Name:
Course: CAP 4601
Semester: Summer 2013
Assignment: Assignment 07
Date: 15 JUL 2013

Complete the following written problems:

NOTE: A programming language that uses Forward Chaining and can be embedded into C++:
C Language Integrated Production System (CLIPS)

NOTE: A programming language that uses Backward Chaining and can be embedded into C++:
SWI Prolog
NOTE: On `linprog4.cs.fsu.edu`, SWI Prolog is available using the `pl` command.

1. Unification (100 Points). State whether the following pairs of atomic sentences can be unified. If they can be unified, write the most general unifier (i.e. the most general substitution that unifies them). If they cannot be unified, write: "No unifier".

a. (20 Points) $Q(x,y)$ and $Q(y,x)$

$$\{x/y\}$$

Because:

$$\begin{aligned} Q(x,y) \stackrel{?}{\equiv} Q(y,x) &\Rightarrow \{x/y\} \\ &\Rightarrow Q(x,x) \stackrel{?}{\equiv} Q(x,x) \\ &\Rightarrow Q(x,x) \equiv Q(x,x) \end{aligned}$$

b. (40 Points) $P\left(g\left(g\left(g(K)\right)\right),K\right)$ and $P\left(g(x),y\right)$

$$\left\{x/g\left(g(K)\right),y/K\right\}$$

Because:

$$\begin{aligned} P\left(g\left(g\left(g(K)\right)\right),K\right) \stackrel{?}{\equiv} P\left(g(x),y\right) &\Rightarrow \left\{x/g\left(g(K)\right),y/K\right\} \\ &\Rightarrow P\left(g(x),y\right) \stackrel{?}{\equiv} P\left(g(x),y\right) \\ &\Rightarrow P\left(g(x),y\right) \equiv P\left(g(x),y\right) \end{aligned}$$

c. (40 Points) $P(g(x),x)$ and $P(z,f(y,z))$

No unifier.

Because:

$$P(g(x),x) \overset{?}{\equiv} P(z,f(y,z)) \Rightarrow \{x/f(y,z)\}$$
$$\Rightarrow P(g(x),x) \overset{?}{\equiv} P(z,x)$$
$$\Rightarrow \{z/g(x)\}$$
$$\Rightarrow P(z,x) \overset{?}{\equiv} P(z,x)$$
$$\Rightarrow \{x/f(y,z), z/g(x)\}$$
$$\Rightarrow \left\{ \underbrace{x/f(y,g(x))}_{\text{Circular}}, z/g(x) \right\}$$
$$\Rightarrow \text{No unifier}$$

Use the following pseudo code for the written problems that follow:

Backward Chaining with Diagrams
by Dr. Chris Darken

Requirements:

- Each time a sentence from the Knowledge Base (KB) is used, the variables will be given unique names by applying a numeric subscript
- Substitutions will be consistently applied across the entire diagram
- P1, P2, … are atoms

BACKWARD-CHAINING-WITH-DIAGRAM (P1, P2, …):

Start diagram by drawing a box for P1, P2, ….
Initialize our subscript counter, *N*, to 1 (i.e. $N \leftarrow 1$).
BACKWARD-CHAIN-THE-NEWEST-BOX().

BACKWARD-CHAIN-THE-NEWEST-BOX:

If all boxes have substitutions or child boxes under them, then exit (The proof has successfully completed).
Choose an unfinished box to work on (i.e. a box with no substitution or child boxes under it).
For this unfinished box, **choose** either MATCH-ATOM or MATCH-RULE (as applicable).

MATCH-ATOM:

Choose a sentence from the knowledge base that unifies with the box.
If there isn't a sentence that unifies with the box, then start the diagram over making different choices.
Write **A**-*AtomID*-*N* under the box.
Write the (possibly empty) substitution under the box.
Increment our subscript counter, *N*, by one (i.e. $N \leftarrow N + 1$).
BACKWARD-CHAIN-THE-NEWEST-BOX().

MATCH-RULE:

Choose a rule whose conclusion on the right-hand-side unifies with the box.
If there isn't a rule that unifies with the box, then start the diagram over making different choices.
Write **R**-*RuleID*-*N* under the box.
Write the (possibly empty) substitution under the box.
Draw child boxes for each premise.
Increment our subscript counter, *N*, by one (i.e. $N \leftarrow N + 1$).
BACKWARD-CHAIN-THE-NEWEST-BOX().

