תרגום לקוד ביניים

שפת ביניים- שפת הרביעיות

- נעבוד עם 4 סוגי פקודות בלבד:

1. פעולה אריתמטית: $t_1 := t_2 + t_3$
2. קפיצה לא-מותנית: goto label
3. קפיצה מותנית: if t_1 relop t_2 goto label
4. תווית: label:

- קיימות פקודות רבות נוספות....

דוגמה

$$a = b + c + d$$

$$\Downarrow$$

$$t_1 := b + c$$

$$t_2 := t_1 + d$$

$$a := t_2$$

- מדוע צריך את t_2? לא צריך (אופטימיזציה)

טעויות נפוצות

- דברים שאינם קוד ביניים:
- if … else …
  - בשפת ביניים אין else.
- if (x > 1 && y > 1) goto …
  - התנאי בקפיצה אינו מורכב.
    - נראה ייצוג חלופי בשפת ביניים לשני המקרים הללו.
- x = l1000

  goto x
  - ניתן לקפוץ רק לתוויות קבועות.

פונקציות לפריסת קוד

- נצטרך את התכונות הסמנטיות הבאות:
  - code: מחזיקה את הקוד הנוצר.
  - var: עבור ביטויים אריתמטיים זהו המשתנה הזמני שהוקצה לביטוי (למשל: כתובת יחסית במחסנית...)

פונקציות לפריסת קוד

- פונקציות עזר:
  - freshVar(): יוצרת משתנה זמני חדש (בעל מזהה ייחודי)
  - freshLabel(): מחזירה תווית חדשה.

שיטות לייצור קוד

נציג 2 שיטות לייצור קוד:

<u>שיטה ראשונה:</u> הקוד נאגר בתכונה סמנטית בשם code:

- לכל משתנה בדקדוק יש תכונה סמנטית בשם code, אשר אוגרת את הקוד שנוצר מהמשתנים שכבר נגזרו.
- בסוף הניתוח התכונה code של המשתנה התחילי מכילה את כל הקוד של התוכנית.

שיטה ראשונה ליצירת קוד - דוגמה

- דוגמה: תרגום ביטויים אריתמטיים
  - נגדיר למשתנה E את התכונות code ו-var

(var יכיל את המשתנה הזמני אליו נשמר ערך הביטוי).

$E \rightarrow E_1 + E_2$

יצירת מקום לתוצאה

E.var = freshVar();

E.code = E_1.code || E_2.code

|| E.var || ':=' || E_1.var || '+' || E_2.var;

יצירת הקוד:

$t_1 \leftarrow E_1$

$t_2 \leftarrow E_2$

$t_3 := t_1 + t_2$

שיטה ראשונה ליצירת קוד - דוגמה

- דוגמה: תרגום ביטויים אריתמטיים
  - נגדיר למשתנה E את התכונות code ו-var

(var יכיל את המשתנה הזמני אליו נשמר ערך הביטוי).

יצירת מקום לתוצאה

יצירת הקוד:
$t_1 \leftarrow E_1$
$t_2 \leftarrow E_2$
$t_3 := t_1 + t_2$

```
Visit (binop& node ) {
    node.var = freshVar();
    visit(child1);
    visit(child2);
    node.code =child1.code || child2.code
        || node.var || ':=' || child1.var || '+' || child2.var;   }
```

שיטה שנייה ליצירת קוד

נחזיק buffer גלובלי שיכיל את כל הקוד שנוצר עד כה.

- קוד חדש יודפס ישירות לתוך הbuffer בעת ביצוע reduce לכלל.
- בעת סיום הגזירה, הbuffer יכיל את כל הקוד שנוצר.

שיטה שנייה ליצירת קוד - דוגמה

- דוגמה: תרגום ביטויים אריתמטיים
  - ל-E נגדיר תכונה var (אין צורך בתכונת code).
  - הפונקציה emit מדפיסה פקודה ל-buffer.

$E \rightarrow E_1 + E_2$ {

E.var = freshVar();

emit(E.var || ":=" || E_1.var || "+" || E_2.var); }

- היכן הקוד של E_1 ו-E_2?

תרגום פשוט - הגדרה

- לכל קטע קוד (המתאים למשתנה גזירה מסויים) נקודת כניסה אחת (בראשיתו) ונקודת יציאה בסופו.
- התוצאה:
  - בתום ביצוע קטע קוד, מבוצע קטע הקוד המשורשר אחריו.
  - עובדה זו מפשטת את התרגום.

