UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Verificarea algoritmului DPLL in F*

propusă de

Alexandru Donica

Sesiunea: iunie/iulie, 2023

Coordonator științific

Conf. Dr. Ștefan Ciobâcă

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Verificarea algoritmului DPLL in F*

Alexandru Donica

Sesiunea: iunie/iulie, 2023

Coordonator științific

Conf. Dr. Ștefan Ciobâcă

Avizat,

Îndrumător lucrare de licență,

Conf. Dr. Ștefan Ciobâcă.

Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Donica Alexandru** domiciliat în **România, jud. Iași, mun. Iași, strada Costache Negri, nr. 35, bl. A1, ap. 42**, născut la data de **07 aprilie 2000**, identificat prin CNP **5000407226761**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2022, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Verificarea algoritmului DPLL in F*** elaborată sub îndrumarea domnului **Conf. Dr. Ștefan Ciobâcă**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: ..... Semnătura: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Verificarea algoritmului DPLL in F***, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea ”Alexandru-Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Alexandru Donica**

Data: ..... Semnătura: .....

Cuprins

Motivație

Rolul unui 'SAT solver' este de a rezolva problema satisfiabilității booleene, făcându-se si in ziua de azi cercetări care au scopul de a îmbunătăți algoritmi existenți. Acest 'SAT solver' găsește o soluție pentru o formula data in cazul in care formula este satisfiabilă, in caz contrar formula este nesatisfiabilă.

Corectitudinea oricărui rezultat al unei formula satisfiabile poate fi verificat folosind algoritmi simpli. Însă un 'SAT solver' complex creat pentru procesa formule de dimensiuni din ce in ce mai mari sau pentru a avea o viteza de rezolvare a problemei mai rapida decât alți 'SAT solveri' folosiți pot conține erori de programare ce ar produce rezultate false, cum ar fi, in urma procesării unei formula nesatisfiabile acesta sa enunțe ca este satisfiabilă, sau invers.

De aceea este importanta crearea unor programe care verifica corectitudinea acestor 'SAT solveri' si am creat un 'SAT solver' verificat formal folosind limbajul de programare F* (FStar).

Introducere

detalii despre problema sat
conexiuni cu probleme reale
solvere create pana acum
probleme care apar la aceste solvere
importanta verificarii solverelor
acest solver implementeaza alg DPLL
descriere la alg DPLL
verificare formala
fstar - limbaj de programare si verificare foloseste z3 smt solver
ce presupune un solver verificat
de ce acest solver e verificat si in ce mod de catre fstar

Capitolul 1

Algoritmul DPLL

(Davis-Putnam-Logemann-Loveland)

Capitolul 2

Detalii de implementare

2.1 Dividerea pe module?

rolul fiecarui fisier + cea mai importanta functie de acolo?
link catre fisiere de input?

2.2 despre tipurile de date

2.3 cum functioneaza?

despre asserturi, putin cod efectiv, multe lemme/asserturi
compara cate linii is in .ml fata de .fst

2.4 despre tipurile de date simple

2.5 proofness / soundness / completeness / etc..

2.6 Metrici orientative? / poate in concluzie?

timp sa returneze sat
timp sa dea unsat cam mult
500 secunde sa compileze, verifice si extraga cod pt toate fisierele
datatypes - 8 secunde
datatypeUtils - 33 sec

dpllpropagation - 285 sec

occmatrix - 91 sec

fileparser - 4 sec

dpll - 71 sec

convertorToString - 3 sec

main - 3 sec

Capitolul 3

Pasi necesari pentru a reproduce

Informatii despre replicarea conditiilor necesare executiei programelor scrise folosind F* se pot gasi pe pagina de github a limbajului F*. [1]

In sectiunea urmatoare se prezinta pasii care au fost luati pentru crearea mediului in care s-a dezvoltat 'SAT solver-ul'. Instructiunile urmatoare sunt compatibile cu sistemul de operare Windows, verificat cu versiunile 10/11.

3.1 Programe si resurse necesare

Urmatoarele aplicatii/programe/resurse trebuie descarcate de la locatia specificata fiecareia in fisierul 'INSTALL.md' gasit pe pagina de github a limbajului FStar.

