DIVINE (via LLVM)

Charakteristika

- Cieľ: rýchlosť, spoľahlivosť, všeobecné použitie, easy-to-use
- Vhodný na verifikáciu rozsiahlych systémov (oproti sekvenčným model checkerom – problém so space explosion) – využíva efektívne space-reduction techniky (Partial Order Reduction, Path Compression)
- Keďže podporuje implementácie väčšiny POSIX thread APIs (pthread.h), umožňuje verifikáciu multivláknových programov
- Model checking LTL vlastností, vhodný aj na vývoj mod. checking algoritmov, alebo na experimentálne porovnávanie a vyhodnocovanie

Charakteristika II

- Verifikácia modelu v jazyku DVE
- Verifikácia cez LLVM
- UPAAL timed automata
  - LTL model checking
  - Deadlock detection
  - Implementuje Upaal Time Automata Parser Library, DBM library a interpreter pre timed automata
  - Akceptuje .xml
- MurPHI models
  - Implementovaný kompilátor, ktorý generuje natívny kód
  - Distribuovaná aj paralelná state-space analýza, detekcia deadlockov.

Charakteristika – kompresia stavového priestoru

- Bezstratová
- Založená na stromovej kompresii
  - Efektívne na veľkých modeloch
- Až 90% kompresia
- Zanedbateľný čas komprimovania
- Podporovaná všetkými algoritmami
- --compression alebo –compression=tree
- Podporované aj v paralelnej verifikácii, ale odporúča sa použitie zdieľanej pamäte.

Charakteristika – Windows verzia

- Od verzie 2.1 poskytuje grafické prostredie
- Windows verzia podporuje len paralelné systémy so zdieľanou pamäťou
- Zatiaľ neumožňuje verifikáciu C/C++ kódu cez LLVM

GUI

- Vo Windows verzii je vlastné
- Unix verzia – využíva Qt
- Verifikácia jedným klikom
- Grafický simulátor (napr. v prípade vygenerovania protipríkladu)
- Grafický debugger (kontrola premenných, krokovanie, náhodný beh)

Inštalácia I - požiadavky

- HW: 2GB miesta na disku a aspoň 4GB RAM
- Pre Windows: 32bit systém, MinGW kompilátor, CMake
- Pre Unix:
  - GNU C++ (4.7.3) alebo clang (3.2)
  - Cmake
  - Ďalšie doplnky:
    - LLVM (3.2)
    - Clang (3.2)
    - Qt (4.5) – GUI
    - Boost
    - libxml2
    - Pandoc (+ pdflatex/bibtex)
    - MPI (OpenMPI)
    - flex, byacc

Inštalácia II

- Rozbaliť tar do priečinka
- ./configure
- make
- make check
- make install
- Pozor na kompatibilitu verzíi!

Príkazy

- divine - -version
  - Zobrazí zoznam možnosti, dostupných v nainštalovanej verzii programu (závisí od toho, s akými prídavnými časťami bol skompilovaný)

- divine info <model>
  - Zobrazí informácie o konkrétnom modeli, resp. zoznam vlstností, ktoré sa dajú verifikovať (použije sa v ďalších príkazoch) – napríklad deadlock alebo assert, prípadne LTL

Príkazy II - combine

- divine combine [-f <formula file>] <model file>
- Niektoré jazyky majú vnútornú podporu pre LTL vlastnosti – vtedy su overované vlastnosti dostupné automaticky (verify - - property), pri overovaní cez DVE treba špecifikovať LTL vlastnosť v samostatnom súbore.
- divine combine [-f formula.ltl] [-p N] [-o] [-q] model.dve
- divine combine [-f ...] [...] model.mdve [P1=VAL] [P2=VAL] ..
- Príkaz combine preloží LTL na Büchi automat a ten sa zahrnie do DVE súboru. Pre každú LTL vlastnosť sa vytvorí samostatný .dve súbor

LTL - pripomenutie

- overovanie modelu pomocou formúl lineárnej temporálnej logiky
- vytvorí sa formálny model M systému (systém je množina nekonečných behov), ktorý chceme verifikovať a to, čo konkrétne chceme overiť vyjadríme ako formulu
- φ pomocou LTL a rozhodneme, či M |= F (tj či je M modelom φ).
- 2 rôzne behy sú ekvivalentné ak sa zhodujú v interpretácii at. premenných (buď ich spĺňajú alebo nie)
- φ sa vyhodnocuje nad jedným behom a vyjadruje sa o platnosti atomických premenných v stavoch behu.

