Forward & Backward Chaining Inference

November 2021

1

FOPC – Inference Rules

- Inference rules for Quantifiers (∀, ∃)
  - Universal Instantiation
  - Existential Instantiation
- Generalized Modus Ponens
  - Unification
- Forward & Backward Chaining
  - Definite Clauses
  - Logic Programming in Prolog
- Resolution
  - Reductio ad Absurdum

2

2

Generalized Modus Ponens

For atomic sentences p_i, p_i', and q

where there is a substitution θ such that

SUBST(θ, p_i') = SUBST(θ, p_i), for all i

p_1'
p_2'
…
p_n'
$p_1' \wedge p_2' \wedge \ldots \wedge p_n' \Rightarrow q$

SUBST(θ, q)

e.g.

King(John)
Greedy(John)
∀x [King(x) ∧ Greedy(x) ⇒ Evil(x)]
θ is {x/John}
SUBST(θ, Evil(x)) is Evil(John)

3

3

Unification - UNIFY(p, q) = θ

How to find substitutions that make different logical expressions look identical?

For wffs p and q (sentences with universally quantified variables)

UNIFY(p, q) = θ

e.g.
p = Knows(John, x)
q = Knows(John, Mary) θ = {x/Mary}

p = Knows(John, x)
q = Knows(y, Bill) θ = {y/John, x/Bill}

p = Knows(John, x)
q = Knows(x, Elizabeth) θ = { } i.e. Fail! (no unification)

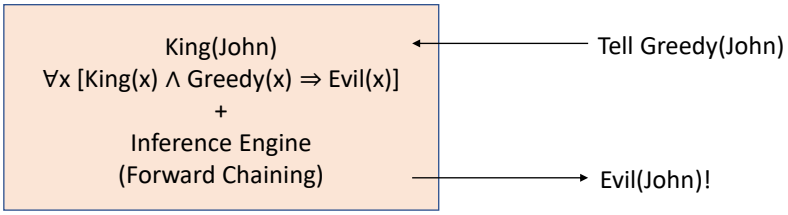
p = Knows(John, x_1)
q = Knows(x_2, Elizabeth) θ = {x_1/Elizabeth, x_2/John}

4

4

Forward Chaining Inference

- Tell-Ask Systems

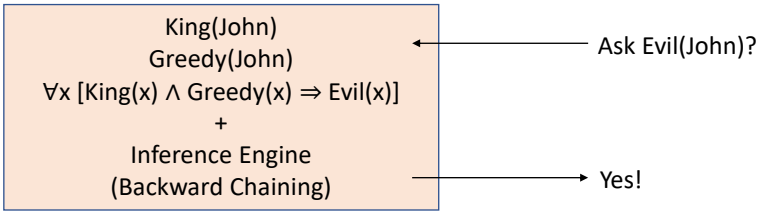


5

5

Backward Chaining Inference

- Tell-Ask Systems

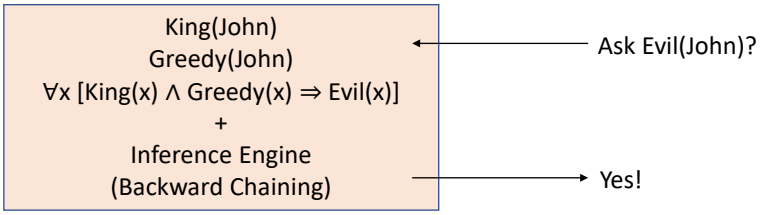


6

6

Backward Chaining Inference

- Tell-Ask Systems



- **Requires wffs to be in Definite Clause form!**

7

7

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

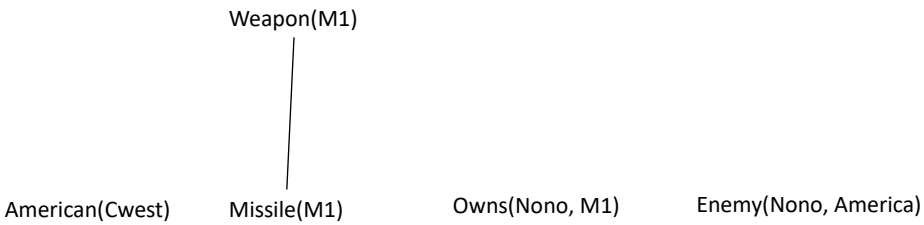
American(Cwest) Missile(M1) Owns(Nono, M1) Enemy(Nono, America)

