An Assignment Report

On

Knowledge Representation using Predicate Logic

by

Aditya Sharma
AU18B1009

Under the guidance of

Ashish Bansal

School of Engineering

Avantika University, Ujjain

2020-2021

Content: **Page**

Problem Statement 1: Family Tree

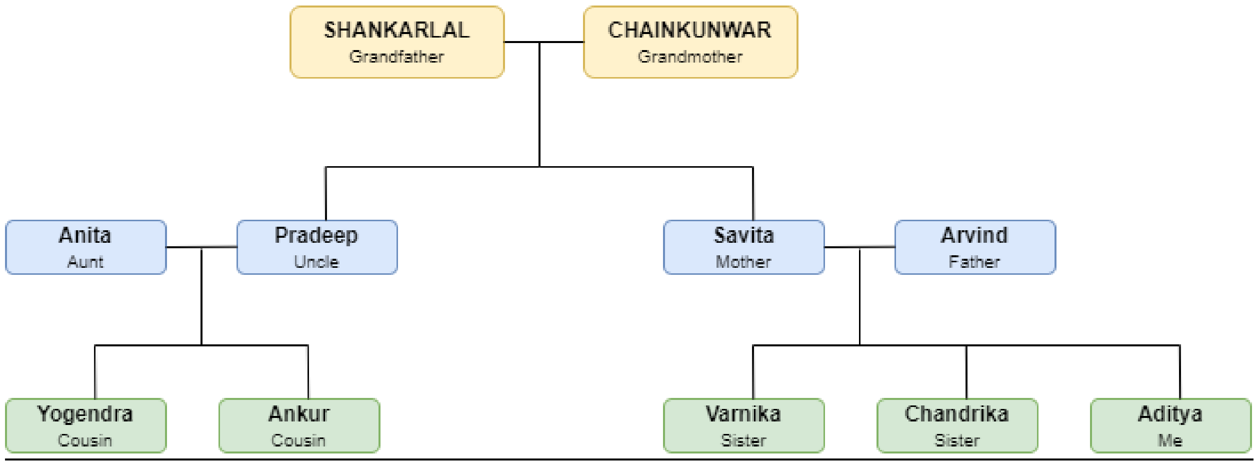
Knowledge Representation:

Knowledge Representation in AI describes the representation of knowledge. Basically, it is a study of how the **beliefs, intentions**, and **judgments** of an **intelligent agent** can be expressed suitably for automated reasoning. One of the primary purposes of Knowledge Representation includes modeling intelligent behavior for an agent.

Knowledge Base:

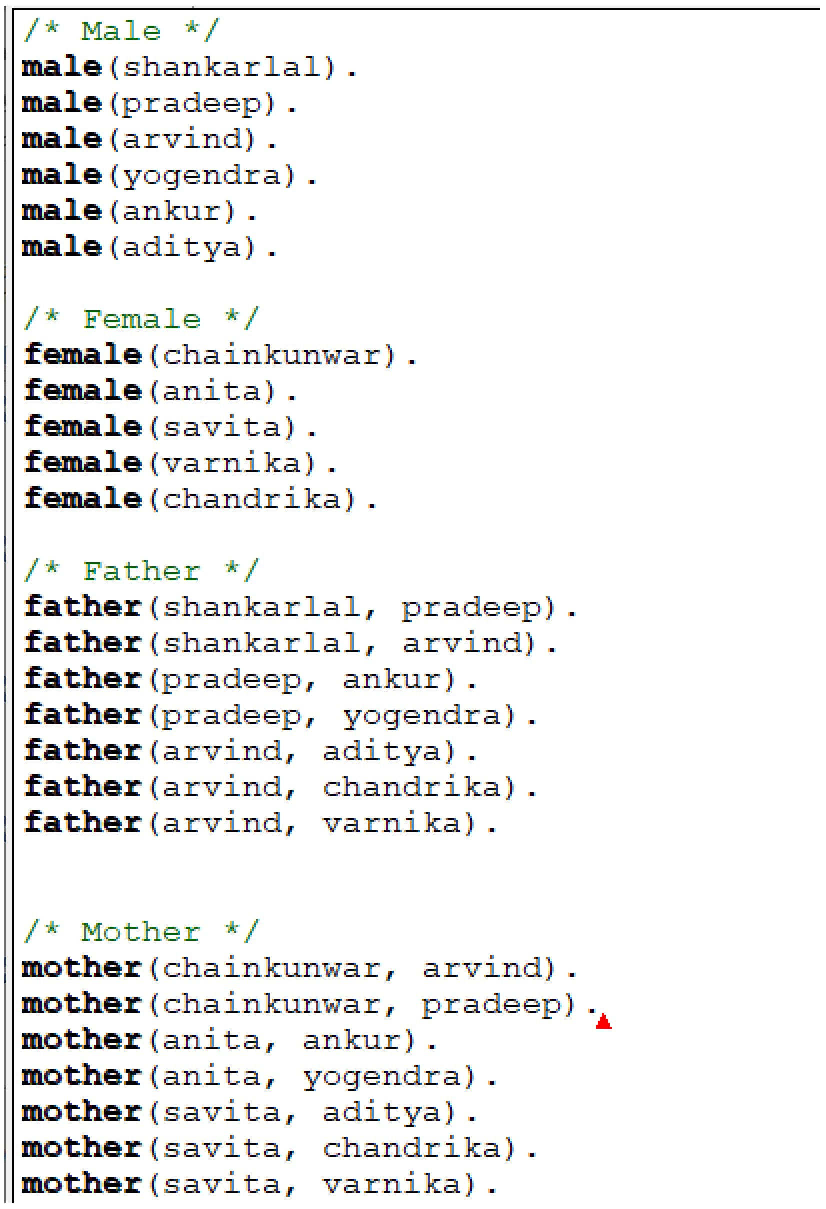
A knowledge base is a database used for knowledge sharing and management. It promotes the collection, organization and retrieval of knowledge. Many knowledge bases are structured around artificial intelligence and not only store data but find solutions for further problems using data from previous experience stored as part of the knowledge base.

