A formális szemantika interaktív oktatása tételbizonyító rendszerben

Horpácsi Dániel[1], Németh Dávid János[2], Kaposi Ambrus[3]

[1]daniel-h@elte.hu, [2]ndj@inf.elte.hu, [3]akaposi@inf.elte.hu
ELTE IK

Absztrakt. Az ELTE informatikai képzéseiben gazdag múltra tekint vissza a programozási nyelvek formális szemantikájának oktatása. Karunkon, más elméleti tárgyakhoz hasonlóan, hagyományosan a Formális szemantikát is tábla és papír köré szerveztük, a 2008-ban indult programtervező informatikus mesterképzésben pedig a tárgy gyakorlata sajnos egyáltalán nem kapott helyet. Sokéves kihagyás után az előző félévben ezt géptermi kurzusként indítottuk újra, amely keretében immáron interaktív és végrehajtható módon mutatjuk be a programozási nyelvek tételbizonyító rendszerben történő modellezését. Cikkünkben ismertetjük a gyakorlat megtervezése és lebonyolítása során szerzett tapasztalatainkat; kiértékeljük a tételbizonyító rendszer használatának látható következményeit; végül szót ejtünk a módszerünk más kurzusokra való kiterjeszthetőségéről is.

Kulcsszavak: formális szemantika, tételbizonyító rendszer, Coq, interaktív gyakorlat

1. Bevezetés

Egyetemi szintű programozó-képzésben elengedhetetlen a matematikai, formális leírások használata, oktatása. Az ELTE programtervező mesterképzésen kötelező tantárgy a Formális szemantika, amelynek keretében a hallgatók elsajátítják a programozási nyelvek matematikai alapokon nyugvó definíciós módszereit. A tárgy nagy mennyiségű absztrakt fogalommal és formális jelölésrendszerrel dolgozik, amelyet a kevésbé matematikus beállítottságú hallgatók nehezen fogadnak be. Mint minden „elméleti" tárgynál, itt is szükséges a matematika „csomagolása", amellyel könnyebben emészthetővé tesszük az egyébként száraz anyagot. Részletes, gyakorlatias példákkal lényegesen könnyíthető az absztrakt fogalomrendszer megértése, ám a fiatal generáció akkor fogadja be a legkönnyebben az elvont fogalmakat, ha megérinthetővé, kipróbálhatóvá, kísérletezhetővé tesszük azokat. A formális szemantika kurzus gyakorlatán nemcsak prezentálunk szemléletes példákat, de bevezettük a tételbizonyító rendszer használatát is, amelynek segítségével a definíciókon túl a tételek és bizonyítások is interaktívan oktathatóak.

2. A formális szemantika kurzus

A programozási nyelvek a természetes nyelvekkel ellentétben mesterségesek: azért alkotta őket az ember, hogy a számítógépet irányítsa. A természetes nyelvekkel szemben a programozási nyelvek mondatainak egyértelmű és pontos jelentéssel kell rendelkezniük, amely alapján a számítógép mérlegelés és további kérdések nélkül végre tudja hajtani őket, ahol ezen végrehajtás hatása matematikai precizitással megfogalmazható. Számítógépes programok jelentésfogalmával, és általában a programozási nyelvek formális definícióival foglalkozik a Formális szemantika.

Az ELTE programtervező matematikus és programtervező informatikus MSc szakok tanulmányi hálójának mindig fontos eleme volt a Formális szemantika tantárgy, aminek keretében a programozóhallgatók elsajátíthatják a programozási nyelvek definícióinak megértéséhez és formalizációjához szükséges matematikai eszközöket. A kurzus körbejárja a szintaxis, statikus szemantika és dinamikus szemantika matematikai leírásának lehetőségeit, beleértve az operációs, denotációs és axiomatikus stílusban megadott szemantikákat. A különböző definíciós módszerek minden esetben valós, elterjedt (főleg imperatív) programozási absztrakciók szemantikájának leírásával kerülnek bemutatásra, ami

elősegíti a megértést, kapcsolja a megszerzendő tudást a korábbi tanulmányokhoz, és hasznos a meglévő tudás elmélyítésében is.

2.1. A kurzus célja

A kurzusnak több célja is van. Az első és legfontosabb a programozási nyelvek precíz, formális definiálásának népszerűsítése, továbbá a formális definíciók szerepének tisztázása a nyelvek megértésében, összehasonlításában, valamint a programok funkcionális tulajdonságainak, helyességének vizsgálatában. A hallgató a különböző általa ismert programozási absztrakciók (pl. vezérlési szerkezetek) formális megadása útján mélyebben megérti ezen absztrakciók működését, kölcsönhatásait, korlátait, valamint átfogóbb képet kap a különböző nyelvekben különbözőképp leírt absztrakciók hasonlóságairól és különbségeiről (pl. elágazások és ciklusok, függvényabsztrakciók és kivételkezelés megannyi formája a modern nyelvekben).

