Program Analysis

Graduate Course by Mooly Sagiv

Spring 2007

שיעור 1

מהו ניתוח סטטי

הגדרה. ניתוח סטטי הוא תהליך שפועל על תוכנית מחשב ומסיק תכונות קבועות של התוכנית, אשר תהיינה תקפות בכל הרצה שלה ובכל קלט שהוא.

דוגמה. בתוכנית C שלהלן

```
int p(int x) {
        return x*x;
}
int main() {
        int z;
        if (getc())
                z = p(6) + 8;
        else
                z = p(-7) – 5;
        printf("%d\n", z);
}
```

מתקיימת התכונה, שבשורה האחרונה (*printf*) הערך של z הוא 44, בלי תלות בקלט. זאת על-אף שמסלול הריצה של התוכנית תלוי בקלט, עקב מבנה הבקרה *if*; משום שבכל ענף הערך המחושב הוא זהה:

$6^2+8 = (–7)^2–5 = 44$

דוגמה נוספת. תוכנית קצת יותר מורכבת:

```
int x;
void p(int a) {
        int c;
        read(c);
        if (c > 0) {
                a = a – 2;
                p(a);
                a = a + 2;
        }
        x = -2 * a + 5;
        print(x)
}
int main() {
        p(7);
        print(x);
}
```

במקרה הזה המשתנה הכללי x מקבל ערכים שונים במהלך התוכנית. זאת עקב הקריאה הרקורסיבית של הפונקציה p עבור ערכים משתנים של הפרמטר a. ואולם, לפני כל יציאה מהפונקציה מתקיימת השמורה:

$x = –2a_0 + 5$

כאשר a_0 הוא הערך שניתן ל a באותה קריאה.

לכן, בשורה האחרונה של התוכנית מתקיימת תמיד התכונה $x = 2*(–7)+5 = –9$.

הערכה אישטיבית – Iterative Approximation

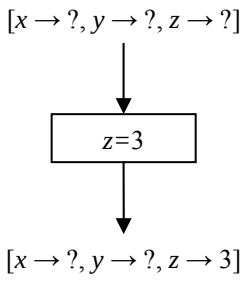
השאלה האם משתנה מסוים קבוע לערך יחיד בנקודה מסוימת בתוכנית נקראת בעיית *constant propagation*. בשיטה זו עוקבים אחרי מצבים אפשריים של התוכנית בהתאם למבני הבקרה הקיימים בה.

הגדרה. מצב בתוכנית הוא מיפוי $f: V \to D \cup \{?\}$ כאשר V היא קבוצת המשתנים בתוכנית ו D מרחב הערכים האפשריים עבורם. הסימן '?' משמש כערך מיוחד לציין שערך המשתנה אינו ידוע בנקודה זו בתוכנית.

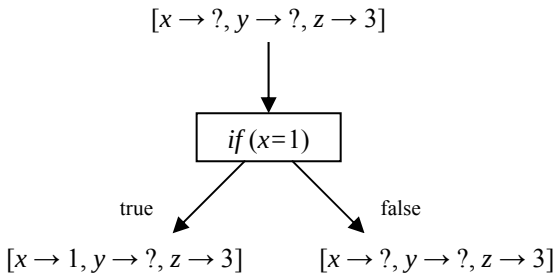
המצב ההתחלתי הוא הפונקציה הקבועה $f(x)=?$.

עוברים על הפקודות בתוכנית אחת אחרי השנייה, בדומה למפרש (*interpreter*), ומסיקים מהמצב הקודם את המצב הבא, במידת האפשר.

