Programozási feladatok megoldásaiban elkövetett hibák felderítése annotációk segítségével a ProgCont rendszerben

Detection of errors in solutions of programming tasks using annotations in ProgCont system

Dr. PÁNOVICS János[1], Dr. KÁDEK Tamás[2], Dr. BIRÓ Piroska[3], Dr. KÓSA Márk[4]

Debreceni Egyetem, Informatikai Kar
H–4028 Debrecen, Kassai út 26.
telefon: +36 52 512 900, honlap: https://www.inf.unideb.hu/
e-mail: [1]panovics.janos@inf.unideb.hu, [2]kadek.tamas@inf.unideb.hu,
[3]biro.piroska@inf.unideb.hu, [4]kosa.mark@inf.unideb.hu

Abstract

In response to the challenges caused by the pandemic situation, the University of Debrecen, Faculty of Informatics, had to switch to distance learning on several occasions. This form of education requires significantly more autonomy on the students, which increases the role of independent preparation. Our ProgCont system, which has existed for more than a decade, needs to respond to these challenges. We wanted to support independent preparation by improving the system's automatic feedback, particularly, by adding new annotations to the subtasks of the selected tasks, which help identify errors in partially good solutions. In this article, we describe how to develop annotations based on three different use cases and summarize the experience and results obtained with the new technique.

Keywords: ProgCont system, annotations, automated evaluation, programming

Kivonat

A járványhelyzet okozta kihívásokra reagálva a Debreceni Egyetemen Informatikai Karán is több ízben kellett átállni a távolléti oktatásra. Ez az oktatási forma lényegesen több önállóságot feltételez a hallgatók részéről, melyben megnő az önálló felkészülés szerepe. Több mint egy évtizedes múltra visszatekintő ProgCont rendszerünkben is szükségessé vált, hogy reagáljunk az ebből adódó kihívásokra. Az önálló felkészülést azzal kívántuk segíteni, hogy javítjuk a rendszer automatikus visszajelzéseit, olyan új annotációkkal ellátva a kitűzött feladatok részfeladatait, melyek segítenek a részben jó megoldásokban fellelhető hibák feltárásában. Cikkünkben bemutatjuk az annotációk kialakításának útját, három különböző felhasználási területet alapul véve, és összefoglaljuk az új technikának köszönhető tapasztalatokat és eredményeket.

Kulcsszavak: ProgCont rendszer, annotációk, automatizált kiértékelés, programozás

BEVEZETÉS

2020 tavasza óta jelentős átalakuláson ment át az oktatás. Reagálni kellett a járványhelyzet okozta kihívásokra [1]. A Debreceni Egyetemen is ki kellett dolgozni azokat a módszereket, amelyek segítségével a távolléti oktatás hatékonyan kivitelezhető. Erősödött az e-learning eszközök szerepe az oktatásban, és megjelentek új módszerek, gyakorlatok, melyek vélhetően a pandémia lecsengését követően is velünk maradnak. Az elmúlt időszak egyik nagy kihívását az jelentette, hogy a hallgatók rákényszerültek az önálló felkészülésre, melynek szerepe jelentősen megnőtt, és melynek támogatására hatékony eszközöket kellett találjunk (elektronikus tananyagok, visszanézhető előadások, online feladatsorok stb.).

A hagyományos programozásoktatásban, a számítógépes laborokban erősen építettünk a közös hibakeresésre az elkészített megoldásokban. A távolléti oktatás ezt a lehetőséget sokszor elvette vagy elnehezítette. Olyan eszközre van szükség, amely a hallgatót képes rávezetni az elkövetett hibára.

