

Derivation Trees of a Tree Adjoining Grammar

Sylvain Schmitz

LORIA, INRIA Nancy - Grand Est, Nancy, France

MOSTRARE Seminars, April 18, 2008

Contents

Tree Adjoining Grammars

- Principles

- Feature Unification

Two-level Syntax

- Derivation Tree Languages

Surface Realization

- Targeting Derivation Trees

- Feature Unification in Derivation Trees

Test Trees

- Overgeneration

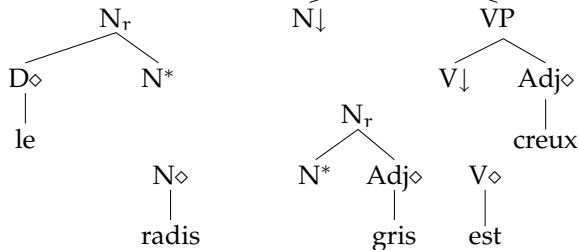
- “Exhaustive” Generation (work in progress)

Conclusion

Tree Adjoining Grammars

Joshi et al. [1975], Joshi and Schabes [1997]

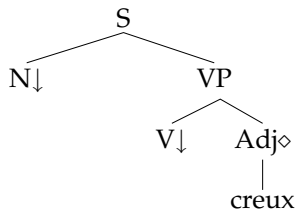
Elementary trees:



Tree Adjoining Grammars

Joshi et al. [1975], Joshi and Schabes [1997]

Two operations:

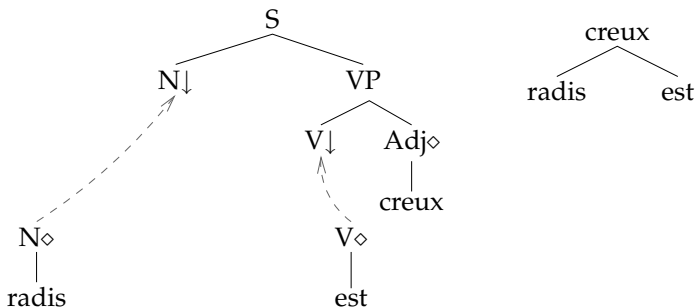


creux

Tree Adjoining Grammars

Joshi et al. [1975], Joshi and Schabes [1997]

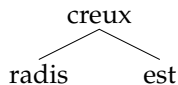
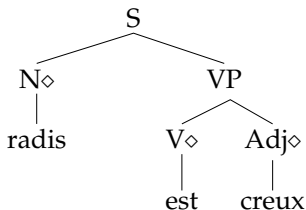
Two operations: substitution:



Tree Adjoining Grammars

Joshi et al. [1975], Joshi and Schabes [1997]

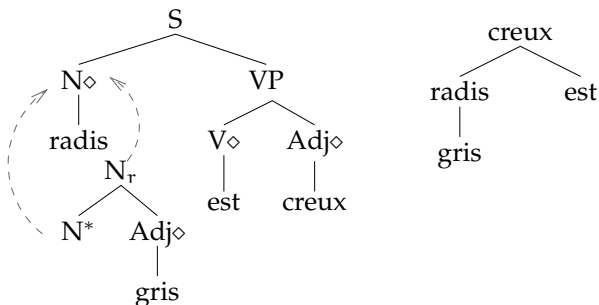
Two operations: substitution:



Tree Adjoining Grammars

Joshi et al. [1975], Joshi and Schabes [1997]

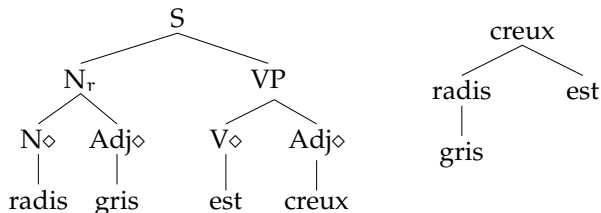
Two operations: adjunction:



Tree Adjoining Grammars

Joshi et al. [1975], Joshi and Schabes [1997]

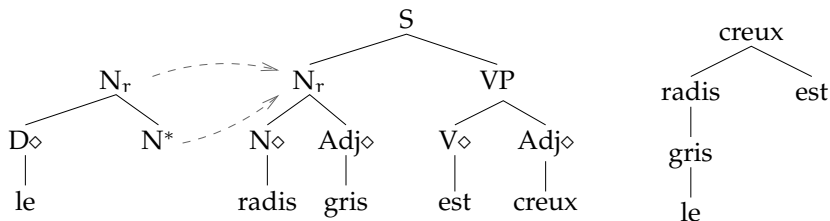
Two operations: adjunction:



Tree Adjoining Grammars

Joshi et al. [1975], Joshi and Schabes [1997]

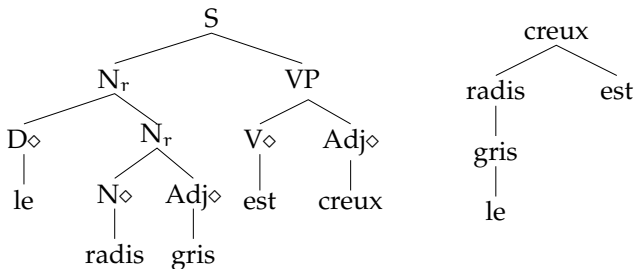
Two operations: adjunction:



Tree Adjoining Grammars

Joshi et al. [1975], Joshi and Schabes [1997]

Derived and derivation trees:



