מבוא

המושג **רקורסיה** נחשב לאחד המושגים המרכזיים במדעי-המחשב בכלל, ובתכנית הלימודים לבית הספר התיכון בפרט. מורים למדעי המחשב מתלבטים לא אחת בדבר דרכי ההוראה המתאימות לתלמידיהם הלומדים רקורסיה, בדבר הפעילויות הלימודיות אותן כדאי לבצע בכיתה ובדבר המועד המתאים לטיפול ברקורסיה במסגרת תכנית הלימודים במדעי המחשב. זאת ועוד, רקורסיה נחשבת ומוצגת בספרים כנושא מורכב וקשה להוראה.

הספרות החינוכית והמחקרית הענפה העוסקת בלמידה ובהוראה של רקורסיה פורשת את מגוון ההתלבטויות והבעיות, מציעה פתרונות ומתארת דרכי הוראה, ומציגה ממצאים של מחקרים העוסקים באוכלוסיות מגוונות של לומדי מדעי-המחשב. מטרתו של סקר ספרות זה היא להביא בפני המורים למדעי המחשב בישראל מעט מן הרעיונות וממצאי המחקרים שנערכו בנושא, תוך חשיפת אוכלוסית המורים לבעיות שאותרו בקשר ללמידת רקורסיה, לגישות להוראת הנושא ולפתרונות שנוסו בהתמודדות עם הבעיות.

חלק שני זה של סקר הספרות מציג תקצירים מתורגמים של חמישה מאמרים שפורסמו בעיתונות המדעית המקצועית. מאמרים אלה נבחרו בקפידה מתוך מבחר מקורות, כך שיביאו בפני המורה למדעי המחשב הן היבטים שונים הקשורים בלמידה והוראה של רקורסיה והן פתרונות פרקטיים אותם ניתן ליישם בבתי ספר תיכוניים בישראל ובמסגרת תכנית הלימודים החדשה במדעי המחשב. בפרק הפותח את החלק השני, מובא רקע כולל הנוגע למושג רקורסיה, ללמידתו ולהוראתו, תוך התייחסות אינטגרטיבית למאמרים שתקציריהם מופיעים בפרקים הבאים כמו גם לקשת רחבה בהרבה של פרסומים ומקורות חינוכיים העוסקים ברקורסיה.

רקע כללי – רקורסיה, למידתה והוראתה

רקורסיה: מבט רב תחומי

רקורסיה מופיעה בתכניות הלימודים בדרך כלל בקורסים העוסקים ביסודות התכנות, לאחר לימוד כלים תכנותיים בסיסיים כמו משתנה, פרוצדורה ולולאה ולאחר הכרות עם מבני הנתונים הבסיסיים. המרכזיות של המושג רקורסיה במדעי-המחשב נובעת ממספר גורמים, וביניהם האפשרות לתת פתרון לבעיות אלגוריתמיות מסוימות, היכולת לבטא ולייצג מבני נתונים מורכבים, והכוח לתאר בפשטות תהליכים חישוביים מסובכים. בנוסף לגורמים הדיסציפלינריים, תורמים גם גורמים אינטרדיסציפלינריים להיות הרקורסיה מושג מפתח. מוסיף לכך המסתורין המיוחס למושג ואשר בא לידי ביטוי באמירות כמו "אלגוריתם מיון מיזוג הוא דוגמה טובה לעוצמה הכמעט-מסתורית ולאלגנטיות של רקורסיה" (היליס, 2000, עמ' 76) וכן "פלאיה של הרקורסיה במתמטיקה הם רבים מספור" (Hofstadter, 1979, p. 138).

בתוך הדיסציפלינה של מדעי-המחשב, המושג רקורסיה מבטא בדרך כלל את הרעיון של התייחסות עצמית ואת אפשרויות יישום הרעיון לאלגוריתמים ולתכניות מחשב (Harvey & Wright, 1994). ההגדרות המופיעות בספרי לימוד של מדעי-המחשב מתייחסות לרקורסיה כאל כלי תכנותי, או טכניקה תכנותית, כמו בהגדרה הבאה - "רקורסיה היא טכניקה שבה פונקציה או פרוצדורה קוראת לעצמה, ואלגוריתם רקורסיבי הוא כזה שבו תהליך מתייחס לעצמו" (Garland, 1987, p.321). אלגוריתם רקורסיבי מוגדר כ"אלגוריתם שאחד (או יותר) ממשימותיו הוא הפעלה חוזרת של אותו אלגוריתם על קלט פשוט יותר" (בן-ארי וחוב', 1998, עמ' 193). הגדרות אלה מבטאות את תיאור הפן **התהליכי** של מושג הרקורסיה. אולם חשיבותו של המושג "רקורסיה" במדעי-המחשב חורגת מההתייחסות לרקורסיה כאל טכניקה לכתיבת אלגוריתמים, והיא נוגעת במידה רבה גם ל**מבנה** העצמים המשמשים כנתונים לאלגוריתמים. ההתייחסות הכללית והמופשטת יותר למושג "רקורסיה" מתמקדת בתיאור המאפיינים של **הגדרה רקורסיבית** שעשויה להיות מיושמת הן על מבנים והן על תהליכים. כותב על כך היליס: "(ה)שיטה להגדרה רקורסיבית של דברים, מתברר, היא רבת עוצמה. רבים מסוגי הנתונים שאנו עושים בהם מניפולציות – וביחוד תכניות המחשב עצמן – הם בעלי מבנה רקורסיבי. ההגדרות הרקורסיביות נוחות מאין כמוהן לתיאור פעולות שיש לבצע בנתונים רקורסיביים" (היליס, 2000, עמ' 52).

במובן הכללי ביותר, הגדרה רקורסיבית היא "הגדרה שבה המונח המוגדר הוא עצמו חלק מההגדרה" (Gersting, 1996, p.56) או הגדרת משהו במונחים של עצמו. להגדרה רקורסיבית אופיינית יש שני חלקים: חלק אחד מתאר מה צריך לקרות במקרה פשוט מסוים, וחלק אחר מתאר איך אפשר לצמצם מקרה מורכב למקרה פשוט יותר אך דומה לו. הגדרה רקורסיבית של אלגוריתם בנויה לפיכך מחלק אחד המתאר מה האלגוריתם צריך לבצע במקרה הפשוט ביותר האפשרי, ומחלק שני המתאר כיצד האלגוריתם מפעיל העתק פשוט יותר של עצמו. באופן דומה, הגדרה רקורסיבית של מבנה-נתונים בנויה מחלק אחד המתאר כיצד נראה מבנה-הנתונים הפשוט ביותר האפשרי, ומחלק שני המתאר כיצד מבנה-הנתונים מכיל גירסה קטנה יותר, ובדרך כלל פשוטה יותר, של עצמו. בתרשים 1 מובאות שלוש דוגמאות להגדרה רקורסיבית של אלגוריתמים, ובתרשים 2 מובאות שלוש דוגמאות להגדרה רקורסיבית של מבני-נתונים.

תרשים 1: **הגדרות רקורסיביות של אלגוריתמים**

א. הגדרה רקורסיבית של אלגוריתם מתמטי לביצוע פעולת האיחוד (ע"פ Gersting, 1996)

מקרה בסיסי פשוט ביותר

1. $A_1 \vee A_2$ defined by the truth table
2. $A_1 \vee \ldots \vee A_n = (A_1 \vee \ldots \vee A_{n-1}) \vee A_n$ for $n > 2$

קריאה רקורסיבית

ב. הגדרה רקורסיבית של אלגוריתם לביצוע פעולת כפל בשפת התכנות Pascal (ע"פ Gersting, 1996)

```
function Product(m, n:integer): integer;
{recursively computes the product of m and n}
begin
   if n = 1 then
      Product := m
   else
      Product := Product (m, n - 1) + m;
end;
```

מקרה בסיסי פשוט ביותר (Product := m)

קריאה רקורסיבית (Product (m, n - 1))

ג. הגדרה רקורסיבית של אלגוריתם לסיכום אברי רשימה בשפת התכנות DrScheme (ע"פ לפידות, לוי ופז, 2001)

```
(define  (sum-all  L)
    (cond
           [ ( empty?  L )   0  ]
           [ else  ( +  ( first  L )  ( sum-all  (rest  L ) )  ] ) )
```

מקרה בסיסי פשוט ביותר

קריאה רקורסיבית

תרשים 2: **הגדרות רקורסיביות של מבני נתונים**

א. הגדרה רקורסיבית של מבנה הנתונים "מזהה" בשפת תכנות (BNF) (ע"פ Gersting, 1996)

קריאה רקורסיבית

<**identifier**> ::= <letter>|<**identifier**><letter>|<**identifier**><digit>
<letter> ::= a|b|c|…|z
<digit> ::= 0|1|…|9

מקרים בסיסיים פשוטים ביותר

ב. הגדרה רקורסיבית של מבנה הנתונים "עץ בעל שרשרת X" (ע"פ ברנדס וחוב', 1997)

(1) עץ בעל שרשרת X במקרה הפשוט הוא עץ בינרי ריק

מקרה בסיסי פשוט ביותר

(2) **עץ בעל שרשרת X** הוא עץ בינרי שתוכן השורש שלו הוא X ולפחות אחד מהתת עצים של הוא **עץ בעל שרשרת X**

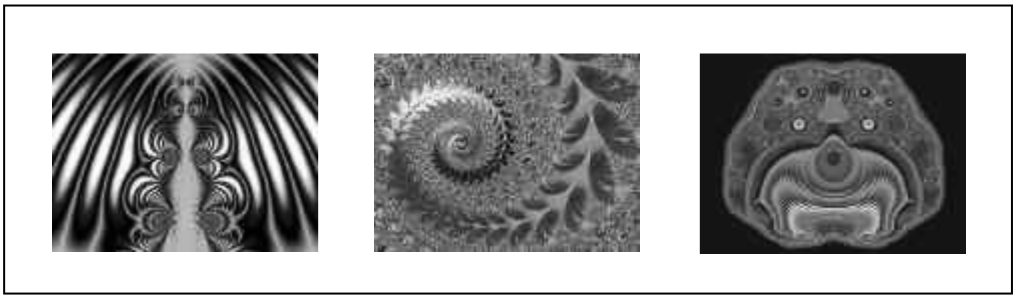
קריאה רקורסיבית

לסיכום, בספרות של מדעי-המחשב מופיעות ההתייחסויות לרקורסיה בעיקר באמצעות התייחסות לאלגוריתמים רקורסיביים ובאמצעות התייחסות להגדרות רקורסיביות. ניתן לתת הגדרות רקורסיביות עבור סדרות של עצמים, קבוצות של עצמים, ופעולות על עצמים, כאשר ידוע המידע הבסיסי הפשוט ביותר וכאשר מידע חדש תלוי במידע קיים. כך, אלגוריתם רקורסיבי מספק דרך טבעית לפתור בעיות מסוימות באמצעות הפעלת אותה המשימה על גרסאות קטנות יותר של אותה בעיה. דרך זו היא טבעית במיוחד כאשר האלגוריתם משתמש במבני נתונים שהגדרתם רקורסיבית. ההגדרות הרקורסיביות מופיעות בספרות במגוון שפות-תיאור, כולל שפה מתמטית, שפות תכנות שונות, ושימוש בפסאודו-קוד לסוגיו.

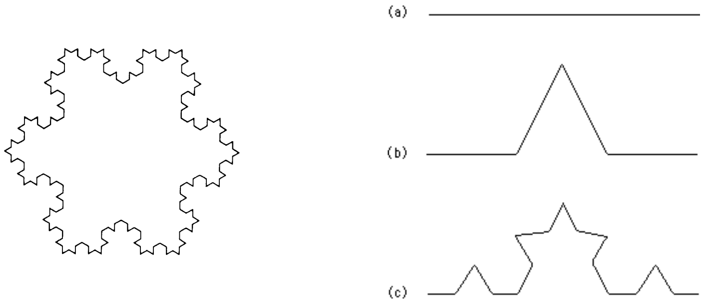
כל ההתייחסויות הללו עשויות לבטא את הגבלת מרחב הלימוד של מושג הרקורסיה להקשרים תכנותיים, ובעיקר לבעיות אלגוריתמיות שניתנות לחישוב. אולם מושג הרקורסיה הוא אינטרדיסציפלינרי במהותו, והיקפו חורג מהתחום התכנותי-אלגוריתמי. ההתייחסות המקיפה ביותר מופיעה בספר Godel,Escher,Bach - An Eternal Golden Braid (Hofstadter, 1979), ושם הספר מעיד על התחומים השונים בהם הוא עוסק: לוגיקה, אמנות ומוסיקה. הופשטטר טוען כי "לפעמים נדמה כי רקורסיה מתחככת בפרדוקס. למשל, יש הגדרות רקורסיביות. הגדרה כזו עשויה לעשות רושם שמשהו מוגדר במונחים של עצמו. זה יהיה מעגלי ויוביל לרגרסיה אינסופית, אם לא לפרדוקס ממש. למעשה, הגדרה רקורסיבית (מנוסחת היטב) אף פעם לא מובילה לרגרסיה אינסופית או לפרדוקס. וזאת משום שהגדרה רקורסיבית אף פעם לא מגדירה משהו במונחים שלו עצמו, אלא תמיד במונחים של גירסאות פשוטות יותר של עצמו" (שם, עמ' 67). ההסבר הזה מזכיר את ההתייחסויות התכנותיות שהובאו לעיל, אולם המחבר משתמש בדוגמאות ממגוון תחומים לא תכנותיים כדי להמחיש את טענותיו. למשל, אחת הדרכים הנפוצות בהן רקורסיה מופיעה בחיי יומיום היא כאשר דוחים את השלמת ביצועה של משימה לטובת ביצוע של משימה פשוטה יותר מאותו סוג. הופשטטר מביא דוגמה של מנהל שמדבר בטלפון ומבקש מהדובר איתו להמתין כדי לדבר עם מישהו אחר. דוגמאות נוספות בספרו לקוחות מתוך ציוריו של אשר (Escher), מתווי יצירותיו של באך, מדיונים המתנהלים במסורת המיתולוגיה היוונית, מהמבנה (הפיזיקלי) של החומר, מדקדוק של שפות, מהפיסיקה של מצב מוצק, ועוד. הוא מסכם כי "רקורסיה היא תחום שבו העניין של דמיון בתוך שוני משחק תפקיד מרכזי. רקורסיה מבוססת על כך שאותו אירוע מתרחש ברמות שונות בבת אחת. אבל האירועים ברמות השונות אינם בדיוק אותו דבר - הם חולקים תכונה אינואריאנטית כלשהי, ומלבדה הם יכולים להיות שונים זה מזה בדרכים רבות" (שם, עמ' 9-148).

סוג מעניין במיוחד של תופעות רקורסיביות מכונה פרקטלים (תרשים 3). הפרקטלים תופסים מקום של כבוד בתיאוריות העוסקות בכאוס, והם זוכים לספרות ענפה ורב תחומית שלא תסקר כאן מקוצר היריעה. פרקטל משמש דוגמה לרקורסיה רב-ממדית, דהיינו תופעה שמשתמשת בכמה וכמה העתקים פשוטים יותר של עצמה בו זמנית. אחד הפרקטלים המפורסמים מכונה פתית השלג של קוך, ששלבי יצירתו תוך שימוש בהעתקים פשוטים יותר מוצגים בתרשים 4.

תרשים 3: **דגמים שונים של פרקטלים**[1]



תרשים 4: **שלבים ביצירת פתית השלג של קוך**



הקשרים של רקורסיה לתחומי-דעת מדעיים ואמנותיים אחרים, והאופנים המגוונים שבהם מתבטא הרעיון של התייחסות עצמית בתחומי-דעת אלה, תורמים לחשיבותו החינוכית של המושג; לאפשרות קישורו לחיי יומיום ולהעדפות המקצועיות של הלומדים; להרחבת האופקים; ולהעשרת אופני ההתבוננות, הניתוח וההבנה של רקורסיה.

למידת רקורסיה

חשיבותו של המושג רקורסיה ללומדי מדעי-המחשב אינה מוטלת בספק, שכן הבנה טובה של רקורסיה היא נכס מהותי עבור לומדי מדעי-המחשב. כפי שנאמר בסעיף הקודם, הרקורסיה נחשבת כטכניקה רבת עוצמה לפתרון בעיות וכרעיון שעוזר להבין טוב יותר במבני-נתונים מורכבים. אולם, באותה מידה נחשב מושג מפתח זה כאחד המושגים הקשים ללמידה ולהוראה. בשנים האחרונות רבים כתבו על כך שללומדים יש בעיות מיוחדות בלמידה והבנה של רקורסיה וביישום עקרונותיה לפתרון בעיות. הספרות החינוכית והמחקרית מתייחסת גם לקשיים טיפוסיים של לומדים העוסקים בתכנות תוך שימוש ברקורסיה ומפרטת מספר סיבות לקשיים אלה.

[1] מקורות העוסקים בפרקטלים מצויים בשפע באינטרנט. למשל, באתר http://home.att.net/~Novak.S/ .

אחת הסיבות לקשיים בהבנת רקורסיה מתוארת על-ידי לירון (1996). לירון טוען כי הן הכוח והן המורכבות של המושג רקורסיה נובעים שניהם מאותו מקור: יכולתה להפיק תיאורים פשוטים מאד לתופעות מורכבות מאד. ההסבר שהוא מציע לקושי עושה שימוש במונחי 'השלישיה החישובית' – ה**תכנית** שמפעילה את ה**תהליך**, שבסופו של דבר מפיק את ה**תוצר**.[2] במונחים אלה, הקושי המהותי בהבנת רקורסיה נובע מפער המורכבות בין תכניות והליכים רקורסיביים לבין התהליכים הנוצרים על ידם. ה**תכנית** הרקורסיבית היא בדרך כלל קצרה ופשוטה למראה, אולם היא יוצרת **תהליך** מורכב עד מאד, כמו למשל תהליכים רקורסיביים שפותרים את בעיית מגדלי הנוי או מסרטטים את הפרקטלים שתוארו בסעיף הקודם. התכנית הרקורסיבית הפשוטה למראה היא ייצוג סטטי (טקסטואלי) של התהליך, בעוד התהליך הוא הייצוג הדינמי של התכנית. הפער שבין הפשטות של הייצוג הסטטי לבין המורכבות של הייצוג הדינמי במקרה של רקורסיה, הוא אחת הסיבות לקשיים בהם נתקלים הלומדים. מזיהויו של פער זה נובע גם הפתרון: הואיל והמורכבות מסתתרת בעיקר בתהליך, ממליץ לירון (שם) להדחיק את החשיבה עליו. על השאלה "מה ההליך עושה?" יש לענות במונחים של התוצר, לא התהליך. במילים אחרות, עלינו לראות את התכנית כתיאור של התוצר, לא של התהליך.

