CS 4260 and CS 5260
Vanderbilt University

Lecture on Inference, Planning, and Learning with First-Order Representations

This lecture assumes that you have

- Read Chapter 13, through 13.4.3 of ArtInt

ArtInt: Poole and Mackworth, Artificial Intelligence 2E
at http://artint.info/2e/html/ArtInt2e.html

Douglas H. Fisher

First Order Logic
Inference

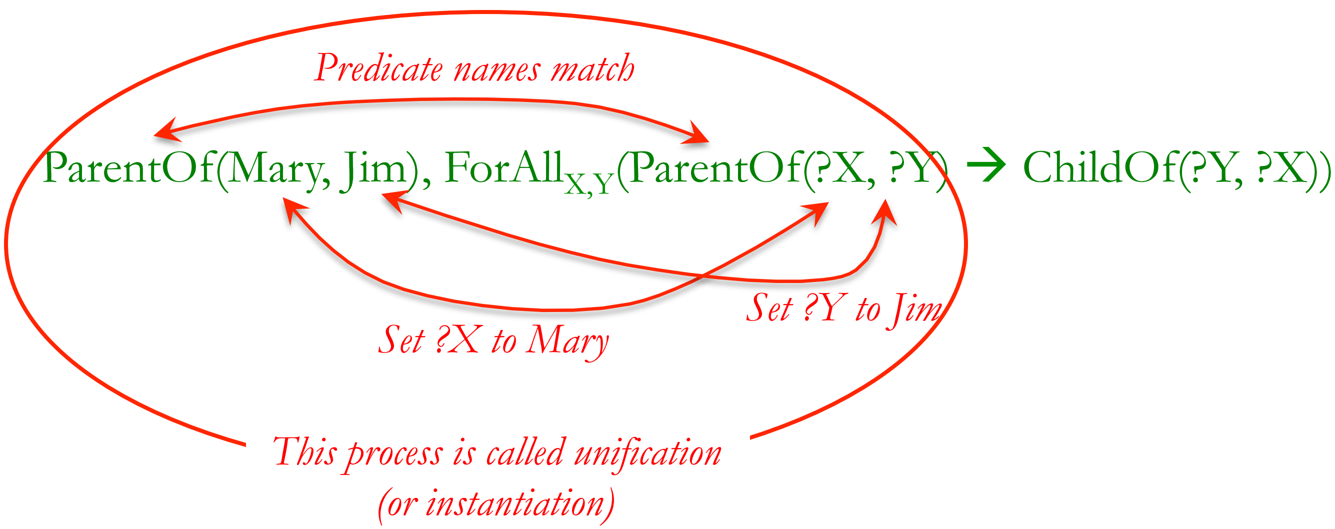
Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- $ForAll_{?X,?Y}(ParentOf(?X, ?Y) \rightarrow ChildOf(?Y, ?X))$
- $ForAll_{?X,?Y}(ParentOf(?X, ?Y) \leftarrow ChildOf(?Y, ?X))$
- $ForAll_{?X,?Y,?Z}(ParentOf(?X, ?Y) \land ParentOf(?Y, ?Z)) \rightarrow GrandParentOf(?X, ?Z)$

First Order Logic
Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- $\text{ForAll}_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- $\text{ForAll}_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- $\text{ForAll}_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z)) → GrandParentOf(?X, ?Z)



Douglas H. Fisher

First Order Logic
Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- $\text{ForAll}_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- $\text{ForAll}_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- $\text{ForAll}_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z)) → GrandParentOf(?X, ?Z)

Mary Mary

ParentOf(Mary, Jim), $\text{ForAll}_{?X,?Y}$(ParentOf(~~?X~~, ~~?Y~~) → ChildOf(~~?Y~~, ~~?X~~))

Jim Jim

Substitute variable values throughout

First Order Logic
Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- $ForAll_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- $ForAll_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- $ForAll_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z)) → GrandParentOf(?X, ?Z)

ParentOf(Mary, Jim), ~~ForAll~~$_{X,Y}$(ParentOf(Mary, Jim) → ChildOf(Jim, Mary))

Modus Ponens now applies

ParentOf(Mary, Jim), ~~ForAll~~$_{X,Y}$(ParentOf(Mary, Jim) → ChildOf(Jim, Mary)) |- ChildOf(Jim, Mary)

Douglas H. Fisher

First Order Logic
Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- $ForAll_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- $ForAll_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- $ForAll_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z))
  → GrandParentOf(?X, ?Z))
- $ForAll_{?X,?Y}$(GrandParentOf(?X, ?Y) → GrandParent(?X))

GrandParent(Mary) ?

Douglas H. Fisher

First Order Logic Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- $\text{ForAll}_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- $\text{ForAll}_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- $\text{ForAll}_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z))
  → GrandParentOf(?X, ?Z)
- $\text{ForAll}_{?X,?Y}$(GrandParentOf(?X, ?Y) → GrandParent(?X))

unify

GrandParent(Mary)

GrandParentOf(Mary, ?Y)

Douglas H. Fisher

First Order Logic
Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- $\text{ForAll}_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- $\text{ForAll}_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- $\text{ForAll}_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z))
  → GrandParentOf(?X, ?Z)
- $\text{ForAll}_{?X,?Y}$(GrandParentOf(?X, ?Y) → GrandParent(?X))

GrandParent(Mary)

↑

unify

GrandParentOf(Mary, ?Y)

Douglas H. Fisher

First Order Logic
Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- ForAll$_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z))
  → GrandParentOf(?X, ?Z)
- ForAll$_{?X,?Y}$(GrandParentOf(?X, ?Y) → GrandParent(?X))

GrandParent(Mary)

↑

GrandParentOf(Mary, ?Z)

↑

ParentOf(Mary, ?Y) ∧ ParentOf(?Y, ?Z)

First Order Logic Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- ForAll$_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z))
  → GrandParentOf(?X, ?Z)
- ForAll$_{?X,?Y}$(GrandParentOf(?X, ?Y) → GrandParent(?X))

unify

GrandParent(Mary)

GrandParentOf(Mary, ?Z)

ParentOf(Mary, ?Y) ∧ ParentOf(?Y, ?Z)

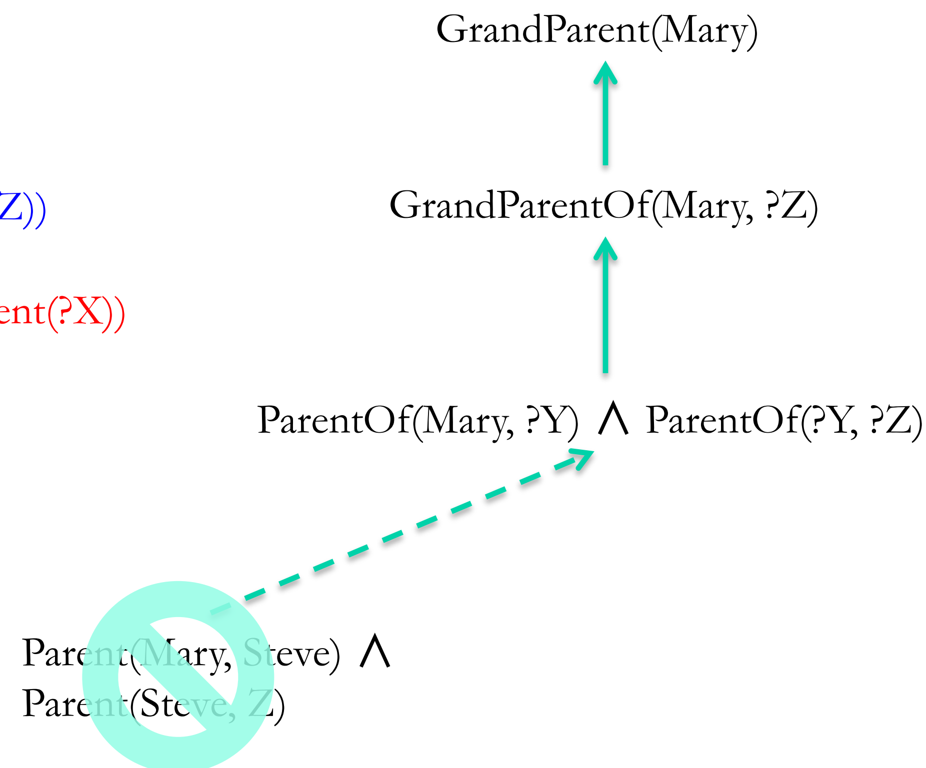
ParenOft(Mary, Steve) ∧
ParentOf(Steve, ?Z)

Douglas H. Fisher

First Order Logic Inference

Knowledge Base

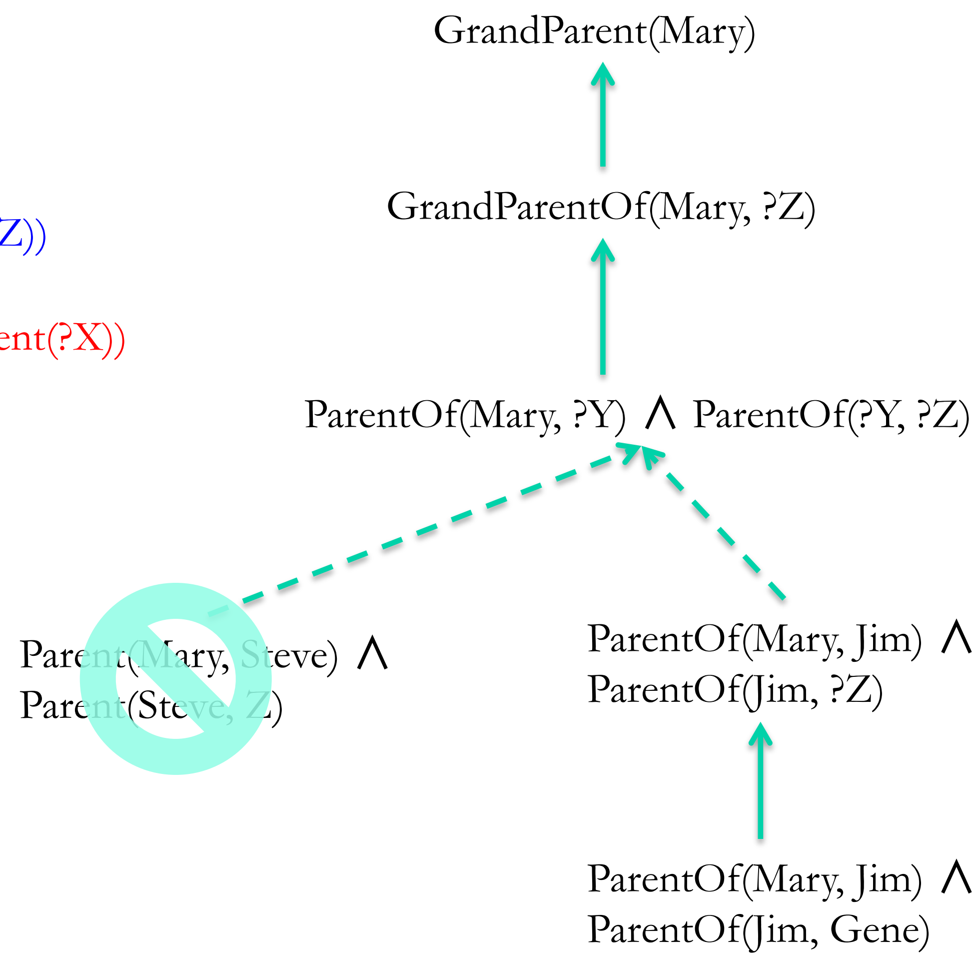
- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- ForAll$_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z))
  → GrandParentOf(?X, ?Z))
- ForAll$_{?X,?Y}$(GrandParentOf(?X, ?Y) → GrandParent(?X))



First Order Logic Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- ForAll$_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z))
  → GrandParentOf(?X, ?Z)
- ForAll$_{?X,?Y}$(GrandParentOf(?X, ?Y) → GrandParent(?X))

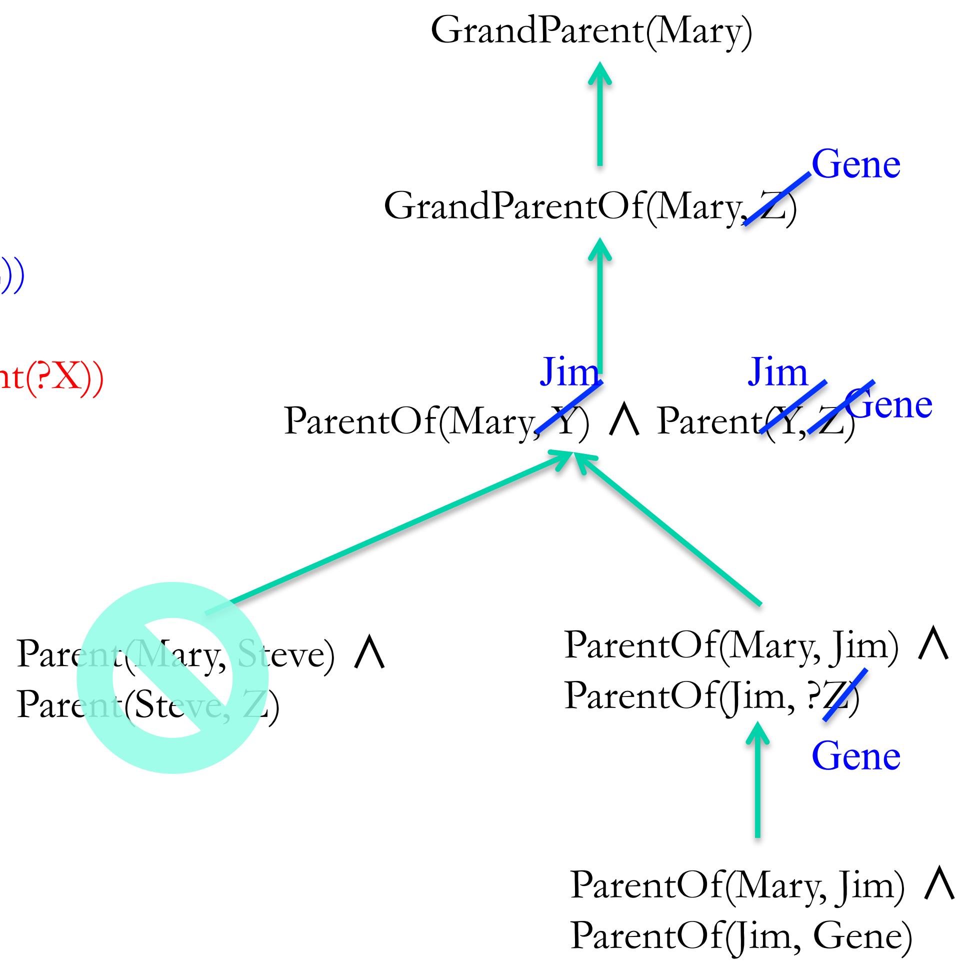


Douglas H. Fisher

First Order Logic
Inference

Knowledge Base

- ParentOf(Mary, Jim)
- ParentOf(Mary, Steve)
- ParentOf(Jim, Gene)
- ParentOf(Hua, Jia)
- ChildOf(Jim, Mary)
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) → ChildOf(?Y, ?X))
- ForAll$_{?X,?Y}$(ParentOf(?X, ?Y) ← ChildOf(?Y, ?X))
- ForAll$_{?X,?Y,?Z}$(ParentOf(?X, ?Y) ∧ ParentOf(?Y, ?Z))
  → GrandParentOf(?X, ?Z)
- ForAll$_{?X,?Y}$(GrandParentOf(?X, ?Y) → GrandParent(?X))



Douglas H. Fisher

Resolution Proofs

Axioms:

1. Exists(?x) (Dog(?x) and Owns(Jack, ?x))
2. ForAll(?x) (Exists(?y) (Dog(?y) and Owns(?x,?y)) → AnimalLover(?x))
3. ForAll(?x) (AnimalLover(?x) → (ForAll(?y) (Animal(?y) → ~Kills(?x,?y))))
4. Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ForAll(?x) (Cat(?x) → Animal(?x))

~Exists(?y) (Dog(?y) and Owns(?x,?y)) or AnimalLover(?x))

ForAll(?y) ~(Dog(?y) and Owns(?x,?y)) or AnimalLover(?x))

Clause Form:

1. Dog(D), Owns(Jack, D)
2. ~Dog(?y) or ~Owns(?x,?y) or AnimalLover(?x)
3. ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)
4, Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

Douglas H. Fisher

Axioms in Clause Form: Prove: Kills(Curiosity, Tuna)

1. A) Dog(D), B) Owns(Jack, D)
2. ~Dog(?y) or ~Owns(?x,?y) or AnimalLover(?x)
3. ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)
4. Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

negation: ~Kills(Curiosity, Tuna)

In a proof by contradiction, assume the negation is true

Resolution Refutation (contradiction) Proof:

~Kills(Curiosity, Tuna)   4 Kills(Jack, Tuna) or Kills(Curiosity, Tuna)

If both these statements are true
then Kills(Jack, Tuna) must be true

Axioms in Clause Form:

Prove: Kills(Curiosity, Tuna)

1. A) Dog(D), B) Owns(Jack, D)
2. ~Dog(?y) or ~Owns(?x,?y) or AnimalLover(?x)
3. ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)
4. Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

negation: ~Kills(Curiosity, Tuna)

Resolution Refutation (contradiction) Proof:

~Kills(Curiosity, Tuna)

4 Kills(Jack, Tuna) or Kills(Curiosity, Tuna)

{}

No variables involved, so substitution list is empty

Kills(Jack, Tuna)

Axioms in Clause Form:

Prove: Kills(Curiosity, Tuna)

1. A) Dog(D), B) Owns(Jack, D)
2. ~Dog(?y) or ~Owns(?x,?y) or AnimalLover(?x)
3. ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)
4. Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

negation: ~Kills(Curiosity, Tuna)

Resolution Refutation (contradiction) Proof:

~Kills(Curiosity, Tuna)   4   3 ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)

Jack Tuna Jack Tuna

{}

Kills(Jack, Tuna)

{?z/Jack, ?w/Tuna}

If both statements true, then it must be true that ~AnimalLover(Jack) or ~Animal(Tuna)

Axioms in Clause Form:

1. A) Dog(D), B) Owns(Jack, D)
2. ~Dog(?y) or ~Owns(?x,?y) or AnimalLover(?x)
3. ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)
4. Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

Prove: Kills(Curiosity, Tuna)

negation: ~Kills(Curiosity, Tuna)

Resolution Refutation (contradiction) Proof:

~Kills(Curiosity, Tuna)   4   3 ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)

{}

Kills(Jack, Tuna)

{?z/Jack, ?w/Tuna}

~AnimalLover(Jack) or ~Animal(Tuna)

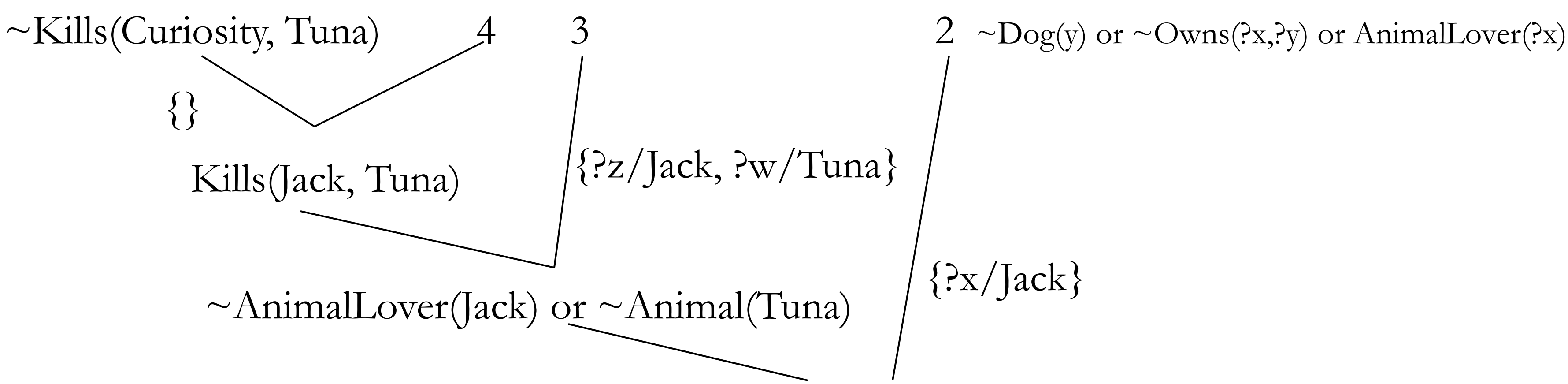
Axioms in Clause Form:

Prove: Kills(Curiosity, Tuna)

1. A) Dog(D), B) Owns(Jack, D)
2. ~Dog(?y) or ~Owns(?x,?y) or AnimalLover(?x)
3. ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)
4. Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

negation: ~Kills(Curiosity, Tuna)

Resolution Refutation (contradiction) Proof:



If AnimalLover(Jack) then ~Animal(Tuna)
If ~AnimaLover(Jack) then ~Dog(?y) or ~Owns(Jack, ?y),
So ~Dog(?y) or ~Owns(Jack,?y) or ~Animal(Tuna)

Douglas H. Fisher

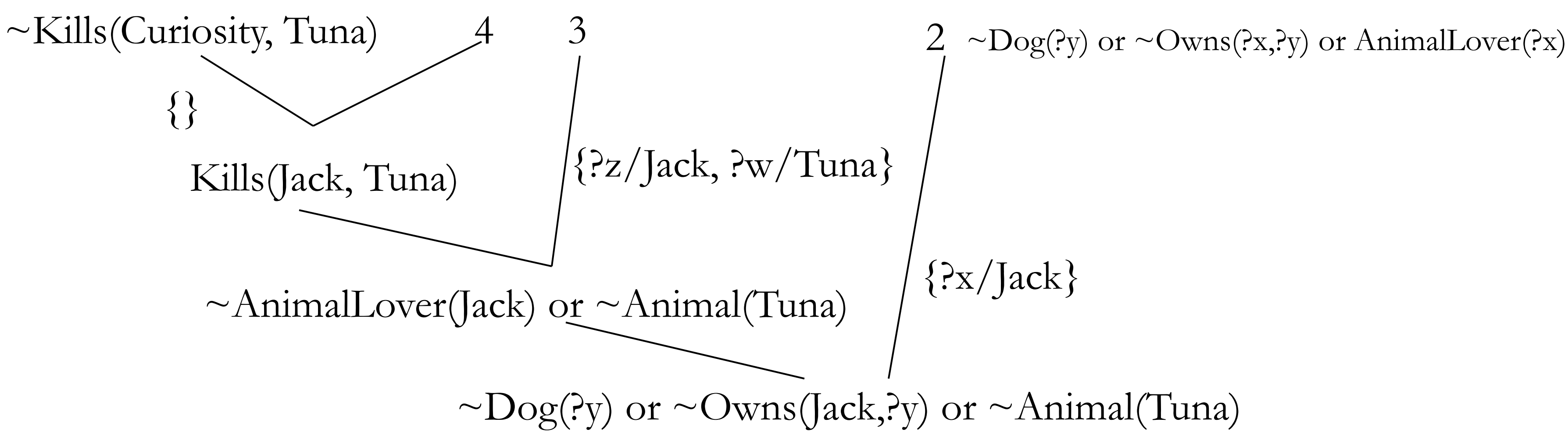
Axioms in Clause Form:

Prove: Kills(Curiosity, Tuna)

1. A) Dog(D), B) Owns(Jack, D)
2. ~Dog(?y) or ~Owns(?x,?y) or AnimalLover(?x)
3. ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)
4. Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

negation: ~Kills(Curiosity, Tuna)

Resolution Refutation (contradiction) Proof:



Axioms in Clause Form:

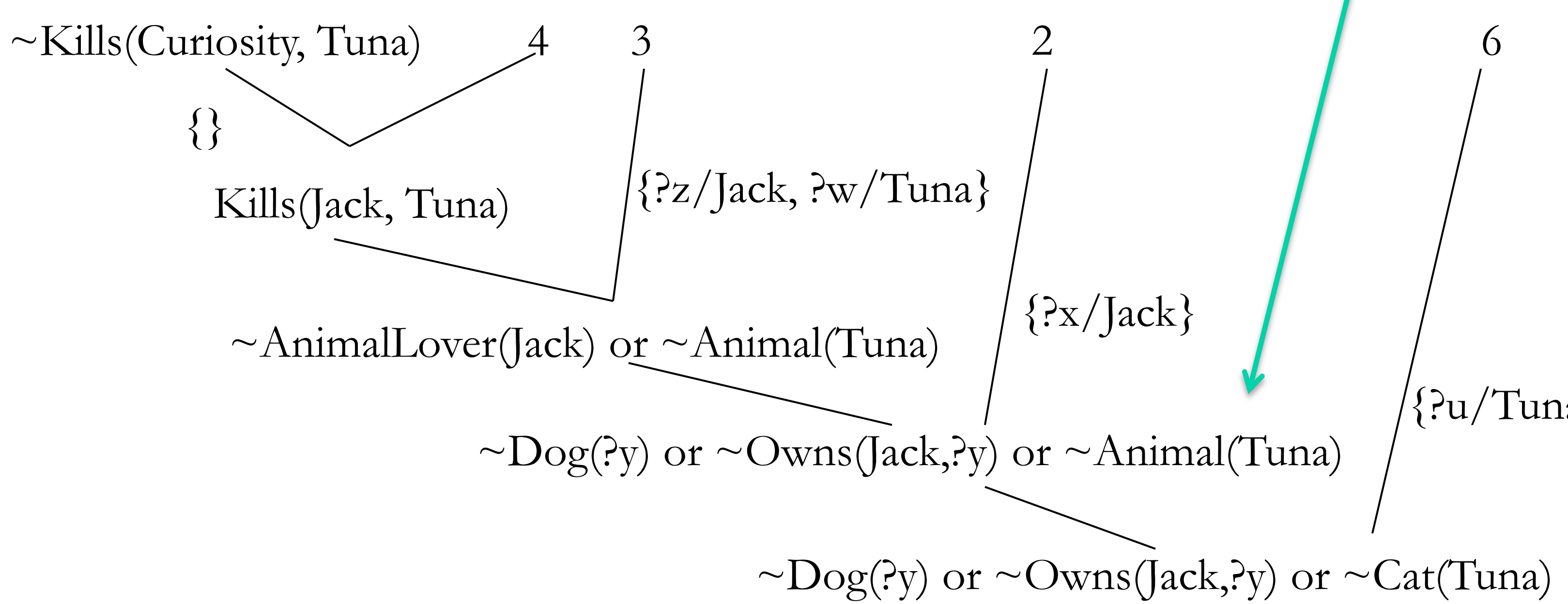
Prove: Kills(Curiosity, Tuna)

1. A) Dog(D), B) Owns(Jack, D)
2. ~Dog(?y) or ~Owns(?x,?y) or AnimalLover(?x)
3. ~AnimalLover(?z) or ~Animal(?w) or ~Kills(?z, ?w)
4. Kills(Jack, Tuna) or Kills(Curiosity, Tuna)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

negation: ~Kills(Curiosity, Tuna)

Look for propositions that unify, one negated and one not

Resolution Refutation (contradiction) Proof:



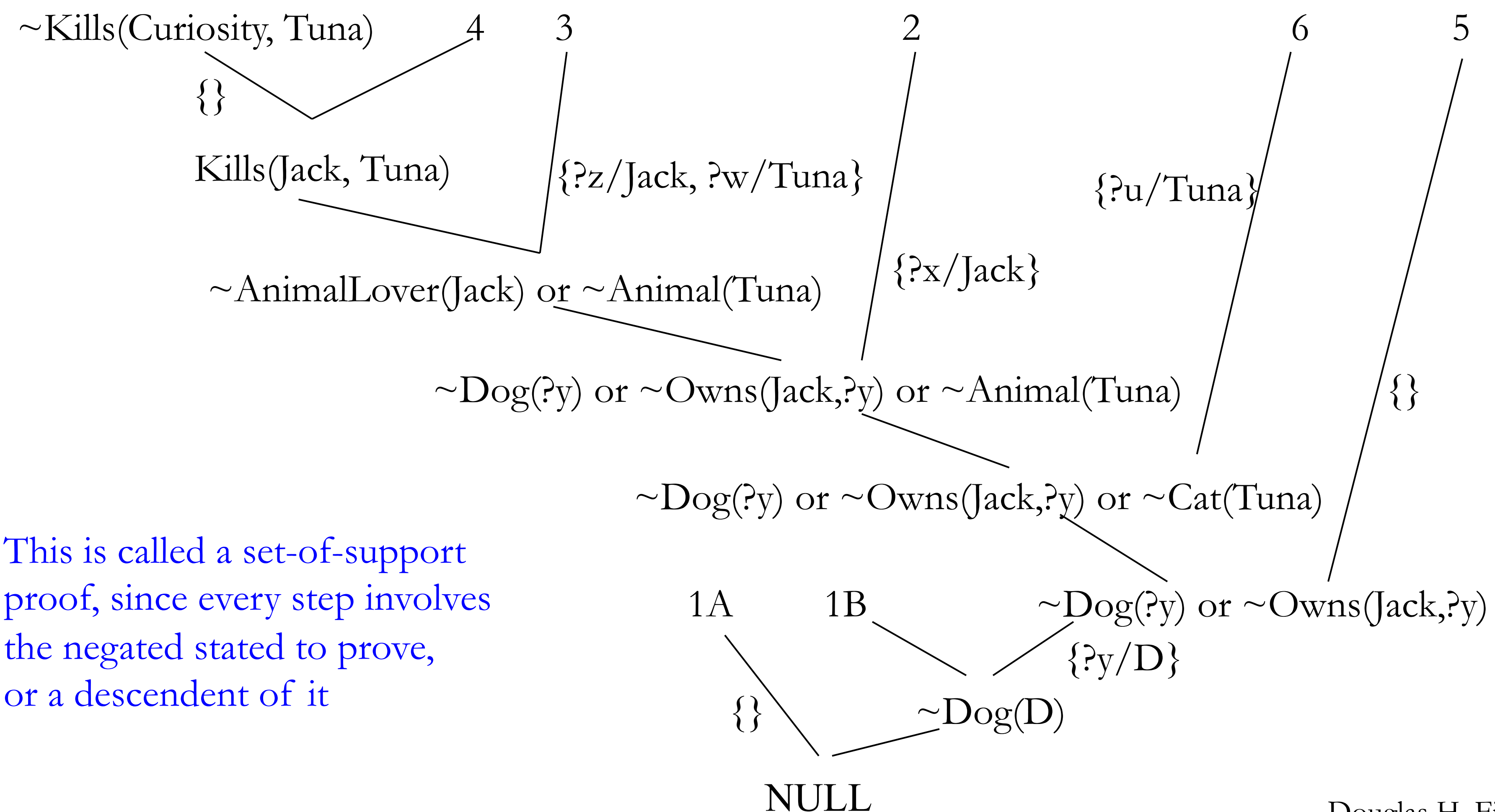
Douglas H. Fisher

Axioms in Clause Form:

1. A) Dog(D), B) Owns(Jack, D)
5. Cat(Tuna)
6. ~Cat(?u) or Animal(?u)

Prove: Kills(Curiosity, Tuna)

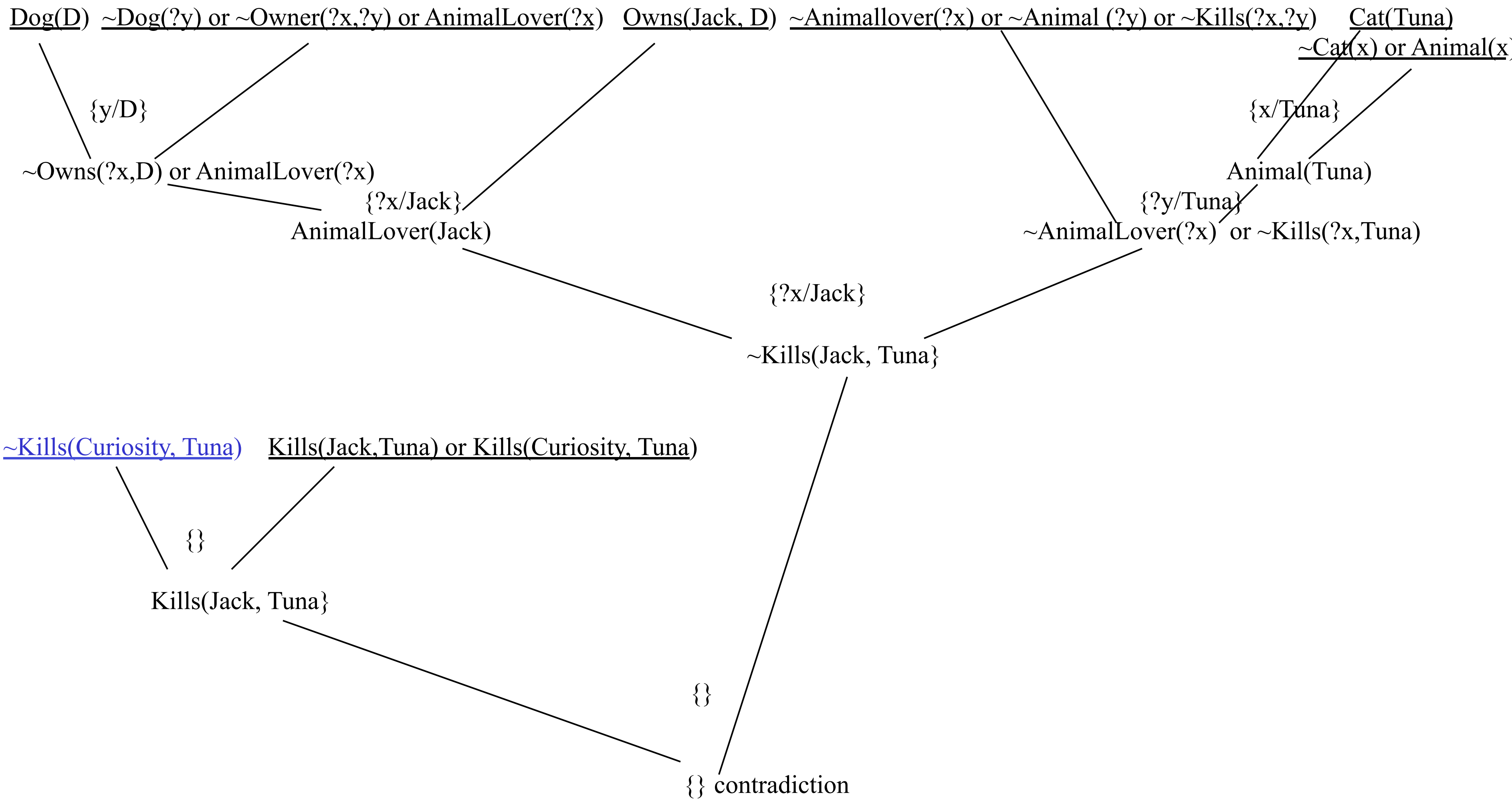
negation: ~Kills(Curiosity, Tuna)



This is called a set-of-support proof, since every step involves the negated stated to prove, or a descendent of it

Douglas H. Fisher

Equivalent (refutation) proof for Curiosity (not set-of-support)

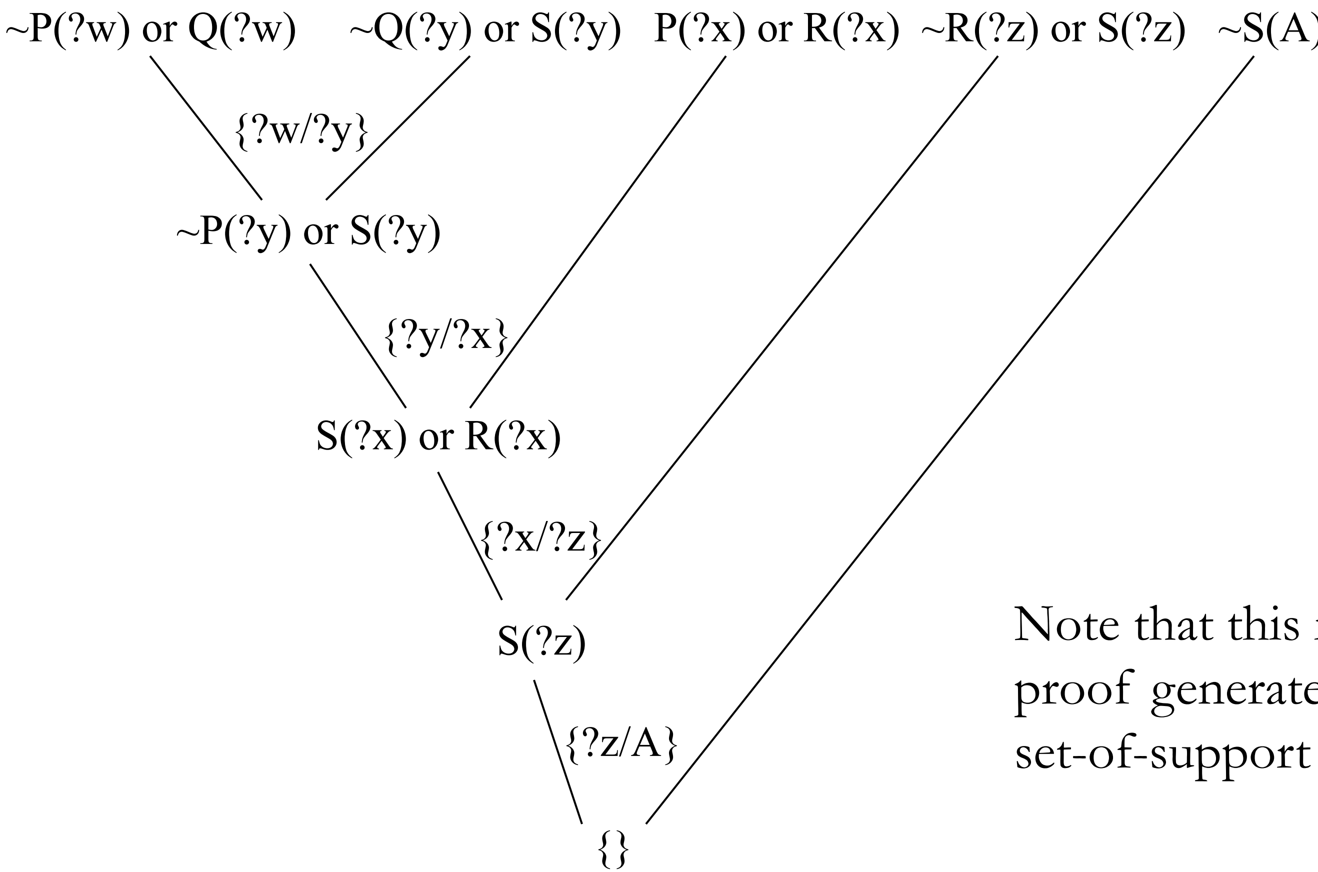


Douglas H. Fisher

Prove S(A) by resolution refutation proof

~P(?w) or Q(?w)   ~Q(?y) or S(?y)   P(?x) or R(?x)   ~R(?z) or S(?z)

Equivalent (refutation) proof for proving S(A)

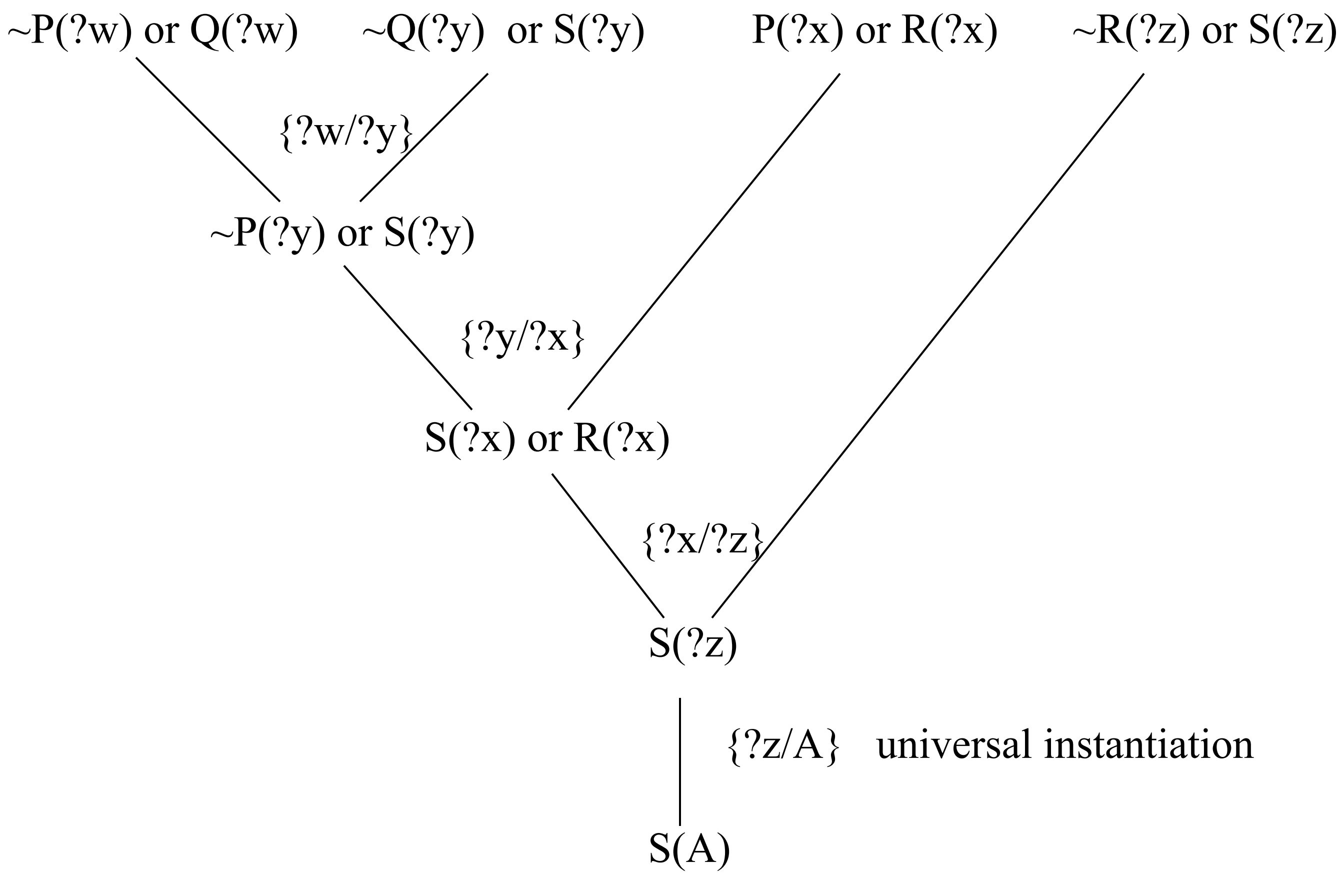


Note that this is not a proof generated by a set-of-support strategy.

Prove S(A) from axioms along top line using resolution (without refutation)

~P(?w) or Q(?w)   ~Q(?y) or S(?y)   P(?x) or R(?x)   ~R(?z) or S(?z)

Prove S(A) from axioms along top line using resolution



Douglas H. Fisher

Show the result of resolving **P(y, x, B, y) or Q(x, y, x, A)** with **~P(z, z, w, A)**

Unifying P(y, x, B, y) with P(z, z, w, A) results in {y/z, x/z, w/B, z/A} or some alternate, equivalent substitution list

z z

y/z P(~~y~~, x, B, ~~y~~) with P(z, z, w, A) **Q(x, ~~y~~, x, A)**

z z z

x/z P(z, ~~x~~, B, z) with P(z, z, w, A) **Q(~~x~~, z, ~~x~~, A)**

B

w/B P(z, z, B, z) with P(z, z, ~~w~~, A) **Q(z, z, z, A)**

A A A A A A A A

z/A P(~~z~~, ~~z~~, B, ~~z~~) with P(~~z~~, ~~z~~, B, A) **Q(~~z~~, ~~z~~, ~~z~~, A)**

Final answer: Q(A,A,A,A)

Given the axioms:

A,B are constants

?x's, ?y's, and ?z's are (universally-quantified) variables

ontable(A)

~clear(A)

~above(?x1, ?y1) or below(?y1,? x1)

~below(?x2, ?y2) or above(?y2, ?x2)

~on(?x3, ?y3) or above(?x3, ?y3)

~on(?x4, ?y4) or ~above(?y4, ?z4) or above(?x4, ?z4)

clear(?x5) or on(B, ?x5)

~ontable(?x6) or ~holding(?x6)

~ontable(?x7) or ~on(?x7, ?y7)

Prove **below(A, B)** with a *refutation resolution* proof using a *set-of-support* strategy. Show the proof tree and substitution lists.

If axioms like those shown represent conditions in a warehouse where humans and robots collaboratively engage in loading and movement tasks involving large, heavy objects, sketch an example where an initially inconsistent knowledge base could lead to dangerous interactions for humans and/or robots.

