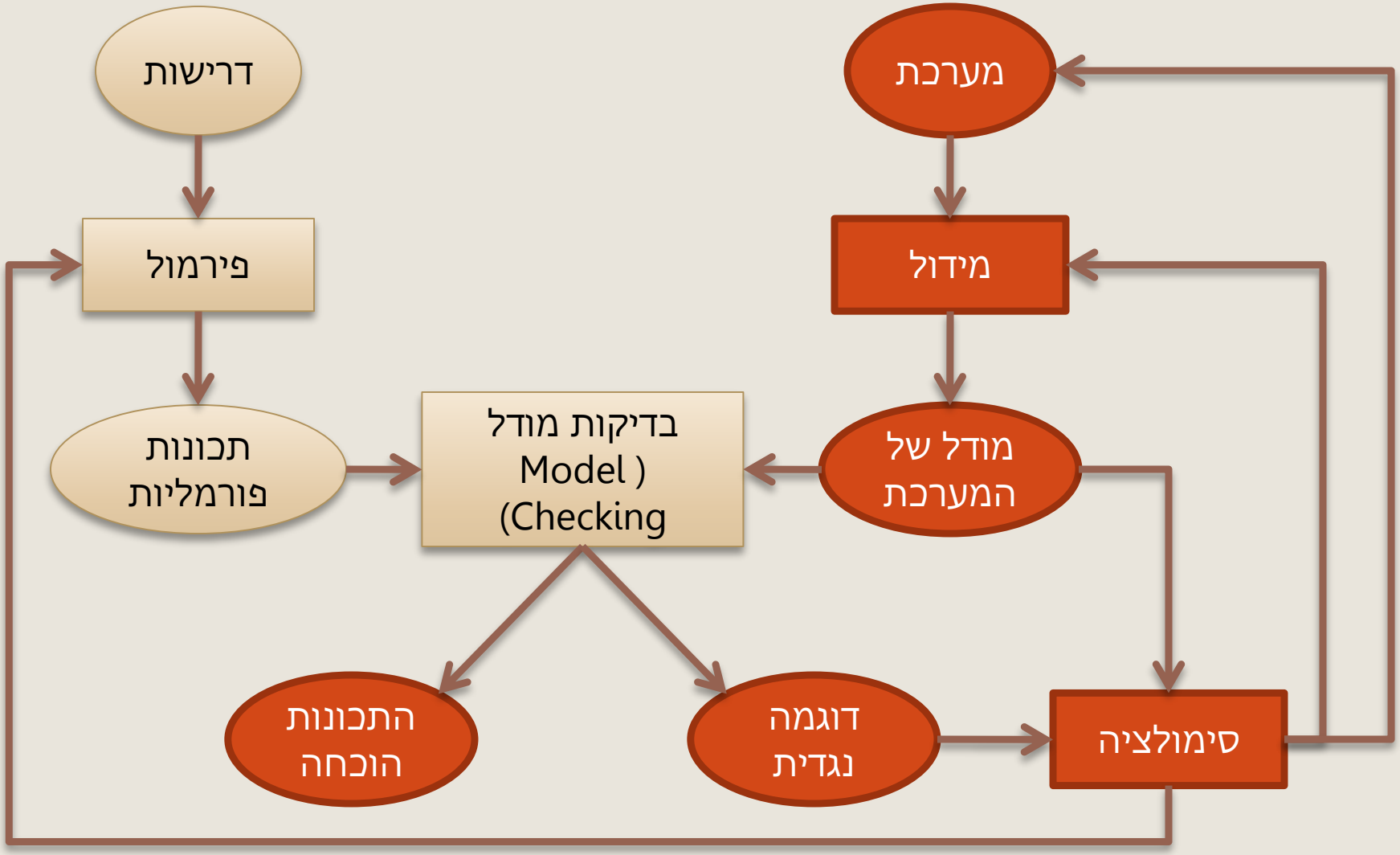
239

תכונות שמורה
Invariant Properties

גרא וייס
המחלקה למדעי המחשב
אוניברסיטת בן-גוריון

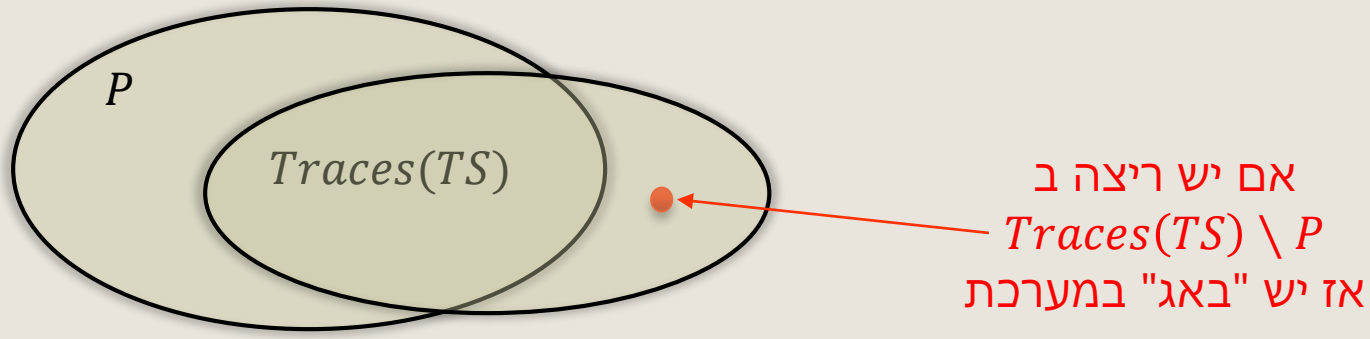
בדיקות מודל (Model Checking)



תזכורת: תכונות זמן ליניארי
Linear Time Properties

הגדרה: תכונת זמן ליניארי P היא קבוצה של מילים אינסופיות מעל האלף-בית 2^{AP}

- מערכת מקיימת תכונה, $TS \vDash P$, אם כל רצפי העקבות שהיא משאירה נמצאים ב-P:



