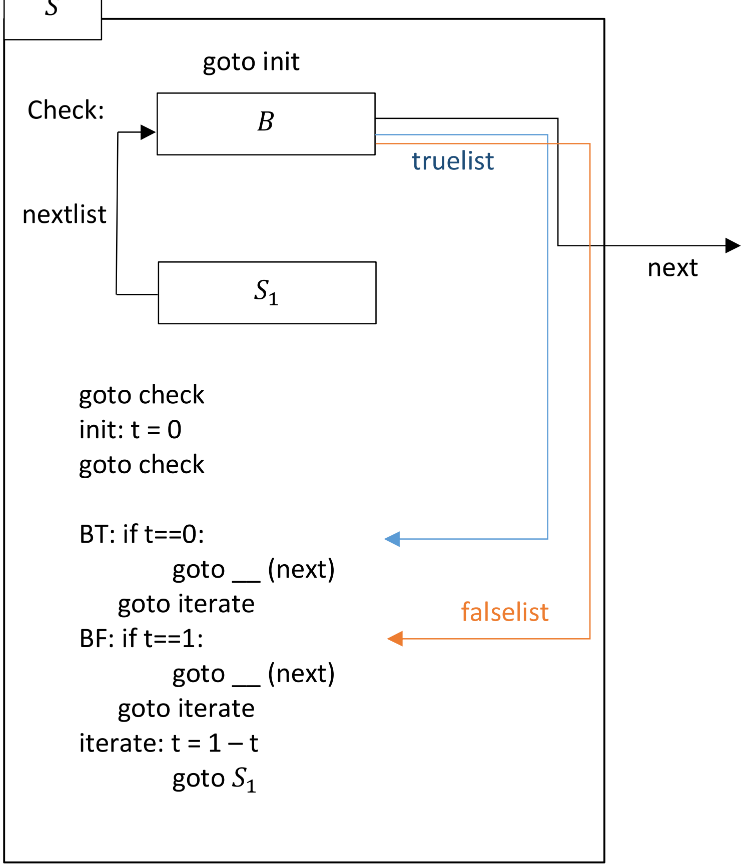
שאלה 1 – Backpatching

המארקרים שנגדיר:

$S \rightarrow trickyWhile\ \boldsymbol{N}\ \boldsymbol{M_1}\ B\ do\ \boldsymbol{M_2}\ S_1\ \boldsymbol{M_3}\ od\ \boldsymbol{M_4}$

פריסת קוד:



תכונות:

$S: nextlist$

$B: truelist, falselist$

סכימת תרגום:

$S \rightarrow trickyWhile\ \boldsymbol{N}\ \boldsymbol{M_1}\ B\ do\ \boldsymbol{M_2}\ S_1\ \boldsymbol{M_3}\ od\ \boldsymbol{M_4}$

```
{
        bp(S_1.nextlist, M_4.quad);
        bp(N.nextlist,nextquad());          // goto init
        emit(goto M_1.quad);                // goto check
        emit(t = newtemp());
        emit(goto M_1.quad);                // goto check
        bp(B.truelist,nextquad());
        S.nextlist = makelist(nextquad());
        emit("if" ||t|| "= 0: goto");
        emit("goto" || M_3.quad);
        bp(B.falselist,nextquad());
        S.nextlist=merge(S.nextlist,makelist(nextquad());
        emit("if" ||t|| "= 1: goto");
```

```
        emit("goto" || M3.quad);
        bp(M3.quad,nextquad());
        emit(t = 1 – t);
        emit ("goto" || M2.quad);
}
```

שאלה 2 – Gen/Kill DFA

א. נגדיר את הדומיינים הבאים ולאחר מכן נבצע מכפלה ביניהם:

$$Funcs = All\ available\ functions$$
$$Vars = All\ available\ varisables$$
$$N = Vars \times Funcs$$

פריטי המידע שנשמור הם:

$$in(B) \in 2^N$$
$$out(B) \in 2^N$$

ב. צורת החישוב היא סריקה אחורית – התכונה המתוארת תלויה בפקודות שבאות אחרי הנקודה P ולכן יש לפעפע את המידע מהסוף עד לנקודה שבה רוצים לבחון את מצב המשתנה.

