NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

MDE

MODELAÇÃO EM FRAMES – GOLOG

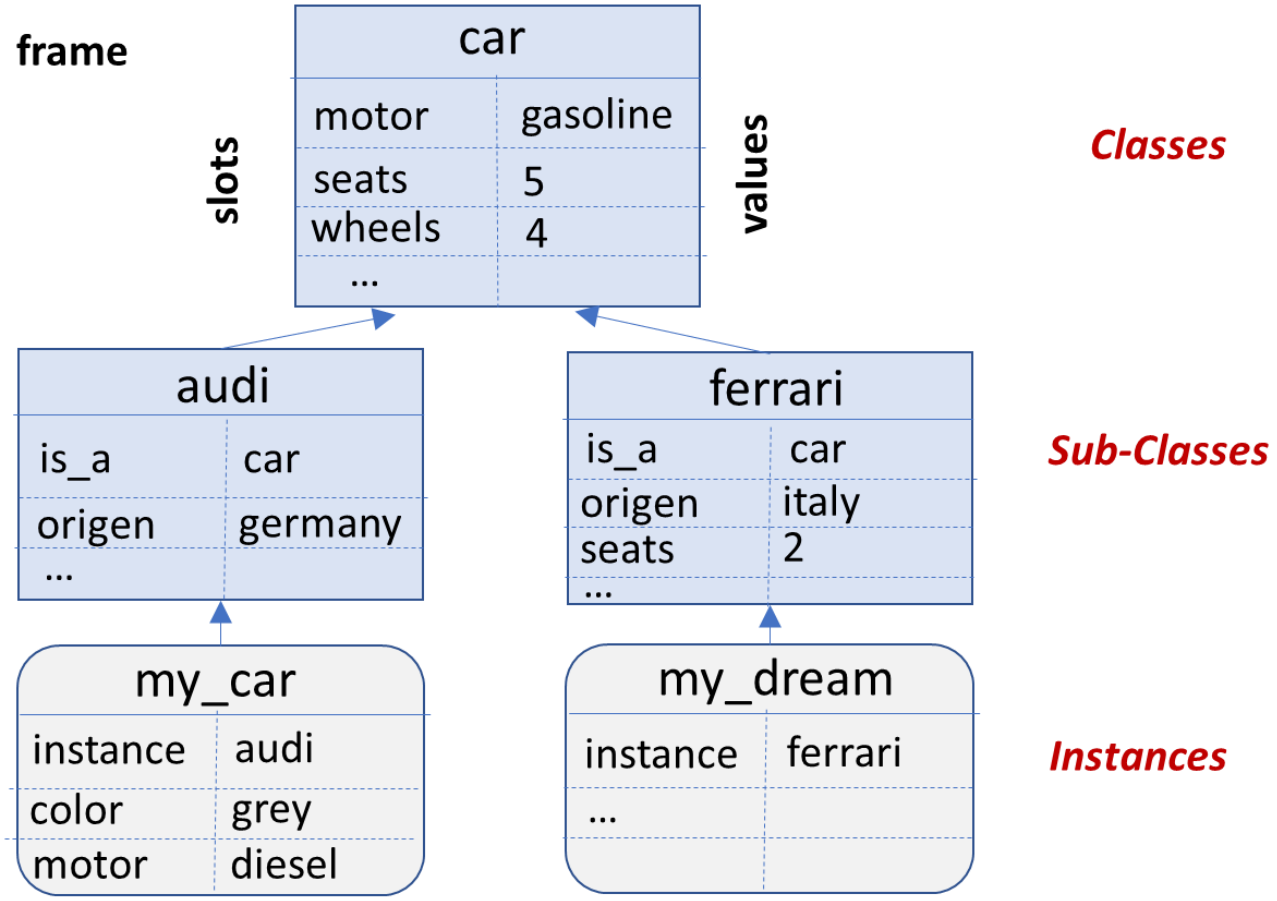
LAB 3

(com base nos slides das aulas teóricas)

2020-2021

TAXONOMIA DE “FRAMES”

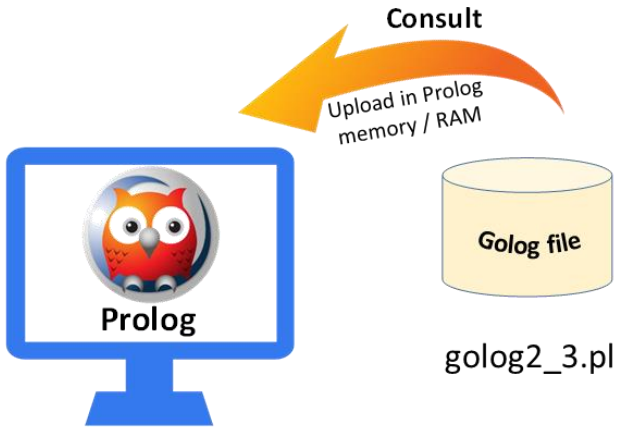
Frames.
Classes. Instâncias.
Slots. Relações. Herança.



