GenI in a Grounding Dialogue System

Alexandre Denis
postdoc on CCCP 2009 > 2011

02/24/09

Context

- Dialogue systems
- Grounding process (Clark & Schaefer, 89) :
  - establishing a common interpretation of an utterance
  - applied on reference resolution
- Requires a real generation module
  - to check what the system really understood
  - to produce utterances the system is able to understand

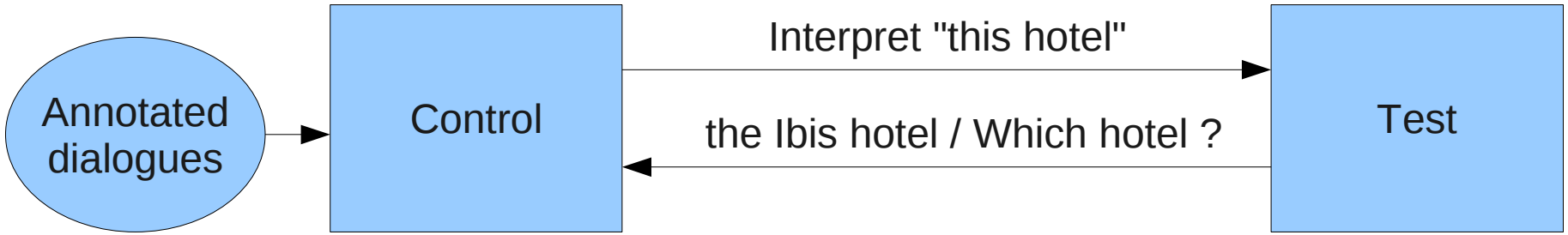
Example

- U : How much is **the first room** ?
- S : **The single room at the Ibis hotel** is 20 euros.

