NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

MDE

MODELAÇÃO EM FRAMES – GOLOG

LAB 3

2024-2025

TOPICS

- Labwork 3 classes planning
- Some revisions
- Lab Work Requirements
- Air Conditioning example - Implementation

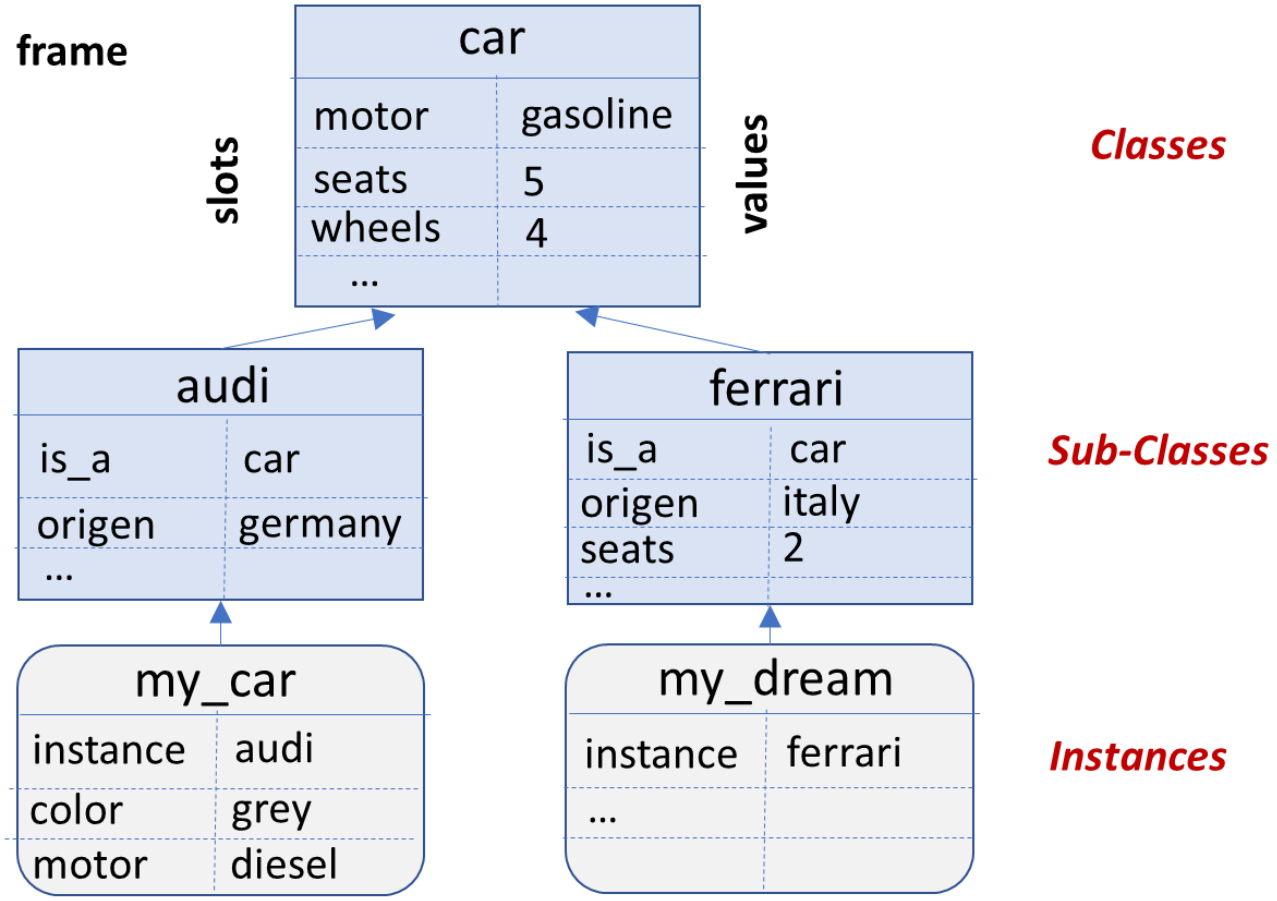
LABWORK 3 CLASSES PLANNING

- **Class 1 [27.05.2025 – 30.05.2025]:**
  - Presentation of the work assignment and the tools to be used.
  - Review of concepts related to programming using the Golog library.
  - Presentation of the problem.
  - Implementation of the AC control system