הקושי

- כאשר מתרגמים מבנה בקרה, יעדי הקפיצות לא תמיד ידועים.
- לדוגמה: if B then S_1 else S_2
- לכאורה, יש יותר מנקודת יציאה אחת:
  - יש לבצע קטע קוד אחד במקרה והתנאי הבוליאני מתקיים,
  - או קטע קוד אחר במקרה והתנאי אינו מתקיים.
- איך המנתח יידע לאיזה שורה משני קטעי הקוד לקפוץ מיד אחרי קטע הקוד לבדיקת התנאי B?

הפתרון – הוספת תכונות סמנטיות

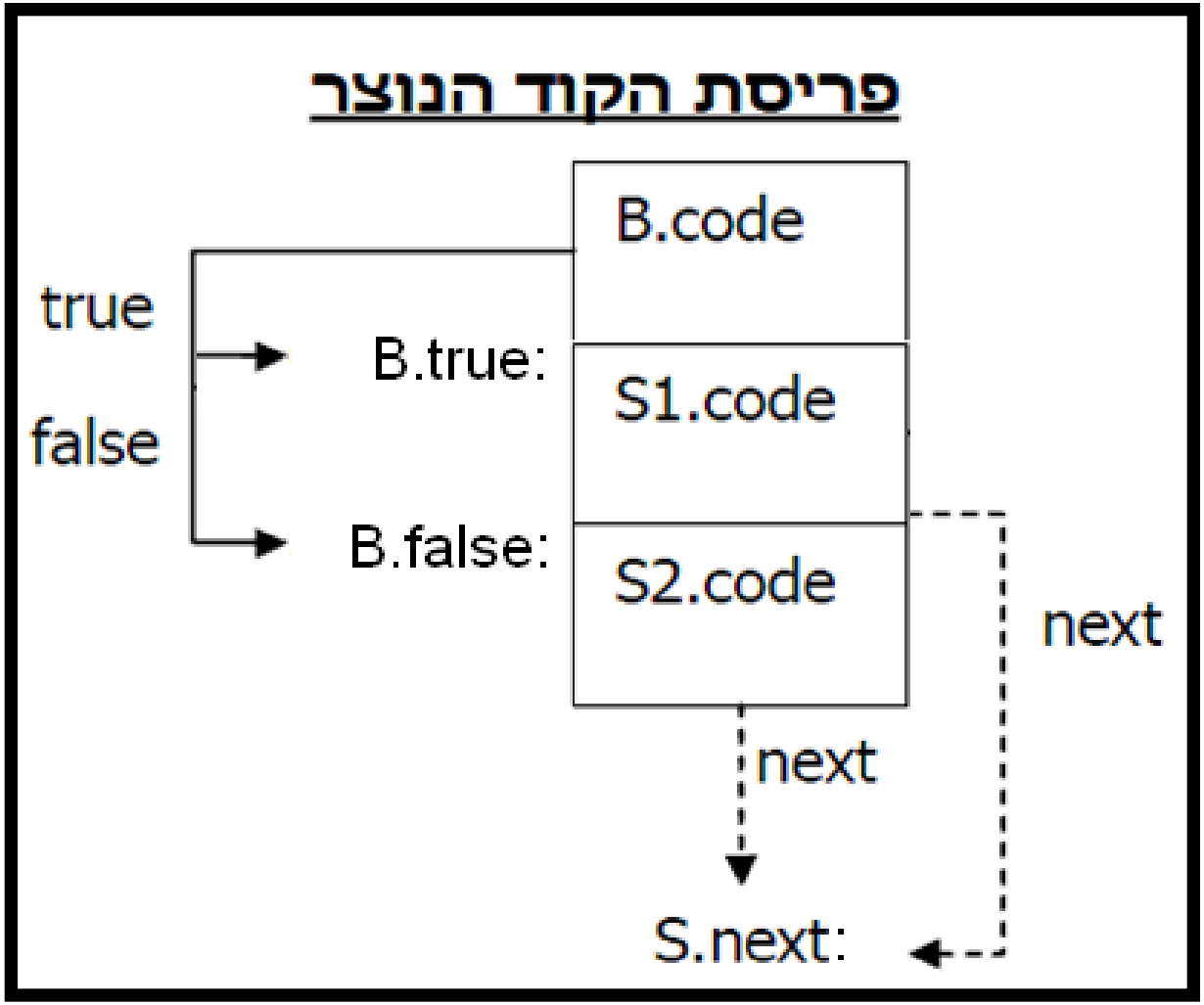
$$S \rightarrow \underline{\text{if}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2$$
$$B \rightarrow B_1\ \underline{\text{or}}\ B_2$$
$$B \rightarrow E_1\ \underline{\text{'>'}}\ E_2$$