8

8

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

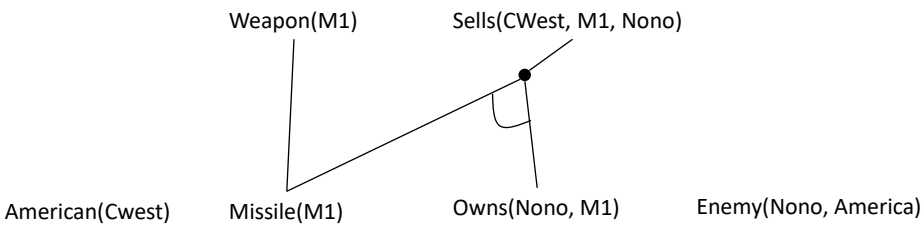


9

9

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

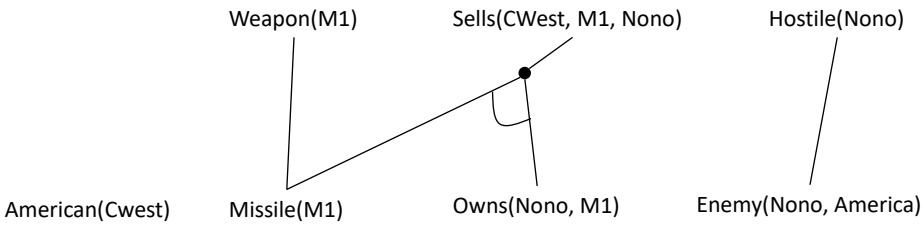


10

10

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

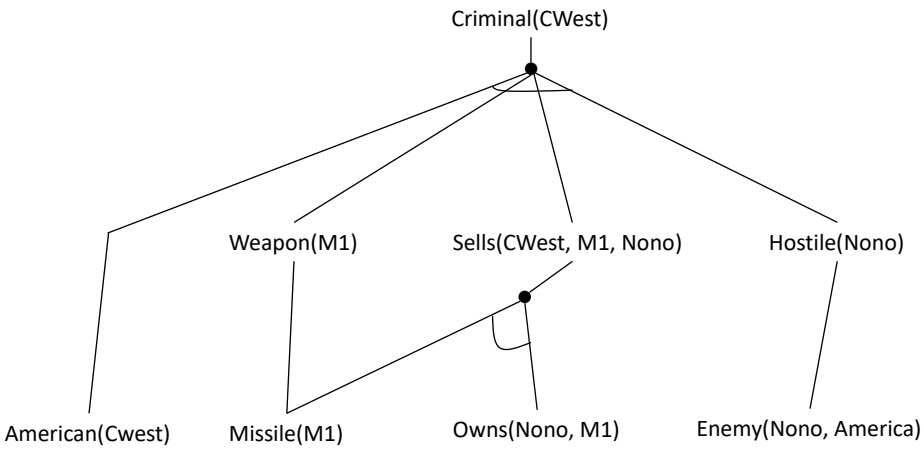


11

11

Forward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)



12

12

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x, America) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

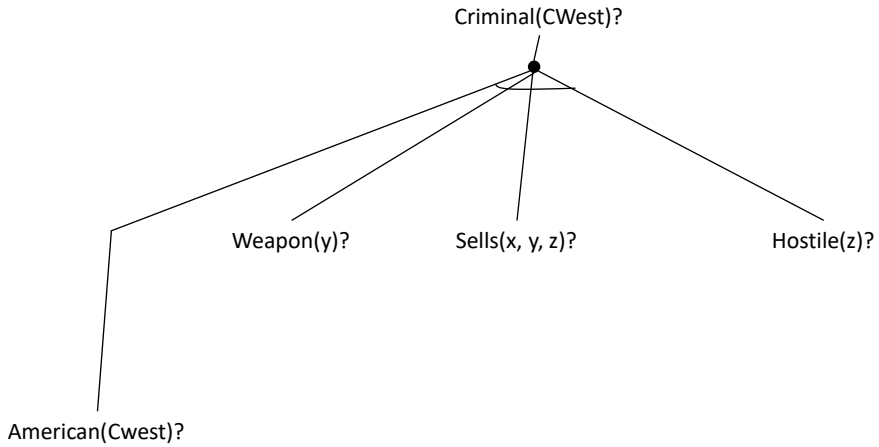
Criminal(CWest)?

