פתרון SAT

<u>רעיון בסיסי:</u> נעזר ב backtrack search. זה אולי DPLL

* נשמור את אוסף הפסוקיות ב CNF, וננסה לספק את כולן
* נציב ערכים במשתנים. בכל פעם שנציב השמה למשתנה, נעדכן את הקבוצה של הפסוקיות (אם v משתנה או שלילתו והצבנו $v=1$, אז פסוקית מהצורה $(C \vee \bar{v})$ תהפוך לפסוקית (C) ופסוקית מהצורה $(C \vee v)$ נסיר מהקבוצה. C זה $\vee$ של כמה פסוקיות).
* אם נקבל פסוקית $()$ באוסף, נבטל את ההשמה האחרונה
* אם קיבלנו אוסף ריק, נחזיר שהפסוקיות ספיקות (וכך φ הנוסחה)

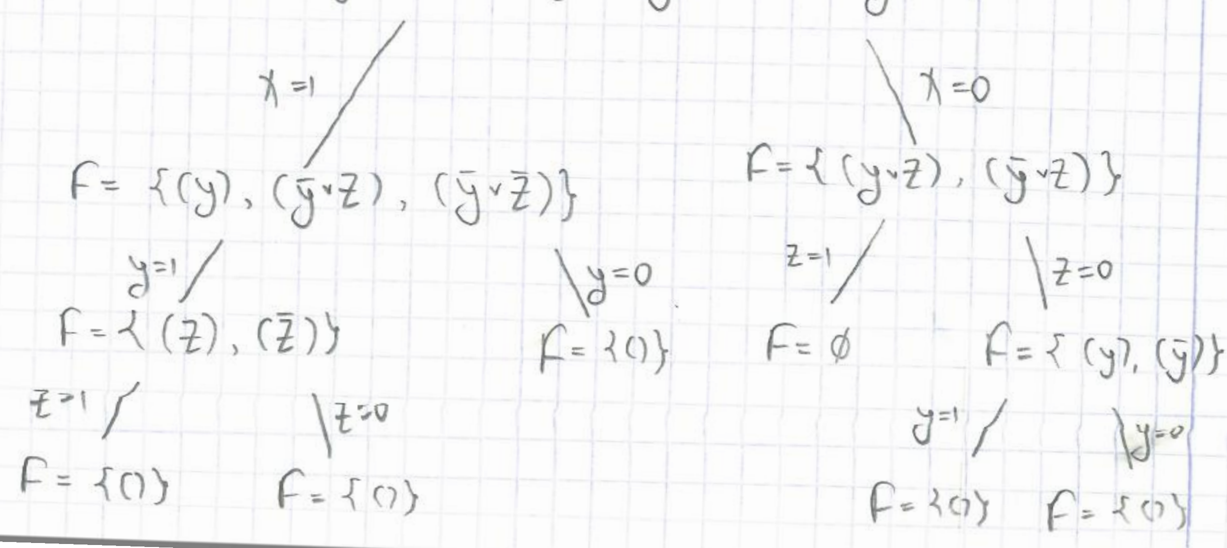
<u>באופן פורמלי:</u> F אוסף פסוקיות:

```
Search(F):
    if F = ∅
        return SAT
    if () ∈ F
        return UNSAT
    Pick x ∈ Var(F)          משתנה או שלילתו מופיע בפסוקית ב-F:
    if Search(F[x=1]) = SAT   ננסה להציב x=1
        return SAT
    return Search(F[x=0])     אם לא הצליח, אזי בהכרח x=0
```

<u>דוגמה</u> עבור הפסוקית $\varphi = (x \vee y \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{y} \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$

נקבל את הקבוצה:

$F = \{(x \vee y \vee z), (\bar{x} \vee y), (\bar{y} \vee z), (\bar{x} \vee \bar{y} \vee \bar{z})\}$



הערה: נוכל לשפר את האלגוריתם מעט אם נמצא פסוק שהוא $(x)\in F$
נציב $x=1$ ולא ננסה בכלל להציב $x=0$. "ההסקה" הזו קוראים

Unit Propagation

CDCL Conflict Driven Clause Learning

זוהי היוריסטיקה שמשפרת את ביצוע DPLL with Unit Propagation.
ע"י כך שהיא מסיקה מסקנות לגבי הקשר בין ערכי השמה במשתנים שונים, ואת מה שהיא לומדת היא שומרת כפסוקיות נוספות.
לשם כך, האלג' בונה מהנוסחה הנתונה Implication graph
הגרף יזכור את המסקנות שהסקנו לגבי ערכי משתנים, ואיך הסקנו אותן.

