

# תכונות (Attributes) (Attributes)

9:31 AM Sunday, April 26, 2020

## Syntax Directed Translation

\* מדריך תרגום סימטי attributes - מתקנים

\* מדריך תרגום סימטי attributes - תפקידים

(Recursive Descent Parsing)

!!! Attribute Grammars \*

## תרגום סימטי attributes

\* מדריך תרגום סימטי attributes

. Expr.type : first.terminal ifExpr.type == second.terminal then

(reduce ->) : הו אפסי הו אפסי הו אפסי

. Expr.type : first.type ifExpr.type == second.type then

Expr → Expr + Term

```
{ if ((second Expr).type == Term.type)
    (first Expr).type = (second Expr).type;
    else error; }
```

$E_1:T$   $E_2:T$

$E_1+E_2:T$

## מבנה של AST

\* רצף של סימטיקות נזקינית כוונתית סימטיות קומפקטיבית

\* סימטיקה של AST

Expr → Expr + Term

```
Expr → Expr1 + Term
{ if (Expr.type == Term.type)
    Expr.type = Expr1.type;
    else error; }
```

\* מבנה סימטיקות AST

## AST - תרשים

\* AST - תרשים

\* סימטיקה של AST

מבחן:

בנוסף ל- $b_1, \dots, b_m$  ב- $a = f(b_1, \dots, b_m)$  מוגדרת  $a$ .

$a$  ב- $f$  מוגדר ב- $b_1, \dots, b_m$ .

$a \leftarrow b_1, \dots, b_m \rightarrow a$  מוגדר ב- $b_1, \dots, b_m$ .

מבחן: מילוי כוון קולע

AST-הו מילוי.

מילוי מילויים.

מילוי מילויים.

מילוי מילויים.

מילוי מילויים.

מבחן 3 - CPAC

מבחן 3 - CPAC: קורס פונקצייתית - מילוי מילויים.

מילוי מילויים.

ל- $L$  ב-type-ה מילוי מילויים.  
מילוי מילויים מילויים.  
מילוי מילויים מילויים.  
מילוי מילויים מילויים.

| Production              | Semantic Rule                                |
|-------------------------|--|
| $D \rightarrow T L$     | $L.in = T.type$                              |
| $T \rightarrow int$     | $T.type = integer$                           |
| $T \rightarrow float$   | $T.type = float$                             |
| $L \rightarrow L_1, id$ | $L_1.dt = L.dt$<br>$addType(id.entry, L.dt)$ |
| $L \rightarrow id$      | $addType(id.entry, L.dt)$                    |

| Production              | Semantic Rule  |
|-------------------------|--|
| $D \rightarrow T L$     | $L.in = T.type$                                      |
| $T \rightarrow int$     | $T.type = integer$                                   |
| $T \rightarrow float$   | $T.type = float$                                     |
| $L \rightarrow L_1, id$ | $L_1.dt = L.dt$<br>$L.dmy = addType(id.entry, L.dt)$ |
| $L \rightarrow id$      | $L.dmy = addType(id.entry, L.dt)$                    |

מבחן 3 - CPAC:

מילוי מילויים.

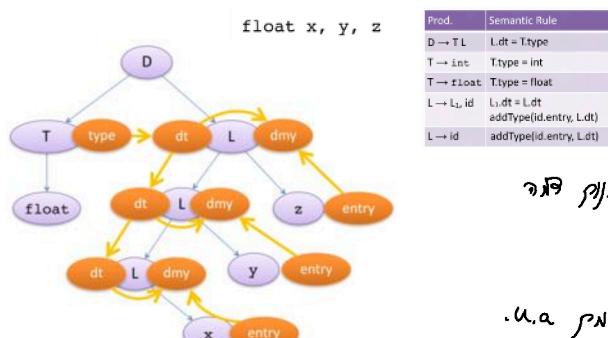
$attr_1 = f_1(attr_1, attr_2, \dots)$   
 $attr_2 = f_2(attr_1, attr_2, \dots)$

מילוי מילויים.

מילוי מילויים.

מילוי מילויים.

מילוי מילויים.

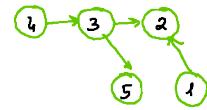


לינר קולר:

$|V|=k$ ,  $G=(V,E)$  גראף גיאומטרי.

רעיון הכל: רצף  $\langle v_1, v_2, \dots, v_k \rangle$  - גראף גיאומטרי.

לינר קולר: זוגות  $(v_i, v_j) \in E$



$\langle 14352 \rangle, \langle 43512 \rangle$ : לא מושג אוניברסלי.

? פתרון מה נון?

לפנינו קולר גיאומטרי קומפקט או לא-קומפקט.

- סימוכין יסוד (למשל).

- צורה של רוחב שטח (בנוסף לאלגוריתם).

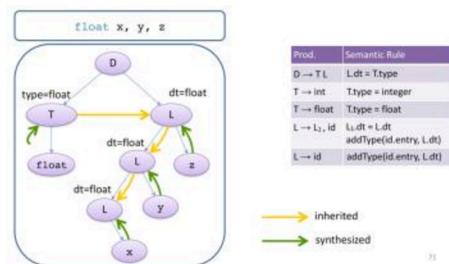
- (לא) קולר גיאומטרי.

לפנינו קולר גיאומטרי לא-קומפקט.

## הכוריק נווקודט (Inherited) או סינטזיזט (Synthesized)

- \* **הכוריק רותך ורץ:** תכורה של ה-*תבנית* לאומת רשותה ב-*תפקידים* של *טבלה* ו-*טבלה* של *טבלה*.
- \* **הכוריק סינטזיזט:** תכורה של ה-*תבנית* לאומת רשותה ב-*טבלה* של *טבלה* של *טבלה*.
- \* גדרון של דיוויד נחיש ב-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*.

