Three reasons to adopt TAG-based surface realisation

Claire Gardent
CNRS/LORIA
615, rue du Jardin Botanique
F-54600 Villers-Les-Nancy
gardent@loria.fr

Eric Kow
INRIA/LORIA
Universite Henri Poincare
615, rue du Jardin Botanique
F-54600 Villers-Les-Nancy
kow@loria.fr

Surface Realisation (Parsing in reverse?)

Input: grammar/lexicon + logical formula (input semantics). e.g.

`like(l,f,m), faye(f), music(m)`

Output: sentences in that grammar, e.g,

Faye likes music

Substitution before Adjunction

Problem: Lack of ordering information. The input to surface realisation is a set of literals. Supposing each literal selects exactly one constituent in the lexicon, then the number of possible combinations between these constituents will be 2^n, where n is the size of the input semantics (number of literals).

Why TAG: Substitution and adjunction apply independently of each other. Two phase generation strategy (modifiers only added to complete syntactic trees)

1) Do all substitutions, then discard all unsaturated trees (trees with unfilled substitution sites).

2) Do all adjunctions.

Result: Modifiers only combine with syntactically complete S-trees (i.e., where all substitution sites are filled).

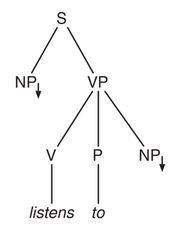
Extended domain of locality

Problem: Words with empty semantics (e.g., complementiser that, infinitival to). All trees with an empty semantics be considered as potential constituent candidate at each combining step (naive approach). This means increasing the size of the input n.

Why TAG: Can treat them as co-anchors in a TAG elementary tree

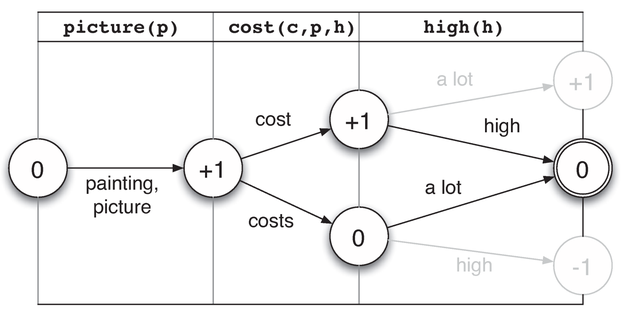
Result: No special treatment, and no impact on efficiency

Co-anchor example



literals	improvement chart items	improvement CPU time
≤ 3	1.1×	1.0×
4	1.0×	1.0×
5	1.2×	1.0×
6	1.6×	3.3×
7	3.0×	3.3×
≥ 8	Time out	

Effect of substitution-before-adjunction



Polarity automaton example

literals	improvement ambiguity	improvement chart items	improvement CPU time
1-6	35.6×	1.8×	1.0×
7-9	161.3×	2.6×	1.2×
10-13	313.9×	10.6×	1.9×
14-16	441.6×	10.8×	5.3×

Effect of polarity filtering
(substitution-before-adjunction as baseline)

Polarity filtering

Problem: lexical ambiguity. One semantic literal [e.g. like(l,m,f)] might be associated with more than one lexical entries. Total ambiguity is product of ambiguities for each literal!

Why TAG: Root nodes and substitution sites can be used as basis for "polarity filtering".

1) Each lexical item assigned to a bag of polarities:
   +R for root node category R (resource)
   -S for each substitution site, where S is its category (requirement)
2) Build polarity automaton representing all lexical combinations and its net polarity.
3) Only do surface realisation on combinations with net polaritity of zero.

Result: Effect of lexical ambiguity greatly reduced (on largest problems: 660 instead of 290 000 combinations, 2.21 instead of 11.6 seconds)