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 [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, ez a 7. oldalon olvasható. Anyagot gyűjtöttem és dolgoztam fel a SAT megoldókról. Ezekben megjelentek különböző definíciók, mint az 1.16. definíció a tautológia, az 1.13. definíció a CNF, és az 1.14. definíció a DNF, valamint programozási nyelvek, mint a Python az 1.3. szakaszban és a Wolfram Alpha az 1.2. szakaszban.

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. Operátor: Azt mondjuk, hogy egy szimbólum operátor, akkor és csak akkor, ha egy Formulán, klózon, vagy literálon módosítást végez, oly módon, hogy kiinduló állapotát megváltoztatja, megfelelő elő feltételek mellett. Azaz két ítéletváltozó értékéből, azokat értelmezve egy harmadik értéket eredményez.

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

Klózban értelmezve,

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 operátor: ha $B \wedge \neg B$, akkor $\wedge$ egy és operátor.

4. vagy operátor: ha $B \vee \neg B$, akkor $\vee$ egy vagy operátor.

Pozitív vagy negatív literál is feltűnhet egy klózban, de sohasem egyszerre. Az is lehetséges, hogy valamelyik változó egyik formában sem tűnik fel egy klózban, hiszen egyszerűsíthetünk: $B \wedge \neg B \equiv \emptyset$.

1.7. *Megjegyzés.* Létezik konjunktív és diszjunktív klóz is. Vegyes egy klóz tartalma, ha vagy, valamint és operátorokat is tartalmaz.

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: $l_1 \wedge \ldots \wedge l_n$

1.8. Definíció. Implikáció: Azt mondjuk hogy egy operátor 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: $A \implies B$

1.9. Definíció. Ekvivalencia: Azt mondjuk, hogy egy operátor 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. Jele: $\equiv$

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

1.11. Definíció. És operátor, más néven konjukció: Azt mondjuk, hogy egy operátor 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: $\wedge$

1.12. Definíció. Vagy operátor, más néven diszjunkció: Azt mondjuk, hogy egy operátor 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: $\vee$

1.13. 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.

1.14. Definíció. Diszjunktív normál forma, 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.

1.15. 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.

1.16. 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.

1.17. 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.

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

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

1.20. *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 magát a formulát is. Soronként minden atomhoz értéket rendel, minden lehetséges sorrendben, és a formulába behelyettesítve kiszámítja a formula értékét. Az eredményét az utolsó oszlopból olvashatjuk ki.

Logikai törvény, tautológia: Ezek segítségével felírhatunk olyan formulákat, és igazság táblákat, amelyek minden lehetséges esetre igaz értéket adnak eredményül. Az eredményekből könnyebben átláthatjuk, ha tautológiát kapunk.

SAT probléma: A logikai kielégíthetőség egy olyan probléma, ami azt határozza meg, hogy létezik olyan interpretációja egy logikai formulának, ami kielégíti az adott logikai formulát.

SAT megoldó, angolul SAT solver: Hosszú, és bonyolult SAT problémák megoldására képes.

1.1.1. A tanáraim ajánlása

1.21. Definíció. Egy irányított gráf teljes, akkor és csak akkor, ha minden pár külön álló csúcsokból össze van kötve egy pár egyedi éllel (eggyel minden irányba). Egy irányított gráf erősen összetett, vagy erősen irányított gráf, akkor és csak akkor, ha van út minden csúcsból minden más csúcsba. Vegyük figyelembe, hogy a teljes gráf egyben erős is. És azt is, hogy az erősen irányított gráf tartalmaz egy kört, ami tartalmaz minden csúcsot [SYNASC2020].

1.22. Definíció. Erősen összetett gráf, más néven erős gráf: Azt mondjuk, hogy egy gráf erős gráf, akkor és csak akkor, ha van út minden pár csúcs között.

1.23. Definíció. Erősen összetett komponens, más néven erős komponens, angolul strongly connected component, röveiden SCC: Azt mondjuk, hogy egy algráf erősen összetett komponens, akkor és csak akkor, ha egy irányított gráf maximális erősen összekötött algráfja.

