

7 למידת MCP דרך תרגיל הליגה

תרגיל ליגת זוגי/אי-זוגי אינו רק תרגיל תכנות. הוא מהוות מודל פדגוגי שלם להבנת פרוטוקול MCP ועקרונות סוכני AI. בפרק זה נסביר כיצד התרגיל מלמד את עקרונות היסוד של סוכני AI ופרוטוקול MCP.

7.1 השחקן כסוכן AI

7.1.1 האם סוכן השחקן הוא סוכן AI?

השאלה הראשונה שיש לשאול היא: האם סוכן השחקן (Player Agent) בliga הוא באמת סוכן AI? התשובה היא חד-משמעות: כן.

סוכן AI מוגדר כישות המקיים אינטראקציה עם הסביבה על מנת להשיג מטרות מוגדרות [1]. להבדיל מתוכנית רגילה המבצעת הוראות קבועות מראש, סוכן AI הוא תוכנה אוטונומית שמקבלת מידע מהסביבה, מעבדת אותו, ומחליטה בעצמה מה לבצע על בסיס המצב הנוכחי.

7.1.2 ארבעת המאפיינים של סוכן AI

נבחן את סוכן השחקן בliga לאור ארבעת המאפיינים העיקריים של סוכן AI:

1. **אוטונומיות** – הסוכן פועל באופן עצמאי. בהקשר המשחק, סוכן השחקן מחליט באופן אוטונומי איזו אסטרטגיה לבחור: "זוגי" (even) או "אי-זוגי" (odd). אף אחד לא אומר לו מה לבחור.

2. **תפיסה** – הסוכן קולט מידע מהסביבה. השחקן קולט הודעות הזמן למשחק מושכים (GAME_OVER) מהשופט ומנהל הליגה.

3. **פעולה** – הסוכן משפייע על הסביבה. השחקן מבצע פעולות על ידי שליחת בחירות (GAME_INVITATION), בקשות לבחירת זוגיות (GAME_JOIN_CALL), ותוצאות משחקים (GAME_RESPONSE) (CHOOSE_PARITY_RESPONSE).

4. **תכליתיות** – יש לו מטרה מוגדרת. מטרתו היא לשחק, לנצח משחקים ולעדכן את מצבו הפנימי, כגון היסטוריית ניצחונות והפסדים.

סוכן השחקן יכול אף להשתמש במודל שפה גדול (LLM) כדי לבחור את האסטרטגיה הטובה ביותר. בכך הוא מגדים "חשיבה" או "הסקת מסקנות" לפני ביצוע הפעולה.

7.2 השחקן בארכיטקטורת MCP

7.2.1 שרת או ל��וח?

בארQUITטורת ליגת זוגי/אי-זוגי, השחקן הוא בעיקרו **שרת MCP**. שרת MCP הוא הרכיב שהושך יכולות ושירותים, המכונים "כלי" (Tools), "כליים" (Tools), "משאבים" (Resources) או "הנחיות" (Prompts). השירות מוגדר כתהליך נפרד הפועל על פורט מוגדר ומספק "שער" לעולם החיצון [2].

סוכן השחקן נדרש למשר MCP HTTP שמקבל בקשות POST בנתיב `mcp`. הכלים שהוא חושף נקראים באמצעות פרוטוקול JSON-RPC 2.0. הכלים שהשחקן מחויב למשר כוללים:

- `handle_game_invitation` – טיפול בהזמנה למשחק.
- `choose_parity` – בחירת "זוגי" או "אי-זוגי".
- `notify_match_result` – קבלת הודעה על תוצאת המשחק.

7.2.2 היחסים מול השופט ומנהל הליגה

בהתאם שהשחקן הוא שרת, מי שקורא לשירותיו הוא הלקוח (Client). במערכת הליגה, השופט (Referee) ומנהל הליגה (League Manager) הם שפועלים כלוקחות או אורקסטרטורים (Orchestrators).

השופט הוא זה שיזכר את בקשת JSON-RPC הקוראת לכל `choose_parity` של השחקן. כאשר השופט רוצה לאסוף בחירות מהשחקנים, הוא שולח בקשת `CALL_PARITY_CALL` לכל שחקן.

לסייע: אף על פי שסוכן השחקן הוא סוכן AI אוטונומי, מבחינות מימוש פרוטוקול MCP, הוא ממלא את תפקיד השירות המציע יכולות לאורקסטרטורים המרכזיים.

