```
% she announced a program to promote safety in trucks and vans
exists(
  [She1,Announced1,Program1,Promote1,Vans1,Trucks1,SafetyLevel1,SafetyLevel2],
        and(isa(She1,tFemale),
            isa(Announced1,actAnnouncment),
            bodilyDoer(She1,Announced1),
            isa(Announced1,actMakingAvailable),
            madeAvailable(Announced1,Program1),
            isa(Program1,actWorkflow),
            workflowGoals(Program1,Promote1),
            isa(Promote1,actProcessChange),
            isa(Vans1,xtCollectionDenotedByFn("vans")),
            isa(Trucks1,xtCollectionDenotedByFn("trucks")),
            isa(SafetyLevel1,xtValueDenotedByFn("truck safety")),
            isa(SafetyLevel2,xtValueDenotedByFn("van safety")),
            levelOfSafety(actOperationFn(Trucks1),SafetyLevel1),
            levelOfSafety(actOperationFn(Vans1),SafetyLevel2),
            goalChanges(Promote1,increaseCausually(SafetyLevel1)),
            goalChanges(Promote1,increaseCausually(SafetyLevel2))))).
```

[BACK TO TOC](#)

```
#!prolog

------------------------------------------------------------------------------------------------------------------------------------------------------------
[the, shopper, drank, the, pepsi]
(thereExists ?shopper4675    (and (isa ?shopper4675 (FrequentPerformerFn Shopping ) )     (thereExists ?pepsi6      (and (isa ?pepsi6
PepsiCola )      (and (occursDuring ?drank7550 Past )       (and (isa ?drank7550 DrinkingEvent )       (performedBy ?drank7550
?shopper4675 )    (consumedObject ?drank7550 ?pepsi6 ) ) ) ) ) ) )

[the, shopper, drinks, the, pepsi]
(thereExists ?shopper4675  (and (isa ?shopper4675 (FrequentPerformerFn Shopping ) )
   (thereExists ?pepsi6 (and (isa ?pepsi6 PepsiCola ) (occursDuring ?drank7550 Now ) (isa ?drank7550 DrinkingEvent )
     (performedBy ?drank7550 ?shopper4675 )  (consumedObject ?drank7550 ?pepsi6 ) ) ) )

[the, shopper, bought, fries]
(thereExists ?shopper4794
   (and (isa ?shopper4794 (FrequentPerformerFn Shopping ) )
   (and (and (isa ?fries20 Frying )
   (objectOfStateChange ?fries20 ?POSSESSOR13610 ) )
   (and (occursDuring ?bought3 Past )
   (and (isa ?bought3 Buying )
```

```
   (buyer ?bought3 ?shopper4794 )
   (objectPaidFor ?bought3 ?fries20 ) ) ) ) ) )

[the, shopper, bought, fries, with, his, pepsi]
(thereExists ?shopper4863
    (and (isa ?shopper4863 (FrequentPerformerFn Shopping )
     (isa ?pepsi42 PepsiCola )
    (and (and (and (isa ?fries103 Frying )
       (objectOfStateChange ?fries103 ?POSSESSOR13613 ) )
    (and (isa ?bought4 Buying ) (seller ?bought4 ?shopper4863 )
    (objectPaidFor ?bought4 ?fries103 )
    (buyingPerformer ?bought4 ?shopper4863 ) ) )
    (equals ?POSSESSOR13613
            (OneOfFn (PronounFn ThirdPerson-NLAttr Singular-NLAttr Masculine-NLAttr PossessivePronoun-Post )
             (PronounFn ThirdPerson-NLAttr Singular-NLAttr Masculine-NLAttr PossessivePronoun-Pre ) ) )
    (controls (OneOfFn (PronounFn ThirdPerson-NLAttr Singular-NLAttr Masculine-NLAttr PossessivePronoun-Post )
                       (PronounFn ThirdPerson-NLAttr Singular-NLAttr Masculine-NLAttr PossessivePronoun-Pre ) )
                  ?pepsi42 ) ) ) ) )



|    english2Kif("I see you").
assert(known_phrase(verbSemTrans(#$and(#$isa(?ACTION1, #$VisualPerception), #$pe
rformedBy(?ACTION1, ?Speaker), #$perceivedThings(?ACTION1, ?TargetAgent)), true)
))


e2c("George fell last year").

[[txt, 'George'], [tag, staart, nnp, np]]
[[txt, fell], [tag, z, vbd]]
[[txt, last], [tag, z, jj, ap, 'Adjective', 'Adverb', 'CountNoun', 'WHAdverb']]
[[txt, year], [tag, laast, nn, 'CountNoun']]

seg([p('VP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('PP_non_cyclic', 3, 3), p('VP_non_cyclic', 4, 4)])

(implies
  (equals ?GEORGE2
    (OneOfFn EddieGeorge-FootballPlayer-American
      (CityNamedFn "George" RepublicOfSouthAfrica ) ) )
  (and
    (isa ?FELL1 Event )
    (implies
      (and
        (isa ?YEAR5 CalendarYear )
        (isa ?YEAR5 ContemporaryArtifact ) )
      (or
        (and
          (isa ?FELL1 CuttingDownATree )
          (objectOfStateChange ?FELL1 ?YEAR5 )
          (doneBy ?FELL1 ?GEORGE2 ) )
        (and
          (organismKilled ?FELL1 ?YEAR5 )
          (isa ?FELL1 CuttingDownATree )
          (performedBy ?FELL1 ?GEORGE2 ) ) ) ) ) )
```

```
e2c("I see you").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, see], [tag, z, vb]]
[[txt, you], [tag, laast, ppss, ppo, prp, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB',
'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]

seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('ADVP_non_cyclic', 3, 3)])

(thereExists ?SPEAKER
  (and
    (isa ?SPEAKER Person )
    (and
      (isa ?SEE1 Event )
      (thereExists ?TARGETAGENT
        (and
          (isa ?TARGETAGENT Person )
          (or
            (awareOf ?SPEAKER ?TARGETAGENT )
            (and
              (isa ?SEE1 VisualPerception )
              (performedBy ?SEE1 ?SPEAKER )
              (perceivedThings ?SEE1 ?TARGETAGENT ) ) ) ) ) ) ) )


[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, seen], [tag, z, vbn]]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, wood], [tag, laast, nn, 'CountNoun', 'MassNoun', 'MassNoun-Generic']]

seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('QP', 3, 4)])
?- e2c("I saw the wood").
(thereExists ?SPEAKER
  (and
    (isa ?SPEAKER Person )
    (and
      (isa ?SEEN1 Event )
      (thereExists ?WOOD5
        (and
          (isa ?WOOD5
            (OneOfFn Wood Wood-GolfClub WoodedArea ) )
          (or
            (awareOf ?SPEAKER ?WOOD5 )
            (and
              (isa ?SEEN1 VisualPerception )
              (performedBy ?SEEN1 ?SPEAKER )
              (perceivedThings ?SEEN1 ?WOOD5 ) ) ) ) ) ) ) )

e2c("I sawed the wood").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
```

```