13

13

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

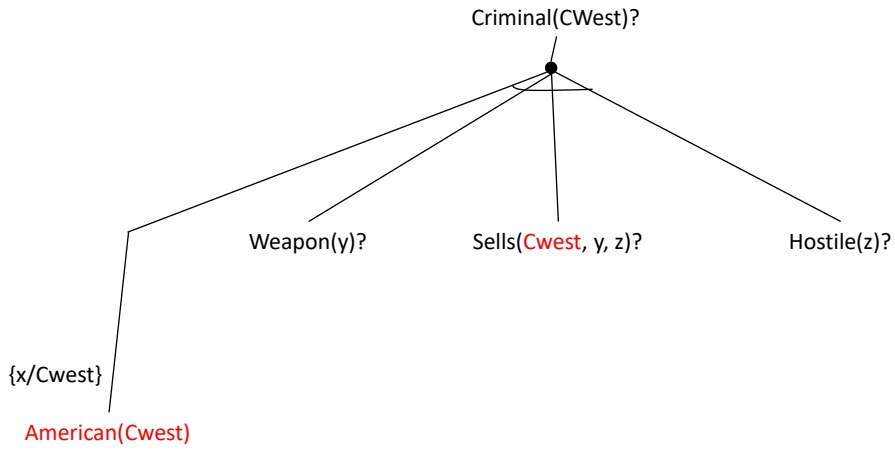


14

14

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

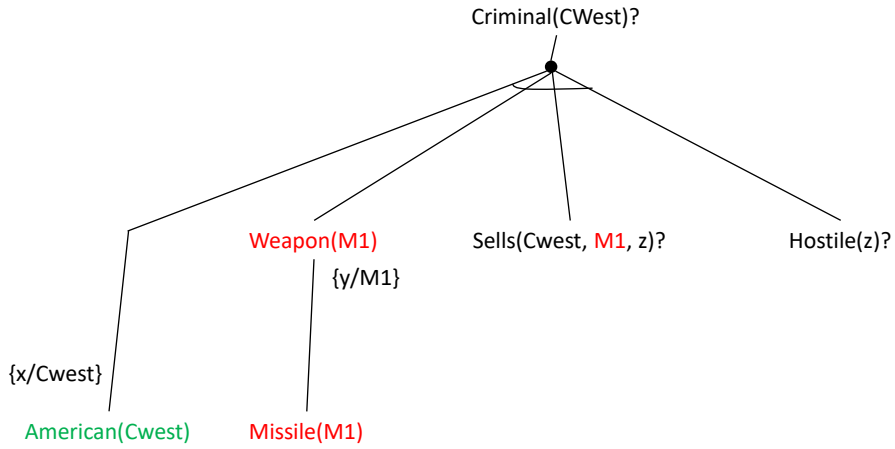


15

15

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

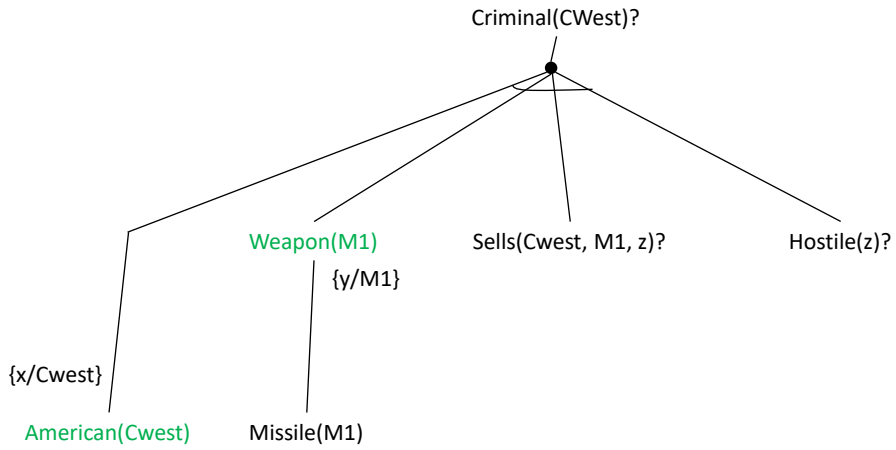


16

16

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

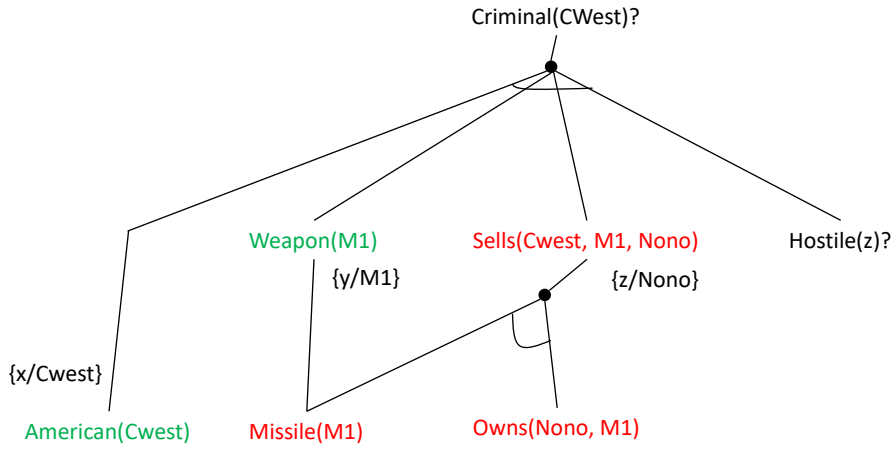