Lassan egy évtizede használjuk a programozás oktatásában a ProgCont rendszert, amely a programozási feladatok megoldásainak automatikus kiértékelésére való [2], [3], [4], [5]. Kézenfekvő, hogy a távolléti oktatás során is támaszkodjunk erre a képességére. Ugyanakkor ezt a szoftvert nem úgy alakítottuk ki, hogy az az önálló felkészülést támogassa, ehelyett eredetileg csupán az automatikus kiértékelésre koncentráltunk. A rendszert először programozási versenyeken vetettük be, ahol a hibakeresés a versenyzők fontos feladata, és az értékelés kimerül pusztán annyiban, hogy helyesnek vagy helytelennek minősítjük a beküldött programot. A rendszer ezen tulajdonsága a később hozzáadott funkciók esetében is megmaradt. Régóta használjuk számonkérések lebonyolítására és gyakorló feladatok kitűzésében is. Utóbbiak esetén a hibásnak ítélt programokban a hibák felderítését már nem a rendszer végezte, ha a hallgató e tekintetben segítségre szorult, akkor azt az oktatótól várhatta el. Ez az önálló felkészülés támogatásához természetesen nem elegendő.

Olyan megoldást szerettünk volna kidolgozni, amely hibás feladatmegoldás esetén rávezeti a hallgatót a hiba lehetséges okára. Ebben segítségünkre volt a ProgCont rendszer azon tulajdonsága, ahogy a beküldések értékelése zajlik. Egy-egy feltöltött program többször is lefuttatásra kerül több különböző bemenettel, és annak függvényében kerül elfogadásra vagy elutasításra, hogy ezek esetén a megfelelő, elvárt kimenetet produkálja vagy sem. Ebből következően megjelenhetnek olyan részben jónak ítélt feltöltések, melyek bizonyos teszteseteken ugyan átmentek, de nem mindegyiken. Eredetileg ugyanakkor arról semmilyen információnk nincs, hogy mely körülmények (milyen jellegzetességgel rendelkező bemenetek) esetében volt helyes a program. Logikus fejlesztési irány, hogy a ProgCont rendszer ezen visszajelzéseinek bővítésével irányítsuk ilyenkor a hibakeresés folyamatát, ötleteket adjunk arra, hogy hol kell keresni a problémát. E gondolatoktól vezérelve vezettük be a teszteset-annotációk lehetőségét.

ANNOTÁCIÓK A PROGCONT RENDSZERBEN

A teszteset-annotáció olyan kulcs-érték párokként megjelenő címkéket takar, melyek a teszteset sajátosságainak leírására szolgálnak. Ha bizonyos sajátossággal rendelkező teszteken egy program átmegy, másokon azonban nem, az már segíthet beazonosítani a hiba okát. Innentől kezdve nem csak azt tudjuk elmondani egy hibás beküldésről, hogy részben jó, hanem azt is, hogy a tesztesetek mely részében.

A tesztesetek sajátosságait két csoportba sorolhatjuk:

- globális vagy a konkrét feladattól független annotációk, olyan kulcs-érték címkék, melyek több különböző feladat teszteseteiben is előfordulhatnak;
- problémaspecifikus annotációk, amelyek a tesztesetekben megjelenő olyan sajátosságokat emelnek ki, amelyek kizárólag az adott probléma megoldása során érdekesek.

Elsőként a következő globális annotációkat vezettük be:

- **`empty input = true:`** a tesztesethez üres bemenet tartozik
- **`empty input = false:`** a tesztesethez nem üres bemenet tartozik
- **`test cases = none:`** a teszteset nem tartalmaz egyetlen feladatot sem
- **`test cases = one:`** a teszteset csak egy feladatot tartalmaz
- **`test cases = more:`** a teszteset egynél több feladatot tartalmaz

A ProgCont rendszerben megtalálható feladatok közül igen sok esetben azt várjuk el a fejlesztendő programtól, hogy a bemenetén megjelenő több feladatot is fel tudjon dolgozni. Ezen esetekben a bemenet feldolgozása, a bemenetben található feladatok egymástól való megkülönböztetése plusz kihívást jelent. Érdemes ennek tesztelésére olyan teszteseteket is készíteni, amelyek kizárólag egy feladatleírást tartalmaznak, és olyanokat is, amelyek többet vagy éppen egyet sem. Néhány feladat esetében előfordul, hogy az üres bemenet is megfelel a bemeneti specifikációnak, megjegyzendő ugyanakkor, hogy az egyetlen feladatot sem tartalmazó teszteset (**`test cases = none`**) nem feltétlenül üres bemenetű (**`empty input = true`**).