ג'ורג' מתייחס אף הוא לפער שבין הייצוג הסטטי לבין הייצוג הדינמי (George, 2000a). לומדים רבים מפתחים, לטענתו, באופן טבעי מודל הפעלה טקסטואלי-סטטי שבו הפרוצדורות נתפסות כקטעים סטטיים של קוד התכנית, ואינם מסוגלים לפיכך לראות בעיני רוחם את התהליכים הדינמיים השונים המייצגים באופן לוגי את ביצוע הפרוצדורות. קושי מיוחד מתעורר אצל לומדים בנוגע להבנת התפקיד של השהיית החישוב (suspended computation), השהייה המהווה מרכיב מרכזי של כל אלגוריתם רקורסיבי (Bhuiyan, Greer & McCalla, 1994). קשיים אלה בהבנת רקורסיה הם חלק מבעייה כללית יותר של בניית מודל מנטלי אפקטיבי של הפעלת תת-תכניות. בספרות מופיעים פתרונות שונים לבעייה זו, כולל שימוש במודלים ויזואליים של רקורסיה (ראה המודל שמציע היינס (Haynes, 1995) במאמרו המופיע בהמשך, מודל האנשים הקטנים (לוי 1991; Harvey, 1985), מודל top-down המכונה גם מודל המסגרות (Harvey, 1985) והמודל הלוגי-דינמי אצל ג'ורג' (George, 2000a)).

מרבית המחקרים שהתייחסו לקשיים של תלמידים התבצעו במסגרת תכנותית, דהיינו בהקשר של הבנת אלגוריתמים רקורסיביים הרשומים בניסוח פורמלי בשפת תכנות. באחד המקורות הראשונים שחקר קשיים של תלמידים בהבנת רקורסיה, נטען כי הקושי מתעורר בשל הצגת רקורסיה לאחר שתורגל השימוש במבני לולאה רגילים (Kurland & Pea, 1983). יש הטוענים כי הקשיים המדווחים בהוראת הנושא נובעים מהשילוב של רקורסיה בתכנות פרוצדורלי (אימפרטיבי), ומוסיפים כי לא מדווחים קשיים דומים בלמידת רקורסיה בהקשר של תכנות פונקציונלי (Valazquez-Iturbide, 2000). מקורות אחרים מציינים כי דווקא השילוב בין המושג המופשט "רקורסיה" למושג המופשט "פונקציה" הוא אחד ממקורות הקושי העיקריים (Troy & Early, 1992). ואולם, למעט מחקריהן של חדשי (1996) ושל לוי (2001) שעוסקים בהוראה ולמידה בין-תחומית של רקורסיה, לא מדווח על מחקרים שנעשו בהקשר לא-תכנותי, ואשר בחנו עניינים הנוגעים להיבטים רחבים יותר של מושג הרקורסיה, כמו יכולת ראייה רקורסיבית, מיומנויות ניסוח רקורסיבי שהוא לאו דווקא ניסוח פורמלי, ושימוש ברקורסיה לפתרון בעיות לא אלגוריתמיות. זאת ועוד, אפילו המסגרת התכנותית אינה עוסקת במגוון ההיבטים התכנותיים של רקורסיה. כך למשל, ממחקרו של אהרוני (1999) עולה כי אפילו סטודנטים למדעי-המחשב באוניברסיטה המגלים הבנה של המושג **תהליך רקורסיבי**, אינם מגלים הבנה הקשורה להיבטים אחרים של רקורסיה כמו למשל הבנה של המושג **מבנה רקורסיבי**. במלים אחרות, המסגרת התכנותית שבה מוצג בדרך כלל מושג הרקורסיה מתמקדת בעיקר בהיבטים תהליכיים. לפיכך היא לא תורמת להבנת היבטים תכנותיים אחרים של רקורסיה ובוודאי לא תורמת להבנת היבטים שאינם תכנותיים.

---

[2] באנגלית מכנה לירון מונחים אלה The three P's of computation – Procedure, Process and Product .

עד כה הוצגו שלושה הגורמים המרכזיים המתוארים בספרות המחקרית, אשר תורמים לקשיים של לומדים במהלך לימוד רקורסיה במדעי-המחשב. הגורם הראשון קשור בפער המורכבות בין ההליך הרקורסיבי לתהליך הרקורסיבי; הגורם השני קשור בבניית מודלים מנטליים לא אפקטיביים המייצגים את תהליך הפעלתם של הליכים רקורסיביים; והגורם השלישי קשור בהצגת רקורסיה במסגרת תכנותית צרה. לגורמי קושי אלה ניתן להוסיף גם את הדרכים בהן מקובל להציג את מושג הרקורסיה הן בספרי הלימוד של מדעי-המחשב והן בשיטות ההוראה המסורתיות. בן-ארי ורייך (1996) מסבירים כי שתי הגישות המקובלות להצגת רקורסיה הן שימוש בעצמים מהעולם האמיתי כאנלוגיה (כמו בובות רוסיות או משחקים), ושימוש בדיאגרמות שמאפשרות לתלמיד לעקוב אחר ביצוע ההפעלות הרקורסיביות (כמו מחסנית). אלא שלטענתו, שתי הגישות לוקות בחסר ואינן מאפשרות התמודדות ממשית עם הקשיים של הלומדים, ובמיוחד עם קשייהם של לומדים מתחילים. זאת ועוד, גם כאשר משתמשים כאנלוגיה בדוגמאות מחיי היומיום, שימוש זה אינו רחב היקף ואינו מגוון דיו (צצקיס, 1985; לוי, 1991). בפרקים הבאים של סקר הספרות יובאו מאמרים העוסקים בעיקר בגורם השני לקושי, דהיינו במודלים מוצעים להסבר רקורסיה (המאמר של Haynes, 1994) ובקשר בין סוג המודלים לסגנון הקוגניטיבי של הלומדים (המאמר השלישי של Wu, Dale & Bethel 1998).

הוראת רקורסיה

בעקבות זיהוי הקשיים המלווים את תהליך הלמידה וההוראה של רקורסיה, מציעה הספרות החינוכית דרכי הוראה והמלצות למורים למדעי-המחשב המבקשים לשלב מושג זה במהלך הוראתם (Anderson, Pirolli & Farrel, 1988; Harvey, 1985). חלק מההמלצות מופיעות גם במאמרים שיסקרו בהמשך, אולם הרשימה להלן כללית ומקיפה יותר ומבוססת גם על מקורות נוספים.

1. מאחר והמורכבות העיקרית מסתתרת בתהליך הרקורסיבי, מומלץ ל"הדחיק את החשיבה על התהליך ולראות את ההליך הרקורסיבי כתיאור של התוצר ולא של התהליך (Leron , 1988).
2. כדאי לעשות שימוש במודלים ממוחשבים או ויזואליים-דינמיים, המאפשרים להתבונן בשלבים שונים של ביצוע האלגוריתם הרקורסיבי (George, 2000-b; Bhuiyan, Greer & McCalla, 1994).
3. מומלץ להמחיז את התהליך הרקורסיבי על ידי הלומדים עצמם, באמצעות מודל האנשים הקטנים (Harvey, 1985; לוי, 1991). גם בן-ארי משתמש ברעיון ההמחזה, וממליץ על צירוף בעיה מהעולם האמיתי שניתן להמחיזה, לבעיה תכנותית שפתרונה מקביל בדיוק להמחזה. ההמחזה משלבת לטענת בן-ארי הנאה מהלימוד יחד עם ויזואליזציה דינמית של התהליך הרקורסיבי (Ben-Ari, 1997).
4. למעט ההמלצה הראשונה, ההמלצות שמובאות לעיל ממוקדות בהבנת התהליך הרקורסיבי. ואולם כדי להרחיב ולהעמיק את הבנת המושג "רקורסיה", מומלץ לעסוק גם בהיבטים נוספים שלו. בין השאר מופיעות בספרות המלצות לעסוק במהלך הלימוד במבני-נתונים רקורסיביים ובמבנים רקורסיביים מורכבים כמו פרקטלים (אהרוני, 1999), והמלצות להשתמש במספר שיטות שונות לבניית אלגוריתמים רקורסיביים ולניתוחם (Harvey & Wright, 1994)).
5. מומלץ להתחיל את לימוד המושג "רקורסיה" עם בעייה מתאימה. נהוג הוא להתחיל את לימוד הרקורסיה עם תכנית לחישוב עצרת, או לחישוב מספרי פיבונצ'י. שתי אלה בעיות מתמטיות, לראשונה יש גם פתרון איטרטיבי טריוויאלי והשנייה מממשת רקורסיה דו-ממדית ולכן קשה מדי עבור מתחילים (Ben-Ari, 1997). גם בעיית מגדלי הנוי המפורסמת מחייבת שימוש בשתי קריאות רקורסיביות. לצורך הקלה על קשיי הלומדים, במיוחד המתחילים שבהם, מומלץ לבחור כבעיה ראשונה בעיה עם רקורסיה ליניארית, ואשר אין עבורה פתרון איטרטיבי פשוט.
6. רצוי לפתח אצל הלומדים מיומנויות כלליות של ראייה רקורסיבית וחשיבה רקורסיבית (Roberts, 1986), תוך שימוש במגוון רחב של תופעות רקורסיביות מתחומי חיים שונים (חדשי, 1996; לוי, 1991; צצקיס, 1985).

חלק מההמלצות משתלבות במגמה כללית המאפיינת כיום את השינויים החלים בדרכי הוראת מושגים מתמטיים ומדעיים, ואשר מושפעים בין השאר מתפיסת עולם קונסטרוקטיביסטית של הלמידה, על פיה הלומד בונה את הידע שלו באופן פעיל, תוך כדי חתירה לתת משמעות להתנסויותיו האישיות. ההמלצות הללו היוו בסיס רעיוני לפיתוח חלק מחומרי הלימוד במדעי-המחשב במסגרת תכנית הלימודים החדשה לחטיבה העליונה בישראל, וניתן לראות להן ביטוי מפורש בחומרי הלימוד בהם משתמשים התלמידים ובגישות ההוראה בהם משתמשים המורים למדעי-המחשב. המושג רקורסיה נדון בתכנית הלימודים התיכונית בדגשים שונים וברמות קושי שונות, הן ביחידת הלימוד השניה "יסודות מדעי המחשב – 2" (בן-ארי וחוב', 1998), הן בחלופות של יחידת הלימוד השלישית "תכנות פונקציונלי" (לפידות, לוי ופז, 1999) ו"תכנות לוגי" (שרץ, הברמן, בן-זקן וגלוברמן, 1996) והן ביחידת הלימוד הרביעית "עיצוב תכנה" (ברנדס וחוב', 1997).

לדוגמה, לימוד מושג הרקורסיה ביחידה "תכנות פונקציונלי" נפתח, בהתאם להמלצות לעיל, בסדרת פעילויות רב-תחומיות לא-תכנותיות שמטרתן פיתוח החשיבה הרקורסיבית של התלמידים, בהצגת תופעות רקורסיביות הלקוחות ממגוון תחומים, ובדיון במגוון היבטים של מושג הרקורסיה תוך דגש על ההיבט המבני. בהמשך תקופת הלימוד העוסקת ברקורסיה מתנסים התלמידים בתכנות של פונקציות רקורסיביות והם משתמשים לשם כך בשיטות תכנות מגוונות ובמודלים שונים להדמיית התהליך הרקורסיבי (ראה תרשים 5). באופן זה מנסה יחידת הלימוד לתת מענה הן לחשיבות הכוללת של רקורסיה, והן לקשיי הלומדים עליהם מדווח בספרות.

תרשים 5: **דרכים שונות לתיאור פונקציה רקורסיבית** (ע"פ לפידות, לוי ופז 2001)

הפונקציה fact מקבלת מספר n ומחזירה את העצרת שלו. לדוגמה: (fact 3) ⇒ 1 * 2 * 3 = 6

תאור מילולי של הפונקציה עצרת n!

הערך של **עצרת** עבור מספר טבעי כלשהו n הוא:
אם n שווה ל – 0 הערך הוא **1** .
אחרת: הערך הוא המכפלה של n
בערך של **עצרת** עבור n - 1 .

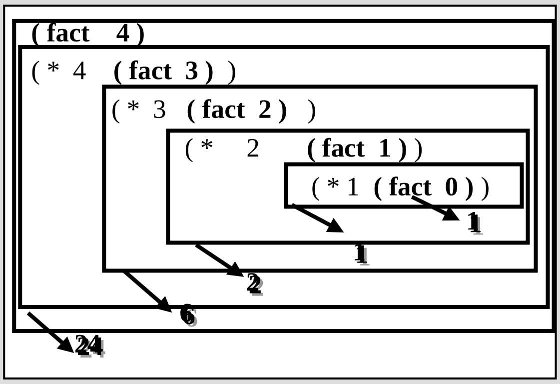
הגדרת הפונקציה

```
(define (fact n)
   (cond
    [( = n 0 ) 1]
    [else ( * n (fact (sub1 n )))] ))
```

מודל ההצבה

(fact 4) =
(* 4 **(fact 3)**) =
(* 4 (* 3 **(fact 2)**)) =
(* 4 (* 3 (* 2 **(fact 1)**))) =
(* 4 (* 3 (* 2 (* 1 **(fact 0)**)))) =
(* 4 (* 3 (* 2 (* 1 **1**)))) =
(* 4 (* 3 (* 2 **1**))) =
(* 4 (* 3 2)) =
(* 4 6) =
24

מודל המסגרות (Top-Down)



על המאמרים שתקציריהם מופיעים בהמשך

המאמרים המופיעים בהמשך נבחרו כך שיציגו היבטים שונים הקשורים בלמידה והוראה של רקורסיה, ופתרונות מעשיים אותם ניתן ליישם בכיתות במסגרת תכנית הלימודים החדשה במדעי המחשב. תקצירי המאמרים מובאים על פי מבנה הסעיפים במאמר המקורי ותוך שימוש בכותרות המקוריות. התקצירים מתמקדים בחלקי המאמר המשמעותיים ביותר למורי מדעי-המחשב בישראל, ובשל קוצר היריעה הושמטו מהם החלקים הפחות רלוונטיים [הערות עריכה יופיעו בסוגריים מרובעות, ד"ל].

המאמר הראשון - **מסגרת להוראת רקורסיה** (Ford, 1984) - מציע מסגרת כללית להתייחסות, מספק הסבר משלו למושג רקורסיה כמבנה חזרה כללי, ומציע שיטה לקבוע עבור אילו בעיות תכנותיות כדאי להשתמש ברקורסיה ועבור אילו בעיות כדאי להשתמש באיטרציה. הדגש במאמר הוא על פרדיגמות תכנותיות פרוצדורליות, אם כי התיאורים הרקורסיביים המובאים בו מנוסחים בשפה טבעית ולא תכנותית. המאמר מדגיש גם את המבנה בן שני החלקים של תיאורים רקורסיביים, כך שחלק אחד משתמש בפירוק רקורסיבי של הבעייה לתת-בעיות, וחלק שני מתאר את המקרה או המקרים הבסיסיים הפשוטים ביותר [שימו לב לכך שחשוב שיהיו שני חלקים כאלה בכל תיאור רקורסיבי, אולם הסדר אינו קבוע - כך שלעתים החלק המתאר את המקרה הבסיסי מופיע ראשון ולעתים הוא מופיע שני, ד"ל]. האופן הכללי שבו מוצג המושג רקורסיה במאמר, מהווה מעין מבוא לעיסוק הממוקד יותר בהיבטים ספציפיים של למידת והוראת רקורסיה בהם דנים שאר המאמרים.

המאמר השני - **כיצד להסביר רקורסיה ללומדים מתחילים** (Haynes, 1994) - מתאר את הגישות המקובלות להסבר כיצד מתבצע אלגוריתם רקורסיבי נתון ומציע גישה נוספת של סרטוט "עץ הפעלה". מאמר זה ממוקד בשאלה כיצד ניתן להסביר ללומדים קוד רקורסיבי נתון, והוא כמעט שאינו עוסק בשיטות ליצירת קוד כזה. הגישות המתוארות במאמר מהוות חלק מהיבט רחב של הוראת רקורסיה המכונה בספרות "מודלים להסבר רקורסיה".

במודלים כאלה, ובבחינת הקשר שביניהם לבין הסגנון הקוגניטיבי של התלמידים הנחשפים אליהם, עוסק גם המאמר השלישי – **מודלים וסגנונות קוגניטיביים בהוראת רקורסיה** (Wu, Dale & Bethel, 1998). חלק מהמודלים המוצגים במאמר זה מהווים חזרה על הגישות שהוצגו במאמר הקודם, אולם ההתמקדות כאן היא בשאלה עד כמה השימוש בסוג מוחשי או מופשט של מודלים קונספטואליים להסבר רקורסיה עשוי לעזור לתלמידים בעלי סגנון קוגניטיבי מוחשי או מופשט. מודלים מוחשיים, נטען במאמר, עוזרים יותר למתכנתים מתחילים.

המאמר הרביעי עוסק ביתר פירוט בשאלה **מה מתכנתים מתחילים יודעים אודות המושג רקורסיה** (Kahney, 1989). במאמר זה נבדקים המודלים המנטליים של מתכנתים מתחילים – 'טירונים' - בטרם חשיפתם למודל קונספטואלי מסוים להסבר של רקורסיה. המחקר המתואר במאמר העלה את הממצא כי טירונים נוטים להשתמש במודל המכונה "מודל הלולאה" בעוד מומחים מסתייעים ב"מודל ההעתקים" להסבר של תהליכים רקורסיביים. המאמר מסביר מודלים מנטליים אלה ומדגים את השימוש בהם עבור מספר תכניות הרשומות כאלגוריתם רקורסיבי מילולי (במעין פסיאודו-קוד). בנוסף, מאפיין המאמר מספר מודלים נוספים למודל הלולאה בהם מחזיקים תלמידים המאופיינים כטירונים בתחום התכנות, ומסכם בחשיבות השימוש במודלים מנטליים – גם אם אינם מדויקים.