: תולב

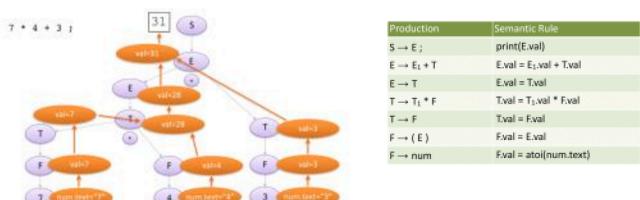


## : S-Attributed תרגום

- \* קבוצת פונקציות ב-*תפקידים* נוכחות.
- \* מושגנו רוח תכורה נווקודט (S-attributed).
- \* מושגנו תכורה נווקודט (סינטזיזט) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- \* מושגנו תכורה נווקודט (סינטזיזט) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- \* מושגנו תכורה נווקודט (סינטזיזט) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- \* מושגנו תכורה נווקודט (סינטזיזט) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- \* מושגנו תכורה נווקודט (סינטזיזט) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- \* מושגנו תכורה נווקודט (סינטזיזט) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).

: S-Attributed תרגום תולב

\* מושגנו תכורה נווקודט (סינטזיזט) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).



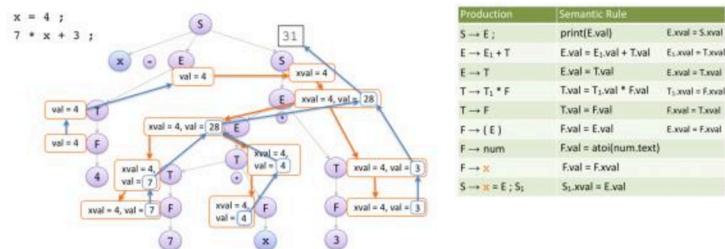
## : L-Attributed תרגום

- \* מושגנו תכורה נווקודט (L-Attributed) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- **L -** תכורה נווקודט (L-Attributed) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- מושגנו תכורה נווקודט (L-Attributed) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- מושגנו תכורה נווקודט (L-Attributed) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).
- מושגנו תכורה נווקודט (L-Attributed) (ה-*תבנית* ב-*טבלה* של *טבלה* של *טבלה*).

א ב מילויים נזק -

ו S-Attributed ו T-Attributed נסוברים במאמר.

## L-Attributed באלגוריתם נסוברים



## טבלה סופר של L-Attributed באלגוריתם

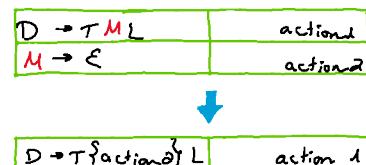
במקרה זה מושג ב (Top-Down) Recursive-Descent.

במקרה זה מושג ב (Bottom-Up) Shift-Reduce.

(Marker Variables) יונן ערך.

## (Marker Variables) יונן ערך

בכדי לאפשר מילויים נזק, ניתן לשים את כל הפעולות בפונקציית מילויים.



(mid-rule action) סדרת מילויים action1 action2.

במקרה זה מושג ב (Bottom-Up) Shift-Reduce.

במקרה זה מושג ב (Top-Down) Recursive-Descent.

במקרה זה מושג ב (Top-Down) Recursive-Descent.

במקרה זה מושג ב (Bottom-Up) Shift-Reduce.

## דיאוד:

הנראה דיאוד מילויים נזק (bottom-up).

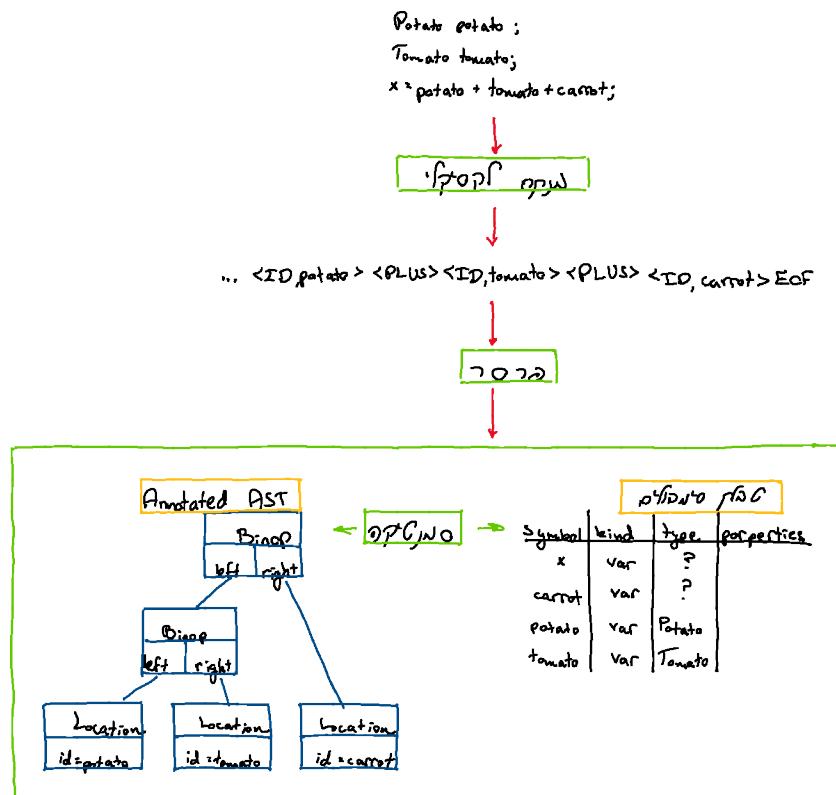
במקרה זה מילויים נזק מושג ב (Top-Down) Recursive-Descent.

במקרה זה מילויים נזק מושג ב (Bottom-Up) Shift-Reduce.

L-attributed, S-attributed: מילויים נזק מושגים ב (Top-Down) Recursive-Descent.

במקרה זה מילויים נזק מושגים ב (Bottom-Up) Shift-Reduce.