A rendszerünkben számos különböző feladattípus megtalálható (sztringműveletek, gráfkereső algoritmusok, aritmetikai problémák, geometriai feladatok stb. [6], [7]), ezért valódi segítséget a problémaspecifikus annotációk nyújthatnak. Ezek esetében minden feladatot külön-külön kell megvizsgálni, így egy teljes feladatsor problémaspecifikus annotációkkal történő ellátása időigényes feladat.

A meglévő feladatsorok esetében még nem számoltunk az annotációk lehetőségével, ezért az eredetileg kidolgozott tesztesetek nem is feltétlenül jól annotálhatók, jellemzően az annotációk

összegyűjtése után új, ezeknek megfelelő tesztesetek legyártása szükséges. Az új tesztesetekkel végzett értékelés szokásos következménye, hogy megváltozik a sikeresen, illetve sikertelenül teljesített tesztek aránya, legalábbis a nem is teljesen jó és nem is teljesen rossz beküldések esetében. Jól megfogalmazott annotációk mellett arra számítunk, hogy a részben jó megoldások száma is megnő. Valójában az annotációk a részben jó megoldásokban végzett hibakeresést képesek elősegíteni.

EREDMÉNYEK

3.1. Első kísérlet az annotációk bevezetésére

Az annotációk bevezetésének ötletét egy, a rendszerünkben már régóta létező és számos beküldéssel rendelkező feladattal próbáltuk ki. Az „Időpontok”[1] című feladatra esett a választásunk, melyben olyan programot kellett készítenie a hallgatóknak, amely a bemeneten érkező 24 órás formátumú időpontokat az angol nyelvterületen szokásos 12 órás formátumúvá konvertálja. 2014 és 2021 között összesen 1387 beküldést rögzítettünk erre a feladatra, melyek közül mindössze 30% volt helyes. Két problémaspecifikus jellemzőt emeltünk ki:

- külön tesztesetekben gyűjtöttük össze az időpontokat aszerint, hogy az órákat miként kell konvertálni: **`hour = 0, hour = 1-11, hour = 12, hour = 13-23`**;
- külön tesztesetekben gyűjtöttük össze azokat az időpontokat, amelyekben a perc értéke egy számjegyű (**`minute = 0-9`**) és amelyekben két számjegyű (**`minute = 10-59`**).

Tekintettel arra, hogy két független jellemzőről van szó, ez összesen nyolc tesztesetet igényelt, ezen felül további egy teszteset készült, melyben a bemenet üres volt, és ezért **`empty input = true, test cases = none`** globális annotációkat kapott. Az annotációk megfogalmazása után elkészített új tesztesetekkel a részben jó megoldások aránya 13%-ról 56%-ra nőtt. Arra is fény derült, hogy az egy számjegyű percek (63%) és a 12 óra kezelését (59%) rontották el a legtöbben.

3.2. Dolgozatfeladatsor kiértékelése annotációk segítségével

Az első kísérlet fényt derített arra is, hogy az annotációk segítségével kinyerhető plusz információ az oktatók munkáját is megkönnyíti. A feltöltött programok vizsgálata nélkül is meg lehet határozni a hallgatók számára problémás részfeladatokat, melyeket aztán ki lehet emelni a következő tanórán, és előre fel lehet hívni rájuk a figyelmet a következő évi csoportnak. Ennek megfelelően a következő annotációval ellátott feladatok egy számonkérés feladatsorából kerültek ki, ahol az újraírt és annotált tesztesetek hasznosságát egy 2019-ben 45 hallgató által már teljesített vizsgafeladatsor újraértékelésével tudtuk leméri. A vizsgafeladatsor a *Programozási nyelvek 1* tárgyhoz[2] tartozott, és egy időben három gyakorlati csoport is teljesítette, melyek teljesítményét is össze kívántuk hasonlítani. Az új teszteseteken két globális annotációt alkalmaztunk:

- a korábbiakban már bevezetett **`test cases = one, test cases = more`**;
- **`stretched limits = true, stretched limits = false`** annotációkkal láttuk el a teszteseteket annak függvényében, hogy a tesztesetben megadott bemeneti adatok tartalmazták-e a feladatleírásban rögzített szélsőségeket vagy sem.

A fentiek mellett minden feladathoz egy-egy kettő vagy három különböző értékkel rendelkező problémaspecifikus annotáció is megfogalmazásra került. Ezzel 8–12 új tesztesetet kellett készíteni.

Az eredmények részletes értékelését [8]-ben tettük közzé. A csoportok által leggyakrabban elkövetett hibák összevetéséből jól kiolvasható volt, hogy mennyiben igényelnek más segítséget, útmutatást az elsőévesek csoportjai, mint az évismétlőket magába foglaló gyakorlati csoport. A kezdők számára a **`test cases = more`** annotációval ellátott feladatok bemenetének feldolgozása okozza a kihívást, míg az évismétlők esetében a problémaspecifikus annotációk jelölik ki azon részfeladatokat, melyeken a leggyengébben teljesítenek. Az nem volt meglepetés, hogy a hallgatók ezen két csoportját érdemes külön kezelni, de immár arról is kapunk automatikusan némi iránymutatást, hogy hogyan.

[1] Magas szintű programozási nyelvek 1, 2014. március 11., K12 ZH, Időpontok című feladat, https://progcont.hu/progcont/100029/?pid=200502

[2] Programozási nyelvek 1, 2019. április 10., Sz10 ZH, https://progcont.hu/progcont/100301/

3.3. Gyakorlást segítő annotációk

Nem szem elől tévesztve az eredeti célt (az önálló felkészülés lehetőségeinek javítását), egy gyakorló feladatsort is átalakítottunk abban a reményben, hogy a hallgatók a részben jó feladatairól annotációk által olyan információt kaphatnak, amely behatárolja az elkövetett hiba lehetséges helyét. Ehhez egy 2021-ben készített gyakorló feladatsor három feladatát választottuk ki, melyeket mind a mai napig ajánlunk az önálló felkészüléshez.

Az, hogy közel egy évtizednyi beküldés áll a rendelkezésünkre, ismét segített visszaellenőrizni, hogy az újraírt és felannotált tesztesetek segítségével megkapható-e a kívánt plusz információ. Ennek elemzését részletesen a [9] cikkben mutattuk be.

A három feladathoz összesen 47 darab új problémaspecifikus annotáció és ennek megfelelően számos új teszteset készült. Az egyes feladatok átalakítása során mindig sikerült növelni a feladatok részben jó megoldásának arányát. Eredetileg egy-egy teszteset sok különböző részfeladatot egyszerre vizsgált, és amennyiben bármelyik megoldása sikertelen volt, úgy a program a teljes teszteseten elbukott. Az annotációk által leszűkített, egy-egy részproblémára összpontosító tesztek alapján ellenben sok korábban teljesen hibásnak ítélt program mégis részben jónak bizonyult. A részben jó megoldások számának növelése azért is kívánatos, mert ezek esetében tudunk így ötleteket adni a javításhoz.

