EGER 1774
ESZTERHÁZY KÁROLY KATOLIKUS EGYETEM
Matematikai és Informatikai Intézet

A kommunikációs gráfok modelljeinek vizsgálata Python programozási nyelvvel

Készítette
Mohai Ferenc
programtervező informatikus BSc

Témavezető
Dr. Kusper Gábor
egyetemi docens

Eger, 2022

Tartalomjegyzék

Bevezetés

A szakdolgozati szemináriumon, amikor hallottam Kusper Gábor tanár úr magyarázatát a kutatásról, annak eredményeiről, céljáról, felhasználásáról. és már akkor nagyon megtetszett a téma. A SAT megoldó széles körű felhasználásáról beszélgettünk. Korábbi előadásokon, gyakorlatokon is voltak tanáraim, akik ezt a témát felvetették, és már akkoriban meghozták a kedvemet hozzá. Amikor választanom kellett, nem volt nagy kérdés, hogy ez egy számomra érdekes téma, amivel szívesen dolgozok, és lehetőséget ad a fejlődésre.

Szaktársammal, Rajna Franciskával csak mi ketten érdeklődtünk ebben a témában, úgyhogy mindenki örömmel beszélte meg a részleteket és közös megegyezéssel találtuk ki melyik ágát dolgozza ki a témának. Pozitív és energikus első benyomás után örömmel kezdtünk a munkának. Bíró Csaba és Balla Tamás tanár urakkal dolgoztunk a témával kapcsolatos házi TDK-hoz hasonló előadásokon és kutatásokon ([1, ICAI2020, AM2020]) vettünk részt. Az egyik alkalommal egy plakátot is készítettünk, ezekkel megalapozva egy lendületes kezdést.

Megbeszéltük hol szorul fejlesztésre a SAT megoldó, amin tudok programozással javítani. Valamint a korábbi általuk írt angol nyelvű szakirodalmakkal elsajátíthatom az elméleti hátteret, felzárkózhatok a jelenlegi helyzethez és ezeken dolgozva könnyedén belerázódjak a szakdolgozatom megfogalmazásába. Így kezdtem el a munkámat.

1. fejezet

Az alapok

A munkámat azzal kezdtem, hogy szakirodalmakat olvastam, fordítottam és értelmeztem, amiket korábban témavezetőim írtak [RÉSZLETEZD az alapokat SZAKIRODALOM végén]. Anyagot gyűjtöttem és dolgoztam fel a SAT megoldókról. Ezekben megjelentek különböző fogalmak, mint a tautológia [RÉSZLETEZD az alapokat alfejezetben SZAKIRODALOM FOGALMAI] cnf, és dnf, valamint programozási nyelvek, mint a Python [RÉSZLETEZD az alapokat PYTHON] és a Wolfram Alpha [RÉSZLETEZD az alapokat WOLFRAM ALPHA].

1.1. Szakirodalom

Alapfogalmak, definíciók:

1.1. Definíció. Atomi formula, röviden atom: Azt mondjuk, hogy egy szimbólum atomi formula, vagy atom, akkor és csak akkor, ha egy kifejezést jelöl. Ilyen atomok, az igaz és hamis ítéletváltozók. Szimbólumuk általában az I és H betűk magyarul, de szoktuk használni az angol megfelelőjét is a T és F betűket.

1.2. Definíció. Literál: Azt mondjuk, hogy egy szimbólum literál, akkor és csak akkor, ha egy atom, vagy annak negáltja.

1.3. Definíció. Jól formázott formula, röviden formula: Azt mondjuk, hogy egy szimbólum sorozat jól formázott formula, vagy formula, akkor és csak akkor, ha F formula a következő alakok egyikében van:

(a) A, ahol A egy atom;

(b) $\neg A$, ahol A egy formula;

(c) $(A \wedge B)$, ahol A és B formulák;

(d) $(A \vee B)$, ahol A és B formulák;

(e) $(A \implies B)$, ahol A és B formulák;

(f) $(A \Leftrightarrow B)$, ahol A és B formulák.

Minden formula a fenti esetek véges sokszori alkalmazásával áll elő.

1.4. Definíció. Formula: Adott ítéletváltozónak egy véges, nem üres V halmaza.

1. Ítéletváltozó: ha $A \in V$, akkor A formula.
2. Negáció: ha A formula, akkor $\neg A$ is formula.

1.5. Definíció. Klóz, angolul clause: Azt mondjuk, hogy egy formula klóz, akkor és csak akkor, ha adott literáloknak és formális összekötőknek egy véges, nem üres, egynél több elemű W halmaza.