1.24. Definíció. Fekete hozzárendelés, más néven fekete klóz: Azt mondjuk, hogy egy klóz fekete klóz, akkor és csak akkor, ha a gráf minden csúcsának negáltja megtalálható benne.

1.25. Definíció. Fehér hozzárendelés, más néven fehér klóz: Azt mondjuk, hogy egy klóz fehér klóz, akkor és csak akkor, ha a gráf minden csúcsa megtalálható benne.

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. Amiből kiindultam

Egy élet úgy írunk le, hogy a csomópont, amiből kiindul az él, azt megjelöljük egy negatív literállal, amelyik csomópontba mutat az él, azt pedig egy pozitív literállal. Ezt egy klózban lehet tárolni.

Egy kört úgy írunk le, hogy minden csomópont, ami szerepel benne, azt egy negatív literállal jelöljük és egy klózba írjuk, úgy hogy mindegyik csak egyszer szerepeljen. Amennyiben a körből kifelé mutat él, az összeset a klóz végére fűzzük.

Erős modell: A legegyszerűbb modell. Csak az éleket tartalmazza csomópont páronként. A végére illesztve a fekete és a fehér klózokal.

Balaton boglár modell: Az alap ötlete, hogy nem köröket ír le, hanem 3as csoportosításban a csomópontokat, amik közt van út. Ezzel csak annyi a baj, hogy ezzel még nem tudunk 3-SAT problémából irányított gráfot generálni.

Egyszerűsített Balatonboglár: Ha kevesebb klózból álló erősen összetett irányított gráfot készítünk, akkor egy fekete-fehér 3-SAT modellt kapunk.

Gyenge modell: Egy kört úgy írunk le benne, hogy az összes körben szereplő csomópontot megjelöljük egy negatív literállal, annak kilépési pontjait pedig pozitív literálként. Egy gyenge modellben az összes klózban legalább egy negatív és legalább egy pozitív literálnak kell lennie. Egy csomópont csak akkor van leírva a modellben, ha van kimenő él belőle. Egy kör csak akkor van leírva a modellben, ha van kilépési pontja [1].

TODO ezt még ki kell egészíteni, és a chapter tetejétől végig olvasni (rework needed) Ahogyan azt a szakdolgozati jelentkezésembe is beleírtam, a tanárom arra kért hogy a gráfok gyenge modelljét vizsgáljam, és azokkal dolgozzak. Munkám során azonban félre értettem pár dolgot, így a gyenge modelltől elkanyarodtam az erős felé, és ott értelmeztem egy összefoglalt lépés sorozatot. Ennek eredménye, hogy egy általánosított erős modell generáló szoftvert írtam. Lásd a 2.2. fejezetben.

Amit félre értettem, hogy én az SCC-ket írtam össze és azok kilépési pontjait tettem

bele egy klózba. A weak modell pedig úgy ír le egy kört, hogy a gráf teljes hosszú klóz halmazából éseli össze a klózokat, oly módon, hogy leírja az éleket és a köröket. Tehát én egy SCC elemeit és kilépési pontjait néztem, a gyenge modellnél pedig a teljes hosszú klózokat kell össze éselni

TODO példa: *teljes hosszú klóz halmaz* kiemelni a 6os 7es sort *kép* kiemelni az a-ból kimenő éleket *gyenge modellje* előtte leírni milyen körök vannak benne, utána leírni hogy a 6os és 7es klózok az eredeti halmazból adják a-nak a kimenő éleit [1, a 24. oldalon].

2.2. Kiterjesztett erős modell

Ezt úgy értem el, hogy ha egy irányított gráf több SCC-ből áll, akkor a neki megfelelő modell mindig kielégíthető, akkor is, ha hozzáadjuk a fekete és a fehér klózokat. Ez a megállapítás független attól, hogy a gráf, melyik modelljét generáljuk le. Akkor is igaz, ha az erős, vagy ha a gyenge, vagy mondjuk a Balatonboglár modell segítségével generáljuk le a neki megfelelő SAT példányt.