First Order Planning

Representation of operators: A PREcondition list and an EFFects list

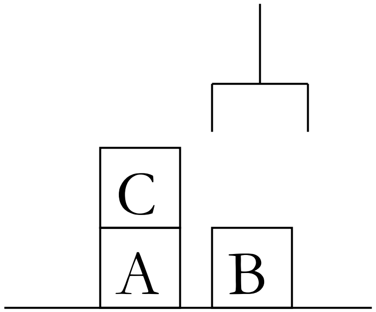
pickup(?x): PRE: ONTABLE(?x), CLEAR(?x), HANDEMPTY
EFF: ~ONTABLE(?x), ~CLEAR(?x), ~HANDEMPTY,
HOLDING(?x)

putdown(?x): PRE: HOLDING(?x)
EFF: ~HOLDING(?x),
ONTABLE(?x), CLEAR(?x), HANDEMPTY

stack(?x, ?y): PRE: HOLDING(?x), CLEAR(?y)
EFF: ~HOLDING(?x), ~CLEAR(?y),
HANDEMPTY, ON(?x, ?y), CLEAR(?x)

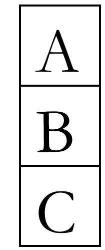
unstack(?x, ?y): PRE: HANDEMPTY, CLEAR(?x), ON(?x,?y)
EFF: ~HANDEMPTY, ~CLEAR(?x), ~ON(?x,?y),
HOLDING(?x), CLEAR(?y)

Tweak : the following slides represent *one* path in a search for a plan



ON(C,A)
ONTAB(A)
ONTAB(B)
CLEAR(B)
CLEAR(C)
HANDEMPTY

Initial State



ON(A,B)
ON(B,C)

Goal Spec

Douglas H. Fisher

Tweak

STACK(A,B)

CLEAR(B) | HANDEMPTY
HOLDING(A) | ON(A,B)
~CLEAR(B)
~HOLDING(A)
CLEAR(A)

?x/A,?y/B

ON(C,A)
ONTAB(A)
ONTAB(B)
CLEAR(B)
CLEAR(C)
HANDEMPTY

Causal link (protected)
No operator can be placed between STACK(A,B) and goal that undoes (clobbers) the protected condition, ON(A,B).

ON(A,B)
ON(B,C)

Step addition

Tweak

STACK(A,B)

CLEAR(B)	HANDEMPTY
HOLDING(A)	ON(A,B)
	~CLEAR(B)
	~HOLDING(A)
	CLEAR(A)

ON(C,A)
ONTAB(A)
ONTAB(B)
CLEAR(B)
CLEAR(C)
HANDEMPTY

Causal links (protected)

STACK(B,C)

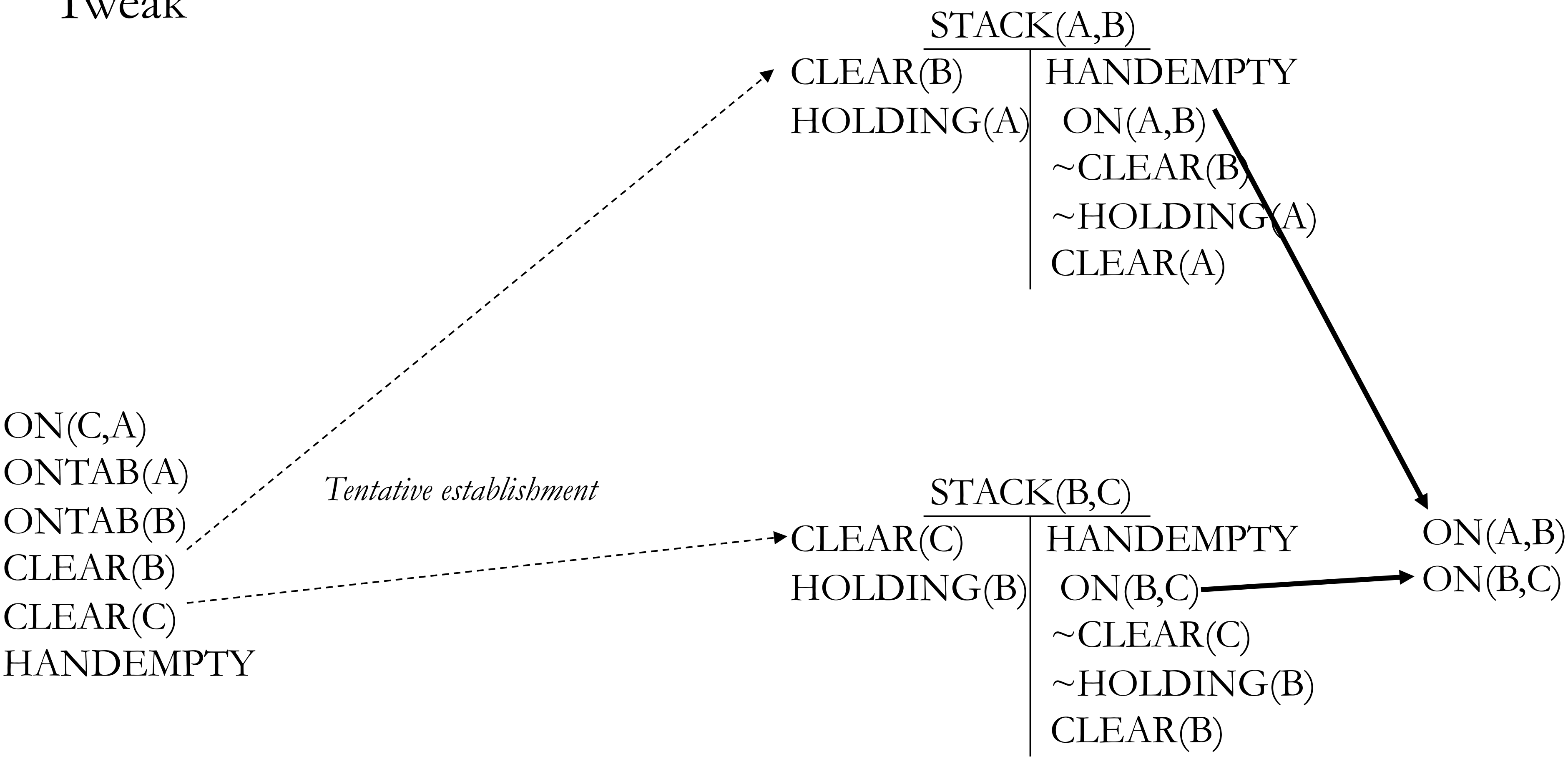
CLEAR(C)	HANDEMPTY
HOLDING(B)	ON(B,C)
	~CLEAR(C)
	~HOLDING(B)
	CLEAR(B)

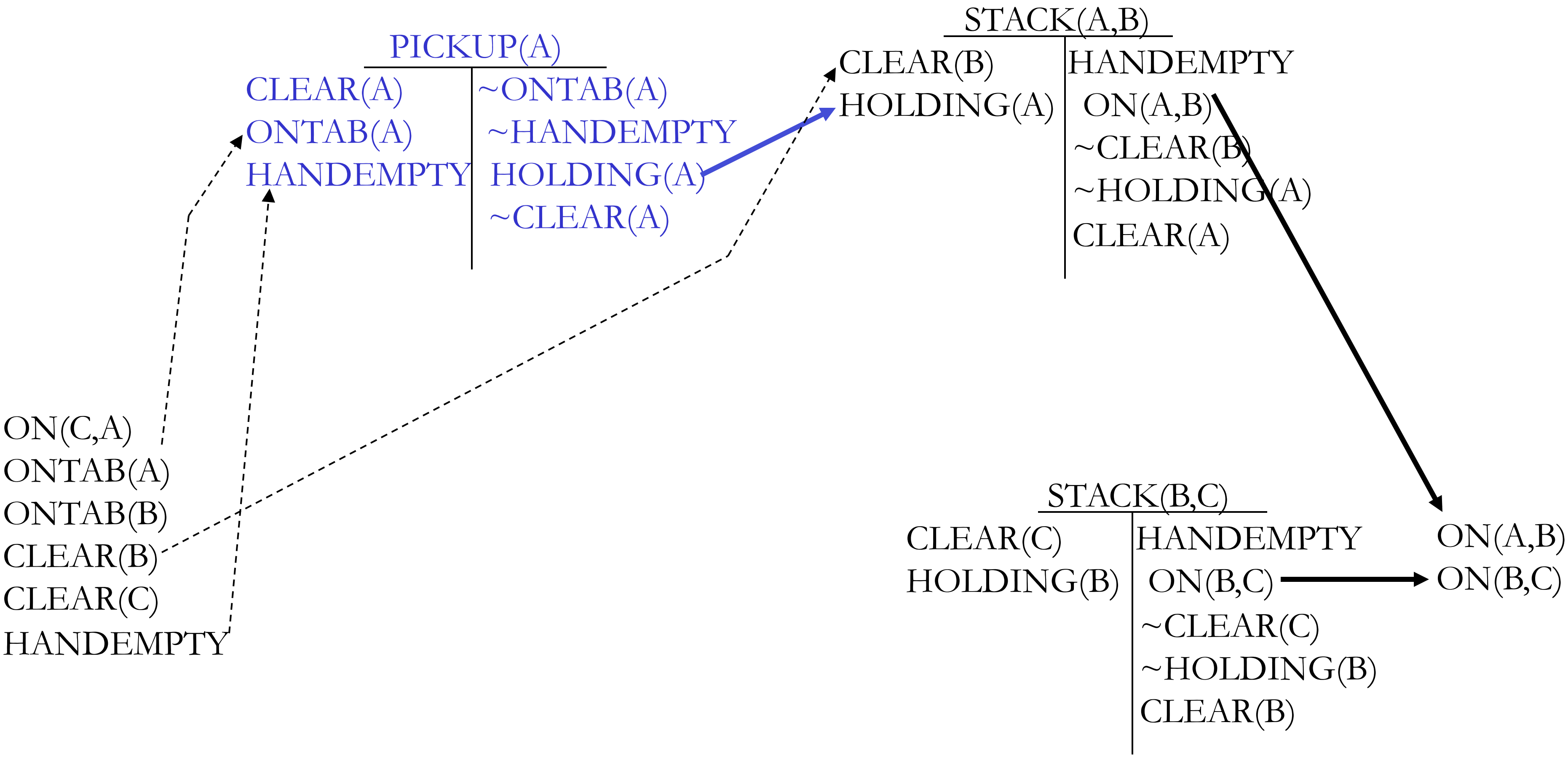
?x/B,?y/C

ON(A,B)
ON(B,C)

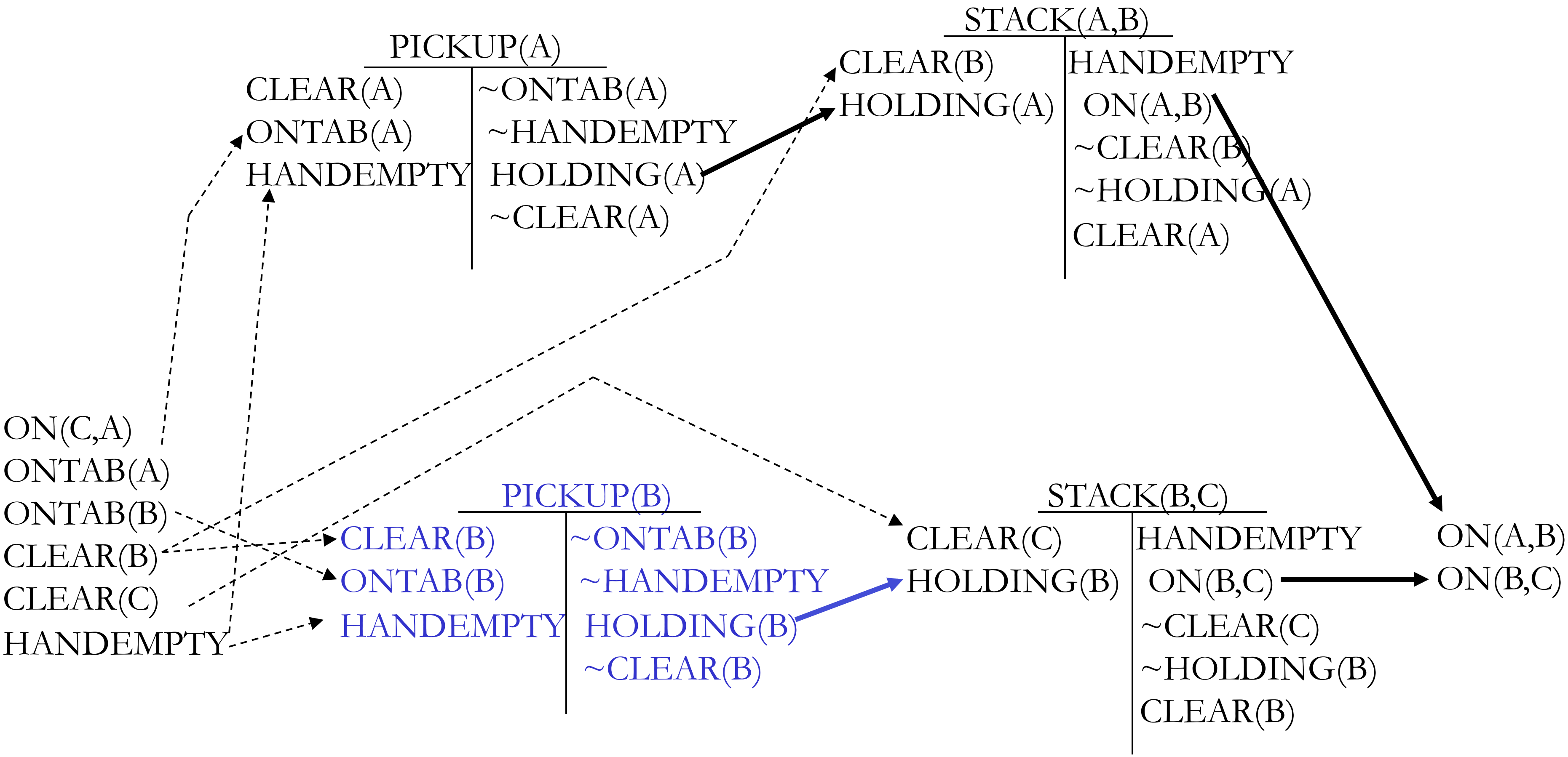
Step addition

Tweak

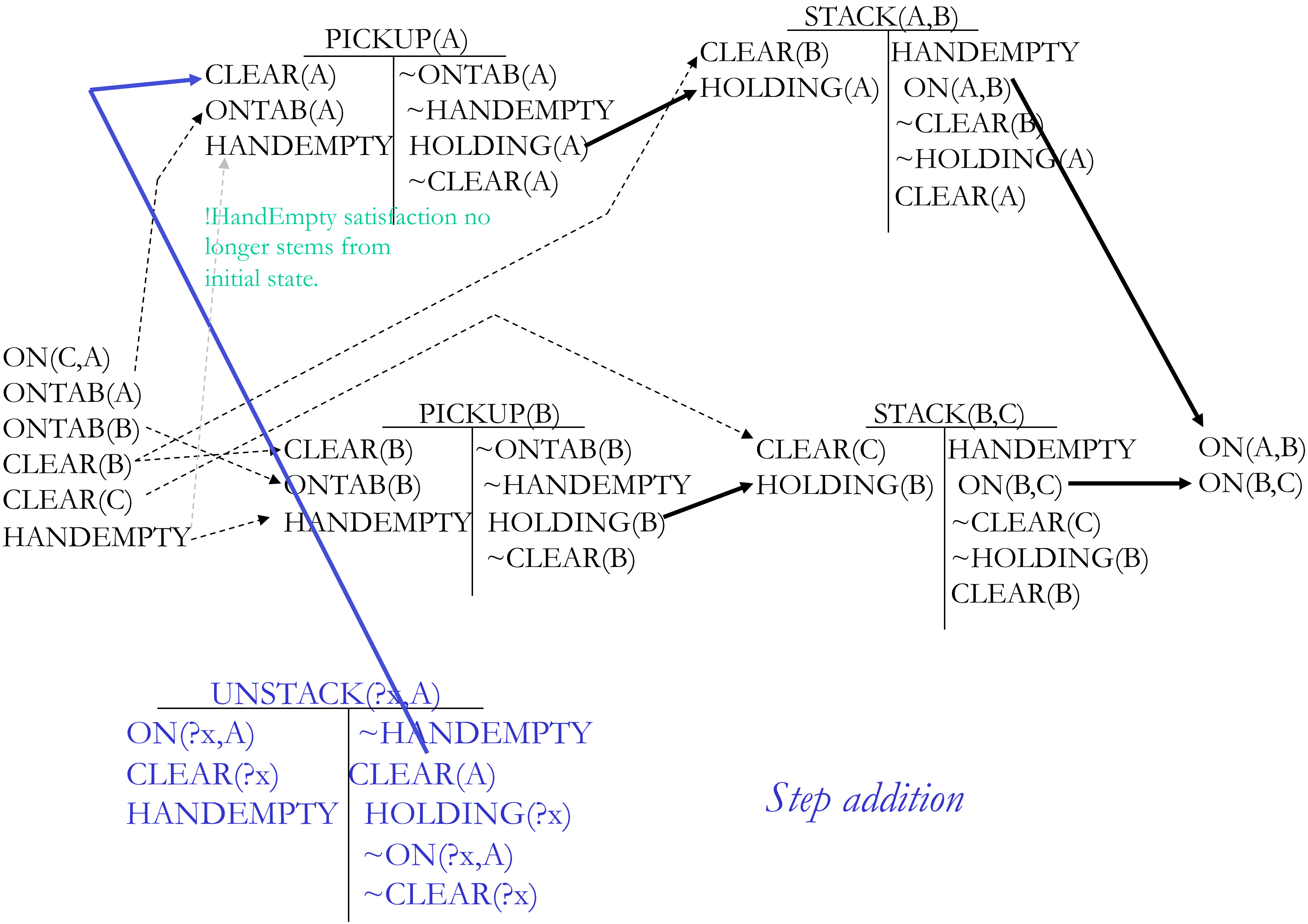

ON(C,A)
ONTAB(A)
ONTAB(B)
CLEAR(B)
CLEAR(C)
HANDEMPTY
Tentative establishment
STACK(A,B)
CLEAR(B)
HOLDING(A)
HANDEMPTY
ON(A,B)
~CLEAR(B)
~HOLDING(A)
CLEAR(A)
STACK(B,C)
CLEAR(C)
HOLDING(B)
HANDEMPTY
ON(B,C)
~CLEAR(C)
~HOLDING(B)
CLEAR(B)
ON(A,B)
ON(B,C)