7.3 השופט ומנהל הליגה כסוכני AI

7.3.1 סוכנים בדרجة גבוהה

גם השופט ומנהל הליגה מוגדרים כסוכני AI. הם עומדים אותם ארבעה מאפיינים: הסוכנים הללו אינם פסיביים. הם מנהלים את המערכת כולה בהתאם לכלים ומטרות קבועות. זהה מוחות האוטונומיות והתכליתיות של סוכן AI.

7.3.2 שירותי MCP שפועלים גם בכלוקות

שני הסוכנים הללו מוגדרים כשירות MCP:

- מנהל הליגה פועל כשרת MCP בפורט 8000. הוא מימוש כלים כמו `register_referee`, `report_match_result` ו-`register_player`.
- השופט פועל כשרת MCP בפורט 8001. הוא מימוש כלים כמו `start_match` ו-`collect_choices`.

הערה חשובה: השופט ומנהל הליגה, אף שהם מוגדרים כשירותים, חייבים לפעול גם בכלוקות MCP כדי למלא את תפקידם המרכזי. לדוגמה:

טבלה 14: מאפייני סוכן AI עבור השופט ומנהל הליגה

מאפיין	מנהל ליגת	שופט
תכליתיות	ניהול הליגה כולה, רישום שופטים וশחקנים, ייצירת לוח משחקים, חישוב דירוג	רישום למנהל הליגה, ניהול משחק בודד, אימות חוקיות מהלכים, קביעת מנצח
אוטונומיות	פועל באופן עצמאי לרישום שופטים ולקביעת סבבי משחק	נרשם באופן עצמאי ליגת ומנהל את שלבי המשחק
תפיסה	קולט בקשות רישום משופטים וশחקנים, תוצאות מהשופטים	קולט אישורי הגעה, בחירות זוגיות/אי-זוגיות מהשחקנים
פעולה	מאשר רישום שופטים וশחקנים, שלוח הכרזות מחזור, מעدقן טבלאות דירוג	שלוח בקשת רישום ליגת, שלוח הזמנות מושך, בקשות בחירה, מדוח תוצאות

- השופט חייב לפעול(Clk) כדי להירשם למנהל הליגה (REGISTER_RE_AI). (QUEST)
- השופט חייב לפעול(Clk) כדי לקרוא לכלי choose_parity של סוכן השחקן.
- מנהל הליגה חייב לפעול(Clk) כדי לשלווח את הכרזות המוחזר לסטטוס השחקנים. במערכת זו, השירותים המרכזיים הם למעשה קוחות-אורקסטרטורים כאשר הם צריכים להניע פעולה אצל השירותים השחקנים.

7.4 היפוך התפקידים: תובנה מרכזית

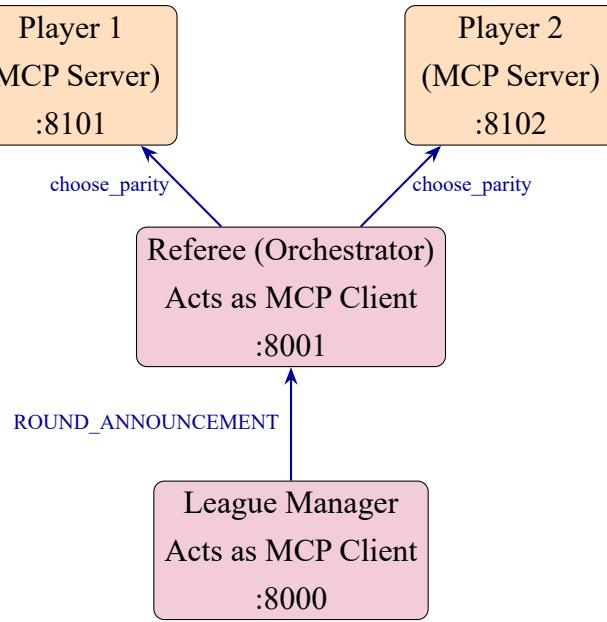
7.4.1 הפרדיגמה המסורתית

בארכיטקטורת שרת-לקוח הטיפוסית, הלקוח הוא הרכיב האקטיבי ששולח בקשות, והשרת הוא הרכיב הפסיבי שמחכה לבקשתו. בliget ה-AI, מתקיים היפוך תפקידים יצרתי.