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, sawed], [tag, z, 'NLWordForm', 'Verb', 'Adjective']]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, wood], [tag, laast, nn, 'CountNoun', 'MassNoun', 'MassNoun-Generic']]

?- e2c("I sawed the wood").
(thereExists ?SPEAKER
  (and
    (isa ?SPEAKER Person )
    (and
      (isa ?SAWED1 Event )
      (thereExists ?WOOD11
        (and
          (isa ?WOOD11
            (OneOfFn Wood Wood-GolfClub WoodedArea ) )
          (and
            (isa ?SAWED1 SawingSomething )
            (objectOfStateChange ?SAWED1 ?WOOD11 )
            (doneBy ?SAWED1 ?SPEAKER ) ) ) ) ) ) )

e2c("I see two books on a shelf").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, see], [tag, z, vb]]
[[txt, two], [tag, z, cd, 'CountNoun', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun', 'PossessivePronoun-Post', 'Postdeterminer',
'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, books], [tag, z, nns, 'CountNoun', 'ProperCountNoun', 'ProperNoun']]
[[txt, on], [tag, z, in, 'Adverb', 'Preposition', 'Preposition-Directional-Telic', 'Preposition-Locative', 'VerbParticle', 'WHAdverb']]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, shelf], [tag, laast, nn, 'CountNoun']]

seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('QP', 3, 4), p('PP_non_cyclic', 5, 5), p('QP', 6, 7)])

(thereExists (?SPEAKER ?SEE)   (and     (isa ?SPEAKER Person )     (isa ?SEE2 VisualPerception )     (thereExistExactly 2 ?TWO2
(isa ?TWO2 BookCopy )                    (awareOf ?SPEAKER ?TWO2 )                                (performedBy ?SEE2 ?SPEAKER )
(perceivedThings ?SEE2 ?TWO2 ) ) )             (thereExists ?SHELF5               (and                  (isa ?SHELF5ShelfInABuilding )
(on-UnderspecifiedSurface ?TWO2 ?SHELF5 ) ) ) ) ) ) ) )

e2c("I see two books sitting on a shelf").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, see], [tag, z, vb]]
[[txt, two], [tag, z, cd, 'CountNoun', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun', 'PossessivePronoun-Post', 'Postdeterminer',
'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, books], [tag, z, nns, 'CountNoun', 'ProperCountNoun', 'ProperNoun']]
[[txt, sitting], [tag, z, vbg, 'GerundiveCountNoun', 'GerundiveNoun', 'MassNoun-Generic', 'Adjective']]
[[txt, on], [tag, z, in, 'Adverb', 'Preposition', 'Preposition-Directional-Telic', 'Preposition-Locative', 'VerbParticle', 'WHAdverb']]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, shelf], [tag, laast, nn, 'CountNoun']]
```

```
seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('QP', 3, 4), p('VP_non_cyclic', 5, 5), p('PP_non_cyclic', 6, 6), p('QP', 7, 8)])

(thereExists ?SPEAKER
  (and
    (isa ?SPEAKER Person )
    (and
      (isa ?SEE3 Event )
      (thereExistExactly 2 ?TWO4
        (and
          (and
            (isa ?TWO4 BookCopy )
            (isa ?TWO4 Supported) )
          (and
            (or
              (awareOf ?SPEAKER ?TWO4 )
              (and
                (isa ?SEE3 VisualPerception )
                (performedBy ?SEE3 ?SPEAKER )
                (perceivedThings ?SEE3 ?TWO4 ) ) )
            (thereExists ?SHELF11
              (and
                (isa ?SHELF11 ShelfInABuilding )
                (on-UnderspecifiedSurface ?TWO4 ?SHELF11 ) ) ) ) ) ) ) ) )

e2c("I can see you").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun', 'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, can], [tag, z, md, 'Modal', 'CountNoun']]
[[txt, see], [tag, z, vb]]
[[txt, you], [tag, laast, ppss, ppo, prp, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun', 'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]

seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('VP_non_cyclic', 3, 3), p('ADVP_non_cyclic', 4, 4)])

(thereExists ?SPEAKER
  (and
    (isa ?SPEAKER Person )
    (and
      (behavourCapable ?SPEAKER ?SEE4 )
      (and
        (isa ?SEE4 Event )
        (thereExists ?TARGETAGENT
          (and
            (isa ?TARGETAGENT Person )
            (or
              (awareOf ?SPEAKER ?TARGETAGENT )
              (and
                (isa ?SEE4 VisualPerception )
                (performedBy ?SEE4 ?SPEAKER )
                (perceivedThings ?SEE4 ?TARGETAGENT ) ) ) ) ) ) ) ) )

e2c("Fido is a dog").
```

```
[[txt, 'Fido'], [tag, staart]]
[[txt, is], [tag, z, bez, vbz, 'BeAux']]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, dog], [tag, laast, nn, 'CountNoun']]

?- e2c("Fido is a dog").
(implies
  (properNameString ?FIDO3 "Fido" )
  (thereExists ?DOG8
    (or
      (and
        (isa ?DOG8 InsultingSomeone )
        (subjectOfInfo ?DOG8 :POSSESSOR ) )
      (and
        (isa ?IS1 Living )
        (occursDuring ?IS1 Now )
        (preActors ?IS1 ?FIDO3 )
        (actors ?IS1 ?DOG8 ) ) ) ) )

?- e2c("Who is the president of the United States of America ?").

[[txt, 'Who'], [tag, staart, nnp, prp, wp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, is], [tag, z, bez, vbz, 'BeAux']]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, president], [tag, z, nn, 'CountNoun', 'ProperCountNoun', 'ProperNoun', 'Adjective']]
[[txt, of], [tag, z, in, 'Preposition', 'Preposition-Of']]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, 'United', 'States', of, 'America'], [tag, z]]
[[txt, ?], [tag, laast, ?, 'Punctuation-SP']]


?- e2c("Who is the president of the United States of America ?").
(CYC-QUERY
  `(thereExists ?Who
    (thereExists ?PRESIDENT46
      (and
        (and
          (and
            (isa ?PRESIDENT46 Thing )
            (isa ?PRESIDENT46 Thing ) )
          (isa ?PRESIDENT46
            (OneOfFn PresidentOfOrganization
              (SubcollectionOfWithRelationFromTypeFn Person president Organization ) ) ) )
        (and
          (and
            (isa ?IS10 Living )
            (occursDuring ?IS10 Now )
            (preActors ?IS10 ?Who )