למשל, כאשר בתוכנית מופיע משפט של השמה:

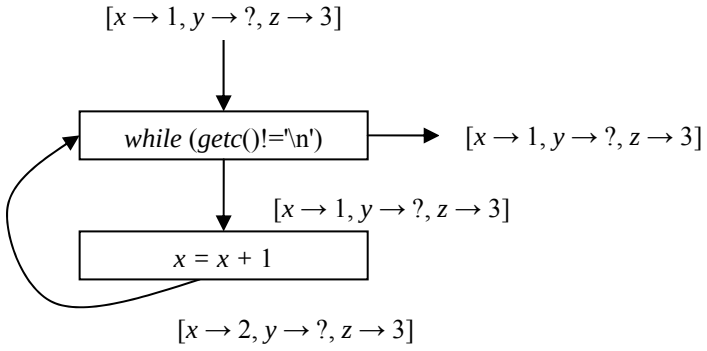


אם יש התפצלות בתוכנית, בודקים את כל הענפים האפשריים:

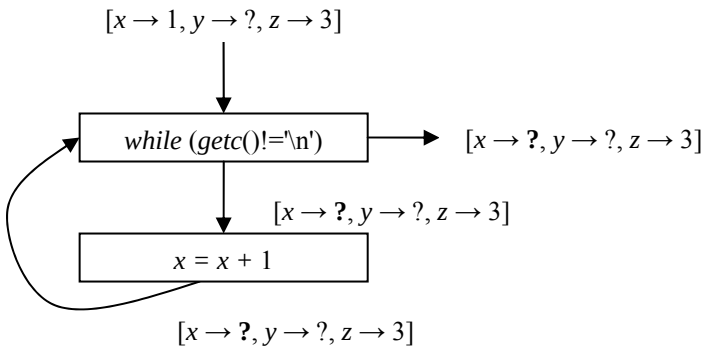


כאשר מטפלים בלולאות, יש לבצע את התהליך על גוף הלולאה, ולאחר-מכן לחזור ולתקן את ההערכות במידת הצורך:

מעבר ראשון

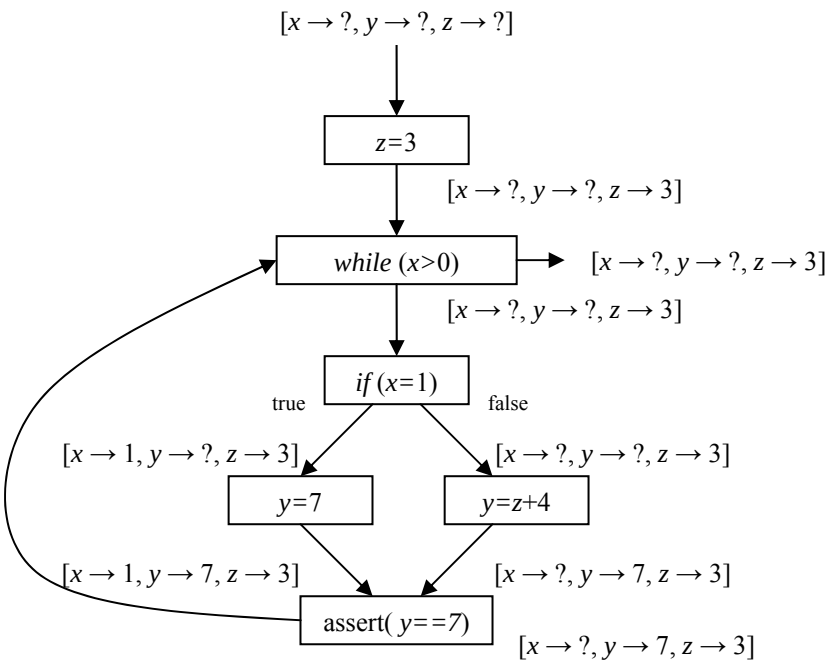


מעבר שני



במקרה זה התקבלו ערכים שונים במעבר הראשון לעומת המעבר השני, ולכן נקבע ערכו של x ל ?.

דוגמה.



מציאת דליפות זיכרון

אלגוריתם שיוצג בהמשך מאפשר לזהות דליפות זיכרון פוטנציאליות בתוכניות.

דוגמה. בתוכנית הבאה המבצעת היפוך של סדר האיברים ברשימה מקושרת, קיימת אפשרות לדליפת זיכרון באיבר המוצבע על-ידי *head*:

```
List reverse(Element *head)
{
	List rev, n;
	rev = NULL;
	while (head != NULL) {
		n = head->next;
		head->next = rev;
		head = n;
		rev = head;
	}
}
```

השגיאה נובעת מכך שהמשפטים `head=n` ו `rev=head` נכתבו בסדר הפוך. אם משנים את הסדר מתקבלת תוכנית תקינה ללא דליפות זיכרון, והאלגוריתם יכול לוודא זאת.