אלג' בניית הגרף (מתחילים עם "רמה" $=0$)

שלב 1: אם באוסף הפסוקיות יש "יחידון" (פסוקית עם משתנה יחיד)
אזי "נסיק" שהמשתנה חייב להיות 1. נוסיף לגרף קודקוד שמייצג את המשתנה.

* נשמור בקודקוד את הפסוקית המקורית אליה היה שייך המשתנה (כלומר לפני שהפסוקית הפכה ליחידון). נרשום לו את הערך "רמה" (שבוחרים).
* לכל משתנה נוסף בפסוקית המקורית, נסמנו x, נוסיף בגרף קשת מ $\bar{x}$ לקודקוד החדש שיצרנו.
* בכל פסוקית אחרת בה המשתנה מופיע, נמחק את הפסוקית מאוסף הפסוקיות הנוכחי.
* בכל פסוקית אחרת בה שלילת המשתנה מופיעה, נמחק את "שלילת המשתנה" מהפסוקית (כלומר אם הסקנו $y=1$ אזי פסוקית $(C\vee\bar{y})$ תהפוך ל (C)).

נמשיך כך עד שלא יהיו עוד יחידונים.

שלב 2: אם הסקנו (כלומר יש בגרף) משתנה ושלילתו, נעצור.

שלב 3: נעלה את ה"רמה" ב-1. נבחר משתנה נוסף שמופיע בפסוקית כלשהי, ונשים לו ערך כלשהו (0 או 1). אם הצבנו $x=1$ נוסיף קודקוד עם ערך "x", ואחרת "$\bar{x}$". נרשום שזה קודקוד "מנחש", נזכור את הרמה שלו, ונעדכן את שאר הפסוקיות.
נחזור לשלב 1.

דוגמא: (מסובכת יותר מאשר) ננית והפסוקית (כיו)

$$\varphi = (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_5) \wedge (\bar{x}_4 \vee x_5 \vee x_6) \wedge$$
$$\wedge (\bar{x}_7 \vee x_8) \wedge (\bar{x}_8 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_{10}) \wedge (x_3 \vee \bar{x}_8 \vee x_{10})$$

שלב 1: אין יחידונים

שלב 2: רמה $\rightarrow 1$

שלב 3: נבחר $x_1 = 1$. נוסיף קודקוד x_1 מרמה 1 (קודקוד מיוחד)

שלב 1: לאחר עדכון φ (שלא אפרט), (x_2) יחידון ולכן נסיק $x_2 = 1$,
נוסיף לו קודקוד וקשת מ $\bar{\bar{x}}_1$ (כי הוא איתו בפסוקית שהפכה יחידון)
(הנ"ל) $x_1 \rightarrow x_2$ ונזכור לקודקוד x_2 את הפסוקית
שהפכה ליחידון: $(\bar{x}_1 \vee x_2)$

באותו אופן אחרי עדכון נסיק $x_5 = 0$ ולכן $\bar{x}_5 = 1$ ונוסיף קודקוד
$\bar{x}_5$ שנזכר את פסוקית $\bar{x}_2 \vee \bar{x}_5$, ולכן יש לו קשת מ-x_2.

הגרף:

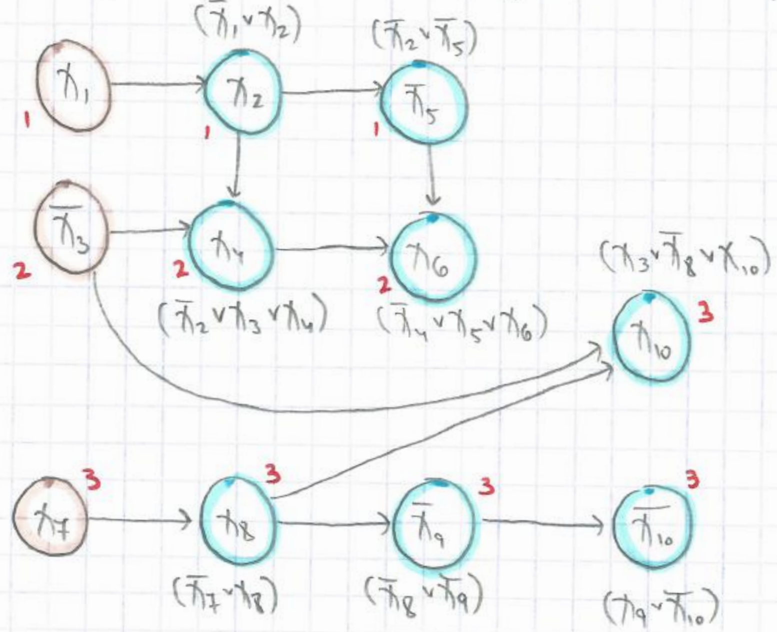
$$x_1 \rightarrow \underset{\bar{x}_1 \vee x_2}{x_2} \rightarrow \underset{\bar{x}_2 \vee \bar{x}_5}{\bar{x}_5}$$