            (actors ?IS10 ?PRESIDENT46 ) )
          (thereExists ?G300312
            (and
              (eq ?G300312 UnitedStatesOfAmerica )
              (possessiveRelation ?G300312 ?PRESIDENT46 ) ) ) ) ) ) ) )
```

```
e2c("I saw him").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, saw], [tag, z, vbd, 'CountNoun']]
[[txt, him], [tag, laast, ppo, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]

seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('ADVP_non_cyclic', 3, 3)])

(thereExists ?SPEAKER
  (and
    (isa ?SPEAKER Person )
    (and
      (isa ?SAW1 Event )
      (thereExists ?HIM
        (and
          (equals ?HIM
            (PronounFn ThirdPerson-NLAttr Singular-NLAttr Masculine-NLAttr ObjectPronoun ) )
          (or
            (awareOf ?SPEAKER ?HIM )
            (and
              (isa ?SAW1 SawingSomething )
              (objectOfStateChange ?SAW1 ?HIM )
              (doneBy ?SAW1 ?SPEAKER ) )
            (and
              (isa ?SAW1 VisualPerception )
              (performedBy ?SAW1 ?SPEAKER )
              (perceivedThings ?SAW1 ?HIM ) ) ) ) ) ) ) )

e2c("I saw him Monday").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, saw], [tag, z, vbd, 'CountNoun']]
[[txt, him], [tag, z, ppo, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, 'Monday'], [tag, laast, nnp, nr, 'CountNoun', 'ProperCountNoun', 'ProperNoun']]

seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('ADVP_non_cyclic', 3, 3), p('VP_non_cyclic', 4, 4)])

?- e2c("I saw him Monday").
(thereExists ?SPEAKER
  (and
    (isa ?SPEAKER Person )
    (implies
      (occursDuring ?SAW2 Monday )
      (holdsIn ?SAW2
        (and
          (isa ?SAW2 Event )
          (thereExists ?HIM
            (and
              (equals ?HIM
                (PronounFn ThirdPerson-NLAttr Singular-NLAttr Masculine-NLAttr ObjectPronoun ) )
              (or
                (awareOf ?SPEAKER ?HIM )
```

```
                (and
                  (isa ?SAW2 SawingSomething )
                  (objectOfStateChange ?SAW2 ?HIM )
                  (doneBy ?SAW2 ?SPEAKER ) )
                (and
                  (isa ?SAW2 VisualPerception )
                  (performedBy ?SAW2 ?SPEAKER )
                  (perceivedThings ?SAW2 ?HIM ) ) ) ) ) ) ) ) )

e2c("I saw him January").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, saw], [tag, z, vbd, 'CountNoun']]
[[txt, him], [tag, z, ppo, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, 'January'], [tag, laast, nnp, np, 'ProperCountNoun', 'ProperNoun']]

seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('ADVP_non_cyclic', 3, 3), p('VP_non_cyclic', 4, 4)])

(thereExists ?SPEAKER
  (and
    (isa ?SPEAKER Person )
    (implies
      (occursDuring ?SAW3 January )
      (holdsIn ?SAW3
        (and
          (isa ?SAW3 Event )
          (thereExists ?HIM
            (and
              (equals ?HIM
                (PronounFn ThirdPerson-NLAttr Singular-NLAttr Masculine-NLAttr ObjectPronoun ) )
              (or
                (awareOf ?SPEAKER ?HIM )
                (and
                  (isa ?SAW3 SawingSomething )
                  (objectOfStateChange ?SAW3 ?HIM )
                  (doneBy ?SAW3 ?SPEAKER ) )
                (and
                  (isa ?SAW3 VisualPerception )
                  (performedBy ?SAW3 ?SPEAKER )
                  (perceivedThings ?SAW3 ?HIM ) ) ) ) ) ) ) ) ) )

e2c("I saw him January twenty first").

[[txt, 'I'], [tag, staart, ppss, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, saw], [tag, z, vbd, 'CountNoun']]
[[txt, him], [tag, z, ppo, prp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, 'January'], [tag, z, nnp, np, 'ProperCountNoun', 'ProperNoun']]
[[txt, twenty], [tag, z, cd, 'CountNoun', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB',
'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, first], [tag, laast, jj, od, rb, 'Adjective', 'Adverb', 'CountNoun', 'NongradableAdjective', 'OrdinalAdjective', 'WHAdverb']]
```

```
seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('ADVP_non_cyclic', 3, 3), p('VP_non_cyclic', 4, 4), p('QP', 5, 5),
p('PP_non_cyclic', 6, 6)])

e2c("Monday ' s concert should be good").

[[txt, 'Monday\'s'], [txt, 'Monday', '\'s'], [txt, 'Monday'], [tag, staart, '\'s', 'nr$']]
[[txt, concert], [tag, z, nn, 'CountNoun']]
[[txt, should], [tag, z, md, 'Modal']]
[[txt, be], [tag, z, be, vb, 'BeAux']]
[[txt, good], [tag, laast, jj, 'Adjective']]


e2c("We did it in Jan").

[[txt, 'We'], [tag, staart, prp, ppss, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun', 'PossessivePronoun-Post', 'Postdeterminer',
'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, did], [tag, z, dod, vbd, 'DoAux']]
[[txt, it], [tag, z, prp, pps, ppo, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, in], [tag, z, in, 'Adjective', 'Adverb', 'CountNoun', 'Preposition', 'Preposition-Directional-Telic', 'Preposition-Locative',
'VerbParticle', 'WHAdverb']]
[[txt, 'Jan'], [tag, laast, nnp, np]]

seg([p('ADVP_non_cyclic', 1, 1), p('VP_non_cyclic', 2, 2), p('ADVP_non_cyclic', 3, 3), p('PP_non_cyclic', 4, 4), p('VP_non_cyclic', 5,
5)])

(thereExists ?WE
  (and
    (equals ?WE
      (PronounFn FirstPerson-NLAttr Plural-NLAttr Ungendered-NLAttr SubjectPronoun ) )
    (and
      (and
        (occursDurring ?IN3 ?G431374 )
        (bodilyDoer ?WE ?IN3 ) )
      (implies
        (properNameString ?JAN68 "Jan" )
        (and
          (occursDuring ?IN3 Now )
          (holdsIn ?IN3
            (in-UnderspecifiedContainer ?WE ?JAN68 ) ) ) ) ) ) )


e2c("Failure to comply may result in dismissal").

[[txt, 'Failure'], [tag, staart, nn, 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, to], [tag, z, to, in, 'InfinitiveComp', 'Preposition', 'Preposition-Directional']]
[[txt, comply], [tag, z, vb, 'Adjective', 'Adverb', 'WHAdverb']]
[[txt, may], [tag, z, md, 'Modal', 'ProperCountNoun', 'ProperNoun']]
