

Szoftver-modellellenőrző eszköz: CPAchecker

dr. Majzik István

BME Méréstechnika és Információs Rendszerek Tanszék

Szoftver-modellellenőrzés: CPAchecker

- The Configurable Software-Verification Platform
- Bemenet: **C program** + az ellenőrzés specifikációja
 - Ellenőrzések: Assertion, error címkéjű utasítás, deadlock, null dereference, ...
- Célkitűzések összehangolása
 - Program analízis: Nagy programokról, egyszerű tulajdonságok
 - Modellellenőrzés: Egyszerűbb programokról, teljes viselkedési analízissel felderíthető tulajdonságok
- Nagymértékben **konfigurálható**
 - Többféle absztrakció (nem csak predikátum absztrakció)
- Elérhető:
 - <http://cpachecker.sosy-lab.org/> (Java alapú, letölthető)
 - <http://cpachecker.appspot.com/> (online használatra)

Alkalmazott módszerek

- CEGAR: predikátum absztrakció, lusta absztrakciófinomítás, interpoláció
- Szimbolikus végrehajtás
- Korlátos modellellenőrzés

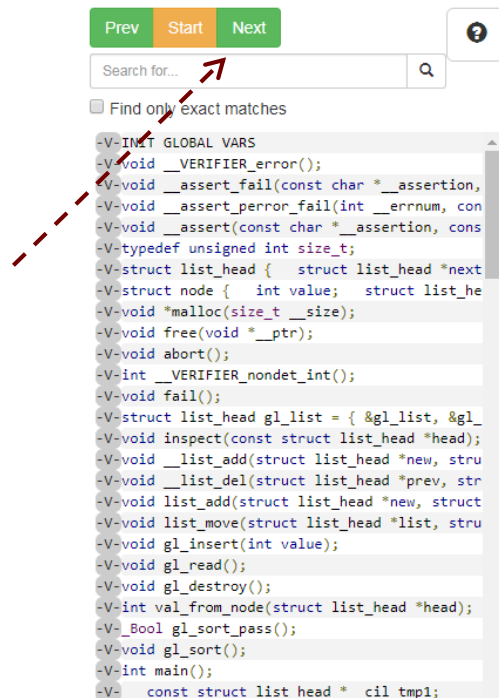
Verifier	CEGAR	Predicate Abstraction	Symbolic Execution	Bounded Model Checking	k-Induction	Property-Directed Reachability	Explicit-Value Analysis	Numerical Interval Analysis	Shape Analysis	Separation Logic	Bit-Precise Analysis	ARG-Based Analysis	Lazy Abstraction	Interpolation	Automata-Based Analysis	Concurrency Support	Ranking Functions
CPACHECKER	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓			

Bemenet

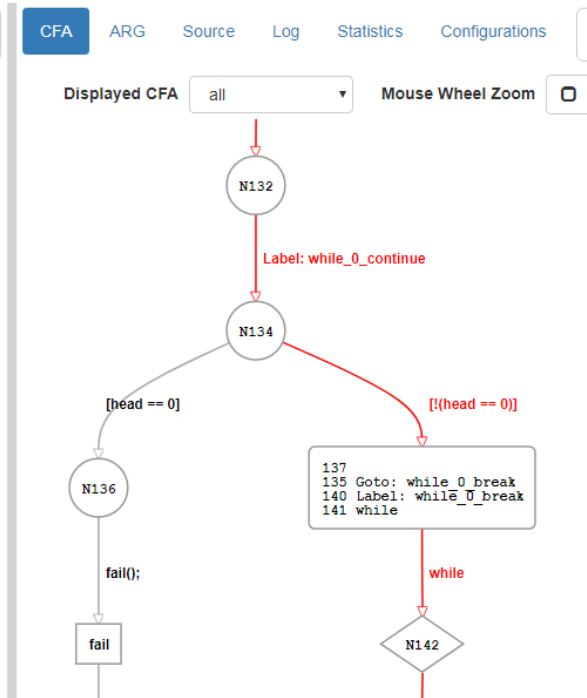
- C program előfeldolgozás után
 - Makrók, include, ... feloldása
 - `gcc -E program.c > program.i`
- Támogatott nyelvi konstrukciók:
Az alkalmazott SMT megoldótól is függ
(pl. Linux: MathSAT5, Windows: SMTInterpol)
 - Integer, float
 - Array, bitvector, list
 - Pointer, heap
 - Dinamikus memóriakezelés,
 - Rekurzió, többszálú konkurens működés, ...
- Több fájl együttes kezelése