Formal Properties

Strings

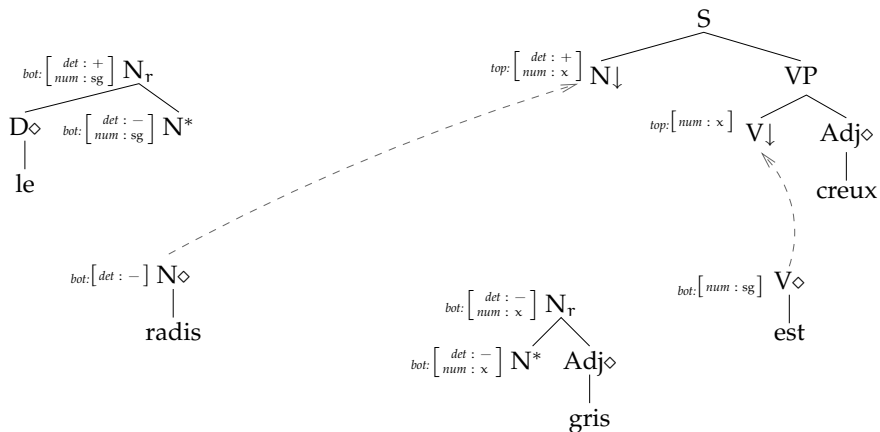
- ▶ beyond context-free: *mildly context-sensitive*
- ▶ polynomial time parsing: $\mathcal{O}(|G| n^6)$

Trees

- ▶ beyond regular
- ▶ a restricted class of context-free tree languages:
linear and *monadic* [Kepster and Rogers, 2007]

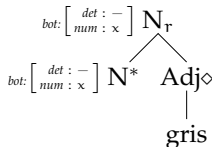
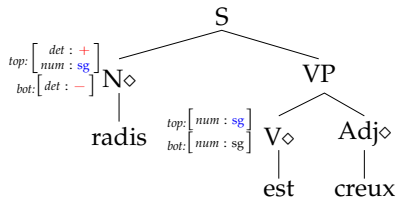
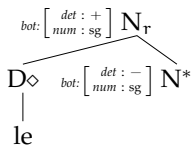
Feature-Based TAG

Vijay-Shanker and Joshi [1988]



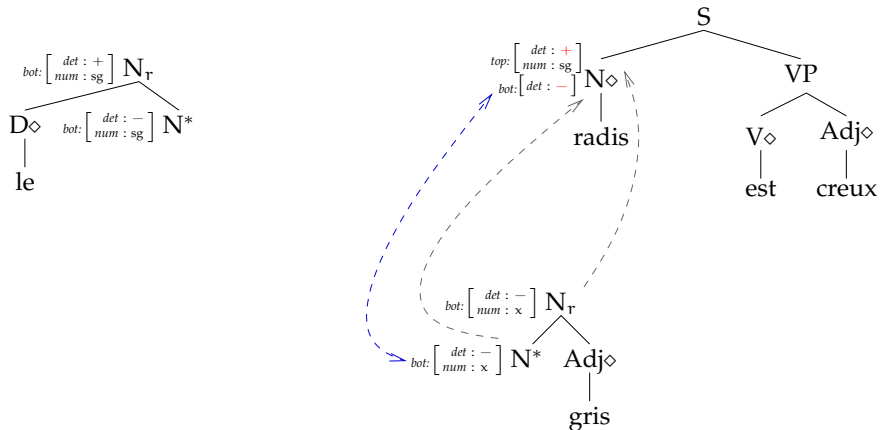
Feature-Based TAG

Vijay-Shanker and Joshi [1988]



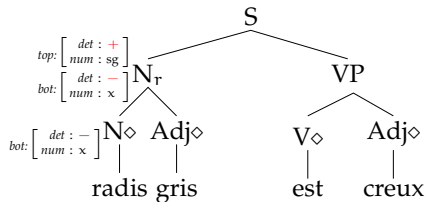
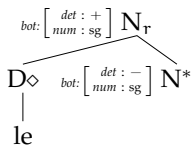
Feature-Based TAG

Vijay-Shanker and Joshi [1988]



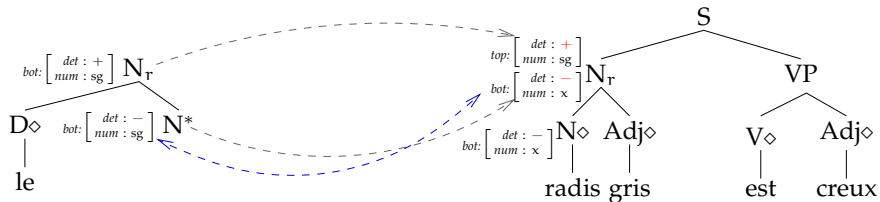
Feature-Based TAG

Vijay-Shanker and Joshi [1988]



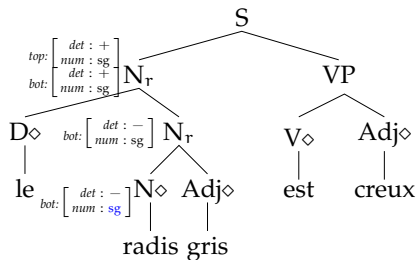
Feature-Based TAG

Vijay-Shanker and Joshi [1988]



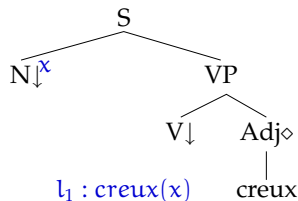
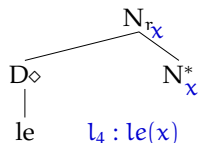
Feature-Based TAG

Vijay-Shanker and Joshi [1988]

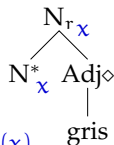


Semantics Through Unification

Gardent and Kallmeyer [2003]



$l_2 : radis(r)$



$l_3 : gris(x)$

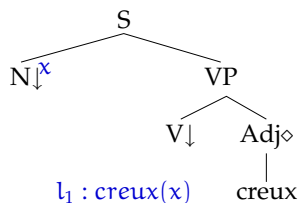
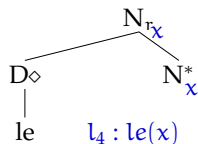


Yields flat semantics:

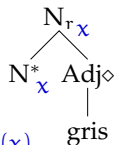
$l_1 : creux(r) \wedge l_2 : radis(r) \wedge l_3 : gris(r) \wedge l_4 : le(r)$

Semantics Through Unification

Gardent and Kallmeyer [2003]



$l_2 : radis(r)$



Yields flat semantics:

$l_1 : creux(r) \wedge l_2 : radis(r) \wedge l_3 : gris(r) \wedge l_4 : le(r)$

Two-level Syntax

Pogodalla [2004], Shieber [2006], and many others

- ▶ derivation trees as intermediate representations
- ▶ morphisms / transductions into
 1. derived trees (or strings)
 2. semantics
- ▶ inverse morphisms / transductions for parsing and generation

Derivation Tree Language (1)

(reduced) Regular tree grammar

$$S_I \rightarrow \text{creux}(S_A, N_I, VP_A, V_I, \text{Adj}_A)$$

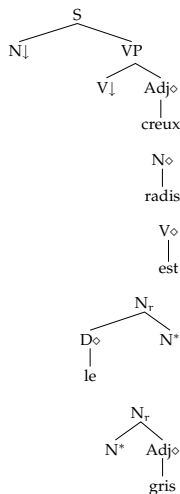
$$N_I \rightarrow \text{radis}(N_A)$$

$$V_I \rightarrow \text{est}(V_A)$$

$$N_A \rightarrow \text{le}(N_A, D_A)$$

$$N_A \rightarrow \text{gris}(N_A, \text{Adj}_A)$$

$$\text{Adj}_A \rightarrow \varepsilon(), D_A \rightarrow \varepsilon(), N_A \rightarrow \varepsilon(), S_A \rightarrow \varepsilon(), V_A \rightarrow \varepsilon(), VP_A \rightarrow \varepsilon()$$



Derivation Tree Language (1)

(reduced) Regular tree grammar

$$S_I \rightarrow \text{creux}(S_A, N_I, VP_A, V_I, \text{Adj}_A)$$

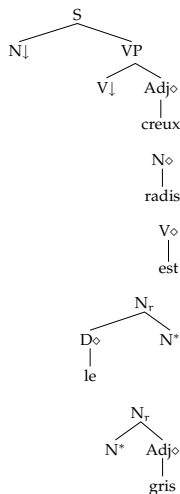
$$N_I \rightarrow \text{radis}(N_A)$$

$$V_I \rightarrow \text{est}(V_A)$$

$$N_A \rightarrow \text{le}(N_A, D_A)$$

$$N_A \rightarrow \text{gris}(N_A, \text{Adj}_A)$$

$$\text{Adj}_A \rightarrow \varepsilon(), D_A \rightarrow \varepsilon(), N_A \rightarrow \varepsilon(), S_A \rightarrow \varepsilon(), V_A \rightarrow \varepsilon(), VP_A \rightarrow \varepsilon()$$



Derivation Tree Language (1)

(reduced) Regular tree grammar

$$S_I \rightarrow \text{creux}(S_A, N_I, VP_A, V_I, \text{Adj}_A)$$

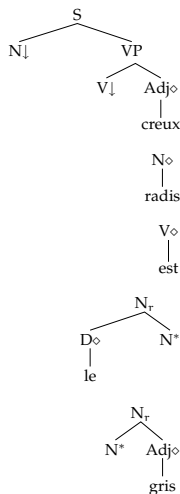
$$N_I \rightarrow \text{radis}(N_A)$$

$$V_I \rightarrow \text{est}(V_A)$$

$$N_A \rightarrow \text{le}(N_A, D_A)$$

$$N_A \rightarrow \text{gris}(N_A, \text{Adj}_A)$$

$$\text{Adj}_A \rightarrow \varepsilon(), D_A \rightarrow \varepsilon(), N_A \rightarrow \varepsilon(), S_A \rightarrow \varepsilon(), V_A \rightarrow \varepsilon(), VP_A \rightarrow \varepsilon()$$



Derivation Tree Language (1)

(reduced) Regular tree grammar

$$S_I \rightarrow \text{creux}(S_A, N_I, VP_A, V_I, \text{Adj}_A)$$

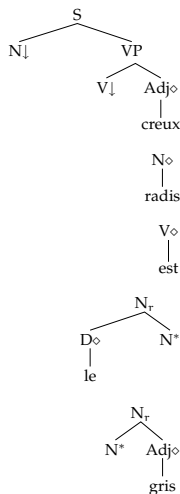
$$N_I \rightarrow \text{radis}(N_A)$$

$$V_I \rightarrow \text{est}(V_A)$$

$$N_A \rightarrow \text{le}(N_A, D_A)$$

$$N_A \rightarrow \text{gris}(N_A, \text{Adj}_A)$$

$$\text{Adj}_A \rightarrow \varepsilon(), D_A \rightarrow \varepsilon(), N_A \rightarrow \varepsilon(), S_A \rightarrow \varepsilon(), V_A \rightarrow \varepsilon(), VP_A \rightarrow \varepsilon()$$



Derivation Tree Language (1)

(reduced) Regular tree grammar

$$S_I \rightarrow \text{creux}(S_A, N_I, VP_A, V_I, \text{Adj}_A)$$

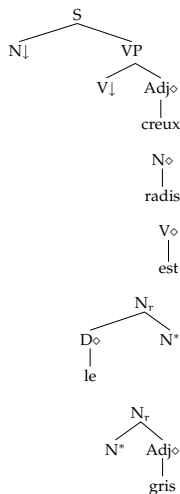
$$N_I \rightarrow \text{radis}(N_A)$$

$$V_I \rightarrow \text{est}(V_A)$$

$$N_A \rightarrow \text{le}(N_A, D_A)$$

$$N_A \rightarrow \text{gris}(N_A, \text{Adj}_A)$$

$$\text{Adj}_A \rightarrow \varepsilon(), D_A \rightarrow \varepsilon(), N_A \rightarrow \varepsilon(), S_A \rightarrow \varepsilon(), V_A \rightarrow \varepsilon(), VP_A \rightarrow \varepsilon()$$



Derivation Tree Language (1)