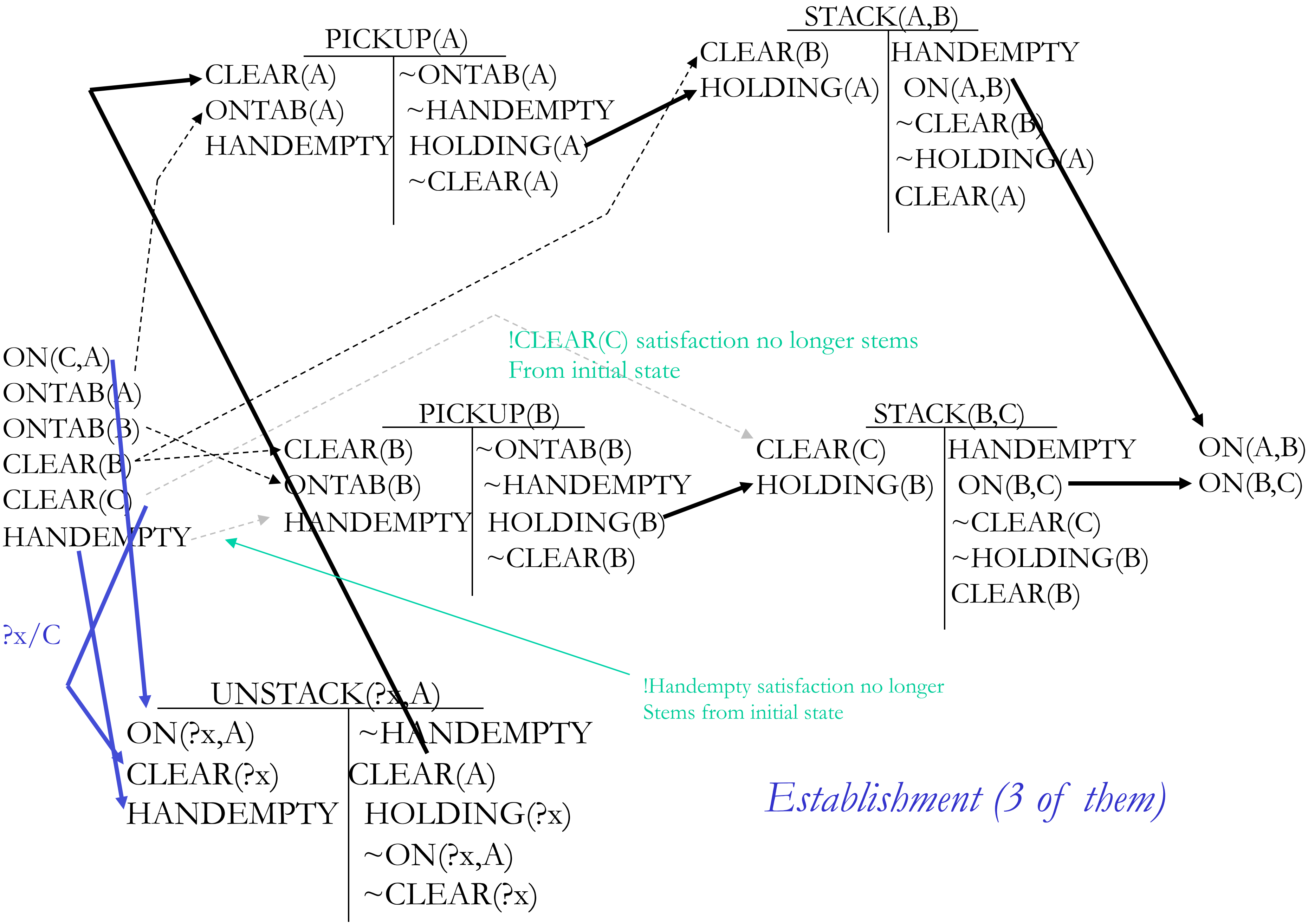


Step addition and tentative establishment

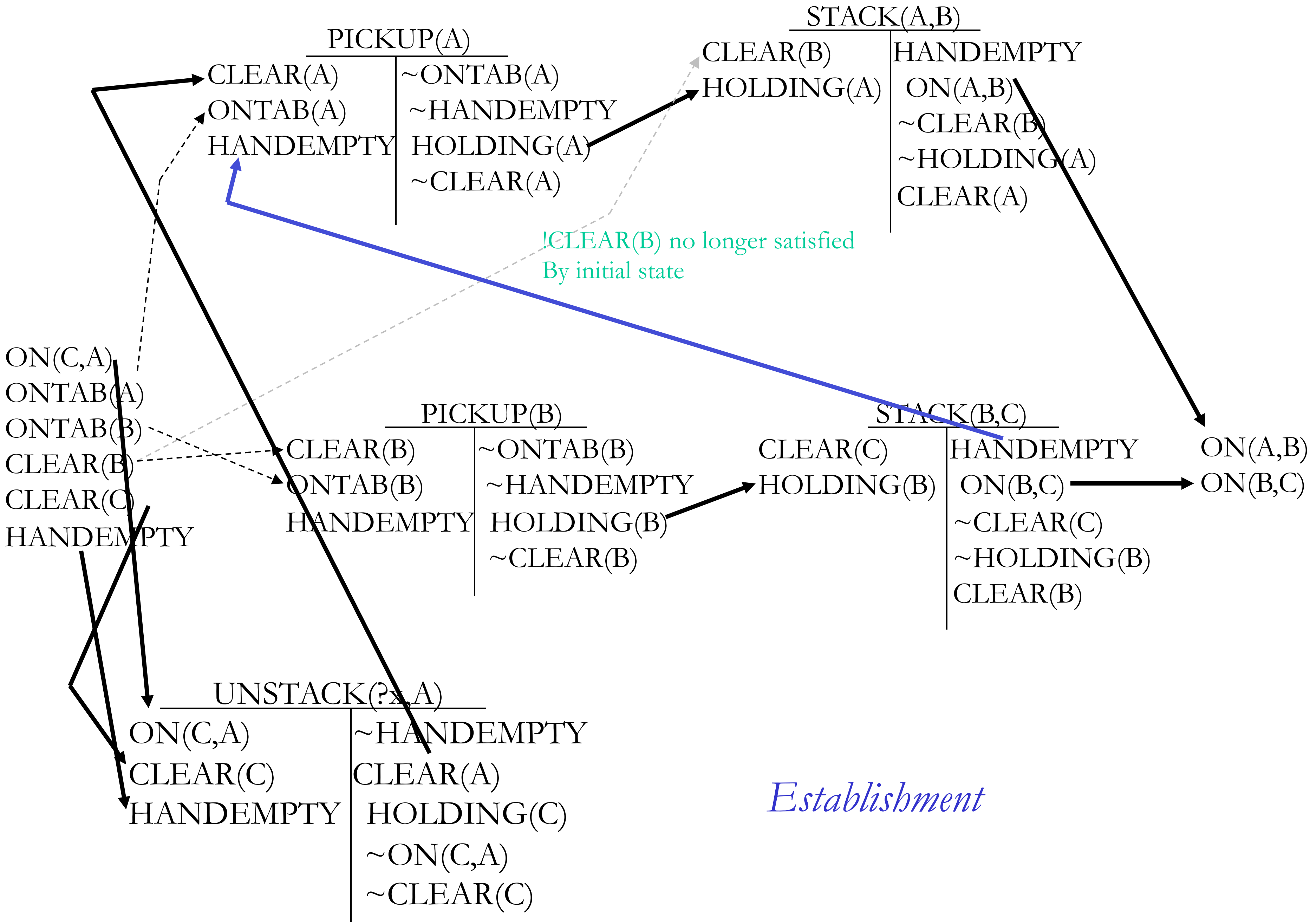


Step addition and tentative establishment

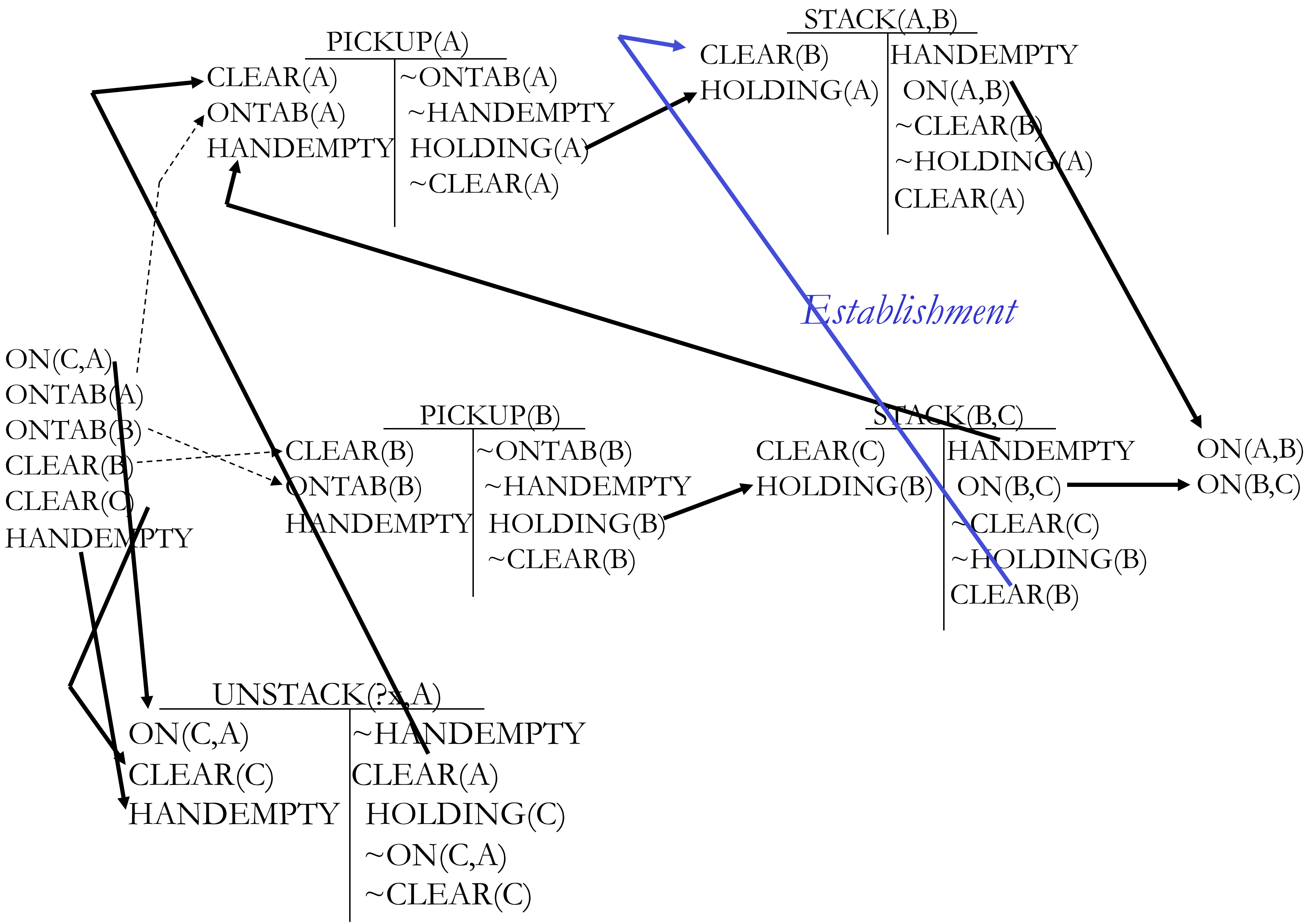

PICKUP(A)
CLEAR(A)
ONTAB(A)
HANDEMPTY
~ONTAB(A)
~HANDEMPTY
HOLDING(A)
~CLEAR(A)
STACK(A,B)
CLEAR(B)
HOLDING(A)
HANDEMPTY
ON(A,B)
~CLEAR(B)
~HOLDING(A)
CLEAR(A)
!HandEmpty satisfaction no longer stems from initial state.
ON(C,A)
ONTAB(A)
ONTAB(B)
CLEAR(B)
CLEAR(C)
HANDEMPTY
PICKUP(B)
CLEAR(B)
ONTAB(B)
HANDEMPTY
~ONTAB(B)
~HANDEMPTY
HOLDING(B)
~CLEAR(B)
STACK(B,C)
CLEAR(C)
HOLDING(B)
HANDEMPTY
ON(B,C)
~CLEAR(C)
~HOLDING(B)
CLEAR(B)
ON(A,B)
ON(B,C)
UNSTACK(?x,A)
ON(?x,A)
CLEAR(?x)
HANDEMPTY
~HANDEMPTY
CLEAR(A)
HOLDING(?x)
~ON(?x,A)
~CLEAR(?x)
Step addition



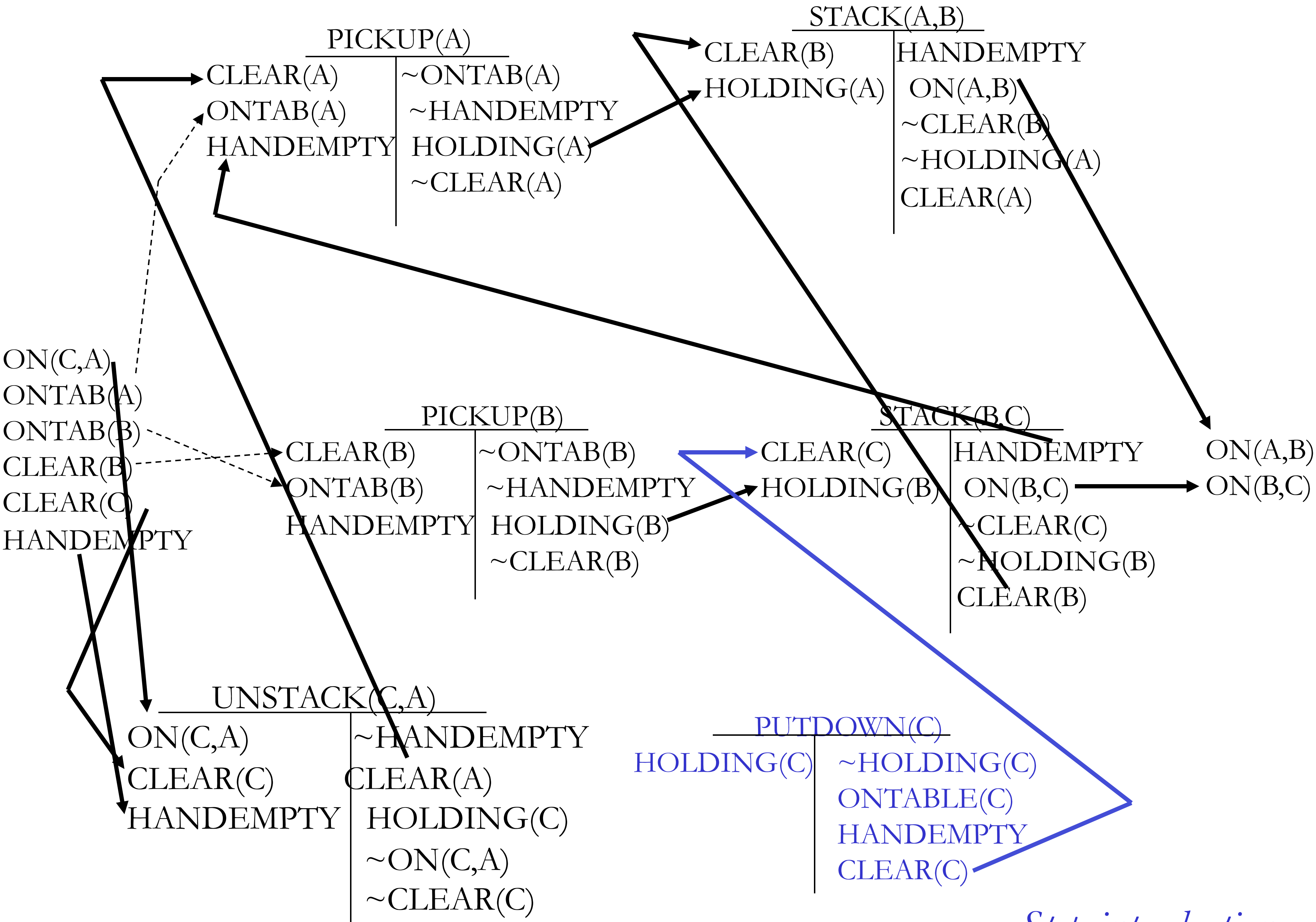
Establishment (3 of them)



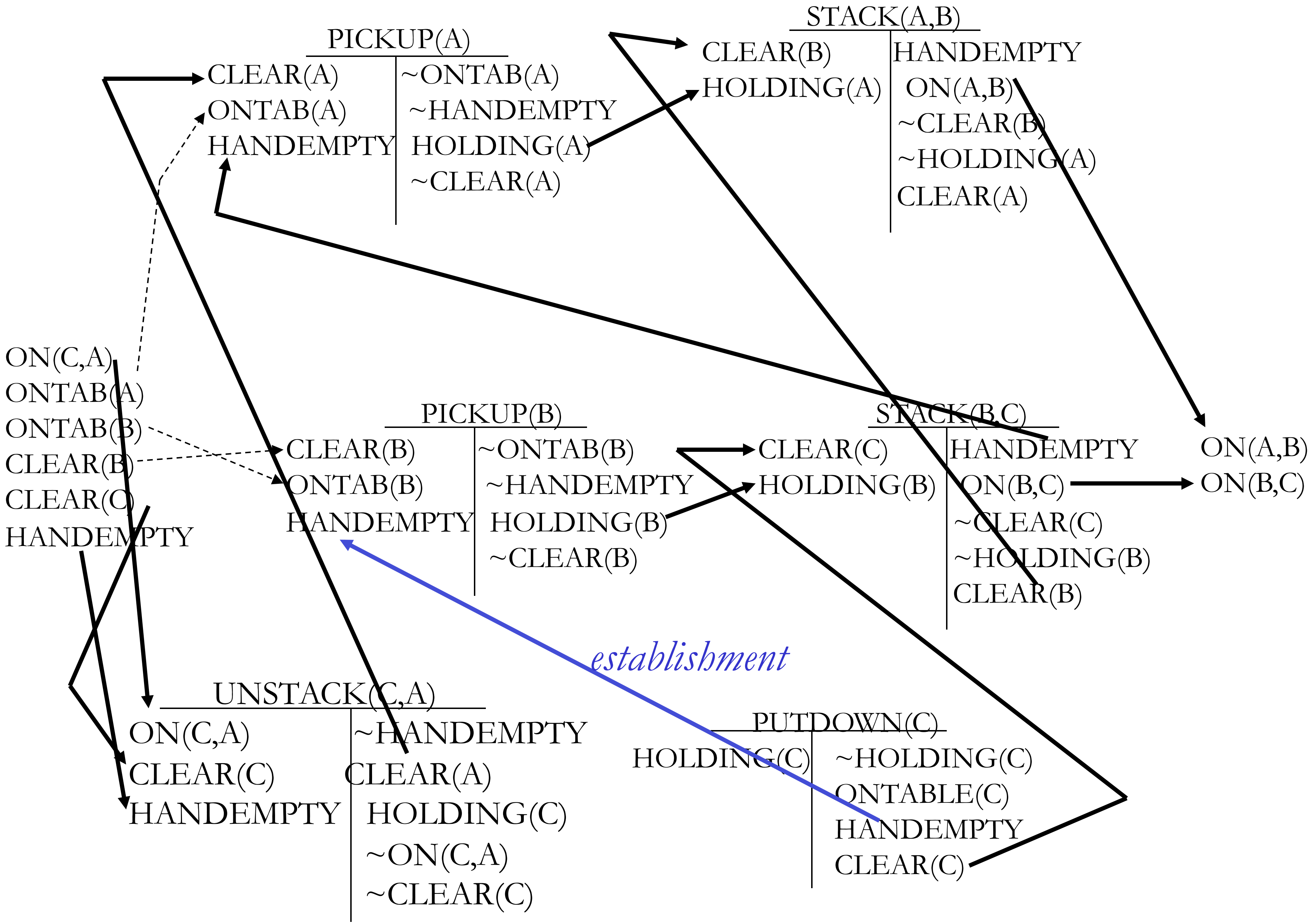
Establishment



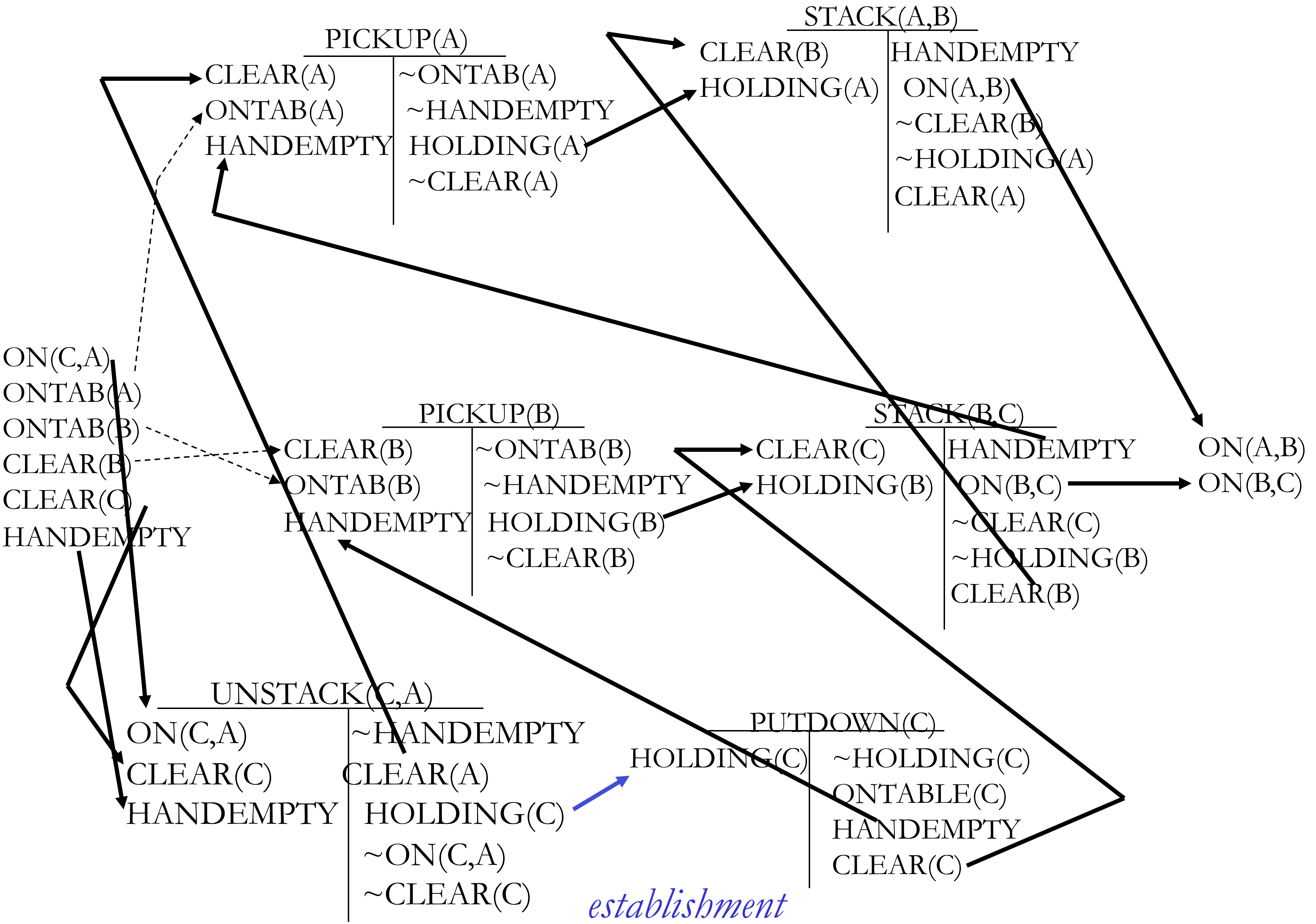
PICKUP(A)
CLEAR(A)
ONTAB(A)
HANDEMPTY
~ONTAB(A)
~HANDEMPTY
HOLDING(A)
~CLEAR(A)
STACK(A,B)
CLEAR(B)
HOLDING(A)
HANDEMPTY
ON(A,B)
~CLEAR(B)
~HOLDING(A)
CLEAR(A)
Establishment
ON(C,A)
ONTAB(A)
ONTAB(B)
CLEAR(B)
CLEAR(C)
HANDEMPTY
PICKUP(B)
CLEAR(B)
ONTAB(B)
HANDEMPTY
~ONTAB(B)
~HANDEMPTY
HOLDING(B)
~CLEAR(B)
STACK(B,C)
CLEAR(C)
HOLDING(B)
HANDEMPTY
ON(B,C)
~CLEAR(C)
~HOLDING(B)
CLEAR(B)
ON(A,B)
ON(B,C)
UNSTACK(?x,A)
ON(C,A)
CLEAR(C)
HANDEMPTY
~HANDEMPTY
CLEAR(A)
HOLDING(C)
~ON(C,A)
~CLEAR(C)