[[txt, result], [tag, z, nn, vb, 'CountNoun']]
[[txt, in], [tag, z, in, 'Adjective', 'Adverb', 'CountNoun', 'Preposition', 'Preposition-Directional-Telic', 'Preposition-Locative',
'VerbParticle', 'WHAdverb']]
[[txt, dismissal], [tag, laast, nn, 'CountNoun', 'Adjective']]

seg([p('VP_non_cyclic', 1, 1), p('PP_non_cyclic', 2, 2), p('VP_non_cyclic', 3, 3), p('VP_non_cyclic', 4, 4), p('VP_non_cyclic', 5, 5),
p('PP_non_cyclic', 6, 6), p('VP_non_cyclic', 7, 7)])
```

```
(implies
  (properNameString ?FAILURE3 "Failure" )
  (and
    (and
      (isa ?RESULT3
        (AdverbFn Comply-TheWord ) )
      (and
        (behavourCapable ?FAILURE3 ?RESULT3 )
        (implies
          (and
            (isa ?DISMISSAL9
              (NounFn Dismiss-TheWord ) )
            (agentPopularity ?DISMISSAL9
              (HighAmountFn PopularityLevel ) ) )
          (and
            (occursDuring ?RESULT3 Now )
            (holdsIn ?RESULT3
              (eventOutcomes ?FAILURE3 ?DISMISSAL9 ) ) ) ) ) )
    (implies
      (and
        (isa ?G461366 Thing )
        (isa ?G461366 Thing ) )
      (to-UnderspecifiedLocation ?FAILURE3 ?G461366 ) ) ) )

e2c("Absence to comply may result in dismissal").

[[txt, 'Absence'], [tag, staart, 'CountNoun', 'CountNoun-Generic', 'NLWordForm', 'Noun']]
[[txt, to], [tag, z, to, in, 'InfinitiveComp', 'Preposition', 'Preposition-Directional']]
[[txt, comply], [tag, z, vb, 'Adjective', 'Adverb', 'WHAdverb']]
[[txt, may], [tag, z, md, 'Modal', 'ProperCountNoun', 'ProperNoun']]
[[txt, result], [tag, z, nn, vb, 'CountNoun']]
[[txt, in], [tag, z, in, 'Adjective', 'Adverb', 'CountNoun', 'Preposition', 'Preposition-Directional-Telic', 'Preposition-Locative',
'VerbParticle', 'WHAdverb']]
[[txt, dismissal], [tag, laast, nn, 'CountNoun', 'Adjective']]


(implies
  (properNameString ?ABSENCE3 "Absence" )
  (and
    (and
      (isa ?RESULT7
        (AdverbFn Comply-TheWord ) )
      (and
        (behavourCapable ?ABSENCE3 ?RESULT7 )
        (implies
          (and
            (isa ?DISMISSAL18
              (NounFn Dismiss-TheWord ) )
            (agentPopularity ?DISMISSAL18
              (HighAmountFn PopularityLevel ) ) )
          (and
            (occursDuring ?RESULT7 Now )
            (holdsIn ?RESULT7
              (eventOutcomes ?ABSENCE3 ?DISMISSAL18 ) ) ) ) ) )
    (implies
      (and
        (isa ?G495123 Thing )
```

```
        (isa ?G495123 Thing ) )
      (to-UnderspecifiedLocation ?ABSENCE3 ?G495123 ) ) ) )
```

```
e2c("He made a request that there be an investigation").
```

```
[[txt, 'He'], [tag, staart, pps, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, made], [tag, z, vbd, vbn]]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, request], [tag, z, nn, 'CountNoun']]
[[txt, that], [tag, z, in, cs, dt, wdt, 'Adverb', 'Complementizer', 'Determiner', 'Determiner-Central', 'Determiner-ClassA',
'Determiner-ClassB', 'Determiner-ClassC', 'Determiner-Definite', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHAdverb', 'WHDeterminer', 'WHPronoun']]
[[txt, there], [tag, z, ex, rb, 'Adverb', 'AdverbOfPlace', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun',
'PossessivePronoun', 'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'There-Existential', 'WHAdverb', 'WHPronoun']]
[[txt, be], [tag, z, be, vb, 'BeAux']]
[[txt, an], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer']]
[[txt, investigation], [tag, laast, nn, 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
```

```
(thereExists ?HE   (and     (isa ?HE Male )     (and       (isa ?MADE1 Event )       (thereExists?REQUEST8        (and           (and
(isa ?REQUEST8               (OneOfFn InformationRequest-IBT RequestRequesting-CommunicationAct ) )             (isa ?REQUEST8 Thing ) )
(and             (or              (causes-Underspecified ?HE ?REQUEST8 )               (createdBy ?REQUEST8 ?HE )
(doneBy?REQUEST8 ?HE )               (and                (isa ?MADE1 MakingSomething )
                (performedBy ?MADE1 ?HE )
                (products ?MADE1 ?REQUEST8 ) ) )
            (and
              (isa ?BE29
                (AdverbFn There-TheWord ) )
              (thereExists ?INVESTIGATION7
                (and
                  (isa ?INVESTIGATION7 Research )
                  (and
                    (isa ?BE29 Living )
                    (occursDuring ?BE29 Now )
                    (preActors ?BE29 ?REQUEST8 )
                    (actors ?BE29 ?INVESTIGATION7 ) ) ) ) ) ) ) ) ) )
```

```
e2c("He stated the fact that there be an investigation").
```

```
[[txt, 'He'], [tag, staart, pps, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun']]
[[txt, stated], [tag, z, vbn, vbd, 'Adjective']]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, fact], [tag, z, nn, 'CountNoun']]
[[txt, that], [tag, z, in, cs, dt, wdt, 'Adverb', 'Complementizer', 'Determiner', 'Determiner-Central', 'Determiner-ClassA',
'Determiner-ClassB', 'Determiner-ClassC', 'Determiner-Definite', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHAdverb', 'WHDeterminer', 'WHPronoun']]
[[txt, there], [tag, z, ex, rb, 'Adverb', 'AdverbOfPlace', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun',
'PossessivePronoun', 'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'There-Existential', 'WHAdverb', 'WHPronoun']]
```

```
[[txt, be], [tag, z, be, vb, 'BeAux']]
[[txt, an], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer']]
[[txt, investigation], [tag, laast, nn, 'CountNoun', 'MassNoun', 'MassNoun-Generic']]


(thereExists ?HE
  (and
    (isa ?HE Male )
    (thereExists ?FACT10
      (and
        (ist :POSSESSOR ?FACT10 )
        (and
          (implies
            (isa ?STATED3
              (VerbFn State-TheWord ) )
            (eventSOT ?STATED3 ?HE ?FACT10 Past ) )
          (and
            (isa ?BE31
              (AdverbFn There-TheWord ) )