- אנחנו מכוונים ללוגיקה שבה ניתן יהיה לתאר תכונות של מערכות ושאפשר לבקש, למשל מ-SPIN, לבדוק אם מערכת מעברים מקיימת תכונה נתונה
- נראה דרכים שונות לתאר תכונות (תכונות שמורה, אוטומטים, לוגיקה,...)

מאפיינים של מערכות שאינם תכונות זמן לינארי

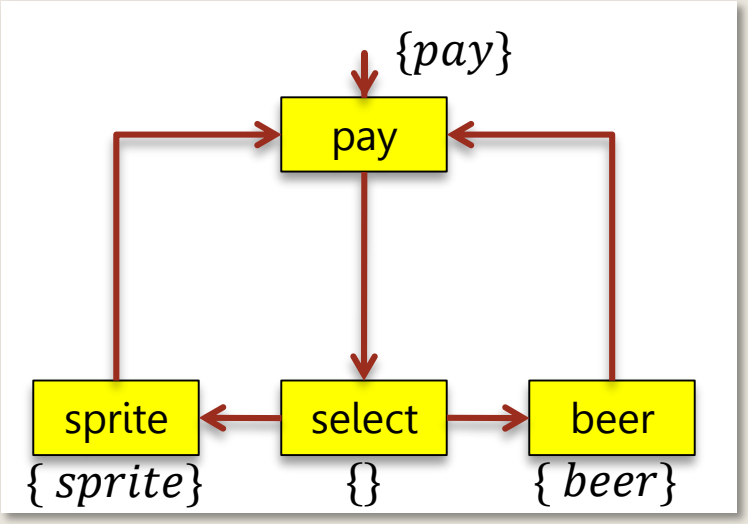
- יש ארבעה מצבים במערכת המעברים
- לכל מצב יש עוקב שיש לו עוקב המתויג באופן מסוים
- מכל מצב במערכת יש מסלול למצב עם תיוג מסוים
- יש למערכת ריצות מסוימות