דוגמה נוספת. בתוכנית הבאה ישנה סכנה לחריגה מגבולות חוצץ בזיכרון:

```
void foo(char *s) {
	while (*s != ' ')
		s++;
	*s = 0;
}
```

כדי להימנע מכך, יש להניח הנחות על הקלט לפונקציה. למשל, אם מניחים שהפרמטר *s* הוא מחרוזת מסוג *"null terminated string"* (שהאיבר האחרון בה הוא '\0') ניתן להגן מפני חריגה כך:

```
void foo(char *s) {		@require string(s)
	while (*s != ' ' && *s != 0)
		s++;
	*s = 0;
}
```

הסימול *@require* הוא רמז לאלגוריתם לבצע שני דברים:

- להתייחס ל *s* כאל *null terminated string*,
- לוודא כי בכל מקום שבו קוראים לפונקציה *foo* בתוכנית, מתקיים התנאי הזה.

משתנים חיים

הגדרה. משתנה נקרא "חי" בנקודה מסוימת בתוכנית, אם קיים מסלול חישוב העובר בנקודה זו אשר לפניה מתבצעת השמה למשתנה ולאחריה נדרש הערך של המשתנה.

דוגמה.

		live variables
		c
L0:	a = 0	
		a, c
L1:	b = a + 1	
		b, c
	c = c + b	
		b, c
	a = b * 2	
		a, c
	if (c < N) goto L1	
		c
	return c	

במקרה זה ניתן לראות, כי קיימות נקודות בתוכנית שבהן המשתנים a ו c חיים בו-זמנית, קיימות נקודות שבהן המשתנים b ו c חיים בו-זמנית, אבל לא קיימות נקודות שבהן המשתנים a ו b חיים בו-זמנית. זה מידע שימושי למהדר (*compiler*) המתבקש להקצות אוגרים לאחסון ערכי משתנים בזמן ביצוע של פונקציה – במקרה זה ניתן להשתמש באותו אוגר לאחסון הערכים של a ושל b.

הקושי בניתוח סטטי

- **אי-כריעות.** ישנן בעיות ניתוח סטטי אשר ניתן להראות שהן שקולות לבעיית העצירה, ולכן לא ניתן לתת פתרון מדויק – יש לוותר על *soundness* או על *completeness*.
- **יעילות הניתוח.** במקרים רבים הניתוח דורש זמן מעריכי. זה הופך את התהליך לבלתי-ישים מכיוון שקיימות תוכניות גדולות מאד, בנות מיליונים של שורות קוד. תהליך עשוי להניב תוצאות טובות על תוכניות קטנות, אבל הוא לא יהיה שימושי אם הוא אינו סקלבילי.

האתגרים הללו מתעצמים עם הזמן, כיוון שתוכניות המחשב הולכות ונהיות גדולות יותר, והמחשבים – משוכללים ומורכבים יותר. למשל, בעת ש *Kernighan & Ritchie* תכננו את שפת *C*, הם התכוונו שלא יהיה צורך בביצוע אופטימיזציות מכיוון שהשפה נותנת למתכנת שליטה מלאה. למעשה, מהדרים מודרניים מבלים את רוב זמנם בביצוע ניתוח סטטי, תהליכים שאף מתכנת לא מסוגל לבצע בזמן סביר עקב המורכבות של התוכניות וגודלן.

הבסיס לניתוח סטטי

ניתן להציג מנתח סטטי כסוג מיוחד של אינטרפרטר, כאשר במקום לפעול על ערכים אמיתיים של המשתנים ושל הקלט בתוכנית, הוא פועל על "תחום מופשט" של ערכים. התחום המופשט מייצג קבוצות-על של מצבים אפשריים בתוכנית, ובחירת ההפשטה תלויה בתכונה שאותה רוצים לנתח.

דוגמה. מחשבון הוא אינטרפרטר הפועל על ערכים מספריים (נאמר, טבעיים). ניתן לחלק את המספרים הטבעיים לתשע מחלקות שקילות –

$$0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8$$

כשהחלוקה תתבצע באופן הבא: אם $n<9$, אז n שייך למחלקה n. אחרת, n שייך למחלקה שאליה שייך סכום ספרותיו.

