גירסה 1.19 – 26.2.2009

מבוא לאימות תוכנה

סוכם ע"י: ניר אדר

תוכן עניינים

1. מבוא

מסמך זה הינו מסמך תיאורטי בתחום המתמטי אימות תוכנה (Software Verification) ובדיקת מודל (Model Checking). תחום אימות התוכנה עוסק בשיטות להוכחת נכונות של תוכניות, מעגלים חשמליים ומערכות שונות. מסמך זה יציג חלק מהשיטות הקיימות.
המסמך מסתמך במידה רבה על הקורס **מבוא לאימות תוכנה** בטכניון אך הוא אינו חומר רשמי של הקורס אלא סיכום אישי בלבד, וכן על ויקיפדיה האנגלית ועל מספר מאמרים מקצועיים נוספים בתחום. רשימת המקורות המלאה נמצאת בסוף המסמך. ברצוני להודות **לפרופסור ארנה גרימברג** על שעות רבות של שאלות ועל סבלנות אינסופית בדרכי להבין את החומר.

כדי להגדיר נכונות נדבר על דרישות מהתוכנית. תוכנית הינה נכונה ביחס לדרישות שלה. הדרישות יכונו מפרט או ספסיפיקציה. נתאר את תוכניות המחשב כישויות מתמטיות, ונציג כלים להתייחס אליהן:

- מהו המצב ההתחלתי של התוכנית?
- באיזה מצב היא עוצרת (אם בכלל)?
- האם התוכנית מבצעת את החישוב שהיא היתה אמורה לבצע? (כלומר, עומדת במפרט המתקבל)

מפרט הינו דרך מתמטית לתאר את מצב התוכנית הסופי הרצוי בהנתן מצב התחלתי. נציג מפרטים שונים, ובדרכים לבדוק את התוכנית מולם.
נתרכז גם בהגדרת **כללי הוכחה**, שהם רשימות צעדים ובמידה ונצליח לבצע אותם בהצלחה על תוכנית, נוכל להסיק עליה מסקנות.

בדיקה דינאמית היא בדיקה המבוצעת בזמן ריצה. דוגמא בשפת C היא פקודת assert(condition) העוצרת את ביצוע התוכנית אם התנאי שבה לא מתקיים. בדיקה דינמאית נותנת לנו תוצאה על ריצה מסויימת של התוכנית.
בדיקה סטאטית: באופן אידיאלי: בדיקת טענות בזמן קומפילציה על הקוד. אם הקומפילציה עוברת בהצלחה – הטענה מתקיימת בכל ריצה של התוכנית. הבעיה הכללית של בדיקת תוכנה ופלט איננה ניתנת להכרעה, כפי שניתן ללמוד בתחום החישוביות. עם זאת, עבור מקרים פרטיים: סוגים מסויימים של תוכנות, סוגים מסויימים של בדיקות – ניתן בהחלט להוכיח נכונות ואנו נציג מקרים אלה.

מסמך זה מתרכז בבדיקות סטאטיות של תוכניות.

הנושאים העיקריים בהם מסמך זה עוסק:

1. הוכחת נכונות של תוכניות ע"י לוגיקה – תוכניות בעלות מספר מצבים אינסופי.
2. בדיקת מודלים במערכות בעלות מספר סופי של מצבים.

הבלדים בין תוכניות עם מס' מצבים סופי לתוכניות עם מס' מצבים אינסופי:

תוכניות עם מספר מצבים אינסופי	תוכניות עם מספר מצבים סופי
שיטת לאימות: אין הוכחה שהתוכנית לא מספקת את המפרט, אם לא הצלחנו להוכיח שהיא כן מספקת אותו.	שיטות לאימות: אלגוריתמים (יעילים) המחזירים תשובה חד משמעית אם התוכנית מקיימת את המפרט.
מפרט: הינו שפה בסדר ראשון. הרחבה של תחשיב היחסים.	מפרט: שימוש בלוגיקה טמפורלית פסוקית הכוללת דרך לבטא התנהגות לאורך זמן.
התוכנית מיוצגת כטרנספורמצית קלט-פלט. מקבלת x, y נתונים ומחזירה לדוגמא $x \cdot y$. אנו מתעניינים בתוצאות ופחות בדרך החישוב.	תוכניות ריאקטיביות: מערכות הפעלה, פרוטוקולי תקשורת. מערכות שאמורות לפעול לנצח ולא לעצור. מגיבות לקלט מהסביבה.
מחשוב: תוכנות כגון Teoram Prover. דורשות התערבות של המשתמש.	מחשוב: בדיקת מודל - כלים אוטומטיים לחלוטין. שיטות מעולם האלגוריתמים בתורת הגרפים.

מומלץ לחזור לעיין בטבלה זו בהמשך קריאת המסמך לאחר שתקבלו הכרות ראשונית עם התחום.

2. לוגיקה מסדר ראשון - תקציר

במהלך מסמך זה אנו נעבוד עם תחשים היחסים, כאשר סימני הפונקציה וסימני היחס שלנו יהיו סימנים סטנדרטיים במשמעותם המוכרת. ערכי המשתנים ימלאו את תפקיד ההשמה. הכתיב $\sigma \vDash q(\bar{x})$ מבטא שהנוסחה q נכונה כאשר במקום המשתנים החופשיים שמים ערכים מ- σ.

הדפים הבאים יציגו רענון של תחשיב היחסים מעולם הלוגיקה, ללא ההפשטות בהן נשתמש במהלך המסמך. במידה והינכם שולטים בלוגיקה, ניתן לדלג ישירות לפרק הבא.

2.1. תזכורת – תחשיב היחסים

<u>הסימנים של תחשיב היחסים</u>

הסימנים מתחלקים ל-2 קבוצות:

1. סימנים לוגיים – סימנים המשותפים לכל השפות של תחשיב היחסים.
2. פרמטרים של השפה – מילון – סימנים המיוחדים לשפה.

<u>הסימנים הלוגיים</u>

- כמתים: לכל $\forall$, קיים $\exists$.
- קשרים של תחשיב הפסוקים: $\{\rightarrow, \wedge, \vee, \neg\}$.
- סוגריים ופסיק.
- סימן שוויון $\approx$.
- משתנים: $\{v_i \mid i \in \mathbb{N}\}$.

<u>מילון: פרמטרים של השפה</u>

- **סימני קבועים** אישיים מסומנים ב- C_α כאשר α אינדקס מספרי.
- **סימני יחס** המסומנים ב- $R_{n,\alpha}$ כאשר $n, \alpha \in \mathbb{N}$. n הוא מספר הארגומנטים ביחס ו- α הוא אינדקס מספרי.
- **סימני פונקציה** המסומנים ב- $F_{n,\alpha}$. n הוא מספר הארגומנטים ביחס ו- α הוא אינדקס מספרי.

מילון: אוסף סימני יחס, סימני פונקציה וסימני קבועים אישיים, כאשר לכל סימן פונקציה ולכל סימן יחס ידוע מספר הארגומנטים. מילון מסומן לרוב באות τ. לדוגמא:

$\tau_0 = \langle R_{1,0}, R_{1,1}, F_{2,0}, F_{2,3}, C_2, C_3 \rangle$.

מילון סופי הוא מילון המכיל מספר סופי של סימנים. **מילון יחסי** הוא מילון המכיל רק סימני יחס.

<u>הגדרת אוסף הטענות בשפה</u>

שלב 1: הגדרת אוסף העצמים עליהם מדברים. **שמות עצם** אינם טענות שניתן לשאול עליהם נכון / לא נכון.

שלב 2: נגדיר את אוסף הטענות בשפה שתיקראנה **נוסחאות** על סמך שמות העצם.

הגדרת אוסף שמות העצם

ההגדרה הינה באינדוקציה: (סימני פיסוק, משתנים $\cup$ קבועים)X.

<u>קבוצת האטומים</u>: סימני הקבועים מהמילון בתוספת המשתנים $\{v_i \mid i \in \mathbb{N}\}$.

<u>קבוצת הפעולות</u>: לכל סימן פונקציה $F_{n,\alpha}$ במילון נגדיר פעולה המקבלת שמות עצם $t_1,...,t_n$ ומוציאה כפלט את $F_{n,\alpha}(t_1,...,t_n)$.

<u>הערה</u>: מספר הפעולות הינו כמספר הפונקציות במילון של השפה. ישנן שפות ללא פונקציות, ולכן ללא פעולות ליצירת שמות עצם. מספר שמות העצם בשפות אלו זה למספר האטומים.

הגדרת אוסף הטענות/נוסחאות.

ההגדרה הינה באינדוקציה. קבוצת האטומים הינה אוסף סדרות הסימנים מהצורה $R_{n,\alpha}(t_1,...,t_n)$ כאשר $t_1,...,t_n$ שמות עצם, ו- $R_{n,\alpha}$ הינו סימן יחס n-מקומי מהמילון. כל סדרת סימנים כזו תקרא **נוסחה אטומית**.

אבחנה: בשם עצם יכולות להיות כלולות כמה פונקציות. בנוסחה אטומית יש רק סימן יחס אחד (ואחד או יותר שמות עצם).

עבור הגדרת אוסף הנוסחאות נתייחס לסימן "=" כאל סימן יחס דו מקומי. דוגמא:

$F(x,x) = F(x,y)$ זוהי נוסחה אטומית.

קבוצת אוסף הפעולות מתחלקת לשני חלקים:

1. הפעלת הקשרים של תחשיב הפסוקים. אם α, β נוסחאות, אזי גם הביטויים הנ"ל הם נוסחאות: $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \to \beta), (\neg\alpha)$.
2. הפעלת **כמתים**: לכל נוסחה α ומשתנה x, גם $\forall x\alpha$ ו- $\exists x\alpha$ הן נוסחאות.

הערה: מניחים קדימות לכמתים: $\forall v_i P(v_i) \to R(v_r) \equiv (\forall v_i P(v_i)) \to R(v_r)$

<u>הבדלים בין סימני יחס לפונקציות</u>

$F(F(\ ))$ הינו שם עצם. $R(R(\ ))$ לא מוגדר כחלק מתחשיב היחסים!

$R(F(\ ))$ הינה נוסחה אטומית. $F(R(\ ))$ לא קיים – אסור להפעיל סימני פונקציה על יחס.

$R \to R$ מותר. $F \to F$ - אסור. יש לתת סימני יחס בין הקשרים והכמתים.

$\forall xR$ מותר. $\forall xF$ אסור.

<u>סמנטיקה לתחשיב היחסים - הגדרה אינטואיטיבית</u>

בהינתן מילון τ, $\tau = \langle R_1,...,R_m,F_1,...,F_k,C_1...,C_l \rangle$ **מבנה** עבור τ מוגדר כך:

$$M = \langle D^M, R_1{}^M,...,R_m{}^M, F^M{}_1,...,F_k{}^M, C_1{}^M ...,C_l{}^M \rangle$$

כאשר D^M הוא התחום, R_i^M הוא הפירוש של סימן היחס R_i במבנה M, F_i^M הוא הפירוש של סימן הפונקציה F_i במבנה M, C_i^M הוא הפירוש של סימן הקבוע C_i במבנה M.

דוגמא: $\tau = \langle R_{2,0}, F_{1,0}, C_0, C_1 \rangle$. $M_1 = \langle \mathbb{N}, \leq, +1, 2, 3 \rangle$, $M_2 = \langle P(A), \subseteq, \text{completion}, \phi, A \rangle$.

בהינתן מבנה עבור מילון τ, נגדיר **השמה** z המתאימה למשתנים ערכים מהתחום:

$$z : \{v_i \mid i \in \mathbb{N}\} \to D^M$$

נגדיר **הרחבה של ההשמה** z, שתתאים לכל שם עצם t מעל המילון τ איבר בתחום שיסומן $\overline{z}(t)$

.

נגדיר באינדוקציה על מבנה שמות העצם:

בסיס: t משתנה $\Leftarrow \overline{z}(t) = z(t)$. t קבוע $\Leftarrow t = c_i \Leftarrow \overline{z}(c_i) = c_i^M$.

סגור: $\overline{z}(F_{n,\alpha}(t_1,...,t_n)) = F_{n,\alpha}^M(\overline{z}(t_1),...,\overline{z}(t_n))$.

משתנים חופשיים וקשורים

1. $\forall v_1 R_{2,0}(v_1, v_1)$ - כל האיברים בתחום הם ביחס עם עצמם.
2. $R_{2,0}(v_1, v_1)$ - v_1 ביחס $R_{2,0}$ עם עצמו.

בדוגמא הראשונה אין צורך לדעת מה הערך של v_1 בהשמה, ובנוסחה השניה כן. בנוסחה הראשונה v_1 מופיע קשור (תחת השפעת הכמת), לכן הוא אינו מופיע בתרגום הנוסחה למילים ואין צורך לדעת את הערך שההשמה נתנה לו. בנוסחה השניה v_1 חופשי.

טענה: תהי α נוסחה מעל מילון τ ו- M מבנה עבור τ. אם z_1 ו- z_2 השמות ב- M המזדהות על המשתנים החופשיים ב- α, אז ערך האמת של α תחת z_1 ותחת z_2 זהה.

מסקנה: אם α נוסחה ללא משתנים חופשיים, אז ערך האמת של α אינו תלוי בהשמה.

הגדרה: נוסחה α בלי משתנים חופשיים נקראת **פסוק**.

הגדרה פורמלית: נגדיר באינדוקציה על מבנה הנוסחה α מתי v_i הוא **משתנה חופשי** ב- α.

בסיס: נוסחאות אטומיות: v_i חופשי ב- α אם v_i מופיע ב- α.

סגור: קשרים: עבור $\circ \in \{\wedge, \vee, \rightarrow\}$, v_i חופשי ב- $(\alpha \circ \beta)$ אם v_i חופשי ב- α או v_i חופשי ב- β.

v_i חופשי ב- $(\neg\alpha)$ אם v_i חופשי ב- α.

כמתים: v_i חופשי ב- $\forall v_j \alpha$ או ב- $\exists v_j \alpha$ אם v_i חופשי ב- α וגם $i \neq j$.

משתנה שאינו חופשי נקרא **משתנה קשור**.

סמנטיקה לתחשיב היחסים - הגדרה פורמלית

הגדרת מבנה: בהינתן מילון τ: $\tau = \langle R_1, ..., R_m, F_1, ..., F_k, C_1, ..., C_l \rangle$ מבנה עבור τ הוא:

$$M = \langle D^M, R_1{}^M, ..., R_m{}^M, F^M{}_1, ..., F_k{}^M, C_1{}^M ..., C_l{}^M \rangle$$

כאשר D^M הוא התחום.

- לכל $1 \le i \le m$ אם R_i הוא סימן יחס n-מקומי אזי $R_i^M \subseteq \underbrace{D^M \times ... \times D^M}_{n \text{ times}}$, כלומר R_i^M הוא יחס n-מקומי מעל D^M.
- לכל $1 \le i \le k$ אם F_i סימן פונקציה n-מקומי אז $F_i^M : \underbrace{D^M \times ... \times D^M}_{n \text{ times}} \rightarrow D^M$, כלומר F_i^M מתארת מה הערך של F_i עבור כל n-יה סדורה של איברים מ- D^M.
- לכל $1 \le i \le l$, מתקיים: $C_i^M \in D^M$.

ראינו כיצד בהינתן מבנה M מגדירים השמה z ב- M: $z:\{v_i \mid i\in\mathbb{N}\}\to D^M$ וראינו כיצד להרחיב את z כך שתתאים ערך מהתחום לכל שם עצם.

בהינתן נוסחה α מעל מילון τ, מבנה M והשמה z עבור M נסמן $M\underset{z}{\models}\alpha$ את הטענה ש- α **מסתפקת** ב- M תחת z.

הגדרת ערך האמת: נגדיר באינדוקציה על מבנה α מתי $M\underset{z}{\models}\alpha$.

בסיס: α נוסחה אטומית: $\alpha=(t_1\approx t_2)$ עבור t_1,t_2 שמות עצם, $M\underset{z}{\models}\alpha\Leftrightarrow\overline{z}(t_1)=\overline{z}(t_2)$.

כמו כן כאשר $\alpha=R(t_1,...,t_n)$, עבור $t_1,...,t_n$ שמות עצם, $M\underset{z}{\models}\alpha\Leftrightarrow(\overline{z}(t_1),...,\overline{z}(t_n))\in R^M$.

סגור:

1. קשרים: מחשבים את ערך האמת בעזר טבלאות האמת. לכל נוסחה α, מבנה M והשמה z, מתקיים כי $M\underset{z}{\models}\alpha$ או $M\underset{z}{\models}\neg\alpha$ אבל לא שניהם.
2. כמתים: נניח שלכל מבנה M' והשמה z' אנו יודעים האם $M'\underset{z'}{\models}\alpha$. נרצה לדעת האם $M\underset{z}{\models}\forall v_i\alpha$, $M\underset{z}{\models}\exists v_i\alpha$ למבנה והשמה מסויימים. בהינתן השמה z, משתנה v_i ואיבר $d\in D^M$, נגדיר השמה מתוקנת $z[v_i\leftarrow d]$:

$$z[v_i\leftarrow d](v_j)=\begin{cases}z(v_j) & i\neq j\\ d & i=j\end{cases}$$

נגדיר: לכל $d\in D^M$ מתקיים $M\underset{z}{\models}\forall v_i\alpha\Leftrightarrow M\underset{z[v_i\leftarrow d]}{\models}\alpha$.

קיים $d\in D^M$ עבורו $M\underset{z}{\models}\exists v_i\alpha\Leftrightarrow M\underset{z[v_i\leftarrow d]}{\models}\alpha$.

2.2. *תזכורת – מונחים מעולם הלוגיקה*

נוסחה α הינה **אמת לוגית** אם לכל מבנה m ולכל השמה S מתקיים: $m,S\models\alpha$.

סימון: $\models\alpha$. לדוגמה: $\forall x(R(x,y)\vee\neg R(x,y))$

מערכת הוכחה: אוסף של כללים ליצירת טענות.

אם ניתן ליצור את α על ידי מערכת הוכחה D אז נאמר ש α **יכיחה** במערכת D ונסמן: $\vdash_D\alpha$.

אם לא ניתן ליצור את α מהמערכת D נאמר כי α **איננה יכיחה** ב D ונסמן: $\nvdash_D\alpha$.

מערכת הוכחה **נאותה** היא מערכת בה כל טענה יכיחה הינה ספיקה. $\vdash_D\alpha\rightarrow\models\alpha$

מערכת הוכחה **שלמה** היא מערכת בה ניתן להוכיח כל טענה נכונה. $\models\alpha\rightarrow\vdash_D\alpha$

3. הוכחת נכונות של תוכניות

בחלק זה נציג תוכניות כיחס קלט-פלט עם מספר מצבים אינסופי, ונפרט את הדרישות מהן באמצעות מפרט שישתמש בנוסחאות מסדר ראשון.

<u>מרכיבים נדרשים למערכת ההוכחה של תוכנית:</u>

1. **שפת מפרט** לתיאור הפונקציונאליות הנדרשת מן המערכת. שפה מתמטית עם סמנטיקה מדוייקת.
2. **שפת תכנות** עם סמנטיקה פורמאלית.
3. **כללי הוכחה** שיאפשרו להוכיח <u>טענות נכונות</u>.

<u>המטרה שלנו:</u> בהינתן תוכנית P, נרצה להוכיח שהתוכנית מקיימת את המפרט שלה.

<u>דוגמא למפרט</u> המתאר תוכניות מיון: $\forall 0 \le i, j \le n \;\; a[i] \le a[j]$

<u>הגדרה</u>: תהי תוכנית P ומפרט φ. נאמר כי P **נכונה** ביחס ל-φ אם מתקיים $P \models \varphi$ (**P מספק את φ**).

איך נבדוק ספיקות של תוכנית?

- תוכנית עם מספר מצבים סופי: נעבור על כל המצבים ונבדוק את ספיקות התוכנית.
- תוכנית עם מספר מצבים אינסופי (שהוא טרנספורמציית קלט-פלט): דרושה מערכת הוכחה.

נטפל בתוכניות דטרמיניסטיות בלבד, כלומר ישנו חישוב יחיד לכל פלט $\Leftarrow$ לכל קלט יש תוצאה אחת לכל היותר. (לכל היותר = התוכנית יכולה לא לעצור כלל).

<u>נאותות (תקפות) ושלמות בהקשר למערכת ההוכחה:</u>

נדרוש שמערכת ההוכחה איתה נעבוד תהיה שלמה ונאותה.

- **נאותות**: אם הצלחנו להוכיח טענת נכונות, אז הטענה אכן נכונה.
- **שלמות**: אם הטענה נכונה, ניתן להוכיח אותה במערכת ההוכחה.

3.1. הגדרות בסיסיות

עבור תוכנית P, **מצב של תוכנית** σ יוגדר כהשמה מסויימת למשתני התוכנית.
מרחב המצבים (state-space) הינו מספר המצבים האפשריים בתוכנית. **מרחב המצבים הישיגים במערכת** הינו מספר המצבים אליהם ניתן להגיע מהמצב ההתחלתי של המערכת.
ניסוח נוסף: **מצב של תוכנית** הינו פונקציה ממשתני התוכנית לתחום שבו הם מקבלים את ערכיהם.

1. **משתני התוכנית**: המשתנים יתוארו ע"י הווקטור: $\bar{x} = (x_1, x_2, ..., x_n)$.
2. **מצב התוכנית** σ הינו פונקציה ממשתני התוכנית לתחום הערכים שלהם.
   הערך של המשתנה x במצב σ מסומן על ידי $\sigma(x)$. לדוגמא: $\sigma(x) = 5$.
3. **מפרט** הינו זוג טענות מהצורה $< q_1(\bar{x}), q_2(\bar{x}) >$ כאשר $q_1(\bar{x}), q_2(\bar{x})$ הן טענות מתחשיב היחסים מעל משתני התוכנית $\bar{x}$. המשמעות של המפרט היא שתנאי הקדם $q_1(\bar{x})$ גורר את קיום התנאי $q_2(\bar{x})$ עם סיום התוכנית. (q_1 המצב בתחילת התוכנית, q_2 המצב בסופה).
   - דוגמא: תהא תוכנית בעלת המשתנים (x, y, b) כאשר: $x \in \mathbb{N}, x \in \mathbb{R}, b \in \{T, F\}$
     מפרט לדוגמא יהיה $\langle x > 0 \wedge b = T, y > x \wedge b = F \rangle$.
   - דגש: צריך לשים לב שבמפרט המצב של $\bar{x}$ ב-q_1 אינו המצב של $\bar{x}$ ב-q_2.

3.2. חישוב של תוכנית

חישוב של תוכנית P ממצב σ, המסומן על ידי $\boldsymbol{\pi(P,\sigma)}$, הינו סדרה סופית של מצבים $\sigma_1,...,\sigma_k$ או סדרה אינסופית של מצבים $\sigma_1,\sigma_2,...$, המקיימים כי לכל i, המעבר מ-σ_i אל σ_{i+1} הוא בהתאם לתוכנית P, וכן כי $\sigma_1=\sigma$.

המצב הסופי של החישוב יסומן $\boldsymbol{Val(\pi)}$. במידה והחישוב סופי, הרי ש-$Val(\pi)=\sigma_k$. במידה והחישוב הוא אינסופי נסמן $Val(\pi)=\perp$. הערך $\perp$ הוא ערך לא מוגדר, השונה מכל הערכים (**Bottom**).

<u>הגדרה</u>: עבור מצב σ וטענה $q(\bar{x})$, נסמן $\sigma \models q(\bar{x})$ (**המצב מספק את הנוסחה/טענה**) אם הטענה $q(\bar{x})$ נכונה כאשר מציבים במקום המשתנים החופשיים ב-$q(\bar{x})$ את הערכים המתאימים להם ב-σ.

לדוגמא, תהי הטענה: $q(y)=\forall x(x>y \vee x<y \vee x|2)$ מעל השלמים.

- עבור המצב $\sigma_1(y)=1$ מתקיים $\sigma_1 \not\models q(y)$. (במקרה בו $x=1$).
- עבור $\sigma_2(y)=4$, מתקיים כי $\sigma_2 \models q(y)$.

נשים לב שאנחנו מדברים בדוגמא רק על המשתנה y מכיוון שרק משתנה זה הינו משתנה חופשי. שיטת העבודה היא הצבת הערך של y בטענה, ובדיקה האם הטענה נכונה.

<u>הגדרה</u>: **קבוצת מצבי התוכנית** הינה הקבוצה הבאה: נסמן ב-D_i את התחום של המשתנה ה-i בתוכנית. קבוצת המצבים הינה: $D_1 \times D_2 \times ... \times D_n \cup \{\perp\}$.

<u>הגדרה</u>: עבור כל טענה $q(\bar{x})$, מתקיים כי $\perp \not\models q(\bar{x})$. (במקרה והתוכנית אינה עוצרת – אף טענה אינה מסתפקת). בפרט: $\perp \not\models true$

3.3. תכונה של תוכנית - נכונות חלקית

3.3.1. הגדרה

תוכנית P היא **נכונה חלקית (partially correct)** ביחס למפרט $<q_1(\bar{x}), q_2(\bar{x})>$ אם ורק אם לכל חישוב π של התוכנית P, המתחיל ממצב התחלתי σ_0 המספק את $q_1(\bar{x})$, כלומר $\sigma_0 \vDash q_1(\bar{x})$, מתקיים שאם החישוב עוצר $\left(val(\pi(P,\sigma_0)) \neq \perp\right)$ אזי $q_2(\bar{x})$ מתקיים בסוף החישוב. $\left(val(\pi(P,\sigma_0)) \vDash q_2(\bar{x})\right)$

ניסוח מתמטי: $\sigma_0 \vDash q_1(\bar{x}) \wedge Val(\pi(P,\sigma_0)) \neq \perp \rightarrow Val(\pi(P,\sigma_0)) \vDash q_2(\bar{x})$

מסקנה: חישוב אינסופי מספק כל טענת נכונות חלקית.

שאלה: האם בניסוח המתמטי ניתן לוותר על $Val(\pi(P,\sigma_0))$? התשובה היא לא. נניח כי π אינסופי, וכי $\sigma_0 \vDash q_1$, נקבל $true \Rightarrow false$ שזהו ביטוי $false$. נכונות חלקית צריכה להיות נכונה על חישוב לא עוצר, ולכן יש צורך בחלק זה כדי להגדיר את הנכונות החלקית.

סימון לנכונות חלקית: $\{q_1\}P\{q_2\}$

סימונים קשורים נוספים:

- $\vDash \{q_1\}P\{q_2\}$ הטענה נכונה.
- $\vdash_D \{q_1\}P\{q_2\}$ הטענה יכיחה ("ניתנת להוכחה") במערכת ההוכחה D.

טענות:

- רק תוכנית שלעולם לא עוצרת מספקת את המפרט $\{true\}P\{false\}$.
- כל תוכנית מספקת את המפרט $\{false\}P\{true\}$.

כדי להסביר את הטענות נתחיל מההגדרה: אם המצב לפני התוכנית מספק את $q_1 = false$, אז אם התוכנית עוצרת, אחרי התוכנית המצב הסופי יספק את $q_2 = true$.

איזה מצב מספק את $false$?

אף מצב - ולכן המפרט $\{false\}P\{true\}$ מתקיים (באופן ריק) על כל תוכנית P.

המצב ה-"מעניין" יותר הוא המצב ההפוך: $\{true\}P\{false\}$.

אם המצב לפני התוכנית מספק את true, אז אם התוכנית עוצרת המצב יספק את $false$.

איזה מצב מספק את $true$? כל מצב. איזה מצב מספק את $false$? אף מצב.

ניתן להסיק ש-P לעולם לא עוצרת. כי אם כן, המצב היה מספק את $false$ וזה לא יתכן.

לסיכום: באופן אינטואיטיבי, המפרט $\{true\}P\{false\}$ מתאים לכל תוכנית שאינה עוצרת.

3.3.2. דוגמא למצב בעייתי

<u>מפרט</u>: "בסוף התוכנית ערכו של x כפול מערכו בהתחלה".

<u>כתיבה שגויה למפרט</u>: $< true, x = 2x >$ הביטוי אינו נכון מתמטית. לא ניתן להגיד ש-$x = 2x$ (זו לא שפת תכנות אלא מתמטיקה).

<u>שימוש במשתנה חדש</u>: ביטוי נכון ניתן לכתוב באמצעות משתנה חדש y: $< y = x, x = 2y >$.

לדוגמא: $\{y = x\}P\{x = 2y\}$

משתנה לוגי הינו משתנה שאינו מופיע בתוכנית, ולכן אינו משנה את ערכו במהלך התוכנית.

נכתוב את הדוגמא כך: $< x = X, x = 2 \cdot X >$.

מוסכמה: משתנה לוגי עבור המשתנה x יסומן ב- X , משתנה לוגי עבור y יסומן ב- Y וכו'.

הסימון $\overline{x}$ - אוסף המשתנים של התוכנית, יכלול הן את משתני התוכנית והן את המשתנים הלוגיים.

3.4. תכונה של תוכנית - נכונות מלאה

תוכנית P **נכונה באופן מלא (Total correctness)** אם ורק אם כל חישוב π שמתחיל ממצב σ_0 שמספק את $q_1(\bar{x})$ עוצר, ומצבו הסופי מספק את $q_2(\bar{x})$.

ניסוח מתמטי: $\sigma_0 \models q_1(\bar{x}) \rightarrow Val(\pi(p,\sigma_0)) \models q_2(\bar{x})$

סימון: $< q_1 > P < q_2 >$.

מתקיים: נכונות מלאה $\Leftarrow$ נכונות חלקית.