- OCaml - necesar compilarii si executarii fisierelor OCaml (.ml), care rezulta in urma compilarii fisierelor FStar (.fst)
- opam - necesar pentru a instala pachetele necesare la compilarea fisierelor specifice limbajului de programare OCaml (versiune folosita - 4.14.0)

  (versiunea folosita pentru lucrare - 2.0.10)
- cygwin - ofera posibilitatea compilarii si executarii a programelor tipice sistemelor de operare Unix si Linux, ceea ce include suport pentru fisiere 'Makefile'

  (versiunea folosita pentru lucrare - 3.4.6)
- Z3 - folosit pentru a valida fisierele ce contin programe scrise folosind F*

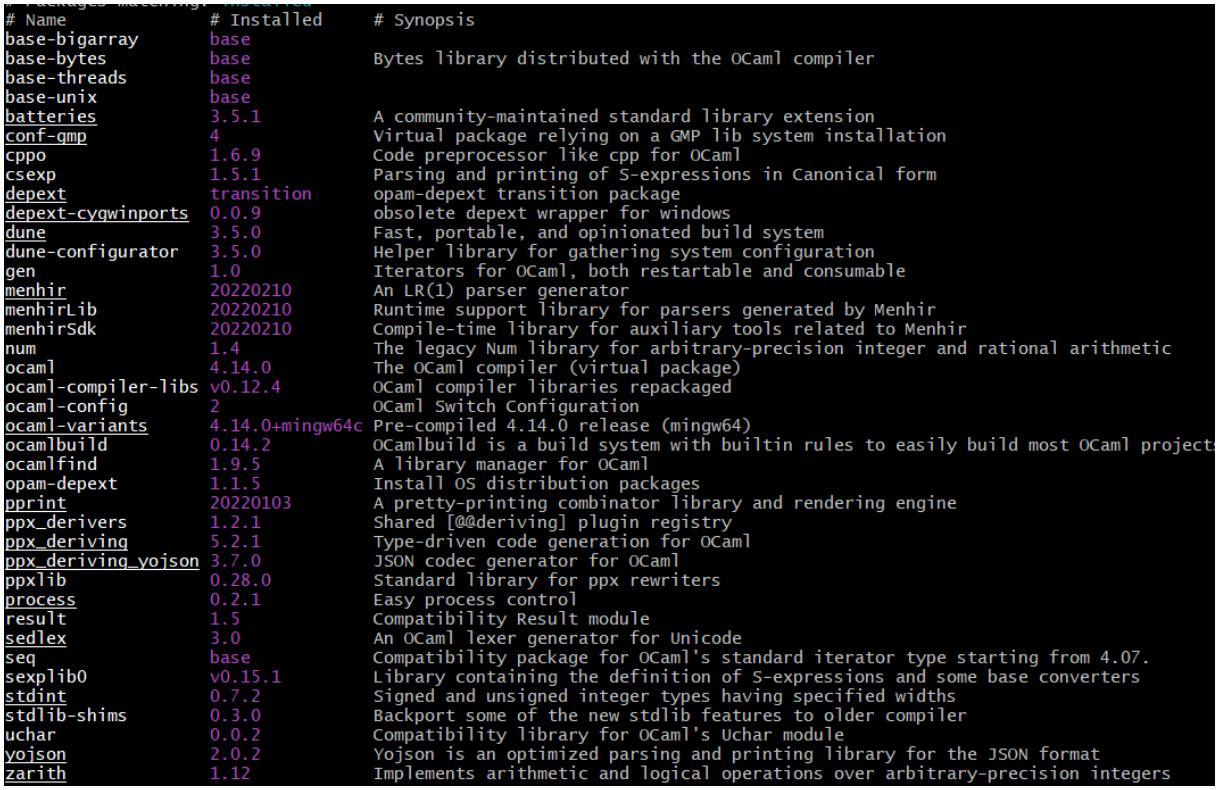
  (arhiva folosita pentru Windows - z3-4.8.5-x64-win.zip)

[1] https://github.com/FStarLang/FStar/blob/master/INSTALL.md

Dupa descarcarea/instalarea acestor resurse, trebuie clonata ramura 'master' a proiectului FStar pe dispozitivul local, denumind folder-ul "fstar". (locatia clonei pentru acest proiect: "D:/fstar", versiunea - F* 2023.04.26 dev)

Trebuie adaugate path-urile absolute catre ".../fstar/bin" si ".../z3-win/bin" in variabila 'Path' a sistemului.