התכונות (הנוצרות):

- B.false_label : התווית אליה תעבור התוכנית במקרה שערך הביטוי הוא false.
- B.true_label: התווית אליה תעבור התוכנית אם ערך הביטוי הוא true.
- S.next_label: התווית אליה נעבור בסיום הפקודה S.

דוגמה – תרגום משפט if מורכב

$S \rightarrow$ <u>if</u> B <u>then</u> S_1 <u>else</u> S_2
$B \rightarrow B_1$ <u>or</u> B_2
$B \rightarrow E_1$ <u>'>'</u> E_2



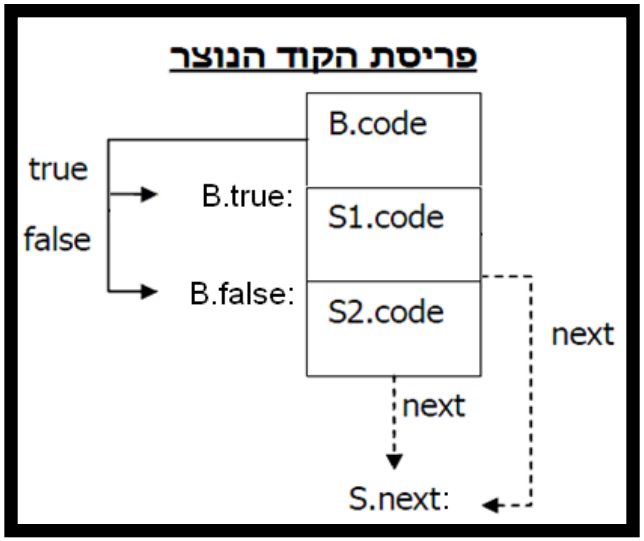
דוגמה – תרגום משפט if מורכב

<u>סכימת התרגום:</u>

S → <u>if</u> B <u>then</u> S_1 <u>else</u> S_2

```
Visit(If & node){
node.b.true_lable = freshLabel(); node.b.false_lable = freshLabel();
node.s1.next_label = freshLabel();
node.s2. .next_label = node.s1.next_label;

}
```



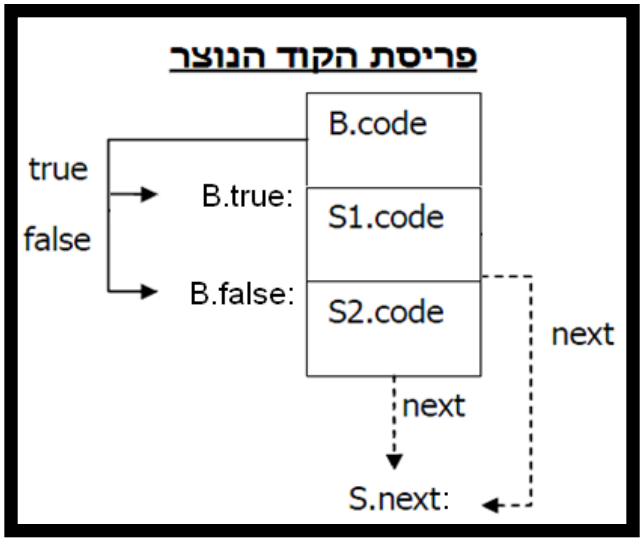
דוגמה – תרגום משפט if מורכב

<u>סכימת התרגום:</u>

S → <u>if</u> B <u>then</u> S_1 <u>else</u> S_2

```
Visit(If & node){
node.b.true_lable = freshLabel(); node.b.false_lable = freshLabel();
node.s1.next_label = freshLabel();
node.s2. .next_label = node.s1.next_label;
Visit(node.b);

}
```