(reduced) Regular tree grammar

$$S_I \rightarrow \text{creux}(S_A, N_I, VP_A, V_I, \text{Adj}_A)$$

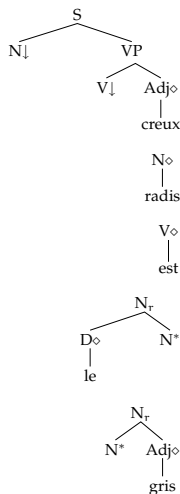
$$N_I \rightarrow \text{radis}(N_A)$$

$$V_I \rightarrow \text{est}(V_A)$$

$$N_A \rightarrow \text{le}(N_A, D_A)$$

$$N_A \rightarrow \text{gris}(N_A, \text{Adj}_A)$$

$$\text{Adj}_A \rightarrow \varepsilon(), D_A \rightarrow \varepsilon(), N_A \rightarrow \varepsilon(), S_A \rightarrow \varepsilon(), V_A \rightarrow \varepsilon(), VP_A \rightarrow \varepsilon()$$



Derivation Tree Language (1)

(reduced) Regular tree grammar

$$S_I \rightarrow \text{creux}(S_A, N_I, VP_A, V_I, \text{Adj}_A)$$

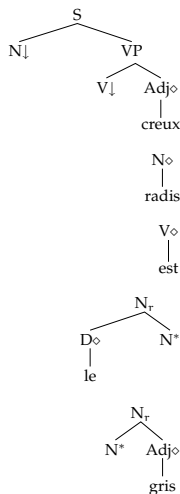
$$N_I \rightarrow \text{radis}(N_A)$$

$$V_I \rightarrow \text{est}(V_A)$$

$$N_A \rightarrow \text{le}(N_A, D_A)$$

$$N_A \rightarrow \text{gris}(N_A, \text{Adj}_A)$$

$$\text{Adj}_A \rightarrow \varepsilon(), D_A \rightarrow \varepsilon(), N_A \rightarrow \varepsilon(), S_A \rightarrow \varepsilon(), V_A \rightarrow \varepsilon(), VP_A \rightarrow \varepsilon()$$



Derivation Tree Language (1)

(reduced) Regular tree grammar

$$S_I \rightarrow \text{creux}(S_A, N_I, VP_A, V_I, Adj_A)$$

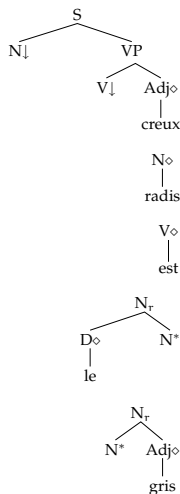
$$N_I \rightarrow \text{radis}(N_A)$$

$$V_I \rightarrow \text{est}(V_A)$$

$$N_A \rightarrow \text{le}(N_A, D_A)$$

$$N_A \rightarrow \text{gris}(N_A, Adj_A)$$

$$Adj_A \rightarrow \varepsilon(), D_A \rightarrow \varepsilon(), N_A \rightarrow \varepsilon(), S_A \rightarrow \varepsilon(), V_A \rightarrow \varepsilon(), VP_A \rightarrow \varepsilon()$$



From Derivations to Derived Trees

Macro tree transducer [Shieber, 2006]

$$q_I(\text{creux}(x_1, x_2)) \rightarrow S(q_I(x_1), VP(q_I(x_2), \text{Adj}(\text{creux})))$$

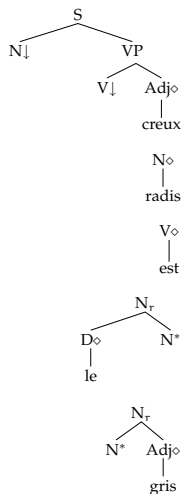
$$q_I(\text{radis}(x)) \rightarrow q_A(x, N(\text{radis}))$$

$$q_I(\text{est}()) \rightarrow V(\text{est})$$

$$q_A(\text{le}(x), y) \rightarrow q_A(x, N(D(\text{le}), y))$$

$$q_A(\text{gris}(x), y) \rightarrow q_A(x, N(y, \text{Adj}(\text{gris})))$$

$$q_A(\varepsilon(), y) \rightarrow y$$



From Derivations to Derived Trees

Macro tree transducer [Shieber, 2006]

$$q_I(\text{creux}(x_1, x_2)) \rightarrow S(q_I(x_1), VP(q_I(x_2), \text{Adj}(\text{creux})))$$

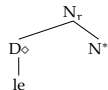
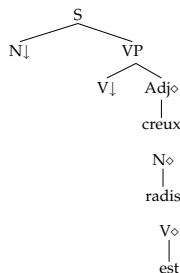
$$q_I(\text{radis}(x)) \rightarrow q_A(x, N(\text{radis}))$$

$$q_I(\text{est}()) \rightarrow V(\text{est})$$

$$q_A(\text{le}(x), y) \rightarrow q_A(x, N(D(\text{le}), y))$$

$$q_A(\text{gris}(x), y) \rightarrow q_A(x, N(y, \text{Adj}(\text{gris})))$$

$$q_A(\varepsilon(), y) \rightarrow y$$



From Derivations to Derived Trees

Macro tree transducer [Shieber, 2006]

$$q_I(\text{creux}(x_1, x_2)) \rightarrow S(q_I(x_1), VP(q_I(x_2), \text{Adj}(\text{creux})))$$

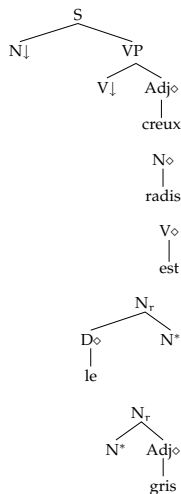
$$q_I(\text{radis}(x)) \rightarrow q_A(x, N(\text{radis}))$$

$$q_I(\text{est}()) \rightarrow V(\text{est})$$

$$q_A(\text{le}(x), y) \rightarrow q_A(x, N(D(\text{le}), y))$$

$$q_A(\text{gris}(x), y) \rightarrow q_A(x, N(y, \text{Adj}(\text{gris})))$$

$$q_A(\varepsilon(), y) \rightarrow y$$



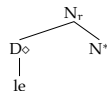
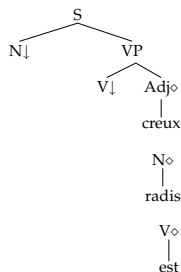
From Derivations to Semantics?