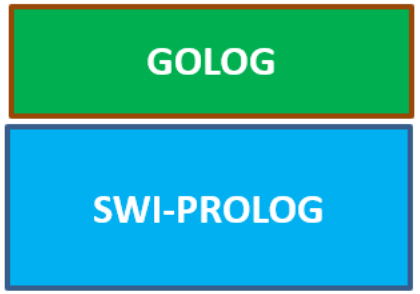
- **Class 2 [03.06.2025 – 06.06.2025]:**
  - Implementation of the functional requirements for the fruit intelligent greenhouse management system.
  - UML diagrams required in part II.
  - Finalization of the assignment.

“FRAMES” TAXONOMY

Frames.
Classes. Instances.
Slots. Relations.

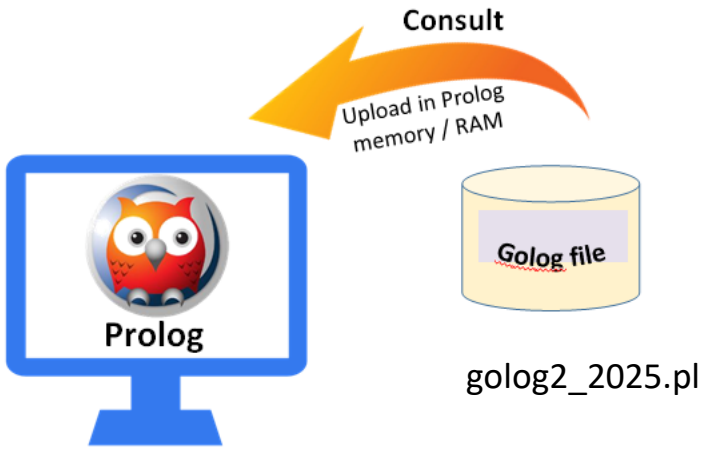


GOLOG – MINI FRAME ENGINE IN PROLOG



A mini Frame Engine written in Prolog

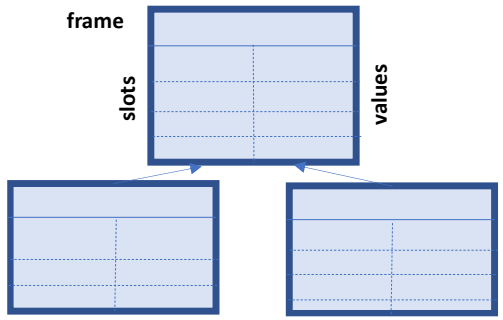
- Creation and manipulation of frames and their slots and values.
- Definition of relations and inheritance mechanisms.
- Definition of methods and reactive programming.



golog2_2025.pl

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

BASIC OPERATIONS



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)

D. On relations

- new_relation(Relation,Transitivity, Restriction, Inverse)
- delete_relation(Relation)
- frame_actual_relations(F,LR)

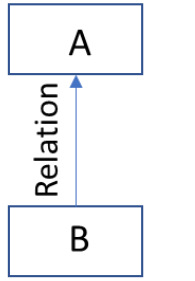
E. On methods

- new_slot(F, S, Method)
- call_method(F, S, LPar)
- 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)

F. On attached predicates or demons

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

BASIC OPERATIONS



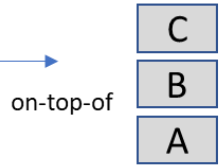
B is related to A

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

BASIC OPERATIONS

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.