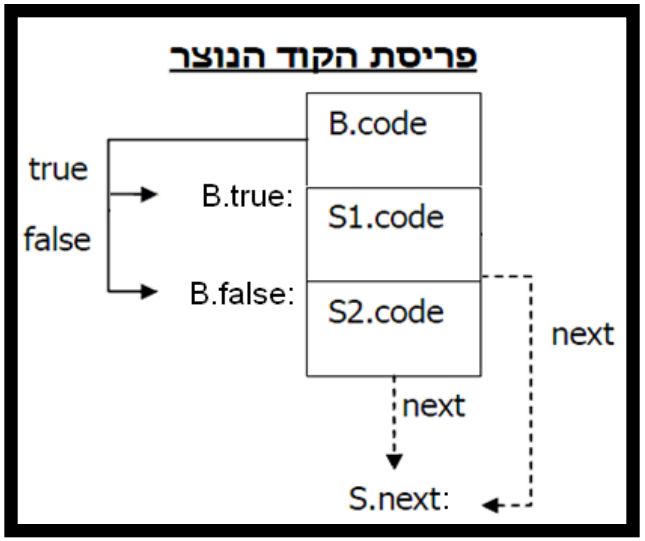
דוגמה – תרגום משפט if מורכב

<u>סכימת התרגום:</u>

S → <u>if</u> B <u>then</u> S_1 <u>else</u> S_2

```
Visit(If & node){
node.b.true_lable = freshLabel(); node.b.false_lable = freshLabel();
node.s1.next_label = freshLabel();
node.s2. .next_label = node.s1.next_label;
Visit(node.b);
emit(node.b.true_lable || ":");

}
```



דוגמה – תרגום משפט if מורכב

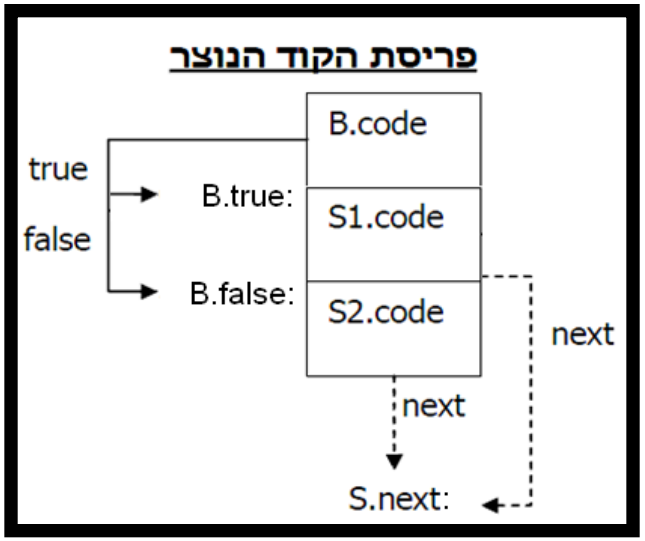
<u>סכימת התרגום:</u>

S → <u>if</u> B <u>then</u> S_1 <u>else</u> S_2

```
Visit(If & node){
node.b.true_lable = freshLabel(); node.b.false_lable = freshLabel();
node.s1.next_label = freshLabel();
node.s2. .next_label = node.s1.next_label;
Visit(node.b);
emit(node.b.true_lable || ":");
Visit(node.s1);
emit(node.bfalse_label || ":");



}
```