## דקדוק-הפקה קומילטיבית



'carrot' is undefined    'potato' used before initialized    Cannot add 'Potato' and 'Tomato'

הנום און און

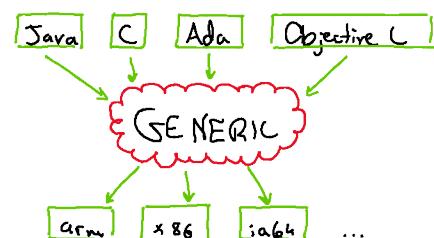
לפניהם מושג שפה סטנדרטית ומיינדרטיבית.

back-end - front end מושג שפה סטנדרטית ומיינדרטיבית.

לפניהם מושג שפה סטנדרטית ומיינדרטיבית.

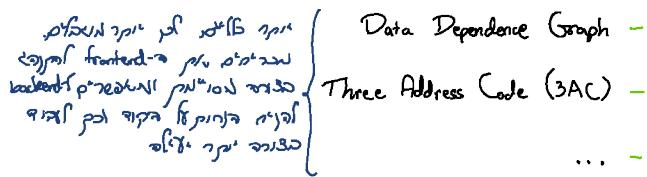
לפניהם מושג שפה סטנדרטית ומיינדרטיבית.

(כ-ל-ב גע-ן GENERIC (gen) פאון דאסט פאון IR פאון.



:IR le ore ova \*

, סיבי וריאנטים מודולריים ב-AST Annotated AST -



לפנינו מוצג תהליכי קידום מודולריים ב-AST

C ה- basic building blocks שלו. C מודולריים יוצרים generic program mode ב-LLVM.

C מודולריים יוצרים generic program mode ב-LLVM.

:Annotated AST - דוגמאות

| production                | semantic rule  |
|---------------------------|--|
| $S \rightarrow id := E$   | $S.\text{nptr} = \text{new Node('assign', new Node('id', id.entry), E.\text{nptr})}$ |
| $E \rightarrow E_1 + E_2$ | $E.\text{nptr} = \text{new Node('+', E_1.\text{nptr}, E_2.\text{nptr})}$             |
| $E \rightarrow E_1 * E_2$ | $E.\text{nptr} = \text{new Node('*', E_1.\text{nptr}, E_2.\text{nptr})}$             |
| $E \rightarrow -E_1$      | $E.\text{nptr} = \text{new Node('uminus', E_1.\text{nptr})}$                         |
| $E \rightarrow (E_1)$     | $E.\text{nptr} = E_1.\text{nptr}$  |
| $E \rightarrow id$        | $E.\text{nptr} = \text{new Node('id', id.entry)}$                                    |

.ה-3 - new Node (op, operand, [operands]) \*

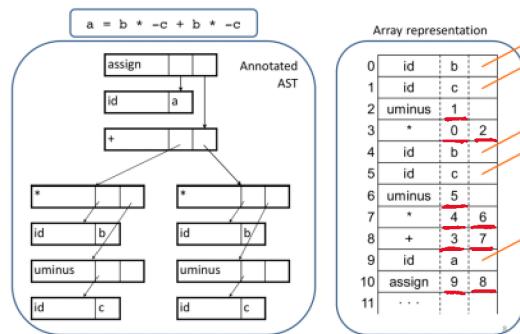
.ה-3 - new Node ('id', id.entry) \*

.ה-3 - new Node ('id', id.entry) \*

ה-3 - new Node ('id', id.entry) \*

ה-3 - new Node ('id', id.entry) \*

:ת. נ. ד. \*



\* לא מושג מהטבלה  
ה-3 - new Node ('id', id.entry) \*

## :Three Address Code (3AC)

רезультат של 3 מטריצות ב-3 כוּם ו-

result := operand1 operator operand2

\* גורם אחד שמייצג את הערך ומיופיע בכל מטריצה

- גורם אחד שמייצג את הפעלה

\* 3AC-ה מגדיר את הפעלה כפונקציית מילוי של הפעלה

\* גורם אחד שמייצג את הערך או הכתובת של הערך

## :3AC מילוק הנקודות

| הנקודות          | הפעולות                              |
|------------------|--------------------------------------|
| $x := 502$       |                                      |
| $x := 0$         |                                      |
| $x := y$         |                                      |
| $x := &y$        |                                      |
| $x := *y$        |                                      |
| $*x = y$         |                                      |
| $x := y[i]$      |                                      |
| $t_1 := &y$      | ה- $t_1$ אינט y ל-ptr ימויים         |
| $t_2 := t_1 + i$ | ה- $t_2$ אינט $t_1 + i$ ל-ptr ימויים |
| $x := *t_2$      | ה- $x$ אינט $t_2$ ל-deref ימויים     |
| $x[i] := y$      |                                      |
| $t_1 := &x$      | ה- $t_1$ אינט x ל-ptr ימויים         |
| $t_2 := t_1 + i$ | ה- $t_2$ אינט $t_1 + i$ ל-ptr ימויים |
| $*t_2 = y$       | ה- $t_2$ ל-deref ימויים              |

## :3AC ו-ptrים ו-addr

:3AC מילוק הנקודות \*

$x := y[i]$

$t_1 := &y$  ה- $t_1$  אינט y ל-ptr ימויים  
 $t_2 := t_1 + i$  ה- $t_2$  אינט  $t_1 + i$  ל-ptr ימויים  
 $x := *t_2$  ה- $x$  אינט  $t_2$  ל-deref ימויים

$x[i] := y$

$t_1 := &x$  ה- $t_1$  אינט x ל-ptr ימויים  
 $t_2 := t_1 + i$  ה- $t_2$  אינט  $t_1 + i$  ל-ptr ימויים  
 $*t_2 = y$  ה- $t_2$  ל-deref ימויים

ה- $t_1$  מילוק הנקודות 3 מטריצות ב-3 כוּם ו-

.ptr מילוק הנקודות.

