Agentes Libres

La Inteligencia Artificial, IA, suena a tema de Ciencia Ficción y de proyectos distantes. En esta charla presentaremos un esfuerzo real y endógeno para desarrollar software de IA en un lenguaje que ha sido el centro de una polémica política. Aun cuando el proyecto apunta al objetivo tradicional de la IA (El dispositivo esclavo perfecto), creemos que es posible extraer algunas lecciones mas generales sobre Inteligencia. Una de esas lecciones es la posibilidad de representar sistemáticamente el conocimiento que guía nuestra conducta (la humana) y de aprender (también sistemáticamente y en el mismo sentido que se usa en Datamining) a partir de esas representaciones. La charla no requiere de experiencia previa en IA, pero los interesados pueden revisar el texto y el software asociado en http://gloria.sourceforge.net

Jacinto Dávila
jacinto@ula.ve
Universidad de Los Andes. Mérida, Venezuela

Lógica Computacional como el componente pensante de un agente inteligente

El Universo

Para **ciclar**,

observe el universo,

piense,

decida que hacer,

actúe,

ciclar de nuevo.

Bob Kowalski, http://doc.ic.ac.uk/~rak

Para **ciclar**,

observe el universo,

aprenda de sus acciones pasadas,

piense,

decida que hacer,

actué,

ciclar de nuevo.

To **cycle**,

think,

decides what actions to perform,

act,

observes the world,

learns from its past actions,

cycle again.

Observables

Metas

- Metas de logro
- Metas de mantenimiento

Creencias

Ejecutables

Observación: Ocurre un evento.

Razonamiento hacia adelante:

El evento corresponde con las condiciones de una meta de mantenimiento o una creencia.

Meta de logro:

En algún punto, la conclusión de una meta de mantenimientose convierte en una meta de logro.

Razonamiento hacia atrás:

Se usan las creencias para reducir las metas de logro a acciones específicas.

Acciones:

Se seleccionan las acciones a ejecutar.

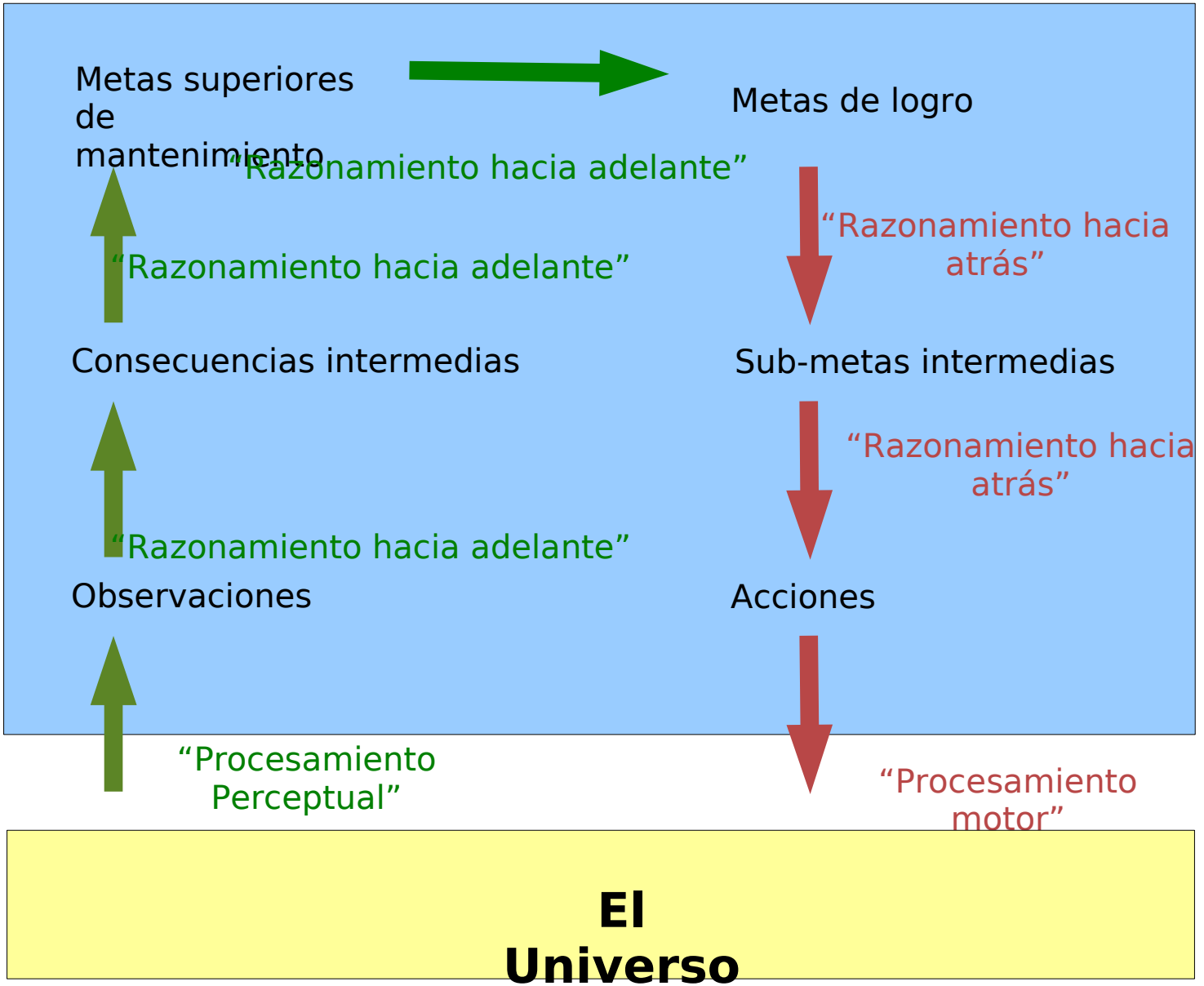
Observación:

El agente observa si su acción tiene éxito o falla.

Ocurre un evento

El agente intenta la acción

Bob Kowalski, http://doc.ic.ac.uk/~rak



Bob Kowalski, http://doc.ic.ac.uk/~rak

Meta: If there is an emergency then I get help.

Creencias: A person gets help if the person alerts the driver.

A person alerts the driver if the person presses the alarm signal button.

There is an emergency if there is a fire.

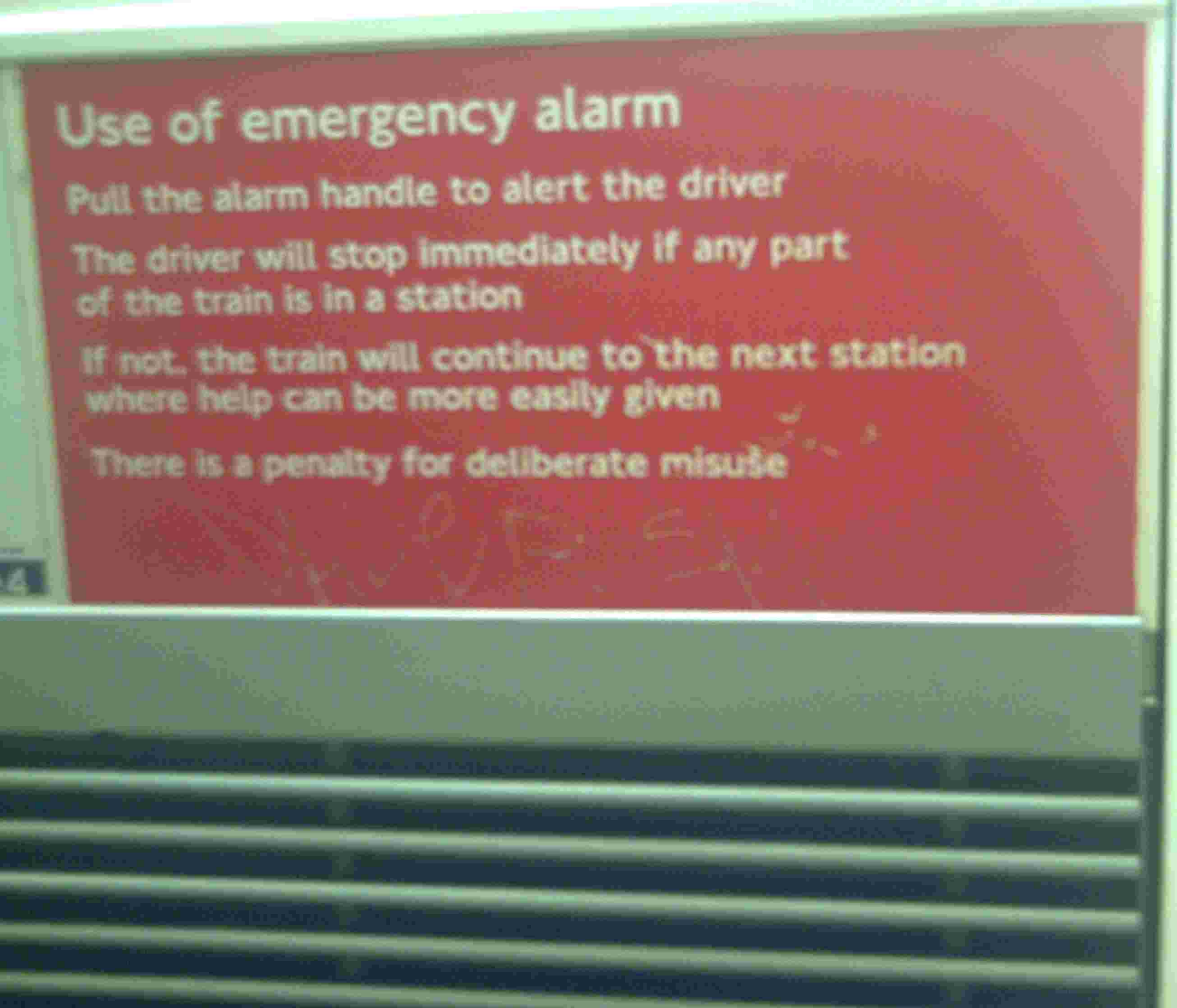
There is an emergency if one person attacks another.