GOLOG – MINI FRAME ENGINE EM PROLOG

A mini Frame Engine written in Prolog

- Creation and manipulation of frames and their slots and values.
- Definition of relations and inheritance mechanisms.
- Definition of methods and reactive programming.



GOLOG was inspired on a commercial product (KNOWLEDE CRAFT), one of the first frame engines

USAR NOVA VERSÃO GOLOG2_3.PL

- Mensagens que ajudam a detectar problemas na criação de frames.
- Mecanismo para guardar a base de conhecimento em ficheiro.

  ?- save_kb(nomeFicheiro).
- Mecanismo para apagar a base de conhecimentos.

  ?- delete_kb.
- Outros…

OPERAÇÕES

A. On frames

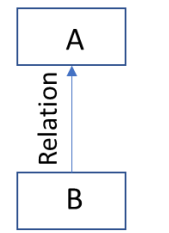
- **new_frame(F)**
- **frame_exists(F)**
- **show_frame(F)**
- **delete_frame(F)**

B. On slots

- **new_slot(F,S)**
- **new_slot(F,S,V)**
- **delete_slot(F,S)**
- **frame_local_slots(F,LS)**
- **get_all_slots(F,LS)**

C. On values

- **new_value(F,S,V)**
- **new_values(F,S,LV)**
- **add_value(F,S,V)**
- **add_values(F,S,LV)**
- **get_value(F,S,V)**
- **get_values(F,S,LV)**
- **delete_value(F,S,V)**
- **delete_values(F,S)**



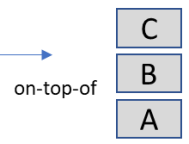
B is related to A

D. On relations

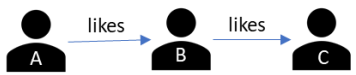
- **new_relation(Relation,Transitivity, Restriction, Inverse)**

Defines a new relation with name Relation.
The definition requires 4 parameters

Transitivity: transitive, intransitive
Restriction: all
none
inclusion(LS)
exclusion(LS)
Inverse: name of the inverse relation or *nil*



The relation "on-top-of" is transitive:
If B is on-top-of A and C is on-top-of B, then we can infer that C is on-top-of A



The relation "likes" is intransitive:
If A likes B and B likes C, we cannot infer that A likes C

- **delete_relation(Relation)**
- **frame_actual_relations(F,LR)**

To delete a relation of name Relation

Obtains a list LR with the names of all relations associated to a given frame F

Allows to automatically create an inverse relation. If we don't want it, put "nil" in this parameter

Specifies the level of inheritance:
all – B inherits all slots of A
none – B does not inherit any slot from A
inclusion(LS) – B only inherits from A the slots mentioned in list LS
exclusion(LS) – B inherits all slots from A except those indicated in list LS

OPERAÇÕES

E. On methods

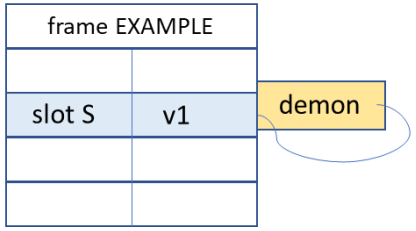
A **method** in frames is a procedure associated with a **class**.
A **method** defines the behavior of the **class** and its sub-classes and instances. A **method** is an action that an object (class) is able to perform.

- **new_slot(F, S, Method)** → *Creates a new Method, which is "stored" in a new slot S.*
- **call_method(F, S, LPar)** → *Calls the method identified by slot S, passing the list of parameters Lpar to the corresponding procedure*
- **call_method_0(F, S)**
- **call_method_1(F, S, P)**
- **call_method_2(F, S, P1, P2)**
- **call_method_3(F, S, P1, P2, P3)**

Particular cases of call_method for the cases that the corresponding procedure has 0, 1, 2, or 3 parameters, respectively

call_method_0(F, S) equivalent to call_method(F,S,[])
call_method_1(F,S,P) " " call_method(F,S,[P])
call_method_2(F,S,P1,P2) " " call_method(F,S,[P1, P2])
call_method_3(F,S,P1,P2,P3) " " call_method(F,S,[P1, P2, P3])

