CS161 SPR '13 **HW-3** **due**: on CourseWeb before midnight, Wednesday, May 22

You will write a Lisp function, UNIFRAME, which is a *variant* on the unification function in your textbook (see Fig. 9.1, p. 328). UNIFRAME will unify frames (instead of relations and functions). In your textbook, for any two structures to unify, they must have an identical number of arguments and the order of the arguments matters. In contrast, with frames the order of the arguments does not matter (because each filler is named by a slot) and for UNIFRAME the second frame can have more slots than the first frame and frame unification can still succeed.

1. (UNIFRAME XX YY feta) UNIFRAME unifies **XX** with **YY** and returns a bindings-list **feta**. **XX** and **YY** can each be: 1. a list of frames, 2. a single frame, or 3. a variable.

Let us consider the core case, namely, when **XX** and **YY** are each a single frame – let us call them **frm1** and **frm2**.

UNIFRAME will recursively go through each slot in **frm1** and look for a corresponding slot in **frm2**. If a corresponding slot is found and the fillers are recursively unified in a successful manner, then **UNIFRAME** will return **feta** (i.e., whatever bindings **feta** already had, along with those added during the unification process).

Note-1: Since **UNIFRAME** looks only at **frm1** for slots to be unified with corresponding slots in **frm2**, it is possible that **frm2** could have <u>additional</u> slots (that do not appear in **frm1**) and the unification would still be successful. So UNIFRAME is seeking to unify only that subset of the slots in frm2 that are found in frm1.

Note-2: UNIFRAME will <u>not</u> implement the OCCUR-CHECK? function.

The UNIFY in your textbook calls on a helper function UNIFY-VAR. UNIFRAME should also call on a similar helper function (except it should skip the call to OCCUR-CHECK?). Name this function UNIVAR.

In your textbook, theta (θ) is a ***set*** of bindings, but since you are using Lisp, **feta** will be a ***list*** of bindings. **feta** will take the form: (T (atom binding) ... (atom binding)), (T) or NIL.

(T) indicates successful unification in the case in which there were no variables

and so there are no bindings to consider. NIL indicates that UNIFRAME failed to unify. Recall from HW #1 that *variables* in frames are indicated as (V atom). The atoms in **feta** will come from *variables* appearing in the arguments **XX** and **YY**.

Note (as specified in HW#1) that frame *variables* appear in a frame only as the *filler* of a slot.

Here are some cases on which to test your UNIFRAME function:

Test-case-1: (variables are underlined below to make it easier to spot them)

FRM-1 =
(MTRANS
(**AGENT** (HUMAN
(**GENDER** (FEMALE))
(**F-NAME** (V NAME1))
(**L-NAME** (GRENTZ)))))
FRM-2 =
(MTRANS
(**AGENT** (HUMAN
(**F-NAME** (SALLY))
(**GENDER** (V GEND1))
(**L-NAME** (GRENTZ))))
(**RECIP** (HUMAN
(**GENDER** (MALE))
(**REL** (SON
(**F-NAME** (FREDDY))
(**OF** (HUMAN (**GENDER** (FEMALE))
(**F-NAME** (SALLY))
(**L-NAME** (GRENTZ))))))
(**AGE** (TEEN))
(**L-NAME** (V L-NAME2)))))

(UNIFRAME FRM-1 FRM-2 '(T)) returns two bindings:
(T (NAME1 (SALLY)) (GEND1 (FEMALE)))

Notice several things about test-case-1:

- UNIFRAME succeeded even though FRM-2 has an extra slot at the top level and the order of **GENDER** and **F-NAME** are different.
- The variable L-NAME2 was ignored because it appeared within the additional slot of **RECIP** in the second argument to UNIFRAME

Instead, if we reverse the arguments:

(UNIFRAME FRM-2 FRM-1 ‘(T)) returns: NIL

because FRM-2 has a **RECIP** slot in the *first* argument to UNIFRAME and that slot fails to also appear in the second argument to UNIFRAME.