1. pozitív literál: ha $B \in W$, és B egy pozitív literál.
2. negatív literál: ha B pozitív literál, akkor $\neg B$ negatív literál.
3. és formális összekötő: ha $B \wedge \neg B$, akkor $\wedge$ egy és formális összekötő.
4. vagy formális összekötő: ha $B \vee \neg B$, akkor $\vee$ egy vagy formális összekötő.

1.6. *Megjegyzés.* Létezik konjunktív és diszjunktív klóz is. Ezeket később részletezem. Diszjunktív klóz: Azt mondjuk, hogy l_i szimbólumok egy klózt alkotnak, akkor és csak akkor, ha minden l_i literál: $l_1 \vee \ldots \vee l_n$ Konjunktív klóz: Azt mondjuk, hogy l_i szimbólumok egy klózt alkotnak, akkor és csak akkor, ha minden l_i literál: $l_1 \wedge \ldots \wedge l_n$

1.7. Definíció. Implikáció: Azt mondjuk hogy formális összekötő implikáció, akkor és csak akkor, ha mindkét oldalán van egy literál, és egy harmadik literált állít elő a kettő értékéből, oly módon, hogy az összes bal oldali értékből pozitív literált állít elő, kivéve, ha a bal oldalán pozitív, a jobb oldalán negatív literál van. Mivel a negatív értékből nem következik a pozitív érték. Jelölése: $A \implies B$

1.8. Definíció. Ekvivalencia: Azt mondjuk, hogy formális összekötő ekvivalencia, akkor és csak akkor, ha mindkét oldalán van egy literál, és egy harmadik literált állít elő a kettő értékéből, oly módon, hogy ha mindkét oldalán ugyan az a pólusú literál van, akkor pozitív literált állít elő, kivéve, ha eltérnek a pólusok.

1.9. *Megjegyzés.* Pólus alatt a negatív vagy pozitív jelzőt értjük.

1.10. Definíció. Konjukció: Azt mondjuk, hogy formális összekötő konjukció, akkor és csak akkor, ha mindkét oldalán van egy literál, és egy harmadik literált állít elő a kettő értékéből, oly módon, hogy ha mindkét oldalán pozitív literál van, akkor és csak akkor pozitív literált állít elő, különben negatív literált. Jele: $\wedge$

1.11. Definíció. Diszjunkció: Azt mondjuk, hogy formális összekötő diszjunkció, akkor és csak akkor, ha mindkét oldalán van egy literál, és egy harmadik literált állít elő a kettő értékéből, oly módon, hogy ha mindkét oldalán negatív literál van, akkor negatív literált állít elő, különben pozitív literált. Jele: $\vee$

1.12. Definíció. Konjunktív normál forma, röviden KNF, angolul conjunctive normal form, röviden CNF: Azt mondjuk, hogy logikai formula konjunktív normál forma, akkor és csak akkor, ha egy vagy több klózt egymáshoz kötünk konjunkcióval.

1.13. Definíció. Diszjunktív normál forma, röviden DNF, angolul disjunctive normal form, röviden DNF: Azt mondjuk, hogy logikai formula diszjunktív normál forma, akkor és csak akkor, ha egy vagy több klózt egymáshoz kötünk diszjunkcióval.

1.14. Definíció. Interpretáció: Adott ítéletváltozóknak egy véges sok, nem üres V halmaza. Azt mondjuk, hogy a J hozzárendelés az F formula egy interpretációja, akkor és csak akkor, ha F minden atomjához vagy az igaz, vagy a hamis értéket rendeljük, de csak az egyiket.

1.15. Definíció. Logikai törvény, más néven tautológia: Azt mondjuk, hogy az F formula logikai törvény, vagy tautológia, akkor és csak akkor, ha F minden interpretációjában igaz.

1.16. Definíció. Logikai ellent mondás, angolul contradiction, unsatisfiable, röviden UNSAT: Azt mondjuk, hogy az F formula logikai ellentmondás, akkor és csak akkor, ha F minden interpretációjában hamis.

1.17. Definíció. Kielégíthető, angolul satisfiable, röviden SAT: Azt mondjuk, hogy az F formula kielégíthető, akkor és csak akkor, ha F legalább egy interpretációjában igaz.

1.18. Definíció. Hamissá tehető, angolul falseable: Azt mondjuk, hogy az F formula hamissá tehető, akkor és csak akkor, ha F legalább egy interpretációjában hamis.

1.19. *Megjegyzés.* Kielégítő ellentéte a logikai ellentmondás, mivel ha valami nem kielégíthető, akkor abból következik, hogy logikai ellentmondás. Hamissá tehető ellentéte a logikai törvény, mivel ha valami nem hamissá tehető, akkor abból következik, hogy logikai törvény.

