

7 למידת MCP דרך תרגיל הליגה

תרגיל ליגת זוגי/אי-זוגי אינו רק תרגיל תכנות. הוא מהווה 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) בליגה הוא באמת סוכן AI? התשובה היא חד-משמעית: כן.

סוכן AI מוגדר כישות המקיימת 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 בליגה 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 הזמנה למשחק (GAME_INVITATION), בקשות לבחירת זוגיות (CHOOSE_PARITY_CALL), ותוצאות משחקים (GAME_OVER) מהשופט ומנהל הליגה.
 3. **פעולה** – הסוכן משפיע על הסביבה. השחקן מבצע פעולות על ידי שליחת בחירות (CHOOSE_PARITY_RESPONSE) ואישורי הגעה (GAME_JOIN_ACK) למשחקים.
 4. **תכליתיות** – יש לו מטרה מוגדרת. מטרתו היא לשחק, לנצח במשחקים 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 מדגים "חשיבה" או "הסקת מסקנות" לפני ביצוע הפעולה.

7.2 השחקן בארכיטקטורת MCP

7.2.1 שרת או לקוח?

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

סוכן השחקן נדרש לממש שרת HTTP שמקבל בקשות POST בנתיב /mcp. הכלים שהוא חושף נקראים באמצעות פרוטוקול JSON-RPC 2.0. הכלים שהשחקן מחויב לממש 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

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

השופט הוא זה שיוצר את בקשת ה-JSON-RPC הקוראת לכלי choose_parity של השחקן. כאשר השופט רוצה לאסוף בחירות מהשחקנים, הוא שולח בקשת CHOOSE_PARITY_CALL לכל שחקן.

לסיכום: אף על פי שסוכן השחקן הוא סוכן AI אוטונומי, מבחינת מימוש פרוטוקול MCP, הוא ממלא את תפקיד השרת המציע יכולות לאורקסטרטורים המרכזיים.

7.3 השופט ומנהל הליגה כסוכני AI

7.3.1 סוכנים בדרגה גבוהה

גם השופט ומנהל הליגה מוגדרים כסוכני AI. הם עומדים באותם ארבעה מאפיינים: הסוכנים הללו אינם פסיביים. הם מנהלים את המערכת כולה בהתאם לכללים ומטרות קבועות. זוהי מהות האוטונומיות והתכליתיות של סוכן AI.

7.3.2 שרתי MCP שפועלים גם כלקוחות

שני הסוכנים הללו מוגדרים כשרתי MCP:

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

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

טבלה 14: מאפייני סוכן IA עבור השופט ומנהל הליגה

מאפיין	מנהל ליגה	שופט
תכליתיות	ניהול הליגה כולה, רישום שופטים ושחקנים, יצירת 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 לליגה ומנהל את שלבי המשחק
תפיסה	קולט בקשות רישום משופטים ומשחקנים, דוחות תוצאות מהשופטים	קולט אישורי הגעה, בחירות זוגיות/אי-זוגיות מהשחקנים
פעולה	מאשר רישום שופטים ושחקנים, שולח הכרזות מחזור, מעדכן טבלאות דירוג	שולח בקשת רישום לליגה, שולח הזמנות משחק, בקשות בחירה, מדווח תוצאות

- השופט חייב לפעול כלקוח כדי להירשם למנהל הליגה (REFEREE_REGISTER_RE-QUEST).

- השופט חייב לפעול כלקוח כדי לקרוא לכלי choose_parity של סוכן השחקן.
 - מנהל הליגה חייב לפעול כלקוח כדי לשלוח את הכרזות המחזור לסוכני השחקנים.
 במערכת זו, השרתים המרכזיים הם למעשה לקוחות-אורקסטרטורים כאשר הם צריכים להניע פעולה אצל שרתי השחקנים.

7.4 היפוך התפקידים: תובנה מרכזית

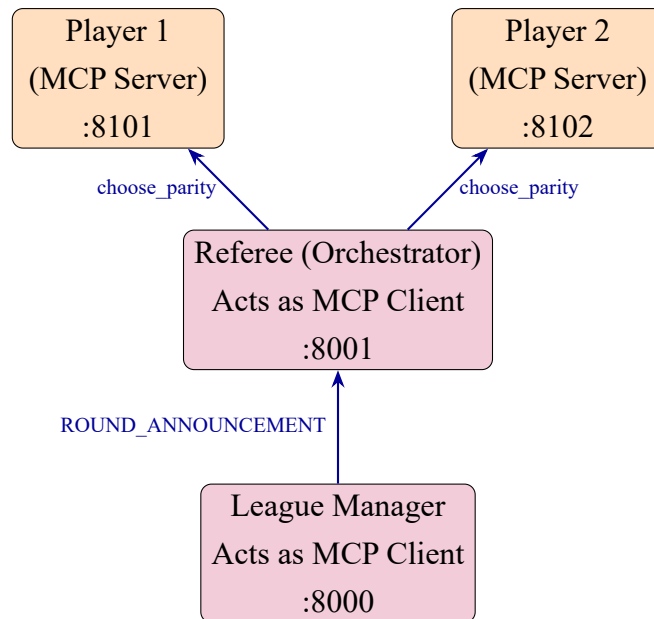
7.4.1 הפרדיגמה המסורתית

בארכיטקטורת שרת-לקוח הטיפוסית, הלקוח הוא הרכיב האקטיבי ששולח בקשות, והשרת הוא הרכיב הפסיבי שמחכה לבקשות. בליגת ה-AI, מתקיים היפוך תפקידים יצירתי.