המאמר החמישי והאחרון מביא אף הוא את הפן של הלומדים. המאמר **כיצד לומדי תכנות פונקציונלי מפרשים פונקציות רקורסיביות** (Segal, 1995) עוסק בלומדים הנדרשים לפענח פונקציות רקורסיביות. בעקבות סידרת מחקרים אמפיריים שהתבססו על תצפיות ועל ראיונות עם לומדים, זוהו שני קשיים עיקריים: הראשון קשור בתפיסה מוטעית של הלומדים אודות תהליכים רקורסיביים, והשני נובע מערבוב שהם עושים בין ההקשר המתמטי של המושג "משתנים" לבין ההקשר התכנותי שלו. במאמר נידונות האסטרטגיות שהלומדים מפעילים כדי להבין פונקציות רקורסיביות, והוא ממליץ על אסטרטגיה עדיפה – אסטרטגיה של "חישוב מושהה" (delayed evaluation). בחלקו הראשון של המאמר נסקרים בהרחבה היבטים שונים של רקורסיה, ומתוארים מספר מחקרים קודמים שעסקו בנושא. בכך מהווה מאמר זה גם סיכום לקובץ המאמרים כולו.

מסגרת להוראת רקורסיה

ג. פורד

Ford, Gary (1982). A framwork for teaching recursion. *SIGCSE Bulletin* 14(2), 32-39.

אודות המאמר

המסגרת ההוראתית המתוארת במאמר זה שימשה בהצלחה את הכותב בהוראת רקורסיה ללומדים בכל הרמות, כחלק מתכנית הלימודים הנהוגה במדינת אריזונה שבארה"ב. הרעיון הכללי הוא להתייחס **לרקורסיה** כאל מבנה בקרה תכנותי כללי (generalized control structure), אשר **איטרציה** היא מקרה פרטי שלו. בחלק הראשון של המאמר מוסבר המושג רקורסיה כמבנה חזרה כללי, ובחלק השני מתוארים מקרים בהם כדאי להשתמש ברקורסיה ומתוארת שיטה לקבוע עבור אילו בעיות תכנותיות כדאי להשתמש ברקורסיה ועבור אילו בעיות כדאי להשתמש באיטרציה. האלגוריתמים מתוארים במאמר זה במונחים כלליים ולא באמצעות שפת תכנות ספציפית, אם כי הדגש הוא על פרדיגמות תכנותיות פרוצדורליות. בסוף המאמר מובאים תיאורים רקורסיביים למבחר בעיות תכנותיות אופייניות, תוך הדגשת המבנה המקובל של תיאורים רקורסיביים הבנויים משני חלקים: חלק אחד המשתמש בפירוק רקורסיבי של הבעייה לתת-בעיות, וחלק שני המתאר את המקרה הבסיסי הפשוט ביותר.

<u>רשימת מושגים</u>

- איטרציה - Iteration
- זרימת בקרה – Flow of control
- מבני בקרה – Control structures
- מיון בחירה – Selection sort
- מקרה פשוט ביותר – Smallest instance
- פירוק בעייה – Problem decomposition
- פתרון בעיות – Problem solving
- רקורסיה - Recursion

הקדמה

בעבור מתכנתים רבים, רקורסיה מצויה במעין "אזור דמדומים" של מושגים תכנותיים המובנים רק באופן חלקי. בשל חשיבות המושג רקורסיה, על כל מי שעוסק בחינוך בתחום מדעי המחשב מוטלת אחריות להבטיח כי תלמידיו יבינו מושג זה היטב ויוכלו להשתמש בו כהלכה. המשימה החינוכית הזו אינה בלתי אפשרית, אם המסגרת המשמשת להוראת רקורסיה הולמת גם את שאר מושגי היסוד בתכנות. מסגרת כזו מוצעת להלן.

בעת הצגת מושג תכנותי חדש, אנו נדרשים לענות על שתי שאלות:

1. מהו המושג?
2. כיצד משתמשים במושג?

בהמשך המאמר, מוצגות תשובות לשאלות אלה במקרה של המושג רקורסיה.

מהי רקורסיה?

בתכנות פרוצדורלי ניתן להשתמש בשלושה סוגים של מבני בקרה:

(1) **מבנה סדרתי** – שבו משפטי התכנית מתבצעים על פי הסדר שבו הם רשומים.

(2) **מבנה הסתעפות** – המשמש לבחירה בין שני משפטים או יותר, או בין גושי משפטים (blocks).

(3) **מבנה חזרה** – המציין כי יש לחזור על ביצועו של משפט בתכנית או על ביצוע גוש של משפטים.

הסוג השלישי של מבני בקרה מציין לא רק על מה לחזור, אלא גם את התנאים שבקיומם החזרה אמורה להיפסק. זאת כמובן כדי למנוע חזרה אינסופית.

מבנה חזרה נראה כגוש של משפטים, שהאחרון בהם הוא משפט של החלטה האם לבצע שוב את הגוש אם לאו:

משפט-1
משפט-2
.
.
.
משפט-n (החלטה אם לבצע שוב)

אם מצרפים מבנה חזרה כזה ל**מבנה הסתעפות**, יתקבל מבנה בקרה שעשוי לא לבצע את גוש המשפטים אפילו פעם אחת:

משפט (החלטה אם לבצע פעם אחת)
משפט-1
משפט-2
.
.
.
משפט-n (החלטה אם לבצע שוב)

השתמשנו בביטוי ***החלטה אם לבצע שוב*** בתיאור מבני החזרה. זהו מושג מופשט וקריטי להבנה של מבני בקרה. אלא שעבור לומדים רבים, ההכרות עם מבנה בקרה כלשהו מלווה בדוגמה של מימוש מוחשי של אותו מבנה, מה שמקשה על הלומד באבחנה שבין המושג המופשט ובין המימוש שלו. לדעתנו, חשוב שהלומדים לא יזהו את המושגים המופשטים הקשורים בבקרה עם מימושים ספציפיים, שמוצגים מטבע הדברים ברמת הפשטה נמוכה יותר.

ניתן עתה להגדיר **רקורסיה** כהכללה של מבנה בקרה מהסוג השלישי. ראשית, נגמיש את הדרישה כי משפט *החלטה אם לבצע שוב* יופיע **בסוף** גוש המשפטים לביצוע חוזר, ונאפשר לו להופיע בכל מקום **בתוך** הגוש:

משפט-1
משפט-2
.
.
.
משפט-k (החלטה אם לבצע שוב)
.
.
.
משפט-n

שנית, ניתן להגמיש את הדרישה הסמויה כי בגוש המשפטים יופיע רק משפט אחד שגורם לביצוע הגוש שוב. כפי שנראה בהמשך, לעתים קרובות רצוי לאפשר כמה משפטי החלטה כאלה בתוך אותו גוש.

נשתמש במונח **איטרציה** (iteration) עבור המקרה המיוחד שבו המשפט היחיד של ***החלטה אם לבצע שוב*** מופיע בסוף גוש המשפטים. המונח **רקורסיה** יציין את אותם מקרים שבהם משפט זה או מספר משפטים כאלה מופיעים באמצע הגוש. [המחבר משתמש במונח איטרציה לתיאור מקרים של רקורסית קצה או רקורסית זנב, ד"ל].

עתה נבחן שוב את הביטוי ***החלטה אם לבצע שוב.*** אם ההחלטה שלילית, זרימת הבקרה של התכנית עוברת למשפט הבא לאחר גוש המשפטים. אם ההחלטה חיובית, מתרחשת למעשה הפרעה זמנית בזרימת הבקרה הטבעית. בתום הביצוע החוזר של גוש המשפטים, הבקרה חוזרת למשפט אשר מופיע מיד לאחר המשפט ***החלטה אם לבצע שוב.*** התיאור הזה דומה מאד לתיאור הקשרים שבין התכנית הראשית לתת-תכניות.

במימוש של מבנה בקרה, מתכנני שפת התכנות מנסים להגיע ליעילות. ההכרה בכך שמיקום משפט ההחלטה בסוף גוש המשפטים היא מקרה פרטי, מובילה למימוש יעיל יותר. בדרך כלל, הקישור לתת-תכנית מחייב שמירה של כתובת לחזרה (return address). במקרה הפרטי בו אנו דנים כעת, הכתובת לחזרה היא כתובתו של המשפט הבא מיד לאחר גוש המשפטים. לכן במקרה זה יעיל יותר להמנע משמירת הכתובת לחזרה, שכן זרימת הבקרה ממילא תמשיך למשפט הבא. כל המימושים של מבני בקרה איטרטיביים מנצלים עובדה זו.

במבני בקרה רקורסיביים, לעומת זאת, משפטי ההחלטה <u>לא</u> מופיעים **בסוף** הגוש, ועם סיום הביצוע החוזר אמורה הבקרה לחזור למשפט המצוי **בתוך** הגוש ולא מיד לאחריו. לפיכך רקורסיה מחייבת שמירה של כתובת לחזרה, בדיוק כמו במקרה של קישור לתת-תכנית. זוהי סיבה אחת למימוש רקורסיה באמצעות תבנית תחבירית דומה לתבנית המשמשת להפעלה של תת-תכנית. אולם לא הכרחי לממש רקורסיה באמצעות קריאה לתת-תכנית. אפשרות נוספת היא ליצור כלי תכנותי מיוחד לצורך כך. הכלי החדש יצטרך לאפשר **שמירה של מידע** שנוצר בכל פעם שמתרחשת הפרעה זמנית לזרימת הבקרה הטבעית, וכן יצטרך לתת אפשרות **לכנות בשם** את גוש המשפטים עליו חוזרים, כדי שניתן יהיה להפנות את זרימת הבקרה אליו וממנו. שני מנגנונים אלה קיימים כמובן גם בעת שימוש בתת-תכנית.

ניתן אם כן להתייחס לרקורסיה כאל מבנה בקרה כללי, בלי קשר לאופן המימוש שלו. התייחסות כזו מאפשרת להציג את המושג ללומדים כצעד קטן נוסף קדימה, מעבר לידע שלהם על מבני הבקרה הבסיסיים.

כיצד משתמשים ברקורסיה?

מבני הבקרה עבור חזרה, הן האיטרטיביים והן הרקורסיביים, התפתחו משתי סיבות. ראשית התפתחה ההכרה בכך שאם רוצים להשתמש n פעמים באותו גוש משפטים, אין זה הגיוני לרשום את גוש המשפטים הזה n פעמים [מה גם שלעתים קרובות, n לא ידוע מראש – ד"ל]. שנית, נראה היה כי מבנה חזרה מתאים לא רק שימנע כתיבה כפולה ומכופלת של שורות קוד, אלא גם יאפשר הכללה תוך כדי שימוש במשתנה לייצוג מספר הפעמים לחזרה. אם כן, מתי וכיצד יש להשתמש במבנה חזרה, ובאיזה סוג של מבנה חזרה? להלן העיקרון הבסיסי שעל הלומדים לזכור:

ניתן להשתמש ברקורסיה כדי לפתור בעיה, אם הבעיה היא בעלת התכונות הבאות –

(1) כדי לפתור חלק כלשהו מהבעיה, יש לפתור חלק קטן יותר (או חלקים קטנים יותר) של אותה בעיה.

(2) ניתן לפתור בקלות את המקרים הפשוטים ביותר של הבעיה.

עיקרון זה מכוון אותנו גם לאופן שבו משתמשים ברקורסיה. **אלגוריתם רקורסיבי** יצטרך לקבוע אם מקרה מסוים הוא המקרה הפשוט ביותר של הבעיה. אם כן, ניתן להשתמש בפתרון ישיר. אם לא, יש לזהות מקרה פשוט יותר של אותה בעיה.

נדגים את העיקרון הנ"ל עבור מספר בעיות תכנותיות [המאמר המקורי מביא 7 בעיות, וכולל גם את המימוש בפסקל; בתקציר זה הסתפקנו ב-3 בעיות וויתרנו על המימוש – ד"ל].

דוגמה 1: המספר הגדול ביותר

הבעיה היא למצוא את המספר השלם הגדול ביותר במערך שמכיל n איברים. ניתן להשתמש כאן ברקורסיה משום ש:

(1) כדי למצוא את המספר הגדול ביותר מתוך n מספרים במערך, ניתן למצוא את המספר הגדול ביותר מבין n-1 המספרים הראשונים במערך ולהשוות אותו למספר האחרון במערך.

(2) במקרה הפשוט שבו יש רק מספר אחד, קל מאד לקבוע כי זהו גם המספר הגדול ביותר.

דוגמה 2: הדפסת מערך מהסוף להתחלה

הבעיה היא להדפיס את הערכים במערך בסדר הפוך. ניתן להשתמש כאן ברקורסיה משום ש:

(1) כדי להדפיס n ערכים במערך בסדר הפוך, ניתן להדפיס תחילה את הערך המצוי במקום האחרון במערך ואז להדפיס את n-1 הערכים המצויים בתחילת המערך בסדר הפוך.

(2) במקרה הפשוט שבו יש רק ערך אחד, קל מאד להדפיס אותו ישירות.

דוגמה 3: מיון בחירה

הבעיה היא לסדר מערך של n מספרים בסדר עולה בשיטת מיון בחירה (selection sort). בשיטה זו, בוחרים בכל פעם איבר מהחלק הלא-ממוין של המערך וממקמים אותו במקום הסופי הרצוי. ניתן להשתמש כאן ברקורסיה משום ש:

(1) אפשר למצוא תחילה את המספר הגדול ביותר במערך ולהחליפו עם המספר המצוי במקום האחרון במערך. לאחר מכן נותר למיין באותה שיטה את n-1 האיברים הראשונים במערך.

(2) במקרה הפשוט שבו יש רק מספר אחד, המערך כבר ממוין.

[בהמשך המאמר מובאת רשימת בעיות נוספות הנחשבות על ידי המחבר כדוגמאות טובות אותן ניתן להציג בשלבים שונים של לימוד, וביניהם חיפוש וטיול בעץ בינארי, מיון-מיזוג ועוד. הבעיות הללו אינן מוצגות במאמר בפירוט – ד"ל].

לחזור או לא לחזור – זו השאלה

לעתים משתמשים ברקורסיה כדי לפתור בעיות שניתן לפתרן גם באמצעות מבנה חזרה איטרטיבי. כיצד ניתן לקבוע איזה מבנה חזרה עדיף? התשובה תלויה באופן שבו מחלקים את הבעיה לחלקים. במקרים רבים, ניתן לחלק בעיה לתת-בעיות שהן דומות זו לזו אבל אינן דומות לבעיה המקורית. תת-הבעיות הן אם כן עצמאיות, כך שפתרון כל אחת מהן יכול להסתיים בטרם ניגשים לפתרונה של האחרת. במלים אחרות, בהינתן גוש משפטים שפותר תת-בעיה כלשהי, ניתן לשלב את המשפט ***החלטה אם לבצע שוב*** בסופו. במקרים אלה איטרציה היא מבנה החזרה המועדף.

סוג אחר של פירוק בעיה לתת-בעיות מופיע בדוגמאות שהובאו לעיל. בכל הדוגמאות פורקה הבעיה המקורית לשתיים: תת-בעיה אחת שאינה דומה למקורית (זהו תמיד המקרה הפשוט ביותר), ותת-בעיה אחרת **זהה** למקורית אבל קטנה יותר (פשוטה יותר). למרות שהבעיות שהובאו כדוגמה הן בעלות התכונות ההכרחיות לפתרון רקורסיבי, ההכרעה אם להשתמש במבנה חזרה איטרטיבי או רקורסיבי תהיה תלויה במיקומו של המשפט ***החלטה אם לבצע שוב*** בגוש המשפטים המממש את הפתרון. כאשר משפט זה מופיע בסוף הגוש, עדיף להשתמש במבנה איטרטיבי, וזה המצב בדוגמה של הדפסת מערך בסדר הפוך או בדוגמה של מיון בחירה. ואילו כאשר משפט ההחלטה חייב להופיע בתוך הגוש ולא בסופו, יש להשתמש במבנה רקורסיבי, כמו בדוגמה של מציאת המספר הגדול ביותר. יחד עם זאת, ניתן לפרק את בעיית מציאת המספר הגדול ביותר גם בדרך שונה מזו שהצענו לעיל, באופן שיאפשר פתרון איטרטיבי.

הבעיה של מגדלי הנוי מהווה דוגמה לבעיות שבהן מפרקים את הבעיה המקורית ליותר מתת-בעיה אחת זהה למקורית. בעבור בעיות כאלה, רקורסיה היא מבנה החזרה המועדף. למעשה, לבעיות מסוג זה קשה מאד לתת פתרון איטרטיבי.

סיכום

הראינו כי רקורסיה היא מבנה חזרה כללי, אשר איטרציה היא מקרה פרטי שלו. אם מציגים סוג זה של מבני בקרה בפני הלומדים במנותק מפרטי המימוש של כל אחד מהמבנים, ניתן להתייחס לרקורסיה כמבנה השונה רק במעט מאיטרציה.

הראינו כי הבעיות שבעבורן רקורסיה מספקת פתרון עדיף הן בעיות בעלות שני מאפיינים: ניתן לפרק את הבעיה המקורית לתת-בעיות זהות לה, וניתן לפתור באופן מיידי את המקרה הפשוט ביותר של הבעיה. במצבים שבהם פתרון המקרה הפשוט ביותר מופיע בסוף האלגוריתם, ניתן להשתמש במבנה איטרטיבי ולעתים הוא אף עדיף. לאחר שהלומדים נחשפים למספר דוגמאות של עיקרון זה, הם בדרך כלל מסוגלים לזהות בעיות שבעבורן רקורסיה אפשרית, וכאלה שבעבורן עדיפה איטרציה.

ולבסוף, האופן שבו משתמשים ברקורסיה כדי לפתור בעיות מייצג את התכונות שמצדיקות את המימוש של מבנה חזרה רקורסיבי באמצעות קריאה לפרוצדורה או לפונקציה. תכונות אלה הן: הצורך להיכנס לתוך גוש משפטים מזוהה בשם ולצאת ממנו, הצורך לשמור ערכי משתנים במהלך ההפרעה הזמנית לזרימת הבקרה, והצורך לתת ערכים חדשים לחלק מהמשתנים במהלך הביצוע. זאת ועוד, המושגים הקשורים במבני בקרה ובפיתוח אלגוריתמים תוך כדי פירוק בעיה לתת-בעיות הם מושגים בולטים בכל תכניות הלימודים במדעי המחשב. לפיכך, ראוי להשתמש במושגים אלה כמסגרת להוראת רקורסיה.

[מן הראוי להוסיף כי בשנים שחלפו מאז פורסם המאמר, נערכו מחקרים שבדקו כיצד משפיעה הכרות מוקדמת עם מבנה איטרטיבי של לולאה על האופן שבו לומדים מתחילים תופסים מבנים רקורסיביים, ונמצא כי אחת מהתפיסות המוטעות הרווחות ביותר של רקורסיה קשורה בהתייחסות לרקורסיה כאל לולאה. לפיכך, על המורים לשקול היטב באיזה אופנים הם מיישמים את ההשוואה הנדונה במאמר לעיל עם תלמידיהם – ד"ל].