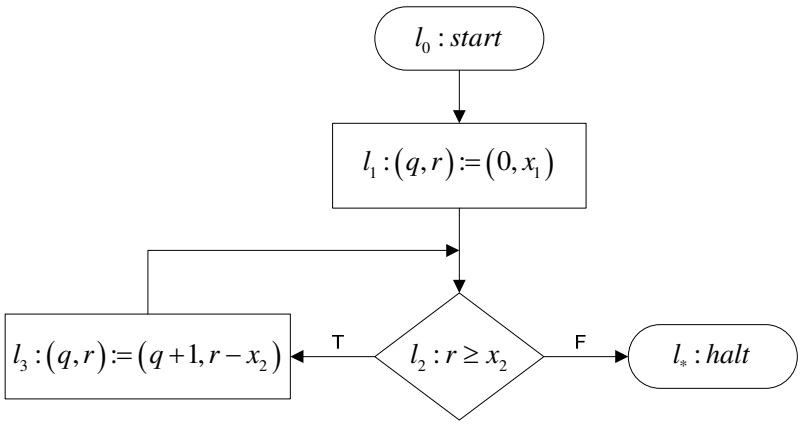
הערה: כאשר מביטים בביטוי $\langle q_1(\bar{x}), q_2(\bar{x}) \rangle$ עבור מפרט נתון, $\bar{x}$ מתייחס לערכי המשתנים בשני זמנים שונים של התוכנית (שני מצבים שונים).

4. תרשימי זרימה – Flowcharts

שפת התכנות הראשונה איתה נתעסק היא flowchart. השפה מאפשרת תיאורים גרפיים של תוכניות.

<u>דוגמת פתיחה</u>:

נביט בתרשים הזרימה הבא, המתאים לתוכנית חלוקה בשלמים עם שארית (P_{idiv}):



<u>טענה לגבי התוכנית</u>: אם x_1 שלם לא שלילי ו-x_2 שלם חיובי אז P_{idiv} ובסיומה ל-q יהיה ערך המנה השלמה בחלוקת x_1 ב-x_2 וכן r יכיל את השארית השלמה.

<u>הצורה הפורמאלית לכתוב זאת היא</u>:

$$\langle x_1 \geq 0 \wedge x_2 > 0 \wedge x_1 \in \mathbb{Z} \wedge x_1 = X_1 \wedge x_2 = X_2 \rangle P_{idiv}$$
$$\langle x_1, x_2, r, q \in \mathbb{Z} \wedge 0 \leq r \leq X_2 \wedge X_1 = qX_2 + r \rangle$$

נקודות חשובות:

- נשים לב שכאשר אנחנו מדברים על תרשימי זרימה אנו למעשה מדברים על גרפים. כל צומת בגרף מסומן בתווית. מיד נרחיב עוד על תרשימי זרימה כגרפים.
- סימון מקובל הוא סימון צומת ההתחלה ב-l_0 וצומת הסיום ב-l_*. במידה וזהו לא המצב יש לציין באופן מפורש מה הם צמתי ההתחלה והסיום.

4.1. הגדרות

<u>תיאור לא פורמלי של הפקודות</u>:

- **אתחול, start**: מציינת את נקודת ההתחלה של התוכנית. לתוכנית ישנה נקודת התחלה יחידה.
- **הצבה** $\bar{x} := \bar{e}$: יהי $\bar{x}$ וקטור משתנים ממשתני התוכנית, בלי חזרות. יהי $\bar{e}$ ווקטור באותו אורך של ביטויים מעל משתני התוכנית, וקבועים. ערכי הביטויים מחושבים לפי המצב הנוכחי, ולאחר מכן מוצבים ל- $\bar{x}$ סימולטנית. דוגמא: יהיה מצב: $\sigma(x)=2, \sigma(y)=4$, ותהא השמה $(x, y) := (x+2, y \cdot x)$, אזי מצב התוצאה הוא $\sigma'(x)=4, \sigma'(y)=8$.
- **בדיקה** $B(\bar{x})$: B ביטוי בוליאני מעל משתני התוכנית במצב נתון σ. הביטוי $\sigma \models B$ מחושב ואנחנו ממשיכים לקשת true/false לפי ערכו של החישוב.
- **עצירה, halt**: נקודת סיום של התוכנית. יכולה להיות יותר מנקודת עצירה אחת.

<u>הגדרה מדוייקת של תרשים זרימה (כגרף)</u>:

תכנית P בשפת תרשימי הזרימה PLF הינה גרף סופי מכוון שבצמתיו פקודות. לכל צומת מותאמת תווית המופרדת מהפקודה בנקודותיים. בנוסף על הגרף לספק את המגבלות הבאות:

1. לצומת אתחול אין אב ויש בן אחד בדיוק.
2. לצומת עצירה אין בנים.
3. לצומת הצבה בן אחד בדיוק.
4. לצומת בדיקה שני בנים בדיוק: אחד מסומן ב-T והשני מסומן ב-F.
5. צומת האתחול הוא יחיד.
6. כל צומת נמצא על מסלול מצומת אתחול לצומת עצירה.
7. לצמתים שונים תוויות שונות.

<u>הגדרת מסלולים</u>:

<u>הגדרה</u>: בהינתן תוכנית תרשימי זרימה, נגדיר בשם **מסלול** סדרת מצבים עוקבים בגרף תרשימי הזרימה.

<u>הגדרה</u>: **מסלול מלא** הוא מסלול המתחיל ב-start ומסתיים ב-halt.

<u>הגדרה</u>: **מסלול מקסימאלי** הוא מסלול מלא או אינסופי.

4.2. סמנטיקה

המטרה שלנו: הגדרת חישוב של תוכנית. לצורך כך נגדיר את מושג הקונפיגורציה ואת רלציית המעברים:

קונפיגורציה – זוג $C=\langle l,\sigma\rangle$ כאשר l תווית המציינת את הצומת בתוכנית בו נמצא החישוב. σ הינו מצב (השמה למשתנים).
משמעות קונפיגורציה: הפקודה שנמצאת בצומת עם התווית l מתבצעת ממצב σ.

רלציית המעברים $\rightarrow$ בין קונפיגורציות

$\langle l,\sigma\rangle \rightarrow \langle l',\sigma'\rangle$ אמ"מ קיימת קשת מ-l ל-l' בתוכנית ובנוסף מתקיים אחד מהתנאים הבאים:

1. ב-l יש פונקצית אתחול וגם $\sigma'=\sigma$.
2. ב-l פקודת הצבה מהצורה $\overline{x}:=\overline{e}$ ומתקיים $\sigma'=\sigma\left[\overline{x}\leftarrow\sigma\left(\overline{e}\right)\right]$.

   $\sigma'=\sigma\left[\overline{x}\leftarrow\sigma\left(\overline{e}\right)\right]$ הינה **הצבה חלקית**, בה מוחלפים ב-σ כל ערכי המשתנים המצוינים בווקטור.
3. ב-l פונקצית בדיקה $B\left(\overline{x}\right)$, מתקיים $\sigma\left(B\right)=true$ וגם קיימת קשת מ-l ל-l' בתוכנית המסומנת T, וכן $\sigma'=\sigma$ <u>או</u> $\sigma\left(B\right)=false$ וגם קיימת קשת מ-l ל-l' בתוכנית המסומנת F, וכן $\sigma'=\sigma$.

בנוסף: כאשר ב-l יש halt אין מעבר לאף קונפיגורציה אחרת.

הארה: רלצית המעברים לוקחת את גרף תרשימי הזרימה ונותנת לנו את היכולת להשתמש בו ככלי לניתוח התוכנית. היא מאפשרת לנו לעשות בו צעדים רק בהתאם לתוכנית עצמה.

הגדרת חישוב:

חישוב של תוכנית P מקונפיגורציה C מסומן $\pi(P,C)$ הינו סידרה מקסימלית של קונפיגורציות C_i כך ש- $C_0 = C$ ולכל i מתקיים $C_i \to C_{i+1}$.

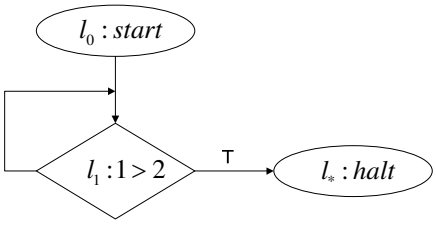
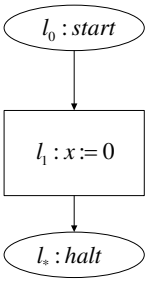
חישוב סופי תמיד יסתיים בקונפיגורציה עוצרת.

קונפיגורציה עוצרת: $C = \langle l, \sigma \rangle$ היא קונפיגורציה עוצרת אם l מציינת צומת עצירה בתוכנית.

קונפיגורציה התחלתית: $C = \langle l, \sigma \rangle$ היא קונפיגורציה התחלתית אם l מציינת את צומת האתחול בתוכנית.

עבור חישוב סופי π המסתיים בקונפיגורציה (עוצרת) $\langle l, \sigma_k \rangle$ מגדירים את $Val(\sigma_k)$ להיות σ_k.

בהינתן חישוב בתוכנית יש מסלול יחיד שמתאים לו. בהינתן מסלול יכול להיות שאין חישוב שמתאים לו, או לחילופין יכולים להיות אין סוף חישובים מתאימים.

אין חישוב מתאים	אינסוף חישובים מתאימים
	
$l_0 \to l_1 \to l_*$	$l_0 \to l_1 \to l_*$

4.3. *למת ההפרדה*

לכל תוכנית P ומפרט $< q_1, q_2 >$ מתקיים $< q_1 > P < q_2 >$ (נכונות מלאה) אם ורק אם $\models < q_1 > P < true >$ (כל החישובים שמתחילים ב- q_1 עוצרים) וגם $\models \{q_1\} P \{q_2\}$ (נכונות חלקית).

הלמה מאפשרת לנו לפרק הוכחת נכונות מלאה ל-2 הוכחות: 1. עצירה לכל חישוב. 2. נכונות חלקית.

4.4. שיטת / כללי ההוכחה

כדי להוכיח נכונות של תוכנית בשפת תרשימי הזרימה נציג כללים, שבמידה והם יתקיימו נאמר כי התוכנית נכונה. נתחיל במקרה הפשוט בו בתוכנית אין מעגלים (לולאות). במקרה זה מובטחת עצירה של התוכנית, ולכן נכונות חלקית ונכונות מלאה נובעות זו מזו. לאחר מכן נרחיב את הכללים גם למצב בו התוכניות כוללת לולאה אחת או יותר.

<u>בשלב הראשון: המטרה שלנו בהנתן תרשים זרימה</u>: נרצה להראות שכל חישוב מלא (חסר מעגלים) שמספק את $q_1(\bar{x})$ מספק בסופו את $q_2(\bar{x})$.

4.4.1. הגדרות – טרנספורמצית המצבים, תנאי הישיגות

נתון מסלול: $\tau = l_{i0}, l_{i1}, ..., l_{in}$. נרצה לחשב:

1. בהינתן מצב התוכנית ההתחלתי נרצה לדעת את מצב התוכנית הסופי.
2. בהינתן מצב התוכנית ההתחלתי נרצה לדעת האם המסלול עביר.

בהינתן מסלול $\tau = l_{i0}, l_{i1}, ..., l_{in}$ נגדיר:

1. **טרנספורמציית המצבים (state transformer)** $\boldsymbol{T_\tau(\bar{x})}$ על מסלול τ שמתחיל ממצב $\bar{x}$. $T_\tau(\bar{x})$ מתאר את המצב הסופי של התוכנית בעזרת המשתנים של תחילת התוכנית.
2. **תנאי הישיגות (reachability condition)** $\boldsymbol{R_\tau(\bar{x})}$ - ביטוי שערכו $true$ אם ורק אם τ הוא מסלול עביר עבור המצב ההתחלתי $\bar{x}$.

<u>משפט</u>: בהינתן מסלול סופי $\tau = l_{i0}, l_{i1}, ..., l_{ik}$ שחישובו מתחיל ממצב σ_0:

1. המסלול עביר (כלומר l_{ik} ישיג מ-l_{i0} לאורך המסלול τ) אם ורק אם $\sigma_0 \models R_\tau(\bar{x})$.
2. אם המסלול מסתיים ב-l_{ik} במצב σ_k אזי מתקיים ש-$\sigma_k = \sigma_0\left[\bar{x} \leftarrow T_\tau(\bar{x})\right]$.

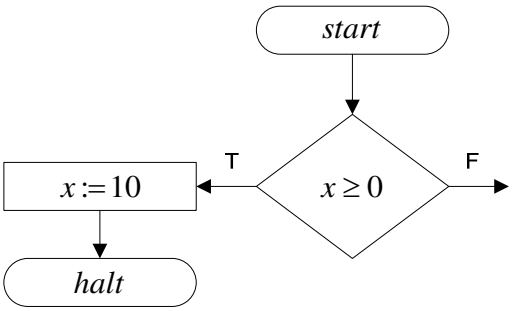
<u>שימוש</u>: בהנתן מצב σ שנותן ערכים מסויימים למשתנים, ניתן לחשב את ערכם בסוף המסלול אם ביצענו אותו ממצב σ.

<u>דוגמא</u>: יהא $T_\tau(x,y) = (x+y, y)$ וגם $\sigma_0(x,y) = (2,4)$ אז $\sigma_k(x,y) = (6,4)$.

4.4.2. דוגמאות לעבירות

דוגמא 1

המסלול הבא יהיה עביר רק אם במצב ההתחלתי $x \geq 0$:



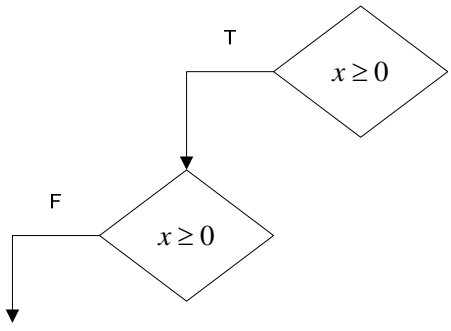
שימו לב: יכולים להיות מסלולים אחרים בגרף שהם עבירים. המסלול הספציפי הזה עביר רק בהתקיים התנאי.

שאלה: האם חישוב לאורך המסלול המתואר סותר את המפרט $\langle x < 0, x = 0 \rangle$?

לא, כי המסלול אינו עביר, ולא ניקח אותו בחשבון בהוכחה על מנת שלא נקלקל שלמות.

דוגמא 2

המסלול הבא אינו עביר כלל (כמסלול, לא כתוכנית):



4.4.3. הגדרה אינדוקטיבית של $T_\tau(\bar{x})$ ושל $R_\tau(\bar{x})$

יהי מסלול $\tau=(l_{i0},...,l_{ik})$. נחשב סדרה של ביטויים $T_\tau^j(\bar{x}),R_\tau^j(\bar{x})$ על המסלול מהסוף להתחלה, כלומר נתחיל ב-$j=k$ ונסיים ב-$j=0$.

- $T_\tau^j(\bar{x})$ מציין את טרנספורמצית המצבים על המסלול $(l_{ij},...,l_{ik})$
- $R_\tau^j(\bar{x})$ הינו תנאי הישיבות המסלול $(l_{ij},...,l_{ik})$

הערה: צריך לשים לב שכאשר אנחנו מסתכלים על חישוב מסלול, החישוב $(l_{ij},...,l_{ik})$ אינו כולל את ביצוע הפקודה הרשומה ב-l_{ik}.

<u>בסיס ההגדרה</u>: $T_\tau^k(\bar{x})=(\bar{x})$, $R_\tau^k(\bar{x})=true$

הסבר: בשלב זה המסלול הוא l_{ik} ולכן:

א. המשתנים בסיום זהים למשתנים בהתחלה.

ב. התנאי למעבר הוא $true$.

<u>צעד האינדוקציה</u>: בהינתן T_τ^{k+1}, R_τ^{k+1} נרצה לחשב T_τ^k, R_τ^k.

א. הצבה: $\bar{x}:=\bar{e}$:

$$R_\tau^k(\bar{x})=R_\tau^{k+1}[\bar{x}\leftarrow\bar{e}]$$
$$T_\tau^k(\bar{x})=T_\tau^{k+1}[\bar{x}\leftarrow\bar{e}]$$

ב. תנאי : $b(\bar{x})$: אם זה הצד החיובי של התנאי:

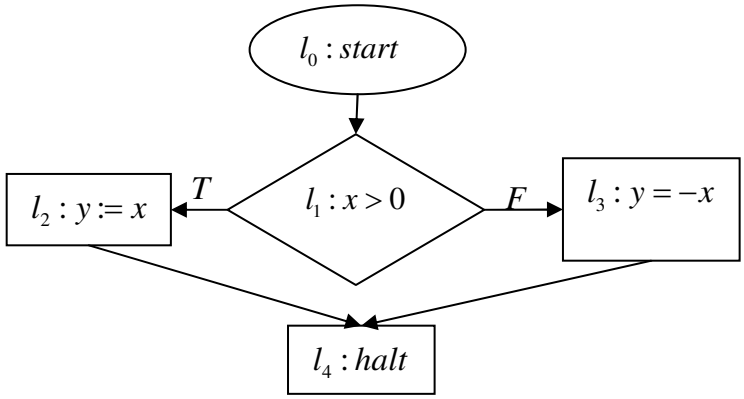
$$R_\tau^k(\bar{x})=R_\tau^{k+1}(\bar{x})\wedge b(\bar{x})$$
$$T_\tau^k(\bar{x})=T_\tau^{k+1}(\bar{x})$$

אם זה הצד השלילי של התנאי:

$$R_\tau^k(\bar{x})=R_\tau^{k+1}(\bar{x})\wedge\neg b(\bar{x})$$
$$T_\tau^k(\bar{x})=T_\tau^{k+1}(\bar{x})$$

<u>אופן החישוב</u>: נחשב את $T_\tau(\bar{x})$ ואת $R_\tau(\bar{x})$ ע"י חישוב "אחורנית": בהינתן מסלול $\tau=l_{i0},l_{i1},...,l_{ik}$ נתחיל עם $T_\tau^k(\bar{x})$ ונסיים עם $T_\tau^0(\bar{x})$. יתקיים: $T_\tau(\bar{x})=T_\tau^0(\bar{x})$.

<u>דוגמא</u>: יהא תרשים הזרימה הבא:



הביטו כעת בטבלה. שימו לב כשאתם קוראים את הטבלה הבאה שבנייתה החלה מ-l_4 ונמשכה כלפי מעלה אל הצמתים הראשונות. באפור – המסקנות אליהן אנו מגיעים.

מסלול ימני $\tau = l_0, l_1, l_3, l_4$		מסלול שמאלי $\tau = l_0, l_1, l_2, l_4$	
$l_0 : start$	$T_\tau^0(\bar{x}) = (x, -x)$	$l_0 : start$	$T_\tau^0(\bar{x}) = (x, x)$
$l_1 : x > 0$	$T_\tau^1(\bar{x}) = (x, -x)$	$l_1 : x > 0$	$T_\tau^1(\bar{x}) = (x, x)$
$l_3 : y := -x$	$T_\tau^2(\bar{x}) = (x, -x)$	$l_2 : y := x$	$T_\tau^2(\bar{x}) = (x, x)$
$l_4 : halt$	$T_\tau^3(\bar{x}) = (x, y)$	$l_4 : halt$	$T_\tau^3(\bar{x}) = (x, y)$
$T_\tau(\bar{x}) = T_\tau^0(\bar{x}) = (x, -x)$		$T_\tau(\bar{x}) = T_\tau^0(\bar{x}) = (x, x)$	
$l_0 : start$	$R_\tau^0(x, y) = x \le 0$	$l_0 : start$	$R_\tau^0(x, y) = x > 0$
$l_1 : x > 0$	$R_\tau^1(x, y) = x \le 0$	$l_1 : x > 0$	$R_\tau^1(x, y) = x > 0$
$l_3 : y := -x$	$R_\tau^2(x, y) = True[y \leftarrow -x] = True$	$l_2 : y := x$	$R_\tau^2(x, y) = True[y \leftarrow x] = True$
$l_4 : halt$	$R_\tau^3(x, y) = True$	$l_4 : halt$	$R_\tau^3(x, y) = True$
$R_\tau(\bar{x}) = R_\tau^0(x, y) = x \le 0$		$R_\tau(\bar{x}) = R_\tau^0(x, y) = x > 0$	

4.4.4. כלל להוכחת נכונות עבור תוכניות ללא מעגלים

כדי להוכיח נכונות של תוכנית בשפת תרשימי הזרימה נציג כלל, שבמידה והוא מתקיים נאמר כי התוכנית נכונה. נתחיל כאמור במקרה הפשוט בו בתוכנית אין מעגלים (לולאות):

תוכנית ללא מעגלים P נכונה ביחס למפרט $< q_1, q_2 >$ (כלומר $< q_1 > P < q_2 >$) אם ורק אם לכל מסלול מלא τ של P מתקיים: $q_1(\bar{x}) \wedge R_\tau(\bar{x}) \rightarrow q_2(T_\tau(\bar{x}))$. (אם תנאי ההתחלה מתקיים והמסלול עביר אז תנאי הסיום מתקיים על הטרנספורמר של משתני ההתחלה).

אופן הפעולה:

1. לכל מסלול מ-start ל-halt נחשב $T_\tau(\bar{x})$ ו-$R_\tau(\bar{x})$.
2. לכל מסלול כנ"ל נוכיח: $\forall \bar{x}\left[q_1(\bar{x}) \wedge R_\tau(\bar{x}) \rightarrow q_2(T_\tau(\bar{x})) \right]$

4.4.5. כלל ההוכחה F (Floyd) לתוכניות תרשים זרימה עם חוגים

כלל ההוכחה של Floyd משמש אותנו להוכחת $\{q_1\} P \{q_2\}$. נשתמש בכלל הבא:

1. נבחר "נקודות חתך" בתוכנית בצורה הבאה:

   א. נקודת ההתחלה l_0 $(start)$ תהיה נקודת חתך.

   ב. נקודת הסיום l_{halt} $(halt)$ תהיה נקודת חתך.

   ג. כל מעגל בגרף התוכנית יכיל לפחות נקודת חתך אחת.

2. לכל נקודת חתך l, נמצא טענה אינדוקטיבית (invariant שמורה) $I_l(\bar{x})$. כמו כן, תמיד נבחר: $I_{start}(\bar{x}) = q_1(\bar{x})$, $I_{halt}(\bar{x}) = q_2(\bar{x})$.

3. לכל מסלול $\alpha = (l_i, l_j)$ שלא עובר דרך נקודות חתך אחרות נוכיח את תנאי הנכונות:

   $I_{li}(\bar{x}) \wedge R_\alpha(\bar{x}) \rightarrow I_{lj}\left[\bar{x} \leftarrow T_\alpha(\bar{x})\right]$.

   (אם מתקיים התנאי הראשון על המשתנים בנקודה הראשונה וגם המסלול עביר אז יתקיים התנאי השני על הטרנספורמר של המסלול על המשתנים).

אם הפעלנו את הכלל בהצלחה נסמן: $\vdash_F \{q_1\} P \{q_2\}$

הערות:

- כאשר מדובר על תוכנית עם מעגלים נכונות חלקית ונכונות מלאה לא מתלכדים. כלל Floyd מוכיח נכונות חלקית בלבד.
- הרעיון מאחורי בחירת נקודות החתך הוא לדאוג שכל מסלול בין 2 נקודות חתך יהיה סופי.

4.4.5.1. יתרונות וחסרונות שיטת Floyd

חסרונות של שיטת Floyd:

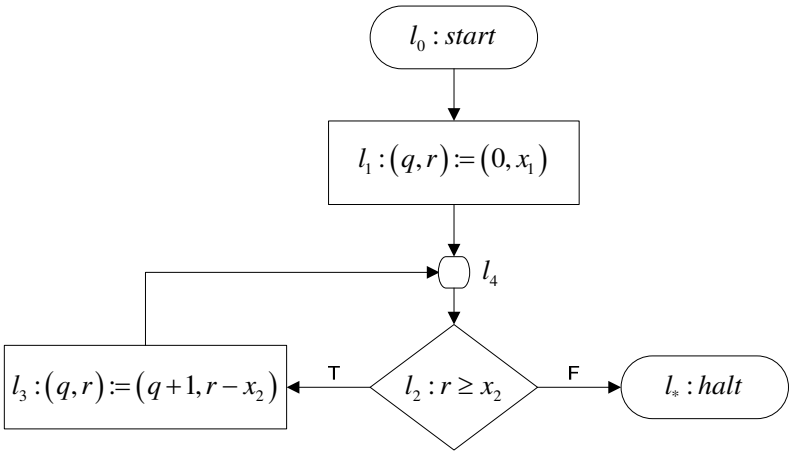
- שיטת Floyd מורכבת ויקרה. היא דורשת כמות גדולה של חישובים.
- השיטה מערבת את המשתמש הבוחר את האינווריאנטות. זהו חסרון מבחינתנו מכיוון שמטרת העל שלנו היא למצוא שיטות אוטומטיות ככל האפשר.

יתרונות שיטה Floyd:

- לקחנו 2 טענות מסדר ראשון q_1, q_2 שאיננו יכולים להגיד הרבה על הקשר ביניהן, והפכנו אותן לאוסף טענות בלוגיקה רגילה עליהן ניתן להשתמש בשיטות המוכרות לנו מלוגיקה.
- השיטה נאותה ושלמה. החשיבות של עובדה זו היא שניתן ליצור במחשב תוכניות שפותרות מקרים פרטיים של שפת תרשימי הזרימה. נציג מיד הוכחה לנאותות השיטה.

4.4.5.2. דוגמא להפעלת Floyd

נציג שוב את התוכנית המחשבת את המנה והשארית של חלוקת x_1 ב-x_2. בשרטוט הצומת l_4 אינו עושה דבר וקיים רק לצרכי פישוט ההוכחה. ניתן להוכיח את התוכנית גם בלעדיו.



הפעלת Floyd:

1. נגדיר את $q_1(\bar{x})$ ואת $q_2(\bar{x})$:
   a. כל ערכי המשתנים בתוכנית הינם מעל $\mathbb{Z}$.
   b. $q_1(x_1,x_2,q,r,X_1,X_2)=(x_1=X_1\wedge x_2=X_2\wedge x_1\geq 0\wedge x_2>0)$
   c. $q_2(x_1,x_2,q,r,X_1,X_2)=(X_1=qX_2+r\wedge 0\leq r<X_2)$
2. נבחר טענות עבור נקודות החתך:
   a. נצמיד לצומת l_0 את הטענה $q_1(\bar{x})$ ונצמיד ל-l_* את הטענה $q_2(\bar{x})$.
   b. נציע טענה: $I_{l_4}=(X_1=qX_2+r\wedge r\geq 0\wedge x_1=X_1\wedge x_2=X_2)$. אם האינוריאנטה מתקיימת ובחרנו לצאת מהלולאה, הרי שהטענה תבטיח את q_2 בסיום, ולכן זו נראית טענה מוצלחת.
3. צריך להוכיח את תנאי הנכונות לכל המסלולים הסופיים. המסלולים הקיימים: $l_0l_4,\ l_4l_4,\ l_4l_5$.
   a. נראה את תנאי הנכונות עבור l_4l_4.

צ"ל: $I_{l_4}(\bar{x})\wedge R_{l_4l_4}(\bar{x})\rightarrow I_{l_4}\left[\bar{x}\leftarrow T_{l_4l_4}(\bar{x})\right]$

$$R_{l_4l_4}(\bar{x})=(r\geq x_2)$$
$$T_{l_4l_4}(x_1,x_2,q,r,X_1,X_2)=(x_1,x_2,q+1,r-x_2,X_1,X_2)$$

נציב בתנאי:

$$(X_1=qX_2+r\wedge r\geq 0\wedge x_1=X_1\wedge x_2=X_2)\wedge(r\geq x_2)\rightarrow$$
$$(X_1=(q+1)X_2+(r-X_2)\wedge r-X_2\geq 0\wedge x_1=X_1\wedge x_2=X_2)$$

קיבלנו נוסחה בלוגיקה. לצורך הוכחת אימות תוכנה מספיק לנו לראות שהנוסחה נכונה מבחינה מתמטית – אין צורך להכנס להוכחה מורכבת בלוגיקה. ממתמטיקה טריויאלית ניתן לראות את נכונות הביטוי.

   b. תנאי נכונות עבור l_0l_4 ו-l_4l_5 יחושבו בצורה דומה.
   c. אחרי ההוכחות נאמר כי $\vdash_F\{q_1\}P\{q_2\}$.

4.4.5.3. נאותות ושלמות שיטת Floyd

משפט הנאותות של F – (מערכת ההוכחה של פלויד):

אם מתקיים $\vdash_F \{q_1\}P\{q_2\}$ אז $\models \{q_1\}P\{q_2\}$. במילים: אם ניתן להוכיח את הטענה במערכת ההוכחה Floyd, אז הטענה נכונה.

על מנת להוכיח את המשפט, נציג למת עזר:

למה: אם $\vdash_F \{q_1\}P\{q_2\}$ אזי לכל חישוב π של P, שמתחיל ממצב σ_0 שמקיים: $\sigma_0 \models q_1(\bar{x})$, אם החישוב מגיע לנקודת חתך l' במצב σ' אזי $\sigma' \models I_{l'}(\bar{x})$.

כלומר: אם ניתן במערכת ההוכחה להוכיח את $\{q_1\}P\{q_2\}$ אז כל חישוב שמתחיל ממצב σ_0 ומספק את תנאי ההתחלה, בהגיעו לנקודת החתך במצב כלשהו, מצב זה מספק את התנאי של נקודת החתך.

הוכחת הלמה תעשה באינדוקציה על מספר נקודות החתך בהן עבר החישוב:

בסיס: חישוב שעבר בנקודת חתך אחת: $l_0 = start, \sigma' = \sigma_0, I_{l'} = q_1$. מתקיים $\sigma' \models I_{l'}(\bar{x})$ כי נתון שמתקיים $\sigma_0 \models q_1(\bar{x})$.

צעד: ניקח מסלול בו $i+1$ נקודות חתך. הנחת האינדוקציה היא: $\sigma_i \models I_{l_i}(\bar{x})$ וכן מתקיים $\sigma_i \models R_{i\to i+1}(\bar{x})$ (המסלול עביר). צריך להוכיח כי מתקיים: $\sigma_{i+1} \models I_{l_{i+1}}(\bar{x})$.