Step introduction

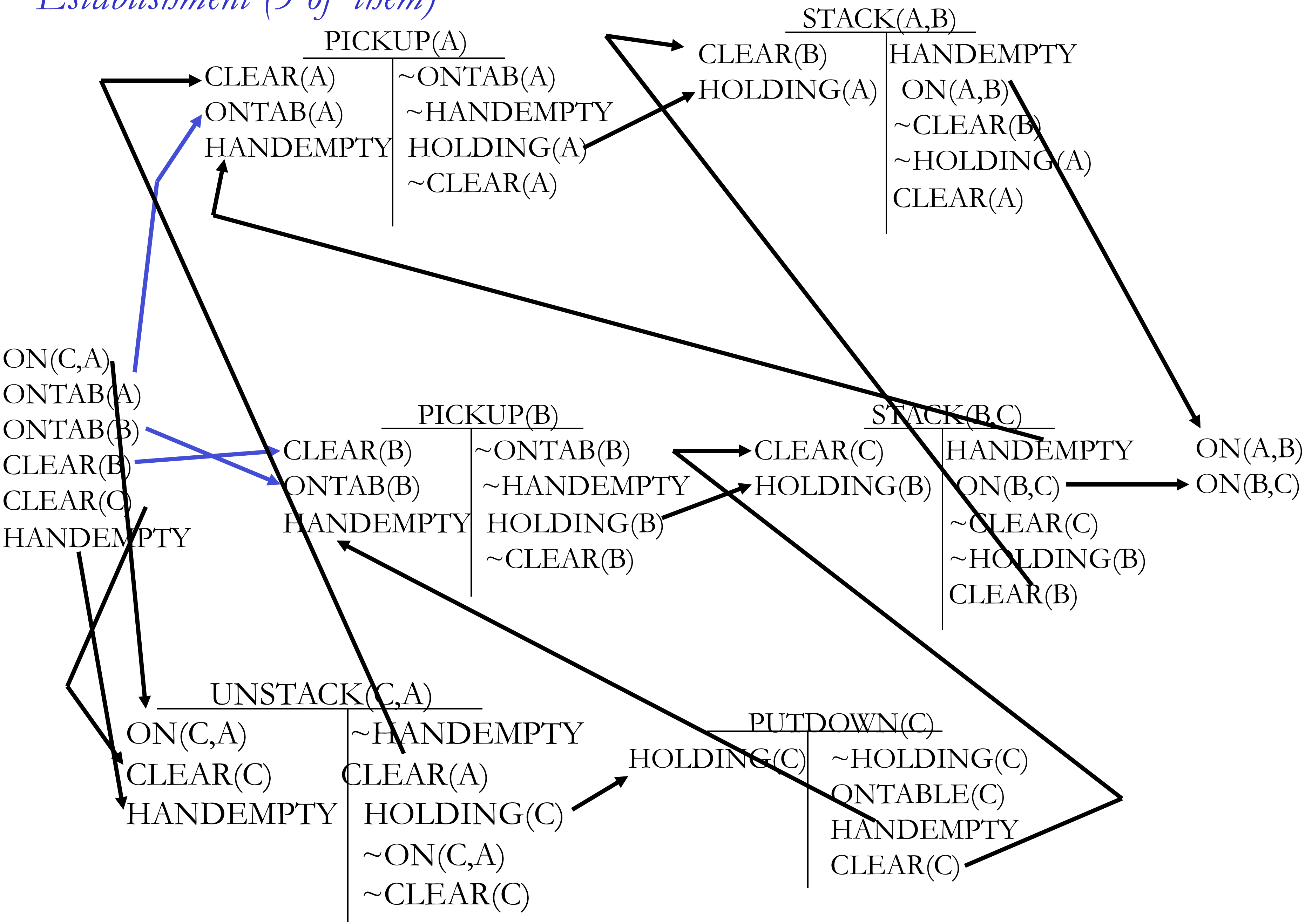


Douglas H. Fisher



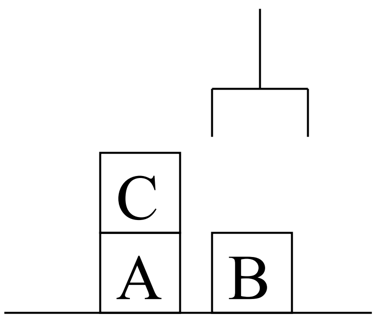
Douglas H. Fisher

Establishment (3 of them)

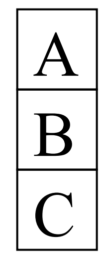


UNSTACK(C,A) → PUTDOWN(C) → PICKUP(B) → STACK(B,C) → PICKUP(A) →STACK(A,B)

ON(C,A)
ONTAB(A)
ONTAB(B)
CLEAR(B)
CLEAR(C)
HANDEMPTY

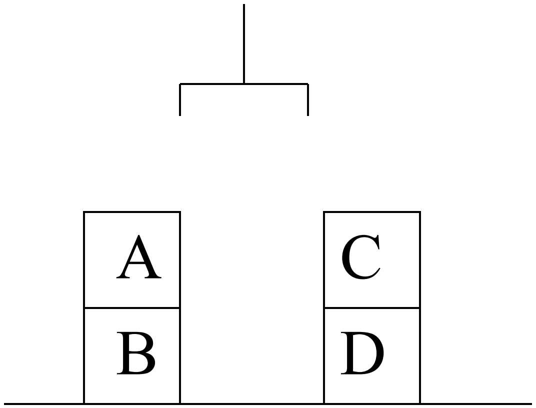


ON(A,B)
ON(B,C)



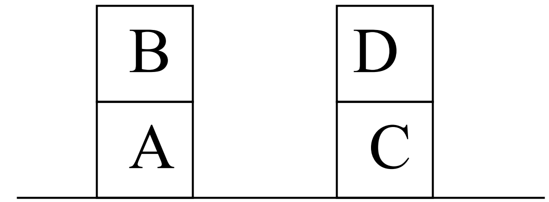
Step introduction

Douglas H. Fisher



Initial State

ON(A,B)
ONTAB(B)
CLEAR(A)
ON(C,D)
ONTAB(D)
CLEAR(C)
HANDEMPTY



Goal spec

ON(B,A)
CLEAR(B)
ON(D,C)
CLEAR(D)

Step addition

Stack(B,A)

Preconditions	Effects
HOLDING(B)	~HOLDING(B)
CLEAR(A)	~CLEAR(A)
	HANDEMPTY
	ON(B,A)
	CLEAR(B)

?x/B, ?y/A

ON(A,B)
ONTAB(B)
CLEAR(A)
ON(C,D)
ONTAB(D)
CLEAR(C)
HANDEMPTY

ON(B,A)
CLEAR(B)
ON(D,C)
CLEAR(D)

establishment

Stack(B,A)

Preconditions	Effects
HOLDING(B)	~HOLDING(B)
CLEAR(A)	~CLEAR(A)
	HANDEMPTY
	ON(B,A)
	CLEAR(B)

ON(A,B)
ONTAB(B)
CLEAR(A)
ON(C,D)
ONTAB(D)
CLEAR(C)
HANDEMPTY

ON(B,A)
CLEAR(B)
ON(D,C)
CLEAR(D)

Step addition + establishment

Stack(B,A)

Preconditions	Effects
HOLDING(B) CLEAR(A)	~HOLDING(B) ~CLEAR(A) HANDEMPTY ON(B,A) CLEAR(B)

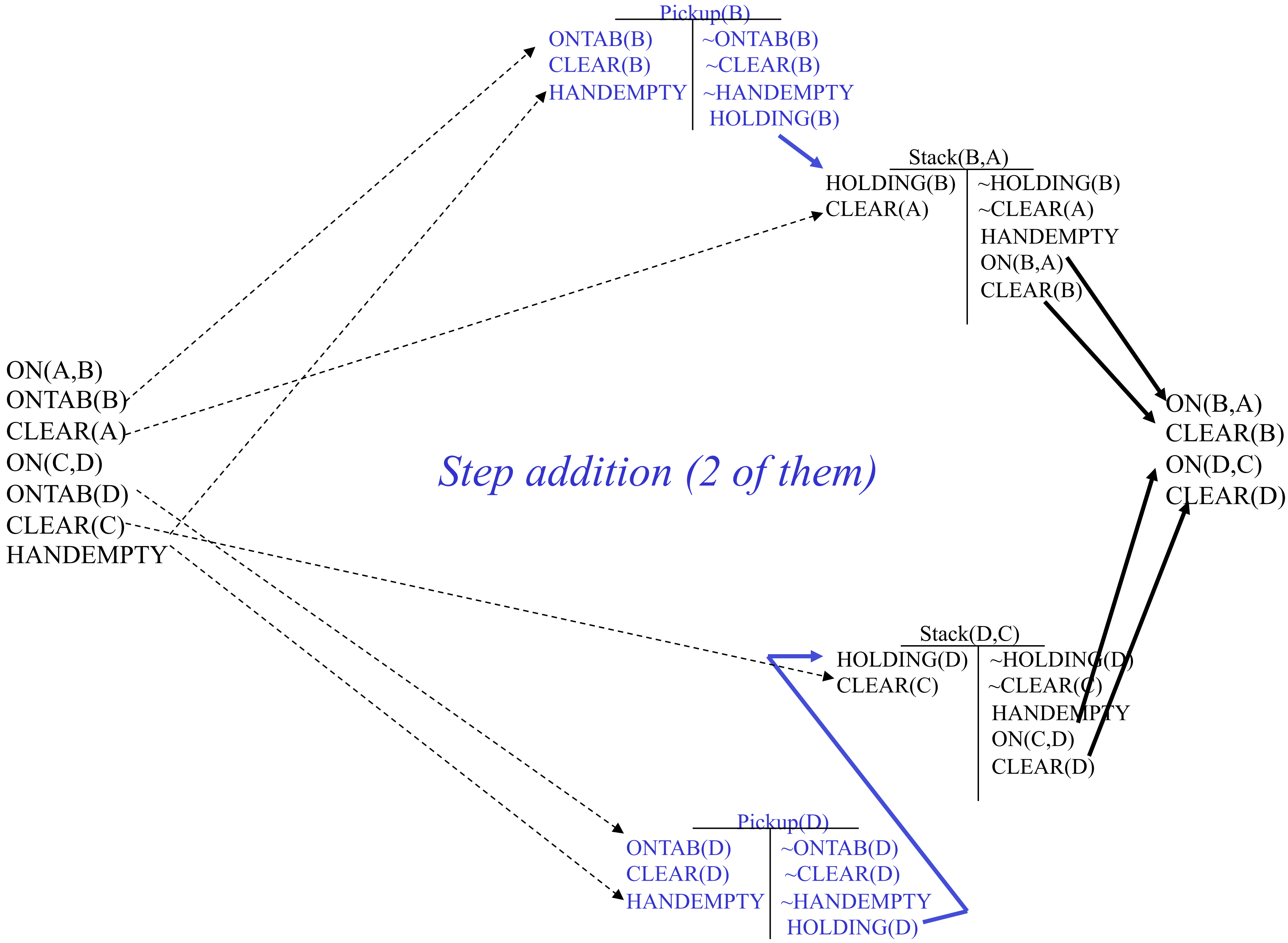
ON(A,B)
ONTAB(B)
CLEAR(A)
ON(C,D)
ONTAB(D)
CLEAR(C)
HANDEMPTY

ON(B,A)
CLEAR(B)
ON(D,C)
CLEAR(D)

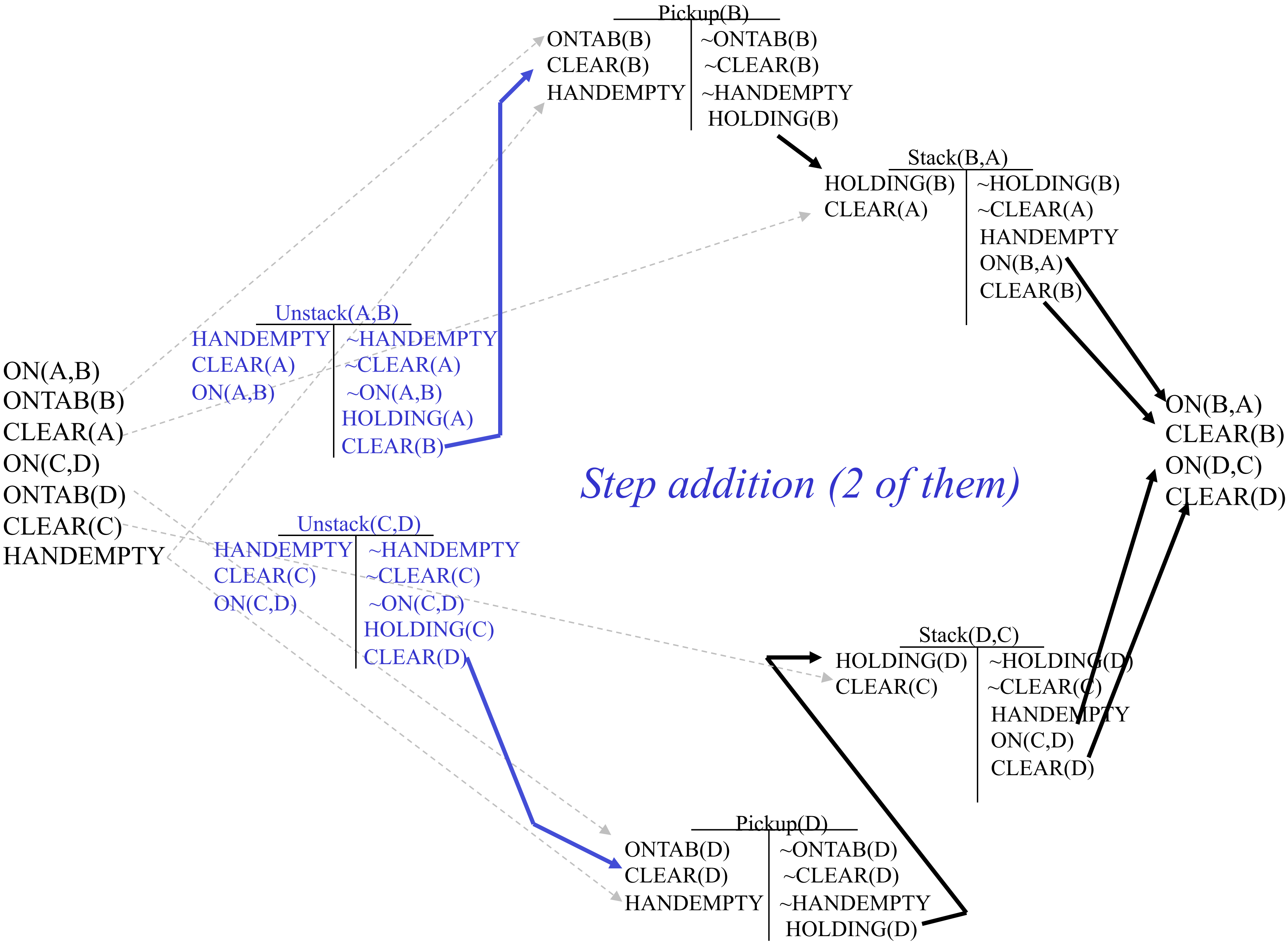
Stack(D,C)

Preconditions	Effects
HOLDING(D) CLEAR(C)	~HOLDING(D) ~CLEAR(C) HANDEMPTY ON(C,D) CLEAR(D)

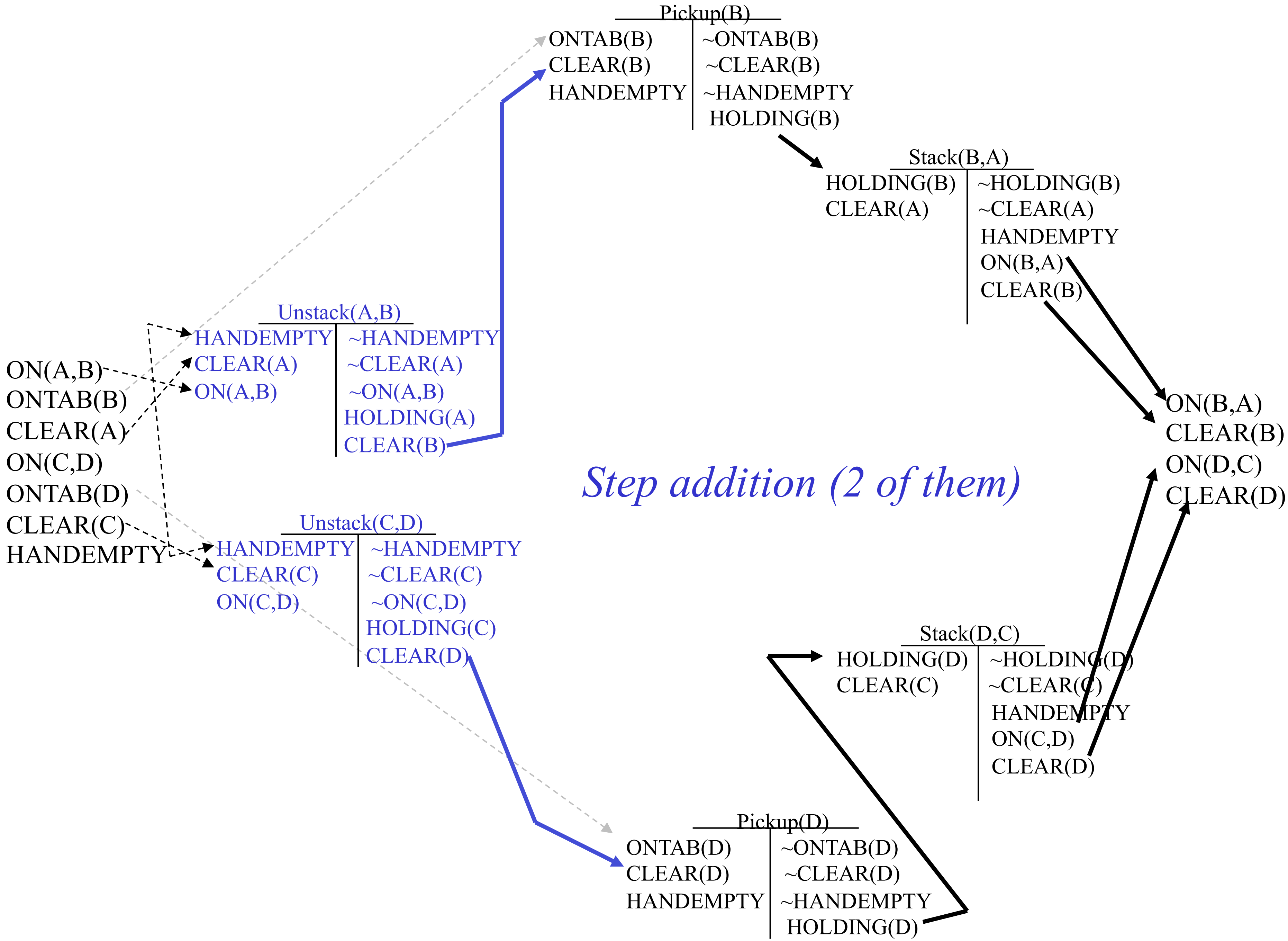
Step addition (2 of them)



Step addition (2 of them)