7.4.2 היפוך התפקידים בליגה

השחקן (הסוכן האוטונומי) הוא השרת: למרות שהשחקן הוא הישות האוטונומית שצריכה לבצע פעולה, הוא נדרש לחשוף את יכולותיו כשרת MCP.

השופט ומנהל הליגה (האורקסטרטורים) הם הלקוחות: השופט הוא האורקסטרטור שפועל כלקוח 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. **שכבת הליגה** (מנוהלת על ידי מנהל הליגה) – גיוס שחקנים, לוח משחקים (Round-Robin), וטבלת דירוג.
2. **שכבת השיפוט** (מנוהלת על ידי השופט) – ניהול משחק בודד ואימות מהלכים.
3. **שכבת חוקי המשחק** (מנוהלת על ידי מודול נפרד) – הלוגיקה הספציפית למשחק זוגי/אי-זוגי.

7.5.2 היתרון של ההפרדה

השחקן, בכך שהוא חושף ממשק MCP סטנדרטי (JSON-RPC 2.0 על גבי HTTP), מאפשר לליגה להישאר אגנוסטית לשפת הפיתוח או לאסטרטגיה הפנימית שלו. זהו פתרון לבעיית הפרגמנטציה שבה לכל סוכן ולכל מודל 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 השרת

7.6.1 הדילמה

עולה שאלה מעניינת: מצד אחד, השחקן מוגדר כשרת MCP שחושף יכולות. מצד שני, הוא מתואר כסוכן AI אוטונומי שיכול להשתמש ב-LLM כ"מוח" לבחירת אסטרטגיה. בהגדרות מסורתיות, שרת אינו מפעיל "מוח" אלא ממלא בקשה.

7.6.2 הפתרון: הפרדת תפקידים

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

הסוכן הוא גם שרת וגם לקוח (בפועל): כל אחד מהסוכנים הוא בפועל גם שרת וגם לקוח. תפקיד השרת נדרש לכל סוכן המארח את עצמו כדי לאפשר לסוכנים אחרים לקרוא לכליו. תפקיד הלקוח נדרש לכל סוכן שצריך ליזום אינטראקציה.

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

כאשר השרת מקבל בקשת `choose_parity`:

1. שכבת ה-MCP (השרת) קולטת את הבקשה.
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 השרת, והוא אינו מפר את מודל שרת-לקוח. הרעיון המרכזי ב-MCP הוא להבטיח שגם כאשר ה"מוח" נמצא בתוך השרת, התקשורת החיצונית תישאר סטנדרטית באמצעות JSON-RPC.

7.6.3 אנלוגיה: תחנת שירות לקוחות

ניתן לדמיין את הארכיטקטורה כתחנת שירות לקוחות:

- MCP (פרוטוקול) – הוא הטלפון והשפה שבה מדברים (JSON-RPC על HTTP).
 - השחקן (שרת) – הוא משרד השירות עם קו טלפון משלו.
 - האסטרטגיה/LLM (המוח) – הוא היועץ החכם היושב בתוך המשרד, שמקבל את השיחה, מחשב את המענה, ומכתיב לשכבת ה-MCP איזו תשובה לשלוח בחזרה.
- הכלים הפנימיים (ה-LLM והלוגיקה) אינם חשופים ישירות לפרוטוקול MCP, אלא משרתים את הכלים הציבוריים שהסוכן חושף, כגון `choose_parity`.

7.7 תפקיד האורקסטרטור

7.7.1 מנהל הליגה – הארכיטקט

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

יתרון ההפרדה: אם הליגה תרצה להחליף את המשחק מזוגי/אי-זוגי לאיקס-עיגול (Tic-Tac-Toe), מנהל הליגה כמעט ולא ישתנה. זוהי הדגמה מושלמת של עקרון הפרדת התפקידים שמקדם ה-MCP.

7.7.2 השופט – המיישם הדינמי

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

MCP מאפשר את חלוקת התפקידים הברורה: השופט ומנהל הליגה אחראים על ה"איך" (הפרוטוקול והתקשורת), בעוד השחקנים אחראים על ה"מה" (האסטרטגיה 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 הודעות מהמערכת.
- פעולה – השחקן שולח תגובות.
- תכליתיות – השחקן שואף לנצח.

7.8.2 עקרונות יסוד של MCP

התרגיל מלמד את עקרונות הליבה של פרוטוקול MCP:

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

7.8.3 חוויית הלמידה

בסיום התרגיל, הסטודנט יבין:

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

7.9 סיכום

תרגיל ליגת זוגי/אי-זוגי מהווה מודל פדגוגי מושלם להבנת פרוטוקול MCP וסוכני AI. המשחק הפשוט מאפשר להתמקד בעקרונות הארכיטקטוניים מבלי 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 שניתן להחליף את משחק הליגה בעתיד מבלי לשנות את הפרוטוקול הכללי.

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