\* מילוק הנקודות מילוק הנקודות ב-3 כוּם ו-

\* מילוק הנקודות מילוק הנקודות ב-3 כוּם ו-

ה- $t_1$  מילוק הנקודות int

## :3AC ביר

בנוסף ל-3AC ישנו סעיפים נוספים ב-3AC שיכולים להיות מוגדרים:

לפניהם נקבעו דיבריה:

לפניהם נקבעו דיבריה:

לפניהם נקבעו דיבריה:

לפניהם נקבעו דיבריה:

לפניהם נקבעו דיבריה.

בנוסף ל-3AC ישנו סעיפים נוספים ב-3AC שיכולים להיות מוגדרים:

## :3AC מז'

לעתה נראה כיצד ניתן ליצור קידמיון של 3AC מז'?

לפניהם נקבעו דיבריה:

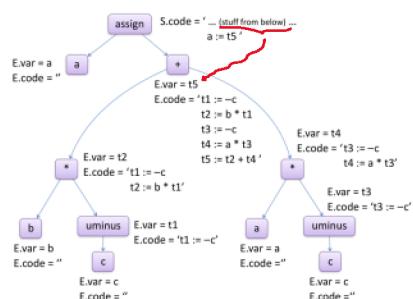
Syntax Directed Translation של 3AC מז' \*

לפניהם נקבעו דיבריה:

לפניהם נקבעו דיבריה:

לפניהם נקבעו דיבריה:

## :PCTN - 3AC ביר מז'



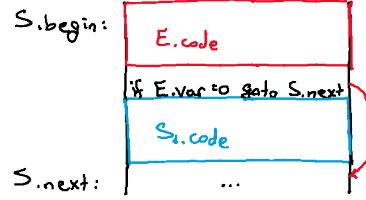
| production                | semantic rule  |
|---------------------------|--|
| $S \rightarrow id := E$   | $S.code = E.code    (id.name := E.var)$  |
| $E \rightarrow E_1 + E_2$ | $E.var = freshVar();$<br>$E.code = E_1.code    E_2.code    (E.var := E_1.var + E_2.var)$ |
| $E \rightarrow E_1 * E_2$ | $E.var = freshVar();$<br>$E.code = E_1.code    E_2.code    (E.var := E_1.var * E_2.var)$ |
| $E \rightarrow - E_1$     | $E.var = freshVar();$<br>$E.code = E_1.code    (E.var := -E_1.var)$                      |
| $E \rightarrow ( E_1 )$   | $E.var = E_1.var;$<br>$E.code = E_1.code$  |
| $E \rightarrow id$        | $E.var = id.name;$<br>$E.code = ''$  |

ולפניהם נקבעו דיבריה:

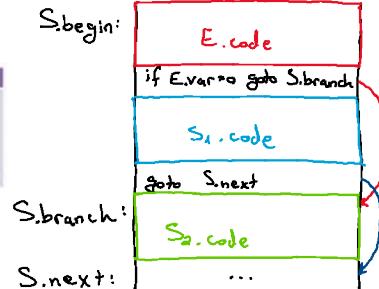
## :תגון - 3AC ביר מז'

לפניהם נקבעו דיבריה:

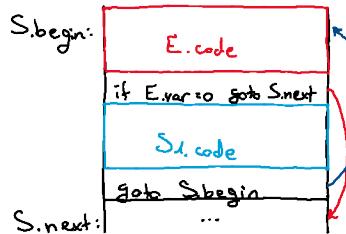
| production   | semantic action   |
|--|---|
| $S \rightarrow \text{if } E \text{ then } S_1 \text{ else } S_2$ | <pre> S.begin = freshLabel(); S.next = freshLabel(); S.code = (S.begin ':')    E.code              ('if E.var != 0' goto' S.next)              S1.code    (S.next ':')         </pre> |



| production   | semantic action   |
|--|---|
| $S \rightarrow \text{if } E \text{ then } S_1 \text{ else } S_2$ | <pre> S.begin = freshLabel(); S.branch = freshLabel(); S.next = freshLabel(); S.code = (S.begin ':')    E.code              ('if E.var != 0' goto' S.branch)    S1.code              ('goto' S.next )    (S.branch ':')    S2.code    (S.next ':')         </pre> |



| production                                      | semantic action   |
|---|---|
| $S \rightarrow \text{while } E \text{ do } S_1$ | <pre> S.begin = freshLabel(); S.next = freshLabel(); S.code = (S.begin ':')    E.code              ('if E.var == 0' goto' S.begin)    S1.code              ('goto' S.begin )    (S.next ':')         </pre> |



3AC 434

איך יגערו פונקציית \*

היפר פונקציית מילויים מילויים \*

(op, arg1, arg2, result) מילויים מילויים מילויים \*

מילויים מילויים מילויים מילויים \*

|     | op     | arg1 | arg2 | result |
|-----|--------|------|------|--------|
| (1) | uminus | c    |      | t1     |
| (2) | *      | b    | t1   | t2     |
| (3) | uminus | c    |      | t3     |
| (4) | *      | b    | t3   | t4     |
| (5) | +      | t2   | t4   | t5     |
|     | :      | t5   |      | a      |

## הנחתה בheap

\* הנחתה בheap מושג בפונקציית `enter`

.ptr\_to\_val = -

.ptr\_to\_heap = -  
ptr\_to\_struct = -

(offset -> גורם שמשתמש בptr\_to\_struct)

(ptr\_to\_struct = גורם שמשתמש בptr\_to\_heap)