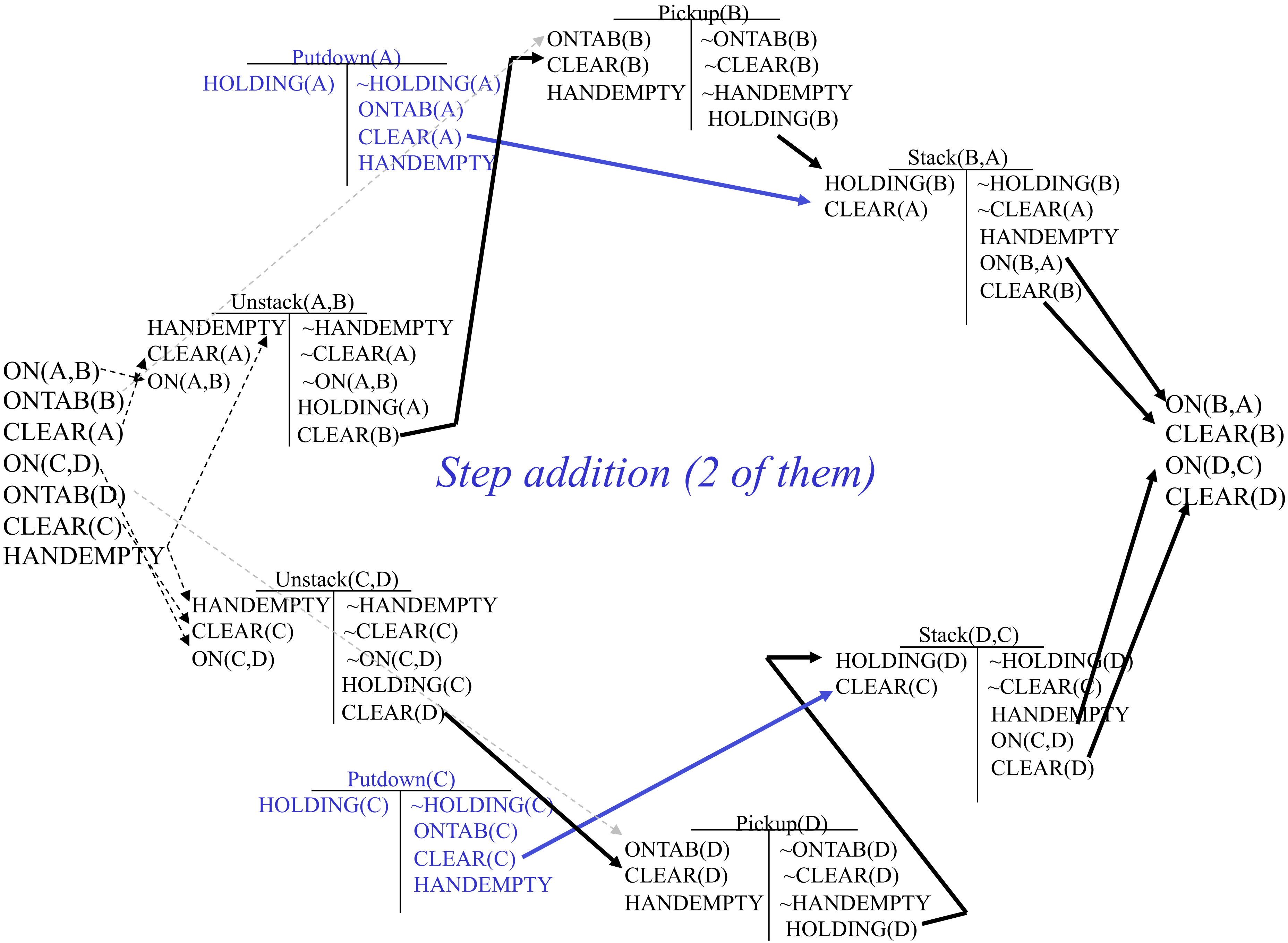


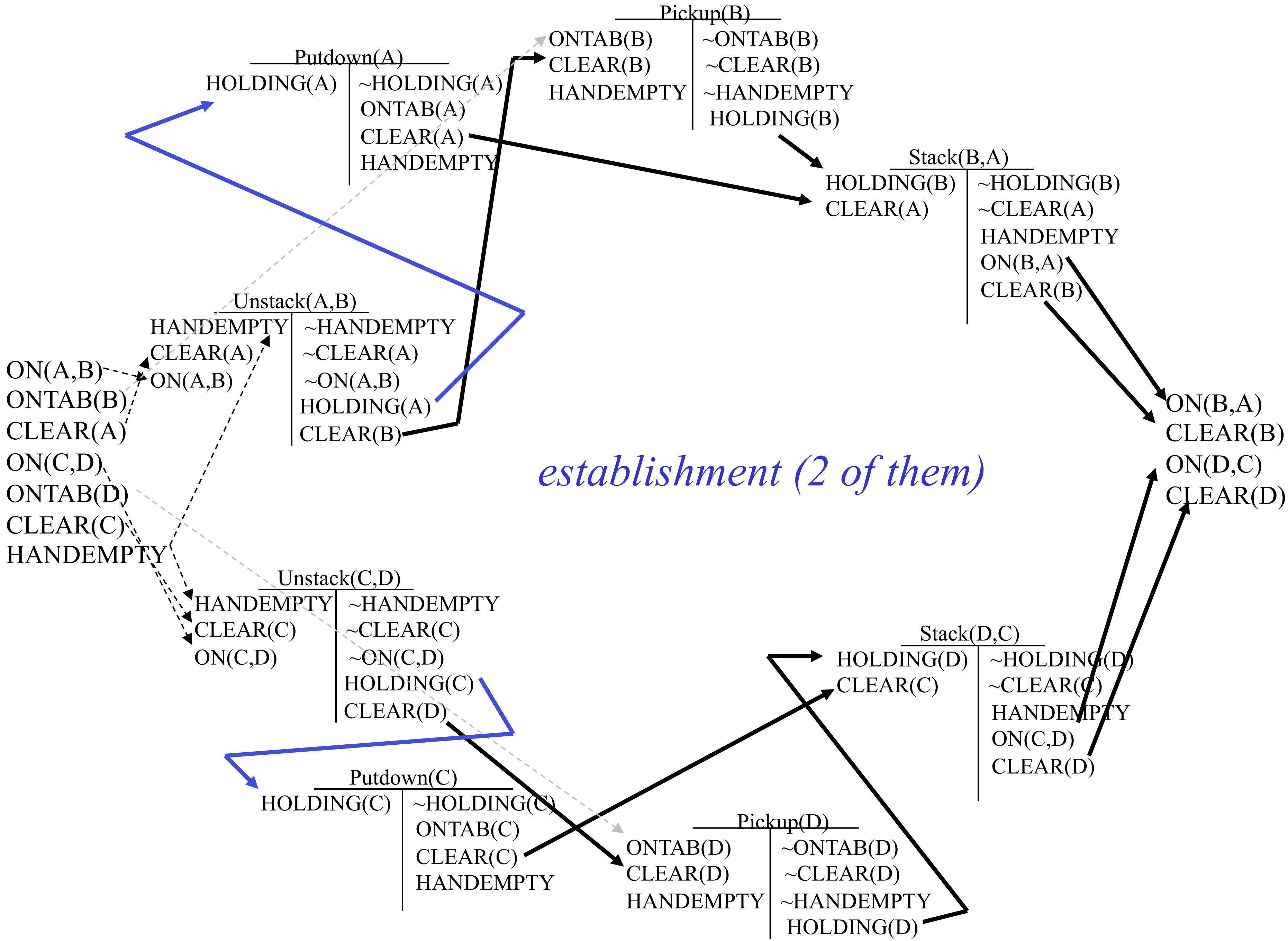
Douglas H. Fisher

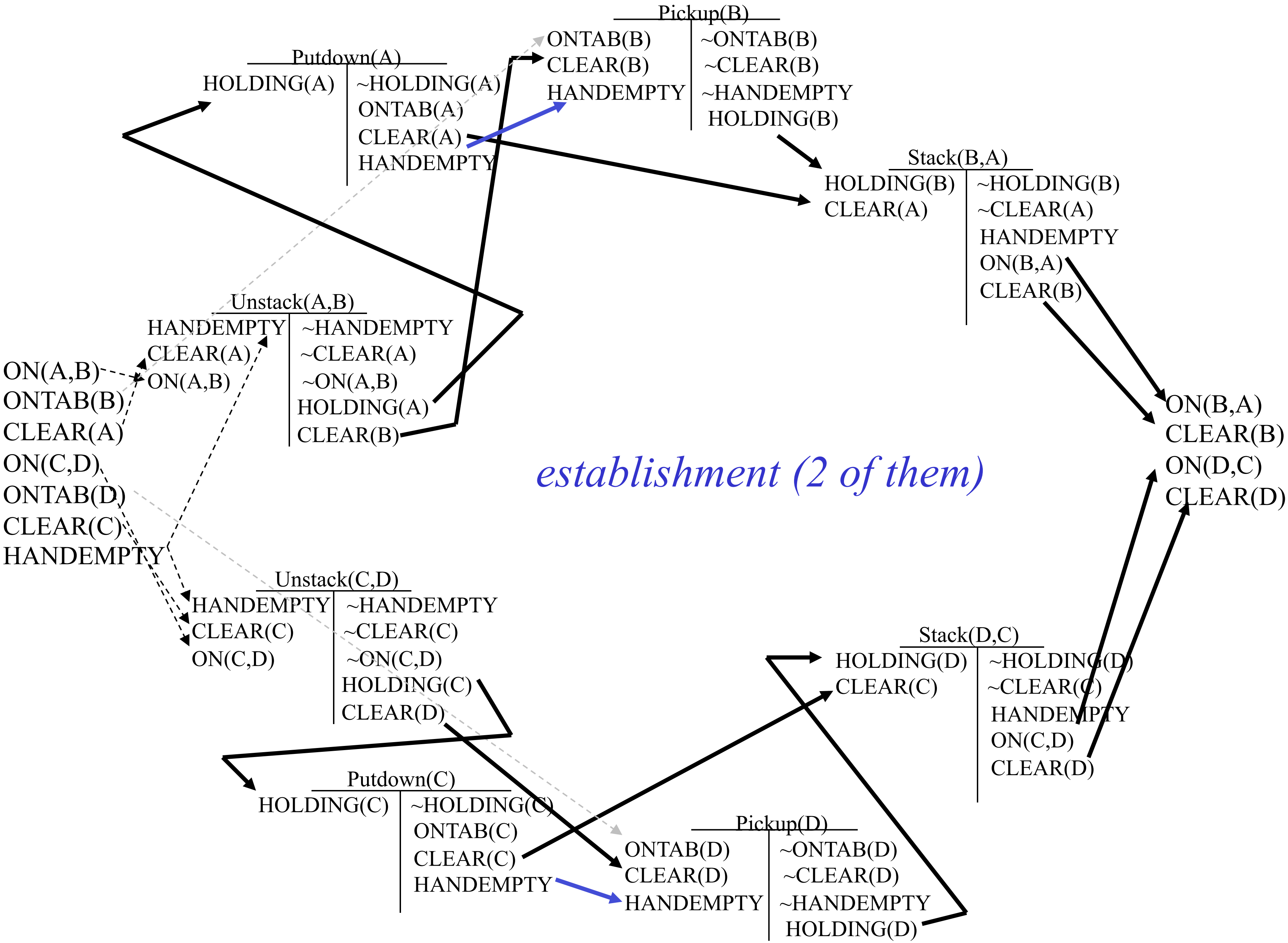


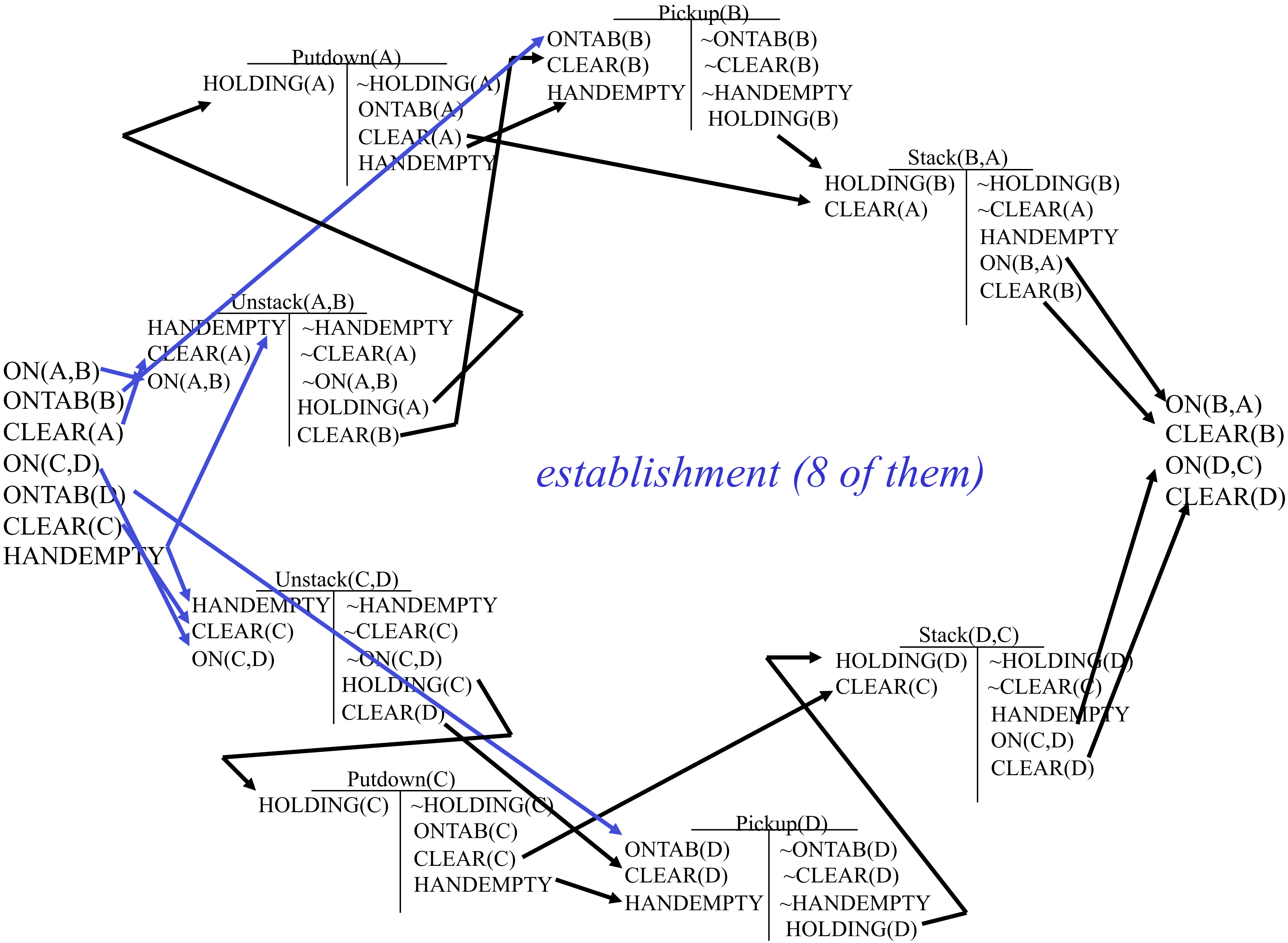
Step addition (2 of them)

Step addition (2 of them)




Pickup(B)
ONTAB(B)
CLEAR(B)
HANDEMPTY
~ONTAB(B)
~CLEAR(B)
~HANDEMPTY
HOLDING(B)
Putdown(A)
HOLDING(A)
~HOLDING(A)
ONTAB(A)
CLEAR(A)
HANDEMPTY
Stack(B,A)
HOLDING(B)
CLEAR(A)
~HOLDING(B)
~CLEAR(A)
HANDEMPTY
ON(B,A)
CLEAR(B)
Unstack(A,B)
HANDEMPTY
CLEAR(A)
ON(A,B)
~HANDEMPTY
~CLEAR(A)
~ON(A,B)
HOLDING(A)
CLEAR(B)
ON(A,B)
ONTAB(B)
CLEAR(A)
ON(C,D)
ONTAB(D)
CLEAR(C)
HANDEMPTY
ON(B,A)
CLEAR(B)
ON(D,C)
CLEAR(D)
establishment (2 of them)
Unstack(C,D)
HANDEMPTY
CLEAR(C)
ON(C,D)
~HANDEMPTY
~CLEAR(C)
~ON(C,D)
HOLDING(C)
CLEAR(D)
Stack(D,C)
HOLDING(D)
CLEAR(C)
~HOLDING(D)
~CLEAR(C)
HANDEMPTY
ON(C,D)
CLEAR(D)
Putdown(C)
HOLDING(C)
~HOLDING(C)
ONTAB(C)
CLEAR(C)
HANDEMPTY
Pickup(D)
ONTAB(D)
CLEAR(D)
HANDEMPTY
~ONTAB(D)
~CLEAR(D)
~HANDEMPTY
HOLDING(D)

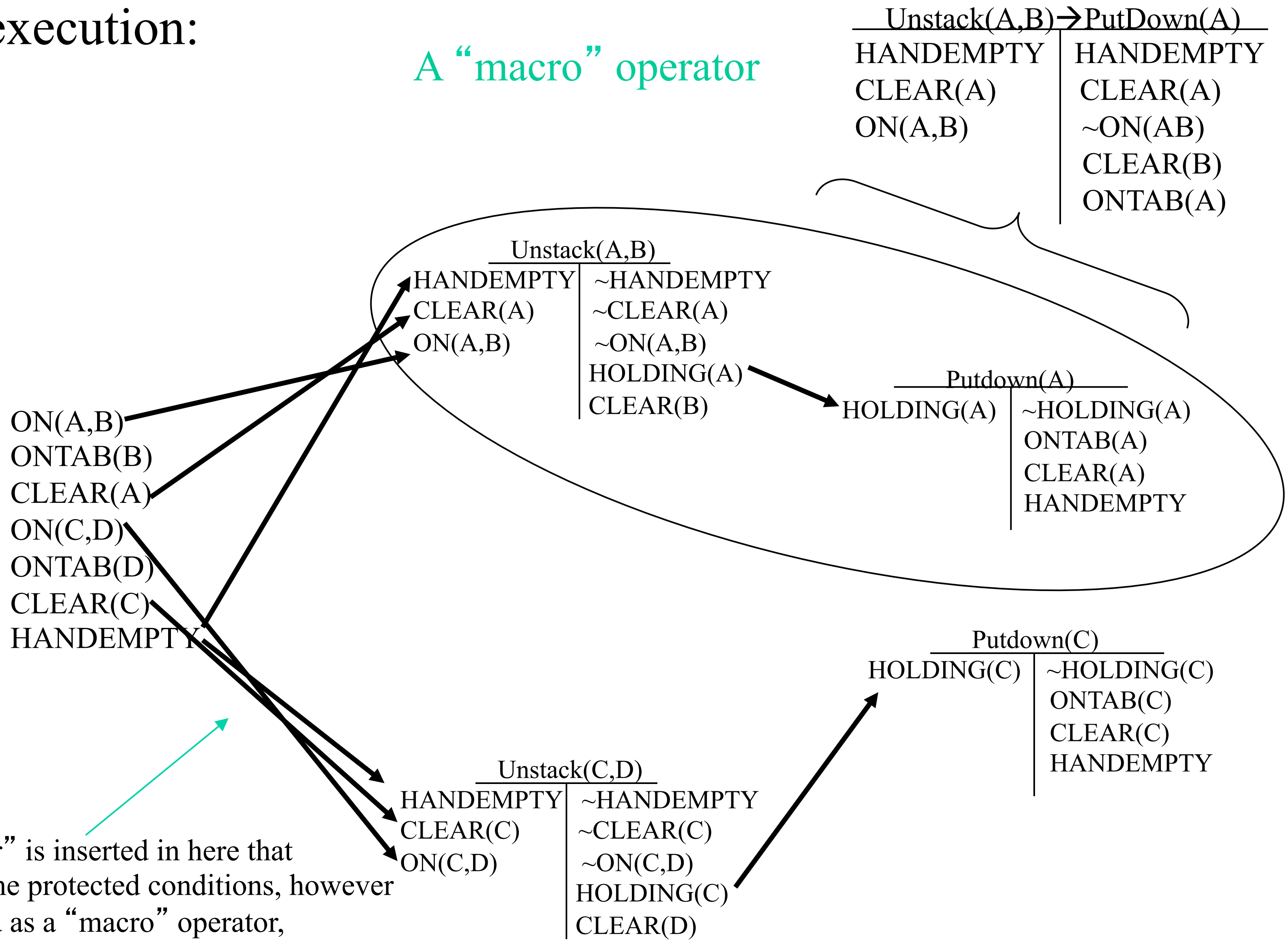

Pickup(B)
ONTAB(B)
CLEAR(B)
HANDEMPTY
~ONTAB(B)
~CLEAR(B)
~HANDEMPTY
HOLDING(B)
Putdown(A)
HOLDING(A)
~HOLDING(A)
ONTAB(A)
CLEAR(A)
HANDEMPTY
Stack(B,A)
HOLDING(B)
CLEAR(A)
~HOLDING(B)
~CLEAR(A)
HANDEMPTY
ON(B,A)
CLEAR(B)
Unstack(A,B)
HANDEMPTY
CLEAR(A)
ON(A,B)
~HANDEMPTY
~CLEAR(A)
~ON(A,B)
HOLDING(A)
CLEAR(B)
ON(A,B)
ONTAB(B)
CLEAR(A)
ON(C,D)
ONTAB(D)
CLEAR(C)
HANDEMPTY
ON(B,A)
CLEAR(B)
ON(D,C)
CLEAR(D)
establishment (2 of them)
Unstack(C,D)
HANDEMPTY
CLEAR(C)
ON(C,D)
~HANDEMPTY
~CLEAR(C)
~ON(C,D)
HOLDING(C)
CLEAR(D)
Stack(D,C)
HOLDING(D)
CLEAR(C)
~HOLDING(D)
~CLEAR(C)
HANDEMPTY
ON(C,D)
CLEAR(D)
Putdown(C)
HOLDING(C)
~HOLDING(C)
ONTAB(C)
CLEAR(C)
HANDEMPTY
Pickup(D)
ONTAB(D)
CLEAR(D)
HANDEMPTY
~ONTAB(D)
~CLEAR(D)
~HANDEMPTY
HOLDING(D)



Douglas H. Fisher

Plan execution:

A "macro" operator



No "operator" is inserted in here that will negate the protected conditions, however when viewed as a "macro" operator, Unstack(A,B)→PutDown(A) does not clobber any protected subgoals (or more exactly, it restores any subgoals (e.g., handempty) that are temporally "clobbered")

Douglas H. Fisher

```
Assign(?r1, ?v1, ?r2, ?v2)   /*  ?r1 = ?r2  */
   PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
   DEL: CONT(?r1, ?v1)
   ADD: CONT(?r1, ?v2)
```

CONT(P,a)
CONT(Q,b)
CONT(S,c)

CONT(Q,a)
CONT(P,b)

Initial State

P a

Q b

S c

Goal Spec

P b

Q a

Douglas H. Fisher

Assign(?r1, ?v1, ?r2, ?v2)   /* ?r1 = ?r2 */
PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
DEL: CONT(?r1, ?v1)
ADD: CONT(?r1, ?v2)

Assign(?r1, ?v1, ?r2, ?v2)

CONT(?r1, ?v1)	~CONT(?r1, ?v1)
CONT(?r2, ?v2)	CONT(?r1, ?v2)

?r1/Q, ?v2/a

CONT(P,a)
CONT(Q,b)
CONT(S,c)

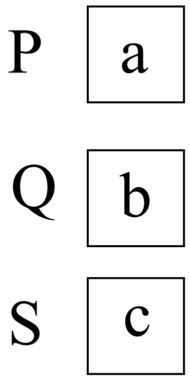
CONT(Q,a)
CONT(P,b)

Assign(?r1, ?v1, ?r2, ?v2)

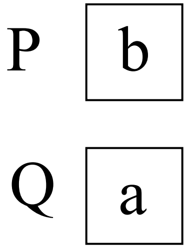
CONT(?r1, ?v1)	~CONT(?r1, ?v1)
CONT(?r2, ?v2)	CONT(?r1, ?v2)

?r1/P, ?v2/b

Initial State

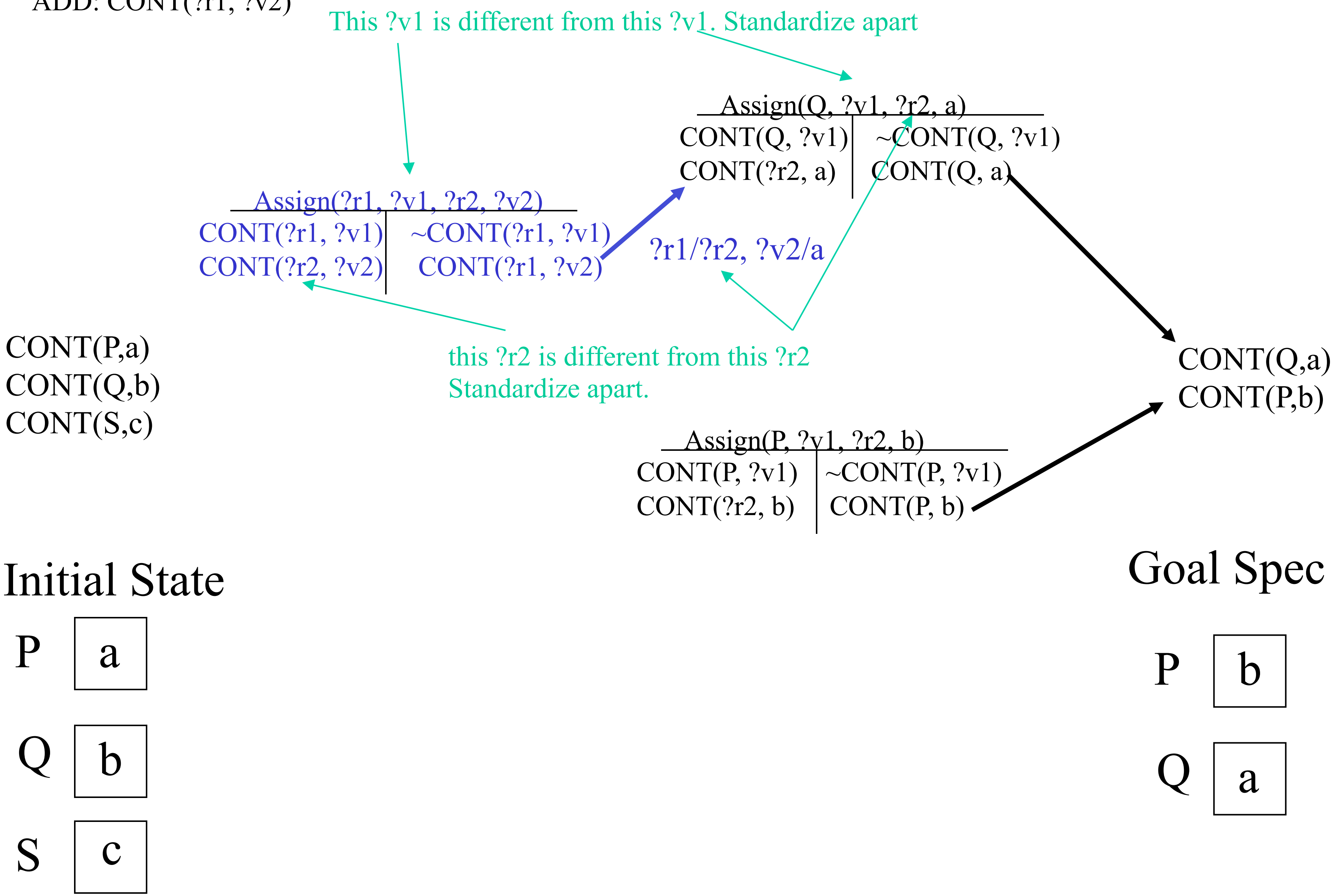


Goal Spec



Douglas H. Fisher

Assign(?r1, ?v1, ?r2, ?v2) /* ?r1 = ?r2 */
PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
DEL: CONT(?r1, ?v1)
ADD: CONT(?r1, ?v2)



