Round trips with meaning stopovers

(illustrated with English and Japanese examples)

Alastair Butler

Tohoku University

This talk describes a full pipeline

starting form an Historical Penn-treebank parse,

$$\Downarrow$$

semantic processing creates a standard predicate logic based meaning representation (see e.g., Davidson 1967; Dowty, Wall and Peters 1981),

$$\Downarrow$$

which is converted to PENMAN notation (Matthiessen and Bateman 1991) to form the basis for generation,

$$\Downarrow$$

which proceeds as structure growth producing an output parse tree which can yield a language string.

Keeping to the same language tests the combined success of building meaning representations from parsed input and of generating parsed output.

Switching languages during the round trip would achieve translation.

The method will first be illustrated by round tripping on English, so

- taking English parsed sentences,
- going to meaning representations,
- and then back to parsed sentences of English.

The front or back end of the pipeline can be changed, e.g., to calculate a meaning representation for English input but use Japanese generation rules.

No modification to the stopover meaning representation will return English words and concepts with Japanese parse structure (= Japanese word order with a yield).

Meaning representations arrived at from parsed parallel corpora show the shortfall left for generating sentences of one language from another.

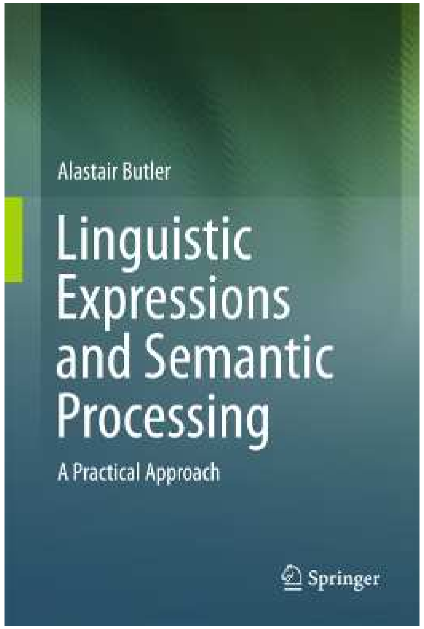
Reaching meaning representations

The first requirement is a way to reach meaning representations from natural language input.

There are many ways to go. E.g., Schubert (2015) overviews 12 distinct approaches, many with multiple implementations.

In what follows, use is made of **Treebank Semantics** (`http://www.compling.jp/ts`; Butler 2015),

- the same kind of parse tree to be generated as output can be taken as input,
- produced meaning representations are of high quality.



Treebank Semantics: Input trees are converted to SCT expressions (a Dynamic Semantics language) which are processed against a sequence based information state (cf. Vermeulen 2000, Dekker 2012).

```
1: Treebank Annotation

(IP-MAT (PP (NP (NPR 田中さん))
            (P が))
        (NP-SBJ *が*)
        (PP (NP (N ピザ))
            (P を))
        (NP-OB1 *を*)
        (VB 食べ)
        (AX まし)
        (AXD た)
        (PU 。))
```

```
2: SCT expression

val ex1 =
  ( fn fh =>
   ( fn lc =>
    ( ( npr "entity" "田中さん")
      "arg0"
      ( ( some lc fh "entity"
          ( nn lc "ピザ"))
        "arg1"
       ( past "event"
         ( verb lc "event" ["arg0", "arg1"]
             "食べ_まし_た")))))
   ["arg1", "arg0", "h"])
  ["constant", "entity", "event"]
```

```
3: Reduced SCT expression

Hide ("constant",
 CClose ("constant",
  Hide ("entity",
   Close ("∃", ("entity","entity"),
     Hide ("event",
      Close ("∃", ("event","event"),
      Clean (0, ["arg0"], "c",
       CUse (C
         Lam ("constant", "arg0",
          Clean (0, ["arg1"], "c",
            Use ("entity",
            Lam ("entity", "arg1",
             ...
```

4: Meaning Representation Output

$$\exists x_1 e_2 (\text{ピザ}(x_1) \wedge \text{past}(e_2) \wedge \text{食べ\_まし\_た}(e_2, \text{田中さん}, x_1))$$

Treebank annotation

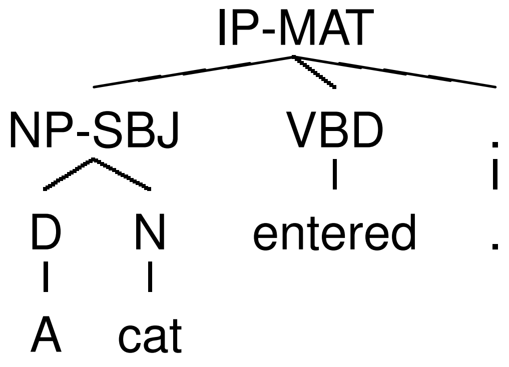
The Treebank Semantics system accepts parsed data conforming to the *Annotation manual for the Penn Historical Corpora and the PCEEC* (Santorini 2010).

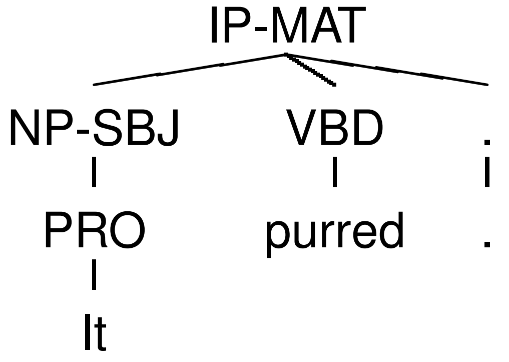
This widely and diversely applied scheme forms the basis of annotations for **English** (Taylor et al 2003, Kroch, Santorini and Delfs 2004, Kroch, Santorini and Delfs 2004), **French** (Martineau et al 2010), **Icelandic** (Wallenberg et al 2011), **Portuguese** (Galves and Britto 2002), **Ancient Greek** (Beck 2013), **Japanese** (Butler et al 2012), and **Chinese** (Zhou 2015) among other languages.

There are also parsing systems to produce annotated trees from raw language input (e.g., Kulick, Kroch and Santorini 2014, Fang, Butler and Yoshimoto 2014).

With the annotation scheme constituent structure is represented with labelled bracketing and augmented with grammatical functions.

Parse trees for “*A cat entered. It purred.*” look like:





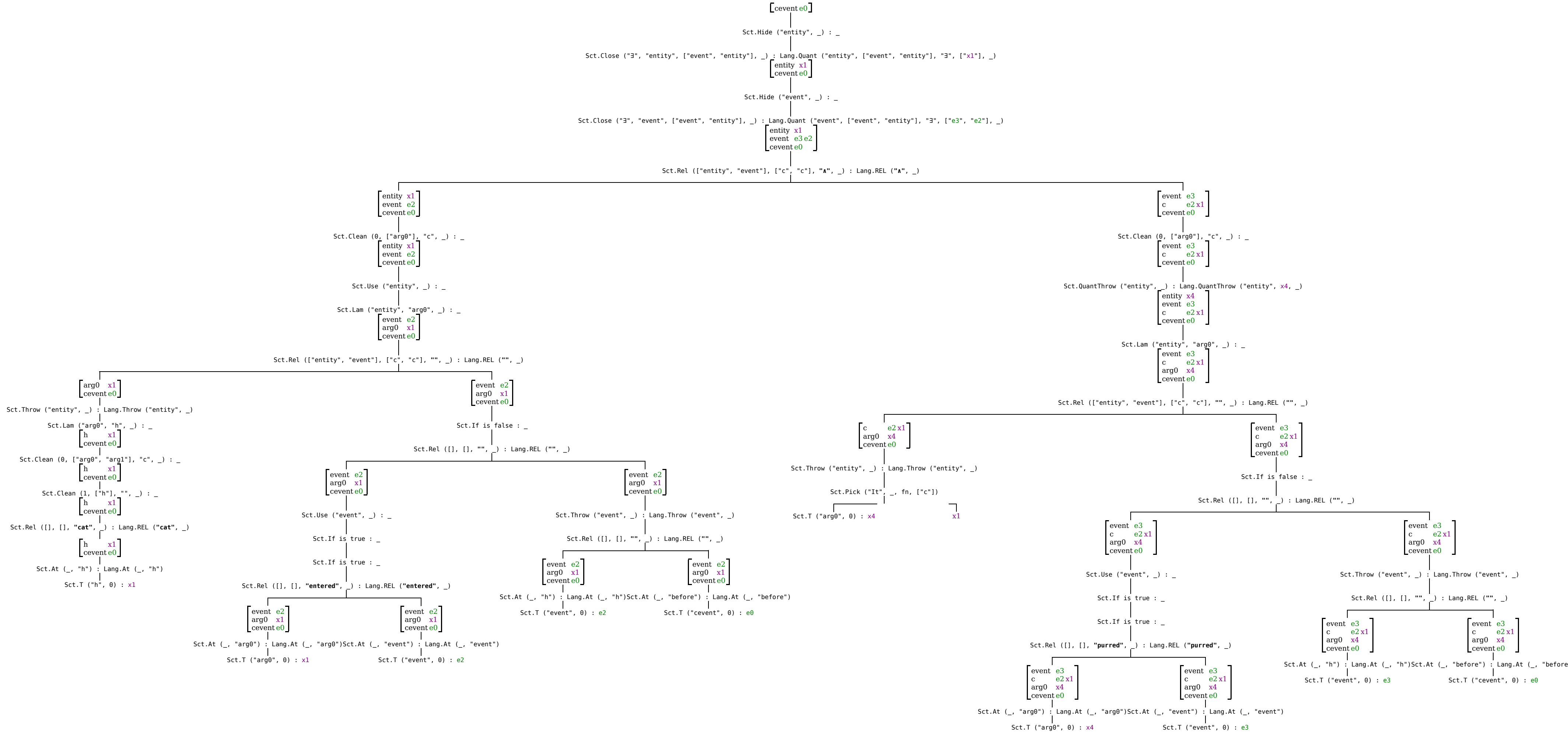
First step: convert trees for “*A cat entered. It purred.*” into SCT expressions

```
val sent1 =
( fn fh =>
  ( fn lc =>
    ( some lc fh "entity"
      ( nn lc "cat")
      "arg0"
      ( past "event"
        ( verb lc "event" ["arg0"] "entered"))))
  ["arg0", "arg1", "h"])
["entity", "event"]
;
val sent2 =
( fn fh =>
  ( fn lc =>
    ( pro ["c"] fh ["entity"] ( "entity", "entity") "It"
      "arg0"
      ( past "event"
        ( verb lc "event" ["arg0"] "purred"))))
  ["arg0", "arg1", "h"])
["entity", "event"]
;
val discourse =
( fn fh =>
  ( conj fh "∧" free
    [sent1, sent2]))
["entity", "event"]
;
```

`discourse` of “*A cat entered. It purred.*” as a fully resolved SCT expression:

```
discourse =
Sct.Hide ("entity",
 Sct.Close ("∃", ("entity","entity"),["event", "entity"],
  Sct.Hide ("event",
   Sct.Close ("∃", ("event","event"),["event", "entity"],
    Sct.Rel (["entity", "event"], ["c", "c"], "∧", [
     Sct.Clean (0, ["arg0"], "c",
      Sct.Use ("entity",
       Sct.Lam ("entity", "arg0",
        Sct.Rel (["entity", "event"], ["c", "c"], "", [
         Sct.Throw ("entity",
          Sct.Lam ("arg0", "h",
           Sct.Clean (0, ["arg0", "arg1"], "c",
            Sct.Clean (1, ["h"], "", Sct.Rel ([], [], "cat", [Sct.At (T ("h", 0), "h")]))))),
         Sct.If (fn,
          Sct.Use ("event",
           Sct.If (fn,
            Sct.If (fn,
             Sct.Rel ([], [], "entered", [Sct.At (T ("arg0", 0), "arg0"),
              Sct.At (T ("event", 0), "event")]),
             Sct.Rel ([], [], "entered", [Sct.At (T ("arg0", 0), "arg0"),
              Sct.At (T ("h", 0), "h"), Sct.At (T ("event", 0), "event")])),
            Sct.If (fn,
             Sct.Rel ([], [], "entered", [Sct.At (T ("arg0", 0), "arg0"),
              Sct.At (T ("arg1", 0), "arg1"), Sct.At (T ("event", 0), "event")]),
             Sct.Rel ([], [], "entered", [Sct.At (T ("arg0", 0), "arg0"),
              Sct.At (T ("arg1", 0), "arg1"), Sct.At (T ("h", 0), "h"),
              Sct.At (T ("event", 0), "event")])))),
          Sct.Rel ([], [], "", [
           Sct.Use ("event",
            Sct.If (fn,
             Sct.If (fn,
              Sct.Rel ([], [], "entered", [Sct.At (T ("arg0", 0), "arg0"),
               Sct.At (T ("event", 0), "event")]),
              Sct.Rel ([], [], "entered", [Sct.At (T ("arg0", 0), "arg0"),
               Sct.At (T ("h", 0), "h"), Sct.At (T ("event", 0), "event")])),
             Sct.If (fn,
              Sct.Rel ([], [], "entered", [Sct.At (T ("arg0", 0), "arg0"),
               Sct.At (T ("arg1", 0), "arg1"), Sct.At (T ("event", 0), "event")]),
              Sct.Rel ([], [], "entered", [Sct.At (T ("arg0", 0), "arg0"),
               Sct.At (T ("arg1", 0), "arg1"), Sct.At (T ("h", 0), "h"),
               Sct.At (T ("event", 0), "event")])))),
           Sct.Throw ("event",
            Sct.Rel ([], [], "", [Sct.At (T ("event", 0), "h"),
             Sct.At (T ("cevent", 0), "before")]))])))]))))),
     Sct.Clean (0, ["arg0"], "c",
      Sct.QuantThrow ( ("entity","entity"),
       Sct.Lam ("entity", "arg0",
        Sct.Rel (["entity", "event"], ["c", "c"], "", [Sct.Throw ("entity", Sct.Pick ("It", T ("arg0", 0), ["c"])),
         Sct.If (fn,
          Sct.Use ("event",
           Sct.If (fn,
            Sct.If (fn,
             Sct.Rel ([], [], "purred", [Sct.At (T ("arg0", 0), "arg0"),
              Sct.At (T ("event", 0), "event")]),
             Sct.Rel ([], [], "purred", [Sct.At (T ("arg0", 0), "arg0"),
              Sct.At (T ("h", 0), "h"), Sct.At (T ("event", 0), "event")])),
            Sct.If (fn,
             Sct.Rel ([], [], "purred", [Sct.At (T ("arg0", 0), "arg0"),
              Sct.At (T ("arg1", 0), "arg1"), Sct.At (T ("event", 0), "event")]),
             Sct.Rel ([], [], "purred", [Sct.At (T ("arg0", 0), "arg0"),
              Sct.At (T ("arg1", 0), "arg1"), Sct.At (T ("h", 0), "h"),
              Sct.At (T ("event", 0), "event")])))),
          Sct.Rel ([], [], "", [
           Sct.Use ("event",
            Sct.If (fn,
             Sct.If (fn,
              Sct.Rel ([], [], "purred", [Sct.At (T ("arg0", 0), "arg0"),
               Sct.At (T ("event", 0), "event")]),
              Sct.Rel ([], [], "purred", [Sct.At (T ("arg0", 0), "arg0"),
               Sct.At (T ("h", 0), "h"), Sct.At (T ("event", 0), "event")])),
             Sct.If (fn,
              Sct.Rel ([], [], "purred", [Sct.At (T ("arg0", 0), "arg0"),
               Sct.At (T ("arg1", 0), "arg1"), Sct.At (T ("event", 0), "event")]),
              Sct.Rel ([], [], "purred", [Sct.At (T ("arg0", 0), "arg0"),
               Sct.At (T ("arg1", 0), "arg1"), Sct.At (T ("h", 0), "h"),
               Sct.At (T ("event", 0), "event")])))),
           Sct.Throw ("event",
            Sct.Rel ([], [], "", [Sct.At (T ("event", 0), "h"),
             Sct.At (T ("cevent", 0), "before")]))])))]))))])))))
```

The SCT language primitives access and possibly alter the content of a sequence based information state that serves to retain binding information by assigning (possibly empty) sequences of values to binding names.