A formális szemantikadefiníció motivációja és közvetlen gyakorlati haszna mellett legalább ugyanolyan fontos a matematikai eszközkészlet megismertetése és megtanítása. A hallgató átismétli a fordítóprogramok tanulmányozása során megismert szintaxisdefiníciós módszereket, majd megismeri az operációs és denotációs definíciós módszert a jelentés megadására. Az operációs szemantikákat (small-step és big-step) a konfigurációknak, valamint azok átmeneteit induktívan megadó levezetési szabályok formátumának és megtervezésének megértésével, a denotációs szemantikákat pedig a szemantikus függvények felírásával, a kompozicionalitás elvének megismerésével sajátítja el a hallgató.

Az alkalmazott matematikai eszközkészlet felhasználja a halmazelmélet és a csoportelmélet egyes elemeit, induktív definíciókat és azokon végzett mintaillesztést, indukciót, a fixpontelmélet releváns részeit. A tárgy által feldolgozott téma, valamint a feldolgozás módja igen absztrakt, amit a hallgatók egy része nehezen tud befogadni. A példák sokat segítenek egy-egy új fogalom megértésében, ám a gyakorlat tudja csak igazán elmélyíteni az előadás anyagát.

2.2. A kurzus korábbi megvalósítása

A Formális szemantika tárgy a programtervező matematikus képzésben előadás és gyakorlat keretében került oktatásra, ahol a gyakorlaton az előadás által tárgyalt matematikai eszközök használatával foglalkoztak a hallgatók. Habár volt gyakorlat, az papír alapú volt, nem alkalmazott semmilyen interaktív környezetet a definíciók és tételek kipróbálására, a szemantikákkal való kísérletezésre. 2008-ban aztán a programtervező informatikus képzésben a tárgy gyakorlata nem is kapott helyet, így az osztott képzésben már csupán előadás keretében tanulhatták az MSc hallgatók a Formális szemantikát. Ez nyilvánvalóan nem tett jót a kurzus hatékonyságának és népszerűségének, hiszen a tantárgy kiterjedt és absztrakt fogalomrendszerét gyakorlás nélkül nehéz teljes mértékben megérteni és elsajátítani.

A 2008-2018 közötti tízéves időszak közepén oktatóváltás történt, aminek következményeként gyakorlatiasabb stílust kapott az előadás. Az órák elején és végén népszerű programozási nyelvek bonyolult konstrukciói kerültek bemutatásra és elemzésre, a kedvcsinálás mellett pedig a hallgatók bevonása is nagyobb hangsúlyt kapott: a denotációs szemantikákat implementáltuk a Haskell programozási nyelvben, az operációs szemantikákat pedig formalizáltuk egy termátíró keretrendszerben, ezzel végrehajtható, módosítható, kipróbálható definíciókat adva a hallgatók kezébe. Ezek a lépések érezhetően javítottak a tárgy megítélésén, de a következő természetes lépés a gyakorlat újbóli bevezetése volt.

2.3. Az új gyakorlat

A 2018-ban újragondolt programtervező informatikus mesterképzési tantervbe visszakerült a Formális szemantika gyakorlat, amely újabb lendületet tudott adni a szemantika gyakorlatias oktatásának. A tárgy oktatói – akik egyúttal a cikk szerzői – nekiláttak a gyakorlat megtervezésének és lebonyolításának. A cikk a sokéves kihagyás utáni első gyakorlati csoportok új tematika szerinti oktatásának tapasztalatait mutatja be.

A gyakorlat tematikáját és számonkérést úgy kellett megalkotnunk, hogy bármelyik elsőéves programtervező mesterhallgató képes legyen elsajátítani a tudást és elvégezni a kurzust. Az oktatók között abban teljes egyetértés volt, hogy a Formális szemantika gyakorlatias oktatásnak elengedhetetlen eleme a definíciókkal való kísérletezés, újabb és újabb nyelvi elemek megadása; ahogy abban is egyetértettünk, hogy ennek a kísérletezésnek interaktívan, számítógépen kell történnie. Mivel a definíciók mellett a tételek kimondását és bizonyítását is szerettük volna gyakoroltatni a hallgatókkal, olyan rendszert kellett választanunk, amely tételbizonyításra és/vagy ellenőrzésre is alkalmas. A következő fejezet részletesen tárgyalja, hogy mely funkcionális nyelvek és tételbizonyítók közül választottunk, és hogy milyen múltra tekint vissza a gépi tételbizonyítás bevonása az egyetemi gyakorlatokba.

