Unification-Based Grammar Engineering

Dan Flickinger
Stanford University & Redbird Advanced Learning
danf@stanford.edu

Stephan Oepen
Oslo University
oe@ifi.uio.no

ESSLLI 2016; August 15–19, 2016

Dative Shift: A Productive Process

$\{hand_1, give_1, send_1, ...\}$

$$\begin{bmatrix} \texttt{HEAD} & verb \\ \texttt{SPR} & \langle \cdots \rangle \\ \texttt{COMPS} & \left\langle \begin{bmatrix} \texttt{HEAD} & noun \\ \texttt{SPR} & \langle\rangle \\ \texttt{COMPS} & \langle\rangle \end{bmatrix}_{phrase}, \begin{bmatrix} \texttt{HEAD} & noun \\ \texttt{SPR} & \langle\rangle \\ \texttt{COMPS} & \langle\rangle \end{bmatrix}_{phrase} \right\rangle \end{bmatrix}$$

$\{hand_2, give_2, send_2, ...\}$

$$\begin{bmatrix} \texttt{HEAD} & verb \\ \texttt{SPR} & \langle \cdots \rangle \\ \texttt{COMPS} & \left\langle \begin{bmatrix} \texttt{HEAD} & noun \\ \texttt{SPR} & \langle\rangle \\ \texttt{COMPS} & \langle\rangle \end{bmatrix}_{phrase}, \begin{bmatrix} \texttt{HEAD} & prep\begin{bmatrix}\texttt{PFORM}\ to\end{bmatrix} \\ \texttt{SPR} & \langle\rangle \\ \texttt{COMPS} & \langle\rangle \end{bmatrix}_{phrase} \right\rangle \end{bmatrix}$$

Lexical Variation: Lexical Rules

- Dative shift, passivization, et al. are systematic processes in the lexicon;
- use of *monotonic* inheritance is insufficient to relate $give_1$ and $give_2$;
- *lexical rules* are unary grammar rules that operate 'within the lexicon';
- take as input a lexical sign (*expression*) and output a derived lexical sign.

Rough Approximation of Passive Lexical Rule

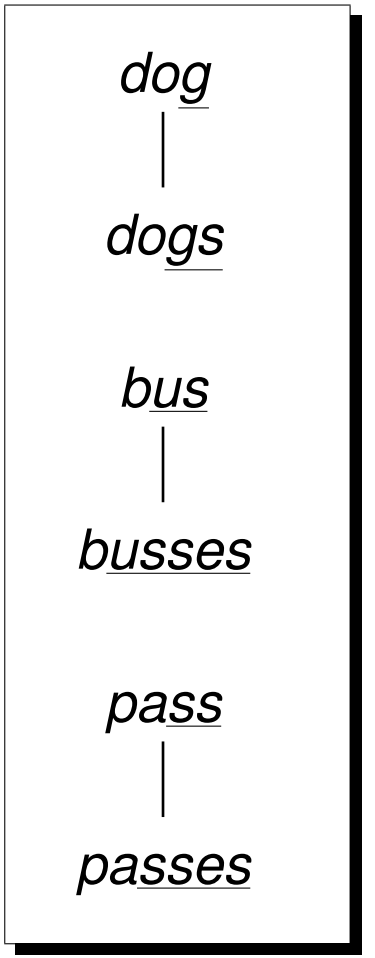
$$\left[\begin{array}{ll} \texttt{HEAD} & \boxed{1} \\ \texttt{SPR} & \langle \boxed{2} \rangle \\ \texttt{COMPS} & \boxed{3} \\ \texttt{ARGS} & \left\langle \left[\begin{array}{ll} \texttt{HEAD} & \boxed{1} \\ \texttt{SPR} & \left\langle \textit{phrase}\left[\texttt{HEAD}\ \textit{noun}\right] \right\rangle \\ \texttt{COMPS} & \left[\begin{array}{ll} \texttt{FIRST} & \boxed{2}\ \textit{phrase}\left[\texttt{HEAD}\ \textit{noun}\right] \\ \texttt{REST} & \boxed{3} \end{array}\right] \end{array}\right] \right\rangle \end{array}\right]$$

