CS 6360
Vanderbilt University

Notes on Relational Learning, Including Explanation-Based Learning

This lecture assumes that you have

- Completed a introductory course on AI with an intro to ML
- Read 19.1 and 19.2 of Russell and Norvig

Douglas H. Fisher

Relational Learning

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

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

More compact than a disjunction of all possible hands satisfying FH

Sample FHs

Q H	Q C	5 S	5 C	5 D	2 D	9 H	2 C	2 H	9 D
6 S	6 H	7 C	7 C	7 H	A C	3 C	3 D	A H	3 H

Relational Learning

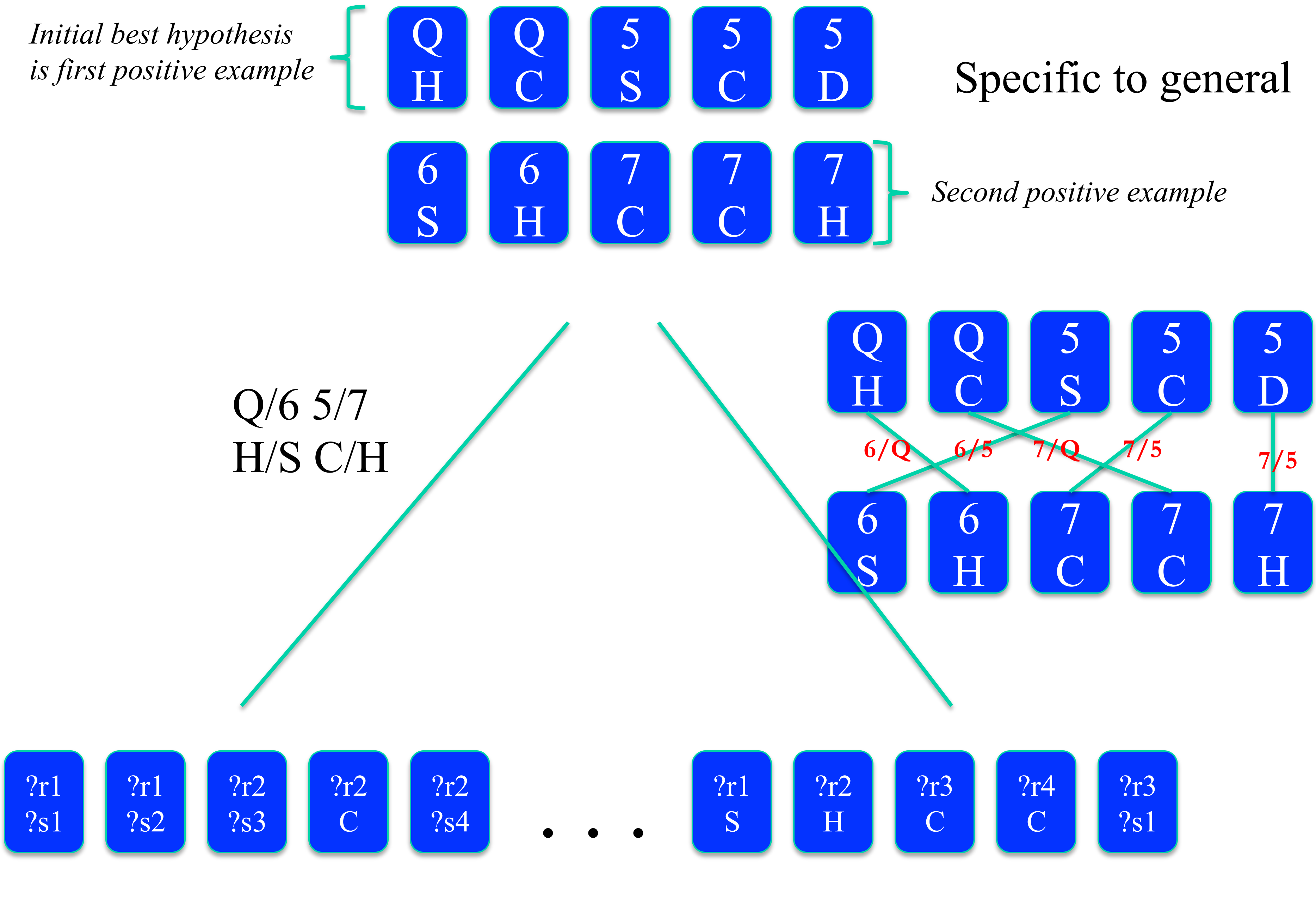
Relational (e.g., first-order) representations, such as:

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

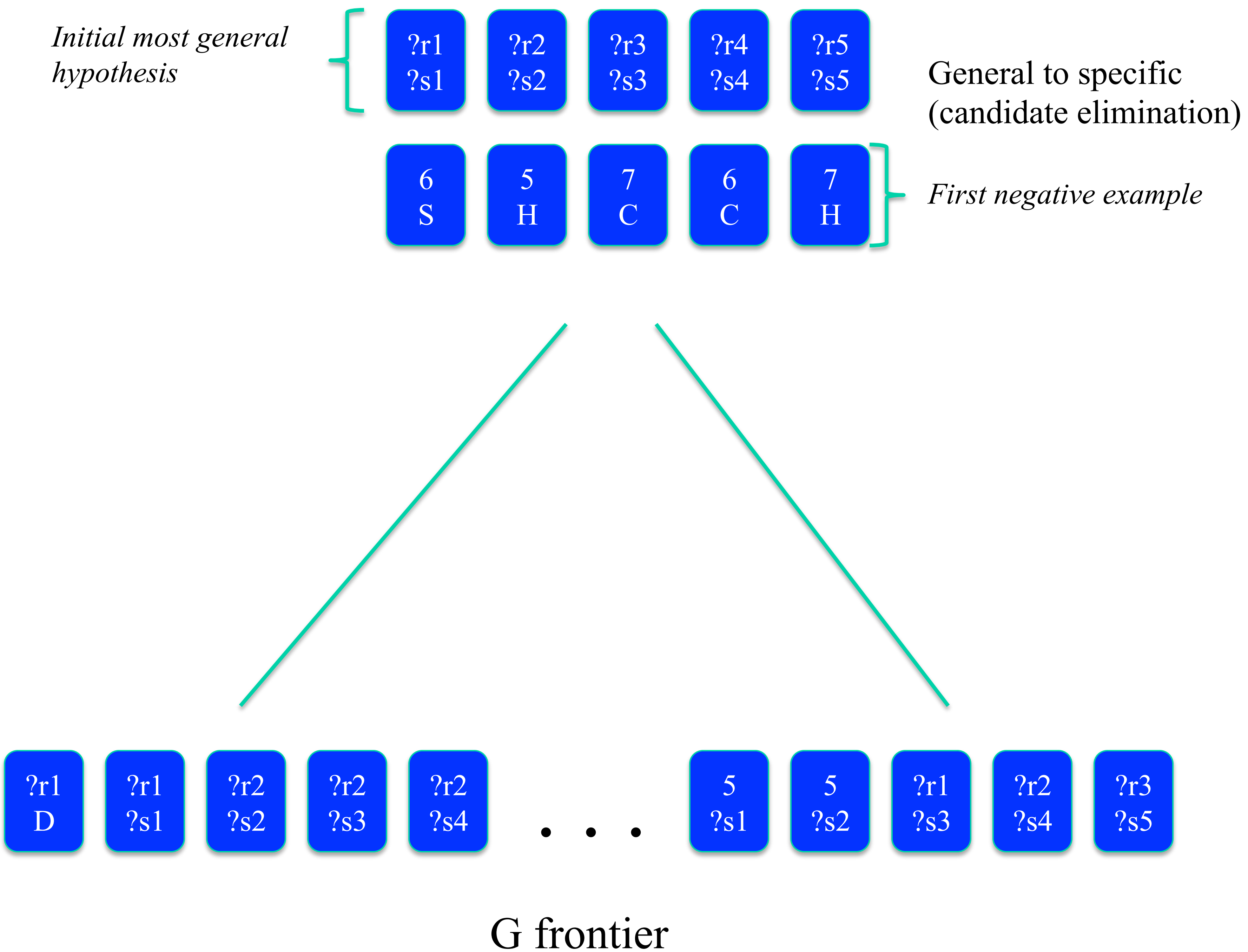
Sample Straights

9 H	8 C	7 S	6 C	5 D	8 D	9 H	7 C	J H	10 D
6 S	5 H	4 C	3 D	2 H	4 D	6 H	7 D	5 H	8 D

Douglas H. Fisher



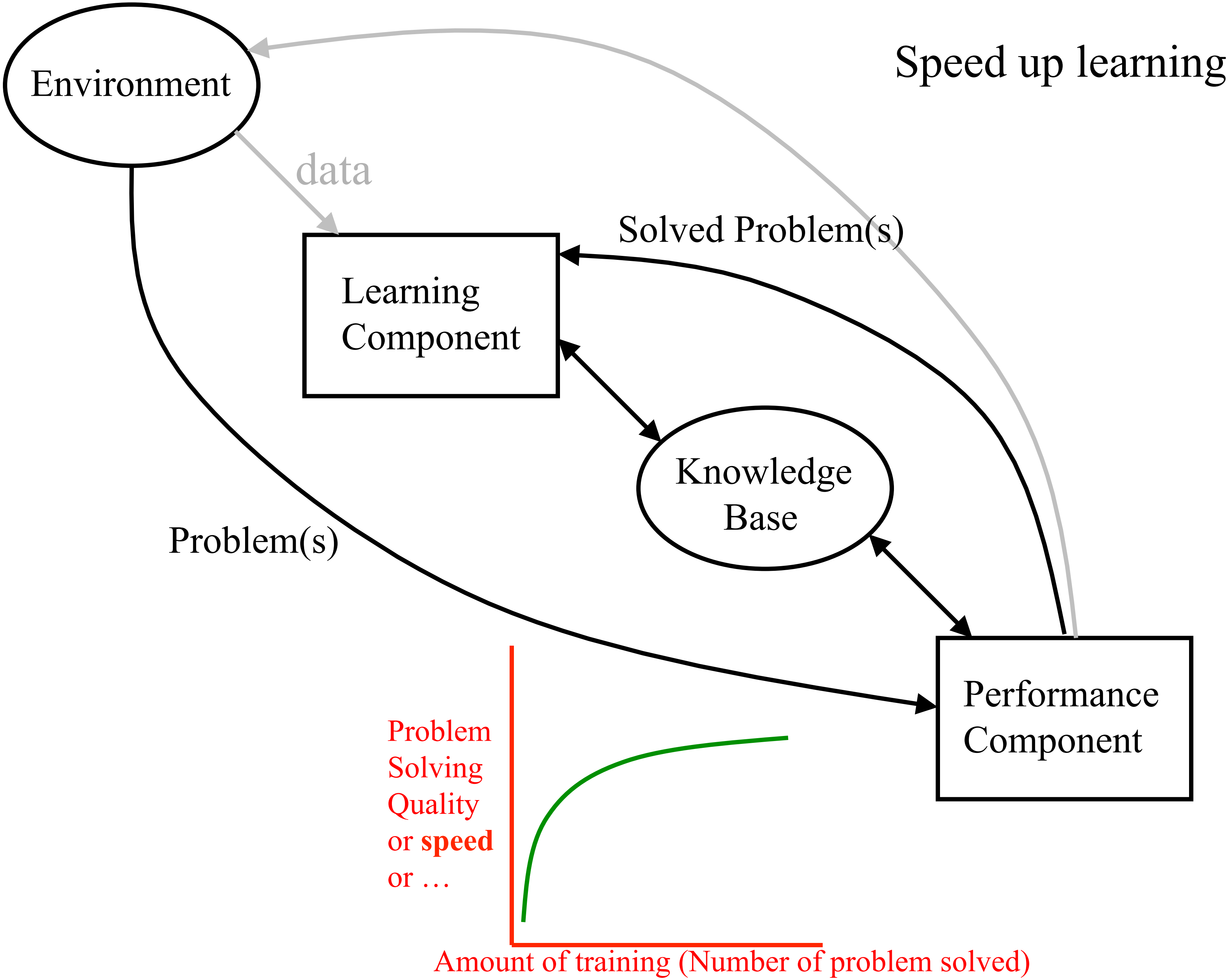
Douglas H. Fisher



Douglas H. Fisher

Exercises

1) Continue the specific-to-general search for 1-2 more positive examples
2) Add additional entries to the G frontier for the general-to-specific search step shown
3) Continue the general-to-specific search for 1-2 more negative examples (you need only show 2 members of the expanded G frontier)
4) Show an example of a situation where a member of the S frontier is more general than an item on the G frontier. What is the relevance of this to a bidirectional search that combines specific-to-general and general to specific (see discussion on p. 775)
5) Advanced: What if we did the following? Instead of expanding G with negative examples as described in the text, we generalized negative examples with members of the S frontier, which would create a G' frontier that was part of the inconsistent region.