[cevent e0]
Sct.Hide ("entity", _) : _
Sct.Close ("∃", "entity", ["event", "entity"], _) : Lang.Quant ("entity", ["event", "entity"], "∃", ["x1"], _)
[entity x1 / cevent e0]
Sct.Hide ("event", _) : _
Sct.Close ("∃", "event", ["event", "entity"], _) : Lang.Quant ("event", ["event", "entity"], "∃", ["e3", "e2"], _)
[entity x1 / event e3 e2 / cevent e0]
Sct.Rel (["entity", "event"], ["c", "c"], "∧", _) : Lang.REL ("∧", _)
[entity x1 / event e2 / cevent e0]
Sct.Clean (0, ["arg0"], "c", _) : _
[entity x1 / event e2 / cevent e0]
Sct.Use ("entity", _) : _
Sct.Lam ("entity", "arg0", _) : _
[event e2 / arg0 x1 / cevent e0]
Sct.Rel (["entity", "event"], ["c", "c"], "", _) : Lang.REL ("", _)
[arg0 x1 / cevent e0]
Sct.Throw ("entity", _) : Lang.Throw ("entity", _)
Sct.Lam ("arg0", "h", _) : _
[h x1 / cevent e0]
Sct.Clean (0, ["arg0", "arg1"], "c", _) : _
[h x1 / cevent e0]
Sct.Clean (1, ["h"], "", _) : _
[h x1 / cevent e0]
Sct.Rel ([], [], "cat", _) : Lang.REL ("cat", _)
[h x1 / cevent e0]
Sct.At (_, "h") : Lang.At (_, "h")
Sct.T ("h", 0) : x1
[event e2 / arg0 x1 / cevent e0]
Sct.If is false : _
Sct.Rel ([], [], "", _) : Lang.REL ("", _)
[event e2 / arg0 x1 / cevent e0]
Sct.Use ("event", _) : _
Sct.If is true : _
Sct.If is true : _
Sct.Rel ([], [], "entered", _) : Lang.REL ("entered", _)
[event e2 / arg0 x1 / cevent e0]
Sct.At (_, "arg0") : Lang.At (_, "arg0")
Sct.T ("arg0", 0) : x1
[event e2 / arg0 x1 / cevent e0]
Sct.At (_, "event") : Lang.At (_, "event")
Sct.T ("event", 0) : e2
[event e2 / arg0 x1 / cevent e0]
Sct.Throw ("event", _) : Lang.Throw ("event", _)
Sct.Rel ([], [], "", _) : Lang.REL ("", _)
[event e2 / arg0 x1 / cevent e0]
Sct.At (_, "h") : Lang.At (_, "h")
Sct.T ("event", 0) : e2
[event e2 / arg0 x1 / cevent e0]
Sct.At (_, "before") : Lang.At (_, "before")
Sct.T ("cevent", 0) : e0
[event e3 / c e2 x1 / cevent e0]
Sct.Clean (0, ["arg0"], "c", _) : _
[event e3 / c e2 x1 / cevent e0]
Sct.QuantThrow ("entity", _) : Lang.QuantThrow ("entity", x4, _)
[entity x4 / event e3 / c e2 x1 / cevent e0]
Sct.Lam ("entity", "arg0", _) : _
[event e3 / c e2 x1 / arg0 x4 / cevent e0]
Sct.Rel (["entity", "event"], ["c", "c"], "", _) : Lang.REL ("", _)
[c e2 x1 / arg0 x4 / cevent e0]
Sct.Throw ("entity", _) : Lang.Throw ("entity", _)
Sct.Pick ("It", _, fn, ["c"])
Sct.T ("arg0", 0) : x4
x1
[event e3 / c e2 x1 / arg0 x4 / cevent e0]
Sct.If is false : _
Sct.Rel ([], [], "", _) : Lang.REL ("", _)
[event e3 / c e2 x1 / arg0 x4 / cevent e0]
Sct.Use ("event", _) : _
Sct.If is true : _
Sct.If is true : _
Sct.Rel ([], [], "purred", _) : Lang.REL ("purred", _)
[event e3 / c e2 x1 / arg0 x4 / cevent e0]
Sct.At (_, "arg0") : Lang.At (_, "arg0")
Sct.T ("arg0", 0) : x4
[event e3 / c e2 x1 / arg0 x4 / cevent e0]
Sct.At (_, "event") : Lang.At (_, "event")
Sct.T ("event", 0) : e3
[event e3 / c e2 x1 / arg0 x4 / cevent e0]
Sct.Throw ("event", _) : Lang.Throw ("event", _)
Sct.Rel ([], [], "", _) : Lang.REL ("", _)
[event e3 / c e2 x1 / arg0 x4 / cevent e0]
Sct.At (_, "h") : Lang.At (_, "h")
Sct.T ("event", 0) : e3
[event e3 / c e2 x1 / arg0 x4 / cevent e0]
Sct.At (_, "before") : Lang.At (_, "before
Sct.T ("cevent", 0) : e0

Reading off accumulated information from the evaluation gives:

$$
\begin{array}{l}
\exists x_4 x_1 e_2 e_3\,(\\
\quad \texttt{past}(e_2) \wedge \\
\qquad \texttt{past}(e_3) \wedge \\
\qquad\quad \text{cat}(x_1) \wedge \\
\qquad\qquad x_4 = \text{It}\{x_1\} \wedge \text{entered}(e_2,\ x_1) \wedge \text{purred}(e_3,\ x_4))
\end{array}
$$

This assumes a Davidsonian theory (Davidson 1967) in which verbs are encoded with minimally an implicit event argument which is existentially quantified over and may be further modified.

This encodes truth-conditional content in a standard way, but also contains clues to assist generation. Most notably variables have sort information, thus: $e_1, e_2, \ldots$ are events, while $x_1, x_2, \ldots$ are objects, etc. Also, a candidate for the main predicate is the most deeply embedded right-side predicate.

Generation

The idea behind the approach to generation is, from a meaning representation presented as a tree, to follow a series of meaning preserving transformations to arrive back at a parse tree, that is, to a representation of the kind fed to the Treebank Semantics system at the start of the pipeline.

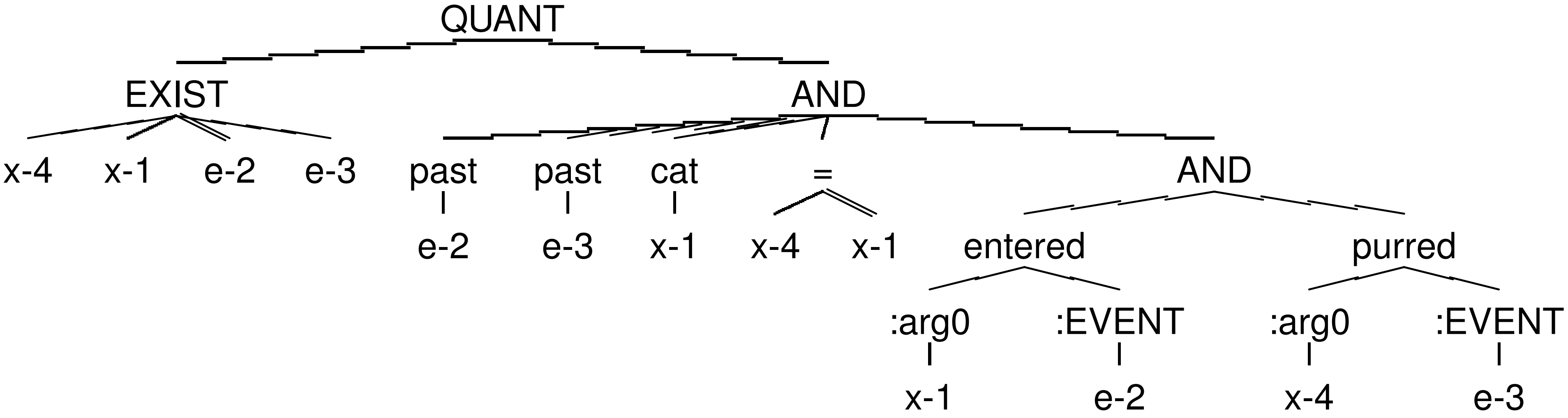
There are two major steps.

- First, there is preparation,
- then there is generation, as growing and manipulating tree structure.

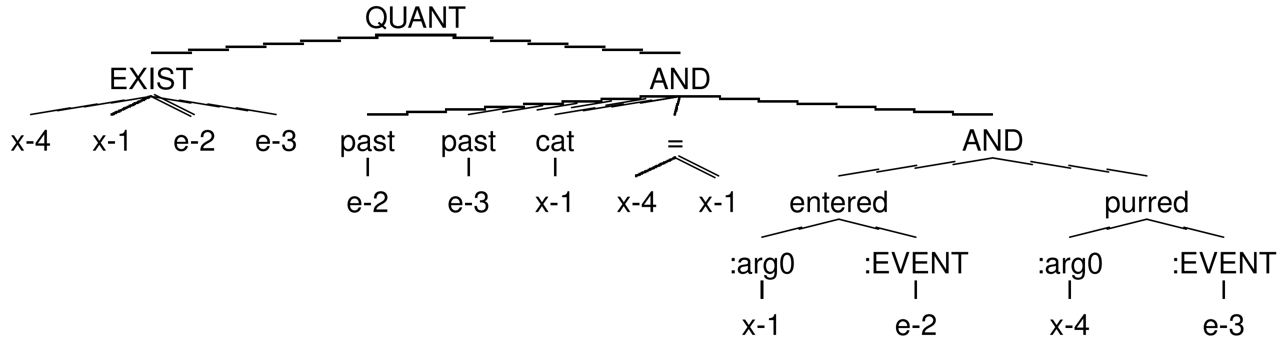
Preparing for generation

Preparation for generation involves obtaining a tree-based representation of the output produced by Treebank Semantics.

Rendering the meaning representation for “*A cat entered. It purred.*” as a tree with argument role information made explicit gives:

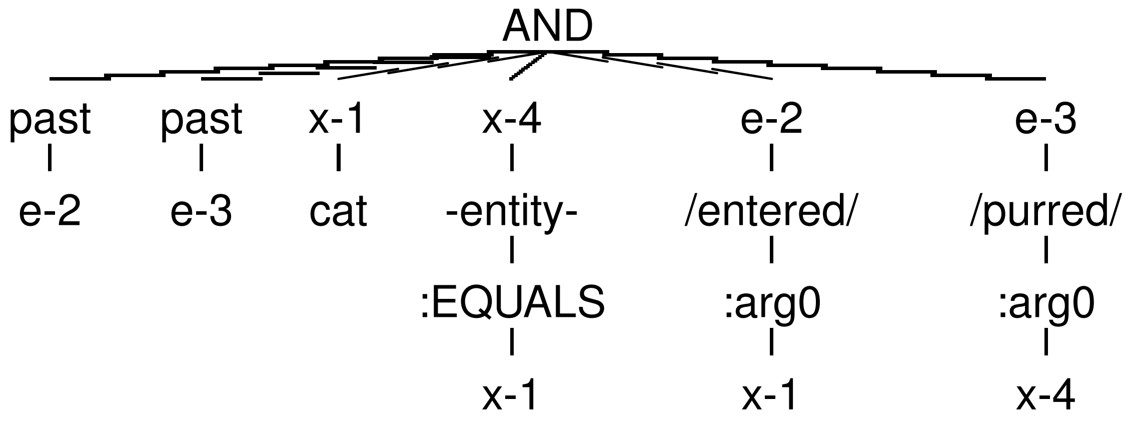


Content meeds to be further re-packaged to a tree format optimised for generation.



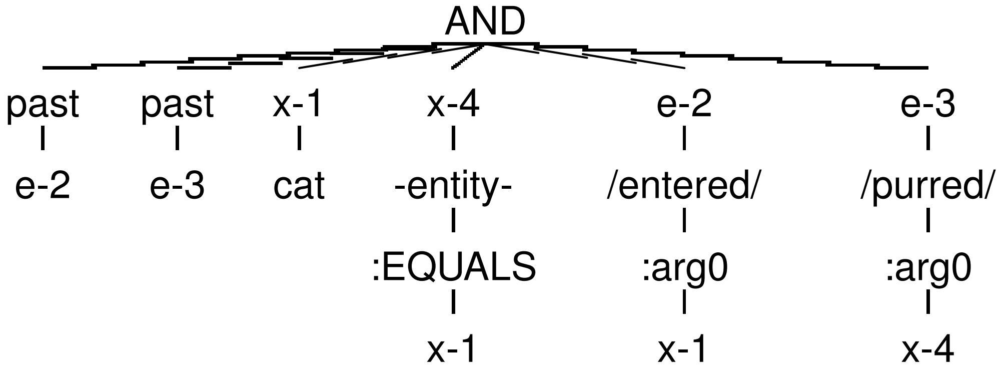
Firstly, the binding of wide-scope existentials is made implicit with the removal of the top quantification level.

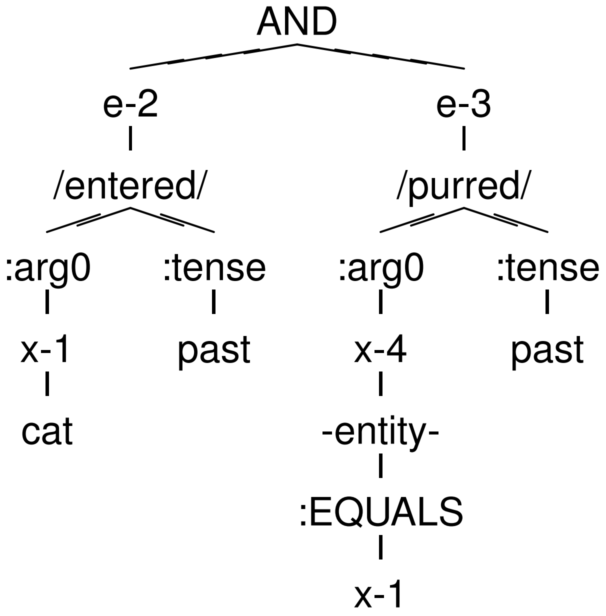
Next, an argument of each predicate is promoted to become the parent of the predicate, notably: the left-hand argument of an equality relation, or an `event` argument if present, or the sole argument of a one-place predicate.



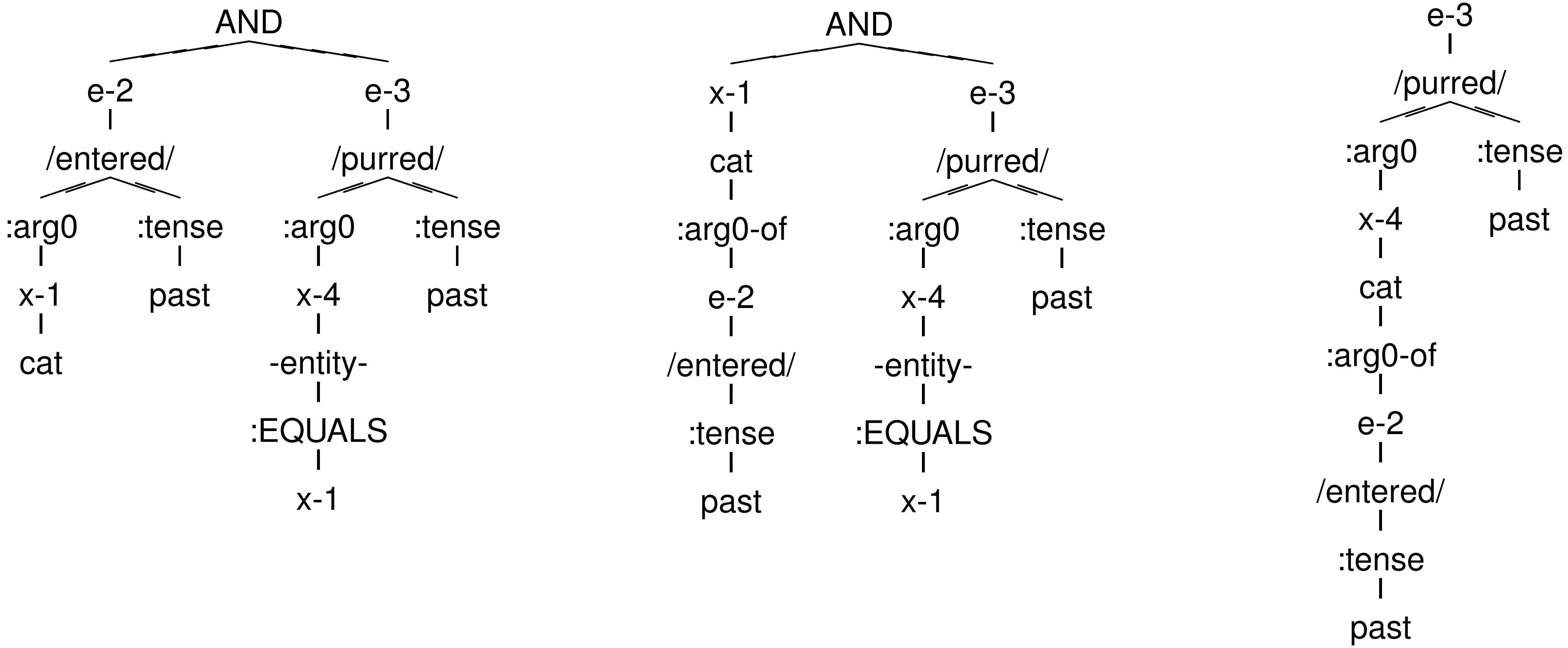
Next, tense information of the top level AND is relocated.