There is an emergency if someone becomes seriously ill.

There is an emergency if there is an accident.

There is a fire if there are flames.

There is a fire if there is smoke.


Use of emergency alarm
Pull the alarm handle to alert the driver
The driver will stop immediately if any part of the train is in a station
If not, the train will continue to the next station where help can be more easily given
There is a penalty for deliberate misuse

Observación: **There is smoke.**

Razonar hacia adelante, nueva creencia: **There is a fire.**

Razonar hacia adelante, nueva creencia:

There is an emergency.

Razonar hacia adelante, meta de logro:

I get help!

Razonar hacia atrás, sub-meta: **I alert the driver!**

Razonar hacia atrás, acción: **I press the alarm signal button!**

Bob Kowalski, http://doc.ic.ac.uk/~rak

Meta: If I am told to build an arch then I build the arch.

Creencias: A person build an arch if the person builds a pair of columns and a beam.

A person builds a pair of columns if the person finds the parts and builds one column and the other.

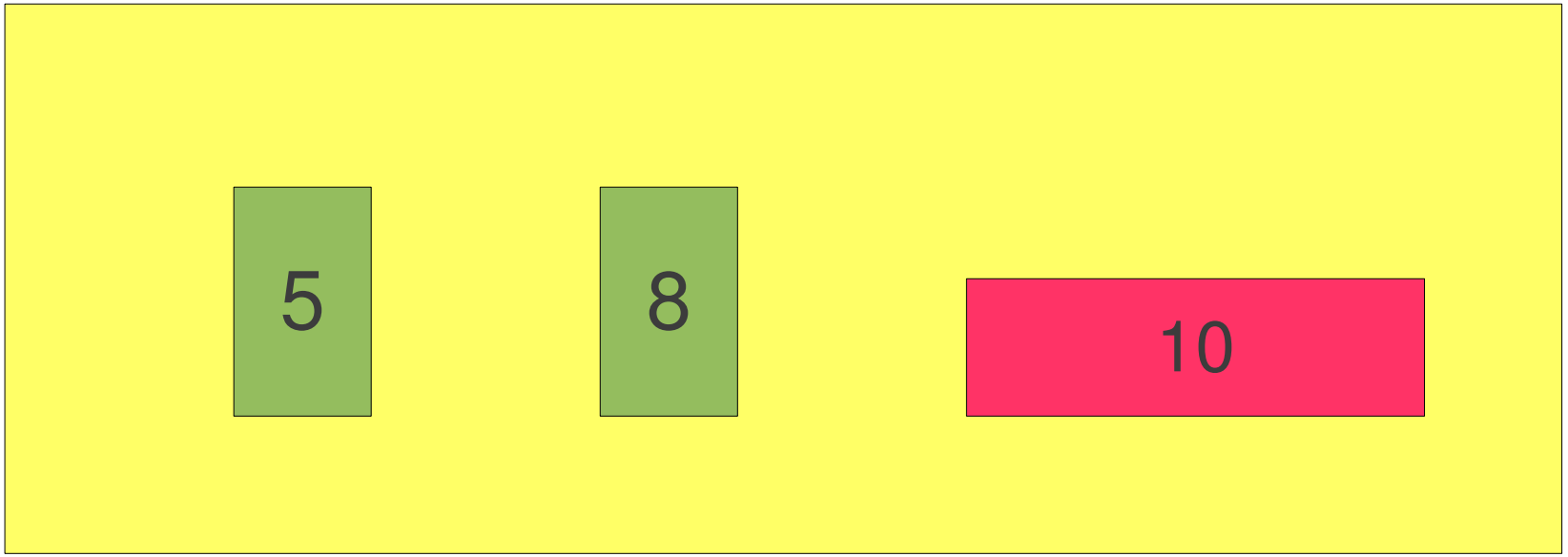
A person builds a beam if the person finds an object and puts it as beam.