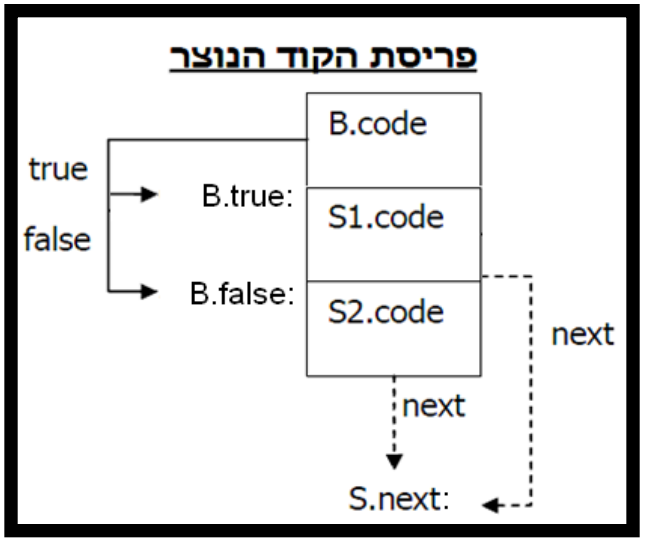


דוגמה – תרגום משפט if מורכב

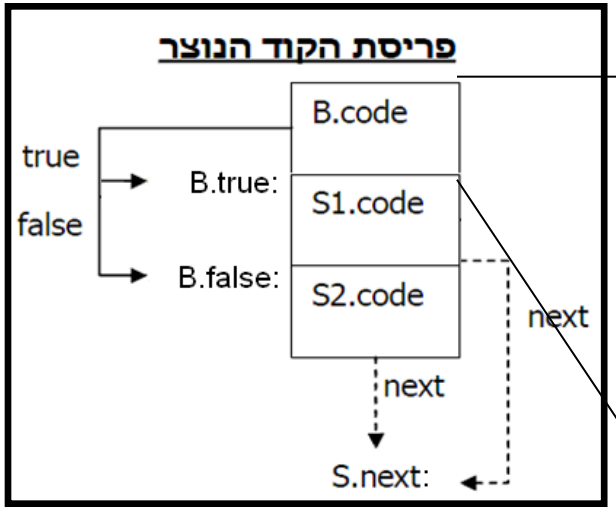
<u>סכימת התרגום:</u>

S → <u>if</u> B <u>then</u> S_1 <u>else</u> S_2

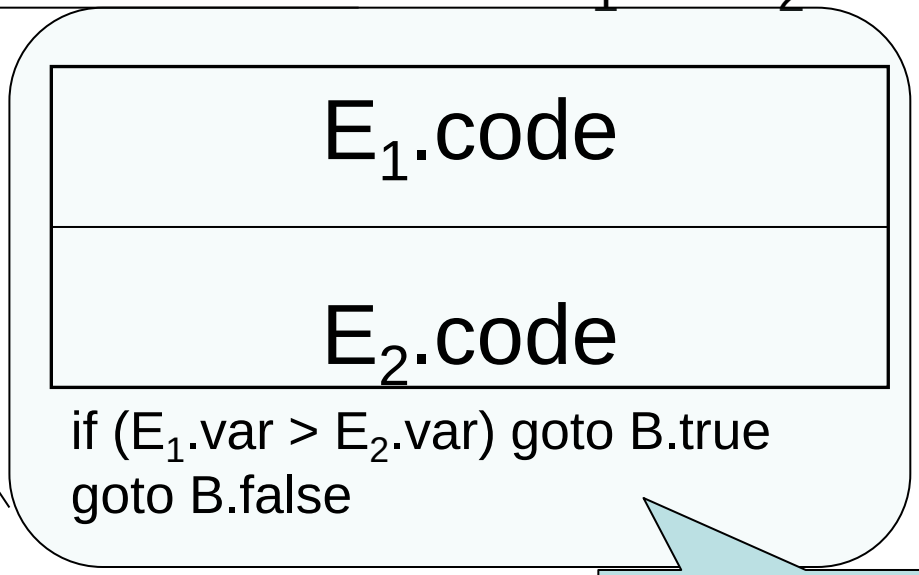
```
Visit(If & node){
node.b.true_lable = freshLabel(); node.b.false_lable = freshLabel();
node.s1.next_label = freshLabel();
node.s2. .next_label = node.s1.next_label;
Visit(node.b);
emit(node.b.true_lable || ":");
Visit(node.s1);
emit(node.bfalse_label || ":");
Visit(node.s2);
emit(node .s1.next_label || ":");
emit("jmp" || node.next);
}
```