Speed up learning
Environment
data
Solved Problem(s)
Learning Component
Knowledge Base
Problem(s)
Performance Component
Problem Solving Quality or **speed** or …
Amount of training (Number of problem solved)

Explanation-based learning

Pseudo-psychology axioms from Mooney and DeJong

Hate(?x, ?y) and Possess(?x, ?z) and Weapon(?z) → Assault(?x, ?y)

Depressed(?w) → Hate(?w, ?w)

Buy(?u, ?v) → Possess(?u, ?v)

Gun(?s) → Weapon(?s)

Depressed(John)

Buy(John, Ob1)

Gun(Obj1)

Augmented Proof tree for Assault(John, John)

Assault(John, John)

Assault(?x,?y) {?x/John, ?y/John}

Hate(?x, ?y) and Possess(?x, ?z) and Weapon(?z)

Hate(John, John) and Possess(John, ?z) and Weapon(?z)

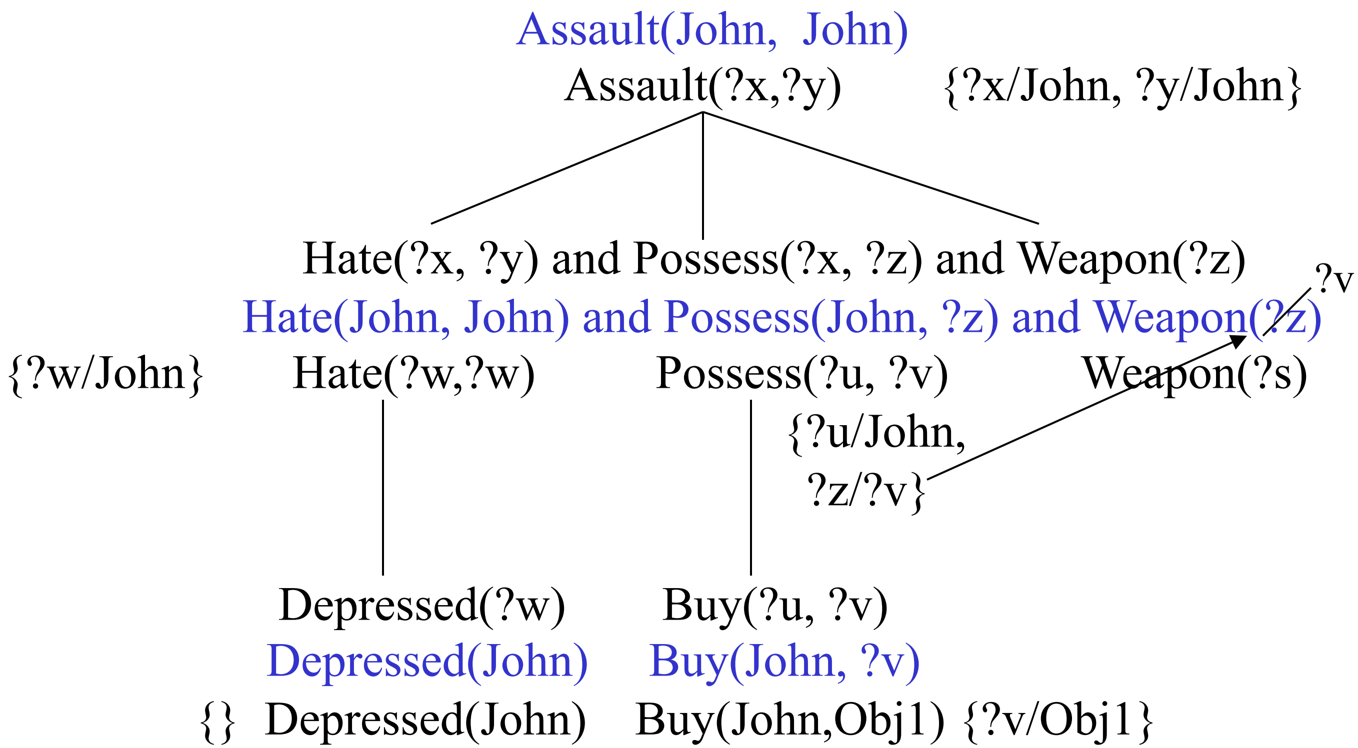
{?w/John} Hate(?w,?w) Possess(?u, ?v) Weapon(?s)

Depressed(?w)

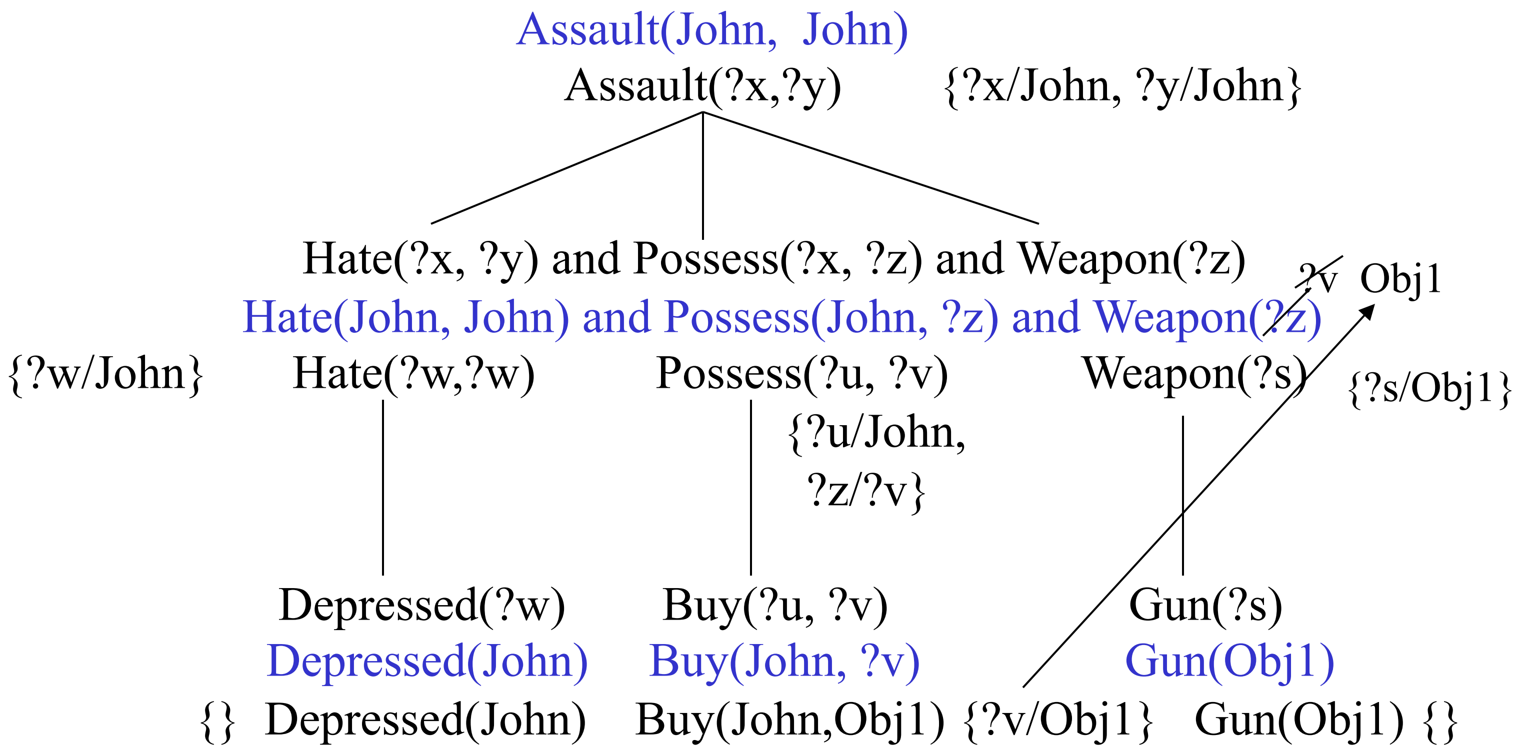
Depressed(John)