Next, a daughter D of the top level AND is moved inside a sister S when the name at the root of D is contained as an argument within S. Movement is to only one location (the left-most).





An internal argument is promoted to become root of a daughter of AND if this enables further inclusion into a sister. Promotion folds tree material around inverse roles from the PENMAN notation (Matthiessen and Bateman 1991).



Back to a parsed representation

Generation proceeds as a series of tree transformations, implemented as a tsurgeon script (Levy and Andrew 2006) with hundreds of transformation rules.

A tsurgeon script contains pattern/action rules, where the pattern describes tree structure and the action transforms the tree, e.g., moving, adjoining, copying or deleting auxiliary trees or relabelling nodes.

Transformations are repeatedly made until the pattern that triggers change is no longer matched.

An example tsurgeon rule

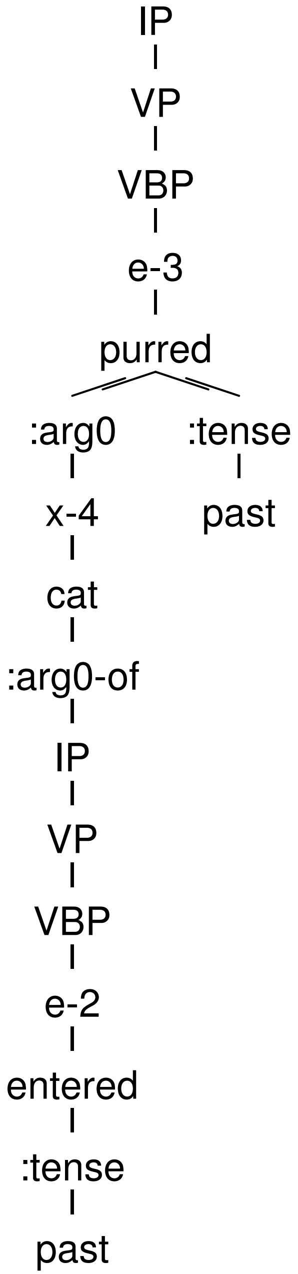
Clause structure is built by identifying a main predicate as being headed by an event variable (so: match *e*- followed by a number), and adjoining the projection of a VBP part-of-speech tag, a VP layer and an IP layer.

```
/^e-[0-9]+$/=x !> VBP
```

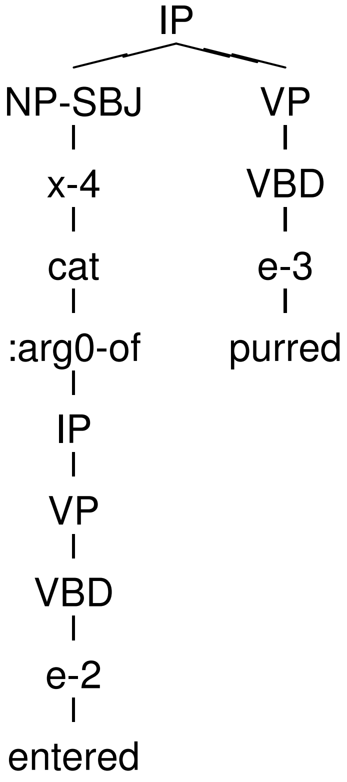
```
adjoinF (IP (VP (VBP @))) x
```

Action `adjoinF` adjoins the specified auxiliary tree into the specified target node, preserving the target node as the foot of the adjoined tree.

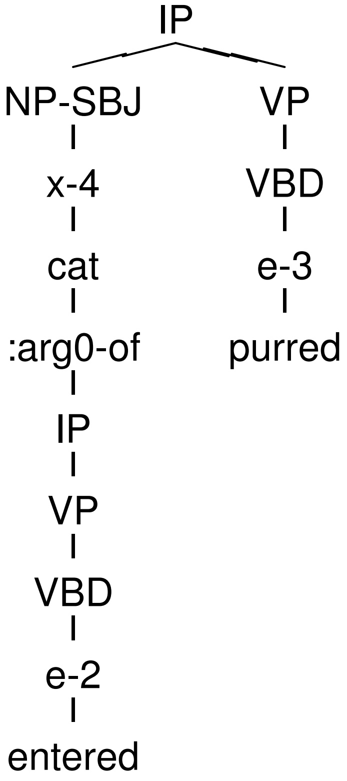
VBP (present tense verb) may subsequently change, e.g., `tense past` triggers change to VBD (past tense verb), while *was* when generating English brings about further change to BED (past tense copula).

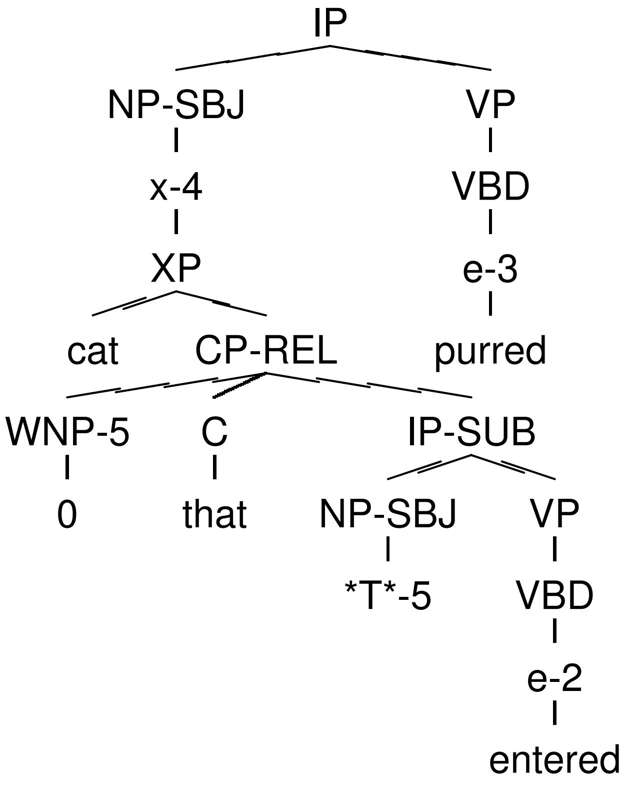


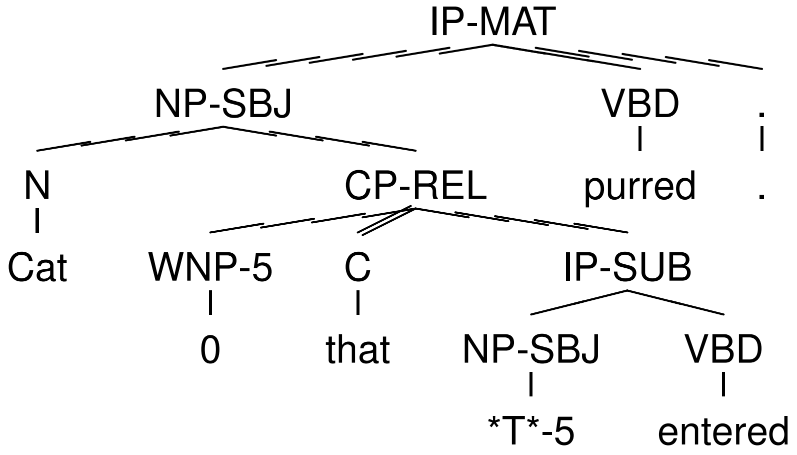
Subsequent changes involve moving all structure under a main predicate into the clause, starting with the creation of NP-SBJ from an `arg0` argument.



The inverse role `arg1-of` is the foundation for relative clause structure with an NP-SBJ (subject) trace.





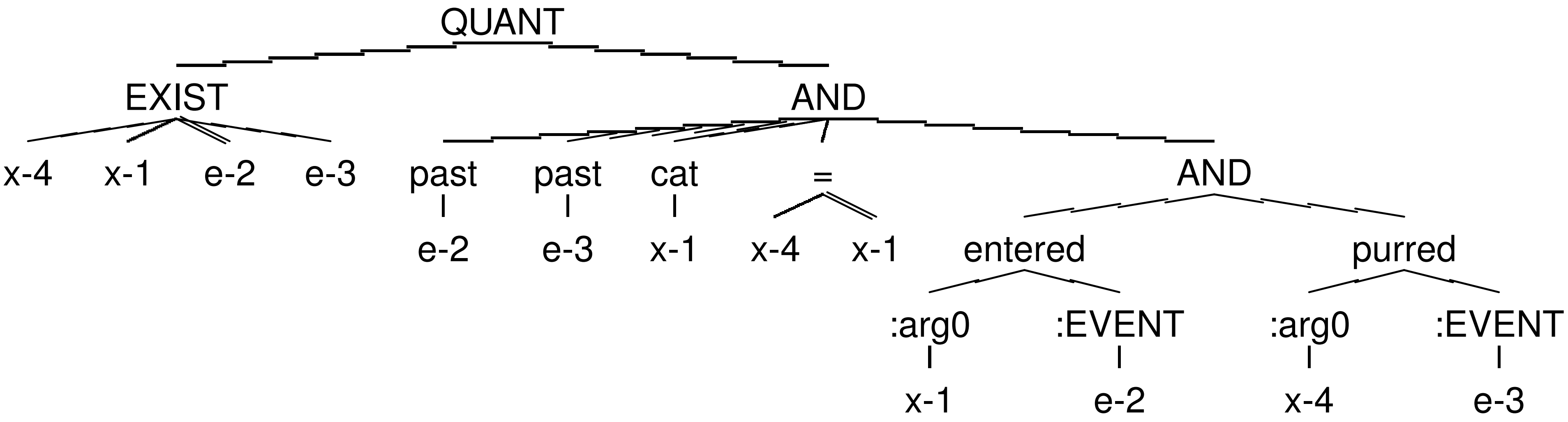


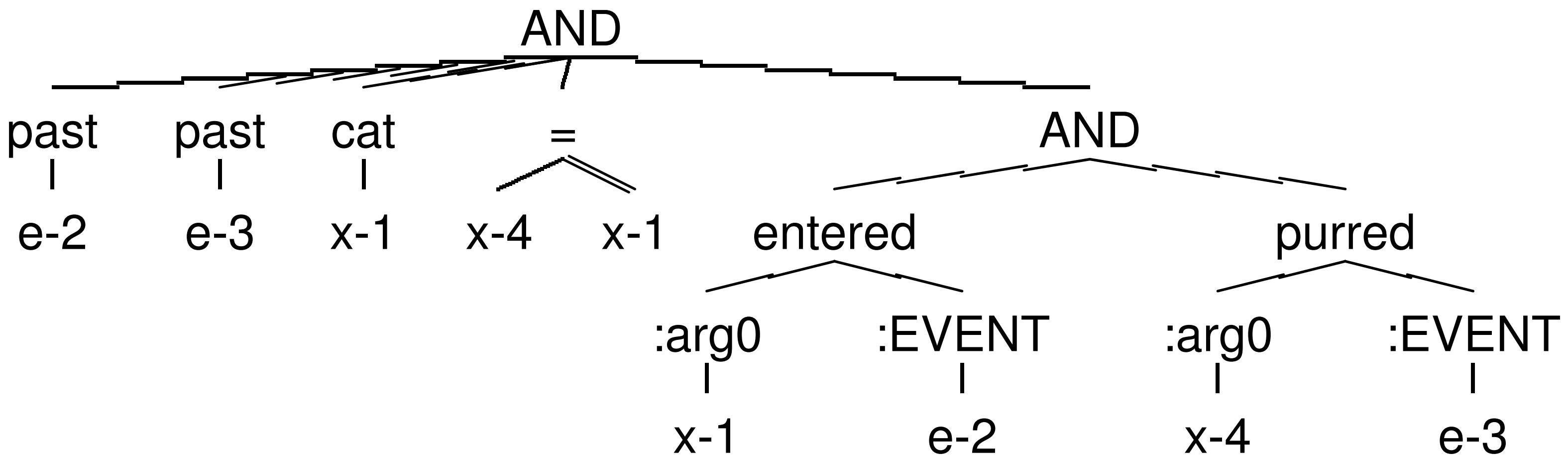
Generation possibilities

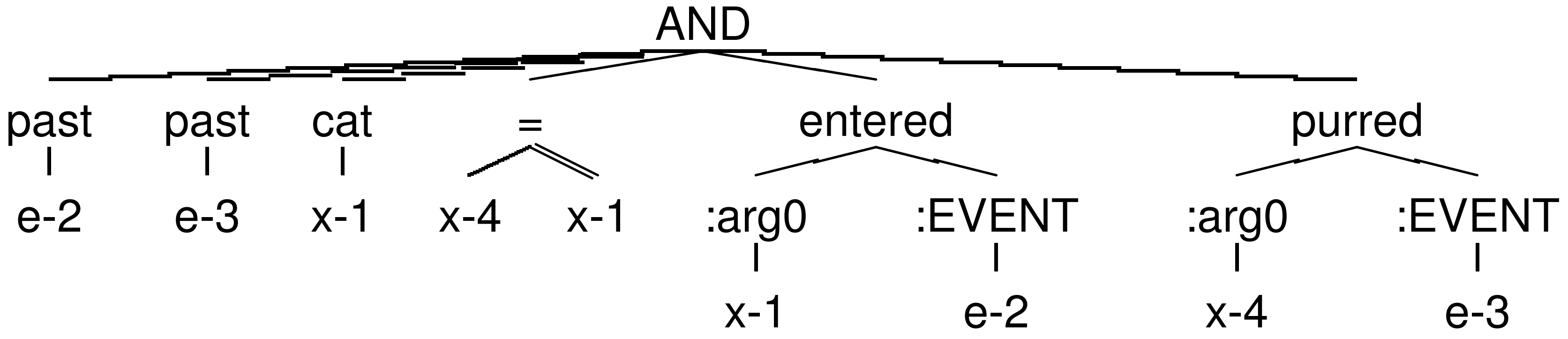
If an `arg0` argument happened to be missing, either a passive transformation may result or there may be inclusion of a subject expletive *it* or *there* for English.

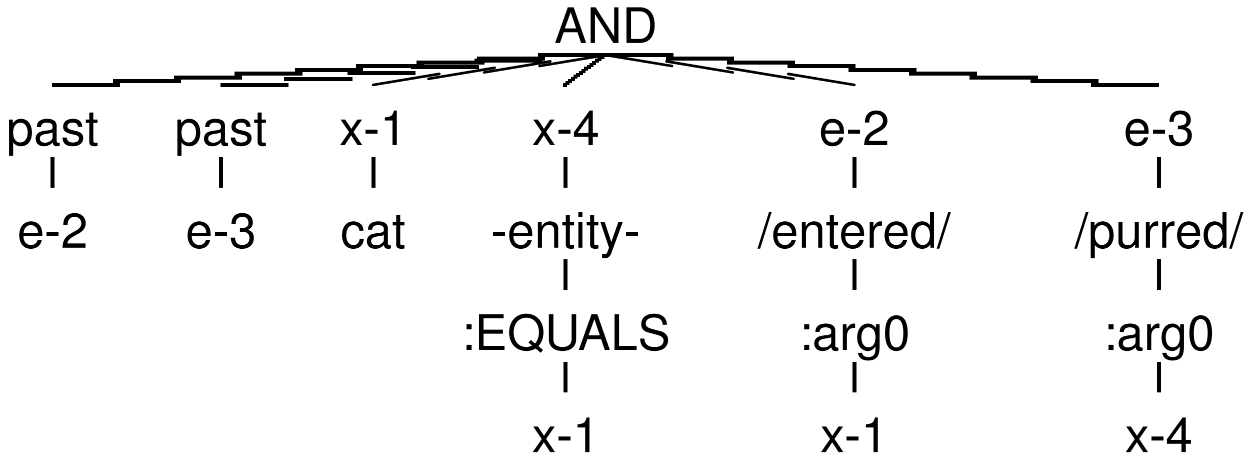
Adjunct materials can find placement based on argument role information or subtree size, e.g., vocatives (NP-VOC) are always clause initial, a temporal NP (NP-TMP) will typically be clause initial, while, for English, clause final positioning will be favoured for a heavy PP or NP (whose children reach large depths).