Family Tree

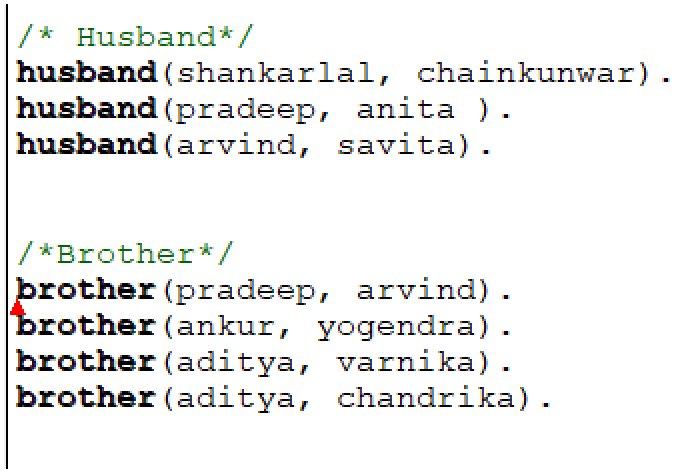


Predicates:

male, female, father, husband, mother, brother, sister, grandfather, grandmother, uncle, aunt



```
/* Male */
male(shankarlal).
male(pradeep).
male(arvind).
male(yogendra).
male(ankur).
male(aditya).

/* Female */
female(chainkunwar).
female(anita).
female(savita).
female(varnika).
female(chandrika).

/* Father */
father(shankarlal, pradeep).
father(shankarlal, arvind).
father(pradeep, ankur).
father(pradeep, yogendra).
father(arvind, aditya).
father(arvind, chandrika).
father(arvind, varnika).


/* Mother */
mother(chainkunwar, arvind).
mother(chainkunwar, pradeep).
mother(anita, ankur).
mother(anita, yogendra).
mother(savita, aditya).
mother(savita, chandrika).
mother(savita, varnika).
```

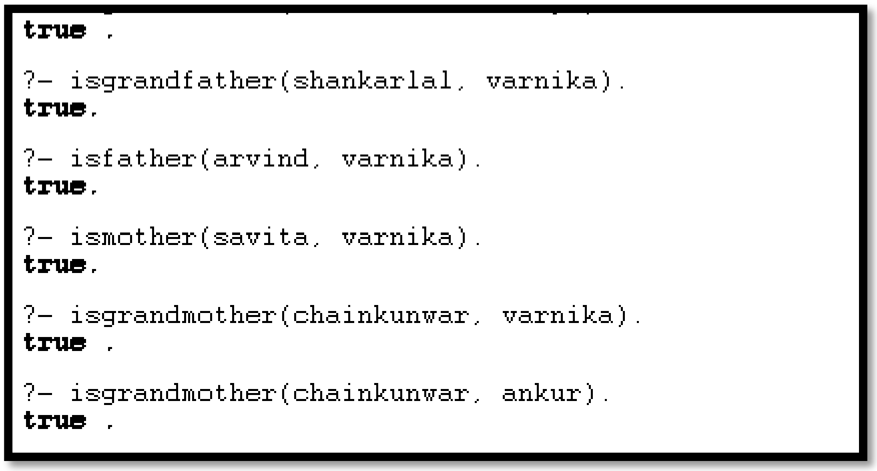


```
/* Husband*/
husband(shankarlal, chainkunwar).
husband(pradeep, anita ).
husband(arvind, savita).


/*Brother*/
brother(pradeep, arvind).
brother(ankur, yogendra).
brother(aditya, varnika).
brother(aditya, chandrika).
```

Clauses:

Isgrandfather, Isgrandmother, iscousin,isgrandson,isgranddauther,isaunt,isuncle

```
/* Clauses*/

isfather(X, Y):- male(X), father(X,Y).

ismother(X,Y):- female(X), mother(X,Y).

isgrandfather(X,Y):- male(X), father(X,Z) , father(Z, Y).

isgrandmother(X,Y):- female(X), mother(X,Z), father(Z,Y).

iscousin(X,Y):- father(P,X), father(Q, Y) , brother(P,Q).

isgrandson(Y,X):- father(X,Z), father(Z,Y), male(Y).

isgranddaughter(Y,X):- father(X,Z), father(Z, y), female(Y).

isaunt(X,Y):- father(P, Y), brother(P, Q), husband(Q, X).

isuncle(X,Y):- father(Z,Y),brother(Z,X).
```



```
true .

?- isgrandfather(shankarlal, varnika).
true.

?- isfather(arvind, varnika).
true.

?- ismother(savita, varnika).
true.

?- isgrandmother(chainkunwar, varnika).
true .

?- isgrandmother(chainkunwar, ankur).
true .
```

Prolog output.

