Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Radostyán Bertalan

Molnár Balázs

AUTOMATIZÁLT FELADATGENERÁLÁS ABSZTRAKCIÓS KOMPETENCIA TESZTHEZ

KONZULENS

Dr. Forstner Bertalan

BUDAPEST, 2016

Tartalomjegyzék

Összefoglaló

Az emberi intelligencia mérésében a figurális absztrakciós teszt (pl. RPM – Raven's Progressive Matrices [1]) egy széles körben elfogadott módszer. A teszt olyan feladatokat tartalmaz, melyben a kitöltőnek egy 3x3-as mátrix utolsó elemét kell kitalálni, és nyolc lehetséges válasz közül kiválasztania az előző nyolc mátrixelem alapján. A mátrix különböző ábrákat, alakzatokat tartalmaz, melyek valamilyen logika szerint összefüggnek egymással. Az elmúlt egy év alatt ennek egy okostelefonon futó változatán dolgoztunk. A feladatok nehézsége adaptívan változott a teszt kitöltése során, vagyis a soron következő feladat nehézsége az előzőekre adott válaszok alapján került meghatározásra. A feladatok véletlenszerűen generáltak voltak előre meghatározott minták és logikák alapján, ezzel sziginifikánsan növelve a különböző feladatok számát ezek kézzel való megadásához képest. A következő lépés a feladatok mögött álló minták automatikus generálása, így még tovább növelve a lehetséges feladatok számát. A logikák, vagy a mi elnevezésünk szerint szabályok leírására egy általunk kitalált leíró nyelvet használunk. Az egyik legfontosabb probléma azon szabályok leszűrése, melyek értelmesnek és emberek által megoldhatónak mondhatóak. Ezt egy sor feltétellel próbáljuk megoldani, melyek ha igazak egy adott szabályra, akkor azt megoldhatónak tekintjük. A második probléma a feladatok nehézségének meghatározása. Egy úgynevezett „naív" nehézség számolható a szabály különböző tulajdonságai alapján. Ez a nehézségi szint nem lesz pontos, viszont különböző módszerekkel később finomhangolható lesz.

Abstract

Figural abstraction teszt (e.g. RPM – Raven's Progressive Matrices [1]) was proven to be one of the best ways of measuring intelligence. This type of test contains examples in which the test subject is presented with a three by three matrix of different figures with the last one missing. The goal is to find the appropriate answer out of eight different possible solutions by determining the logic behind the elements of the matrix. Over the last year we have been working on a smartphone version of such a test. This test included adaptiveness, which means that the difficulty of the next given exercise was determined by the previous answers. In this application the exercises were randomly generated using predetermined patterns. This way the number of different exercises were significantly higher than in the case of such a test, where all the exercises are created one by one. The next step is to make the generation of these patterns automatic, thus raising the number of differently generated exercises even more. We use a descriptional language to describe the logic behind the exercises, calling the different logics rules. One of the major challenges is to determine which of these rules are considered solvable by humans. We formulate conditions and if these conditions are met in a generated rule, the rule is considered solvable. The second problem is determining the difficulty of each rule. A „naive" difficulty level can be determined for each rule based on different different properties of the given rule. This difficulty level can by later on fine-tuned using different methods.

1 Bevezetés

Az elmúlt néhány évtizedben a technológia fejlődése jelentős mértékben felgyorsult. A különböző megjelenő új technológiák és eszközök könnyebbé és kényelmesebbé tették mindennapjaink legkülönböző aspektusait. Az okoseszközök megjelenésének hála az információ szinte végtelen mennyiségben rendelkezésre áll bárki számára egy gombnyomás hatására. Egy terület azonban érezhetően le van maradva a technológia fejlődéséhez képest, elmúlasztva a rengeteg lehetőséget, mely a legmodernebb eszközök használatából adódik: az oktatás.

Manapság szinte minden diák zsebében ott lapul egy okoseszköz, melyet az iskolák által szolgáltatott egyéb eszközökkel egyetemben felhasználva jelentősen növelni lehetne az oktatás hatékonyságát, például személyre szabott feladatokkal, gyakorlással stb. A kutatócsoportunkban jelenleg is folyó kutatás egy olyan rendszer kialakítását célozza meg, melyben a tanulók adaptívan változó nehézségű feladatokon keresztül tudnák fejleszteni különböző képességeiket, ezzel biztosítva azt, hogy a diák megfelelő mértékű kihívást kapjon. Az adaptivitáshoz természetesen különböző visszacsatoló elemekre van szükség. Mindamellett, hogy a tanuló jelenlegi teljesítményét figyelembe vesszük a következő feladat nehézségének megállapításakor, számos egyéb faktort is bevehetünk a számításba. Ilyen elem lehet például még az illető érdeklődése, melynek meghatározására különböző módszerek állnak rendelkezésre. Egy jól alkalmazható eljárás például Gyarmathy Éva pszichológus által kifejlesztett ún. Érdeklődési Térkép. Ennek alkalmazásával képesek vagyunk felmérni az egyén érdeklődési területeit egy gyors, kb. 15 percet igénybe vevő teszt segítségével. Ehhez szintén készítettünk egy okostelefonokon elérhető alkalmazást (1. ábra). Egyéb visszacsatolási elem lehet még a tanuló fiziolgóiai állapota is. Szintén a kutatócsoportunkban zajlik annak kutatása, hogy egyszerűbb, megfizethető és hordozható, különböző fiziológiai jelek mérésére alkalmazható eszközök (pl. EEG, EKG) által mért adatok alapján hogyan lehet megállapítani az alany mentális állapotát és ezt hogyan lehet felhasználni a kiadandó feladat nehézségének megállapításában, illetve a tanulók különböző módon történő motiválására. [2]

Az imént felsoroltak mellett az egyik legfontosabb visszacsatolási tényező a tanuló kognitív képességeinek ismerete. Ha ismerjük az illető ezen képességeit, képesek

vagyunk már a kezdetektől olyan nehézségű feladatokat adni, melyek a tanuló képességeinek megfelelőek. A legutóbbi kutatások alapján a különféle sztenderd intelligencia tesztek nem megfelelőek erre a célra. Az egyik legnagyobb gyengesége ezen teszteknek, hogy a tesztalanyokat nem szeparálják különböző szempontok szerint, hanem egy nagy csoportként kezelik őket és értékelik a teljesítményüket. A mi elképzelésünk szerint olyan kognitív képességeket mérő tesztekre van szükség, melyek önsztenderdizálóak, vagyis minden nemnek, korosztálynak és mentális betegségben küzdőknek külön csoportot alakítva mérik az egyének képességeit. Ezek természetesen változhatnak a jövőben, így olyan tesztekre van szükség, melyek nagyszámú és változatos feladatokat tartalmaznak.

Az emberi intelligencia és különböző kompetenciák mérésének problémája hosszú múltra tekint vissza. Az évtizedek során rengeteg különböző megoldás született erre a problémakörre, melyek többféle kompetenciát, illetve különböző aspektusát mérik az emberi intelligenciának. [3] Ahogy a technológia fejlődött, felmerült az igény ezen tradicionális tesztek digitalizálására. Egyre több teszt elérhető az interneten online, illetve különböző eszközökön applikációk formájában. A hagyományos tesztelés különböző lépéseit képesek vagyunk automatizálni, például az értékelési folyamatot, vagy éppen a feladatok elkészítésének feladatát.

Az egyik, mai napig is aktívan használt és elfogadott módszer a figurális absztrakciós teszt. Az első dokumentált implementációja egy ilyen típusú tesztnek az RPM (Raven's Progressive Matrices) [1] teszt. A figurális absztrakciós tesztek olyan feladatokból állnak, melyek 3x3-as mátrixokat tartalmaznak. Ezen mátrixok celláiban különböző ábrák láthatók. A tesztalany feladata, hogy az mátrix utolsó, hiányzó elemét nyolc lehetséges válasz közül kiválassza. A mátrix elemei valamilyen logikai kapcsolatban állnak egymással, ezt felfedezve a válasz egyértelműen kikövetkeztethető. Az évek során a teszt különböző alkalmazási területekre sztenderdizálva is lett. [4]

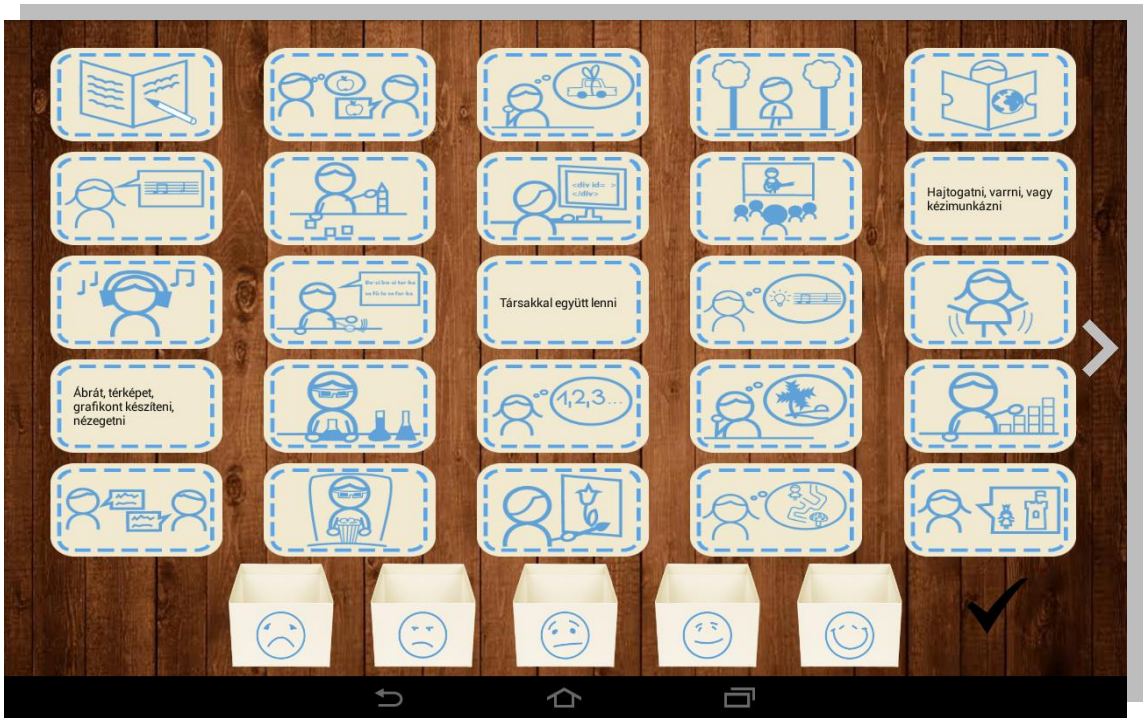
Ahogyan azt korábban említettem, az egyik legfontosabb cél egy olyan teszt létrehozása, mely rengeteg változatos feladatot tartalmaz. Erre a célra remekül megfelel az figurális absztrakciós teszt. A jelenlegi implementációk, melyek esetében sokszor a feladatokat kézzel határozzák meg, nem képesek elég nagy számú feladatot előállítani, emiatt a pszichológusok és matematikusok egyik nyitott kérdése az ilyen tesztek generálása. Természetesen egy ilyen módszer megalkotása során sok kihívással szembesülünk. Az első és talán legnehezebb kihívás, hogy a feladatok, melyeket

automatikusan generálunk értelmesek, vagyis egyértelműen megoldhatóak legyenek. Emellett fontos, hogy az előállított feladatok nehézségét meg tudjuk becsülni, majd későbbi mérésekkel pontosítani azt. Miután a rendelkezésre állnak a legenerált feladatok, egy érdekes kérdés a válaszjelöltek előállítása. Ebben a lépésben például figyelnünk kell arra, hogy a lehetséges válaszok valamilyen tekintetben hasonlítsanak a tényleges megoldásra. Ezzel próbáljuk megakadályozni azt, hogy a megoldás túlságosan egyértelmű legyen. Végül pedig szükséges, hogy változatosan tudjuk vizualizálni a feladatokat. A tradicionális tesztekhez képest a vizualizáció is sokkal bővebb lehet, hiszen nem vagyunk a papír statikusságához kötve. Itt egy komoly kihívás, hogy elkerüljük a feladatok megjelenítéséből adódó esetleges kétértelműségeket, vagyis olyan vizualizációt alkalmazzunk, mely nem teszi a feladatot ezáltal értelmetlenné.