Test-case-2:

FRM-3 =
(MTRANS
    (**AGENT** (HUMAN
        (**F-NAME** (SALLY))
        (**GENDER** (V GEND1))
        (**L-NAME** (GRENTZ))))
    (**RECIP** (HUMAN
        (**GENDER** (MALE))
        (**REL** (SON (**F-NAME** (FREDDY))
            (**OF** (HUMAN (**GENDER** (FEMALE))
                (**F-NAME** (SALLY))
                (**L-NAME** (GRENTZ)))))))))

FRM-4 =
(MTRANS
    (**AGENT** (HUMAN
        (**GENDER** (FEMALE))
        (**F-NAME** (SALLY))
        (**L-NAME** (V L-NAME1))))
    (**RECIP** (HUMAN
        (**GENDER** (MALE))
        (**REL** (SON (**OF** (V OF-1)))))))

(UNIFRAME FRM-4 FRM-3 ‘(T)) returns 3 bindings:
(T (L-NAME1 (GRENTZ))
  (GEND1 (FEMALE))
  (OF-1 (HUMAN (**GENDER** (FEMALE))
    (**F-NAME** (SALLY))
    (**L-NAME** (GRENTZ)))))

(UNIFRAME FRM-3 FRM-4 ‘(T)) returns NIL

because FRM-3 has an additional slot **F-NAME** (under SON) that FRM-4 does

not have.

Now we will consider the case in which *lists* of frames are given to UNIFRAME. UNIFRAME will go through the first list **XX** and attempt to unify each frame in **XX** with a frame in **YY**. If a frame in **XX** cannot unify with any frame in **YY**, then UNIFRAME fails and returns NIL. Each frame in **YY** may be unified with at most one frame in **XX**. So there can be more frames in **YY** than in **XX**, but not vice versa. Below, FR-L1 has 2 frames while FR-L2 has 3 frames.

```
FR-L1 =
(  (MTRANS (AGENT (V X1))
           (RECIP (V Y1))
           (OBJECT (V OBJ1)))
   (BELIEVE (AGENT (V Y1))
            (OBJECT (IS-LIAR (AGENT (V X1)))))
)

FR-L2 =
(  (BELIEVE (AGENT (HUMAN (F-NAME (SALLY))
                          (GENDER (FEMALE))))
            (OBJECT (IS-LIAR (AGENT (HUMAN (F-NAME (HANK))
                                           (GENDER (MALE)))))))
   (STEALS (AGENT (HUMAN (F-NAME (HANK))
                         (GENDER (MALE))))
           (OBJECT (VEHICLE)))
   (MTRANS (AGENT (HUMAN (F-NAME (HANK))
                         (GENDER (MALE))))
           (OBJECT (BUY (AGENT (HUMAN (F-NAME (HANK))
                                      (GENDER (MALE))))
                        (OBJECT (DONUTS))
                        (TIME (FUTURE))))
           (RECIP (HUMAN (F-NAME (SALLY))
                         (GENDER (FEMALE)))))
   )
```

(UNIFRAME FR-L1 FR-L2 ‘(T)) returns 3 bindings:

```
(T   (OBJ1 ((BUY (AGENT (HUMAN (F-NAME (HANK))
                               (GENDER (MALE))))
                 (OBJECT (DONUTS))
```

(**TIME** (FUTURE))))
(Y1 (HUMAN (**F-NAME** (SALLY)) (**GENDER** (FEMALE))))
(X1 (HUMAN (**F-NAME** (HANK)) (**GENDER** (MALE))))

(UNIFRAME FR-L2 FR-L1 '(T)) returns: NIL
because FR-L2 will look for a STEALS frame in FR-L1 and not find it.