קצת לוגיקה: בעצם מהנתונים מתקיימת טענת הנכונות:

$$\sigma_i \models \left(I_{l_i}(\bar{x}) \wedge R_{i\to i+1}(\bar{x}) = I_{l_{i+1}}\left[T_{l_i\to l_{i+1}}(\bar{x})\right]\right)$$

מטענת הנכונות מתקיים $\sigma_i \models I_{l_{i+1}}\left[T_{l_i\to l_{i+1}}(\bar{x})\right]$. זהו אינו הביטוי המדוייק אותו אנו צריכים להוכיח, לכן נשתמש במשפט מלוגיקה:

תזכורת: מתקיים תמיד $S\left[x \leftarrow s(e)\right] \models \varphi \Leftrightarrow s \models \varphi\left[x \leftarrow e\right]$

- $S\left[x \leftarrow s(e)\right]$ זוהי השמה חדשה (אצלנו = מצב חדש) שזהה ל-s חוץ מהערך של x שהוא $s(e)$, כאשר $\varphi\left[x \leftarrow e\right]$ זו נוסחה בה החלפנו את x בביטוי e.

על סמך המשפט בלוגיקה נסיק: $\sigma_i\left[\overline{x} \leftarrow T_{l_{i+1}}\left(\overline{x}\right)\right] \vDash I_{l_{i+1}}$.

העברנו שינויים בנוסחה לשינויים במצב. מתקיים $\sigma_{i+1} \vDash I_{l_{i+1}}\left(\overline{x}\right)$ כנדרש.

<u>הוכחת משפט הנאותות</u>:

נניח כי מתקיים $\vdash_F \left\{q_1\right\}P\left\{q_2\right\}$, אזי המצבים האפשריים:

- אם חישוב π של תוכנית אינו עוצר, אז אין דרישה עבורו בנכונות חלקית.
- חישוב π שמתחיל מ-start ממצב σ כך ש-$\sigma \vDash q_1\left(\overline{x}\right)$ עוצר. על סמך הלמה ב-halt מתקיימת הטענה $q_2\left(\overline{x}\right)$.

משפט השלמות עבור F: אם מתקיים $\vDash \left\{q_1\right\}P\left\{q_2\right\}$ אז $\vdash_F \left\{q_1\right\}P\left\{q_2\right\}$. כלומר ניתן למצוא נקודות חתך וטענות אינווריאנטיות שמספקות את תנאי הנכונות.

4.4.6. הוכחת עצירה של תוכניות בשפת PLF

נרצה כעת להראות כי תוכנית P נתונה עוצרת, כלומר: כל חישוב של התוכנית הוא סופי. הרעיון יהיה הצמדה של סדרה סופית יורדת אל כל חישוב. חשוב לשים לב כי מספיק להראות שכל לולאה היא סופית, ואין צורך להוכיח בבת אחת על כל התוכנית.
בכל תוכנית שנבדוק נחפש "מידה יורדת" כדי לחשוב על הפתרון. "מידה יורדת טובה" היא מידה שיורדת ממש בכל ביצוע של החוג.

לצורך המימוש נגדיר מושג חדש, קבוצה מבוססת היטב, בו נשתמש בכלל ההוכחה. ניתן להוכיח כי שימוש בקבוצת הטבעיים בלבד יספיק לנו לצרכי ההוכחה, אך לעתים נוח לבחור קבוצות אחרות.

4.4.6.1. קבוצה מבוססת היטב

<u>הגדרה</u>: **קבוצה מבוססת היטב (Well founded set)** היא קבוצה עם יחס (יכול להיות חלקי) $(W,>)$ שבה לא קיימת סדרה יורדת אינסופית. כלומר לא קיימת סדרה אינסופית $w_0, w_1, w_2, ... \in W$ כך שמתקיים $w_0 > w_1 > w_2 > w_3 ...$.

תזכורת:

- **יחס מלא**: יחס בו כל שני איברים ביחס ניתנים להשוואה. דוגמאות: טבעיים, ממשיים, רציונליים עם הסדר הרגיל.
- **יחס חלקי**: לא כל שני איברים ניתנים להשוואה. היחס מקיים טרנזיטיביות.

<u>דוגמאות לקבוצות מבוססות היטב</u>:

1. $(\mathbb{N},>)$ כאשר המספרים הם המספרים בסדר הרגיל.
2. בהינתן A קבוצה סופית ויחס '$>$' הכלה, הקבוצה $(p(A),\supset)$ הינה קבוצה מבוססת היטב.
3. $(N\times N,>)$ - $(n',m')>(n,m) \Leftrightarrow (n'>m)\vee(n'=n\wedge m'>m)$ (יחס סדר לקסיקוגרפי) גם קבוצה מבוססת היטב.

   לאחר שבחרנו זוג התחלתי (n_0,m_0) נוכל אומנם לרדת ל (n_0-1,m_1) כך ש m_1 גדול כרצוננו, אולם הוא עדין סופי, ולכן אפשר לשים לכל היותר m_1 איברים עד שנאלץ לרדת ל n_0-2 וכך בסופו של דבר, הסדרה חייבת להיות סופית.

<u>דוגמאות לקבוצות שאינן מבוססות היטב</u>:

1. $(\mathbb{R}^+,>)$ - אינה קבוצה מבוססת היטב – למשל ניתן לבחור בסדרה: $r_1, \frac{r_1}{2}, \frac{r_1}{4}, ...$
2. $(p(A),\supset)$ עבור A אינסופית אינה קבוצה מבוססת היטב – למשל אם $A=\mathbb{N}$ נבחר $A_0=\mathbb{N}, A_{i>0}=A_{i-1}\setminus\{i\}$

4.4.6.2. כלל F* להוכחת עצירה

מטרת כלל זה היא להוכיח $\langle q_1 \rangle P \langle true \rangle$. דרך הפעולה:

1. בחרו קבוצה מבוססת היטב W עם סדר חלקי $(W, <)$.
2. בחרו נקודות חתך (באופן זהה לזה שבחרנו נקודות חתך ב-F).
3. לכל נקודת חתך, התאימו טענה אינדוקטיבית פרמטרית, $I(\bar{x}, w)$ כאשר w משתנה שתחומו W ו- $\bar{x}$ הם משתני התוכנית.
4. הוכיחו את תנאי הנכונות הבאים:
   א. $(Init) \quad \forall \bar{x}\left(q_1(\bar{x})\right) \rightarrow \exists_w I_{Start}(\bar{x}, w)$
   ב. עבור כל מסלול $\alpha = (l, l')$ ללא נקודות חתך באמצע

$$(Dec) \quad \forall \bar{x} \forall w I_l(\bar{x}, w) \wedge R_\alpha(\bar{x}) \rightarrow \exists \underbrace{w'}_{\in W}\left[w' < w \wedge I_{l'}\left(T_a(\bar{x}), w'\right)\right]$$

בצורה כזאת נוכיח $\langle q_1 \rangle P \langle true \rangle$ - אם תנאי ההתחלה מתקיים, אז התוכנית עוצרת.

באופן אינטואיטיבי לא פורמאלי, נשים לב כי מכיוון שמתקיים שבין כל שתי נקודות חתך ערכו של w קטן, וכן לא קיימת סדרה יורדת אינסופית ב- W, הרי שחייב להיות סוף לתהליך ומכאן התוכנית עוצרת.

אם היינו רוצים גם להוכיח נכונות "באותה הזדמנות" היינו צריכים להוכיח:
$\forall \bar{x} \forall w I_{halt}(\bar{x}, w) \rightarrow q_2(\bar{x})$ (האינווריאנטה של נקודת העצירה מספקת את תנאי הסיום).

<u>משפט (ללא הוכחה)</u>: הכלל F* הוא נאות ושלם.
נאותות F*: אם $\vdash_{F^*} \langle q_1 \rangle P \langle true \rangle$ אז $\models \langle q_1 \rangle P \langle true \rangle$

<u>למה עבור F*</u>: אם $\vdash_{F^*} \langle q_1 \rangle P \langle True \rangle$ אזי לכל חישוב π של P שמתחיל מ- $Start$ במצב σ כך ש- $\sigma \models q_1(\bar{x})$, אם החישוב מגיע לנקודה l' במצב σ' אז קיים w' כך ש- $\sigma' \models I_{l'}(\bar{x}, w')$.

אינטואציה להוכחת שלמות F*: אם החישוב סופי, נתאים לכל שלב את מספר הצעדים שנשארו עד העצירה.

שאלה לדוגמא

נתונה תוכנית P בשפת תרשימי הזרימה PLF ונתונות טענות q_1, q_2 מסדר ראשון מעל משתני התוכנית. נאמר ש- $P \models q_1 \rightarrow GLOBALLY q_2$ אם לכל חישוב של P, אם הוא מתחיל ממצב שמספק את q_1 אז כל מצב על מסלול החישוב מספק את q_2.

הציעו כלל נאות ושלם להוכחת $P \models q_1 \rightarrow GLOBALLY q_2$. נמקו את נאותות ושלמות הכלל.

<u>פתרון:</u>

1. בחר את כל צמתי התוכנית כנקודות חתך.
2. לכל תווית הצמד טענה אינדוקטיבית $I_l(\bar{x})$ כאשר $I_{l_0} = q_1(\bar{x})$ ו $I_{l_*}(\bar{x}) = true$
3. הוכח: $\forall \bar{x}\left[q_1(\bar{x}) \rightarrow q_2(\bar{x})\right]$
4. לכל מסלול בסיסי $\tau = l, l'$ הוכח: $\forall \bar{x}\left[I_l \wedge R_{l,l'} \wedge q_2 \rightarrow I_{l'}(\bar{x}) \wedge q_2\left[\bar{x} \leftarrow T_{l,l'}(\bar{x})\right]\right]$

5. לוגיקה של תוכניות

5.1. הגישה המודולרית של *Hoare*

בתחילת המסמך הצגנו כי עבור מערכת הוכחה צריך:

- שפת תכנות עם סמנטיקה פורמלית.
- שפת מפרט עם סמנטיקה פורמלית.
- שיטת/כללי הוכחה.

כעת נציג שפת תכנות נוספת שאינה PLF.

- <u>שפת המפרט</u>: נבחר בשפת מפרט זהה לזו שהשתמשנו בה עד כה.
- <u>שפת התכנות</u>: שפת התכנות תהיה שפה דמויית C בשם **while-program**.
- <u>שיטת / כללי הוכחה</u>:
  - השם "הגישה המודולרית" נובע מכיוון שיהיה ניתן להוכיח טענה על חלקים מהתוכנית, ולאחר מכן לחבר טענות לטענות גדולות יותר.
  - נוכיח את הטענות באמצעות אקסיומות וכללי היסק, ולכן נציג בעצם חישובים שהם "לוגיקה מעל תוכניות". (בניגוד לדוגמאות שהצגנו ב-PLF, בהם הצגנו שיטות להוכחה, אבל לא היתה הוכחה פורמלית מסודרת על שלבי התוכנית).

5.1.1. שפת תוכניות while

נגדיר את השפה על ידי כלל הגזירה הבא:

$$S :: \quad x := e \mid skip \mid S_1; S_2 \mid \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi} \mid \text{while } B \text{ do } S \text{ od}$$

<u>הפעולות</u>:

1. $x := e$ - הצבה.
2. $skip$ - פקודה שאינה עושה דבר. (שימושי למשל בתנאי ה-if הדורשים לפי התחביר else, ולא בכל מקרה נרצה לבצע פעולה במקרה כזה).
3. $S_1; S_2$ - שרשור של שתי פקודות אחת אחרי השניה.
4. $\text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}$ - משפט תנאי. כל חלקי המשפט הם חובה.
5. $\text{while } B \text{ do } S \text{ od}$ - לולאה.

כאשר:

- B הינו תנאי בוליאני מעל משתני התוכנית.
- $skip, x := e$ הינן פעולות אטומיות. שאר הפעולות הינן פעולות מורכבות.

מכיוון ששפת התכנות מודולרית אנחנו יכולים להשתמש במערכת הוכחה מודולרית.

סמנטיקה אופרטיבית של תוכניות while:

- σ הינו **מצב התוכנית**, פונקציה בין משתני התוכנית לערכיהם.
- **קונפיגורציה** $C = \langle S_i, \sigma \rangle$ כאשר S_i היא תוכנית while, ומתארת את המשך החישוב (בכל רגע S_i היא יתרת התוכנית לביצוע).
- עבור הגדרת הסמנטיקה נגדיר **תוכנית ריקה E** המקיימת: $S; E = E; S = S$. התוכנית E היא תוכנית שאינה עושה דבר – אם נשרשר אותה לפני או אחרי תוכנית אחרת כלשהי, נקבל את אותה תוכנית.
- **קונפיגורציה עוצרת** היא קונפיגורציה מהצורה $\langle E, \sigma \rangle$.

5.1.2. רלציית המעברים $\rightarrow$

רלציית המעברים $\rightarrow$ הינה הרלציה הקטנה ביותר המקיימת את ההגדרה האינדוקטיבית הבאה:

1. $\langle x := e, \sigma \rangle \rightarrow \langle E, \sigma[x \leftarrow \sigma(e)] \rangle$
2. $\langle skip, \sigma \rangle \rightarrow \langle E, \sigma \rangle$
3. לכל תוכנית T: אם $\langle S, \sigma \rangle \rightarrow \langle S', \sigma' \rangle$ אז $\langle S; T, \sigma \rangle \rightarrow \langle S'; T, \sigma' \rangle$.
   a. החישוב של פקודה אינו תלוי בפקודות בהמשך.
   b. חישוב של $S;T$ הינו חישוב של S במלואו ולאחר מכן חישוב של T.
4. $(\sigma \models B) \Rightarrow \langle \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}; \sigma \rangle \rightarrow \langle S_1, \sigma \rangle$
   $else \Rightarrow \langle \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}; \sigma \rangle \rightarrow \langle S_2, \sigma \rangle$
5. אם $\sigma \not\models B$ אזי $\langle \text{while } B \text{ do } S \text{ od}, \sigma \rangle \rightarrow \langle E, \sigma \rangle$.

אחרת: $\langle \text{while } B \text{ do } S \text{ od}, \sigma \rangle \rightarrow \langle S; \text{while } B \text{ do } S \text{ od}, \sigma \rangle$

הגדרה: $\xrightarrow{*}$ הוא **הסגור הטרנזיטיבי** של $\longrightarrow$. אם נכתוב $C_1 \xrightarrow{*} C_2$ המשמעות היא שניתן לעבור מ-C_1 ע"י מספר סופי של צעדים (כולל 0) ל-C_2.

הגדרה: **חישוב** מקונפיגורציה C המסומן $\pi(C)$ הינו סדרה מקסימלית של קונפיגורציות $C_0, C_1, \ldots$ כך ש-$C = C_0$ ולכל $i \geq 0$ מתקיים $C_i \to C_{i+1}$.

הערה: כל חישוב סופי מסתיים ב-$\langle E, \sigma' \rangle$ - קונפיגורציה עוצרת.

הגדרה:

$$Val(\pi(C)) = \begin{cases} \sigma' & C \xrightarrow{*} \langle E, \sigma' \rangle \\ \perp & else \end{cases}$$

הגדרה: **המשמעות** (סמנטיקה) של תוכנית S היא $M[S](\sigma) = Val(\pi(S, \sigma))$.

דגש: נשים לב כי לפי ההגדרה, חישוב של B לוקח תמיד צעד אחד $\Leftarrow$ חישוב לא יכול "להתקע" ב-B

5.1.3. למת החישובים

למת החישובים מתארת את הצורה של כל חישובי התוכנית האפשריים. יהי $C_0 = \langle S_0, \sigma_0 \rangle$:

1. אם S_0 הוא פעולה אטומית אז $\pi(C_0)$ הוא מהצורה $C_0 \to C_1$ כאשר C_1 היא קונפיגורציה סופית (עוצרת) שנקבעת על ידי (1) או (2) בהגדרת $\to$.
2. $S_0 = S_1; S_2$. ל-$\pi(C_0)$ יש אחת משלוש צורות:

א. **התבדרות** S_1: $C_0 \to C_1 \to \ldots$ כאשר $C_i = \langle T_i; S_2, \sigma_i \rangle$ ו-$i \geq 0, \langle T_i, \sigma_i \rangle$ הוא חישוב אינסופי ב-S_1 מ-σ_0.

ב. **התבדרות** S_2: $C_0 \xrightarrow{*} \langle S_2, \sigma_i \rangle \to C_{i+1} \to \ldots$ כאשר $C_j, j \geq i$ הוא חישוב אינסופי של S_2.

ג. **עצירה**: $C_0 \xrightarrow{*} \langle S_2, \sigma_i \rangle \xrightarrow{*} \langle E, \sigma' \rangle$ כאשר $C_0 \xrightarrow{*} \langle S_2, \sigma_i \rangle$ מתאים לחישוב סופי של S_1 וגם $\langle S_2, \sigma_i \rangle \xrightarrow{*} \langle E, \sigma' \rangle$ הוא חישוב סופי של S_2.

3. אם $S_0 = \text{while } B \text{ do } S \text{ od}$ אז $\pi(C_0)$ מהצורה:

א. **עצירה מיידית**: $C_0 \to \langle E, \sigma_0 \rangle$ אם $\sigma_0 \models \neg B$.

ב. **התבדרות בחוג (S)**: (S לא מסתיים)

$C_0 \xrightarrow{*} \langle S_0, \sigma_1 \rangle \xrightarrow{*} \ldots \xrightarrow{*} \langle S_0, \sigma_k \rangle \to C_{k+1} \to \ldots$ כאשר $\sigma_j \vDash B$ לכל

$0 \le j \le k$ וכן $\langle S_0, \sigma_j \rangle \to \langle S; S_0, \sigma_j \rangle \xrightarrow{*} \langle E; S_0, \sigma_{j+1} \rangle$ הוא חישוב סופי של S

מ- σ_j ו- $\pi(C_{k+1})$ הוא חישוב אינסופי של S.

ג. **התבדרות החוג**: (התנאי B תמיד מתקיים)

$C_0 \xrightarrow{*} \langle S_0, \sigma_1 \rangle \xrightarrow{*} \ldots \xrightarrow{*} \langle S_0, \sigma_k \rangle \to \ldots$ כאשר $\sigma_j \vDash B$ לכל $j \ge 0$ וכן

$\langle S_0, \sigma_j \rangle \xrightarrow{*} \langle E, \sigma_{j+1} \rangle$ הינו חישוב סופי של S מ- σ_j.

ד. **עצירה**: $C_0 \xrightarrow{*} \langle S_0, \sigma_1 \rangle \xrightarrow{*} \ldots \xrightarrow{*} \langle E, \sigma_k \rangle$ כאשר $\sigma_j \vDash B, \sigma_K \vDash \neg B$ לכל

$0 \le j < k$.

4. אם $S_0 = \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}$ אז $\pi(C_0)$ הינו מהצורה:

א. אם $\sigma_0 \vDash B$ אז מבצעים חישוב של S_1: $\langle S_0, \sigma_0 \rangle \longrightarrow \langle S_1, \sigma_0 \rangle \xrightarrow{*} \ldots$

ב. אחרת מבצעים חישוב של S_2: $\langle S_0, \sigma_0 \rangle \longrightarrow \langle S_2, \sigma_0 \rangle \xrightarrow{*} \ldots$

5.1.4. מערכת הוכחה H להוכחת נכונות חלקית

מערכת ההוכחה H הינה מערכת דדוקטיבית. האקסיומות יכללו היסק מול טענות נכונות.
הוכחה במערכת תוגדר להיות סדרה של טענות שכל אחת מהן היא או אקסיומה, או טענה שהתקבלה מהטענות הקודמות על ידי אחד מכללי ההיסק.

הטענה להוכחה על ידי מערכת זו: $\{p\}s\{q\}$ - נכונות חלקית.

5.1.4.1. אקסיומות

אקסיומת ההצבה - $\{p[x \leftarrow e]\}x := e\{p\}$. (תסומן ב-ASS)

האקסיומה הראשונה שנציג הינה אקסיומת ההצבה. אקסיומה זו מעט מסובכת להבנה בהתקלות ראשונה, ולכן נסביר אותה בפירוט. דגש ראשון: <u>את אקסיומת ההצבה קוראים מימין לשמאל</u>.
$p[x \leftarrow e]$ הוא התנאי החלש ביותר שמבטיח ש- p יתקיים לאחר ביצוע $x := e$. משמעותו: לקחנו את התוכנית p והצבנו בכל מקום במקום x את הערך e.

<u>דוגמא 1</u>: נביט בחלק הימני: $\{x < 0\}x := x + 5$ - השאלה שאנחנו שואלים הינה: "מה צריך לדרוש כדי שיתקיים $x < 0$ אחרי ביצוע של הפעולה $x := x + 5$?". התשובה היא $x < -5$, ולכן נכתוב: $\{x < -5\}x := x + 5\{x < 0\}$. מבחינת טכניקה: נוכל לקבל את הצד השמאלי על ידי הצבת e בכל מקום שבו מופיע x ב- p, וכך לייצר את הצד השמאלי.
לחילופין, אם נתון רק הצד השמאלי: $\{p\}y := t\{?\}$ אנחנו יכולים ליצור את הצד הימני באופן אינטואיטיבי, על ידי הפעלת הפקודה שבתוכנית על החלק השמאלי.

<u>ניסוח חלופי לאקסיומת ההצבה</u>: מדוע לא ניסחנו טענה בסגנון $\{true\}x := e\{x = e\}$? מכיוון שכאן אין קשר בין המצב ההתחלתי לסופי. ראו לדוגמא: $\{true\}x := x + 1\{\underbrace{x = x + 1}_{false}\}$. כאשר e אינו תלוי ב- x מתקיים כי $\{true\}x := e\{x = e\}$ היא אקסיומה נכונה. למשל: $\{true\}x := 5\{x = 5\}$

אקסיומת skip - $\{p\}skip\{p\}$. האקסיומה תסומן $SKIP$.

אקסיומות אריתמטיות: כל טענה מהצורה $q \to q'$ שהינה אמת לוגית הינה אקסיומה. אקסיומות אלו יסומנו ב- $ARITH$.

5.1.4.2. כללי הסק

כאשר יש בידינו את האקסיומות, נוכל באמצעות כללי ההיסק לבנות טענות נוספות.

כלל הסק עבור הרכבה סדרתית (SEQ): $\dfrac{\{p\}s_1\{r\} \qquad \{r\}s_2\{q\}}{\{p\}s_1;s_2\{q\}}$

<u>דוגמא לשימוש</u>: רוצים להוכיח את הטענה הבאה:

$$\vdash_h \{x=X \wedge y=Y\}\, z:=x; x:=y; y:=z \{x=Y \wedge y=X\}$$

שימוש באקסיומת ההצבה: מסתכל על הביטוי מצידו הימני: $y:=z\{x=Y \wedge y=X\}$ נפעל כמו שהגדרנו בטכניקה עבור כלל ההצבה, ונקבל:

1. $(ASS)\ \{x=Y \wedge z=X\}\, y:=z \{x=Y \wedge y=X\}$

ניקח את הביטוי מצד שמאל, ונפעיל שוב את אקסיומת ההצבה עם ההצבה $x:=y$.

2. $(ASS)\ \{y=Y \wedge z=X\}\, x:=y \{x=Y \wedge z=X\}$

כעת נשתמש בהרכבה SEQ לשרשור התוצאות:

3. $(SEQ)1,2\ \{y=Y \wedge z=X\}\, x:=y; y:=z \{x=Y \wedge y=X\}$

נשתמש שוב באקסיומת ההצבה כאשר מימין אנו שמים את הצד השמאלי של (3).

4. $(ASS)\ \{y=Y \wedge x=X\}\, z:=x \{y=Y \wedge z=X\}$

לסיום נבצע שוב SEQ והוכחנו את הטענה שהתבקשנו להוכיח.

5. $(SEQ)3,4\ \{y=Y \wedge x=X\}\, z:=x; x:=y; y:=z \{x=Y \wedge y=X\}$

כלל הסק עבור if $(COND)$: $\dfrac{\{p\wedge B\}S_1\{q\} \qquad \{p\wedge\neg B\}S_2\{q\}}{\{p\}\,if\ B\ then\ S_1\ else\ S_2\ \ fi\ \{q\}}$

הכלל נאות: כל חישוב של if הוא חישוב של S_1 או של S_2 ובשני המקרים הוכחנו שהמצב הסופי (אם קיים) מספק את q.

כלל הסק עבור while (REP): $\dfrac{\{p\wedge B\}S\{p\}}{\{p\}\,while\ B\ do\ S\ \ od\,\{p\wedge\neg B\}}$

p היא האינואריאנטה של הלולאה. כלל זה חלש מידי עבור שלמות! קיימות טענות נכונות במערכת שלא נוכל להוכיח ולכן מוסיפים כלל נוסף – CONS.

כלל הסק $(CONS)$: $\dfrac{p\to p_1 \qquad \{p_1\}S\{q_1\} \qquad q_1\to q}{\{p\}S\{q\}}$ כאשר p,q הן טענות מסדר ראשון.

<u>דוגמא לאי-שלמות ללא CONS</u>:

נביט בטענה הבאה: $\{x=0\}$ while true do $x:=x-1$ od $\{x=0\wedge\text{false}\}$

הטענה אומרת "אם יודעים כי $x=0$ ואז מבצעים while true do $x:=x-1$ od נרצה שיתקיים $x=0\wedge\text{false}$".

הטענה נכונה בגלל שאנו מדברים על נכונות חלקית, והלולאה לעולם אינה עוצרת. אנו רוצים להוכיח את הטענה הנכונה, והדרך היחידה ללא $CONS$ לקבל טענת while היא ע"י REP. נוכיח אם כך את מה שמעל הקו של טענת $while$:

$$\{x=0\wedge true\}\ x:=x-1\ \{x=0\}$$

טענה זו <u>אינה נכונה</u> ומכיוון שמע' ההוכחה נאותה לא נוכל להוכיח אותה. קיבלנו שלילה של השלמות, כי לא ניתן להוכיח את הטענה ממנה התחלנו, שהיא טענה נכונה.

נראה כעת איך מוכיחים את הטענה כאשר CONS כן חלק ממערכת ההוכחה.

כאמור, אנו רוצים להוכיח $\{x=0\}$ while true do $x:=x-1$ od $\{x=0\wedge\text{false}\}$

נבחר את הטענה הנכונה הבאה:

$$1.\{\underbrace{true}_{p}\wedge\underbrace{true}_{B}\}x:=x+1\{\underbrace{true}_{p}\}$$

נפעיל REP:

$$2.\{true\}\,\text{while true do}\,x:=x-1\,\text{od}\{true\wedge false\}$$

נשתמש פעמיים באקסיומת ARITH:

$$3.true\wedge false\rightarrow x=0\wedge false$$
$$4.x=0\rightarrow true$$

נפעיל CONS על 2,3,4 ונקבל את הטענה אותה אנו מעוניינים להוכיח.

צריך לשים לב כי לא הראנו כי הוספת CONS גרמה למערכת להיות שלמה. הראנו כי הבעיה שהיתה קיימת לפני הוספת CONS (לא יכולנו להוכיח טענה נכונה) נפתרה.

5.1.4.3. משפט הנאותות של H

משפט: אם $\vdash_h\{p\}S\{q\}$ אז $\vDash\{p\}S\{q\}$

ההוכחה תהיה באינדוקציה על מבנה ההוכחה ב-H. נראה שכל אחת מהאקסיומות תמיד נכונה. נראה כי לכל כלל היסק, אם מתקיים החלק העליון שבכלל, אזי מתקיים גם החלק התחתון.

<u>בסיס</u>

$$\vDash\{p[x\le e]\}x:=e\{p\}$$

אם החישוב התחיל במצב σ כך שמתקיים $\sigma\vDash p[x\leftarrow e]$ אז המצב הסופי σ' יספק את p.

לפי למת החישובים, החישוב היחיד שיש להשמה הוא החישוב הבא:

$$\langle x:=e,\sigma\rangle\rightarrow\langle E,\sigma[x\leftarrow e]\rangle$$

צריך להוכיח כי $\sigma\vDash p[x\leftarrow e]\rightarrow\sigma[x\leftarrow e]\vDash p$. ביטוי זה נכון לפי המשפט מלוגיקה.

כעת נבדוק עבור $skip$: $\models \{p\}\, skip\, \{p\}$

נשתמש שוב בלמת החישובים, ונראה כי צריך להוכיח: $\sigma \models p \rightarrow \sigma \models p$, ביטוי שהוא נכון.

<u>הוכחת כללי ההיסק</u>

<u>הרכבה סדרתית (SEQ)</u>:

<u>נתון</u>: $\models \{p\}\, S_1\, \{r\}, \models \{r\}\, S_2\, \{q\}$

<u>צ"ל</u>: $\models \{p\}\, S_1; S_2\, \{q\}$

נשאל את עצמנו איזה סוג חישובים יש להרכבה סידרתית לפי למת החישובים. ישנם חישובים מתבדרים וישנם חישובים עוצרים. המקרה בו אנו מתעניינים עבור נכונות חלקית הוא מקרה שבו גם S_1 וגם S_2 עוצרים. עפ"י למת החישובים מתקיים $C_0 \xrightarrow{*} \langle S_2, \sigma_1 \rangle \xrightarrow{*} \langle E, \sigma_1' \rangle$.

החלק $C_0 \xrightarrow{*} \langle S_2, \sigma_1 \rangle$ מתאים לחישוב סופי של S_1 מלבד תוספת "$;S_2$" בכל קונפיגורציה.

עפ"י הנתון:

$$(*) \qquad \sigma \models p \Rightarrow \sigma_1 \models r$$

$\langle S_2, \sigma_1 \rangle \xrightarrow{*} \langle E, \sigma' \rangle$ הוא חישוב של S_2 ולכן עפ"י הנתון:

$$(**) \qquad \sigma_1 \models r \Rightarrow \sigma' \models q$$

מ-$(*)$ ומ-$(**)$ נסיק ש- $\sigma \models p \Rightarrow \sigma' \models q$ כנדרש.

