Surface Realisation using Tree Adjoining Grammar. Application to Computer Aided Language Learning

Claire Gardent
CNRS / LORIA, Nancy, France
(Joint work with Eric Kow and Laura Perez-Beltrachini)

March 24, 2014
TAG-Based Surface realisation (SR) ...

- maps data to text
- using a grammar which relates NL expressions, syntactic structures and meaning representations

\[ \text{John loves Mary} \]

<table>
<thead>
<tr>
<th>Parsing</th>
<th>Grammar</th>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lexicon</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Algorithm</td>
<td></td>
</tr>
</tbody>
</table>

\[ l1:john(j), l2:mary(m), l3:love(e,j,m) \]
Outline

1. Grammars: TAG, FB-LTAG and Implementation

2. Algorithms for Surface Realisation

3. Application to Computer Aided Language Learning
Tree Adjoining Grammar

A Tree Adjoining Grammar (TAG) is a tuple $G = \langle N, T, I, A, S \rangle$ such that

- $T$ and $N$ are terminals and nonterminals categories,
- $I$ is a finite set of initial trees, and
- $A$ is a finite set of auxiliary trees,
- $S$ is a distinguished non terminal (the axiom)

The trees in $I \cup A$ are called **elementary** trees.

The trees of $G$ are combined using **adjunction** and **substitution**. Each derivation yields two structures: a derived and a derivation tree.
Example Elementary Trees

- **Initial trees** are elementary trees whose leaves are labelled with non terminal or terminal categories. Leaf nodes labelled with non terminal are substitution nodes marked with ↓

- **Auxiliary trees** are elementary trees with a designated *foot node*. The root and the foot nodes are labelled with the same category.

![Tree Diagram]

```
NP  ↘️ S
   ↘️ VP
      ↘️ VP*    ADV
      ↘️ runs

NP  ↘️ VP  ↘️ ADV
      ↘️ quickly
```
Example TAG Derivation

- **Substitution** inserts a derived or elementary tree at the substitution node of a TAG tree.
- **Adjunction** inserts an auxiliary tree into a tree (Adjunction is not allowed on substitution nodes)
Feature-Based TAG

- Tree nodes are decorated with two feature structures called top and bottom.
- Unifications on these feature structures are performed:
  - During derivation, each time a substitution or an adjunction takes place.
  - After derivation: at the end of the derivation, the top and bot FS of each node are unified.
Using Features to capture Subject/Verb Agreement
Unification based Semantic construction in FTAG

- Semantic representation language: unification-based flat underspecified formulae (aka MRS)
- Each elementary tree is associated with a formula $\phi$ representing its meaning
  Missing semantic arguments are represented by unification variables
- Elementary tree nodes are decorated with semantic indices occurring in $\phi$
- The meaning of a derived tree is the union of the meanings associated with the elementary trees modulo the unifications made during processing
Capturing the Interplay between Syntax and Semantics

$S \rightarrow \text{NP}^X \rightarrow \text{VP} \rightarrow \text{NP}^Y \rightarrow \text{NP}_m$

John loves Mary

$l_j : \text{name}(j, \text{john})$, $l_i : \text{love}(X, Y)$, $l_m : \text{name}(m, \text{mary})$

$l_i : \text{love}(j, m), l_j : \text{name}(j, \text{john}), l_m : \text{name}(m, \text{mary})$
Derived and Derivation Trees

(a) Derived tree

S
  NP
    Det  NP
        id\(x=t\)  num=sg
        gen=f
        the  tatoo
    VP
        id\(x=e\)
        tse=pst
        speaks
    ADV
        loudly

(b) Derivation tree

\(\alpha_{\text{tatoo}}\)
\(\beta_{\text{loudly}}\)
\(1\)
\(2\)
\(0\)
Implementing a TAG

A large coverage TAG consists of several thousands of trees.

For each word type, there are as many trees as there are different possible syntactic contexts for that word.

But these trees often share subtrees.

To implement a FB-LTAG, we use the XMG specification language and compiler.

