6.863 Final Project: A Hierarchical System for Quantifying Who Ate our Breakfast

Grant Emery Sam Hartman Brian Perrin

May 20, 1999

This project extends the semantic interpreter presented in lab 3 in two ways First, it adds a notion of hierarchy: some nouns can be subsets of other nouns. Second, it adds a model of quantification and adds universal/existential quantifiers.

1 Changes to Semantics Handling

The semantic interpreter attempted to represent most phrases as lambda closures, with the exception of names. The main construction of the semantic frame was focused toward the procedure implementing the verb. The problem with this design is that it was too flexible: a change to how the frame worked involved a change to every verb.

In addition, components in the frame were atoms (usually symbols) representing words. This structure was insufficient for our needs; we needed to describe noun phrases based on their properties like the quantifiers associated with them.

Thus, the rules to compose phrases together have been moved out of the definition of the verbs and into the semantic rules associated with the phrases.

In addition, the frame-type object (a tagged property list) is used as a generic data structure. The frame type describes which phrase is being described, and the tags describe the phrase.

The semantic interpretation phase still flattens the sentence. As the phrases are combined into the overall thematic frame, some properties are propagated into the frame. For example, the set of quantifiers in all the subframes are aggregated together into a global quantifier list. Noun phrases do retain some structure in the final thematic frame.

2 Model of hierarchy

Each noun is stored as a frame with a property. In addition, nouns have two properties for hierarchy: and . The property contains a list of nouns that are subsets of the current noun. For example, if John is a person, then John will be in the subset tag of person. The superset tags are also maintained.

This hierarchical linking is important for quantification. For example, if the system needs to quantify a particular predicate for all people, it can look up the noun person and find out what subsets of person exist.

The hierarchy also helps to answer yes/no questions. For example if people like cars and john is a subset of people, then john must like cars. (The hierarchy is strict; a noun can only be a subset if every statement that is true about the superset will be true of the noun.)

3 Model for Quantification

Two types of quantifiers were implemented: universal and existential quantifiers. These quantifiers all apply to a set of objects. For example in the phrase 'a person', the existential quantifier 'a' applies to the subsets of person.

Noun phrases that are quantified have a tag that is set to be a monotonically increasing integer. This allows the code implementing quantification to only look at the global list of all quantifiers stored in the top of the sentence frame. Then, later, the noun phrase in which the quantifier was introduced can be found based on the hole.

Unfortunately, we were not able to come up with a general model of how to deal with quantifier information in both statements and questions. Thus, these sub-problems were handled separately.

3.1 Statements

Universal quantifiers in statements are basically no-operations. If I say "Every person likes Mary," then the statement will be true for any subset of person, so the statement can be associated directly with person.

Existential quantifiers work by creating new nameless nouns that are subsets of the noun being quantified. For example, if I enter "Mary placed a book on the table," then a new subset of booke is created, and a fact frame is added to the database stating that this object was placed on the table by Mary.

This is not quite right. statement quantification does not currently deal with quantifier ordering. In particular, if I say something like"A person ate each pancake," I do not mean that there exists a single person responsible for the consumption of every pancake. However, the empty placeholder noun that is a subset of some real concept appears to be a powerful modeling tool. One extention of this project would be to explore ways of binding particular nameless placeholders together; this could be combined with pronoun binding to allow a discourse system to understand when two instantiations of a common noun are the same and to deduce properties from statements made about the common noun.

3.2 Questions

In questions, all the quantifiers are collected together and sorted based on their ordering. Natural order is preserved in most cases, but quantifiers like each migrate to the outside. The current mechanism does not support c-command because the sentences it deals with are not complex enough for this to be important, although adding grouping to the quantifier ordering so that particular quantifiers could not be internally re-arranged would not be too difficult.

Then, the question logic attempts to fill in the hole left by the quantifier with each of the possible subsets of the noun being quantified.

The question logic then tries to answer the question, possibly dealing with more quantifiers and holes.