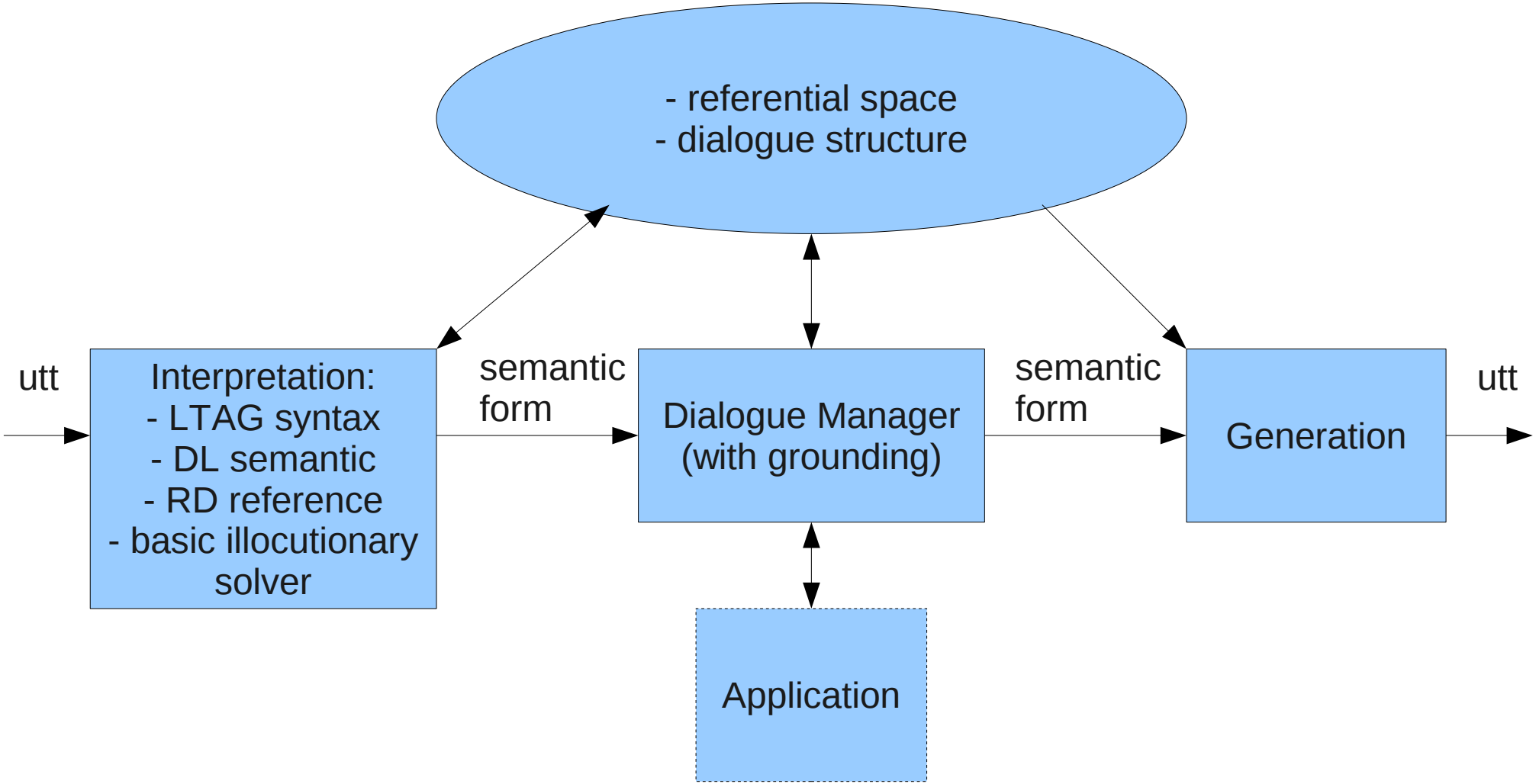
Example

- U : How much is **the first room** ?
- S : **The single room at the Ibis hotel** is 20 euros.

- C : Interpret "How much is **the first room** ?"
- T : **a single room at the Ibis hotel**



