Interogare LLM pentru instrumente de verificare formală

Mateian Tudor, Miron Sebastian-Ioan, Rancov Larisa, Rațț Ioan-Paul

Universitatea de Vest din Timișoara, Facultatea de Informatică
Master: Inginerie Software
Timișoara, România
larisa.rancov03@e-uvt.ro

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: First keyword · Second keyword · Another keyword.

1 Introducere

Verificarea formală a programelor a devenit tot mai prezentă în proiectarea software, având cerințe mari de fiabilitate, de la sisteme critice și infrastructură până la aplicații utilizate în mai multe domenii. În paralel, ecosistemul s-a diversificat rapid fiindcă au apărut instrumente specifice pentru diferite sarcini, precum dovedirea corectitudinii funcționale, demonstrarea terminării, estimarea limitelor de complexitate, verificarea proprietăților pentru rețele neuronale și evaluarea formulărilor booleene cuantificate. Ritmul aparițiilor, repoziționărilor și abandonărilor de proiecte face dificilă menținerea unei imagini actualizate a peisajului, informațiile fiind dispersate în repository-uri, și rezultate ale competițiilor, iar un catalog întreținut manual își pierde actualitatea rapid.

Dificultatea reală nu provine doar din volum, dar și din calitatea neuniformă a metadatelor publice. Există descrieri sumare sau ambigue, licențe neprecizate ori schimbate între versiuni, categorii amestecate și semnale slabe privind activitatea curenta a proiectului îngreunează selecția. Fenomene precum link-rot sau reorganizarea repository-urilor duc la linkuri invalide, iar deduplicarea devine dificilă cand același instrument este reambalat, redenumit sau multiplicat prin fork-uri. În absența unui proces sistematic, listele publice devin incomplete sau inconsecvente, ceea ce descurajează folosirea si reduce utilitatea de cercetare.

LLM-urile mari pot accelera descoperirea inițială, deoarece sunt utile pentru a formula și rafina interogări, a sumariza pagini, a extrage nume, descrieri, linkuri, indicii de licență și o încadrare preliminară pe categorie. Aceste avantaje vin însă cu riscuri precum halucinații, clasificari inexacte, confuzii între „bibliotecă" , „framework" , și „instrument utilizabil direct" , dar și cu tendința de a acorda aceeași greutate surselor cu credibilitate diferită. Din acest motiv, un flux complet automat nu ar fi potrivit pentru un catalog public care își propune rigoare și trasabilitate. Este necesară o etapă de control uman înainte de publicare, care să confirme existența proiectului, corectitudinea încadrării și validitatea informațiilor.

Proiectul de față propune o abordare echilibrată, ce include un flux de descoperire asistat de LLM, rulat periodic, completat cu validare umană. Sistemul interoghează LLM-ul pe direcții corespunzătoare categoriilor vizate (corectitudine funcțională, terminare, limite de complexitate), extrage metadate minimale, le normalizează (nume, descriere scurtă, adresă de repo sau pagină oficială, limbaj, categorie), verifică automat existența linkurilor. Propunerile trec apoi printr-o interfață de revizuire, în care editorul uman poate aproba, respinge sau corecta metadatele, fiindca doar intrările aprobate se propagă către listă.

Scopul proiectului este să identifice în mod recurent instrumente noi de verificare formală care merită atenție, punând accent pe noutate, relevanță și pe calitatea informației care le însoțește. Demersul este important pentru că reduce costul de căutare pentru cercetători și practicieni, scurtează timpul până la evaluări comparative și experimente, scoate la lumină proiecte recente sau nișate care altfel rămân greu de descoperit și oferă un punct de plecare suficient de clar pentru a decide dacă un instrument merită explorat mai departe. În acest sens, LLM-ul are rolul de a propune piste plauzibile, iar verificarea umană

funcționează ca filtru de încredere, astfel încât recomandările finale să fie utile și responsabile. Obiectivul este deci furnizarea constantă de sugestii proaspete si bine motivate, care să susțină activitatea de cercetare și integrare practică.

2 Descrierea problemei

Problema abordată este identificarea, la intervale regulate, a unor instrumente pentru verificarea formală a programelor și prezentarea lor, într-o formă coerentă editorilor care decid dacă merită recomandate mai departe. Prin descoperire periodică se înțelege reluarea aceleiași proceduri de căutare și filtrare la intervale fixe, astfel încât rezultatele să nu depindă de un efort punctual, ci să producă în timp o serie constantă de propuneri. Human-in-the-loop (HIL) face referire la faptul că decizia de includere nu este automată, ci un editor verifică existența proiectului, sensul încadrării pe categorie și caracterul plauzibil al informațiilor asociate, iar recomandarea finală se realizează doar după această confirmare.

Domeniul adoptat este cel al verificării formale, cu accent pe categoriile menționate mai sus. Interesul cade pe instrumente noi sau mai puţin vizibile, utile pentru a inspira experimente, comparaţii sau adoptări rapide în proiecte.