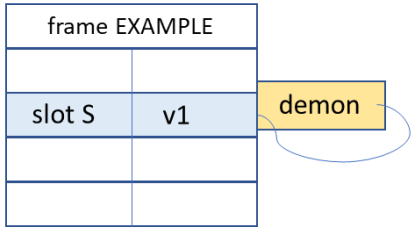
- **new_slot(F, S, Method)** → *Creates a new Method, which is "stored" in a new slot S.*
- **call_method(F, S, LPar)** → *Calls the method identified by slot S, passing the list of parameters Lpar to the corresponding procedure*
- **call_method_0(F, S)**
- **call_method_1(F, S, P)**
- **call_method_2(F,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

BASIC OPERATIONS

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

SOME ADDIONAL FUNCTIONALITIES

- Mechanism to save the knowledge-base to a file
  - ?- save_kb(FileName).
- Mechanism to show the actual knowledge-base
  - ?-show_kb.
- Mechanism to delete the knowledge-base
  - ?- delete_kb.
- Others…

check

golog2_2025.pl

```
% %%
% %%   Predicados Utilitarios
% %%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:-style_check(-singleton).
:-style_check(-discontiguous).

:-dynamic frame_/1.
:-dynamic relation_/1.

% http://stackoverflow.com/questions/130097/real-world-prolog-usage
%
%

save_kb(Filename):-
        tell(Filename),
        forall(show_kb,true),
        told.


show_kb:-
        (   (
          clause(frame_(_),true),
          listing(frame_),
```

LABWORK 3

Departamento de Engenharia Electrotécnica
Secção de Robótica e Manufactura Integrada

Intelligent Greenhouse

Ficha de trabalho Nº 3	
Disciplina	Modelação de Dados em Engenharia
Ano Lectivo	2024/2025
Objectivo	Modelar conhecimento usando Frames e UML
Aulas	2 aulas x 3 horas + 3 horas extra
Data de Entrega	2025/06/09

Objetivos concretos:

1. Modelação baseada em frames
   a. Entender o conceito de modelação com frames e relevância num contexto de engenharia Electrotécnica.
   b. Representação de modelos com a linguagem Prolog usando a biblioteca fornecida GOLOG.
   c. Modelar a informação do problema proposto usando conceito de frames, slots, relações e mecanismos de herança, métodos e demons.

2. Modelação UML
   a. Modelar os tipos de entidades que existem no sistema e como estes se relacionam entre si através, neste caso, de um Diagrama de Classes e outro de Sequência.

Req. Funcionais	Descrição
FR1	Criar, visualizar, alterar e apagar estufas, incluindo parametrização ambiental
FR2	Criar, visualizar, alterar e apagar sensores (adicionar fontes de dados) e frutas (o que está a ser produzido), incluindo simulação de medições e atualizações em tempo real
FR3	Criar encomendas considerando diferentes combinações de conteúdos e quantidades, e disponibilidade
FR4	Visualizar, encomendas feitas considerando o seu estado (em preparação / em entrega / entregue)
FR5	Configurar e simular os níveis de climatização do sistema automático com base nos valores dos sensores
FR6	Configurar lógica adaptativa do sistema de climatização consoante ocupação da estufa
FR7	Gerar e visualizar automaticamente mensagens de alarme com base em eventos críticos
FR8	Diagrama de Classes que suporta o problema