המשך הדוגמה – הקוד עבור הביטוי הבוליאני



$B \rightarrow E_1$ '>' E_2



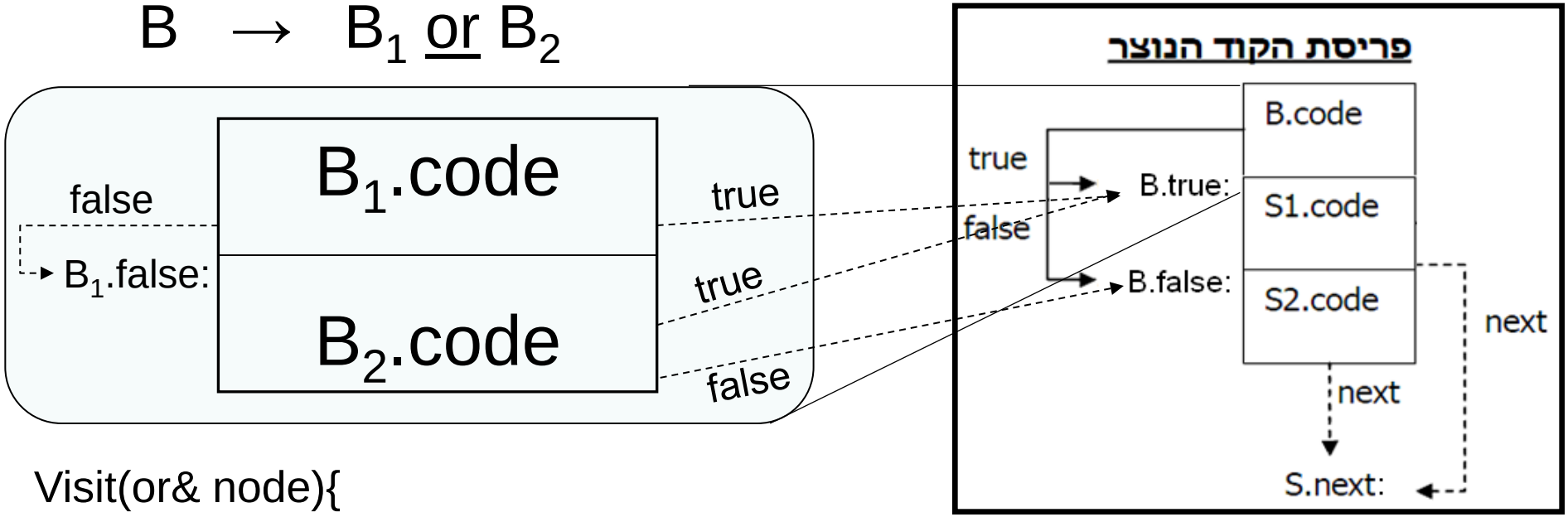
תזכורת: זהו המשתנה הזמני שהוקצה לביטוי (למשל מיקום במחסנית)

B → E1 '>' E2

```
Visit(relop& node)
{
visit(node.e1);
visit(node.e2);
emit('if' || node.e1.var || '>' || node.e2.var || 'goto' || node.true_label);
emit('goto'|| B.false_label) ;
}
```