Igazság tábla: Oszloponként tartalmazza az összes atomot, ami a formulánkban van, és az utolsó oszlopban a formulát is. Soronként minden atomhoz értéket rendel (minden lehetséges sorrendben), és a formulába behelyettesítve kiszámítja a formula értékét. Az eredményét az utolsó oszlopban láthatjuk.

Logikai törvény, tautológia: Ezek segítségével felírhatunk olyan formulákat, és igazság táblákat, amelyek minden lehetséges esetre igaz értéket adnak eredményül.

részletet illeszd be a szakirodalom fordításaidból!

1.2. WolframAlpha válasz motor

A Wolfram általános, több paradigmás programozási nyelv az alapja. Ami a hangsúlyt a szimbolikus számításra, a funkcionális programozásra, a szabályalapú programozásra helyezi, valamint képes tetszőleges struktúrákat és adatokat alkalmazni. Ezekre az alapokra épül a WolframAlpha tudás számító és válasz motor. Képes közvetlenül válaszolni a tényszerű kérdésekre azáltal, hogy külső forrásból származó adatokból számítja ki a választ.

Mi arra tudjuk használni, hogy beviteli mezőbe logikai formulát adunk át neki. Válasznak vissza adja megformázva, amit beírtunk, annak igazság tábláját, normál formáit, logikai áramkörét, Venn diagrammját és az igazság sűrűségét.

Így könnyen leellenőrizhető, hogy a logikai formula amit beírtunk az tautológia-e. Ezt az igazságtábla utolsó oszlopából láthatjuk.

1.3. A Python programozási nyelvről

A Python egy magas-szintű programozási nyelv, ami a funkcionális, az objektumorientált, az imperatív és a procedurális programozási paradigmákat támogatja. Azaz egy olyan több paradigmás nyelv, ami függvényeket, eljárásokat, metódusokat, változókat, használ, ezekkel változtatja meg az állapotát. Bár dinamikusan típusos nyelv, mégis mégis hibát dob a nem jól definiált műveletek használatára. Például nem lehet hozzá adni számot szöveghez viszont dinamikus, mert a változóknak csak nevük van, és ha véletlenül ugyan azzal a névvel egy másik típust akarunk használni, azt felül írja, és az utoljára értékül kapott típust használja. Fontos a szintaxisra nézve, hogy minden behúzás jelentős, mivel ezeket használja a kód részek csoportosítására. Hivatkozás, angolul reference számolást és kör észlelő szemétgyűjtés, angolul garbage collection (GC) amit alkalmaz a memória visszaigényléséhez. Széleskörű az alap könyvtár készlete, azaz sok importálható osztály van, amit más már megírt előttünk. Ez annak köszönhető, hogy nyújthatóvá tették modulokon keresztül az egész nyelvet. Ezen kívül dinamikus név meghatározást használ, ami a késői kötésnek, vagy lusta kiértékelésnek köszönhetően a program futása közben köti össze a neveket és a metódusokat. Funkcionális függvényei is vannak, mint a filter, map és reduce, implementálva vannak fejlett listák, angolul list coomprehension, szótárak, halmazok, és generátor, valamint iterátor kifejezések. Két alap könyvtára van a functools és az itertools. Utóbbit én is használom a programomban, de ezen kívül még a gráfokhoz kidolgozott NetworkX és Pylab könyvtárakat is használom.

2. fejezet

Kutatás

2.1. célja, alap, kiindulás modellek

2.2. eléréséhez mit csináltam

2.3. célja

3. fejezet

Szoftver

3.1. Amiből kiindultam

A munka, amihez én is hozz teszem a részemet, egy saját készítésű SAT megoldó a CSFLOCK-ról szól. Ez ugyan egy Java-ban írt program, amihez én is hozzá tudok tenni, hiszen a bemeneti formátum, amivel dolgozik, az egy .cnf fájl. Ilyen fájlokat generálnak a graph_cnf_GEN nevű programok. Ezek különböző verziókban készültek el, ahogyan előre haladtunk.

3.2. Ahova eljutottam

Összegzés

honnan hova jutottam, mi lett az eredmény. További fejlesztési lehetőségek

Irodalomjegyzék

[1] ICAI2020, AM2020 - Agria Média: ...

Nyilatkozat

Alulírott ....., büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, ... című szakdolgozat önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Aláírásommal igazolom, hogy az elektronikusan feltöltött és a papíralapú szakdolgozatom formai és tartalmi szempontból mindenben megegyezik.

Eger, 2022. április 6.

aláírás