EXAMPLE

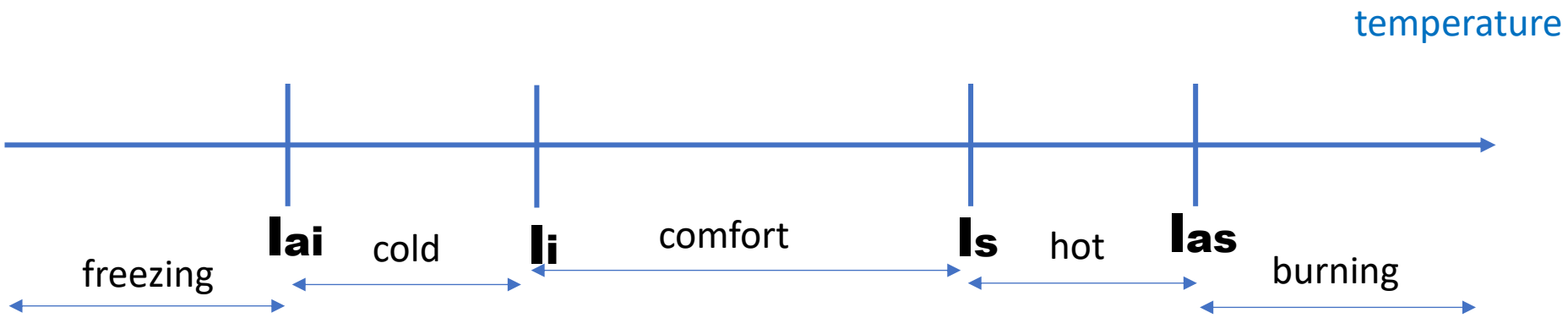
Model an air conditioning system

Use this example on your Labwork 3

AC

EXAMPLE SPECIFICATIONS

The scale of temperatures is characterized by the following diagram:



The level of comfort / discomfort in the room ("**clima**") is classified in 5 levels:
freezing, cold, comfort, hot, burning

Possible **status** of the AC: inactive, warming, cooling

AC offers 3 **operations**: stop, warm, cool

When we check the **temperature** in AC it automatically gets it from the Thermostat

EXAMPLE SPECIFICATIONS

The **control** system is automatic and depends on the temperature in the Thermostat

When the **absolute** upper or lower **limits** Las or Lai are passed, the controller generates an alarm message.

An **alarm** is characterized by:
- event:
- temp: temperature recorded on that occasion
- date: date and time of the alarm event

Generated alarms (represented by frames) are named as alarm1, alarm2, alarm3, …

model_ac:- def_ac, def_thermo, def_alarm.

EXAMPLE: Main “Frames”