Whenever correct answers are found, the quantifier specifies how they are combined. For example in a yes/no question, a universal quantifier requires that a particular value for a hole satisfy all inner substitutions.

Who-style questions are handled by designation of a particular hole as the print-hole. All values that can plug into this hole are presented as answers to the question.

4 Implementation Information

Several files are provided in `/afs/sipb.mit.edu/user/hartmans/6.863/lab5` that contain information about the implementation. These files can be accessed through the `www.mit.edu` web server if AFS is not available.

assumed-procedures Descriptions of all utility procedures added during the course of the project

frame-types A list of all frame types used in the project along with descriptions of how they are used and lists of valid tags for those frames

frame-tags A list of all tags that are used, their meanings and type

In addition, the files implementing the Lisp code can be found in this directory. If all the lisp files are copied into a directory, `semantics.lisp` will initialize the system.

5 Sample Interaction

```
(top-level)
Shall I clear the database?y
>john ate a pancake
 Syntax: (ROOT
          (S (NP (NP-PRO (NAME JOHN)))
             (VP
              (V+ARGS (V2+TNS ATE)
               (NP (NP-PRO (DET A) (AP?) (N PANCAKE)))))))
Semantics: (S :QUANTS
              ((QINFO :QORDER 100 :QPROCF
                #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                     #x850144a>
                :QPROCQ
                #2=#<Interpreted Closure (:INTERNAL MAKE-QPROCQ) @
                     #x8501582>
                :HOLE 126))
              :WORD ATE :ID 2868 :RELATION EAT :TENSE PAST :PATIENT
              (NP :HOLE 126 :QUANTS
               ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2# :HOLE 126)))
              :AGENT (NP :WORD JOHN :ID 2780))
OK.
>did john eat a pancake
 Syntax: (ROOT
          (Q-WH (AUX%DO%MODAL (AUX%DO+TNS DID))
           (NP (NP-PRO (NAME JOHN)))
           (VBAR-TNS (V2-TNS EAT)
            (NP (NP-PRO (DET A) (AP?) (N PANCAKE))))))
Semantics: Question: (Q :QUANTS
                      ((QINFO :QORDER 100 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x84f2eba>
                        :QPROCQ
                        #2=#<Interpreted Closure (:INTERNAL
```

```
                                                 MAKE-QPROCQ)
                           @ #x84f2ff2>
                      :HOLE 127))
                    :WORD EAT :ID 2867 :RELATION EAT :PATIENT
                    (NP :HOLE 127 :QUANTS
                     ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                       :HOLE 127)))
                    :AGENT (NP :WORD JOHN :ID 2780) :TENSE PAST)
finished answering question (T)
yes.
>did mary eat a pancake
 Syntax: (ROOT
          (Q-WH (AUX%DO%MODAL (AUX%DO+TNS DID))
           (NP (NP-PRO (NAME MARY)))
           (VBAR-TNS (V2-TNS EAT)
            (NP (NP-PRO (DET A) (AP?) (N PANCAKE))))))
Semantics: Question: (Q :QUANTS
                      ((QINFO :QORDER 100 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x8613762>
                        :QPROCQ
                        #2=#<Interpreted Closure (:INTERNAL
                                                 MAKE-QPROCQ)
                             @ #x861389a>
                        :HOLE 128))
                      :WORD EAT :ID 2867 :RELATION EAT :PATIENT
                      (NP :HOLE 128 :QUANTS
                       ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                         :HOLE 128)))
                      :AGENT (NP :WORD MARY :ID 2781) :TENSE PAST)
finished answering question (NIL)
i don't think so.
>did john eat every pancake
 Syntax: (ROOT