Assign(?r1, ?v1, ?r2, ?v2)   /* ?r1 = ?r2 */
PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
DEL: CONT(?r1, ?v1)
ADD: CONT(?r1, ?v2)

Assign(Q, ?v1, ?r2, a)

PRE	EFFECTS
CONT(Q, ?v1)	~CONT(Q, ?v1)
CONT(?r2, a)	CONT(Q, a)

Assign(?r2, ?v11, ?r22, a)

PRE	EFFECTS
CONT(?r2, ?v11)	~CONT(?r2, ?v11)
CONT(?r22, a)	CONT(?r2, a)

CONT(P,a)
CONT(Q,b)
CONT(S,c)

CONT(Q,a)
CONT(P,b)

Assign(P, ?v1, ?r2, b)

PRE	EFFECTS
CONT(P, ?v1)	~CONT(P, ?v1)
CONT(?r2, b)	CONT(P, b)

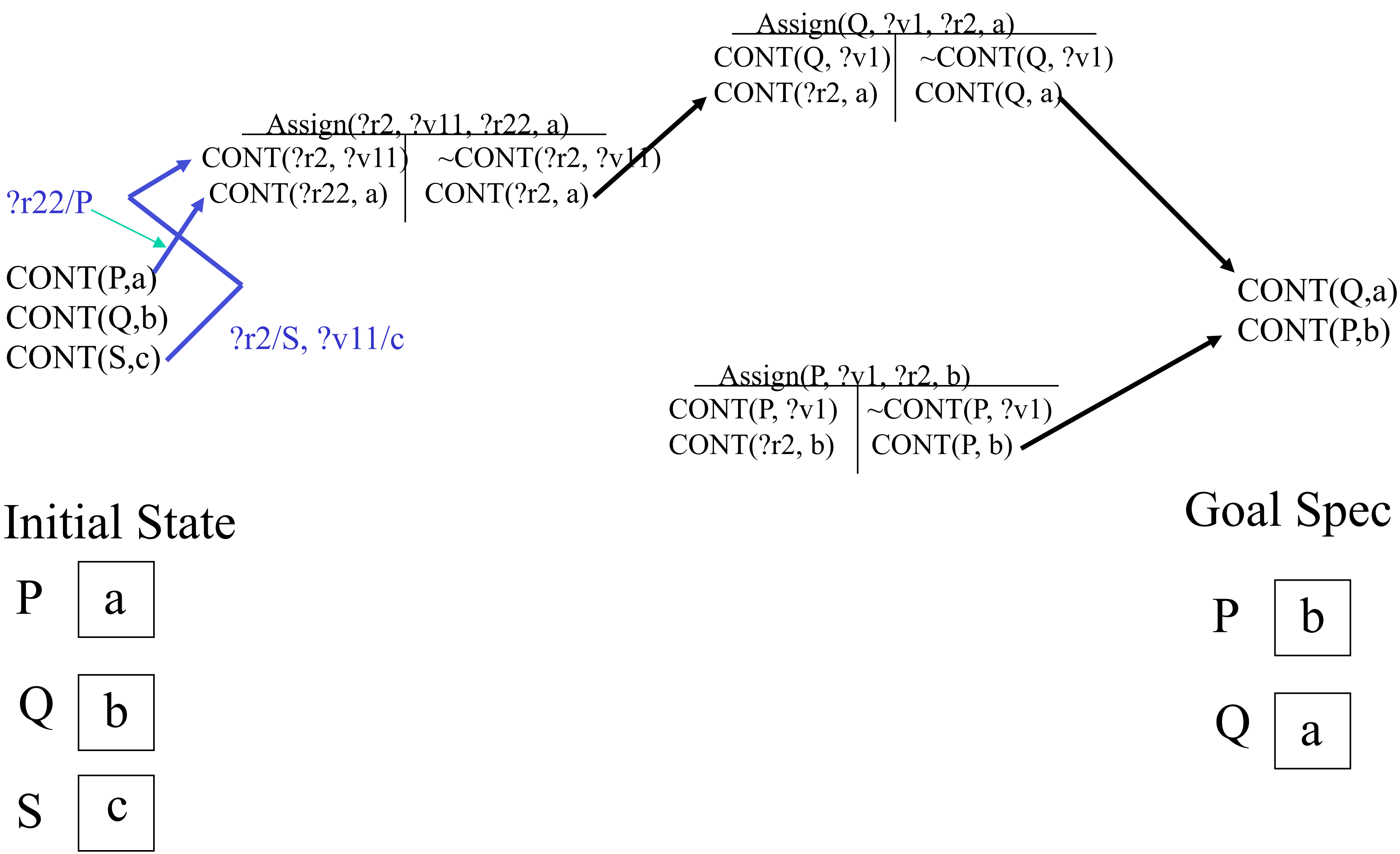
Initial State

P	a
Q	b
S	c

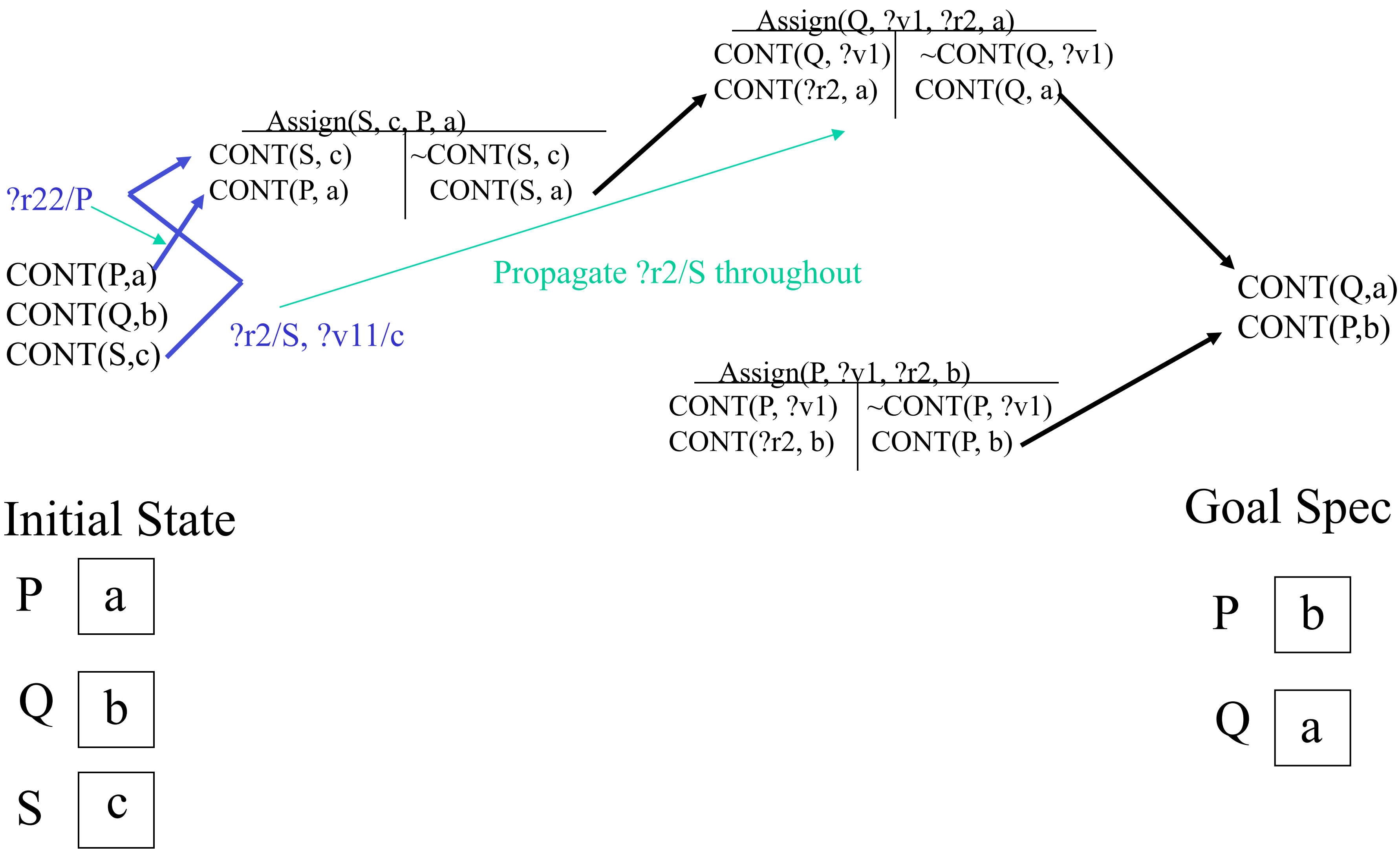
Goal Spec

P	b
Q	a

Assign(?r1, ?v1, ?r2, ?v2) /* ?r1 = ?r2 */
PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
DEL: CONT(?r1, ?v1)
ADD: CONT(?r1, ?v2)

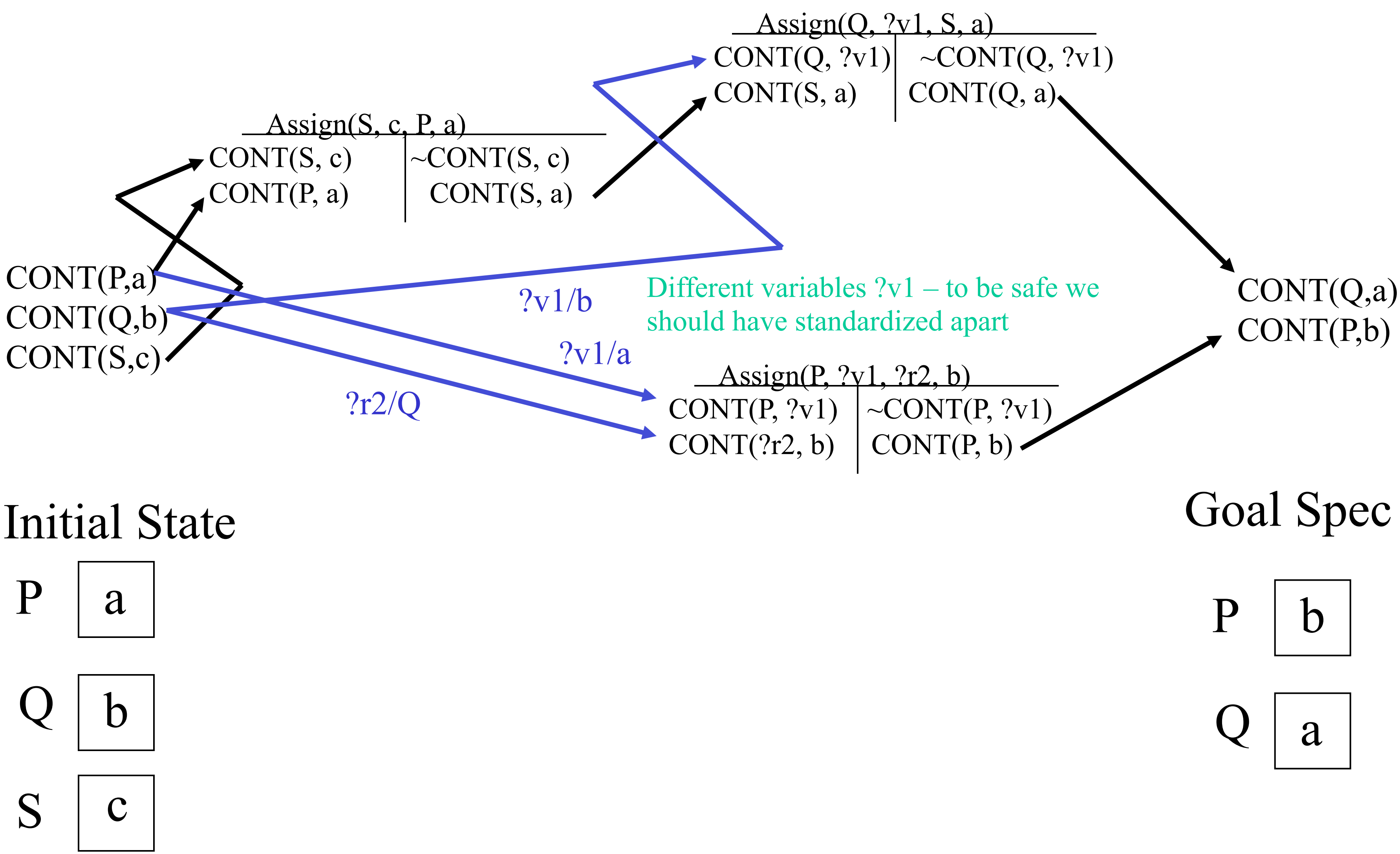


```
Assign(?r1, ?v1, ?r2, ?v2)   /*  ?r1 = ?r2  */
   PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
   DEL: CONT(?r1, ?v1)
   ADD: CONT(?r1, ?v2)
```

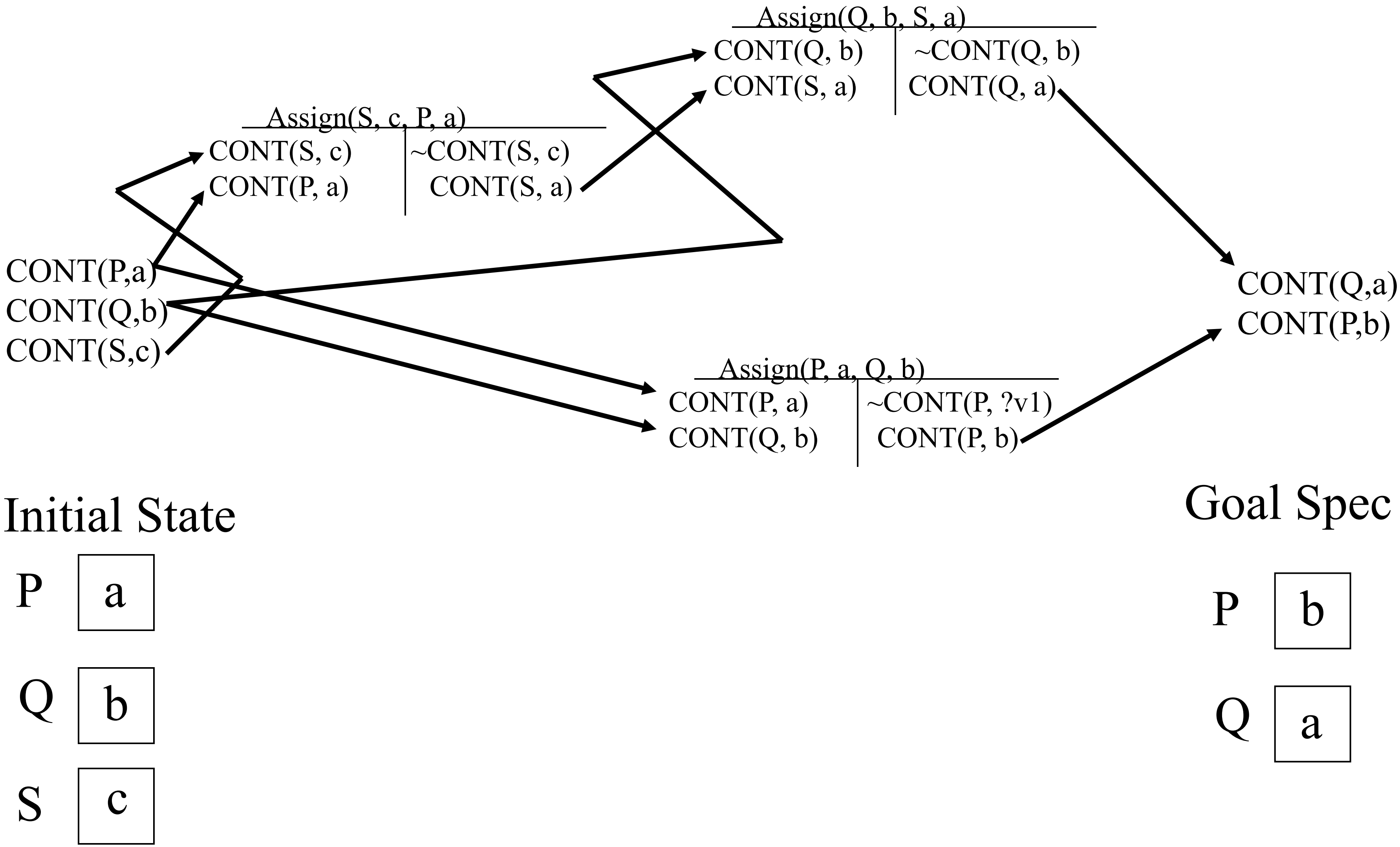


Douglas H. Fisher

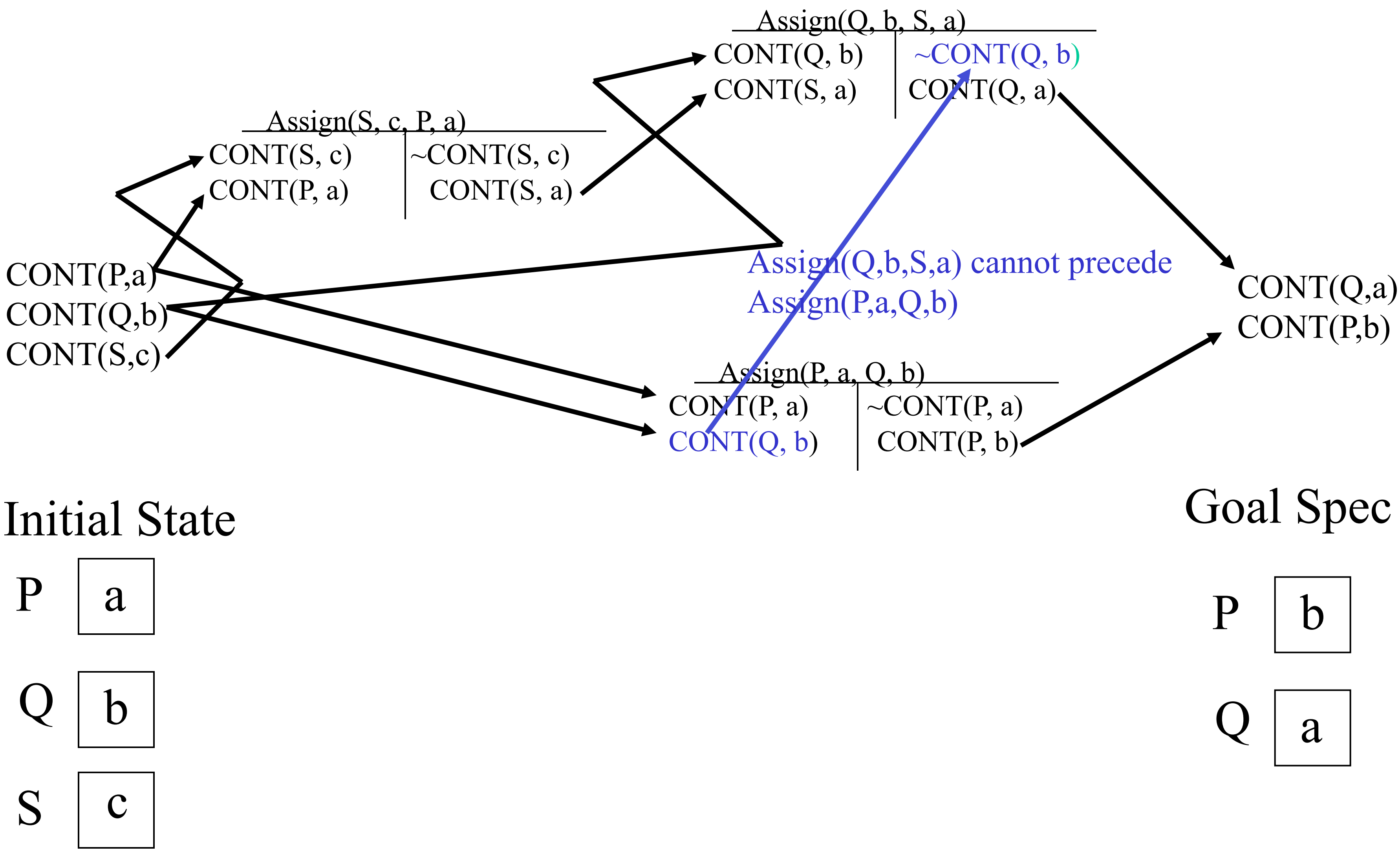
```
Assign(?r1, ?v1, ?r2, ?v2)   /*  ?r1 = ?r2  */
   PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
   DEL: CONT(?r1, ?v1)
   ADD: CONT(?r1, ?v2)
```



```
Assign(?r1, ?v1, ?r2, ?v2)   /*  ?r1 = ?r2  */
   PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
   DEL: CONT(?r1, ?v1)
   ADD: CONT(?r1, ?v2)
```