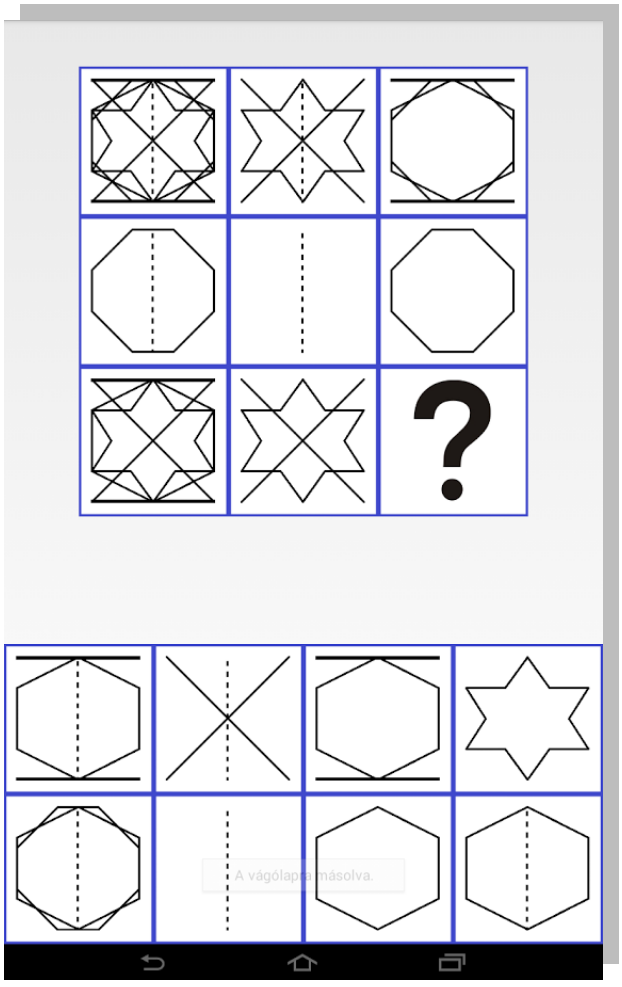
Az elmúlt év során lefejlesztettünk egy okostelefonon (Android) futó alkalmazást (1. ábra), mely az imént említett teszt egy implementációja. A teszt során a feladatok véletlenszerűen generálódnak előre definiált mintákat követve. Ezek a minták lényegében megegyeznek a feladatok mögötti logikákkal, melyeket a tesztalanyoknak kell felderíteni. Az alkalmazás kb. 15 különböző ilyen mintát tartalmaz, melyeknek további variációi illetve különböző nehézségi szintjei vannak. A feladatok tradicionálisan kézzel való elkészítéséhez képest ezzel a módszerrel sikerült nagy mértékben növelni a rendelkezésre álló feladatok számát. Másik fontos eredmény az emberi erőforrásra való igény csökkentése a tesztfeladatok létrehozása folyamán.

A következő lépés, amelyről ez a dolgozat is szól, egy olyan módszer megalkotása, mely az imént bemutatott alkalmazásban is alkalmazott logikai minták automatizált generálását teszi lehetővé. Az előredefiniált mintákból való feladatgenerálást már több, aktív használatban lévő teszt is alkalmazza, viszont ennél általánosabb megoldás még csak kevés született. Természetesen vannak már erre a problémára irányuló kutatások is, melyek közül a legtöbb az RPM tesztet veszi alapul. [5] Ezek a megoldások megpróbálják az RPM tesztet elemezve a benne található különböző logikai mintákat felismerni és általánosítani. Az ilyen módszerek hátránya, hogy mind a feladatok mögötti logikákat, mind pedig a megjelenítésüket tekintve egyaránt túl behatároltak és ezáltal az előállítható feladatok száma nem elegendően nagy. Ezzel szemben az előnyük, hogy egy olyan implementációra (RPM) alapoznak, mely többszörösen is bizonyított az elkészülte óta.

A mi megoldásunk egy másik aspektusból próbálja megközelíteni a problémát. A módszer során csupán a figurális absztrakciós tesztek alaptulajdonságait használjuk fel ahelyett, hogy egy konkrét implementációra alapoznánk. Ezzel egy jóval általánosabb és szabadabban implementálható megoldást hozunk létre. A dolgozat során először egy generáló algoritmust mutatunk be, mely képes nagyszámú logikai minta előállítására. Ehhez definiálunk egy formális leíró nyelvet. Következő lépésben megszűrjük az előállított minták halmazát aszerint, hogy melyeket tartunk értelmesnek, megoldhatónak. Ezután a válaszlehetőségek automatizált generálásáról lesz szó. A következő lépés a legenerált minták vizualizációja lesz, mely konkrét feladatok előállítását teszi lehetővé. Ezt követően a feladatok nehézségének meghatározásáról lesz szó, melyben ismertetünk néhány módszert, amikkel jól becsülhetőek illetve finomhangolhatóak a feladatok nehézségi szintjei. Végül egy áttekintés keretében bemutatjuk az eddig elért eredményeinket, illetve a tesztelés során használt általunk fejlesztett tesztalkalmazást.



1. ábra Pillanatkép az általunk fejlesztett ÉrdeklődésiTérkép mobilalkalmazásból



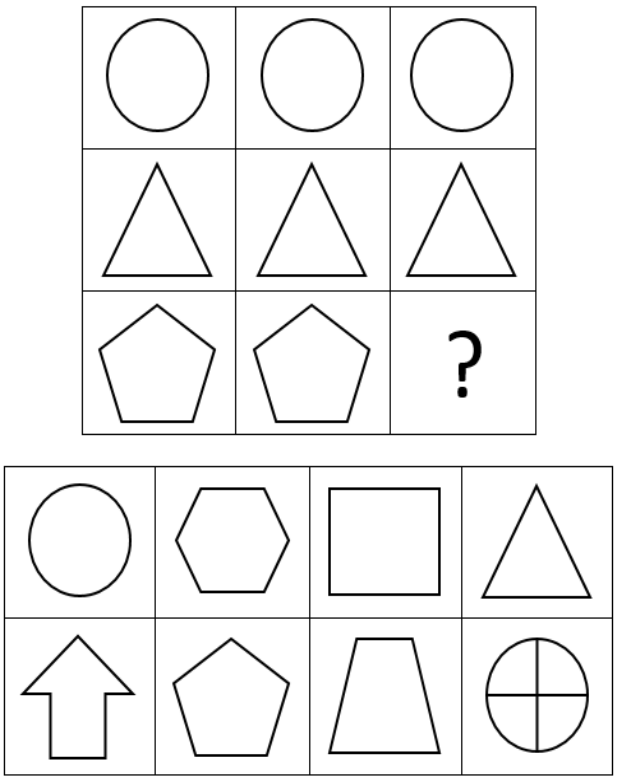
2. ábra Pillanatkép az általunk fejlesztett AdaptiveR mobilalkalmazásból

2 Generáló algoritmus

2.1 Szabályok és azokat leíró formai nyelv

2.1.1 Szabályok

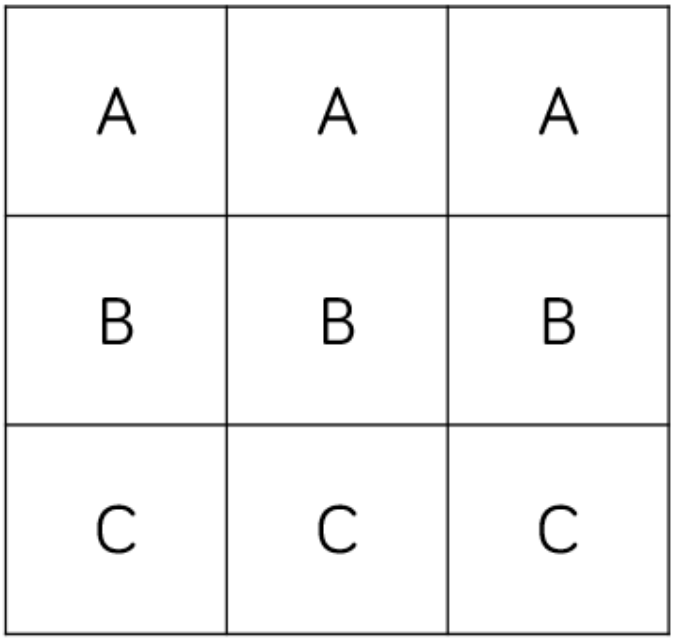
A figurális absztrakciós tesztek olyan feladatokból állnak, melyek tradicionálisan háromszor hármas mátrixokat tartalmaznak (de vannak implementációk, melyek más dimenziójú mátrixokkal is dolgoznak). A mátrixok elemei különböző ábrák, melyek között valamilyen logikai kapcsolat áll. Az utolsó elem hiányzik, a tesztalany feladata ezen hiányzó elem kiválasztása nyolc lehetséges válasz közül. A feladatok mögött álló logikákat szabályoknak nevezzük. A feladat első körben egy olyan módszer megalkotása, mellyel nagyszámú szabályt tudunk generálni. Ehhez először definiálunk egy leíró formai nyelvet.



3. ábra Példa egy nagyon egyszerű figurális absztrakció teszt feladatra

2.1.2 Leíró nyelv

A szabályok leírására egy olyan formai nyelvet definiálunk, mely, a teszt típusából adódóan, háromszor hármas mátrixokat tartalmaz. Egy mátrix celláiban ún. entitásokat helyezünk el. Ezek az entitások absztrakt dolgok, melyek nem hordoznak információt a későbbi vizualizációjukkal kapcsolatban. Lehetséges természetesen egy cellát üresen hagyni, ekkor nem tartalmaz entitást. Minden entitásnak van egy alap típusa, melyeket az A-H betűkkel jelölünk. Ez a típus csupán annyit hordoz magában, hogy két azonos típusú entitás azonosan (vagy legalábbis hasonlóan), két eltérő pedig eltérően lesz a későbbiekben vizualizálva. A 3. ábrán látható egy példa, mely egy egyszerű szabályt ír le, felhasználva különböző típusú entitásokat.