Architecture of the system used in my PhD

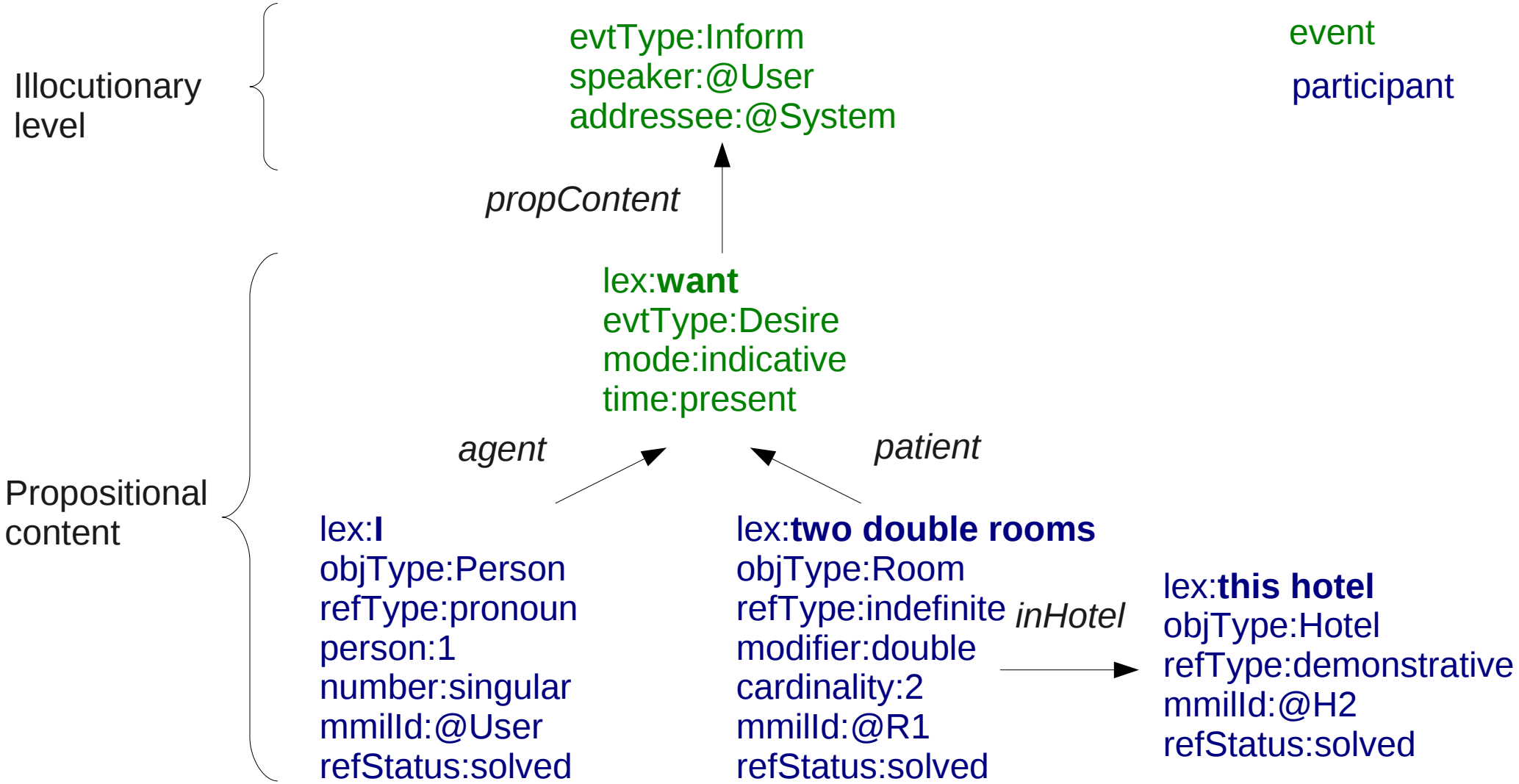


MMIL semantic form (MultiModal Interface Language)

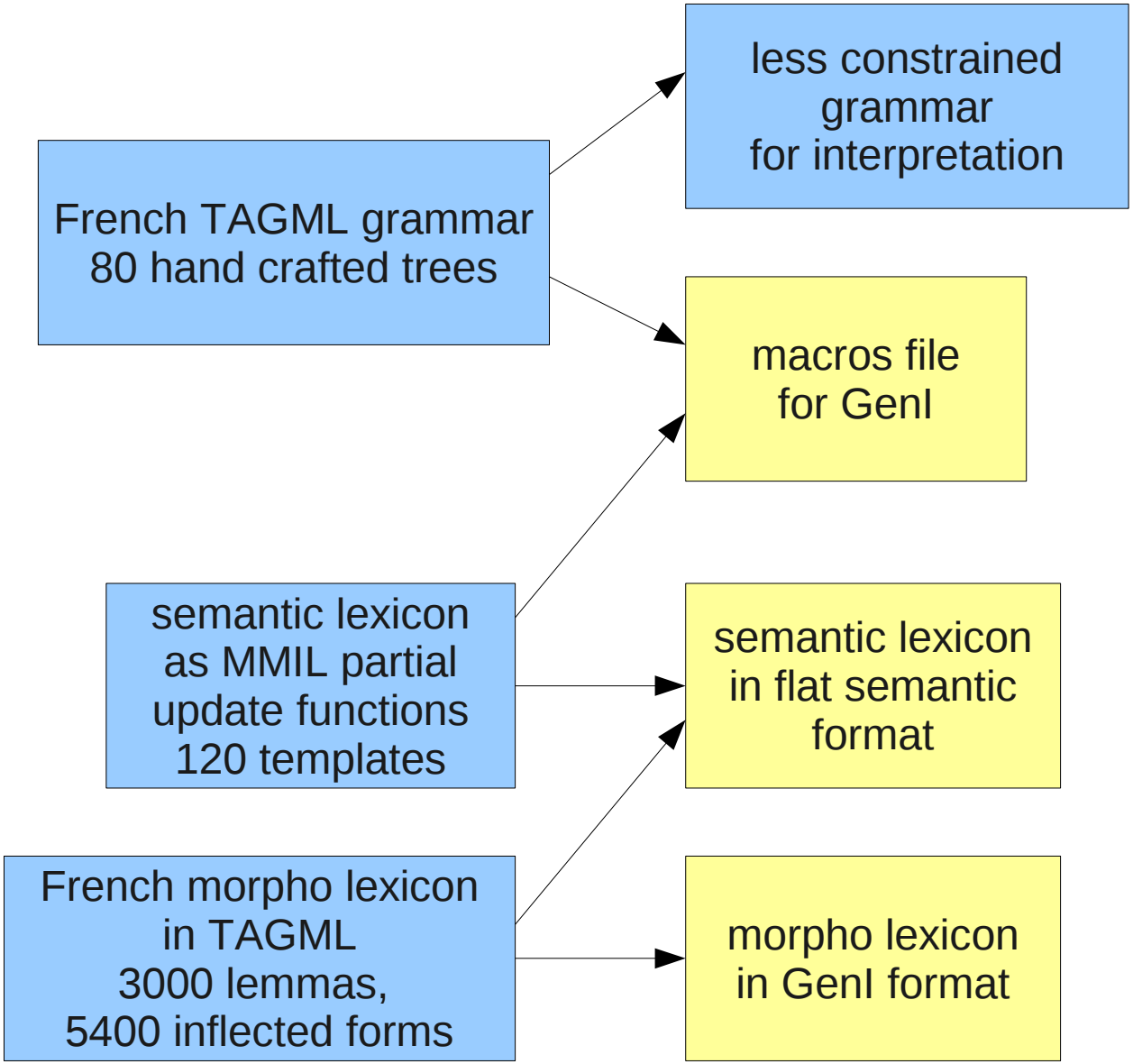
- Landragin et al. 04 (from Romary & Bunt work)
- Multi-level representation format
- Message representation between modules of a dialogue system (including the user)
- Uninterpreted but intended to be projected :
  - prolog : OZONE, for querying application
  - description logic : MEDIA, for reference resolution
  - flat semantic without variables : MEDIA, annotation formalism
  - flat semantic with variables : GenI