Test-case-3:

FR-L3 =
```
(  (BELIEVE (AGENT (HUMAN (F-NAME (SALLY))
                                  (GENDER (FEMALE))))
            (OBJECT (IS-LIAR (AGENT (HUMAN (F-NAME (HANK))
                                                (GENDER (MALE)))))))
   (STEALS (AGENT (HUMAN (F-NAME (HANK))
                                  (GENDER (MALE))))
            (OBJECT (VEHICLE)))
   (MTRANS (AGENT (HUMAN (F-NAME (FREDDY))
                                  (GENDER (MALE))))
            (OBJECT (BUY (AGENT (HUMAN (F-NAME (FREDDY))
                                                (GENDER (MALE))))
                                  (OBJECT (DONUTS))
                                  (TIME (FUTURE))))
            (RECIP (HUMAN (F-NAME (SALLY))
                                  (GENDER (FEMALE)))))
   )
```

(UNIFRAME FR-L1 FR-L3 '(T)) returns: NIL

because X1 initially binds to Freddy and in FR-L3 Sally believes that ***Hank*** is a liar and since Freddy /= Hank, the unification fails to go through.

Once you have UNIFRAME working, it is time to implement FFINFER, which will do a simple form of forward-chaining inference with facts and rules.

2. (FFINFER FACTS RULES NEW) – **FACTS** will hold a list of instantiated frames (no variables). **RULES** will hold a list of rules. Each rule will contain one or more *premises* and a single *conclusion*. Both premises and conclusions will be frames that may also contain variables. **FFINFER** will behave in a way similar (but not identical) to FOL-FC-ASK in your textbook (see p. 332, Fig. 9.3).

FFINFER will take each rule in **RULES** and do the following: a. rename the variables in that rule using **STANDARDIZE-VVARS** (see below). b. attempt to unify (using UNIFRAME) all the premises with facts in FACTS. If they unify and produce a feta, then **UNIFRAME** will use **SUBST-FETA** (see below) to substitute those bindings into the rule's conclusion and add that conclusion to **NEW**. Once all rules have been attempted against all facts, if **NEW** is non-NIL, then **FFINFER** will remove the new facts from **NEW**, add them to the front of **FACTS,** and call itself recursively. **FFINFER** will halt when no new facts are produced and return all new facts that were inferred.

Here are your utility functions:

3. (STANDARDIZE-VVARS RULE) -- You should now know how to create new names, so define a utility function called STANDARDIZE-VVARS that takes a RULE, finds each variable *var*, and renames it to *var-N* (some unique number).

Rules will be of the form:
(premise-1 … premise-n ===> conclusion)

Here is an example rule:

```
RULE-1 =
(  (MTRANS (AGENT (V X1))
           (RECIP (V Y1))
           (OBJECT (V OBJ1)))
   (BELIEVE (AGENT (V Y1))
            (OBJECT (IS-LIAR (AGENT (V X1)))))
   ===> (DOUBTS (AGENT (V Y1))
                (OBJECT (V OBJ1)))
)
```

This rule states:
IF X communicates a concept OBJ1 to Y **and** Y believes that X is a liar,
THEN Y will doubt that OBJ1 is true.

(STANDARDIZE-VVARS RULE-1) returns:

```
(  (MTRANS (AGENT (V X1-1))
           (RECIP (V Y1-1))
           (OBJECT (V OBJ1-1)))
   (BELIEVE (AGENT (V Y1-1))
            (OBJECT (IS-LIAR (AGENT (V X1-1)))))
```

```
   ===> (DOUBTS (AGENT (V Y1-1))
                (OBJECT (V OBJ1-1)))
)
```

A second call: (STANDARDIZE-VVARS RULE-1) will return:

```
(  (MTRANS (AGENT (V X1-2))
           (RECIP (V Y1-2))
           (OBJECT (V OBJ1-2)))
   (BELIEVE (AGENT (V Y1-2))
            (OBJECT (IS-LIAR (AGENT (V X1-2)))))
   ===> (DOUBTS (AGENT (V Y1-2))
                (OBJECT (V OBJ1-2)))
)
```

4. (SUBST-FETA CONCLU FETA) – SUBST-FETA takes a conclusion frame **CONCLU** and if it has any variables, SUBST-FETA returns a conclusion with those variables each replaced by its bindings in FETA.