המשך הדוגמה – הקוד עבור הביטוי הבוליאני

$B \rightarrow B_1 \ \underline{or} \ B_2$

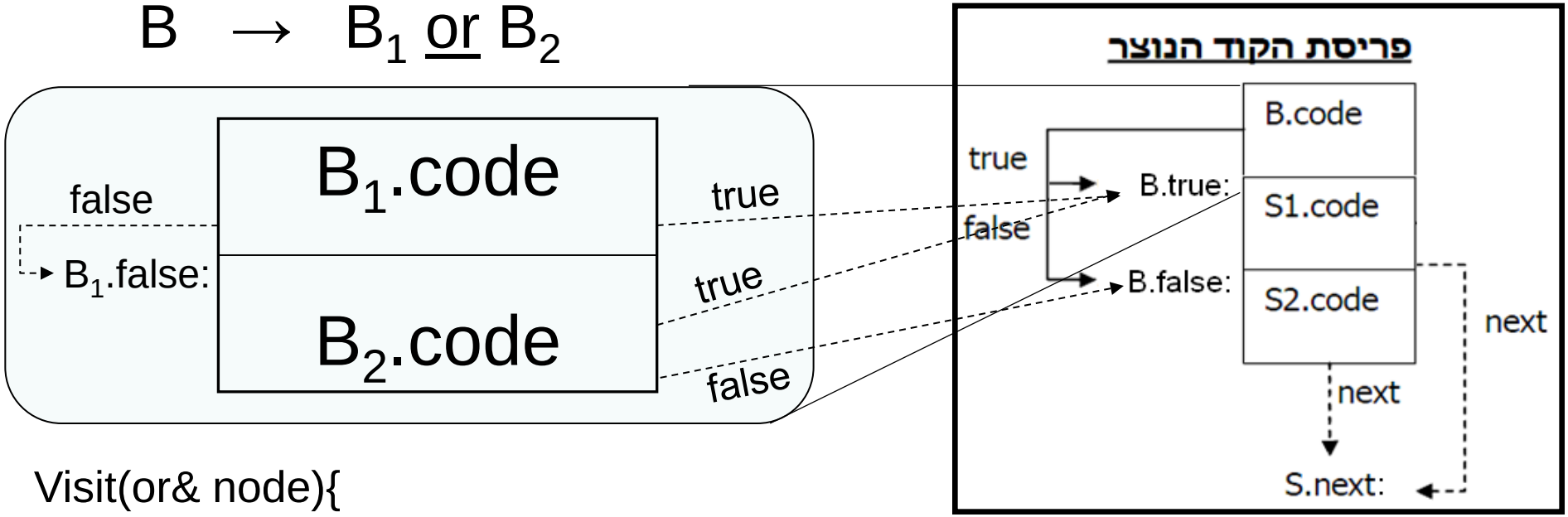


Visit(or& node){

}

המשך הדוגמה – הקוד עבור הביטוי הבוליאני

$B \rightarrow B_1 \underline{\text{or}} B_2$

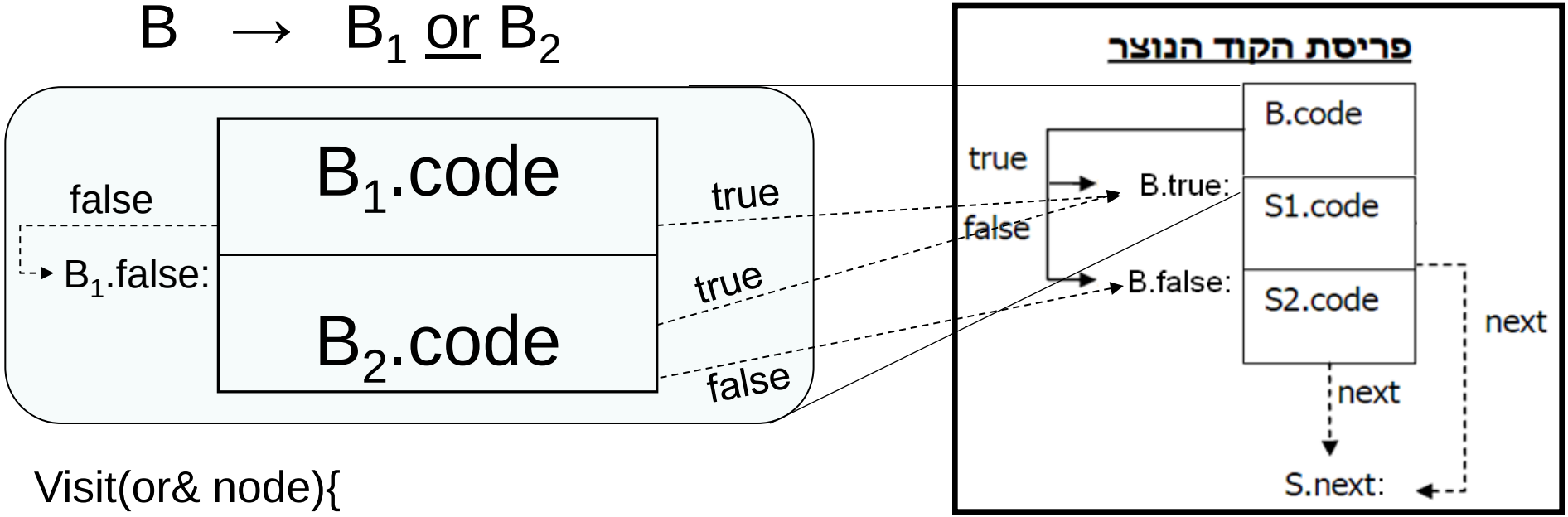



```
Visit(or& node){
node.b1.true_label = node.true_label; node.b1.false_label = freshLabel();
visit(b1);

}
```

