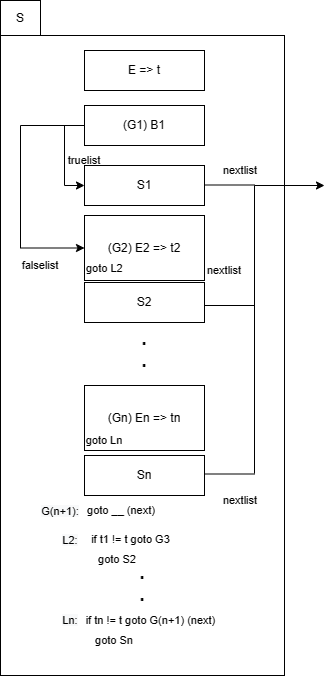
**תרגיל 4**

**שאלה 1**



1. הסבר תכונות:

S: nextlist .

E: place .

B: truelist, falselist.

C: after\_g , after\_s, exp\_address, nextlist, type, value.

CL: after\_g\_list (stack), after\_s\_list (stack) exp\_address\_list (stack),   
 value\_list(stack), nextlist .

G: truelist, falselist, type, value, next.

S, E, B – כמו בהרצאה.

G:

1. Truelist: במקרה של E – ריק.

במקרה של B – רשימת כתובות הקפיצה אותה צריך למלא בS שלאחר התנאי.

1. Falselist: במקרה של E – ריק.

במקרה של B - רשימת כתובות הקפיצה אותה צריך למלא בG הבא במידה והתנאי נכשל.

1. Type – סוג הGuard.
2. Value – שם המשתנה במקרה וזה exp.
3. Next – כתובת פקודת הקפיצה שמופיעה במקרה של E, שאמורה להתמלא בכתובת בה יש השוואת expressions.

C:

1. After\_g – שומר את הכתובת אחרי הGuard.
2. After\_s – שומר את הכתובת אחרי הS (בדיוק לפני הguard הבא.
3. Exp\_address – שומר את רשימת הקפיצות שצריך למלא כדי לקפוץ להשוואת הexp (אם מדובר בguard מסוג B אז יהיה ריק)
4. Nextlist – הכתובת שאליה ממשיכים לאחר ביצוע S.
5. Type – סוג הguard.
6. Value – שם המשתנה של הE (אם מדובר בguard מסוג B זה יהיה ערך זבל)

CL:

1. After\_g\_list – רשימת הכתובות של אחרי הGuard בכל case.
2. After\_s\_list – רשימת הכתובות של אחרי הS בכל case.
3. Exp\_address\_list – רשימת החורים שצריך למלא כאשר ניצור את תנאי הקפיצה.
4. Value\_list – רשימת שמות המשתנים של הexpressions.
5. Nextlist – הכתובת שאליה ממשיכים לאחר ביצוע הS הנבחר.

מימוש:

CL -> C  
{  
 CL.after\_g\_list = newstack();  
 CL.after\_s\_list = newstack();  
 CL.nextlist = C.nextlist;  
 CL.exp\_address\_list = newstack();  
 CL.value\_list = newstack();  
 if( C.type = "Exp"){   
 CL.exp\_address\_list.push(C.exp\_ address);  
 CL.value\_list.push(C.value);  
 CL.after\_g\_list.push(C.after\_g);  
 CL.after\_s\_list.push(C.after\_s);  
 }   
}

CL -> C CL1  
{  
 CL.after\_g\_list = CL1.after\_g\_list;  
 CL.after\_s\_list = CL1.after\_s\_list;  
 CL.nextlist = merge(CL1.nextlist, C.nextlist);  
 CL.exp\_address\_list = CL1.exp\_address\_list;  
 CL.value\_list = CL1.value\_list;  
 if( C.type = "Exp"){   
 CL.exp\_address\_list.push(C.exp\_ address);  
 CL.value\_list.push(C.value);  
 CL.after\_g\_list.push(C.after\_g);  
 CL.after\_s\_list.push(C.after\_s);  
 }   
}

C -> case G : M S; break ;  
{  
 C.after\_g = M.quad;  
 C.after\_s = nextquad();  
 C.type = G.type;  
 C.nextlist = S.nextlist;  
 if (G.type = "Exp"){  
 C.value = G.value;  
 C.exp\_address = G.next;  
 }  
 else if (G.type = "Bool"){  
 backpatch(G.truelist, C.after\_g);   
 backpatch(G.falselist, C.after\_s);  
 }  
}