Ennek az az oka, hogy az elmélet szerint akkor és csak akkor keletkezik irányított gráfból UNSAT példány, ha az irányított gráf egy SCC-ből áll, azaz erősen összefüggő, és hozzáadjuk a modelljéhez a fekete és a fehér klózokat.

Több esetet is vizsgáltam, amikor azt irányított gráf két SCC-ből áll. Ezeknek az erős modelljét elkészítettem kézzel, illetve programmal is, valamint a WolframAlpha weboldalát használtam fel ezek vizsgálatára. A WolframAlpha-t már kifejtettem a 1.2. fejezetben, így erre itt nem térnék ki minden részlete, a használatával kapcsolatban.

A legegyszerűbb gráf, aminek erős modelljét vizsgáltam, két SCC-ből áll. Az első SCC az A és a B változókból áll (számokkal kifejezve: 1, 2), a második SCC a C, valamint a D változókból (számokkal kifejezve: 3, 4), oly módon, hogy az első SCC-ből megy él a másodikba. Visszafelé természetesen nem megy él, hiszen akkor az egész gráfunkból egy nagy SCC lenne, amiről tudjuk, hogy egy fekete-fehér SAT probléma. Kutatásom során, azt találtam, hogy függetlenül attól, hogy az A-ból megy él C-be, azaz $A \to C$, vagy A-ból megy él D-be: $A \to D$, vagy B-ből megy él C-be: $B \to C$, vagy B-ből megy él D-be: $B \to D$, mind a négy esetben az erős reprezentációnak megfelelő DNF formula a következő volt:

WolframAlpha link: `https://www.wolframalpha.com/input?i=%28%28A%3D%3EB%29+and+%28B%3D%3EA%29+and+%28C%3D%3ED%29+and+%28D%3D%3EC%29+and+%28A%3D%3EC%29%29`

$$(A \wedge B \wedge C \wedge D) \vee (\neg A \wedge \neg B \wedge C \wedge D) \vee (\neg A \wedge \neg B \wedge \neg C \wedge \neg D)$$

Azaz: (A és B és C és D), vagy (nem A és nem B és nem C és nem D) vagy (nem A

és nem B és C és D)

Ezt kaptam mind a négy esetben. Mint látható az első megoldás a fekete klóz negáltja, a második a fehér klóz negáltja, amiket majd kizárunk a végső megoldások közül a fekete és fehér klózok hozzá adásával, ahogy azt előírja az eredeti algoritmus Kusper Gábor és társainak cikkében.

Ugyanakkor a harmadik megoldás nem tűnik el, és ez megfelel az eredeti értelmének, habár az eredeti elmélet nem magyarázza meg a harmadik megoldás alakját. Ha jól megnézzük, akkor ez a harmadik megoldás: (nem A és nem B és C és D), azaz az első SCC változói negatívan szerepelnek benne, a második SCC változói pedig pozitívan. Megvizsgáltam több esetet is, az erős modellt legeneráltam programmal, illetve kézzel is, és a fent leírtak mindig tökéletesen beigazolódtak. TODO programmal generált verziót beilleszteni Ez pedig egy WolframAlpha-val készült példa:

Bemeneti adatok:

$$((A \implies B) \land (B \implies A) \land (C \implies D) \land (D \implies E) \land (E \implies C) \land (A \implies C))$$

DNF:

$$(A \land B \land C \land D \land E) \lor (\neg A \land \neg B \land C \land D \land E) \lor (\neg A \land \neg B \land \neg C \land \neg D \land \neg E)$$

Mivel a megfigyelésem többször is visszaigazolódott, ezért a következő sejtést fogalmaztam meg:

TODO csomópont/node elmagyarázása, definíciója. Ha minden irányított gráf, nevezetesen G, két SCC-ből áll, S_1-ből, és S_2-ből, akkor ha S_1-ből legalább egy él vezet S_2-be, akkor függetlenül az élek számától, illetve, hogy konkrétan az élek melyik S_1-ben lévő csomópontból, melyik S_2-ben lévő csomópontba vezetnek, akkor G-nek a SAT modelljének pontosan ez az egy megoldása lesz, amennyiben a modellhez hozzáadjuk a fekete és a fehér klózokat is: $\neg S_1 \land S_2$, ahol $\neg S_1$ ez a formula: $\neg A_1 \land \neg A_2 \land \ldots \land \neg A_k$, ahol S_1 csomópontjai: $A_1, A_2, \ldots, A_k$, és S_2 ez a formula: $B_1 \land B_2 \land \ldots \land B_m$, ahol S_2 csomópontjai: $B_1, B_2, \ldots, B_m$.