<u>לולאה (REP)</u>:

<u>נתון</u>: $\models \{p \wedge B\}\, S\, \{p\}$

<u>צ"ל</u>: $\models \{p\}\, \text{while } B \text{ do } S \text{ od}\, \{p \wedge \neg B\}$

למת החישובים: מעניין אותנו לבדוק חישובים שעוצרים, כי עבור חישובים שלא עוצרים הטענה נכונה באופן טריויאלי.

$C_0 \xrightarrow{*} \langle S_0, \sigma_1 \rangle \xrightarrow{*} ... \xrightarrow{*} \langle E, \sigma_k \rangle$. נסמן ב- $C_i = \langle while..., \sigma_i \rangle$. אנו יודעים כי עבור $0 \le i < k$ מתקיים $\sigma_i \models B$ וכן כי $\sigma_k \not\models B$.

באינדוקציה על מספר ביצועי החוג, נראה כי לכל $i \le k$ מתקיים $\sigma_i \models p$, בתנאי ש- $\sigma_0 \models p$.

בסיס: צ"ל כי $\sigma_0 \models p$. זה נכון מכיוון שזה נתון.

צעד: נניח כי $\sigma_m \models p$. נראה כי $\sigma_{m+1} \models p$ עבור $m+1 \le k$.

על פי $\models \{p \wedge B\} S \{p\}$ אז לכל σ, σ' כך שהחישוב של S מתחיל ב- σ ומסתיים ב- σ' מתקיים $\sigma \models p \wedge B \Rightarrow \sigma' \models p$. מכאן המסקנה $\sigma_{m+1} \models p$.

עבור σ_k , מתקיים כי $\sigma_k \models p$ על פי ההוכחה באינדוקציה. $\sigma_k \models \neg B$ ולכן קיבלנו את מה שצריך להוכיח.

כלל CONS:

נתון:

1. $\models p \to p_1$
2. $\models \{p\} S \{q_1\}$
3. $\models q_1 \to q$

צריך להוכיח: $\models \{p\} S \{q\}$

יהי $\sigma \models$. על סמך נתון 1, מתקיים כי $\sigma \models p_1$ (כל מה שמספק p מספק את p_1).

על סמך נתון 2: $\sigma \models p_1 \to \sigma' \models q_1$

על סמך נתון 3: $\sigma' \models q_1 \to \sigma' \models q$

על סמך טרנזיטיביות הגזירה נקבל: $\sigma \models p \to \sigma \models q$ ומכאן: $\models \{p\} S \{q\}$

כלל COND:

הכלל מוכח באופן דומה ל-SEQ. ההוכחה לא תוצג במסמך זה.

5.1.4.4. משפט השלמות של H

אם $\models \{p\}S\{q\}$ אז $\vdash_H \{p\}S\{q\}$

זוהי שלמות יחסית. נסביר מהי שלמות יחסית ומדוע השלמות של H היא כזו. השלמות אינה מתקיימת תמיד במקרה של H, עקב מספר נושאים:

- לא תמיד ניתן לבטא את תנאי ההתחלה והסיום באמצעות לוגיקה כלשהי. (כוח הביטוי של הלוגיקה מוגבל). לפיכך יתכנו מקרים נכונים שפשוט אין לנו דרך לבטא אותם בכתב.
- ידוע לפי משפט אי השלמות של גדל (משפט שחורג מתחום מסמך זה) שעבור המספרים השלמים אין מע' ההוכחה שלמה.

מכיוון שכך, אנו אומרים כי "אם אין בעיות בלוגיקה מסדר ראשון בה אנו משתמשים", אז מתקיימת שלמות. לכן אנחנו לא נדרשים להוכיח טענות לוגיקה מסדר ראשון בכלל CONS. כלומר, אנחנו עדיין צריכים להסתכל שהטענות ב-CONS נכונות, אבל אנחנו בודקים אותן בעזרת ידע מוקדם. לדוגמא – אנחנו מסתכלים על הטענות $x<1 \rightarrow x<10$ ואומרים – "הטענה נכונה (ARITH) לפי ידע קודם ממתמטיקה".

5.1.4.5. קצת פרקטיקה

איך בפועל אנחנו מוכיחים תוכניות במערכת ההוכחה H? נציג את דרך העבודה.
אנו מקבלים תוכנית בשפת while וטענות שאנחנו רוצים לבדוק אם הן מתקיימות לגבי התוכנית.
בהנתן תוכנית P, טענה התחלתית q_1 וטענה סופית q_2, הרי שהשורה האחרונה בהוכחה שלנו תהיה $\{q_1\}P\{q_2\}$. כדי להוכיח את התוכנית נרצה לרשום סדרת כללי הוכחה המתחילים באקסיומות, ובהמשכם אקסיומות וכללי היסק, עד להגעה לתוצאה הסופית.

נשים לב שעם תחילת ההוכחה, <u>אנחנו יודעים מה אנחנו רוצים לקבל – את השורה האחרונה.</u> אנחנו צריכים לקבוע מה היה קודם. נשים לב לנקודה חשובה נוספת: <u>בכללי ההיסק שנציג הזהות חייבת להיות סינטטקטית, ולא רק סמנטית.</u> ההוכחה שלנו היא הוכחה בלוגיקה.

בהנתן תוכנית – נוח לרוב להתחיל בלולאות. אנחנו לוקחים לולאה ורושמים אותה כשלב בהוכחה.
אם נסתכל על כלל REP, נשים לב כי p היא השמורה של הלולאה. נחשוב על שמורה עבור הלולאה שלנו, ובעזרתה נרכיב את הצעד בהוכחה הכולל את REP. זהו החלק הכי יצירתי לרוב בהוכחת HOARE שהיא מאוד טכנית. איך נמצא שמורה אם היא לא ברורה? דרך מומלצת היא לבצע מספר איטרציות של הלולאה, ולהרגיש מה קורה – איזה ערכים מקיימים חוקיות ביניהם. זו תהיה בחירה טובה כשמורה.

בשלב הבא בהוכחה, לאחר שכתבנו את REP, נכתוב את השלב שלפני REP – למעשה את הביטוי שמעל הקו בכלל ההיסק. זהו הביטוי שאנחנו רוצים להוכיח, כדי שה-REP יתקדם.
בכל שלב בתהליך ההוכחה אנחנו מסתכלים מה היה צריך לקרוא כדי שהביטוי בשלב הנוכחי יתקיים, ורושמים את הביטוי הנדרש, עד שאנחנו מגיעים אל האקסיומות.

5.1.5. מערכת H* להוכחת $\langle p \rangle S \langle q \rangle$

במערכת זו, מלבד כלל REP של מערכת H, נאמץ את כל הכללים כמו שהם תוך כדי החלפת $\{\ \}$ ב-$\langle\ \rangle$. **כלל REP החדש**: נשתמש ב- $\mathbb{N}$ עם הסדר הרגיל (בתור $(w,<)$). נקבע טענות $p(\bar{x},n)$ כאשר $\bar{x}$ משתני התוכנית ו-n מעל הטבעיים (כולל 0):

$$\frac{\left(p(\bar{x},n)\wedge n>0\right)\to B \qquad \langle p(\bar{x},n)\wedge(n>0)\rangle S \langle p(\bar{x},n-1)\rangle \qquad p(\bar{x},0)\to\neg B}{\langle \exists n, p(\bar{x},n)\rangle \text{ while } B \text{ do } S \text{ od } \langle p(\bar{x},0)\rangle}$$

<u>משפט הנאותות</u>: $\vdash_{H^*} \langle p_1\rangle S\langle q_1\rangle \Rightarrow \models \langle p_1\rangle S\langle q_1\rangle$

נוכיח את משפט הנאותות רק עבור מקרה פרטי: עבור תוכנית $S = while\ B\ do\ S'\ od$ כאשר S' אינו מכיל חוגים.

<u>הוכחה</u>:

התוכנית S הינה ללא חוגים, לפי הנאותות של H. כמו כן: $S = while\ B\ do\ S'\ od$. לפי למת החישובים, ביצוע S שאינו עוצר נראה כך: $\langle S,\sigma_0\rangle \xrightarrow{*} \langle S,\sigma_1\rangle \xrightarrow{*} \ldots$ כאשר $\forall i, \sigma_i \models B$.

כדי להוכיח נאותות, נביט איך נראית הוכחה של WHILE, ונראה כי הנאותות מתקיימת:

1. $p(n)\wedge(n>0)\to B$	(נתון מהנאותות)
2. $\langle p(n)\wedge n>0\rangle S\langle p(n-1)\rangle$	(נתון מהנאותות)
3. $p(0)\to\neg B$	(נתון מהנאותות)
4. $\langle \exists n, p(n)\rangle\ while\ B\ do\ S'\ od\ \langle p(0)\rangle$	$(\text{REP } 1,2,3)$
5. $p_1 \to \exists n,\left(p(n)\right)$	
6. $p(0)\to q_1$	
7. $\langle p_1\rangle\ while\ B\ do\ S'\ od\ \langle q_1\rangle$	$(\text{CONS } 4,5,6)$

נסיק שלא קיים חישוב אינסופי כי חישוב שמתחיל מ- σ $\left(\sigma \vDash p_1\right)$ ע"ס S: $\sigma \vDash \exists_n p(n)$ לכן קיים k כך ש- $\sigma \vDash p(k)$.

ניתן להראות באינדיקציה ש- $p(0)$ מתקיים תוך k צעדים עבור σ' $\sigma' \vDash p(0) \Rightarrow \sigma' \vDash \neg B$ (עפ"י 3) ולכן יש עצירה.

תרגיל

<u>המטרה</u>: לנסח כלל REP בו הירידה היא לאו דווקא ב-1 והעצירה לאו דווקא ב-0.

<u>פתרון 1</u>: נגדיר אוסף כללים $REP^*(c)$ - כלל לכל קבוע טבעי אפשרי c:

$$REP^*(c):\ \ p(\bar{x},n) \wedge (n > c) \rightarrow B$$

$$\frac{\begin{array}{l}\left\langle p(\bar{x},n) \wedge (n > c)\right\rangle S \left\langle \exists n' : p(x,n') \wedge c \le n' \le n\right\rangle \\ p(\bar{x},c) \rightarrow \neg B\end{array}}{\left\langle \exists n, p(X,n) \wedge (n \ge c)\right\rangle \ while\ B\ do\ S\ od\ \left\langle p(\bar{x},c)\right\rangle}$$

<u>פתרון 2</u>:

$$\frac{\begin{array}{l}\left\langle p(n) \wedge n > 0 \wedge B\right\rangle S \left\langle \exists n' < n, p(n')\right\rangle \\ p(0) \rightarrow \neg B\end{array}}{\left\langle \exists n, p(n)\right\rangle while\ B\ do\ S\ od \left\langle \exists n', p(n') \wedge \neg B\right\rangle}$$

תרגיל

סעיף א':

נשנה את מערכת ההוכחה ונראה כיצד השינוי משפיע על המערכת. נשנה את כלל REP על ידי שנבטל את ההנחה $\left(p\left(\overline{x},n\right)\wedge n>0\right)\to B$. מה ניתן להגיד כעת על מערכת ההוכחה?

כלל REP החדש:

1. מבטיח שאם $n>0$ והדרישה מתקיימת אז נכנסים אל הלולאה.
2. יכול להיות ש- $n>0$ וגם B הוא false. מכאן: לא מבטיחים עצירה של הלולאה ב-0. הלולאה יכולה לעצור כאשר $n>0$.
3. הכלל עדיין מבטיח את עצירת הלולאה!

האם המערכת שלמה? התשובה היא כן – כי החלשנו את התנאי. השלמות לא תפגע. לעומת זאת, הנאותות איננה נשמרת.

סעיף ב':

כעת נשנה את הכלל השני ב-REP להיות $\left\langle p\left(\overline{x},n\right)\right\rangle S\left\langle p\left(\overline{x},n-1\right)\right\rangle$ במקום $\left\langle p\left(\overline{x},n\right)\wedge\left(n>0\right)\right\rangle S\left\langle p\left(\overline{x},n-1\right)\right\rangle$. איך הנאותות והשלמות של המערכת יושפעות במקרה זה?

במקרה זה אנו פוגעים בשלמות אך לא בנאותות. נשים לב שבעת ההוכחה נצטרך להוכיח כעת יותר: נצטרך להוכיח שהטענה מתקיימת גם עבור כל n שלילי, ולא רק עבור n גדול מ-0.

6. אימות אוטומטי - שיטות לבדיקת מודל

6.1. מבוא

המטרה של חלק זה: אימות אלגוריתמי שתמיד עוצר ועונה "כן" אם המערכת מספקת את המפרט, ו-"לא" + דוגמא נגדית אם המערכת אינה מספקת את המפרט.

.

נביט כעת על מערכות בעלות מספר סופי של מצבים. התוכניות יהיו ריאקטיביות - אינן בהכרח עוצרות: מערכות הפעלה, פרוטוקולי תקשורת וכדו'. מערכות שאמורות לפעול לנצח ולא לעצור ומגיבות לקלט מהסביבה.

המפרט שנבחר בו לצרכי ביטוי והוכחה יהיה מפרט בלוגיקה טמפורלית פסוקית, שזוהי לוגיקה שיכולה לתאר תופעות לאורך זמן.

	הוכחת נכונות	אימות אוטומטי
מספר המצבים	אינסופי או גדול מאוד	סופי וקטן יחסית (עד 10^{300})
סוג התוכנית	טרנספורמטיבית	ריאקטיבית = תגובתית
מפרט	לוגיקה מסדר ראשון (למשל תחשיב היחסים)	לוגיקה טמפורלית (עיתית)
חישוביות	בלתי כריעה	כריעה
אוטומציה	עזרה מהמשתמש	אוטומציה מלאה
כלים	Theorem proves	RuleBase

השוואה בין הנושאים בהם עסקנו בחלקו הראשון של המסמך לבין הנושאים בהם נעסוק בחלק זה

דוגמאות נוספות למערכות בעלות מספר מצבים סופי:

1. מעגל חשמלי.
2. פרוטוקולי תקשורת (תוך התעלמות מתוכן ההודעה).
3. אבסטרקציה סופית של תוכניות עם מספר מצבים אינסופי.

האם מעניין אותנו להוכיח דברים על לוגיקה טמפורלית פסוקית? התשובה היא כן. נציג בקצרה מספר דוגמאות מהעולם האמיתי למפרטים שניתנים לביטוי בלוגיקה טמפורלית פסוקית על מנת להראות זאת:

1. **מניעה הדדית - mutual exclusion**

   CS_i משתנה פסוקי שהוא נכון במצב אם ורק אם תהליך i נמצא בקטע הקריטי CS.

   נוכל לכתוב בלוגיקה טמפורלית: $\neg(CS_1 \wedge CS_2)$ globally – כלומר לעולם לא יהיו שני תהליכים בקטע הקריטי.

2. **אי הרעבה**

   כל דרישה תתמלא בסופו של דבר (תוך זמן סופי, eventually).

   $$\text{globally}\ (\text{request} \rightarrow \text{eventually granted})$$

3. **פרוטוקולי תקשורת**

   כל הודעה שנשלחה תגיע תוך זמן סופי: $globally(sent \rightarrow eventually\ arrived)$.

   כל הודעה שמתקבלת גם נשלחה קודם: $(\neg\ got-message)\ until\ \ sent-message$

6.2. מבנה קריפקה - *Kripke Structure*

מבנה קריפקה הוא סוג של מכונת מצבים סופית אי-דטרמיניסטית המשמשת לייצוג התנהגות של מערכת. מבנה קריפקה הוא בבסיסו גרף שצמתיו מתארים מצבים ישיגים של המערכת והקשתות מייצגות מעבר מצבים. פונקציה מיוחדת מתאימה תווית לכל צומת לקבוצת מאפיינים המתארים את מצבה. מבנה קריפקה משמש אותנו להציג את הטענות השונות שלנו בלוגיקה הטמפורלית.

הגדרה פורמלית:

תהי AP קבוצת נוסחאות אטומיות. מבנה קריפקה הוא הרביעיה: $M=(S,R,L,S_0)$ כאשר:

- S - קבוצת מצבים סופית.
- $R \subseteq S \times S$ רלציית מעברים R טוטאלית (לכל S קיים S' כך ש-$(S,S') \in R$).
- $L: S \rightarrow 2^{AP}$ פונקצית סימון המתאימה לכל מצב את הנוסחאות האטומיות שמתקיימות בו.
- $S_0 \subseteq S$ - קבוצת מצבים התחלתיים (אופציונאלי)

תוצאה של שימוש ברלציה הטוטאלית: כל החישובים במבנה הם אינסופיים.

<u>דוגמא</u>

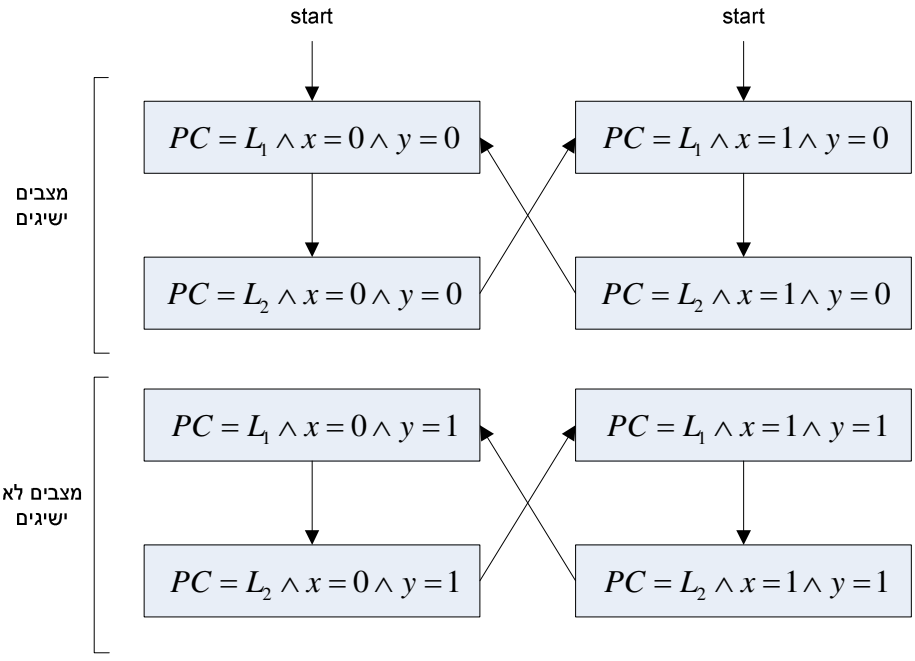
נציג דוגמא לתוכנית וכיצד בונים ממנה מבנה קריפקה. מטרת הדוגמא היא להראות את המוטיבציה לשימוש במבנה, ולכן נציג את בניית המבנה בשלבים.

תהי התוכנית בעלת המשתנים x, y שתחומן $\{0,1\}$ וכן משתנה PC שתחומו $\{L_1, L_2\}$.

התוכנית עצמה היא:

```
L1:   while true do
          L2:  x:=¬x
      od
```

נתון שהתוכנית מתחילה במצב $y = 0$. מתקיים תמיד כי כל אחד ממצבי התוכנית נותן ערך ל- x, y, PC. נבנה את מכונת המצבים הבאה:



נשים לב כי במקרה זה קל לראות מה הם המצבים הישיגים ואילו מצבים אינם ישיגים. לא תמיד נדע מה הם המצבים הישיגים.

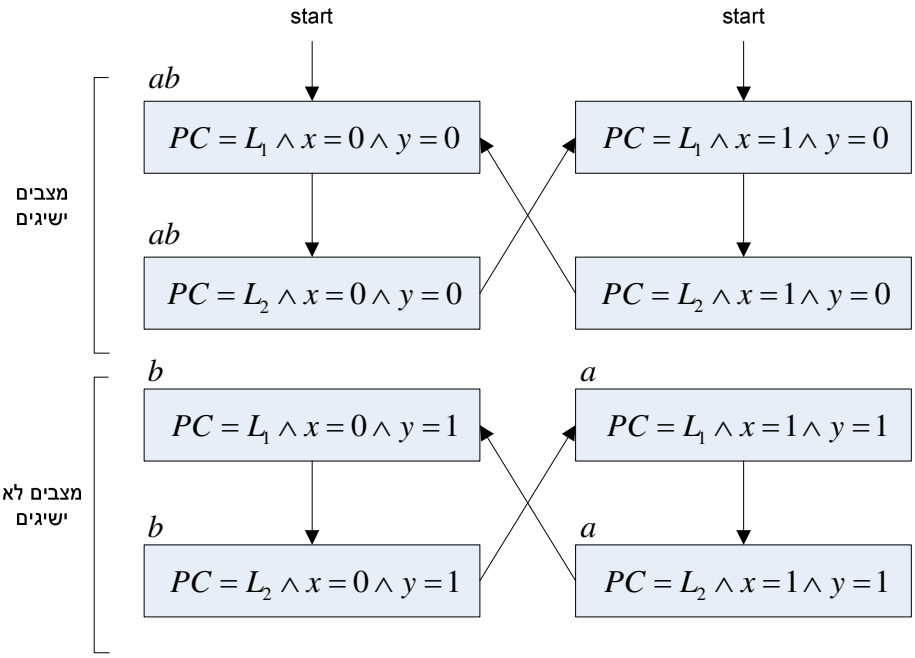
נרצה לבדוק תכונות על התוכנית. נגדיר 2 תכונות:

a: $x = y$

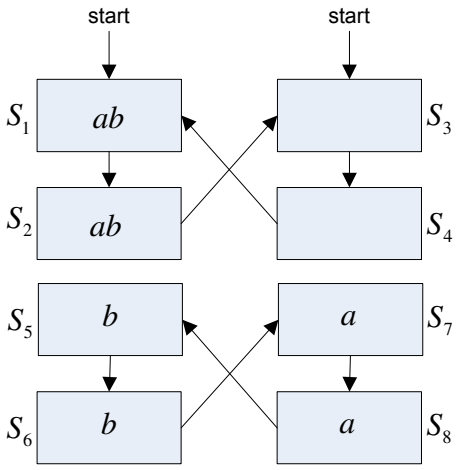
b: $x = 0$

נקרא לתכונות אלו בשם נוסחאות אטומיות.

כעת, ניקח את מכונת המצבים וליד כל מצב נרשום את התכונות האטומיות אותן היא מקיימת:



כעת נעבור למכונת קריפקה. ניתן סימון לכל מצב. ערכי המשתנים המקוריים לא מעניינים אותנו.



עבור מכונת הקריפקה שהוצגה, מתקיים:

$$S = \{S_1, S_2, ..., S_8\}$$
$$S_0 = \{S_1, S_3\}$$
$$R = \{(S_1, S_2), (S_2, S_3), (S_3, S_4), (S_4, S_1), (S_5, S_6), (S_6, S_7), (S_7, S_8), (S_8, S_5)\}$$
$$L(S_2) = L(S_1) = \{a, b\}$$
$$L(S_3) = L(S_4) = \varnothing$$
$$L(S_5) = L(S_6) = \{b\}$$
$$L(S_7) = L(S_8) = \{a\}$$

הצגנו דוגמא לבניית מודל מתוך תוכנית נתונה. החל משלב זה אנו הולכים להתעלם מבניית המודל ולהתרכז במודל עצמו. לא נשאל את עצמנו בהמשך מהי התוכנית ממנה בנו את המודל, אלא ניקח מודלים נתונים, ונשאל עליהם שאלות.

6.3. לוגיקות טמפורליות פסוקיות

נציג כעת מספר לוגיקות טמפורליות פסוקיות שונות בהן נשתמש לבדיקת מודל, והן:

- הלוגיקה CTL (Computation Tree Logic)
- הלוגיקה LTL (Linear Temporal Logic)
- הלוגיקה CTL*.

הלוגיקה CTL* היא הגדולה מבין 3 הלוגיקות והיא מכילה ממש את הלוגיקות CTL ו-LTL.
הלוגיקות CTL ו-LTL הן בעלות יכולת ביטוי שאינה ניתנת להשוואה. הלוגיקות נבדלות בשילובים של כמתי מסלול ואופרטורים טמפורליים המותרים בכל אחת מהן. נציג את הלוגיקות השונות ונעמוד על ההבדלים ביניהן.

מבחינה היסטורית, אמיר פנואלי הציע ב-1977 לעשות שימוש ב-LTL לצורך הוכחת תוכניות. 4 שנים מאוחר יותר, ב-1981 המציאו Clarke ו-Emerson את CTL ואת בדיקת המודל באמצעות CTL. הלוגיקה CTL* הומצאה רק מאוחר יותר, ב-1986. בפרקטיקה LTL ו-CTL נמצאים בשימוש בפועל, ואילו CTL* נפוצה הרבה פחות. הסיבה העיקרית היא כוחות שוק המקדמים לוגיקות אלו.

6.3.1. מרכיבי הלוגיקות הטמפורליות

קבוצת נוסחאות אטומיות: AP סופית

אופרטורים בוליאניים: $\wedge, \vee, \neg, \ldots$

אופרטורים טמפורליים:

- Fp (Future, Eventually) – קיים מצב בעתיד שבו p מתקיים. (החלטה: העתיד כולל גם את הצומת הנוכחי, "ההווה").
- Gp (Globally) - p מתקיים בכל מצב על המסלול (החל מהצומת הנוכחי).
- Xp (next) - p נכון במצב הבא בסדרה. (המצב השני בסדרה)
- pUq (Until) - q מתקיים מתישהו בעתיד ו- p נכון בכל המצבים הקודמים.

כמתי מסלול:

- $A\varphi$ - φ מתקיים על כל המסלולים, כולל המצב הנוכחי.
- $E\varphi$ - קיים לפחות מסלול אחד היוצא מהמצב הנוכחי עבורו φ מתקיים.

6.3.2. אופרטורים טמפורליים

1. **Globally (מסומן ב-G)** - תכונה שמתקיימת לכל אורך התוכנית.

   לדוגמה: $G\left(\neg at\_critical_1 \vee \neg at\_critical_2\right)$.

   (לא יתכן ששני חוטים יהיו בקטע הקריטי בו זמנית).

2. **Future (Eventually)** - תכונה שתתקיים מתישהו בעתיד.

   לדוגמה: $request \rightarrow F\left(granted\right)$

   (אם תהליך ביקש משאב כלשהו, בסופו של דבר הוא יקבל אותו מתישהו).

   <u>דוגמאות</u>:

   - <u>מתקיים</u>: $G\left(p\right)=\neg F\left(\neg p\right)$ כלומר ש P מתקיים תמיד זה כמו לומר שלא יקרה בעתיד שיתקיים "לא P".
   - GFp - בכל רגע בתוכנית מתקיים שמתישהו בעתיד יתקיים P.
   - FGp - החל מרגע מסוים בעתיד, P יתקיים לנצח.

3. **neXt(p)** - p יתקיים בצעד הבא. יסומן ב- X.

   <u>דוגמאות</u>:

   - XGp - החל מהצעד הבא, p יתקיים תמיד.
   - GXp - בכל מצב יתקיים שבמצב הבא לאחריו p יתקיים.
   - קיבלנו: $XGp = GXp$

4. **(p) Until (q)** - יסומן ב-**U**.

   pUq - מתישהו בעתיד q מתקיים, ועד אז בהכרח p מתקיים.

   <u>דוגמא</u>:

   "אם הודעה 1 נשלחה לפני הודעה 2 אזי הודעה 1 תגיע לפני הודעה 2".

   - האטומים הם $sent_1, sent_2, received_1, received_2$.
   - הביטוי המתאים: $\neg sent_2 U sent_1 \rightarrow \neg received_2 U received_1$

5. **(p) Release (q)** – יסומן ב-**R**.

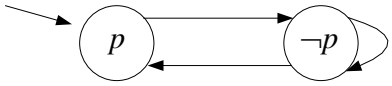
   pRq - q מתקיים עד שיתקיים p, כולל המצב הראשון שבו יתקיים p. לא בהכרח יתקיים בעתיד p.

טענה: האופרטורים U, X מספיקים כדי להגדיר את R, F, G.

- $F(p) = (True)U(p)$
- $G(p) = \neg F(\neg p) = \neg((True)U(\neg p))$

כמתי מסלול: $A - for\ all$ $E - exist,$

דוגמא:



מתקיים: EXp אבל לא מתקיים AXp. כלומר: קיים מסלול בו מתקיים Xp אבל לא בכל מסלול מתקיים Xp.

CTL* .6.3.3

הגדרת CTL* .6.3.3.1

CTL* הינה סוג של לוגיקה טמפורלית, המוגדרת באמצעות 2 סוגי נוסחאות:

1. נוסחאות מצב שמתפשרות במצב נתון.
2. נוסחאות מסלול שמתפרשות במסלול נתון.

<u>הגדרה אינדוקטיבית של הלוגיקה:</u>

<u>נוסחאות מצב</u>

1. <u>בסיס</u>: נוסחאות אטומיות $p \in AP$.
2. <u>כללי יצירה</u>:
   - אם f, g נוסחאות מצב אז $\neg f, f \vee g$ הן נוסחאות מצב.
   - אם f נוסחת <u>מסלול</u> אז Ef היא נוסחת מצב.

<u>נוסחאות מסלול</u>

1. <u>בסיס</u>: כל נוסחת מצב היא נוסחת מסלול. (הנוסחה תבחן על המצב הראשון במסלול).
2. <u>כללי יצירה</u>: אם f, g נוסחאות מסלול אז $\neg f, f \vee g$ Xf, fUg הן נוסחאות מסלול.

<u>CTL*</u> <u>היא קבוצת נוסחאות המצב המוגדרות לפי הגדרה זו</u>. נסביר זאת: CTL* מוגדרת באופן אינדוקטיבי באמצעות נוסחאות מצב ונוסחאות מסלול. נוסחאות המסלול משמשות אותנו להגדרת קבוצת הנוסחאות CTL* (כצעד ביניים), אך הבנייה לעולם לא עוצרת כשיש לנו ביד נוסחת מסלול.

