ארבע בשורה

פרוייקט בינה מלאכותית בפרולוג

מגיש: ערן קמפף

הגדרת העבודה והדרישות

מטרת התוכנית לאפשר לשחקן לשחק את המשחק "ארבע בשורה" נגד המחשב. הדרישות מהתכנית הן:

1. **ממשק נוח למשתמש:** התכנית צריכה לתת למשתמש הסבר ברור וקריא בזמן הריצה. על הממשק לאפשר למשתמש לשחק ולבצע פעולות שונות בנוחות. כמו כן על הממשק לאפשר לבצע פעולות אלמנטריות כמו יציאה, קבלת עזרה וכדומה.
2. **טפול בשגיאות:** התכנית לא יכולה להניח שהקלט מהמשתמש תקין ואליה לטפל בקלט שגוי.
3. התכנית תגדיר לוח בגודל NxN ותגביל את N לתחום הרצוי, **התכנית עצמה תהיה כללית,** כלומר החוקים יעבדו לכל מספר N.

מימוש התכנית: התכנית ממושת תוך שימוש בפרולוג של AMZI (www.amzi.com)

תיאור המשחק "ארבע בשורה"

לוח המשחק הוא למשל בגודל 8x8. הכוונה המוחשית היא לשמונה עמודים שעליהם ניתן להשחיל טבעות. השחקן והמחשב, כל אחד בתורו משחיל טבעת על עמוד לפי בחירתו. הטבעת "נופלת" למטה עד כמה שניתן. המטרה היא ליצור שורה של ארבע טבעות: במאוזן, במאונך או באלכסון.

תיאור מבנה התכנית והאלגורתמים בהם היא משתמשת

התכנית מורכבת משני חלקים:

- חלק אחד אחראי על המשחק עצמו ועל הבינה המלאכותית. חלק זה משתמש בחוקים שונים כדי לבצע מניפולציות על לוח המשחק (ביצוע המהלכים) ושיטות חיפוש על מנת לבצע את מהלכי המחשב.
- החלק השני אחראי על הקלט והפלט. חלק זה משתמש בפונקציות שונות להגדרת ממשק פקודה נוח למשתמש.

חלק ראשון: הבינה המלאכותית

החוק המרכזי האחראי לפעולת הבינה המלאכותית הוא think(Pos) המקבל את מצב הלוח הנוכחי Pos ובודק איזה מהלך יעיל ניתו לבצע.
לחוק 6 חלקים כשכל אחד מהם מייצג אסטרטגיה משחק כלשהיא לפי סדר העדיפות שלה (לדוגמא ביצוע מהלך מנצח עדיף על חסימת היריב).

1. **בודק תבוסה.** דבר ראשון בודק המחשב אם הוא הובס ע״י השחקן, במקרה זה יסתיים המשחק.
2. **מהלך התקפי (נסיון לנצח).** המחשב בודק אם הוא יכול לבצע מהלך מנצח – מהלוח שיגרום לו לנצח במשחק, או מהלך בדרך לניצחון – בדרך להשלמת רצף.
3. **מהלך הגנתי.** בחלק זה מבצע המחשב מהלך הגנתי – חסימת היריב או מהלך רגיל (מהלך שמטרתו לפתוח אפשרויות להתקפה). חלק זה שומר את המהלך הטוב האחרון שהוא מצא כחוק דינמי – last_good_move(Pos).
4. **ביצוע המהלך הטוב האחרון שנמצע.** אם מהלך כלשהוא נמצע בחלק הקודם אז הוא יבוצע.
5. **מהלך בלית ברירה.** המחשב לא מצא אף מהלך ״טוב״ ולכן מבצע מהלך סתמי.

אסטרטגיה 1: בדיקת תבוסה

המחשב בודק ע״י שימוש בחוק victory(Player, Pos) אם בלוח הנוכחי יש רצף של השחקן. אם כן, מסתיים המשחק.

אסטרטגיה 2: מהלך התקפי

המחשב מנסה לבצע מהלך מנצח. החוק שאחראי על ביצוע חיפוש המהלך הוא try_to_win(Pos), חוק זה מבצע מהלך אפשרי ואז בודק אותו ע״י החוק is_it_win(Pos), כלומר הוא עובר על כל הבנים של הלוח הנוכחי בעץ המשחק ובודק כל בן ע״י החוק is_it_win(Pos).
לחוק is_it_win(Pos) שני חלקים:

1. בודק אם בלוח הנבדק Pos (אחד הבנים של לוח המשחק הנוכחי בעץ המשחק) יש למחשב רצף. אם כן יבצע במחשב את המהלך והמשחק מסתיים בנצחון המחשב.
2. המחשב מחפש מהלך שיעזור לו לנצח אבל לא יתן ליריב אפשרות לבצע מהלך מנצח. כדי לחפש מהלך שכזה משתמש is_it_win/1 בחוק

win(Player, Pos, Deep) כשעומק הסריקה נקבע ע״י העובדה win_deep(X).
החוק win סורק עד לעומק המוגדר ומנסה להגיע לניצחון, על על ירידה בעץ המשחק (ביצוע מהלך) הוא בודק שהיריב לא חכול להגיב בתגובה שתביים לניצחונו.