Orthographemic Variation: Inflectional Rules

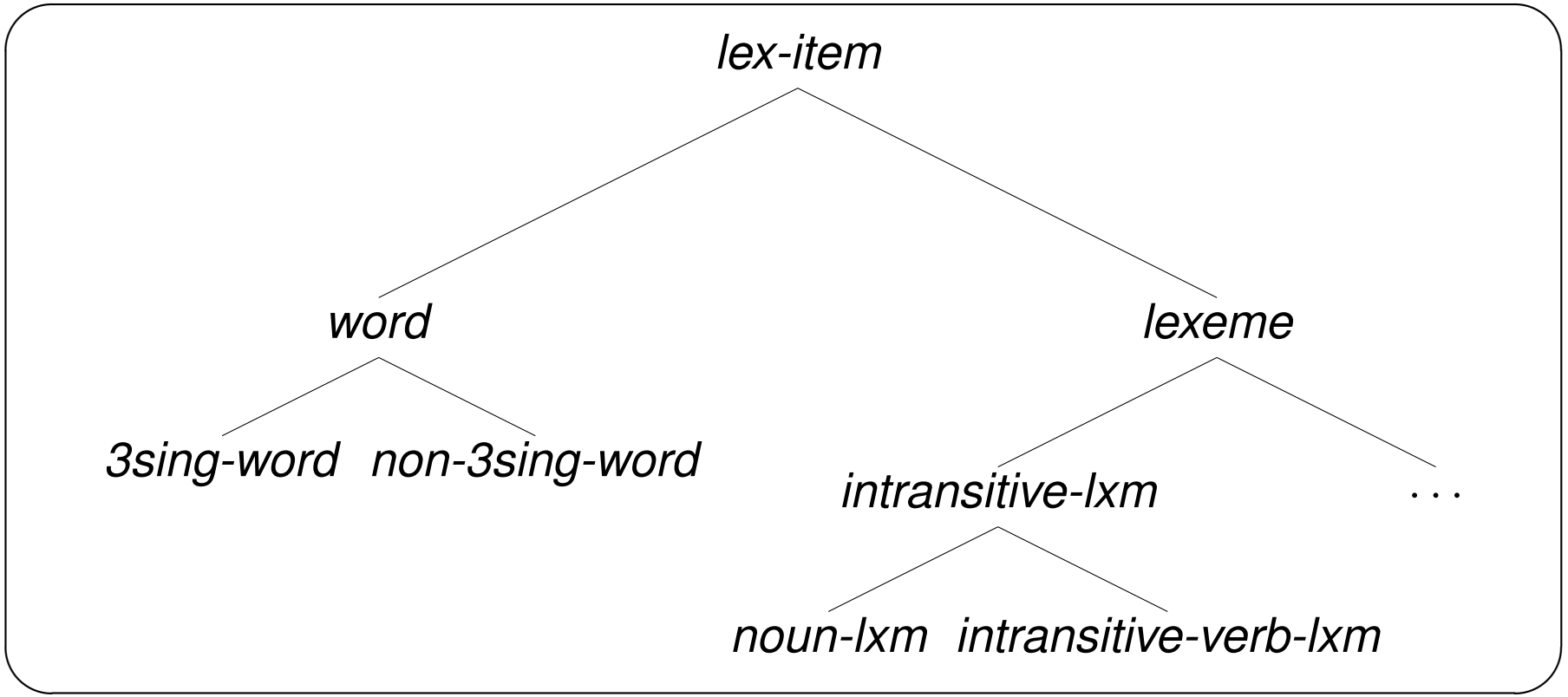
```
%(letter-set (!s abcdefghijklmnopqrtuvwxyz))

noun-non-3sing_irule :=
%suffix (!s !ss) (!ss !ssses) (ss sses)
non-3sing-word &
[ HEAD [ AGR non-3sing ],
  ARGS < noun-lxm > ].

noun-3sing_irule :=
3sing-word &
[ ORTH #1,
  ARGS < noun-lxm & [ ORTH #1 ] > ].
```



The Lexeme vs. Word Distinction



- Lexical entries are *uninflected*; cannot enter syntax by themselves;
- inflectional rules 'make' *word* from *lexeme*, possibly with 'null' suffix.

Recursion in the Type Hierarchy

- Type hierarchy must be finite *after* type inference; illegal type constraint:

```
*list* := *top* & [ FIRST *top*, REST *list* ].
```

- needs additional provision for empty lists; indirect recursion:

```
*list* := *top*.
*ne-list* := *list* & [ FIRST *top*, REST *list* ].
*null* := *list*.
```

- recursive types allow for *parameterized list types* ('list of X'):

```
*s-list* := *list*.
*s-ne-list* := *ne-list* & *s-list* &
                [ FIRST expression, REST *s-list* ].
*s-null* := *null* & *s-list*.
```