{} Depressed(John)

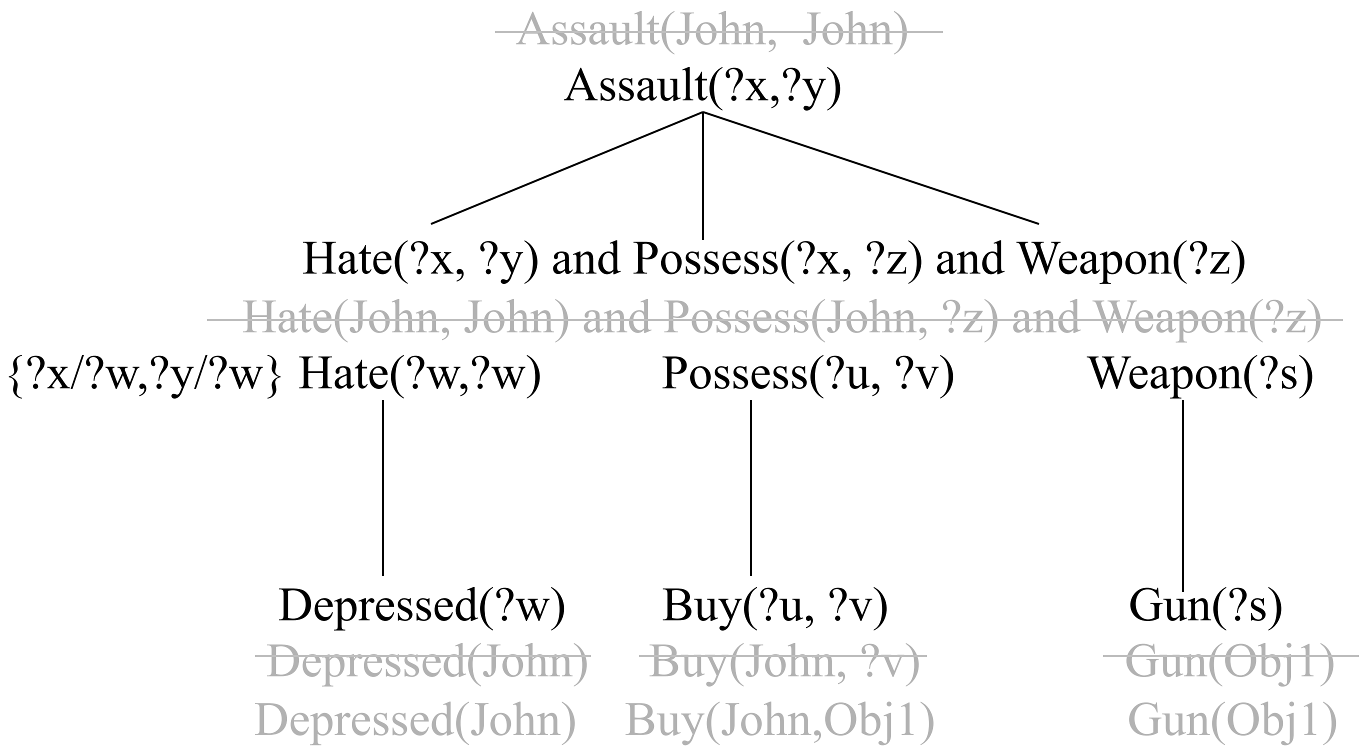
Augmented Proof tree for Assault(John, John)



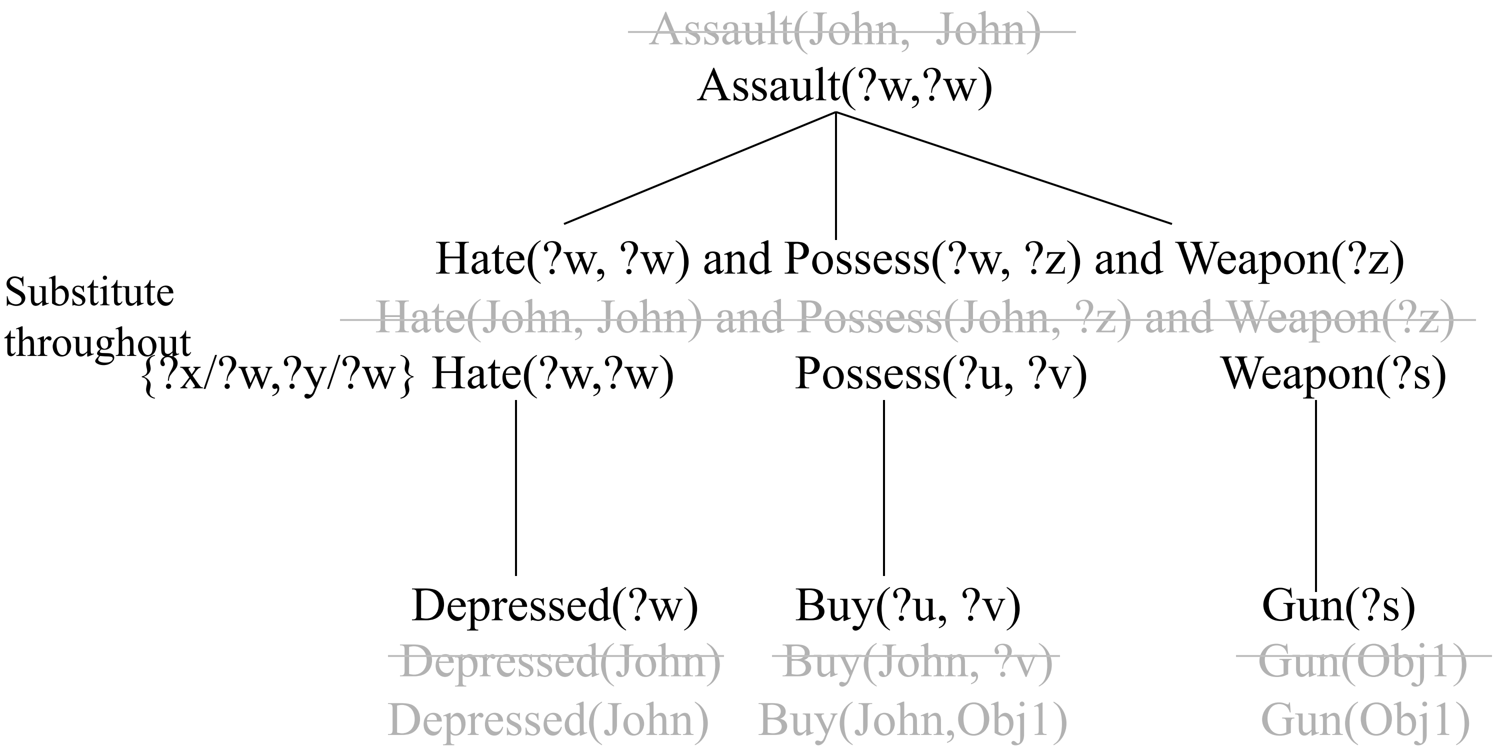
Augmented Proof tree for Assault(John, John)