17

17

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

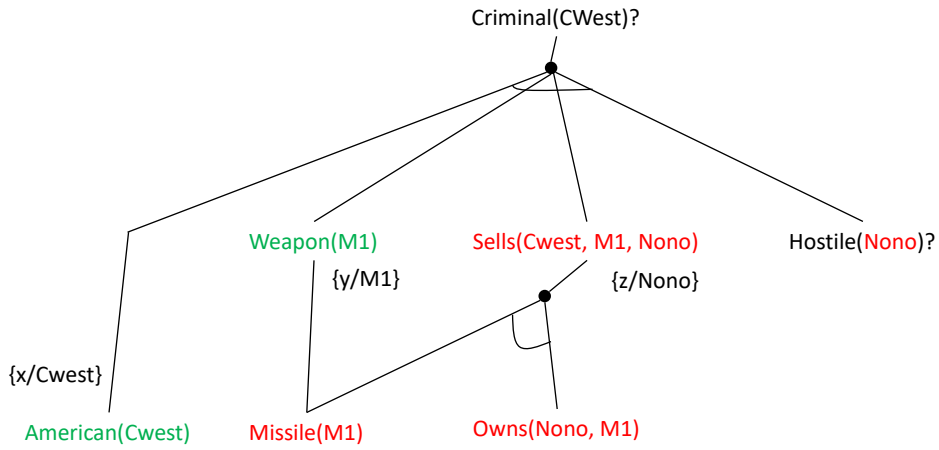


18

18

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

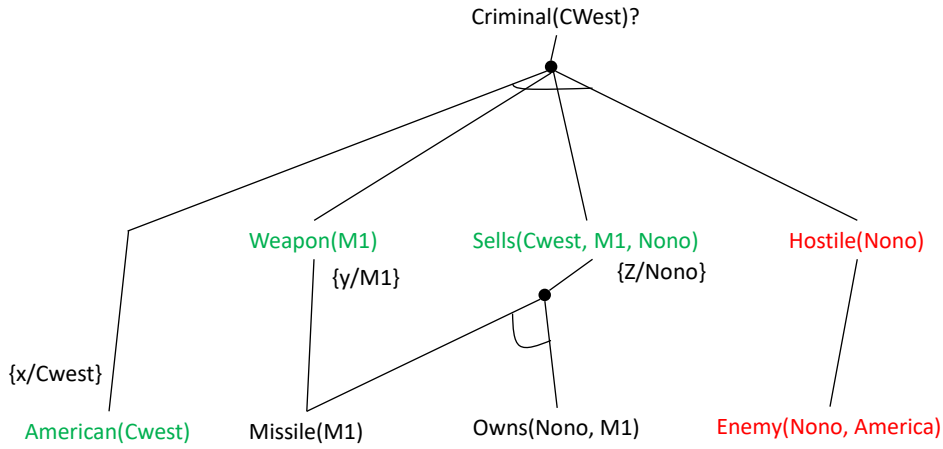


19

19

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

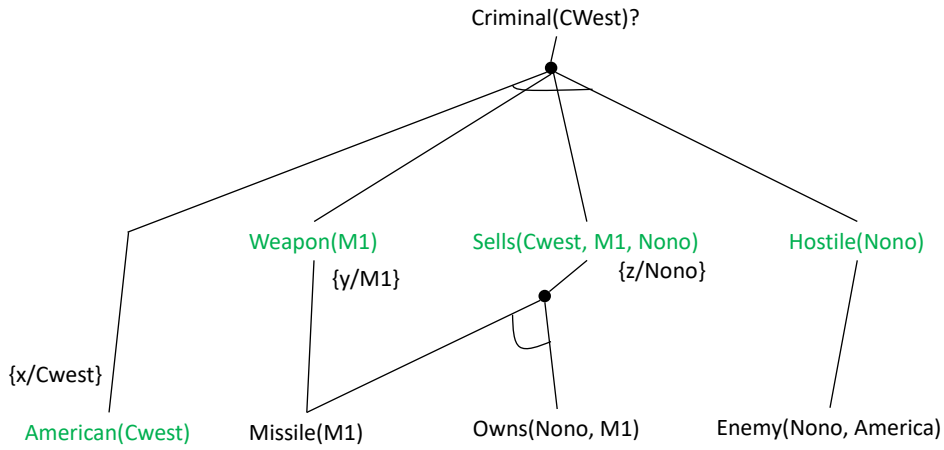


20

20

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)