Problem Statement 2: Goal Stack Planning to solve Block world problem**.**

Blocks World Problem:

There is a table on which some blocks are placed. Some blocks may or may not be stacked on other blocks. We have a robot arm to pick up or put down the blocks. The robot arm can move only one block at a time, and no other block should be stacked on top of the block which is to be moved by the robot arm. Our aim is to change the configuration of the blocks from the Initial State to the Goal State.

Goal Stack Planning:

Planning is process of determining various actions that often lead to a solution. Planning is useful for non-decomposable problems where sub goals often interact. Goal Stack Planning is one of the earliest methods in artificial intelligence in which we work **backwards from the goal state to the initial state.** We start at the goal state and we try fulfilling the preconditions required to achieve the initial state. These preconditions in turn have their own set of preconditions, which are required to be satisfied first. We keep solving these "goals" and "sub-goals" until we finally arrive at the Initial State. **We make use of a stack to hold these goals that need to be fulfilled as well the actions that we need to perform for the same**.

Representing the configurations as a list of "predicates"

Predicates can be thought of as a statement which helps us convey the information about a configuration in Blocks World. Given below are the list of predicates as well as their intended meaning:

1. ON(A,B) : Block A is on B
2. ONTABLE(A) : A is on table
3. CLEAR(A) : Nothing is on top of A

4. HOLDING(A) : Arm is holding A.
5. ARMEMPTY : Arm is holding nothing

This are the system state. Using these predicates, we represent the Initial State and the Goal State.

Operations performed by the Robotic Arm:

Pickup(A): Pickup block A from the table

Putdown(B): Put down the block B on the table.

Stack (A, B): Stacking Block A on Block B.

Unstack (A, B): Picking up Block A which is on top of Block B.

Predicates for the following operations:

Pickup(A):

Prerequisite: Clear(A) ⋀ Armempty

Outcome: Holding(A)

Putdown(B):

Prerequisite: Holding(B)

Outcome: Ontable(B) ⋀ Armempty

Stack (A, B)

Prerequisite: Holding(A) ⋀ Clear(B)

Outcome: On(A, B) ⋀ Armempty ⋀ Clear(A)

Unstack (A, B)

Prerequisite: Armempty ⋀ Clear(A) ⋀ On (A, B)

Outcome: Holding(A) ⋀ Clear(B)

Initial State: on (A, B) ⋀ on (C, D) ⋀ ontable(B) ⋀ ontable(D) ⋀ clear(A) ⋀ clear(C) ⋀ Armempty

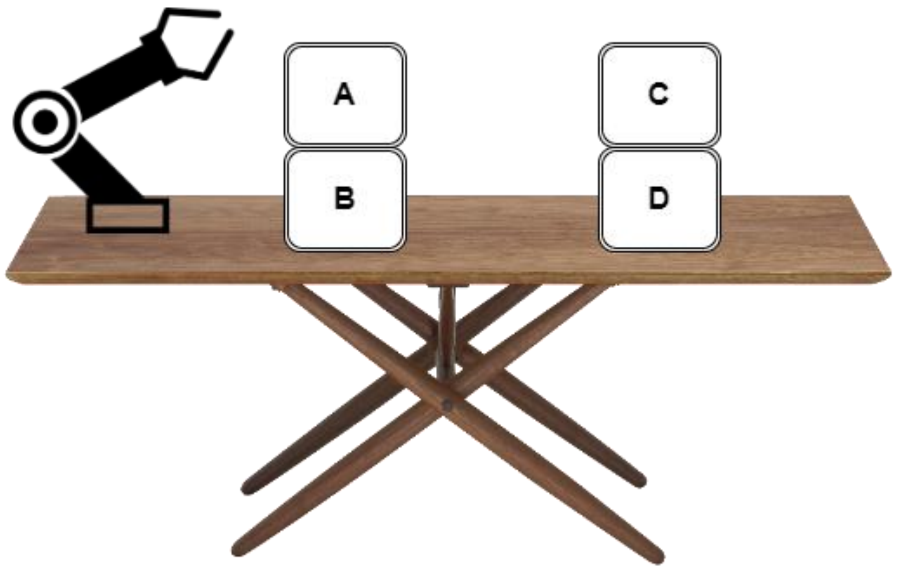


Fig. 1 Given problem initial state

Goal State: on (A, C) ⋀ on (B, D) ⋀ ontable(C) ⋀ ontable(D) ⋀ clear(A) ⋀ clear(B) ⋀ Armempty