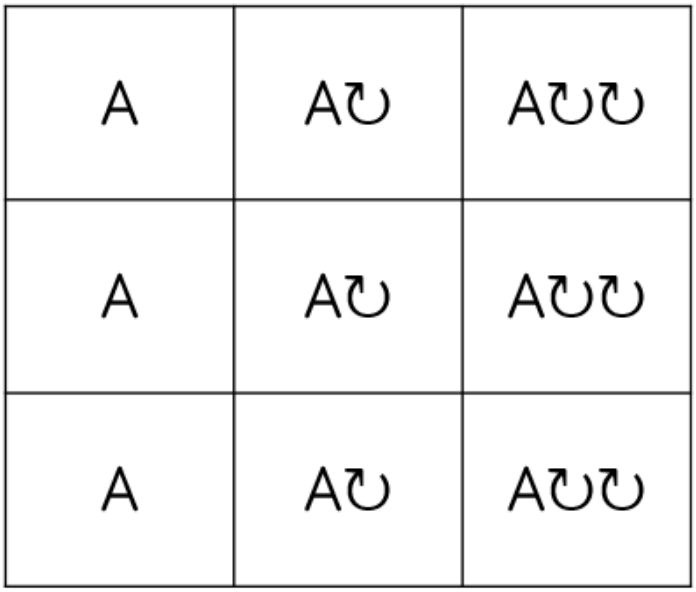


4. ábra Példa egy egyszerű szabály leírására

Az entitásokra különböző transzformációk is alkalmazhatók, melyek a következők: elforgatás, eltolás, tükrözés. A 1. táblazat tartalmazza a különböző transzformációk altípusait, hogy melyikből maximum hány alkalmazható egy entitáson, illetve hogy milyen szimbólummal jelöljük. A 4. ábra egy példát tartalmaz, melyben az entitásokra elforgatásokat is alkalmaztunk.

1. táblázat A transzformációk altípusai, maximum száma és szimbóluma

	Type	Max number	Symbol
Rotation	Clockwise	3	↻
	Counter clockwise	3	↺
Mirroring	Horizontal	1	-
	Vertical	1	|
	Left diagonal	1	\
	Right diagonal	1	/
Offsetting	Up	2	↑
	Down	2	↓
	Left	2	←
	Right	2	→



5. ábra Példa egy transzformációkat tartalmazó szabály leírására

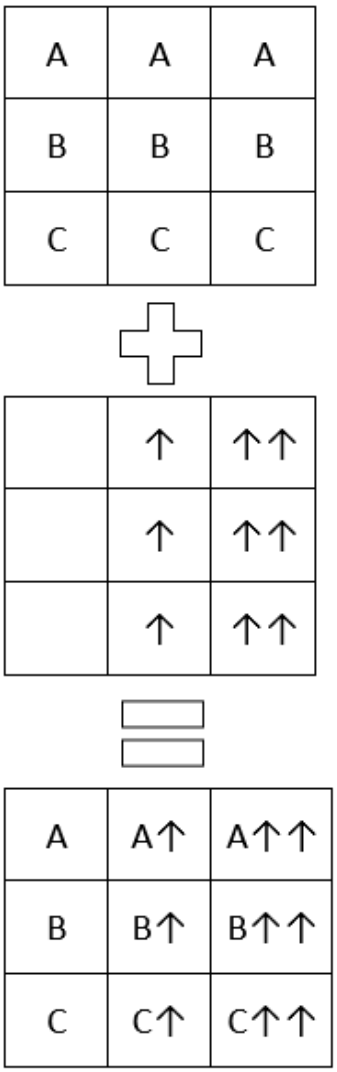
A transzformációk mellett különböző logikai függvényeket is használhatunk, melyek paraméterei entitások. Ezek a logikai függvények a következők: AND (logikai ÉS), OR (logikai VAGY), XOR (kizáró VAGY), NEG (negálás). Természetesen ezen függvények konkrét implementációja nincs megszabva, ez a későbbi vizualizáción múlik.

2.2 Szabály generálás

Könnyen belátható, hogy az előző pontban definiált formai nyelvvel leírható szabályok száma végtelen. Ahhoz, hogy ezt végesre redukáljuk, néhány megkötést fogalmazunk meg a szabályokat illetően. Ezek a megkötések a következők

- Az entitások csak alfabetikus sorrendben jelenhetnek meg. Ez azt jelenti, hogy a szabály első (balról jobbra olvasást feltételezve) entitásának típusa csak A lehet, a másodiknak A vagy B és így tovább.
- Egy entitásra csak egyféle transzformációt lehet egyszerre alkalmazni.
- Logikai függvények paraméterei csak olyan entitások lehetnek, melyekre nincs semmilyen más transzformáció vagy logikai függvény alkalmazva.
- Logikai függvények paraméterszáma pontosan kettő (negálás esetén egy).
- Csak azokra az entitásokra lehet bármilyen transzformációt, vagy logikai függvény alkalmazni, mely legalább egyszer mindenféle módosító nélkül önmagában is szerepel a szabályban.
- Egy szabályon belül csak egyféle transzformáció vagy logikai függvény használható, viszont transzformáció esetén azok altípusai közül akár több is.

A fenti megkötésekkel a formai nyelv által leírható szabályok száma véges, így képesek vagyunk minden létezőt legenerálni. A folyamatot optimalizálva, két részre bontjuk a lehetséges szabályokat. Először legeneráljuk azokat a szabályokat, melyek nem tartalmaznak semmiféle transzformációt vagy logikai függvényt. Ezután (vagy ezzel egyidőben) olyan mintákat generálunk, melyek csak transzformációkat vagy logikai függvényeket tartalmaznak, vagyis konkrét entitásokat nem. Látható, hogy ezen két csoportból két elemet kombinálva kaphatunk egy érvényes szabályt. Természetesen az első csoport elemei önmagában is szabályokat alkotnak. Amire a kombináció során figyelni kell, hogy ne sértsük meg egyik fent megfogalmazott megkötést sem, illetve hogy üres cellára ne alkalmazzunk transzformációt vagy logikai függvényt. Az 5. ábrán egy példa látható, melyben egy csak egyszerű entitásokat tartalmazó szabályt kombinálunk egy ún. transzformációs mintával, végeredményként kapva egy érvényes szabályt.



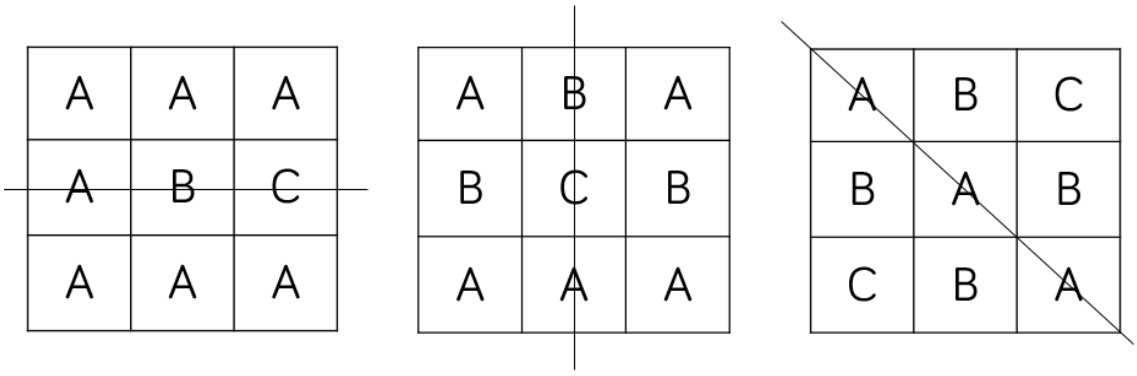
6. ábra Két minta kombinálása a generálás során

2.3 Megoldhatóság vizsgálata

Az előző pontban ismertetett leíró nyelvet használva képesek vagyunk nagyszámú szabályt generálni. A következő lépés ezen szabályok megszűrése az alapján, hogy melyik szabályt tartjuk emberek által megoldhatónak, tehát valójában összeségében értelmesnek. A generálás folyamatának ez az egyik leglényegesebb pontja, hiszen itt választjuk ki azokat a szabályokat, melyekkel a későbbiekben dolgozni fogunk és végeredményben feladatokat készítünk.

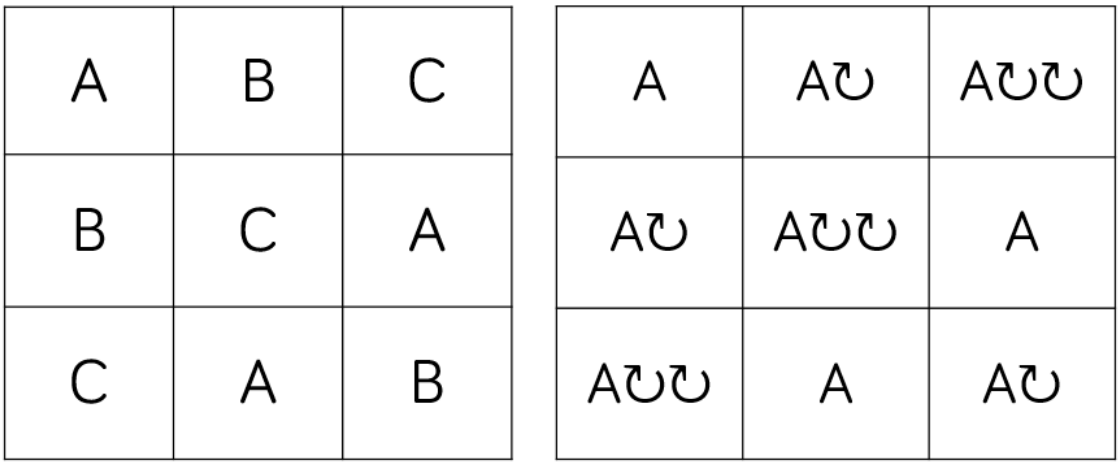
A megoldhatóság vizsgálatát feltételek megfogalmazásával végezzük, melyek közül ha valamelyik igaz egy adott szabályra, akkor azt megoldhatónak mondjuk és megtartjuk későbbi használatra. A feltételek a következők:

- Szimmetria: A szabályban felismerhető valamilyen szimmetria, vertikális, horizontális vagy diagonális. A 6. ábrán látható példa mindhárom szimmetriára. Ezen kívül két különböző típusú szimmetriát különböztetünk meg. Az első esetben maga a szimmetria tengely is szimmetrikus (ilyen a példán látható harmadik szabály), a másikban pedig nem (ilyen a példán látható másik kettő szabály). Az utóbbi gyenge szimmetriának nevezzük. Gyenge szimmetria esetén a balról jobbra diagonális szimmetriát nem tartjuk megoldhatónak, mivel ez nagyon sok esetben nem egyértelmű szabályokhoz vezet.



