Logic-Based Agents

Aim To understand how resolution works, and how to integrate resolution into the decision making component of logical agents.

Preparation Go through the slide material of Lecture 7 and 8 on the course website.

About the lab This lab consists of three parts, and it can be done by at most three persons.

Lab examination demonstrate the tasks you solved during any scheduled lab session.

Grading Task A $\rightarrow$ 3 / Tasks A, B $\rightarrow$ 4 / Tasks A, B, C $\rightarrow$ 5

Task A
This task is about to implement the resolution inference mechanism applied to formulae (called clauses) in Conjunctive Normal Form (CNF) for propositional logic. In particular, you have to write a program that implements:

1. The resolution of two clauses in CNF. That is, given two clauses the program calculates their resolvent by applying one resolution step.

2. The resolution mechanism applied to a given set S of clauses. Given S, the program selects two arbitrary clauses from S, or any previously calculated resolvent, and calculates the new resolvents[1]. The program applies the resolution step until no new resolvent can be derived. See the Solver for Clauses in CNF on p.4.

Write your program in Java if you are going to solve Task C since the Wumpus World is in Java.

[1] To do so, you can use the program you developed in task A.1

Task B
Choose one of the three puzzles below and formalize it in propositional logic. Then, use the program you implemented in Task A.2 to solve it.

1. Robbery puzzle

There was a robbery in which a lot of goods were stolen. The robber(s) left in a truck. It is known that:

1. Nobody else could have been involved other than A, B and C.
2. C never commits a crime without A's participation.
3. B does not know how to drive.

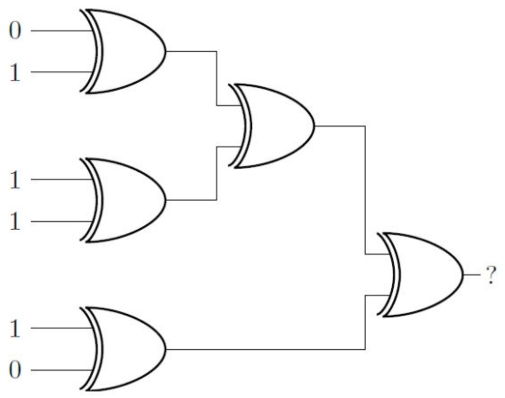
So, is A innocent or guilty?

2. Hat Color puzzle

Three people are in a room wearing hats. Two of them are wearing blue hats and one of them is wearing a red hat. They can't see what color hat they are wearing, but they can see everyone else's hats. They are playing a game where their objective is to figure out the color of their own hat.

If a player sees that the other two players have a red hat and a blue hat, what color hat is the player wearing?

3. Logic Gates puzzle



The image on the left has five XOR gates. A XOR gate outputs a 1 if the inputs are a 1 and a 0, otherwise it outputs a 0.

What is the output at the end?

Task C (Difficult)
This task is about to design and implement the decision making component of an AI agent. The AI agent we consider is a logic-based agent that uses resolution as inference mechanism. The scenario we consider is the one of the Wumpus World (described on page 6).

- Download the file WumpusWorld.zip[2] from the course website.
- The WumpusWorld.zip consists of a Wumpus World simulator and an empty logic class called MyAgent.java.
- Your task is to build an AI agent (by implementing the file MyAgent.java) for playing the Wumpus World game. To do this, you can reuse the code you developed in Task A.2.