באופן זה ניתן לכתוב "מחשבון מופשט", כאשר במקום לפעול על המספרים, הוא מופעל על מחלקות השקילות שאליהן הם שייכים.

123	*	457	+	76543	$=^{?}$	132654		עבור הביטוי –
6	*	7	+	7	$=^{?}$	3		החישוב המופשט יהיה

המשמעות של המחשבון המופשט הזה, היא שהוא יכול לגלות טעויות בתרגילים חשבוניים כמו זה שלעיל. במקרה הנתון, המחשבון גילה כי פתרון התרגיל שגוי (צד ימין שווה למחלקה 4, צד שמאל שווה למחלקה 3). תכונה זו נובעת מתכונה אלגברית, שהפעולה *mod 9* שומרת על הפעולות +, * וסכום ספרות:

$(10a + b) \bmod 9 = (a + b) \bmod 9$

$(a + b) \bmod 9 = (a \bmod 9) + (b \bmod 9) \bmod 9$

$(a * b) \bmod 9 = (a \bmod 9) * (b \bmod 9) \bmod 9$

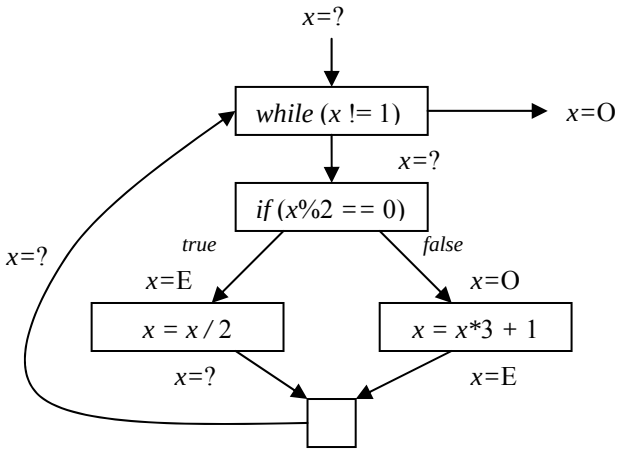
המחשבון המופשט מקיים תכונה של נאותות; אם הוא מדווח על שגיאה בתרגיל, אזי שגיאה כזו אכן קיימת. ואולם, ייתכן מצב שבו קיימת שגיאה והמחשבון לא יגלה אותה. תהליכי ניתוח סטטי נדרשים לקיים תכונת נאותות (בדרך-כלל מעדיפים דווקא שהתכונה תתקיים בכיוון ההפוך, כלומר שהתהליך יגלה כל שגיאה, גם אם יש תוכניות תקינות שהתהליך יזהה כשגויות).

דוגמה נוספת. רוצים לנתח את התכונה שהערך של משתנה מסוים זוגי או אי-זוגי. נתונה תוכנית:

```
while (x != 1) {
       if (x % 2 == 0) {
              x = x / 2;
       }
       else {
              x = x * 3 + 1;
              assert(x % 2 == 0);
       }
}
```

מגדירים תחום מופשט שהוא {O, E, ?}. הסמל *E* מייצג מספר זוגי. *O* מייצג מספר אי-זוגי. *?* מציין כי הזוגיות אינה ידועה מראש.

כעת ניתן להפעיל כללים אלגברים: אם *x%2==0* אזי המספר הוא זוגי; אחרת, הוא אי-זוגי. לכן מסיקים את תנאי הכניסה המתאימים לענפים שבתוכנית שבדוגמה. בנוסף, מכפלה של שני מספרים אי-זוגיים היא אי-זוגית, וסכום של שני מספרים אי-זוגיים הוא זוגי.