$$q(\text{creux}(x_1, x_2), r) \rightarrow \wedge(\text{creux}(r), q(x_1, r))$$

$$q(\text{radis}(x), r) \rightarrow \wedge(\text{radis}(r), q(x, r))$$

$$q(\text{le}(x), r) \rightarrow \wedge(\text{le}(r), q(x, r))$$

$$q(\text{gris}(x), r) \rightarrow \wedge(\text{gris}(r), q(x, r))$$

$$q(\varepsilon(), r) \rightarrow \text{true}$$


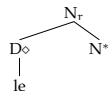
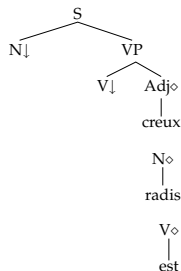
From Derivations to Semantics?

$$q(\text{creux}(x_1, x_2), r) \rightarrow \wedge(\text{creux}(r), q(x_1, r))$$

$$q(\text{radis}(x), r) \rightarrow \wedge(\text{radis}(r), q(x, r))$$

$$q(\text{le}(x), r) \rightarrow \wedge(\text{le}(r), q(x, r))$$

$$q(\text{gris}(x), r) \rightarrow \wedge(\text{gris}(r), q(x, r))$$

$$q(\varepsilon(), r) \rightarrow \text{true}$$


From Derivations to Semantics?

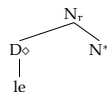
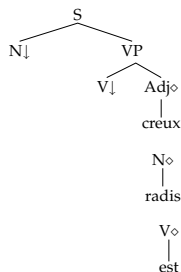
$$q(\text{creux}(x_1, x_2), r) \rightarrow \wedge(\text{creux}(r), q(x_1, r))$$

$$q(\text{radis}(x), r) \rightarrow \wedge(\text{radis}(r), q(x, r))$$

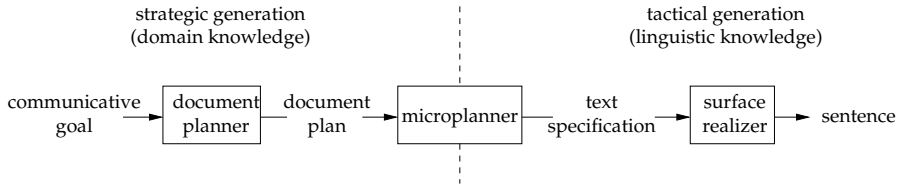
$$q(\text{le}(x), r) \rightarrow \wedge(\text{le}(r), q(x, r))$$

$$q(\text{gris}(x), r) \rightarrow \wedge(\text{gris}(r), q(x, r))$$

$$q(\varepsilon(), r) \rightarrow \text{true}$$



Sentence Generation Pipeline



Surface Realization Algorithms

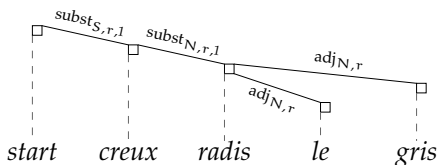
E.g. Carroll et al. [1999], Gardent and Kow [2007], ...

1. select all elementary trees that subsume parts of the input semantics
2. compute all combinations—tabulate intermediate results
3. extract sentences

NP-complete! [Koller and Striegnitz, 2002]

Generation as Dependency Parsing

Koller and Striegnitz [2002]: Topological Dependency Grammar



atom	labels	valency
start	\emptyset	$\{\text{subst}_{S,r,1}\}$
creux	$\{\text{subst}_{S,r,1}\}$	$\{\text{subst}_{N,r,1}\}$
radis	$\{\text{subst}_{N,r,1}\}$	$\{\text{adj}_{N,r}^*\}$
le	$\{\text{adj}_{N,r}\}$	\emptyset
gris	$\{\text{adj}_{N,r}\}$	\emptyset

Generation as Planning

Koller and Stone [2007]: Planning Domain Definition Language

Action start().

Effect: subst(S,r), sem(creux,r), sem(radis,r), sem(le,r), sem(gris,r)

Action creux(r). Precond: sem(creux,r), subst(S,r)

Effect: \neg sem(creux,r), \neg subst(S,r), subst(N,r), subst(V)

Action radis(r). Precond: sem(radis,r), subst(N,r)

Effect: \neg sem(radis,r), \neg subst(N,r), canadjoin(N,r)

Action est(). Precond: subst(V)

Effect: \neg subst(V)

Action le(r). Precond: sem(le,r), canadjoin(N,r)

Effect: \neg sem(le,r), \neg mustadjoin(N,r)

Action gris(r). Precond: sem(gris,r), canadjoin(N,r)

Effect: \neg sem(gris,r), \neg mustadjoin(N,r)

Action end(). Precond: $\forall A,u. \neg$ sem(A,u) \wedge \neg subst(A,u)

\wedge \neg mustadjoin(A,u)

Features in RTG

- ▶ \mathcal{D} : set of feature structures
- ▶ work on terms over $\mathbb{N} \times \mathcal{D}$
- ▶ rules of form $(A, d) \rightarrow a((B_1, d'_1), \dots, (B_n, d'_n))$
- ▶ rewrites $(s, e) \Rightarrow (t, e')$ maintain a global environment for substitutions in feature structures

$$s = C[(A, d)], \quad t = C[a((B_1, \sigma(d'_1)), \dots, (B_n, \sigma(d'_n)))]$$

$$\sigma = \text{mgu}(d, e(d')), \quad e' = \sigma \circ e$$

- ▶ language

$$L(G) = \{t \in T(\mathcal{F}) \mid \exists e, ((S, \top), id) \Rightarrow^* (t, e)\}$$

Derivation Tree Language (2)

$$(S_I, \top) \rightarrow \text{creux} (N_I \left[\begin{smallmatrix} \text{top} : [\text{det} : +] \\ \text{bot} : [\text{num} : x] \end{smallmatrix} \right], V_I \left[\text{top} : [\text{num} : x] \right])$$

$$N_I \left[\text{top} : t \right] \rightarrow \text{radis} (N_A \left[\begin{smallmatrix} \text{top} : t \\ \text{bot} : [\text{det} : -] \end{smallmatrix} \right])$$