$$Traces(TS_1) \subseteq Traces(TS_2)$$

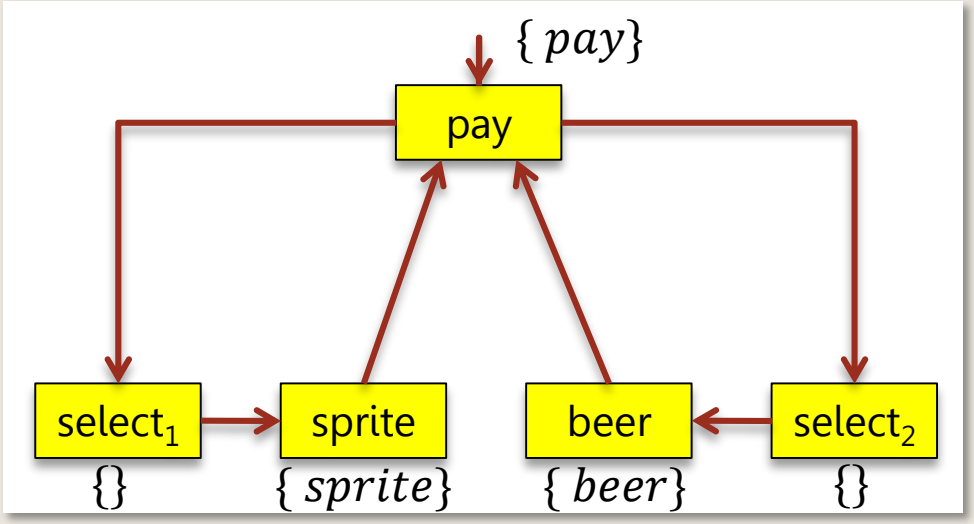
אם ורק אם

לכל תכונת זמן לינארי P מתקיים $TS_2 \vDash P \;\Rightarrow\; TS_1 \vDash P$

דוגמה: שתי מכונות שתייה



≡



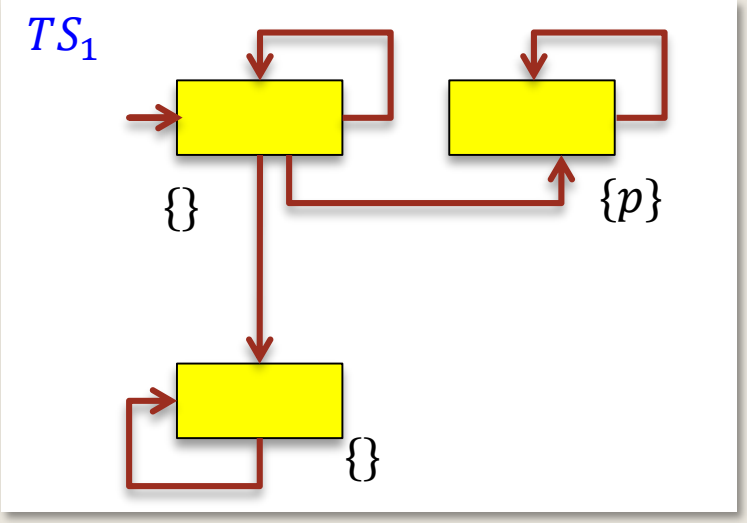
$$AP = \{ pay, sprite, beer \}$$

אין תכונת זמן ליניארי שיכולה להבדיל בין שתי המכונות האלה

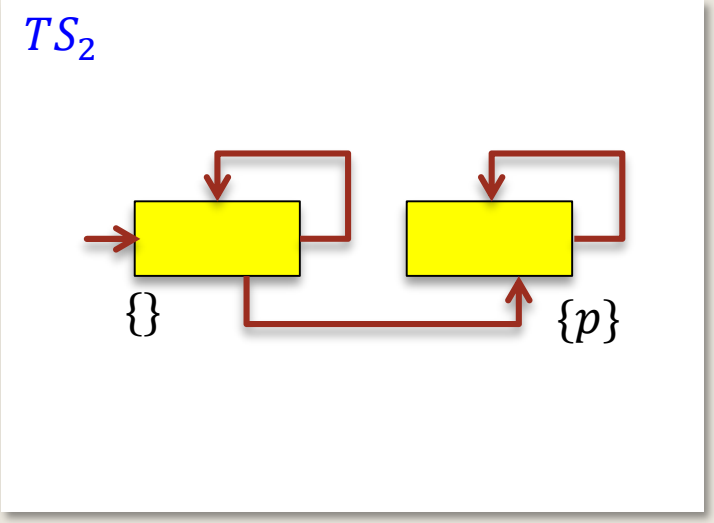
מסקנה: האפיון "יש ל-TS ארבעה מצבים" אינו תכונת זמן לינארי.