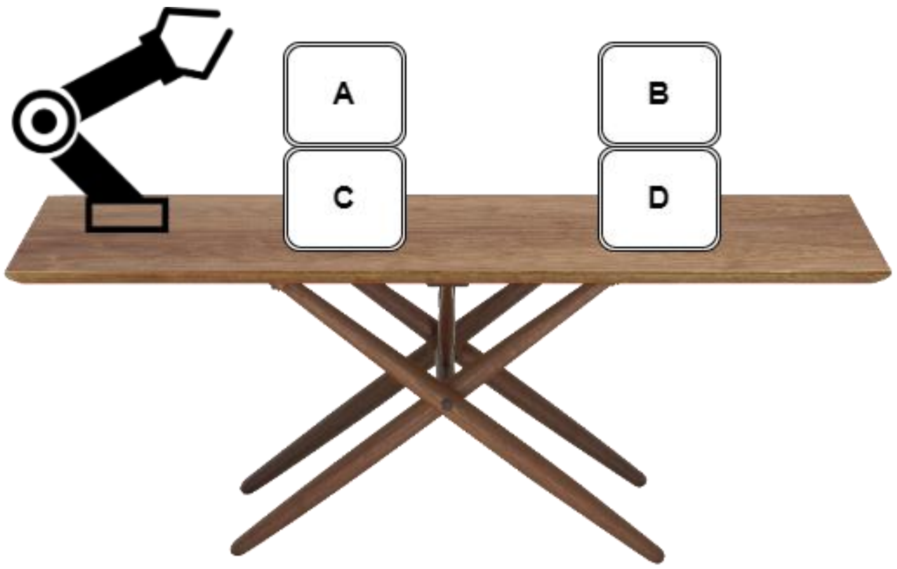
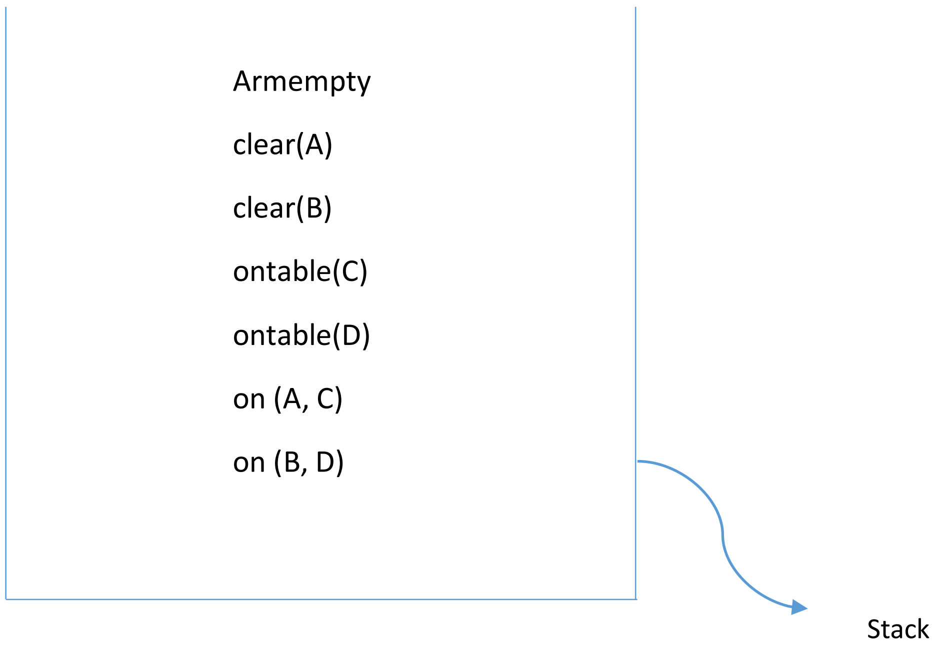
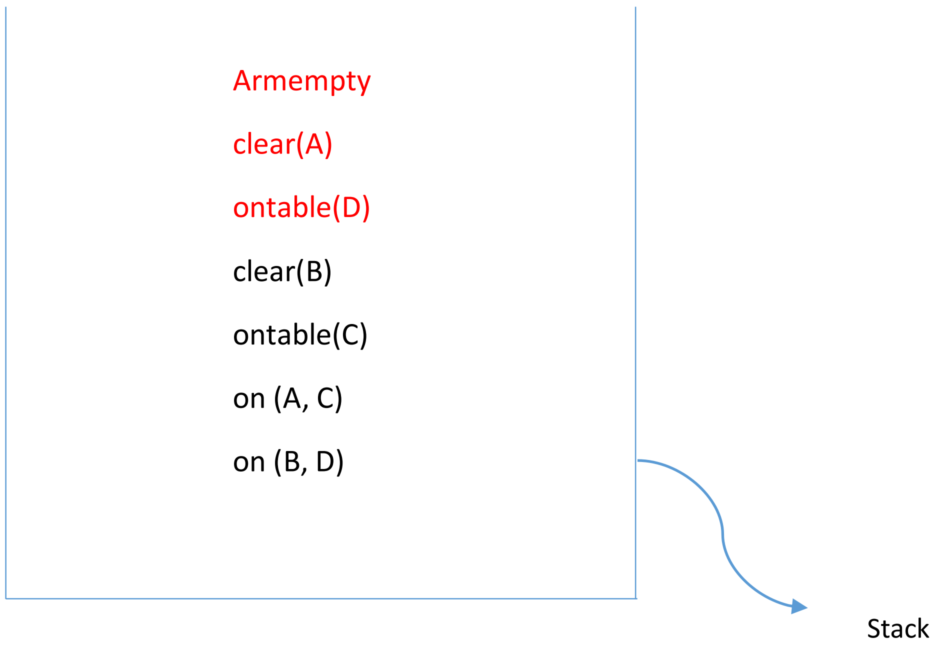


Fig. 1 Given problem goal state.