ac	
status	inactive
temp	
stop	stopf
warm	warmf
cool	coolf

Demon that reacts to read actions and gets the temperature from the Thermostat

readtd

Operations

thermo	
temp	0
li	14
lai	0
ls	25
las	35
clima	

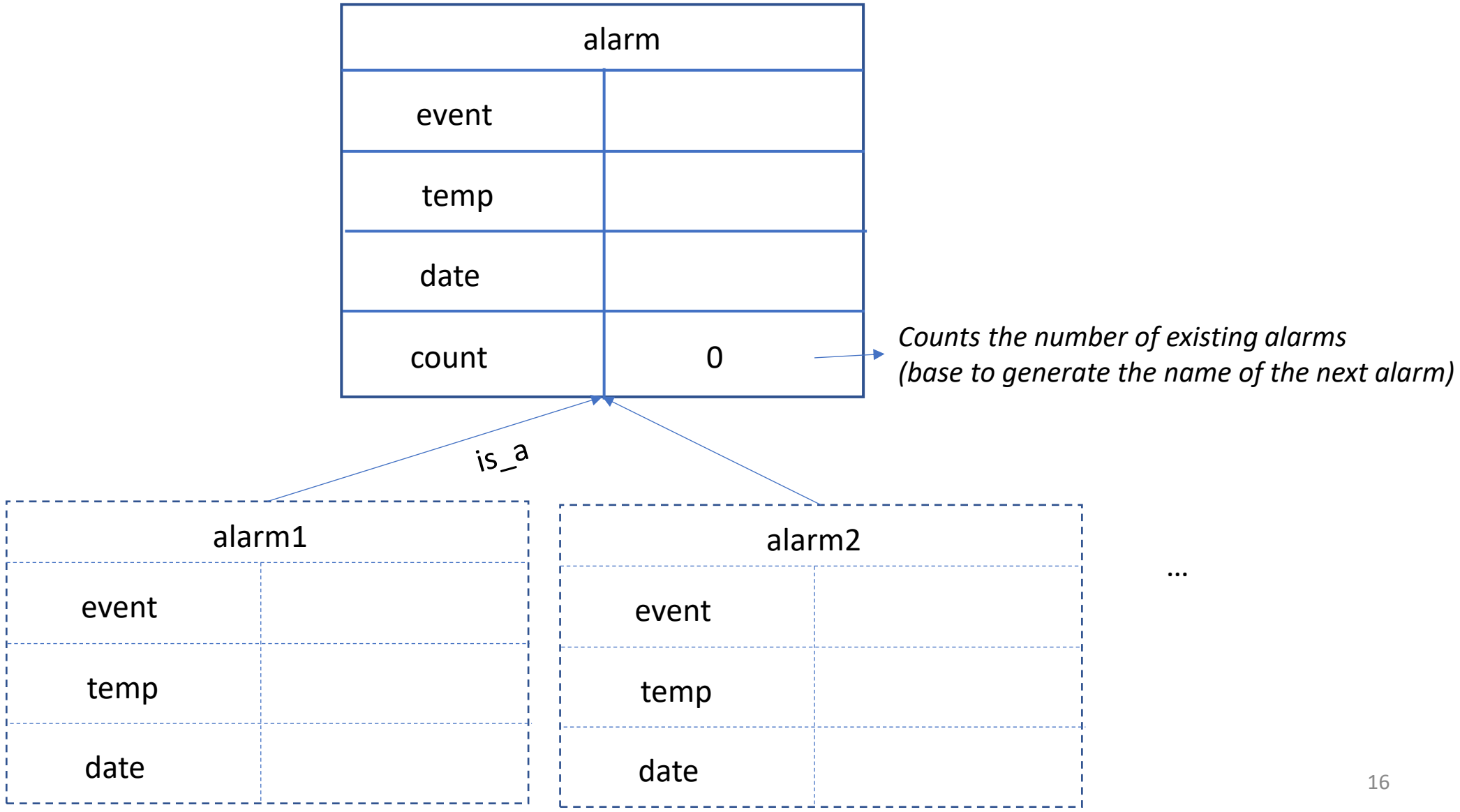
Demon that reacts to write actions and activates the controller