Since this is just slightly more rigorous than the what Dr. Dusgupta presented, let's look at an example … esp. since we will be doing this on the Final Exam:

Given the following *West* example:

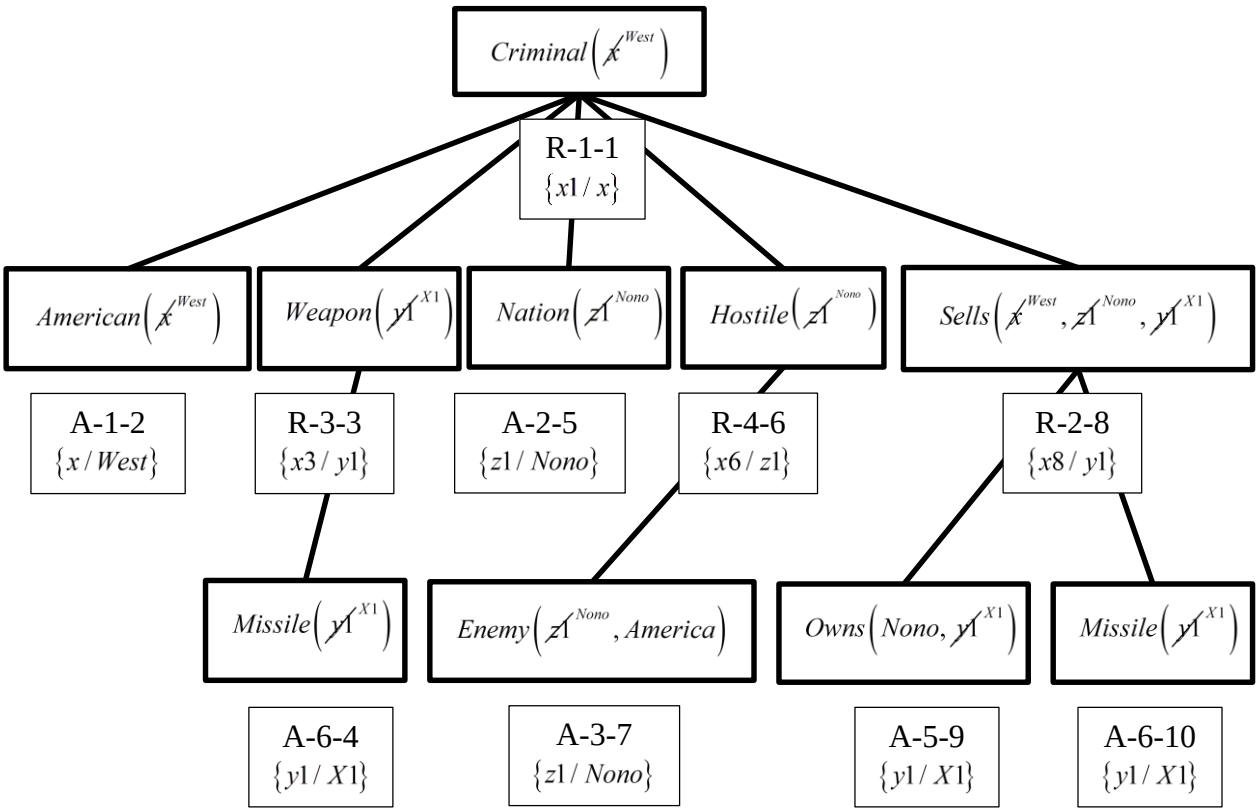
Rules:

1) $American(x) \wedge Weapon(y) \wedge Nation(z) \wedge Hostile(z) \wedge Sells(x, y, z) \Rightarrow Criminal(x)$
2) $Owns(Nono, x) \wedge Missile(x) \Rightarrow Sells(West, Nono, x)$
3) $Missile(x) \Rightarrow Weapon(x)$
4) $Enemy(x, America) \Rightarrow Hostile(x)$

Knowledge Base:

1) $American(West)$
2) $Nation(Nono)$
3) $Enemy(Nono, America)$
4) $Nation(America)$
5) $Owns(Nono, X1)$
6) $Missile(X1)$

Simulate BACKWARD-CHAINING-WITH-DIAGRAM($Criminal(x)$)



When you instantiate a rule, then for each variable used in the rule, you need to use a name that doesn't already exist in the knowledge base. For simplicity, we just use the variable names used in the rule and append the current value of the subscript counter, N. Note: As you instantiate the rules, the these names (i.e. $x1$, $y5$, $z78$, …) are temporarily added to the knowledge base until that line of reasoning/instantiation is complete.

For example, if we have $Hostile(z1)$, our subscript counter is currently at $N = 6$, and the following rule exists:

$$Enemy(x, America) \Rightarrow Hostile(x)$$

Then we can instantiate that rule ... but we need to use a name for the x that doesn't already exist in the knowledge base ... so we choose $x6$ … because x was used in the rule and our subscript counter was $N = 6$ at that point.