כיצד להסביר רקורסיה ללומדים מתחילים

ס. היינס

Haynes, S.M. (1995). Explaning recursion to the unsophisticated. *SIGCSE Bulletin* Vol. 27, No. 3, 3-6.

אודות המאמר

במוקד המאמר הנוכחי עומדות גישות להסבר כיצד מתבצע אלגוריתם רקורסיבי נתון. נקודת המוצא של הכותב היא שהאלגוריתם הרקורסיבי, כאשר הוא נרשם באמצעות שפת תכנות, נותן למעשה ביטוי להגדרה אינדוקטיבית של בעייה כלשהי. אבל ההגדרה האינדוקטיבית הזו היא קומפקטית ומופשטת, כך שמתכנתים מתחילים מתקשים בהבנתה. לכן מתאר הכותב מספר דרכים ידועות להסבר של אלגוריתם רקורסיבי נתון ומציע דרך אחת חדשה אותה הוא מכנה "גישת עץ ההפעלה". השימוש בגישת עץ ההפעלה נוסה על ידי הכותב בהצלחה עם סטודנטים מתחילים הלומדים מדעי המחשב. המאמר מתאר את הגישה ומדגים אותה באמצעות התייחסות לפונקציית אקרמן הרקורסיבית, הנתונה בשפת C. כמו כן מתייחס המאמר לאופן שבו ניתן לקשר בין הגישה המוצעת לבין יצירתן של הגדרות אינדוקטיביות. [הגדרה אינדוקטיבית היא המונח המוצע על-ידי הכותב עבור מה שכונה בחלק הראשון של סקר הספרות 'הגדרה רקורסיבית', ראו תרשים 1 במבוא,ד"ל].

<u>רשימת מושגים</u>

- אינדוקציה - Induction
- טיול לעומק בעץ - Tree depth first traversal
- מחסנית הפעלה - Runtime stack
- מספרי פיבונצ'י – Fibonacci numbers
- עץ הפעלה - Activation tree
- עץ, צמתים ועלים - Tree, nodes and leaves
- פונקצית אקרמן – ackermann function
- רשומת הפעלה - activation record

הגדרת הבעייה

המושג "רקורסיה" נחשב כקשה עבור מתכנתים מתחילים, למרות חשיבותו הרבה אפילו בשלבים המוקדמים של תכנית הלימודים. לכן אתגר הוא להסביר מהי רקורסיה לקהל של מתחילים, אשר נוטים ללכת לאיבוד ביער הפרטים הנלווה למעקב אחר תכניות רקורסיביות.

כותב המאמר נוהג להשתמש בכל חמש הדרכים המוצעות בהמשך כדי להסביר רקורסיה, ארבע הראשונות מהן מוכרות וידועות ולהן הוספתי את הגישה של שימוש בעץ ההפעלה activation tree. תלמידי מוצאים את הגישה האחרונה מאירת עיניים, באשר היא מאפשרת בקרה על כל הפרטים הנחוצים.

המאמר מתאר בקצרה את הגישות הנפוצות להסבר רקורסיה: גישת האינדוקציה, גישת המחסנית, גישת המעקב וגישת העץ הרקורסיבי. לאחר מכן מתוארת גישת עץ ההפעלה, ומוסבר כיצד היא קשורה לגישות האחרות. לבסוף מובאת דוגמה של תכנית רקורסיבית מסובכת, ושל המעקב אחריה באמצעות עץ ההפעלה. כמו כן מובאת דוגמה פשוטה יותר המסבירה כיצד לעזור ללומדים לשפר את הבנת המושג "אינדוקציה".

גישות להסבר המושג רקורסיה

א. גישת האינדוקציה

בגישה זו מביאים את ההגדרה של הפונקציה, כלומר – הפונקציה מוגדרת במונחים שלה עצמה ושל מקרה בסיסי. ההגדרה של ביטויים בשיטת BNF היא דוגמה אחת כזו. ההגדרה האינדוקטיבית של סדרת פיבונצ'י היא:

```
Fib(k)=Fib(k-1) + Fib(k+2);
Fib(2)=1;
Fib(1)=1;
```

גישה זו היא בעלת כוח רב, אולם הלומדים בדרך כלל מתקשים להבין את המלים ואת הסימון בלי להבין את המשמעות. הבנה של אינדוקציה מחייבת הפשטה ברמת-מומחיות שאינה קיימת בדרך כלל אצל מתחילים.

ב. גישת המחסנית

רקורסיה ממומשת באמצעות דחיפה (push) והוצאה (pop) של רשומות הפעלה למחסנית הביצוע או ממנה. בכל קריאה לפרוצדורה, נדחפת למחסנית רשומת ההפעלה הכוללת את המשתנים הלוקליים, הפרמטרים, הכתובת לחזרה ומידע נוסף. כאשר הפרוצדורה מסתיימת ומחזירה את הבקרה לפרוצדורה שקראה לה, מוצאת רשומת ההפעלה מהמחסנית.

היתרון של שימוש במחסנית הביצוע להסבר רקורסיה הוא בכך שרשומת ההפעלה שומרת על כל המידע ועל כל הפרטים הנחוצים. החסרון הוא בשימוש במדיום סטטי כמו נייר ועפרון או לוח, בעוד מחסנית הביצוע היא תופעה דינמית.

ג. גישת המעקב

בגישה זו, בכל פעם שקוראים לפרוצדורה, מדפיסים את שמה ואת ערכי הפרמטרים שלה, ובכל פעם שביצוע הפרוצדורה מסתיים, מדפיסים את שמה ואת הערך המוחזר.

אפשר לחשוב על גישת המעקב כעל ייצוג סטטי של המחסנית הדינמית. המידע היחיד מהמחסנית שבו משתמשים בגישת המעקב הוא ערכי הפרמטרים, הערך המוחזר וסדר הקריאה. גישת המעקב עשויה להיות מאד אפקטיבית, במיוחד בשילובה עם גישות הסבר נוספות. יחד עם זאת, גישת המעקב הופכת להיות מורכבת למדי במקרה של רקורסיה רב-מימדית, כמו המקרים של סידרת פיבונצ'י ופונקציית אקרמן.

ד. גישת העץ הרקורסיבי

בעץ רקורסיבי כל צומת מכיל את הפרמטרים ואת המשתנים המקומיים של סביבת ההפעלה, כך שלמעשה כל צומת מייצג הפעלה של פרוצדורה מסוימת בסביבה מסוימת. מאחר שהעץ מאפשר ייצוג דו מימדי, קל יותר לבחון ולהשתמש בגישת העץ מאשר בגישת המעקב. יתרון נוסף הוא בכך שניתן להגיע בקלות יחסית מגישת העץ לגישת האינדוקציה. מעבר זה מסובך יותר אם יוצאים מגישת המחסנית או מגישת המעקב.

חסרונה של גישה זו הוא בכך שהצומת לא מכיל את הערך המוחזר. כמו כן מתעוררים קשיים בבניית העץ כאשר הפונקציה מכילה יותר משתי קריאות לעצמה. שני חסרונות אלה ניתנים לשיפור בגישה של עץ ההפעלה שתוצג להלן.

ה. גישת עץ ההפעלה

גישה זו היא קומבינציה של גישת המחסנית ושל גישת העץ הרקורסיבי, והיא משלבת בין היתרונות של שתיהן. ארגון המידע המוכל בכל אחת מרשומות ההפעלה של מחסנית הביצוע באמצעות עץ רקורסיבי, הופך את המעקב אחר הביצוע לפשוט יותר. למעשה הגישה הנוכחית היא הרחבה של גישת העץ הרקורסיבי, אבל היא הוכיחה עצמה כיעילה במיוחד עבור התלמידים.

דוגמה לשימוש בגישת עץ ההפעלה

כדי להסביר מהו עץ הפעלה, נשתמש באותו מתכון אותו ניתן לתת גם לתלמידים. ניקח לדוגמה את פונקציית אקרמן, הרשומה להלן בתחביר של שפת C [ובניסוח מתמטי מקובל – ד"ל].

```
int ack(int m, int n) {
    if (m==0) return (n + 1);
    if (n==0) return (ack (m-1,1) );
    return (ack (m –1, ack (m, n – 1) ) );
 }
```

$$f(m,n) = \begin{cases} n+1 & m=0 \\ f(m-1,1) & n=0 \\ f(m-1,f(m,n-1)) & \end{cases}$$

הוראות:

[ההוראות מנוסחות באופן כללי, להפעלה של פרוצדורות ופונקציות גם יחד. לשם קיצור משתמש הכותב במונח פרוצדורה והכוונה היא לפרוצדורה/פונקציה, ד"ל].

1. בנו תבנית של רשומת הפעלה. על התבנית להכיל את שם הפרוצדורה, הפרמטרים, והערך המוחזר.

תבנית ack

ack m n	Returned value

תבנית כללית

Name Par1 Par2	Returned value

2. התבוננו בקוד. אם הפרוצדורה קוראת לעצמה יותר מפעם אחת, סמנו כל קריאה רקורסיבית כזו במספר – למשל ack_1, ack_2 וכדומה (למעשה, הסימון הזה הוא מעין הפשטה של הכתובת לחזרה)

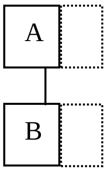
```
int ack(int m, int n) {
    if (m==0) return (n + 1);
    if (n==0) return (ack1 (m-1,1) );
    return (ack2 (m –1, ack3 (m, n – 1) ) );
 }
```

3. ציירו את עץ ההפעלה באמצעות מעקב אחר התכנית. התחילו עם ההפעלה הראשונה של הפרוצדורה, וציירו בשורש של העץ את התבנית עם הערכים המתאימים של הפרמטרים והמשתנים המקומיים.

4. כל קריאה לפרוצדורה מיוצגת בעץ כצומת-בן של הצומת הנוכחי. כדי לבצע את הקריאה, יש לרדת למטה לצומת החדש.

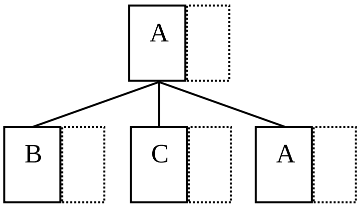
א. אם בצומת הנוכחי יש מספר קריאות, מציירים את הצמתים הבנים על פי הסדר שבו הם מופעלים, משמאל לימין.

לדוגמה: אם A מפעיל את B העץ נראה כך –



ואם A מפעיל את B ולאחר מכן

את C ואת A, העץ נראה כך -



ב. אם ערכי הפרמטרים מחושבים באמצעות הקריאה הרקורסיבית, יש לחשב את הערכים לפני שניתן להעביר אותם לפרוצדורה כלשהי. במקרה כזה יש להוסיף לעץ צומת-בן לצומת הנוכחי.

ג. אם ערכי הפרמטרים מחושבים במהלך ביצוע הקריאה הרקורסיבית, ואז מועברים לקריאה רקורסיבית אחרת (כמו במקרה של פונקצית אקרמן בהפעלה של ack_3), יש לחשב תחילה את הערכים ואז לקרוא לפרוצדורה עם ערכים אלה. המשמעות היא שיש צומת-בן עבור כל חישוב של אחד מהפרמטרים ואחר כך נוסף צומת-בן עבור הפרוצדורה שמשתמשת בערכים המחושבים.

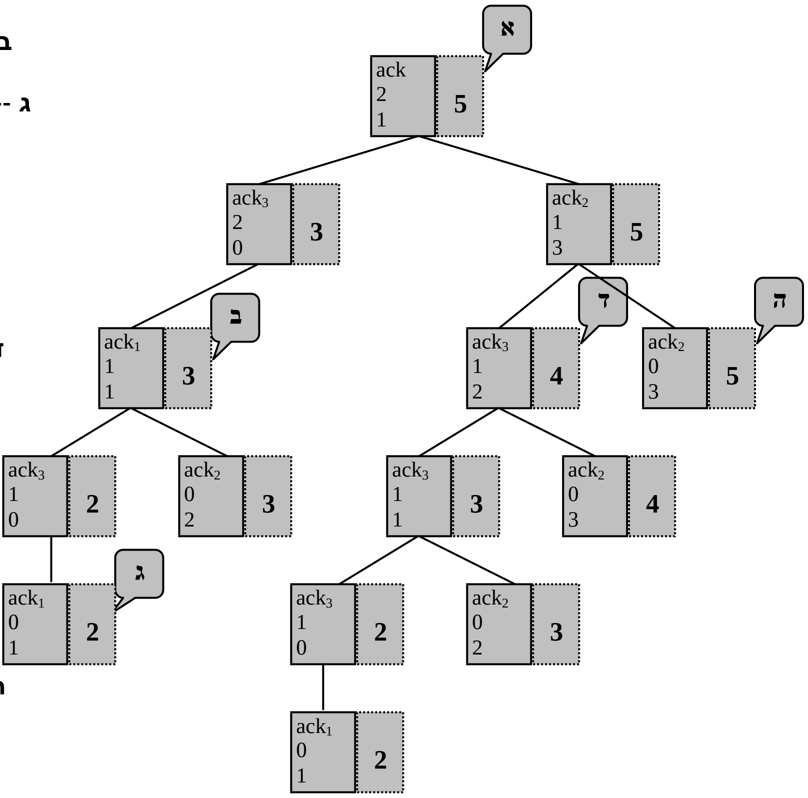
5. החזרה מהפרוצדורה משמעותה מילוי הערך המוחזר בתבנית, ועלייה בעץ לצומת האב. העלים בעץ מייצגים פרוצדורות שאינן מפעילות פרוצדורות אחרות.

6. הביצוע הדינמי של התכנית תואם את "הטיול לעומק" (depth-first traversal) בעץ ההפעלה.

כדי להשלים את הדוגמה, ניתן להביא את המעקב אחר פונקצית אקרמן כאשר ערכי הפרמטרים ההתחלתיים שלה הם m=2 ו n=1. נתבונן לשם כך בהדפסות המעקב המובאות בעמוד הבא, ונשווה אותו לעץ ההפעלה המתאים המובא לצד המעקב. שימו לב לסימונים א,ב,ג,ד,ה במקומות תואמים בהדפסת המעקב ובעץ ההפעלה.

[במאמר המקורי מתוארת דוגמה נוספת, של פונקציה לחישוב מספרי פיבונצ'י – ד"ל].

עץ הפעלה



הרצת מעקב

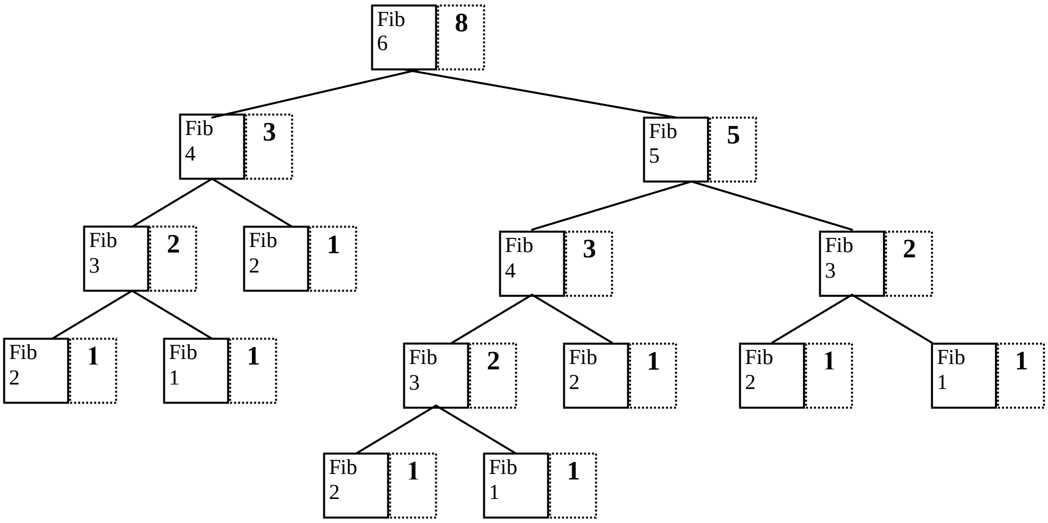
```
> (ack 2 1)
Entering: ack, Argument list: (2 1)            -- א
  Entering: ack, Argument list: (2 0)
    Entering: ack, Argument list: (1 1)  -- ב
      Entering: ack, Argument list: (1 0)
        Entering: ack, Argument list: (0 1) -- ג
        Exiting: ack, Value: 2
      Exiting: ack, Value: 2
      Entering: ack, Argument list: (0 2)
      Exiting: ack, Value: 3
    Exiting: ack, Value: 3
  Exiting: ack, Value: 3
  Entering: ack, Argument list: (1 3)
    Entering: ack, Argument list: (1 2) -- ד
      Entering: ack, Argument list: (1 1)
        Entering: ack, Argument list: (1 0)
          Entering: ack, Argument list: (0 1)
          Exiting: ack, Value: 2
        Exiting: ack, Value: 2
        Entering: ack, Argument list: (0 2)
      Exiting: ack, Value: 3
      Entering: ack, Argument list: (0 3)
      Exiting: ack, Value: 4
    Exiting: ack, Value: 4
    Entering: ack, Argument list: (0 4) -- ה
    Exiting: ack, Value: 5
  Exiting: ack, Value: 5
Exiting: ack, Value: 5
```

המעבר לשימוש בגישת האינדוקציה

לאחר שהלומדים מתנסים במעקב אחר פרטי הביצוע של אלגוריתמים רקורסיביים, חשוב גם שירכשו מומחיות בחשיבה המופשטת יותר. על הלומדים להיות מסוגלים לעבור מרמת התהליך המפורט והמידע הרב המעובד במהלכו לרמה של הגדרה אינדוקטיבית מופשטת של הפונקציה המתבצעת תוך שימוש ברקורסיה. מכל הגישות שתוארו קודם, גישת עץ ההפעלה היא המתאימה ביותר בשלב זה. מבנה העץ נובע מתהליך הביצוע של האלגוריתם, והביצוע קשור באופן הדוק להגדרה האינדוקטיבית ששימשה לכתיבת הפונקציה.

נתבונן למשל בעץ שיצוייר עבור ביצוע של חישוב האיבר השישי של מספרי פיבונצ'י. העץ מראה כיצד לצומת של הפעלת Fib 6 יש שני בנים, כלומר החישוב של הערך המוחזר מ- Fib 6 עושה שימוש בערכים המוחזרים מההפעלות של Fib 5 ו- Fib 4.

[הדוגמה המקורית המובאת במאמר היא של הפעלת Fib 5. כדי למנוע בלבול בין הפרמטר של ההפעלה הזו לבין הערך המוחזר, בחרתי לתאר כאן את הפעלת Fib 6 שבה הפרמטר שונה מהערך המוחזר, ד"ל].