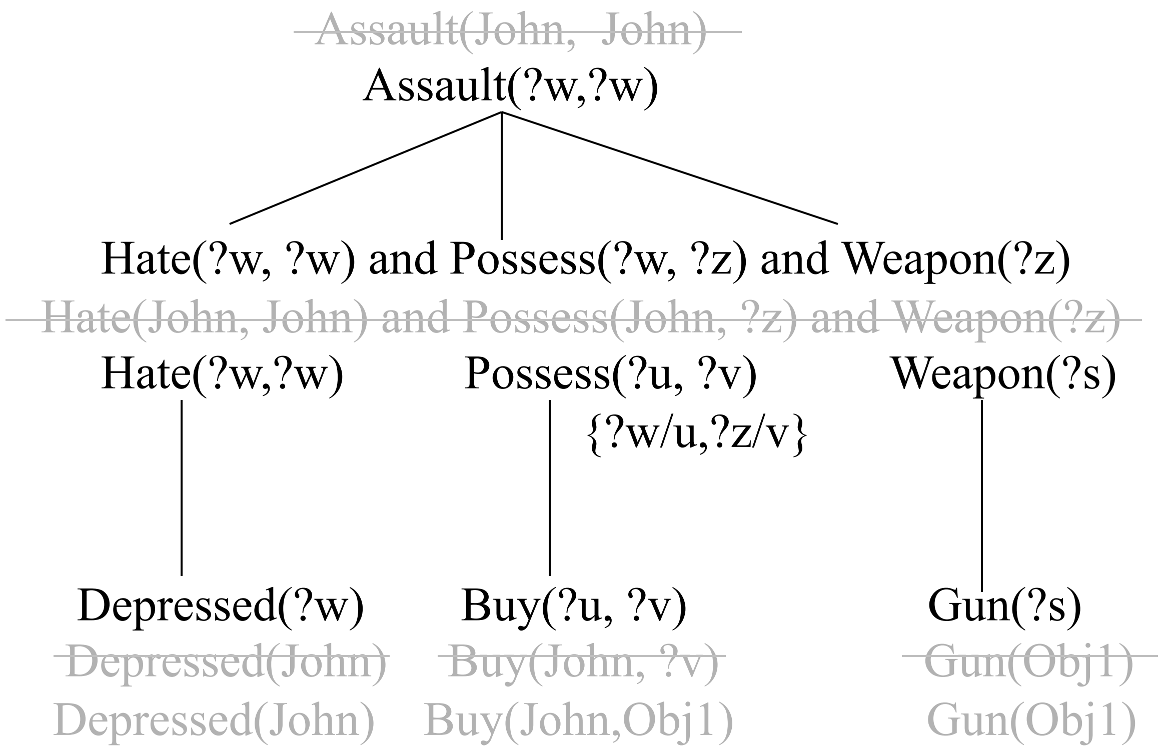
Generalize Proof tree for Assault(John, John)



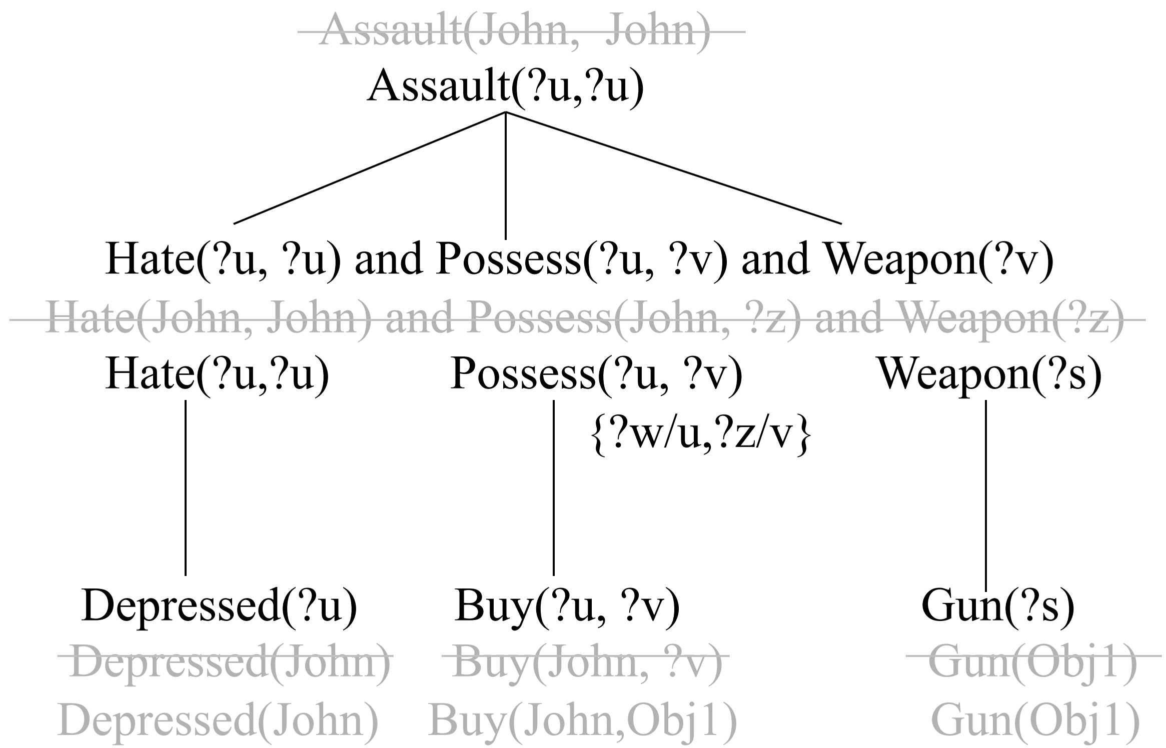
Generalize Proof tree for Assault(John, John)