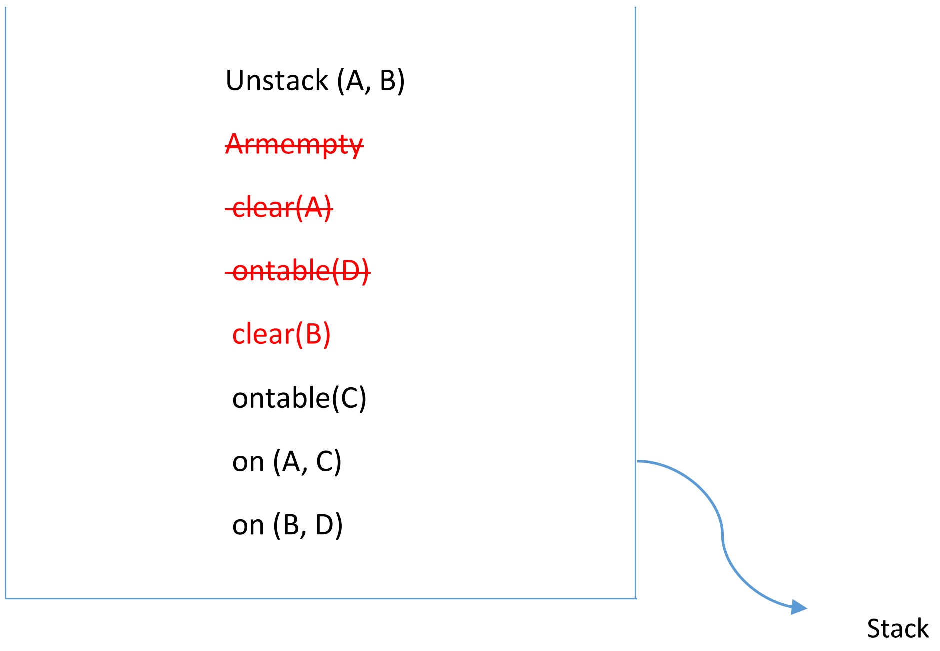
As we had defined the goal state, now we will push the goal to the stack:



Here on popping we can see that Armempty, Clear(A) and Ontable(D) are true in our current(Initial) world, therefore we will not worry for this predicates.



Now, POP the stack. Clear(B) is not true in our current world, So the next step is to push the relevant action which could achieve the subgoal clear(B) in to the stack.



Unstack (A, B)

Prerequisite: Armempty ⋀ Clear(A) ⋀ On (A, B)

Outcome: Holding(A) ⋀ Clear(B)

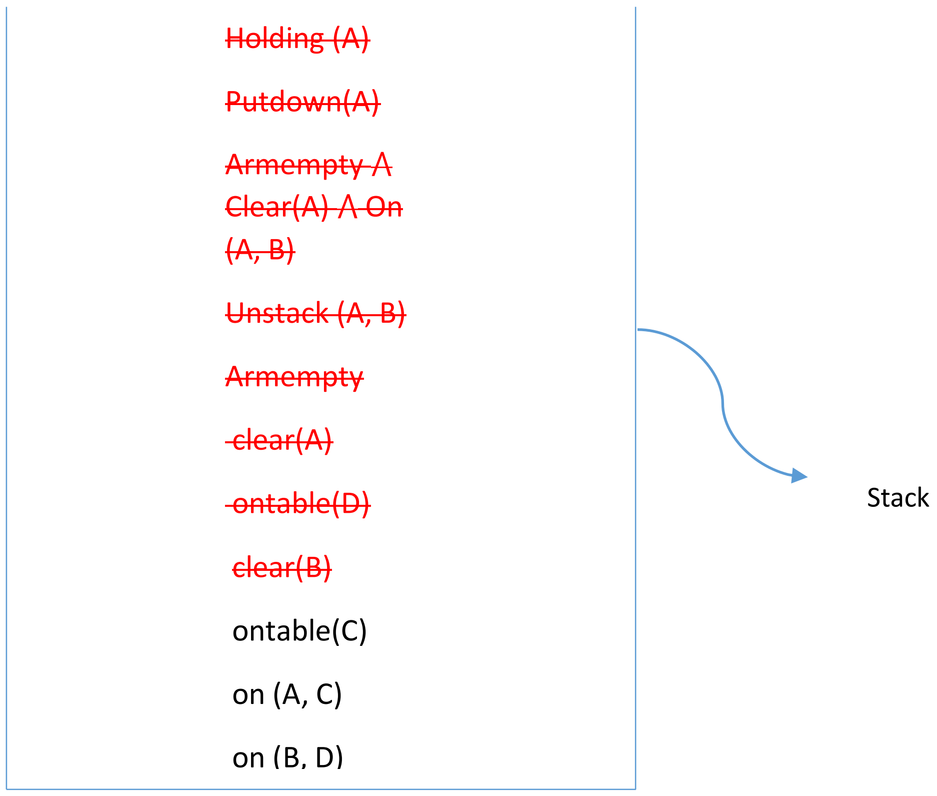
After unstacking the A and B, we need to empty the Arm in order to facilitate other operations.