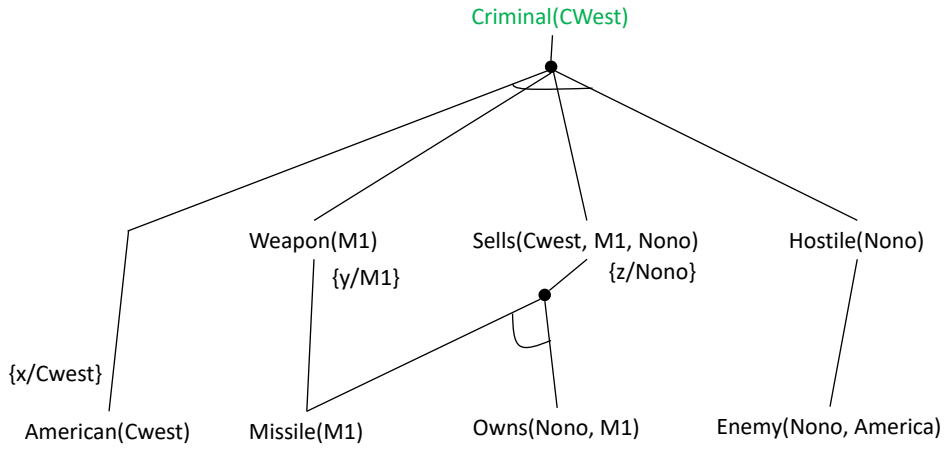


21

21

Backward Chaining

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono, M1)
Missile(M1)
Missile(x) ∧ Owns(Nono, x) ⇒ Sells(CWest, x, Nono)
Missile(x) ⇒ Weapon(x)
Enemy(America, x) ⇒ Hostile(x)
American(Cwest)
Enemy(Nono, America)



22

22

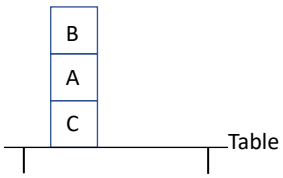
Reasoning about states & actions - Planning

Deepak Kumar

November 2021

23

Blocksworld



- **Objects** A, B, C, Table
- **Relations** On^2, $Clear^1$

On(C, Table) On(A, C) On(B, A)
Clear(B) Clear(A) Clear(C) Clear(Table)