אסטרטגיה 3,4: מהלך הגנתי

המחשב מנסה לבצע מהלך הגנתי. החוק שאחראי על ביצוע חיפוש המהלך הוא try_nolose(Pos) , חוק זה מבצע מהלך ובודק אותו ע״י הסריקה nolose(Player, Pos, Deep) כשהעומק נקבע ע״י העובדה nolose_deep(X).
הסריקה של nolose היא בעצם הפוכה ל-win ומהווה בעצם את סריקת ה-win של השחקן היריב, המחשב בודק איזה צעדים יכול לנסות היריב כדי לנצח.
המחשב מקבל צעד כצעד ״טוב״ (צעד שניצן לבצעו) אם החוק nolose לא הצליח להגיע לנצחון היריב לאחר צעד זה. הצעד ה״טוב״ שהתקבל נשמר כעובדה דינמית: last_good_move(Pos). במידה ויבוצע backtracing והמחשב יחפש וימצא צעד אחר תעודכן עובדה זו בפרטי הצעד ה״טוב״ החדש.
גורם הרנדומיזציה: כדי להפוך את משחק המחשב ליותר מגוון ולשפר את ביצועי המשחק והזמן, מוגדר סיסוי מסויים שהמחשב יקבל צעד מסויים ולא ימשיך ב-backtracing כדי למצוא עוד צעדים, סיכוי זה הוא 1/(X-1) כש-X נקבע ע״י העובדה random_factor(X).

חלק שני: קלט/פלט

בקבלת הקלט מהמשתמש רציתי שהמשתמש יוכל להשתמש בפקודות בשפה אנגלי פשוטה כמו 'go 2' , 'goto 2' וכדומה ולכן היתי צריך לפתח מערכת שתקבל שורת קלט ותנתח אותה תחבירית לפקודה ופרמטרים של פקודה.

לשם כך הגדרתי את חוקי התחביר של הפקודות ע"י החוק command, המגדיר ארבעה סוגי פקודה:

1. פקודה המורכבת משם פקודה ופרמטר אחד (לדוגמא: "go 1").
2. פקודה המורכבת משם פקודה ושני פרמטרים (לדוגמא: "set random_factor 2") .
3. פקודה המורכבת משם פקודה בלבד (לדוגמא: "quit").
4. פקודה מיוחדת לפעולת ביצוע מהלך המקבלת פרמטר בלבד (את מספר העמודה בה מבצע השחקן את התזוזה).

כל פקודה מורכבת מפועל (verb) ומשם או שמות עצם.

הפועל מוגדר ע"י verb(Type, Verb) כשלכל סוג פועל מוגדרים השמות שמבטאים אותו. לדוגמא, את הפועל המגדיר מהלך של השחקן ניתן לבטא ע"י מספר מילים המוגדרות בתכנה: go, goto, move, move to, ועוד...

ביטוי שם העצם מוגדר בדומה לפועל ע"י nounphrase(Type, Noun) המגדיר את שם העצם שיתקבל לפי הסוג של ביטוי שם העצם.

בנוסף להגדרות התחביר יש צורך בחוק שיקרא את הקלט מהמשתמש ויחזיר אותו בצורה של רשימת מילים כך שיהיה ניתן לנתח את המשפט.

החוק המבצע זאת הוא read_list(L) שמראה שורת פקודה, מקבל את שורת הפקודה מהמשתמש ומחזיר שורה זו כרשימת מילים L.

הפקודות המוגדרות ע"י המערכת:

1. **ביצוע מהלך**. הפקודות 'go','go to','move','move to','g' ולאחריהן ערך העמוד יבצעו את המהלך על העמוד המבוקש. ניתן גם להקיש רק את ערך העמוד.
2. **מסך עזרה.** ניתן לראות את מסך העזרה ע"י הפקודה 'help'.
3. **הצגת הלוח.** ניתן להציג את לוח המשחק ע"י הפקודות 'show','show board','pos','board'.
4. **אתחול המשחק.** ניתן לאתחל את המשחק ע"י הפקודה 'reset'.
5. **יציאה.** ניתן לצאת מהמשחק ע"י הפקודות 'quit','exit','end','bye'.
6. **שינוי ערך של קבועי המשחק.** ניתן לשנות את ערכם של שלושת קבעי המשחק תוך כדי הריצה ע"י 'set <game var name> <value>'.
   קבועי המשחק הם:
   - win_deep – קובע כמה עמוק תתבצע הסריקה ההתקפית (ראה תאור בחלק הראשון).
   - molose_deep - קובע כמה עמוק תתבצע הסריקה ההגנתית (ראה תאור בחלק הראשון).
   - random_factor – קובע את הסיכוי של המחשב לקבל את המהלך ה"טוב" הנוכחי ולא להמשיך את החיפוש (ראה תאור בחלק הראשון).

ביבליוגרפיה

1. PROLOG – Programming for Artificial Language של Ivan Bratko.
2. Adventure in Prolog Tutorial המצורף ל-AMZI – מדריך לשפת פרולוD ותכנות משחק בסגנון הקווסט הישן – מדריך לבניית מנגנון קלט Natural Language Processing.