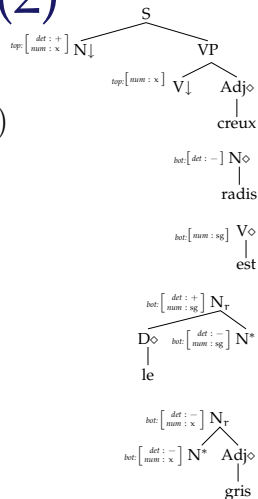
$$V_I \left[\text{top} : t \right] \rightarrow \text{est} (V_A \left[\begin{smallmatrix} \text{top} : t \\ \text{bot} : [\text{num} : \text{sg}] \end{smallmatrix} \right])$$

$$N_A \left[\begin{smallmatrix} \text{top} : t \\ \text{bot} : [\text{det} : -] \\ \text{num} : \text{sg} \end{smallmatrix} \right] \rightarrow \text{le} (N_A \left[\begin{smallmatrix} \text{top} : t \\ \text{bot} : [\text{det} : +] \\ \text{num} : \text{sg} \end{smallmatrix} \right])$$

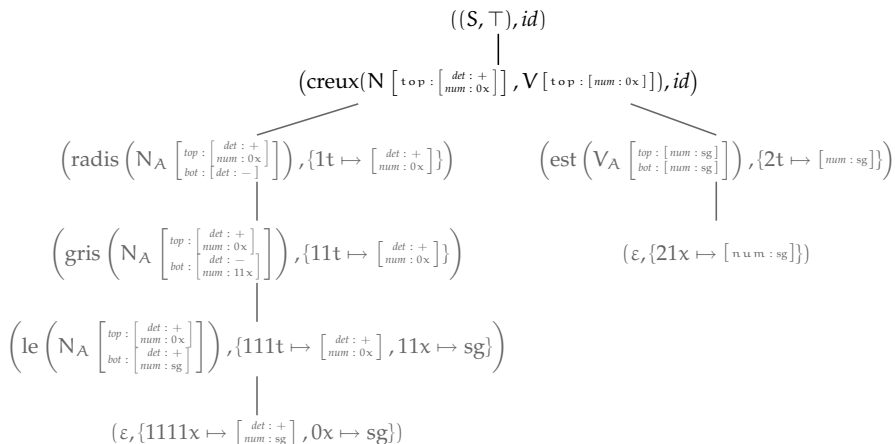
$$N_A \left[\begin{smallmatrix} \text{top} : t \\ \text{bot} : [\text{det} : -] \\ \text{num} : x \end{smallmatrix} \right] \rightarrow \text{gris} (N_A \left[\begin{smallmatrix} \text{top} : t \\ \text{bot} : [\text{det} : -] \\ \text{num} : x \end{smallmatrix} \right])$$

$$N_A \left[\begin{smallmatrix} \text{top} : x \\ \text{bot} : x \end{smallmatrix} \right] \rightarrow \varepsilon()$$

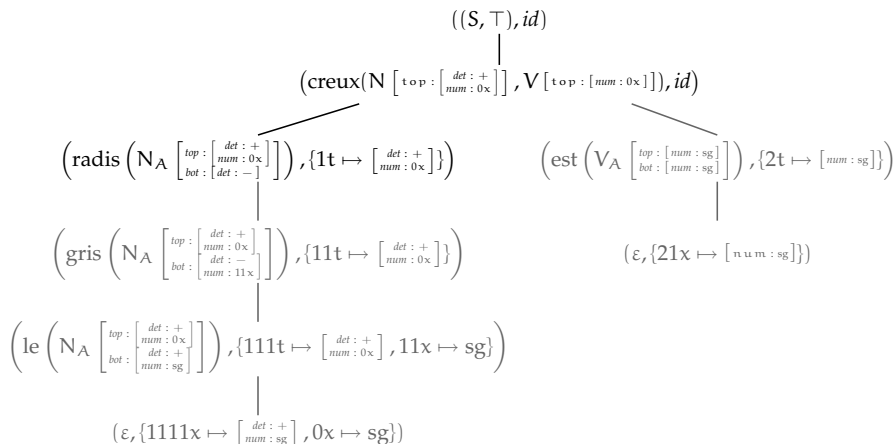
$$V_A \left[\begin{smallmatrix} \text{top} : x \\ \text{bot} : x \end{smallmatrix} \right] \rightarrow \varepsilon()$$