          (Q-WH (AUX%DO%MODAL (AUX%DO+TNS DID))
           (NP (NP-PRO (NAME JOHN)))
           (VBAR-TNS (V2-TNS EAT)
            (NP (NP-PRO (DET EVERY) (AP?) (N PANCAKE))))))
Semantics: Question: (Q :QUANTS
                      ((QINFO :QORDER 100 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x8605202>
                        :QPROCQ
                        #2=#<Interpreted Closure (:INTERNAL
                                                 MAKE-QPROCQ)
```

```
                          @ #x860535a>
                     :HOLE 129))
                   :WORD EAT :ID 2867 :RELATION EAT :PATIENT
                   (NP :HOLE 129 :QUANTS
                    ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                      :HOLE 129)))
                   :AGENT (NP :WORD JOHN :ID 2780) :TENSE PAST)
 0: (LIST-AND (T))
   1: (LIST-AND NIL)
   1: returned T
 0: returned T
finished answering question (T)
yes.
>mary ate a pancake
 Syntax: (ROOT
          (S (NP (NP-PRO (NAME MARY)))
             (VP
              (V+ARGS (V2+TNS ATE)
               (NP (NP-PRO (DET A) (AP?) (N PANCAKE)))))))
Semantics: (S :QUANTS
              ((QINFO :QORDER 100 :QPROCF
                #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                     #x84f92e2>
                :QPROCQ
                #2=#<Interpreted Closure (:INTERNAL MAKE-QPROCQ) @
                     #x84f941a>
                :HOLE 130))
              :WORD ATE :ID 2868 :RELATION EAT :TENSE PAST :PATIENT
              (NP :HOLE 130 :QUANTS
               ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2# :HOLE 130)))
              :AGENT (NP :WORD MARY :ID 2781))
OK.
>did john eat a pancake
 Syntax: (ROOT
          (Q-WH (AUX%DO%MODAL (AUX%DO+TNS DID))
           (NP (NP-PRO (NAME JOHN)))
           (VBAR-TNS (V2-TNS EAT)
            (NP (NP-PRO (DET A) (AP?) (N PANCAKE))))))
Semantics: Question: (Q :QUANTS
                      ((QINFO :QORDER 100 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x861830a>
                        :QPROCQ
                        #2=#<Interpreted Closure (:INTERNAL
                                                   MAKE-QPROCQ)
```

```
                          @ #x8618442>
                     :HOLE 131))
                   :WORD EAT :ID 2867 :RELATION EAT :PATIENT
                   (NP :HOLE 131 :QUANTS
                    ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                      :HOLE 131)))
                   :AGENT (NP :WORD JOHN :ID 2780) :TENSE PAST)
finished answering question (T)
yes.
>did john eat every pancake
 Syntax: (ROOT
          (Q-WH (AUX%DO%MODAL (AUX%DO+TNS DID))
           (NP (NP-PRO (NAME JOHN)))
           (VBAR-TNS (V2-TNS EAT)
            (NP (NP-PRO (DET EVERY) (AP?) (N PANCAKE))))))
Semantics: Question: (Q :QUANTS
                      ((QINFO :QORDER 100 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x860be0a>
                        :QPROCQ
                        #2=#<Interpreted Closure (:INTERNAL
                                                  MAKE-QPROCQ)
                             @ #x860bf62>
                        :HOLE 132))
                      :WORD EAT :ID 2867 :RELATION EAT :PATIENT
                      (NP :HOLE 132 :QUANTS
                       ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                         :HOLE 132)))
                      :AGENT (NP :WORD JOHN :ID 2780) :TENSE PAST)
 0: (LIST-AND (T NIL))
   1: (LIST-AND (NIL))
   1: returned NIL
 0: returned NIL
finished answering question (NIL)
i don't think so.
>who ate a pancake
 Syntax: (ROOT
          (Q+WH (PRONP+WH WHO)
           (VP
            (V+ARGS (V2+TNS ATE)
             (NP (NP-PRO (DET A) (AP?) (N PANCAKE)))))))
Semantics: Question: (Q :PRINT-HOLE -1 :QUANTS