3. Tételbizonyító rendszerek az oktatásban

A tételbizonyító rendszerek olyan szoftverek, amelyek lehetőséget adnak:

- absztrakt modellek minden részletre kiterjedő definiálására,
- ezen modellek felett értelmezett állítások kimondására,
- majd az igaz állítások matematikailag megalapozott szabályrendszer mentén történő, gépileg ellenőrizhető, interaktív bizonyítására.

A Formális szemantika kurzus géptermi gyakorlatának tervezése során több ilyen rendszert is fontolóra vettünk, figyelembe véve az irodalomban hozzájuk köthetően fellelhető oktatási tapasztalatokat, valamint a körvonalazott céljainkhoz való illeszthetőségüket.

A Haskell programozási nyelv [6] szigorú értelemben véve ugyan nem tekinthető tételbizonyító rendszernek, viszont tisztán funkcionális jellege és egyszerű szintaxisa alkalmassá teszi programozási nyelvek modellezésére. Benne az absztrakt szintaxist algebrai adattípusokkal, a szemantikát pedig függvényekkel írhatjuk le, denotációs stílusban, végrehajtható módon. A gyakorlatban építettek már rá szemantika kurzust például az Umeåi Egyetemen (Svédország) [11], viszont az alkalmazását mi, dedikált tételbizonyítási funkcionalitás hiányában, elvetettük.

Az Agda [1] egy, a Haskellénél erősebb, függő típusrendszert támogató funkcionális programozási nyelv. Az ilyen típusrendszerek sajátossága, hogy a bennük előforduló típusokat értékektől tehetjük függővé, leírva például a statikusan 4-hosszú vektorok típusát. Egy ennyire erős típusrendszer, a típusok és a matematikai állítások között kapcsolatot teremtő Curry-Howard izomorfizmus révén lehetőséget ad arra, hogy benne állításokat típusok, bizonyításokat pedig futtatható programok formájában, konstruktívan írjunk le. Az Agda nyelvet jelenleg is aktívan használják szemantika oktatására többek között az Edinburgh-i Egyetemen (Egyesült Királyság) és a Vermonti Egyetemen (Amerikai Egyesült Államok) [10], mi viszont célszerűbbnek láttuk egy olyan rendszer bevezetését, amely élesebben leválasztott, a hallgatók által papíron megszokottakhoz közelebb álló tételbizonyító funkcionalitással rendelkezik.

Ennek a követelménynek eleget tesz az Isabelle/HOL tételbizonyító [7], mely szintén, gyakorlatban alátámasztottan alkalmas programozási nyelvek szemantikájának oktatására (Müncheni Műszaki Egyetem, Németország) [2]. Ugyanakkor a típusrendszere nem támogat függő típusokat, így benne szükségessé válhat, hogy egy erősebb típusrendszerben egyébként a típusokba (állításokba) kódolt, és így a rendszer által automatikusan kezelt információkat a tételkimondások és bizonyítások során manuálisan kelljen karban tartani. Az ezzel járó komplexitást más eszközben potenciálisan elkerülhetjük.

A választásunk végül a Coq tételbizonyító rendszerre [5] esett. Az Agda nyelvezetéhez hasonló, funkcionális és függő típusos leírónyelve kellően kifejező és konstruktív, viszont dedikált, imperatív stílusú bizonyítási résznyelvvel is rendelkezik, ami szerencsés összhangban áll az általánosságban imperatív paradigmához szokott hallgatók előismereteivel. Széleskörű pedagógiai alkalmazása (például a

svédországi Chalmers Műszaki Egyetemen és az amerikai egyesült államokbeli Pennsylvaniai Egyetemen is felhasználták szemantika oktatására) [3], részletesen kidolgozott segédanyagai [9] és aktív támogatottsága [4] tovább növelték a belé vetett bizalmunkat.

4. Interaktív formális szemantika gyakorlat

A Formális szemantika gyakorlat feladata, hogy segítse az előadás anyagának megértését az ott tárgyalt definíciók, példaprogramok, valamint további példák formalizálásával. Mivel a hallgatók túlnyomó többségének nincsen gyakorlata semmilyen tételbizonyító rendszerrel, a gyakorlati óráknak gyorsan és hatékonyan kell a hallgatót rutinos tételbizonyító (Coq) felhasználóvá tenniük anélkül, hogy a hangsúly a szemantika fogalmairól a tételbizonyító technikai részleteire tolódna el.