: attributes a struct -

.ptr\_to\_struct = offset -

.ptr\_to\_struct = width -

ptr\_to\_struct = גורם שמשתמש בptr\_to\_struct  
!Top-Down 방식

.ptr\_to\_struct = offset -> גורם שמשתמש בptr\_to\_struct

.ptr\_to\_struct = offset -> גורם שמשתמש בptr\_to\_struct

.ptr\_to\_struct = offset -> גורם שמשתמש בptr\_to\_struct

| production               | semantic action   |
|--------------------------|---|
| $P \rightarrow D$        | $D.offset = 0$  |
| $D \rightarrow D_1 D_2$  | $D_1.offset = D.offset;$<br>$D_2.offset = D.offset + D_1.width;$<br>$D.width = D_1.width + D_2.width$ |
| $D \rightarrow T id_2$   | <code>enter(id.name, T.type, D.offset); D.width = T.width</code>                                      |
| $T \rightarrow int$      | $T.type = int; T.width = 4$   |
| $T \rightarrow long$     | $T.type = long; T.width = 8$  |
| $T \rightarrow float$    | $T.type = float; T.width = 4$   |
| $T \rightarrow T_1[num]$ | $T.type = array(num.vol, T_1.type); T.width = num.vol * T_1.width$                                    |
| $T \rightarrow *T_1$     | $T.type = pointer(T_1.type); T.width = 4$   |

ptr\_to\_struct = גורם שמשתמש בptr\_to\_struct  
ptr\_to\_struct = offset -> גורם שמשתמש בptr\_to\_struct

ptr\_to\_struct = גורם שמשתמש בptr\_to\_struct

= Bottom Up ↴ היררכיה

:Bottom-Up גורם שמשתמש בptr\_to\_struct \*

| production               | semantic action  |
|--------------------------|--|
| $P \rightarrow M D$      |  |
| $M \rightarrow E$        | $offset = 0$   |
| $D \rightarrow D D$      |  |
| $D \rightarrow T id_2$   | <code>enter(id.name, T.type, offset); offset += T.width</code>     |
| $T \rightarrow int$      | $T.type = int; T.width = 4$  |
| $T \rightarrow long$     | $T.type = long; T.width = 8$                                       |
| $T \rightarrow float$    | $T.type = float; T.width = 4$                                      |
| $T \rightarrow T_1[num]$ | $T.type = array(num.vol, T_1.type); T.width = num.vol * T_1.width$ |
| $T \rightarrow *T_1$     | $T.type = pointer(T_1.type); T.width = 4$                          |

# יצירת קוד IR

יום ראשון 03 במאי 2020 21:18

:IR בירכט

\* 2 סעיפים:

1. AST → IR (attributed) #1

incremental translation  
2. IR → קידוד סרגן: מתרגסן IR → קידוד סרגן #2

לפנינו מוגדרת מילה מושבנית (word) כGROUP 33, כלומר מילה מושבנית מוגדרת כGROUP 33.

ונא דעתה נזכיר.

במקרה הבא יתבצע אינטראקציית מילויים בין מילים מושבניות ומשמעותם.

במיון כל מילה מושבנית (group 33) רוחנה מילה מושבנית.

:ריצוף מילים

מילים מושבניות - exit

| production                       | semantic action   |
|----------------------------------|---|
| $S \rightarrow id := E$          | $p := \text{lookup}(id.name);$ if $p \neq \text{null}$ then <b>emit(p :=' E.var)</b> else error |
| $E \rightarrow E_1 \diamond E_2$ | $E.var := \text{freshVar}();$ <b>emit(E.var :=' E_1.var \diamond E_2.var)</b>                   |
| $E \rightarrow - E_1$            | $E.var := \text{freshVar}();$ <b>emit(E.var :=' 'uminus' E_1.var)</b>                           |
| $E \rightarrow ( E_1 )$          | $E.var := E_1.var$  |
| $E \rightarrow id$               | $p := \text{lookup}(id.name);$ if $p \neq \text{null}$ then $E.var := p$ else error             |

◊ סימני טרנסלט – מילויים מושבניים  
סימני טרנסלט – מילויים מושבניים  
סימני טרנסלט – מילויים מושבניים  
סימני טרנסלט – מילויים מושבניים

:ריצוף מילים

| production                      | semantic action   |
|---------------------------------|---|
| $E \rightarrow E_1 op E_2$      | $E.var := \text{freshVar}();$ <b>emit(E.var :=' E_1.var op E_2.var)</b> |
| $E \rightarrow \text{not } E_1$ | $E.var := \text{freshVar}();$ <b>emit(E.var :=' 'not' E_1.var)</b>      |
| $E \rightarrow ( E_1 )$         | $E.var := E_1.var$  |
| $E \rightarrow \text{true}$     | $E.var := \text{freshVar}();$ <b>emit(E.var :=' '1')</b>                |
| $E \rightarrow \text{false}$    | $E.var := \text{freshVar}();$ <b>emit(E.var :=' '0')</b>                |

op ∈ {and, or} !  
אנו שופע מילויים מושבניים true/false ← כ"ז  
ריצוף מילים

| production                          | semantic action  |
|-------------------------------------|--|
| $E \rightarrow id_1 \triangle id_2$ | $E.var := \text{freshVar}();$<br><b>emit('if' id_1.var <math>\triangle</math> id_2.var 'goto' nextInstr+3);</b><br><b>emit(E.var :=' '0');</b><br><b>emit('goto' nextInstr+2);</b><br><b>emit(E.var :=' '1')</b> |

▷ סימני טרנסלט – מילויים מושבניים

loop: if a < b goto loop (nextInstr=loop: מילויים מושבניים) \*

loop: t1:=0

loop: goto loop

loop: t1:=1

loop: ...

Short-circuit, so when a < b, it can skip the comparison.