השימוש בתיבה מיוחדת עבור הערכים המוחזרים מכל הפעלה מקלה אף היא על הלומדים. עתה נותר רק להחליף את המקרה הפרטי של הפרמטר 6 בהפעלה של Fib 6 בפרמטר כללי n, ומבנה העץ כבר ממחיש את השימוש אותו Fib (n) עושה בערכים של Fib (n-1) ו- Fib (n-2). העלים של העץ תואמים כמובן למקרה הבסיסי - base case (או למקרים הבסיסיים) של ההגדרה האינדוקטיבית.

מאחר שההגדרה האינדוקטיבית נותנת תיאור קומפקטי ומאד מופשט של רקורסיה, חשוב להבין אותה. עץ ההפעלה, עם מבנהו הויזואלי, מציע סיוע נוח לתלמיד המתאמן בחשיבה אינדוקטיבית.

מסקנות

הבעיות הרקורסיביות הניתנות ללומדי תכנות מתחילים אינן קשות במיוחד, ויחד עם זאת הסבר בעיות אלו, אפילו של הפשוטות מביניהן, הוא משימה מורכבת. אחת הסיבות לכך היא מורכבות המעקב אחר קוד המכיל הרבה פרוצדורות ממתינות בעלות אותו שם. קיימות גישות אחדות לעזור למתחילים להבין רקורסיה. לאלה מוסיף המאמר הנוכחי את גישת עץ ההפעלה – גישה שתלמידים המשתמשים בה בצירוף עם גישת המעקב מוצאים אותה נוחה ומבהירה.

מודלים וסגנונות קוגניטיביים בהוראת רקורסיה

צ'. וו, נ. דייל ול. בתל

Wu, C., Dale, N.B., & Bethel, L.J. (1998). Conceptual models and cognitive styles in teaching recursion.*SIGCSE 30*, Atlanta GA, USA, 292-296.

אודות המאמר

המאמר מתאר מחקר שבדק עד כמה השימוש בסוג מוחשי או מופשט של מודלים קונספטואליים להסבר רקורסיה עשוי לעזור לתלמידים בעלי סגנון קוגניטיבי מוחשי או מופשט. תחילה מוסבר מהו מודל קונספטואלי ומתוארים סוגים שונים של מודלים כאלה להסבר רקורסיה. בהמשך מוצגת התיאוריה של קולב (Kolb) בדבר מימדים לאבחון סגנון קוגניטיבי של למידה. לבסוף מובאים ממצאי המחקר, לפיהם המודלים המוחשיים עוזרים יותר למתכנתים מתחילים אולם אין בכך כדי לקבוע קשר בין סוג המודל לסגנון הקוגניטיבי.

<u>**רשימת מושגים**</u>

- מודל אינדוקציה מתמטית – Mathematic induction model
- מודל הבובות הרוסיות – Russian dolls model
- מודל המעקב – Trace model
- מודל התבניות – Structure template mode
- מודל מנטלי - Mental model
- מודל סימולציה של מחסנית – Stack simulation model
- מודל קונספטואלי – Conceptual model
- סגנון קוגניטיבי מוחשי – Concrete cognitive style
- סגנון קוגניטיבי מופשט– Abstract cognitive style

הקדמה

מודל מנטלי הוא ייצוג של מושג מופשט או של מערכת מורכבת, אשר עשוי לספק הסברים ויכולת חיזוי למי שמנסה להבין את המושג באמצעות המודל. *מודל קונספטואלי* הוא מודל שמאפייניו מוגדרים על ידי מורים, מדענים או מהנדסים, ואשר אמור לספק ייצוג מתאים למערכת-מטרה כלשהי. ניתן להגדיר הבנה של מערכת כרכישה של מודל מנטלי מתאים של המערכת.

במסגרת הוראת מדעי המחשב, מקובל לחשוב כי רקורסיה הוא מושג קשה מאד ללמידה ולהוראה, ויש הטוענים כי אחת הסיבות לקושי קשורה בהעדרן של אנלוגיות יומיומיות למושג רקורסיה. במחקרים אף נמצא כי התלמידים נוטים לפתח מודלים מנטליים לא מתאימים של רקורסיה. מורים עושים שימוש במודלים קונספטואליים, כמו המודל של הבובות הרוסיות, המודל של מעקב אחר התהליך הרקורסיבי והמודל של אינדוקציה מתמטית. זאת מתוך תקווה שהמודלים הללו יסייעו ללמידת רקורסיה ויעזרו ללומדים לפתח מודל מנטלי מתאים יותר של המושג. בנוסף למודלים הקונספטואליים, גם הכישורים הקוגניטיביים האישיים של התלמידים משחקים תפקיד נכבד בתהליך יצירתם של מודלים מנטליים של רקורסיה. מאמר זה מנסה לענות על השאלה אלה מודלים קונספטואליים עוזרים לתלמידים מתחילים ללמוד רקורסיה, וביתר פירוט – לבחון אם מודלים קונספטואליים מוחשיים (קונקרטיים) עוזרים יותר או פחות ממודלים קונספטואליים מופשטים (אבסטרקטיים), ואם תלמידים עם סגנון קוגניטיבי מופשט לומדים רקורסיה אחרת מתלמידים עם סגנון קוגניטיבי מוחשי. כמו כן נבחן הקשר בין שימוש בסוג מסוים של מודל (מוחשי או מופשט) לסגנון הקוגניטיבי (מוחשי או מופשט).

מודלים קונספטואליים של רקורסיה

מודל קונספטואלי משמש כ"תחום בסיס" שבאמצעותו ניתן להסיק מסקנות על "תחום מטרה". מודל קונספטואלי מופשט מתבסס על תחום בסיס מופשט, כמו במקרה של מודלים מתמטיים. מודל קונספטואלי מוחשי משתמש בעצמים מוחשיים. בתחום של הוראת תכנות, יש הממליצים על שימוש במודלים מוחשיים להתייחסות לתהליכים המתרחשים ב"קופסה השחורה" של המחשב, ולעומתם אחרים טוענים כי עדיף ללמד תכנות לתלמידים מתחילים ללא התייחסות לאופן המוחשי שבו פועלת המכונה - המחשב.

לצורך הוראת רקורסיה, נהוג להשתמש בסוגים שונים של מודלים קונספטואליים. נבחן להלן חמישה מהמודלים המקובלים ביותר, כאשר השלושה הראשונים נוטים לקצה המוחשי יותר של הרצף, והשניים האחרונים נוטים לקצה המופשט שלו.

1. **מודל הבובות הרוסיות**. בובה רוסית מתפרקת להרבה בובות הולכות וקטנות בעלות אותה צורה בדיוק. המודל הזה מציג את התהליך שבו מקרה מפעיל מקרה פשוט יותר של עצמו (זהו החלק הרקורסיבי), עד שלבסוף התהליך הרקורסיבי נפסק עם הבובה הקטנה ביותר שאינה מכילה בובה אחרת וגם נראית שונה מהאחרות (זהו החלק של המקרה הבסיסי, הפשוט ביותר).
2. **מודל המעקב**. מודל זה מתמקד במעקב אחר התהליך הנוצר על ידי פונקציות רקורסיביות, כלומר במעקב אחר איך פועלות הפונקציות הללו [בדומה לכל הגישות שתוארו במאמר הקודם, ד"ל]. זהו מודל מוחשי בעליל, אולם דרגת המוחשיות שלו משתנה בהתאם לשיטת המעקב בה משתמשים.
3. **מודל סימולציה של מחסנית**. רקורסיה מוצגת כאן במונחים של ארכיטקטורת המחשב בה משתמשים לביצוען של תכניות רקורסיביות. המעקב אחר הקריאות לפונקציות או לפרוצדורות מתבצע תוך שימוש במנגנון המחסנית שבו משתמשות גם שפות התכנות.
4. **מודל האינדוקציה המתמטית**. רקורסיה מוצגת במונחים של היסודות המתמטיים המשמשים להוכחת נכונותה, כלומר במונחים של הוכחה באמצעות אינדוקציה.

5. **מודל התבניות.** מודל זה מספק למתכנתים מתחילים דוגמאות של תכניות רקורסיביות, תוך תיאור מקרי הבסיס והמקרים הרקורסיביים. במודל זה, פתרון של בעייה רקורסיבית מתבצע באמצעות השלמת החלקים החסרים בתבנית הכללית הנתונה.

במחקרים קודמים נמצא אמנם כי מודלים מוחשיים (כדוגמת 1-3) מקדמים את הלמידה, אך ממצא זה לא נבחן לעומק מבחינת הידע המושגי המורכב הכרוך בהבנת קטעי תכניות גדולים כמו לולאות או שימוש במבני נתונים. המחקר הנוכחי בדק איזו השפעה יש לשימוש בשני הסוגים של המודלים הקונספטואליים על ההבנה בתחום המורכב של רקורסיה.

סגנונות קוגניטיביים של למידה

המחקר המתואר כאן מתבסס על התיאוריה של קולב בדבר הממדים של סגנון הלמידה האישי (Kolb, 1985). בתיאוריה זו מוגדרים שני ממדים הקשורים לאופנים בהם אנשים לומדים. המימד האחד הוא הרצף שבין מוחשי למופשט, והוא מתאר את האופן שבו אנשים תופסים מידע חדש. במצבים לא מוכרים, יש כאלה שמעדיפים לחוש את סביבתם (התנסות מוחשית) ולעומתם אחרים מעדיפים לתכנן את דרכם באופן מחושב מראש (המשגה מופשטת). המימד השני מופיע כרצף שבין פעילות (אקטיביות) להתבוננות (רפלקטיביות), והוא נוגע לאופן שבו אנו מעבדים את המידע.

על פי תיאוריה זו, צפוי שתלמידים עם סגנונות למידה שונים יגיבו באופן שונה לשיטות הוראה אלה או אחרות. כך גם מומלץ להתאים את שיטת ההוראה לסגנון הלמידה, שכן תלמידים מסוימים בודאי תופסים מושגים מופשטים באופן אחר מחבריהם, אשר זקוקים לסיוע של דימויים מוחשיים לצורך הלמידה. יש גם עדויות לכך שלומדים בעלי סגנון קוגניטיבי מופשט מרוויחים יותר ממודל מופשט, ונרתעים ממודלים מוחשיים.

ממצאי המחקר

המחקר התבצע בקרב 237 סטודנטים למדעי המחשב באוניברסיטה, בקורס מבוא למדעי המחשב. סגנונות הלמידה של הסטודנטים הללו אובחנו ע"פ מבחנים הנובעים מהתיאוריה של קולב, והם חולקו באופן אקראי לשתי קבוצות – באחת התבצעה ההיכרות עם רקורסיה באמצעות מודלים מוחשיים והשניה באמצעות מודלים מופשטים.

הממצא הראשון נוגע לשאלה האם מודלים מוחשיים תורמים ללמידת רקורסיה יותר ממודלים מופשטים, ונמצא שבעבור הסטודנטים המתחילים שנבדקו התשובה היא חיובית, אולם ההשפעה איננה לאורך זמן. כלומר, במבחן נוסף שנערך מספר שבועות מאוחר יותר, לא בלטה תרומה זו.

ממצא נוסף קשור בשאלה האם סטודנטים עם סגנון קוגניטיבי מופשט מצליחים יותר בלמידת רקורסיה מסטודנטים עם סגנון קוגניטיבי מוחשי, וגם כאן התשובה היתה חיובית. ממצא זה עקבי עם תוצאות מחקרים אחרים מהם עולה כי בעלי סגנון קוגניטיבי מופשט או אנליטי נוטים לתפקד טוב יותר בלמידת תכנות. הממצא הזה תואם גם השלכות מהתיאוריה של קולב הנוגעות לקשר בין סגנון קוגניטיבי למקצוע, דהיינו להתאמה הקיימת בין הסגנון הקוגניטיבי לבין הדרישות המקצועיות. אנשים בתחום מדעי המחשב נוטים יותר לסגנון קוגניטיבי מופשט. למידת מדעי המחשב מחייבת שימוש בלוגיקה ובסמלים, הבנה של מושגים מופשטים, פיתוח תיאוריות ומודלים, וניתוח בעיות ומערכות. בעלי סגנון קוגניטיבי מופשט חשים יותר בנוח במצבי למידה מסוג זה. זו אולי הסיבה לכך שהלומדים בעלי הסגנון המופשט תפקדו טוב יותר מאלה בעלי הסגנון המוחשי בלמידת רקורסיה.

הממצא האחרון עוסק בקשר שבין הסגנון הקוגניטיבי של הסטודנטים לבין אופי המודל שהוצג בפניהם. במחקר הנוכחי לא נמצא כל קשר, דהיינו – בעלי הסגנון המופשט לא בהכרח הרוויחו יותר משימוש במודלים מופשטים, ובעלי הסגנון המוחשי לא בהכרח הרוויחו משימוש במודל המוחשי.

מסקנות

חשיבות השימוש במודלים קונספטואליים בהוראת מושגים מופשטים ידועה לכל, וממחקר זה עולה אף תמיכה לשימוש במודלים מוחשיים בהוראת רקורסיה למתכנתים מתחילים. אולם השאלה איזה סוג של מודל עוזר יותר למתכנתים ברמת הביניים נשארה עדיין פתוחה. על המורים להיות זהירים בתכנון מודלים מוחשיים, שכן אלה צריכים להמחיש ברמת מתאימה של פירוט את המכניזם הפנימי של רקורסיה.

מהמחקר הנוכחי עולה עוד כי תלמידים עם סגנון קוגניטיבי מופשט נוטים לתפקד באופן טוב יותר מאחרים בלמידת תכנות. במלים אחרות, בעלי סגנון קוגניטיבי מוחשי עשויים להתקל ביותר קשיים בעת למידת רקורסיה. חשוב שהמורים יזהו תלמידים כאלה ויספקו להם תמיכה. המודעות לסגנונות הלמידה של תלמידיהם תעזור למורים בהתאמת שיטות ההוראה לכיתה.

ולבסוף, הטענה כי מומלץ לקשר בין סגנון הלמידה לסוג המודל נראית באופן תיאורטי נכונה, אולם ממצאי המחקר לא תומכים בטענה כזו. יש צורך במחקרים נוספים בנושא, שיתמקדו לא רק במימד המוחשי-מופשט בו התמקדנו אנו, אלא גם במימד של פעילות-התבוננות הקשור בהיבטים של מעורבות אקטיבית בלמידה. לדוגמה, לומדים מתבוננים מעדיפים צפייה והקשבה, כך שהוראה פרונטלית עשויה להתאים להם יותר מללומדים פעילים המעדיפים התנסות ומעורבות פעילה בעשייה ובדיונים. דרוש מחקר נוסף על מנת לבחון קשרים בין מימד זה של סגנונות הלמידה של התלמידים לבין שיטות ההוראה של המושג רקורסיה.

מקורות

Kolb, D.A. (1985). Learning style inventory. McBer and Company, Boston, MA.

מה מתכנתים 'טירונים' יודעים אודות רקורסיה?

ה. קהני

Kahney, H. (1989). What do novice programmers know about recursion? Soloway, E. & Sphorer, J.C. (eds), *Studying the novice programmer*, LEA publishers. 209-228.

אודות המאמר

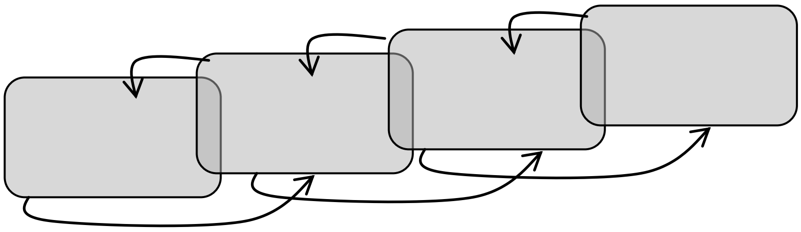
מתכנתים מתחילים וחסרי נסיון יוצרים, ככל הנראה, מודלים מנטליים להסבר של תהליכים רקורסיביים, והמודלים הללו שונים מאלו שבהם משתמשים מתכנתים מומחים. המאמר מתבסס על הנחה זו, ואף מציג השערה מפורטת יותר בדבר האופי השונה של המודלים שיש לטירונים לעומת מומחים. לצורך בדיקת ההשערה, ערך קהני מחקר שבו התבקשו הן טירונים והן מומחים להשיב לשאלון שכלל מספר פתרונות אפשריים לבעייה תכנותית. ניתוח התשובות לשאלון וההסברים המפורטים שהנבדקים רשמו בו מאושש הן את ההנחה הכללית והן את ההשערה כי טירונים נוטים להשתמש במודל הלולאה, בעוד מומחים מסתייעים במודל ההעתקים להסבר של תהליכים רקורסיביים. בנוסף, מאפיין המאמר מספר מודלים נוספים בהם מחזיקים הטירונים.

<u>רשימת מושגים</u>

- מודלים אישיים - Idiosyncratic models
- מודל ההעתקים – Copies model
- מודל הלולאה – Loop model
- מתכנת טירון – Novice programmer
- מתכנת מומחה – Expert programmer

הקדמה

המשימה בה דן מאמר זה תוכננה כדי לבחון את ההשערה כי המודל שיש למתכנתים טירונים אודות התהליך הרקורסיבי שונה מזה שיש למתכנתים מומחים, וכדי לזהות מהם המודלים בהם מחזיקים המתכנתים הטירונים. לשם כך, הוצג בפני תלמידים מודל קונספטואלי המגדיר רקורסיה כתהליך שביכולתו להפעיל עותקים חדשים של עצמו, ואשר מסוגל להעביר את הבקרה אל ומהעותק המופעל. זהו המודל המשוער של המומחים, והוא מכונה **מודל העתקים** (ראה תרשים).



ההשערה היתה כי התלמידים המתחילים מחזיקים במודל אחר של רקורסיה, הוא **מודל הלולאה**. כלומר, הם רואים את הפרוצדורה הרקורסיבית כעצם בודד ולא כסידרה של עותקים חדשים. התכונות המשוערות של העצם הבודד הזה הן:

1. קיימת **נקודת כניסה** המיוצגת באמצעות שם הפרוצדורה ורשימת הפרמטרים שלה.
2. קיים **חלק ביצועי**.
3. קיים **מנגנון** כלשהו להעברת ערכים בחזרה לחלק הקדמי, או לנקודת הכניסה של הפרוצדורה.

כיצד 'תתנהג' תכנית רקורסיבית בהתאם למודלים השונים

ההשערה אודות הבדלים בין מתכנתים טירונים למומחים, נבדקה באמצעות שאלון אחיד. בשאלון הופיעו שלוש תכניות הפותרות את אותה בעייה, כאשר שתיים מהתכניות מאפשרות לאבחן את המודל בו מחזיק המתכנת. נדגים להלן את תהליכי ההיסק הלוגיים המתרחשים, על פי ההנחה, כאשר תלמידים עם מודלים שונים של רקורסיה בוחנים את שתי התכניות המאבחנות הללו.