4.1. Irányelvek

Előadáson megértés, gyakorlaton kódolás.

Az előadások során a tananyag esetenként kifejezetten összetett példákon kerül bemutatásra, hogy a példák minél inkább mintázzák a valódi programozási nyelvekben előforduló konstrukciókat. Ezeket részletesen tárgyalja az előadás, de a formalizációjuk komplexebb annál, mint amit egy kezdő magától le tud írni a tételbizonyító rendszerben. A kész formalizációkat kiadhatnánk a hallgatóknak (a korábbi végrehajtható szemantikákkal ezt tettük), de az új gyakorlatokon még inkább a hallgató bevonása a cél. A gyakorlati órák többségén nem meglévő formalizációkat olvasnak vagy módosítanak a hallgatók, hanem informális és fél-formális definíciókat, tételeket visznek be a rendszerbe önállóan, "üres lappal" indulva. Ezzel biztosítjuk, hogy az idő nem a komplex példák megértésére és magyarázatára, hanem – még ha kicsi példákon is, de – a formalizáció gyakorlására kerül felhasználásra. Ennek köszönhetően a szemantikán marad a fókusz, viszont a folyamatos önálló kódolással viszonylag rövid idő alatt gyakorlatot szerez a hallgató a tételbizonyító rendszer használatában.

Először megértés, aztán kódolás.

Az előző gondolat persze nem jelenti azt, hogy a hallgatónak a definíció, tétel vagy bizonyítás megértése nélkül kellene elkezdenie kódolni. Sőt, a gyakorlatoknak feladata, hogy az anyag elméleti megértését különválassza a Coqban való formalizációjától. A gyakorlat kisebb feladatokat próbál teljes egészében megoldatni a hallgatókkal, de a tételbizonyítóban való formalizációt minden esetben megelőzi az elmélet tökéletes megértése. Ha szükséges, ehhez a táblára vagy füzetbe is írunk, de ez az „analóg" kidolgozás jóval kisebb hangsúlyt kap, mint más matematikai tantárgyak gyakorlatain.

Fokozatosság a tételbizonyító megismerésében.

A Coq megannyi nyelvi elemmel és funkcióval segíti a kényelmes és hatékony formalizációt, de ha kezdettől fogva arra törekszünk, hogy a hallgató jó Coq programozó legyen, nem szemantikát fogunk tanítani neki, hanem Coqot szemantika példákkal. A tételbizonyító nyelvét és funkcióit fokozatosan vezetjük be gyakorlatról gyakorlatra három-négy újabb elemmel (hasonló sorrendben, mint a Programming Language Foundations könyv [9]), ennek köszönhetően a hallgató fokozatosan szerez jártasságot a Coq fogalomrendszerében és nyelvezetében, nem veszik el annak részleteiben.

Fokozatosság a tételek bonyolultságában.

A hallgatóknak sokszor nemcsak a tételbizonyítókkal, a tételbizonyítással sincs elég tapasztalatuk. A Formális szemantika gyakorlatoknak ezért sokszor bizonyításelméletet is kell tanítaniuk, és ezzel egy időben a bizonyítás formalizációjának módszerét is tárgyalniuk kell. Ahhoz, hogy a hallgató ne veszítse el idejekorán a kedvét és lelkesedését a tételbizonyító rendszer használata iránt, kellő körültekintéssel kell egyre hosszabb és bonyolultabb bizonyításokat készíteni és készíttetni a gyakorlatok során. Tapasztalataink szerint egyszerű esetszétválasztással megadott függvények formalizációja például nem

jelent gondot egyik mesterszintű hallgatónak sem, de a levezetési szabályok induktív definíciókká alakítása már esetenként fejtörést okoz. Kellő előkészítéssel folyamatos sikerélményt tudunk biztosítani a hallgatónak, aminek következményeként a tételbizonyító rendszer használatát pozitív élményként fogja elkönyvelni, és ez fontos mérföldkő lehet a szakmai fejlődésében.

Folyamatos munka és számonkérés.