Here is an example:

```
(SUBST-FETA '(DOUBTS (AGENT (V Y1))
                     (OBJECT (V OBJ1)))
             ' (T (Y1 (HUMAN (GENDER (FEMALE))
                             (F-NAME (SALLY))))
                  (X1 (HUMAN (GENDER (MALE))
                             (F-NAME (HANK))))
                  (OBJ1 (BUY (AGENT (HUMAN
                                      (F-NAME (HANK))
                                      (GENDER (MALE))))
                             (OBJECT (DONUTS))
                             (TIME (FUTURE)))))  )
```

returns:

```
(DOUBTS (AGENT (HUMAN (GENDER (FEMALE))
                      (F-NAME (SALLY))))
        (OBJECT (BUY (AGENT (HUMAN
                              (F-NAME (HANK))
                              (GENDER (MALE))))
                     (OBJECT (DONUTS))
                     (TIME (FUTURE)))))
```

Now you are ready to try out FFINFER and see if it is doing forward chaining:

Recall the story:

Sally warned her teenage son Freddy about becoming friends with Hank because he had stolen a car. Hank asked Freddy to drive him to the Seven Eleven for some donuts. Freddy stayed in the car. Suddenly, Hank ran out holding cash and a gun. He jumped into Freddy's car. The owner Harold Smythe came out mad as hell. Freddy drove away but Smythe wrote down his plate. The police arrested Freddy for armed robbery.

We will *assume* that a conceptual lexicon (with frames and associated demons) already magically exists; that these demons magically fired for the first half of the story and that they created the following (greatly simplified) facts in some episodic memory:

```
STORY-FACTS-1 =
(
  (TELL (AGENT (SALLY))
        (RECIP (FREDDY))
        (OBJECT (C-CAUSE (AGENT (HANK))
                         (OBJECT (GOAL-FAIL-FOR
                                   (AGENT (FREDDY)))))))

  (TELL (AGENT (HANK))
        (RECIP (FREDDY))
        (OBJECT (WANT (AGENT (HANK))
                      (OBJECT (DONUTS)))))


 (ASK (AGENT (HANK))
      (RECIP (FREDDY))
      (OBJECT (PTRANS (AGENT (FREDDY))
                      (OBJECT (HANK))
                      (DESTIN (SEVN-ELVN))
                      (INSTRU (CAR2)))))
 (VEHICLE (IS (CAR1)))
 (VEHICLE (IS (CAR2)))
 (SNACK (IS (DONUTS)))
```

```
  (IN (AGENT (FREDDY))
      (OBJECT (CAR2)))
  (STEAL (OBJECT (CAR1))
         (AGENT (HANK)))
  (AT (AGENT (HANK))
      (LOC (SEVN-ELVN)))
  (HOLD (AGENT (HANK))
        (OBJECT (GUN)))
)
```

Recall the rules that appeared in your mid-term. We are going to use these to test out your frame-forward-chaining inference function FFINFER.

$\forall$ x, y, o TELL(x, y, o) $\Rightarrow$ KNOW(y, o),
$\forall$ x, y, o STEAL(x, o) & VALUABLE(o) $\Rightarrow$ CRIMINAL(x),
$\forall$ x VEHICLE(x) $\Rightarrow$ VALUABLE(x),
$\forall$ x, y, estb, z ASK(x, y, PTRANS(y, x, estb, z)) & VEHICLE(z) & IN(y, z)
$\Rightarrow$ COMPLY-REQ(y, x, PTRANS(y, x, estb, z)),
$\forall$ x, y, estb, z, o COMPLY-REQ(x, y, PTRANS(x, y, estb, z))
& KNOW(x, WANT(y, o)) & SNACK(o) $\Rightarrow$ BELIEVE(x, WILL-BUY(y, o, estb))
$\forall$ x, y, z AT(x, loc) & HOLD(x, GUN) & CRIMINAL(x)
$\Rightarrow$ ROB(x, loc)
$\forall$ x, y, z BELIEVE(x, WILL-BUY(y, o, estb)) & ROB(y, estb) $\Rightarrow$ SURPRISED(x)