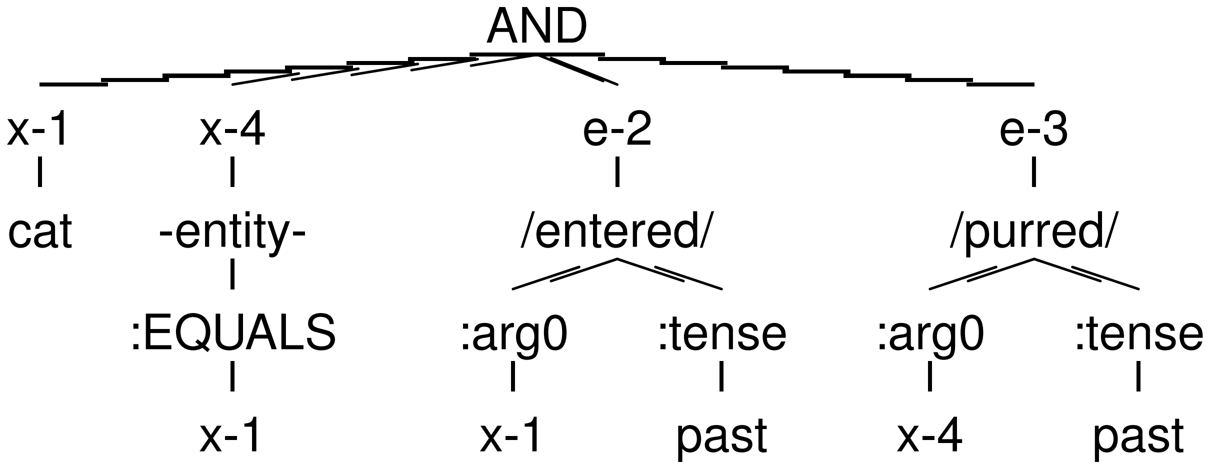
Having arguments with the same referent can trigger the introduction of infinitival or participial clause structure to create control configurations or various types of ellipsis, such as VP ellipsis.

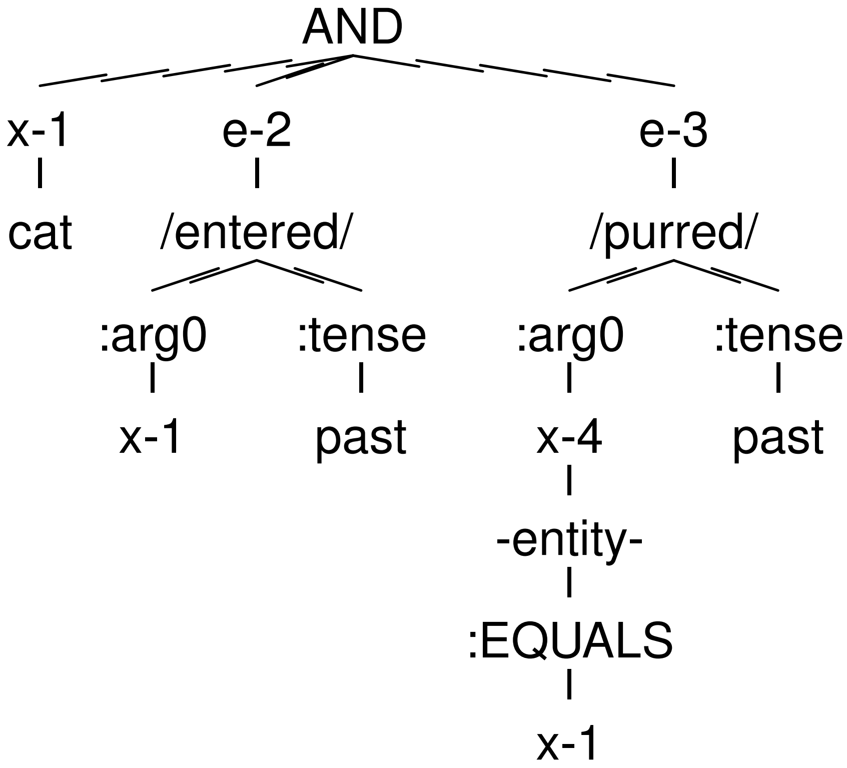

QUANT
EXIST
x-4
x-1
e-2
e-3
AND
past
e-2
past
e-3
cat
x-1
=
x-4
x-1
AND
entered
:arg0
x-1
:EVENT
e-2
purred
:arg0
x-4
:EVENT
e-3

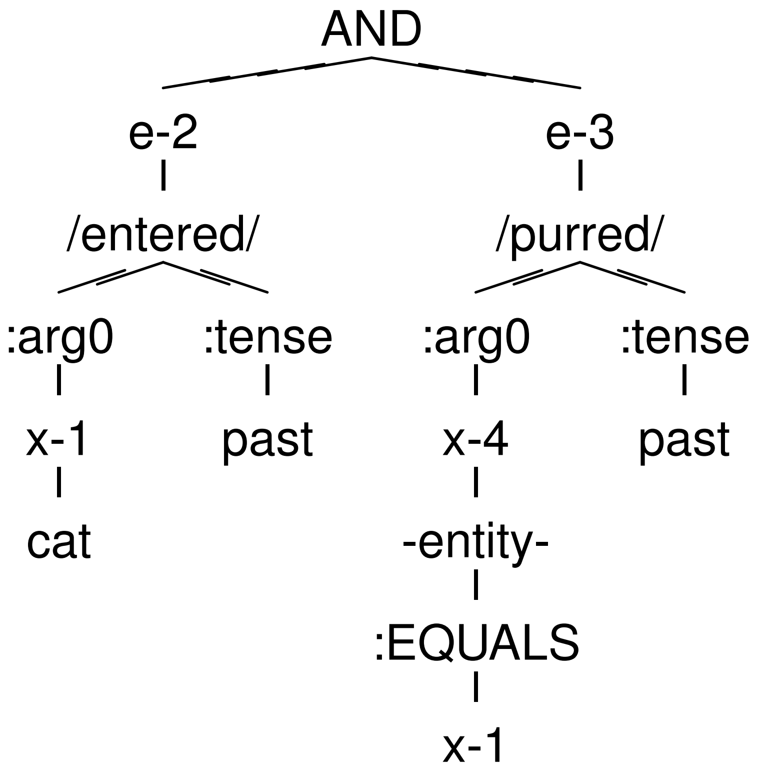

AND
past
e-2
past
e-3
cat
x-1
=
x-4
x-1
AND
entered
:arg0
x-1
:EVENT
e-2
purred
:arg0
x-4
:EVENT
e-3

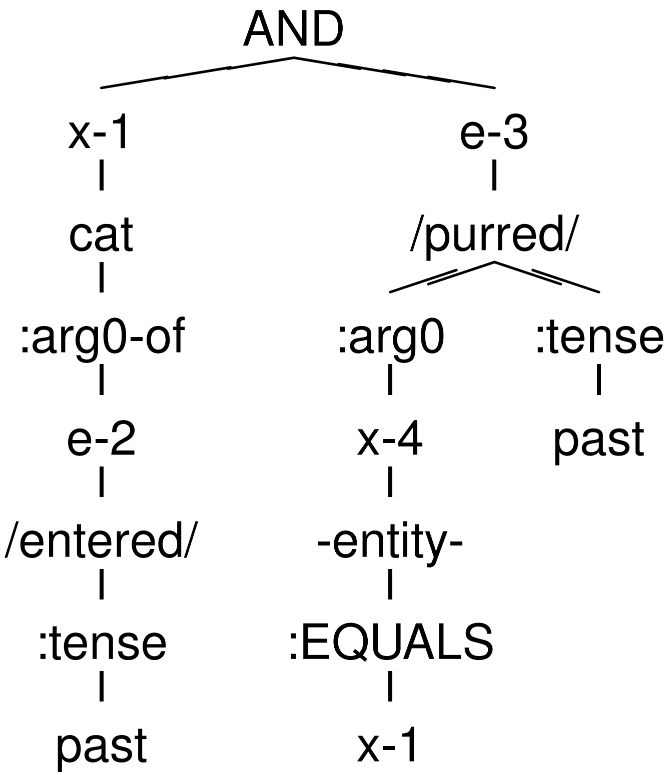

AND
past
e-2
past
e-3
cat
x-1
=
x-4
x-1
entered
:arg0
x-1
:EVENT
e-2
purred
:arg0
x-4
:EVENT
e-3

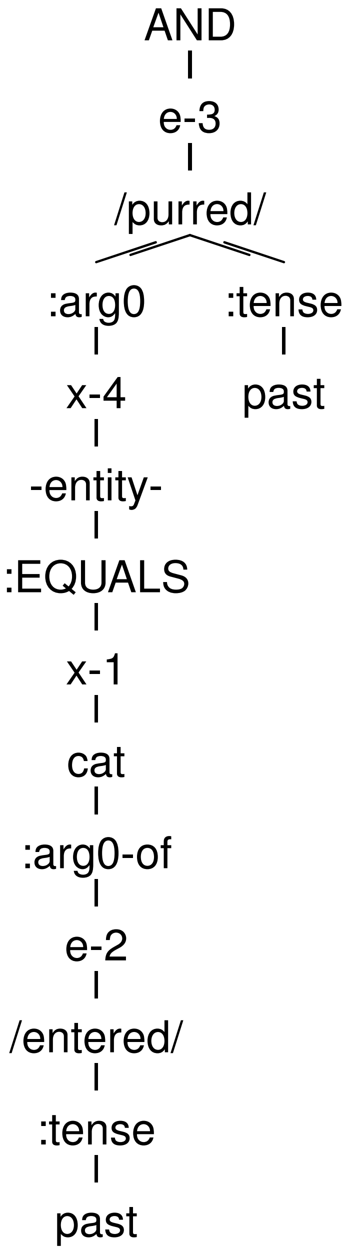

AND
past
e-2
past
e-3
x-1
cat
x-4
-entity-
:EQUALS
x-1
e-2
/entered/
:arg0
x-1
e-3
/purred/
:arg0
x-4


AND
x-1
cat
x-4
-entity-
:EQUALS
x-1
e-2
/entered/
:arg0
x-1
:tense
past
e-3
/purred/
:arg0
x-4
:tense
past

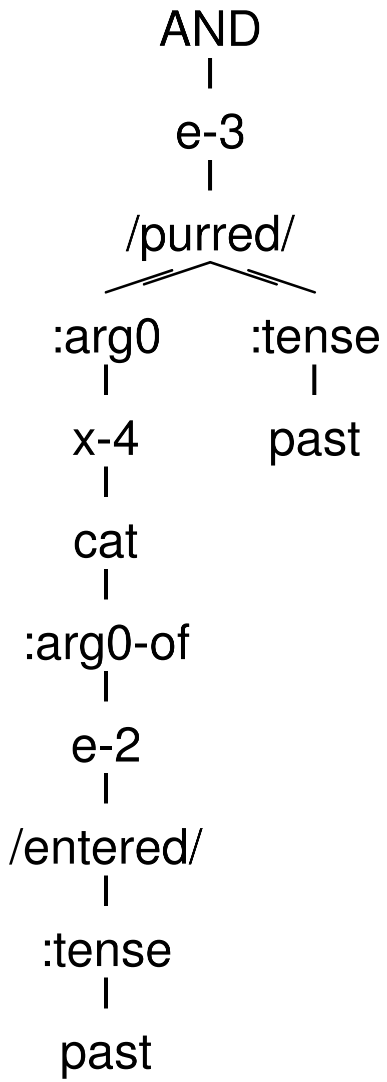

AND
x-1
cat
e-2
/entered/
:arg0
x-1
:tense
past
e-3
/purred/
:arg0
x-4
-entity-
:EQUALS
x-1
:tense
past

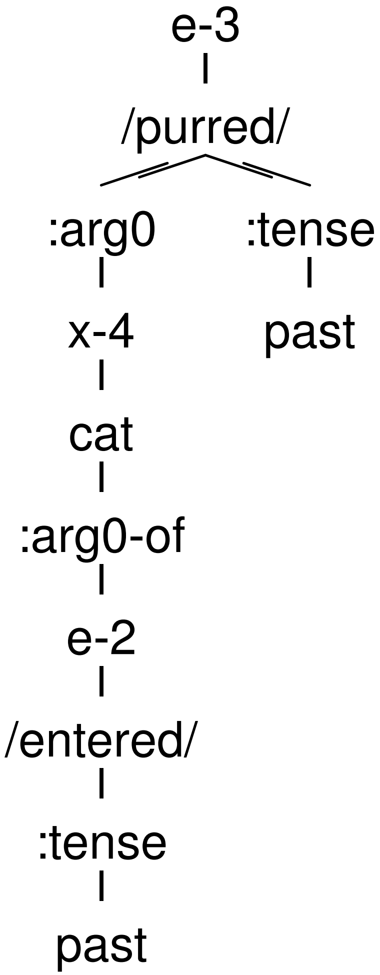

AND
e-2
e-3
/entered/
/purred/
:arg0
:tense
:arg0
:tense
x-1
past
x-4
past
cat
-entity-
:EQUALS
x-1

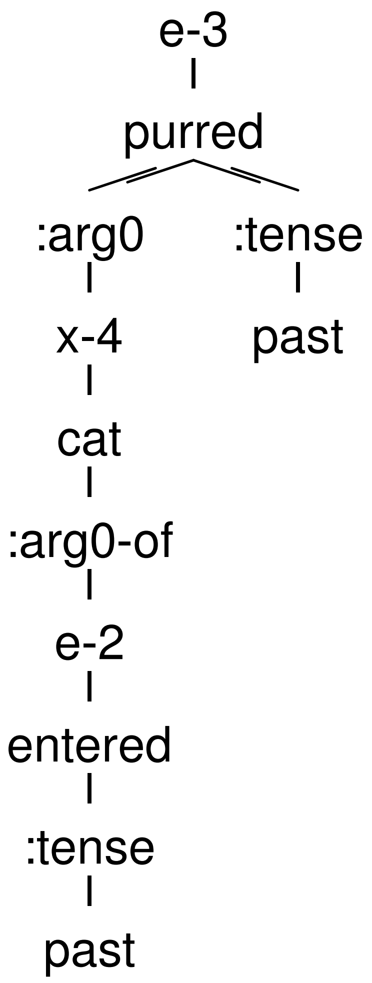

AND
x-1
cat
:arg0-of
e-2
/entered/
:tense
past
e-3
/purred/
:arg0
x-4
-entity-
:EQUALS
x-1
:tense
past



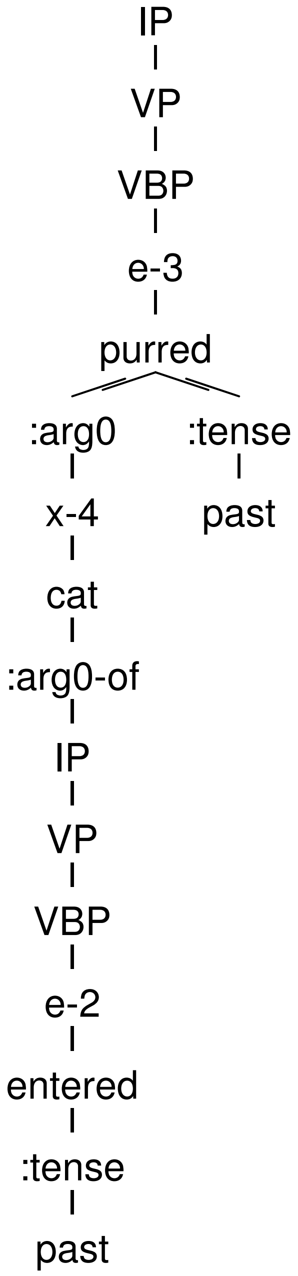
AND
e-3
/purred/
:arg0
:tense
x-4
past
-entity-
:EQUALS
x-1
cat
:arg0-of
e-2
/entered/
:tense
past