controld

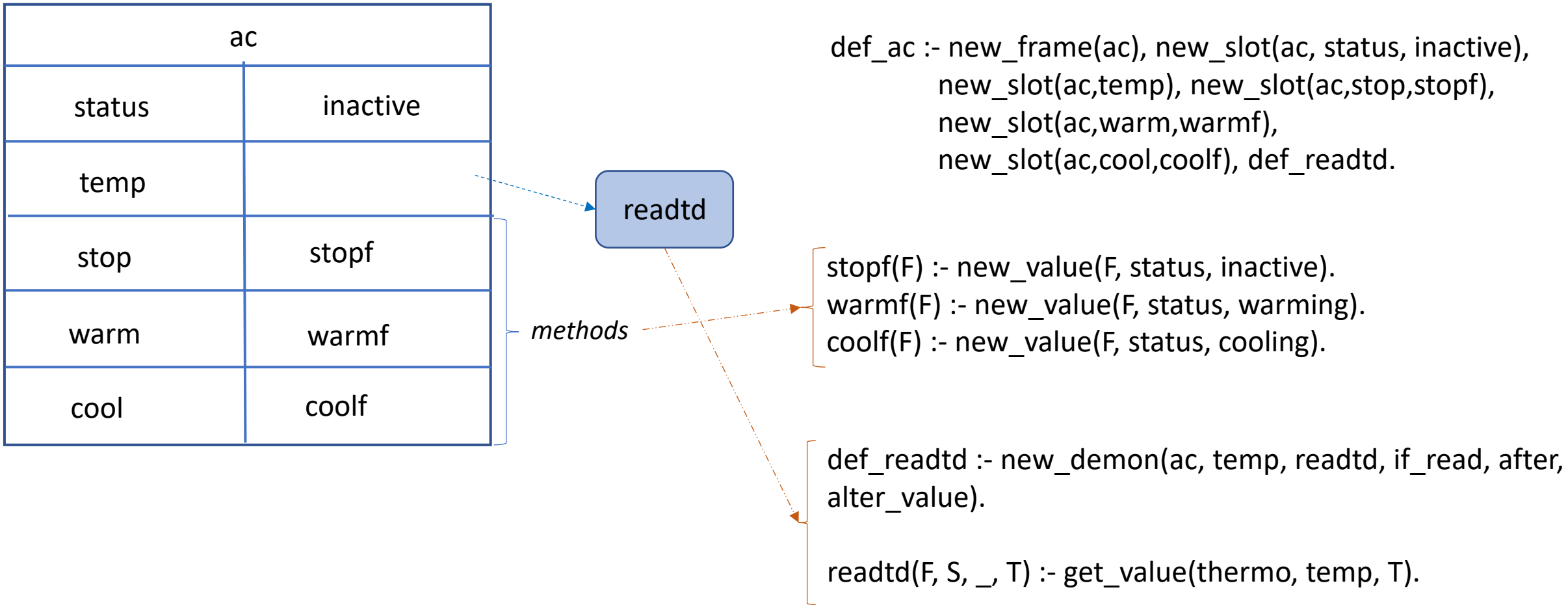
Demon that generates the value of “clima” as a function of the temperature

climad

EXAMPLE: “Frame” alarm



EXAMPLE: Details AC



EXAMPLE: Details THERMO

thermo	
temp	0
li	14
lai	0
ls	25
las	35
clima	

controld

climad

% THERMOSTAT

Complete!

```
def_thermo:- (…),
              def_climad, def_controld.
```

```
def_climad:-new_demon(thermo, clima, climad, if_read, after,
alter_value).
```

```
climad(F,S,_,C) :- get_value(F, temp,T), get_value(F,li,Li),
                   get_value(F,lai,Lai), get_value(F,ls,Ls),
                   get_value(F,las,Las), classify(T,Lai,Li,Ls,Las,C).
```

```
classify(T,_,Li,Ls,_,comfort) :- T>=Li, T=<Ls.
classify(T,Lai,_,_,_,freezing) :- T<Lai.
classify(T,_,_,_,Las, burning) :- T> Las.
classify(T,Lai,Li,_,_, cold) :- T>=Lai, T<Li.
classify(T,_,_,Ls,Las, hot) :- T>=Ls, T=<Las.
```

EXAMPLE: Details CONTROL

% CONTROL

```
def_controld :- new_demon(thermo, temp, controld, if_write, before, side_effect).

controld(F,S,T,T) :- get_value(F,temp,Ta), get_value(F,li,Li), get_value(F,lai,Lai), get_value(F,ls,Ls),
                     get_value(F,las,Las), act(T,Ta,Li,Lai,Ls,Las).


act(T,Ta,Li,Lai,Ls,Las) :- T>Ls, call_method_0(ac,cool), malarm(T,Ta,Li,Lai,Ls,Las).
act(T,Ta,Li,Lai,Ls,Las) :- T<Li, call_method_0(ac,warm), malarm(T,Ta,Li,Lai,Ls,Las).
act(_,_,_,_,_,_):-call_method_0(ac,stop).

malarm(T,Ta,_,_,_,Las) :- T>Las, Ta<Las, getdate(D), genmsg(T, burning,D).
malarm(T,Ta,_,Lai,_,_) :- T<Lai, Ta>Lai, getdate(D), genmsg(T,freezing,D).
malarm(_,_,_,_,_,_).
```

Generate an alarm each time a temperature goes beyond Las or Lai.