                      ((QINFO :QORDER 100 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x85ff312>
```

```
                         :QPROCQ
                         #2=#<Interpreted Closure (:INTERNAL
                                                   MAKE-QPROCQ)
                              @ #x85f026a>
                         :HOLE 133)
                        (QINFO :QPROCQ
                         #3=#<Interpreted Closure (:INTERNAL
                                                   MAKE-QPROCQ)
                              @ #x849d7c2>
                         :QORDER 100 :PRINT-HOLE -1 :HOLE -1))
                       :WORD ATE :ID 2868 :RELATION EAT :TENSE PAST
                       :PATIENT
                       (NP :HOLE 133 :QUANTS
                        ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                          :HOLE 133)))
                       :AGENT
                       (NP :WORD WHO :ID 2834 :HOLE -1 :PRINT-HOLE -1
                        :QUANTS
                        ((QINFO :QPROCQ #3# :QORDER 100 :PRINT-HOLE -1
                          :HOLE -1))))
finished answering question (NIL)
finished answering question (NIL)
finished answering question (NIL)
finished answering question (NIL)
NIL
>john is a person
 Syntax: (ROOT
          (S (NP (NP-PRO (NAME JOHN)))
             (VP (AUX%BE+TNS IS)
              (NP (NP-PRO (DET A) (AP?) (N PERSON))))))
Semantics: (VP :QUANTS
            ((QINFO :QORDER 100 :QPROCF
              #1=#<Interpreted Closure (:INTERNAL QPROCF) @ #x84f1d52>
              :QPROCQ
              #2=#<Interpreted Closure (:INTERNAL MAKE-QPROCQ) @
                   #x84f1e8a>
              :HOLE 134))
            :WORD IS :ID 2968 :TENSE PRESENT :RELATION BE :PATIENT
            (NP :HOLE 134 :QUANTS
             ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2# :HOLE 134)))
            :AGENT (NP :WORD JOHN :ID 2780))
OK.
>who ate a pancake
 Syntax: (ROOT
          (Q+WH (PRONP+WH WHO)
```

```
           (VP
            (V+ARGS (V2+TNS ATE)
             (NP (NP-PRO (DET A) (AP?) (N PANCAKE)))))))
Semantics: Question: (Q :PRINT-HOLE -1 :QUANTS
                       ((QINFO :QORDER 100 :QPROCF
                         #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                              #x86108e2>
                         :QPROCQ
                         #2=#<Interpreted Closure (:INTERNAL
                                                   MAKE-QPROCQ)
                              @ #x8610a1a>
                         :HOLE 135)
                        (QINFO :QPROCQ
                         #3=#<Interpreted Closure (:INTERNAL
                                                   MAKE-QPROCQ)
                              @ #x849d7c2>
                         :QORDER 100 :PRINT-HOLE -1 :HOLE -1))
                       :WORD ATE :ID 2868 :RELATION EAT :TENSE PAST
                       :PATIENT
                       (NP :HOLE 135 :QUANTS
                        ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                          :HOLE 135)))
                       :AGENT
                       (NP :WORD WHO :ID 2834 :HOLE -1 :PRINT-HOLE -1
                        :QUANTS
                        ((QINFO :QPROCQ #3# :QORDER 100 :PRINT-HOLE -1
                          :HOLE -1))))
finished answering question (NIL)
finished answering question (((JOHN)))
finished answering question (NIL)
finished answering question (((JOHN)))
((JOHN))
>mary is a person
 Syntax: (ROOT
          (S (NP (NP-PRO (NAME MARY)))
             (VP (AUX%BE+TNS IS)
              (NP (NP-PRO (DET A) (AP?) (N PERSON))))))
Semantics: (VP :QUANTS
            ((QINFO :QORDER 100 :QPROCF
              #1=#<Interpreted Closure (:INTERNAL QPROCF) @ #x86095b2>
              :QPROCQ