ÖSSZEFOGLALÁS

A feladatok tesztjeinek annotálása újragondolt tesztesetek megfogalmazásával jár együtt. Ezen tesztesetek segítségével újraértékelve a beküldött megoldásokat azt tapasztalhatjuk, hogy megnövekedik a részben jónak ítélt megoldások száma a hibásnak ítélt beküldések kárára. Ez azért is szerencsés, mert az annotációk valójában a részben jó megoldást beküldő hallgatók hibakeresését könnyíthetik meg. A részfeladatok annotációkkal történő beazonosítása az oktatók számára nem kevésbé hasznos, rávilágít az egyes feladatok során a hallgatóság által elkövetett tipikus hibákra, lehetőséget teremtve ezáltal a differenciált oktatás erősítésére. Az elvégzett munka után kialakultak olyan feladatsorok, amelyek támogatják az önálló felkészülés lehetőségét. Bár próbáltunk globális annotációkat megfogalmazni (melyek előnye az lenne, hogy automatikusan hozzákapcsolhatók a meglévő tesztesetekhez), vizsgálataink során mégis úgy tapasztaltuk, hogy ezek nem veszik fel a versenyt a problémaspecifikus jellemzőket megjelenítő egyedi annotációkkal. A rendelkezésünkre álló feladatgyűjtemények átalakítása jelentős kihívást jelent számunkra, a kezdeti eredmények alapján azonban többszörösen megtérül a befektetett munka.

KÖSZÖNETNYILVÁNÍTÁS

A kutatást a „Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein” (EFOP-3.6.3-VEKOP-16-2017-00002) című projekt támogatta. A projekt a Magyar Kormány és az Európai Szociális Alap társfinanszírozásában valósult meg.

IRODALMI HIVATKOZÁSOK

[1] Kádek, T. & Biró, P.: *A távolléti oktatás hatásai a ProgCont rendszerre*, in ENELKO SzámOkt 2020, 2020, pp. 104–109, https://ojs.emt.ro/index.php/enelko-szamokt/article/view/325/264.

[2] Biró P. & Kádek, T.: *Automatic evaluation of programming tasks at the University of Debrecen*, INTED2020 Proceedings, pp. 3522-3527. https://doi.org/10.21125/inted.2020.0994.

[3] Kádek, T. & Biró P.: *Úton a szakköralkalmazás felé a ProgCont API-val*. INFODIDAKT 2019 konferencia, Zamárdi, Webdidaktika Alapítvány, https://people.inf.elte.hu/szlavi/InfoDidact19/Manuscripts/KTBP.pdf.

[4] Kádek, T. & Biró, P.: *A ProgCont API: programozási feladatok megoldásainak újszerű kiértékelése*. SZÁMOKT 2019, Temesvár, Románia: Erdélyi Magyar Műszaki Tudományos Társaság (EMT) kiadó, 2019, pp. 191-195.

[5] Tóth, R., Kósa, M., Kádek, T. & Pánovics, J.: *The development of evaluation systems at the Faculty of Informatics, University of Debrecen*, in INTED2019 Proceedings, 2019, pp. 5552–5559.

[6] Biró, P. & Kádek, T.: *The Mathability of Computer Problem Solving with ProgCont*, Acta Polytechnica Hungarica, 2021 – accepted for publications.

[7] Kádek, T., Kósa, M. & Pánovics, J.: Informatikai versenyfeladatok, Digitális Tankönyvtár, 2014, https://regi.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0103_04_informatikai_versenyfeladatok/index.html (Utolsó letöltés: 2021. 09.10).

[8] Biró, P., Kósa, M., Pánovics, J. & Kádek, T.: *Test case annotation of programming tasks in the ProgCont system for differential education and strengthening stand-alone preparation opportunities*, in 13th International Conference on Education and New Learning Technologies, EDULEARN21 Proceedings, 2021, pp. 5433-544, https://doi.org/10.21125/edulearn.2021.11091.

[9] Balázs, P., Biró, P., Kádek, T., Kósa, M. & Pánovics, J.: *Mathability and exploring mathematical skills of programmers with exercises' annotations*, in 2021 12th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), 2021 – accepted for publication.