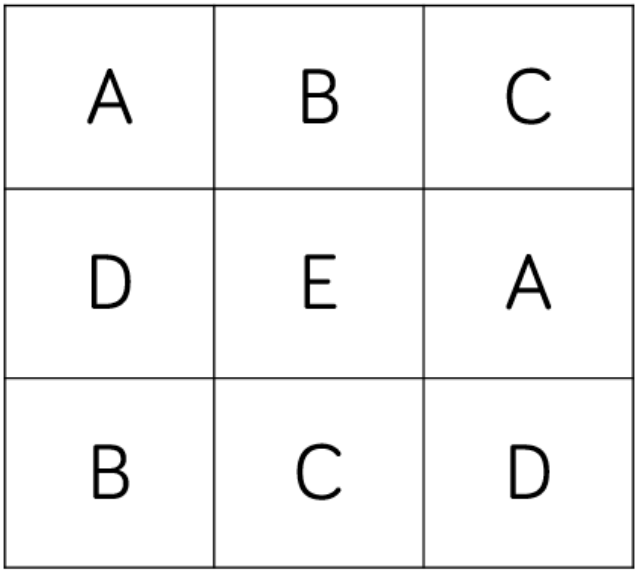
7. ábra Példa horizontális, vertikális és diagonális szimmetriára

- Három különbözőség: A szabályban három különböző entitás található (beleértve a típusukat és a rajtuk alkalmazott transzformációt vagy logikai függvényt). Minden sorban és oszlopban megtalálható mindhárom entitás. A 7. ábrán látható két példa közül az elsőben az entitások típusaikban térnek el, míg a másodikban a rajtuk alkalmazott transzformációkban.



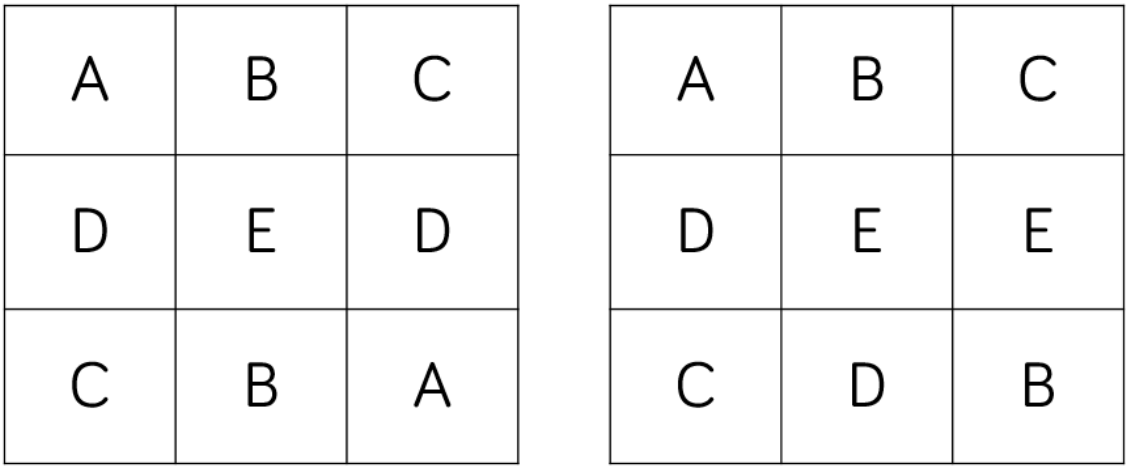
8. ábra Példák három különbözőségre

- Szekvencia: A szabályban valahány számú entitás szekvenciálisan ismétlődik (balról jobbra való olvasást feltételezve). Alacsony számú entitás esetén a szekvencia lényegében megegyezik a szimmetriával, túl magas elemszám esetén pedig aluldefiniált lesz a szabály. A 8. ábrán egy példa látható öt elemű szekvenciára.



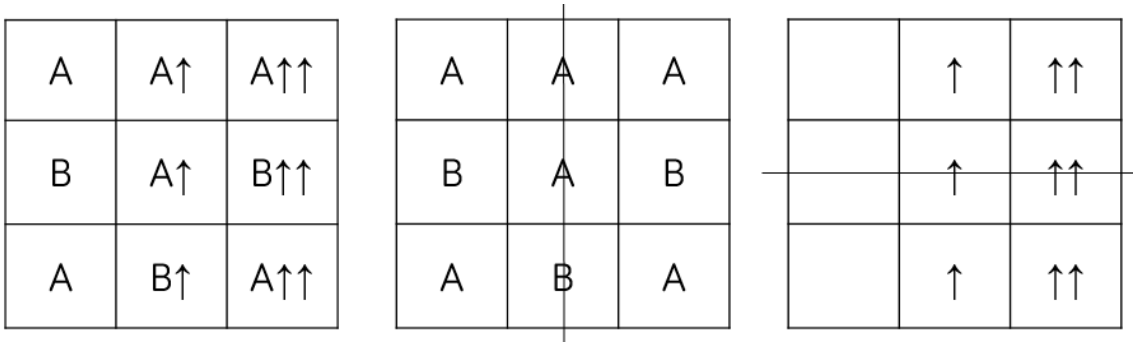
9. ábra Példa szekvenciára

- Fordított szekvencia: A szabályban valamely számú entitás szekvenciálisan ismétlődik megfordítva a sorrendet minden iterációban. A 9. ábrán látható két különböző típusú példa fordított szekvenciára. Az első esetben az iteráció utolsó eleme megegyezik a következő iteráció első elemével.



10. ábra Példa fordított szekvenciára

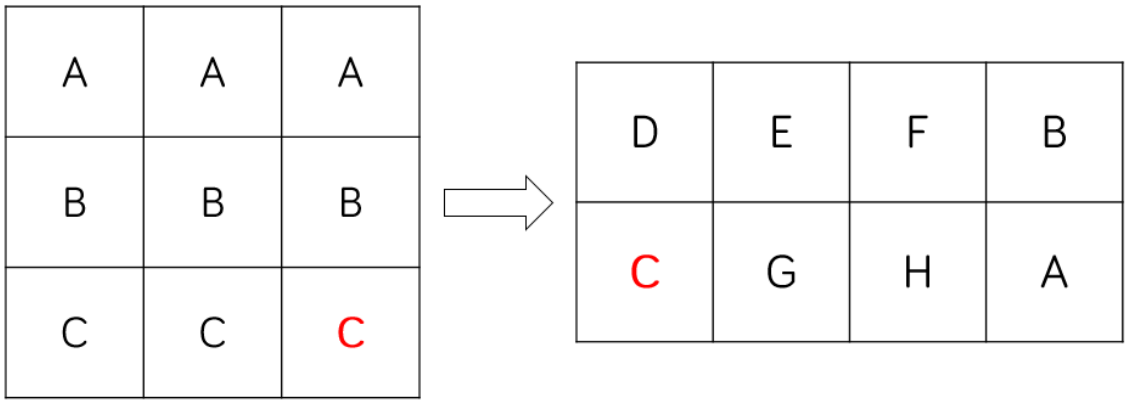
Ahelyett, hogy a megoldhatóság vizsgálata során az imént felsorolt feltételeket a szabályokra, mint egészekre vizsgálnánk, a generálás során felhasznált mintákat, melyek kombinálása eredményezte a kész szabályokat, külön-külön tanulmányozzuk. Ez a gyakorlatban azt jelenti, hogy például egy szabályra, melyben transzformációk is találhatók, egyik feltétel sem igaz, de az előállítása során használt két minta mindegyikéhez találunk olyan feltételt, amely igaz, akkor a szabályt megoldhatónak jelöljük. A 10. ábrán látható egy példa a megoldhatóságnak ilyen módú szeparált vizsgálatára. Amint látható, a szabály összességében nem szimmetrikus (illetve bármely egyéb feltétel sem érvényesül), viszont a két minta, mely a felépíti a szabályt, külön-külön szimmetrikus (vertikálisan és horizontálisan).



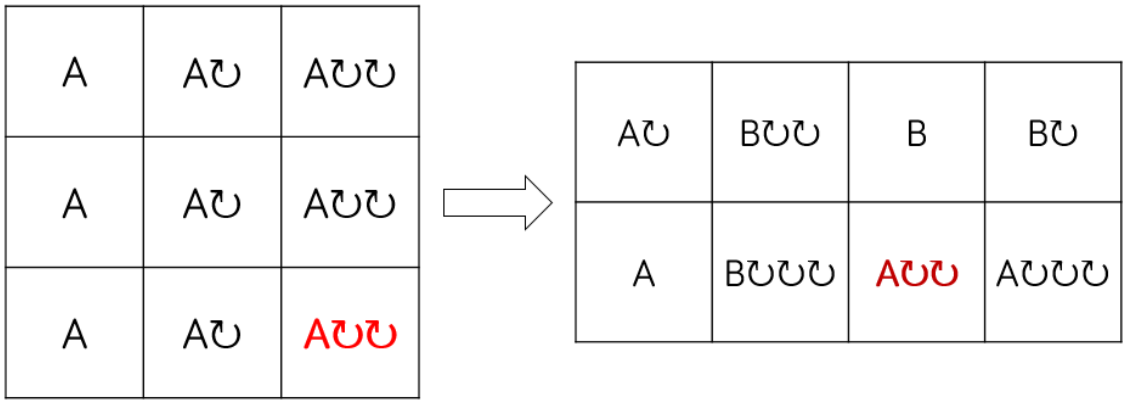
11. ábra Példa szeparált megoldhatóság vizsgálatra

2.4 Válaszjelöltek generálása

A tesztünk természetéből adódóan a feladatok előállításához a szabályokhoz lehetséges válaszokat kell generálnunk. Minden szabályhoz hét lehetséges megoldást kell előállítanunk, melyeknek egymástól különbözőknek kell lennie. Ehhez egy egyszerű algoritmust használunk: amennyiben a szabály nem tartalmaz semmilyen transzformációt vagy logikai függvényt, egyszerűen adunk hét különböző típusú entitást (lásd 11. ábra), ellenkező esetben pedig a szabályban található transzformációt vagy logikai függvényt, entitás típusokat, illetve egy extra új típust felhasználva generálunk véletlenszerűen entitásokat (lásd 12. ábra).



12. ábra Példa módosító mentes szabályhoz való válaszgenerálásról



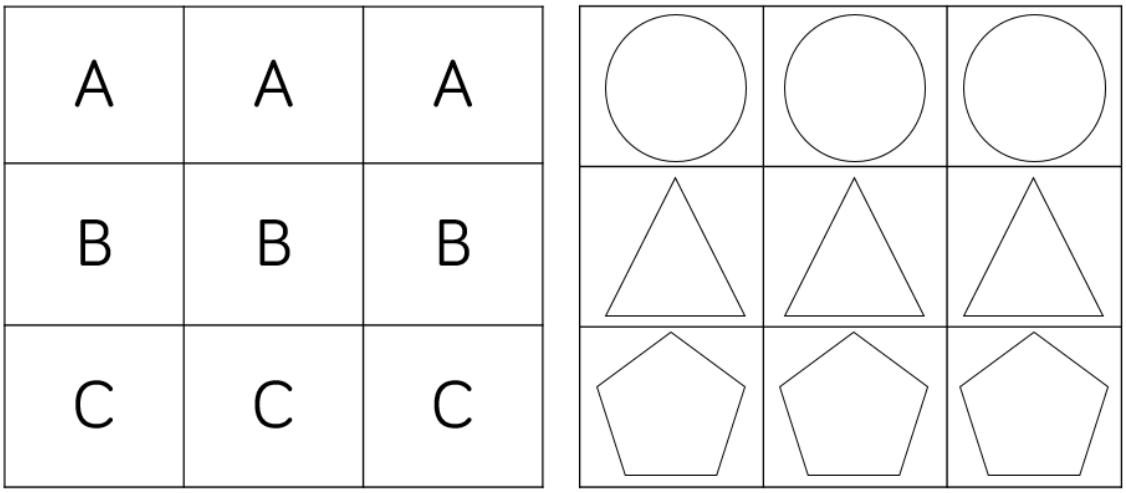
13. ábra Példa transzformációt tartalmazó szabályhoz való válaszgenerálásról

Előfordulhat, hogy két szabály csak az utolsó elemében, vagyis a megoldásban különbözik. Emiatt a válaszgenerálás folyamat közben figyelnünk kell, hogy ne generáljunk olyan válaszlehetőséget, mely megoldása lehet egy másik, nagyon hasonló szabálynak.