As a result, each TAG tree is associated with the set of XMG classes used to produce it.
Structure Sharing in TAG

(Canonical Subject)

(Active Verb)

Relative Subject, Active Verb
(e.g. the boy who sleeps)
SemFRAG, a grammar for French

- An FB-LTAG for French with unification-based compositional semantics
- Roughly 6 000 elementary trees
- Coverage:
  - 35 basic verbal subcategorisation frames
  - Argument redistributions: active, passive, middle, neuter, reflexivisation, impersonal, passive impersonal
  - Argument realisations: cliticisation, extraction, omission, permutations, etc.
  - Predicative (adjectival, nominal and prepositional) and light verb constructions
  - Basic descriptions for adverbs, determiners and prepositions.
XXTAG, a grammar for English

- An FB-LTAG for English
- Roughly 1200 elementary trees
- Reimplementation of the XTAG grammar developed at UPenn
- Coverage: most of the Penn Treebank
Complexity

Surface realisation from flat semantics is exponential in the length of the input (Brew92,Kay96)

- **Unordered Input**: At least $2^n$ possible combinations with $n$ the number of literals in the input
- **intersective modifiers**: $2^{m+1}$ possible intermediate structures with $m$ the number of modifiers for a given structure
- **lexical ambiguity**: $\prod_{i=1}^{i=n} Lex_i$ possible combinations with $Lex_i$ the number of lexical entries associated with literal $l_i$ and $n$ the number of literals in the input
SR Algorithm 1: GenI

3 steps:

1. **Lexical selection**: select trees whose semantics subsumes the input semantics.

2. **Combination**: perform substitutions or adjunctions between selected trees.

3. **Extraction**: Return the trees which are syntactically complete and whose semantics matches the input semantics.

Claire Gardent and Eric Kow.
A Symbolic Approach to Near-Deterministic Surface Realisation using Tree Adjoining Grammar.
ACL 2007, Prague
Example

\[ S \]

\[ NP_X \]
\[ NP_j \]
John

\[ NP_Y \]
\[ NP_m \]
Mary

\[ V \]
loves

\[ l_j : name(j, john) \]
\[ l_i : love(X, Y) \]
\[ l_m : name(m, mary) \]

\[ l_i : love(j, m), l_j : name(j, john), l_m : name(m, mary) \]
Optimisations

- Semantic indices used to limit the impact of unordered input
- Substitutions before Adjunctions to deal with intersective modifiers
- Polarity based filtering to reduce the initial search space and limit the impact of lexical ambiguity
Intersective modifiers

\texttt{fierce(x), little(x), cat(x), black(x)}

15 intermediate structures:
\texttt{F,L,B,FL,FB,BL,BF,LF,FLB,FBL,BLF,BFL,LBF,LFB}
multiplied by the context :
\texttt{x 2: the F,L,B,FL,FB,BL,BF,LF,FLB,FBL,BLF,BFL,LBF,LFB}
\texttt{x 2: the F,L,B,FL,FB,BL,BF,LF,FLB,FBL,BLF,BFL,LBF,LFB runs}

45 structures built
Substitutions before Adjunctions

Adjunction restricted to syntactically complete trees
The $2^{m+1}$ intermediate structures are not multiplied out by the context:

the cat runs

the fierce cat runs, the black cat runs, the little cat runs, the fierce little cat runs, the fierce black cat runs, the black fierce cat runs, ...

16 structures built
Polarity based filtering (Perrier 2003)

Polarity based filtering filters out all combinations of lexical items which cannot result in a grammatical structure

- The grammar trees are associated with polarities reflecting their syntactic resources and requirements
- A combination of trees covering the input semantics but whose polarity is not zero is necessarily syntactically invalid and is therefore filtered out.
- A finite state automata is built which represent the possible choices (transitions) and the cumulative polarity (states)
- The paths leading to a state with polarity other than zero are deleted (automata minimisation)
Polarity Filtering

Many combinations are syntactically incompatible. Polarity filtering aims to detect these combinations and to filter them out.

<table>
<thead>
<tr>
<th>john(j)</th>
<th>drink(e,j,w)</th>
<th>water(w)</th>
<th>Polarity Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1np</td>
<td>SFIN-2np</td>
<td>+1np</td>
<td>+0np</td>
</tr>
<tr>
<td>SFIN-1np</td>
<td></td>
<td>+1np</td>
<td>+1np</td>
</tr>
</tbody>
</table>

(mode: fin)

S

NP

X

V

drinks

VP

(mode: inf, controller: X)

S

NP

X

V

drinks

VP

NP

Y
SR Algorithm 2: RTGen

- Builds derivation rather than derived trees ...
- using a conversion from FB-LTAG to Feature Based Regular Tree Grammar (FB-RTG, Schmitz and Leroux 2009)
- Earley algorithm with packing and sharing

Derived and Derivation Tree

(a) Derived tree

(b) Derivation tree
Converting a TAG to an RTG

\[
\begin{align*}
S & \rightarrow NP_S \\
NP_S & \rightarrow john(NP_A) \\
NP & \rightarrow S_S \\
S_S & \rightarrow runs(S_A NP_S VP_A V_A) \\
V & \rightarrow VP_T \\
VP_T & \rightarrow often(VP_A) \\
NP & \rightarrow VP_A \\
VP_A & \rightarrow \epsilon \\
V_A & \rightarrow \epsilon \\
VP & \rightarrow \epsilon
\end{align*}
\]
Earley Algorithm

Axiom

\[ S' \rightarrow \bullet S, \emptyset \]

Goal

\[ S' \rightarrow S \bullet, \phi \] where \( \phi \) is the input semantics.

