تمرین تئوری + گزارش تمرین عملی
HW6

بکتاش انصاری
99521082

برای اثبات روش inference logic دو روش وجود دارد.

1. Model-checking
2. Theorem-proving

Model-Checking :

در این روش فرض میکنیم میخواهیم یک جمله منطقی را اثبات کنیم.

برای مثال در مسئله مورد نظر جمله ای را در نظر میگیریم.

مثلا در بازی Wumpus world جمله مورد نظر این است :

در خانه (2,2) چاه وجود دارد.

در این روش برای اثبات تمامی حالت های مختلف جهان بازی را در نظر میگیریم و به ازای حالات مختلف چک میکنیم که ایا جمله ما با توجه به knowledge base های ما درست است یا نه.

یا به طور دقیق تری آیا KB ما جمله مورد نظر ما را entails میکند یا نه.

که در این مورد باید تک تک حالاتی که چاه در خانه 2,2 است را چک کنیم تا ببینیم با KB ما تناقض دارد یا نه. که اگر ندارد میتوان نتیجه گرفت که درست است.

این روش برای جهان های محدود و کوچک مناسب است ولی در حالتی که جهان ما بزرگ یا نامحدود است و تعداد متغیر های ما بسیار زیاد هستند. به صرفه نیست.

: Theroem-proving

در این روش ما برای اثبات جمله خود از منطق ریاضی استفاده میکنیم. بجای آنکه تمامی حالات جدول را چک کنیم از چند عبارت منطقی استفاده میکنیم تا نتیجه جدیدی بگیریم.

در این روش بجای جدول درستی از قواعد منطقی استفاده میکنیم.

به طوری که قواعد مختلف را بصوری ترتیبی با هم در نظر میگیریم و نتیجه گیری میکنیم تا درنهایت از a به b برسیم (a entails b).

یکی از قواعدی که میتوان از آن استفاده کرد modus ponens است.

برای مثال فرض کنیم طبق knowledge base خود ما میدانیم اگر ما تقلب کنیم، از نمره ما کم میشود.

حال ما تقلب کردیم.

در نتیجه نمره ما کم میشود.

در اصل جمله اول knowledge base ما بود و جمله دوم اتفاق حتمی ای بود که رخ داده است که با استفاده از modus ponens جمله سوم را نتیجه گرفتیم.

گزارش تمرین عملی

در این تمرین قصد داریم tic tac toe را توسط الگوریتم MCTS پیاده کنیم.

این الگوریتم از چند بخش تشکیل شده است.

1. Selection
2. Expansion
3. Simulation
4. Propagation

هر مرحله از ورودی کاربر یک جدول از بازی به عنوان root درخت در نظر گرفته میشود. و توسط آن root و بچه های هر state درخت را ایجاد میکنیم.

در هر مرحله توسط تابع selection_expansion نود بعدی را انتخاب میکنیم.

هر state از بازی شامل چند متغیر کلیدی هستند.

- تعداد دفعات visit شدن آن استیت
- Score آن استیت
- مقدار UCB آن استیت

که توسط مقدار UCB ، ما state بعدی را انتخاب میکنیم.

اگر در استیتی بودیم که بچه های آن ویزیت نشده بودند باید با expand کردن آن همسایه های آن را ایجاد کنیم.

و اگر برای استیتی تمامی بچه های آن expand شده بودند، باید بچه ای را انتخاب کنیم که مقدار UCB کمتری داشته باشد.

پس از انتخاب استیت مورد نظر باید تابع simulation را فراخوانی کنیم.

در این تابع بصورت رندوم با شروع از استیت بازی را پیش میبریم تا به یک نود ترمینال برسیم.

و سپس نتیجه را ذخیره میکنیم که آیا X برنده بازی شده یا O و یا مساوی شدند.

سپس با فراخوانی تابع propagation بصورت بازگشتی به روت برمیگردیم و مقادیر هر استیت را آپدیت میکنیم. به طوری که به مقدار visits هر نود یک واحد اضافه میکنیم و اگر در استیتی نوبت X باشد و نتیجه simulation نیز X باشد مقدار score را یک واحد اضافه کرده در غیر این صورت مقدار را یک واحد کم میکنیم و همین فرآیند را برای O نیز پیاده میکنیم. و همینطور مقادیر UCB را نیز آپدیت میکنیم.

تمامی این مراحل را به تعداد 1000 بار تکرار میکنیم تا به یک نتیجه مطلوبی برسیم.

در نهایت از بچه های root استیتی که مقدار UCB بیشتری داشته باشد را خروجی میدهیم. و در این حالت موقعیت O در بازی در آن نوبت مشخص میشود.