LTL - pripomenutie

- Operátory:
- Fφ - (future) - niekde v behu platí φ
- Gφ - (globally) - v celom behu platí φ
- φUΨ - (until) - niekde v behu platí Ψ a do tej doby platí φ
- Xφ - (next) - v ďalšom stave platí φ
- φWΨ - (weak until) - tak ako until, ale Ψ nemusí nastať
- φRΨ - (release) - Ψ platí kým neplatí (Ψ AND φ), potom neplatí ani jedno (+tiež nemusí nastať to, že platí (Ψ AND φ))

LTL - syntax

```
#define at1 (Proces1.vStaveX)
#define at2 (Proces2.vStaveY)
#define at3 (premenna1 == 100)

#property F (at1 && at2)
#property G at1
#property !F at3
```

- #define zadáva symbolický názov pre atomickú propozíciu
- #property špecifikuje konkrétnu LTL formulu

Príkazy III - metrics

- divine metrics <prepínače> <model>
- [--reduce=R]
- [--no-reduce]
- [--fair]
- [--report[=<report format>] | -r]
- [--property=N]
- [engine options]

- Zisťuje dostupnosť (stavov) na celom state space daného systému
- Vypisuje štatistiky – počet states, transitions, accepting a deadlocks.

Príkazy IV - draw

- divine draw
- [--distance=N]
- [--trace=N,N,N...]
- [-l|--labels|--trace-labels]
- [--bfs-layout]
- [--reduce=R]
- [--no-reduce]
- [-f|--fair]
- [--render=<cmd>|-r <cmd>]
- [--compression]

Príkazy V - verify

- divine verify <prepíače> <model>
- [--reachability|--owcty|--map|--nested-dfs] – ak chceme použiť konkrétny algoritmus (inak automaticky vyberie vhodný algoritmus na overenie modelu, vzhľadom na typ vlastnosti)
- [--property=<name> | -p <name>] - akú vlastnosť chceme overiť, dá sa použiť divine info na zistenie zoznamu dostupných vlastností
- [--fair] – akceptuje len behy ktoré sú weakly fair. Zatiaľ dostupné len pre DVE modely, vhodné pri použití LTL.

Príkazy V – verify pokr.

- [--reduce=<reduction>] - vynucuje použitie heuristík, ktoré zmenšujú stavový priestor.
- [--report[=<report format>] | -r] – generuje report, formát napr. text, text:file, plain, atď.
- [--no-counterexample | -n] – zakazuje generovanie protipríkladov
- [--display-counterexample | -d] – prikazuje generovanie protipríkladov.
- [engine options] – *nezdokumentované!*

Príkazy VI - gen-explicit

- divine gen-explicit
- [--fair]
- [--reduce=<reduction>]
- [--report[=<report format>]]
- [engine options]
- [--no-save-states]
- [-o <file> | --output=<file>]

- Generuje state space modelu do súboru ktorý môže byť neskôr použitý DIVINE alebo iným nástrojom, schopným pracovať s DIVINE Explicite Space Format

LLVM

- Low Level Virtual Machine
- Infraštruktúra pre prekladač (knižnice a rozhrania)
- Napísané v C++
- Využíva ho Clang, ale aj veľa iných kompilátorov pre rôzne jazyky (napr. pre Python, Haskell). Clang (ale aj GCC s pluginmi) vie generovať optimalizovaný aj neoptimalizovaný bitcode.
- Podporuje life long compilation model, vrátane link-time, install-time, run-time

DIVINE a LLVM

- Dá sa použiť hocijaký kód v C/C++
-
- Skompiluje sa pomocou divine compile –llvm prog.c
-
- Takto sa vytvorí celé runtime prostredie – prog.bc
-
- Môžeme použiť divine info (a pozrieť si properties)
-
- Alebo divine metrics (počet states, transitions, accepting, deadlocks)
  - Pozor na deadlocks! Deadlock v DIVINE je iný ako deadlock v C. Deadlock v DIVINE je aj stav ktorý nemá následníka.
  - Vypnutie redukcií – navýši stavy a prechody (redukcie môžu byť náročné na výpočet)

DIVINE a LLVM

- Verifikácia – divine verify prog.bc -p <vlastnosť>, napríklad porušenie assertov atď
- Prípadne aj s -d
- Pri multivláknových programoch DIVINE prechádza všetky interakcie vlákien systematicky, na úrovni jednotlivých bitkódových inštrukcií. To mu umožňuje dokázať absenciu deadlockov/porušení assertov, čo je so štandardnými technikami nemožné.

DVE I

- Formalizmus pre modelovanie asynchrónnych systémov-protokolov.
- Čiastočne odvodený z modelovacieho jazyka pre Uppaal, ale sústreďuje sa viac na expresibilitu modelu ako na pohodlné modelovanie.
- Vytvára abstraktný model - automat resp. viacero rozšírených konečných automatov.
- Synchrónny/asychrónny mód (ale syn. nie je veľmi podporovaný)

DVE II - syntax

- Skladá sa z procesov, tie obsahujú premenné, prechody, stavy.
- Komunikácia cez kanály (buff/nbuff)
- Globálne/lokálne premenné
  - Zdieľanie premenných (sync)
- Prechody - guards+effects
- 2 dát typy: byte, int + jednorozmerné polia týchto typov
- Commited states

DVE II – syntax príklad

```
int glob_var = 0;

process P {
int loc_var = 0;
state s1, s2, s3;
init s1;
trans
s1 -> s1 {guard glob_var<3; effect loc_var = loc_var +1;},
s1 -> s2 {effect glob_var = glob_var +1;},
s2 -> s3 {};
}

system async;
```