Prediction

\[
\frac{[A \rightarrow a(\alpha \bullet B_x \beta), \varphi]}{[\sigma(B^0 \rightarrow b(\bullet B^1, ..., B^n), \psi)]}
\]

with \( \langle B \rightarrow b(B^1, ..., B^n), \psi \rangle \) a grammar rule

\( \sigma = \text{mgu}(B, B^0), P[x] \in \psi \) and \( \varphi \cap \psi = \emptyset \)

Completion

\[
\frac{[A \rightarrow a(\alpha \bullet B \delta), \varphi][B \rightarrow b(\beta) \bullet, \psi]}{[\sigma(A \rightarrow a(\alpha(B, f) \bullet \delta), \varphi \cup \psi)]}
\]

with \( \sigma = \text{mgu}(B, B^0), \varphi \cap \psi = \emptyset \)
RTGen vs GenI

- All trees are taken into account while filtering (GenI’s polarity filtering ignores auxiliary trees)
- All features can be used (GenI’s polarity filtering can only use ground features i.e., categories)
- All syntactic constraints are applied (not just counting)
- Intersective Modifiers are handled using packing and sharing
Comparison on two Automatically Generated Benchmarks

- Modifiers benchmark: modification differently distributed over the predicate argument structures + lexical ambiguity in modifiers. 1789 input formulae.
- All benchmark: modification, varying number and type of verb arguments. 890 input formulae.

Modification

- when using no features (\textsc{RTGEN-level0}) over-generation increases the number of intermediate structures.
- when using all the features (\textsc{RTGEN-all}) or only a selected set of them (\textsc{RTGEN-selective}) (almost) the same number of intermediate structures are produced.

Space performance results on the \textsc{Modifiers}-benchmark.
Overall efficiency

For more complex cases, RTGen’s sharing and packing mechanisms perform better.
Generating Grammar Exercises

Generate sentences

Use the detailed linguistic information output by the generator to select and build exercises

Three types of exercises: FIB, Shuffle and Reformulation

C. Gardent and L. Perez-Beltrachini.
Using FB-LTAG Derivation Trees to Generate Transformation-Based Grammar Exercises.

L. Perez-Beltrachini, C. Gardent and G. Kruszewski
Generating Grammar Exercises.
The 7th Workshop on Innovative Use of NLP for Building Educational Applications, NAACL-HLT Worskhop 2012, Montreal, Canada, June.
Grammar Exercises

Built from a single sentence.

[FIB] Complete with an appropriate personal pronoun.

(S) Elle adore les petits tatous
(She loves the small armadillos)

(Q) _____ adore les petits tatous  (gender=fem)
(K) elle

[Shuffle] Use the words below to make up a sentence.

(S) Tammy adore les petits tatous
(Tammy loves the small armadillos)

(Q) tatous / les / Tammy / petits / adore
(K) Tammy adore les petits tatous.
Grammar Exercises

Built from a pair of syntactically related sentences

[Reformulation] Rewrite the sentence using passive voice

(Q) C’est Tex qui a fait la tarte.
   (It is Tex who has baked the pie.)

(K) C’est par Tex que la tarte a été faite.
   (It is Tex by whom the pie has been baked.)

Active/Passive, NP/Pronoun, Assertion/Wh-Question,
Assertion/YN-Question
The *GramEx* framework: generating and selecting sentences to build exercises
Creating a grammar exercise

\[
aime(e, be, bi), bijou(bi), les(bi), betty(be)
\]
Creating a grammar exercise

\textit{aime(e,be,bi), bijou(bi), les(bi), betty(be)}

\textit{Bette aime le bijou.}
\textit{C’est Bette qui aime les bijoux.}
\textit{Bette aime les bijoux.}
Creating a grammar exercise

aime(e, be, bi), bijou(bi), les(bi), betty(be)

Goal: Plural form of irregular nouns.  
Exercise type: Fill-in-the-blank.