דוגמה: "הכל צפוי והרשות נתונה" (אבות ג' טו')

http://www.leibowitz.co.il/leibarticles.asp?id=62



≡

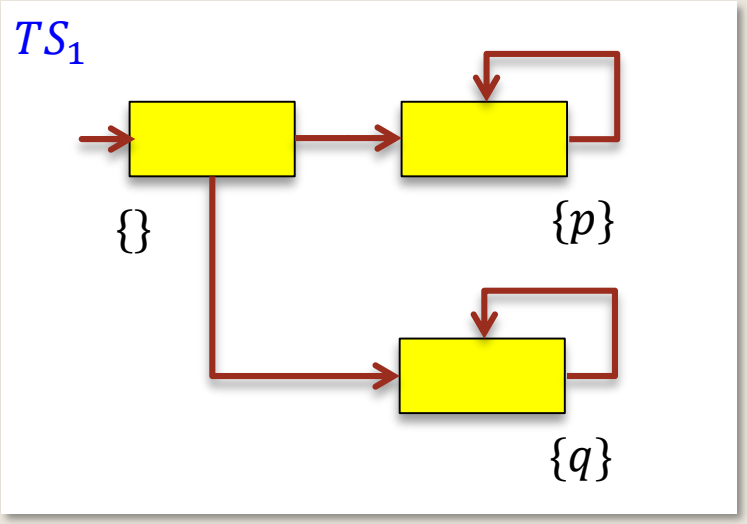


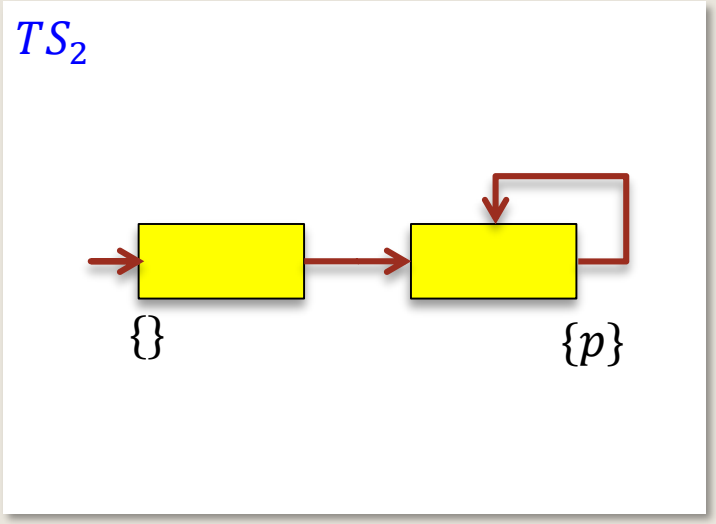
לכן $Traces(TS_1) = Traces(TS_2)$

אין תכונת זמן ליניארי שיכולה להבדיל בין שתי המכונות האלה

מסקנה: האפיון "מכל מצב יש מסלול למצב המקיים את התכונה p" אינו תכונת זמן לינארי.

דוגמה: תכונה הנוגעת לקיום ריצות





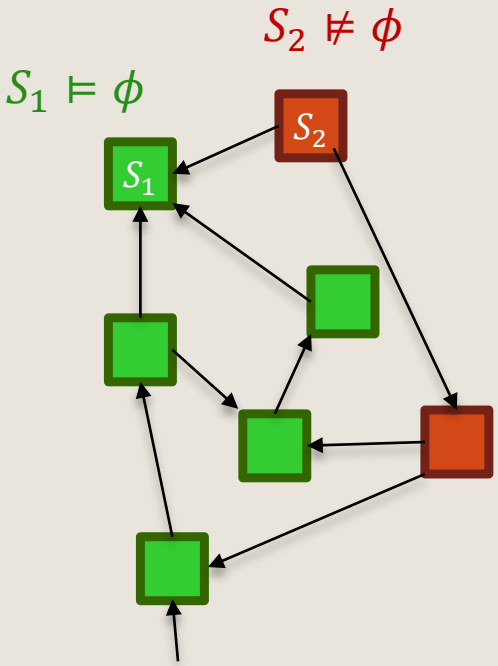
לכן $Traces(TS_2) \subseteq Traces(TS_1)$

כל תכונת זמן לינארי ש-TS_1 תקיים גם TS_2 תקיים

מסקנה: המאפיין "$\{\ \}\{q\}^\omega \in Traces(TS)$" אינו תכונת זמן לינארי.