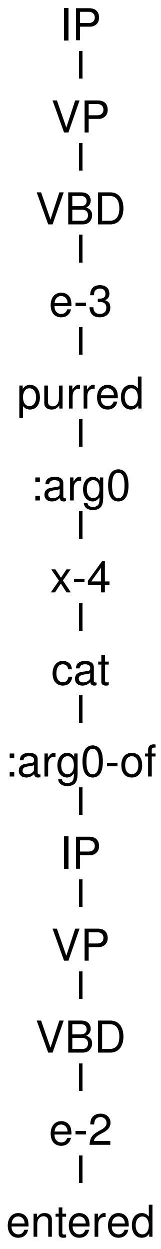

AND
e-3
/purred/
:arg0
:tense
x-4
past
cat
:arg0-of
e-2
/entered/
:tense
past

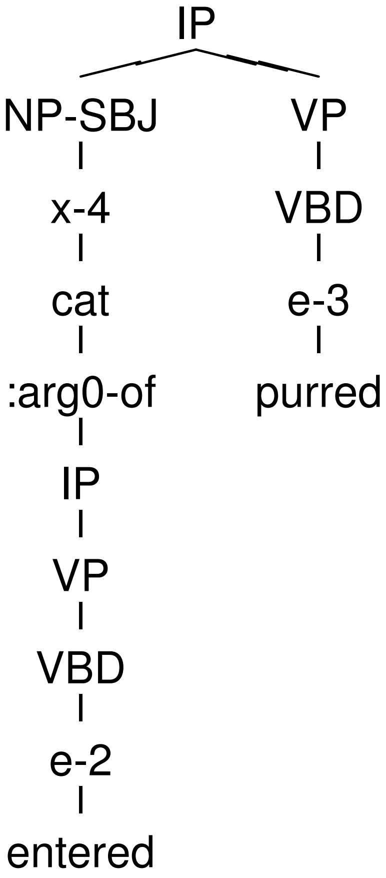

e-3
/purred/
:arg0
:tense
x-4
past
cat
:arg0-of
e-2
/entered/
:tense
past

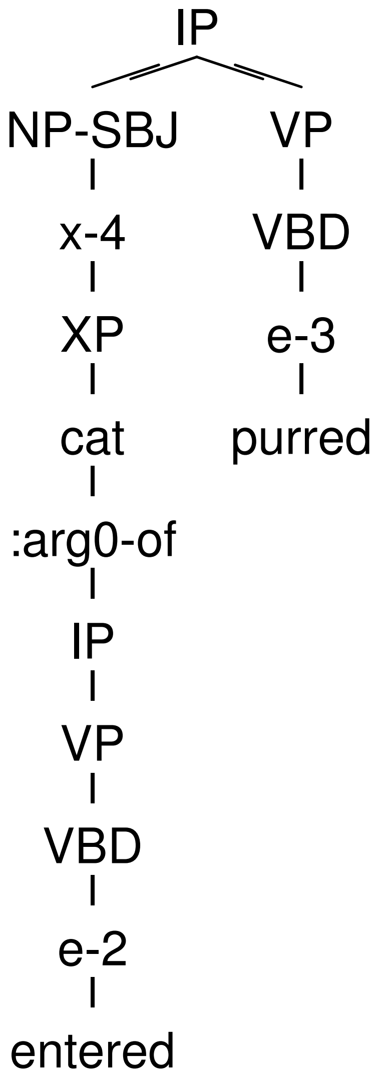


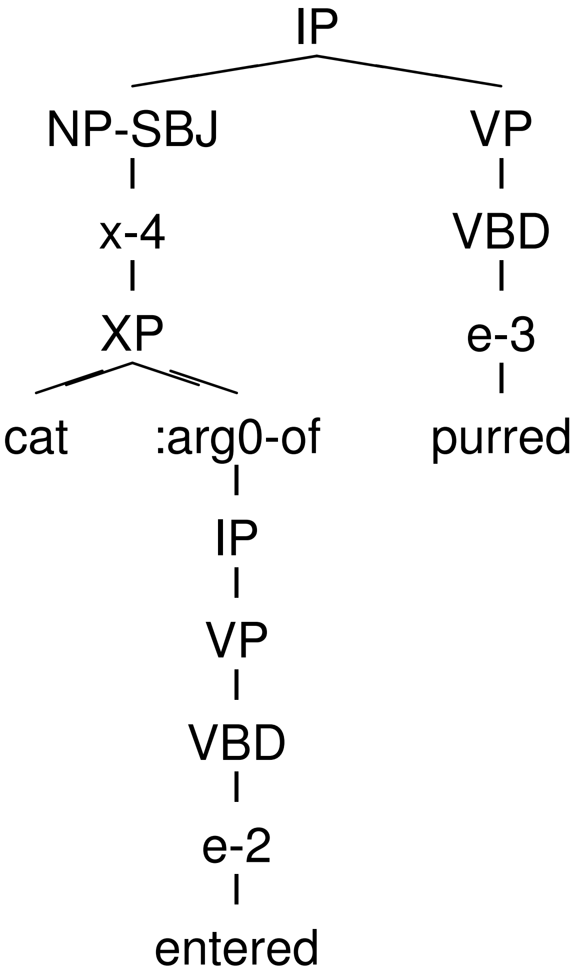
e-3
purred
:arg0
:tense
x-4
past
cat
:arg0-of
e-2
entered
:tense
past

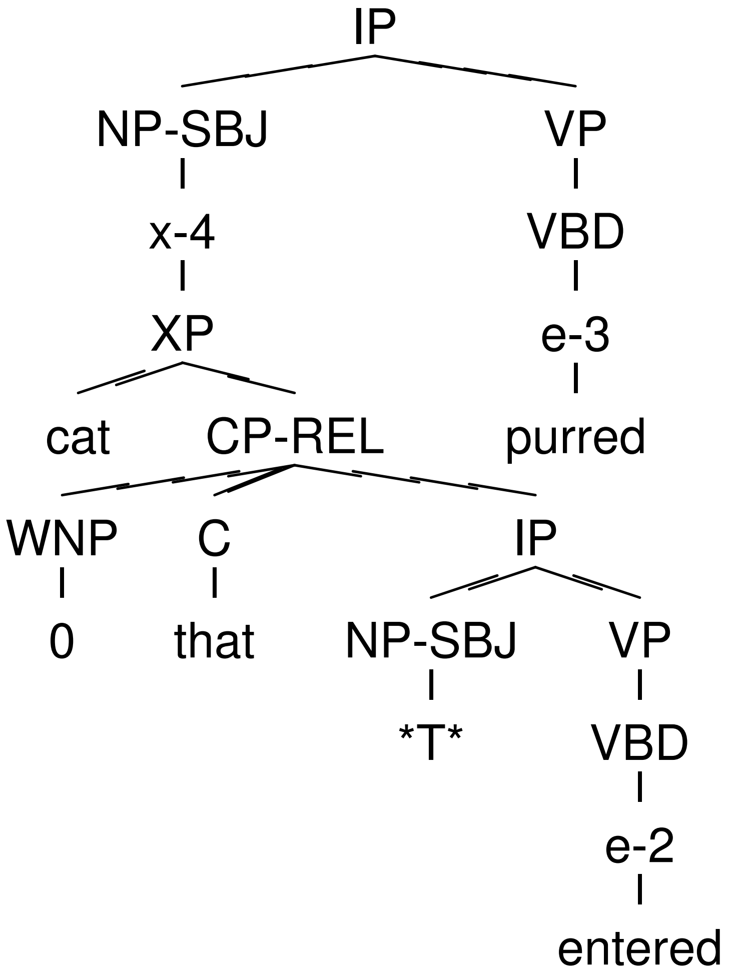

IP
VP
VBP
e-3
purred
:arg0
:tense
x-4
past
cat
:arg0-of
IP
VP
VBP
e-2
entered
:tense
past