Bette aime le bijou.
C’est Bette qui aime les bijoux.
Bette aime les bijoux.
Creating a grammar exercise

aime(e, be, bi), bijou(bi), les(bi), betty(be)

Bette aime le bijou.
C’est Bette qui aime les bijoux.
Bette aime les bijoux.

Goal: Plural form of irregular nouns.
Exercise type: Fill-in-the-blank.

1. Select sentences
   NP[\(num = pl \wedge plural = irreg\)]
   \(\wedge\) CanonicalOrder
Creating a grammar exercise

\[ \text{aime}(e, be, bi), \text{bijou}(bi), \text{les}(bi), \text{betty}(be) \]

Bette aime le bijou.

C'est Bette qui aime les bijoux.

Bette aime les bijoux.

Goal: Plural form of irregular nouns.
Exercise type: Fill-in-the-blank.

\[
\downarrow
\]

1. Select sentences
   \[ \text{NP}[num = pl \land \text{plural} = \text{irreg}] \land \text{CanonicalOrder} \]
Creating a grammar exercise

**aime(e, be, bi), bijou(bi), les(bi), betty(be)**

*Bette aime le bijou.*

*C’est Bette qui aime les bijoux.*

Bette aime les bijoux.

**Goal:** Plural form of irregular nouns.

**Exercise type:** Fill-in-the-blank.

1. Select sentences

   \[NP[num=pl \land plural=irreg] \land CanonicalOrder\]

2. Process the selected sentence

   \[NP[num=pl] \Rightarrow \text{blank}\]
   \[NP[lemma=bijou][num=pl] \Rightarrow \text{hint}\]
Creating a grammar exercise

\[ \text{aime(e,be,bi), bijou(bi), les(bi), betty(be)} \]

Bette aime le bijou.
C'est Bette qui aime les bijoux.
Bette aime les bijoux.

Goal: Plural form of irregular nouns.
Exercise type: Fill-in-the-blank.

1. Select sentences
   \[ \text{NP[num = pl} \land \text{plural = irreg]} \land \text{CanonicalOrder} \]

2. Process the selected sentence
   \[ \text{NP[num = pl]} \Rightarrow \text{blank} \]
   \[ \text{NP[lemma = bijou]} \Rightarrow \text{hint} \]
   (Q) Bette aime les _______. (bijou)
   (K) bijoux

\{CanonicalObject, CanonicalSubject, ActiveVerb\}
Selecting appropriate sentences

*GramEx*'s boolean constraint language over linguistic constants

**morpho-syntactic** (feature structures)

**syntactic** (tree properties)
Selecting appropriate sentences

*GramEx*'s boolean constraint language: syntax and use

Boolean constraint language

- conjunction, disjunction and negation of
  morpho-syntactic and syntactic properties

Describes the linguistic requirements imposed by pedagogical goals

- Permits retrieving appropriate sentences from the DB
Selecting appropriate sentences

Some examples

**Pedagogical goal:** Pre/post nominal irregular adjectives

\[ \text{Epith} \land \text{flexion: irreg} \]

- ✓ Tammy a une voix douce (Tammy has a soft voice)
- X Tammy a une jolie voix (Tammy has a nice voice)

**Pedagogical goal:** Prepositions with infinitives

\[ \text{POBJinf} \land \text{CLAUSE} \]

\[ \text{POBJinf} \Leftrightarrow (\text{DE-OBJinf} \lor \text{A-OBJinf}) \]

\[ \text{CLAUSE} \Leftrightarrow \text{Vfin} \land \neg \text{Mod} \land \neg \text{CCoord} \land \neg \text{Sub} \]

- ✓ Tammy refuse de chanter (Tammy refuses to sing)
- X Jean dit que Tammy refuse de chanter (John says that Tammy refuses to sing)
Transformation-based grammar exercises
Finding syntactically related sentences (e.g. active/passive)

(Q) C’est Tex qui a fait la tarte.
   (It is Tex who baked the pie.)

✗ (K) Tex a fait la tarte.
    (Tex baked the pie.)

✗ (K) La tarte a été faite par Tex.
    (The pie was baked by Tex.)

✗ (K) C’est par Tex que la tarte sera faite.
    (It is Tex who will bake the pie.)

✗ (K) Est-ce que la tarte a été faite par Tex ?
    (Has the pie been baked by Tex ?)

✓ (K) C’est par Tex que la tarte a été faite.
    (It is Tex by whom the pie was baked.)
Creating transformation-based grammar exercises

To identify pairs of sentences that are identical up to a single syntactic transformation, we ...