הערות להגדרה האינדוקטיבית:

- חשוב לשים לב: כל נוסחה אטומית ב-AP היא נוסחת מצב של CTL*.
- המסלול מספק נוסחת מצב אם המצב הראשון במסלול מספק את הנוסחה.
- שאלה: האם ניתן לעשות $\vee$ בין נוסחת מצב לנוסחת מסלול? לפי ההגדרה – כן, כי כל נוסחת מצב היא נוסחת מסלול. φ_1 נוסחת מצב, φ_2 נוסחת מסלול, אזי לפי הבסיס φ_1 היא גם נוסחת מסלול ו- $\varphi_1 \vee \varphi_2$ היא נוסחת מסלול.

נוסחאות CTL* מתפרשות מעל מבנה קריפקה M: $M=(S,R,L,S_0)$. S_0 הינו אופציונלי בלבד.

<u>הגדרה</u>: **מסלול** ב- M ממצב S הינו סדרה אינסופית $S_0,S_1,S_2,...$ כך שלכל $i\geq 0$, מתקיים $(S_i,S_{i+1})\in R$ ובנוסף $S_0=S$.

<u>סימונים</u>:

- π^j - הסיפא של π ממצב S_j (π^0 - כל המסלול)

<u>סמנטיקה</u>:

- $M,S\models f$ - אם f נוסחת מצב נכונה במבנה M במצב S.
- $M,\pi\models f$ - אם f נוסחת מסלול הנכונה במבנה M במסלול π.
- $M\models f$ אם M כוללת הגדרת S_0 ולכל $s_0\in S_0$ $M,s_0\models f$.

6.3.3.2. הגדרה אינדוקטיבית של $\models$

נגדיר סמנטיקה עבור נוסחאות CTL* באינדוקציה על מבנה הנוסחה.

<u>סימונים בהגדרה</u>:

- f_1,f_2 - נוסחאות מצב.
- g_1,g_2 - נוסחאות מסלול.

<u>נוסחאות מצב</u>:

- $S\models p$ עבור $p\in L(S)\Leftrightarrow p\in AP$
- $S\not\models f\Leftrightarrow S\models\neg f$
- $S\models f_1\vee f_2\Leftrightarrow S\models f_1$ או $S\models f_2$
- $S\models Eg_1\Leftrightarrow$ קיים מסלול π מ- S כך ש- $\pi\models g_1$

נוסחאות מסלול:

- $S_0 \models f \Leftrightarrow \pi \models f$ כאשר S_0 המצב הראשון ב- π
- $\pi \not\models g_1 \Leftrightarrow \pi \models \neg g_1$
- $\pi \models g_1$ או $\pi \models g_2 \Leftrightarrow \pi \models g_1 \vee g_2$
- $\pi^1 \models g_1 \Leftrightarrow \pi \models Xg_1$
- $\pi \models g_1 U g_2$ קיים $k \geq 0$ כך ש: $\pi^j \models g_1$, $\pi^k \models g_2$ לכל $0 \leq j < k$

קיצורים: (אופרטורים המוגדרים בעזרת האופרטורים הבסיסיים)

- $f_1 \wedge f_2 \equiv \neg\left(\neg f_1 \vee \neg f_2\right)$
- $g_1 \wedge g_2 \equiv \neg\left(\neg g_1 \vee \neg g_2\right)$
- $Af \equiv \neg E \neg f$
- $Ff \equiv \left(true U f\right)$
- $Fg \equiv \left(true U g\right)$
- $Gg \equiv \neg F \neg g$

הארה

נסתכל על הקיצור האחרון $Gg \equiv \neg F \neg g$. האם היינו יכולים לכתוב אותו בעזרת Until?

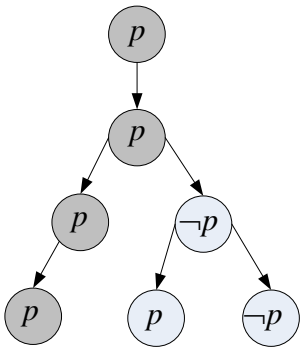
כיוון אפשרי יכל להיות $gUfalse$ אך זה איננו כיוון טוב – כי U מבטיח שבהכרח false יתקיים בסופו של דבר. האופרטור weak-until (הידוע גם בכינויו unless) היה יכול להתאים: $gWfalse$. אם weak-until היה חלק מהבסיס זו היתה הגדרה נכונה.

6.3.3.3. דוגמאות לנוסחאות + משמעות אינטואיטיבית

בדוגמאות הבאות, $p \in AP$.

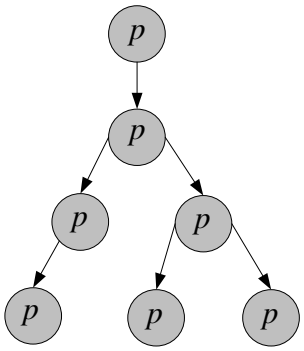
דוגמא 1

הגרף מקיים EGp עבור המסלול השמאלי ביותר:



דוגמא 2

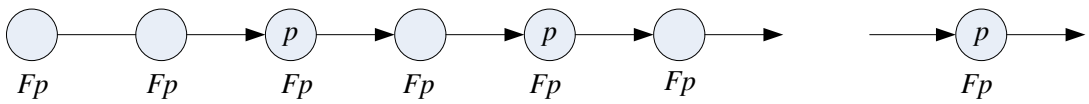
הגרף מקיים AGp :



דוגמא 3

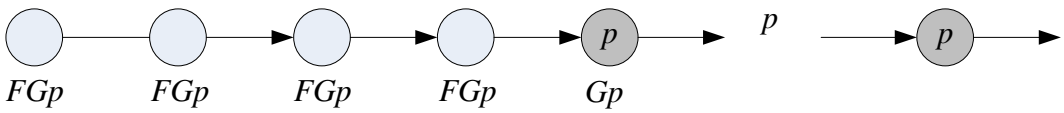
נסתכל על נוסחת המסלול הבאה: $\pi \models GFp$.

משמעות הנוסחה: p מתקיים על π אינסוף פעמים. "קיימת סדרה אינסופית של p במסלול".



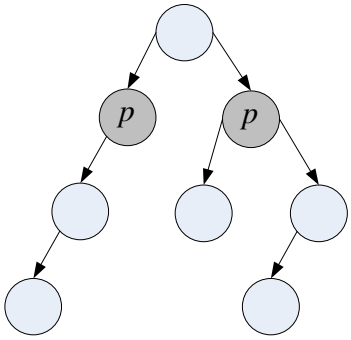
דוגמא 4

הנוסחה $\pi \models FGp$. קיים מצב על המסלול שממנו והלאה מסתפק p .



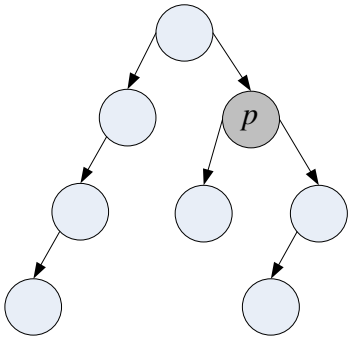
דוגמא 5

הנוסחה $\pi \models AXp$:



דוגמא 6

הנוסחה $\pi \models EXp$:



דוגמאות נוספות (ללא שרטוט):

1. $S \vDash EGFp$

קיים מסלול שעליו אינסוף פעמים p נכון.

2. $S \vDash EGEFp$

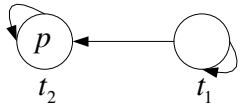
קיים מסלול מ- S שלכל מצב עליו קיים מסלול עליו מתקיים p (לא בהכרח המשך המסלול הראשון שיוצא מ- S).

<u>הגדרה</u>: נוסחת CTL* φ נקראת **ספיקה** אם קיים מבנה M כך ש- $M \vDash \varphi$.

<u>דוגמאות</u>:

1. האם הנוסחה $\varphi = E\left(GEXp \wedge FX\neg p\right)$ ספיקה?

הנוסחה ספיקה, נראה מבנה שמספק אותה:



המסלול המוכיח: $\pi : t_1 \to t_1 \to t_1 \to t_1 \to ...$

2. $\varphi = EXp \wedge X\neg P$ אינה ספיקה.

CTL .6.3.4

CTL הינה לוגיקה טמפורלית נוספת, שהיא למעשה תת שפה של CTL*: CTL מוגדרת מעל נוסחאות מצב בלבד. **CTL** היא השפה המוגדרת ע"י:

1. אם $p \in AP$ אז p נוסחת CTL.
2. אם f, g נוסחאות CTL אז $\neg f, f \vee g$ $E(fUg), EXg, EGg$ נוסחאות CTL.

סמנטיקה ישירה של נוסחאות CTL: (מעל מבנה M ומצב S)

- $p \in L(S) \Leftrightarrow S \vDash p$
- $S \nvDash f \Leftrightarrow S \vDash \neg f$
- $S \vDash g$ או $S \vDash f \Leftrightarrow S \vDash f \vee g$
- $S \vDash EXf \Leftrightarrow$ קיים S' כך ש- $(S, S') \in R$ ו- $S' \vDash f$

ניסוח חלופי: $S \vDash EXf \Leftrightarrow$ קיים מסלול $\pi = S_0 S_1 ...$ ממצב $S = S_0$ כך שמתקיים $S_1 \vDash f$.

נשים לב כי f היא נוסחת מצב. ב-CTL* הביטוי $S_1 \vDash f$ היה מתפרש שונה.

- $S \vDash E(fUg) \Leftrightarrow$ קיים מסלול $\pi = S_0, S_1, ...$ מ- S כך שקיים $k \geq 0$ כך ש- $S_k \vDash g$ ולכל $0 \leq j < k$ $S_j \vDash f$.
- $S \vDash EGf \Leftrightarrow$ קיים מסלול π מ- S כך שלכל $i \geq 0$ $S_i \vDash f$

קיצורים:

- $AXf_1 \equiv \neg EX \neg f_1$
- $EFf_1 \equiv E(true\ Uf_1)$
- $AGf_1 \equiv \neg EF \neg G_1$
- $AFf_1 \equiv \neg EG \neg f_1$
- $A(f_1 U f_2) \equiv \neg\left(EG\neg f_2 \vee E\left(\neg f_2 U\left(\neg f_1 \wedge \neg f_2\right)\right)\right)$

המצבים עבורם $f_1 U f_2$ לא מתקיים (אף פעם לא מתקיים f_2, או שקיים מצב בו f_1 לא מתקיים לפני ש- f_2 התקיים)

דגשים:

- נשים לב שבמקרה של CTL יש צורך ב-3 אופרטורים טמפורליים כדי להגדיר את האופרטורים האחרים.
- קל לזהות נוסחת CTL על ידי המבנה שלה. ב-CTL הנוסחאות מורכבות מצמדים שהם הרכבה של כמת מסלול (E, A) ואופרטור טמפורלי (G, F, U, X), למשל: EF, AG וכו'.

דוגמא

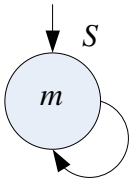
אילו מהמבנים הבאים הם מודלים לנוסחה, כלומר מספקים את הנוסחה?

הנוסחה היא:

$$AP = \{p, m, q\}$$
$$\varphi = A\left(pU\left(E\left(qUm\right)\right)\right)$$

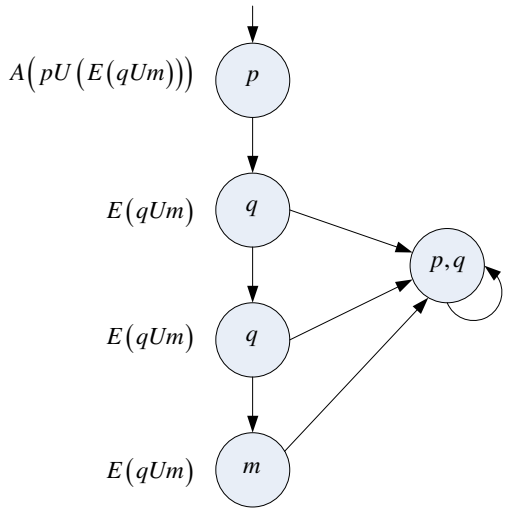
אבחנה: נשים לב שזו נוסחת CTL – האופרטורים מופיעים בה בזוגות.

מבנה ראשון:



המבנה מספק את הנוסחה. מתקיים $S \models m$ ומכאן $S \models E\left(qUm\right)$ ולכן מתקיים גם $S \models \varphi$.

מבנה שני:



המבנה מספק את הנוסחה. חשוב לשים לב שכל הנוסחאות הן נוסחת מצב - אפשר לסמן בכל מצב את הנוסחאות אותן הוא מספק.

LTL .6.3.5

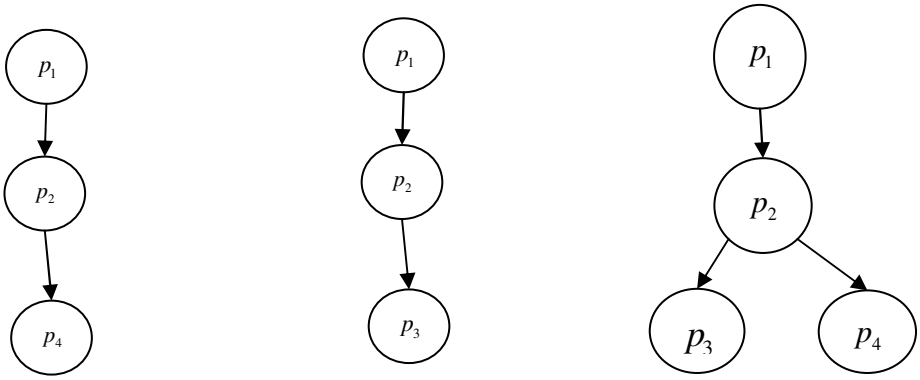
לוגיקה עיתית ליניארית Linear temporal logic: נוסחאות LTL הן מהצורה Af כאשר f נוסחת מסלול שבה נוסחאות המצב הן נוסחאות אטומיות בלבד. (אין כמתי מסלול ב-LTL מלבד ה- A החיצוני).

נוסחת מסלול LTL היא מהצורה:

- $p \in AP$
- אם f, g נוסחאות מסלול LTL, אז גם $\neg f, fUg, Xf, f \vee g$ הן נוסחאות מסלול LTL.

נוסחת LTL היא מהצורה Af כאשר f היא נוסחת מסלול LTL.

דוגמאות לנוסחאות מסלול LTL: $GFp, FGp, (Fb)Ua$.



Linear Time Logic Branching Time Logic

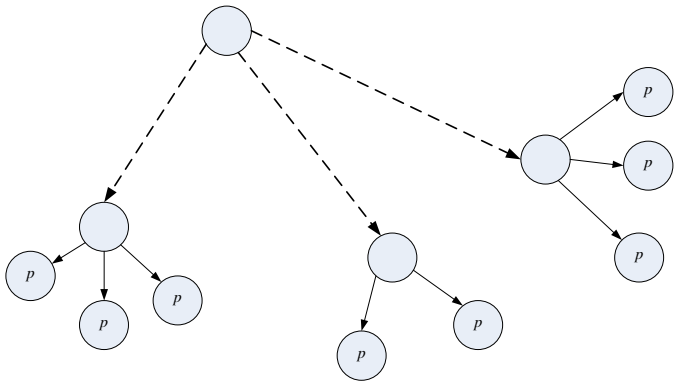
יש הבדל בין שתי צורות הבדיקה (הימנית לעומת שתי השמאליות): בצורה הימנית בנקודה p_2 עוד לא נעשתה ההחלטה לאן לנוע ואילו בצורות השמאליות ההחלטה כבר התקבלה.

- ב-CTL יש אפשרות בחירה בעתיד.
- ב-LTL אין אפשרות בחירה בעתיד.

דוגמא

נראה את ההבדל בין CTL ל-LTL על ידי דוגמא. נביט בביטוי ה-CTL הבא: $AFAXp$.

איך יראה המבנה המתאים לביטוי זה? המבנה יהיה מהצורה:



במבנה מסלולים באורכים שונים, ובכל מסלול שכזה קיים מצב שבו כל הבנים מספקים p.

הלוגיקה LTL לא מאפשרת לנו לבטא מבנה כזה. נוסחה קרובה ב-LTL הינה $AFXp$. הנוסחה אומרת כי בכל מסלול במבנה, נגיע למצב שבו המצב הבא מספק את p . עם זאת, אנחנו לא יכולים להבטיח ש- p יסתפק על כל הבנים של מצב כלשהו בהמשך.

6.3.6. זיהוי ביטויים מהלוגיקות השונות

- CTL – הביטויים בלוגיקה זו מורכבים תמיד מזוגות – כמת+אופרטור.
- LTL – ביטויים בלוגיקה זו מתחילים תמיד בכמת A, ולאחריו במהלך הביטוי אין כמתים נוספים.
- *CTL – הביטויים מתחילים בכמת ולאחריו אין הגבלות כלשהן. לחילופין הביטוי הוא מצב.

משפט (ללא הוכחה):

- ל- LTL ו- CTL כוח ביטוי בלתי ניתן להשוואה. בכל אחת מהן יש נוסחה שאין לה נוסחה שקולה בלוגיקה השניה.
- *CTL הינה בעלת כוח ביטוי גדול ממש גם מ- CTL וגם מ- LTL .

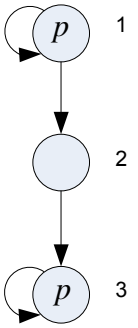
דוגמא נוספת להבדל בין CTL ל-LTL:

נראה כי $AFAGp \in CTL$ איננה שקולה לנוסחה $AFGp \in LTL$.

$AFAGp$ אומרת: קיימת נקודה בגרף החל ממנה AGp.

$AFGp$ אומרת: בכל המסלולים בגרף, קיימת נקודה החל ממנה מתקיים p.

כדי להראות שהנוסחאות אינן שקולות יש להראות מבנה המספק אחת אך אינו מספק את השניה:



המבנה מספק את $AFGp$ אך אינו מספק את $AFAGp$. המסלול הבעייתי הוא מסלול בו נלך על 1 אינסוף פעמים – אין סיפא שממנה והלאה p יתקיים.

משפט (ללא הוכחה):

בהנתן נוסחת CTL p, אם קיימת ב-LTL נוסחה q השקולה ל- p, נקבלה על ידי מחיקת כל הכמתים מ- p, והוספת A בתחילת הביטוי שנוצר.

6.4. אלגוריתם מפורש לבדיקת מודל CTL

בעיית בדיקת המודל: נתון מבנה $M=(S,S_0,R,L)$ ונוסחת CTL f.

- נחשב את קבוצת כל המצבים ב- M שמספקים את f : $S_f=\{s\in S \mid M,S\vDash f\}$
- נבדוק האם $S_0\subseteq S_f$ - כלומר האם קבוצת המצבים ההתחלתיים מוכלת ב- S_f.
- פלט האלגוריתם: אם קבוצת המצבים ההתחלתיים מוכלת ב- S_f אז נחזיר כי $M\vDash f$ ואחרת נחזיר $M\nvDash f$. תזכורת: מבנה M מספק נוסחה f ($M\vDash f$) אם כל אחד ממצביו ההתחלתיים מספק את הנוסחה.

אנו קוראים לתהליך "בדיקת מודל" כי אנחנו בודקים האם מבנה M הוא מודל לנוסחה f. ($M\vDash f$)

6.4.1. הבעיה והאלגוריתם

בדיקת מודל מפורשת (**Explicit model checking**) היא אלגוריתם המבצע בדיקת מודל, העובד ישירות על הגרף - מבנה קריפקה. המודל M נתון באופן מפורש כייצוג של הגרף – וממומש לרוב כמטריצת סמיכויות או כרשימת בנים.
האלגוריתם המפורש הוא קל להבנה, אך בעייתי למימוש עקב "בעיית התפוצצות המצבים": במערכת בה n משתנים בוליאנים, יהיו 2^n מצבים אפשריים למערכת. מספר המצבים השונים גדל מהר מאוד כאשר המערכת גדלה, וכמות הזכרון לא מספיקה לייצוג מלא של המודל.
כדי לפתור את בעיה זו נציג בהמשך 2 אלגוריתמים אחרים שהם אלגוריתמים סימבוליים – אנו נציג קידוד של המבנה M וגם של תוצאות הביניים המאפשר שימוש בפחות זכרון.

האלגוריתם לבדיקת מודל CTL מטפל בנוסחאות מהצורה: $EGf, EFf, E(fUg), EXf$

$a\in AP, f\vee g, \neg f$ כאשר f, g הן נוסחאות CTL. אם נרצה להתאים את האלגוריתם לאופרטורים נוספים – נצטרך לפתח עבורם אלגוריתמים, או לחילופין לתרגם את האופרטורים הנוספים לאופרטורים אלה.

<u>תיאור כללי של האלגוריתם</u>:

נצרף לכל מצב S משתנה $label(S)$ שיחזיק קבוצה של תתי נוסחאות של f שנכונות ב- S.

- <u>טיפול בתת נוסחה</u> g: $M,S \models g \Leftrightarrow f \in label(S)$
- <u>בסיום האלגוריתם</u> $M,S \models f \Leftrightarrow f \in label(S)$. נקבל את $S_f = \{S \mid f \in label(S)\}$ קבוצת המצבים שבהם הנוסחה נכונה.

האלגוריתם עובד באיטרציות על תתי הקבוצות של f מהנוסחאות האטומיות אל הנוסחאות המורכבות יותר, על כל תתי הקבוצות של f.

דוגמא לסדר הטיפול: עבור הנוסחה הבאה $f = (\neg b \vee c) \vee EG(EXa \vee \neg b)$

- <u>איטרציה 1</u>: נטפל בנוסחאות האטומיות: a, b, c
- <u>איטרציה 2</u>: $\neg b, EXa$
- <u>איטרציה 3</u>: $\neg b \vee c, EXa \vee \neg b$
- <u>איטרציה 4</u>: $EG(EXa \vee \neg b)$
- <u>איטרציה 5</u>: f

בעמודים הבאים נציג מה הפירוש של "טיפול בתת נוסחה g" שהוזכר בתיאור האלגוריתם.
בכל איטרציה g יהיה תת הנוסחה איתה אנו מתעסקים.

האלגוריתם לסימון:

צעד 0: **נוסחאות אטומיות** $(g \in AP)$

לכל $s \in S$, $label(s) := L(s)$.

סיבוכיות: $0(|S|)$

צעד זה הוא היחיד שלא קשור לנוסחה g בה אנו מטפלים כרגע. בתחילת ריצת האלגוריתם נסמן כל מצב בנוסחאות האטומיות המספקות אותו.

צעד 1:

סימון ב- $\neg g$

לכל $s \in S$:

$$if \quad g \notin label(s) \quad then \quad label(s) := label(s) \cup \{\neg g\}$$

סיבוכיות: $0(|S|)$

סימון ב- $g = f_1 \vee f_2$

לכל $s \in S$:

$$if \quad f_1 \in label(s) \quad or \quad f_2 \in label(s) \; then$$

$$label(s) := label(s) \cup \{f_1 \vee f_2\}$$

סיבוכיות: $0(|S|)$

סימון ב- $g = EXf_1$

לכל מצב s:

$$if \quad \exists s'((s,s') \in R \wedge f_1 \in label(s')) \; then \quad label(s) := label(s) \cup \{EXf_1\}$$

פירוש: קיימת קשת מצומת האב לצומת אחר, וגם מתקיים f_1 בצומת הבן.

סיבוכיות $0(|S|+|R|)$

האלגוריתם לסימון $g = E(f_1 U f_2)$

1. מצאו את כל המצבים שמסומנים ב- f_2 וסמנו אותם ב- g . נכניס את המצבים לקבוצה T ונסמן אותם כ-visit.

כלומר, לכל $s \in S$:

$$if \;\; f_2 \in label(s) \;\; then$$

$$label(s) := label(s) \cup \{g\}$$

$$visit(s) = true$$

$$T := T \cup \{s\}$$

2. <u>המטרה שלנו</u>: סריקה אחורנית של הגרף החל מכל מצב שסומן ב- g וסימון כל מצב שיש לו בן שמסומן ב- g והוא עצמו מסומן ב- f_1 , ב- g . אנו מבצעים סריקה אחורנית על מנת לא לסגור מעגל.

שלבי הפעולה:

a. נבחר $s \in T$ ונוציאו מ- T.

b. לכל t המקיים $R(t,s) \wedge \neg visit(t)$ (t אב של s אותו עדיין לא ביקרנו), אם מתקיים $f_1 \in label(t)$:

   i. $label(t) := label(t) \cup \{g\}$

   ii. $T := T \cup \{t\}$

c. עוצרים כאשר T ריקה.

ניסוח חלופי לאלגוריתם צעד-צעד להבהרת הפעולה:

$$if \;\; g \in label(s) \wedge (t,s) \in R \wedge f_1 \in label(t) \; then \;\; label(t) := label(t) \cup \{g\}$$

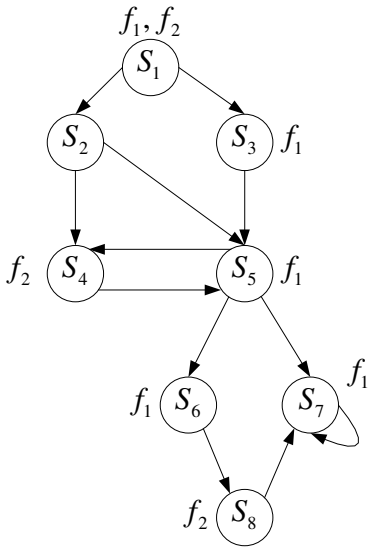
האלגוריתם מסתמך על השקילות הבאה: $E(f_1 U f_2) = f_2 \vee (f_1 \wedge EXE(f_1 U f_2))$

סיבוכיות האלגוריתם: $O(|S|+|R|)$

הערות על האלגוריתם:

1. מעגלים בגרף "לא נכנסים למשחק". אנחנו בוחנים רק מצבים שיש מהם דרך ל- f_2 .
(הסריקה האחורנית דואגת לכך).
2. הסימון ב-visit מונע סריקות מיותרות של העץ. על ידי סימון זה אנחנו משיגים את הסיבוכיות הליניארית של האלגוריתם.
3. בסוף האלגוריתם המצבים המסומנים הם מצבים שיש להם מסלול סופי אל f_2 וגם מקיימים את התנאי.

דוגמא:



שלב ראשון: מסומנים ב- f_2 : S_1, S_4, S_8 .

שלב שני: האבות שלהם: $\not{S_2}, S_5, S_6$

אנחנו לא מסמנים את S_2 כי הוא אינו מסומן ב- f_1 .

שלב שלישי: האבות של האבות: $\not{S_2}, \not{S_4}, S_3, \not{S_5}$

אנו לא מסמנים ולא מטפלים שוב במצבים שכבר סומנו (S_4, S_5).

ולבסוף: $\not{S_1}$

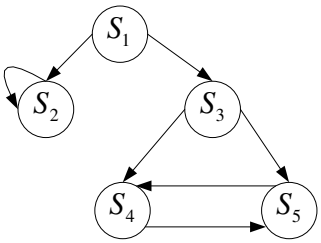
מתקבל בסוף: $S_{E(f_1 U f_2)} = \{S_1, S_3, S_4, S_5, S_6, S_8\}$

האלגוריתם לסימון $g = EGf_1$

הגדרות: **רכיב קשיר היטב: Strongly Connected Component – (SCC)** הינו תת גרף C, שבו מכל צומת יש מסלול לכל צומת ב-C, דרך צמתים ב-C.

- רכיב קשיר היטב הוא **טריוויאלי** אם הוא מורכב מצומת בודד ללא חוג עצמי.
- רכיב קשיר היטב הוא **מקסימאלי** (MSCC) אם אינו מוכל ממש ברכיב קשיר היטב אחר.
  תכונה חשובה של רכיבים מקסימליים: אם C_1, C_2 שניהם MSCC, אזי $C_1 \bigcap C_2 = \varnothing$.
- רכיבים קשירים היטב מקסימאליים הם זרים ולכן סכום הצמתים בהם הוא $|S|$. סכום הצמתים ברכיבים קשירים היטב לא בהכרח מקסימאליים הוא $2^{|S|}$.

דוגמא:



$$SCC = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{4,5\},\{3,4,5\}\}$$

- רכיבים קשירים היטב מקסימאליים: $\{\{3,4,5\},\{2\}\}$
- רכיבים קשירים היטב טריוויאליים: $\{\{1\},\{3\},\{4\},\{5\}\}$.

האלגוריתם של Tarjan מוצא את כל הרכיבים המקסימאליים בסיבוכיות $O(|S|+|R|)$ (כולל רכיבים טריויאליים).

נגדיר את המבנה המצומצם הבא: בהינתן $M = (S, R, L)$ נגדיר $M' = (S', R', L')$ ע"י:

- $S' = \{S \mid M, S \vDash f_1\}$
- $R' = (S' \times S') \bigcap R$ הערה: R' אינה בהכרח טוטאלית
- $L' = L \mid S' \leftarrow$ הטלה של L על S'

נסתמך על המשפט הבא:

משפט: $\Leftrightarrow M,s \vDash EGf_1$

1. $s \in S'$
2. קיים מסלול ב- M' ממצב s למצב t ברכיב קשיר היטב מקסימאלי לא טריוויאלי של M'.

המשפט מבטיח לנו שקיים מסלול אינסופי בו מסתפק f_1 (ולא רק רישא סופית שבה הוא מסתפק).

<u>האלגוריתם עצמו</u>:

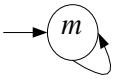
1. בונים $M' = (S', R', L')$ כמתואר לעיל. $O(|S|+|R|)$
2. מוציאים את כל הרכיבים הקשירים היטב המקסימאליים ב- M'. $O(|S|+|R|)$
3. כל מצב ברכיב קשיר היטב לא טריוויאלי יסומן ב- g. $O(|S|)$
4. מכל מצב שמסומן ב- g נבצע סקירה אחורנית ב- M' ונסמן כל מצב שממנו יש מסלול לרכיב קשיר היטב. $O(|S|+|R|)$

6.4.2. תרגילים

תרגיל 1

אילו מהמבנים הבאים הם מודלים לנוסחה: $\varphi = A(pU(EqUm))$?