Solver for Clauses in CNF

Notation

- Let Σ be a set of propositional symbols. A literal is any propositional symbol $a \in \Sigma$ or its negation $\neg a$. a is called positive literal while $\neg a$ is called negative literal. If $\Sigma = \{a, b\}$, then the set of literals is $\{a, b, \neg a, \neg b\}$.
- A clause A is represented by the two sets $A.p$ and $A.n$ containing the positive and negative literals in A, respectively. If $A = a \vee b \vee \neg c$, then $A.p = \{a, b\}$ and $A.n = \{c\}$. Note that both $A.p$ and $A.n$ do not contain duplicates since they are sets. This corresponds to the fact that having duplicates in a clause does not change its meaning. That is, $A = a \vee b \vee \neg c \vee b$ is equal to $a \vee b \vee \neg c$.
- The order of the literals in a clause is not important. The clause $A = a \vee b \vee \neg c$ is equivalent to $A = b \vee \neg c \vee a$.
- If a clause A contains a propositional symbol a and its negation $\neg a$ (that is, $a \in A.p \cap A.n$), then the clause is a tautology. A tautology is always true (it does not add anything to the knowledge base).
- A clause A subsumes or is equal to a clause B, written as $A \preceq B$, if it holds that $A.p \subseteq B.p$ and $A.n \subseteq B.n$. A subsumes B, written as $A \prec B$, if it holds that $A \preceq B$, and $A.p \subset B.p$ or $A.n \subset B.n$.

  Let S_1 and S_2 be two sets. We write $S_1 \subseteq S_2$ to indicate that S_1 is a subset of S_2, and $S_1 \subset S_2$ to indicate that S_1 is a proper (strict) subset of S_2.

```
**function** Resolution(A,B) **return** resolvent of A and B, or false
    **input:** clauses $A$ and $B$; $A$ and $B$ are local variables
    **if** $A.p \cap B.n = \{\}$ and $A.n \cap B.p = \{\}$ **do**
        **return** false
    **if** $(A.p \cap B.n) \neq \{\}$ **do**
        $a \leftarrow$ Pick_random_element_from($A.p \cap B.n$)
        $A.p \leftarrow A.p - \{a\}$
        $B.n \leftarrow B.n - \{a\}$
    **otherwise**
        $a \leftarrow$ Pick_random_element_from($A.n \cap B.p$)
        $A.n \leftarrow A.n - \{a\}$
        $B.p \leftarrow B.p - \{a\}$
    $C.p \leftarrow A.p \cup B.p$
    $C.n \leftarrow A.n \cup B.n$
    **if** $C.p \cap C.n \neq \{\}$ **do**            // $C$ is a tautology
        **return** false
    $C \leftarrow$ Remove_duplicates($C$)
    **return** C
```

```
function Solver(KB) return set of clauses
    Input: set of clauses KB

    repeat
        $S = \{\}$
        $KB' \leftarrow KB$

        for each A, B in KB :
            $C \leftarrow$ Resolution(A,B)
            if $C \neq \mathsf{false}$ do
                $S \leftarrow S \cup \{C\}$

        if $S = \{\}$
            return KB

        $KB \leftarrow$ Incorporate($S, KB$)

    until $KB' = KB$


function Incorporate(S,KB) return set of clauses
    Input: set of clauses $S$, set of clauses $KB$

    for each $A$ in $S$ :
            $KB \leftarrow$ Incorporate_clause($A, KB$)

    return $KB$


function Incorporate_clause(A,KB) return set of clauses
    Input: clause $A$, set of clauses $KB$

    if there is a clause $B \in KB$ such that $B \preceq A$ do
        return KB

    for each B in KB :
        if $A \prec B$ do
            $KB \leftarrow KB - \{B\}$

    $KB \leftarrow KB \cup \{A\}$
    return $KB$
```

Examples

Resolution

- Given $A = a \vee b \vee \neg c$ and $B = c \vee b$

  $a \vee b \leftarrow$ Resolution(A,B)

- Given $A = a \vee b \vee \neg c$ and $B = d \vee b \vee \neg g$

  false $\leftarrow$ Resolution(A,B)

- Given $A = \neg b \vee c \vee t$ and $B = \neg c \vee z \vee b$

  false $\leftarrow$ Resolution(A,B)

  since by applying one resolution step to A and B one obtains a resolvent that is a tautology.

