Deterministic natural language generation from meaning representations for machine translation

Alastair Butler
National Institute for Japanese Language and Linguistics
ajb129@hotmail.com

Abstract

This paper describes a deterministic method for generating natural language suited to being part of a machine translation system with meaning representations as the level for language transfer. Starting from Davidsonian/Penman meaning representations, syntactic trees are built following the Penn Parsed Corpus of Modern British English, from which the yield (i.e., the words) can be taken. The novel contribution is to highlight exploiting the presentation of meaning content to inform decisions regarding the selection of language constructions: active vs. passive, argument subject vs. expletive it vs. existential there, discourse vs. intra-sentential coordination vs. adverbial clause vs. participial clause vs. purpose clause, and infinitive clause vs. finite clause vs. small clause vs. relative clause vs. it cleft.

1 Introduction

This paper pursues the idea that the arrangement of information contained by a meaning representation can provide sufficient clues to drive a deterministic generation of natural language. This is particularly suited to being part of a machine translation system with meaning representations as the level for language transfer because the clues exploited for generating the target language can be gathered when converting the source language to a meaning representation. The paper is structured as follows. Section 2 gives background for the approach. Section 3 sketches the generation procedure with a simple example. Section 4 is the core of the paper, detailing different grammatical constructions and triggers for their creation. Section 5 is a conclusion.

2 Background

The generation of this paper aims to form part of a pipeline approach to machine translation, where a semantic parser transforms source language sentences into meaning representations that are inputs for generation to produce target language sentences.

While many semantic parsing systems are now available, representations reached are typically either (neo-)Davidsonian predicate language formulas (Davidson, 1967; Parsons, 1990; Landman 2000), or Penman style representations (Matthiessen and Bateman, 1991). Generation in this paper will start from Penman representations that are derived from Davidsonian predicate language formulas. Systems producing (neo-)Davidsonian output are typically based on methods from formal semantics with compositional assembly of sentence/discourse meanings rooted in first obtaining a syntactic parse, e.g., Copestake et al. (2006), Bos (2008), Butler (2015a), and Mineshima et al (2015). Systems that natively produce Penman representations include the growing number of systems trained on the sembanks of Abstract Meaning Representation (AMR; Banarescu et al., 2013), e.g., Flanigan et al. (2014), Artzi et al. (2015), and Pust et al. (2015).

For generation from Penman representations, there is currently the Nitrogen system (Langkilde and Knight, 1998). Nitrogen relies on a statistical component to filter results generated from a base system with phrase structure like rules. Other generation systems typically start from representations of

Proceedings of the 2nd Workshop on Semantics-Driven Machine Translation, pages 1–9,
San Diego, California, June 16, 2016.

argument structure or quasi-logical forms, e.g., Alshawi (1992) and Humphreys et al (2001).

The generation of this paper builds on Butler (2015b), in that a series of transformations are followed to construct parsed trees. Trees constructed will conform to the Penn Parsed Corpus of Modern British English (PPCMBE; Kroch et al., 2010), with the yield (i.e. words) producing target sentences of English. This scheme is similar to the Penn Treebank scheme (Bies et al., 1995), but with more consistency in the projection of phrase structure (following X-bar theory; Chomsky, 1970) and handling of coordination, while clause structure is generally flat since VP occurs only with coordination.

An implementation of the generation of this paper is available from `http://www.compling.jp/generation`. The assumed engine to transform trees is provided by tsurgeon (Levy and Andrew, 2006). This works with tsurgeon scripts that contain patterns with associated actions. Patterns describe tree structure with the tree description language of tgrep (Pito, 1994) and actions transform the tree, e.g., moving, adjoining, copying or deleting auxiliary trees or relabelling nodes. Alternative programs to transform trees with scripts are CorpusSearch (Randall, 2009) and TTT (Purtee and Schubert, 2012).

3 A sketch of the generation procedure

This section sketches generation of a canonical sentence with a transitive verb:

(1) Girls see a boy.