            (thereExists ?INVESTIGATION15
              (and
                (isa ?INVESTIGATION15 Research )
                (and
                  (isa ?BE31 Living )
                  (occursDuring ?BE31 Now )
                  (preActors ?BE31 ?FACT10 )
                  (actors ?BE31 ?INVESTIGATION15 ) ) ) ) ) ) ) ) ) )


e2c("Many were angered by the hearings").

[[txt, 'Many'], [tag, staart, dt, jj, nnp, pdt, rb, 'Adverb', 'Determiner', 'Determiner-Central', 'Determiner-ClassA',
'Determiner-ClassB', 'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHAdverb', 'WHDeterminer', 'WHPronoun', 'Adjective']]
[[txt, were], [tag, z, bed, vbd, 'BeAux']]
[[txt, angered], [tag, z, vbn, vbd, 'Adjective']]
[[txt, by], [tag, z, in, 'Adverb', 'Preposition', 'Preposition-Directional-Telic', 'WHAdverb', 'Adjective']]
[[txt, a], [tag, z, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, hearings], [tag, laast, nns, 'CountNoun', 'GerundiveCountNoun']]

seg([p('QP', 1, 1), p('VP_non_cyclic', 2, 2), p('VP_non_cyclic', 3, 3), p('PP_non_cyclic', 4, 4), p('QP', 5, 5), p('QP', 6, 6)])

(thereExists ?MANY
  (and
    (or
      (and
        (equals ?MANY Many-NLAttr )
        (isa ?MANY
          (CollectionUnionFn Individual NLGenQuantAttribute ) ) )
      (and
        (equals ?MANY Many-Quant )
        (isa ?MANY
          (CollectionUnionFn Individual PositiveIntegerExtent ) ) )
      (and
```

```
        (equals ?MANY many )
        (isa ?MANY
          (CollectionUnionFn GeneralizedQuantifier-Conservative Individual ReflexiveBinaryPredicate TruthFunction ) ) ) )
    (and
      (occursDurring ?ANGERED3 Past )
      (implies
          (isa ?G626264 Thing )
        (and
          (implies
            (isa ?ANGERED3
              (VerbFn Anger-TheWord ) )
            (eventSOT ?ANGERED3 ?MANY ?G626264 Past ) )
          (thereExists ?HEARINGS5
            (and
              (and
                (isa ?HEARINGS5 Hearing )
                (objectActedOn ?HEARINGS5 :POSSESSOR ) )
              (by-Underspecified ?G626264 ?HEARINGS5 ) ) ) ) ) ) ) )


e2c("Many who initially supported Thomas later changed their minds").

[[txt, 'Many'], [tag, staart, dt, jj, nnp, pdt, rb, 'Adverb', 'Determiner', 'Determiner-Central', 'Determiner-ClassA',
'Determiner-ClassB', 'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHAdverb', 'WHDeterminer', 'WHPronoun', 'Adjective']]
[[txt, who], [tag, z, wps, wp, 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHPronoun',
'WHPronoun-Subject']]
[[txt, initially], [tag, z, rb, 'Adverb', 'WHAdverb', 'Adjective']]
[[txt, supported], [tag, z, vbn, vbd, 'Adjective']]
[[txt, 'Thomas'], [tag, z, np, nnp]]
[[txt, later], [tag, z, rbr, rb, 'Adjective', 'Adverb', 'AgentiveNoun', 'AdverbOfTime', 'WHAdverb']]
[[txt, changed], [tag, z, vbn, vbd, 'Adjective', 'CountNoun']]
[[txt, their], [tag, z, 'pp$', 'prp$', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun', 'PossessivePronoun-Post',
'PossessivePronoun-Pre', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, minds], [tag, laast, nns, 'CountNoun']]

seg([p('QP', 1, 1), p('X', 2, 2), p('PP_non_cyclic', 3, 3), p('VP_non_cyclic', 4, 4), p('VP_non_cyclic', 5, 5), p('QP', 6, 6),
p('VP_non_cyclic', 7, 7), p('NX', 8, 9)])

(thereExists ?MANY
  (and
    (or
      (and
        (equals ?MANY Many-NLAttr )
        (isa ?MANY
          (CollectionUnionFn Individual NLGenQuantAttribute ) ) )
      (and
        (equals ?MANY Many-Quant )
        (isa ?MANY
          (CollectionUnionFn Individual PositiveIntegerExtent ) ) )
      (and
        (equals ?MANY many )
        (isa ?MANY
```

```
          (CollectionUnionFn GeneralizedQuantifier-Conservative Individual ReflexiveBinaryPredicate TruthFunction ) ) ) )
    (implies
      (and
        (or
          (and
            (equals ?MANY Agent-PartiallyTangible )
            (isa ?MANY
              (CollectionUnionFn Collection ExistingObjectType StuffType TemporalStuffType ) ) ) )
        (and
          (isa ?CHANGED1 Event )
          (and
            (controls ?THEIR ?MINDS4 )
            (and
              (equals ?THEIR
                (PronounFn ThirdPerson-NLAttr Plural-NLAttr Ungendered-NLAttr PossessivePronoun-Pre ) )
              (implies
                (and
                  (isa ?MINDS4 TakingCareOfSomething )
                  (beneficiary ?MINDS4 :POSSESSOR ) )
                (or
                  (and
                    (isa ?CHANGED1 IntrinsicStateChangeEvent )
                    (performedBy ?CHANGED1 ?MANY )
                    (objectOfStateChange ?CHANGED1 ?MINDS4 ) )
                  (and
                    (isa ?CHANGED1 ReplacingAnObject )
                    (objectActedOn ?CHANGED1 ?MINDS4 )
                    (doneBy ?CHANGED1 ?MANY ) ) ) ) ) ) ) )
      (and
        (isa ?SUPPORTED1
          (AdverbFn Initial-TheWord ) )
        (and
          (isa ?SUPPORTED1
            (AdverbFn Late-TheWord ) )
          (and
            (isa ?SUPPORTED1 Event )
            (implies
              (equals ?THOMAS2 JackThomas-911FOAF )
              (or
                (satisfiesStandard ?MANY ?THOMAS2 )
                (and
                  (isa ?SUPPORTED1 ShowingSupportForSomeone )
                  (beneficiary ?SUPPORTED1 ?THOMAS2 )
                  (performedBy ?SUPPORTED1 ?MANY ) )
                (and
                  (isa ?SUPPORTED1 SupportingSomething )
                  (supportedObject ?SUPPORTED1 ?THOMAS2 )
                  (supportingObject ?SUPPORTED1 ?MANY ) ) ) ) ) ) ) ) ) )


e2c("Those that want to come can come").

[[txt, 'Those'], [tag, staart, dt, nnp, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB',
'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, that], [tag, z, in, cs, dt, wdt, 'Adverb', 'Complementizer', 'Determiner', 'Determiner-Central', 'Determiner-ClassA',
```

