Podobnosti v zdrojovom kóde

Marek Roštár

Vedúci: Ing. Michal Kompan, PhD.

Úvod

- Cieľom práce bolo analyzovať, navrhnúť a implementovať metódu zisťovania podobnosti v zdrojových kódoch
- Pomocou zisťovania podobnosti zdrojových kódov vieme zlepšiť kvalitu zdrojových kódov a odhaliť plagiátorstvo

Plagiátorstvo

- Akt vydávania cudzej práce alebo nápadu za vlastný
- Dá sa rozdeliť na doslovné a inteligentné
- Plagiátorstvo v zdrojových kódoch sa líši od plagiátorstva v textových dokumentoch
- V práci sme identifikovali plagiátorské útoky (zmena bielych znakov, zmena komentárov, zmena poradia častí zdrojového kódu, premenovanie premenných, zmena algebraických výrazov), ktorých kombinácia tvorí časť reálnych útokov
- Ako podobnosť medzi zdrojovými kódmi rôznych jazykov sme uvažovali prípad, keď prekladateľ nepridal žiadnu dodatočnú hodnotu okrem prekladu

Metódy zisťovania podobnosti

- Textové [2] – porovnávanie reťazcov textu
  - + veľmi rýchle, jednoduché na implementáciu
  - - náchylné na útoky
- Stromové [2] – porovnávanie pomocou stromov
  - + schopné odhaliť väčšinu útokov
  - - pomalšie
- Tokenové [1] – porovnávanie abstrahovaných reťazcov
  - + rýchle, schopné odhaliť veľa útokov
  - - vyžaduje predspracovanie
- Semantické [1] – vyhodnocujú správanie zdrojového kódu
  - + vedia odhaliť útoky ktoré iné metódy nedokážu
  - - pomalé

[1] S. Park, S. Ko, J. J. Choi, H. Han, and S.-J. Cho, “Detecting Source Code Similarity Using Code Abstraction Categories and Subject Descriptors,” Proc. 7th Int. Conf. Ubiquitous Inf. Manag. Commun. - ICUIMC ’13, pp. 1–9, 2013.

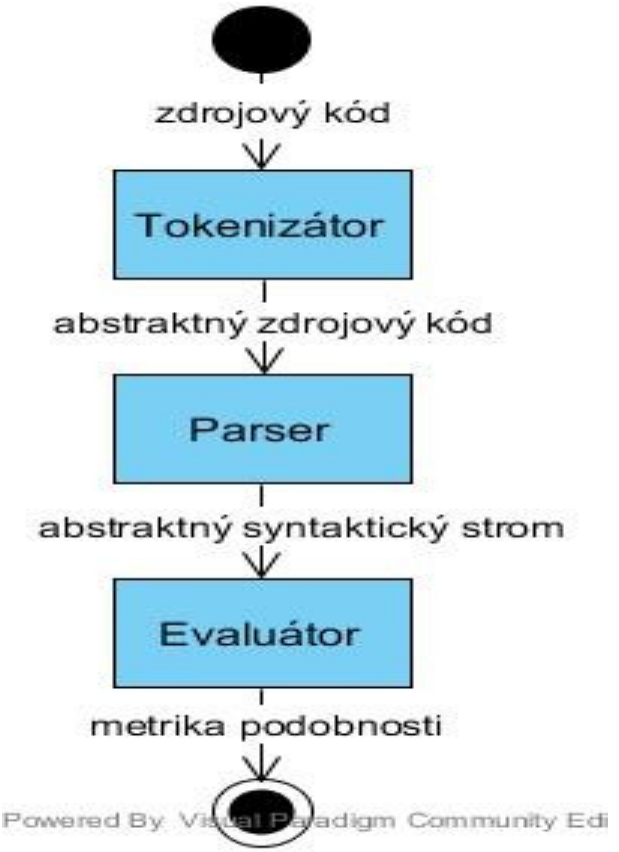
[2] Y. Zhang, X. Gao, C. Bian, D. Ma, and B. Cui, “Homologous detection based on text, Token and abstract syntax tree comparison,” Proc. 2010 IEEE Int. Conf. Inf. Theory Inf. Secur. ICITIS 2010, pp. 70–75, 2010.