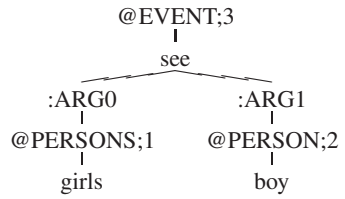
A typical Davidsonian meaning representation for (1) is as follows:

```
∃ EVENT[3] PERSON[2] PERSONS[1]
(girls(PERSONS[1]) ∧ boy(PERSON[2])
∧ see(EVENT[3], PERSONS[1],
PERSON[2]))
```

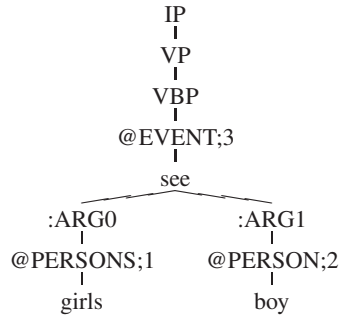
With information for argument roles (‘`:ARG0`’ and ‘`:ARG1`’) sourced from the arity of the ‘`see`’ predicate, the same content converted to a Penman representation is as follows:

```
( EVENT-3 / see
      :ARG0 ( PERSONS-1 / girls)
      :ARG1 ( PERSON-2 / boy))
```

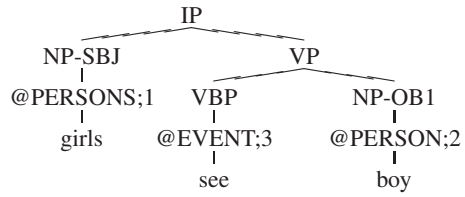
Already the Penman notation provides a base for growing tree structure:



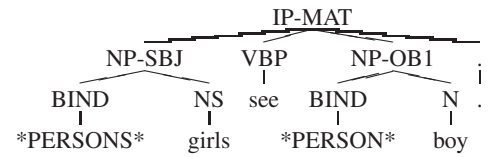
Clause structure is built by adjoining VBP, VP and IP layers to nodes beginning with ‘@EVENT’.



Next, arguments of what has been made the main predicate are moved to populate the clause, with ‘`:ARG1`’ as the object inside VP, while ‘`:ARG0`’ creates the subject outside VP.



With arguments in place, it is safe to remove the VP layer and type the clause as IP-MAT (matrix clause), as well as add punctuation. Entity information is retained with BIND, which also contributes to the projection of noun part-of-speech tags (N; singular vs. NS; plural).



Noun phrases are left bare when indefinite, with the assumption that further post-processing might add an indefinite determiner (*a* or *an*) when the noun head is singular.

4 Different constructions

This section is concerned with how the make up of a meaning representation can determine generation of particular English language constructions as PPCMBE trees, with both meaning representation content and how the content is packaged influencing output.

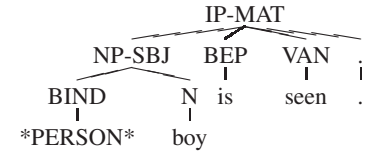
4.1 Passivisation

With '`:ARG0`' missing, but '`:ARG1`' present, the content of '`:ARG1`' can be taken to form the grammatical subject to create a passive clause with the verb tag altered to VAG (passive participle) and BEP (present tense copula) added.

(2) A boy is seen.

```
∃ EVENT[2] PERSON[1] (boy(PERSON[1])
∧ seen(EVENT[2], _, PERSON[1]))
```

```
( EVENT-2 / seen
      :ARG1 ( PERSON-1 / boy))
```



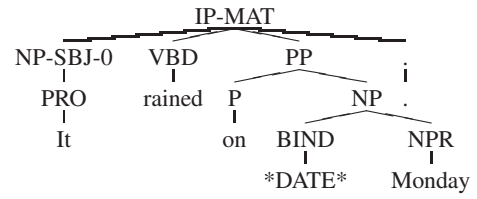
4.2 Expletive it

If there is no core argument ('`:ARG0`', '`:ARG1`', or '`:ARG2`') then expletive *it* should be created to fulfil the grammatical subject role.