## :(Short-Circuit Evaluation) ↗ בירן פוליה

אנו יתרכז על פונקציית **וריאנט שORTHOCODE** \*  
**if x then y else false : if true then (x and y) -**

**if x then true else y : if true then (x or y) -**  
**.הזהר שORTHOCODE מושך את הוראות \***

**short-circuit → מילוי הוראות סידוריים מימין למשמאל** \*

**ולפיה גוררת מושך הוראות סידוריים מימין למשמאל** \*

**ולפיה גוררת מושך הוראות סידוריים מימין למשמאל** \*

**ולפיה גוררת מושך הוראות סידוריים מימין למשמאל** \*

**(sin), (sqrt), (random)** \*  
**ולפיה גוררת מושך הוראות סידוריים מימין למשמאל** \*

## לכט נסיעה

**S = if B then S<sub>1</sub>  
 if B then S<sub>1</sub> else S<sub>2</sub>  
 | while B do S<sub>1</sub>** \*  
**:הנתק \***

**ולפיה גוררת מושך הוראות סידוריים מימין למשמאל** \*

|                                  |               |
|----------------------------------|---------------|
| false → מושך B ונקרא בולסן בולסן | :B.falseLabel |
| true → מושך B ונקרא בולסן בולסן  | :B.trueLabel  |

**ולפיה גוררת מושך הוראות סידוריים מימין למשמאל** \*

**S = next : מושך הוראות סידוריים מימין למשמאל** \*

**.ולפיה גוררת מושך trueLabel-1 falseLabel מושך הוראות סידוריים מימין למשמאל** \*

## לכט נסיעה

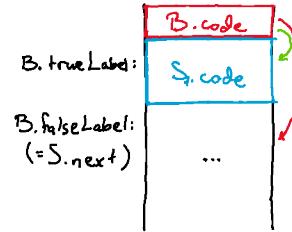
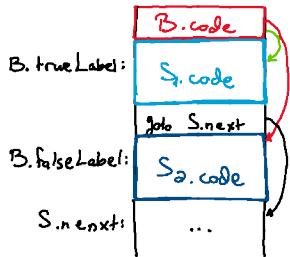
| production                        | semantic action   |
|-----------------------------------|---|
| P → S                             | S.next = freshLabel();<br>P.code = S.code    (S.next ':')   |
| S → S <sub>1</sub> S <sub>2</sub> | S <sub>1</sub> .next = freshLabel();<br>S <sub>2</sub> .next = S.next;<br>S.code = S <sub>1</sub> .code    (S <sub>1</sub> .next ':')    S <sub>2</sub> .code |

**.ולפיה גוררת מושך trueLabel-1 falseLabel מושך הוראות סידוריים מימין למשמאל** \*

## לכט נסיעה-דינמי

| production  | semantic action  |
|---|--|
| S → if B then S <sub>1</sub><br>else S <sub>2</sub> | B.trueLabel = freshLabel();<br>B.falseLabel = freshLabel();<br>S <sub>1</sub> .next = S.next;<br>S <sub>2</sub> .next = S.next;<br>S.code =<br>B.code    (B.trueLabel ':')    S <sub>1</sub> .code    ('goto' S.next)<br>   (B.falseLabel ':')    S <sub>2</sub> .code |

| production                   | semantic action   |
|------------------------------|---|
| S → if B then S <sub>1</sub> | B.trueLabel = freshLabel();<br>B.falseLabel = S.next;<br>S <sub>1</sub> .next = S.next;<br>S.code = B.code    (B.trueLabel ':')    S <sub>1</sub> .code |



: פונקציית סינון

| production  | semantic action   |
|---|---|
| $B \rightarrow B_1 \text{ or } B_2$               | $B_1.\text{trueLabel} = B.\text{trueLabel};$<br>$B_1.\text{falseLabel} = \text{freshLabel};$<br>$B_2.\text{trueLabel} = B.\text{trueLabel};$<br>$B_2.\text{falseLabel} = B.\text{falseLabel};$<br>$B.\text{code} = B_1.\text{code}    (B_1.\text{falseLabel} ':')    B_2.\text{code}$                       |
| $B \rightarrow B_1 \text{ and } B_2$              | $B_1.\text{trueLabel} = \text{freshLabel};$<br><del><math>B_1.\text{falseLabel} = B.\text{falseLabel};</math></del><br>$B_2.\text{trueLabel} = B.\text{trueLabel};$<br>$B_2.\text{falseLabel} = B.\text{falseLabel};$<br>$B.\text{code} = B_1.\text{code}    (B_1.\text{trueLabel} ':')    B_2.\text{code}$ |
| $B \rightarrow \text{not } B_1$                   | $B_1.\text{trueLabel} = B.\text{falseLabel};$<br>$B_1.\text{falseLabel} = B.\text{trueLabel};$<br>$B.\text{code} = B_1.\text{code};$  |
| $B \rightarrow ( B_1 )$                           | $B_1.\text{trueLabel} = B.\text{trueLabel};$<br>$B_1.\text{falseLabel} = B.\text{falseLabel};$<br>$B.\text{code} = B_1.\text{code}    (\text{if } B_1.\text{trueLabel} \text{ goto } B_1.\text{trueLabel})    (\text{goto } B_1.\text{falseLabel})$   |
| $B \rightarrow \text{id}_1 \triangle \text{id}_2$ | $B.\text{code} = (\text{if } \text{id}_1.\text{var} \triangle \text{id}_2.\text{var} \text{ goto } B_1.\text{trueLabel})    (\text{goto } B_1.\text{falseLabel})$   |
| $B \rightarrow \text{true}$                       | $B.\text{code} = \text{'goto' } B.\text{trueLabel}$   |
| $B \rightarrow \text{false}$                      | $B.\text{code} = \text{'goto' } B.\text{falseLabel}$  |

?  $B.\text{falseLabel}$  הינה היעד של הוצאתם \*

- פונקציית סינון בירך בזירה יתבצע על כל הנקודות שמשמשות כיעדים

: מילוי גלגולים

. (IR -> ST, LR, PC) LR -> ST -> IR -> PC -> מילוי גלגולים

? מילוי גלגולים יתבצע לאחר כל הנקודות שמשמשות כיעדים \*

: Backpatching - מילוי גלגולים

\* מילוי גלגולים יתבצע לאחר כל הנקודות שמשמשות כיעדים.