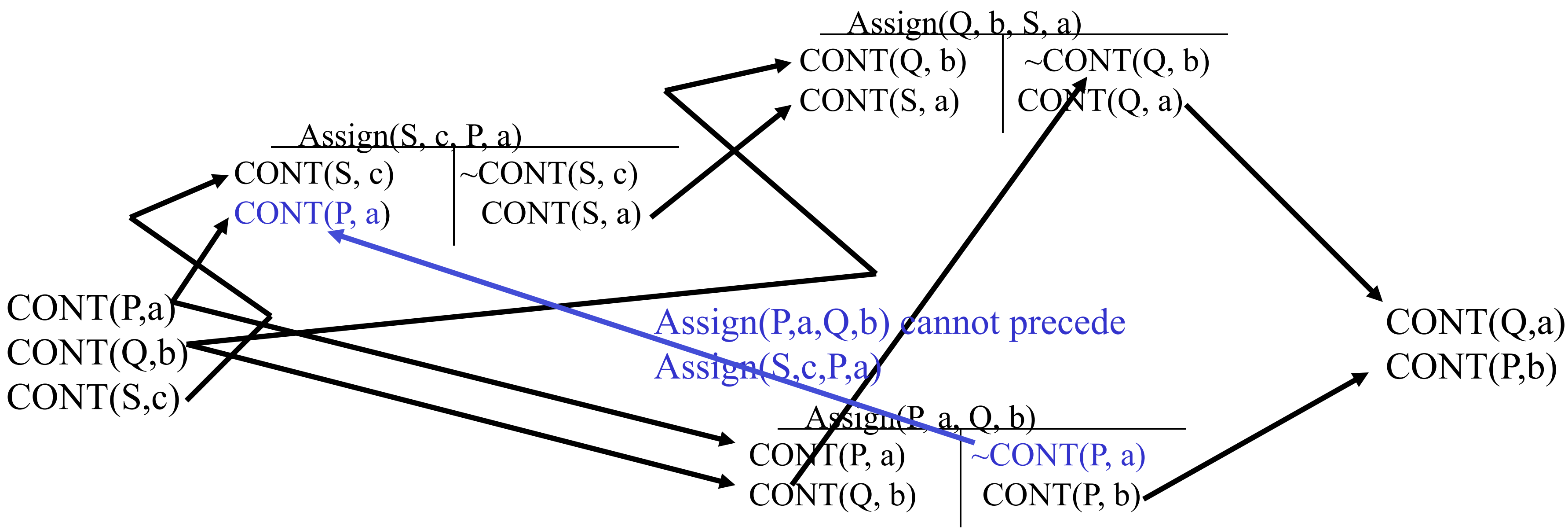


Assign(?r1, ?v1, ?r2, ?v2) /* ?r1 = ?r2 */
PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
DEL: CONT(?r1, ?v1)
ADD: CONT(?r1, ?v2)

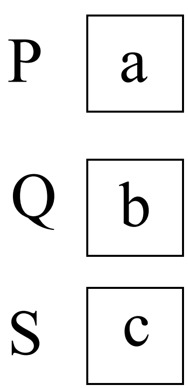


Douglas H. Fisher

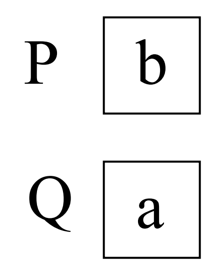
Assign(?r1, ?v1, ?r2, ?v2) /* ?r1 = ?r2 */
PRE: CONT(?r1, ?v1), CONT(?r2, ?v2)
DEL: CONT(?r1, ?v1)
ADD: CONT(?r1, ?v2)



Initial State



Goal Spec



Only one consistent order:

Assign(S,c,P,a) [S = P]

→ Assign(P,a,Q,b) [P = Q]

→ Assign(Q,b,S,a) [Q = S]

Douglas H. Fisher

Relational Learning

Relational (e.g., first-order) representations, such as:

IF R(?c1, ?r1) $\wedge$ R(?c2, ?r1) $\wedge$ R(?c3, ?r2) $\wedge$ R(?c4, ?r2) $\wedge$ R(?c5, ?r2)
$\wedge$ $\neq$(?c1, ?c2) $\wedge$ $\neq$(?c3, ?c4) $\wedge$ $\neq$(?c3, ?c5) $\wedge$ $\neq$(?c4, ?c5)
THEN FullHouse(?c1, ?c2, ?c3, ?c4, ?c5)

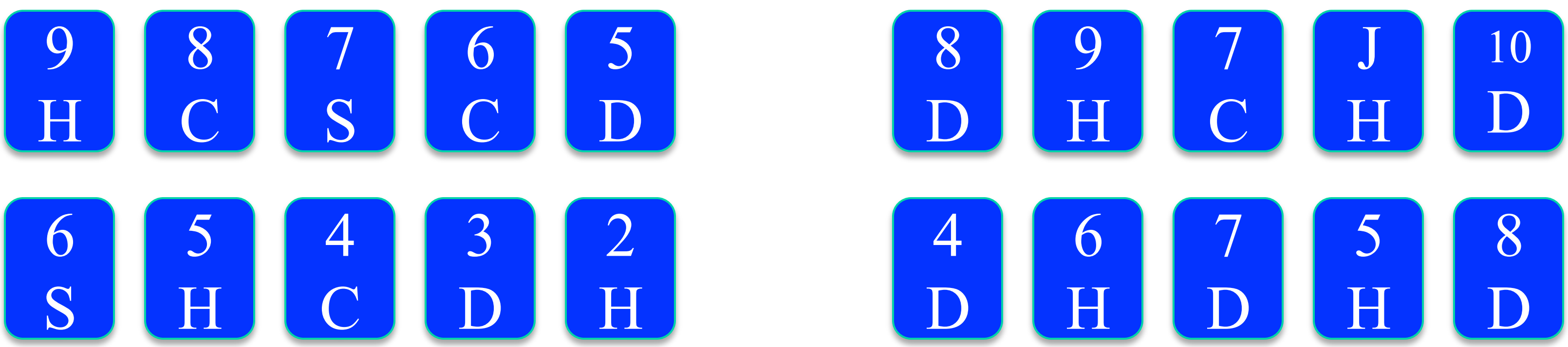
Q H	Q C	5 S	5 C	5 D
6 S	6 H	7 C	7 C	7 H

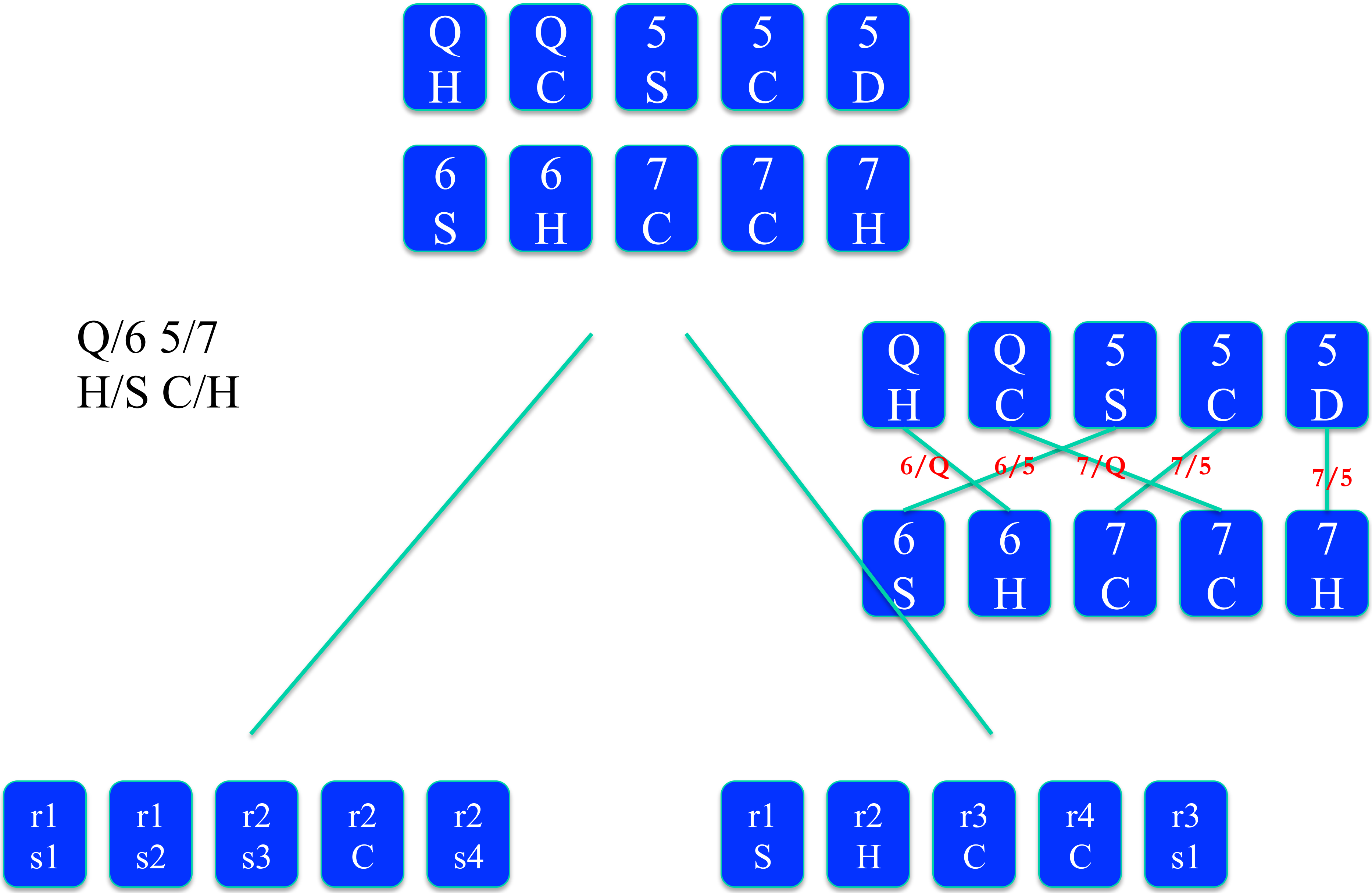
2 D	9 H	2 C	2 H	9 D
A C	3 C	3 D	A H	3 H

Relational Learning

Relational (e.g., first-order) representations, such as:

IF R(?c1, ?r1) ∧ R(?c2, ?r2) ∧ R(?c3, ?r3) ∧ R(?c4, ?r4) ∧ R(?c5, ?r5)
∧ +1(?r1, ?r2) ∧ +1(?r2, ?r3) ∧ +1(?r3, ?r4) ∧ +1(?r4, ?r5)
THEN Straight(?c1, ?c2, ?c3, ?c4, ?c5)

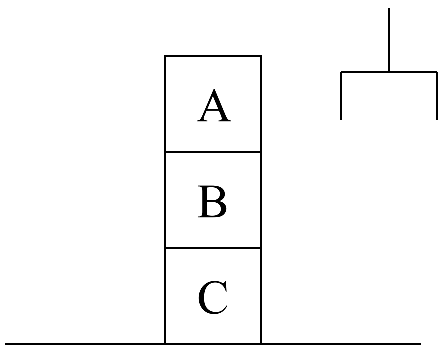




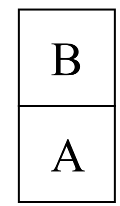
Douglas H. Fisher

Learning macros: Given a plan, generalize the plan so that the generalized plan can be applied in a greater number of situations

Objective: reusing previously-developed generalized plans (aka macro-operators) will reduce the cost (improve the "speed") of subsequent planning



Start State

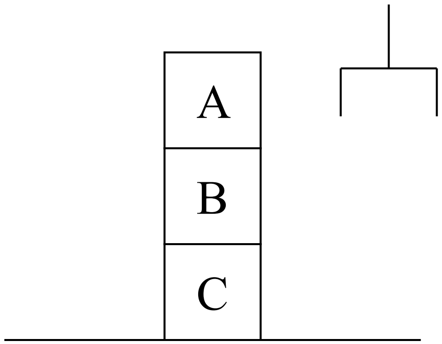


GoalSpec

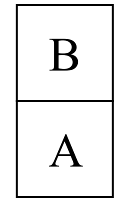
Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)

(Generalize) ➔

Unstack(?x1, ?y1) → Putdown(?x1) → Unstack(?y1, ?z1) → Stack(?y1, ?x1)



Start State



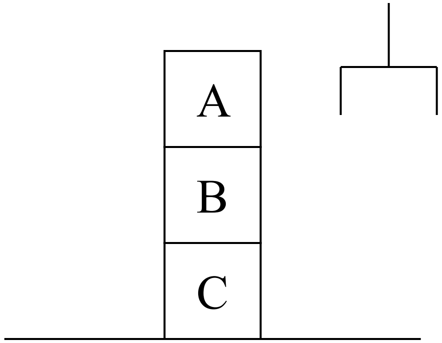
GoalSpec

Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)

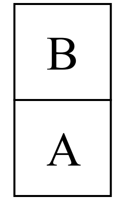
Unstack(?x1, ?y1) → Putdown(?x2) → Unstack(?y2, ?z1) → Stack(?y3, ?x3)

Unstack(?x1, ?y1) pre	Unstack(?x1, ?y1) eff	Putdown(?x2) pre	Putdown(?x2) eff	Unstack(?y2, ?z1) pre	Unstack(?y2, ?z1) eff	Stack(?y3, ?x3) pre	Stack(?y3, ?x3) eff
X On(?x1,?y1)	On(?x1,?y1)	Holding(?x2)	Holding(?x2)	X On(?y2,?z1)	On(?y2,?z1)	Holding(?y3)	Holding(?y3)
X Clear(?x1)	Clear(?x1)		Clear(?x2)	Clear(?y2)	Clear(?y2)	Clear(?x3)	Clear(?x3)
X Handemp()	Handemp()		Handemp()	Handemp()	Handemp()		Handemp()
	Holding(?x1)		OnTab(?x2)		Holding(?y2)		Clear(?y3)
	Clear(?y1)				Clear(?z1)		**On(?y3,?x3)**

Learning macros:



Start State



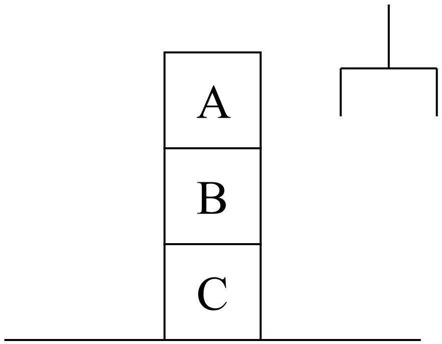
GoalSpec

Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)

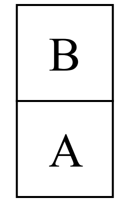
{?x3/?x2}

Unstack(?x1, ?y1)		→ Putdown(?x2)		→ Unstack(?y2, ?z1)		→ Stack(?y3, ?x3)	
X On(?x1,?y1)	On(?x1,?y1)	Holding(?x2)	Holding(?x2)	X On(?y2,?z1)	On(?y2,?z1)	Holding(?y3)	Holding(?y3)
X Clear(?x1)	Clear(?x1)		Clear(?x2)	Clear(?y2)	Clear(?y2)	Clear(?x3)	Clear(?x3)
X Handemp()	Handemp()		Handemp()	Handemp()	Handemp()	{?y3/?y2}	Handemp()
	Holding(?x1)		OnTab(?x2)		Holding(?y2)		Clear(?y3)
	Clear(?y1)				Clear(?z1)		On(?y3,?x3)

Learning macros:



Start State



GoalSpec

Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)

{?x3/?x2}

Unstack(?x1, ?y1) → Putdown(?x2) → Unstack(?y2, ?z1) → Stack(?y2, ?x2)

Unstack(?x1, ?y1)
- Preconditions: X On(?x1,?y1), X Clear(?x1), X Handemp()
- Effects: On(?x1,?y1), Clear(?x1), Handemp() / Holding(?x1), Clear(?y1)

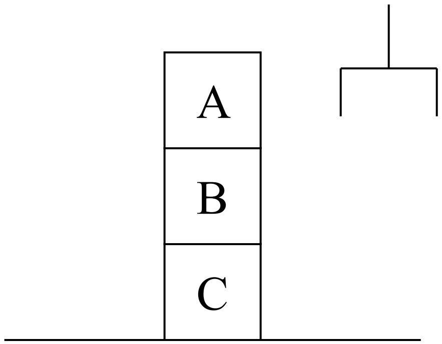
Putdown(?x2)
- Preconditions: Holding(?x2)
- Effects: Holding(?x2) / Clear(?x2), Handemp(), OnTab(?x2)

Unstack(?y2, ?z1)
- Preconditions: X On(?y2,?z1), Clear(?y2), Handemp()
- Effects: On(?y2,?z1), Clear(?y2), Handemp() / Holding(?y2), Clear(?z1)

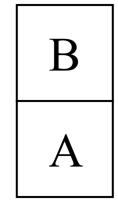
{?y3/?y2}

Stack(?y2, ?x2)
- Preconditions: Holding(?y2), Clear(?x2)
- Effects: Holding(?y2), Clear(?x2), Handemp() / Clear(?y2), On(?y2,?x2)

Learning macros:



Start State



GoalSpec

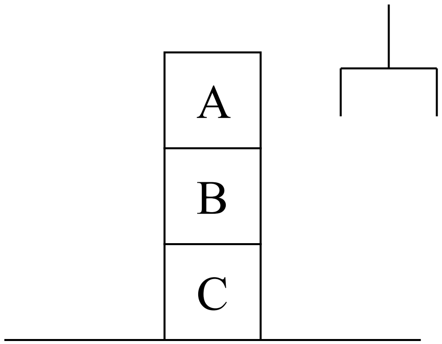
Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)

Unstack(?x1, ?y1)		→ Putdown(?x2)		→ Unstack(?y2, ?z1)		→ Stack(?y2, ?x2)	
X On(?x1,?y1)	On(?x1,?y1)	Holding(?x2)	Holding(?x2)	X On(?y2,?z1)	On(?y2,?z1)	Holding(?y2)	Holding(?y2)
X Clear(?x1)	Clear(?x1)		Clear(?x2)	Clear(?y2)	Clear(?y2)	Clear(?x2)	Clear(?x2)
X Handemp()	Handemp()		Handemp()	Handemp()	Handemp()		Handemp()
	Holding(?x1)		OnTab(?x2)		Holding(?y2)		Clear(?y2)
	Clear(?y1)				Clear(?z1)		On(?y2,?x2)

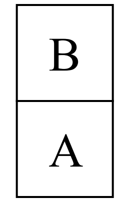
{?y2/?y1}

Unstack(?x1, ?y1)		→ Putdown(?x2)		→ Unstack(?y1, ?z1)		→ Stack(?y1, ?x2)	
X On(?x1,?y1)	On(?x1,?y1)	Holding(?x2)	Holding(?x2)	X On(?y1,?z1)	On(?y1,?z1)	Holding(?y1)	Holding(?y1)
X Clear(?x1)	Clear(?x1)		Clear(?x2)	Clear(?y1)	Clear(?y1)	Clear(?x2)	Clear(?x2)
X Handemp()	Handemp()		Handemp()	Handemp()	Handemp()		Handemp()
	Holding(?x1)		OnTab(?x2)		Holding(?y1)		Clear(?y1)
	Clear(?y1)				Clear(?z1)		On(?y1,?x2)

Learning macros:



Start State



GoalSpec

Unstack(?x1, ?y1) → Putdown(?x2) → Unstack(?y1, ?z1) → Stack(?y1, ?x2)

Unstack(?x1, ?y1)
- X On(?x1,?y1)
- X Clear(?x1)
- X Handemp()

| On(?x1,?y1) |
| Clear(?x1) |
Handemp()
Holding(?x1)
Clear(?y1)

Putdown(?x2)
- Holding(?x2)

Holding(?x2)
Clear(?x2)
Handemp()
OnTab(?x2)

{?x2/?x1}

Unstack(?y1, ?z1)
- X On(?y1,?z1)
- Clear(?y1)
- Handemp()

| On(?y1,?z1) |
| Clear(?y1) |
Handemp()
Holding(?y1)
Clear(?z1)

Stack(?y1, ?x2)
- Holding(?y1)
- Clear(?x2)

| Holding(?y1) |
Clear(?x2)
Handemp()
Clear(?y1)
On(?y1,?x2)

Unstack(?x1, ?y1) → Putdown(?x1) → Unstack(?y1, ?z1) → Stack(?y1, ?x1)

Unstack(?x1, ?y1)
- X On(?x1,?y1)
- X Clear(?x1)
- X Handemp()

| On(?x1,?y1) |
| Clear(?x1) |
Handemp()
Holding(?x1)
Clear(?y1)

Putdown(?x1)
- Holding(?x1)

Holding(?x1)
Clear(?x1)
Handemp()
OnTab(?x1)

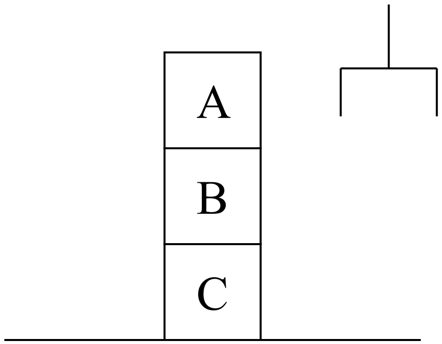
Unstack(?y1, ?z1)
- X On(?y1,?z1)
- Clear(?y1)
- Handemp()

| On(?y1,?z1) |
| Clear(?y1) |
Handemp()
Holding(?y1)
Clear(?z1)

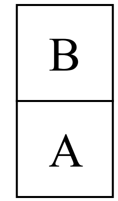
Stack(?y1, ?x1)
- Holding(?y1)
- Clear(?x1)

| Holding(?y1) |
Clear(?x1)
Handemp()
Clear(?y1)
On(?y1,?x1)

Learning macros:



Start State



GoalSpec

Unstack(?x1, ?y1) → Putdown(?x1) → Unstack(?y1, ?z1) → Stack(?y1, ?x1)

Unstack(?x1, ?y1)

Preconditions	Effects
X On(?x1,?y1)	On(?x1,?y1)
X Clear(?x1)	Clear(?x1)
X Handemp()	Handemp()
	———
	Holding(?x1)
	Clear(?y1)

→ Putdown(?x1)

Preconditions	Effects
Holding(?x1)	Holding(?x1)
	———
	Clear(?x1)
	Handemp()
	OnTab(?x1)

→ Unstack(?y1, ?z1)

Preconditions	Effects
X On(?y1,?z1)	On(?y1,?z1)
Clear(?y1)	Clear(?y1)
Handemp()	Handemp()
	———
	Holding(?y1)
	Clear(?z1)

→ Stack(?y1, ?x1)

Preconditions	Effects
Holding(?y1)	Holding(?y1)
Clear(?x1)	Clear(?x1)
	———
	Handemp()
	Clear(?y1)
	On(?y1,?x1)

Macrop(?x1, ?y1, ?z1)

Preconditions	Effects
On(?x1, ?y1)	On(?x1, ?y1)
On(?y1, ?z1)	Clear(?x1)
Clear(?x1)	On(?y1, ?z1)
Handemp()	———
	Clear(?y1)
	OnTab(?x1)
	Clear(?z1)
	On(?y1, ?x1)

Douglas H. Fisher