(3) It rained on Monday.

```
∃ EVENT[1] (past(EVENT[1]) ∧
rained(EVENT[1]) ∧ on(EVENT[1]) =
DATE[Monday])
```

```
( EVENT-1 / rained
      :MOD ( mod-1 / past)
      :ON ( DATE-Monday / DATE
             :name ( n-2 / name
                    :op1 "Monday")))
```



This example also demonstrates creation of a PP adjunct from an '`:ON`' argument, as well as the effect of `past` tense information altering the verb tag to VBD (past tense verb).

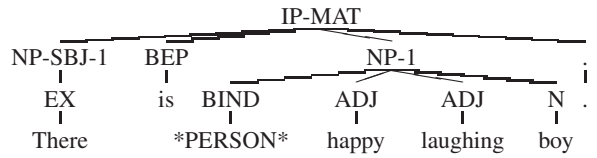
4.3 Existential construction

Another variation in clause construction occurs when the predicate is the copula and there is no '`:ARG1`', but '`:ARG0`' is present. This triggers creation of a *there* subject that is coindexed with the contribution of '`:ARG0`' captured as a noun phrase that immediately follows the copula.

(4) There is a happy laughing boy.

```
∃ ATTRIB[3] ATTRIB[2] EVENT[4]
PERSON[1] (
 happy(ATTRIB[2]) ∧
  laughing(ATTRIB[3]) ∧
is_boy_ATTRIBUTE(PERSON[1],
ATTRIB[3]) ∧ is_boy_ATTRIBUTE(PERSON[1],
ATTRIB[2]) ∧ copula(EVENT[4],
PERSON[1]))
```

```
( EVENT-4 / copula
   :ARG0 ( PERSON-1 / boy
    :ATTRIBUTE ( ATTRIB-3 / laughing)
    :ATTRIBUTE ( ATTRIB-2 / happy)))
```



4.4 Discourse

A discourse is created when there is content for two or more clauses that are conjuncts of `multi-sentence` in the Penman representation.

(5) A boy is happy. He laughs.

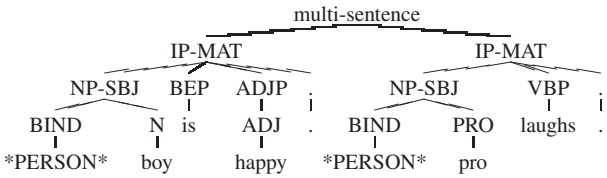
```
∃ PERSON[3] EVENT[2] EVENT[4]
PERSON[1] (boy(PERSON[1]) ∧ PERSON[3]
= PERSON[1] ∧ copula_happy(EVENT[2],
PERSON[1]) ∧ laughs(EVENT[4],
PERSON[3]))
```

The presence of the equative link '`PERSON[3] = PERSON[1]`' is a sufficient clue when converting

from the Davidsonian predicate language formula to create conjuncts conjoined with `multi-sentence`.

```
( CONJ-1 / multi-sentence
      :snt1 ( EVENT-2 / copula_happy
             :ARG0 ( PERSON-1 / boy))
      :snt2 ( EVENT-4 / laughs
             :ARG0 PERSON-1))
```

With there being multiple argument roles for the same entity in distinct conjuncts, it should be the first instance in the Penman representation that is populated with information about the entity, e.g., `(PERSON-1 / boy)`, while subsequent instances are bare references, e.g., `PERSON-1`. It is such a bare reference that leads to the creation of a pronoun with the generated output.



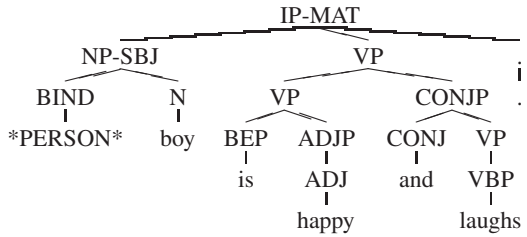
4.5 VP coordination

A relation projecting a label beginning `CONJ` is a relation of coordination. Such a relation name that is not `multi-sentence` is the foundation for forming intra sentential coordination. If the content for arguments that become subjects is shared between conjuncts then VP coordination is established to share the same subject.