ג. סוג הבעיה היא $Must$ כי לפי הדרישה $x.f()$ נקראת לפני ההשמה הבאה ל-x ו-$x.g()$ לא נקראת לפני ההשמה הבאה ל-x **בכל מסלול חישוב**. מספיק שיהיה מסלול חישוב אחד שהדרישה לא מתקיימת כדי ש-x לא יקרא f-חיוני-g-קטלני לפי הגדרת הבעיה.

ד. נגדיר את $gen(B)$ שתכיל את הזוגות (x,k) לכל $k \in Funcs$ אם מתבצעת השמה ל-x:

$$gen(B) = \{x | x \coloneqq ? \in B\} \times Funcs$$

ה. נגדיר את $kill(B)$ שתכיל את הזוגות (x,k) כך שמתבצעת ב-B הקריאה $x.k()$:

$$kill(B) = \{(x,k) | x.k() \in B\}$$

ו. משוואות הזרימה:

$$out(B) = f\big(in(B)\big)$$
$$in(B) = \bigcap_{B,D \in CFG} out(D)$$

ז. מאופן פעולת החיתוך, בכל פעם אנחנו מורידים פריטים מ-$in(B), out(B)$.
לכל בלוק נאתחל באופן שמרני עבור בעיית must: $in(B) = out(B) = N$
האתחול תקף גם עבור הבלוק בסוף התכנית, כי לא יהיו עוד פקודות השמה ולכן כל התנהגות תהיה חוקית.

ח. נוכל לקבל את קבוצה זו ע"י אנליזת הבלוק של נקודה P באופן הבא:
$$X = \{x | \forall f, g: (x,g) \in in(B) \land (x,f) \notin in(B)\}$$

הקבוצה X תהיה קבוצת המשתנים שמתאימים להגדרה המבוקשת.
<u>הסבר:</u> בכל הפעלה של f_B אנו מסירים מ-$in(B)$ את הזוגות המתארות קריאה לפונקציה. לכן, כדי לוודא ש-$x.g()$ לא נקראת נרצה לבדוק ש-(x,g) עדיין נמצא ב-in. בנוסף, נראה לוודא שהפונקציה $x.f()$ נקראה ולכן נבדוק ש-(x,f) לא ב-in. מאופן פעולת החיתוך, נקבל שהתכונה חייבת להישמר בכל מסלול שמתחיל בנקודה P.

שאלה 3 – Static Analysis

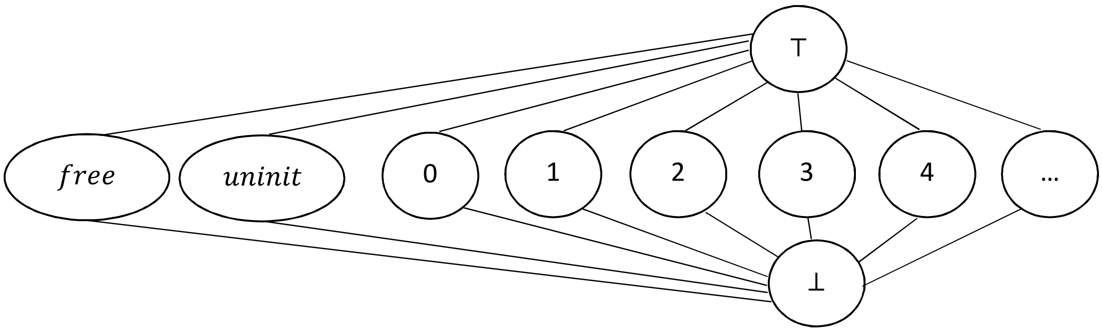
א.

a. תחילה נגדיר לכל משתנה בתכנית סריג משלו. נניח שבוצעה אנליזה מקדימה שמבצעת ספירה של כמות המשתנים – נסמן ב-m את מספר המשתנים בתכנית. בנוסף, נניח שבוצעה אנליזה לזמן המקסימאלי שאובייקט יכול להיות נגרר ונסמן בזמן זה ב-k. עבור משתנה x_i נסמן ב-L_i את הסריג המתאים לו.