הנוסחה נראית כעת כך: $\varphi = (x_3 \vee x_4) \wedge (\bar{x}_4 \vee x_6) \wedge$ ....

אותו דבר כמו שורה שניה

אבל נזכרים ש $x_3 \vee x_4$
נוצר מ $\bar{x}_2 \vee x_3 \vee x_4$

ניתן מכאן לעקוב אחרי המצגת, הדוגמה ממשיכה עד לקבלת הגרף:



- הסקנו
- בחרנו

זהו הקונפליקט, כי הסקנו $x_{10}, \bar{x}_{10}$

* הקודקוד כתום (decision) עבור כל החלטה של השמה שקבענו לבחור כרצוננו. דרגת הכניסה שלהם היא 0, ויש אחד מכל רמה (שאינה 0)
* קודקוד כחול הוא מוסק (implied) מהקודקודים של הקשתות אליו, כלומר אם הם שווים 1 אזי גם הוא שווה 1. הרמה שלו זה למעשה הרמה המקסימלית של הקודקוד כתום שמוביל אליו.
* אם מצאנו השמה לכל המשתנים ללא קונפליקטים, היא השמה מספקת של ולכן φ ספיקה
* אם יש קונפליקט, נרצה לבדוק איך הגענו לו וללמוד ממנו כדי לנסות באופן עיוור את כל שאר ההשמות. (בגרף לדוגמה, ראינו שיש קונפליקט אך אין מסלול מ-x_1 לקונפליקט, ולכן אין טעם שננסה לשנות את x_1 (בינתיים)).

ניתוח הקונפליקט:

נניח ובנינו גרף והגענו לקונפליקט. נסמן ב-S את שני הקודקודים בקונפליקט (בדוגמה: $x_{10}, \bar{x}_{10}$). S מהווה חתך. נתחיל להוסיף קודקודים ל-S מהרמה האחרונה בסדר, כסדר ההפוך: כלומר, כל פעם שיש קודקוד x ברמה האחרונה, עם הקודקוד $y \to x$ אחד מתקיים $y \in S$, נוסיף את x ל-S.
מכיוון שהקודקוד הראשון ברמה האחרונה (הקודקוד הכתום) הוא קודקוד decision, ממנו נובעים כל שאר קודקודי הרמה האחרונה ולכן הוא יהיה האחרון להתווסף ל-S.
אנחנו נעצור את הגדלת החתך כשהתנאי הבא מתקיים עבור S והרמה האחרונה n: קיים קודקוד יחיד $x \in S$ מרמה n, כך שקיים $y \in S$ עם $y \to x$. בהכרח נעצור, כי הקודקוד הכתום מרמה n מקיים זאת (ראינו שהוא אחרון להתווסף ל-S, בפרט הוא אחרון מרמה n עם $y \to x$ עבור $y \in S$).
נסמן את הקודקוד x בו עצרנו ב-UIP. למרות שהקודקוד הכתום מרמה n יכול להיות UIP, נשים לב שיש עוד מועמדים (בדוגמה בגרף הקודם, עבור $S=\{x_{10}, \bar{x}_{10}, \bar{x}_9\}$, הקודקוד x_8 הוא UIP)

UIP = Unique Implication Point

ניתוח קונפליקט: המשך

מצאנו חתך S והקודקוד UIP. כעת נגדיר את קבוצת המשתנים/הקודקודים הבאה: $A=\{x\notin S \mid \exists y\in S \quad y\to x\}$ אלו הקודקודים של הקשתות יוצאות שחוצה את החתך. ברור כי מאופן הגדרת A, אם ננית את A נקבל את S ובפרט הקונפליקט, לכן לא ייתכן שכל ההשמות שקבענו ב-A מתקיימות. לכן נגדיר את הפסוקית $C=\bigvee_{x\in A}\bar{x}$, כלומר יש הצבה שבה טעינו ב-A (שים לב, ש-φ ספיקה אמ"מ $C\wedge\varphi$ ספיקה). לכן נוסיף את C לאוסף הפסוקיות שלנו. לפסוקית C קוראים asserting clause כי היא לא משנה ספיקות אלא רק עוזרת לנו להסיק מסקנות לגבי ההצבות.