- Use the information contained in derivation trees
- Define tree filters on pairs of derivation trees
- Retrieve sentences pairs that match those tree filters
Why Derivation Trees?

- Detailed syntactic information
- More compact than derived trees. Allow fewer and simpler filters.
Derivation vs Derived trees

\[
\begin{align*}
\alpha\text{-faire:} & \{\text{Active, CleftSubj, CanObj}\} \\
\alpha\text{-tex:} & \{\cdots\} \\
\alpha\text{-avoir:} & \{\cdots\} \\
\alpha\text{-tarte:} & \{\cdots\} \\
\beta\text{-la:} & \{\cdots\}
\end{align*}
\]
Derivation Tree Filters

Tree filter types

- **Active/passive**
  - $s\{\text{Active, CleftSubj, CanObj}\} \leftrightarrow t\{\text{Passive, CleftAgent, CanSubj}\}$

- **NP/Pronoun**
  - $s\{\text{CanSubj}\} \leftrightarrow t\{\text{CliticSubj}\}$

- **Assertion/YN-Question**
  - $\emptyset \leftrightarrow q\{\text{questionMark}\}$
Meaning Preserving Transformations
Same core meaning (e.g. active/passive)

(Q) C’est Tex qui a fait la tarte. ↔ (K) C’est par Tex que la tarte a été faite.
(It is Tex who has baked the pie) (It is by Tex that the pie has been baked)
↔ (K) La tarte a été faite par Tex.
(The pie has been baked by Tex)

\[\alpha\text{-faire:}\{\text{Active, CleftSubj, CanObj}\}\]
\[\alpha\text{-tex:}\{\ldots\}\quad \beta\text{-avoir:}\{\ldots\}\quad \alpha\text{-tarte:}\{\ldots\}\]
\[\beta\text{-la:}\{\ldots\}\]

\[\alpha\text{-faire:}\{\text{Passive, CleftAgent, CanSubj}\}\]
\[\alpha\text{-tex:}\{\ldots\}\quad \beta\text{-avoir:}\{\ldots\}\quad \alpha\text{-tarte:}\{\ldots\}\]
\[\beta\text{-la:}\{\ldots\}\]

\[\bullet_s\{\text{Active, CleftSubj, CanObj}\}\]
↔ \[\bullet_t\{\text{Passive, CleftAgent, CanSubj}\}\]
Meaning Altering Transformations

Related core meaning: content deleted, added or replaced (e.g. Assertion/Wh-Question)

α-dort: { CanSubj }
    | α-tatou: { ... }
    | β-chante: { ... }
    | β-petit: { ... }
    | β-le: {defDet}

α-dort: { whSubj }
    | α-tatou: { ... }
    | β-petit: { ... }
    | β-quel: { WhDet }

Quel petit tatou dort?
Which small armadillo sleeps?

Le petit tatou qui chantera dort.
The small armadillo that will sing sleeps

β-qui: { WhPron }
Qui dort?
Who sleeps?

β-quel: { WhDet }
Quel tatou dort?
Which armadillo sleeps?
Correctness, Productivity, Integration

Manual annotation of a sample of generated exercises

- using SemFraG and lexicon tailored to *Tex’s French Grammar* vocabulary
- around 80% of the automatically generated exercises are correct
- 52 input formulae $\Rightarrow$ around 5000 exercises

Exercises generated by *GramEx* are integrated in I-FLEG (serious game) and WFLEG (web interface)
WFLEG

Welcome!

ALLEGRO is an EU funded INTERREG IV A project which focuses on the development of new technologies for second language learning. Our aim is to exploit research technologies from Natural Language Generation to automatically generate grammar exercises and Learner/Computer Dialog Systems which enable self practice. While the learner has full autonomy to decide on the exercises to be practiced, the system keeps tracks of the learner’s activities and results. This in turn opens the door for adaptive training systems i.e., systems which promote learning by suggesting new activities based on the learner’s history.

To showcase the power of our technology, we embedded our exercise generator tool both in WFLEG (this web service) and in the IFLEG (Interactive French Learning Game) serious game.

Please select the exercise you want to play with during this session:

**W-FLEG Vocabulary exercises**

W-FLEG includes exercises designed to help learning French vocabulary. The learner is shown an image depicting an object and prompted for its name. All interactions are logged in a database thereby supporting a detailed analysis of the learner’s activities. In the future, we plan to use this data to develop adaptive learning systems which make use of a learner’s history to assist the learner in choosing activities likely to enhance his/her progress. The database recording W-FLEG activities (vocabulary and grammar) is common to the IFLEG serious game so that a learner’s activities in both IFLEG and W-FLEG can equally be taken into account to analyse his/her progress.