Below they are turned into frame-related rules:

```
RULE-LIST-1 =
(    ( (TELL (AGENT (V AG))
             (OBJECT (V OBJ))
             (RECIP (V RC)))    ===>  (KNOW (AGENT (V RC))
                                            (OBJECT (V OBJ))) )
       ( (STEAL (AGENT (V AG))
                (OBJECT (V OBJ)))
         (VALUABLE (IS (V OBJ)))  ===> (CRIMINAL (IS (V AG)))  )

       (  (VEHICLE (IS (V OBJ)))  ===>  (VALUABLE (IS (V OBJ)))  )

       ( (ASK (AGENT (V AG))
              (OBJECT (PTRANS (AGENT (V RC))
                              (OBJECT (V AG))
                              (DESTIN (V ESTB))
                              (INSTRU (V VH))))
```

```
            (RECIP (V RC)))
        (VEHICLE (IS (V VH)))
        (IN (AGENT (V RC))
            (OBJECT (V VH)))
        ===>  (COMPLY-REQ (AGENT (V RC))
                          (FROM (V AG))
                          (OBJECT (PTRANS (AGENT (V RC)
                                                 (OBJECT (V AG))
                                                 (DESTIN (V ESTB))
                                                 (INSTRU (V VH))))))  )

   (  (COMPLY-REQ (AGENT (V X))
                  (FROM (V Y))
                  (OBJECT (PTRANS (AGENT (V X))
                                  (OBJECT (V Y))
                                  (DESTIN (V  Z))
                                  (INSTRU (V VH)))))
     (KNOW (AGENT (V X))
           (OBJECT (WANT (AGENT (V Y))
                         (OBJECT (V O)))))
     (SNACK (IS (V O))) ===>  (BELIEVE
                                 (AGENT (V X))
                                 (OBJECT (WILL-BUY (AGENT (V Y))
                                                   (OBJECT (V O))
                                                   (FROM (V  Z)))))  )

  (  (AT (AGENT (V X))
         (LOC  (V LC)))

     (HOLD (AGENT (V X))
           (OBJECT (GUN)))

     (CRIMINAL (IS (V X)))  ===>   (ROB (AGENT (V X))
                                        (OBJECT (V LC)))   )
  (  (BELIEVE (AGENT (V AG1))
              (OBJECT (WILL-BUY (AGENT (V  AG2))
                                (OBJECT (V  THING))
                                (FROM (V ESTB)))))
     (ROB (AGENT (V AG2))
          (OBJECT (V ESTB)))  ===>  (SURPRISED (AGENT (V AG1)))  )
)
```

(FFINFER STORY-FACTS-1 RULE-LIST-1 NIL) should return, in a list, the following 9 new facts (not necessarily in this order):

(KNOW (**AGENT** (FREDDY))
(**OBJECT** (C-CAUSE (**AGENT** (HANK))
(**OBJECT** (GOAL-FAIL-FOR
(**AGENT** (FREDDY)))))))
(KNOW (**AGENT** (FREDDY))
(**OBJECT** (WANT (**AGENT** (HANK))
(**OBJECT** (DONUTS)))))

(VALUABLE (**IS** (CAR1)))
(VALUABLE (**IS** (CAR2)))

(CRIMINAL (**IS** (HANK)))

(COMPLY-REQ (**AGENT** (FREDDY))
(**FROM** (HANK))
(**OBJECT** (PTRANS (**AGENT** (FREDDY))
(**OBJECT** (HANK))
(**DESTIN** (SEVN-ELVN))
(**INSTRU** (CAR2)))))

(BELIEVE
(**AGENT** (FREDDY))
(**OBJECT** (WILL-BUY (**AGENT** (HANK))
(**OBJECT** (DONUTS))
(**FROM** (SEVN-ELVN)))))

(ROB (**AGENT** (HANK))
(**OBJECT** (SEVN-ELVN)))

(SURPRISED (**AGENT** (FREDDY)))