EXAMPLE: Details CONTROL

getdate(D) :- **get_time**(T), **stamp_date_time**(T,D,'UTC').

Notes:

?- **get_time(T).**
T = 1684358077.870409

A TimeStamp is a floating point number expressing the time in seconds since 1970-01-01

?- get_time(T),**stamp_date_time(T, D, 'UTC').**
T = 1684358077.870409,
D = date(2023, 5, 17, 21, 14, 37.870409011, 0, 'UTC', -).

Coordinated Universal Time
=> See: https://www.timeanddate.com/time/zones/

?- get_time(T),**stamp_date_time(T, D,local).**
T = 1588683952.532988,
D = date(2023, 5, 17, 22, 14, 37.87040901184082, -3600, 'GMT Daylight Time', true).

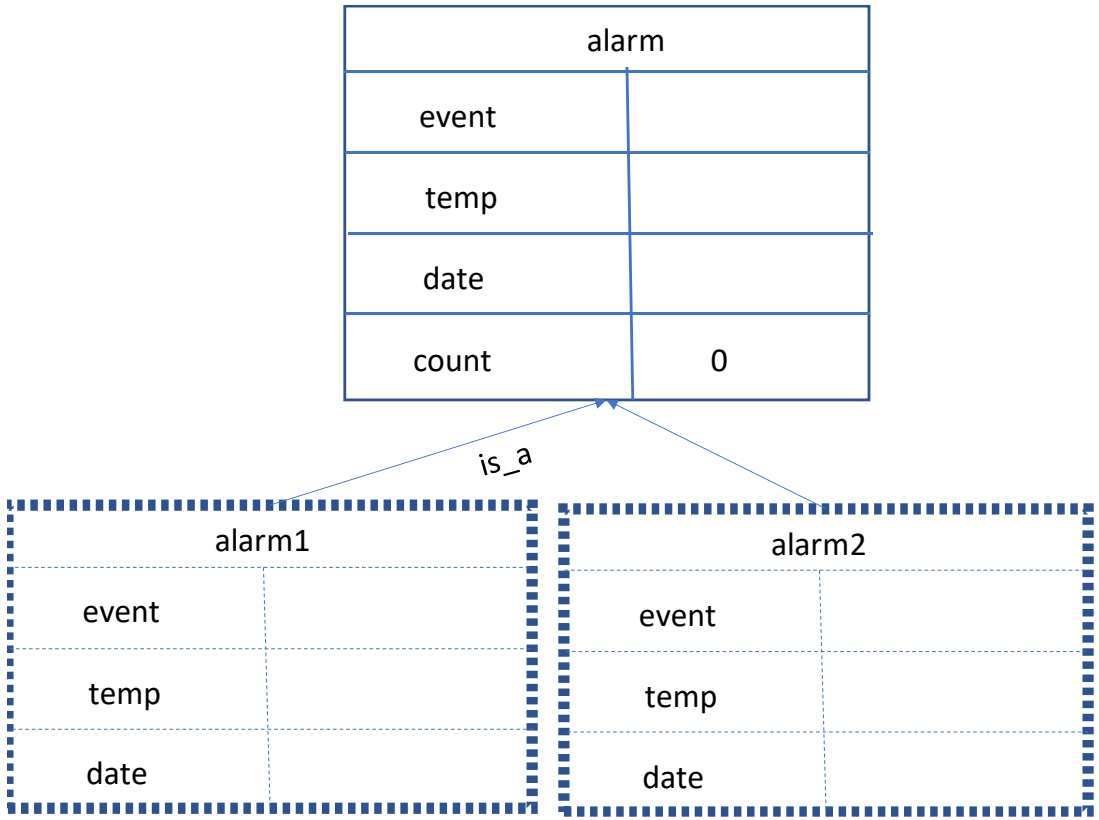
year month day hour minutes seconds

offset relative to UTC in seconds

local timezone

true if daylight saving time applies to the current time

EXAMPLE: Details ALARMS



...