Pentru ca o propunere să fie evaluată eficient, fiecare instrument identificat trebuie să fie însoțit de un set minimal de informații funcționale, suficient pentru o înțelegere de primă lectură. Sunt necesare deci un nume clar și o scurtă descriere în câteva fraze, un link către repository sau către pagina oficială, o indicare a limbajului ori a tehnologiilor predominante, menționarea licenţei atunci când poate fi dedusă din sursa oficială, o etichetă de categorie dintre cele enumerate mai sus şi un indiciu al activităţii proiectului, de tipul existenţei unor commit-uri sau versiuni recente. Acest nucleu informativ are rolul de a scădea costul de verificare pentru editor şi de a permite o decizie rapidă, fără a presupune parcurgerea întregii documentaţii a proiectului.

Calitatea procesului trebuie să fie evaluabilă prin câteva obiective măsurabile care să nu depindă de detalii de implementare. În mod firesc, se dorește o rată de aprobare după verificarea umană care să fie semnificativă, dar nu maximală, semn că filtrarea automată aduce candidaţi promiţători fără a forţa includerea lor; o ţintă rezonabilă poate fi ca cel puţin două treimi dintre propuneri să fie confirmate într-o perioadă de referinţă. De asemenea, procesul ar trebui să producă un volum stabil de descoperiri, de pildă câteva instrumente noi pe lună, în funcţie de ritmul real al ecosistemului. Completitudinea informaţiilor oferite pentru fiecare intrare poate fi urmărită ca proporţie a câmpurilor obligatorii populate corect, cu aşteptarea de a atinge un prag ridicat şi consistent în timp. În fine, timpul dintre identificare şi verdictul editorului trebuie menţinut scurt, astfel încât recomandările să rămână proaspete şi utile.

În ansamblu, nu se urmărește stabilirea valorii tehnice definitive a fiecărui proiect, ci furnizarea unui mecanism repetabil de descoperire și triere inițială, care aduce în atenție candidați relevanți, atașează informația minimă necesară unei decizii informate și menține controlul la nivel uman acolo unde este necesar.

3 State-of-the-art

State-of-the-art în descoperirea instrumentelor de verificare formală evidențiază un peisaj fragmentat, dificil de cartografiat și lipsit de un director centralizat, în ciuda numărului mare de instrumente existente. Sursele actuale care cataloghează astfel de unelte sunt variate, dar fiecare prezintă limitări importante. Paginile generale, precum categoria „Formal methods tools" de pe Wikipedia sau diversele wiki-uri comunitare, oferă o acoperire redusă și adesea învechită, reflectând rareori aparițiile recente din domeniu. În special wiki-urile dedicate metodelor formale depind aproape integral de contribuții voluntare, iar multe secțiuni păstrează informații din anii 2000, ceea ce le transformă mai degrabă în resurse istorice decât în instrumente actuale de descoperire.

O altă sursă notabilă o reprezintă site-urile organizațiilor profesionale, precum Formal Methods Europe, care întrețin liste structurate de instrumente împărțite pe categorii precum model checking, teoreme-proving sau solvere SMT. Deși aceste liste sunt considerate printre cele mai credibile și bine organizate, ele sunt totuși dependente de contribuțiile dezvoltatorilor, iar instrumentele neanunțate de autori rămân în afara catalogării. Chiar FME recunoaște limitele clasificării pe categorii fixe și sugerează trecerea la un sistem flexibil de etichete, care însă nu este încă implementat. În mod similar, listele comunitare de pe GitHub, de tip „awesome lists", pot fi utile pentru orientare rapidă, dar sunt neoficiale, incomplete și lipsite de rigoare în selecție și actualizare, motiv pentru care nu pot fi considerate surse standardizate.

Pentru anumite subdomenii au apărut portaluri specializate precum Termination-Portal, care oferă liste de instrumente pentru terminarea programelor, incluzând unelte precum AProVE sau TTT2. Cu toate acestea, aceste portaluri suferă de probleme severe de mentenanță; multe secțiuni nu au mai fost actualizate din 2008, ceea ce compromite utilitatea lor în raport cu evoluțiile recente. În analiza complexității programelor situația este și mai fragmentată, multe instrumente fiind cunoscute doar prin participările la competiții sau prin literatura științifică, fără a exista un director care să le centralizeze în mod coerent. În schimb, competițiile internaționale precum SV-COMP oferă surse actualizate și standardizate, întrucât obligă participanții să își documenteze uneltele. Portalul Tools for Formal Methods întreținut de SOSY-Lab conține astfel de informații, incluzând instrumente precum CPAchecker, Ultimate Automizer sau ESBMC, împreună cu detalii despre tehnicile folosite și rezultatele obținute. Totuși, acoperirea acestor resurse se concentrează în principal pe verificarea programelor C și Java, lăsând mai puțin vizibile instrumentele din alte ecosisteme.