המשך הדוגמה – הקוד עבור הביטוי הבוליאני

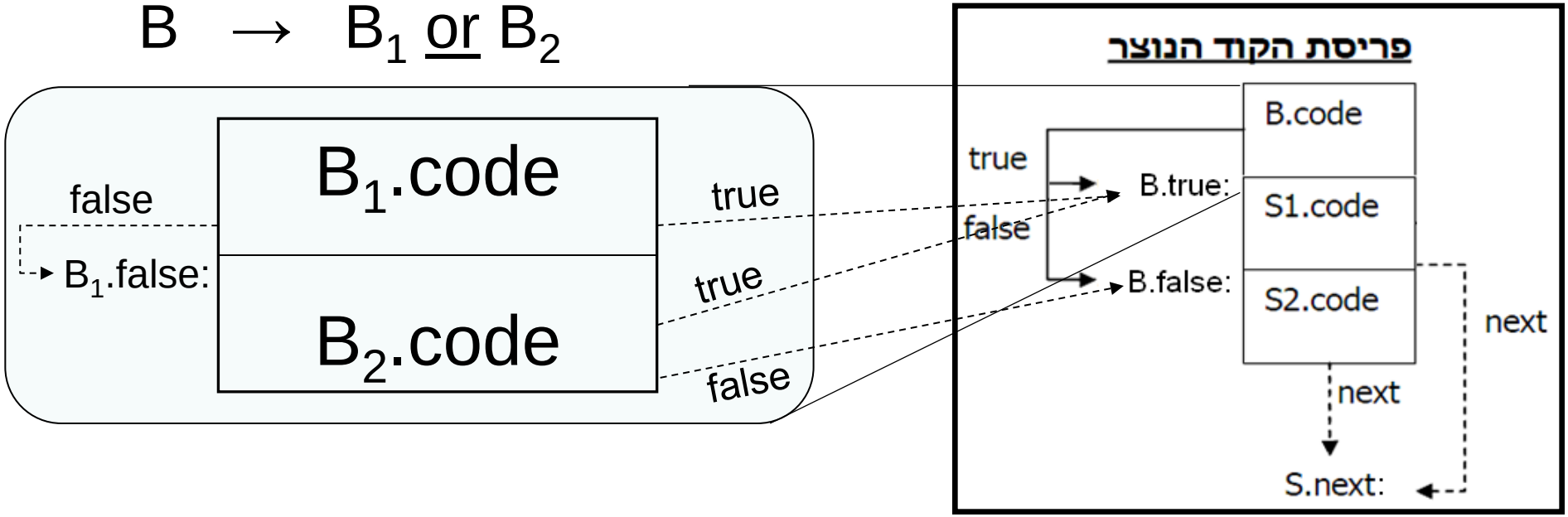
$B \rightarrow B_1 \underline{\text{or}} B_2$




```
Visit(or& node){
node.b1.true_label = node.true_label; node.b1.false_label = freshLabel();
visit(b1);
emit(node.b1.false_label  || ":");
node.b2.true_label = node.true_label; node.b2.false_label = node.false_label;
visit(b2);
}
```

המשך הדוגמה – הקוד עבור הביטוי הבוליאני

$B \rightarrow B_1 \underline{\text{or}} B_2$



הערות:

- הבדיקות מבוצעת בשיטת Short circuit lazy evaluation :
  - ברגע שהתשובה ידועה מפסיקים לחשב.
  - המשמעות היא כמו של האופרטור logical Or ב C.

דוגמא

- מה יהיה פלט המנתח שלנו עבור

if 4<5 then f1() else f2()

$op: <$

$$S \rightarrow if\ \boldsymbol{B}\ then\ \boldsymbol{S_1}\ else\ \boldsymbol{S_2}$$

E_1 E_2 $(f1)$ $(f2)$

4 5

if 4<5 then f1() else f2()

$$S \rightarrow if\ \boldsymbol{B}\ then\ \boldsymbol{S_1}\ else\ \boldsymbol{S_2}$$

פלט

פקודות בבאפר

1. t1=4
2. t2=5
3. if t1<t2 goto B.trueLabel
4. goto B.falseLabel
5. B.trueLabel:
6. f1()
7. goto S.next
8. B.falseLabel:
9. f2()
10. goto S.next

תכונות

1. E1.var = t1
2. E2.var = t2
3. B prints
4. B prints
5. IF prints
6. S1 prints
7. S1 prints
8. IF prints
9. S2 prints
10. S2 prints