כעת, אנחנו יכולים "בדיעבד" להשתמש בפסוקית C כדי להסיק את הערך של UIP מכל שאר המשתנים בפסוקית (UIP מהרמה הכי גבוהה). נסמן ב-m את הרמה המקסימלית של משתנה ב-A שאינו UIP. נניח $m<n$. נבצע חזרה אחורה (backtracking) עד הרמה של הנקודה בה ביצענו את הבחירה של $m+1$, ובמקום נניח שאנחנו יודעים את ה-m בחירות הראשונות והמסקנות את הערך של UIP (דרך פסוקית C). לכן נשנה את ערכו ונמשיך באלגוריתם.

תכונות חשובות:

* חשוב לבחור את ה-UIP שהכי קרוב לקונפליקט על מנת ש-C תהיה קצרה ביותר ושנוכל להסיק יותר מההצבה של UIP.
* כאשר ה-UIP קרוב יותר לקונפליקט, S קטנה יותר וכך גם A (שהגדרנו בתחילת הפרק) קטנה יותר ולכן גם m. חשוב ש-m יהיה קטן כדי שנוכל להסיק מסקנות כבר בהתחלה.

טענה: נסמן $k=(k_0,k_1,\dots,k_N)$ וקטור שמתאר את כמות המשתנים ברמה i (N זו כמות המשתנים, k_i - כמות המשתנים ברמה i). אזי k גדל לקסיקוגרפית באלגוריתם.

הוכחה: בבחירה של ערך, הופך 0 ל-1 ולכן k גדל. כנ"ל בהסקה.

הוכחה: המשך

כאשר מגיעים לקונפליקט ברמה n, אנחנו מוסיפים פסוקית חדשה C ומבצעים backtracking לרמה $m<n$. כפי שראינו, מיד לאחר שעושים backtracking אנחנו מוסיפים את הערך של קוד ברמה m במקום n, לכן נקבל שלכל $i>m$ מתקיים $k_i=0$, ול-m מתקיים $k_m \leftarrow k_m+1$, כלומר הגדלנו את ה ההשמות של המשתנים אחרי m בתמורה למשתנה נוסף בשלב m. לכן k גדל לקסיקוגרפית.

לכן, מהטענה נסיק שאם לא נמצא השמה מספקת, האלגוריתם ימשיך (וידע לקסיקוגרפית) עד ש-k_0 גדל מספיק, כלומר נסיק שתי פסוקיות (x) ו-$(\bar{x})$ ונגיע לקונפליקט ברמה 0. במקרה הזה נקבל ש-φ לא ספיקה.

היוריסטיקות בחירה

ישנן דרכים כיצד לבחור בכל רמה באיזה משתנה נציב 1. לדוגמה:

1. לבחור במשתנה שיספק כמה שיותר פסוקיות.
2. לבחור במשתנה x שממקסם את $\sum_{x\in C} \frac{1}{2^{|C|}}$ כאשר $|C|$ כמות המשתנים. (C פסוקית)
3. VSIDS Variable State Independent Decaying Sum

שיטה מתוחכמת יותר שלכל משתנה x שומרת ערך $s(x)$. $s(x)$ מאותחל לכמות הפסוקיות שמכילות את x. מחלקים את $s(x)$ פי 2 באופן מחזורי. כאשר מוסיפים פסוקית חדשה, כל המשתנים בה מקבלים $+1$ ל-s שלהם.

שיפורים נוספים:

ישנם עוד אלמנטים בהם משתמשים כדי לייעל את האלג' כיום, כמו לדוגמה:

* לאתחל את כל הבחירות בשלב מסוים (אחרי שהוספנו כמה פסוקיות חדשות)
* למחוק פסוקיות (כי הן מיותרות או כבר לא מועילות)
* ללמוד מתוך נתונים ספציפיים לבעיה הזו (כדי לשפר מאוד ביצועים).

Certifying SAT Solvers

* אם אנחנו יודעים לפתור את SAT. קל להוכיח שהיא ספיקה ע"י הצבת השמה מספקת.
* אם נוסחה איננה ספיקה, קשה יותר להוכיח שאין השמה מספקת. כיצד נראה שנוסחה איננה ספיקה? נעשה זאת ע"י Resolution

Resolution - בהינתן שתי פסוקיות $(C \vee x)$, $(C' \vee \bar{x})$, אנחנו יודעים שאם שתיהן מסופקות אזי גם $(C' \vee C)$. זה נוח לקבל ישר פסוקיות הנ"ל Resolution לקבלת פסוקית חדשה.