הבעייה שהוצגה בפני המתכנתים היתה:

"יש לכתוב תכנית שתבנה בסיס נתונים מתאים לתנאים הבאים: אם ל – X יש שפעת, אז כל מי ש-X מנשק אותו נעשה אף הוא חולה בשפעת, וכל מי שנדבק מפיץ את השפעת לאדם שהוא או היא מנשקים, וכך הלאה. "

הפתרונות המוצעים היו הבאים [כאן מופיע תרגום למעין קוד בעברית. במקור מופיע פסיאודו-קוד באנגלית, ד"ל]:

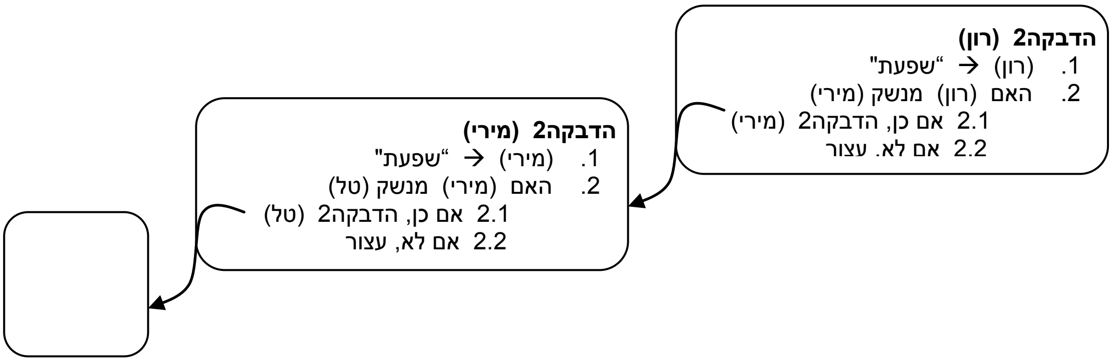
הדבקה1 (X)
1. האם (X) מנשק (מישהו)
   1.1 אם כן, (מישהו) → "שפעת"
   1.2 אם לא, עצור

הדבקה2 (X)
1. (X) → "שפעת"
2. האם (X) מנשק (מישהו)
   2.1 אם כן, הדבקה2 (מישהו)
   2.2 אם לא, עצור

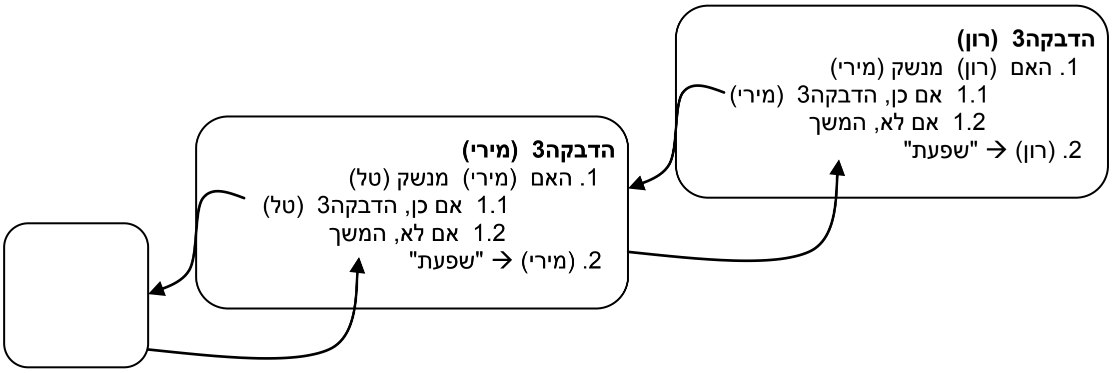
הדבקה3 (X)
1. האם (X) מנשק (מישהו)
   1.1 אם כן, הדבקה3 (מישהו)
   1.2 אם לא, המשך
2. (X) → "שפעת"

הפתרון הראשון לא מתאים, שכן פרוצדורה כזו תעצר כבר לאחר הבדיקה הראשונה. שני הפתרונות האחרים עובדים, כאשר הפתרון השלישי יוצר מחסנית שבעת הפריקה שלה מתבצעות ההשמות הדרושות.

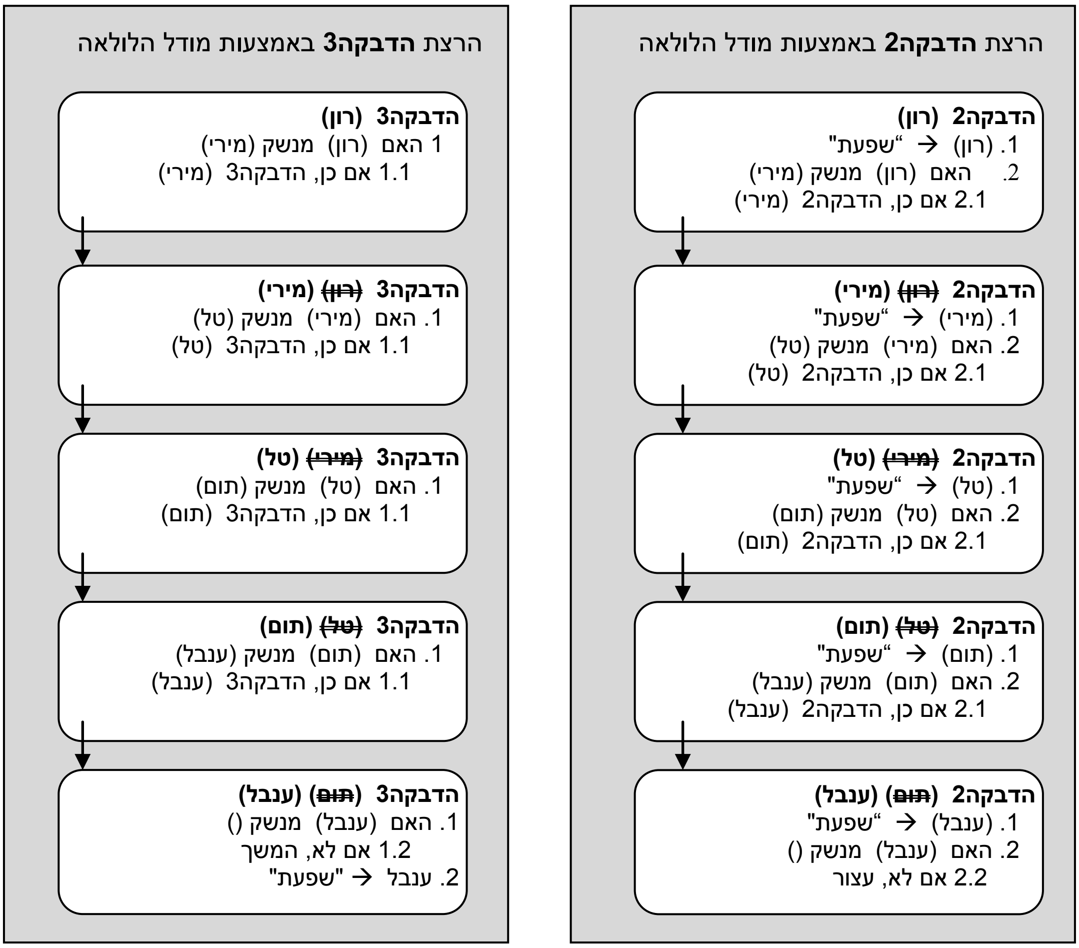
כיצד תראה הרצת התכניות הללו (הדבקה2 והדבקה3) תוך שימוש במודל ההעתקים? התרשים הבא מציג את תהליך ההיסק המשוער עבור התכנית **הדבקה2** אצל מי שמחזיק במודל כזה. נניח לדוגמה שידוע כי לרון יש שפעת והוא מנשק את מירי, ואילו מירי מנשקת את טל. בתרשים מופיעים שני העותקים הראשונים.



זהו תהליך ההיסק המשוער עבור התכנית **הדבקה3**, תוך שימוש במודל ההעתקים:



נבחן עתה כיצד תראה הרצת התכניות הללו תוך שימוש **במודל הלולאה**. לשם כך נגדיל את בסיס הנתונים, ונניח שרון מנשק את מירי שמנשקת את טל, וטל מנשק את תום שמנשק את ענבל, שאיננה מנשקת אף אחד אחר. ההרצה המפורטת תחת מודל הלולאה נראית ככל הנראה כמו בתרשים הבא, כשבצידו הימני ההרצה של תכנית הדבקה2 ובצידו השמאלי ההרצה של תכנית הדבקה3. במונחים של מודל הלולאה, הקריאה הרקורסיבית מתפרשת כחזרה לתחילת התכנית, כאשר ליד שם הפרוצדורה נמחק הערך הקודם של הפרמטר ומופיע הערך החדש. שימו לב לכך שעל פי מודל הלולאה, לא חוזרים יותר לפרוצדורה "הקודמת" לביצוע ההוראה הממוספרת 2.2, ולכן במקרה של הדבקה3 למעשה מתקבלת תוצאה לא נכונה – רק ענבל חולה בשפעת! לכן צפוי שמי שמחזיק במודל הלולאה, יאמר בעקבות נסיון ההרצה כי רק תכנית **הדבקה2** מתאימה לנתוני הבעיה. לעומת זאת, צפוי כי המחזיקים במודל ההעתקים של רקורסיה יאמרו כי הן **הדבקה2** והן **הדבקה**3 משיגות את המטרה. משום כך נחשבות זוג התכניות הללו לתכניות מאבחנות.



לסיכום, עדות לכך שמישהו מחזיק במודל הלולאה, יהיה קבלה של הפתרון הדבקה2 כנכון ודחייה של הפתרון הדבקה3. חיזוק לעדות זו יתקבל אם התלמיד יסביר גם כי מהרצת התכנית השלישית עם הנתונים הנ"ל יתקבל שרק ענבל חולה בשפעת.

ממצאי המחקר

המחקר המתואר במאמר התבסס על ההנחה שנוסחה לעיל. במחקר השתתפו סטודנטים מתחילים ללא כל רקע תכנותי (שלושים מהם השיבו את השאלון), ועוד תשעה מתכנתים מומחים. כולם קיבלו את השאלון עם הבעייה ושלושה הפתרונות, התבקשו לציין מי מהפתרונות מתאים לבעייה, וכמו כן להסביר במלים שלהם כיצד מתבצע כל אחד מהפתרונות.

מבין המומחים, שמונה טענו שגם הפתרון השני וגם הפתרון השלישי יעבדו כהלכה. התשיעי טען שרק הפתרון השני מתאים. תגובה טיפוסית למתכנת מומחה היא התגובה הבאה, לגבי הפתרון השני:

> "כן, התכנית תעבוד... זה נעשה על ידי כך שבשלב הראשון מציינים עבור כל פרמטר אקטואלי ש'יש לו שפעת'. אחר כך בודקים אם יש קשר נשיקה בין הפרמטר הנוכחי לבין ערך חדש ואם כן הפרוצדורה מופעלת באופן רקורסיבי עם הפרמטר האקטואלי החדש. החיפוש הזה לעומק ממשיך (כמו הדבקה אמיתית – למרות שבמציאות מעורבת איטרציה בעניין וכאן לא) עד שאין יותר זוגות מתנשקים ואז מתבצע מסלול של יציאה (כל הדרך חזרה בין הרמות של הרקורסיה עד למעלה)".

זו תגובתו של אותו מומחה לפתרון השלישי:

"גם הפרוצדורה הזו תבצע את הדרוש. היא עושה זאת בעצם באותו אופן שעבד הפתרון השני, רק שכאן הציון כי למישהו יש שפעת מתבצע רק לאחר שנפסק החיפוש לעומק אחר זוגות מתנשקים....".

מתוך הסטודנטים המתחילים, ארבעה טענו כי אף אחת מהתכניות לא מתאימה לבעייה; אחד חשב שרק הפתרון הראשון יעבוד, ועוד אחד חשב שגם הפתרון הראשון וגם הפתרון השני יעבדו; שישה-עשר מהסטודנטים טענו שרק הפתרון השני מתאים; חמישה טענו שרק הפתרון השלישי יתאים; ועוד שלושה חשבו שגם הפתרון השני וגם השלישי יעבדו כהלכה. הטבלה הבאה מציגה את התפלגות התשובות, הן אצל המתחילים והן אצל המומחים.

התגובה	**טירונים**	**מומחים**
רק ההליך **הדבקה2** מתאים	16	1
שני ההליכים **הדבקה2** ו- **הדבקה3** מתאימים	3	8
תשובה אחרת	11	0

דיון בממצאים

מתגובות הסטודנטים המתחילים ניתן להסיק כי כמחצית מהם אימצו את מודל הלולאה (בטענתם כי רק הפתרון השני מתאים), וכי רק עשירית מהם מחזיקים במודל ההעתקים. לגבי חמישית מהסטודנטים נמצא כי הם לא הבינו כמעט דבר מהתבוננות באף אחת מהתכניות. לגבי עוד כחמישית מהסטודנטים, יש עדות חלשה לשימוש במודל ההעתקים – אלה ציינו את התכנית השלישית כתכנית המתאימה היחידה. כדי לקבוע ביתר דיוק כיצד חשבו הסטודנטים שהתכניות מתנהגות, נתבונן בנימוקים שהם ציינו.

נפתח בתגובותיהם של המעטים שציינו כי גם הפתרון השני וגם השלישי נכונים, כלומר אלה שיתכן ומחזיקים במודל ההעתקים. אחד מהם, בהסבר לפתרון השני, הסביר כי "התכנית תפתור את הבעייה, אבל היא תנסה להדביק את כולם פעמיים, חוץ מאשר את ענבל, ולכן צעד מספר 2 מיותר". צעד זה הוא הצעד הרקורסיבי, וברור שאינו מיותר, כך שנדמה כי הסטודנט הזה לא הבין את התכנית השניה כהלכה. גם מדברי הסטודנט השני שציין כי שתי התכניות מתאימות ניתן להתרשם כי הוא מציג ניחוש לא מנומק. ואילו הסטודנט השלישי, כתב כך על הפתרון השני: "מדביק את הראשון, מוצא קשר של נשיקה, מדביק את השני וכך ממשיך בשרשרת ההדבקה". על הפתרון השלישי הוא כותב: "מדביק את הראשון וממשיך כמו קודם. ההבדל הוא רק בסדר ביצוע ההוראות". רק הסטודנט הזה מכל השלושה הביא נימוקים משכנעים מספיק כדי לשער שהוא מחזיק בואריאציה מסוימת של מודל ההעתקים ולא במודל הלולאה. כלומר, נמצא כי רק אצל אחד משלושים סטודנטים יש עדות כלשהי להתפתחות של הבנת רקורסיה ברמת מומחיות גבוהה יותר. מן ההסברים שכתבו הסטודנטים עולה גם כי לחלקם הבנות אישיות וייחודיות של פרוצדורות רקורסיביות, וחלקם אף מתקשים בזיהוין של פרוצדורות כאלה.

נעבור לתגובותיהם של אלה שעבורם יש עדות כי הם מחזיקים במודל הלולאה, ולכן בחרו בפתרון השני כפתרון המתאים היחיד. תגובה אופיינית נראית כך: "צריך לעבור שלושה שלבים. א) למישהו יש שפעת, ב) בדיקה את מי הוא מנשק, ג) הדבקה באמצעות נשיקה". הסברים מפורטים יותר הופיעו רק אצל מעטים, והתגובה הבאה היא היחידה שמציינת במפורש את השימוש במנגנון של לולאה: "כאשר מבצעים הדבקה של רון רושמים שיש לו שפעת, בודקים את מי הוא מנשק, ואז חוזרים בחזרה [loops back במקור – ד"ל] להדביק את האיש הזה... ממשיכים בלולאה עד שכולם נדבקים".

מתוך שישה עשר הסטודנטים שבחרו בפתרון השני כמתאים, ארבעה דחו את הפתרון השלישי בנימוק שעל פי פתרון זה "רק ענבל תחלה בשפעת". זוהי העדות החזקה ביותר לשימוש במודל הלולאה, ובמחקרנו נמצא כי כ- 13% מהסטודנטים המתחילים מחזיקים בו. כך למשל כתב אחד הסטודנטים הללו על הפתרון השלישי: "לאף אחד אין שפעת בהתחלה. זה פשוט יבדוק מי מנשק את מי, אבל השפעת לא מועברת הלאה. זה מסתובב סביב 1 ו- 1.1, ו X מקבל שפעת רק אם הוא לא מנשק אף אחד". סטודנט אחר הסביר "זה לא יפתור את הבעייה. התכנית תסתובב בלולאה באופן דומה לתכנית השניה, אבל תציין בסוף רק שלענבל יש שפעת".

לסיכום, ברור שהסטודנטים שהשיבו לשאלון לא החזיקו במודל ההעתקים, אבל גם השימוש במודל הלולאה זכה רק לעדויות נחרצות בודדות. במלים אחרות, שני המודלים הללו הופיעו רק אצל שישית מהסטודנטים, ומעניין לאבחן את המודלים האישיים של כל השאר. חלק מהשאר, התברר, לא הבינו רקורסיה בכלל ולא השתמשו באף מודל מנטלי. סטודנטים אלה טענו שאף פתרון לא מתאים או שהפרוצדורות לא חוקיות. במאמר מכנים זאת "שימוש במודל ריק" [במקור מופיע המונח null model. אולם מן הראוי לציין כי על פי הגישה הקונסטרוקטיביסטית ללמידה, לכל הלומדים יש מודל מנטלי כלשהו, גם אם לא תמיד השימוש בו מביא לתהליך ההיסק המצופה – ד"ל]. אוסף אחר של מודלים אישיים מכונה במאמר "מודל מוזר" (odd model). מפירוש תגובות הסטודנטים שסווגו כמחזיקים ב"מודל מוזר" עולה שלסטודנטים אלה יש מודל כלשהו של התנהגות התכניות שהופיעו בשאלון, אבל משום שיש להם רעיונות ייחודיים הנוגעים לחלק ממרכיבי התכנית, הם לא יכלו לצפות את התנהגות התכניות באופן מדויק. אחד מהעניינים הבולטים היה קשור בהסבר של מה שמתבצע כאשר תנאי העצירה מתקיים, ובעניין זה היו לסטודנטים בעלי המודל "המוזר" הסברים שונים (ולא מדויקים). המודל האחרון שזוהה במחקר מכונה "המודל התחבירי" או "מודל המסתורין". זהו מודל תחבירי במובן זה שהסטודנטים מזהים תכנית רקורסיבית על פי המבנה של התכנית, וכל מבנה דומה לזה המופיע בתכנית השניה (ושהוצג לסטודנטים אלה גם בשיעורים קודמים) נתפס על ידם כתבנית רקורסיבית. המודל התחבירי הוא מסתורי במובן זה שלמרות הסטודנט יודע מה התכנית עושה, אין לו מושג כיצד היא מגיעה לתוצאות אליהן היא מגיעה. הערות רבות שהופיעו בתגובות הסטודנטים לשאלון מרמזות על שימוש במודל התחבירי, כאשר הסטודנטים מוכיחים רגישות למצב היחסי של קטעי התכנית ולסדר שבו הם רשומים.