'Determiner-ClassB', 'Determiner-ClassC', 'Determiner-Definite', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun', 'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun', 'SubjectPronoun', 'WHAdverb', 'WHDeterminer', 'WHPronoun']]
[[txt, want], [tag, z, vb, vbp, 'CountNoun']]
[[txt, to], [tag, z, to, in, 'InfinitiveComp', 'Preposition', 'Preposition-Directional']]
[[txt, come], [tag, z, vb, vbn]]
[[txt, can], [tag, z, md, 'Modal', 'CountNoun']]
[[txt, come], [tag, laast, vb, vbn]]

seg([p('PP_non_cyclic', 1, 2), p('VP_non_cyclic', 3, 3), p('PP_non_cyclic', 4, 4), p('VP_non_cyclic', 5, 5), p('VP_non_cyclic', 6, 6), p('VP_non_cyclic', 7, 7)])

```
(thereExists ?THOSE
  (and
    (or
      (and
        (equals ?THOSE Backreference-ClassB-NLAttr )
        (isa ?THOSE
          (CollectionUnionFn Backreference-NLAttrType Individual ) ) )
      (and
        (equals ?THOSE Demonstrative-ClassB-NLAttr )
        (isa ?THOSE
          (CollectionUnionFn Demonstrative-NLAttrType Individual ) ) ) )
    (and
      (eventSOT ?EVENT88 ?THOSE ?OBJ88 ?TIME88 )
      (implies
        (occursDuring ?WANT3
          (OneOfFn ArrivingAtAPlace HavingAnOrgasm ) )
        (holdsIn ?WANT3
          (implies
            (and
              (isa ?G738781 Thing )
              (isa ?G738781 Thing ) )
            (and
              (implies
                (isa ?WANT3
                  (VerbFn Want-TheWord ) )
                (eventSOT ?WANT3 ?THOSE ?G738781 Now ) )
              (implies
                (and
                  (isa ?COME4 Emission )
                  (isa ?COME4
                    (OneOfFn Buttocks Can CanningFood ) ) )
                (to-UnderspecifiedLocation ?G738781 ?COME4 ) ) ) ) ) ) ) )
```

