מעולה. קיבלתי. אנחנו נכנסים כעת לרזולוציה הגבוהה ביותר. התוכנית הבאה היא מסמך ארכיטקטוני מפורט לתת-מערכת ניהול הסיכונים בלבד. מסמך זה יכול לשמש כמתווה (Blueprint) לצוות הפיתוח.

### **תוכנית ארכיטקטונית מפורטת: תת-מערכת MARL לניהול סיכונים דינמי (M-RMS)**

**גרסה 1.0**

#### 1.0 פילוסופיה, מטרות ויעדים

##### 1.1 פילוסופיית המערכת

ה-M-RMS נוטשת את פרדיגמת ניהול הסיכונים מבוססת-החוקים הסטטיים, ומאמצת גישה של **מדיניות נלמדת ודינמית**. המערכת אינה "מפחדת" מסיכון, אלא לומדת לכמת, לתמחר ולהקצות אותו באופן אופטימלי בהתבסס על ההזדמנות בשוק וההקשר של חשבון המסחר.

##### 1.2 מטרה ראשית

לייצר באופן אוטונומי המלצת ניהול עסקה ({גודל פוזיציה, מרחק SL, יחס R/R}) אשר, בטווח הארוך, תמקסם את **יחס סורטינו (Sortino Ratio)** של עקומת ההון, תוך עמידה מלאה במגבלות שהוגדרו מראש.

##### 1.3 יעדים מדידים (KPIs)

* **יחס סורטינו (Sortino Ratio):** מעל 2.0 על פני תקופת בדיקה OOS.
* **משיכה מקסימלית (Max Drawdown):** נמוכה ככל האפשר, ואינה חוצה את מגבלות הכללים.
* **שיעור הפרת כללים:** 0%.
* **פקטור רווח (Profit Factor):** מעל 1.5.

#### 2.0 ארכיטקטורה מערכתית ורכיבים

##### 2.1 תרשים זרימה כללי

טריגר סינרגיה -> בניית וקטורים -> רשת הטמעה -> סוכן MARL -> המלצת תוכנית עסקה

##### 2.2 פירוט רכיבים

| **רכיב** | **קלט (Inputs)** | **עיבוד (Processing)** | **פלט (Outputs)** |
| --- | --- | --- | --- |
| **2.2.1 מרכיב וקטור הסינרגיה** | אותות גולמיים מ-MLMI, NW-RQK, FVG | הרכבת וקטור נומרי. חישוב "זמן מאז אות". נרמול ראשוני. | Synergy\_Vector (וקטור גולמי) |
| **2.2.2 מודול ניטור חשבון** | חיבור ל-API של הברוקר / סימולטור | קריאה רציפה של Equity, P&L, Drawdown. חישוב יחסים מנורמלים. | Account\_State\_Vector |
| **2.2.3 מנוע חוקים** | קובץ תצורה (JSON/YAML) | טעינת חוקי החשבון הסטטיים (למשל, max\_daily\_loss: 1000). | אובייקט Rules זמין לסביבת האימון |
| **2.2.4 רשת הטמעה** | Synergy\_Vector | העברת הווקטור דרך ארכיטקטורת NN (למשל, LSTM) להפחתת ממדיות ולמידת מאפיינים. | Synergy\_Embedding (וקטור דחוס) |
| **2.2.5 סוכן MARL (MAPPO)** | State = [Synergy\_Embedding, Account\_State\_Vector] | הפעלת לוגיקת Actor-Critic לקבלת החלטה. | Trade\_Plan\_Proposal |

#### 3.0 ניסוח הבעיה כלמידת חיזוק (The MDP Formulation)

זוהי ליבת המערכת.

##### 3.1 מרחב המצב (State Space, S)

וקטור נומרי בגודל קבוע, המורכב משני חלקים המחוברים יחדיו (concatenated):

* **3.1.1 Synergy\_Embedding**: הפלט הדחוס של רשת ההטמעה. מייצג את איכות ההזדמנות.
* **3.1.2 Account\_State\_Vector**: וקטור מנורמל (ערכים בין 0 ל-1, או 1- ל-1) המתאר את מצב החשבון. רכיבים לדוגמה:
  + Daily\_Drawdown\_Ratio: (Current\_Daily\_Loss / Max\_Daily\_Loss)
  + Account\_Drawdown\_Ratio: (Current\_Account\_Drawdown / Max\_Trailing\_Drawdown)
  + Is\_In\_Trade\_Flag: דגל בינארי (0 או 1).
  + Open\_Trade\_PnL\_Ratio: (Unrealized\_PnL / Account\_Size)

##### 3.2 מרחב הפעולות (Action Space, A)

פעולה היברידית שה-Actor מייצר:

* **3.2.1 Position\_Size (בדיד):** Action\_1 ∈ {1, 2, 3}. מספר החוזים.
* **3.2.2 SL\_ATR\_Multiplier (רציף):** Action\_2 ∈ [0.5, 3.5]. המרחק ל-SL ככפולה של ה-ATR הנוכחי.
* **3.2.3 RR\_Ratio (רציף):** Action\_3 ∈ [1.0, 5.0]. יחס סיכון/תגמול לקביעת מרחק ה-TP.