(6) A boy is happy and laughs.

```
∃ EVENT[2] EVENT[3]
PERSON[1] (boy(PERSON[1]) ∧
CONJ_and(copula_happy(EVENT[2],
PERSON[1]), laughs(EVENT[3],
PERSON[1])))
```

```
( CONJ-4 / and
      :op1 ( EVENT-2 / copula_happy
             :ARG0 ( PERSON-1 / boy))
      :op2 ( EVENT-3 / laughs
             :ARG0 PERSON-1))
```



If there are other shared entities between conjuncts formed with the same argument role, then they are projected outside the VP layer, typically to the left, but to the right when the argument role is '`:ARG1`' and the verbs are active (also have '`:ARG0`' arguments), or when the argument is heavy, e.g., containing many terminal nodes.

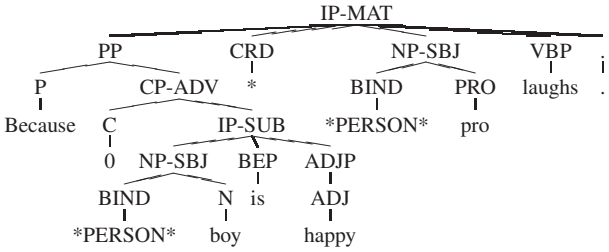
4.6 Adverbial clause

A binary relation projecting a label beginning `CND` (conditional) or `CRD` (coordinating relation) creates an adverbial clause from the first conjunct, with the relation name forming a subordinate conjunction (e.g., *if*, *when*, *unless*, *although*, *because*) that introduces the adverbial clause to a containing clause formed from the content of the second conjunct. If the same entity fills arguments in distinct conjuncts, it should be the instance in what will form the adjunct clause that is populated with information about the entity, while subsequent instances are bare references, with bare references leading to the creation of pronouns. This is similar to the treatment of bare references when there is a `multi-sentence` relation creating discourse (see section 4.4).

(7) Because a boy is happy he laughs.

```
∃ PERSON[3] EVENT[2] EVENT[4]
PERSON[1] (
 boy(PERSON[1]) ∧
  PERSON[3] = he{PERSON[1]} ∧
CRD_Because(copula_happy(EVENT[2],
PERSON[1]), laughs(EVENT[4],
PERSON[3])))
```

```
( CRD-5 / Because
      :op1 ( EVENT-2 / copula_happy
             :ARG0 ( PERSON-1 / boy))
      :op2 ( EVENT-4 / laughs
             :ARG0 PERSON-1))
```



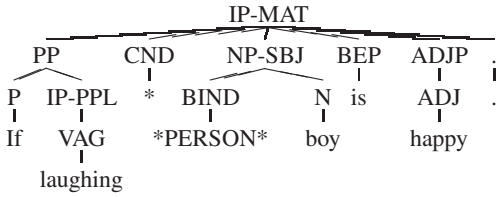
4.7 Participal clause

Instead of an adverbial clause being created from the first conjunct of a binary relation projecting a label beginning `CND` or `CRD`, a participal clause with a controlled subject is created when there is a bare reference for the '`:ARG0`' of the first conjunct that is coreferential with the content of a core argument ('`:ARG0`', '`:ARG1`' or '`:ARG2`') of the second conjunct.

(8) If laughing a boy is happy.

```
∃ EVENT[2] EVENT[3]
PERSON[1] (boy(PERSON[1]) ∧
CND_If(laughing(EVENT[2], PERSON[1]),
copula_happy(EVENT[3], PERSON[1])))
```

```
( CND-4 / If
    :op1 ( EVENT-2 / laughing
           :ARG0 PERSON-1)
    :op2 ( EVENT-3 / copula_happy
           :ARG0 ( PERSON-1 / boy)))
```



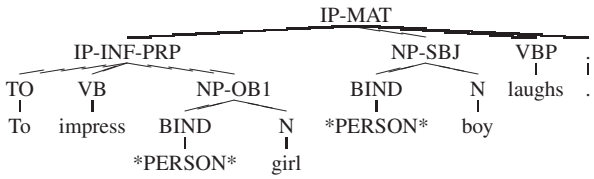
4.8 Purpose clause

A to-infinitive clause can function as an adverbial expressing ideas of purpose or outcome. The content for such a clause falls under a '`:PRP`' tag, while inside there is '`:ARG0`' containing a bare reference coreferential with (i.e., controlled by) the content of the '`:ARG0`' of the containing clause. This creates a subjectless IP-INF-PRP projection containing `(TO to)` and a non-finite verb (`VB`).