\* מילוי גלגולים יתבצע לאחר כל הנקודות שמשמשות כיעדים.

\* מילוי גלגולים: רצף של הנקודות שמשמשות כיעדים (ST, LR, PC) יתבצע מילוי גלגולים (LR, PC) ולבסוף מילוי גלגולים (ST).

. מילוי גלגולים יתבצע לאחר כל הנקודות שמשמשות כיעדים

: מילוי גלגולים (LR, PC) ולבסוף מילוי גלגולים (ST)

- מילוי גלגולים יתבצע לאחר כל הנקודות שמשמשות כיעדים (LR, PC) ולבסוף מילוי גלגולים (ST) - B.trueList

(B.trueLabel) true ורשות B לפרט

מילוי גלגולים יתבצע לאחר כל הנקודות שמשמשות כיעדים (LR, PC) ולבסוף מילוי גלגולים (ST) - B.falseList

(B.falseLabel) false ורשות B לפרט

? מילוי גלגולים יתבצע לאחר כל הנקודות שמשמשות כיעדים (LR, PC) ולבסוף מילוי גלגולים (ST) -

{ גלגולים יתבצעים לפני גלגולים לאחר. פונקציית סינון תבצע את הגלגולים לאחר גלגולים לפני. }

- מילוי גלגולים יתבצע לאחר כל הנקודות שמשמשות כיעדים (LR, PC) ולבסוף מילוי גלגולים (ST) -

• **תא נרכז** (תאי נרכז) הם גורמים קלאבינג המתגדרים כטיטרליזר לברוח (טיטרליזר) אשר מגדירים את תאי ה-CD4.

\* **לעומת** מילויים, מילויים נקראים **מילויים מודולריים**.

\* **סלאם** כפוזה או פוזה רגילה (ב) כפוזה דיאlect נורווגית הנורווגית כפוזה (ב-נורווגית)

\* **ਹੈਰਿਊਲਡ ਕਾਨੂੰਨ** ਦੀ ਸੰਖੇਪ ਰੂਪ ਨੂੰ ਕਿਵੇਂ ਕਿਸੇ ਜਾਣੇ?

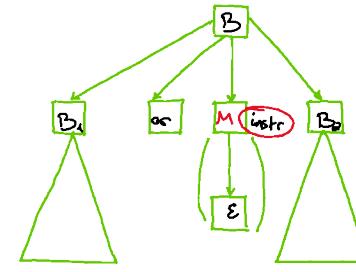
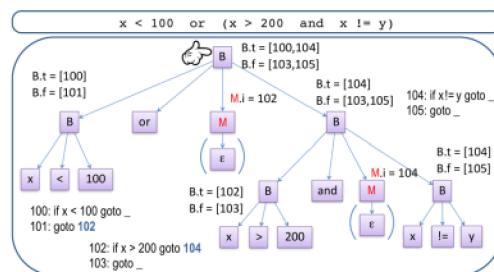
לעומת זה, מטרת ה-LL(1) היא לחשוף תבניות שמייצגות אוטומט סופי.

: Backpatch für  $\rho^{\text{rec}}/\rho^{\text{ref}}$

`makelist(addr)` .addr גורף צדקה צדקה אוניברסיטאות-[addr] \*

\* $\cdot p$  goto  $\rightarrow$  prisjället till fe att ta addrs till  $\text{3n-backpatch}(p, \text{addr})$

מִקְרָא:



M.instr = nextInstr();

\* רפואה כ-*N* נבייה ער ectodermectocytoid גיר, ומי שוקה ב.

## દ્વારા પણ કે Backpatching

| production                             | semantic action   |
|--|---|
| $B \rightarrow B_1 \text{ or } M B_2$  | <b>backpatch</b> ( $B_1.falseList$ , $M.instr$ );<br>$B.trueList = B_1.trueList ++ B_2.trueList;$<br>$B.falseList = B_2.falseList;$                               |
| $B \rightarrow B_1 \text{ and } M B_2$ | <b>backpatch</b> ( $B_1.trueList$ , $M.instr$ );<br>$B.trueList = B_2.trueList;$<br>$B.falseList = B_1.falseList ++ B_2.falseList;$                               |
| $B \rightarrow \text{not } B_1$        | $B.trueList = B_1.falseList;$<br>$B.falseList = B_1.trueList;$  |
| $B \rightarrow ( B_1 )$                | $B.trueList = B_1.trueList;$<br>$B.falseList = B_1.falseList;$  |
| $B \rightarrow id_1 \triangle id_2$    | $B.trueList = [\text{nextInstr}];$<br>$B.falseList = [\text{nextInstr}+1];$<br><b>emit</b> ('if' $id_1.var \triangle id_2.var$ 'goto _'); <b>emit</b> ('goto _'); |
| $B \rightarrow \text{true}$            | $B.trueList = [\text{nextInstr}]; B.falseList = [];$<br><b>emit</b> ('goto _');   |
| $B \rightarrow \text{false}$           | $B.falseList = [\text{nextInstr}]; B.trueList = [];$<br><b>emit</b> ('goto _');   |
| $M \rightarrow \epsilon$               | $M.instr = \text{nextInstr};$   |