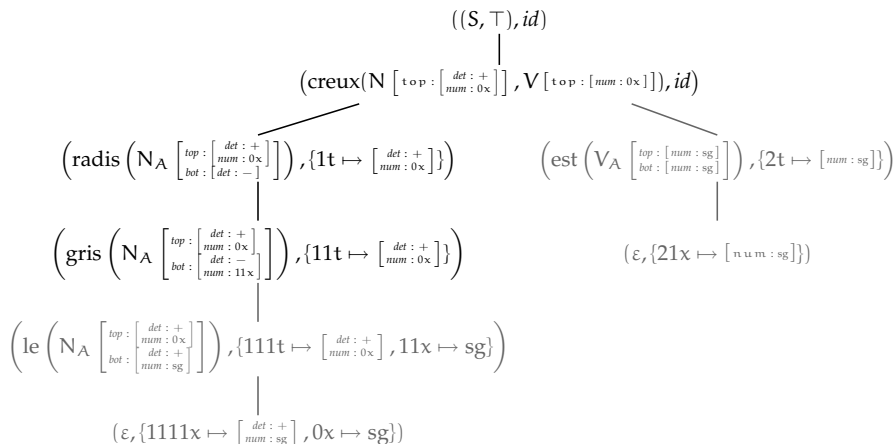
Example



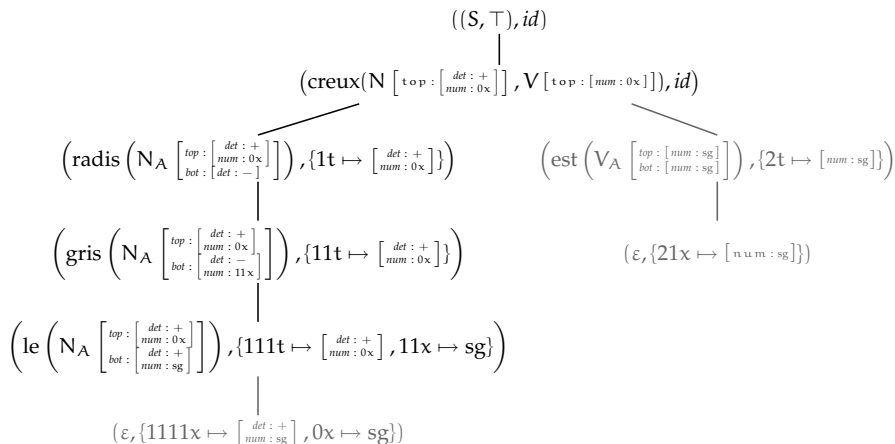
Example



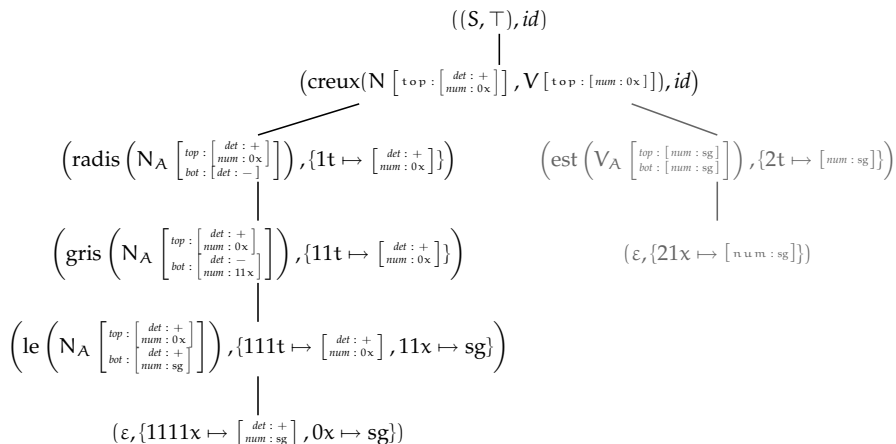
Example



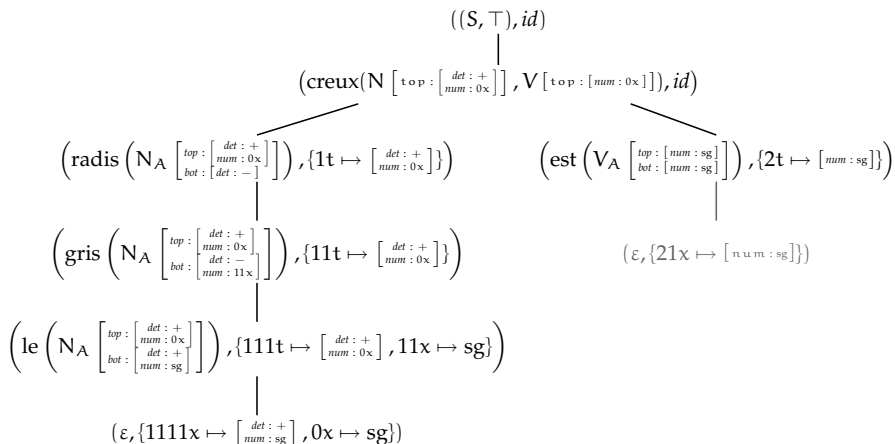
Example



Example



Example



Left Corner Transform

Rosenkrantz and Lewis II [1970]

- ▶ top feature structures of root nodes usually empty
- ▶ reverse order of root adjunctions on initial trees

Overgeneration

Definition

Equivalently, if a grammar

- ▶ assigns incorrect structure to grammatical sentences
- ▶ accepts agrammatical sentences
- ▶ generates agrammatical sentences

precision score in “precision and recall” metrics.

Error Mining

van Noord [2004], Sagot and Éric de la Clergerie [2006]

Applied to grammatical coverage (“recall”)

1. parse a large corpus of correct sentences
2. failures indicate coverage issues
3. statistical analysis identifies a probable culprit for each failure
4. might attempt to provide corrections

Error Mining

van Noord [2004], Sagot and Éric de la Clergerie [2006]

Applied to grammatical coverage (“recall”)

1. parse a large corpus of correct sentences
2. failures indicate coverage issues
3. statistical analysis identifies a probable culprit for each failure
4. might attempt to provide corrections

For Overgeneration?

Which test suite for pass/failure?

- ▶ a TreeBank
- ▶ a corpus of incorrect sentences
- ▶ sentences generated from the grammar
 - ▶ which input?
 - ▶ generation from logic formulae is computationally expensive

For Overgeneration?

Which test suite for pass/failure?

- ▶ a TreeBank
- ▶ a corpus of incorrect sentences
- ▶ sentences generated from the grammar
 - ▶ which input?
 - ▶ generation from logic formulae is computationally expensive

For Overgeneration?

Which test suite for pass/failure?

- ▶ a TreeBank
- ▶ a corpus of incorrect sentences
- ▶ sentences generated from the grammar
 - ▶ which input?
 - ▶ generation from logic formulae is computationally expensive

For Overgeneration?

Which test suite for pass/failure?