טענה CNF איננה ספיקה אמ"מ יש סדרה של Resolution שיוצרת את הפסוקית הריקה

הערה יש נוסחאות של הוכחות ארוכות מאוד (רק, שכן אין ספיקות).
(הוכחה למשפט בשיעורי הבית)

PH_n - זה ה CNF המוגדר באופן הבא:

$$PH_n = \left(\bigwedge_{i=0}^{n} \left(\bigvee_{j=1}^{n} x_{i,j}\right)\right) \wedge \left(\bigwedge_{j=1}^{n} \left(\bigwedge_{i \neq k} (\overline{x_{i,j}} \vee \overline{x_{k,j}})\right)\right)$$

טענה סדרת ה Resolution שמוכיחה את אי הספיקות של PH_n היא באורך לכל n $2^{n/20}$.

הקשר בין Resolution ו Conflict Clause

באלג' CDCL הסקנו פסוקיות חדשות על סמך הקונפליקטים. כל פסוקית כזו ניתן להסיק ע"י Resolution של הפסוקיות הישירות לקונפליקט. באופן פורמלי, במקום להגדיר את החתך עד הפעם UIP להסיק פסוקית חדשה, ניתן להתחיל עם החתך המקורי $S = \{x, \bar{x}\}$, ולבצע:

* נעשה Resolution בין שתי הפסוקיות הנוכחיות (בשלב הראשון - שתי הפסוקיות של הקודקודים $x, \bar{x}$).
* נוסיף את הקודקוד לחתך, כך שאין קשתות חוצות יוצאות ממנו.
* נוסיף את הפסוקית של הקודקוד הנ"ל לסט הפסוקיות ששומרים
* נחזור לשלב הראשון

כך בכל שלב נעשה את Resolution של פסוקיות של הקודקודים. ו לקבלת

הפסוקית החדשה (asserting clause). נמשיך באופן הזה ניצור את
פסוקיות א יחידות $(\bar{x})$, (x) ואז נבצע Resolution ונקבל הפסוקית
הריקה.

Solving SAT

CNF Conjuctive Normal Form

CNF זו צורה של נוסחה בוליאנית. (אומר שנוסחה היא בצורת CNF אם היא conjuction of disjunctions, כלומר פעולת $\wedge$ של כמה פסוקיות, כשפסוקית היא פעולת $\vee$ של כמה משתנים או שלילתם. את השלילה של x מסמנים $\bar{x}$.

דוגמה $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4 \vee x_5) \wedge (x_2 \vee \bar{x}_3 \vee x_5)$

SAT SAT זו הבעיה הבאה: בהינתן נוסחה בוליאנית בצורת CNF, האם יש השמה למשתנים שמספקת את הנוסחה?

הערה: יודעים $SAT \in NPC$ כלומר $P=NP \leftarrow SAT \in P$

לכן משערים שאין פתרון לSAT בזמן פולינומי בכמות המשתנים וכמות הפסוקיות.

k-SAT אותה בעיה כמו SAT, אבל יודעים שבכל פסוקית לכל היותר k משתנים (או שלילתם).

דוגמה בנוסחה הבאה $(x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4) \wedge (x_2 \vee \bar{x}_3 \vee x_5)$ יש רק פסוקיות 3 משתנים לכן 3-SAT "יכולה להכריע" האם הנוסחה הנ"ל ספיקה.

הערה: $3\text{-}SAT \in NPC$ (כלומר אם קיים פתרון פולינומי אז $P=NP$)

$2\text{-}SAT \in P$ (יש פתרון ליניארי בכמות הפסוקיות)

תחשיב הפסוקים נוסחה בוליאנית של תחשיב הפסוקים היא נוסחה עם הקשרים $\wedge, \vee, \bar{\ }, \to$.

דוגמה נוסחה תקינה בתחשיב הפסוקים: $x_3 \wedge (x_1 \to ((\overline{\bar{x}_2 \vee x_3}) \to x_4))$

SAT הבעיה של SAT (כפי שהגדרנו מקודם) היא בעיה קשה ששואלת עבור כל נוסחה בתחשיב הפסוקים האם היא ספיקה (לא רק נוסחאות בצורת CNF).

