03-12-2024 WEEK-8 KNOWLEDGE BASE USING FORWARD REASONING

ALGORITHM-

GOKITIWI-	1177 26		
ALGORITH	M. Danislans 9	STONATON.	Bulon avois!
Forward CI	harncies, a):	A 1200	The Payr enp
Instialize	an empry se	t of der	ived fact
Initializa	an empty	set of a	ppued rules
	inges=True		
	anges as True		
	ranges = false		
	rule R in KB		
	a already 1		
	ep mes rule		
	all " if" condi		
	un's fied:		
pert	red one "ther	" fact	a exphanal
	it to KB 14 n		
	K R as appli		
set	changer = True	AND ONL	2-00 0 x 4-/
	Q å PN KB f		1
			derived fact
else, rem	rn " Not prove	n "	Pacif
			1) brownsa
HERITAGE IN			The second of

CODE-

```
knowledge base = {
  "facts": [
     {"type": "Food", "object": "Banana"},
     {"type": "Food", "object": "Pizza"},
     {"type": "Consumes", "subject": "Sam", "object": "Idli"},
     {"type": "NotHarmed", "subject": "Sam", "object": "Idli"}
  ],
  "rules": [
     {"if": [{"type": "Consumes", "subject": "x", "object": "y"},
           {"type": "NotHarmed", "subject": "x", "object": "y"}],
     "then": {"type": "Food", "object": "y"}},
     {"if": [{"type": "Food", "object": "x"}],
     "then": {"type": "Likes", "subject": "Ravi", "object": "x"}}
  1
def forward chain(kb, query):
  derived facts = set()
  applied rules = set()
  changes = True
  while changes:
     changes = False
     for rule id, rule in enumerate(kb["rules"]):
       if rule id in applied rules:
          continue
       if conditions = rule["if"]
       satisfied = all(any(all(fact.get(k) == cond.get(k) for k in cond if k != 'object' and k !=
'subject')
                    for fact in kb["facts"]) for cond in if conditions)
       if satisfied:
          applied rules.add(rule id)
          derived fact = rule["then"]
          if not any(all(fact.get(k) == derived fact.get(k) for k in derived fact) for fact in
kb["facts"]):
           kb["facts"].append(derived fact)
           derived facts.add(tuple(derived fact.items()))
           changes = True
```

```
for fact in kb["facts"]:
    if all(fact.get(k) == query.get(k) for k in query):
        return True, derived_facts

return False, derived_facts

query = {"type": "Likes", "subject": "Ravi", "object": "Idli"}

result, derived_facts = forward_chain(knowledge_base, query)
print("\nDerived Facts:")
for fact in derived_facts:
    print(dict(fact))

print("\nQuery Result:")
if result:
    print(f"The query {query} is PROVEN.")
else:
    print(f"The query {query} is NOT PROVEN.")
```

OUTPUT-

```
Derived Facts:
{'type': 'Likes', 'subject': 'Ravi', 'object': 'x'}
{'type': 'Food', 'object': 'y'}

Query Result:
The query {'type': 'Likes', 'subject': 'Ravi', 'object': 'Idli'} is NOT PROVEN.
```

PROOF TREE-

PROOF TREE: Prove Ravi (ike side) Ravi enjoys all food Idu is a food Food (Banana) Food (Pizza) sam eat idu Jam not harmed by Idu eats (sam, idi) harm (sam, idi) eats (Eiel, enjoys (di)) Sam eat (ears, idu)	1110	
Prove Ravi like sidie Ravi enjoys all food Idu is a food Food (Banana) Food (Pizza) sam eatidu sam not narmed by Idu eats (Samildio narm (samildio) eats (Eill, enjoys idu)	10/2	PROOF TRE
Ravi enjoys all food Iduis a food Food (Banana) food (Pizza) sam eatidu sam not narmed by Idui eats (Sam, 1dto harm (sam, idii) eats (Bill, enjoys idii) (ravi, idii)		The sale soul but some in some
Ravi enjoys all food Iduis a food Food (Banana) Food (Pizza) sam eatidui sam not harmed by Idui eats (Sam, Idio harm (sam, Idii) eats (Biel, enjoys Idii) (eary, Idui	-	Prove Rays Iskes 1 410
eats (sam, Idlo harm (sam, Idli) eats (eats)	-	
eats (sam, Idlo harm (sam, Idli) eats (eats)		Rays encous
eats (sam, 1000 harm (sam, 100) eats (eats) enroys ldi)		lde so and
eats (som, 1000 harm (som, 100) eats (estel, enroys 100)	1.	
eats (som, 1010 narm (som, 1010) eats (Biel, enroys 1010)	-	ood (Barana) food (Duza)
eats (som, 1010 narm (som, 1010) eats (Biel, enroys 1010)	11	sam eat idli sam not
eats (Bill, enjoys (du)	11	
eats (Bill, enjoys (Ramidu)		nai m (sam, idli)
		Mas IMA prison eats (Bill en 2011)
		(du) (ravidu
	-(
Sam eats Idli sam not narmed by idli		sam of harmed by the
arrived by law		armed by law
: Dvm4		: Dvm9