# RIVar

בטקסט זה אנו מציגים את *RIVar*, קונספט חדשני עבור מימוש *עקביות במידע* שמופיע במערכות תוכנה *ריאקטיביות*. מערכות ריאקטיביות הן מערכות שמקיימות אינטראקציה עם הסביבה לאורך זמן. מימוש עקביות במידע קיים במגוון גישות תכנותיות (להלן מידע הוא ערך של משתנים):

- Event - מגדירים קוד שיופעל תגובה לשינוי של ערך של משתנה, כאשר קוד זה מפעיל עדכון של משתנה אחר בהתאם למשתנה הראשון (וכך גם לגבי קבוצות של משתנים)
- מכונת מצבים - (כמו העבודות של דוד הראל, Redux ועוד) מגדירים למערכת מצבים לוגים הנגזרים ממצב המשתנים, ופעולות אשר מעבירות את המערכת בין המצבים. כמו כן, אירועים שיכולים לקרות במעבר בין המצבים.
- Data Binding - משתמשים בהפשטה אשר מפרידה בין הגדרות של *קשרים בין המשתנים*, לבין *מנגנון הסנכרון*.
- זרמים - קונספט מורכב ללמידה, ניתן באמצעותו להשתמש בתכנות פונקציונלי ולהתיחס להפשטה של הסטוריה של משתנה בשונה מהקונספט המוכר של משתנה כתא זכרון.
- Signals מגדירים משתנים בדומה לתאים באקסל, כך שניתן להגדיר משתנה input או משתנה מחושב בהתאם לנוסחה, וכתוצאה מכך מתבצעים חישובים באופן אוטומטי.

## בעיה

דרישה של ניהול עקביות מצביעה על *יתירות* במידע, כלומר על קיום של מידע משוכפל. המושג יתירות רלוונטי גם כן לגבי כמות הפעמים שמעדכנים משתנים (או מידע) לעומת הכמות הנדרשת בפועל. בנוסף, ניהול עקביות וניהול יתירות מתקשר גם לגבי כימות של קוד, כי קוד הוא גם מידע. אם קטע קוד מסוים משוכפל במקומות שונים במערכת, ורק חלק מההעתקים הללו משתנים (בהתאם לדרישות משתנות של לקוחות), המערכת עלולה להתנהג באופן שונה ובלתי צפוי במצבים זהים. בהתאם לכך, בקרב מתכנתים ישנה פרקטיקה Don't Repeat Yourself (DRY). יוצא שהסוגים השונים של היתירות כרוכים זה בזה: יתירות של מידע במשתנים, יתירות בעדכונים שלהם ויתירות בקוד.

להלן נציג קיומם של מקרים אשר בהם הגישות לניהול עקביות של משתנים תחת דרישה להימנע מעדכונים מיותרים כרוכות בבעיות שכפול של קוד:

- שימוש בevents - בעת מימוש של חישוב של משתנה שיהיה בהתאם לערכי משתנים אחרים בעזרת eventים, יש לחזור ולהפעיל את החישוב בכל אחד מהאירועים של המשתנים הרלוונטים (אם משתמשים באירוע אחד מתבצעים עדכונים מיותרים)
- signals - נוצר שכפול במשתנים כאשר מייצגים נתון מסוים, פעם כcomputed value ופעם כמשתנה קלט. תופעה זו קורית ממגבלה השפה: כל משתנה חייב להיות מכוון למקור נתונים יחיד. התוצאה של משתנים כפולים היא שכפול קוד כאשר נדרש להתייחס לשניהם.
- binding וכן state machine - גישות אלו או המימושים שלהן סובלים מריכוזיות, וגורמות לכך שלא ניתן לבצע שימוש בקוד חוזר בחלק מרכיב קיים, ובכך נגרם גם כן שכפול של קוד.
- זרמים - גישה מורכבת שגם אינה מספקת מענה ברור לבעיה.

לפני שנציג את RIVar וגישתנו למניעת שכפול קוד, עולה שאלה האם גישה זו אקטואלית. אנו בעידן שנחקרות גישות שעל פיהן כותבי הקוד העתידיים יהיו סוכני AI בעזרת מודלי שפה, ואלו מאופיינים בשימוש רב של יתירות. מצד שני, על פי הפרוטוקול הפופולארי MCP נראה שהפרומפטים של שפה טבעית לא מחליפים את הצורך בשפה פורמלית לתקשורת בין אדם למחשב.

עבור שפות פורמליות יש צורך בserver חכם בגישה של LSP, שיציע הצעות רלוונטיות למשתמש בשפה, ואשר יבצע לו השלמות אוטומטיות בהתאם למידע שהוא מזין. השלמות אוטומטיות והצעות להשלמות אוטומטיות קשורים לאתגר על שמירה על עקביות במידע. לכן שימוש בRIVar נשאר לדעתינו רלוונטי.

## גישה

RIVar הוא הפשטה של משתנה, שהמענה היחודי שלו הוא בזכות שילוב של התכונות Reactivity וExposability:

Reactivity בדומה לSignal ניתן לקשר נוסחה למשתנה כך שיתחשב אוטומטי בהתאם לה ולערכי המשתנים שבנוסחא. וכן ניתן להזין input למשתנה, אשר יגרום למשתנים אחרים להתעדכן בהתאם לנוסחאות שלהם.