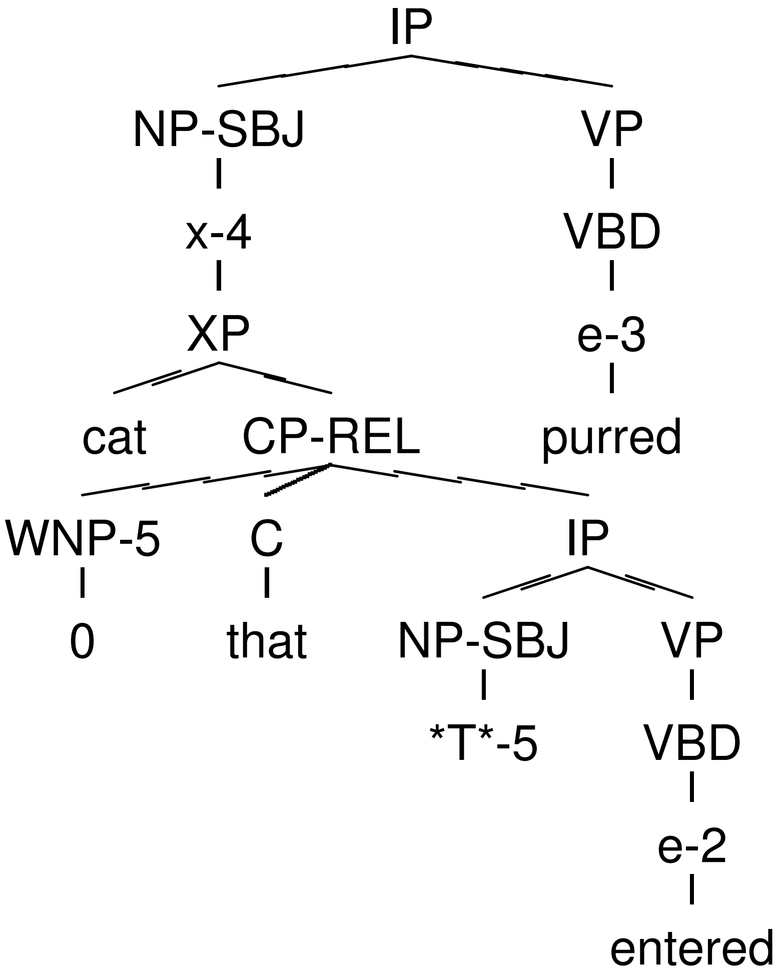

IP
VP
VBD
e-3
purred
:arg0
x-4
cat
:arg0-of
IP
VP
VBD
e-2
entered

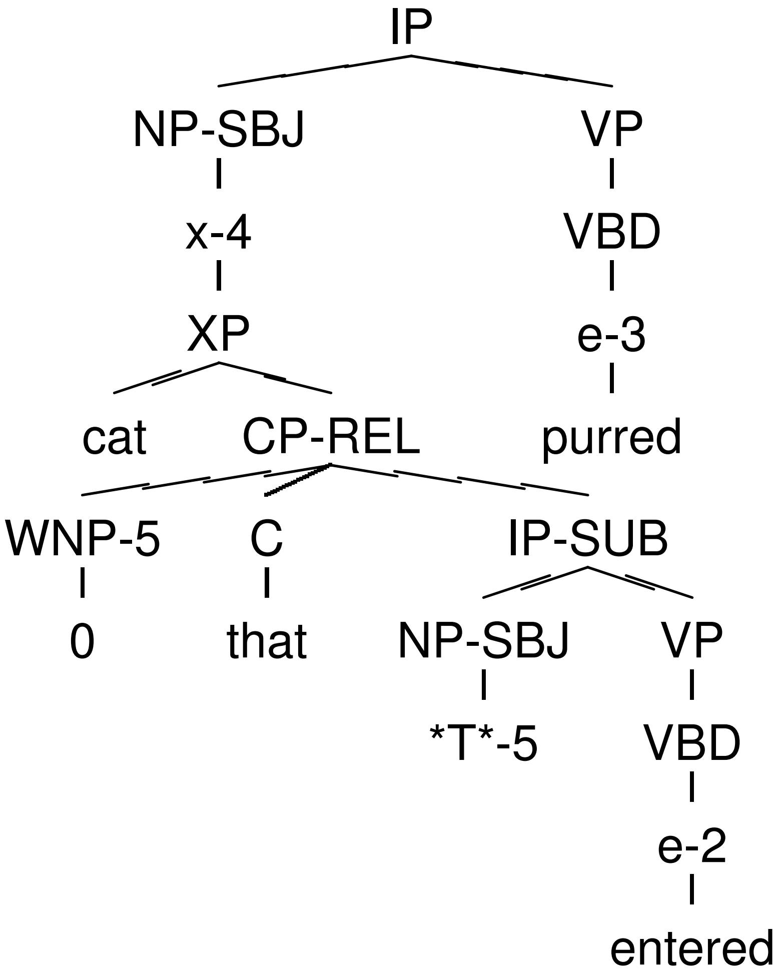

IP
NP-SBJ
VP
x-4
VBD
cat
e-3
:arg0-of
purred
IP
VP
VBD
e-2
entered

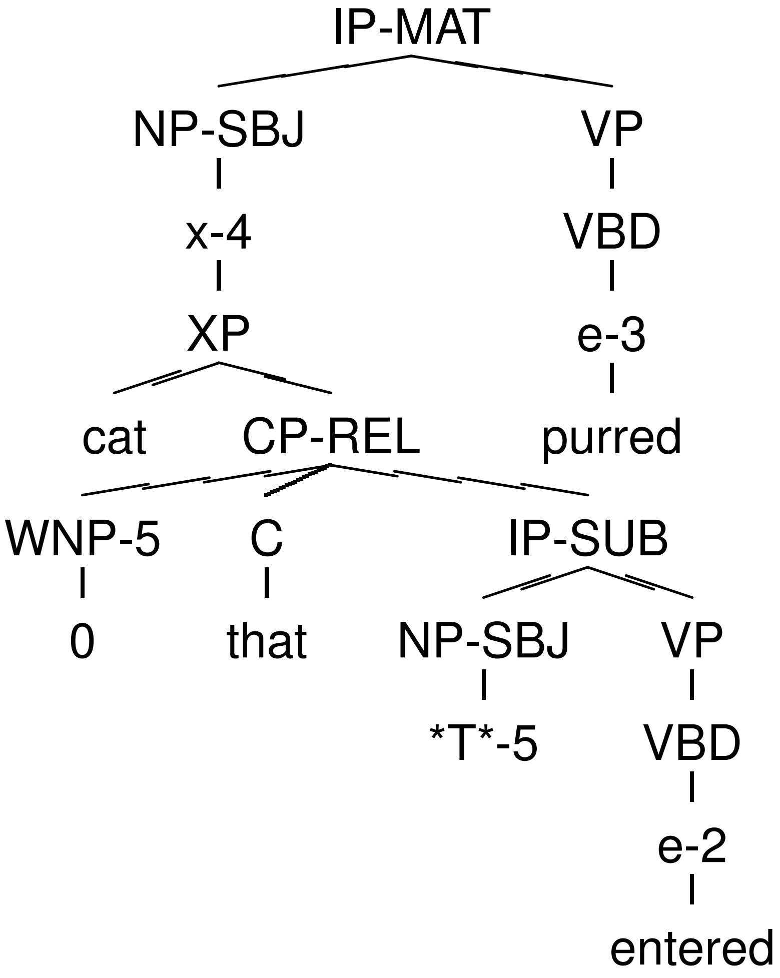

IP
NP-SBJ
VP
x-4
VBD
XP
e-3
cat
purred
:arg0-of
IP
VP
VBD
e-2
entered

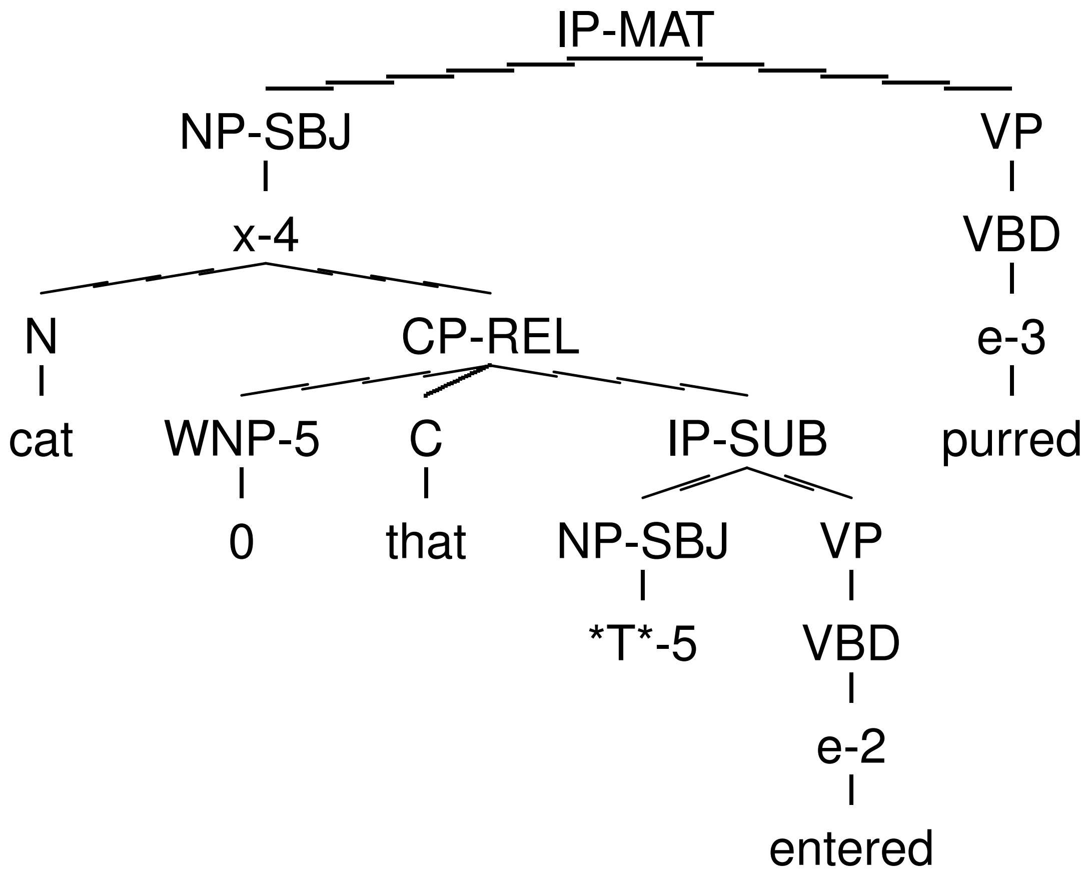

IP
NP-SBJ
VP
x-4
VBD
XP
e-3
cat
:arg0-of
purred
IP
VP
VBD
e-2
entered

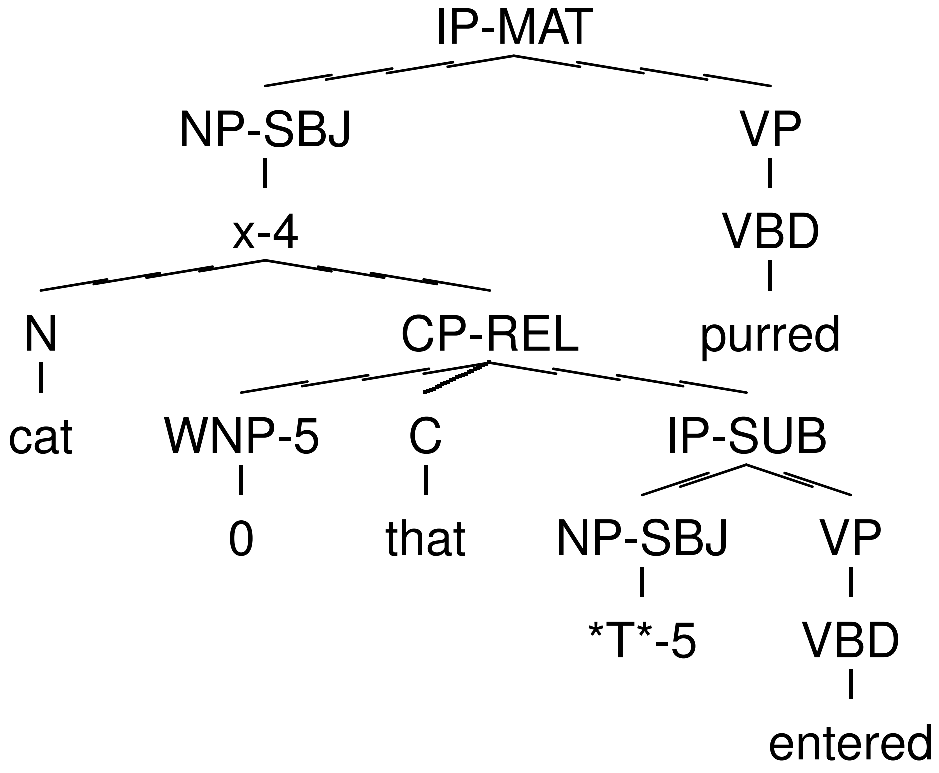

IP
NP-SBJ
VP
x-4
VBD
XP
e-3
cat
CP-REL
purred
WNP
C
IP
0
that
NP-SBJ
VP
T
VBD
e-2
entered

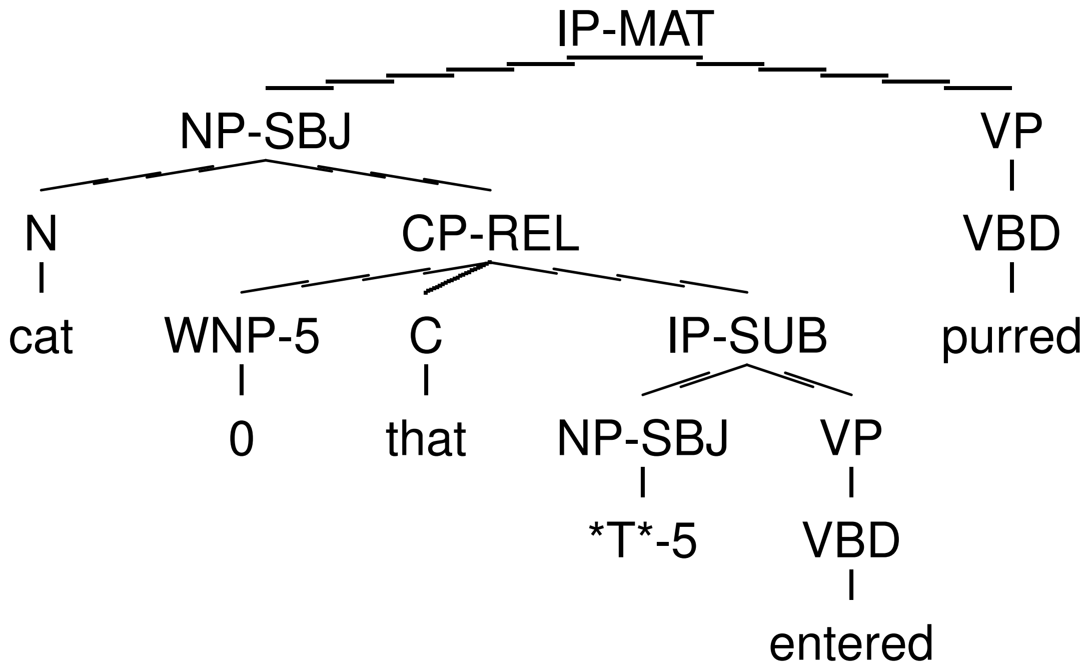

IP
NP-SBJ
VP
x-4
VBD
XP
e-3
cat
CP-REL
purred
WNP-5
C
IP
0
that
NP-SBJ
VP
T-5
VBD
e-2
entered

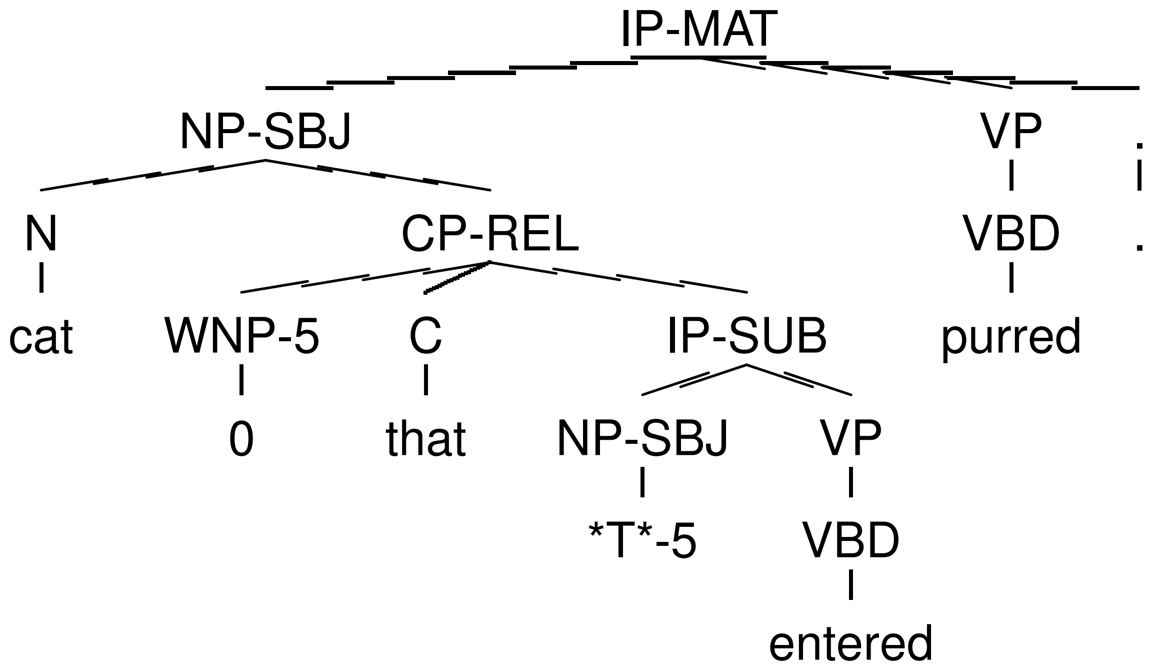

IP
NP-SBJ
VP
x-4
VBD
XP
e-3
cat
CP-REL
purred
WNP-5
C
IP-SUB
0
that
NP-SBJ
VP
T-5
VBD
e-2
entered

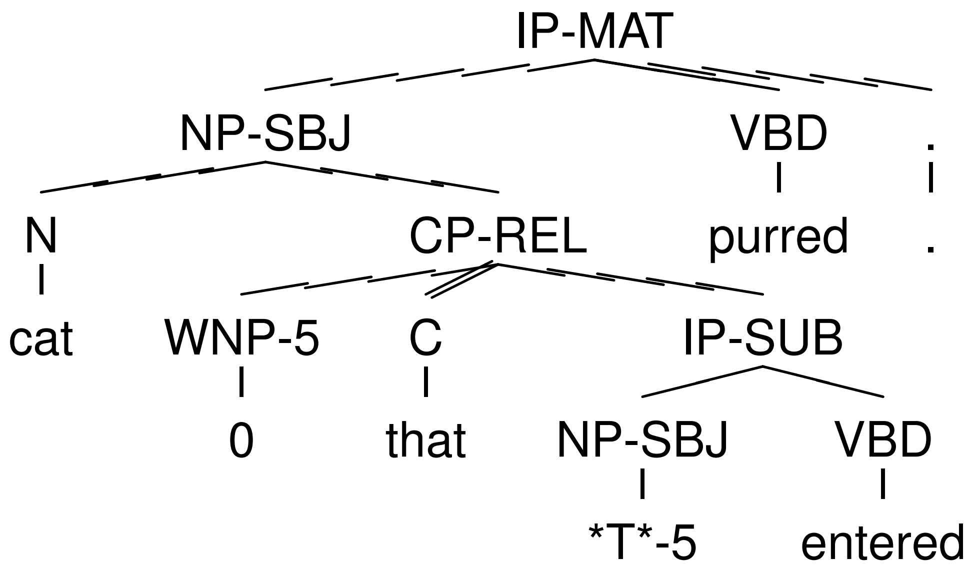

IP-MAT
NP-SBJ
VP
x-4
VBD
XP
e-3
cat
CP-REL
purred
WNP-5
C
IP-SUB
0
that
NP-SBJ
VP
T-5
VBD
e-2
entered

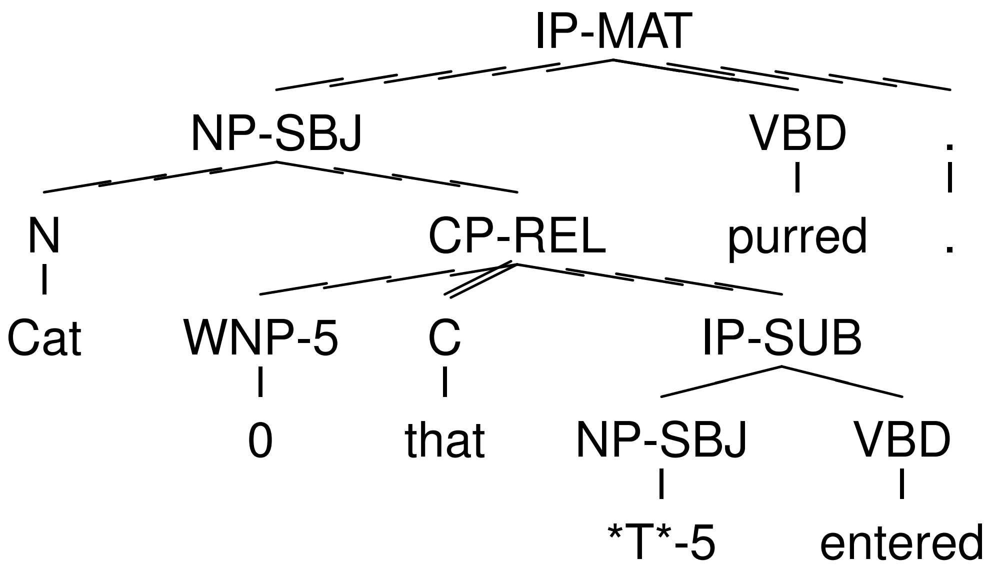

IP-MAT
NP-SBJ
VP
x-4
VBD
N
CP-REL
e-3
cat
WNP-5
C
IP-SUB
purred
0
that
NP-SBJ
VP
T-5
VBD
e-2
entered


IP-MAT
NP-SBJ
VP
x-4
VBD
N
CP-REL
purred
cat
WNP-5
C
IP-SUB
0
that
NP-SBJ
VP
T-5
VBD
entered


IP-MAT
NP-SBJ
VP
N
CP-REL
VBD
cat
WNP-5
C
IP-SUB
purred
0
that
NP-SBJ
VP
T-5
VBD
entered


IP-MAT
NP-SBJ
VP
.
N
CP-REL
VBD
.
cat
WNP-5
C
IP-SUB
purred
0
that
NP-SBJ
VP
T-5
VBD
entered


IP-MAT
NP-SBJ
VBD
.
N
CP-REL
purred
.
cat
WNP-5
C
IP-SUB
0
that
NP-SBJ
VBD
T-5
entered


IP-MAT
NP-SBJ
VBD
.
N
CP-REL
purred
.
Cat
WNP-5
C
IP-SUB
0
that
NP-SBJ
VBD
T-5
entered

Experiments

The smatch metric (Cai and Knight 2013) can be used to evaluate the success of round tripping on English.

This is a metric to measure whole-sentence semantic analysis by calculating the degree of overlap between meaning representations.

Results for 1452 annotated sentences (14,118 tokens) from four different registers are as follows:

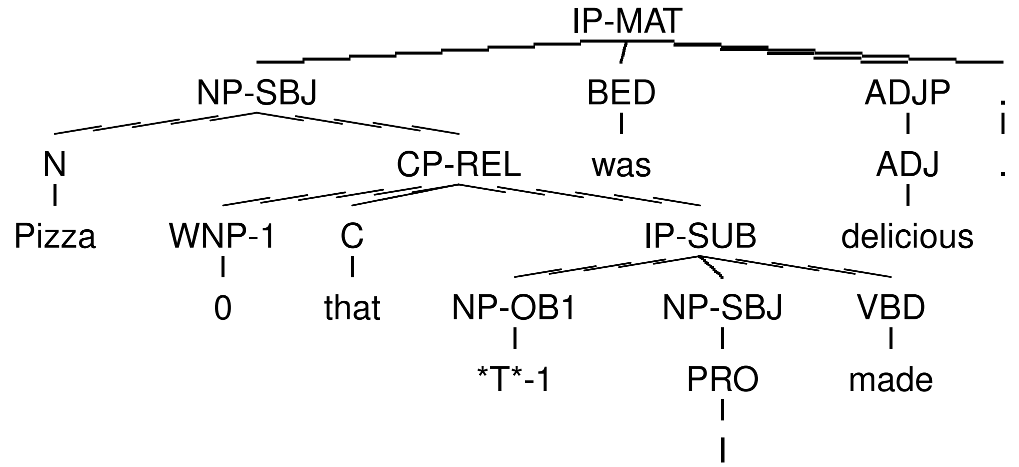
register	sentences	tokens	precision	recall	F-score
textbook	687	5194	0.98	0.98	0.98
newswire	121	2381	0.97	0.96	0.97
(simple) fiction	547	5241	0.96	0.96	0.96
non-fiction	97	1302	0.93	0.93	0.93

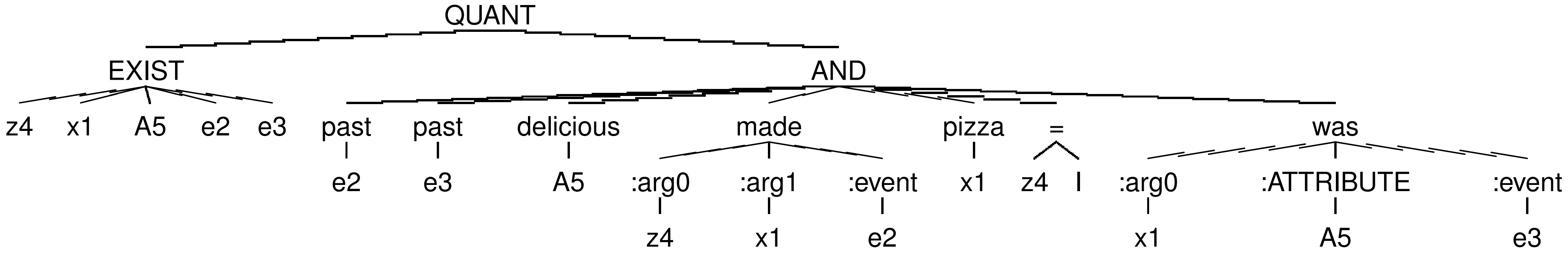
Results show that in round tripping with English, so building a meaning representation A from a gold standard parse and generating back to an English sentence and then building a meaning representation B from the generated sentence, and then comparing A with B, it is possible to retain the bulk of semantic content with high precision and recall.