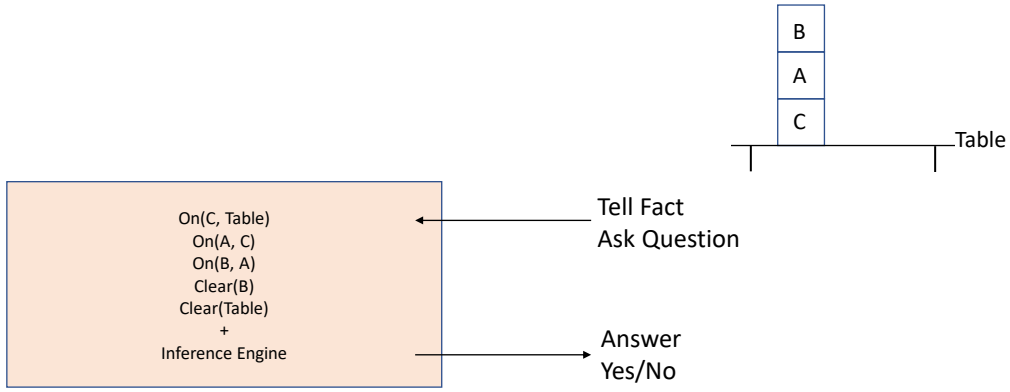
- Example Knowledge Base

Δ = {On(C, Table), On(A, C), On(B, A), Clear(B), Clear(Table)}

24

24

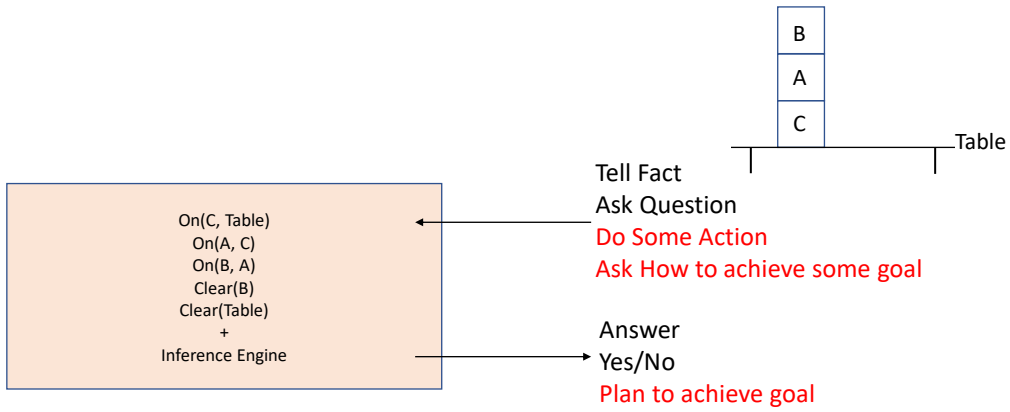
Tell-Ask Systems



25

25

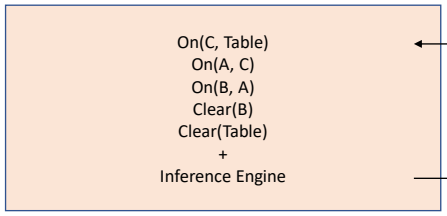
Tell-Ask-Do Systems



26

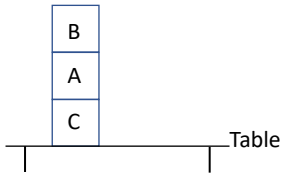
26

Tell-Ask-Do Systems



Tell Fact
Ask Question
Do Some Action
Ask How to achieve some goal

Answer
Yes/No
Plan to achieve goal

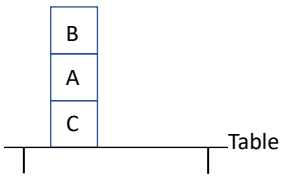


Can specify a goal as a wff:
Achieve On(A, B) ∧ On(B, C) ∧ On(C, Table)
Achieve On(A, B)
Achieve ∃x On(x, B)
Etc.

27

27

Reasoning about states & actions



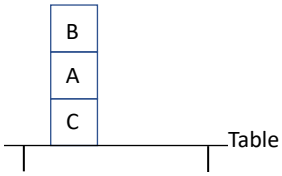
Can specify a goal as a wff:
Achieve On(A, B) ∧ On(B, C) ∧ On(C, Table)
Achieve On(A, B)
Achieve ∃x On(x, B)
Etc.

We can test if a goal is satisfied by the KB:
$\Delta \models_R^? \text{Goal}$

But, how to find a set/sequence of actions for achieving a goal satisfied by a wff?

28

Reasoning about states & actions



Can specify a goal as a wff:
Achieve On(A, B) ∧ On(B, C) ∧ On(C, Table)
Achieve On(A, B)
Achieve ∃x On(x, B)
Etc.

We can test if a goal is satisfied by the KB:
$\Delta \models_R^? \text{Goal}$

But, how to find a set/sequence of actions for achieving a goal satisfied by a wff?

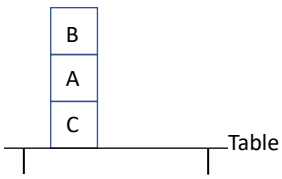
Search a space of logical expressions/state descriptions…

29

Situation Calculus

- McCarthy & Hayes, 1969
- Went out of fashion, but made a come back – Reiter 2001, Lakemeyer, 2014.
- If S_0 is the state:

  S_0 = {On(B, A), On(A, C), On(C, Table),…}



- Add the state to the relation:

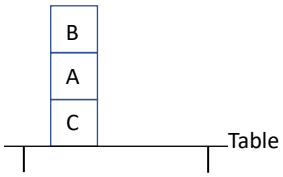
  {On(A, B, S_0), On(A, C, S_0), On(C, Table, S_0)…}

30

Situation Calculus

- McCarthy & Hayes, 1969
- Went out of fashion, but made a come back – Reiter 2001, Lakemeyer, 2014.
- If so is the state:

  S_0 = {On(B, A), On(A, C), On(C, Table),…}



- Add the state to the relation:

  {On(A, B, S_0), On(A, C, S_0), On(C, Table, S_0)…}

These are called Fluents.