Dupa instalarea programului 'opam', trebuie instalate mai anumite pachete de date. Minimul necesar de pachete se poate gasi si pe instructiunile de instalare gasite la link-ul de mai sus, insa pentru a avea la dispozitie toate resursele din proiectul FStar descarcat fara erori, sunt necesare urmatoarele pachete:



# Name	# Installed	# Synopsis
base-bigarray	base	
base-bytes	base	Bytes library distributed with the OCaml compiler
base-threads	base	
base-unix	base	
batteries	3.5.1	A community-maintained standard library extension
conf-gmp	4	Virtual package relying on a GMP lib system installation
cppo	1.6.9	Code preprocessor like cpp for OCaml
csexp	1.5.1	Parsing and printing of S-expressions in Canonical form
depext	transition	opam-depext transition package
depext-cygwinports	0.0.9	obsolete depext wrapper for windows
dune	3.5.0	Fast, portable, and opinionated build system
dune-configurator	3.5.0	Helper library for gathering system configuration
gen	1.0	Iterators for OCaml, both restartable and consumable
menhir	20220210	An LR(1) parser generator
menhirLib	20220210	Runtime support library for parsers generated by Menhir
menhirSdk	20220210	Compile-time library for auxiliary tools related to Menhir
num	1.4	The legacy Num library for arbitrary-precision integer and rational arithmetic
ocaml	4.14.0	The OCaml compiler (virtual package)
ocaml-compiler-libs	v0.12.4	OCaml compiler libraries repackaged
ocaml-config	2	OCaml Switch Configuration
ocaml-variants	4.14.0+mingw64c	Pre-compiled 4.14.0 release (mingw64)
ocamlbuild	0.14.2	OCamlbuild is a build system with builtin rules to easily build most OCaml project
ocamlfind	1.9.5	A library manager for OCaml
opam-depext	1.1.5	Install OS distribution packages
pprint	20220103	A pretty-printing combinator library and rendering engine
ppx_derivers	1.2.1	Shared [@@deriving] plugin registry
ppx_deriving	5.2.1	Type-driven code generation for OCaml
ppx_deriving_yojson	3.7.0	JSON codec generator for OCaml
ppxlib	0.28.0	Standard library for ppx rewriters
process	0.2.1	Easy process control
result	1.5	Compatibility Result module
sedlex	3.0	An OCaml lexer generator for Unicode
seq	base	Compatibility package for OCaml's standard iterator type starting from 4.07.
sexplib0	v0.15.1	Library containing the definition of S-expressions and some base converters
stdint	0.7.2	Signed and unsigned integer types having specified widths
stdlib-shims	0.3.0	Backport some of the new stdlib features to older compiler
uchar	0.0.2	Compatibility library for OCaml's Uchar module
yojson	2.0.2	Yojson is an optimized parsing and printing library for the JSON format
zarith	1.12	Implements arithmetic and logical operations over arbitrary-precision integers

La finalul acestor pasi, folosind terminalul Cygwin si instructiunile de tipul 'make' in folder-ul 'fstar', ar trebui sa functioneze verificarea si executarea oricaror fisiere surse scrise in F*, fisiere proprii sau exemple ce faceau deja parte din proiect.

3.2 Executarea solver-ului SAT

Sursele corespunzatoare proiectului prezentat se gasesc la: FStar-DPLL github.

Aceste surse trebuie descarcate, salvate intr-un folder in proiectul 'fstar'. Fisierul 'Makefile' trebuie modificat, astfel incat variabila 'FSTAR-HOME' trebuie sa faca referire folder-ul 'fstar'. Aceiasi pasi trebuie facuti pentru fisieru 'Makefile' din folder-ul 'output'.

Apoi, in terminalul cygwin deschis in folder-ul proiectului DPLL-FStar, trebuie executata comanda 'make', la finalul careia in folder-ul 'output' vor aparea pentru fiecare fisier sursa '.fst' cate un fisier '.ml' care contin codul Ocaml extras din sursele FStar. De asemenea in folder-ul 'output' se va afla executabilul "Main.exe".

Imediat dupa pornirea programului "Main.exe", trebuie introdus de la tastatura calea relativa catre un fisier de input. Cateva fisiere de input exista in folder-ul "input-files" si orice alt fisier de intrare trebuie sa respecte acel pentru ca parsarea implementata a datelor sa functioneze.

La finalul unei astfel de executii, va aparea mesaj la consola cygwin cu rezultatul obtinut, fie ca formula data este nesatisfiabila, fie ca e satisfiabila si alaturi o varianta de raspuns ce contine variabilele formulei si valorile lor astfel incat fiecare clauza a formulei sa aibe valoarea de adevar true.

—INTRODU EXEMPLU POZA INPUT / OUTPUT DUPA EXECUTIE

Concluzii

acest solver nu prezinte cele mai eficiente structuri de date si euristici

insa prezinta cum arata specificatii functionale pt algoritmului dpll si implicit reprezinta o baza pt specificatiile oricarei extensii ale sale

solverul este sound, complet, garantat ca se termina? ,verificat formal

schimbarea structurilor de date spre o forma mai eficienta nu ar fi una dificila

Bibliografie

- fstar tutorial,
- fstar github,
- toate linkurile referentiate mai sus?