LING 696G: Syntax and Computation

- Lecture 5

- Sandiway Fong
- University of Arizona
- sandiway.arizona.edu
- sandiway@arizona.edu

Announcement

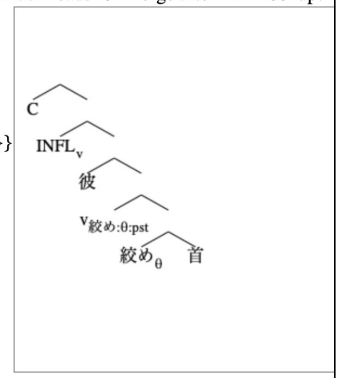
- Next time:
  - SMT Parser: *see next slides*
- Guest lecture today:
  - *50 Years after the Chomsky-Piaget Debate*
  - Massimo Piattelli-Palmarini

SMT Parser: building from scratch

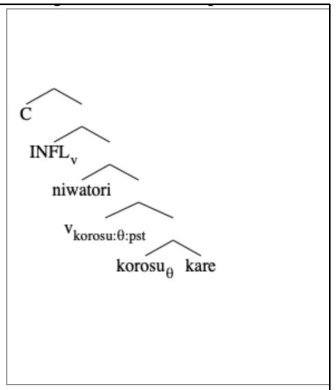
- *Some examples from Jason's talk about Japanese*

▶ Help: blue = parse inside. **Abbr.:** WS: Workspace; Initial WS: initial heads for Merge after LEX lookup.

Words: 彼 は 首 を 絞めた
▼ **Initial WS 1:** 絞め$_\theta$ $v_{絞め:\theta:pst}$ $INFL_v$ 首 彼
▶ **WS 1:** {絞め$_\theta$, 首} $v_{絞め:\theta:pst}$ $INFL_v$ 彼
INT/EXT: ⩘ {C, {$INFL_v$, {彼, {$v_{絞め:\theta:pst}$, {絞め$_\theta$, 首}}}}}
EXT: 彼 は 首 を 絞め pst 3
Spellout: 彼 は 首 を 絞めた
Parse found: 彼 は 首 を 絞め pst 3
EXT: 彼 が 首 を 絞め pst 3
Spellout: 彼 が 首 を 絞めた
Blocked: inconsistent with input!
▶ **WS 1:** {絞め$_\theta$, 首} $v_{絞め:\theta:pst}$ $INFL_v$ 彼
▶ **WS 1:** {絞め$_\theta$, 彼} $v_{絞め:\theta:pst}$ $INFL_v$ 首
INT/EXT: ⩘ {C, {$INFL_v$, {首, {$v_{絞め:\theta:pst}$, {絞め$_\theta$, 彼}}}}}
EXT: 首 は 彼 を 絞め pst 3
Spellout: 首 は 彼 を 絞めた
Blocked: inconsistent with input!
EXT: 首 が 彼 を 絞め pst 3
Spellout: 首 が 彼 を 絞めた
Blocked: inconsistent with input!
▶ **WS 1:** {絞め$_\theta$, 彼} $v_{絞め:\theta:pst}$ $INFL_v$ 首
▶ **WS 1:** {首, 彼} 絞め$_\theta$ $v_{絞め:\theta:pst}$ $INFL_v$
INT/EXT: ⩘ {C, {$INFL_v$, {{首, 彼}, {$v_{絞め:\theta:pst}$, {絞め$_\theta$, {首, 彼}}}}}}



Words: kare ga niwatori o korosita
▼ **Initial WS 1:** korosu$_\theta$ $v_{korosu:\theta:pst}$ $INFL_v$ niwatori kare
▶ **WS 1:** {korosu$_\theta$, niwatori} $v_{korosu:\theta:pst}$ $INFL_v$ kare
INT/EXT: ⩘ {C, {$INFL_v$, {kare, {$v_{korosu:\theta:pst}$, {korosu$_\theta$, niwatori}}}}}
EXT: kare wa niwatori o korosu pst 3
Spellout: kare wa niwatori o korosita
Blocked: inconsistent with input!
EXT: kare ga niwatori o korosu pst 3
Spellout: kare ga niwatori o korosita
Parse found: kare ga niwatori o korosu pst 3
▶ **WS 1:** {korosu$_\theta$, niwatori} $v_{korosu:\theta:pst}$ $INFL_v$ kare
▶ **WS 1:** {korosu$_\theta$, kare} $v_{korosu:\theta:pst}$ $INFL_v$ niwatori
INT/EXT: ⩘ {C, {$INFL_v$, {niwatori, {$v_{korosu:\theta:pst}$, {korosu$_\theta$, kare}}}}}
EXT: niwatori wa kare o korosu pst 3
Spellout: niwatori wa kare o korosita
Blocked: inconsistent with input!
EXT: niwatori ga kare o korosu pst 3
Spellout: niwatori ga kare o korosita
Blocked: inconsistent with input!
▶ **WS 1:** {korosu$_\theta$, kare} $v_{korosu:\theta:pst}$ $INFL_v$ niwatori
▶ **WS 1:** {niwatori, kare} korosu$_\theta$ $v_{korosu:\theta:pst}$ $INFL_v$
INT/EXT: ⩘ {C, {$INFL_v$, {{niwatori, kare}, {$v_{korosu:\theta:pst}$, {korosu$_\theta$, {niwatori, kare}}}}}}



SMT Parser: building from scratch

Building **LEX** from scratch: an example

Sandiway Fong (September 2026) University of Arizona

Preliminaries

You will first need to download and modify the bare files

`lexN.prolog` and
`code_N.prolog`.

Place them in the same directory as the other *696A SMT Parser* files.

The (generic) empty **LEX** is `lexN.prolog`.