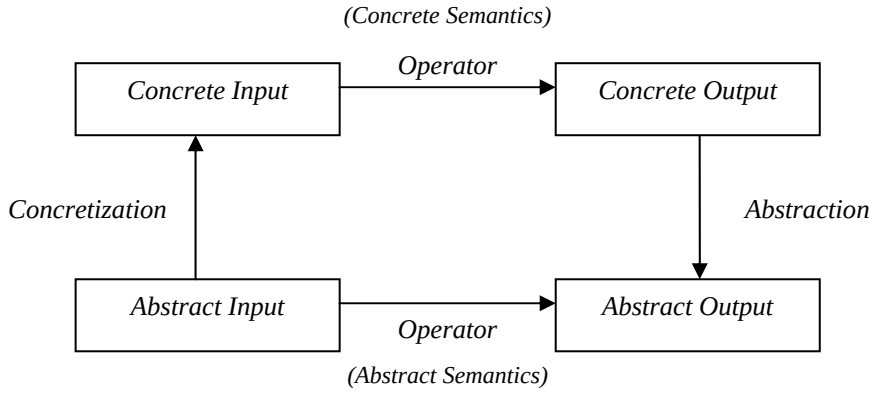
מהניתוח הסטטי הזה ניתן להסיק כמה מסקנות שימושיות:

- כאשר מגיעים למשפט *assert* הערך של *x* תמיד זוגי, ולכן ה *assertion* יתקיים.
- אם *x* אי-זוגי באיטרציה הנוכחית, הוא יהיה זוגי באיטרציה הבאה, וניתן לבצע ייעול מסוג *loop-unrolling* שיקטין את מספר האיטרציות ומספר ה*if*ים בריצת התוכנית.
- אם התוכנית מסתיימת, אזי *x* אי-זוגי.

הניתוח לא יכול לזהות האם התוכנית עוצרת. זאת בעיה הידועה כ"סדרת קולאז'" ונחשבת לבעיה פתוחה.

מודל ההפשטה הזה מייצג משפחה של שיטות לניתוח סטטי: ממפים תת-קבוצות של קבוצת הערכים האפשריים עבור x לאיברים בתחום המופשט. במקרה זה קבוצות שבהן כל הערכים זוגיים מופו ל E; קבוצות שבהן כל הערכים אי-זוגים – ל O; קבוצות מעורבות – ל $?$. באופן הזה מאבדים מידע על התוכנית, כיוון שקבוצות שונות מאד (למשל $\{1\}$ ו $\{3, 5, 7, 9, \ldots\}$) ממופות לאותו איבר; אבל המודל המתקבל פשוט יותר.

לכל משפט בתוכנית, ניתן לחשב את המשמעות שלו בעולם המופשט על-ידי תרגומו לקבוצת הערכים הקונקרטיים אותם הוא מייצג (קונקרטיזציה), הפעלת האופרטור על כל איבר וביצוע ההפשטה מחדש על הקבוצה המתקבלת. (כמובן שאין הכוונה לבצע חישוב ממצה בעזרת מחשב מכיוון שמדובר בדרך-כלל בקבוצות אין-סופיות)



דוגמה. מתוך לוחות החיבור והכפל אפשר לבנות את הפעולות המתאימות של חיבור וכפל מופשטים:

+'	?	O	E
?	?	?	?
O	?	E	O
E	?	O	E

*'	?	O	E
?	?	?	E
O	?	O	E
E	E	E	E

במקרים מסוימים, המעבר מהיחסים הקונקרטיים ליחסים המופשטים יכולה להיות בעיה קשה מאד.

דוגמה נוספת. *rule of signs* – רוצים לנתח את תכונת הסימן (חיובי/שלילי) של ערכים.

התחום המופשט הוא $\{+, -, ?\}$ ופעולת הקונקרטיזציה מוגדרת על-ידי הפונקציה:

$$\gamma(a) = \begin{cases} \{0,1,2,\ldots\} & a = + \\ \{-1,-2,\ldots\} & a = - \\ \mathbb{Z} & a = ? \end{cases}$$

הקונקרטיזציה מובילה לרשימה אינסופית של ערכים שעליה צריך להפעיל את האופרטור. למשל, אם יש שני משתנים x, y הנמצאים במצב המופשט $\langle -,- \rangle$ (שניהם שליליים), והאופרטור הוא $x := x*y$