Putdown(A):

Prerequisite: Holding(A)

Outcome: Ontable(A) ⋀ Armempty

Now we are on the stage were, we have Clear(B), Clear(A). Will check for the next i.e. ontable(C), which is not true, So the next step is to push the relevant action which could achieve the subgoal ontable(C) in to the stack.



We will now perform:

Unstack (C, D)

Prerequisite: Armempty ⋀ Clear(C) ⋀ On (C, D)

Outcome: Holding(A) ⋀ Clear(D)

After unstacking the C and D, we need to empty the Arm in order to facilitate other operations.

Putdown(C):

Prerequisite: Holding(A)

Outcome: Ontable(A) ⋀ Armempty

The stack will now,

~~Holding(C)~~

~~Putdown(C)~~

~~Armempty ⋀ Clear(C) ⋀ On (C, D)~~

~~Unstack (C, D)~~

~~Holding (A)~~

~~Putdown(A)~~

~~Armempty ⋀ Clear(A) ⋀ On (A, B)~~

~~Unstack (A, B)~~

~~Armempty~~

~~clear(A)~~

~~ontable(D)~~

~~clear(B)~~

~~ontable(C)~~

on (A, C)

on (B, D)

Now we will check the predicate, On(A, C), which is not true, So the next step is to push the relevant action which could achieve the sub goal ON(A, C) in to the stack.

Stack (A, C)

Prerequisite: Holding(A) ⋀ Clear(C)

Outcome: On(A, B) ⋀ Armempty ⋀ Clear(A)

Pickup(A):

Prerequisite: Clear(A) ⋀ Armempty

Outcome: Holding(A)

~~Clear(A) ⋀ Armempty~~

Holding(A) ~~⋀ Clear(C)~~

~~Stack(A,C)~~

~~Holding(C)~~

~~Putdown(C)~~

~~Armempty ⋀ Clear(C) ⋀ On (C, D)~~

~~Unstack (C, D)~~

~~Putdown(A)~~

~~Holding (A)~~

~~Armempty ⋀ Clear(A) ⋀ On (A, B)~~

~~Unstack (A, B)~~

~~Armempty~~

~~clear(A)~~

~~ontable(D)~~

~~clear(B)~~

~~ontable(C)~~

~~on (A, C)~~

on (B, D)

Now we will check the predicate, On(B, D), which is not true, So the next step is to push the relevant action which could achieve the sub goal ON(B, D) in to the stack.

Stack (B, D)

Prerequisite: Holding(B) $\wedge$ Clear(D)

Outcome: On(B, D) $\wedge$ Armempty $\wedge$ Clear(B)

Pickup(B):

Prerequisite: Clear(B) $\wedge$ Armempty

Outcome: Holding(B)