הדומיין של הסריגים הללו יהיה:

$$D = [k] \cup \{\top, \bot\} \cup \{uninit, free\}$$

לשם המחשה, הסריג L_i יוגדר באופן הבא:



כעת, הסריג המתאים לאנליזה יהיה $L = L_1 \times L_2 \times \ldots \times L_m$ כאשר m זה מספר המשתנים בתכנית.

הדומיין של הסריג L הוא D^m.

b. נגדיר את $\sqcup$ באופן הבא:

$$\forall x, y \in [k] \cup \{free, uninit\}: x \sqcup y = \top$$
$$\forall x \in [k] \cup \{free, uninit\}: \bot \sqcup x = x$$
$$\forall x \in [k] \cup \{free, uninit\}: x \sqcup \top = \top$$
$$\bot \sqcup \top = \top$$

c. בהינתן פקודה בתכנית, הפונקציה תוגדר באופן הבא: הביטוי בתוך [] מבטא את תוכן השורה בתכנית:

$$f_{[v_j = new\ T()]}(v) = v'$$

$$v_i' = \begin{cases} 0, & i = j \\ \bot, & elif\ v_i = \bot \\ \top, & elif\ v_i = \top \\ v_i + size(T), & elif\ v_i \in [k] \\ v_i, & else \end{cases}$$

$$f_{[some\ usage\ of\ v_j]}(v) = \left(i_1, i_2, \dots, \underbrace{0}_{\substack{j's \\ coordinate}}, \dots, i_m \right)$$

$$f_{[free(v_j)]}(v) = \left(\left(i_1, i_2, \dots, \underbrace{free}_{\substack{j's \\ coordinate}}, \dots, i_m \right) \right)$$

$$f_{[?]}(v) = v$$

<u>הוכחת מונוטוניות:</u>

נוכיח תחילה עבור כל L_i בנפרד ואז לפי משפט מההרצאה נקבל שמכיוון ש-f מונוטונית עבור כל כניסה בוקטור, אז f מונוטונית עבור הוקטורים.

יהיו X, Y זוג וקטורים כך ש-Y מכסה את X. נסמן ב-x_i, y_i את הכניסה ה-i של הוקטורים בהתאמה. $X \sqsubseteq Y$ ולכן לפי משפט $\forall i \in [m]: x_i \sqsubseteq y_i$ נפריד למקרים:

1. $x_i = \bot$ ו-y_i הוא כל מצב. אז מהגדרת f נובע שלאחר הפעלת f על הוקטור X מתקיים $f(X)_i = \bot$ ולכל מצב שאליו y_i יעבור תתקיים מונוטוניות כי $\bot$ הוא חסם תחתון.
2. $y_i = \top$ ו-x_i הוא כל מצב. אז מהגדרת f נובע שלאחר הפעלת f על הוקטור Y מתקיים $f(Y)_i = \top$ ולכל מצב שאליו x_i יעבור תתקיים מונוטוניות כי $\top$ הוא חסם עליון.

- כעת, מכיוון שלכל $i \in [m]$ מתקיים $f(X)_i \sqsubseteq f(Y)_i$ אז לפי המשפט $f(X) \sqsubseteq f(Y)$.