F. On attached predicates or demons



A predicate associated to a slot (hidden) that reacts (executes) when certain actions are performed on the slot

- **add_demon(F,S,Demfunc, Demtype, When, Effect)**
- **remove_all_demons(F,S)**
- **new_demon(F,S,Demfunc, Demtype, When, Effect)**

Demtype:
- if_read — *Demon reacts when some reading action is performed on the slot. E.g. it reacts to get_value, get_values*
- if_write — *Demon reacts when some writing action is performed on the slot. E.g. it reacts to new_value, new_values, add_value, add_values*
- if_executed — *Demon reacts when a call_method is applied*
- if_deleted — *Demon reacts when a delete action is applied*

When:
- before
- after

→ *Indicates if the demon is executed before or after the function that triggered it*

Effect:
- alter_value
- side_effect

→ *Indicates if the demon interferes i.e. changes the parameter of the triggering function*

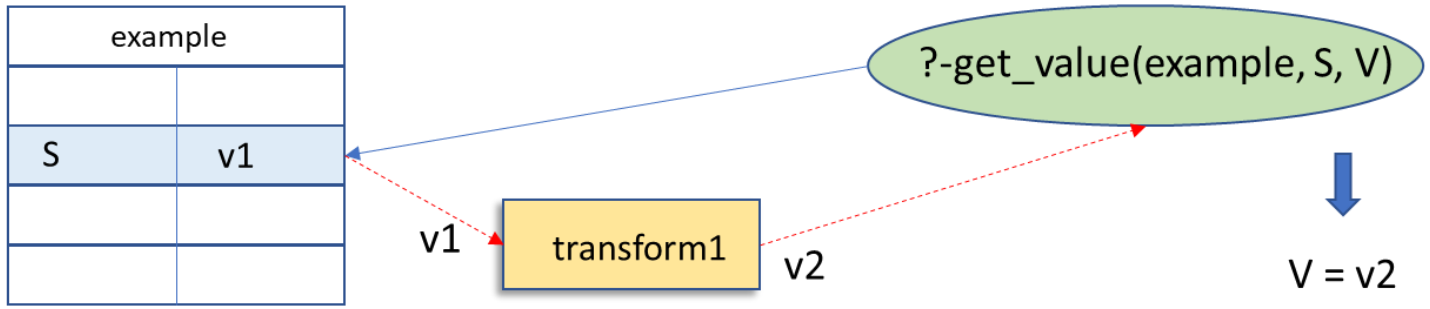
Format of the rule associated to the slot:
E.g.:

```
transform(Frame, Slot, Received_value, Returned_value) :- ....
```

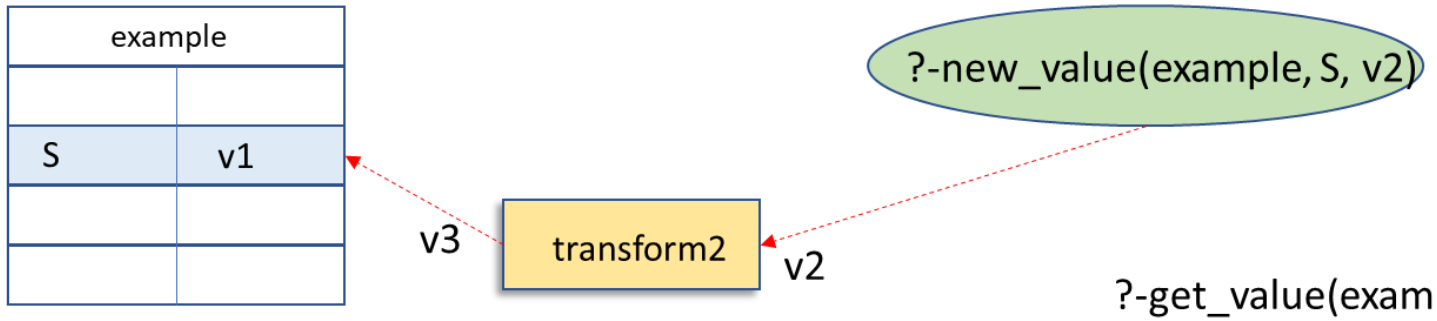
The function (rule) that implements the demon has always 4 parameters

COMO FUNCIONA

?-**new_demon(example, S, transform1, if_read, after, alter_value).**



?-**new_demon(example, S, transform2, if_write, before, alter_value).**



?-get_value(example, S, V).
V= v3