מסקנות

ממצאי המחקר מדגימים קשת רחבה של אפשרויות. ניתן לסווג את המתכנתים הטירונים בהתאם למבנים הפרטיים בהם בונים את מערכות המושגים שלהם. הנתונים מראים כי לפחות חלק מהטירונים מסוגלים, לאחר תקופת אימון תכנותית קצרה, לזהות ולחזות את התנהגותן של תכניות רקורסיביות. במלים אחרות, לסטודנטים האלה יש מודל מנטלי בדבר האופן שבו תכנית רקורסיבית פועלת. עבור משימות מסוימות, המודלים האישיים הללו לא יתנו סיוע מספיק. אבל במשימות רבות אחרות אפילו טירונים יכולים לתפקד היטב. אם למישהו יש מודל מנטלי של תהליך, אפילו אם מודל זה שונה במקצת מהמודל הקונספטואלי של התהליך, אותו לומד יוכל לחזות כיצד יתבצע התהליך ומה יהיו תוצאותיו. גם אם התחזית לא מדויקת, נודעת חשיבות ליכולת חיזוי זו. במלים אחרות, תלמידים המפתחים את מודל הלולאה להבנת רקורסיה יוכלו לתכנן פרוצדורות תוך שימוש במודל המנטלי שלהם ויוכלו לנסות להסביר תכניות לא מוכרות באמצעות מודל זה. אם יתקלו התלמידים בתוצאות לא חזויות או בסתירות למה שחשבו מראש, הדבר יעמיד אותם בפני הצורך לשפר את המודל המנטלי שלהם ולהסביר לעצמם היכן הסתירות. כך גם יוכלו, אולי, להתפתח לכיוון של שימוש במודלים מנטליים אחרים, 'מקצועיים' יותר.

כיצד לומדי תכנות פונקציונלי מפרשים פונקציות רקורסיביות

י. סגל

Segal, J. (1995). Empirical studies of functional programming learners evaluating recursive functions. *Instructional Science 22*, 385-411.

אודות המאמר

המאמר עוסק בלומדים המנסים להבין ולהעריך פונקציות רקורסיביות נתונות הכתובות בשפת תכנות פונקציונלית. המחקר המתואר במאמר זה התבסס על תצפיות בלומדים המבצעים משימות תכנותיות ובראיונות עם לומדים אלה, וממצאיו עולים שני קשיים עיקריים בהם נתקלו הלומדים. הקושי הראשון קשור בתפיסה מוטעית של הלומדים אודות תהליכים רקורסיביים, והקושי השני נובע מערבוב שהם עושים בין ההקשר המתמטי של המושג "משתנים" לבין ההקשר התכנותי שלו. המאמר מסכם בהדגשת ההשפעה שיש לאסטרטגיה בה משתמשים הלומדים כדי לחשב פונקציה רקורסיבית על הצלחתם לנבא את ערך הפונקציה, וממליץ מפורשות ללמד את האסטרטגיה של חישוב מושהה.

<u>**רשימת מושגים**</u>

- אסטרטגית חישוב מושהה – Delayed evaluation
- אסטרטגית חישוב מיידי – Immediate evaluation
- חישוב מושהה מלמעלה למטה - Top-down delayed evaluation
- חישוב מושהה מלמטה למעלה - Bottom-up delayed evaluation
- יחס של התרחשות חוזרת (יחס רקורסיבי) - Recurrence relation
- שפת תכנות פונקציונלית – Functional programming language
- תפיסות שגויות – Misconceptions

הקדמה

המוקד העיקרי במאמר זה הוא הקשיים בהם נתקלים התלמידים המנסים להבין פונקציות רקורסיביות הכתובות בשפת תכנות פונקציונלית (מירנדה – Miranda). הקשיים הללו זוהו בעקבות מחקר שדה אמפירי שכלל תצפיות בלומדים המנסים לבצע משימות תכנותיות וראיונות עמם. בעקבות המחקר ניתן לדבר גם על הסיבות לקשיים שזוהו, ולהמליץ על אסטרטגיה לבחינת פונקציות רקורסיביות. המאמר נפתח בדיון בשאלות למה מעניין לעסוק בתכנות פונקציונלי ומדוע מעניין לעסוק ברקורסיה בתכנות פונקציונלי. כמו כן מוצגות גישות שונות לרקורסיה ונבחנת שאלת ההערכה של תכניות. בחלק העיקרי של המאמר מתוארות התצפיות על קשיי התלמידים, ומוצגים בהרחבה שני קשיים: הראשון קשור בתפיסה שגויה הנוגעת לתהליך הרקורסיבי והשני קשור בבלבול בין משתנים מתמטיים למשתנים תכנותיים. לבסוף נידון תפקידן של האסטרטגיות בהם משתמשים הלומדים להערכת פונקציות רקורסיביות ומוצגות האסטרטגיות היותר מוצלחות מביניהן.

הרקע למחקר

הפרדיגמה של תכנות פונקציונלי צמחה בשנים האחרונות להיות אחת מהפרדיגמות החשובות בתכנות. מקובל לטעון כי לתכניות שמפותחות תוך שימוש בפרדיגמה הפונקציונלית כוח ביטוי מתמטי עשיר, וכי ניתן להתאימן בפשטות יחסית לתכנות מקבילי. בעקבות טיעונים אלה שולבה ההתנסות בתכנות פונקציונלי בקורסים רבים של מדעי המחשב, אולם השאלות עד כמה פשוט ללמוד, ללמד ולהשתמש בגישות של תכנות פונקציונלי נותרו עד כה ללא תשובה מחקרית. יחד עם זאת, מקובל להניח שאחד מהקשיים בלמידת שפת תכנות פונקציונלי הוא רקורסיה.

שפת התכנות "מירנדה", שהיא שפה פונקציונלית דקלרטיבית, פותחה בעשור האחרון כאחד הצאצאים של שפת LISP. בשפה דקלרטיבית הדגש מושם על הבעיה התכנותית, ולא על הדרך לפתור אותה. המתכנת צריך להשכיל לתרגם את הבעיה לצורה הנדרשת על ידי השפה, בעוד המימוש המסוים של הפתרון לא אמור כלל לעניין אותו. תכונה זו עומדת בסתירה לשפות פרוצדורליות שבהן המתכנת צריך לתאר איך לפתור את הבעייה תוך התחשבות באילוצים ששפת התכנות מטילה עליו. שפות פונקציונליות כמו "מירנדה" נעזרות בפונקציות כדי להמיר קלט נתון בפלט רצוי, והמשימה של המתכנת היא להגדיר את הפונקציות הדרושות. קיימות פונקציות ספריה שהמתכנת יכול להרכיבן יחדיו כדי ליצור פונקציות חדשות, או לבחור מביניהן באמצעות משפטי תנאי, או לחזור באופן רקורסיבי על ביצוען. בשפות כאלה, רקורסיה היא מבנה החזרה היחיד, ולפיכך רקורסיה היא מרכיב חיוני של הידע הדרוש אפילו למתכנתים מתחילים.

אפשר לראות רקורסיה ככלי שמאפשר להגדיר פונקציות הפועלות על תחום אינסופי של משתנים בצורה תמציתית, "מכווצת". פונקציות רקורסיביות מתוארות במונחים של תנאי עצירה מצד אחד, ושל יחס התרחשות חוזרת (recurrence relation) מצד שני. למשל, אם התחום של הפונקציה f הוא קבוצת המספרים הטבעיים שיסומנו ב- n, תנאי העצירה יכול להיות הערך של הפונקציה כאשר n=0, כלומר הערך של f(0). יחס ההתרחשות החוזרת במקרה זה יבטא את האופן שבו הפונקציה f(n) מתייחסת ל- f(p), כאשר p מספר טבעי קטן מ- n. דוגמה נפוצה היא הדוגמה של פונקצית העצרת fact. הייצוג המתמטי הרקורסיבי של פונקציה זו נתון כך: fact(0) = 1 - זהו תנאי העצירה. יחס ההתרחשות החוזרת הוא fact(n)=n*fact(n-1) כאשר n>0. הייצוג הרקורסיבי של פונקצית העצרת בשפת תכנות פונקציונלית כמו מירנדה לא שונה בהרבה:

```
Fact n  = 1                  if n = 0
        = n * fact (n-1)     if n > 0
```

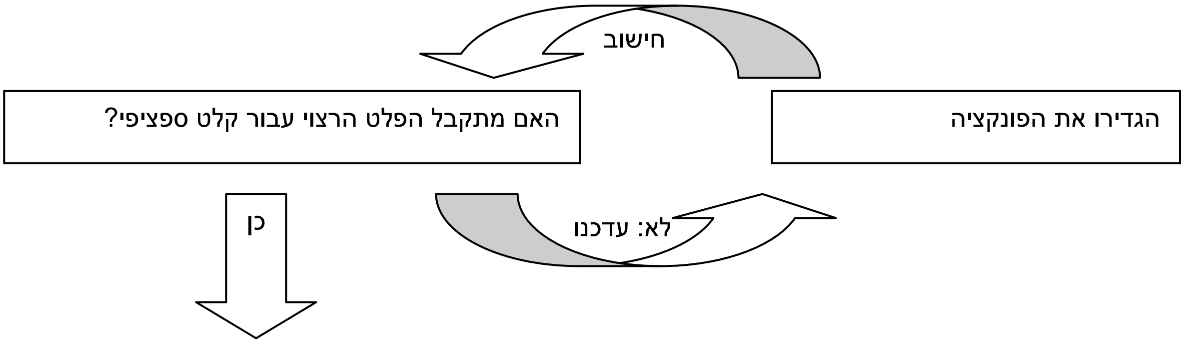
בהינתן ייצוג דקלרטיבי כזה של פונקציה רקורסיבית, יש צורך להשתמש בהליך כלשהו להערכת הפונקציה, כלומר לחישוב הערך שהיא תייצר עבור קלט נתון כלשהו. הליך ההערכה יכיל מספר פעולות מנטליות, במקרה שהוא מתבצע באופן ידני ולא על ידי המחשב. למשל, במקרה של פונקצית העצרת עשוי הליך ההערכה של הפונקציה הנתונה לעיל להכיל מספר פעולות של הצבה (או שכתוב) [ראה דוגמה לחישוב עצרת של המספר 2, ד"ל].

Fact 2 = 2 * fact (1)
= 2 * 1 * fact (0)
= 2 * 1 * 1
= 2 * 1
= 2

במאמר של קהני [Kahney 1989, המופיע אף הוא בסקר ספרות זה – ד"ל], מוצגת גישה אחרת לתיאור רקורסיה, תוך שימת דגש על תהליך הביצוע של הפונקציה הרקורסיבית ולא על רקורסיה ככלי להגדרת פונקציות. במונחים אלה של תהליך הביצוע, אם נניח שהמחשב "יודע" את ההגדרה של פונקצית העצרת ועליו לבצע חישוב של fact 2, אזי מה שיתרחש הוא ש fact 2 תזמן את fact 1 אשר תזמן את fact 0 – כל אלה זימונים של העתקים חדשים של הפונקציה המקורית fact. הזימון האחרון יחזיר תוצאה ל – fact 1 ולאחר מכן תוחזר תוצאה לזימון המקורי של fact 2. גישה נוספת לרקורסיה, היא כאל טכניקה לפתרון בעיות. על פי גישה זו, פתרון בעית העצרת מתקבל מפירוק הבעייה המקורית (של חישוב fact n) לתת-בעייה פשוטה יותר מאותו סוג. המקרה הבסיסי נחשב לתת-הבעייה הפשוטה ביותר שאותה ניתן לפתור ישירות, בלי צורך להזדקק לפירוק נוסף.

למרות שיש קשר ברור וידוע בין אופן הייצוג של פונקציה רקורסיבית לבין תהליך חישוב ערכה של פונקציה כזו, יש עדויות לכך שהידע הדרוש לייצוג פונקציה רקורסיבית שונה במהותו מהידע הדרוש לצורך חישוב הערך של הפונקציה. המאמר הנוכחי בחר להתמקד בידע מהסוג השני, ובחירה זו עלולה להפתיע את מי שמתמקד במטרת המתכנת בגישה דקלרטיבית לבחור ייצוג נאות לפונקציה ו"שם בצד" שיקולים של כיצד הפונקציה מתבצעת בסופו של דבר על ידי המחשב. ואמנם, מרבית המחקרים על למידת רקורסיה התמקדו דווקא ביצירתן של הפונקציות הרקורסיביות על ידי המתכנת ולא בתהליך הערכתן וחישובן על ידו, וגם המעטים שעסקו בתהליך החישוב עשו זאת במגמה לבחון את המודל שיש למתכנתים על התהליך ולא כדי להבין את תהליך החישוב בפני עצמו. אולם, היכולת לבצע חישוב "בראש" של פונקציה רקורסיבית, ללא סיוע מחשב, היא מרכיב חיוני בידע הרקורסיה של לומדים ומומחים כאחד.

בעקבות נסיונה של המחברת, מתואר במאמר המודל הבא. המודל משמש לומדים בתהליך הפיתוח והבדיקה של פונקציות רקורסיביות:



על פי המודל הזה, נודעת חשיבות רבה לידע כיצד לחשב בראש מה מחזירה פונקציה עבור קלט נתון. חשיבות זו גדולה במיוחד עבור מתכנתים מתחילים, אולם גם המומחים נזקקים לו. בהמשך המאמר, מוצגים מספר ממצאים הקשורים לסוג זה של ידע רקורסיה.

ממצאים

המחקרים שממצאיהם מוצגים כאן התבצעו עם אוכלוסיית סטודנטים מתנדבים לתואר ראשון באוניברסיטה, שהיו לקראת סיום הקורס הראשון בתכנות פונקציונלי בשפת מירנדה או שהשלימו זה מכבר קורס כזה. הסטודנטים למדו לקראת תוארים במגוון מקצועות כמו מדעי המחשב ומתמטיקה, אך כולם השלימו גם קורס בתכנות פרוצדורלי (בפסקל) וכולם חשבו שהם "יודעים" רקורסיה. הסטודנטים התבקשו לבצע הרצה ידנית של מספר פונקציות רקורסיביות. במחקר הראשון, שמונה סטודנטים התבקשו לבצע רק פונקציה רקורסיבית אחת, תוך שהם מתבקשים לחשוב בקול רם ולרשום את הפתרון שלהם על דף. במהלך המחקר הוקלט תהליך הפתרון, וההקלטות שימשו כנתוני המחקר יחד עם הרישומים על הדף. במחקר השני, 33 סטודנטים התבקשו לחשב ולרשום מה תבצענה שלוש פונקציות רקורסיביות, כל אחת עוסקת בתחום אחר של ארגומנטים (מספרים טבעיים, רשימות של מספרים ורשימות של תווים). הסטודנטים חולקו לשתי קבוצות כך שכל קבוצת סטודנטים קיבלה את אותן פונקציות בסדר אחר. כולם התבקשו לרשום את הפתרון על דפים – כולל מה שאינם מצליחים לפתור.

תפיסה מוטעית בולטת: המקרה הבסיסי כתנאי העצירה

על פי תפיסה זו, החישוב נפסק ברגע שמגיעים לקיומו של התנאי הקשור במקרה הבסיסי, ואין תהליך של החזרת ערכים לפונקציות הקוראות. במחקר הראשון, התפיסה המוטעית הנ"ל הופיעה ברישומיהם של שלושה משמונת הנחקרים, אולם בעקבות ניתוח ההקלטות התברר כי עוד שניים מחזיקים בה. באחד המקרים יוצגה תפיסה מוטעית זו באופן קיצוני, כאשר הסטודנט הצהיר שהערך של המקרה הבסיסי הוא הוא הערך של הפונקציה בקריאה המקורית.

לתפיסה מוטעית זו יש ככל הנראה שני מקורות. המקור הראשון קשור באופן הייצוג הדקלרטיבי, שעשוי לתת רמזים מטעים בדבר תהליך החישוב ובמיוחד בדבר האופן שבו תהליך זה נעצר. המקור השני נעוץ בהכרות מוקדמת עם רקורסית קצה: במקרים של פונקציות רקורסיביות המיוצגות באמצעות רקורסית קצה, המחשבה כי המקרה הבסיסי עוצר את התהליך היא מחשבה הגיונית. רוב הדוגמאות שהוצגו לנחקרים בעת לימודיהם בקורס תכנות פונקציונלי, היו דוגמאות של רקורסית קצה. ואכן, אחד הסטודנטים שהסביר את קשייו טען "כל הרקורסיה שראינו עד עכשיו מפסיקה במקרה הבסיסי". אולם, במחקר השני נמצא גם, שיש קשר לתחום הארגומנטים שבו עוסקות הפונקציות הרקורסיביות. כאשר הפונקציה עסקה ברשימות, הסטודנטים חשו כי אין זה הגיוני לחשוב שהמקרה הבסיסי עוצר את התהליך. זאת משום שבדרך כלל המקרה הבסיסי קשור ברשימה ריקה, ואין זה נראה לסטודנטים הגיוני לעצור דווקא אז.

תפיסות מוטעות בקשר לתפקידם של משתנים

לתפיסות מוטעות אלה אין קשר הכרחי לרקורסיה, אבל הן באות לידי ביטוי בולט בעת הנסיון לבצע בראש את תהליך החישוב של פונקציות רקורסיביות.

משתנים בשפת מירנדה מתנהגים כמו משתנים במתמטיקה ולא כמו משתנים בשפות פרוצדורליות. אבל חלק מהסטודנטים במחקרים שערכנו התייחסו למשתנה הפונקציה אותה הם התבקשו לחשב לא כאל משתנה מתמטי שמקבל את אותו ערך בכל אחד מהמופעים שלו, אלא כאל משתנה תכנותי שערכו אחר בכל מופע. התפיסה המוטעית הזו הופיעה רק בהקשר של פונקציות הפועלות על ארגומנטים מסוג רשימה, ולא בהקשר של פונקציות מתמטיות מעל המספרים הטבעיים. לדוגמה, הפונקציה g פועלת על רשימה L של מספרים ומוגדרת באופן הבא –

g (L) = first(L) + g (tail (L)) + first(L).