Dôsledky analýzy metód

- Chceme odhaliť plagiáty čo najpresnejšie – potrebujeme byť schopní odhaliť čo najviac typov útokov
- Vyvážiť pomalosť stromových metód využitím abstrakcie na redukovanie veľkosti porovnávaných kódov
- V rámci analýzy sme v dostupnej literatúre nenašli publikované využitie kombinácie týchto dvoch prístupov
- Zameranie na jazyk C – dominantný jazyk vo výučbe na fakulte

Návrh riešenia

- Kombinácia abstraktných syntaktických stromov a abstrakcie zdrojového kódu
- Navrhnutá metóda je:
  - pomalšia ako textové a tokenové metódy
  - schopná odhaliť väčšinu plagiátorských útokov
- Zrýchlenie porovnávania pomocou stromov, zmenšením veľkosti porovnávaných súborov



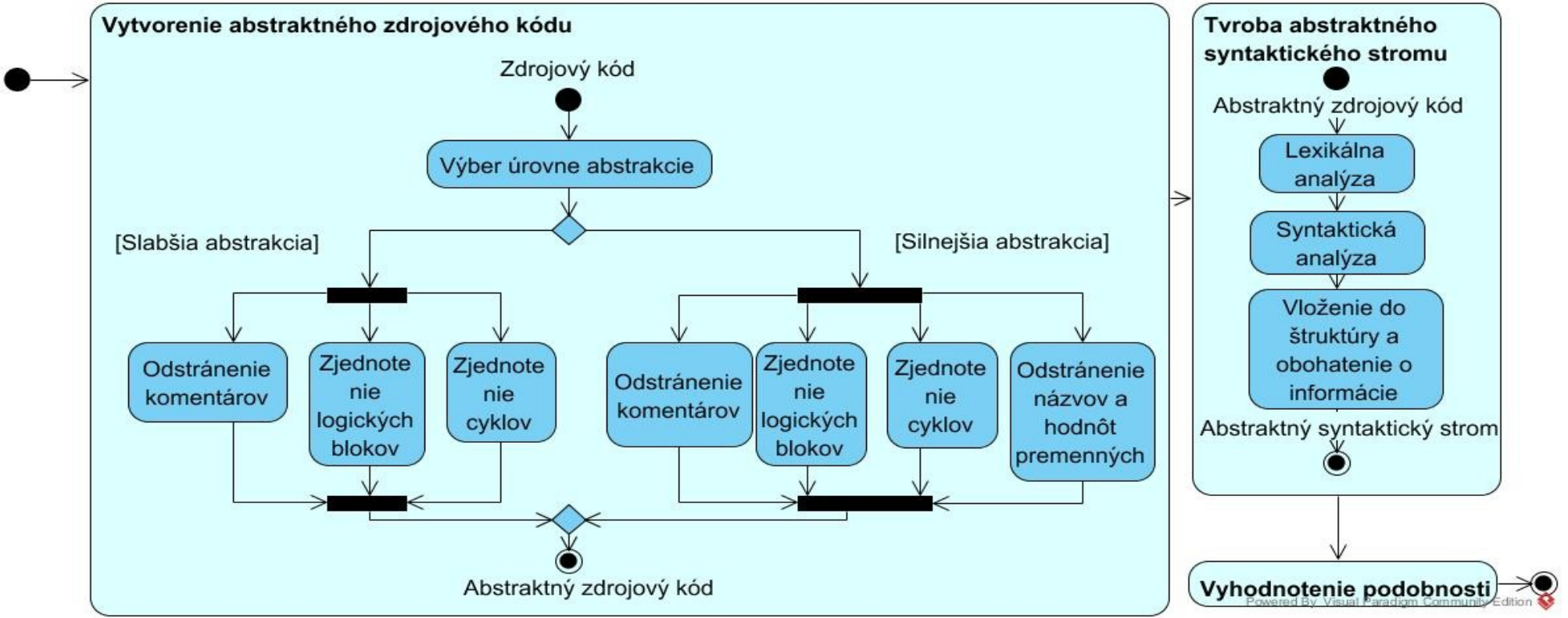
Abstraktné syntaktické stromy

- Dátová štruktúra využívaná na reprezentáciu zdrojového kódu
- Strom pozostáva z uzlov obsahujúcich typ uzla, hodnotu uzla a odkaz na deti uzla
- Tvorba prebieha pomocou lexikálnej a syntaktickej analýzy
- Pre porovnávanie je vhodné vypočítavať hash hodnotu uzlov
- Pri porovnávaní počítame hash hodnotu uzla ako súčet hash hodnôt detí uzla a výsledok hash funkcie pre daný uzol

Abstrakcia zdrojového kódu

- Odstraňovanie nepotrebných častí zdrojového kódu
- Výrazne zmenšenie veľkosti oproti pôvodnému zdrojovému kódu
- Nebezpečie odstránenia podstatnej časti zdrojového kódu
- Možnosť využitia viacerých úrovní abstrakcie
- Dá sa využiť pri porovnávaní zdrojových kódov v rôznych programovacích jazykoch
- Naše riešenie porovnávalo dve úrovne abstrakcie
  - Slabšia úroveň – odstránenie komentárov, unifikovanie blokov
  - Silnejšia úroveň – slabšia úroveň a odstránenie mien a hodnôt premenných

Spracovanie dát



Implementácia riešenia

- Implementované v jazyku ruby
- Na implementáciu abstraktných syntaktických stromov sme použili opensource knižnicu, v ktorej sme museli upraviť výpočet hash hodnoty uzla
- Na výpočet hash hodnoty uzla sme využili vstavanú hash funkciu ruby,
  - Na báze murmur hash funkcie
  - Najrýchlejšia v rámci porovnávaných (xxHash, SHA, MD5)
- Postupne sme skúšali viaceré parsre (parser, rubyparser, jparser, cast)
- Vybraný parser pracuje s knižnicou na štruktúru a vie spracovať abstrahovaný zdrojový kód