##### 3.3 פונקציית התגמול (Reward Function, R)

התגמול ניתן לסוכן בסוף כל עסקה, ומחושב באופן הבא:

Python

def calculate\_reward(trade\_result, account\_history):  
 # תנאי עצירה קריטי: הפרת חוק  
 if trade\_result.caused\_rule\_violation:  
 return -10.0 # עונש כבד וחד משמעי  
  
 # 1. חישוב רכיב התשואה מותאמת הסיכון  
 # חשב את יחס סורטינו של החשבון לפני ואחרי העסקה  
 sortino\_before = calculate\_sortino(account\_history.before\_trade)  
 sortino\_after = calculate\_sortino(account\_history.after\_trade)  
 sortino\_reward = sortino\_after - sortino\_before # תגמל על שיפור, הענש על הרעה  
  
 # 2. רכיב עיצוב תגמול (Reward Shaping) - אופציונלי אך מומלץ  
 shaping\_reward = 0  
 # בונוס קטן על יחס סיכון/תגמול טוב  
 if trade\_result.rr\_ratio > 2.0:  
 shaping\_reward += 0.1  
 # עונש קטן על סיכון יתר בעסקה בודדת (SL רחב מדי)  
 if trade\_result.sl\_atr\_multiplier > 2.5:  
 shaping\_reward -= 0.1  
  
 # 3. שקלול הרכיבים  
 # w\_sortino צריך להיות המשקל הדומיננטי  
 W\_SORTINO = 1.0  
 W\_SHAPING = 0.2  
  
 final\_reward = (W\_SORTINO \* sortino\_reward) + (W\_SHAPING \* shaping\_reward)  
 return final\_reward

#### 4.0 זרימת נתונים ותהליך תפעולי

1. **המתנה:** M-RMS במצב המתנה.
2. **טריגר:** המערכת הגדולה מזהה סינרגיה ושולחת אובייקט Synergy\_Event ל-M-RMS.
3. **איסוף מצב:**
   * רכיב 2.2.1 בונה Synergy\_Vector מהאירוע.
   * רכיב 2.2.2 בונה Account\_State\_Vector עדכני.
4. **הטמעה:** Synergy\_Vector מוזן לרשת 2.2.4 לקבלת Synergy\_Embedding.
5. **החלטה:** וקטור המצב המאוחד S מוזן ל-Actor של סוכן ה-MARL.
6. **הצעה:** ה-Actor מייצר אובייקט Trade\_Plan\_Proposal, לדוגמה: {position\_size: 2, sl\_distance\_points: 15.5, tp\_distance\_points: 38.75}.
7. **שליחה:** ההצעה נשלחת חזרה למערכת הראשית (לרמה 3, שער ההחלטה הסופי) לאישור/דחייה.
8. **לולאת משוב:** אם העסקה בוצעה ונסגרה, trade\_result מועבר חזרה ל-M-RMS לצורך חישוב התגמול ועדכון המדיניות של הסוכן (עדכון משקולות ה-Actor וה-Critic).

#### 5.0 אסטרטגיית אימון, ולידציה ואופטימיזציה

##### 5.1 סביבת אימון (סימולטור)

נדרש סימולטור ייעודי ב-Python שיכלול:

* **Data Feeder:** מספק נתוני OHLCV וערכי אינדיקטורים היסטוריים.
* **Virtual Account Manager:** מחזיק את מצב החשבון (Account\_State\_Vector), ומעדכן אותו לאחר כל עסקה.
* **Execution Engine:** מדמה ביצוע פקודות SL/TP על בסיס נתוני העבר.
* **Rule Engine Integration:** בודק הפרת כללים לאחר כל עסקה.

##### 5.2 פרוטוקול אימון (Curriculum Learning)

* **שלב 1 ("לימוד הבסיס"):** אימון הסוכן עם Position\_Size קבוע ל-1. המטרה היא ללמוד להתאים SL/TP לסינרגיות כדי לייצר סורטינו חיובי.
* **שלב 2 ("ניהול הון"):** פתיחת מרחב הפעולות של Position\_Size. הסוכן כעת לומד מתי להגדיל/להקטין סיכון בהתבסס על מצב החשבון.

##### 5.3 ולידציה

**Walk-Forward Optimization (WFO)** היא חובה. המערכת תאומן על חלון IS (למשל, 12 חודשים) ותיבדק על חלון OOS עוקב (למשל, 3 חודשים), תוך הזזת החלונות קדימה.

##### 5.4 אופטימיזציית היפר-פרמטרים (HPO)

שימוש ב-Optuna על סביבת האימון.

* **פרמטרים לכיוונון:** שיעורי למידה של ה-Actor/Critic, גודל רשתות ה-NN, פרמטרי אלגוריתם MAPPO, וחשוב מכל - **משקלי פונקציית התגמול (W\_SORTINO, W\_SHAPING)**.
* **פונקציית מטרה ל-Optuna:** יחס סורטינו ממוצע על פני כל מקטעי ה-OOS של ה-WFO.

#### 6.0 אינטגרציה עם המערכת הראשית

* **ממשק קלט (API In):** המערכת תחשוף פונקציה/Endpoint propose\_risk\_plan(synergy\_event).
* **ממשק פלט (API Out):** הפונקציה תחזיר אובייקט Trade\_Plan\_Proposal.
* **תפקיד במערכת הגדולה:** ה-M-RMS מתפקד כ"יועץ סיכונים" (Risk Consultant) למערכת ההחלטה הסופית. הוא אינו מבצע פקודות בעצמו.

#### 7.0 דרישות טכנולוגיות ומימוש

* **שפה:** Python 3.9+
* **למידת מכונה:** PyTorch, Ray RLlib (למימוש MAPPO)
* **אופטימיזציה:** Optuna
* **מניפולציית נתונים:** Pandas, NumPy
* **מקורות נתונים:** חיבוריות ל-API של Databento, Rithmic, או dxFeed.

תוכנית זו מספקת מתווה מפורט ובר-ביצוע לבניית אחד הרכיבים המתוחכמים והחשובים ביותר בארכיטקטורה הכוללת שלך.