במקרה כזה, יש סטודנטים שחושבים שהערך של L במרכיב השלישי (הימני ביותר) של הביטוי משתנה לאחר שהתבצע המרכיב השני. באופן זה, אם הרשימה L היתה בהתחלה [3 5 1], אז המרכיב הראשון בביטוי יחזיר את המספר 1, אבל במרכיב השני קוצצים את ראש הרשימה ונשארת הרשימה [3 5], ולכן – לפי התפיסה המוטעית – מה שיחזיר המרכיב השלישי בביטוי הוא המספר 5.

g (L) = first(L) + g (tail (L)) + first(L)

L= [1 5 3]   L= [1 5 3]   L= [5 3]

first(L)=1   tail(L)=[5 3]   first(L)=5

[הייצוג כאן הוא תרגום מייצוג בשפת מירנדה לייצוג שיהיה בהיר יותר לקוראים. משמעות first היא המספר הראשון ברשימה, ומשמעות tail היא הרשימה ללא המספר הראשון שלה – ד"ל].

מה מקורו של הבלבול ביחס לתפקידם של המשתנים? האמנם הסטודנטים הללו באמת חושבים שערכי המשתנים מתעדכנים תוך כדי התקדמות החישוב של הביטויים משמאל לימין? מנתוני המחקר לא היה ברור באופן חד משמעי האם אכן מדובר בתפיסה מוטעית, או שאולי הבלבול הוא תוצאה של בחירה לא מתאימה באסטרטגיה לחישוב המכונה "חישוב מיידי" אשר תתואר בהמשך.

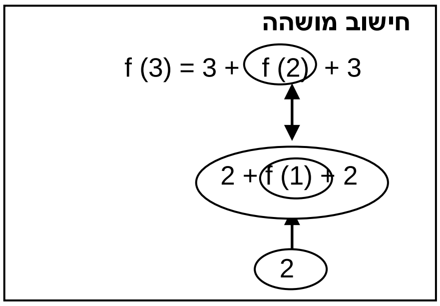
ההשפעה של אסטרטגית החישוב

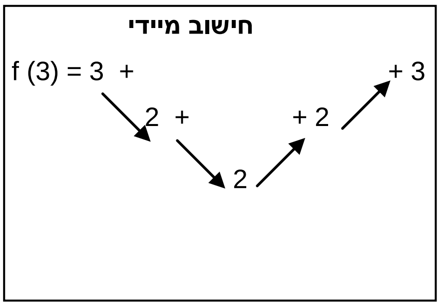
ממצא נוסף מהמחקרים קשור בהשפעתן של האסטרטגיות בהם השתמשו הלומדים לחישוב ערכן של פונקציות רקורסיביות על הביצועים של הסטודנטים שהשתתפו במחקרים. מהתצפיות מתברר כי הסטודנטים משתמשים בשתי אסטרטגיות בולטות לחישוב: חישוב מושהה (delayed evaluation) וחישוב מיידי (immediate evaluation). האסטרטגיה הראשונה דומה לחיפוש לרוחב (breadth first): הסטודנטים מחשבים ומציבים את ערכן של כל הפונקציות המרכיבות את ההגדרה, פרט לערך של הקריאה הרקורסיבית לפונקציה – שחישוב ערכה מושהה. במלים אחרות, הם רואים את הערך של הפונקציה בכל נקודה כביטוי שלם, שיתכן וחלק ממרכיביו עדיין לא חושבו. בחישוב מיידי, לעומת זאת, החישוב מתקדם משמאל לימין, כאשר חישוב הערך של קריאה רקורסיבית מתבצע מיד כשנתקלים בקריאה כזו. אסטרטגיה זו דומה יותר לחיפוש לעומק (depth first).

בתרשים הבא מושוות שתי האסטרטגיות לחישוב הפונקציה הרקורסיבית הבאה –

$$\begin{cases} f(n) = n + f(n-1) + n \\ f(1) = 2 \end{cases}$$

החיצים בתרשים מייצגים את סדר החישוב, במקרה של הפעלת הפונקציה כאשר n=3.





האינטואיציה כיוונה את המחברת לחשוב שמי שיפעיל אסטרטגיה של חישוב מיידי עשוי "ללכת לאיבוד", בעוד אלה המשתמשים בחישוב מושהה לא צפויים להציג בלבול ותפיסות מוטעות. הנתונים הצדיקו את האינטואיציה: במחקר הראשון, כל החמישה שענו תשובות נכונות השתמשו באסטרטגיה של חישוב מושהה. במחקר השני, תיאר אחד הסטודנטים את ההשפעה של השימוש באסטרטגיה של חישוב מיידי: "אני חושב ש ... כשיש לך פונקציה שיש בה רקורסיה באמצע – שיש לה חלק מרכזי שהוא רקורסיבי – אתה שוכח את מה שיש בסוף. אז אתה עושה את כל הדרך למטה ואז אתה עשוי לחשוב שמוכרחים לזכור את זה לרמה העליונה, ואז אתה בכלל שוכח את הרמה האמצעית".

שימוש באסטרטגיה של חישוב מושהה מוביל, אם כן, לביצוע מוצלח יותר של משימות הערכה של פונקציות רקורסיביות. לאסטרטגיה זו יש מספר ואריאציות, וביניהן "חישוב מושהה מלמעלה למטה" (top-down delayed evaluation) כמו בתרשים לעיל, וכן "חישוב מושהה מלמטה למעלה" (bottom-up delayed evaluation) שבו החישוב מתחיל מהמקרה הבסיסי. יש שיקולים בעד ונגד שימוש בכל אחת מהגרסאות של החישוב המושהה, והבחירה ביניהן תלויה במידה רבה בפונקציה אותה מנסים לחשב. בכל מקרה, רצוי שהסטודנטים יכירו את שתיהן, וילמדו מתי כדאי להפעיל כל אחת.

סיכום

כמו בכל מחקר שמזהה קשיים של לומדים, יש צורך גם כאן בהתייחסות לסוגייה של התגברות על הקשיים, או בהצעת גישה מתקנת לבעיות שזוהו. ראשית, בעניין התפיסה המוטעית הראשונה שאובחנה במחקר אשר רואה במקרה הבסיסי את תנאי העצירה, אנו סבורים כי ניתן למנוע אותה אם מציגים בפני הלומדים לא רק דוגמאות של רקורסית קצה. שנית, אנו ממליצים לעודד לומדים להשתמש באסטרטגיות של חישוב מושהה, וזאת כדי להתגבר על כל סוגי התפיסות המוטעות שזוהו ולשפר את הביצועים באופן כללי. השילוב של שתי הגרסאות של חישוב מושהה בהן דנו קודם מספק ללומד מודל טוב ומספק של כיצד מבצע המחשב את הפונקציות הרקורסיביות, ושילוב זה גם מאפשר ללומד לצפות ולשער במידה רבה של דיוק מה תהיה תוצאת החישוב.

רשימת מקורות למבוא

1. אהרוני, ד. (1999). *תפיסת מושגים במבני נתונים על ידי סטודנטים לתואר ראשון*. חיבור לתואר דוקטור, המחלקה להוראת הטכנולוגיה והמדעים - הטכניון, חיפה.
2. בן-ארי, מ. וצוות הפיתוח (1998). *יסודות מדעי המחשב*, פרק 3. המחלקה להוראת המדעים, מכון ויצמן למדע והמרכז הישראלי להוראת המדעים ע"ש עמוס דה-שליט.
3. בן-ארי מ. ורייך נ. (1996). רקורסיה - מהמחשה לתכנית. *הבטים בהוראת מדעי המחשב* 7, עמודים 45-47.
4. ברנדס, ע. וצוות מדעי המחשב (1997). *עיצוב תכנה*. המרכז להוראת המדעים, האוניברסיטה העברית בירושלים.
5. היליס, ד. ו. (2000). *התבנית שבאבן - הרעיונות הפשוטים שביסוד פעולתו של המחשב*. הוצאת הד ארצי, אור יהודה. תרגום: עמנואל לוטם.
6. חדשי ר. (1996). *הבטים אינטרדיסציפלינריים של מדעי-המחשב*. חיבור לתואר M.Sc, המחלקה להוראת הטכנולוגיה והמדעים - הטכניון, חיפה.
7. לוי, ד. (1991). *מטפורות בהוראת מדעי-המחשב בתיכון*. חיבור לתואר M.Sc, המחלקה להוראת הטכנולוגיה והמדעים - הטכניון, חיפה.
8. לוי, ד. (2001). *התפתחות מושגים במדעי המחשב באמצעות פעילויות סיווג והכללה*. חיבור לתואר דוקטור, המחלקה להוראת הטכנולוגיה והמדעים - הטכניון, חיפה.
9. לירון א. (1996). למה קשה להבין רקורסיה? *הבטים בהוראת מדעי המחשב* 7, עמודים 6-13.
10. לפידות, ת., לוי, ד. ופז, ת. (2001). *תכנות פונקציונלי – מדריך למורה*. המחלקה להוראת הטכנולוגיה והמדעים, הטכניון והמרכז הישראלי להוראת המדעים ע"ש עמוס דה-שליט.
11. צצקיס, ר. (1985). *רקורסיה - רעיון רב עוצמה במתמטיקה, מחשבים וחינוך*. חיבור לתואר שני, הפקולטה לחינוך - אוניברסיטת חיפה.
12. שרץ, ז., הברמן, ב., בן-זקן, א., וגלוברמן נ. (1996). *תכנות לוגי (פרולוג)*. המחלקה להוראת המדעים, מכון ויצמן למדע והמרכז הישראלי להוראת המדעים ע"ש עמוס דה-שליט.
13. Anderson J.R., Pirolli P. & Farrel R. (1988). Learning to program recursive functions. In M.T. Chi, R. Glaser & M.J. Farr (Eds.), *The nature of expertise* (pp. 153-183). Lawrence Erlbaum Associates, Hillsdale, NJ.
14. Ben-Ari, M. (1997). Recursion: From Drama to Program. *Journal of Computer Science Education, 11(3),* 9-12.
15. Bhuiyan, S., Greer, J.E. & McCalla, G.I. (1994). Supporting the learning of recursive problem solving. *Interactive Learning Environments*, 4 (2), 115-139.
16. Ford, G. (1982). A framwork for teaching recursion. *SIGCSE Bulletin* 14(2), 32-39.
17. Garland, S.J. (1987). *Introduction to computer science with applications in Pascal*. NY: Addison-Wesley.

18. George, C.E. (2000-a). EROSI – Visualising recursion and discovering new errors*. The Proceedings of the 31st SIGCSE technical symposium on computer science education,* Austin, Texas. 305-309.

19. George, C.E. (2000-b). Experiences with novices: The importance of graphical representations in supporting mental models. In A.F. Blackwell & E. Bilotta (Eds). *Proceedings of the 12th Workshop of the Psychology of Programming Interest Group*, Cozenza Italy.

20. Gersting, J.L. (1996). *Mathematical structures for computer science* (third edition). NY: W. H. Freeman and Company.

21. Harvey, B. & Wright, M. (1994). *Simply Scheme*. Cambridge, Mass.: MIT press.

22. Harvey, B. (1985). *Computer science logo style, volume 1: Intermediate programming*. Cambridge, Mass.: MIT press.

23. Haynes, S.M. (1995). Explaining recursion to the unsophisticated. *SIGCSE Bulletin* Vol. 27, No.3, 3-6.

24. Hofstadter, D. (1979). *Godel, Escher, Bach – An Eternal Golden Braid*. NY: Vintage books.

25. Kahney, H. (1989). What do novice programmers know about recursion? Soloway, E. & Sphorer, J.C. (eds), *Studying the novice programmer*, LEA publishers. 209-228.

26. Kurland D.M. & Pea. R.D. (1983). Children's mental models of recursive LOGO programs. In *Proceedings of the 5th Annual Conference of the Cognitive Science Society, session 4*, 1-5.

27. Leron U. (1988). What makes recursion hard*. Proceedings of the Sixth International Congress on Mathematics Education (ICME6*), Budapest, Hungary.

28. Roberts E.S. (1986). *Thinking Recursively*. John Wiley & Sons.

29. Segal, J. (1995). Empirical studies of functional programming learners evaluating recursive functions. *Instructional Science 22*, 385-411.

30. Troy M.E. & Early G. (1992). Unraveling recursion, Part I and II. *The Computing Teacher, March-April 1992*.

31. Valazquez-Iturbine, J.A. (2000). Recursion in gradual steps (is recursion really that difficult?). *The Proceedings of the 31st SIGCSE technical symposium on computer science education*, 310-314. Austin, Texas

32. Wu, C., Dale, N.B., & Bethel, L.J. (1998). Conceptual models and cognitive styles in teaching recursion. *SIGCSE 30*, Atlanta GA, USA, 292-296.

רשימת מקורות לנושא "למידה והוראה של רקורסיה"

1. Anderson J.R., Pirolli P. & Farrel R. (1988). Learning to program recursive functions. In M.T. Chi, R. Glaser & M.J. Farr (Eds.), *The nature of expertise* (pp. 153-183). Lawrence Erlbaum Associates, Hillsdale, NJ.
2. Ben-Ari, M. (1997). Recursion: From Drama to Program. *Journal of Computer Science Education, 11(3),* 9-12.
3. Bhuiyan, S., Greer, J.E. & McCalla, G.I. (1994). Supporting the learning of recursive problem solving. *Interactive Learning Environments*, 4 (2), 115-139.
4. Ford, G.A. (1984). An implementation-independent approach to teaching recursion, *ACM SIGCSE BULLETIN*, 16(1), 213-216.
5. George, C.E. (2000-a). EROSI – Visualising recursion and discovering new errors. *The Proceedings of the 31st SIGCSE technical symposium on computer science education,* Austin, Texas. 305-309.
6. George, C.E. (2000-b). Experiences with novices: The importance of graphical representations in supporting mental models. In A.F. Blackwell & E. Bilotta (Eds). *Proceedings of the 12th Workshop of the Psychology of Programming Interest Group*, Cozenza, Italy.
7. Ginat, D. and Shifroni, E. (1999). Teaching recursion in a procedural environment – how much should we emphasize the computing model? *The proceedings of the 30th SIGCSE technical symposium on computer science education*, New-Orleans, Louisiana. 127-131.
8. Greer J.E. (1987). *Empirical Comparison of Techniques for Teaching Recursion in Introductory Computer Science.* Ph.D dissertation, The University of Texas at Austin, 1987.
9. Harvey, B. & Wright, M. (1994). *Simply Scheme*. Cambridge, Mass.: MIT press.
10. Henderson, P. B., & Romero, F. J. (1989). Teaching recursion as a problem-solving tool. *ACM SIGCSE BULLETIN*, 21(1), 27-31
11. Kurland D.M. & Pea. R.D. (1983). Children's mental models of recursive LOGO programs. In *Proceedings of the 5th Annual Conference of the Cognitive Science Society, session 4*, 1-5.
12. Kurland D.M., Pea R.D & Mawby (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, 2(4), 429-459.
13. Leron U. (1988). What makes recursion hard. *Proceedings of the Sixth International Congress on Mathematics Education (ICME6*), Budapest, Hungary.
14. Leron, U. & Zazkis, R. (1986). Mathematical induction and computational recursion. *For the Learning of Mathematics,* 6(2), 25 - 28.
15. Levy, D. & Lapidot, T. (2000). Recursively Speaking: Analyzing Students' discourse of Recursive Phenomena. In S. Haller (Ed.), *Proceedings of the annual meeting of the special interest group for computer science education* (SIGCSE 2000). Austin, Texas: The ACM.

16. Levy, D. (2001). 'It's just like the whole picture, but smaller': Expressions of gradualism, self-similarity, and other pre-conceptions while classifying recursive phenomena. *Proceedings of the 13th annual workshop of the psychology of programming interest group* (PPIG 2001): Bournemouth, UK.

17. Levy, D. (2001). Insights and Conflicts in Discussing Recursion: A Case Study. *Computer Science Education,* 11(4).

18. McCalla G.I. & Greer J.E. (1993). Two and one-half approaches to helping novices learn recursion. In E. Lemout, B. Du Boulay & G. Dettori (Eds.), *Cognitive models and intelligent environments for learning programming (*pp. 185-197). NY: Springer Verlag,

19. Pirolli P.L. & Anderson J.R. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, 39, 240-272.

20. Roberts E.S. (1986). *Thinking Recursively*. John Wiley & Sons.

21. Troy M.E. & Early G. (1992). Unraveling recursion, Part I and II. *The Computing Teacher, March-April 1992*.

22. Valazquez-Iturbine, J.A. (2000). Recursion in gradual steps (is recursion really that difficult?). *The Proceedings of the 31st SIGCSE technical symposium on computer science education*, 310-314. Austin, Texas

23. Wiedenbeck S. (1988). Learning recursion as a concept and as a programming technique. *The Proceedings of the 19st SIGCSE technical symposium on computer science education*, 275-278.

24. Wiedenbeck S. (1989). Learning iteration and recursion from examples. *International Journal of Man-Machine studies*, 30, 1-22.

25. בן-ארי מ. ורייך נ. (1996). רקורסיה - מהמחשה לתכנית. *הבטים בהוראת מדעי המחשב* 7, עמודים 45-47.

26. חדשי ר. (1996). *הבטים אינטרדיסציפלינריים של מדעי-המחשב*. חיבור לתואר M.Sc, המחלקה להוראת הטכנולוגיה והמדעים - הטכניון, חיפה.

27. לוי, ד. (2001). *התפתחות מושגים במדעי המחשב באמצעות פעילויות סיווג והכללה*. חיבור לתואר דוקטור, המחלקה להוראת הטכנולוגיה והמדעים - הטכניון, חיפה.

28. לוי ד. ולפידות ת. (1996). בניית המושג רקורסיה באמצעות סיווג. *הבטים בהוראת מדעי המחשב* 7, עמודים 33-44.

29. לוי, ד. (2000). 'בחזרה לשורשים או: מיהו יהודי': ביטויים של תפיסת המושג רקורסיה אצל מורים למתמטיקה ומדעי-המחשב. כנס *איל"ה 12 – המחקר בחינוך ויישומו בעולם משתנה,* 234-238. אוניברסיטת תל-אביב.

30. לירון א. (1996). למה קשה להבין רקורסיה? *הבטים בהוראת מדעי המחשב* 7, עמודים 6-13.

31. צצקיס, ר. (1985). *רקורסיה - רעיון רב עוצמה במתמטיקה, מחשבים וחינוך*. חיבור לתואר שני, הפקולטה לחינוך, אוניברסיטת חיפה.