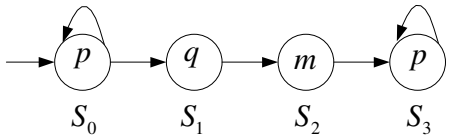
.1



המבנה מספק את הנוסחה.

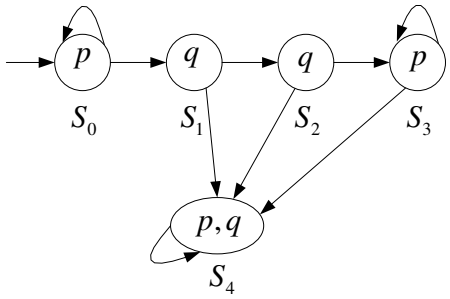
המצב היחיד מספק $EqUm$ כי הוא מספק את m ולכן הוא מספק את φ

.2



המבנה לא מודל לנוסחה. $S_1, S_2 \models EqUm$ אבל $S_0 \not\models \varphi$ בגלל המסלול $S_0 \to S_0 \to S_0 \to \ldots$.

.3



$P \models S_0, S_4$, $EqUm \models S_4, S_2, S_1$. כל מסלול עבור דרך S_0, S_1 ולכן כל מסלול מספק את φ.

תרגיל 2

נתונות הנוסחאות: $\varphi_2 = AXEXp$ $\varphi_1 = AXXp$. האם $\varphi_1 \Rightarrow \varphi_2$?

תשובה: כן.

יהיו M, S כך ש- $M, S \models \varphi_1$. לכל מסלול רק ש- $S_0 = S$ מתקיים $\pi \models XXp$.

$\Leftarrow$ יהי $\pi' = S'_0, S'_1 \ldots$ מסלול כלשהו ב- M שמתחיל ב- S כאמור $\pi' \models XXp$

$\Leftarrow$ $\pi'^1 \models Xp \Leftarrow S'_1 \models EXp$

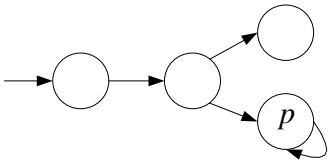
$\Leftarrow$ $\pi'^1 = EXp$

$\Leftarrow$ $\pi' \models XEXp$

הראינו זאת לכל π' שמתחיל ב- S ולכן $M, S \models AXEXp = \varphi$

האם $\varphi_2 \Rightarrow \varphi_1$?

תשובה: לא. דוגמה נגדית:



$M,S \models \varphi_2$ אבל $M,S \not\models AXXp$ בגלל המסלול העליון.

6.4.3. סיבוכיות בדיקת מודל

לוגיקה	CTL	LTL
סיבוכיות	$O\left(\lvert M\rvert \cdot \lvert f\rvert\right)$	$O\left(\lvert M\rvert \cdot 2^{\lvert f\rvert}\right)$

$\lvert M\rvert$ הוא מספר המצבים במבנה M ו- $\lvert f\rvert$ הוא אורך הנוסחה f.

חלק מהדרך להצגת הסיבוכיות של CTL הוצגה: האלגוריתם לסימון, כאשר אנחנו מתחילים לסמן את הנוסחאות האטומיות ועולים כלפי מעלה, ובכל שלב אנחנו מניחים שנוסחאות מהשלב הקודם ניתן לקבל את ערכן ב-$O(1)$ (מתקיים כי ערכן כבר חושב).

בדיקת מודל ב-LTL היא מעבר להיקפו של מסמך זה, והסיבוכיות מוצגת לידיעת הקורא בלבד.

לכאורה LTL בסיבוכיות יותר גבוהה מ-CTL אבל זה נובע רק מצורת הכתיבה. ניתן לרשום למשל:

עבור CTL: $O\left(2^{\lvert R(M)\rvert} \cdot \lvert f\rvert\right)$ כאשר $\lvert R(M)\rvert$ הוא מספר הביטים הדרושים לייצוג המבנה M,

עבור LTL: $O\left(2^{\lvert R(M)\rvert} \cdot 2^{\lvert f\rvert}\right)$

6.4.4. הוספת הוגנות ל-CTL

נגדיר שפה לוגית חדשה CTL^F בעלת אותו Syntax כמו CTL.

6.4.4.1. הגדרות ודוגמאות ראשונות

נגדיר **מבנה קריפקה עם הוגנות**: $M=(S,R,L,H)$

כאשר S,R,L כמו קודם ו- S_0 אופציונאלי, כמו קודם.

$$H=\{h_1,h_2,...,h_k\}\subseteq 2^S$$

- H הינה קבוצה של קבוצות מצבים המגדירה תנאי הוגנות.
- h_i - יכול להיות גם נוסחת CTL שמייצגת את קבוצת כל המצבים שמספקים אותה

מסלול π הוא **הוגן** אם הוא מבקר בכל h_i אינסוף פעמים.

<u>ניסוח שונה:</u>

$$\inf(\pi)=\{S \mid S=S_i \text{ for infinite } i\},\ \pi=S_0,S_1,S_2...$$

π הוא הוגן אם לכל $1\le i\le k$ $\quad \inf(\pi)\cap h_i\neq\varnothing$

<u>דוגמא:</u>

$$H=\{\{1,4\},\{1,6\},\{3\}\}$$

מסלול הוגן יבקר ב-$1,3$ אינסוף פעמים או שיבקר ב-$4,6,3$ אינסוף פעמים

<u>דוגמא:</u> מעגל חשמלי עם קלט input. נרצה לציין שהקלט יהיה 1 אינסוף פעמים.

$$H=\{input=1,...\}$$

<u>דוגמא:</u> 3 קלטים in_1,in_2,in_3 בכל אחד מהם 1 אינסוף פעמים (לאו דווקא באותו זמן).

$$H=\{in_1,in_2,in_3\}$$

דוגמא: בכולם אינסוף פעמים 1 (באותו זמן) $H=\{in_1 \wedge in_2 \wedge in_3\}$

נקודות:

- כאשר $H=\{\ \}$ כל המסלולים הוגנים.
- אם $H=\{\{\ \}\}$ אז אין מסלול הוגן.

6.4.4.2. סמנטיקה

הגדרת **ספיקות** של נוסחאות CTL^F נעשית מעל מבנה קריפקה הוגן $M,S \models_F \varphi$

הסמנטיקה זהה ל-$\models$, למעט כמתי מסלולים:

- $S \models_F EX\varphi_1 \Leftrightarrow$ קיים מסלול הוגן היוצא מ-S כך ש-$S_1 \models_F \varphi_1$
- $S \models_F AX\varphi_1 \Leftrightarrow$ לכל חישוב הוגן היוצא מ-S כך ש-$S_1 \models_F \varphi_1$
- $S \models_F E(\varphi_1 U \varphi_2) \Leftrightarrow$ קיים מסלול הוגן π וכן קיים $k \geq 0$ כך שמתקיים $\pi^k \models \varphi_2$ וכן $\pi^j \models \varphi_1$ לכל $0 \leq j < k$
- $S \models_F A(\varphi_1 U \varphi_2) \Leftrightarrow$ לכל מסלול הוגן π קיים $k \geq 0$ כך שמתקיים $\pi^k \models \varphi_2$ וכן $\pi^j \models \varphi_1$ לכל $0 \leq j < k$

נסמן $E_F\varphi$, $A_F\varphi$ כשנרצה להגיד כי "קיים מסלול הוגן" או כאשר "כל המסלולים ההוגנים היוצאים מהמצב ...".

יתכן שממצב S אין אף מסלול הוגן ואז:

- $S \models_F A\varphi$ לכל φ
- $S \not\models_F E\varphi$ לכל φ

6.4.4.3. אלגוריתם לחישוב קבוצת המצבים המספקים $E_F G\varphi_1$

רכיב קשיר היטב הוגן – רכיב קשיר היטב C הינו הוגן אם לכל $h_i \in H$ $C \bigcap h_i \neq \varnothing$

נבנה $M' = (S', R', L', H')$ באופן הבא (נשים לב כי S', R', L' מוגדרים כמו קודם):

$$M' = (S', R', L', H')$$
$$S' = \{S | M, S \models_F \varphi_1\}$$
$$R' = (S' \times S') \bigcap R$$
$$L'(s) = L(s), s \in S'$$
$$H' = \left\{ \underbrace{h_i \bigcap S'}_{h_i'} | h_i \in H \right\}$$

הערה: אם $h_i' = \varnothing$ נסיק כי M מסלול הוגן המספק את $G\varphi$ (אין מסלולים הוגנים).

<u>למה</u>: $M, s \models_F EG\varphi_1$ אמ"מ מתקיימים 2 התנאים:

1. $s \in S'$
2. יש מסלול ב- M' מ- s למצב t ברכיב קשיר היטב מקסימאלי, לא טריוויאלי והוגן.

טיפול ב- $E_F G\varphi_1$

<u>האלגוריתם:</u>

1. בנו M' $|H|+|R|+|S|$
2. מצאו את הרכיבים הקשירים היטב המקסימאליים, כולל הטריוויאלים $|S|+|R|$
3. מצאו את הרכיבים הקשירים היטב המקסימאליים, הלא טריוויאליים וההוגנים. $|S| \cdot |H|$
4. סמנו את כל המצבים מ-3 ב- φ_1 וכן בסריקה אחורנית סמנו את כל המצבים שמהם ישיגים מצבים שמסומנים ב- φ_1. $|S|+|R|$

סיבוכיות כוללת: $0(|S| \cdot |H| + |R|)$

6.4.4.4 טיפול ב- $E_F X\varphi_1$, $E_F\left(\varphi_1 U \varphi_2\right)$

נוסיף נוסחה אטומית חדשה $fair$, שנכונה במצב S אמ"ם מ- S יוצא מסלול הוגן.

$S \models_F EG\ true$ (נוסחה שקולה $S \models E_F G\, true$) אמ"מ קיים מסלול הוגן π מ- S המספק $G\ true$

דרך נוספת להגדיר את $fair$:

$$fair \triangleq E_F G\ true$$

אבחנות:

- אם π מסלול הוגן אז כל סיפא של π היא מסלול הוגן.
- אם π הוגן אז הוספת רישא סופית להתחלת π $\pi' = \pi_1 \cdot \pi$ גם כן חישוב הוגן.

בעזרת $fair$ אנו מבצעים באלגוריתמים הבאים רדוקציה מ-CTLF ל-CTL ופותרים את הבעיה בכלים שאנו מכירים:

אלגורתים לטיפול ב- $E_F X\varphi_1$

כדי לסמן את המצבים שמספקים $E_F X\varphi_1$ נבדוק (ללא הוגנות) $EX\left(\varphi_1 \wedge fair\right)$

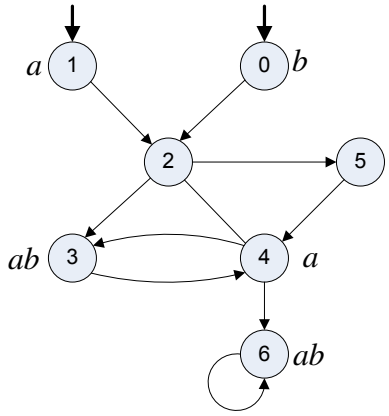
אלגוריתם לטיפול ב- $E_F\left(\varphi_1 U \varphi_2\right)$

כדי לסמן את המצבים שמספקים $E_F\left(\varphi_1 U \varphi_2\right)$ נסמן (ללא הוגנות) $E\left(\varphi_1 U\left(\varphi_2 \wedge fair\right)\right)$

סיבוכיות עבור $EfU\ \ E_F x$: $0\left(|S|\cdot|H|+|R|\right)$

6.4.4.5. דוגמא

נביט במבנה הבא ובתנאי ההוגנות הנתון:



$$H = \{\{1,2\},\{1,5\}\}$$

חישוב fair - $fair = E_F G\ true$

- בניית M': מתקיים כי $M' = M$. נשתמש בהגדרה - קבוצת המצבים המספקים את $\varphi_1 = true$ היא כל המצבים.
- מציאת הרכיבים הקשירים היטב המקסימלים:

$$MSCC = \{\{2,3,4,5\},\{6\},\{0\},\{1\}\}$$

כאשר $\{0\},\{1\}$ הם רכיבים טריויאליים.

- מציאת הרכיבים הקשירים היטב ההוגנים:

$$FMSCC = \{\{2,3,4,5\},\{1\}\}$$

- מציאת fair:

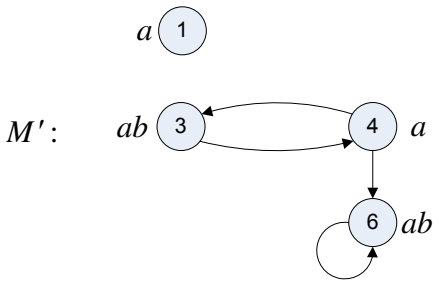
$$fair = \{2,3,4,5,0,1\} = S - \{6\}$$

חישוב $E_f(aUb)$

- על מנת לבדוק את הנוסחה נבדוק $E(aU(b \wedge fair))$.
- מתקיים $b \wedge fair = \{0,3\}$
- הולכים אחורנית למצבים המסומנים ב-a, ונקבל: $E(aU(b \wedge fair)) = \{0,3,4\}$

חישוב $E_f Ga$

- נחשב את M' - קבוצת המצבים המספקים את a והקשתות ביניהם:



- נמצא MSCC:

$$MSCC = \{\{3,4\},\{6\},\{1\}\}$$

- נמצא FMSCC:

$$FMSCC = \varnothing$$

6.5. *Binary Decision Diagram - BDD*

BDD הוא מבנה נתונים לייצוג פונקציה בוליאנית בצורה לשפעמים מצומצמת בזכרון.

פונקציה בוליאנית: $x_1,...,x_n,x_{n+1}\in\{0,1\}, f(x_1,...,x_n)=x_{n+1}$

יתרונות BDD לייצוג פונקציות בוליאניות:

1. לעיתים קרובות מקבלים ייצוג מצומצם בזיכרון.
2. ביצוע יעיל (בגודל ה-BDD) של פעולות בוליאניות.
3. ייצוג קנוני – ל-2 פונקציות בוליאניות שקולות יהיה ייצוג BDD זהה, ולכן מאפשר לבדוק שקילות בקלות.

BDD מאפשר לנו לייצר באופן סימבולי את המודל, ולטפל בבעיות "מהעולם האמיתי". מעט היסטוריה: בתחילת שנות ה-80 השיטות לבדיקת מודל שפותחו היו שיטות מפורשות שהתאימו רק לבעיות עם מספר מצומצם של משתנים – "בעיות צעצוע". החל משנות השמונים, במשך שנים רבות, כל כלי בדיקת המודל היו מבוססים על שימוש ב-BDDs.

6.5.1. תיאור קבוצה ע"י פונקציה בוליאנית

טענה: כל קבוצה ניתנת לייצוג על ידי פונקציה בוליאנית.

תהי f מהצורה $f:\{0,1\}^K\to\{0,1\}$, כך ש-$f(x_1...x_x)=X_{k+1}$ ו-$X_1...X_k,X_{k+1}\in\{0,1\}$.

בהינתן קבוצה U ותת קבוצה A, הפונקציה האופיינית של A שתסומן f_A הינה:

$$f_A(u)=\begin{cases}1 & u\in A\\ 0 & u\notin A\end{cases}$$

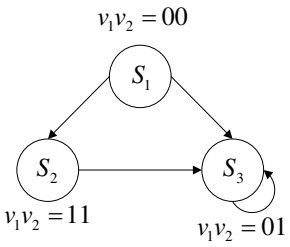
נקודד את איברי U ע"י סדרות של $0,1$ (באורך $\log|U|$) ונסמן ב-$\overline{u}$ את הקידוד ל-u.

$$f_A(\overline{u})=\begin{cases}1 & u\in A\\ 0 & u\notin A\end{cases}$$

6.5.2. יצוג מבנה קריפקה על ידי פונקציה בוליאנית - דוגמא

<u>קידוד של מצבים</u>

נשתמש במשתנים v_1, v_2 כדי לייצג קידוד של מצבים. לדוגמא:



$$f_{\{S_1\}}(v_1, v_2) = \neg v_1 \wedge \neg v_2$$

נוסחה f_A **מייצגת** קבוצה A אמ"מ ההשמות המספקות ל- f_A מתאימות בדיוק לקידוד של איברי A.

$$f_{\{S_3\}}(v_1, v_2) = \neg v_1 \wedge v_2$$

$$f_{\{S_1, S_3\}}(v_1, v_2) = (\neg v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) = \neg v_1$$

$$f_{\{S_2\}}(v_1, v_2) = v_1 \wedge v_2$$

<u>ייצוג רלציית המעברים</u>

אם נתונים המשתנים $v_1, \ldots v_n$ לתיאור מצב אזי נוסיף משתנים $v_1', \ldots v_n'$ לתיאור "המצב הבא" (המצב הסופי ברשת). בדוגמא:

$$f_{\{S_1 \to S_2\}}\{v_1, v_2, v_1', v_2'\} = \underbrace{\neg v_1 \wedge \neg v_2}_{S_1} \wedge \underbrace{v_1' \wedge v_2'}_{S_2}$$

ונוכל אף לייצג את כל הרלציה:

$$f_R(v_1, v_2, v_1', v_2') = (\neg v_1 \wedge \neg v_2 \wedge v_1' \wedge v_2') \vee (\neg v_1 \wedge \neg v_2 \wedge \neg v_1' \wedge v_2') \vee$$

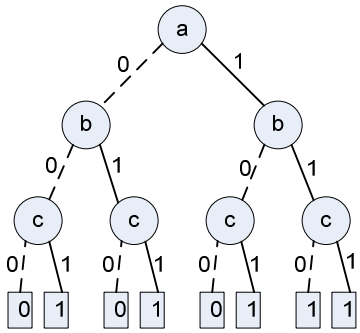
$$(v_1 \wedge v_2 \wedge \neg v_1' \wedge v_2') \vee (\neg v_1 \wedge v_2 \wedge \neg v_1' \wedge \neg v_2')$$

הערה: כל השמה שמספקת את f_R מספקת בדיוק אחת מהפסוקיות שבה.

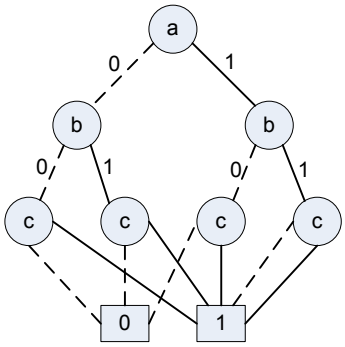
6.5.3. BDD - דוגמא ראשונה

נתחיל להציג את נושא ה-BDD על ידי דוגמא. נתחיל בעץ החלטה בינארי ונניח שמטפלים בנוסחה $(a \wedge b) \vee c$. סימונים: קו שלם מסמן ערך 1 למשתנה, קו מרוסק מסמן ערך 0 למשתנה.

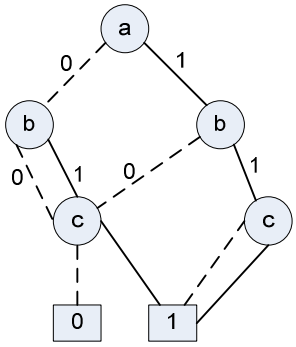
הגרף הינו גרף מכוון ללא מעגלים (Directed – DAG Acyclic Graph).

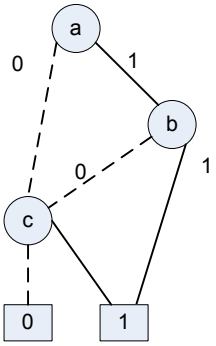


<u>רדוקציה ראשונה:</u> איחוד עלים לשני מצבים - אפס ואחד.



<u>רדוקציה שניה:</u> איחוד תתי עצים איזומורפיים.





<u>רדוקציה שלישית</u>: הורדת משתנים מיותרים (שאין להם השפעה על התוצאה) למשל בדוגמה שלנו המשתנה c הימני הינו מיותר כי גם הענף הימני שלו וגם השמאלי מובילים ל- 1 ולכן אין לו השפעה על התוצאה. כך גם המשתנה b השמאלי.

לאחר הרדוקציות הנ"ל יש בידינו BDD.

חשוב לדעת כי במימושים הקיימים בפועל לא בונים את העץ ואז מצמצים אותו ל-BDD, אלא בזמן בניית העץ בונים אותו באופן מצומצם על מנת לחסוך בזכרון.

6.5.4. BDD כמבנה נתונים

BDD עבור פונקציה בוליאנית $f\left(x_1...x_n\right)$ הינו גרף מכוון חסר מעגלים עם שורש ושני סוגי צמתים: צמתי קצה וצמתים פנימיים. שורש העץ יכול להיות צומת קצה או צומת פנימי.

לצומת פנימי v יש את השדות הבאים:

- משתנה $\mathrm{var}\left(v\right)$
- שני מצביעים: $low\left(v\right), high\left(v\right)$.

לצומת קצה יש ערך בודד: $value\left(v\right)\in\left\{0,1\right\}$

על מנת להבטיח יצוג קנוני נוסיף את התנאים הבאים:

- כל משתנה מופיע פעם אחת לכל היותר על כל מסלול מהשורש לעלה.
- נגדיר סדר למשתנים. המשתנים מופיעים באותו סדר לאורך כל מסלול מהשורש לעלה (לא בהכרח כולם מופיעים). מוסכמה – הסדר הינו סדר עולה מהשורש לעלים.
- הגרף אינו מכיל תתי גרפים איזומורפיים או צמתים מיותרים (שאין תלות בערך המשתנה בהם על המסלול המסוים).

עם מגבלות אלו מתקבל Reduced Ordered BDD (ROBDD). בהמשך, כאשר נדבר על BDD נתכוון תמיד ל-ROBDD.

עבור ROBDD בהינתן סדר מסויים של המשתנים, לכל הפונקציות השקולות זו לזו יש ייצוג זהה עד כדי איזומורפיזם.

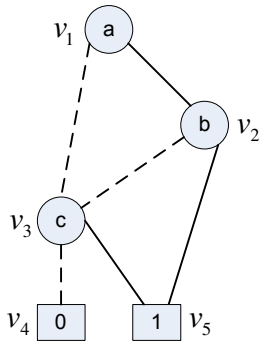
<u>בהינתן BDD – איזו פונקציה הוא מגדיר:</u>

בהינתן BDD עם שורש v הוא מגדיר פונקציה $f_v\left(x_1,...x_n\right)$ באופן הבא:

1. אם v צומת קצה - $f_v\left(x_1...x_n\right)=value\left(v\right)$
2. אם v אינו צומת קצה ואם $var\left(v\right)=x_i$ אזי:

$$f_v\left(x_1...x_n\right)=\left(\neg x_i\wedge f_{low(v)}\left(x_1...x_n\right)\right)\vee\left(x_i\wedge f_{high(v)}\left(x_1...x_n\right)\right)$$

נמשיך עם שרטוט הדוגמא:



נחשב את f_v עבור ה-BDD:

מתקיים: $\mathrm{var}(v_1)=a, \mathrm{var}(v_2)=b, \mathrm{var}(v_3)=c$ וכמו כן $value(v_4)=0, value(v_5)=1$.

$$f_{v_4}(a,b,c)=0$$

$$f_{v_5}(a,b,c)=1$$

$$f_{v_3}(a,b,c)=\left(\neg c \wedge f_{v_4}(a,b,c)\right)\vee\left(c \wedge f_{v_5}(a,b,c)\right)=\left(\neg c \wedge 0\right)\vee\left(c \wedge 1\right)=c$$

$$f_{v_2}(a,b,c)=\left(\neg b \wedge c\right)\vee\left(b \wedge 1\right)=b\vee c$$

$$f_{v_1}(a,b,c)=\left(\neg a \wedge c\right)\vee\left(a \wedge \left(b\vee c\right)\right)=\left(a \wedge b\right)\vee c$$

<u>עובדות על BDD:</u>

- כל צומת ב-BDD מייצג איזשהו "זכרון".
- לא לכל פונקציה בוליאנית קיים BDD "קטן" פולינומיאלית במספר משתני הפונקציה. יש 2^{2^n} פונקציות בוליאניות עם n משתנים, ויש הרבה פחות DAG בגודל פולינומיאלי ב-n.
- לפונקצית הכופל אין BDD פולינומיאלי לכל סדר שהוא.

פונקצית הכופל היא הפונקציה המקבלת שני מספרים a,b ביצוג בינארי: $b_1...b_n, a_1...a_n$ ומחזירה את המכפלה שלהם c ביצוג $c_1...c_{2n}$.

- מציאת הסדר האופטימלי של המשתנים ב-BDD היא בעיית NP-קשה.

6.5.5. פעולות על BDDs:

6.5.5.1. פעולת צמצום – (Reduce)

בהינתן BDD שאינו מצומצם נפעיל את הפעולות הבאות לקבלת BDD מצומצם. במבט ראשון נראה שאין צורך בפעולה כזו, הרי הגדרנו ש-BDD הוא כבר ביצוג מצומצם. הצורך נובע כאשר אנו מבצעים פעולות על BDD. לאחר ביצוע פעולה יתכן שקיבלנו BDD שאינו מצומצם, ולכן יש צורך בפעולת reduce.

א. ביטול צמתי קצה כפולים:
נשאיר צומת קצה בודד עם ערך 0 ואחד עם ערך 1, ונפנה את כל ההצבעות בהתאם.

ב. ביטול צמתים פנימיים כפולים:
בהינתן צמתים u, v כך שמתקיים

$$\mathrm{var}(v) = \mathrm{var}(u), high(v) = high(u), low(v) = low(u)$$

אזי מבטלים את אחד הצמתים ומפנים את כל ההצבעות שהיו אליו אל הצומת שנשאר.

ג. ביטול צמתים מיותרים $low(v) = high(v)$
נבטל את הצומת v ונפנה ההצבעות אליו אל הבן היחיד שלו.

טענה: הפעלת 3 הפעולות שהוצגו בסדר כלשהו עד שלא ניתן להפעילן יותר תיתן כתוצאה BDD מצומצם.

אלגוריתם reduce: עובד מהעלים כלפי מעלה ומצמצם את ה-BDD בזמן שהוא ליניארי בגודל ה-BDD.

6.5.5.2 פעולת הגבלה - restriction

$f|_{x_i:=b}(x_1,...x_n) = f(x_1,...,x_{i-1},b,x_{i+1},...,x_n)$ When $b \in \{0,1\}$

הפעולה מציבה ערך קבוע לאחד המשתנים של הפונקציה.

בהינתן BDD המתאר את f נבנה BDD עבור $f|_{x_i:=b}$ באופן הבא:

1. נבצע DFS על ה-BDD של f.
2. לכל צומת v שיש לו בן u כך ש $\text{var}(u) = x_i$ נשנה ההצבעה מ-v אל u אל $low(u)$ אם $b=0$ ול-$high(u)$ אם $b=1$.
3. בסוף התהליך - נבצע reduce.

דוגמא:

$$f(x_1,x_2,x_3) = (x_1 \wedge x_2) \vee x_3$$

$$f(x_1,x_2,x_3)|_{x_2=0} = (x_1 \wedge 0) \vee x_3 = x_3$$

נשים לב שניתן לממש את הכמתים בעזרת ההצבות. נגדיר את הכמתים "קיים" ו-"לכל":

קיים	$\exists x : f$	$\equiv$	$f|_{x=0} \vee f|_{x=1}$
לכל	$\forall x : f$	$\equiv$	$f|_{x=0} \wedge f|_{x=1}$

תזכורת – אלגוריתם DFS:

```
dfs-visit (Graph G, Vertex u)
{
      the vertex u is painted gray

      for all white successors v of u
      {
          dfs-visit(G, v)
      }

      u is painted black
}


dfs (Graph G)
{
      all vertices of G are first painted white

      dfs-visit(G, root of G)
}
```

6.5.5.3. פעולות בוליאניות על BDD (Apply)

בהינתן שני BDD: f, f' נרצה לחשב את ה-BDD של $f \circ f'$ כאשר $\circ$ הוא אחת מ-16 הפונקציות הבינאריות. סימונים: v, v' הם השורשים של f, f' בהתאמה. אם v, v' אינם צמתי קצה אז $\mathrm{var}(v) = x$ וכן $\mathrm{var}(v') = x'$.

נסתמך על **"הרחבת Shonon"**: $f = (\neg x \wedge f|_{x=0}) \vee (x \wedge f_{|x=1})$

Apply עובדת לפי ארבעה מקרים שונים: (חישוב $f \circ f'$):

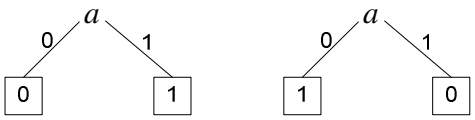
1. אם v, v' הם צמתי קצה אז תוצאת הפעולה היא $f \circ f' = value(v) \circ value(v')$ וה-BDD הוא גם צומת יחיד v'' כך שמתקיים $value(v'') = value(v) \circ value(v')$
2. אם $\mathrm{var}(v) = \mathrm{var}(v') = x$ אז $f \circ f' = (\neg x \wedge (f|_{x=0} \circ f'|_{x=0})) \vee (x \wedge (f|_{x=1} \circ f'|_{x=1}))$.

בניית ה-BDD: צומת חדש עם משתנה v'' כך שיתקיים:

a. $low(v'')$ יצביע אל ה-BDD: $f|_{x=0} \circ f'|_{x=0}$

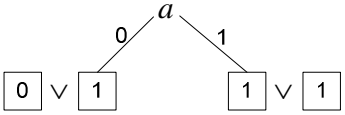
b. $high(v'')$ יצביע אל ה-BDD: $f|_{x=1} \circ f'|_{x=1}$.