טענה: כל נוסחה פסוקית ניתן להמיר לצורת CNF

שלב 1: נפטר מכל הקשר → ע"י השקילות $A \to B \equiv \bar{A} \vee B$

שלב 2: נדחוף את הקשר השלילה "למטה", כלומר אל המשתנים, ע"י שימוש בחוקי דה מורגן, ושלילה כפולה:

$$\overline{A \vee B} \equiv \bar{A} \wedge \bar{B} \qquad \overline{A \wedge B} \equiv \bar{A} \vee \bar{B} \qquad \bar{\bar{A}} \equiv A$$

שלב 3: נשתמש בחוק הפילוג כדי שהקשר $\vee$ יופיע רק על משתנים (וליטרלים):

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

(חוק הפילוג הזה נובע מחוק הפילוג הרגיל)

הערה: ההמרה הזו אינה פולינומית. ניתן להוכיח באינדוקציה כי בעזרת האלגוריתם הנ"ל, הנוסחה $\bigvee_{i=1}^{n} (x_i \wedge x_i')$ מומרת לצורת CNF עם 2^n פסוקיות, ולכן האלג' לוקח $O(2^n)$ (כי בכל צעד מוספים לכל היותר 2 פסוקיות).

נרצה למצוא המרה מנוסחה לצורת CNF בזמן פולינומי. לשם כך נעזר ב: Tseiting encoding: הקידוד מוסיף ליטרלים (משתנים) לנוסחה, אבל חוסך זמן בתהליך ההמרה של נוסחה ל CNF.

טענה: הקידוד הנ"ל (שיוצג בהמשך) ממיר כל נוסחה לצורת CNF שקולה בזמן פולינומי.

שלב 1: כמו בהמרה המקורית, נפטר מכל הקשר → ע"י השקילות $A \to B \equiv \bar{A} \vee B$. מכיוון שיש לכל היותר כמות ליניארית של הקשר →, השלב הזה ליניארי.

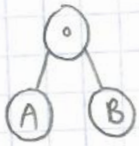
שלב 2: נדחוף את קשרי השלילה "למטה", כמו בהמרה המקורית. אנחנו עושים דחיפה אחת לכל היותר לכל הקשר $\wedge$, $\vee$ בנוסחה, ולכן כל שלב זה לוקח זמן ליניארי.

שלב 3: נבנה לנוסחה Parse Tree. f_0 זה הוא f_A שבו כל הקשר הוא צומת, הבנים של הם תת הנוסחאות בינהן הוא מקשר, והעלים משתנה (או שלילתו) הוא עלה. כלומר עבור $A \wedge B$ נבנה את

, וכנ"ל עבור $\vee$, כאשר (A), (B) הם העצים המתאימים לתתי הנוסחאות A, B בהתאמה.

שלב 4: עבור כל צומת בעץ שאינו עלה נגדיר משתנה חדש. ניצור אוסף נוסחאות (נוסחה לכל צומת), באופן הבא:

עבור הצומת



, כאשר ∘ הקשר מהסוג ∧ או ∨.

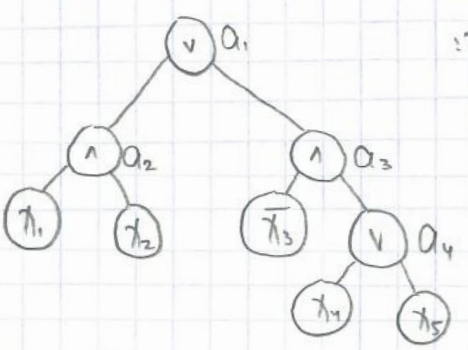
נוסיף לאוסף שלנו את הנוסחה $a_1 \leftrightarrow a_2 \circ a_3$ כאשר a_1 המייצג (החדש שהגדרנו לצומת, a_2 המייצג של הצומת (A), a_3 המייצג של הצומת (B). אם A או B הם בעצמם משתנים או שלילתם, נגדיר $a_2 = A$ או $a_3 = B$ (בהתאמה). נוסיף לאוסף את המשתנה של השורש.
יש כמות לינארית של צמתים ולכן גם שלב זה לינארי:

שלב 5: נמיר כל נוסחה באוסף הנוסחאות שלנו לצורת CNF, ואז נקשר את כל הנוסחאות ביחד עם הקשר ∧. מכיוון שכל הנוסחאות באוסף הן מהצורה $a_1 \leftrightarrow a_2 \wedge a_3$ או $a_1 \leftrightarrow a_2 \vee a_3$ אזי לוקח זמן קבוע להמיר אותן לצורת CNF ולכן גם שלב זה לינארי (לינארי בכמות הנוסחאות, וזה לינארי באורך הנוסחה המקורית).