~~Clear(B) Λ~~
~~Armempty~~

Holding(B) ~~Λ~~
~~Clear(D)~~

~~Stack(B, D)~~

~~Clear(A) Λ~~
~~Armempty~~

~~Holding(A) Λ~~
~~Clear(C)~~

~~Stack(A,C)~~

~~Holding(C)~~

~~Putdown(C)~~

~~Armempty Λ~~
~~Clear(C) Λ On~~
~~(C, D)~~

~~Unstack (C, D)~~

~~Holding (A)~~

~~Putdown(A)~~

~~Armempty Λ~~
~~Clear(A) Λ On~~
~~(A, B)~~

~~Unstack (A, B)~~

~~Armempty~~

~~clear(A)~~

~~ontable(D)~~

~~clear(B)~~

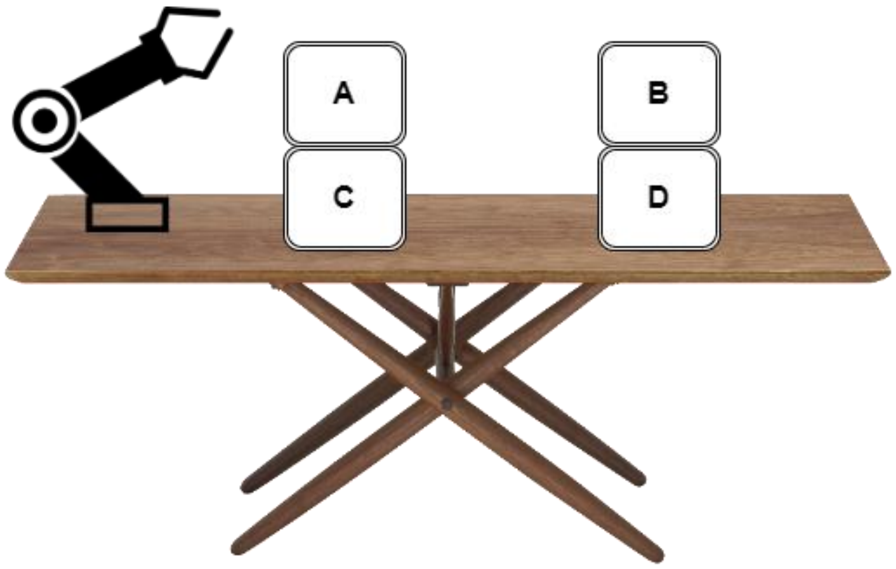
~~ontable(C)~~

~~on (A, C)~~

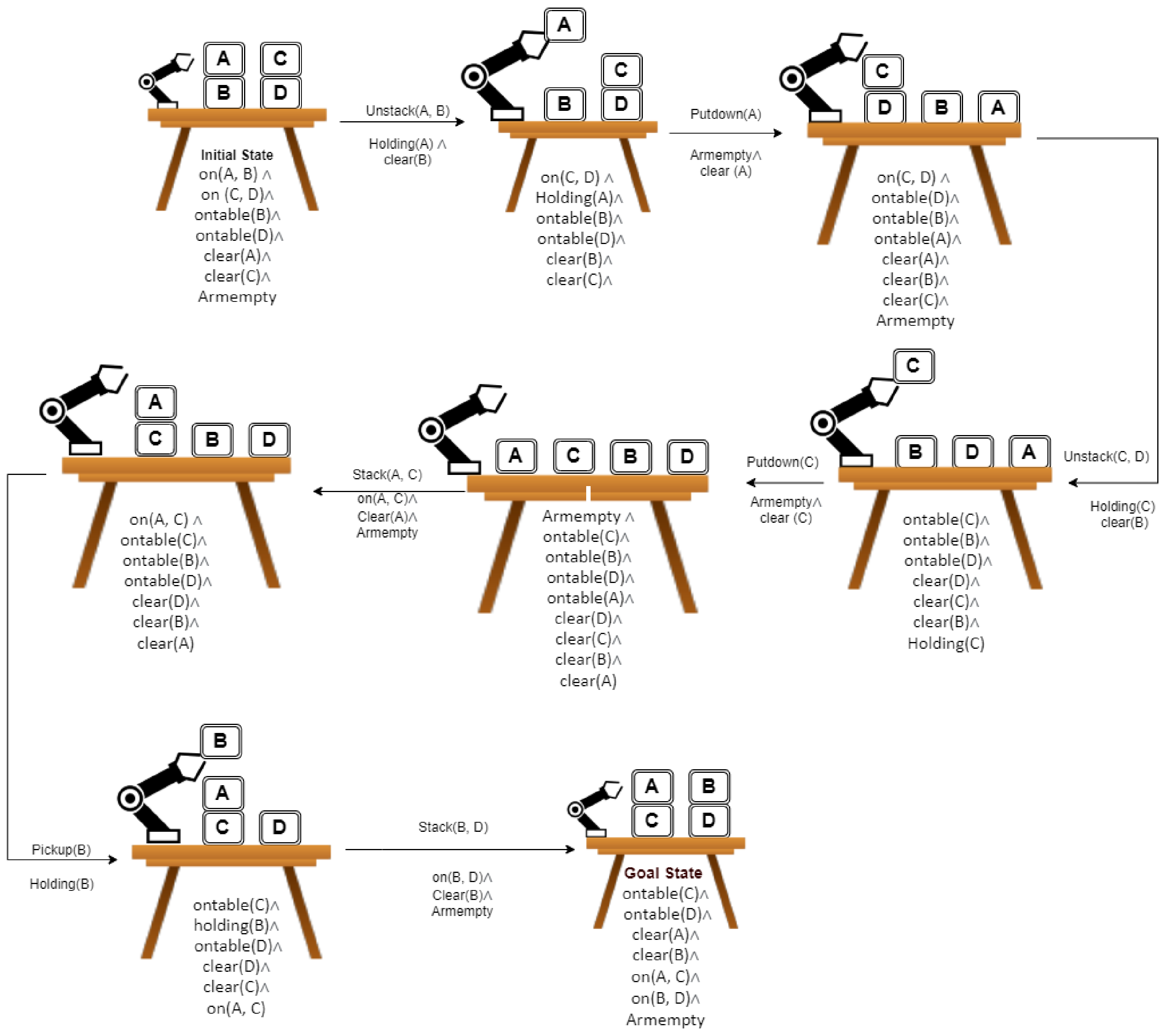
on (B, D)

Now, we had reached to our final goal state.

Goal State: Armempty Λ ontable(C) Λ ontable(D) Λ clear(A) Λ clear(B) Λ on(A, C) Λ on(B, D)