3 Vizualizáció

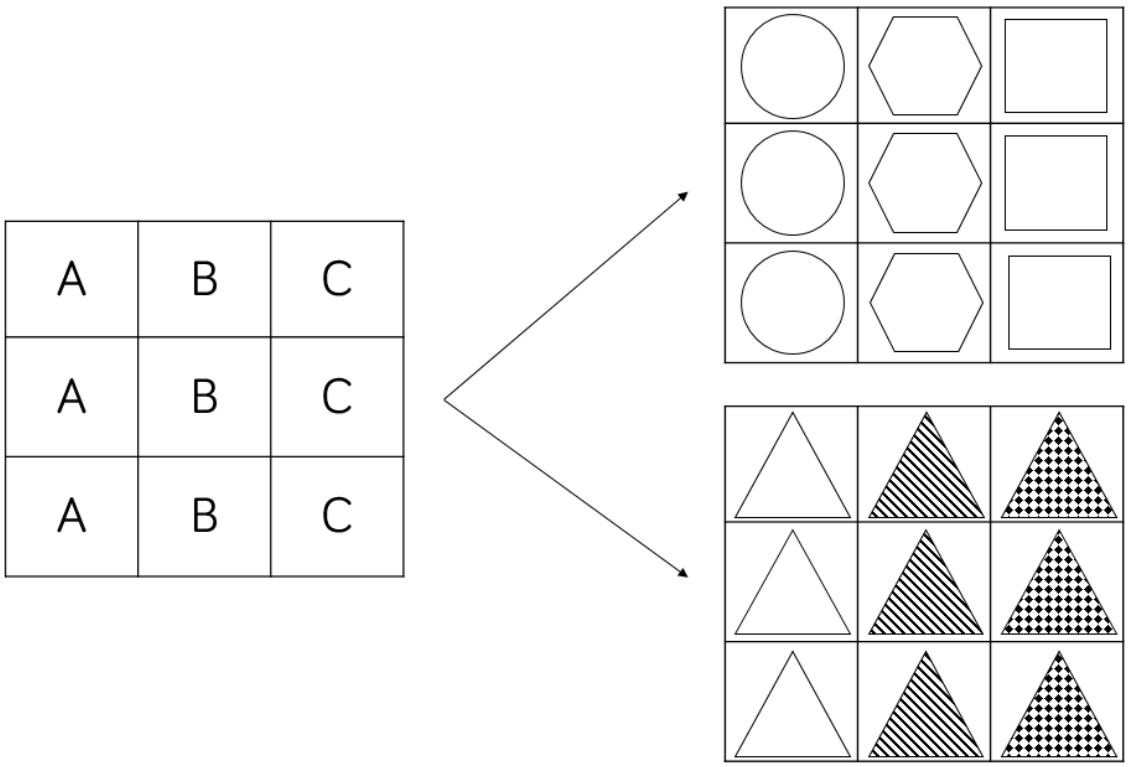
A feladatok előállításának végső lépése az előző pontban leírt módszerrel generált szabályok vizualizációja. A szabály, illetve entitás fogalmának megalkotása során arra törekedtünk, hogy minél általánosabbak, absztraktabbak legyenek. Ez a gyakorlatban azt jelentette, hogy minél kevesebb információt hordozzanak a tényleges vizualizálásukról. Ez azért fontos, mert így a szabályok sokkal szabadabban vizualizálhatóak lettek. Egy entitást megjeleníthetünk egy alakzatként, képként, animációként vagy bármi egyébként.

A 13. ábrán egy példa található, mely a már korábban is látott szabály egy egyszerű vizualizálását mutatja be. Ebben az esetben az entitások különböző alakzatoknak felelnek meg, így az „A” entitás egy körnek, a „B” egy háromszögnek, a „C” pedig egy ötszögnek.



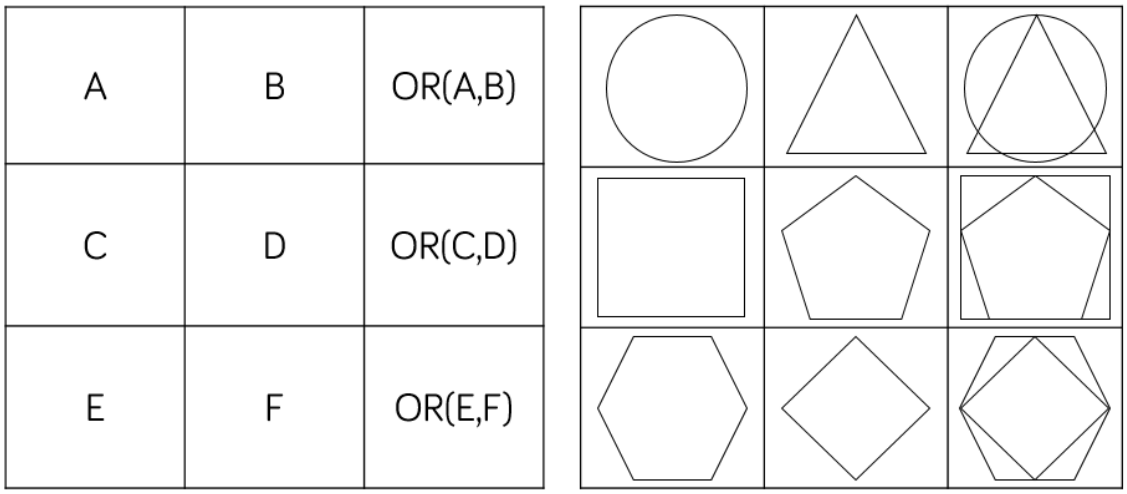
14. ábra Példa egy szabály egyszerű vizualizálására

A 14. ábrán egy hasonló szabály két különböző vizualizálását láthatjuk. A felső a már az előző példában is használt egyszerű alakzatokkal dolgozik, míg a második az entitásokat különböző textúrákkal különbözteti meg. Ebből a példából jól látszik, hogy az entitások típusa a konkrét vizualizáció során utalhat alakzatukra, méretükre, textúrájukra stb..



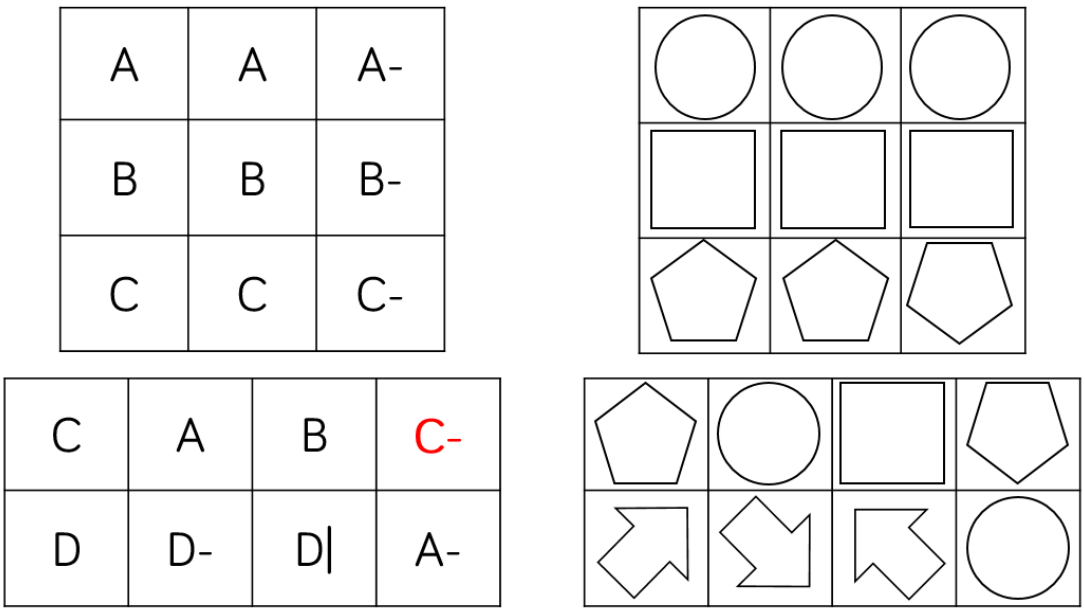
15. ábra Példa egy szabály többféle vizualizálására

Bár a szabályok és entitások elég általánosak, mégis hordoznak olyan információt, amely megszabhatja, hogy mely vizualitás képes mely szabályt megjeleníteni. Ilyen például, ha a szabály tartalmaz logikai függvényeket. Az előbbi példákban használt vizualizáció, mely egyszerű alakzatokat használ, képes például olyan szabályok megjelenítésére, melyek logikai VAGY függvényt tartalmaznak, viszont semelyik másik logikai függvény vizualizálását nem teszi lehetővé. Erre láthatunk egy példát a 15. ábrán.



