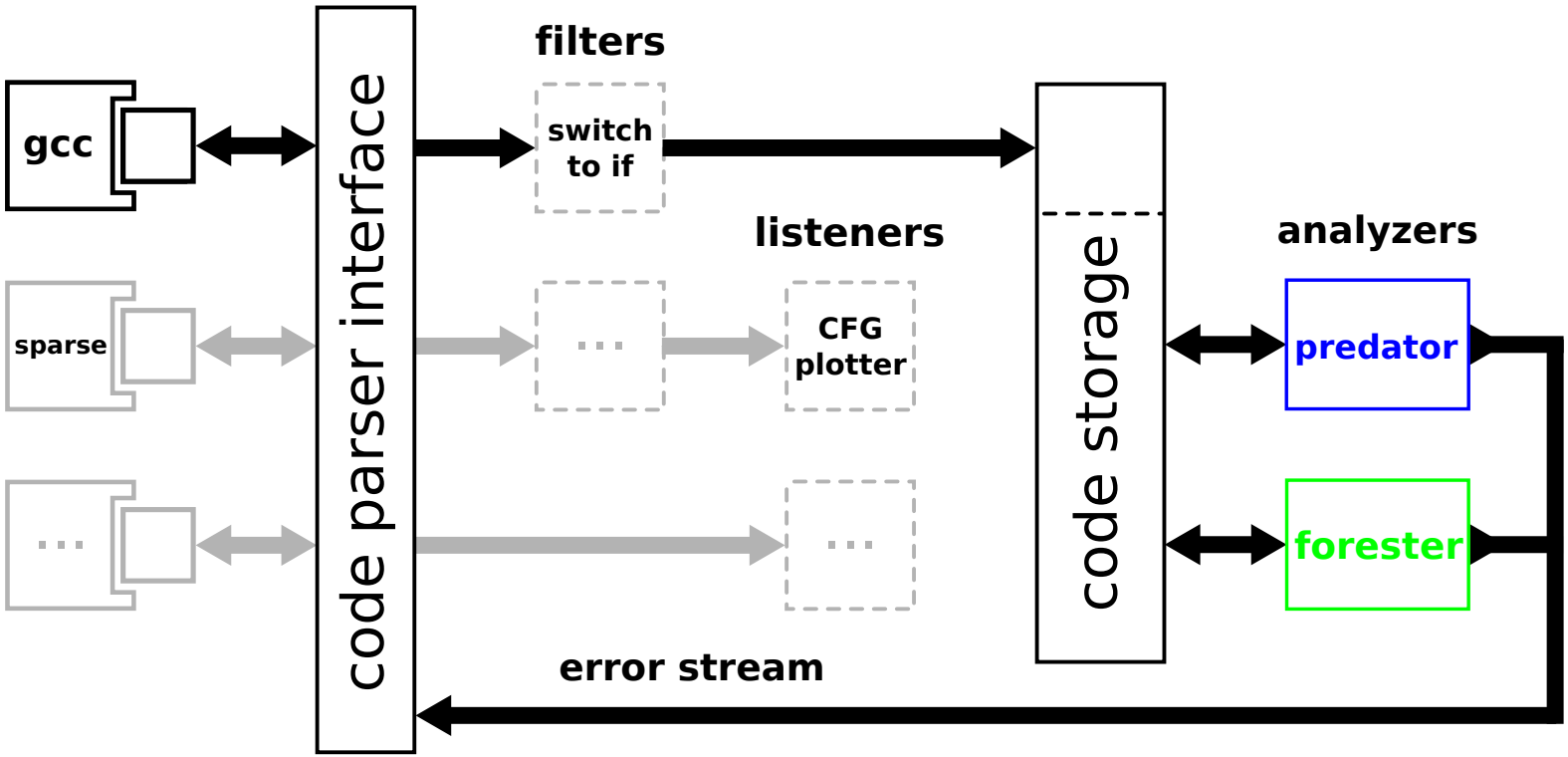
Example	Forester [sec]	Invader [sec]	ARTMC [sec]	Example	Forester [sec]	Invader [sec]	ARTMC [sec]
SLL (delete)	0.04	0.10	0.5	SLL (reverse)	0.04	0.03	
SLL (bubblesort)	0.12	error		SLL (insertsort)	0.09	0.10	
SLL (mergesort)	0.12	error		SLL of CSLLs	0.11	timeout	
SLL+head	0.04	0.06		SLL of 0/1 SLLs	0.13	timeout	
SLL_{Linux}	0.05	timeout		DLL (insert)	0.07	0.08	0.4
DLL (reverse)	0.05	0.09	1.4	DLL (insertsort1)	0.35	0.18	1.4
DLL (insertsort2)	0.16	error		CDLL	0.04	0.09	
DLL of CDLLs	0.32	timeout		SLL of $2CDLLs_{Linux}$	0.11	timeout	
tree	0.11		3.0	tree+stack	0.10		
tree+parents	0.18			tree (DSW)	0.41		o.o.m.

❖ In all cases, random creation and destruction of data structures, possibly with some further (indicated) data structure manipulation.

Code Listener Framework

- Both Predator and Forester are open source:
  - `http://www.fit.vutbr.cz/research/groups/verifit/`.

- Built as gcc plug-ins to be able to input any code that natively compiles with gcc.



Future Work

- Predator:
  - Optimisations of the internal structures based on the experience gained from the first prototype.
  - Support for more data structures (e.g., trees) in the form they appear in system code (Linux kernel).
  - Light-weight support for non-pointer data.

- Forester:
  - Predicate language abstraction.
  - Support for procedure summaries.
  - Automatic learning of nested automata.
  - Light-weight support for non-pointer data.