G -> B  
{  
 G.type = "Bool";  
 G.truelist = B.truelist;  
 G.falselist = B.falselist;   
}

G -> is E  
{  
 G.type = "Exp";  
 G.value = E.place;  
 G.next = nextquad();  
 emit("goto \_\_");  
}

S -> switch (E) CL  
{  
 S.nextlist = merge ( CL.nextlist, makelist(nextquad());   
 emit ("goto \_\_");  
 while(!CL.exp\_address\_list.empty()){  
 after\_g = CL.after\_g\_list.pop();  
 after\_s = CL.after\_s\_list.pop();  
 exp\_address = CL. exp\_address\_list.pop();  
 value = CL.value\_list.pop();  
 backpatch(exp\_address, nextquad());  
 emit("if" || value || "!=" || E.place || "goto" || after\_s);  
 emit("goto" || after\_g);  
 }

}

1. הguard החדש אותו נציע הוא G -> is not E.

נשים לב שעבור הדקדוק הנוכחי באמת אי אפשר לגזור את הguard הזה. כי is E אינו יכול לגרום לביצוע S בעת אי שוויון ו B אינו משווה לערך הביטוי של הswitch כלל.  
השינוי הדרוש הוא בגזירה של S, להוסיף תנאי על השורה:  
emit("if" || value || "!=" || E.place || "goto" || after\_s);

כך שאם הטיפוס הוא Exp השורה תודפס, אם הטיפוס הוא nExp תודפס:

emit("if" || value || "=" || E.place || "goto" || after\_s);

וגם בבדיקות של האם type הוא exp נוסיף בכל המקרים "or nExp".

שאלה 2:

*א. 1. נגדיר את הסריג באופן הבא:*

* *האיברים: קבוצות של מחרוזות שמייצגות את הביטוי. איבר בדומיין לדוגמה: {“t1+8”, “t1+t2”,…}*
* *פעולת יחס הסדר : הפעולה תהיה הכלה בכיוון ההפוך .*
* *פעולת הjoin*: חיתוך קבוצות.

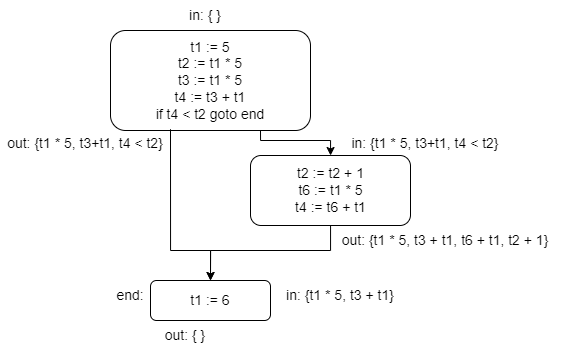
*יחס הסדר ופעולת הjoin נבחרו כך מכיוון שנרצה לדרוש שביטוי יהיה זמין מכל מסלול (אחרת יכול להיות מצב שנצטרך לחשב אותו)*.

*2.*

*לכל statement בblock נעשה:*

|  |  |  |
| --- | --- | --- |
| *Gen* | *kill* | *Statement* |
| *{ e ∈ AExpr(expr) | x ∉ FV(e) }* | *{ e ∈ AExpr | x ∈ FV(e) }* | *X := expr* |
|  |  | *goto label* |
| *AExpr(cond)* |  | *if cond goto label* |
|  |  | *label* |

*הin של המצב הראשון יהיה קבוצה ריקה.*

*ב.   
ג. לכל exp נוסיף רשימה של משתנים שהוא כרגע שמור בהם (עבור cond ניצור משתני דמה שלעולם לא יקראו) .  
השינויים שנבצע בפעולות:  
כאשר יש השמה למשתנה, נוריד אותו מהרשימה שהוא מופיע בה (אם הרשימה התרוקנה נמחק את הביטוי לגמרי) ונוסיף אותו לרשימה של הביטוי החדש (אם לא קיימת כזאת, ניצור רשימה חדשה).*

*חיתוך: נבצע חיתוך של הרשימות לפי ביטוי ונסיר ביטויים עם רשימות ריקות.*

*חיפוש: אם מצאנו את הביטוי, סיימנו. אחרת, לכל משתנה בביטוי נעבור על כל האופציות להחלפה עם משתנים שנמצאים איתו ברשימה ונבדוק האם ההחלפה נותנת ביטוי קיים. אם כן, מצאנו 😊.*