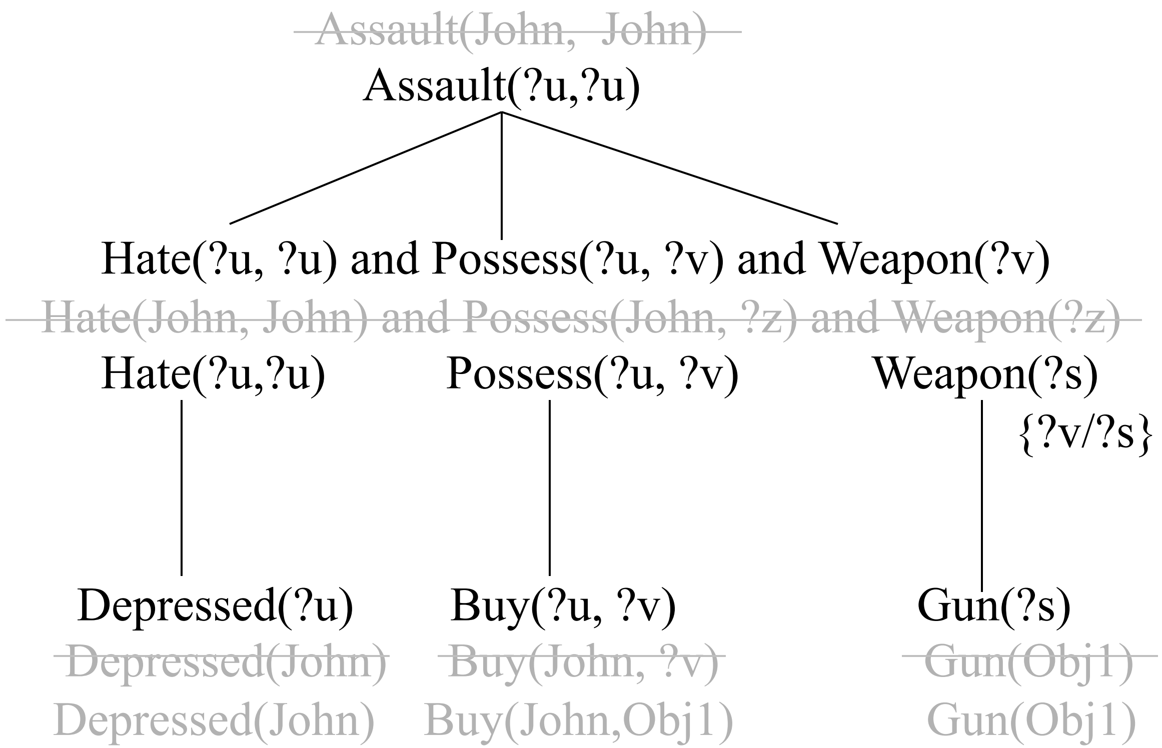
Generalize Proof tree for Assault(John, John)



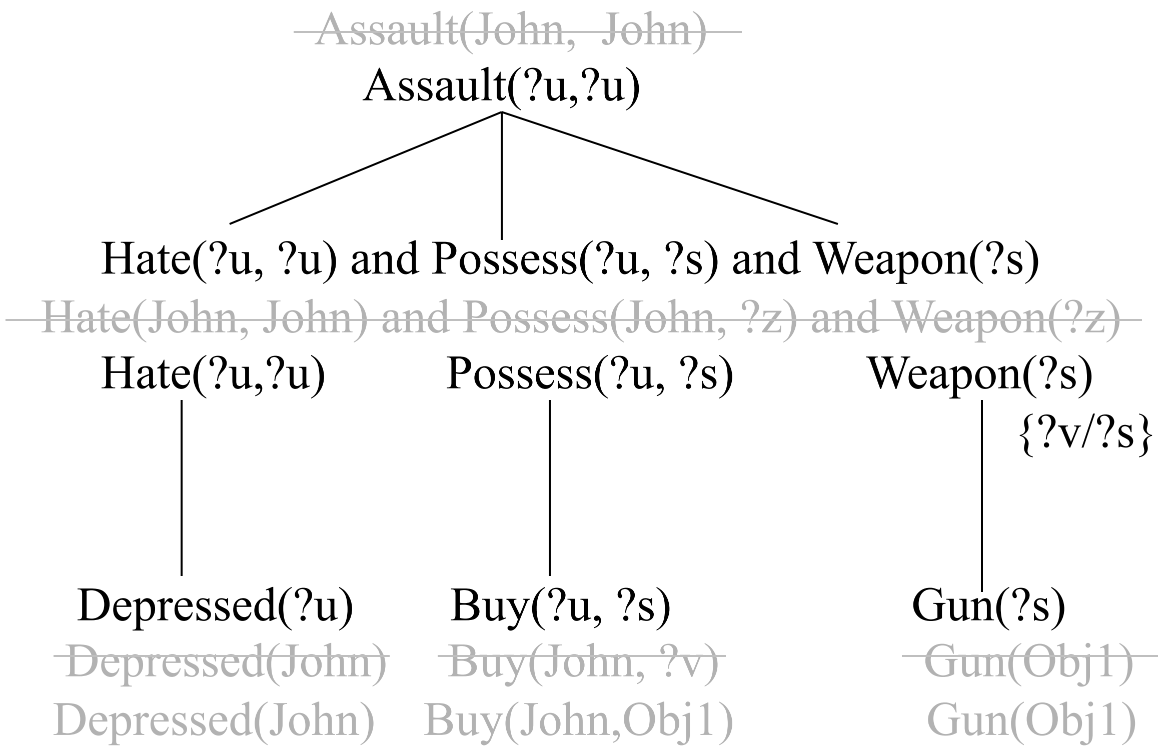
Generalize Proof tree for Assault(John, John)



Generalize Proof tree for Assault(John, John)

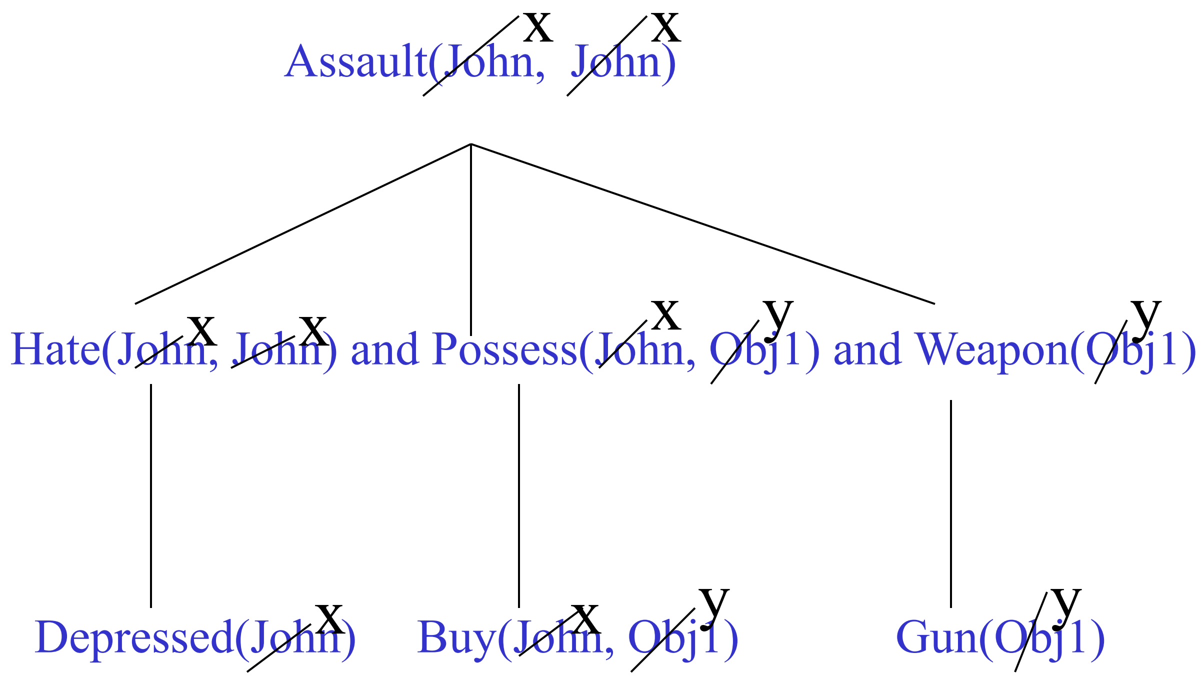


Generalize Proof tree for Assault(John, John)