לפני שנוכיח נכונות (כלומר שקילות של התוצאה לנוסחה המקורית), נראה דוגמה:

דוגמה: נבצע את האלג' על הנוסחה $(x_1 \wedge x_2) \vee (\bar{x}_3 \wedge (x_4 \vee x_5))$ (שני השלבים הראשונים ברורים, כבר ראינו אותם בהמרה הקודמת).

שלב 3: נבנה את העץ המתאים לנוסחה:



שלב 4: נגדיר לכל צומת שאינו עלה משתנה חדש, וניצור אוסף נוסחאות:

$a_1 \leftrightarrow a_2 \vee a_3$

$a_2 \leftrightarrow x_1 \wedge x_2$

$a_3 \leftrightarrow \bar{x}_3 \wedge a_4$

$a_4 \leftrightarrow x_4 \vee x_5$

a_1

שלב 5: כל נוסחה מהצורה $x \leftrightarrow y \circ z$ נמיר לצורת CNF ואז נחבר "∧" של כולם (זה סתם ארוך ומייגע).

צורת CNF שקולה ל $x \leftrightarrow y \vee z$:

$$x \leftrightarrow y \vee z \equiv (x \rightarrow y \vee z) \wedge (y \vee z \rightarrow x) \equiv$$
$$\equiv (\bar{x} \vee y \vee z) \wedge (\overline{y \vee z} \vee x) \equiv (\bar{x} \vee y \vee z) \wedge ((\bar{y} \wedge \bar{z}) \vee x) \equiv$$
$$\equiv (\bar{x} \vee y \vee z) \wedge ((\bar{y} \vee x) \wedge (\bar{z} \vee x)) \equiv$$
$$\equiv (\bar{x} \vee y \vee z) \wedge (\bar{y} \vee x) \wedge (\bar{z} \vee x)$$

צורת CNF שקולה ל $x \leftrightarrow y \wedge z$:

$$x \leftrightarrow y \wedge z \equiv \bar{x} \leftrightarrow \overline{y \wedge z} \equiv \bar{x} \leftrightarrow \bar{y} \vee \bar{z} \equiv$$
$$\equiv (\bar{\bar{x}} \vee \bar{\bar{y}} \vee \bar{\bar{z}}) \wedge (\bar{\bar{y}} \vee \bar{x}) \wedge (\bar{\bar{z}} \vee \bar{x}) \equiv$$

ראינו אצל

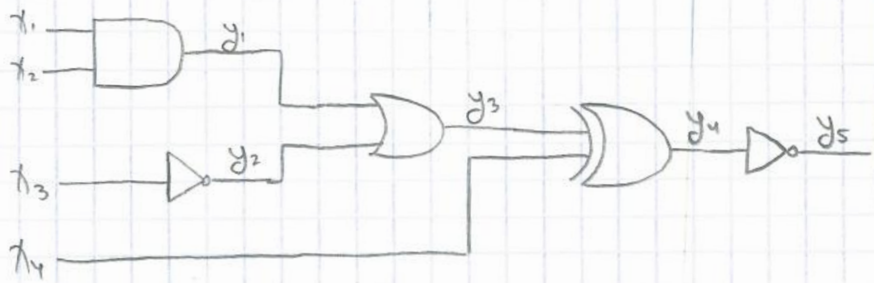
$$\equiv (x \vee \bar{y} \vee \bar{z}) \wedge (y \vee \bar{x}) \wedge (z \vee \bar{x})$$

הערות: האלג' מספק נוסחה שקולה לנוסחה המקורית. ניתן לראות באינדוקציה עבור ה צומת עלה. שאם האלג' נכון על הבנים של הצומת, אז האלג' נכון גם עליו. ההוכחה הפורמלית היא ארוכה אבל לא מסובכת.

הרעיון שעומד מאחורי האלג' הוא "לגז" נוסחה ע"י משתנים חדשים. ה C_i הם הוא משתנה. כל שער הוא השער (או צומת ב parse tree), וה C_i הוא המשתנה החדש שיצרנו עבור השער.