MMIL semantic form (MultiModal Interface Language)

- "I want two double rooms in this hotel"

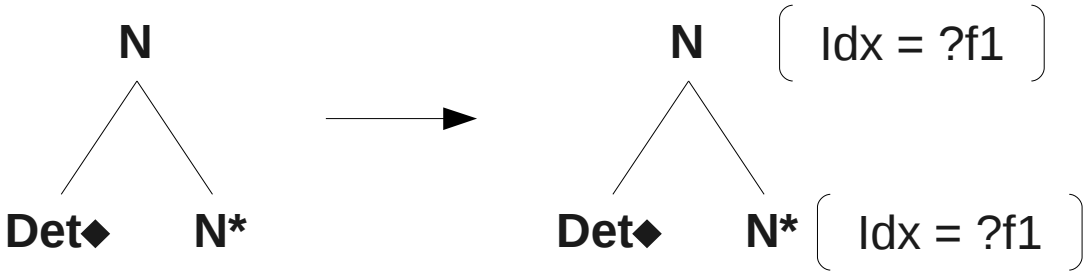


The linguistic resources



Adapting the grammar

- Adding semantic arguments into the grammar with simple rules :
  - Each subst node is a semantic argument
  - Each foot node is a semantic argument
  - If there no subst nor foot node, the anchor node is associated to a semantic argument



Excerpt of the grammar

```
det_n(?f1_1 ! gender:?g number:?n) auxiliary
   n0_0 [cat:Nom det:+ idx:?f1_1 gender:?g number:?n] !
           [cat:Nom det:+ idx:?f1_1 gender:?g number:?n]
   {
      n0_1 anchor [cat:Det gender:?g number:?n] !
                        [cat:Det gender:?g number:?n]
      n1_1 type:foot [cat:Nom det:- idx:?f1_1 gender:?g number:?n] !
                           [cat:Nom det:- idx:?f1_1 gender:?g number:?n]
   }
```

Adapting the semantic lexicon

- An entry is compound of : families or trees, lemmas, type ∈ {participant, event, feature, relation}, set of feature operations

```
<semantic tree="det_n" lemma="un" type="feature">
        <attr name="refType" value="indefinite" mode="add"/>
        <attr name="number" value="@bind" mode="conflict"/>
 </semantic>
```

Argumental structure from the grammar

Feature to bind using morpho

```
"un" det_n(?f1_1 ! number:?number)
semantics:[refType(?f1_1 indefinite) number(?f1_1 ?number)]

"chambre" n(?P ! gender:Fem)
semantics:[participant(?P) objType(?P "chambre")]
```

Morphological feature (!)

Adapting the morphological lexicon

- No big problem (as far as I remember)

TAGML

```
<morph lex="un">
    <fs>
        <f name="gender"><sym value="Mas"/></f>
        <f name="number"><sym value="SG"/></f>
        <f name="cardinality"><sym value="1"/></f>
    </fs>
    <lemmaref cat="Det" name="un"/>
 </morph>
```