Now, you can think of $Enemy(x6, America) \Rightarrow Hostile(x6)$ as now existing in the knowledge base and our $Hostile(z1)$ has something to bind with during backward chaining (using $\{x6 / z1\}$) so that we can come up with $Enemy(z1, America)$.

2. Backward Chaining (200 Points). Consider the following set of rules and knowledge base that represents parsing sentences and various sentence fragments from a sentence represented as a list of words:

Rules:

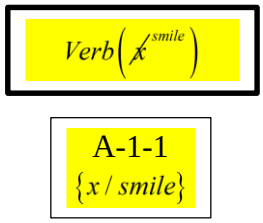
1) $Parse(cons(NounPhrase, cons(VerbPhrase, nil))) \Rightarrow Parse(cons(Sentence, nil))$
2) $Parse(cons(x, y)) \wedge Noun(x) \Rightarrow Parse(cons(NounPhrase, y))$
3) $Parse(cons(x, cons(y, z))) \wedge Verb(y) \Rightarrow Parse(cons(x, cons(VerbPhrase, z)))$

Knowledge Base:

1) $Verb(smile)$
2) $Noun(men)$
3) $Parse(cons(men, cons(smile, nil)))$

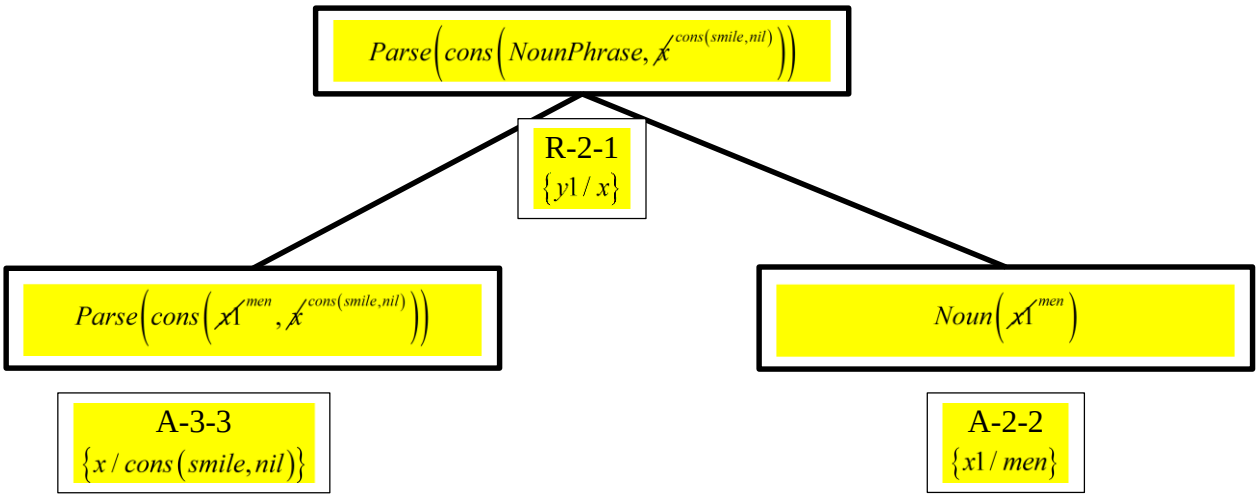
Note: Only x, y, and z are variables.

a. (50 Points) Simulate BACKWARD-CHAINING-WITH-DIAGRAM($Verb(x)$)