(9) To impress a girl a boy laughs.

```
∃ PRP[2] EVENT[4] EVENT[5] PERSON[3]
PERSON[1] (
 boy(PERSON[1]) ∧
  girl(PERSON[3]) ∧
   is_FACT_THAT(PRP[2],
impress(EVENT[4], PERSON[1],
PERSON[3])) ∧
    laughs(EVENT[5], PERSON[1]) ∧
PRP(EVENT[5]) = PRP[2])
```

```
( EVENT-5 / laughs
   :ARG0 ( PERSON-1 / boy)
   :PRP ( EVENT-4 / impress
          :ARG0 PERSON-1
          :ARG1 ( PERSON-3 / girl)))
```



4.9 Infinitive clause with long distance dependency

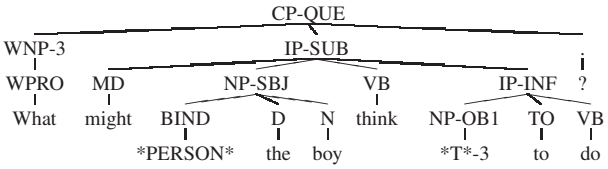
Content for an embedded to-infinitive clause falls under a '`:TOCOMP`' tag, while inside there is '`:ARG0`' containing a bare reference coreferential with (i.e., controlled by) the content of the '`:ARG2`' if present, or alternatively '`:ARG1`' if present, or alternatively '`:ARG0`' of the containing clause. This creates a subjectless IP-INF projection containing `(TO to)` and a non-finite predicate.

(10) What might the boy think to do?

```
∃ ATTRIB[2] EVENT[4] EVENT[3]
PERSON[1] (
 the(ATTRIB[2]) ∧
  is_boy_definite(PERSON[1],
ATTRIB[2]) ∧
   QUEST(MD_might(think_TOCOMP(EVENT[3],
PERSON[1], do(EVENT[4], PERSON[1],
ENTITY[penman2-unknown])))))
```

```
( EVENT-3 / think
   :domain-of ( QUEST-6 / QUEST)
   :domain-of ( MD-5 / might)
   :ARG0 ( PERSON-1 / boy
     :DEFINITE ( ATTRIB-2 / the))
   :TOCOMP ( EVENT-4 / do
```

```
      :ARG0 PERSON-1
      :ARG1 ( ENTITY_UNK-2 /
penman-unknown)))
```



The example also illustrates how a long distance dependency is established with '`:ARG1 (ENTITY_UNK-2 / penman-unknown)`' forming the foundation for an object noun phrase trace `(NP-OB1 *T*-3)` that is coindexed with a WH-phrase `(WNP-3 (WPRO What))` that is placed as the highest constituent of the first commanding question scope marker: '`:domain-of (QUEST-6 / QUEST)`'.

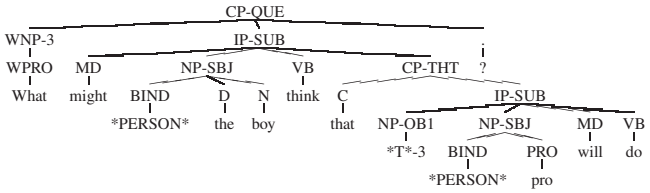
4.10 Embedded clause with long distance dependency

An embedded clause rather than a to-infinitive is established with clause content placed under '`:THAT`'. Inside the content for the embedded clause, bare references form foundations for pronouns.