Anyone can play with W-FLEG! Register W-FLEG & OpenSIM FLEG here

**I-FLEG Grammar exercises**

I-FLEG proposes grammar exercises which were automatically generated using Natural Language Generation techniques. The WFLEG grammar exercises can be practiced using the IFLEG serious games where the learner practice by walking through a house, clicking on objects and selecting a training activity related to that object.

**Tex and Tammy exercises**

These exercises follow the curriculum proposd in the Tex and Tammy French Grammar course which is arranged like many other traditional reference grammars with the parts of speech (nouns, verbs, etc.) used to categorise specific grammar items (gender of nouns, irregular verbs).

NOTE:

“The original Tex & Tammy is about the epic love story of Tex and Tammy, two star-crossed souls, and Bolo, the story takes place in traditional France. In addition to this narrative is a game, the cast of characters include Edouard, a mysterious French anial, Joe-Bob, a dim-witted camel from College Station, and Cosley, a cockroach who prides getting high and watching the X-Files on TV to doing his French homework.”

More can be found here

Register here to play with our Tex grammar based exercises
Chapter 1: Bonjour!

Fill in the blank - missing word: Subject pronouns

Fill in the blank with the appropriate subject pronoun. Remplir le trou avec le pronom personnel approprié.

......... adore l'odeur des pesticides

Type your answer here

Time and score

- Question time: 00:00:21
- Exercise time: 00:00:21
- Session time: 00:00:21
- Current Exercise score: 0
- Exercise score in previous session: 3
- Session score: 0
- Session score in previous session: 9
- Best Exercise score: 47 [FLEG Test]
- Best score: 47 [FLEG Test]
WFLEG

Tag, FB-LTAG and Implemented Grammars Surface Realisation

Tex and Tammy grammar exercises

Amount of tests done: 616
Average score: 43.67%
Average time: 00:01:39

Tests results

Progress ratio

Tests results

Subject pronouns
Singular noun / gender / case
Determiner: definite articles
Determiner: indefinite articles
Determiner: possessive determiner

Type of exercises done

Last Tests results

Amount of tests done
Score of the last tests done
Response time of the last tests done

Date of the tests
Amount of tests done
Score
Time

2013-09-18
2013-08-19
2013-07-29
2013-06-05
2013-05-22
2013-04-30
2013-03-14
2013-02-09
2013-01-13
2012-12-03
2012-11-19
2012-09-25
2012-08-08
2012-07-23
2012-06-18
2012-05-12
2012-04-19
2012-03-31
2012-02-24
2012-01-20
2011-12-06
2011-11-10
2011-10-06
2011-09-13
2011-08-21
2011-07-19
2011-06-14
2011-05-11
2011-04-18
2011-03-24
2011-02-21
2011-01-17
2010-12-13
2010-11-09
2010-10-05
2010-09-28
2010-08-31
2010-08-17
2010-07-19
2010-06-24
2010-06-02
2010-05-03
2010-04-16
2010-03-19
2010-02-21
2010-01-18
2009-12-28
2009-12-14
2009-11-24
2009-11-05
2009-10-22
2009-10-08
2009-09-24
2009-09-09
2009-08-25
2009-08-15
2009-07-23
2009-07-09
2009-06-25
2009-06-10
2009-05-29
2009-05-15
2009-04-28
2009-04-14
2009-03-26
2009-03-12
2009-02-26
2009-02-12
2009-01-29
2008-12-21
2008-12-07
2008-11-23
2008-11-07
2008-10-21
2008-10-07
2008-09-22
2008-09-08
2008-08-21
2008-08-06
2008-07-22
2008-07-08
2008-06-24
2008-06-09
2008-05-25
2008-05-11
2008-04-24
2008-04-10
2008-03-27
2008-03-13
2008-02-27
2008-02-13
2008-01-29
2007-12-21
2007-12-07
2007-11-23
2007-11-07
2007-10-22
2007-10-08
2007-09-22
2007-09-08
2007-08-21
2007-08-06
2007-07-22
2007-07-08
2007-06-24
2007-06-09
2007-05-25
2007-05-11
2007-04-24
2007-04-10
2007-03-27
2007-03-13
2007-02-27
2007-02-13
2007-01-29
Thanks!