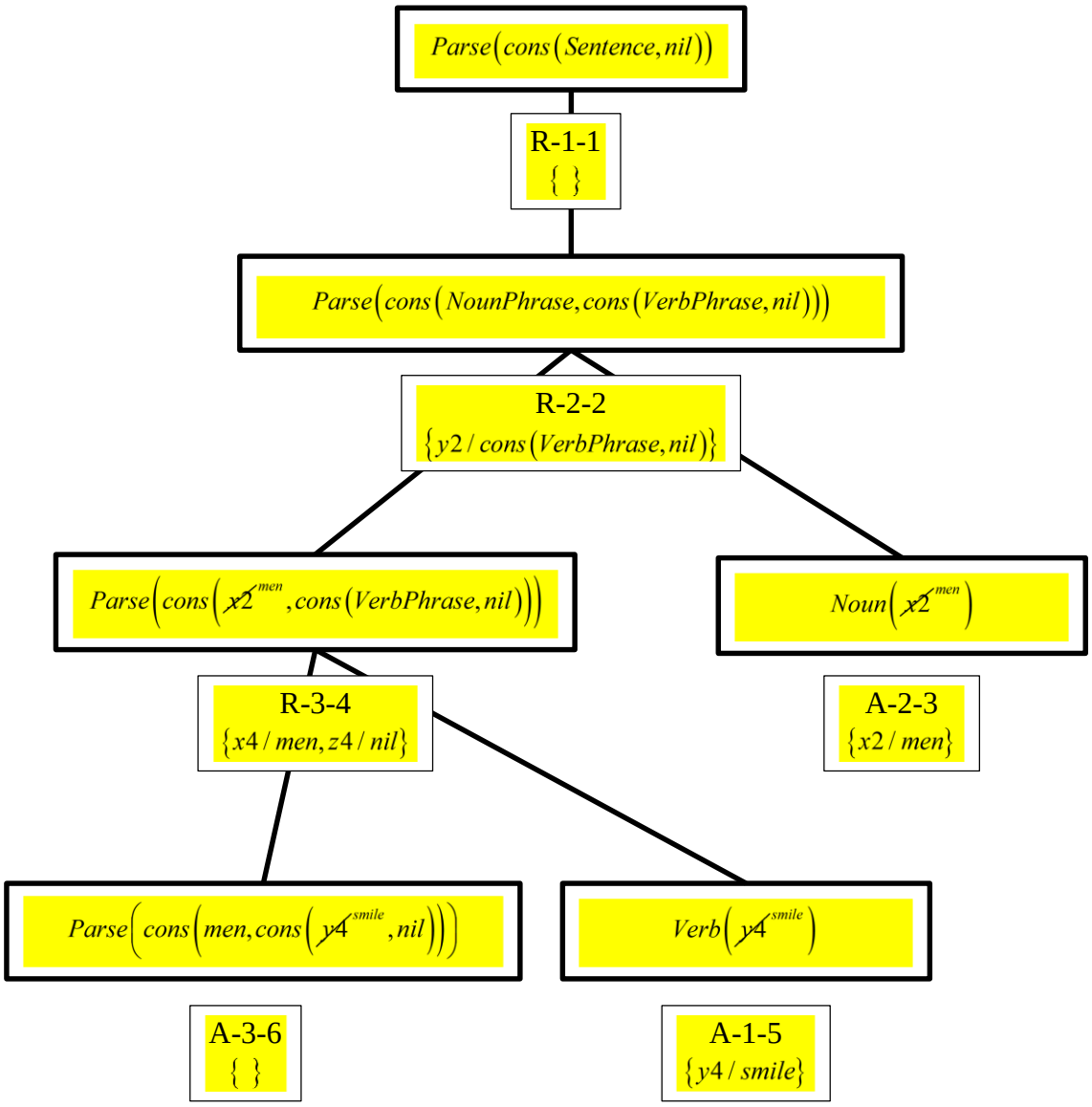
b. (50 Points) Simulate the following:

BACKWARD-CHAINING-WITH-DIAGRAM($Parse(cons(NounPhrase, x))$)



c. (100 Points) Simulate the following:

BACKWARD-CHAINING-WITH-DIAGRAM($Parse(cons(Sentence, nil))$)



3. Backward Chaining (150 Points). Consider the following rules and Knowledge Base describing Integers and Rationals:

Rules:

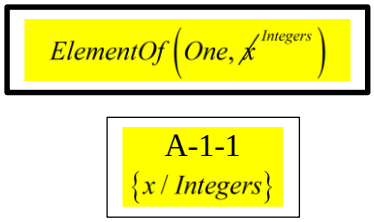
1) $ElementOf(x, y) \Rightarrow ElementOf(x, union(y, z))$
2) $ElementOf(x, y) \wedge Equal(y, z) \Rightarrow ElementOf(x, z)$
3) $Equal(union(x, y), union(y, x))$

Knowledge Base:

1) $ElementOf(One, Integers)$
2) $ElementOf(Two, Rationals)$

Note: Only x, y, and z are variables.

a. (50 Points) Simulate BACKWARD-CHAINING-WITH-DIAGRAM($ElementOf(One, x)$)