(11) What might the boy think that he will do?

```
∃ PERSON[4] ATTRIB[2] EVENT[5]
EVENT[3] PERSON[1] (
 the(ATTRIB[2]) ∧
  is_boy_definite(PERSON[1],
ATTRIB[2]) ∧
   PERSON[4] = he{PERSON[1]} ∧
    QUEST(MD_might(think_THAT(EVENT[3],
PERSON[1], MD_will(do(EVENT[5],
PERSON[4], ENTITY[penman2-unknown])))))))
```

```
( EVENT-3 / think
    :domain-of ( QUEST-9 / QUEST)
    :domain-of ( MD-8 / might)
    :ARG0 ( PERSON-1 / boy
        :DEFINITE ( ATTRIB-2 / the))
    :THAT ( EVENT-5 / do
        :domain-of ( MD-7 / will)
        :ARG0 PERSON-1
        :ARG1 ( ENTITY_UNK-2 /
penman-unknown)))
```



The example again illustrates a long distance dependency established out of the embedding.

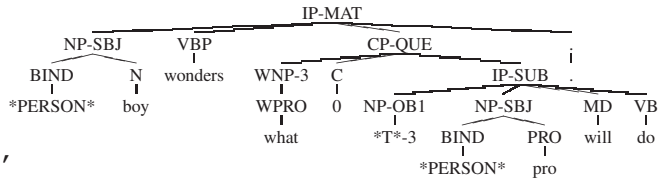
4.11 Embedded question

Having the scope marker for a question local to an embedded clause results in an embedded question.

(12) A boy wonders what he will do.

```
∃ PERSON[3] EVENT[4] EVENT[2]
PERSON[1] (
 boy(PERSON[1]) ∧
  PERSON[3] = he{PERSON[1]} ∧
   wonders_THAT(EVENT[2], PERSON[1],
QUEST(MD_will(do(EVENT[4], PERSON[3],
ENTITY[penman2-unknown])))))
```

```
( EVENT-2 / wonders
   :ARG0 ( PERSON-1 / boy)
   :THAT ( EVENT-4 / do
       :domain-of ( QUEST-8 / QUEST)
       :domain-of ( MD-7 / will)
       :ARG0 PERSON-1
       :ARG1 ( ENTITY_UNK-2 /
penman-unknown)))
```



Together with the examples of section 4.9 and 4.10, this demonstrates how it is not enough for a meaning representation to mark a WH question with `penman-unknown` alone, but that scope marking the level of structure to place a fronted WH phrase is also vital.

4.12 Small clause

Small clauses are embedded clauses that occur with the absence of a finite verb.

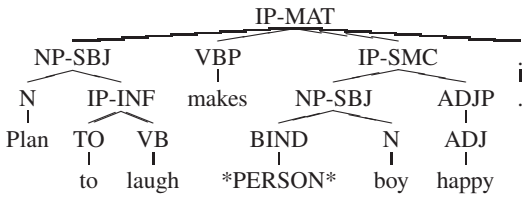
(13) A plan to laugh makes a boy happy.

Lack of a finite verb is reflected by creation of a dummy '`EVENT`' predicate that connects the subject *a boy* to the *happy* attribute in the following Davidsonian formula:

```
∃ ATTRIB[5] PERSON[4] ENTITY[1]
EVENT[2] EVENT[6] EVENT[3] (
 is_plan_TOCOMP(ENTITY[1],
laugh(EVENT[2])) ∧
 boy(PERSON[4]) ∧ happy(ATTRIB[5]) ∧
 makes_TOCOMP(EVENT[3], ENTITY[1],
EVENT(EVENT[6], PERSON[4],
ATTRIB[5])))
```

```
( EVENT-3 / makes
  :ARG0 ( ENTITY-1 / plan
     :TOCOMP ( EVENT-2 / laugh))
  :TOCOMP ( EVENT-6 / EVENT
     :ATTRIBUTE ( ATTRIB-5 / happy)
     :ARG0 ( PERSON-4 / boy)))
```

This leads to projection of an IP-SMC embedded clause with no verb/copula creation.



This example also illustrates creation of a nominal with an infinitive clause embedding, triggered by '`TOCOMP`'.

4.13 Relative clause

The example of this section demonstrates creation of a relative clause with a long distance dependency:

(14) Every boy that a girl says is happy laughs.