              #2=#<Interpreted Closure (:INTERNAL MAKE-QPROCQ) @
                   #x86096ea>
              :HOLE 136))
            :WORD IS :ID 2968 :TENSE PRESENT :RELATION BE :PATIENT
```

```
            (NP :HOLE 136 :QUANTS
             ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2# :HOLE 136)))
            :AGENT (NP :WORD MARY :ID 2781))
OK.
>did a person eat each pancake
 Syntax: (ROOT
          (Q-WH (AUX%DO%MODAL (AUX%DO+TNS DID))
           (NP (NP-PRO (DET A) (AP?) (N PERSON)))
           (VBAR-TNS (V2-TNS EAT)
            (NP (NP-PRO (DET EACH) (AP?) (N PANCAKE))))))
Semantics: Question: (Q :QUANTS
                      ((QINFO :QORDER 200 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x84fdf52>
                        :QPROCQ
                        #2=#<Interpreted Closure (:INTERNAL
                                                  MAKE-QPROCQ)
                             @ #x84fe0aa>
                        :HOLE 138)
                       (QINFO :QORDER 100 :QPROCF
                        #3=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x84fd8ba>
                        :QPROCQ
                        #4=#<Interpreted Closure (:INTERNAL
                                                  MAKE-QPROCQ)
                             @ #x84fd9f2>
                        :HOLE 137))
                      :WORD EAT :ID 2867 :RELATION EAT :PATIENT
                      (NP :HOLE 138 :QUANTS
                       ((QINFO :QORDER 200 :QPROCF #1# :QPROCQ #2#
                         :HOLE 138)))
                      :AGENT
                      (NP :HOLE 137 :QUANTS
                       ((QINFO :QORDER 100 :QPROCF #3# :QPROCQ #4#
                         :HOLE 137)))
                      :TENSE PAST)
finished answering question (NIL)
finished answering question (T)
finished answering question (T)
 0: (LIST-AND (T T))
   1: (LIST-AND (T))
     2: (LIST-AND NIL)
     2: returned T
   1: returned T
 0: returned T
```

```
finished answering question (T)
yes.

who ate a pancake
 Syntax: (ROOT
          (Q+WH (PRONP+WH WHO)
           (VP
            (V+ARGS (V2+TNS ATE)
             (NP (NP-PRO (DET A) (AP?) (N PANCAKE)))))))
Semantics: Question: (Q :PRINT-HOLE -1 :QUANTS
                      ((QINFO :QORDER 100 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x8612682>
                        :QPROCQ
                        #2=#<Interpreted Closure (:INTERNAL
                                                  MAKE-QPROCQ)
                             @ #x86127ba>
                        :HOLE 141)
                       (QINFO :QPROCQ
                        #3=#<Interpreted Closure (:INTERNAL
                                                  MAKE-QPROCQ)
                             @ #x849d7c2>
                        :QORDER 100 :PRINT-HOLE -1 :HOLE -1))
                      :WORD ATE :ID 2868 :RELATION EAT :TENSE PAST
                      :PATIENT
                      (NP :HOLE 141 :QUANTS
                       ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                         :HOLE 141)))
                      :AGENT
                      (NP :WORD WHO :ID 2834 :HOLE -1 :PRINT-HOLE -1
                       :QUANTS
                       ((QINFO :QPROCQ #3# :QORDER 100 :PRINT-HOLE -1
                         :HOLE -1))))
finished answering question (NIL)
finished answering question (((JOHN)))
finished answering question (((MARY)))
finished answering question (((JOHN) (MARY)))
((JOHN) (MARY))
>every communist is a person
 Syntax: (ROOT
          (S (NP (NP-PRO (DET EVERY) (AP?) (N COMMUNIST)))
             (VP (AUX%BE+TNS IS)
              (NP (NP-PRO (DET A) (AP?) (N PERSON))))))
Semantics: (VP :QUANTS
            ((QINFO :QORDER 100 :QPROCF
```

```
               #1=#<Interpreted Closure (:INTERNAL QPROCF) @ #x8618ad2>