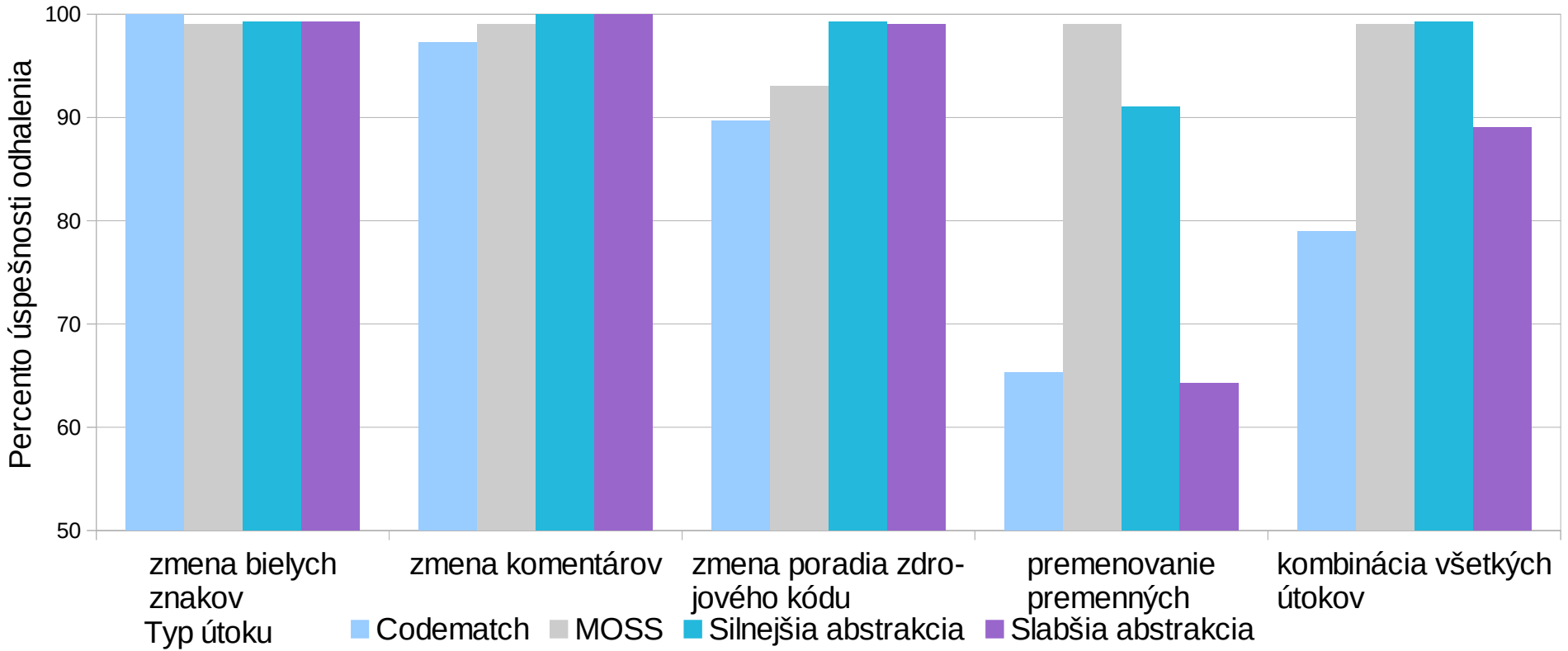
Implementácia riešenia

- Aplikácia pozostáva z dvoch modulov
- Základný modul - tokenizátor slúži na predspracovanie,
  - Obsahuje pravidlá pre zdrojové kódy v jazyku C
  - Rozšíriteľný o pravidlá pre ďalšie jazyky
- Vyhodnocovací modul pozostáva z funkcií parsra a porovnávania abstraktných syntaktických stromov

Overenie riešenia

- Simulovanie identifikovaných útokov na vybranú vzorku zdrojových kódov v jazyku C z predmetu DSA
- Porovnanie výsledkov s existujúcimi nástrojmi CodeMatch a MOSS
- Implementované riešenie odhalilo väčšinu nami identifikovaných útokov
- Pri silnejšej abstrakcii sme dosahovali zmenšenie počtu uzlov súboru približne o 30% voči slabšej abstrakcii
- Útok zmeny algebraických výrazov odhalilo len moje riešenie a len v blokoch (cykloch a logických blokoch)

Graf pecentuálnej úspešnosti odhalenia vybraných typov útokov jednotlivými metódami



Zhrnutie

- Navrhnutá metóda umožňuje vyhodnotiť podobnosť zdrojových kódov
- Riešenie je rozšíriteľné o podporu ďalších programovacích jazykov