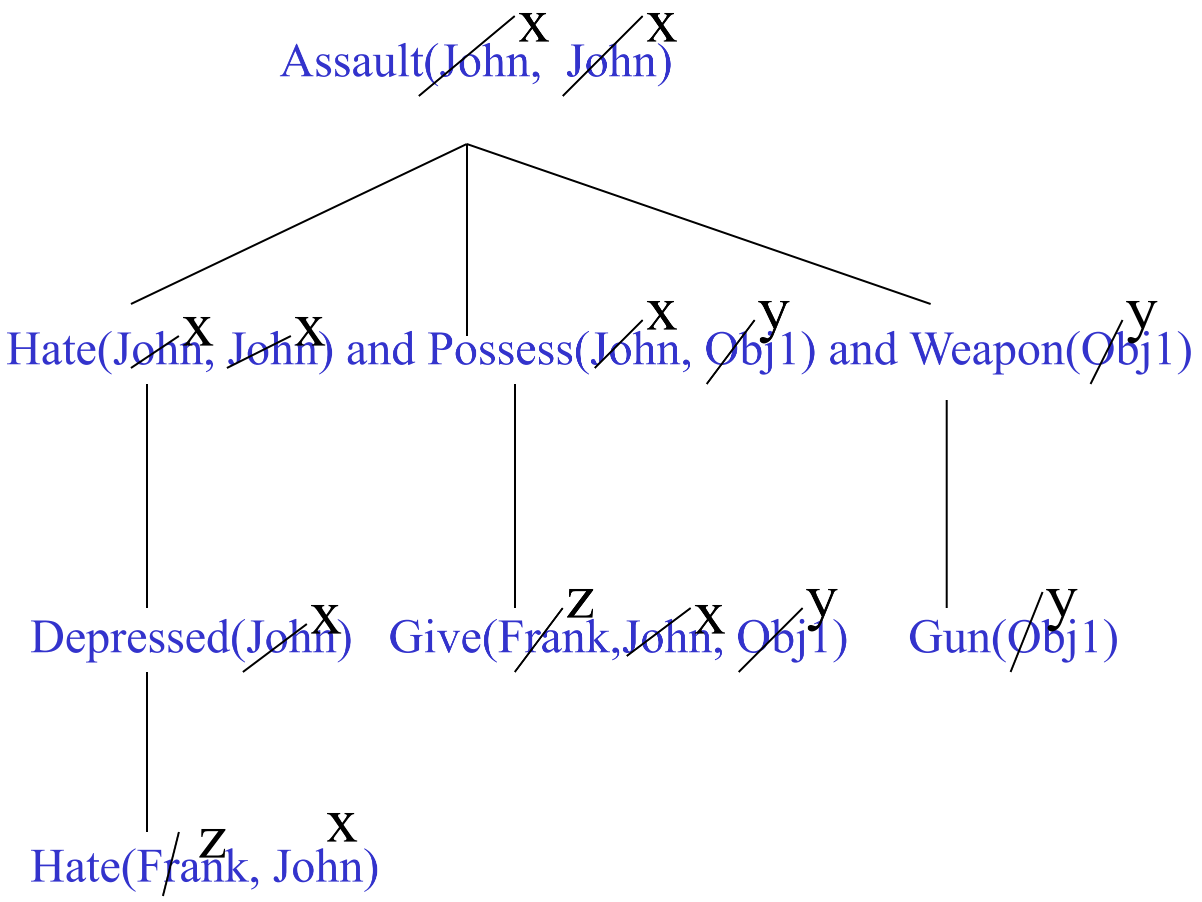
Add (Depressed(?u') and Buy(?u',?s') and Gun(?u',?s')) → Assault(?u',?u')
to knowledge base or
~Depressed(?u') or ~Buy(?u',?s') or ~Gun(?u',?s') or Assault(?u',?u')

Why not simply replace constants by variables in a consistent manner?



Suppose that Depressed(?x) → Hate(?x,?x) was not an axiom, but Depressed(John) → Hate(John, John) was an axiom. The same final proof tree would result, but generalizing as above would not be valid. (*invalid generalization*)

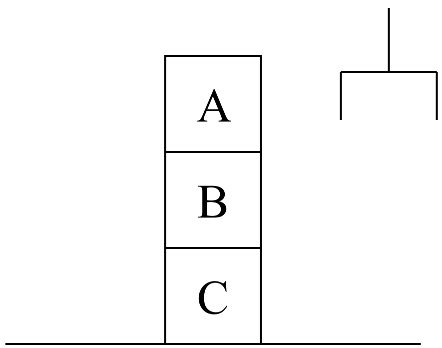
Why not simply replace constants by variables in a consistent manner?



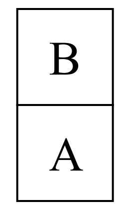
Suppose Hate(?w, John) → Depressed(John) an axiom and Give(Frank, John, Obj1) is an axiom, then generalization above (that the person hating John and the person giving the obj1 have to be the same) is *overly restrictive*.

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

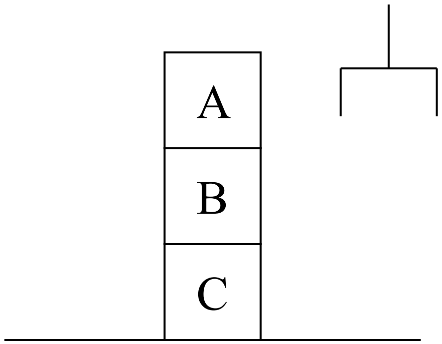


GoalSpec

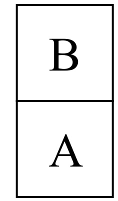
Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)