7.4.2 היפוך התפקידים בליגה

השחקן (הסוכן האוטונומי) הוא השירות: למرات ששהשחקן הוא הישות האוטונומית שצרכיה לבצע פעולה, הוא נדרש לחושף את יכולותיו כשרת MCP.

השופט ומנהל הליגה (אורקסטרטורים) הם הלקוחות: השופט הוא האורקסטרטור שפועל(Clk) MCP וקורא לכלי choose_parity של השחקן כדי להניע את המהלך הבא במשחק.



7.5 עקרון הפרדת השכבות

7.5.1 שלוש שכבות נפרדות

פרוטוקול MCP מאפשר הפרדה ברורה בין התפקידים:

1. **שכבה הליגה** (מנוהלת על ידי מנהל הליגה) – גיוס שחקנים, לוח משחקים- (Round-robin, וטבלת דירוג).
2. **שכבה השיפוט** (מנוהלת על ידי השופט) – ניהול משחק בודד וAIMOS מהלכים.
3. **שכבה חוקי המשחק** (מנוהלת על ידי מודול נפרד) – הלוגיקה הספציפית למשחק זוגי/אי-זוגני.

7.5.2 היתרון של ההפרדה

השחקן, בכך שהוא חושף ממשק MCP סטנדרטי (JSON-RPC 2.0 על גבי HTTP), מאפשר ליגא להישאר אגנוזטיבית לשפת הפיתוח או לאסטרטגייה הפנימית שלו.

זהו פתרון לביעית הrogrammatical שבה לכל סוכן ולכל מודול נדרשה בעבר אינטגרציה ייחודית. פרוטוקול MCP פותר זאת על ידי יצירת ממשק אוניברסלי [2].

כאשר השחקן מקבל בקשה כמו CHOOSE_PARITY_CALL, הנתונים מגיעים במבנה JSON קבוע. השחקן מגיב עם CHOOSE_PARITY_RESPONSE, גם כן במבנה קבוע. זה מבטיח כי כל סוכן, ללא קשר לאופן שבו הוא מחשב את הנתונים, יכול לתקשר באופן עקבי עם כל אורקסטרטור אחר המכבד את הפרוטוקול.

7.6 תפקיד ה-LLM בסוכן השירות

7.6.1 הדילמה

עליה שאלת מעניינית: מצד אחד, השחקן מוגדר כשרת MCP שחושף יכולות. מצד שני, הוא מתואר כסוכן AI אוטונומי שיכל להשתמש ב-LLM כ"מוח" לבחירת אסטרטגיה. בהגדירות מסורתיות, שרת אינו מפעיל "מוח" אלא ממלא בקשה.

7.6.2 הפתרון: הפרדת תפקידיים

הפתרון טמון בהבנה שתפקיד MCP (לקוח/שרת) ומרכיבי ה-AI (מוח/כליים) הם מושגים נפרדים אך משלימים.

הסוכן הוא גם שרת וגם لكוח (בפועל): כל אחד מהסוכנים הוא בפועל גם שרת וגם لكוח. תפקיד השירות נדרש לכל סוכן המארח את עצמו כדי לאפשר לסוכנים אחרים לקרוא לכליו. תפקיד הל��ון נדרש לכל סוכן שצורך ליזום אינטראקציה.

ה-LLM כרכיב פנימי: מודל שפה גדול הוא "מוח" של סוכן AI. אם סוכן השחקן משתמש שירות MCP, ה-LLM הוא פשוט רכיב פנימי בתוך לולאת הסוכן הכללית.

כאשר השירות מקבל בקשה choose_parity:

1. שכבת ה-MCP (השרת) קולטת את הבקשת.
2. הלוגיקה הפנימית של הסוכן (ה-LLM או אסטרטגיה אחרת) מופעלת לקביעת הבחירה.
3. שכבת ה-MCP (השרת) שולחת את התגובה בחזרה.

ה-LLM הוא "הבינה" של השירות, והוא אינו מפרק את מודל השירות-לקוח. הרעיון המרכזי ב-MCP הוא להבטיח שגם כאשר ה"מוח" נמצא בתוך השירות, התקשורות החיצונית תישאר סטנדרטית באמצעות JSON-RPC.