Multext-like used by GenI

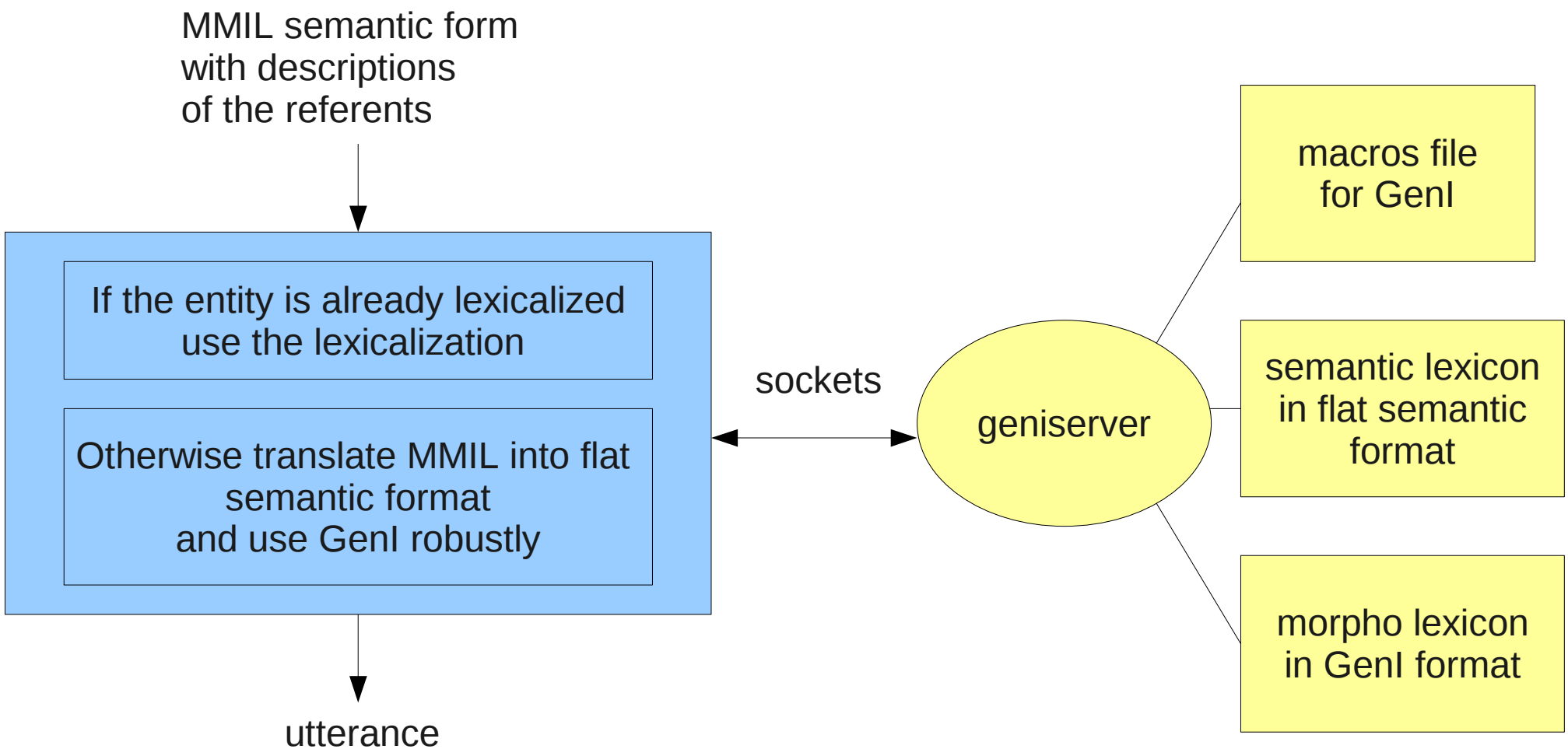
un un [gender:Mas number:SG cardinality:1]

Generating naively referents' description

- Use only asserted description
- Use type of referent to select which part of the description is relevant
- Don't use relational description
  - Not because of recursive loops (see Areces, 08)
  - But because of grounding :
    - C1 : Interpret "... this room ..."
    - T2 : OK (T missed the RE)
    - C3 : no, **a room** at **the Ibis hotel**

The Test system believes falsely that there is an hotel which he missed in C1 !

Interfacing with GenI



Translating MMIL into flat features

- Only use the propositional content without illocutionary force (request or inform e.g)

- Extremely easy : just surround each participant or event by an unary predicate with an id and link each feature to this id

objType:Room
refType:indefinite
modifier:double
cardinality:2

→

participant(p0)
objType(p0 Room)
refType(p0 indefinite)
modifier(p0 double)
cardinality(p0 2)

Generation robustness

- Always take partial realization (thanks Eric)
- If the list of strings returned by GenI is :
  - **empty** : try using a specific pattern-based generation triggered by type
  - **ambiguous** : simply take the longest one (assumed more informative) and the first one if the length is the same

Current direction

- Add the descriptions to the common ground

C : Interpret "the third hotel"
T : OK, the Ibis hotel
C : no, the **EtapHotel** hotel
T : I don't understand. The **Etap** hotel or the Ibis hotel, which hotel ?
C : the **EtapHotel** hotel
T : I don't understand. The **Etap** hotel or the Ibis hotel, which hotel ?
C : nevermind

Current direction

- Add the descriptions to the common ground

C : Interpret "the third hotel"
T : OK, the Ibis hotel
C : no, the **EtapHotel** hotel
T : I don't understand. The **Etap** hotel or the Ibis hotel, which hotel ?
C : the **Etap** hotel
T : OK, the **Etap** hotel
C : OK, interpret ...