În domenii emergente, cum este verificarea rețelelor neuronale, lipsa unor liste centralizate a reprezentat mult timp o dificultate. Apariția VNN-COMP în 2020 a început să suplinească această absență, oferind anual un inventar al instrumentelor precum Marabou, Reluplex sau VeriNet și raportând performanțele acestora. Totuși, nu există încă un portal stabil echivalent QBFLIB, iar evoluția rapidă a domeniului face ca orice listă să devină învechită într-un timp scurt. Spre deosebire de aceste inițiative noi, zona QBF beneficiază de QBFLIB, o bibliotecă matură și bine întreținută de solvere precum DepQBF, CAQE sau

GhostQ. Cu toate acestea, și QBFLIB este dependentă de contribuții voluntare, ceea ce înseamnă că instrumentele necunoscute pot rămâne neincluse.

Descoperirea efectivă a acestor instrumente poate fi realizată prin căutare clasică sau prin interogare asistată de modele de limbaj de tip LLM. Căutarea manuală presupune interogări pe motoare de căutare, consultarea literaturii de specialitate și analiza documentației oficiale a proiectelor. Avantajul acestei metode constă în controlul și acuratețea ridicată a informațiilor, dar procesul este consumator de timp, necesită expertiză și poate omite unelte din afara zonei de vizibilitate a cercetătorului. În schimb, modelele LLM pot oferi rapid liste inițiale și pot agrega informații din surse dispersate, menționând uneori instrumente obscure sau dificil de găsit. Totuși, răspunsurile generate pot conține inexactități, confuzii sau informații învechite, întrucât modelele nu au un mecanism intern de verificare factuală și nici nu garantează exhaustivitatea rezultatelor. În practică, o strategie hibridă, adică generarea unei liste brute cu ajutorul unui LLM, urmată de verificare manuală, este cea mai eficientă.

4 Instalare și configurare

5 Statusul implementării

Stadiul actual al implementării arată că toate componentele principale ale sistemului au fost demarate, însă se află încă în proces de dezvoltare. A fost inițiat un endpoint pentru interogarea unui model LLM extern, precum ChatGPT, folosit pentru extragerea automată de instrumente relevante. De asemenea, a fost începută integrarea frontendului cu backendul printr-un endpoint de tip chat, care facilitează comunicarea cu modelul. Pentru testarea locală s-a configurat modelul Ollama, oferind flexibilitate și control în faza curentă de lucru. În paralel, a început și structurarea documentației în LaTeX, iar prompturile sunt în continuare rafinate pentru a asigura claritate, consistență și aliniere la obiectivele proiectului.

6 Arhitectura sistemului

7 Diagrame UML

Pentru a ilustra arhitectura de ansamblu a aplicației dezvoltate, a fost realizată o diagramă UML[fig. 1] care evidențiază principalele componente ale sistemului și interacțiunile dintre acestea. Diagrama prezintă modul în care frontendul comunică cu backendul prin intermediul unui endpoint de tip chat, precum și modul în care backendul interacționează atât cu un model LLM extern (precum ChatGPT), cât și cu modelul local Ollama, utilizat în procesul de testare.

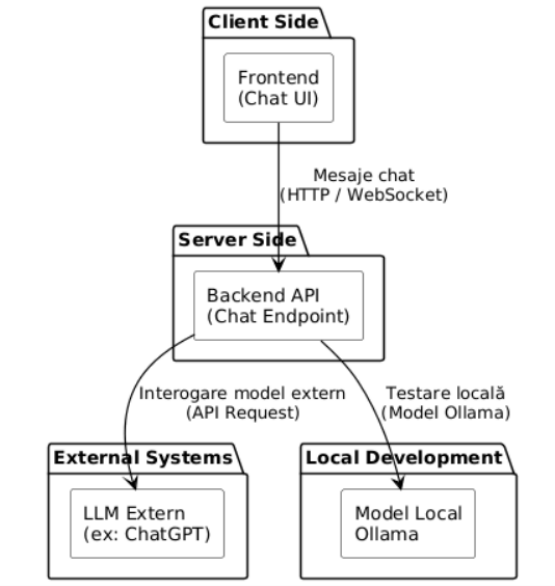

Fig. 1. Structura generală a sistemului

8 Link Github

Toate fișierele asociate proiectului, inclusiv codul sursă, componentele frontend și backend și exemplele de interogări pot fi accesate public la adresa:
https://github.com/tsonicm/ProiectVF

9 Contribuțiile membrilor echipei

Rancov Larisa s-a ocupat de redactarea raportului în Latex, structurând conținutul și organizând componentele academice ale proiectului. Rațț Ioan-Paul a implementat interfața grafică a aplicației folosind frameworkul React, integrând componente UI moderne prin biblioteca ShadCN. Miron Sebastian-Ioan a fost responsabil de partea de prompt engineering, testând și optimizând interacțiunile cu modelele LLM, utilizând Ollama pentru rularea locală și gestionarea fișierelor text. Mateian Tudor a dezvoltat componenta backend a aplicației, creând un conector între interfață și modelul AI cu ajutorul FastAPI, tot prin integrarea cu Ollama.