7.6.3 אנלוגיה: תחנת שירותים לקוחות

ניתן לדמיין את הארכיטקטורה כתחנת שירותים לקוחות:

- **MCP (פרוטוקול)** – הוא הטלפון והשפה שבה מדובר (HTTP).
 - **השחקן (שרת)** – הוא משרד השירות עם קו טלפון מסוילו.
 - **האסטרטגיה/LLM (מוח)** – הוא היועץ החכם היושב בתוך המשרד, שמקבל את השיחה, מחשב את המענה, ומכתיב לשכבת ה-MCP איך תשובה לשולח בחזרה.
- הכליים הפנימיים (ה-LLM והלוגיקה) אינם חשובים לשירות פרוטוקול MCP, אלא משרתים את הכלים הציבוריים שהSOC חושף, כגון choose_parity.

7.7 תפקיד האורקסטרטור

7.7.1 מנהל הליגה – הארכיטקט

מנהל הליגה הוא סוכן ה-AI בדרגה הגבוהה ביותר מבחינה אסטרטגיית, המנהל את שכבות הליגה. הוא אינו מעורב בחוקי המשחק עצמו, אלא בניהול הכללי: לוח משחקים וטבלת דירוג.

יתרון הפרדה: אם הליגה תרצה להחליף את המשחק מזוגי/אי-זוגי לאיקס-עיגול-Tic-Toe, מנהל הליגה כמעט ולא ישתנה. זהה הדגמה מושלמת של עקרון הפרדת התפקידים שמקדם MCP.

7.7.2 השופט – המימוש הדינמי

השופט מגלה את שכבות השיפוט. הוא אינו יודע את כללי המשחק (شمוטופלים על ידי מודול נפרד), אלא הוא אחראי על ניהול השיחה Conversation Lifecycle בין השחקנים. השופט מוודא שהשחקנים עומדים במועד התגובה Deadlines. הוא זה שמבצע את לולאת הסוכן החיצונית עבור השחקנים – הוא קורא לכלי choose_parity שלהם ובכך מניע את הפעולה האוטונומית של השחקן.

MCP מאפשר את חלוקת התפקידים הדרושה: השופט ומנהל הליגה אחראים על ה-"איך" (הפרוטוקול והתקשורת), בעוד השחקנים אחראים על ה-"מה" (הסטרטגיה והתוכן).

7.8 מה התרגיל מלמד

7.8.1 עקרונות יסוד של סוכני AI

התרגיל מלמד את ארבעת המאפיינים של סוכן AI באופן מעשי:

- אוטונומיות – השחקן מחליט בעצמו.
- תפיסה – השחקן קולט הودעות מהמערכת.
- פעולה – השחקן שולח תשובות.
- תכליות – השחקן שואף לנצח.

7.8.2 עקרונות יסוד של MCP

התרגיל מלמד את עקרונות הליבה של פרוטוקול MCP:

1. **משחק סטנדרטי** – כל סוכן חושף כלים דרך JSON-RPC 2.0.
2. **הפרדת תפקידים** – שכבות הליגה, שכבות השיפוט, ושכבת חוקי המשחק.
3. **אגנוסטיות לשפה** – ניתן למשתמש סוכן בכל שפת תכנות.
4. **תקשורת דרך אורקסטטור** – סוכנים לא מדברים ישירות, אלא דרך השופט או מנהל הליגה.
5. **רישום סוכנים** – גם שופטים וגם שחקנים רשומים למנהל הליגה לפני תחילת המשחקים.

7.8.3 חווית הלמידה

בסיום התרגיל, הסטודנט יבין:

- כיצד סוכן AI מתקשר עם סוכנים אחרים.
- כיצד לבנות שרת MCP פשוט.
- מהי משמעות "הכליים" (Tools) בפרוטוקול MCP.
- כיצד אורקסטרטור מנהל אינטראקציה בין סוכנים.
- מדוע הפרדת שכבות חשובה לתכנון מערכות AI.

7.9 סיכום

תרגיל ליגת זוגי/אי-זוגי מהויה מודל פדגוגי מושלם להבנת פרוטוקול MCP וסוכני AI. המשחק פשוט מאפשר להתמקד בעקרונות הארכיטקטוניים מביי להסתבך בלוגיקה מורכבת.

הסטודנט לומד שסוכן AI יכול להיות גם שרת MCP – היפוך תפקידים יצירתי המאפשר לאורקסטרטור לקרוא לסוכנים ולהניע את פעולתם. ההפרדה לשכבות מבטיחה שנייתן להחליף את משחק הליגה בעתיד מביי לשנות את הפרוטוקול הכללי.

לפרטים נוספים על פרוטוקול MCP, ראו את הספר "סוכני AI עם MCP" [1] ואת התיעוד הרשמי של Anthropic [2].