e2c("the best costumes got prizes").

[[txt, a], [tag, staart, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC', 'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, best], [tag, z, jjt, jjs, 'CountNoun', 'Adjective', 'Adverb']]
[[txt, costumes], [tag, z, nns, 'CountNoun']]
[[txt, got], [tag, z, vbd, vbn]]
[[txt, prizes], [tag, laast, nns, 'CountNoun']]

seg([p('QP', 1, 1), p('ADVP_non_cyclic', 2, 2), p('QP', 3, 3), p('VP_non_cyclic', 4, 4), p('QP', 5, 5)])

```
(thereExists ?COSTUMES5
  (and
    (and
      (isa ?COSTUMES5 Costume )
      (isa ?COSTUMES5 Thing ) )
    (implies
      (isa ?PRIZES7 AwardPractice )
      (implies
        (isa ?GOT7
          (VerbFn Get-TheWord ) )
        (eventSOT ?GOT7 ?COSTUMES5 ?PRIZES7 Past ) ) ) ) )

e2c("a best costume got prizes").

[[txt, a], [tag, staart, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, best], [tag, z, jjt, jjs, 'CountNoun', 'Adjective', 'Adverb']]
[[txt, costume], [tag, z, nn, 'CountNoun']]
[[txt, got], [tag, z, vbd, vbn]]
[[txt, prizes], [tag, laast, nns, 'CountNoun']]

seg([p('QP', 1, 1), p('ADVP_non_cyclic', 2, 2), p('VP_non_cyclic', 3, 3), p('VP_non_cyclic', 4, 4), p('QP', 5, 5)])

(thereExists ?COSTUME5
  (and
    (and
      (isa ?COSTUME5 Costume )
      (isa ?COSTUME5 Thing ) )
    (implies
      (isa ?PRIZES15 AwardPractice )
      (implies
        (isa ?GOT14
          (VerbFn Get-TheWord ) )
        (eventSOT ?GOT14 ?COSTUME5 ?PRIZES15 Past ) ) ) ) )

e2c("some best costumes got prizes").

[[txt, some], [tag, staart, dt, dti, 'Adverb', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB',
'Determiner-ClassC', 'Postdeterminer', 'WHAdverb', 'WHDeterminer']]
[[txt, best], [tag, z, jjt, jjs, 'CountNoun', 'Adjective', 'Adverb']]
[[txt, costumes], [tag, z, nns, 'CountNoun']]
[[txt, got], [tag, z, vbd, vbn]]
[[txt, prizes], [tag, laast, nns, 'CountNoun']]

seg([p('QP', 1, 1), p('ADVP_non_cyclic', 2, 2), p('QP', 3, 3), p('VP_non_cyclic', 4, 4), p('QP', 5, 5)])

(thereExists ?COSTUMES10
  (and
    (and
      (isa ?COSTUMES10 Costume )
      (isa ?COSTUMES10 Thing ) )
    (implies
      (isa ?PRIZES23 AwardPractice )
      (implies
        (isa ?GOT21
          (VerbFn Get-TheWord ) )
        (eventSOT ?GOT21 ?COSTUMES10 ?PRIZES23 Past ) ) ) ) )
```

```
e2c("the best five costumes got prizes").

[[txt, a], [tag, staart, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, best], [tag, z, jjt, jjs, 'CountNoun', 'Adjective', 'Adverb']]
[[txt, five], [tag, z, cd, 'CountNoun', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB',
'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, costumes], [tag, z, nns, 'CountNoun']]
[[txt, got], [tag, z, vbd, vbn]]
[[txt, prizes], [tag, laast, nns, 'CountNoun']]

seg([p('QP', 1, 1), p('ADVP_non_cyclic', 2, 2), p('QP', 3, 4), p('VP_non_cyclic', 5, 5), p('QP', 6, 6)])

(thereExistExactly 5 ?FIVE1
  (and
    (and
      (isa ?FIVE1 Costume )
      (isa ?FIVE1 Thing ) )
    (implies
      (isa ?PRIZES31 AwardPractice )
      (implies
        (isa ?GOT28
          (VerbFn Get-TheWord ) )
        (eventSOT ?GOT28 ?FIVE1 ?PRIZES31 Past ) ) ) ) )

e2c("best costumes got prizes").

[[txt, best], [tag, staart, jjt, jjs, 'CountNoun', 'Adjective', 'Adverb']]
[[txt, costumes], [tag, z, nns, 'CountNoun']]
[[txt, got], [tag, z, vbd, vbn]]
[[txt, prizes], [tag, laast, nns, 'CountNoun']]

seg([p('ADVP_non_cyclic', 1, 1), p('QP', 2, 2), p('VP_non_cyclic', 3, 3), p('QP', 4, 4)])

e2c("best five costumes got prizes").

[[txt, best], [tag, staart, jjt, jjs, 'CountNoun', 'Adjective', 'Adverb']]
[[txt, five], [tag, z, cd, 'CountNoun', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB',
'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, costumes], [tag, z, nns, 'CountNoun']]
[[txt, got], [tag, z, vbd, vbn]]
[[txt, prizes], [tag, laast, nns, 'CountNoun']]

seg([p('ADVP_non_cyclic', 1, 1), p('QP', 2, 3), p('VP_non_cyclic', 4, 4), p('QP', 5, 5)])

(implies
  (and
    (and
      (isa ?COSTUMES42
        (NounFn Five-TheWord ) )
      (isa ?COSTUMES42 Costume ) )
    (hasEvaluativeQuantity ?COSTUMES42
```

```
      (MediumToVeryHighAmountFn Goodness-Generic ) ) )
  (implies
    (isa ?PRIZES65 AwardPractice )
    (implies
      (isa ?GOT45
        (VerbFn Get-TheWord ) )
      (eventSOT ?GOT45 ?COSTUMES42 ?PRIZES65 Past ) ) ) )

e2c("five other costumes got prizes").

[[txt, five], [tag, staart, cd, 'CountNoun', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB',
'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, other], [tag, z, jj, ap, 'Adjective', 'CountNoun']]
[[txt, costumes], [tag, z, nns, 'CountNoun']]
[[txt, got], [tag, z, vbd, vbn]]
[[txt, prizes], [tag, laast, nns, 'CountNoun']]

seg([p('QP', 1, 1), p('PP_non_cyclic', 2, 2), p('QP', 3, 3), p('VP_non_cyclic', 4, 4), p('QP', 5, 5)])

(thereExistExactly 5 ?FIVE18
  (and
    (and
      (and
        (isa ?FIVE18 Costume )
        (isa ?FIVE18 Thing ) )
      (isa ?FIVE18 BIOS-OTHERSurveillanceSyndrome ) )
    (implies
      (isa ?PRIZES73 AwardPractice )
      (implies
        (isa ?GOT52
          (VerbFn Get-TheWord ) )
        (eventSOT ?GOT52 ?FIVE18 ?PRIZES73 Past ) ) ) ) )

e2c("the other five costumes got prizes").

[[txt, a], [tag, staart, dt, at, 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB', 'Determiner-ClassC',
'Determiner-Indefinite', 'Postdeterminer', 'WHDeterminer', 'CountNoun', 'MassNoun', 'MassNoun-Generic']]
[[txt, other], [tag, z, jj, ap, 'Adjective', 'CountNoun']]
[[txt, five], [tag, z, cd, 'CountNoun', 'Determiner', 'Determiner-Central', 'Determiner-ClassA', 'Determiner-ClassB',
'Determiner-ClassC', 'ExpletivePronoun', 'IndefinitePronoun', 'Number-SP', 'ObjectPronoun', 'PossessivePronoun',
'PossessivePronoun-Post', 'Postdeterminer', 'Pronoun', 'Pronoun-SubjectOrObject', 'ReciprocalPronoun', 'ReflexivePronoun',
'SubjectPronoun', 'WHDeterminer', 'WHPronoun']]
[[txt, costumes], [tag, z, nns, 'CountNoun']]
[[txt, got], [tag, z, vbd, vbn]]
[[txt, prizes], [tag, laast, nns, 'CountNoun']]

seg([p('QP', 1, 2), p('QP', 3, 4), p('VP_non_cyclic', 5, 5), p('QP', 6, 6)])

(thereExistExactly 5 ?FIVE19
  (and
    (and
      (isa ?FIVE19 Costume )
      (isa ?FIVE19 Thing ) )
    (implies
      (isa ?PRIZES81 AwardPractice )
```

```
      (implies
        (isa ?GOT59
          (VerbFn Get-TheWord ) )
        (eventSOT ?GOT59 ?FIVE19 ?PRIZES81 Past ) ) ) ) )

e2c("other five costumes got prizes").

(implies
  (and
    (and
      (isa ?OTHER12
        (NounFn Five-TheWord ) )
      (isa ?OTHER12 Costume ) )
    (isa ?OTHER12 BIOS-OTHERSurveillanceSyndrome ) )
  (implies
    (isa ?PRIZES89 AwardPractice )
    (implies
      (isa ?GOT62
        (VerbFn Get-TheWord ) )
      (eventSOT ?GOT62 ?OTHER12 ?PRIZES89 Past ) ) ) )


e2c("some other costumes were really bad").


(thereExists ?OTHER52
  (and
    (and
      (and
        (isa ?OTHER52 Costume )
        (isa ?OTHER52 Thing ) )
      (isa ?OTHER52 BIOS-OTHERSurveillanceSyndrome ) )
    (implies
      (and
        (isa ?BAD14
          (NounFn Real-TheWord ) )
        (isa ?BAD14
          (OneOfFn Badness-Generic Badness-Moral ) ) )
      (and
        (isa ?WERE6 Living )
        (occursDuring ?WERE6 Past )
        (preActors ?WERE6 ?OTHER52 )
        (actors ?WERE6 ?BAD14 ) ) ) ) )



English Parser

   Show All
```

