

Types and Features: Appropriateness

Features are introduced

- on a (unique) most general type; e.g. MOD on *pos*
- with a value that specifies most general value
Any occurrence of the feature has *at least* that value;
e.g. HEAD is at least *pos*

Certain information can be inferred from a type definition:

- the presence of a feature implies at least the type on which it has been introduced
- the presence of a type implies the presence of the feature(s) it introduces and inherits, and for each of the features, the values that have been specified or inherited

top

top []

ne-list

ne-list [FIRST *top* []
REST *list* []]

[HEAD noun]

[HEAD pos [MOD *list* []]
noun [MOD *list* []]]
SPR *list* []
COMPS *list* []
syn-struct ARGS *list* []]

Recursive Feature Structures

- Recursive types would lead to infinite feature structures

$$\left[\begin{array}{l} *list* \\ FIRST \quad *top* \\ REST \quad *list* \end{array} \right]$$

- Solution: two kinds of lists

$$\left[\begin{array}{l} *ne-list* \\ FIRST \quad *top* \\ REST \quad *list* \end{array} \right] \quad *null*$$

with a common supertype **list**

- **top** cannot have any features

Multiple Inheritance

- Up to now: types make feature values more specific
- Types can have multiple supertypes: combining information

```
*list* := *top*.
```

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

```
*null* := *list*.
```

```
*o-list* := *list*.
```

```
*o-ne-list* := *ne-list* & *o-list* & [ FIRST [ OPT + ],  
                                         REST *o-list* ].
```

```
*o-null* := *null* & *o-list*.
```

Multiple Inheritance (cont.)

- Features are inherited and their values have to unify too:

```
*o-ne-list* := *ne-list* & [ FIRST [ OPT + ],  
                             REST pos ].
```

Constraints on Feature Structures

- Types may be introduced with features and their values
- More complex constraints are also possible, specified as part of a type definition:
`*o-ne-list* := *ne-list* & [FIRST [OPT +]].`

- Used by inference too

$$*o-ne-list* \rightarrow \left[\begin{array}{l} \text{FIRST} \left[\text{OPT} + \right] \\ \text{REST} \quad *o-list* \end{array} \right] = \langle \left[\text{OPT} + \right] \rangle$$

- In multiple inheritance: constraints must be consistent

Greatest Lower Bounds “on-the-Fly”

Unification of types return the *greatest lower bound of two types*

$*ne-list* \sqcap *o-list* \equiv *o-ne-list*$

As with feature structures, unifications of

- two comparable types results in the most specific one
- two compatible types may result in *new* properties
 - new features (specified on greatest lower bound)
 - new values on inherited features: $*o-list* \sqcap *ne-list*$
 - new constraints

The Current Type Hierarchy