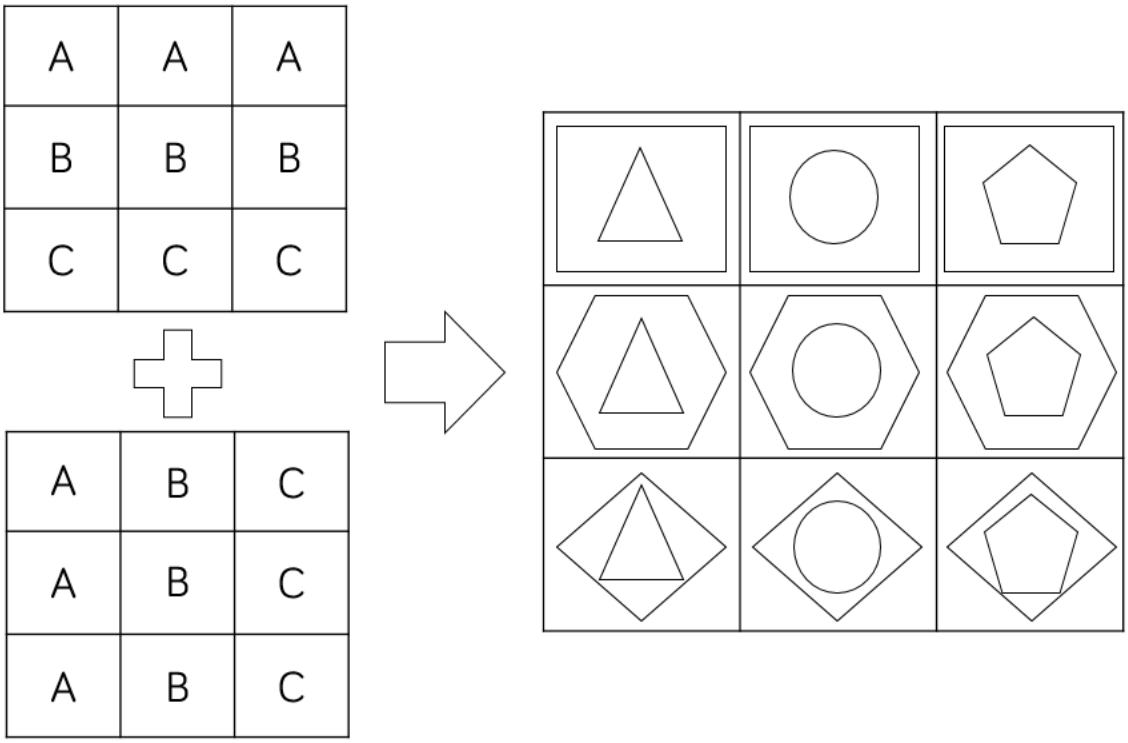
16. ábra Példa logikai függvényt tartalmazó szabály vizualizációjára

A logikai függvények mellett a különböző transzformációkra is oda kell figyelni a vizualizáció során. Ez a gyakorlatban annyit jelent, hogy például ha egy szabályban egy entitásra horizontális tükrözés van alkalmazva, akkor annak a bizonyos entitásnak a vizualitása, legyen az egy ábra vagy akár animáció, nem lehet horizontálisan szimmetrikus, különben a tükrözés művelete nem változtatna a konkrét vizualitáson. Erre egy helytelen példát mutat a 16. ábra, mely jelen esetben egy többértelmű és hibás feladathoz vezet. Látható, hogy mindamelett, hogy a válaszlehetőségek sem különböznek a megjelenítésben, de a feladatnak is két lehetséges megoldása van (amennyiben természetesen csak a vizualizációt nézzük). Meg kell említeni, hogy az elforgatás műveletét alapértelmezetten 90°-os elforgatásként definiáltuk, emiatt alkalmazható maximum három egy entitásra. Ez természetesen nem kötött, a konkrét vizualizációnál megváltoztatható bármilyen más fokban való elforgatásra.



17. ábra Példa egy hibás vizualizációra

Amellett, hogy a szabályokat külön-külön egyesével vizualizáljuk, lehetőség van szabályok összevonására. Ez a gyakorlatban azt jelenti, hogy egy feladat több szabályból áll össze. Ezt úgy lehet elérni, hogy például a vizualizáció során a különböző szabályok különböző dimenziókra érvényesek. A 17. ábra erre mutat egy példát. Ebben az esetben a felső szabály a külső, nagyobb alakzatokra érvényes, míg az alsó a belső, kisebbekre.



18. ábra Példa többdimenziós vizualizációra

4 Nehézségek megállapítása

Az előző fejezetben egy olyan módszert mutattunk be, melynek segítségével szabályok, majd belőlük különféle feladatok generálhatóak. Ahhoz, hogy a végeredményként megkapott feladatokat egy éles tesztben használni lehessen, szükség van a nehézségi szintek megállapítására. A nehézségeket külön érdemes meghatározni a szabályokra, illetve a kész feladatokra, hiszen egy konkrét feladat nehézségét befolyásolja egyrészt értelemszerűen a mögötte álló logika (tehát maga a szabály), illetve a megjelenítés is, hiszen lehetnek egyszerűbb és bonyolultabb vizualizációk is egyaránt. Első körben érdemes a szabályok nehézségének megállapításával foglalkozni, hiszen ezeket egy fokkal objektívabban és egyszerűbben lehet meghatározni.

4.1 Becsült nehézségi szintek

A generálás folyamata közben lehetőség van egy számított nehézséget meghatározni minden szabályhoz. Ez a nehézségi szint a szabály különböző tulajdonságainak súlyozásával és összegzésével áll elő. Ilyen tulajdonságok lehetnek például a következők:

- Különböző entitás típusok száma
- Üres mezők száma
- Különböző transzformáció típusok száma
- Transzformációk/logikai függvények száma
- Megoldhatóság indoka, vagyis hogy melyik feltétel miatt gondoljuk a szabályt megoldhatónak

Ezen nehézségi szint meghatározásának folyamatát egy példán keresztül mutatjuk be. Ebben a példában az imént felsorolt tulajdonságokat és az általunk fejlesztett tesztalkalmazásban is használt súlyokat fogjuk felhasználni egy konkrét, bonyolultabb szabály nehezségi szintjének megbecsülésére. A felhasznált súlyok az egyes paraméterekhez a következők:

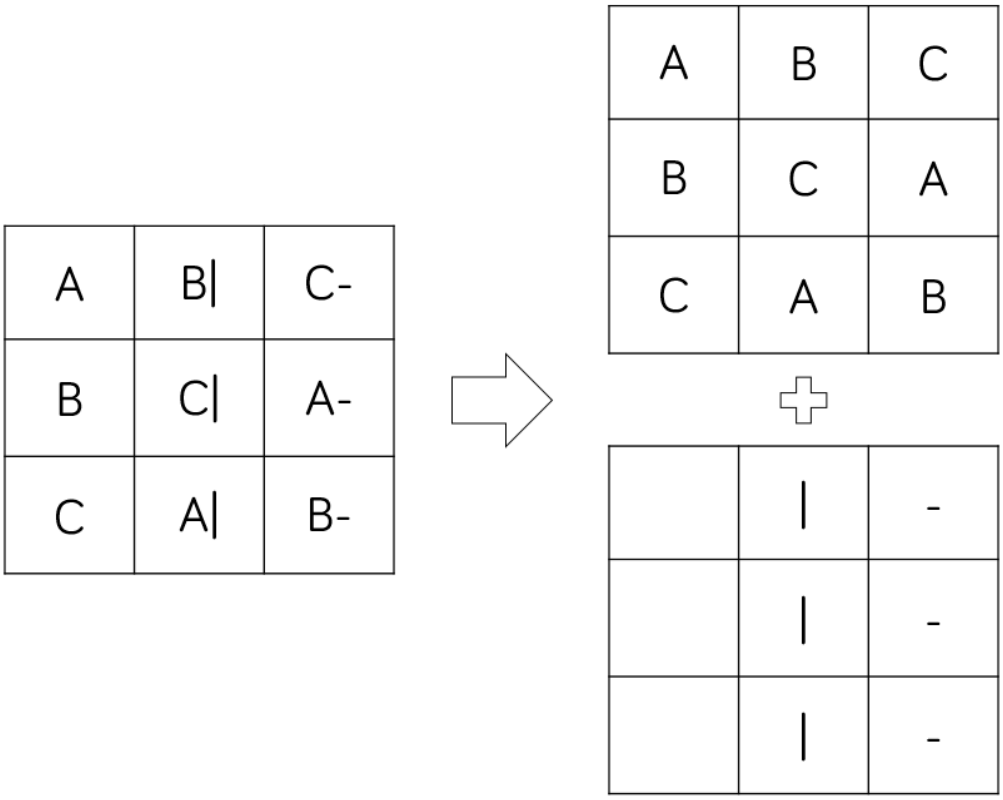
- Entitás típusok száma: n * 10
- Üres mezők száma: n * (-2)

- Transzformáció típusok száma: n * 3
- Transzformációk száma: n * 2
- Logikai függvények száma: n * 4
- Megoldhatóság indoka (két réteg esetében összeadódik):
    - Szimmetria: +0 (+2 ha diagonális)
    - Gyenge szimmetria: +6 (+8, ha diagonális)
    - 3-különbözőség: +4
    - Szekvencia: +15
    - Fordított szekvencia: +20

A konkrét szabály a 18. ábrán látható, felbontva a két mintára, melyeket a generálás során kombináltunk. A felbontás azért fontos, mert a „megoldhatóság indoka” faktor esetében a két réteget külön vizsgáljuk meg. Az egyes tulajdonságok értéke, illetve a végeredmény a következőképpen alakul:

- Entitás típusok száma: 3 → 3 * 10 = 30
- Üres mezők száma: 0
- Transzormáció típusok száma: 2 → 2 * 2 = 4
- Logikai függvények száma: 0
- Megoldhatóság indoka:
    - 3-különbözőség: +4
    - Gyenge szimmetria: +6
- Végeredmény: 30 + 4 + 4 + 6 = **44**

Természetesen az így megkapott eredmény csupán egy durva becslés, mely további finomhangolást igényel a következő alfejezetben bemutatott módszerek egyikével.



19. ábra A szabály és felbontása, melynek a nehézségi szintjét becsüljük

4.2 Nehézségi szintek finomhangolása

Mint ahogy azt az előző alfejezetben láttuk, a szabályok nehézségét képesek vagyunk becsülni. Azonban ez a becsült érték nyilvánvalóan nem lesz pontos, további finomhangolásra szorul. Ezt a finomhangolást többféleképpen megtehetjük, ezek közül most két módszert fogunk felületesen ismertetni. Mindkét esetben szükség van nagyszámú tesztalanyra, illetve sok mérési adatra.

A becsült érték meghatározásánál felhasználtuk az adott szabály néhány tulajdonságát és az ezekhez rendelt súlyokat. Az első módszer lényege, hogy nem a konkrét nehézségi értékeket próbáljuk finomhangolni, hanem ezekhez a faktorokhoz tartozó súlyokat. Ezt a legegyszerűbben úgy tehetjük meg, hogy első körben adunk a súlyoknak valamilyen kezdőértéket (lásd például az előző alfejezetben lévő értékeket), majd méréseket végzünk olyan tesztalanyokon, akikről van valamiféle referencia adatunk a képességi (vagy intelligencia) szintjüket illetően. Mindeközben a beérkező mérési eredmények alapján módosítjuk a súlyokat. Ez a gyakorlatban azt jelentheti, hogy például ha egy magasabb (átlag feletti) képességű tesztalany egy adott szabály szerint előállított feladatot nem oldott meg, akkor a szabályban szereplő faktorok súlyait megemeljük. Ugyanez érvényes természetesen fordított irányban is. A súlyok

módosításának mértéke lehet exponenciálisan csökkenő (tehát kezdetben nagymértékben, majd egyre finomabban módosítjuk az értékeket), illetve szerepet játszhat még az, hogy a referencia képességi szint milyen skálán értelmezett (pl. IQ skála).

Egy másik megközelítés esetén a konkrét nehézségi szinteket módosítjuk a mérési eredmények elemzése alapján. Erre létezik renteget különféle módszer, többek között olyanok is, melyek semmilyen képességi szint referencia értéket nem igényelnek az egyes tesztalanyokról. Egy ilyen módszer megalkotása jelenleg is folyamatban van a kutatócsoportunkban. [6] Ezen módszerek a mi esetünkben nem feltétlenül alkalmazhatóak első körben, hiszen megkövetelik, hogy minden feladatról többszöri mérési eredmény álljon rendelkezésre. Ez a gyakorlatban azt jelentené, hogy a rengeteg legenerált feladatot többször is kiadjuk tesztalanyoknak, ami rendkívül nagy emberi erőforrást venne igénybe.