There is nothing overt to signal a relative clause with the Davidsonian formula:

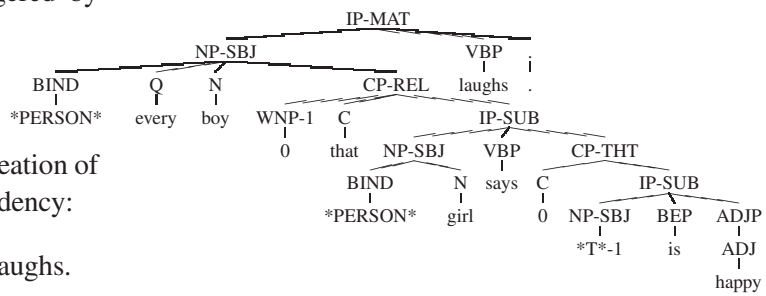
```
∃ ATTRIB[2] EVENT[6] EVENT[5] EVENT[7]
PERSON[4] PERSON[1] (
 every(ATTRIB[2]) ∧
  girl(PERSON[4]) ∧
   says_THAT(EVENT[5], PERSON[4],
copula_happy(EVENT[6], PERSON[1]))
∧ is_boy_QUANTIFIER(PERSON[1],
ATTRIB[2]) ∧
    laughs(EVENT[7], PERSON[1]))
```

But a base for realising a relative clause does emerge when there is conversion to Penman notation:

```
( EVENT-7 / laughs
  :ARG0 ( PERSON-1 / boy
   :QUANTIFIER ( ATTRIB-2 / every)
   :ARG0-of ( EVENT-6 / copula_happy
      :THAT-of ( EVENT-5 / says
        :ARG0 ( PERSON-4 / girl)))))
```

A key requirement for Penman notation is to connect all content around a single rooted node. This privileged node will typically form the main predicate following generation. While not necessary for being a Davidsonian forumla, a convention can be followed to place such a privileged predicate as the most right-side predicate of the formula (so, '`laughs`' of the example). Connection to this single rooted predicate is possible by folding Penman material around inverse roles (signalled by ending a role name with '`-of`') which serves to compact the long distance dependency of the relative clause.

Generation consists of unfolding the dependency, so taking '`-of`' content and reintegrating the content with clausal embedded structure:



4.14 It cleft

As a final example, consider generation of an it cleft:

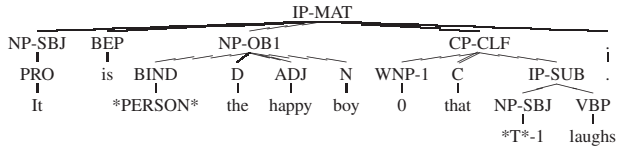
(15) It is the happy boy that laughs.

Such a cleft sentence leads to the presence of a copula predicate that connects 'ARG1' content to an 'ARG0' entity that, with conversion to Penman notation, is given a dummy 'ENTITY' head and is inversed linked to material sufficient to create a clause:

```
∃ ATTRIB[3] ATTRIB[2] ENTITY[4]
EVENT[6] EVENT[7] PERSON[1] (
 the(ATTRIB[2]) ∧
  happy(ATTRIB[3]) ∧
   is_boy_ATTRIBUTE(PERSON[1],
ATTRIB[3]) ∧ is_boy_definite(PERSON[1],
ATTRIB[2]) ∧
    laughs(EVENT[6], ENTITY[4])
∧ copula(EVENT[7], ENTITY[4],
PERSON[1]))
```

```
( EVENT-7 / copula
  :ARG0 ( ENTITY-4 / ENTITY
    :ARG0-of ( EVENT-6 / laughs))
  :ARG1 ( PERSON-1 / boy
    :DEFINITE ( ATTRIB-2 / the)
    :ATTRIBUTE ( ATTRIB-3 / happy)))
```

This leads to generation of the following tree:



Internally, the it cleft has the same structure as a relative clause, but externally, it is a daughter of IP.

5 Conclusion

This paper has focused on the generation of various English language constructions from meaning representations. Starting as Davidsonian predicate language formulas converted to Penman representations, meaning representations were changed to trees to form the basis for generation that proceeds with successive tree structure changes.