A gyakorlat számonkérését úgy terveztük meg, hogy hétről hétre dolgoznia kelljen a hallgatónak: a 90-90 percnyi előadás és gyakorlat mellett heti egy házi feladat és heti egy rövid zárthelyi dolgozat tartja folyamatos fejlődésben a kurzus résztvevőit. Minden kiadott feladat a Coq rendszerben kerül megoldásra, tovább gyorsítva a tételbizonyító használatának elsajátítását. A feladatokat úgy találtuk ki, hogy a heti házi feladat mindig a következő zárthelyit mintázza, tehát a házi feladat megoldása elősegíti a gyakorlati számonkérés jó teljesítését; ez tovább növeli a hallgatói motivációt a folyamatos munkában.

4.2. Tematika

A gyakorlat tematikája elsősorban az előadás meglévő anyagára és a már említett könyv [9] megközelítésére lett felépítve. Szerencsére a könyv kitűnő alapot ad a Coq gyors és gördülékeny bevezetésére, miközben az általa tárgyalt szemantika példák sok helyen kitűnően egybecsengnek az előadáson tárgyaltakkal, mert mindkét forrás az imperatív programozási paradigma absztrakcióit dolgozza fel. Rövid mérlegelés után úgy állítottunk össze a tematikát, hogy a félév első felében a könyvet [9], a félév második felében pedig elsősorban az előadásanyagot követi és formalizálja. A gyakorlat tematikája a következő fontosabb elemekből áll össze:

- Bevezető, a tételbizonyító rendszer használata. Egyszerű típusok Coqban: bool és nat, kapcsolódó lemmák kimondása, bizonyítás esetszétválasztással.
- Szintaxis megadása induktív definícióval (mély beágyazás), egy kifejezésnyelv leírása denotációs szemantikával, kiértékelés szerkezeti rekurzióval. Statikus szemantika: kifejezésen értelmezett egyszerű leképezések (literálok száma, műveletek száma), induktív bizonyítások a kifejezésnyelv szerkezete alapján. Kifejezések leképezése ekvivalens kifejezésekre (optimalizációs lépések), a szemantika megőrzésének induktív bizonyítása.
- Állapot (változókörnyezet) bevezetése, a kifejezésnyelv szintaxisának és szemantikájának bővítése. Állapotokkal kapcsolatos lemmák kimondása és bizonyítása kifejezések speciális eseteire. Statikus elemzés: szabad és kötött változók, zárt kifejezés fogalmának megadása függvénnyel és induktívan definiált relációval. Definíciók ekvivalenciája, zártságra vonatkozó lemmák kimondása és bizonyítása.
- A kifejezésnyelv small-step és big-step operációs szemantikájának megadása induktívan definiált relációval. Levezetési szabályok formalizálásának gyakorlása. A megadott denotációs és operációs szemantikák ekvivalenciájának bizonyítása a kifejezésnyelvre.
- Imperatív utasítások nyelvének formalizálása. Szintaxis, denotációs szemantika. Fixpontelmélet, a terminálás kérdése. Big-step operációs szemantika formalizálása, levezetések, determinisztikusság. A denotációs és operációs szemantikák ekvivalenciája a While nyelvre. Programminták ekvivalenciája (egyszerű optimalizációs lépések helyességének belátása). További vezérlési szerkezetek formalizálása operációs és denotációs szemantika megadásával: for ciklus, repeat-until ciklus.
- Haskell kód exportálása a helyes Coq programból, gyakorlati alkalmazások.

Bonyolultabb fogalmak elmélyítése

Amint említettük, vannak olyan fogalmak és módszerek, amelyeket még mesterhallgatók is lassan, fokozatosan tudnak csak elsajátítani. A gyakorlati kurzus talán abból a szempontból a legfontosabb,

hogy ezeket a bonyolult, absztrakt fogalmakat fokozatosan nehezedő példákon mutatja és gyakoroltatja be. Az eddigi tapasztalataink szerint a következő fogalmak tanítása lényegesen hatékonyabb lett a gyakorlatoknak köszönhetően:

- *Levezetési szabályok és levezetés*: sokszor nem világos a hallgatók számára, hogy egy operációs szemantikát leíró szabályrendszert mért pont úgy formalizálunk, ahogyan; mit lehet a (levezetési szabály premisszáját és konklúzióját elválasztó) vonal fölé, a vonal mellé, és a vonal alá írni. A Coqban megadott induktív típus rákényszeríti a hallgatókat, hogy pontosan megértsék a szabályokat és a formalizmus elemeit. A Coq szemantikájából adódó mintaillesztés kapcsán a hallgató könnyebben el tudja képzelni, miért és hogyan kerülnek alkalmazásra a levezetési szabályok.
- *Strukturális indukció*: bonyolultabb szerkezetű típusok esetén nem mindig érthető a hallgató számára az induktív bizonyításhoz használt séma – nem világos, mit mi után, milyen hipotézis mentén kell belátni. A papíros bizonyítások sokszor könnyebben olvashatóak, de sajnos bármit „eltűrnek"; ezzel ellentétben Coqban interaktívan tudnak bizonyítani, a tételbizonyító rendszer lépésről lépésre vezeti őket a bizonyításon, világosan megadva, hogy mik a hipotézisek és mit kell azokból belátni, valamint hibás lépéseket nem enged meg.
- *Kompozicionalitás*: a kompozicionalitás elvét sokszor csak bemagolják a hallgatók, de nem válik teljesen világossá számukra, hogy miért is kell így megadnunk a denotációkat. Amikor a programok vagy kifejezések szerkezete szerinti indukcióval bizonyítanak, jól látszik a tételbizonyító rendszerben, hogy a kompozicionalitásnak köszönhetően az indukciós hipotéziseik egybevágnak azokkal a belátandókkal, amelyek az összetett szerkezetekre adódnak. További következmény, hogy a denotációs szemantikák rekurzív szerkezetekre való megadásánál kompozicionális definíciók esetében egészen biztosan szerkezetileg csökkenő (primitív) rekurzív hívások adódnak, amit a Coq biztosan termináló rekurziónak fog értelmezni.

5. A tételbizonyító rendszer használatának kihívásai

A tételbizonyító rendszer tudáselmélyítés szempontjából előnyösnek tekinthető tulajdonságai között akadtak olyanok is, amelyek megoldandó pedagógiai problémákat is felvetettek. Ezek egy része a Coq alapnyelvének jellemzőiből következett, míg mások mögött a papírra vetett és a gépbe kódolt bizonyítások közötti részletszigorúság állt.

A programozási nyelvek szintaxisának és operációs szemantikájának megadásához használt induktív típuskonstrukciók helyenként nehezen voltak értelmezhetők az objektumorientált nyelvekhez szokott hallgatók számára – ezt a Coqban definiált típusokhoz Java-beli megfelelőt mutatva igyekeztünk ellensúlyozni.

A hallgatók korábban technikai részleteket mellőző vagy átugró bizonyításokat írtak papíron. Ehhez képest Coqban a bizonyítások legkisebb részleteit is ki kellett fejteni, ez gyakran nehézséget okozott. Ilyen jelenség például, hogy míg többszörös esetszétválasztást szükségessé tevő bizonyításokban papíron elegendő csak az érdekes ágakat kiemelni a triviálisak közül, addig egy tételbizonyítóban akkurátusan foglalkozni kell minden egyes lehetőséggel. Szerencsére ez a probléma sem bizonyult megoldhatatlannak: bizonyítási taktikákat ismétlő nyelvi elemekkel az érintett gépi bizonyítások újra közel hozhatók a megszokott papíros megfelelőikhez.

6. Eredmények

A gyakorlat bevezetésének eredményességét objektíven a gyakorlati és a kollokviumjegyekkel tudjuk mérni. A gyakorlati számonkérés módjából és értékelési skálájából fakadóan az első gyakorlatot végző

évfolyam gyakorlati jegyei nagyon jók lettek, így a képzés eredményességét, a minőség javulását inkább az előadás-vizsgajegyeken kívánjuk szemléltetni.

A Formális szemantika vizsga hagyományosan szóbeli felelet, amely egyrészt a hallgató lexikális tudását, másrészt az anyag elsajátításának módját, a megértés mértékét vizsgálja. A hallgatók nagy számára való tekintettel évek óta egy írásbeli beugró dolgozat is része a vizsgának, amely egyrészt kiszűri a felkészületlen hallgatókat a szóbeli vizsgáról, másrészt a sikeres beugróval szóbeli felelet nélkül elégséges jegyet szerezhetnek a téma iránt kevésbé lelkesedő hallgatók.

6.1. Vizsgaeredmények

A gyakorlat egyértelműen javította a vizsgaeredményeket. Ez valószínűleg annak köszönhető, hogy a hallgatók az elméleti anyag bemagolása helyett a fogalmak folyamatos megértésének köszönhetően értve tudtak felkészülni az előadás tematikájából. A következő számszerűsített javulást könyvelhettük el a 2017/2018/2 és a 2018/2019/2 félévek között.

Átlagok javulása

Az összes vizsgázót tekintve az érdemjegyek átlaga 2,6-ről 3,14-ra nőtt a beugró feladatok jellegének vagy a tételsornak a megváltoztatása nélkül. A vizsgáztató személye is változatlan maradt. Az átlagba minden érdemjegyet (1-5) beleszámítottunk.