דוגמה, נוסחה, $\overline{((x_1 \wedge x_2) \vee \bar{x}_3) \oplus x_4}$, מתאים המעגל

השער xor שלא הגדרנו

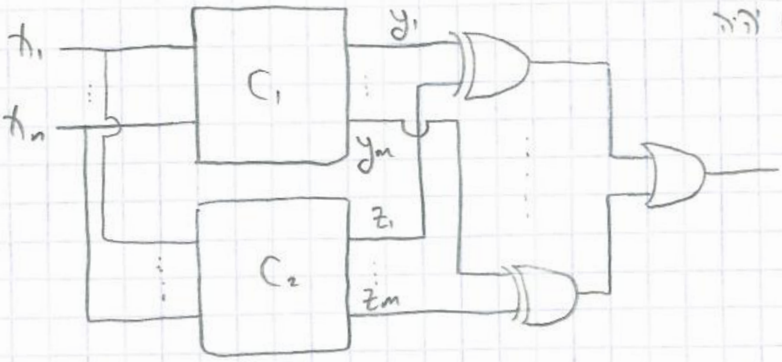


שמתאים לאוסף הנוסחאות

$$y_5 \wedge \begin{cases} y_1 \leftrightarrow x_1 \wedge x_2 \\ y_2 \leftrightarrow \bar{x}_3 \\ y_3 \leftrightarrow y_1 \vee y_2 \\ y_4 \leftrightarrow y_3 \oplus x_4 \\ y_5 \leftrightarrow \bar{y}_4 \end{cases}$$

שימושים של SAT

1. **מציאת שקילות של מעגלים בוליאניים:** בהינתן שני מעגלים בוליאניים עם n קלטים ו- m פלטים כל אחד, נרצה לבדוק האם המעגלים שקולים ע"י SAT (מעגלים שקולים אם הם מחזירים A לכל קלט). אם נסמן ב $\{x_i\}_{i=1}^n$ את הקלט, נגדיר $C(\{x_i\}_{i=1}^n) = \bigvee_{i=1}^m (y_i \oplus z_i)$ כאשר $C_1(x)=y$ $C_2(x)=z$ כאשר C_1, C_2 המעגלים. כלומר המעגל החדש יהיה



והמעגל החדש ספיק אמ"מ קיים i כך ש $y_i \oplus z_i = 1$, כלומר $y_i \neq z_i$, כלומר הוא ספיק אמ"מ $C_1 \neq C_2$

2. **Bounded Model Checking:** נתונים:

* אוסף מצבים. כל מצב מתואר ע"י n ביטים $\{x_i\}_{i=1}^n$
* יחס ρ על המצבים. $\rho(x,y)$ אומר שניתן לעבור ממצב x למצב y בצעד אחד.
* תכונה P על המצבים.

נרצה לבדוק האם, עבור מצב מסוים ו $k \in \mathbb{N}$, יש מסלול באורך קטן שווה k מ-x למצב אחר שמקיים תכונה P.

מכיוון שמצבים מתוארים ע"י ביטים, נוכל לרשום את $\rho(x,y)$ כפונקציה (נוסחה לוגית) של הביטים של y, x. וכנ"ל את $P(x)$ הנוסחה לוגית התלויה רק בביטים של x. נקבל שיש מסלול כנ"ל מ-x^0 אמ"מ הנוסחה הבאה ספיקה: $\left(\bigwedge_{i=0}^{k-1} \rho(x^i, x^{i+1})\right) \wedge \left(\bigvee_{i=1}^{k} P(x^i)\right)$

כאשר לכל $0<i\le k$, x^i הוא n-יה של משתנים. x^0 זה ביטים ידועים מראש.

3. צביעת גרפים בהינתן גרף G ומספר $k \in \mathbb{N}$ נרצה לדעת האם הגרף k צביע. נסמן ב-n את כמות הקודקודים. אז: הגרף k-צביע אמ"מ הנוסחה הבאה ספיקה: $A \wedge B \wedge C$ כאשר:

* $A = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{k} x_{i,j} \right)$ אומר לכל קודקוד יש צבע

* $B = \bigwedge_{i=1}^{n} \left(\bigwedge_{t=1}^{k-1} \bigwedge_{s=t+1}^{k} \left(\overline{x_{i,t}} \vee \overline{x_{i,s}} \right) \right)$ אומר אין קודקוד עם שני צבעים.

* $C = \bigwedge_{(i,j) \in E} \left(\bigwedge_{c=1}^{k} \overline{x_{i,c}} \vee \overline{x_{j,c}} \right)$ אומר לכל שני שכנים ולכל צבע, אחד מביניהם לא בצבע הזה (כלומר אין להם צבע זהה)

כאשר המשתנה $x_{i,j}$ "מייצג" האם קודקוד i צבוע בצבע j.