(Generalize) ➔

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

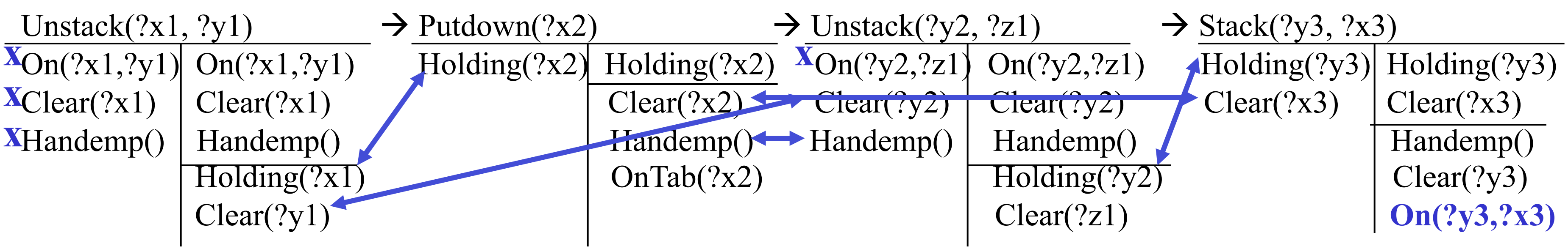


Start State

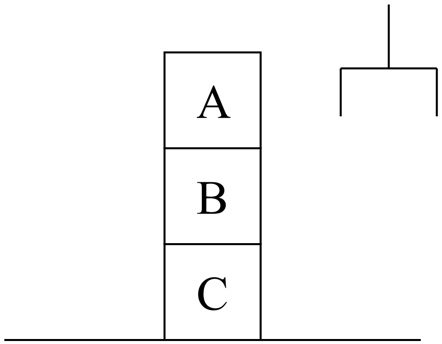


GoalSpec

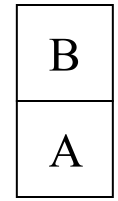
Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)



Learning macros:



Start State



GoalSpec

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

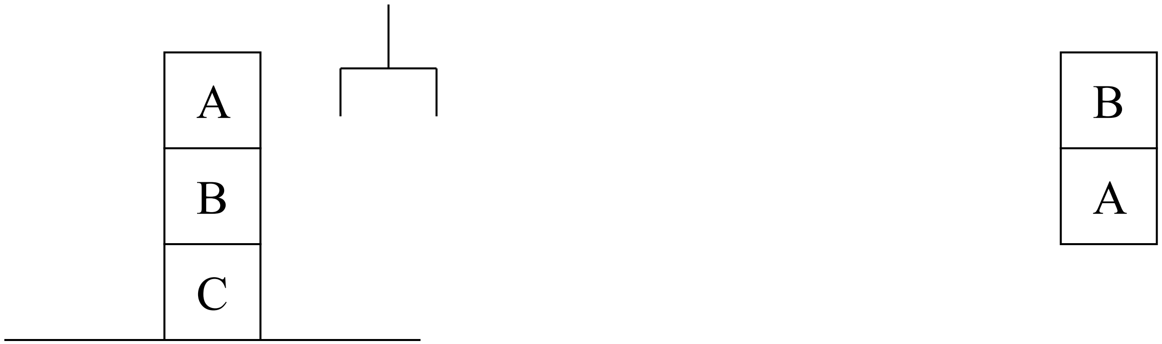
{?x3/?x2}

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

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()		Handemp()
	Holding(?x1)		OnTab(?x2)		Holding(?y2)		Clear(?y3)
	Clear(?y1)				Clear(?z1)		On(?y3,?x3)

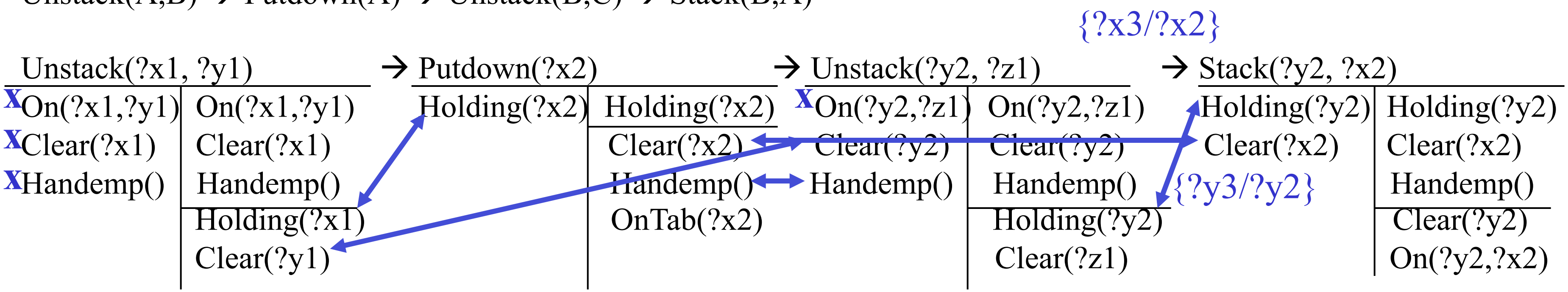
{?y3/?y2}

Learning macros:

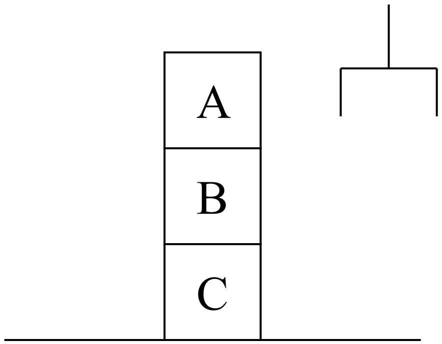


Start State **GoalSpec**

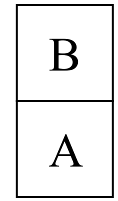
Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)



Learning macros:



Start State



GoalSpec

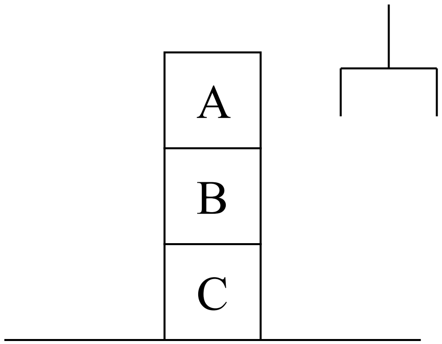
Unstack(A,B) → Putdown(A) → Unstack(B,C) → Stack(B,A)

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)

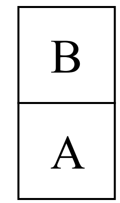
{?y2/?y1}

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)

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

→ Putdown(?x2)

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

{?x2/?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, ?x2)

Preconditions	Effects
Holding(?y1)	Holding(?y1)
Clear(?x2)	Clear(?x2)
	Handemp()
	Clear(?y1)
	On(?y1,?x2)

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)

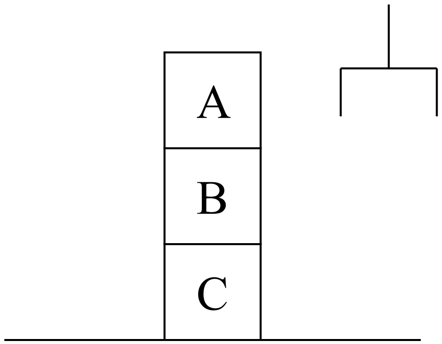
→ Unstack(?y1, ?z1)

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

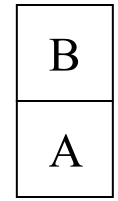
→ Stack(?y1, ?x1)

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

Learning macros:



Start State



GoalSpec

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