31

Situation Calculus

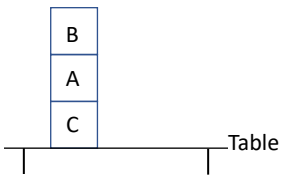
- Add the state to the relation:

  {On(A, B, S_0), On(A, C, S_0), On(C, Table, S_0)…}

- Now, we can make statements about states:

  ∀x∀y∀s [On(x, y, s) ∧ ¬(y=Table) ⇒ ¬Clear(y, s)]

  ∀s Clear(Table, s)



- We can now prove:

  ¬Clear(A, S_0)
  Clear(Table, S_0)

32

Representing Actions – Action Schema

- **Action Schema:** move(x, y, z) - move x, from y, to z
- **Function do():** if α is an action and σ is a state then
  do(α, σ) → σ′
  σ′ is the state resulting from doing α in σ
- **Effects of actions** represented by wffs:
  ∀x∀y∀s∀z [On(x, y, s) ∧ Clear(x, s) ∧ Clear(z, s) ∧ x ≠ z ⇒ On(x, z, do(move(x, y, z), s)]
  ∀x∀y∀s∀z [On(x, y, s) ∧ Clear(x, s) ∧ Clear(z, s) ∧ x ≠ z ⇒ ¬On(x, y, do(move(x, y, z), s)]
  These are positive(+) and negative(-) axioms.
- Need these for every fluent-action pair!

33

Axioms for Clear and move

∀x∀y∀s∀z [On(x, y, s) ∧ Clear(x, s) ∧ Clear(z, s) ∧ (x ≠ z) ∧ (y ≠ z)
⇒ Clear(y, do(move(x, y, z), s)]

∀x∀y∀s∀z [On(x, y, s) ∧ Clear(x, s) ∧ Clear(z, s) ∧ (x ≠ z) ∧ (z ≠ Table)
⇒ ¬Clear(z, do(move(x, y, z), s)]

34

Axioms for Clear and move

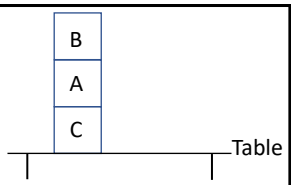
$\forall x \forall y \forall s \forall z$ [On(x, y, s) $\wedge$ Clear(x, s) $\wedge$ Clear(z, s) $\wedge$ (x $\neq$ z) $\wedge$ (y $\neq$ z)
$\Rightarrow$ Clear(y, do(move(x, y, z), s)]

$\forall x \forall y \forall s \forall z$ [On(x, y, s) $\wedge$ Clear(x, s) $\wedge$ Clear(z, s) $\wedge$ (x $\neq$ z) $\wedge$ (z $\neq$ Table)
$\Rightarrow$ $\neg$Clear(z, do(move(x, y, z), s)]

Preconditions & Effects

35

Axioms for Clear and move



$\forall x \forall y \forall s \forall z$ [On(x, y, s) $\wedge$ Clear(x, s) $\wedge$ Clear(z, s) $\wedge$ (x $\neq$ z) $\wedge$ (y $\neq$ z)
$\Rightarrow$ Clear(y, do(move(x, y, z), s)]

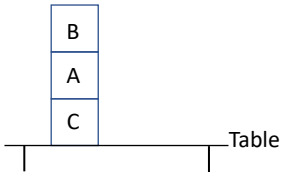
$\forall x \forall y \forall s \forall z$ [On(x, y, s) $\wedge$ Clear(x, s) $\wedge$ Clear(z, s) $\wedge$ (x $\neq$ z) $\wedge$ (z $\neq$ Table)
$\Rightarrow$ $\neg$Clear(z, do(move(x, y, z), s)]

do(move(B, A, Table), S_0)

We get On(B, Table, do(move(B, A, Table), S_0))
$\neg$On(B, A, do(move(B, A, Table), S_0))
Clear(A, do(move(B, A, Table), S_0))

36

Axioms for Clear and move



$\forall x \forall y \forall s$ [On(x, y, s) $\wedge$ Clear(x, s) $\wedge$ Clear(z, s) $\wedge$ (x $\neq$ z) $\wedge$ (y $\neq$ z)
$\Rightarrow$ Clear(y, do(move(x, y, z), s)]

$\forall x \forall y \forall s$ [On(x, y, s) $\wedge$ Clear(x, s) $\wedge$ Clear(z, s) $\wedge$ (x $\neq$ z) $\wedge$ (z $\neq$ Table)
$\Rightarrow \neg$Clear(z, do(move(x, y, z), s)]

For do(move(B, A, Table), S_0)

We get On(B, Table, do(move(B, A, Table), S_0))
$\neg$On(B, A, do(move(B, A, Table), S_0))
Clear(A, do(move(B, A, Table), S_0))

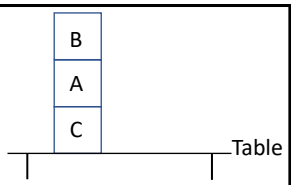
If S_1 = do(move(B, A, Table), S_0), we have

On(B, Table, S_1)
$\neg$On(B, A, S_1)
Clear(A, S_1)

Plus, all wffs for S_0 are still true!