דוגמא:



$f(a) = a$ $\quad$ $f'(a) = \neg a$

נחשב את $f \vee f'$:



ואחרי צמצום:

3. אם $x=\mathrm{var}(v)<\mathrm{var}(v')$ כאשר היחס '$<$' מוגדר על סדר נתון, (כלומר $\mathrm{var}(v)$ לא מופיע ב f' או בניסוח אחר: $f'=f'|_{x=0}=f'|_{x=1}$) אז:

$$f\circ f'=\left(\neg x\wedge\left(f|_{x=0}\circ f'\right)\right)\vee\left(x\wedge\left(f|_{x=1}\circ f'\right)\right)$$

4. באופן דומה ל-3 נטפל במצב בו $x=\mathrm{var}(v)>\mathrm{var}(v')$.

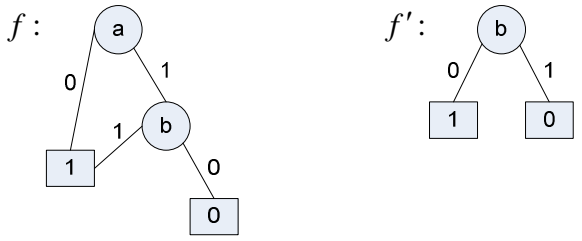
<u>דוגמא</u>

יהיו 2 פונקציות f,f' מעל המשתנים a,b. נקבע את סדר המשתנים להיות $a<b$.

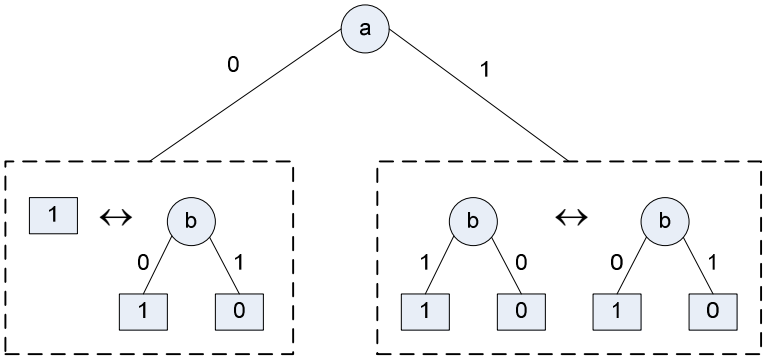
הפונקציות עצמן הינן:

$f:a\to b$

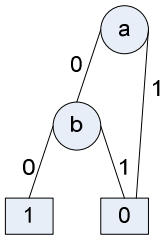
$f':\neg b$



נרצה לחשב את הפונקציה $f\leftrightarrow f'$. המקרה כאן הוא מקרה 3.



ומכאן נקבל:



אנו מקבלים כי:

$$(a \to b) \leftrightarrow \neg b = \neg a \wedge \neg b$$

ניתן להגיע לאותו הפיתוח באמצעות שימוש בלוגיקה, לבדיקה:

$$f \leftrightarrow f' \equiv (a \to b) \leftrightarrow \neg b \equiv (\neg a \vee b) \leftrightarrow \neg b \equiv \neg a \wedge \neg b$$

<u>סיבוכיות פעולה APPLY</u>

טיפול פשטני נותן לנו פתרון אקספוננציאלי בגודל ה-BDD. כדי לפתור את הבעיה ביעילות נשים לב כי כל צומת ב-BDD מייצג פונקציה, ומספר הפונקציות השונות שמיוצגות הוא כמספר הצמתים ב-BDD.

נשתמש בטבלת HASH: לכל הפעלה של APPLY יש מצביעים לצמתים ב-BDD שמהם הפעולה הופעלה, וכן את הפעולה שהופעלה. לא נבצע את אותו חישוב פעמיים אלא נשתמש בתוצאות שכבר חישבנו.

כעת הסיבוכיות תלויה בכמות פעולות ה-APPLY השונות שמבצעים במהלך תהליך APPLY על $f \circ f'$. נסמן ב-$|f|$ את מספר הצמתים ב-BDD של f ומכאן: מספר הפעולות הוא $|f| \cdot |f'|$.

6.5.5.4. פעולת not

נתון BDD לייצור פונקציה $f(\bar{v})$ ורוצים לבנות BDD עבור $g(\bar{v}) = \neg f(\bar{v})$.

1. נבצע זאת על ידי החלפה בין צמתי הקצה: $0 \to 1, 1 \to 0$.
2. דרך נוספת: שימוש ב-APPLY על $f \to false$.

6.5.6. יצוג מבנה קריפקה בעזרת BDD

BDD זו דרך שימושית לייצג יחסים על פני תחום סופי. נראה כיצד ניתן לייצג את מכונת קריפקה באמצעות BDD. שיטת העבודה בה נשתמש: נניח ש-Q זו רלציה n-ממדית מעל $\{0,1\}$ אזי Q יכולה להיות מיוצגת על ידי ה-BDD עם הפונקציה האופיינית הבאה:

$$f_Q(x_1,...,x_n)=1 \Leftrightarrow Q(x_1,...,x_n)$$

אחרת תהא Q רלציה n-ממדית מעל תחום סופי D. ללא הגבלת כלליות, נניח שב-D ישנם 2^m אלמנטים ($m>1$). על מנת לייצג את Q כ-BDD, נקודד את האלמנטים של D על ידי הטרנספורמציה $\phi:[0,1]^m \to D$ שממפה כל ווקטור בוליאני באורך m אל אלמנט ב-D. על ידי שימוש בקידוד ϕ נוכל לבנות רלציה בינארית $\hat{Q}$ כך:

$$\hat{Q}(\tilde{x}_1,...,\tilde{x}_n)=Q(\phi(\tilde{x}_1),...,\phi(\tilde{x}_n))$$

כאשר $\tilde{x}_i$ הוא ווקטור של m משתנים בוליאניים שמקודדים את המשתנה x_i שהינו ערך מ-D.

מכונת קריפקה כאמור הינה $M=(S,R,L)$. כדי לייצג את המכונה במלואה צריכים להציג את הקבוצה S, את הרלציה R ואת פונקצית הסימון L.
כדי לייצג את S צריכים פונקציית קידוד. לצורת הנוחות נניח 2^m מצבים, ואז כמו עבור הרלציה נגדיר פונקצית מיפוי בוליאנית $\phi:[0,1]^m \to S$. ה-BDD המתאים מורכב בצורה דומה לקודם.

הרלציה R מורכבת מאותו הקידוד, כאשר אנחנו משתמשים בשני סטים מקבילים של משתנים בוליאניים כדי להציג את המצב לפני ואת המצב לאחר המעבר $\hat{R}(\tilde{x},\tilde{x}')$.

בעניין פונקצית המיפוי L, למרות שהפונקציה מוגדרת בתוך מיפוי מקבוצת המצבים אל תת קבוצה של הנוסחאות האטומיות, יותר נוח לנו להסתכל על הפונקציה כמיפוי מנוסחאות אטומיות לתת קבוצה של מצבים. הנוסחה האטומית p ממופה לקבוצת המצבים המספקים אותה:
$\{s \mid p \in L(s)\}$. אם נסמן קבוצה זו ב-L_p, נוכל להציג אותה באמצעות הקידוד ϕ כמו שהצגנו קודם. באופן כזה, נוכל להציג כל אחד מהביטויים האטומיים. כלומר:
לכל נוסחה אטומית $a \in AP$ נקבל את ה-BDD שמייצג את קבוצת המצבים המספקים את a ונסמנו ב-$a(\bar{v})$.

6.6. בדיקת מודל סימבולית מבוססת BDD

6.6.1. מבוא והגדרות

בדיקת מודל סימבולית עוסקת באלגוריתמים המשתמשים ביצוג של מכונת קריפקה כ-BDD על מנת לבצע בדיקות מודל. אלגוריתם בדיקת המודל נקרא **סימבולי** מכיוון שהוא מבוסס על מניפולציות מתמטיות על הנוסחאות הבוליאניות המייצגות את המכונה.
BDD מייצג תמיד קבוצות של מצבים ומעברים, ולכן הפעולות שנציג יפעלו על קבוצות שלמות ולא על מצבים בודדים.

אנחנו עובדים על מבנה קריפקה $M\left(S,R,L\right)$ ונוסחה f. המטרה שלנו היא להחזיר את קבוצת המצבים המספקים את f.

את M נקבל באופן הבא (בדומה למה שהצגנו בסעיף על יצוג מכונת קריפקה):

- רלציית מעברים R שנתונה על ידי BDD מייצג $R\left(\overline{v},\overline{v}'\right)$. ($\overline{v}$ מייצג את המצב הנוכחי, $\overline{v}'$ מייצג את המצב הבא).
- L: לכל נוסחה אטומית $a\in AP$ נקבל את ה-BDD שמייצג את קבוצת המצבים המספקים את a ונסמנו ב-$a\left(\overline{v}\right)$.

כמו כן, לפעמים נתונים גם:

- קבוצת המצבים כולה $S\left(\overline{v}\right)$. לא חובה במקרה זה כי לעתים $R\left(\overline{v},\overline{v}'\right)$ מספיק לצרכים.
- קבוצת המצבים ההתחלתיים $S_0\left(\overline{v}\right)$.

באלגוריתם הלא סימבולי "טיפלנו" בכל פעם בתת נוסחה של f עד אשר הגענו לטיפול בנוסחה כולה. האלגוריתם הסימבולי עובד בצורה דומה:

- באלגוריתם הלא סימבולי הראנו כי בהנתן g תת נוסחה של f (הפונקציה הנבדקת), מתקיים כי $\forall s\in S, g\in label\left(s\right)\Leftrightarrow s\vDash g$.
- באלגוריתם הסימבולי "הטיפול" ב-g יהיה כך: נבנה BDD שייצג את קבוצת כל המצבים המספקים את g (תמיד בעבודה עם BDD האלגוריתמים יעבדו על קבוצות מצבים ולא על מצבים בודדים).

בהמשך נציג אלגוריתמים לבדיקת מודל מבוססת BDD. כאשר נכתוב אלגוריתמים נוספים משלנו חשוב לא לבנות אלגוריתם המבוסס BFS או DFS. אלגוריתמים אלה עוברים בכל פעם על מצב בודד, ואלו פעולות מאוד לא יעילות כשאנו מתעסקים עם BDD. האלגוריתמים בהם נשתמש צריכים להיות מבוססים על פעולות על קבוצות כדי לנצל את מבנה הנתונים בצורה הטובה ביותר.

6.6.2. פעולות על קבוצות

פעולות סטנדרטיות:

- איחוד, חיתוך, השלמה, השוואה וכדו'.

פעולות נוספות:

- $\exists x_i, f\left(x_1,...,x_n\right)$ - מחשבים האם יש הצבה של x_i הנותנת לפונקציה ערך אמת.
- $\forall x_i, f\left(x_1,...,x_n\right)$ - מחשבים האם כל הצבה של x_i נותנת לפונקציה ערך אמת.

נשים לב כי מתקיים:

$$\exists x_i, f\left(x_1,...,x_n\right) = f\left(x_1,...,x_n\right)_{x_i=1} \vee f\left(x_1,...,x_n\right)_{x_i=0}$$

$$\forall x_i, f\left(x_1,...,x_n\right) = f\left(x_1,...,x_n\right)_{x_i=1} \wedge f\left(x_1,...,x_n\right)_{x_i=0}$$

כלומר, הפעולות הנוספות לא נותנות לנו עוד כוח ביטוי. הפעולות הנוספות נותנות לנו סימונים נוחים לכתוב פעולות כשנזדקק להן.

דגש נוספת לגבי פעולות אלו הינו שאלו פעולות יקרות יחסית לאיחוד, חיתוך ושאר הפעולות הסטנדרטיות. למשל, חישוב של $\exists x_1, x_2, .., x_n : f\left(x_1,...,x_n\right)$ חסום על ידי 2^n פעולות. (נגיע לחסם זה על ידי שנשים לב שעבור כל x_i מבצעים 2 פעולות, ומבצעים מכפלה n פעמים עבור כל האפשרויות ל- x_i השונים.

6.6.3. בדיקת מודל סימבולית לפי מבנה הנוסחה

אנחנו עובדים על מבנה קריפקה $M(S,R,L)$ ונוסחה f. האלגוריתם מטפל בתתי הנוסחאות של f מהפשוטה למורכבת, וכשמטפלים בתת נוסחה g מניחים שכל תתי הנוסחאות של g כבר טופלו. **נוסחה טופלה** כאשר קיים BDD המייצג את קבוצת המצבים המספקים אותה.

נחלק את בדיקת המודל למקרים שונים. המקרים נבדלים ביניהם על פי מבנה הנוסחה.

1. $\boldsymbol{f = a \in AP}$: נשתמש ב-$a(\overline{v})$ הנתון. נחזיר $f(\overline{v}) = a(\overline{v})$.
2. אם $\boldsymbol{f = f_1 \wedge f_2}$ (וכבר חישבנו BDDs $f_1(\overline{v})$ עבור f_1 ו-$f_2(\overline{v})$ עבור f_2), אזי ה-BDD עבור f יסומן $f(\overline{v})$ ויתקבל על ידי הפעלת $\wedge$ על ה-BDDs של f_1, f_2.

<u>הסבר</u>: למה החישוב הזה נכון?

ה-BDD $f_1(\overline{v})$ מייצג את הפונקציה האופיינית של קבוצת המצבים המספקים את f_1 (נסמן קבוצה זו ב-S_{f_1}). נכתוב:

$$f_1(\overline{v}) = \begin{cases} 1 & \overline{v} \text{ is an encoding of state in } S_{f_1} \\ 0 & \text{else} \end{cases}$$

$$f_2(\overline{v}) = \begin{cases} 1 & \overline{v} \text{ is an encoding of state in } S_{f_2} \\ 0 & \text{else} \end{cases}$$

$$f(\overline{v}) = \begin{cases} 1 & \begin{cases} \overline{v} \text{ is an encoding of state in } S_{f_1} \\ and \\ \overline{v} \text{ is an encoding of state in } S_{f_2} \end{cases} \\ 0 & \text{else} \end{cases}$$

3. אם $\boldsymbol{f = \neg f_1}$ וגם נתון BDD $f_1(\overline{v})$ עבור f_1 (שמייצג S_{f_1}), אזי נחזיר את ה-BDD המתקבלת מהפעלת פעולת $\neg$ על $f_1(\overline{v})$.

4. $f = EXf_1$ וגם נתון BDD $f_1(\bar{v})$ עבור f_1 (שמייצג S_{f_1}).

ה-BDD של הנוסחה הבאה מייצג את קבוצת המצבים שמספקים את EXf_1:

$$f(\bar{v}) = \exists \bar{v}' \left[f_1(\bar{v}') \wedge R(\bar{v}, \bar{v}') \right]$$

קבוצה זו היא קבוצת כל המצבים שיש להם בן שמספק את f_1.

הגדרנו כאן **אופרטור BDD חדש (pre-image)**: $EXf_1(\bar{v})$. האופרטור יוגדר כך: $EXf_1(\bar{v}) = \exists \bar{v}' \left[f_1(\bar{v}') \wedge R(\bar{v}, \bar{v}') \right]$. מביא את כל האבות של מצבים המקיימים f_1.

ניתן להגדיר אופרטורים נוספים: EG, EFf_1 וכן אופרטורים אחרים באותו הרעיון.

5. $f = EFf_1$. הטיפול במקרה זה יהיה באופן הבא: מתחילים מקבוצת המצבים שבמרחק 0 ממצבים שמספקים את f_1. באופן איטרטיבי עולים כל פעם רמה למעלה, בתחילה אל מצבים במרחק 1 ממצבים המספקים את f_1 וכך הלאה, עד ליצירת כל קבוצת המצבים המבוקשת. בכל פעם נאסוף את כל האבות בהם אנו נתקלים בדרך.

אלגוריתם לחישוב קבוצת המצבים המספקים $f = EFf_1$:

$Q := \varnothing$
$Q' := f_1(\bar{v})$
while $Q \neq Q'$ do
    $Q := Q'$
    $Q'(\bar{v}) = Q(\bar{v}) \vee EXQ(\bar{v})$
end while
$f(\bar{v}) = Q(\bar{v})$

הערה חשובה: פעולת $\vee$ על BDD נותנת איחוד של קבוצות.

האלגוריתם מחשב את **נקודת השבת הקטנה ביותר**. x היא נקודת שבת של פונקציה f אם $f(x) = x$. במקרה שלנו אנחנו מתחילים מקבוצת ממצבים ומוסיפים מצבים עד שלא ניתן להוסיף יותר. מאפיין חשוב של נקודת השבת הקטנה ביותר: מתחילים מקבוצה ריקה ומוסיפים איברים עד אשר אי אפשר להוסיף יותר. כשאי אפשר להוסיף איברים הגענו לנקודת השבת.

6. $f = E\left(f_1 U f_2\right)$.

אלגוריתם לחישוב קבוצת המצבים המספקים $f = E\left(f_1 U f_2\right)$:

$Q := \varnothing$
$Q' := f_2\left(\overline{v}\right)$
while $Q \neq Q'$ do
$\quad Q := Q'$
$\quad Q'\left(\overline{v}\right) = Q\left(\overline{v}\right) \vee \left(EXQ\left(\overline{v}\right) \wedge f_1\left(\overline{v}\right)\right)$
end while
$f\left(\overline{v}\right) = Q\left(\overline{v}\right)$

אלגוריתם זה הוא מטיפוס **נקודת שבת קטנה ביותר** (מתחילים מקבוצה שידוע שצריכה להיות בפתרון ומגדילים אותה עד לנקודת שבת).

נביט בשלב הבא:

$$Q'\left(\overline{v}\right) = Q\left(\overline{v}\right) \vee \left(EXQ\left(\overline{v}\right) \wedge f_1\left(\overline{v}\right)\right)$$

$Q\left(\overline{v}\right)$ זוהי קבוצת המצבים שחושבו עד כה. $\left(EXQ\left(\overline{v}\right) \wedge f_1\left(\overline{v}\right)\right)$ זוהי קבוצת המצבים שיש להם בן בקבוצה $Q\left(\overline{v}\right)$ והם עצמם מספקים את $f_1\left(\overline{v}\right)$.

7. טיפול ב- EGf_1

נתחיל מקבוצת המצבים המספקים את f_1 (נתונים על ידי $f_1\left(\overline{v}\right)$). נזרוק צמתים שכל הבנים שלו מספקים את $\neg f_1$ ואז בסריקה אחורנית נזרוק צמתים שזרקנו בן שלהם.

אלגוריתם:

$Q := S\left(\overline{v}\right)$
$Q' := f_1\left(\overline{v}\right)$
while $Q \neq Q'$ do
$\quad Q := Q'$
$\quad Q' = EXQ \wedge Q$
end while
$f\left(\overline{v}\right) = Q\left(\overline{v}\right)$

טענה: כל מצב שנשאר בקבוצה בסוף התהליך מספק את EGf_1.

טענה: כל מצב שמספק את EGf_1 נשאר בקבוצה בסוף התהליך.

מהטענות נובע כי אלגוריתם זה מספק לנו את **נקודת השבת הגדולה ביותר**.

6.6.4. דוגמאות

6.6.4.1 **Reachable**

אלגוריתם סימבולי למציאת כלל המצבים הישיגים מקבוצת מצבים $P(\overline{v})$:

```
reachable(P(v))
{
    new ← P
    reach ← P
    while (new ≠ ∅)
    {
        reach ← reach ∨ new
        new(v') ← ∃v[R(v, v') ∧ new(v)]
        new ← new ∧ ¬reach
    }
}
```

הרעיון בגדול: new מכיל בכל פעם את הצמתים האחרונים שהוספנו, שמהם אנחנו רוצים לחפש קשתות חדשות. בכל פעם אנחנו מוסיפים את כל הצמתים שישיגים מ-new, ומרחיבים את new למעגל רחב יותר.

נקודות:

- $new \wedge \neg reach$ משמעותו שאנחנו לא לוקחים צמתים שאינם "מהמעגל האחרון" של הצמתים הנסרקים.
- $new(\overline{v}')$ בא לציין כי המשתנים החדשים ב-new הם מעל $\overline{v}'$

6.6.4.2 תיאור אלמנטים בגרף באמצעות BDD

נתון BDD $R(\overline{v},\overline{v}')$ המתאר קשתות בגרף כלשהו.

א. רשמו נוסחה עבור קבוצת הקשתות הדו כיווניות בגרף.

פתרון: $H(\overline{v},\overline{v}') = R(\overline{v},\overline{v}') \wedge R(\overline{v}',\overline{v})$

ב. רשמו נוסחה עבור אוסף זוגות הצמתים $(\overline{v},\overline{v}')$ שיש ביניהם מסלול באורך 2:

פתרון: $H(\overline{v},\overline{v}') = \exists \overline{v}'' \left[R(\overline{v},\overline{v}'') \wedge R(\overline{v}'',\overline{v}') \right]$

ג. רשמו נוסחה שתקבל ערך TRUE אם ורק אם הגרף מלא (בין כל שני צמתים יש קשת).

פתרון: $H = \forall \overline{v} \forall \overline{v}' \left[\left(S(\overline{v}) \wedge S(\overline{v}') \right) \rightarrow R(\overline{v},\overline{v}') \right]$

משמעות הנוסחה: לכל זוג צמתים - אם הם בגרף, אז יש ביניהם קשת.

6.7. בדיקת מודל סימבולית מבוססת SAT

6.7.1. תזכורת – בעיית SAT

בעיית SAT היא בעיית הספיקות של נוסחאות פסוקיות: בהנתן נוסחה פסוקית (פסוק CNF), רוצים לדעת האם יש לנוסחה השמה מספקת. SAT עבור נוסחה כללית היא בעיה NP-שלמה (עבור פסוקי CNF היא בעיה NP-קשה). ידוע אלגוריתם אקספוננציאלי שפותר אותה, ואנו מניחים שאין לה פתרון פולינומיאלי.

קיימים בשוק **פותרי SAT (SAT Solvers)** המקבלים נוסחה פסוקית ומחזירים אם היא ספיקה או לא. רובם מבוססים על נוסחאות במבנה CNF.

מושגים:

- **משתנה בוליאני**: משתנה שיכול לקבל 0 או 1.
- **ליטרל**: $x, \neg x$ (משתנה בוליאני או שלילה שלו)
- **פסוקית**: $(x_1 \vee x_2 \vee \neg x_3)$ (רשימה של ליטרלים עם סימן $\vee$ ביניהם)
- **פסוק CNF**: $(x_1 \vee x_2 \vee \neg x_3) \wedge (... \vee ... \vee) \wedge (\ ) ...$ (רשימה של פסוקיות עם $\wedge$ ביניהן).

  חשוב לדעת: ניתן לתרגם כל פסוק לפסוק מקביל במבנה CNF.

השמה:

- **השמה** נותנת ערכים למשתנים של הנוסחה - T או F, (1 או 0).
- דוגמה להשמה: $A = \{x_1 = 0, x_2 = 1, x_3 = 0, ...\}$
- **השמה מלאה** נותנת ערך **לכל** המשתנים.
- **השמה חלקית** נותנת ערך **לחלק** מהמשתנים. למשל: $A = \{x_7 = 0, x_2 = 0\}$.

בהמשך נחזור ונרחיב עוד על מונחים אלה.

6.7.2. בדיקת מודל מבוססת SAT – Bounded Model Checking

המטרה שלנו היא להשתמש ב-SAT Solvers כדי לבצע בדיקת מודל.

בהנתן נוסחה f ומבנה קריפקה M, נרצה לקודד בדיקת מודל $M \models f$ לנוסחה פסוקית.

נציג כדוגמא טיפול בנוסחאות במבנה AGp, כאשר p נוסחה בוליאנית.

<u>הגישה שנציג היא הפרכה ולא אימות</u>: נקודד נוסחה פסוקית כך שאם יש השמה המספקת אותה יתקיים כי $M \not\models f$. במידה וההשמה אינה מספקת את הנוסחה איננו יודעים את המבנה מספק את הנוסחה, ונצטרך לבדוק נוסחה חדשה. נבצע בדיקת מודל חסומה.

<u>הגדרת הבעיה</u>: בהינתן מודל M, נוסף $f = AGp$ (שקולה ל- $\neg f = EF\neg p$) וחסם k, צריך לבדוק האם קיים מסלול ב-M באורך קטן או שווה ל- k ממצב התחלתי שמוביל למצב המספק $\neg p$. אם מצאנו השמה מספקת אז יתקיים כי $M \not\models f$, כלומר $M \not\models AGp$. אם לא מצאנו השמה מספקת נגדיל את k.

<u>עצירה</u>: נעצור את התהליך במקרים הבאים: א. התפוצצות זכרון/זמן. ב. נמצא מסלול המוביל למצב המספק $\neg p$. ג. הגעה לחסם d מקסימלי על אורך המסלול, כך שאם אין שגיאה עד חסם זה מובטח שאין שגיאה בכלל ומתקיים $M \models AGp$.

בדיקת מודל חסומה עבור M ו- $\varphi = EF\neg p$:

1. נקבע k (אפשרי לקבוע $k = 1$)
2. נבנה נוסחה פסוקית f_M^k המתארת את פרישת המודל עד עומק k: הנוסחה תתאר את כל הרישות באורך k של מסלולים אפשריים ב-M ממצב התחלתי.
3. נבנה נוסחה פסוקית f_φ^k המתארת את כל הרישות באורך k המספקות את φ.
4. נבדוק האם הנוסחה $f_M^k \wedge f_\varphi^k$ ספיקה. אם כן אז קיים $s_0 \in S$ כך ש- $M, s_0 \models EF\neg p$ ולכן $M \not\models AGp$.

נקודת מפתח להבנת נושא SAT: SAT מקבלת נוסחה ומחזירה האם קיימת לה השמה מספקת. אם במקרה שלנו הנוסחה מייצגת מבנה של מסלול חוקי, SAT מחזירה האם קיים מסלול כזה.

אם קיימים מספר מסלולים, SAT תחזיר רק את קיומו של אחד מהם. (לכן SAT נוחה להוכחת E ולא להוכחת A).

מתי אפשר לעצור את האלגוריתם?

1. פתרון פשטני: $k=|S|$.
2. חסם הדוק יותר: $d=\max_{S}\left\{\min_{s_0\in S_0} path(s_0,S)\right\}$. כאשר $path(s_0,S)$ הוא אורך המסלול בין s_0 ל-S, כלומר אנחנו מחפשים את המסלול הארוך ביותר מבין המסלולים הקצרים ביותר ממצב התחלתי את כל אחד מהמצבים. בפועל חסם זה לא מעשי – מציאתו NP-שלמה.

איך מקבלים את הנתונים? איך בונים את הנוסחה הפסוקית שנשלח אל SAT?

- מודל $M=(S,R,I,L)$, כאשר S זו קבוצת המצבים, R רלציית המעברים $I\subseteq S$ קבוצת המצבים ההתחלתיים ו-L פונקצית הסימון.
- כל מצב מתואר על ידי קידוד בינארי של משתנים בוליאנים $\overline{V}=(v_1,v_2,...,v_n)$.
- נוסחה פסוקית המתארת את המצבים ההתחלתיים: $I(\overline{v})$.
- עבור $p\in AP$ (בודקים $EF\neg p$): נוסחה פסוקית לתיאור המצבים המספקים את $\neg p$ שנסמנה $\neg p(\overline{v})$.
- נוסחה פסוקית המתארת את רלציית המעברים $R(\overline{v},\overline{v'})$

בניית הפונקציות:

- נסמן $\overline{v_i}=(v_{i1},v_{i2},...,v_{in})$.
- נגדיר את f_M^k בצורה הבאה:

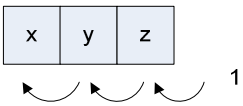
$$f_M^k(\overline{v_0},\overline{v_1},...,\overline{v_k})=I(\overline{v_0})\wedge R(\overline{v_0},\overline{v_1})\wedge R(\overline{v_1},\overline{v_2})...\wedge R(\overline{v_{k-1}},\overline{v_k})$$

- הנוסחה עבור $EF\neg p$ וחסם k:

$$f_\varphi^k(\overline{v_0},\overline{v_1},...,\overline{v_k})=\neg p(\overline{v_0})\vee\neg p(\overline{v_1})\vee...\vee\neg p(\overline{v_k})$$

- אם ידוע שאנחנו בודקים $k=0,1,2,...$ באופן רציף, אפשר להשתמש בנוסחה $f_\varphi^k(\overline{v_0},\overline{v_1},...,\overline{v_k})=\neg p(\overline{v_k})$. הסבר:
  - $f_M^0\wedge f_\varphi^0$ פירושו שאין שגיאה במצבים ההתחלתיים.
  - $f_M^1\wedge f_\varphi^1$ פירושו שאין שגיאה במצבים במרחק 1.

דוגמא



Shift register עם 3 ביטים x, y, z. כל פעם שמכניסים ביט כל הביטים זזים תא אחד שמאלה. המפרט דורש שתמיד ביט אחד לפחות יהיה שווה ל-0.

$$R(x,y,z,x',y',z') = (x' = y) \wedge (y' = z) \wedge (z' = 1)$$

$$I(x,y,z) = (x = 0) \vee (y = 0) \vee (z = 0)$$

אנחנו רוצים לבדוק האם $AG\left[(x=0) \vee (y=0) \vee (z=0)\right]$. כדי לבצע זאת נבדוק את השלילה:

$$EF\left[(x=1) \wedge (y=1) \wedge (z=1)\right]$$

ומכאן:

$$\neg p(x,y,z) = (x=1) \wedge (y=1) \wedge (z=1)$$

נרשום את הנוסחאות עבור $k=2$:

$$f_M^2 = (x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2) =$$
$$(x_0 = 0 \vee y_0 = 0 \vee z_0 = 0) \wedge (x_1 = y_0 \wedge y_1 = z_0 \wedge z_1 = 1) \wedge (x_2 = y_1 \wedge y_2 = z_1 \wedge z_2 = 1)$$
$$f_\varphi^2 = (x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2) =$$
$$(x_0 = 1 \wedge y_0 = 1 \wedge z_0 = 1) \vee (x_1 = 1 \wedge y_1 = 1 \wedge z_1 = 1) \vee (x_2 = 1 \wedge y_2 = 1 \wedge z_2 = 1)$$

דוגמא להשמה מספקת:

x_0	y_0	z_0
1	0	1

מצב התחלתי S_0

x_1	y_1	z_1
0	1	1

מצב 1 - S_1

x_2	y_2	z_2
1	1	1

מצב 2 - S_2

נשים לב שקיימת גם השמה מספקת באורך $k=1$: $111 \to 011$.