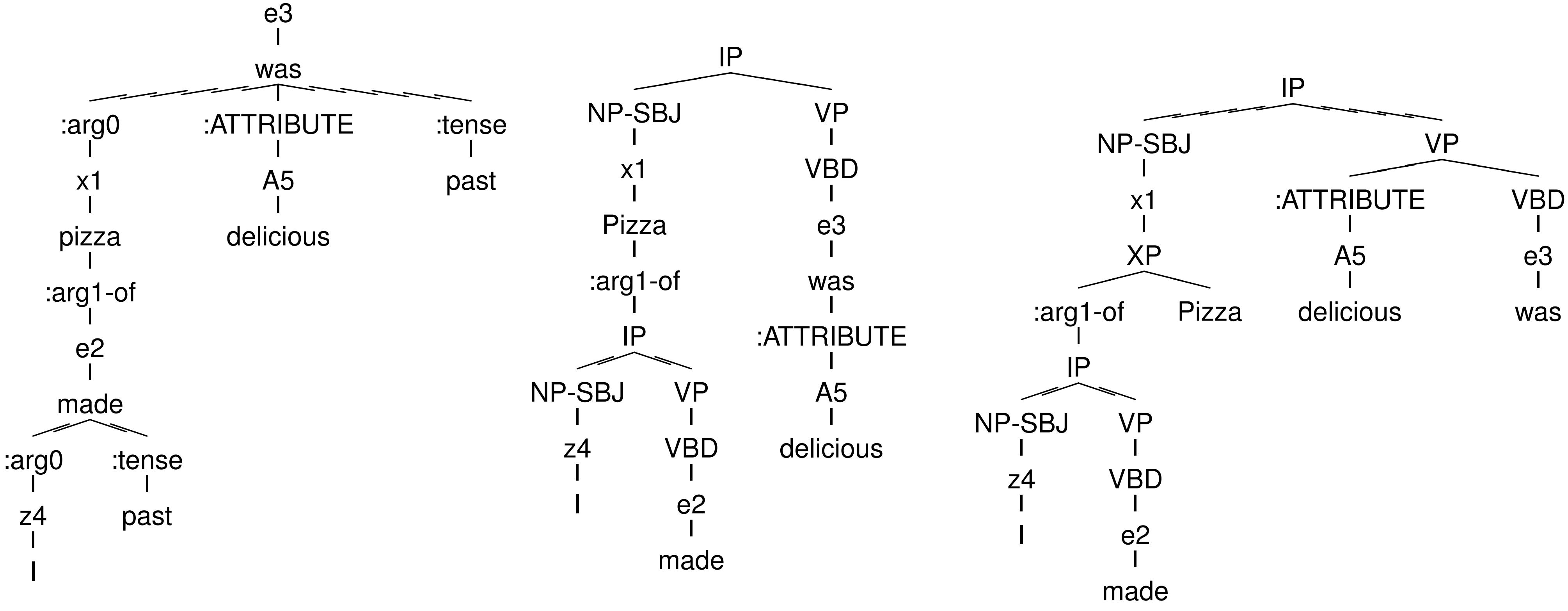
Results also reflect a decline in performance on more challenging data.

Towards translation

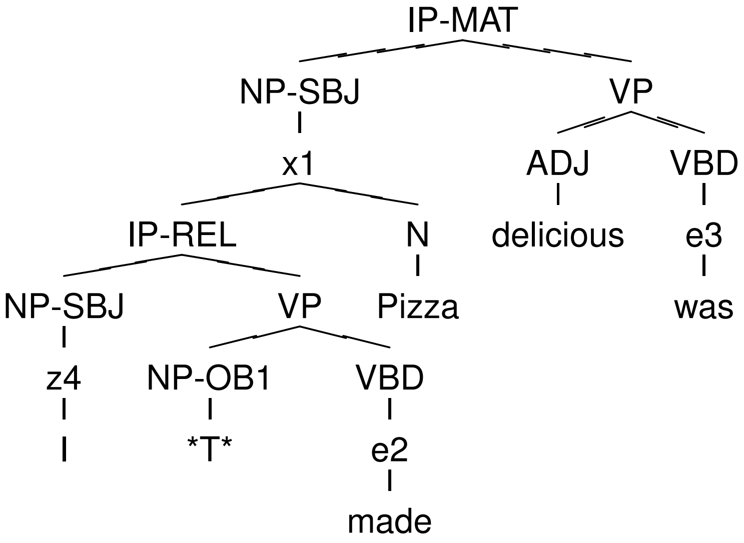
Calculate a meaning representation for English input but use Japanese generation rules



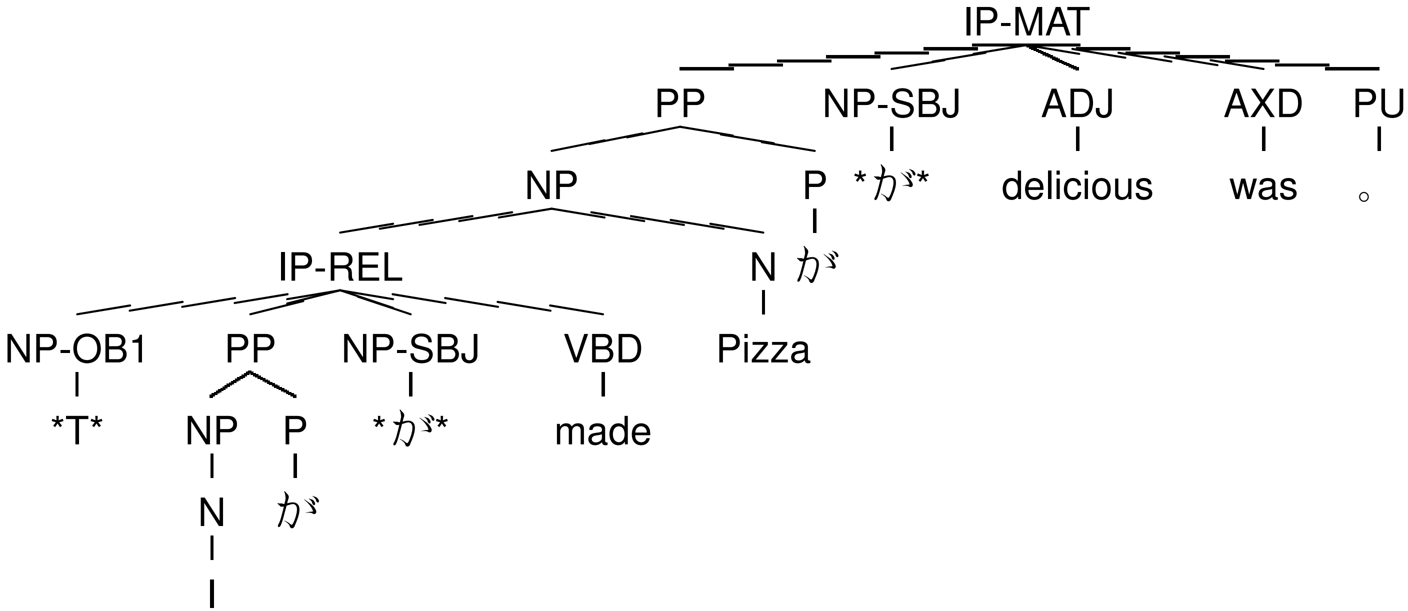


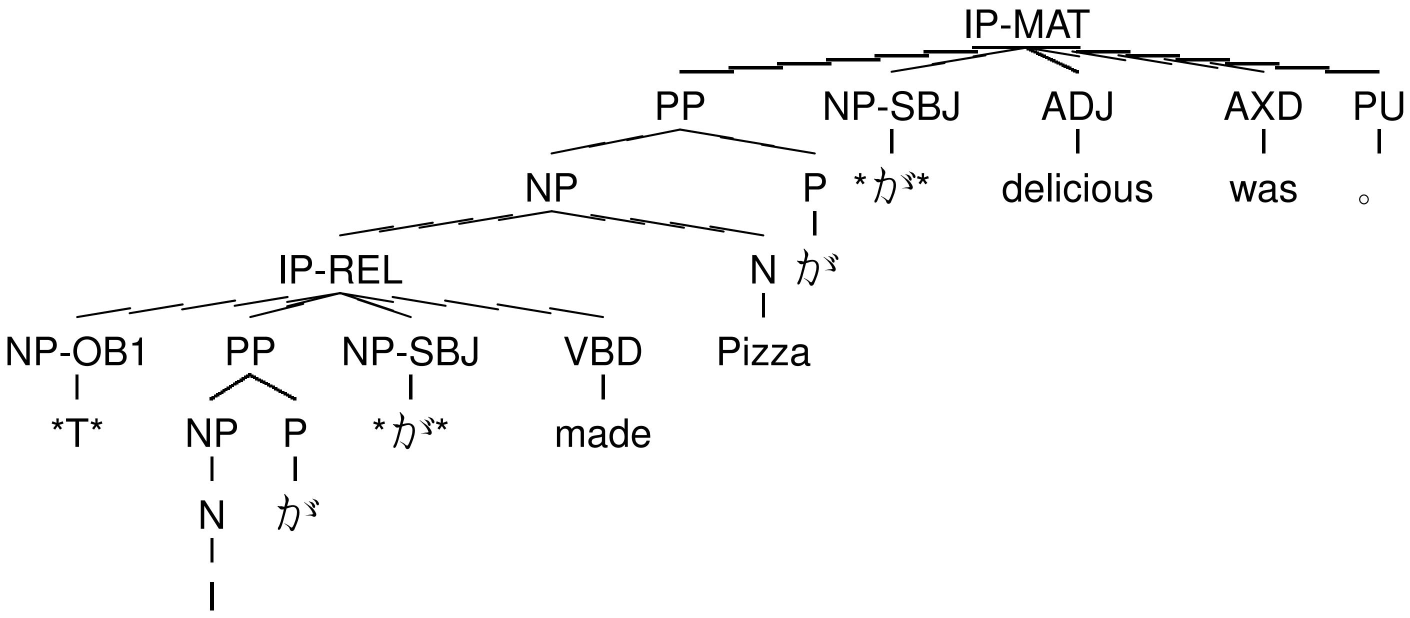


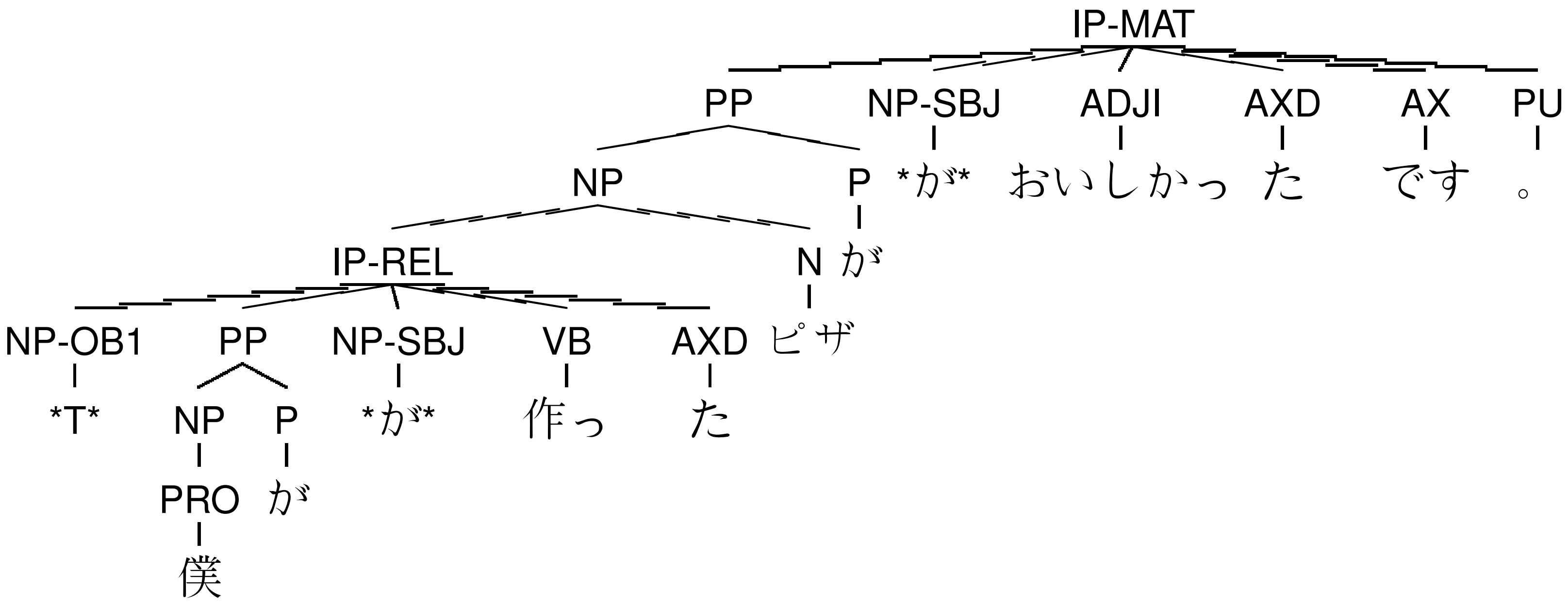
Projection of relative clause structure is again triggered, only for Japanese there is projection of an IP-REL layer to the left side of the head noun.

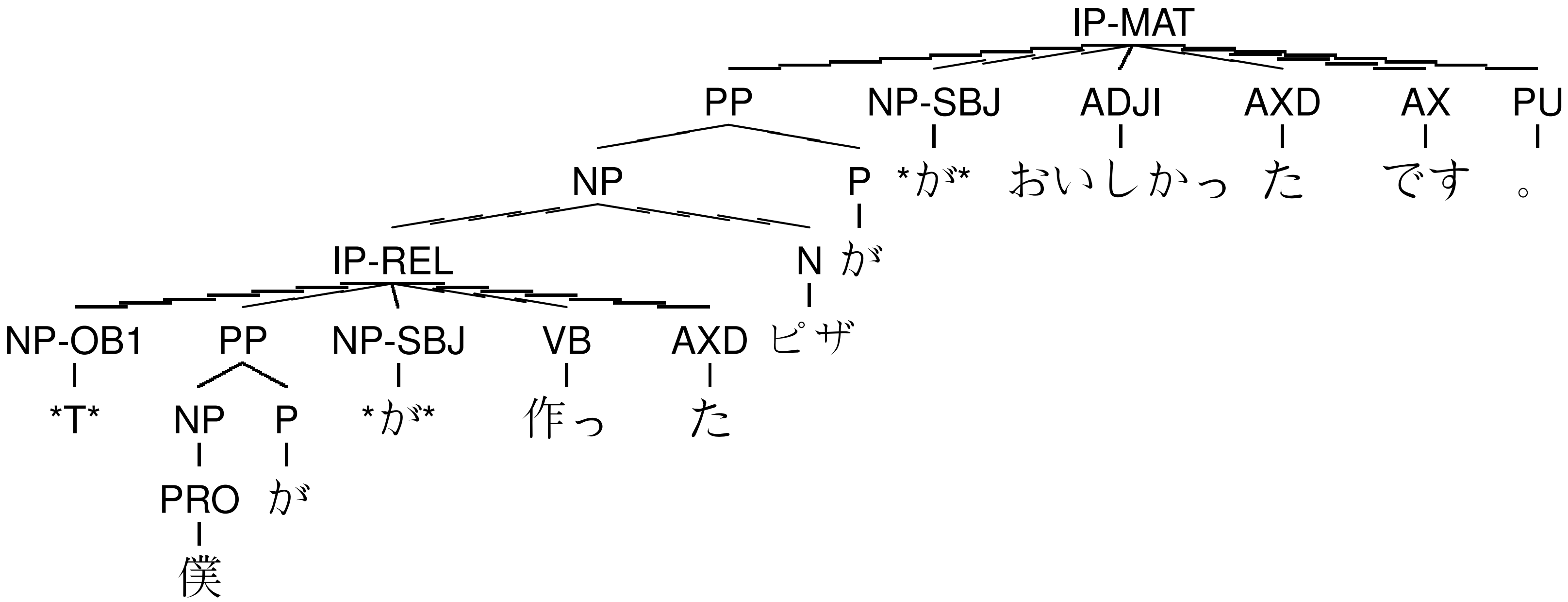


Generation is completed with the addition of case particles.