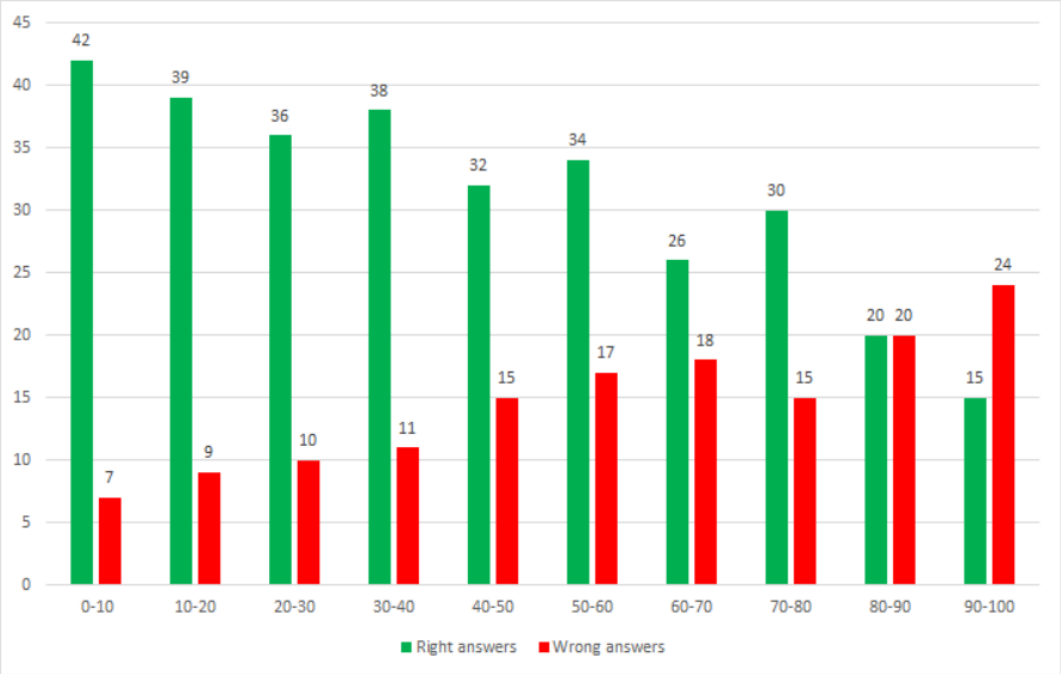
Mint ahogy az korábban említve lett, a feladatok nehézségét nem csak a mögöttük álló szabályok, hanem a megjelenítésük is befolyásolja. Mivel ezt nehéz becsülni, ezért csakis mérések alapján van lehetőség a vizualitás szerepének megállapításában. A mi elképzelésünk, hogy elsőkörben a szabályok becsült nehézségi értékét finomhangoljuk a bemutatott módszerek közül az elsővel, majd ezeket tovább módosítjuk a második típusú módszerek valamelyikével. A kezdeti finomhangolást egy kisebb tesztcsoporton végzett méréssel végeznénk, míg a második szakasz a teszt alkalmazás életciklusa során folyamatosan, egy sztenderdizálás folyamat keretében történne meg, mely már a különböző vizualizációk szerepét is beleszámítaná a nehézségi szintek módosításába. Felmerül a kérdés, hogy a kezdeti finomhangolás esetén is szükség van valamilyen vizualizációra, amely esetleg befolyásolhatja az eredményeket. Mivel elsőkörben a szabályok nehézségi szintjét próbáljuk meg meghatározni, ezért ezt a legegyszerűbb vizualizációkkal tesszük meg, minimalizálva azok hatását a nehézségekre.

5 Eredmények

5.1 Áttekintés

Az előző fejezetekben leírt módszerekkel képesek vagyunk nagy számú szabályt, illetve még nagyobb számú feladatot generálni. A végeredményben legenerált szabályok száma meghaladja az egymilliót. Ez már egy olyan magas szám, amely esetén nyugodtan mondhatjuk, hogy annak az esélye, hogy egy tesztalany kétszer kapja meg ugyanazon feladatot, elég alacsony. Emiatt természetesen az is igaz, hogy szinte lehetetlen „betanulni" a feladatokat és ezáltal lényegében csalni a teszten. A 2. táblázat egy statisztikát mutat, mely a következő alfejezetben bemutatott alkalmazás tesztelése során az egyes nehézségi tartományokhoz tartozó beérkezett jó és rossz válaszok számát mutatja. Jól látható, hogy habár a feladatok nehézségi szintje a statisztika készültekor még csupán egy durva becslés volt, a módszerrel elég jól becsültük a tényleges nehézségi szinteket.

2. táblázat Teszteredmények



5.2 Tesztalkalmazás

Ahhoz, hogy első körben a generált példák nehézségének felmérését, második körben a teszt használatát lehetővé tegyük, készítettünk egy webalkalmazást és egy

androidos alkalmazást. Ahhoz hogy a felhasználók az alkalmazást használni tudják, regisztrálniuk kell a rendszerben. Ugyan a regisztráció és a bejelentkezés kényelmetlen lehet, mégis szükséges, mert így felhasználóhoz tudjuk kötni a teszteket eredményeit. A felhasználóknak is lehetőségük van a teszt felfüggesztésére, és később (akár más eszközön) történő folytatására, továbbá hasznos statisztikákat olvashatnak a korábbi tesztek és gyakorló tesztek eredményeiről. A regisztráció során az eredmények kiértékelése szempontjából is hasznos adatokat tudunk begyűjteni a felhasználótól. Ilyen adat a felhasználó neme, életkora, irányítószáma, és esetleges fogyatékossága. Ahhoz, hogy kommunikálni tudjunk, regisztrációkor elkérjük az email címet is. Így különböző értesítéseket küldhetünk, amiben tájékoztatjuk a felhasználókat a rendszerben történt változásokról, esetleg a kutatás eredményeiről.

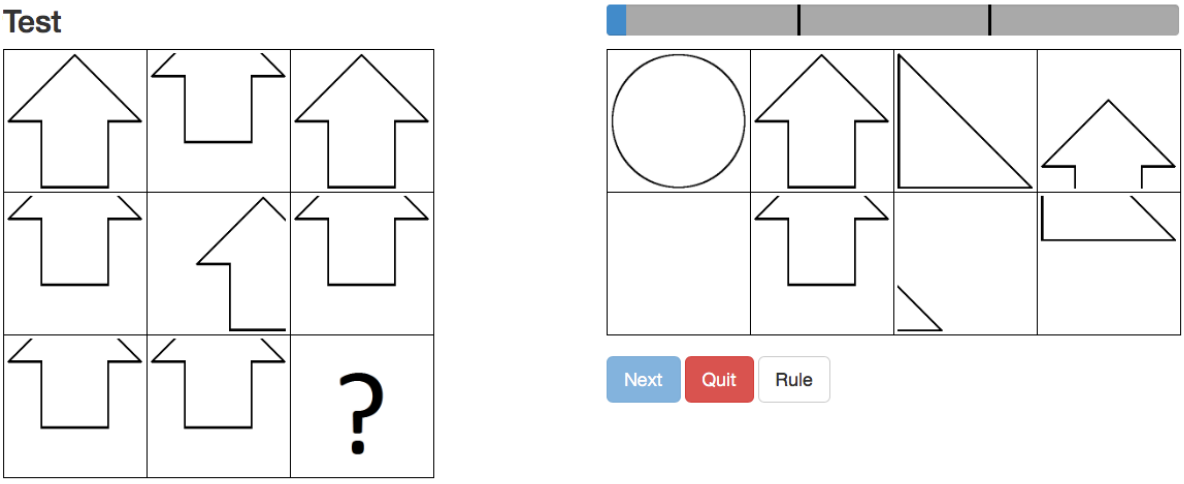
Az alkalmazás létrehozásakor törekedtünk az egyszerűségre. A bejelentkezett felhasználó először a főoldalra érkezik, ahol hasznos információkat találhat a kutatásról és a teszt kitöltésének szabályairól. Létrehoztunk egy teszt oldalt, ahol lehetőség van teszteket kitölteni, egy gyakorló oldalt, ahol a teszteket lehet gyakorolni, és egy statisztika oldalt, ahol a korábbi teszt és gyakorlás eredményei között lehet böngészni.

A teszt és gyakorló oldalra lépéskor új teszt indítható vagy ha van megkezdett tesztmenet, akkor azt is lehet folytatni. Amennyiben van egy megkezdett tesztmenet, de a felhasználó mégis egy újat indít, a megkezdettet lezárjuk és indítunk egy újat. A teszt és gyakorló oldal felépítésében megegyezik. Mind a kettőn található egy 3x3-as tábla jobb alsó sarokban egy kérdőjellel, és egy 2x4-es tábla a kérdőjel helyére választható ábrákkal. A felhasználói interfész megtervezésekor ügyeltünk arra, hogy a teszt kitöltse a rendelkezésre álló helyet minden kijelzőméreten. Figyeltünk továbbá arra is, hogy érintőképernyőn is kényelmesen használható legyen a teszt. A webes és androidos alkalmazásban a megjelenítés teljesen megegyezik, így csökkenteni tudjuk a megjelenítés különbözőségéből származó eredménytorzulást.

A hagyományos figurális absztrakciós tesztekben nincs arra lehetőség, hogy a válasz ábrákat „bepróbáljuk” a kérdőjel helyére, ezért itt is csak kijelölhető a válasz, de a kérdőjel helyén nem jelenik meg. A „Next” gomb akkor válik kattinthatóvá, ha a felhasználó megjelölt egy választ, így addig nem tud tovább lépni, amíg nem válaszolt. A válasz módosítására a „Next” gomb megnyomása előtt van lehetőség. Ha a felhasználó tovább lépett, az alkalmazás elküldi a választ a szervernek, majd megjeleníti a következő tesztfeladatot. A „Quit” gomb megnyomásával a felhasználó megszakítja a

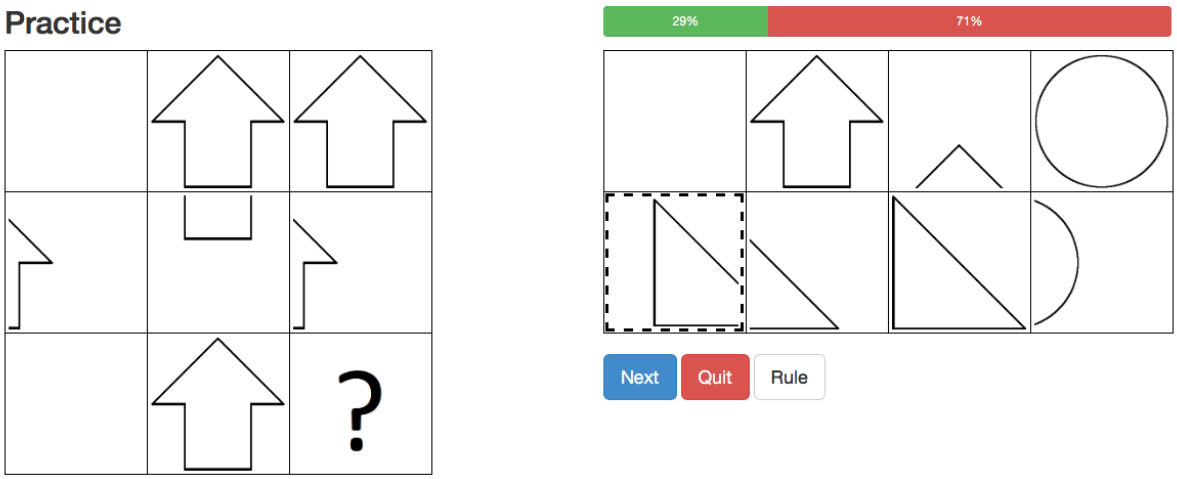
tesztmenetet. Az alkalmazás jelenlegi verziójában a „Role” gombbal lehetőség van a szabály formális leírását megjeleníteni. Ez csak az alkalmazás tesztelését szolgálja, a végleges változatban nem fog szereplni. Ugyan a figurális absztrakciós tesztek kitöltésekor nem kell mérni az időt, mi mégis naplózzuk, hogy mennyi ideig tartott amíg a felhasználó választ adott egy feladatra. Ezzel a feladatok nehézségének finomhangolását szeretnénk megkönnyíteni.