6.7.3. הוכחת נכונות בעזרת SAT – Unbounded Model Checking

נרצה כעת להציג שיטת להוכחת נכונות באמצעות SAT. נציג שוב דוגמא לבדיקת AGp כדי להציג את הנושא. לשיטה המוצגת אופי של הוכחה אינדוקטיבית: בסיס וצעד, שלאחריהם נשלים את האלגוריתם.

הכנה: נוסחה לתיאור מסלול ללא מעגלים:

$$LoopFree(\overline{v}_0,...,\overline{v}_k)=\left(\bigwedge_{i=0}^{k-1}R(\overline{v}_i,\overline{v}_{i+1})\right)\wedge\left(\bigwedge_{0\le i<j\le k}(\overline{v}_i\neq\overline{v}_j)\right)$$

נשתמש ב-2 נוסחאות לצורך בדיקת המודל:

$$\varphi^k_{base}(\overline{v}_0,...,\overline{v}_k)=I(\overline{v}_0)\wedge\left(\bigwedge_{i=0}^{k-1}R(\overline{v}_i,\overline{v}_{i+1})\right)\wedge\left(\bigvee_{j=0}^{k}\neg p(\overline{v}_j)\right)$$

אם הנוסחה ספיקה נסיק שיש שגיאה – קיים מסלול שבו יש מצב המספק $\neg p$.

אם הנוסחה איננה ספיקה – כל מסלול באורך k ממצב התחלתי מכיל רק מצבים המספקים את p

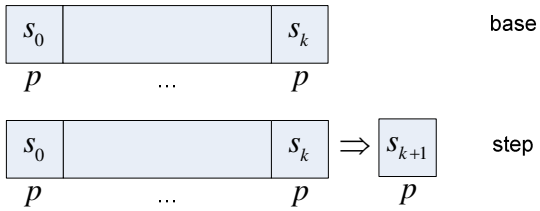
$$\varphi^k_{step}(\overline{v}_0,...,\overline{v}_k,\overline{v}_{k+1})=\left(\bigwedge_{j=0}^{k}R(\overline{v}_j,\overline{v}_{j+1})\wedge p(\overline{v}_j)\right)\wedge\neg p(\overline{v}_{k+1})\wedge\left(\bigwedge_{j=0}^{k-1}(\overline{v}_j\neq\overline{v}_k)\right)$$

המצב המעניין עבור φ^k_{step} הוא המצב בו φ^k_{step} איננה ספיקה. כאשר φ^k_{step} איננה ספיקה אז לכל מסלול (לא בהכרח ממצב התחלתי) באורך $k+1$, אם כל מצביו $1,...,k$ מספקים את p, אז המצב האחרון $k+1$ מספק גם הוא את p.

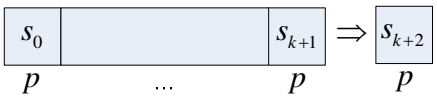
האלגוריתם להוכחת נכונות באמצעות SAT:

1. נבחר k. אם φ^k_{base} ספיקה נחזיר את ההשמה המספקת כדוגמא נגדית ($M\not\models AGp$).
2. אחרת, אם φ^k_{step} לא ספיקה, נעצור ונחזיר true, כלומר $M\models AGp$.
3. אחרת – נגדיל את k ונחזור לשלב 1.

הרעיון מאחור האלגוריתם הוא הצעדים הבאים:



כאשר הוכחנו את אלה, ניתן להזיז את נקודת המבט ונקבל כי:



6.7.4. תרגילים ודוגמאות

6.7.4.1. **תרגיל 1**

בשאלה זו נדון באלגוריתם Bounded Model Checking (BMC) לבדיקת AGp.

סעיף א'

נתון מבנה קריפקה M שמיוצג ע"י נוסחאות CNF. בעת חישוב הנוסחה עבור רלציית המעברים R קרתה "תקלה" וחלק מהפסוקיות בנוסחת ה- CNF הושמטו. נסמן את הנוסחה שהתקבלה ב- R_1.

נריץ בדיקת ספיקות של נוסחה ϕ שהיא נוסחת BMC עבור M, k ו- AGp תוך שימוש ב- R_1 במקום R. האלגוריתם מחזיר "sat" (כלומר – הנוסחה סופקה). האם $M \models_k AGp$?

פתרון סעיף א':

לא בהכרח - אם מורידים פסוקיות מנוסחה שמייצגת קשת בגרף, עלולים ליצור קשתות נוספות ובכך לאפשר מסלולים שלא היו קיימים וכך למצוא מסלול שסותר את AGp.

<u>הדגש החשוב (והמבלבל בתחילה) בדרך להבנה: מחיקת פסוקיות יוצרת מסלולים חדשים המקיימים את הנוסחה</u>. נסו למחוק בדוגמת ה-shift register את אחת הפסוקיות כדי להבין זאת.

סעיף ב'

האלגוריתם מחזיר "unsat". האם בהכרח $M \models_k AGp$?

פתרון סעיף ב':

מתקיים $M \models_k AGp$ - הוספנו עוד מסלולים לגרף. אם לא הצלחנו לספק את הנוסחה עם מסלולים נוספים, אין סיכוי שנצליח לספק אותה עם פחות מסלולים. לכן בכל המסלולים יתקיים AGp .

סעיף ג'

הפעם בעת חישוב הנוסחה עבור R נוספו פסוקיות CNF. נוסחת ה- BMC שהתקבלה מסומנת ב- R_2 והאלגוריתם לבדיקת ספיקות השתמש בה במקום ב-R.

אם מתקיים כי האלגוריתם מחזיר "sat", האם $M \not\models_k AGp$? ואם מתקיים כי האלגוריתם מחזיר "unsat", האם $M \models_k AGp$?

פתרון סעיף ג':

כעת התשובה הפוכה. הוספת פסוקיות CNF מקטינה את רלצית המעברים ולכן מקטינה את הסיכוי למצוא מסלול הסותר את הנוסחה AGp .

6.8. פתרונות לבעיית SAT – SAT Solver

6.8.1. מבוא והצגה חוזרת של הבעיה

בפרק זה נסטה מהדיונים על בדיקת מודל, ונרחיב על בעיית SAT, שהוצגה בקצרה בפרק הקודם.
בעיית SAT היא בעיית הספיקות של נוסחאות פסוקיות: בהנתן נוסחה פסוקית (פסוק CNF), רוצים לדעת האם יש לנוסחה השמה מספקת.

השמות:

- **השמה מלאה** נותנת ערך לכל המשתנים.
- **השמה חלקית** נותנת ערך לחלק מהמשתנים.
- **השמה מספקת** צריכה לספק כל אחת מהפסוקיות.

  דוגמא: נביט בפסוקית הבאה: $(a \vee d) \wedge (\neg c \vee b)$. ההשמה $a = 1, c = 0$ היא השמה חלקית מספקת.
- רוב כלי ה-SAT הקיימים מספקים השמה מלאה על ידי ריפוד השמת חלקית שנמצאת.

בעיית SAT: כאשר קיימים m משתנים בנוסחה קיימות 2^m השמות אפשריות שצריך לבדוק.
כאמור – ידוע פתרון אקספוננציאלי לבעיה. קיימים אלגוריתמים אותם נציג המאפשרים לפתור SAT בזמן סביר.

6.8.2. אלגוריתם בסיסי לפתרון SAT - Davis Putmam (DPLL), 1960, 1962

הגדרה: **פסוקית יחידה** היא פסוקית שתחת השמה חלקית נתונה יש בה בדיוק ליטרל אחד שאין לו ערך, וכמו כן ערכם של כל שאר הליטרלים בפסוקים הוא false.
לדוגמא, תחת ההשמה $b = 1, c = 0$ הפסוקית הבאה היא פסוקית יחידה: $(a \vee \neg b \vee c)$. הדרך היחידה להרחיב את ההשמה הנתונה להשמה מספקת היא לתת לליטרל שנותר את הערך true.

האלגוריתם לפתרון SAT:

1. מתחילים מהשמה חלקית ריקה - שלא נותנת ערך לאף משתנה.
2. **החלטה**: מרחיבים את ההשמה החלקית הנתונה ע"י החלטה (משתנה + ערך).
3. **הסקה (BCP)**: מרחיבים השמה עם כל הגרירות שנובעות מההחלטה.
4. **ניתוח**: אם התקבל קונפליקט (פסוקית קיבלה ערך F או לחילופין משתנה שחייב לקבל גם ערך true וגם ערך false) בהשמה הנוכחית אז מבצעים נסיגה, שבה מבטלים חלק מההחלטות, וחוזרים ל-1.

נסיגה כרונולוגית: מבטלים החלטות עד ההחלטה האחרונה שבה המשתנה עדין לא קיבל את הערך השני, והופכים את הערך.

5. אם אין קונפליקט וההשמה עדיין אינה מספקת, חוזרים ל-2.

עצירת האלגוריתם:

- האלגוריתם עוצר בהצלחה (עם השמה מספקת) כאשר נמצאה השמה שמספקת את כל הפסוקיות.
- האלגוריתם עוצר ללא הצלחה (עם מסקנה שהנוסחה אינה ספיקה) אם יש פסוקית שערך האמת שלה הוא F ואין החלטות שאפשר לשנות אותן.

דוגמא לתהליך:

1. נביט בנוסחה הפסוקית הבאה:

$$(\neg b \vee c) \wedge (\neg a \vee \neg d) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee d) \wedge (a \vee \neg c \vee \neg e)$$

2. החלטה ברמה 1: נבחר $b = 1$:

$$\left(\underbrace{\neg b}_{0} \vee c\right) \wedge (\neg a \vee \neg d) \wedge \left(a \vee \underbrace{b}_{1} \vee \neg c\right) \wedge (\neg a \vee d) \wedge (a \vee \neg c \vee \neg e)$$

3. BCP: צריך לבצע $c = 1$ בגלל פסוקית היחידה השמאלית:

$$\left(\underbrace{\neg b}_{0} \vee \underbrace{c}_{1}\right) \wedge (\neg a \vee \neg d) \wedge \left(a \vee \underbrace{b}_{1} \vee \underbrace{\neg c}_{0}\right) \wedge (\neg a \vee d) \wedge \left(a \vee \underbrace{\neg c}_{0} \vee \neg e\right)$$

4. החלטה ברמה 2: נבחר $a = 1$.

$$\left(\underbrace{\neg b}_{0} \vee \underbrace{c}_{1}\right) \wedge \left(\underbrace{\neg a}_{0} \vee \neg d\right) \wedge \left(\underbrace{a}_{1} \vee \underbrace{b}_{1} \vee \underbrace{\neg c}_{0}\right) \wedge \left(\underbrace{\neg a}_{0} \vee d\right) \wedge \left(\underbrace{a}_{1} \vee \underbrace{\neg c}_{0} \vee \neg e\right)$$

קיבלנו קונפליקט – לא קיים ערך של d שיספק את הנוסחה, ולכן מבצעים נסיגה.

נשים לב שבמקרה הזה זיהינו את הקונפילט באופן מיידי. לעתים הקונפילקט יתגלה רק לאחר כמה שלבים נוספים.

5. נסיגה: נבחר $a=0$.

$$\left(\underbrace{\neg b}_{0}\vee\underbrace{c}_{1}\right)\wedge\left(\underbrace{\neg a}_{1}\vee\neg d\right)\wedge\left(\underbrace{a}_{0}\vee\underbrace{b}_{1}\vee\underbrace{\neg c}_{0}\right)\wedge\left(\underbrace{\neg a}_{1}\vee d\right)\wedge\left(\underbrace{a}_{0}\vee\underbrace{\neg c}_{0}\vee\neg e\right)$$

6. BCP: הפסוקית האחרונה מכריחה אותנו לבחור $e=0$.

$$\left(\underbrace{\neg b}_{0}\vee\underbrace{c}_{1}\right)\wedge\left(\underbrace{\neg a}_{1}\vee\neg d\right)\wedge\left(\underbrace{a}_{0}\vee\underbrace{b}_{1}\vee\underbrace{\neg c}_{0}\right)\wedge\left(\underbrace{\neg a}_{1}\vee d\right)\wedge\left(\underbrace{a}_{0}\vee\underbrace{\neg c}_{0}\vee\underbrace{\neg e}_{1}\right)$$

7. קיבלנו השמה חלקית מספקת: $a=0,b=1,c=1,e=0$. הערך של d יכול להיות 0 או 1.

6.8.3. שיפור לאלגוריתם – יוריסטיקה לבחירת המשתנה הבא שיקבל ערך

מטרת השיפור היא לחשב ערך יוריסטי לכל בחירה אפשרית של קביעת ערך אמת למשתנה. ככל שהערך היוריסטי של משתנה גבוה יותר, נעדיף לנסות לקבוע ערך למשתנה הזה קודם, כאשר אנחנו מנסים לספק את הפסוק.

1. בחירת המשתנה שמופיע הכי הרבה פעמים

לכל משתנה x שומרים את מספר הפסוקיות בהן הוא נמצא (באופן חיובי או שלילי). נבחר את המשתנה שהמונה שלו הגדול ביותר. (בחירה נוספת שיש לבצע – האם לתת לו ערך 0 או 1).
סיבוכיות: יש לעבור על כל הפסוקיות של המשתנה שבחרנו (יש לבצע זאת כל פעם, כי כל החלטה מורידה כמה פסוקיות שהסתפקו), ולכן התקורה של יוריסטיקה זו היא גבוהה.

2. בחירת המשתנים לפי גודל הפסוקיות

נטפל בפסוקיות קצרות קודם, כי הן מגבילות יותר את מרחב החיפוש של ההשמות.

- הפסוקית הקצרה ביותר מכילה ליטרל אחד, ולכן המרחב נחצה לשניים - עבור פסוקית $\alpha=(x)$ מתאים רק $x=T$ ועבור פסוקית $\beta=(\neg x)$ מתאים רק $x=F$. משם ממשיכים.
- פסוקית בעלת 2 ליטרלים, למשל הפסוקית $(x\vee y)$, מקטינה ברבע את מרחב החיפוש, מכיוון שלא יתכן המצב $x=F\wedge y=F$.
- ככל שהפסוקית ארוכה יותר, מרחב החיפוש מצטמצם בחלק קטן יותר (כלומר מרחב החיפוש גדול יותר).

מבחינת הביצוע, נגדיר $\delta(l)=\sum_{l\in w, w\in\varphi} 2^{-|w|}$, כאשר l ליטרל, w פסוקית, ו-φ נוסחת CNF.

בכל רגע נעדיף לבחור קודם את הליטרל שה-δ שלו גדול יותר. המוטיבציה: אם ליטרל מופע בעשר פסוקיות קצרות, הוא עדיף על ליטרל שמופע ב 10 פסוקיות ארוכות. לפי ההגדרה של $\delta(l)$ מתקיים כי ככל שהפסוקית w ארוכה יותר - $2^{-|w|}$ קטן יותר.

6.8.4. שיפור לאלגוריתם – שימוש במבנה נתונים חכם לזרוז BCP

מוטיבציה: זמן ניכר מריצת ה-SAT Solver (יכול להגיע אף ל-90% מהזמן) עסוק האלגוריתם בביצוע BCP. כדי לשפר את מהירות הפעולה נרצה לשפר את ביצוע BCP, כלומר – זיהוי מהיר של פסוקיות יחידה.

הרעיון:

- לכל משתנה תהיה הצבעה ממנו אל כל הפסוקיות שבהן הוא מופיע (חיובי או שלילי).
- לכל פסוקית יהיו שני מצביעים ("שומרים") המצביעים ל-2 ליטרלים שאין להם ערך בהשמה.
- הרעיון: בפסוקית יש הרבה ליטרלים. כל זמן שיש 2 ליטרלים שאין להם השמה – אנו יודעים כי הפסוקית עדיין אינה פסוקית יחידה.

ביצוע:

- בהחלטה או ב-BCP, כאשר משתנה x קיבל ערך, לכל פסוקית ש-x משתתף בה (אופציונלי: ושאינה ספיקה), נבצע את הפעולות הבאות:
  - אם הפסוקית כבר ספיקה – לא עושים כלום (עוצרים).
  - אם x הוא איננו אחד מהשומרים – לא עושים כלום (עוצרים).
  - אם x הוא שומר והליטרל שבו הוא מופיע קיבל ערך true מסמנים את הפסוקית כספיקה ועוצרים.
  - אם x הוא שומר והליטרל הוא false, מחפשים בפסוקית ליטרל חדש שאינו שומר ושאין לו ערך.
    - אם יש כזה – הליטרל החדש הופך לשומר.
    - אם אין כזה – זוהי פסוקית יחידה. מציבים ערך true לשמור השני.

6.8.5. שיפור לאלגוריתם - למידה

הרעיון של השיפור: לימוד מתת בעיה אחת לשניה: כאשר האלגוריתם מגיע לקונפילקט הוא בונה "פסוקית קונפליקט" – פסוקית שמתווספת לנוסחה המקורית. פסוקית זו תאפשר לאלגוריתם לזהות בהמשך את הבעיה במהירות ולהמנע מלחזור עליה.

7. נוסחאון

פרק זה מכיל ריכוזי נוסחאות ומשפטים חשובים שהוצגו במהלך מסמך זה.

7.1. *למת החישובים*

למת החישובים מתארת את הצורה של כל חישובי התוכנית האפשריים. יהי $C_0 = \langle S_0, \sigma_0 \rangle$:

1. אם S_0 הוא פעולה אטומית אז $\pi(C_0)$ הוא מהצורה $C_0 \to C_1$ כאשר C_1 היא קונפיגורציה סופית (עוצרת) שנקבעת על ידי (1) או (2) בהגדרת $\to$.
2. $S_0 = S_1; S_2$. ל-$\pi(C_0)$ יש אחת משלוש צורות:

א. **התבדרות** S_1: $C_0 \to C_1 \to \ldots$ כאשר $C_i = \langle T_i; S_2, \sigma_i \rangle$ ו-$i \geq 0, \langle T_i, \sigma_i \rangle$ הוא חישוב אינסופי ב-S_1 מ-σ_0.

ב. **התבדרות** S_2: $C_0 \xrightarrow{*} \langle S_2, \sigma_i \rangle \to C_{i+1} \to \ldots$ כאשר $C_j, j \geq i$ הוא חישוב אינסופי של S_2.

ג. **עצירה**: $C_0 \xrightarrow{*} \langle S_2, \sigma_i \rangle \xrightarrow{*} \langle E, \sigma' \rangle$ כאשר $C_0 \xrightarrow{*} \langle S_2, \sigma_i \rangle$ מתאים לחישוב סופי של S_1 וגם $\langle S_2, \sigma_i \rangle \xrightarrow{*} \langle E, \sigma' \rangle$ הוא חישוב סופי של S_2.

3. אם $S_0 = \text{while } B \text{ do } S \text{ od}$ אז $\pi(C_0)$ מהצורה:

א. **עצירה מיידית**: $C_0 \to \langle E, \sigma_0 \rangle$ אם $\sigma_0 \models \neg B$.

ב. **התבדרות בחוג (S)**: (S לא מסתיים)

$C_0 \xrightarrow{*} \langle S_0, \sigma_1 \rangle \xrightarrow{*} \ldots \xrightarrow{*} \langle S_0, \sigma_k \rangle \to C_{k+1} \to \ldots$ כאשר $\sigma_j \models B$ לכל $0 \leq j \leq k$ וכן $\langle S_0, \sigma_j \rangle \to \langle S; S_0, \sigma_j \rangle \xrightarrow{*} \langle E; S_0, \sigma_{j+1} \rangle$ הוא חישוב סופי של S מ-σ_j ו-$\pi(C_{k+1})$ הוא חישוב אינסופי של S.

ג. **התבדרות החוג**: (התנאי B תמיד מתקיים)

$C_0 \xrightarrow{*} \langle S_0, \sigma_1 \rangle \xrightarrow{*} \ldots \xrightarrow{*} \langle S_0, \sigma_k \rangle \to \ldots$ כאשר $\sigma_j \models B$ לכל $j \geq 0$ וכן $\langle S_0, \sigma_j \rangle \xrightarrow{*} \langle E, \sigma_{j+1} \rangle$ הינו חישוב סופי של S מ-σ_j.

ד. **עצירה**: $C_0 \xrightarrow{*} \langle S_0, \sigma_1 \rangle \xrightarrow{*} \ldots \xrightarrow{*} \langle E, \sigma_k \rangle$ כאשר $\sigma_j \models B, \sigma_K \models \neg B$ לכל $0 \le j < k$.

4. אם $S_0 = \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}$ אז $\pi(C_0)$ הינו מהצורה:

א. אם $\sigma_0 \models B$ אז מבצעים חישוב של S_1: $\langle S_0, \sigma_0 \rangle \longrightarrow \langle S_1, \sigma_0 \rangle \xrightarrow{*} \ldots$

ב. אחרת מבצעים חישוב של S_2: $\langle S_0, \sigma_0 \rangle \longrightarrow \langle S_2, \sigma_0 \rangle \xrightarrow{*} \ldots$

7.2. מערכת ההוכחה H

אקסיומות:

(ASS) $\{p[x \leftarrow e]\} x := e \{p\}$

(SKIP) $\{p\} skip \{p\}$

(ARITH) $q \to q'$ אמת לוגית

כללי היסק:

(SEQ) $$\frac{\{p\} s_1 \{r\} \qquad \{r\} s_2 \{q\}}{\{p\} s_1; s_2 \{q\}}$$

(IF) $$\frac{\{p \wedge B\} S_1 \{q\} \qquad \{p \wedge \neg B\} S_2 \{q\}}{\{p\} if\ B\ then\ S_1\ else\ S_2\ \ fi\ \{q\}}$$

(REP) $$\frac{\{p \wedge B\} S \{p\}}{\{p\} while\ B\ do\ S\ \ od \{p \wedge \neg B\}}$$

(CONS) $$\frac{p \to p_1 \qquad \{p_1\} S \{q_1\} \qquad q_1 \to q}{\{p\} S \{q\}}$$

7.3. מערכת ההוכחה *H

כל האקסיומות וכללי ההיסק נשארים כמו ב-H למעט כלל REP, בהחלפת סוגריים מסולסלים בסוגריים משולשים. כלל REP החדש:

$$\frac{\left(p(\bar{x},n)\wedge n>0\right)\to B \qquad \left\langle p(\bar{x},n)\wedge(n>0)\right\rangle S\left\langle p(\bar{x},n-1)\right\rangle \qquad p(\bar{x},0)\to\neg B}{\left\langle \exists n, p(\bar{x},n)\right\rangle \text{ while } B \text{ do } S \text{ od } \left\langle p(\bar{x},0)\right\rangle}$$

7.4. לוגיקות טמפורליות

7.4.1. CTL*

CTL* הינה סוג של לוגיקה טמפורלית, המוגדרת באמצעות 2 סוגי נוסחאות:

3. נוסחאות מצב שמתפשרות במצב נתון.
4. נוסחאות מסלול שמתפרשות במסלול נתון.

<u>הגדרה אינדוקטיבית של הלוגיקה:</u>

<u>נוסחאות מצב</u>

3. <u>בסיס</u>: נוסחאות אטומיות $p \in AP$.
4. <u>כללי יצירה</u>:
   - אם f, g נוסחאות מצב אז $\neg f, f \vee g$ הן נוסחאות מצב.
   - אם f נוסחת <u>מסלול</u> אז Ef היא נוסחת מצב.

<u>נוסחאות מסלול</u>

3. <u>בסיס</u>: כל נוסחת מצב היא נוסחת מסלול. (הנוסחה תבחן על המצב הראשון במסלול).
4. <u>כללי יצירה</u>: אם f, g נוסחאות מסלול אז $\neg f, f \vee g$ Xf, fUg הן נוסחאות מסלול.

CTL* היא קבוצת נוסחאות <u>המצב</u> המוגדרות לפי הגדרה זו.

קיצורים: (אופרטורים המוגדרים בעזרת האופרטורים הבסיסיים)

- $f_1 \wedge f_2 \equiv \neg\left(\neg f_1 \vee \neg f_2\right)$
- $g_1 \wedge g_2 \equiv \neg\left(\neg g_1 \vee \neg g_2\right)$
- $Af \equiv \neg E \neg f$
- $Ff \equiv \left(true U f\right)$
- $Fg \equiv \left(true U g\right)$
- $Gg \equiv \neg F \neg g$

CTL .7.4.2

CTL היא השפה המוגדרת ע"י:

3. אם $p \in AP$ אז p נוסחת CTL.
4. אם f, g נוסחאות CTL אז $\neg f, f \vee g$ נוסחאות CTL.

קיצורים:

- $AXf_1 \equiv \neg EX \neg f_1$
- $EFf_1 \equiv E\left(true\ U f_1\right)$
- $AGf_1 \equiv \neg EF \neg G_1$
- $AFf_1 \equiv \neg EG \neg f_1$
- $A\left(f_1 U f_2\right) \equiv \neg\left(EG\neg f_2 \vee E\left(\neg f_2 U \left(\neg f_1 \wedge \neg f_2\right)\right)\right)$

LTL .7.4.3

נוסחת מסלול LTL היא מהצורה:

- $p \in AP$
- אם f, g נוסחאות מסלול LTL, אז גם $\neg f, fUg, Xf, f \vee g$ הן נוסחאות מסלול LTL.

נוסחת LTL היא מהצורה Af כאשר f היא נוסחת מסלול LTL.

קיצורים: הקיצורים ב-LTL זהים לאלו של CTL*.

7.4.4. הגדרת האופרטורים הטמפורלים – ויקיפדיה האנגלית

The temporal operators are the following:

- Quantifiers over paths
  - **A** φ - **A**ll: φ has to hold on all paths starting from the current state.
  - **E** φ - **E**xists: there exists at least one path starting from the current state where φ holds.
- Path-specific quantifiers
  - **X** φ - Ne**x**t: φ has to hold at the next state (this operator is sometimes noted **N** instead of **X**).
  - **G** φ - **G**lobally: φ has to hold on the entire subsequent path.
  - **F** φ - **F**inally: φ eventually has to hold (somewhere on the subsequent path).
  - φ **U** ψ - **U**ntil: φ has to hold until at some position ψ holds. This implies that ψ will be verified in the future.
  - φ **W** ψ - **W**eak until: φ has to hold until ψ holds. The difference with **U** is that there is no guarantee that ψ will ever be verified. The **W** operator is sometimes called "unless".

In CTL*, the temporal operators can be freely mixed. In CTL, the operator must always be grouped in two: one path operator followed by a state operator.

7.5. חשוב לזכור

נקודות חשובות שחשוב לזכור (בעיקר במבחנים בנושא ☺):

- "להציע כלל הוכחה" = להציע אוסף תנאים שאם נוכיח את כולם אז המערכת מקיימת את הנדרשת.
- ב-CTL אפשר לדבר על תתי עצים בהמשך הגרף. ב-LTL אי אפשר לבטא זאת.

8. מקורות

8.1. מקורות בעברית

1. "בעיית הספיקות", 2009, **ויקיפדיה העברית**.
2. "מבוא לאימות תוכנה - סיכומי הרצאות", 2005, **בן ישראל ש.**
3. "מבוא לאימות תוכנה - סיכומי הרצאות", 2006, **כרמלי י.**
4. "מבוא לאימות תוכנה - סיכומי הרצאות", 2008, **אדר נ.**
5. "תחשיב הפרדיקטים", 2009, **ויקיפדיה העברית**

8.2. מקורות באנגלית

1. "An Analysis of SAT-based Model Checking Techniques in an Industrial Environment", Cadence Design Systems, 2005, **Nina Amla, Xiaoqun Du, Andreas Kuehlmann, Robert P. Kurshan and Kenneth L. McMillan**
2. "An Efficiently Checkable, Proof-Based Formulation of Vacuity in Model Checking", 2004, **Kedar S. Namjoshi**
3. "Applying SAT methods in Unbounded Symbolic Model Checking", **K. L. McMillan**, Cadence Berkeley Labs
4. "Computational tree logic", 2006, **Wikipedia**.
5. “CTL*”, 2008, **Wikipedia**.
6. “Davis–Putnam algorithm”, 2009, **Wikipedia**.
7. “DPLL algorithm”, 2009, **Wikipedia**.
8. "Graph Algorithms", 1996, **Renaud Waldura**, http://renaud.waldura.com/portfolio/graph-algorithms/
9. "Model Checking", 1999, MIT Press, **Edmund M Clarke, Orna Grumberg and Doron Peled**.
10. "Linear temporal logic", 2006, **Wikipedia**.
11. "Logic in Computer Science(Second Edition)", 2004, **Michael Huth and Mark Ryan**.
12. "Program Verification", 1992, Addison-Wesley Publishing Company, **France Nissim**.

13. "Overview of the Temporal Logic CTL*", 2003, Knowledge Representation Lab – Texas Tech University, **Edmund M. Clarke, Orna Grumberg, and Doron Peled**.
14. "Semantics and Verification of Software - Operational Semantics of WHILE", 2008, RWTH Aachen University, **Thomas Noll**
15. "Specifying the Semantics of while Programs: A Tutorial and Critique of a Paper by Hoare and Lauer", 1981, **Irene Greif, Albert R. Meyer**
16. "Temporal Logic", 1999, **Stanford Encyclopedia of Philosophy** (http://plato.stanford.edu/entries/logic-temporal/).
17. "Timed Automata and Timed Computation Tree Logic", 2000, **Paul Pettersson**.
18. "Tuning SAT checkers for Bounded Model Checking", **Ofer Shtrichman**
19. "Vacuity Analysis by Fault Simulation", 2008, **Luigi Di Guglielmo, Franco Fummi,Graziano Pravadelli**