Subsumption

- $c \vee a \ \prec \ a \vee b \vee c$
- $b \vee \neg c \ \prec \ a \vee b \vee \neg c$
- $b \vee \neg f \vee \neg c \ \not\prec \ a \vee b \vee \neg c$
- $b \ \prec \ a \vee b \vee \neg c$
- $b \vee \neg c \vee a \ \not\prec \ a \vee b \vee \neg c$ (note that $b \vee \neg c \vee a \ \preceq \ a \vee b \vee \neg c$ holds)

Drawing conclusions

Assume that the following formalae model the behaviour of the agent Bob.

$$\begin{aligned}&sun \wedge money \Rightarrow ice\\&money \wedge \neg ice \Leftrightarrow movie\\&\neg sun \wedge \neg money \Rightarrow cry\end{aligned}$$

If Bob goes to the movie, what can you deduce? To anwer this question, proceed as follows.

1. Transform the formulaes into CNF by using the online converter:

   $\neg sun \vee \neg money \vee ice$

   $\neg money \vee ice \vee movie$

   $\neg movie \vee money$

   $\neg movie \vee \neg ice$

2. Define the KB of Bob by including the converted formulae in CNF and *movie*.

   KB = $\{\neg sun \vee \neg money \vee ice,\ \neg money \vee ice \vee movie,\ \neg movie \vee money,\ \neg movie \vee \neg ice,\ movie\}$

3. Run Solver(KB)

(a) The Solver enters the main loop. For each clause A and B in KB, the Solver applies the resolution step to A and B, and it saves the result in S.
S = $\{\neg ice,\ money,\ \neg sun \lor \neg money \lor \neg movie,\ \neg sun \lor ice \lor \neg movie\}$

(b) Then, the Solver incorporates S into KB. New KB is:
KB = $\{\neg sun \lor \neg money \lor ice,\ movie,\ \neg ice,\ money,\ \neg sun \lor \neg money \lor \neg movie,\ \neg sun \lor ice \lor \neg movie\}$

(c) The Solver cycles into the main loop. It sets S={} and calculates all the resolvents that can be derived by the new KB.
S = $\{\neg sun \lor \neg money, \neg sun \lor \neg movie, \neg sun \lor ice\}$

(d) Again, it incorporate S into KB.
KB = $\{\neg sun \lor \neg money,\ \neg sun \lor \neg movie,\ \neg sun \lor ice,\ movie,\ \neg ice,\ money\}$

(e) Finally, the Solver repeats the main loop. S is now
S = $\{\neg sun\}$

(f) By incorporating S into KB, we obtain:
KB = $\{\neg sun, movie, \neg ice, money\}$

(g) By repeating the main loop, the Solver derives a new S, and it incorporates S into the KB. Being the KB obtained equal to KB' (the knowledge base saved at the beginning of the main loop), the Solver exits the main loop.

By knowing that Bob goes to the movie, one can conclude that Bob has money, Bob does not have ice-cream, and it is not sunny.

The Wumpus World

The Wumpus World is a classic grid-based game first published in 1973. It has since then often been used as a test bed for AI agents since the agent (the hunter) often has to deal with contradicting and non-complete information.

The game has the following objects:

- **Wumpus** - a beast that eats everyone on the same square. If the agent gets eaten by the Wumpus the game is over (and the player loses 1000 points).
- **Pits** - if the agent falls into a pit it has to climb up again (and loses 1000 points).
- **Gold** - the agent wins by moving to the same square as the gold and pick it up (and gain 1000 points).

The agent can:

- **Move** upwards, downward, left or right. Player looses 1 point for each action.
- **Climb** up from a pit.
- **Shoot** the arrow upwards, downward, left or right. The arrow kills the Wumpus, but the player only has one arrow.
- **Grab** the gold.

The agent perceives:

- **Stench** - each square adjacent to the Wumpus (not diagonally) has a Stench.
- **Breeze** - each square adjacent to a pit (not diagonally) has a Breeze.
- **Glitter** - perceived if the agent is at the same square as the gold.
- **Scream** - if the Wumpus is killed by the arrow, a scream is heard in every square.

To win the game:

- **Win** - the AI agent wins the game if it grabs the gold. It looses if it gets eaten by the Wumpus.
- **Win, Best Score** - you can build an AI agent that tries to win while maximizing the game's score. Note that **Win, Best Score** is not a requirement to pass Task C (only **Win**).