תוכנית להמשך: סוגים של תכונות

מתנאים על המצבים לתכונות זמן לינארי



$$S \vDash p \quad \Leftrightarrow \quad p \in L(S)$$

$$S \vDash \neg\phi \quad \Leftrightarrow \quad S \nvDash \phi$$

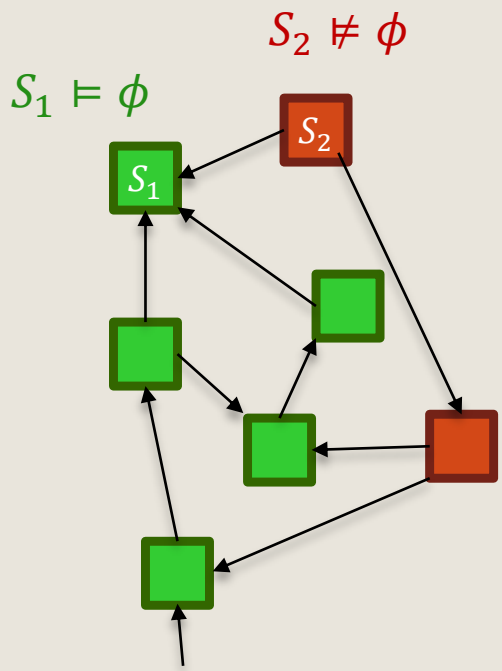
$$S \vDash \phi_1 \wedge \phi_2 \quad \Leftrightarrow \quad (S \vDash \phi_1) \wedge (S \vDash \phi_2)$$

תכונות שְׁמוּרָה (invariants)

הגדרה: תכונת זמן ליניארי P_{inv} היא שְׁמוּרָה אם יש תנאי מצב ϕ כך ש-

$$P_{inv} = \{\, \sigma \in (2^{AP})^{\omega} : \ \forall i \geq 0 \,.\, \sigma[i] \vDash \phi \,\}$$

- ϕ נקרא תנאי השמורה (invariant condition)
- קל להוכיח: $TS \vDash P_{inv}$ אם ורק אם
  - $trace(\pi) \in P_{inv}$ לכל מסלול π של TS
  - $L(s) \vDash \phi$ לכל מצב s השייך למסלול של TS
  - $L(s) \vDash \phi$ לכל מצב $s \in Reach(TS)$
- אינדוקציה:
  - ϕ מתקיים בכל מצב התחלתי
  - אם ϕ נכון במצב הוא נכון בעוקבים שלו



דוגמאות

מי מהתכונות הבאות הן שמורות? הוכיחו טענותיכם.

$$\{\sigma \in (2^{AP})^{\omega}:\ \forall i.\sigma[i] \vDash \overbrace{p \to (q \vee \neg r)}^{\phi}\}$$

$$\{\sigma \in (2^{AP})^{\omega}: \sigma[0] \vDash (p \to q)\ \wedge\ \forall i.\ (q \in \sigma[i] \vee p \notin \sigma[i]) \to \sigma[i+1] \vDash (q \vee \neg p)\}$$

$$\{\sigma \in (2^{AP})^{\omega}: p \in \sigma[0]\ \to \forall i. p \in \sigma[i]\}$$

איך נוכיח שתכונה אינה שמורה?

דוגמאות

איך נוכיח שתכונה אינה שְׁמוּרָה?

<u>טענה</u>: התכונה הבאה אינה תכונת שְׁמוּרָה

$$P = \{\sigma \in (2^{AP})^{\omega} : p \in \sigma[0] \ \rightarrow \forall i. p \in \sigma[i]\}$$

<u>הוכחה</u>:

- נניח בשלילה שקיימת תכונת מצב ϕ כך ש-

$$P = \{\sigma \in (2^{AP})^{\omega} : \ \sigma[i] \vDash \phi \text{ for all } i \in \mathbb{N}\}$$

- לפי הגדרת P, המילה $(\{\}\{p\})^{\omega} \in P$ ולכן, ע"פ הנחת השלילה, $\{\} \vDash \phi$ וגם $\{p\} \vDash \phi$
- לכן, ע"פ הנחת השלילה, המילה $\{p\}\{\}^{\omega} \in P$
- קיבלנו סתירה להגדרה של P

דוגמאות

איך נוכיח שתכונה אינה שְׁמוּרָה?