Sajnos a sejtésemet egyelőre nem sikerült bizonyítani, de az általam kipróbált minden példára működött, akkor is, ha az az erős modellt generáltam, akkor is ha a gyengét, akkor is, ha bármely másikat, azaz szerintem ez egy fontos sejtés.

Első példa képekkel

2.2.1. Első példa

A fenti megfigyelésből az az ötletem támadt, hogy az összes SCC-t egy-egy csomóponttal helyettesítem, azért hogy kisebb bonyolultságú gráfokat kelljen kezelnem, amelyre az általam megírt kódok is gyorsabban futnak.

Az első út, amit kipróbáltam, az az, hogy hogyan lehet kiegészíteni a legenerált SAT modelleket, oly módon, hogy a fenti ismertetett tulajdonság megmaradjon, azaz $\neg S_1$ és S_2 megoldás maradjon, de minden egyes SCC-t egy darab változó képviseljen. Ez egy DIMACS fájlban, azt jelenti, hogy nem az SCC-ben szereplő minden csomópontot írom bele számok ként, hanem csak egy számmal jelölöm el azt.

Hosszas próbálkozás után, ezt kaptam:

Bemeneti adatok:

$$(A \implies B) \wedge (B \implies A) \wedge (A \vee B \implies X) \wedge (\neg A \vee \neg B \implies \neg X)$$

DNF:

$$(A \wedge B \wedge X) \vee (\neg A \wedge \neg B \wedge \neg X)$$

Azaz: ((A=>B) és (B=>A) és (A vagy B => X) és (nem A vagy nem B => nem X)), ahol az A, és B változókból áll az SCC, valamint X változóval lehet helyettesíteni az SCC-t.

A fenti kiegészítést. úgy lehet megkapni, hogy az SCC fehér klóza implikálja az X literált, a fekete klóza pedig a $\neg X$ literált. Ezzel a kiegészítéssel az SCC-nek ugyanúgy csak két megoldása van, mint a kiegészítés előtt. A két megoldása a fekete és a fehér hozzárendelése, annyi kiegészítéssel, hogy most már szerepel bennük az X ítéletváltozó is. Azaz a két megoldás: $(A \wedge B \wedge X)$, valamint $(\neg A \wedge \neg B \wedge \neg X)$

Ezzel a kiegészítéssel azt lehet nyerni. hogy ha van két SCC, mondjuk S_1 és S_2, ekkor S_1-et az X-el egészítem ki és S_2-t az Y-al, akkor minden élt, ami S_1-ből S_2-be megy, azt le tudom írni egy darab klózzal: $(X \implies Y)$, azaz $(\neg X \vee Y)$, ami kisebb SAT modellhez vezethet.

Ráadásul a modellek általam vizsgált összes tulajdonsága megmarad

Egy ettől is egyszerűbb megoldás, ha az első SCC-t az 1-es számmal, azaz az A változóval, a másodikat pedig a 2-es számmal, azaz a B változóval helyettesítem. Azaz rendre egy-egy változót (de mindegyikhez különbözőt) rendelek az erős komponensekhez. Ebből generálok egy erős modellt, majd ennek az erős modellnek a megoldásaimban visszahelyettesítem az SCC-k eredeti változóit a megoldásban kapott előjellel. Így akár nagyon nagy, több ezres irányított gráfok megoldása is milliszekundumok alatt lehetséges.

Ehhez természetesen meg kell találni az összes SCC-t és a köztük lévő kapcsolatokat, amit a következőképpen oldottam meg.

2.3. Kiterjesztett erős modell Python megoldásai

2.4. 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

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

[SYNASC2020] SYNASC2020 submission 77 v20: ...

[1] SAT solving 50:

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