Kimenet

- HTML jelentés (interaktív)
 - CFA: Control Flow Automaton
 - ARG: Abstract Reachability Graph
 - Forráskód (utasításokra ugrás a CFA éleiről)
 - Ellenpélda:
 - Programút bemutatása: forráskód lefutás, CFA-n illetve ARG-n
 - Lépésenként bejárható végrehajtás (CFA és forráskód követés)
 - Változóértékek követése (lásd -V-)
- Statisztikák



```
-V-INIT GLOBAL VARS
-V-void __VERIFIER_error();
-V-void __assert_fail(const char *__assertion,
-V-void __assert_perror_fail(int __errnum, con
-V-void __assert(const char *__assertion, cons
-V-typedef unsigned int size_t;
-V-struct list_head { struct list_head *next
-V-struct node { int value; struct list_he
-V-void *malloc(size_t __size);
-V-void free(void *__ptr);
-V-void abort();
-V-int __VERIFIER_nondet_int();
-V-void fail();
-V-struct list_head gl_list = { &gl_list, &gl
-V-void inspect(const struct list_head *head);
-V-void __list_add(struct list_head *new, stru
-V-void __list_del(struct list_head *prev, str
-V-void list_add(struct list_head *new, struct
-V-void list_move(struct list_head *list, stru
-V-void gl_insert(int value);
-V-void gl_read();
-V-void gl_destroy();
-V-int val_from_node(struct list_head *head);
-V-Bool gl_sort_pass();
-V-void gl_sort();
-V-int main();
-V-const struct list_head *__cil_tmp1;
```



Konfigurálhatóság

- Parancssori interfész
 - `cpa.bat -spec <spec> -config <config> <prog.c>`
- Specifikációk és konfigurációk (nincs "legjobb")

- **Specifikációk** (mit keresünk):
 - Assertion.spc: Assertion megsértése a kódban
 - ErrorLabel.spc: ERROR címke elérése a kódban
 - TerminatingFunctions.spc: Termináló függvény (pl. exit, abort)
 - Deadlock.spc: Holtpont konkurens programban
 - Null-deref.spc: Null-pointer dereferencia
 - UninitializedVariables.spc: Nem inicializált változó, return érték
 - **Memóriakezelés**: MemorySafety-deref (invalid derefencing of pointers), MemorySafety-free (invalid freeing of allocations), MemorySafety-memtrack (memory leaks)

Konfigurálhatóság

- Parancssori interfész
 - `cpa.bat -spec <spec> -config <config> <prog.c>`
- Specifikációk és konfigurációk (nincs "legjobb")

- **Konfigurációk** (hogyan keressük – több száz lehetőség):
 - Absztrakciók (predicate, value, ...), finomítások, ellenőrzések
 - Alapeset: `default.properties`, szekvenciális analízisek
 - Predikátum absztrakció túlcsordulás ellenőrzéshez
 - Lasso analízis terminálás ellenőrzéshez
 - BDD használata konkurens programokhoz
 - Predikátum absztrakció rekurzív programokhoz
 - k-indukció egyéb esetekre
 - Ajánlott konfiguráció:
 - `predicateAnalysis.properties` (CEGAR)
 - `predicateAnalysis-bitprecise.properties`

Megvalósítás elemei

- CPA: Configurable Program Analysis
(D , \rightarrow , merge, stop)
 - D absztrakt domén: Absztrakt állapotok, plusz konkrét állapotokra konkretizáló függvény
 - \rightarrow absztrakt állapotátmeneti reláció: Absztrakt állapotok között a CFA egy éle alapján
 - Merge operátor: Absztrakt állapotok összevonása
 - Stop operátor: Egy absztrakt állapotra megállapítja, hogy más állapotok lefedik-e (tartalmazza-e)

Demo

- Assertion ellenőrzése
 - Sikeres és hibát jelző ellenőrzés (kód módosítás után)
 - Ellenpélda bemutatása: Forráskód, változók, CFA, ARG
- ERROR címke ellenőrzése
 - Ellenpélda bemutatása: Forráskód, CFA, ARG
- Bubbleshort algoritmus verifikálása
 - Több CFA (függvényenként)
- Linux Driver Verification Project
 - Problems in Linux Kernel Found by CPAchecker
<http://linuxtesting.org/results/ldv-cpachecker>