IP-MAT
PP
NP-SBJ
ADJ
AXD
PU
NP
P
が
delicious
was
。
IP-REL
N
が
NP-OB1
PP
NP-SBJ
VBD
Pizza
T
NP
P
が
made
N
が
I


IP-MAT
PP
NP-SBJ
ADJI
AXD
AX
PU
NP
P
が
おいしかっ
た
です
。
IP-REL
N
が
NP-OB1
PP
NP-SBJ
VB
AXD
ピザ
T
NP
P
が
作っ
た
PRO
が
僕

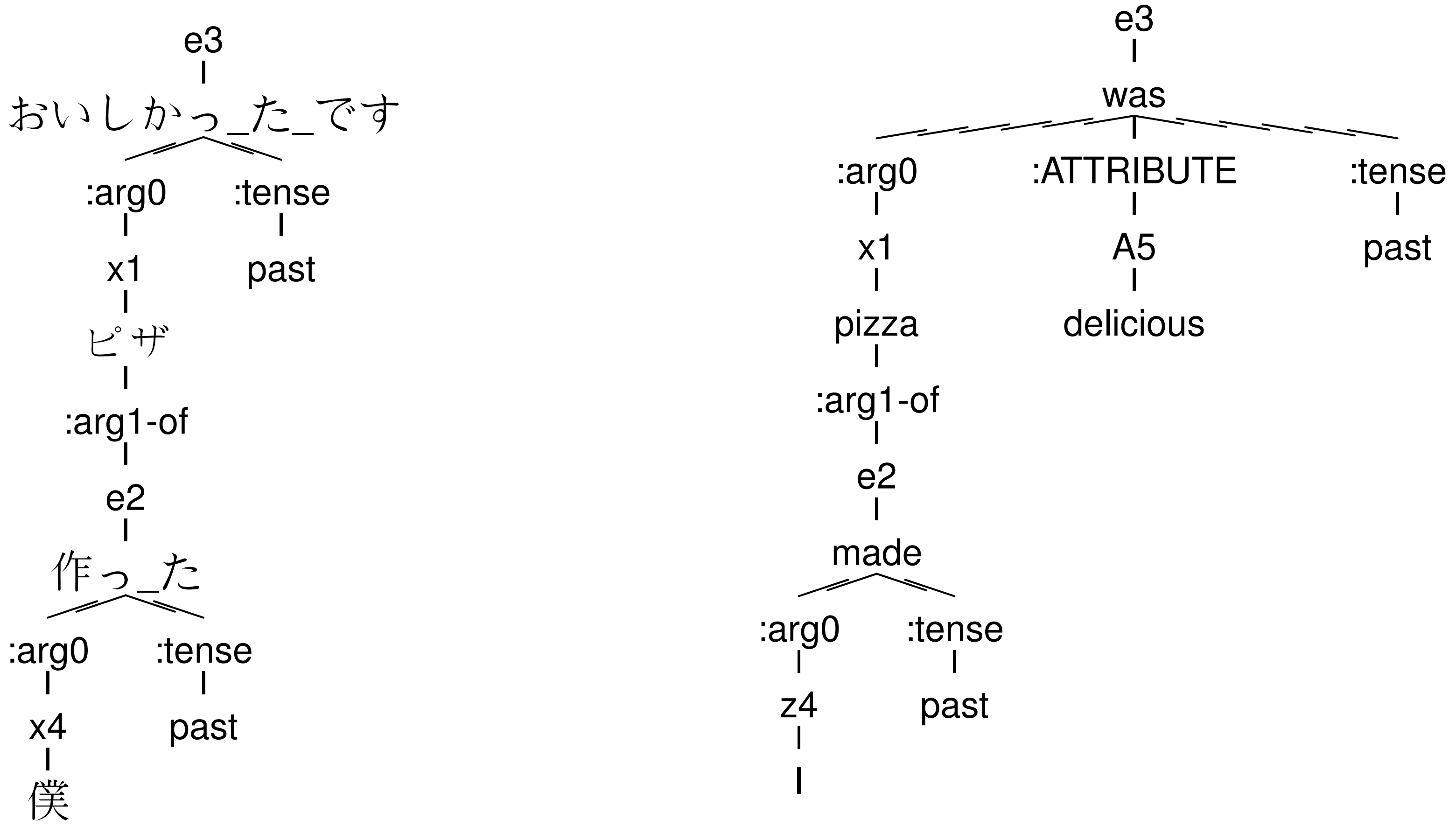


By feeding the Japanese version of the example sentence into the Treebank Semantics system a meaning representation is built:

$$\exists x_4 x_1 e_2 e_3 (\mathtt{past}(e_3) \wedge \mathtt{past}(e_2) \wedge x_4 = \text{僕} \wedge \text{ピザ}(x_1) \wedge \text{作っ\_た}(e_2, x_4, x_1) \wedge \text{おいしかっ\_た\_です}(e_3, x_1))$$

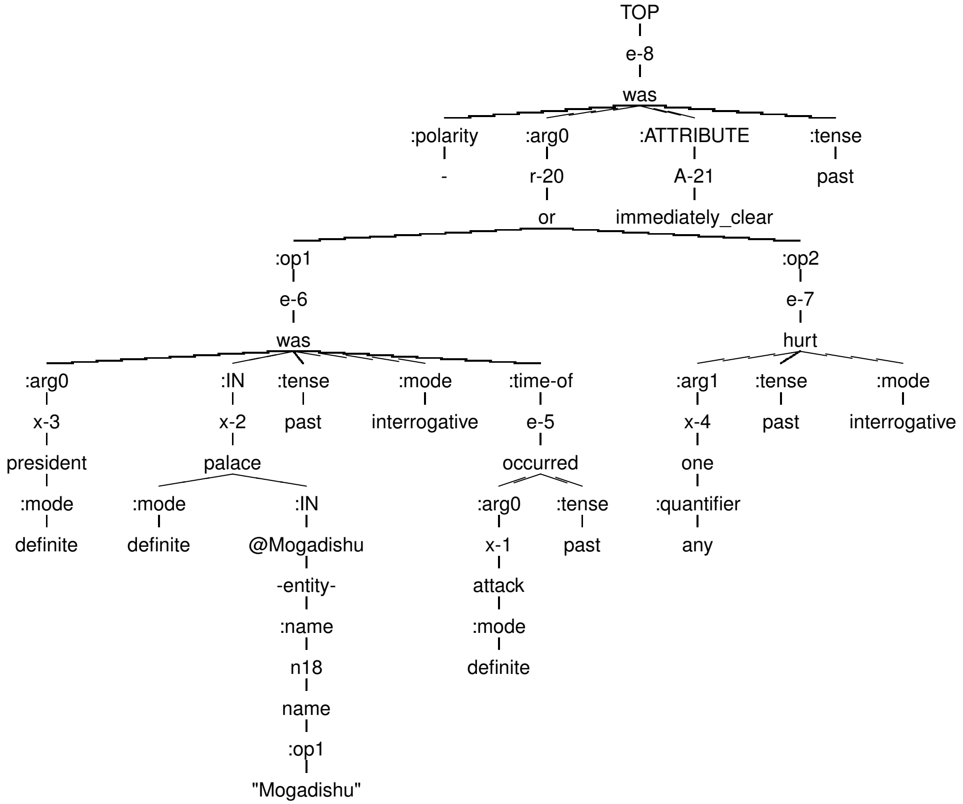
Such a meaning representation can be modified to form the basis for generation, exactly as seen with English.

Having meaning representations for sentences of parallel corpora is a basis for extracting rules for a full translation system.



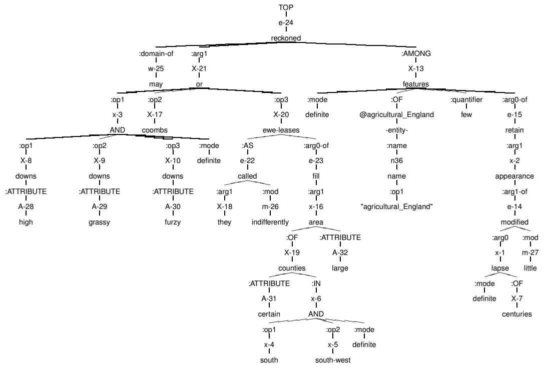
Scalibility of the approach

Input: It was not immediately clear if the president was in the palace in Mogadishu when the attack occurred or if anyone was hurt .



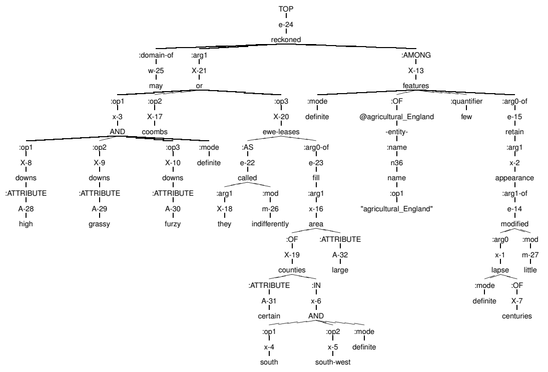
Output: Whether when the attack occurred the president was in the palace in Mogadishu or whether any one was hurt was not immediately clear .

Input: Among the few features of agricultural England which retain an appearance but little modified by the lapse of centuries , may be reckoned the high , grassy and furzy downs , coombs , or ewe-leases , as they are indifferently called , that fill a large area of certain counties in the south and south-west .



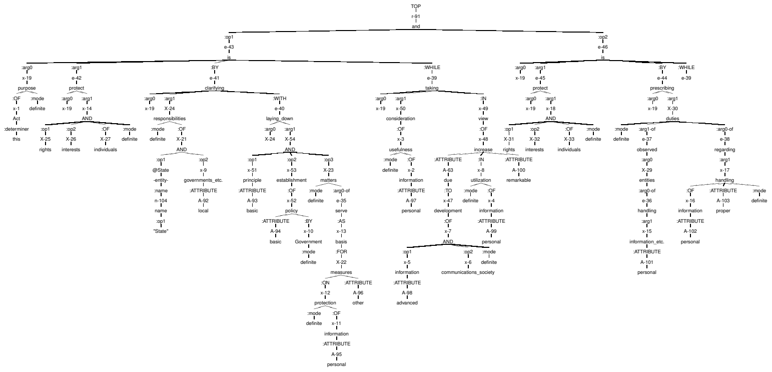
Output: The high downs , grassy downs and furzy downs , coombs or ewe-leases as indifferently they are called that fill large area of certain counties in the south and south-west may be reckoned among the few features of agricultural England that retain appearance that little the lapse of centuries modified .

Input: *Among the few features of agricultural England which retain an appearance but little modified by the lapse of centuries* , may be reckoned the high , grassy and furzy downs , coombs , or ewe-leases , as they are indifferently called , that fill a large area of certain counties in the south and south-west .



Output: The high downs , grassy downs and furzy downs , coombs or ewe-leases as indifferently they are called that fill large area of certain counties in the south and south-west may be reckoned *among the few features of agricultural England that retain appearance that little the lapse of centuries modified* .

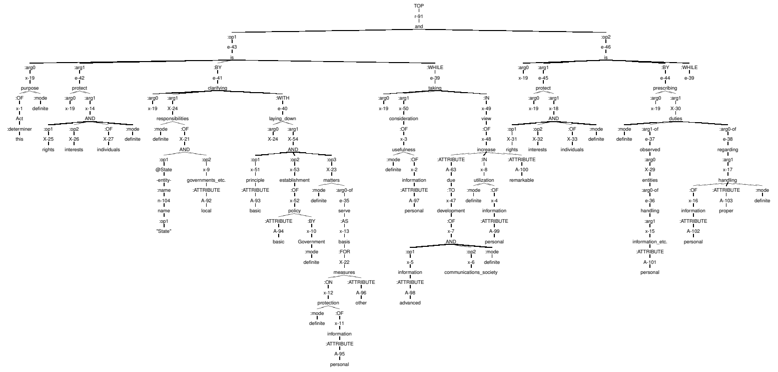
Input: The purpose of this Act is to protect the rights and interests of individuals while taking consideration of the usefulness of personal information , in view of a remarkable increase in the utilization of personal information due to development of the advanced information and communications society , by clarifying the responsibilities of the State and local governments , etc. with laying down basic principle , establishment of a basic policy by the Government and the matters to serve as a basis for other measures on the protection of personal information , and by prescribing the duties to be observed by entities handling personal information , etc. , regarding the proper handling of personal information .


TOP
r-91
and
:op1 :op2
e-43 e-46
is is
:arg0 :arg1 :BY :WHILE :arg0 :arg1 :BY :WHILE
x-19 e-42 e-41 e-39 x-19 e-45 e-44 e-39
purpose protect clarifying taking protect prescribing
:OF :mode :arg0 :arg1 :arg0 :arg1 :WITH :arg0 :arg1 :IN :arg0 :arg1 :arg0 :arg1
x-1 definite x-19 x-14 x-19 X-24 e-40 x-19 x-50 x-49 x-19 x-18 x-19 X-30
Act AND responsibilities laying_down consideration view AND duties
:determiner :op1 :op2 :OF :mode :mode :OF :arg0 :arg1 :OF :OF :op1 :op2 :OF :mode :mode :arg1-of :arg0-of
this X-25 X-26 X-27 definite definite X-21 X-24 X-54 x-3 x-48 X-31 X-32 X-33 definite definite e-37 e-38
rights interests individuals AND AND usefulness increase rights interests individuals observed regarding
:op1 :op2 :op1 :op2 :op3 :mode :OF :ATTRIBUTE :IN :ATTRIBUTE :arg0 :arg1
@State x-9 x-51 x-53 X-23 definite x-2 A-63 x-8 A-100 X-29 x-17
-entity- governments_etc. principle establishment matters information due utilization remarkable entities handling
:name :ATTRIBUTE :ATTRIBUTE :OF :mode :arg0-of :ATTRIBUTE :TO :mode :OF :arg0-of :OF :ATTRIBUTE :mode
n-104 A-92 A-93 x-52 definite e-35 A-97 x-47 definite x-4 e-36 x-16 A-103 definite
name local basic policy serve personal development information handling information proper
:op1 :ATTRIBUTE :BY :AS :OF :ATTRIBUTE :arg1 :ATTRIBUTE
"State" A-94 x-10 x-13 x-7 A-99 x-15 A-102
basic Government basis AND personal information_etc. personal
:mode :FOR :op1 :op2 :mode :ATTRIBUTE
definite X-22 x-5 x-6 definite A-101
measures information communications_society personal
:ON :ATTRIBUTE :ATTRIBUTE
x-12 A-96 A-98
protection other advanced
:mode :OF
definite x-11
information
:ATTRIBUTE
A-95
personal

Output: The purpose of this Act is to protect the rights and interests of individuals by clarifying the responsibilities of State and local governments etc. with laying down basic principle , establishment of basic policy by the Government and the matters that serve as basis for other measures on the protection of personal information while taking consideration of the usefulness of personal information in view of remarkable increase due to development of the advanced information and communications society in the utilization of personal information and is to protect the rights and interests of individuals by prescribing the duties that entities handling personal information etc. observed regarding the proper handling of personal information while taking consideration of the usefulness of personal information in view of remarkable increase due to development of the advanced information and communications society in the utilization of personal information .

Output: The purpose of this Act is to protect the rights and interests of individuals by clarifying the responsibilities of State and local governments etc. with laying down basic principle , establishment of basic policy by the Government and the matters that serve as basis for other measures on the protection of personal information *while taking consideration of the usefulness of personal information in view of remarkable increase due to development of the advanced information and communications society in the utilization of personal information* **and** is to protect the rights and interests of individuals by prescribing the duties that entities handling personal information etc. observed regarding the proper handling of personal information *while taking consideration of the usefulness of personal information in view of remarkable increase due to development of the advanced information and communications society in the utilization of personal information* .

Input: The purpose of this Act is to protect the rights and interests of individuals *while taking consideration of the usefulness of personal information , in view of a remarkable increase in the utilization of personal information due to development of the advanced information and communications society* , by clarifying the responsibilities of the State and local governments , etc. with laying down basic principle , establishment of a basic policy by the Government and the matters to serve as a basis for other measures on the protection of personal information , **and** by prescribing the duties to be observed by entities handling personal information , etc. , regarding the proper handling of personal information .



Conclusion

A complete pipeline was described for taking parsed sentences, going to meaning representations (initially to Davidsonian predicate logic representations, then to PENMAN notation), and then back to parsed sentences (the round trip).

Keeping to the same language tests the combined success of building meaning representations from parsed input and of generating parsed output.

With the described method, the smatch metric reveals that the bulk of semantic content is retained with high precision and recall on a range of data.

While there is no explicit flagging in a conventional Davidsonian meaning representation of what is a verb, noun, adjective, relative clause, passive, control relation, etc., much information is found to facilitate generation of such language elements when there is sort and argument role information and when there is subsequent re-packaging of content, as with the PENMAN format, guided by the aim to form single rooted structures.

Future directions

Future directions are to achieve translation with being able to switch languages at the point of manipulating meaning representations.

Current transfer shortfall is seen with meaning representations built from parsed parallel corpora data.