WHAT CYC ended up doing good

```
the system queries for instance "you love shooting" ?- why_not_wff(loves(cyclbot_1,shoot),X). % result:  X = ~isa(shoot,lovable).

['Hypothetic-I', 'Hypothetic-The-Arabian-Coffee', 'Hypothetic-Drank']
```

Please wait..

Assertion

```
Variable Quantification
Or Constant Name    Refactoring To Customize The Assertion (Persistent)
 Hypothetic-I


 Hypothetic-The-Arabian-Coffee


 Hypothetic-Drank


  (situationConstituents Hypothetic-GenitiveFrame Hypothetic-I)
 T
 (subEvents Hypothetic-GenitiveFrame Hypothetic-Drank)
 T
 (actors Hypothetic-Drank Hypothetic-The-Arabian-Coffee)
 T
 (preActors Hypothetic-Drank Hypothetic-I)
 T
 (isa Hypothetic-Drank DrinkingEvent)
 T
 (performedBy Hypothetic-Drank Hypothetic-I)
 "Formula
  (#$performedBy #$Hypothetic-Drank #$Hypothetic-I)
was not well formed because:
Term #$Hypothetic-I violates arg-isa #$Agent-Generic
applicable to argument 2 of relation #$performedBy
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
Term #$Hypothetic-I violates arg-isa #$Agent-Generic
applicable to argument 2 of relation #$performedBy
(via #$deliberateActors)
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
Term #$Hypothetic-I violates arg-isa #$Agent-Generic
applicable to argument 2 of relation #$performedBy
(via #$awareOf)
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
"
 (primaryObjectMoving Hypothetic-Drank Hypothetic-I)
 "Formula
  (#$primaryObjectMoving #$Hypothetic-Drank #$Hypothetic-I)
was not well formed because:
Term #$Hypothetic-I violates inter-arg-isa #$FluidTangibleThing
applicable to argument 2 of relation #$primaryObjectMoving
when argument 1, #$Hypothetic-Drank, isa #$FluidFlowEvent
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
Term #$Hypothetic-I violates inter-arg-isa #$FluidTangibleThing
applicable to argument 2 of relation #$primaryObjectMoving
(via #$objectMoving)
when argument 1, #$Hypothetic-Drank, isa #$FluidFlowEvent
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
Term #$Hypothetic-I violates inter-arg-isa #$FluidTangibleThing
```

```
applicable to argument 2 of relation #$primaryObjectMoving
when argument 1, #$Hypothetic-Drank, isa #$FluidFlow-Translation
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
Term #$Hypothetic-I violates arg-isa #$SpatialThing
applicable to argument 2 of relation #$primaryObjectMoving
(via #$objectMoving)
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
Term #$Hypothetic-I violates arg-isa #$PartiallyTangible
applicable to argument 2 of relation #$primaryObjectMoving
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
"
 (objectMoving Hypothetic-Drank Hypothetic-I)
 "Formula
  (#$objectMoving #$Hypothetic-Drank #$Hypothetic-I)
was not well formed because:
Term #$Hypothetic-I violates inter-arg-isa #$FluidTangibleThing
applicable to argument 2 of relation #$objectMoving
when argument 1, #$Hypothetic-Drank, isa #$FluidFlowEvent
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
Term #$Hypothetic-I violates arg-isa #$SpatialThing
applicable to argument 2 of relation #$objectMoving
in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the arabian coffee .\"))>.
"
 (consumedObject Hypothetic-Drank Hypothetic-The-Arabian-Coffee)
 T
 (isa Hypothetic-Drank Event)
 T
 (isa Hypothetic-Drank Situation)
 T
 (termStrings Hypothetic-Drank "drank")
 T
 (occursDuring Hypothetic-Drank
      (IntervalBeforeFn Now AnIndefiniteAmountOfTime))
 T
 (isa Hypothetic-The-Arabian-Coffee Individual)
 T
 (isa Hypothetic-The-Arabian-Coffee SomethingExisting)
 T
 (termStrings Hypothetic-The-Arabian-Coffee "the arabian coffee")
 T
 (isa Hypothetic-The-Arabian-Coffee Coffee-Ground)
 T
 (isa Hypothetic-The-Arabian-Coffee Coffee-Beverage)
 T
 (equals Hypothetic-The-Arabian-Coffee ArabianPeninsula)
 T
 (isa Hypothetic-The-Arabian-Coffee ArabianHorse)
 "Formula
  (#$isa #$Hypothetic-The-Arabian-Coffee #$ArabianHorse)
was not well formed because:
#$Hypothetic-The-Arabian-Coffee is known not to be an instance of #$ArabianHorse in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn
#$NLTESTMT \"I drank the arabian coffee .\"))>.
sbhl conflict: (#$isa #$Hypothetic-The-Arabian-Coffee #$ArabianHorse) TRUE #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I
drank the arabian coffee .\"))>
      because: (#$isa #$Hypothetic-The-Arabian-Coffee #$Coffee-Beverage) True-JustificationTruth
(#$genls #$Coffee-Beverage #$Drink) TRUE
(#$genls #$Drink #$LiquidTangibleThing) TRUE
```

```
(#$genls #$LiquidTangibleThing #$FluidTangibleThing) TRUE
(#$disjointWith #$FluidTangibleThing #$Agent-PartiallyTangible) TRUE
(#$genls #<(#$CollectionUnionFn (#$TheSet #$Person #$Animal))> #$Agent-PartiallyTangible) TRUE
(#$genls #$Animal #<(#$CollectionUnionFn (#$TheSet #$Person #$Animal))>) TRUE
(#$genls #$NonHumanAnimal #$Animal) TRUE
(#$genls #$EquineAnimal #$NonHumanAnimal) TRUE
(#$genls #$Horse #$EquineAnimal) TRUE
(#$genls #$Horse-Domesticated #$Horse) TRUE
(#$genls #$ArabianHorse #$Horse-Domesticated) TRUE

"
 (conceptuallyRelated Hypothetic-The-Arabian-Coffee ArabianPeninsula)
 "Formula
  (#$conceptuallyRelated #$Hypothetic-The-Arabian-Coffee #$ArabianPeninsula)
was not well formed because:
(#$isa #$conceptuallyRelated #$IrreflexiveBinaryPredicate) in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I drank the
arabian coffee .\"))>
"
 (isa Hypothetic-I Individual)
 T
 (properNameStrings Hypothetic-I "I")
 "Formula
  (#$properNameStrings #$Hypothetic-I \"I\")
was not well formed because:
Asserting a literal with predicate #$properNameStrings is prohibited in mt #<(#$ParseMtForSourceFn (#$StringInDocumentFn #$NLTESTMT \"I
drank the arabian coffee .\"))>.
"
 (equals Hypothetic-I
      (PronounFn FirstPerson-NLAttr Singular-NLAttr Ungendered-NLAttr SubjectPronoun))
 T
 (isa Hypothetic-I SomethingExisting)
 T
 (termStrings Hypothetic-I "I")
 T
 (equals Hypothetic-I Hypothetic-I)
 T
 (isa Hypothetic-GenitiveFrame Event)
 T
```

```
compareForExtent(M,P,Which,What,How) <-
    isa(M,Which), isa(M,Which),
    t(What,M,MH),t(What,P,PH),
    t(How,MH,PH), ~t(How,PH,MH).

adjectiveExpansion(xtTheWordFn("tallest"),Type,X) ==>  compareForExtent(X,_Y,Type,mudHeight,greaterThan).
adjectiveExpansion(xtTheWordFn("shortest"),Type,X) ==> compareForExtent(X,_Y,Type,mudHeight,lessThan).

relationExpansion(xtTheWordFn("in"),X,Y) ==>  objectFoundInLocation(X,Y).
```

```
relationExpansion(xtTheWordFn("marry"),X,Y) ==>
   isa(S,tRelationshipType),
   mudKeyword(S,"marry"),
   t(S,J,M).


"Janus wants to marry the tallest man in Folsum County."
istAsserted(
 exists(J,
    and(
    txtDenotes("Janus",J),
    wants(J,
     exists([Relates,Man,Place],
       objectFoundInLocation(Man,Place),
       % objectFoundInLocation(Rest,Place),
       compareForExtent(Man,Rest,tHumanMale,mudHeight,greaterThan),
       denotesRelation(xtTheWordFn("marry"),Relates),
       t(Relates,J,Man))))).
```

Broken down from:

⇒

```
txtDenotes("Janus wants to marry the tallest man in Folsum County.",Sit),
istAsserted(Sit),

Sit =
 exists([J,Sit0,Sit1,Sit2,Sit3],
and(
   txtDenotes("Janus",J),
   txtDenotes("wants to",Sit0),
   wants(J,Sit0),
 Sit0 =
 and(
   txtDenotes("marry",Sit1),
   Sit1 = denotesRelation(xtTheWordFn("marry"),Relates), t(Relates,J,Man)),
   txtDenotes("the tallest man",Sit2),
   Sit2 = compareForExtent(Man,Rest,tHumanMale,mudHeight,greaterThan),
   txtDenotes("in Folsum County",Sit3),
   Sit3  =   objectFoundInLocation(Man,Place),
   txtDenotes("Folsum County",Place),
    Sit1,Sit2,Sit3))))).
```

The language below is CycL/KIF that the system uses to *build inside the virtual world.*. It is **not** what LM489 can "understand", These are examples used by the robotic Dungeon Master (executive interface) for setting up and debugging non-verbal experiences for LM489. The code that produced the Logical form below is already working well for developing new MUD games in which LM489 can barely tolerate at present.

Wouldn't it be neat one day if LM489 could learn to control it's logical environment? (That is become its own Dungeon Master) The closest it might come is speaking into the ear of the Dungeon Master phrases like: "I am happy". . Currently several roadblocks are in the way:

realize the Dungeon Master (DM) even exists
speak a the controlled english understood by DM
realize the DM weaves it's (LM489) logical reality

Jan 27 (3 days ago)

Pat Hayes

to ontolog-forum, John

Allow me to interject my reaction to "*The basic unit of the graph of graphs proposed by the Semantic Web is an RDF triple. Using this mathematical foundation, it is possible to represent all existing information in a given context and then to link this graph with other open data graphs.*" The problem is, as soon as you consider more expressive languages than RDF itself, but encoded in RDF (such as RDFS or OWL/RDF), the 'basic unit' is not the triple, but rather smallish groups of RDF triples encoding 'higher-level' assertions. So for example, a single, simple, atomic assertion in OWL such as that hasBrother is the restriction of hasSibling to its range being in the class Male, has to be encoded in RDF as three triples. The issue here is not that one of these three might be false: it is that none of these three even make sense when taken in isolation, and if combined with something other than their intended RDF group, could completely change in its meaning. So, the above quote is correct when we are restricted to simple RDF data comprising 'ground' facts. But that is a severe, almost fatal, restriction: whatever all that data is, and however important, it can hardly be said to even constitute a simple ontology. So we are back to square one: how to combine blockchain ideas *with ontologies* as opposed to simply with data.