A jelenlegi beállítok szerint 3x10 kérdésre kell válaszolni a tesztet kitöltőnek, minden 10-es egység között van egy-egy szünet. Ezek a beállítások konfigurálhatóak. A szünetben kiírjuk a felhasználónak, hogy még hány feladat van hátra. A teszt felett található progressbar ad visszajelzést a felhasználónak arról, hogy hol tart. Ezen a progressbaron nincs visszajelzés arról, hogy a felhasználónak hány válasza helyes vagy helytelen, ez ugyanis segíthetne neki a válaszadásban ami torzíthatja az eredményt. Fontos, hogy a tesztmenet eredményei addig nem jelennek meg a statisztika oldalon, amíg a felhasználó le nem zárta azt. Így nincs lehetőség a statisztikákból kiolvasni az aktuális tesztmenet állását. A 13. ábrán egy pillanatkép látható a teszt felületről.



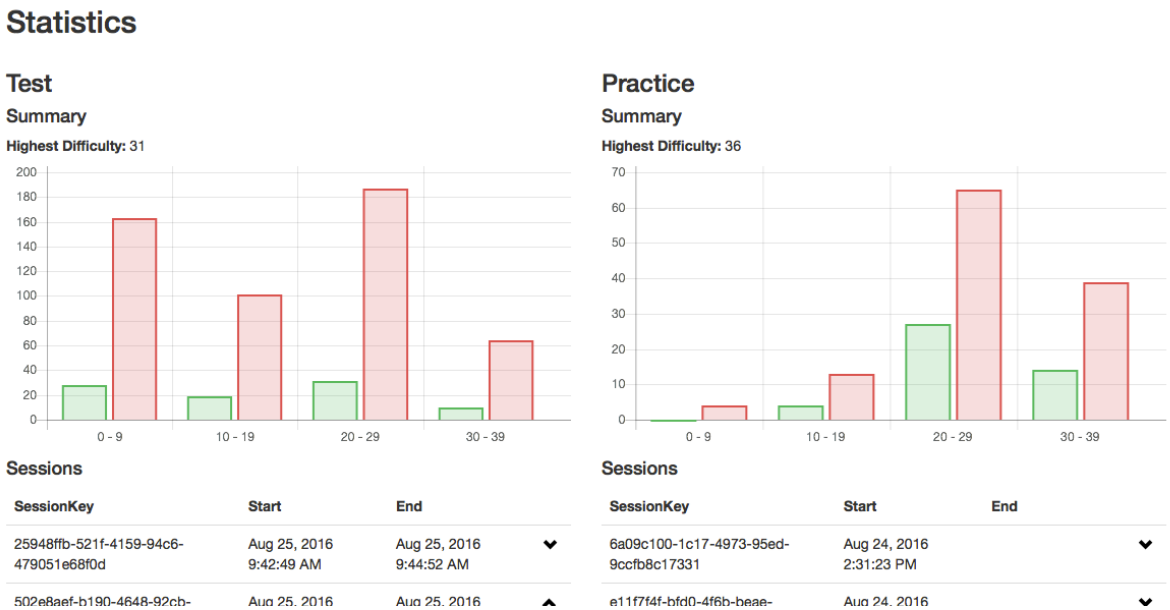
20. ábra Képernyőkép a teszt felületről

A tesztoldaltól a gyakorlóoldal abban különbözik, hogy itt nincs megszabva, hogy hány tesztpéldát kell kitöltenie a felhasználónak. A progressbar azonban itt visszajelzést ad a megoldott feladatok helyességéről. A függőben lévő gyakorlótesztmenetek megjelennek a statisztikák oldolon is. A 14. ábrán egy képernyőkép látható a gyakorló felületről.



21. ábra Képernyőkép a gyakorló felületről

A statisztika oldalon külön-külön oszlopban megjelennek a gyakorló- és tesztfeladatok eredményei, a korábban említett megkötésekkel. Rendelkezésre áll egy összesítő ábra a mindenkori eredményekről. Megtudhatjuk, hogy mi volt a legnehezebb helyesen megoldott példa, továbbá tesztmenetekre lebontva is találhatunk oszlopdiagrammokat. Az aktuális rendszerben szereplő adatok között olyan is van, ami a végleges rendszerben nem fog szerepelni (SessionKey). A diagrammok megjelenítésénél ügyeltünk arra, hogy áttekinthető legyen az ábra, ha túl sok oszlop jelenne meg, akkor intervallumra összegezve jelenítjük meg az adatokat. A 15. ábrán látható egy pillanatkép az adott felhasználóhoz tartozó statisztikákról szóló felületről.



22. ábra Képernyőkép a statisztikákat tartalmazó felületről

Az androidos kliens létrehozásakor ügyeltünk arra, hogy a tesztfeladatok megjelenítése ne térjen el a webes alkalmazásban megvalósítottól. Végeredményben úgy döntöttünk, hogy egy WebView-ba rakjuk a webes alkalmazást és a natív androidos gombok ezen WebView-ban navigálnak.

6 Összefoglalás

A dolgozat elején ismertetett kezdeti célunk az volt, hogy kidolgozzunk egy olyan módszert, mely figurális absztrakciós tesztekhez nagyszámú feladatot képes automatizáltan legenerálni, lényegesen csökkentve az ehhez szükséges emberi erőforrásra való igényt. Ezt egy olyan algoritmus megalkotásával értük el, mely képes a feladatok mögötti logikai mintákat is legenerálni. Ehhez először végigmentünk az alapfogalmakon (szabály, entitás stb.), majd definiáltunk egy leíró formális nyelvet ezen logikai minták, szabályok leírására. A következő lépésben szükség volt különféle megkötések megfogalmazására, hogy képesek legyünk legenerálni az összes, az általunk definiált formai nyelvvel leírható szabályt legenerálni. A generálási folyamatot optimalizálva két csoportba tartozó mintákat generáltunk, majd ezek kombinációjával megkaptuk a tényleges szabályokat. A következő lényeges lépés az volt, hogy ezt a nagy szabályhalmazt megszűrjük, és csak azokat a szabályokat tartsuk meg, melyek értelmesnek, vagyis emberek által megoldhatónak bizonyultak. Ezt néhány feltétel megszabásával tettük meg, melyek közül ha bármelyik igaznak bizonyult egy szabályra, akkor azt megtartottuk későbbi használatra. A végső lépés a szabályok generálásában a hozzájuk tartozó válaszlehetőségek meghatározása volt. Ezt egy egyszerű algoritmussal tettük meg, mely képes hét véletlenszerű lehetséges megoldást generálni az egyes szabályokhoz.

A feladatok előállításának következő lépése ezen szabályok vizualizálása volt. Mivel a szabályokat és entitásokat úgy alkottuk meg, hogy minél kevesebb információt hordozzanak a tényleges megjelenítésükről, így láttuk, hogy az egyes szabályokat rengetegféleképpen lehet megjeleníteni. Miután elkészültek a feladatok, a végső lépés a nehézségi szintjeik meghatározása volt. Láttuk, hogy a szabályok generálása alatt mindegyikhez lehetőség van egy számított nehézségi szint meghatározására felhasználva a szabályok különféle tulajdonságát. Mivel ezek az értékek nem voltak pontosak, hanem csupán durva becslések, bemutattunk néhány módszert ezen nehézségi szintek finomhangolására.

A célunk a jövőben az, hogy elkészítsünk egy online és okostelefonokon elérhető publikus figurális absztrakciós tesztet, mely a dolgozatban leírt algoritmusokat használja feladatok előállításához. Ennek fejlesztése már elkezdődött, a következőkben

a prezentált algoritmusok finomhangolása, illetve a nehézségek meghatározása következik. Emellett egy önsztenderdizáló skála megalkotása is a jövőbeli terveink között van, melynek segítségével, akár más hasonló skálára való konvertálás után (pl. z-index, IQ skála), képesek vagyunk valamilyen visszajelzést adni a teszt kitöltőinek. A teszt kialakításával kapcsolatban pszichológusokkal is egyeztetünk, hogy minél pontosabb eredményeket tudjunk előállítani. A végső verzióban tervezzük, hogy egy más hasonló alkalmazásokat összefogó keretrendszerbe [7] integrálva különböző összefüggéseket tudjunk felismerni és ezáltal pontosabb visszajelzést tudjunk adni a teszt kitöltőinek.

Irodalomjegyzék

[1] J. Raven et al., *"Raven progressive matrices,"* in Handbook of nonverbal assessment. Springer, 2003, pp. 223–237

[2] L. Szegletes, B. Forstner, *„Towards biofeedback-controlled self-rewarding learning with mobile devices"*, Cognitive Infocommunications (CogInfoCom), 2012 IEEE 3rd International Conference on Infocommuncations

[3] R. J. Sternberg, Beyond IQ: *A triarchic theory of human intelligence*. CUP Archive, 1985.

[4] J. C. Raven, *"Standardization of progressive matrices, 1938,"* British Journal of Medical Psychology, vol. 19, no. 1, pp. 137–150, 1941.

[5] L. E. Matzen, Z. O. Benz, K. R. Dixon, J. Posey, J. K. Kroger, and A. E. Speed, *"Recreating raven's: Software for systematically generating large numbers of raven-like matrix problems with normed properties,"* Behavior research methods, vol. 42, no. 2, pp. 525–541, 2010.

[6] M. Szabó, K. D. Pomázi, B. Radostyan, L. Szegletes and B. Forstner, *„Estimating Task Difficulty in Educational Games"*, 7th IEEE International Conference on Cognitive InfoCommunications

[7] B. F. Luca Szegletes, Máté Köles, *"Socio-cognitive gamification: general framework for educational games,"* JOURNAL ON MULTIMODAL USER INTERFACES 9, pp. 395–401, 2015