You will also need to download the 696A *SMT Parser* (v2), which has some updates to the **EXT** and linear processing components.

```
swipl -s code_N.prolog
```

loads in the *SMT Parser* with the empty **LEX** `lexN.prolog`. `code_N.prolog` is defined as follows:

```
:- ensure_loaded(core).
:- ensure_loaded(lexN).       % generic LEX
:- noMorphy.
```

We recommend you copy and rename these files. For example, for this exercise we can copy them as `lexJ.prolog` and `code_J.prolog`, taking care to change `lexN` into `lexJ` inside code_J.prolog.

vStem/4

Japanese has no visible verbal agreement, so we define `vStem/4` to return `null` for AGR and decompose an inflected verb to produce a root and TNS. It should operate as follows for *simeru* (絞める), taking the root (R) to be *sime* (絞め) for `vR/3` (θ-access) as it is a *ichidan* (一段) verb.

```
?- vStem(simemasu,R,TNS,AGR).
R = sime,
TNS = pres,
AGR = null ;
false.

?- vStem(絞めた,R,TNS,AGR).
R = 絞め,
TNS = pst,
AGR = null ;
false.
```

Similarly, we define `vStem/4` for *korosu* (殺す). In this case we take the root (R) for `vR/3` (θ-properties) access to be *korosu* (殺す) as it is a *godan* (五段) verb.

```
?- vStem(korosita,R,TNS,AGR).
R = korosu,
TNS = pst,
AGR = null ;
false.

?- vStem(殺しました,R,TNS,AGR).
R = 殺す,
TNS = pst,
AGR = null ;
false.
```

SMT Parser: building from scratch

Amalgamation

In the case of English and Spanish, we apply affixL/2 rules of the form:

```
affixL([Phi,TNS,Root|Ws],[Word|Wsp]) :-
      φ(Phi), tns(TNS), vR(Root,_,_),
      spell_inflectedV(Phi,TNS,Root,Word), % deal with irregular cases too
      affixL(Ws,Wsp).
```

The above rule is defined in **EXT** and assumes ϕ + TNS + Root maps into a single Word via a language-specific rule spell_inflectedV/4, and proceeds from left to right, mapping subsequent Ws into Ws' = Wsp.

We add the following rule to linear processing in **EXT**:

```
affixL([Root,TNS,Phi|Ws],[Word|Wsp]) :-
      head(final),
      φ(Phi), tns(TNS), vR(Root,_,_),
      spell_inflectedV(Phi,TNS,Root,Word),
      affixL(Ws,Wsp).
```

Japanese linear order is Root + TNS and verbal morphology is insensitive to ϕ.

spell_inflectedV/4 must be defined in **LEX**:

```
%% φ-features irrelevant for Japanese
spell_inflectedV(_Phi,TNS,Root,Word) :- vStem(Word,Root,TNS,null).
```

Note: language-particular vStem/4 was (partially) defined earlier for Japanese ichidan/godan verbs.

Case particle insertion

Returning to our example:

```
彼   鶏      殺す    pst 3 ⇒ 彼   鶏      殺した   ⇒ 彼が   鶏を      殺した
kare niwatori korosu           kare niwatori korosita    kare-ga niwatori-o korosita
```

At **EXT** Case must be spelled out post-nominally.

```
%% caseParticle(initial)  e.g. of the city
%% :- dynamic caseParticle/1.
caseParticle(final).  % e.g. kubi-o (首を)
```

EXT

EXT applies to each head and phrase (recursively) for a computed **SO**, e.g.:

```
{C, {INFL_v:3, {彼, {v_殺す:θ:pst, {殺す_θ, 鶏}}}}}
```

Both phrase order and output for each head must be defined in order for **EXT** to proceed. (**SO**s submitted to **INT** are intrinsically unordered.)

For example, at **EXT** the functional heads in the above **SO**, *viz.* declarative C, INFL and v produce nothing, a ϕ-morpheme ('3' = 3rd person) and TNS (pst = past), respectively.

ϕ-feature output should also be defined here (although in the case of Japanese, we assume it is not taken up by verbal inflection and thus not visible.)

ϕ/1

As part of narrow syntax, universally **MS** applies to INFL to find the nearest θ-relevant nominal. As a result, INFL gets ϕ-feature 3rd person, see φ/2 above. We define φ/1 for ϕ-feature **EXT** as follows:

```
%% VERB φ-FEATURES
%% φ('1sg').

%% EXT on INFL
:- dynamic φ/1.
φ('3').                          % just person
```

This ensures **EXT** on head INFL will produce the morpheme '3' for *Amalgamation* (see below).

Head-complement order is also imposed at **EXT**. We assume the Japanese **LEX** specifies:

```
head(final).
```

With order imposed, we spell out the heads of the (ordered) bracketed structure:

```
[[[彼, [[鶏, 殺す_θ]], v_殺す:θ:pst], INFL_v:3], C]
```

and obtain the linear sequence:

```
彼 鶏 殺す pst 3
```

Next, we must ensure **LEX** verbal inflection rules apply at *Amalgamation* to map verbal root 殺す and affixes *pst 3* into 殺した. We will then obtain:

SMT Parser: building from scratch

- New stuff to download next time:
  - `lexN.prolog` ("blank" **LEX**)
  - `code_N.prolog` (`swipl –s code_N.prolog`)
  - `code.qlf` (Modified **EXT** and **Amalgamation**)

(*Japanese language starter*)

- `lexJ.prolog / code_J.prolog`

```
%% test for head(final) in EXT
%% head(initial).
:- dynamic head/1.
head(final).

%% caseParticle(initial).  e.g. of the city
%% :- dynamic caseParticle/1.
caseParticle(final).            % e.g. kubi-o (首を)

%% option to not pronounce left edge of INFL
% :- dynamic proDrop/0.
proDrop.
```