- Kombinácia abstrakcie a abstraktných syntaktických stromov ešte nebola riešená
- Aplikáciou silnejšej abstrakcie sme redukovali veľkosť porovnávaných zdrojových kódov, približne o 30%, a tým zlepšili výpočtovú efektívnosť
- Riešenie by sa dalo vylepšiť vytvorením vlastnej dátovej štruktúry a tiež parsera

Možné rozšírenia

- Možnosť automatizácie výberu abstrakcie
- Výber prvkov zdrojových kódov, ktoré budú abstrahované
- Možnosť výberu metriky podobnosti
- Možnosť vizualizácie riešenia
- Možnosť spätnej transformácie na zdrojový kód
- Vlastný parser (prípadne využitie ANTLR alebo Parsey McParseface od Google)

Možné metriky podobnosti

- Celkový počet uzlov nachádzajúcich sa v inom strome
- Porovnávanie jedného s jedným
  - Pomalšie
  - Možnosť ukončenia porovnávania vetvy pri nájdení zhodného uzla
- Porovnávanie jedného s viacerými
  - Rýchlejšie
  - Vyššia zhoda kvôli listom
- Počet krokov potrebných na zmenu jedného stromu na druhý

Možnosti vizualizácie

- Výpis stromov s vyznačenými zhodnými uzlami
- Výpis zhodných uzlov
- So spätnou transformáciou možnosť zobrazenia zhodných častí zdrojového kódu