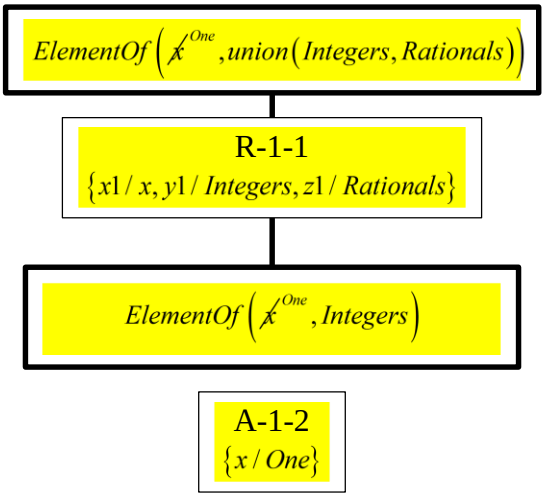


b. (100 Points) Simulate the following:

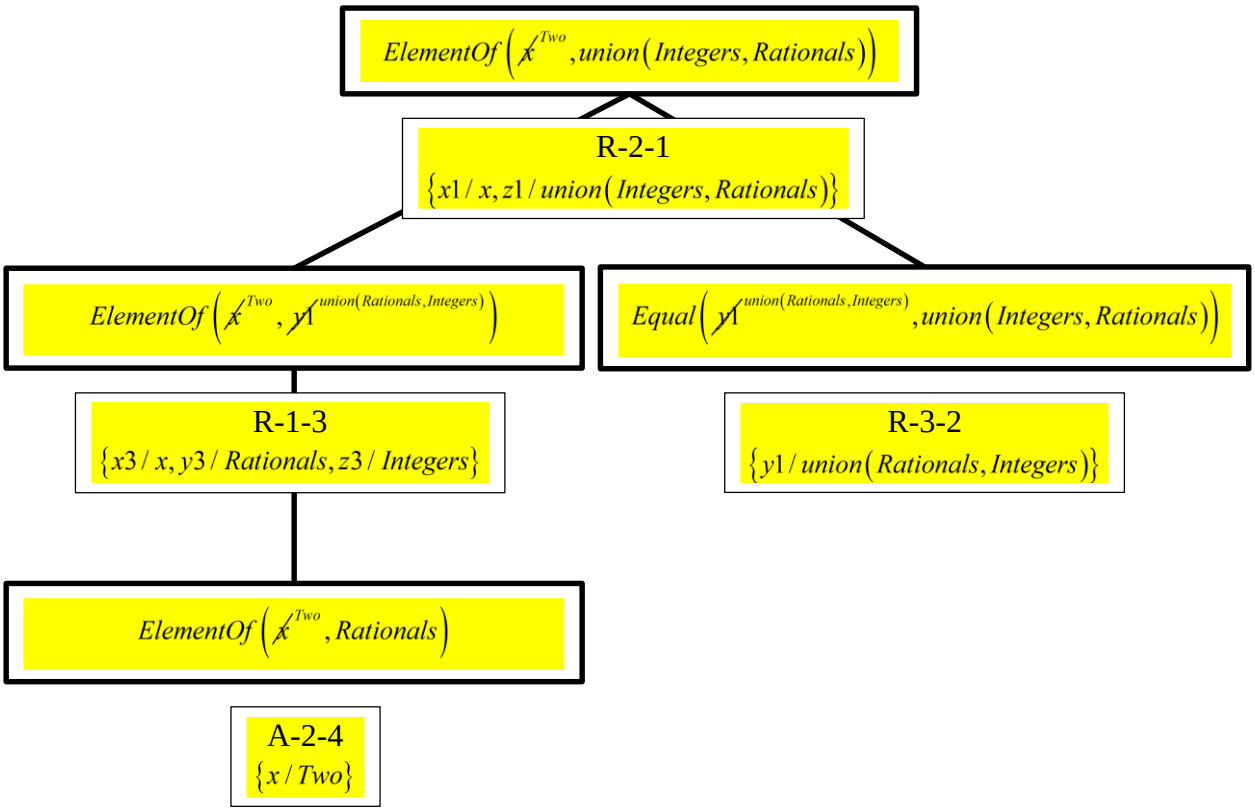
BACKWARD-CHAINING-WITH-DIAGRAM($ElementOf(x, union(Integers, Rationals))$)

Draw one diagram per valid binding of x.

The first of two valid bindings of x:



The second of two valid bindings of x:



4. Research Project (50 Points).

a. Write no more than five sentences summarizing one of the five (or more) references you are using in your research project. (25 Points)

b. Write no more than five sentences summarizing another one of the five (or more) references you are using in your research project. (25 Points)

This assignment has no programming problems.

After completing Assignment 07, create an `assignment_07_`*`lastname`*`.pdf` file for your written assignment.

Upload your `assignment_07_`*`lastname`*`.pdf` file for your written assignment to the Assignment 07 location on the BlackBoard site: https://campus.fsu.edu.