<u>טענה</u>: התכונה הבאה אינה תכונת שְׁמוּרָה

$$P = \{\, \sigma \in (2^{AP})^{\omega} : \quad p \in \sigma[2i] \text{ for all } i \in \mathbb{N}\}$$

<u>הוכחה</u>:

- נניח בשלילה שקיימת תכונת מצב ϕ כך ש-

$$P = \{\sigma \in (2^{AP})^{\omega} : \sigma[i] \vDash \phi \text{ for all } i \in \mathbb{N}\,\}$$

- לפי הגדרת P, המילה $(\{p\}\{\,\})^{\omega} \in P$ ולכן, ע"פ הנחת השלילה, $\{\,\} \vDash \phi$
- לכן, ע"פ הנחת השלילה, המילה $\{\,\}^{\omega} \in P$
- קיבלנו סתירה להגדרה של P

בדיקת שמורות

- בדיקת תכונת שמורה
  - האם תנאי השמורה מתקיים בכל מצב נגיש?
  - שימוש בגרסה של אלגוריתם סריקת גרף (BFS או DFS)
  - בהנחה שמערכת המעברים סופית
- ביצוע חיפוש DFS קדימה
  - אם מצאנו מצב s כך ש $s \not\models \phi$ מסיקים ש ϕ אינו תנאי שמורה
- אפשרות אחרת: חיפוש אחורה
  - מתחילים מהמצבים בהם ϕ אינו מתקיים ($s \not\models \phi$)
  - מחשבים את המצבים הקודמים $Pre^*(s)$ באמצעות DFS או BFS
  - אם הגענו למצב התחלתי ($I \cap Pre^*(s) \neq \emptyset$) אז ϕ אינו תנאי שמורה

בדיקת שמורה באמצעות DFS

קלט: מערכת מעברים סופית TS ונוסחה ϕ
פלט: **true** אם TS מקיימת את השמורה "תמיד ϕ", אחרת **false**

```
set of states R := ∅                    (* קבוצת המצבים בהם ביקרנו *)
stack of states U := ϵ;                 (* מחסנית מצבים "לטיפול" *)
boolean b := true;                      (* כל המצבים ב R מקיימים את φ *)
for all  s ∈ I  do
        if s ∉ R then
            visit(s)                    (* מתחילים DFS מכל מצב התחלתי *)
        fi
od
```