Language generation raises the issue of how to choose between the many ways a language offers to present content. The novel contribution of this paper has been to demonstrate how there can be a significant role for meaning representations to play in influencing the selection of grammatical constructions with the arrangement of information content, notably, handling distinctions of clause type as well as the choice between discourse, or (VP) coordination, or projection of an adverbial clause, or participial clause, or creation of a small clause, or infinitive embedding, or finite embedding, or relative clause, or it cleft. This has simplified what it takes to create a generation component capable of rich, varied and natural output, but also, most importantly, this will ensure there is preservation of meaning.

Preserving meaning is an issue since the constructions on display should rarely be conflated, with, e.g., variation in the placement of noun phrase restriction material, or coordinate material, or adjunct material, or embedded material typically giving differences in meaning. Arguably, results of this paper come at the cost of making the creation of meaning representations a more complex task. In this regard, the approach is especially suitable as a component of a machine translation system with meaning representations as the level for language transfer, since the source language of translation will offer rich information to drive how the content of a derived meaning representation is presented.

Acknowledgements

This paper has benefitted from the comments of three anonymous reviewers, as well as discussions with Pascual Martínez-Gómez, Masaaki Nagata and Kei Yoshimoto, all of which is very gratefully acknowledged. This research is supported by the Japan Society for the Promotion of Science (JSPS), Research Project Number: 15K02469.

References

Hiyan Alshawi. 1992. *The Core Language Engine*. Cambridge, Mass.: MIT Press.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, page 1699–1710. Lisbon, Portugal.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Tech. Rep. MS-CIS-95-06, LINC LAB 281, University of Pennsylvania Computer and Information Science Department.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In J. Bos and R. Delmonte, eds., *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

Alastair Butler. 2015a. *Linguistic Expressions and Semantic Processing: A Practical Approach*. Heidelberg: Springer-Verlag.

Alastair Butler. 2015b. Round-trips with meaning stopovers. In *Proceedings of the 1st Workshop on Semantics-Driven Statistical Machine Translation*, pages 1–10. Beijing, China: Association for Computational Linguistics.

Noam Chomsky. 1970. Remarks on nominalization. In R. Jacobs and P. Rosenbaum, eds., *Readings in English Transformational Grammar*, pages 184–221. Waltham, MA: Ginn.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2006. Minimala Recursion Semantics: an introduction. *Research on Language and Computation* 3(4):281–332.

Donald Davidson. 1967. The logical form of action sentences. In N. Rescher, ed., *The Logic of Decision and Action*. Pittsburgh: University of Pittsburgh Press. Reprinted in: D. Davidson, 1980. *Essays on Actions and Events*. Claredon Press, Oxford, pages 105–122.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of the ACL 2014*.

Kevin Humphreys, Mike Calcagno, and David Weise. 2001. Reusing a statistical language model for generation. In *Proceedings of the 8th European workshop on Natural Language Generation - Volume 8*, pages 1–6. Stroudsburg, PA: Association for Computational Linguistics.

Anthony Kroch, Beatrice Santorini, and Ariel Diertani. 2010. *The Penn-Helsinki Parsed Corpus of Modern British English (PPCMBE)*. Department of Linguistics, University of Pennsylvania. CD-ROM, second edition, (http://www.ling.upenn.edu/hist-corpora).

Fred Landman. 2000. *Events and Plurality: The Jerusalem Lectures*. Dordrecht: Kluwer Academic Publishers.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the ACL/COLING-98*. Montreal, Quebéc.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structure. In *5th International conference on Language Resources and Evaluation*.

Christian Matthiessen and John A Bateman. 1991. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Pinter Publishers.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 2055–2061. Lisbon, Portugal.

Terence Parsons. 1990. *Events in the Semantics of English*. Cambridge: MIT Press.

Richard Pito. 1994. tgrepdoc - documentation for tgrep.

A. Purtee and Lenhart K. Schubert. 2012. TTT: A tree transduction language for syntactic and semantic processing. In *EACL 2012 Workshop on Applications of Tree Automata Techniques in Natural Language Processing (ATANLP 2012)*. Avignon, France.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154. Lisbon, Portugal.

Beth Randall. 2009. CorpusSearch 2 Users Guide. (http://corpussearch.sourceforge.net/CS-manual/Contents.html).