Exposability קישור לנוסחה וכן הזנת קלט מתבצע כשירות או כשליחת הודעה לעצם בOOP. ההשלכות לכך הם שניתן לקשר נוסחה למשתנה, אף אם כבר קיימות נוסחאות אחרות שכבר מקושרות. וכן ניתן להזין קלט למשתנה, אף אם מקושר אליו נוסחה אחת או יותר.

RIVar זה ראשי תיבות Reactive Instance Variable ומכאן תכונותיו. הריאקטיביות הוא מהיותו *משתנה ריאקטיבי* מפרדיגמה "תכנות ריאקטיבי", והExposability הוא מהיותו כעין *משתנה מופע* שהוא משתנה של עצם מהפרדיגמה של "תכנות מונחה עצמים".

## אתגרים ומימוש

הפיזיבליות של הגישה מותנית בקיום מימוש יעיל של *Merge*, *Glitch Freedom* ו*Feedback Loops*. נסביר את האתגרים ואת אופן המימוש שבחרנו.

- האתגר של Merge נובע מכך שמשתנה מתעדכן בהתאם למספר מקורות ולכן יש לשלב ביניהם. בעיה זו מזכירה את בעית הmerge הידועה בפרדיגמה של זרמים ובמימושים בתחום של תכנות ריאקטיבי פונקציונלי.
- האתגר של Glich Freedom הוא כיצד להימנע מעדכון מיותר במקרים כגון, שיש משתנה שממנו מתעדכנים שני משתנים שונים ואשר יש משתנה נוסף אשר מתעדכן מחישוב של שתיהם. לאתגר זה יש פתרונות, אך יש לבחור פתרון שאינו ריכוזי, מה שמאפיין את רוב רובם של הפתרונות.
- האתגר השלישי הוא בכך שיכולים להיווצר מעגלים בזכות ההגדרות היחודיות של RIVar, כך ששינוי בערך של משתנה יוביל בעקיפין שינוי נוסף לאותו משתנה. ההתיחסות הקיימת למעגלים היא מעורבת, רוב המימושים האמינים דואגים שלא יהיו מעגלים, בעוד ישנם טענות ששפות שתומכות בsignals חייבות לתמוך במעגלים.

המימוש שלנו מתבסס על כך שכל משתנה הוא זרם ריאקטיבי. וכן, כל ביטוי הוא זרם ריאקטיבי. כל השמה של ביטוי הוא הפצה של זרם של הביטוי באגף הימני לזרם של משתנה האגף השמאלי. כל הזנה של נתון זה הוספת איבר לזרם. כעת יש לדאוג לפתור את האתגרים.

המפתח לפתרון האתגרים הוא שfeedback loop וglich זהים בכך שהם מבטאים עדכון ישן. בנוסף, יש דמיון בין שניהם לבעית merge בכך שצריך לבצע אבחנות בין ישן לחדש. המימוש כמובן צריך להיות לא ריכוזי אלא מבוסס על הודעות.

להלן תיאור של מימוש. בהתאם לסדר כרונולוגי של אירועי קלט. כל אירוע מקבל מספר סידורי גבוה יותר מקודמו. כמו כן, כאשר מתבצע חישוב של ערך בהתאם לערכים אחרים, נקבע לו קבוצה של ערכים אשר

משוייכים למספרים הסידוריים של האירועים שכתוצאה מהם נוצר הערך. למשל {1,2} מציין קבוצה המשוייכת לערך שחושב מערכים שנקלטו מאירועים שהמספרים הסידוריים שלהם הוא 1 ו2.

כעת כאשר לכל ערך מצורף מידע מלא של המספרים הסידוריים (לצורך פשטות ייוצג תמיד כקבוצה) ממנו הוא נוצר, ניתן להסיק מהו ערך "ישן" ומהו ערך "חדש". יש מצבים שקלים לאבחנה למשל {1} מול {3,4}. אך יש מצבים שיש להבחין בהם שהם כתוצאה מglitch או feedback loop שיולידו השוואות כגון {1,2} מול {1}. משני סוגי אבחנות אלו, הסקנו שערך "חדש" הוא ערך שיש לו בקבוצה ערך שגדול מכל הערכים בקבוצה של הערך השני, אך אין מדובר בקבוצה שהיא superset. זהו אלגוריתם בסיסי שיש לבסוס מול דרישות נוספות כגון ערכי ברירת מחדל.

## סיכום

בטקסט זה הצגנו את RIVar קונספט חדשני שמטרתו לנהל ביעילות יתירות במידע במערכות ריאקטיביות (כלומר ששינוי במידע מוביל באופן לא מפורש לשינוי במידע אחר וכן הלאה). בעזרת RIVar ניתן לקשר נוסחה אחת או יותר לכל משתנה, וכן ניתן להזין ערכים לכל משתנה. בעזרת המימוש המצורף לRIVar ניתן להתנסות בגישה החדשה ולקבל יעילות משופרת.