• 229 per emit in backpacks 830/ 973 1/ not -f 6C6C of 92% \*

## :(Statements) מירג' Backpatching

| production   | semantic action   |
|--|---|
| $S \rightarrow \text{if } (B) M S_1$                                     | <code>backpatch(B.trueList, M.instr);<br/>S.nextList = B.falseList ++ S_1.nextList;</code>  |
| $S \rightarrow \text{if } (B) M_1 S_1 N$<br>$\quad \text{else } M_2 S_2$ | <code>backpatch(B.trueList, M_1.instr);<br/>backpatch(B.falseList, M_2.instr);<br/>temp = S_1.nextList ++ N.nextList;<br/>S.nextList = temp ++ S_2.nextList;</code> |
| $S \rightarrow \text{while } M_1 (B)$<br>$\quad M_2 S_1$                 | <code>backpatch(S_1.nextList, M_1.instr);<br/>backpatch(B.trueList, M_2.instr);<br/>S.nextList = B.falseList;<br/>emit('goto' M_1.instr);</code>                    |
| $S \rightarrow \{ L \}$  | <code>S.nextList = L.nextList;</code>   |
| $S \rightarrow A$  | <code>S.nextList = null;</code>   |
| $M \rightarrow \epsilon$   | <code>M.instr = nextinstr;</code>   |
| $N \rightarrow \epsilon$   | <code>N.nextList = [nextinstr]; emit('goto _');</code>  |
| $L \rightarrow L_1 M S$  | <code>backpatch(L_1.nextList, M.instr);<br/>L.nextList = S.nextList;</code>   |
| $L \rightarrow S$  | <code>L.nextList = S.nextList</code>  |

• if-then-else -> סדר גורף מילוי goto (בבב נ' נ' נ')

• סדר גורף

• (Intermediate Representation) מילוי סדר גורף

• סדר גורף מילוי סדר גורף

• סדר גורף מילוי סדר גורף

• (statements) מילוי סדר גורף goto (בבב נ' נ' נ')

• backpatching

(statements) מילוי סדר גורף goto (בבב נ' נ' נ')

• סדר גורף מילוי סדר גורף

• סדר גורף מילוי סדר גורף goto (בבב נ' נ' נ')

• סדר גורף מילוי סדר גורף goto (בבב נ' נ' נ')

• סדר גורף מילוי סדר גורף goto (בבב נ' נ' נ')

# קוד LLVM IR

10:35 AM Sunday, May 10, 2020

## LLVM-IR

LLVM - Low-Level Virtual Machine \*

.framework - even better -

(Clean, flat) Flat IR, 3AC if even more \*

%num = add i32 %inp, 48 : function signature float float float  
target opode type operands

.clean & clean if we backend -> can see all the opcodes \*

.0-16 % - numbers in LLVM -> different rule \*

## i32 - LLVM IR

: Numeric \*

i1, i32, i16, i64 - Integers -

half, float, double - Floating Point -

i8\* Pointer \*

label (global variable) Label \*

: Aggregate \*

[4 x [10 x float]], [4 x i32] (struct pointer) Array -

{float, [4 x i32]} Struct -

## : LLVM-IR

%a = alloca i32  
Store i32 5, i32\* %a  
%rd = load i32, i32\* %a  
  
%a = alloca [4 x i32]  
%el = getElementPtr [4 x i32],  
[4 x i32]\* %a, i32 0, i32 1  
Store i32 5, i32\* %el  
%rd = load i32, i32\* %el

לינק ל-alloc  
לינק ל-store  
לינק ל-load  
לינק ל-getElementptr  
לינק ל-aggregate  
לינק ל-pointer  
לינק ל-index  
לינק ל-base

.הנה מוגדרים גורם struct - getElementptr \*

struct - מוגדר גורם struct - מוגדר גורם struct -

struct - מוגדר גורם struct - מוגדר גורם struct -

## Hello World - LLVM-IR

```

global
Symbol %str = internal constant [4x8] c "Hello, world!\00"
external
Symbol declare i32 @printf(i8*, ...)

Start
Function define i32 @main() {
    Start
block entry: → p23.g17 label %main
    type annotations
    %tmp1 = getelementptr [4x8], [4x8]* %str, i32 0, i32 0
    end
block %tmp1 = call i32 (i8*, ...) @printf(i8* %tmp1)
    ret i32 42
    end → }
function

```

## Gotchas - LLVM

: מושג זה מופיע ב LLVM מפץ ורמז על קיומו של סימולטן LLVM ב שפה \*

**ret br** switch indirectbr invoke callbr resume

catchswitch catchret cleanupret unreachable

(SSA) Single Static Assignments - מושג אחד הוא מושג LLVM ב שפה \*

: Single Static Assignment - מושג אחד הוא מושג LLVM ב שפה \*

```

p = a + b
q = p - c
p = q * d
p = e - p
q = p + q

```

```

p1 = a + b
q1 = p1 - c
p2 = q1 * d
p3 = e - p2
q2 = p3 + q1

```

.ret מושג אחד הוא מושג אחד והוא מושג LLVM ב שפה \*

.exit מושג אחד הוא מושג LLVM ב שפה \*

.phi מושג אחד הוא מושג LLVM ב שפה \*

.entry מושג אחד הוא מושג LLVM ב שפה \*

```

if (f)
    x = 42;
else
    x = 73;
y = x * a;

```

```

if (f)
    x1 = 42;
else
    x2 = 73;
x3 = f(x1,x2);
y = x3 * a;

```

.join מושג אחד הוא מושג LLVM ב שפה \*

.open מושג אחד הוא מושג LLVM ב שפה \*

.phi מושג אחד הוא מושג LLVM ב שפה \*

.entry if מושג אחד הוא מושג LLVM ב שפה \*

```

cond:
    tb = icmp ult i32 %i, %j
    br il tb, label %then,
    label %else
then:
    amax = or i32 0, %j
    br label %exit
else:
    amax = or i32 0, %i
    br label %exit
exit:
    ret i32 amax

```

```

cond:
    tb = icmp ult i32 %i, %j
    br il tb, label %then,
    label %else
then:
    amax = or i32 0, %j
    br label %exit
else:
    amax = or i32 0, %i
    br label %exit
exit:
    amax = phi i32 [
        %then,
        %else
    ]
    ret i32 amax

```

...dead code elimination, constant propagation: לפני, אחרי מושג אחד הוא מושג LLVM SSA \*

Compiler Explorer - godbolt.org  
- emv: -Xrun -g0