Több szóbeliző

Habár a sikeres írásbeli beugró után a hallgatók megszakíthatják a vizsgájukat egy elégséges érdemjeggyel, szóbeli vizsga tétele nélkül, a szóbelire jelentkező hallgatók aránya 33%-ról 48%-ra emelkedett. Tehát a gyakorlat olyan mértékben javította a hallgatók magabiztosságát az anyag megértését illetően, hogy az összes hallgatónak majdnem fele szóbeli vizsgán szerezte az érdemjegyét, a korábbi egyharmad helyett.

Szóbelizők átlagának javulása

A szóbeli felelet valamely anyagrész részletes bemutatását jelenti, ahol a hallgatónak arról kell bizonyságot tennie, hogy nemcsak megtanulta, de érti is az előadáson elhangzottakat. Nem kis meglepetésünkre a szóbeli feleletek érdemjegyeinek átlaga is emelkedett, 3,9-ről 4,2-re, ami annak is köszönhető, hogy sokkal többen tudták jól elmagyarázni a kihúzott tételüket, és szereztek jó vagy jeles érdemjegyet.

6.2. Szubjektív eredmények

Oktatói szemmel egészen szembetűnő javulás mutatkozott a szóbelik gördülékenységében, a hallgatók magabiztosságában. Jobban megértették a fogalomrendszert, a bizonyítások lépéseit és szerkezetét. A vizsgajegyekben mutatkozó javulás szubjektíven is érezhető volt.

A hallgatók részéről is kifejezetten pozitív visszajelzéseket kaptunk a kurzussal kapcsolatban. Ugyan a Formális szemantika előadás nem volt kifejezetten népszerűtlen, a gyakorlatnak és az interaktív oktatási módszernek köszönhetően a hallgatók lelkesebben álltak hozzá a tárgyhoz. Úgy gondoljuk, hogy a fent vázolt mód az egyetlen, amellyel matematikai tárgyakból hosszú távú, használható tudást tudunk átadni az átlagos mesterhallgatónknak.

6.3. Coq más kurzusokban

Célunk, hogy a Formális szemantika kurzus esetében szerzett tapasztalatainkat a képzésünk más, hasonló jellegű tárgyainak átalakításához is felhasználjuk. Az aktuális, 2019/2020/1 félévben a Nyelvek típusrendszere című, a formális típuselméletbe gyakorlatiasan és szemléletesen bevezető tárgyat egészítettük ki egy Coqra alapozott gyakorlattal, részben a Formális szemantika mintájára. Az átállás ugyan nem zökkenőmentes (a gépi modellezés által megkívánt alaposság miatt a gyakorlat az előadásnál lassabban tud csak haladni), de a Formális szemantikánál érzékelhető (a félév végéig egyelőre jórészt szubjektív) pozitívumok itt is mutatkoznak.

Az eddig említett két kurzuson kívül a jövőben más potenciális tárgyak is szóba jöhetnek a programozáselmélet, a logika és a matematika témaköréből. Az előző félév során (2018/19/2) például az Osztott rendszerek specifikációja és implementációja című kurzusunkhoz született egy hallgatói formalizáció [8]. Hosszabb távon gondolkozva meggyőződésünk, hogy a gépileg ellenőrzött modellezés releváns tárgyakon átívelő szabványosítása számottevő mértékben járulna hozzá a hallgatók szakmai jártasságának elmélyítéséhez, és így az informatikai képzések minőségének, színvonalának emeléséhez.

7. Összefoglalás

Cikkünkben bemutattuk, hogy miként indítottunk el egy tételbizonyító rendszerre épülő, interaktív géptermi kurzust egy korábban sokáig csak elméleti előadást magában foglaló, programozási nyelvek formális szemantikájával foglalkozó tárgyhoz.

A mögöttes motivációnkat az eredeti kurzus képzésünkön belüli pozíciójának, történetének ismertetésével magyaráztuk, kiemelve azt az előnytelen helyzetet – fontos elméleti anyag a megértését segítő gyakorlat nélkül –, ami sok éven át korlátozta a tárgy céljainak megvalósítását.

Az új gyakorlat alapelvének, a tételbizonyító szoftver használatának rögzítése után számba vettük a szóba jövő rendszereket, majd ezek közül kiválasztottuk a Coqot, amely a jól szeparált modellező és tételbizonyító résznyelvei, kifejező típusrendszere, valamint gazdag segédanyag-irodalma révén ideális jelöltnek bizonyult.