לפני	אחרי
$\langle -1,-1 \rangle$	$\langle 1,-1 \rangle$
$\langle -1,-2 \rangle$	$\langle 2,-2 \rangle$
$\langle -1,-3 \rangle$	$\langle 3,-3 \rangle$
$\langle -7,-51 \rangle$	$\langle 357,-51 \rangle$
$\langle -88,-2 \rangle$	$\langle 176,-2 \rangle$
…	…

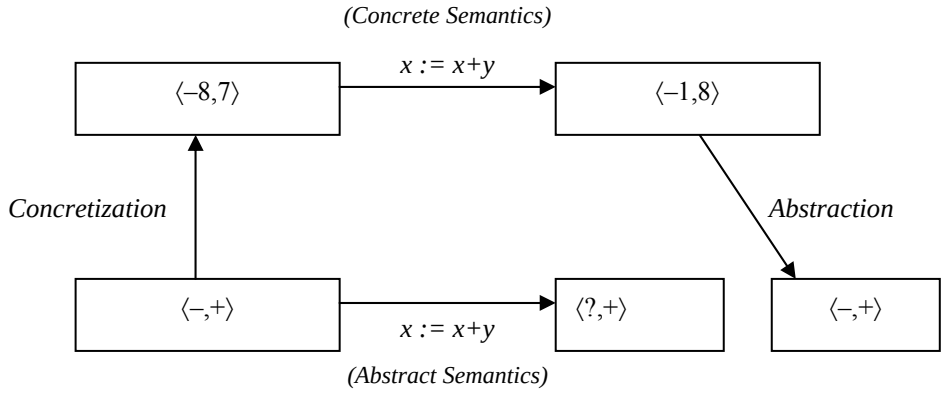
במקרה כזה קל לראות שכל האפשרויות מובילות לכך ש x הוא ערך חיובי. מפעילים את פונקצית ההפשטה:

$$\alpha(C) = \begin{cases} + & \forall c \in C, c \geq 0 \\ - & \forall c \in C, c < 0 \\ ? & \text{otherwise} \end{cases}$$

ומקבלים שהמצב החדש הוא $\langle -,+ \rangle$.

גישה כזו שבה משייכים איבר לקבוצה מופשטת רק אם הערך הקונקרטי שייך לקבוצה בכל מצב ומצב, נקראת שמרנית (*conservative*).

פונקציות הפישוט והקונקרטיזציה לא יכולות להרכיב תמיד מבנה הומומורפי; למשל, אם המצב הוא $\langle -,+ \rangle$ והאופרטור הוא $x := x+y$, אז עבור כל ערך קונקרטי שנבחר מגיעים למצב $\langle -,+ \rangle$ או למצב $\langle +,+ \rangle$; ואולם הפעלת האופרטור המופשט מניבה את המצב $\langle +,? \rangle$.



בכל מקרה, מובטח שבגישה השמרנית לא מחמיצים מצבים, כלומר הקבוצה שמתקבלת במסלול המופשט מכילה את הקבוצה שמתקבלת במסלול הקונקרטי ($\langle -,+ \rangle \subseteq \langle ?,+ \rangle$).

אי-כריעות

הבעיה האם תוכנית מגיעה לנקודה מסוימת במהלך הביצוע שלה היא בלתי-כריעה.

יתרה מכך, מסתבר כי יש בעיות ניתוח סטטי שהן בלתי-כריעות גם אם מניחים שכל הקוד בתוכנית הוא נגיש.

דוגמה. נתון פולינום טבעי p; אם בונים את התוכנית הבאה:

```
while (getc()) {
       if (getc()) x_1 = x_1 + 1;
       if (getc()) x_2 = x_2 + 1;

       if (getc()) x_n = x_n + 1;
}
y = trunc(1 / (1 + p2(x_1, x_2, ..., x_n)))
```

אזי בעיית הניתוח "האם תמיד $y=0$ בסוף התוכנית?" שקולה לבעיה "האם לפולינום p יש שורש במספרים טבעיים?", בעיה הידועה כ"בעיה העשירית של הילברט" והיא אינה כריעה. הקושי של הבעיה אינו נובע במקרה זה מאותם נימוקים התקפים במקרה של בעיית העצירה, אלא מתכונות בסיסיות של מספרים טבעיים.