בדיקת שמורה באמצעות DFS

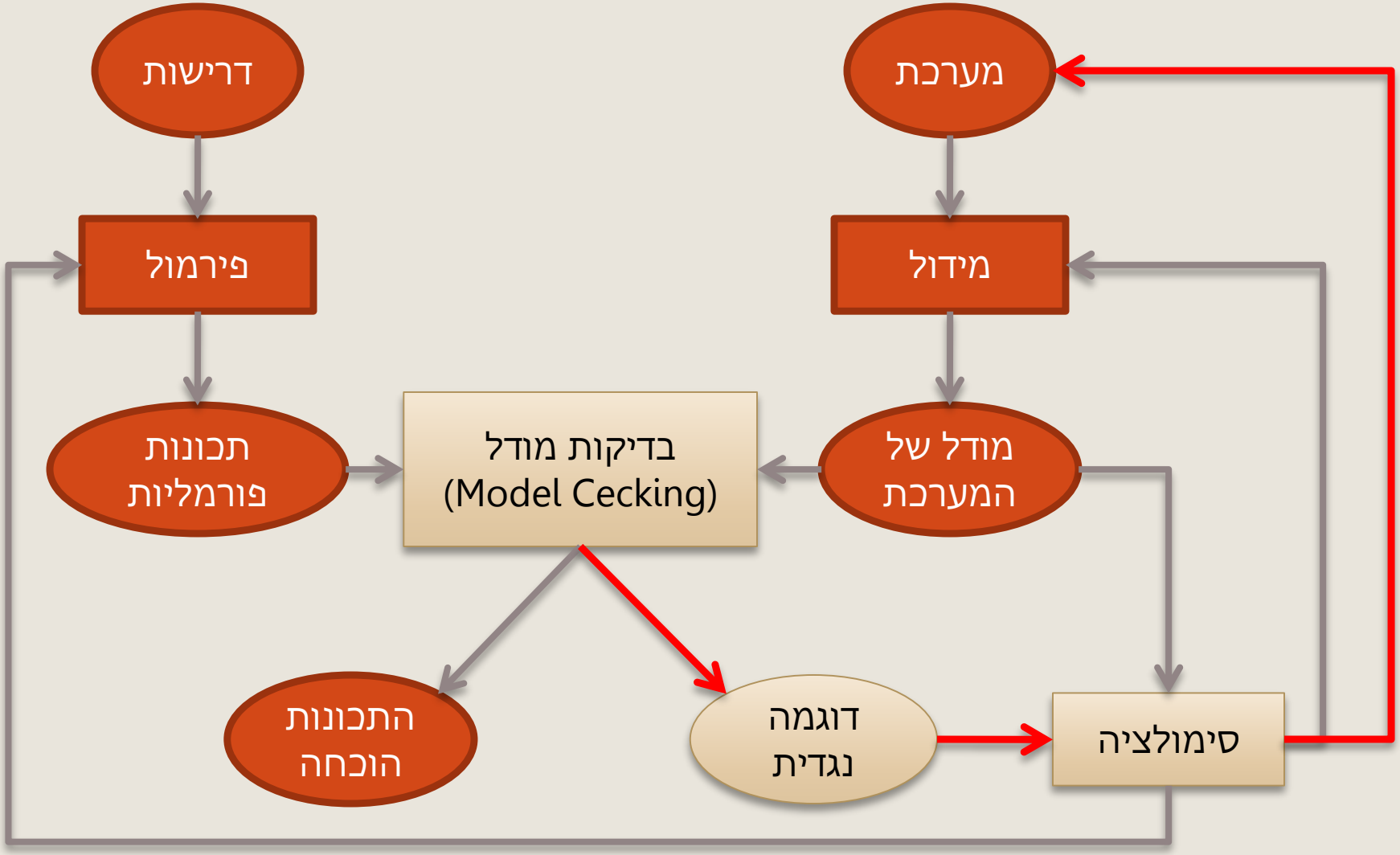
```
procedure visit(state s)                                  (* פרוצדורה לביקור במצב *)
      push(s, U);                                         (* דוחפים את המצב למחסנית *)
      R := R ∪ {s};                                       (* ומוסיפים אותו להמצבים שכבר ביקרנו *)
      repeat
          s' := top(U);
          if  Post(s')⊆ R  then
                pop(U);
                b := b ∧ (s' ⊨ ϕ);                        (* בודקים אם s' מקיימת את ϕ *)
          else
                take any s'' ∈ Post(s')\ R;
                push(s'', U);
                R := R ∪ {s''};                           (* גילינו מצב נגיש חדש s'' *)
          fi
      until  (U = ϵ ∨ ¬b)
endproc
```

סיבוכיות זמן

- נניח שניתן לסרוק כל $s' \in Post(s)$ בזמן $\theta(|Post(s)|)$
  - הנחה תקפה כאשר מייצגים את $Post(s)$ ע"י רשימות סמיכות
- סיבוכיות זמן בדיקת שמורה: $O(N \cdot |\phi| + M)$
  - N מסמל את מספר המצבים הנגישים
  - $M = \sum_{s \in S} |Post(s)|$ הוא מספר המעברים (בחלק הנגיש) של TS
- בדרך כלל לא מייצגים את רשימות הסמיכות באופן מפורש
  - למשל: ניתן להשתמש במעבר ישיר על גרפי התוכנית. במקרה זה, $Post(s)$ מתקבל מהכללים של יחס המעברים.

אלגוריתם שנותן גם דוגמה נגדית

```
set of states R := ∅                          (* קבוצת המצבים בהם ביקרנו *)
stack of states U := ϵ;                       (* מחסנית מצבים "לטיפול" *)
boolean b := true;                            (* כל המצבים ב R מקיימים את ϕ*)
while ((I \ R ≠ ∅) ∧ b) do
        take any s ∈ I \ R
        visit(s)
od
if b ten
        return "yes"
else
        return "no", reverse(U)
fi
```

דוגמה נגדית שימושית יותר מאשר הוכחת נכונות

כי דוגמה נגדית יכולה להצביע על בָּאג אמיתי