■

d. את הפיתרון לשאלה נוכל לקבל ע"י הוקטור המתקבל ב-$In \backslash out$ בהתאם למיקום הנקודה P, ואם נראה באחת הקואורדינטות של הוקטור מספר שגדול מהסף שהגדרנו אז נדע שהמשתנה נגרר:

$$Let\ r\ be\ our\ threshold, then\ the\ solution\ can\ be\ obtained\ by: \{x_i | v_i \geq r\}$$

ב.

a. כעת נרצה לשמור עבור כל משתנה את כל המצבים שבהם הוא יכול להיות בו-זמנית, לכן נשמור בכל כניסה בוקטור כקבוצה. באופן פורמאלי נגדיר כ-L_i' את הסריג של כניסה בוקטור. הדומיין של L_i' הוא:

$$D' = 2^{D - \{\bot, \top\}} \cup \{\bot, \top\}$$

הסריג שלנו הוא $L' = L_1' \times L_2' \times \dots \times L_m'$

הדומיין של הסריג L' יהיה ${D'}^m$.

יחס הסדר יהיה הכלה איבר-איבר בוקטור, כלומר:

$$v \sqsubseteq u \Leftrightarrow \forall i \in [m]: v_i \subseteq u_i$$

לשם המחשת מראה הסריג L_i', הוא בנוי מ-$k+1$ רמות כאשר בכל רמה i יהיו כל הקבוצות עם $i-1$ איברים מתוך D'.

b. וקטור המצבים של המשתנים מכיל קבוצות ולכן נגדיר $\sqcup = point - wise \cup$. המשמעות היא לבצע איחוד איבר-איבר של 2 וקטורים: בהינתן $v, u \in {D'}^m$, נגדיר:

$$(v \sqcup u)_i = (v \cup u)_i = v_i \cup u_i$$

c. ה-$transfer\ function$ תהיה זהה לפונקציה שהגדרנו בסעיף א', אך תפעל בנפרד על כל איבר בתוך הקבוצות. נסמנה ב-f_2. באופן פורמאלי:

$$f_{2_{[?]}}(v) = v'$$
$$v'_k = \left\{f_{[?]}(a) \middle| a \in v_k\right\}$$

בנוסף, נתון שבלוק מכיל פקודה אחת בלבד ולכן בכל בלוק יש לכל היותר כניסה אחת בוקטור שמושפעת מביצוע הבלוק. נסמן ב-i את הכניסה שמושפעת מביצוע הפקודה, לכן:

$$\forall j \neq i: v_j = f_2(v)_j \Rightarrow v_j \subseteq f_2(v)_j$$

f מסעיף א' היא מונוטונית ולכן נקבל:

פונקציה f_2 בסעיף ב' זהה לפונקציה מסעיף א' פרט לצורה שבה תפעל על קבוצות במקום על מספרים ספציפיים. יחס ההכלה מסעיף א' לא הופר על ידי החלפת f בf_2 ולכן גם מונוטוניות f_2 מובטחת.

■

d. נגדיר פונקציות $g: 2^{D'} \to \mathbb{N}$ שתפעל בצורה הבאה:

$$g(A) = \max(\{x | x \in A \cap \mathbb{N}\})$$

נגדיר את max באופן הבא:

$$\forall x, y \in \mathbb{N}: \max(\{x, y\}) = \begin{cases} x, & x > y \\ y, & else \end{cases}$$
$$\forall x \in \{uninit, free\}, \forall y \in \mathbb{N}: \max(\{x, y\}) = y$$
$$\max(\{uninit, free\}) = free$$

נשתמש בפונקציות הנ"ל כדי לקחת איבר $v \in D'^m$ ולכל v_i נמיר אותו למצב ב-D ע"י לקיחת האיבר המקסימלי בקבוצה שהוא המרחק המקסימלי שיכול להיות בין שימוש אחרון של משתנה לשחרור שלו, כלומר מרחק גרירה מקסימלי. ניצור וקטור $dist$ בצורה הבאה:

$$dist = \{g(v_1), g(v_2), \dots, g(v_n)\}$$

ולאחר מכן נקבל תוצאה שתואמת את המבנה של סעיף א' ונקבל את הפתרון לשאלה באותה הדרך כמו שקיבלנו בסעיף א'.