% ALARMS

Complete!

```
def_alarm:- (…),
            def_isa.

def_isa:-new_relation(is_a,…….).

genmsg(T, E, D) :- genname(N), new_frame(N),
                   new_slot(N,is_a,alarm),
                   new_value(N,event,E),
                   new_value(N,temp,T),
                   new_value(N,date,D).

genname(N) :- get_value(alarm,count,A), A1 is A+1,
              new_value(alarm,count,A1),
              atom_concat(alarm,A1,N).
```

EXAMPLE: Details ALARMS

Notes:

atom_concat(*Atom1, Atom2, Atom3*)

Atom3 forms the concatenation of *Atom1* and *Atom2*.
At least two of the arguments must be instantiated to atoms.

```
?- atom_concat(alarm,2,A).
A = alarm2.

?- atom_concat(A,2,alarm2).
A = alarm.

?- atom_concat(alarm,N,alarm2).
N = '2'.
```

EXAMPLE: Final

Main program – entry point.

model_ac:- def_ac, def_thermo, def_alarm.

?- model_ac.
true.

?- show_frame(ac).

```
Frame: ac {
  temp:
    Demons:  readtd
  cool: coolf
  status: inactive
  stop: stopf
  warm: warmf
}

true.
```

?- show_frame(thermo).

```
Frame: thermo {
  clima:
    Demons:  climad
  lai: 0
  las: 35
  li: 14
  ls: 25
  temp: 0
    Demons:  controld
}

true.
```

?- show_frame(alarm).

```
Frame: alarm {
  date:
  event:
  temp:
  count: 0
}

true.
```

?- new_value(thermo, temp, 15).
true.

?- show_frame(ac).

```
Frame: ac {
  temp:
    Demons:  readtd
  cool: coolf
  status: inactive
  stop: stopf
  warm: warmf
}

true.
```

?- get_value(thermo,clima,C).
C = comfort.

EXAMPLE: Final

```
?- new_value(thermo, temp, 30).
true.

?- show_frame(ac).

    Frame: ac {
      temp:
        Demons:  readtd
      cool: coolf
      status: cooling
      stop: stopf
      warm: warmf
    }

true.

?- get_value(thermo,clima,C).
C = hot
```

```
?- new_value(thermo, temp, 48).
true.

?- show_frame(ac).

    Frame: ac {
      temp:
        Demons:  readtd
      cool: coolf
      status: cooling
      stop: stopf
      warm: warmf
    }

true.

?- get_value(thermo,clima,C).
C = burning.

?- get_value(ac,temp,T).
T = 48.
```

Collected from thermostat

```
?- show_frame(alarm).

    Frame: alarm {
      date:
      event:
      temp:
      count: 1
    }

true.

?- show_frame(alarm1).

    Frame: alarm1 {
      date: date(2020,5,5,12,46,7.632052421,0,UTC,-)
      event: burning
      is_a: alarm
      temp: 48
    }

true.
```

EXAMPLE: More interaction

We could, of course, add some I/O to the model …..

```
stopf(F) :- new_value(F, status, inactive), write('AC stopped'), nl.
warmf(F) :- new_value(F, status, warming), write('AC is heating'), nl.
coolf(F) :- new_value(F, status, cooling), write('AC is cooling'), nl.

…

genmsg(T, E, D) :- genname(N), new_frame(N), new_slot(N,is_a,alarm), new_value(N,event,E),
                   new_value(N,temp,T), new_value(N,date,D), write('New alarm: '), write(N), nl.

…
```

LABWORK 3

Intelligent Greenhouse

Use this example to continue with Labwork3…..