A tananyag megalkotásának irányelveit és a meglévő előadásanyaggal való kapcsolatát, illetve a gyakorlati kurzus optimális lebonyolítását szerintünk leginkább segítő számonkérési módszert három pontban foglaltuk össze: először megértés, aztán kódolás; fokozatosság; folyamatos munka és számonkérés. Ezekre építve egy tematikatervezetet is felvázoltunk, amelyet (többek között) abban a szellemben állítottunk össze, hogy pusztán előadással nehezen átadható fogalmak (levezetés, indukció, kompozicionalitás) elsajátíthatóságán javítani tudjunk.

A tételbizonyító rendszer bevezetése természetesen technikai és pedagógiai kihívásokkal is járt, melyek közül a sok hallgató számára szokatlan leírónyelvet, továbbá a gépi bizonyítások papíros megfelelőik után helyenként (logikailag indokolt, de a gyakorlatban) csak nehezen védhető komplexitásnövekedését emeltük ki. Ezen problémákra megoldásokat is javasoltunk rendre a megszokott nyelvekkel való párhuzamvonás, valamint az automatizáció használatának formájában.

A bevezetett géptermi gyakorlat eredményességét az objektív összehasonlításra lehetőséget adó vizsgajegyek mentén értékeltük. Örömmel tapasztaltuk, hogy ezekben számottevő javulás volt kimutatható: a korábbi félévekhez képest nemcsak az átlagok emelkedtek, de a sikeres beugrót követő fakultatív, elégségesnél jobb jegy szerzéséhez szükséges szóbeli feleletre is több hallgató vállalkozott. Kísérletünk sikerességét látva módszerünket az aktuális félévben alkalmazzuk egy másik tárgyra is, a jövőben pedig tervezzük több tárgy irányelveink szerinti átalakítását is.

A programok magas szintű, precíz modellezése azontúl, hogy elengedhetetlen részét képezi a szoftverhelyesség matematikai eszközökkel történő vizsgálatának, hozzájárul az informatikushallgatók programnyelvi jártasságának, absztrakciós készségének fejlesztéséhez is. Bízunk abban, hogy az itt megosztott tapasztalataink legalább bátorítást, de akár tanácsokat, segítséget is adnak ahhoz, hogy ezt és a hozzá kapcsolódó, alapvetően elméleti tudományterületeket gyakorlatias és interaktív módon oktatva hozzuk közelebb a hallgatókhoz.

Irodalom

1. Az Agda programozási nyelv (2007-től)
https://wiki.portal.chalmers.se/agda/pmwiki.php (utoljára megtekintve: 2019.11.04.)

2. Tobias Nipkow, Gerwin Klein: *Concrete Semantics with Isabelle/HOL*. (2014-től)
http://concrete-semantics.org/ (utoljára megtekintve: 2019.11.04.)
3. Coq In The Classroom (2017-től)
https://github.com/coq/coq/wiki/CoqInTheClassroom (utoljára megtekintve: 2019.11.04.)
4. A Coq tételbizonyító GitHub repozitóriuma (2007-től)
https://github.com/coq/coq (utoljára megtekintve: 2019.11.04.)
5. A Coq tételbizonyító rendszer (1989-től)
https://coq.inria.fr/ (utoljára megtekintve: 2019.11.04.)
6. A Haskell programozási nyelv (1990-től)
https://www.haskell.org/ (utoljára megtekintve: 2019.11.04.)
7. Az Isabelle tételbizonyító rendszer (1986-tól)
https://isabelle.in.tum.de/ (utoljára megtekintve: 2019.11.04.)
8. Donkó István: *orsi-formalization*. Az ELTE-IK Osztott rendszerek specifikációja és implementációja című tárgyának hallgatói Idris-formalizációja. (2019)
https://github.com/Isti115/orsi-formalization (utoljára megtekintve: 2019.11.04.)
9. Benjamin C. Pierce et al.: *Programming Language Foundations*. Software Foundations series, volume 2. Elektronikus könyv. (2018)
http://www.cis.upenn.edu/~bcpierce/sf (utoljára megtekintve: 2019.11.04.)
10. Philip Wadler: *Programming Language Foundations in Agda*. Konferenciaanyag, 21st Brazilian Symposium (SBMF 2018). Salvador, Brazil, November 26–30, 2018.
11. O. Johansson: *Programming language semantics*. Kurzusanyag. Umeåi Egyetem, Umeå, Svédország (1993-1998)
http://oldwww.cs.umu.se/local/kurser/TDBC05/ (utoljára megtekintve: 2019.11.04.)