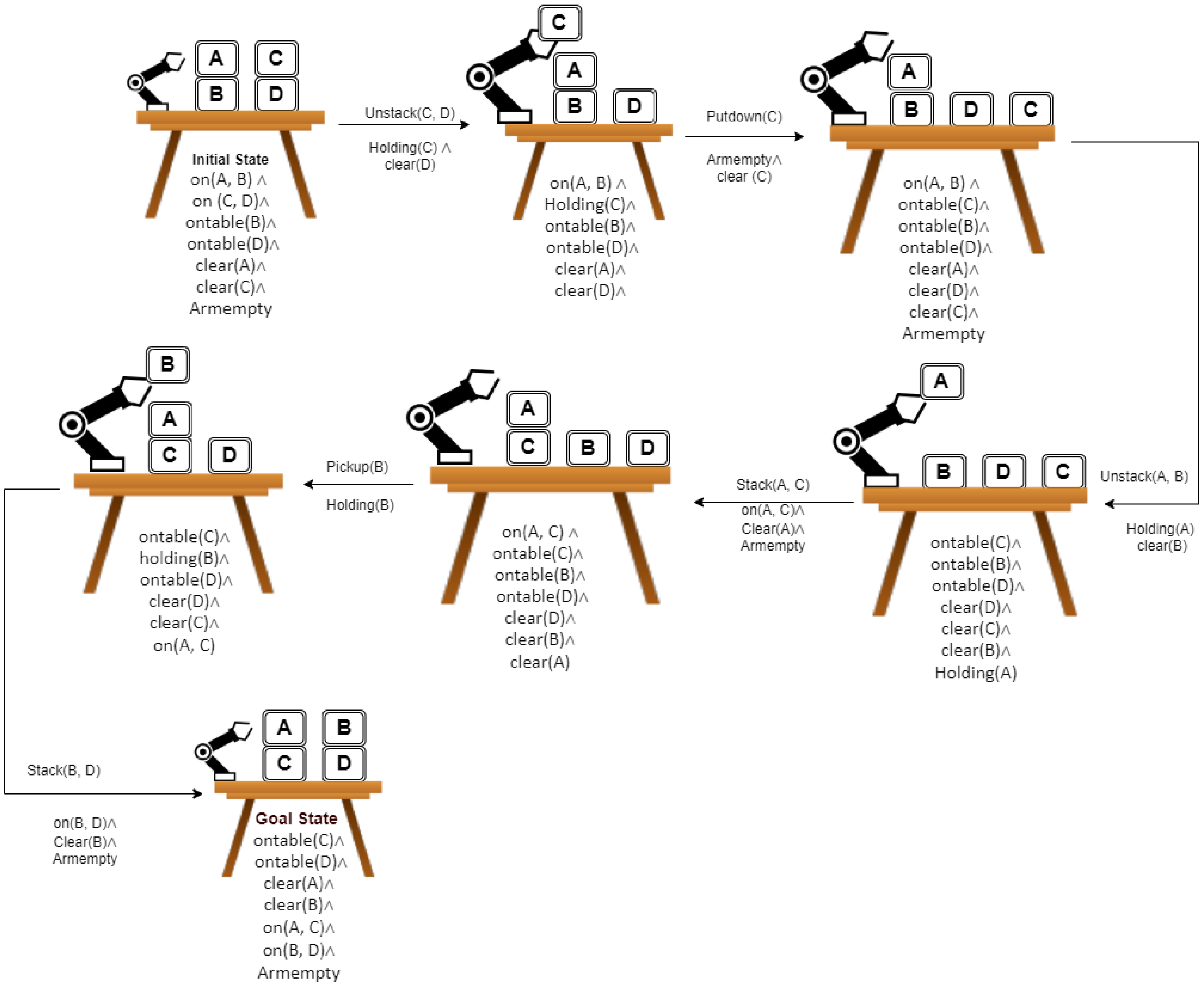


Operations:


A
C
B
D
Initial State
on(A, B) ∧
on (C, D)∧
ontable(B)∧
ontable(D)∧
clear(A)∧
clear(C)∧
Armempty
Unstack(A, B)
Holding(A) ∧
clear(B)
A
C
B
D
on(C, D) ∧
Holding(A)∧
ontable(B)∧
ontable(D)∧
clear(B)∧
clear(C)∧
Putdown(A)
Armempty∧
clear (A)
C
D
B
A
on(C, D) ∧
ontable(D)∧
ontable(B)∧
ontable(A)∧
clear(A)∧
clear(B)∧
clear(C)∧
Armempty
Unstack(C, D)
Holding(C)
clear(B)
C
B
D
A
ontable(C)∧
ontable(B)∧
ontable(D)∧
clear(D)∧
clear(C)∧
clear(B)∧
Holding(C)
Putdown(C)
Armempty∧
clear (C)
A
C
B
D
Armempty ∧
ontable(C)∧
ontable(B)∧
ontable(D)∧
ontable(A)∧
clear(D)∧
clear(C)∧
clear(B)∧
clear(A)
Stack(A, C)
on(A, C)∧
Clear(A)∧
Armempty
A
C
B
D
on(A, C) ∧
ontable(C)∧
ontable(B)∧
ontable(D)∧
clear(D)∧
clear(B)∧
clear(A)
Pickup(B)
Holding(B)
B
A
C
D
ontable(C)∧
holding(B)∧
ontable(D)∧
clear(D)∧
clear(C)∧
on(A, C)
Stack(B, D)
on(B, D)∧
Clear(B)∧
Armempty
A
B
C
D
Goal State
ontable(C)∧
ontable(D)∧
clear(A)∧
clear(B)∧
on(A, C)∧
on(B, D)∧
Armempty

Other way of solving the problem:

Here we can start with unstacking C and D first, so that we can minimize the efforts.



Conclusion:

AI is at the centre of a new enterprise to build computational models of intelligence. The main assumption is that intelligence (human or otherwise) can be represented in terms of symbol structures and symbolic operations which can be programmed in a digital computer. Aspects of intelligent behaviour, such as solving problems, making inferences, learning, and understanding language, have already been coded as computer programs, and within very limited domains, such as identifying diseases of soybean plants, AI programs can outperform human experts.

Knowledge representation in AI is going to be an evolving field. Someday it will provide the system that can be integrated, which has near-human perception and reasoning.