37

And…more axioms



- Not only do we have to say what changes, we have to specify what doesn't change…everything that doesn't change!

  After move(B, A, Table)

  C is still on the Table in S_0 and S_1
  B is still clear in S_0 and S_1
  …etc…

- Need more axioms…

38

Frame Axioms

- Axioms for Move-On

$\forall x \forall y \forall s \forall u \forall v \forall z$ [On(x, y, s) $\wedge$ (x $\neq$ u) $\Rightarrow$ On(x, y, do(move(u, v, z), s)]

$\forall x \forall y \forall s \forall u \forall v \forall z$ [$\neg$On(x, y, s) $\wedge$ [(x $\neq$ u) $\vee$ (y $\neq$ z)] $\Rightarrow$ $\neg$On(x, y, do(move(u, v, z), s)]

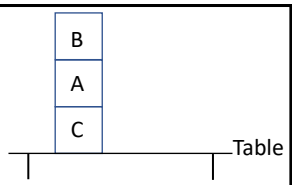
- Axioms for Move-Clear

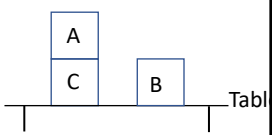
$\forall x \forall y \forall s \forall z$ [Clear(u, s) $\wedge$ (u $\neq$ z) $\Rightarrow$ Clear(u, do(move(x, y, z), s)]

$\forall x \forall y \forall s \forall z$ [$\neg$Clear(u, s) $\wedge$ (u $\neq$ y) $\Rightarrow$ $\neg$Clear(u, do(move(x, y, z), s)]

39

Planning using Situation Calculus

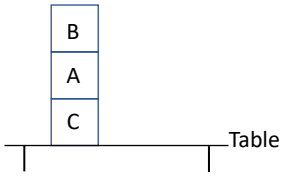




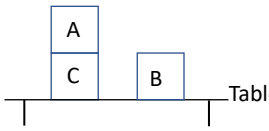
- To get a plan that achieves a goal: Υ(s)
- We need to prove: $\exists s\ \Upsilon(s)$
- We'll get the plan by using Green's Trick: $\exists s\ \Upsilon(s) \vee$ Answer(s)
- Goal: $\exists s$ On(B, Table, s)
- Prove: $\neg$ [$\exists s$ On(B, Table, s)] $\vee$ Answer(s)
- i.e. Add the clause: $\neg$ On(B, Floor, s) $\vee$ Answer(s) to KB (in clause form)
- Answer(do(move(B, A, Table), S_0)

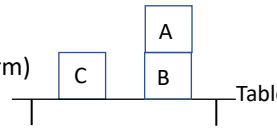
40

Planning using Situation Calculus



- To get a plan that achieves a goal: ϒ(s)
- We need to prove: ∃s ϒ(s)
- We'll get the plan by using Green's Trick: ∃s ϒ(s) ∨ Answer(s)
- Goal: ∃s On(B, Table, s)
- Example 1: Prove: ¬ [∃s On(B, Table, s)] ∨ Answer(s)
  i.e. Add the clause: ¬ On(B, Table, s) ∨ Answer(s) to KB (in clause form)
  Answer(do(move(B, A, Table), S_0)
- Example 2: Prove: ¬ [∃s On(A, B, s) ∧ On(B, Table, s)] ∨ Answer(s)
  i.e. Add the clause: ¬On(A, B, s) ∨ ¬On(B, Table, s) ∨ Answer(s) to KB (in clause form)
  Answer(do(move(A, C, B), do(move(B, A, Table), S_0)))





41

41

Situation Calculus Planning - Summary

- **Action Schema:** move(x, y, z) - move x, from y, to z
- **Function do():** do(α, σ) → σ′
- **Effects Axioms**

  ∀x∀y∀s∀z [On(x, y, s) ∧ Clear(x, s) ∧ Clear(z, s) ∧ x ≠ z ⇒ On(x, z, do(move(x, y, z), s)]

  ∀x∀y∀s∀z [On(x, y, s) ∧ Clear(x, s) ∧ Clear(z, s) ∧ x ≠ z ⇒ ¬On(x, , do(move(x, y, z), s)]

  Need these for every fluent-action pair!
- Also need Frame Axioms
- Prove, using Green's Trick to extract plan: ∃s ϒ(s) ∨ Answer(s)
- Using resolution on: ¬[∃s ϒ(s) ∨ Answer(s)]

42

Problems with Situation Calculus Planning

- Too many Frame Axioms
- Proof effort is too large for even simple problems
- There are more problems:
  - Frame Problem
  - Qualifications Problem
  - Ramification Problem
  - Etc.

43

44