               :QPROCQ
               #2=#<Interpreted Closure (:INTERNAL MAKE-QPROCQ) @
                    #x8618c0a>
               :HOLE 143)
              (QINFO :QORDER 100 :QPROCF
               #3=#<Interpreted Closure (:INTERNAL QPROCF) @ #x86183fa>
               :QPROCQ
               #4=#<Interpreted Closure (:INTERNAL MAKE-QPROCQ) @
                    #x8618552>
               :HOLE 142))
             :WORD IS :ID 2968 :TENSE PRESENT :RELATION BE :PATIENT
             (NP :HOLE 143 :QUANTS
              ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2# :HOLE 143)))
             :AGENT
             (NP :HOLE 142 :QUANTS
              ((QINFO :QORDER 100 :QPROCF #3# :QPROCQ #4# :HOLE 142))))
OK.
>bill is a communist
 Syntax: (ROOT
          (S (NP (NP-PRO (NAME BILL)))
             (VP (AUX%BE+TNS IS)
              (NP (NP-PRO (DET A) (AP?) (N COMMUNIST))))))
Semantics: (VP :QUANTS
             ((QINFO :QORDER 100 :QPROCF
               #1=#<Interpreted Closure (:INTERNAL QPROCF) @ #x8602852>
               :QPROCQ
               #2=#<Interpreted Closure (:INTERNAL MAKE-QPROCQ) @
                    #x860298a>
               :HOLE 144))
             :WORD IS :ID 2968 :TENSE PRESENT :RELATION BE :PATIENT
             (NP :HOLE 144 :QUANTS
              ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2# :HOLE 144)))
             :AGENT (NP :WORD BILL :ID 2776))
OK.
>bill ate a pancake
 Syntax: (ROOT
          (S (NP (NP-PRO (NAME BILL)))
             (VP
              (V+ARGS (V2+TNS ATE)
               (NP (NP-PRO (DET A) (AP?) (N PANCAKE)))))))
Semantics: (S :QUANTS
              ((QINFO :QORDER 100 :QPROCF
                #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                     #x84f4252>
```

```
                :QPROCQ
                #2=#<Interpreted Closure (:INTERNAL MAKE-QPROCQ) @
                     #x84f438a>
                :HOLE 145))
              :WORD ATE :ID 2868 :RELATION EAT :TENSE PAST :PATIENT
              (NP :HOLE 145 :QUANTS
               ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2# :HOLE 145)))
              :AGENT
              (NP :SUP SUPPRESSED :SUB SUPPRESSED :WORD BILL :ID 2776))
OK.
>who ate a pancake
 Syntax: (ROOT
          (Q+WH (PRONP+WH WHO)
           (VP
            (V+ARGS (V2+TNS ATE)
             (NP (NP-PRO (DET A) (AP?) (N PANCAKE)))))))
Semantics: Question: (Q :PRINT-HOLE -1 :QUANTS
                      ((QINFO :QORDER 100 :QPROCF
                        #1=#<Interpreted Closure (:INTERNAL QPROCF) @
                             #x86136ea>
                        :QPROCQ
                        #2=#<Interpreted Closure (:INTERNAL
                                                  MAKE-QPROCQ)
                             @ #x8613822>
                        :HOLE 146)
                       (QINFO :QPROCQ
                        #3=#<Interpreted Closure (:INTERNAL
                                                  MAKE-QPROCQ)
                             @ #x849d7c2>
                        :QORDER 100 :PRINT-HOLE -1 :HOLE -1))
                      :WORD ATE :ID 2868 :RELATION EAT :TENSE PAST
                      :PATIENT
                      (NP :HOLE 146 :QUANTS
                       ((QINFO :QORDER 100 :QPROCF #1# :QPROCQ #2#
                         :HOLE 146)))
                      :AGENT
                      (NP :WORD WHO :ID 2834 :HOLE -1 :PRINT-HOLE -1
                       :QUANTS
                       ((QINFO :QPROCQ #3# :QORDER 100 :PRINT-HOLE -1
                         :HOLE -1))))
finished answering question (NIL)
finished answering question (((JOHN)))
finished answering question (((MARY)))
finished answering question (((BILL)))
finished answering question (((JOHN) (MARY) (BILL)))
```

```
((JOHN) (MARY) (BILL))
>
```