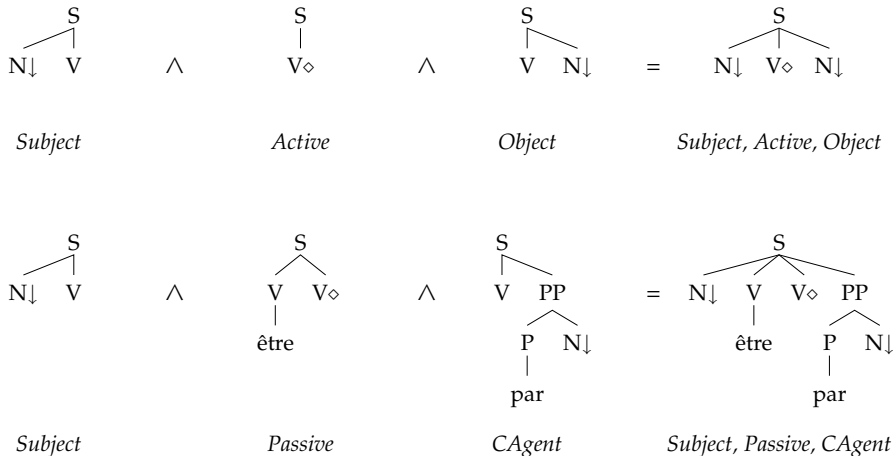
- ▶ a TreeBank
- ▶ a corpus of incorrect sentences
- ▶ sentences generated from the grammar
 - ▶ which input?
 - ▶ generation from logic formulae is computationally expensive

“Exhaustive” Generation

- ▶ not in terms of elementary trees (about 6,000)
- ▶ in terms of *linguistic phenomena*
 - ▶ grammar compiled from a meta grammar
 - ▶ compilation traces
 - ▶ 87 classes match linguistic phenomena

Meta Grammar

[Crabbé, 2005]



Guided Generation

- ▶ use distances from elementary trees to classes
- ▶ generate derivation trees that subsume a target multiset of classes
- ▶ avoid the already generated classes
- ▶ ... in progress

Guided Generation

- ▶ use distances from elementary trees to classes
- ▶ generate derivation trees that subsume a target multiset of classes
- ▶ avoid the already generated classes
- ▶ ... in progress

Concluding Remarks

- ▶ derivation trees capture syntactic dependencies
- ▶ appropriate intermediate level between syntax and semantics
- ▶ not just for TAG

References

- W. S. Brainerd. Tree generating regular systems. *Information and Control*, 14 (2):217–231, 1969. doi: 10.1016/S0019-9958(69)90065-5.
- J. Carroll, A. Copestake, D. Flickinger, and V. Poznanski. An efficient chart generator for (semi-)lexicalist grammars. In *ENLG'99*, pages 86–95, 1999. URL <http://www.cogs.susx.ac.uk/lab/nlp/carroll/papers/ewnl99.pdf>.
- B. Crabbé. Grammatical development with XMG. In P. Blache, E. Stabler, J. Busquets, and R. Moot, editors, *LACL'05*, volume 3492 of *Lecture Notes in Computer Science*, pages 84–100. Springer, 2005. ISBN 978-3-540-25783-7. doi: 10.1007/11422532_6.
- J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. Syst. Sci.*, 31 (1):71–146, 1985. doi: 10.1016/0022-0000(85)90066-2.
- C. Gardent and L. Kallmeyer. Semantic construction in feature-based TAG. In *EACL'03*, pages 123–130. ACL Press, 2003. ISBN 1-333-56789-0. doi: 10.3115/1067807.1067825.
- C. Gardent and E. Kow. A symbolic approach to near-deterministic surface realisation using Tree Adjoining Grammar. In *ACL'07*, pages 328–335. ACL Press, 2007. URL <http://www.aclweb.org/anthology/P07-1042>.

- A. K. Joshi and Y. Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3: Beyond Words, chapter 2, pages 69–124. Springer, 1997. ISBN 3-540-60649-1. URL <http://citeseer.ist.psu.edu/joshi97treeadjoining.html>.
- A. K. Joshi, L. S. Levy, and M. Takahashi. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136–163, 1975. doi: 10.1016/S0022-0000(75)80019-5.
- S. Kepster and J. Rogers. The equivalence of tree adjoining grammars and monadic linear context-free tree grammars. In M. Kracht, editor, *MoL 10*, 2007. URL <http://kracht.humnet.ucla.edu/marcus/mol10/abstracts/kepser-rogers-mol.pdf>.
- A. Koller and M. Stone. Sentence generation as a planning problem. In *ACL'07*, pages 336–343. ACL Press, 2007. URL <http://www.aclweb.org/anthology/P07-1043>.
- A. Koller and K. Striegnitz. Generation as dependency parsing. In *ACL'02*, pages 17–24. ACL Press, 2002. doi: 10.3115/1073083.1073088.
- S. Pogodalla. Computing semantic representation: Towards ACG abstract terms as derivation trees. In O. Rambow and M. Stone, editors, *TAG+7*, pages 64–71, 2004. URL <http://www.cs.rutgers.edu/TAG+7/papers/pogodalla.pdf>.

- D. J. Rosenkrantz and P. M. Lewis II. Deterministic left corner parsing. In *11th Annual Symposium on Switching and Automata Theory*, pages 139–152. IEEE Computer Society, 1970.
- B. Sagot and Éric de la Clergerie. Error mining in parsing results. In *ACL'06*, pages 329–336. ACL Press, 2006. doi: 10.3115/1220175.1220217. URL <http://www.aclweb.org/anthology/P06-1042>.
- S. Schmitz and J. Le Roux. Feature unification in TAG derivation trees. In C. Gardent and A. Sarkar, editors, *TAG+9*, 2008. URL <http://www.loria.fr/~schmitsy/pub/tagunif.en.pdf>.
- S. M. Shieber. Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms. In *EACL'06*. ACL Press, 2006. ISBN 1-932432-59-0. URL <http://www.aclweb.org/anthology/E06-1048>.
- G. van Noord. Error mining for wide-coverage grammar engineering. In *ACL'04*, pages 446–453. ACL Press, 2004. doi: 10.3115/1218955.1219012. URL <http://www.aclweb.org/anthology/P04-1057>.
- K. Vijay-Shanker and A. K. Joshi. Feature structures based tree adjoining grammars. In *COLING'88*, pages 714–719. ACL Press, 1988. ISBN 963-8431-56-3. doi: 10.3115/991719.991783.