Meta: If build(A) then arch(A).

Creencias: To arch(A) do col(C1, C2), beam(B).

To col(C1,C2) do c(C1), c(C2), do(C1), do(C2).

To beam(B) do b(B), do_b(B).

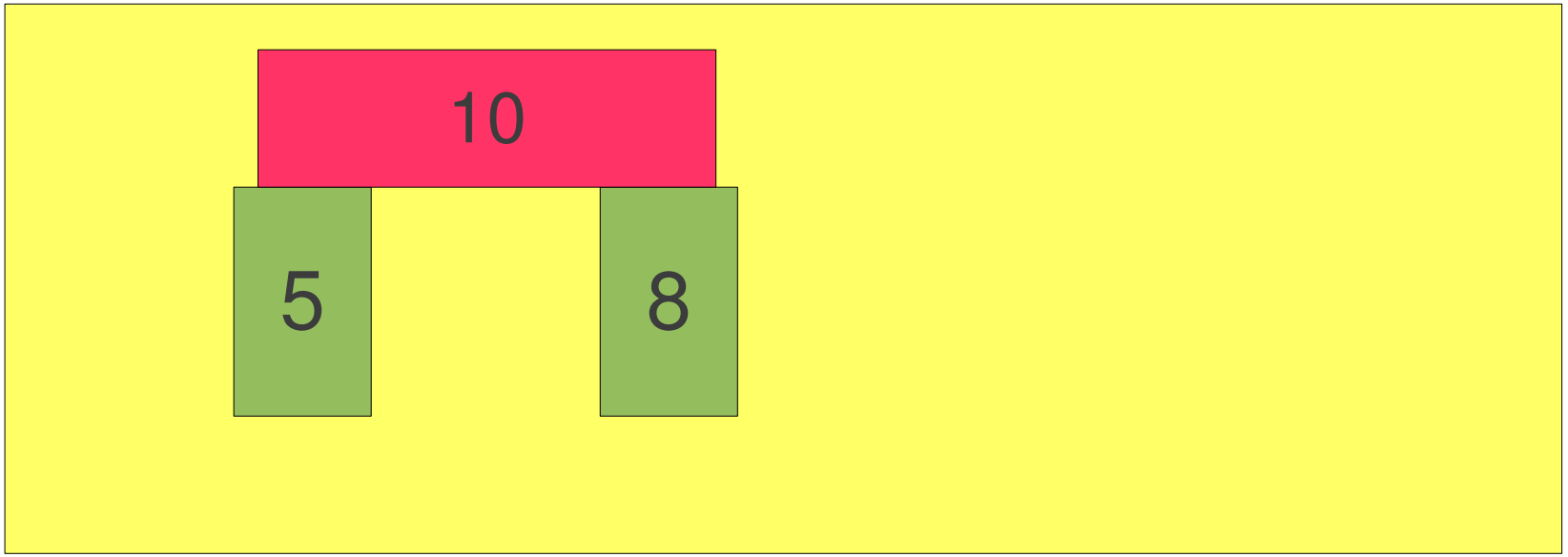


Metas: If build(A) then arch(A).

Creencias: To arch(A) do col(C1, C2), beam(B).

To col(C1,C2) do c(C1), c(C2), do(C1), do(C2).

To beam(B) do b(B), do_b(B).

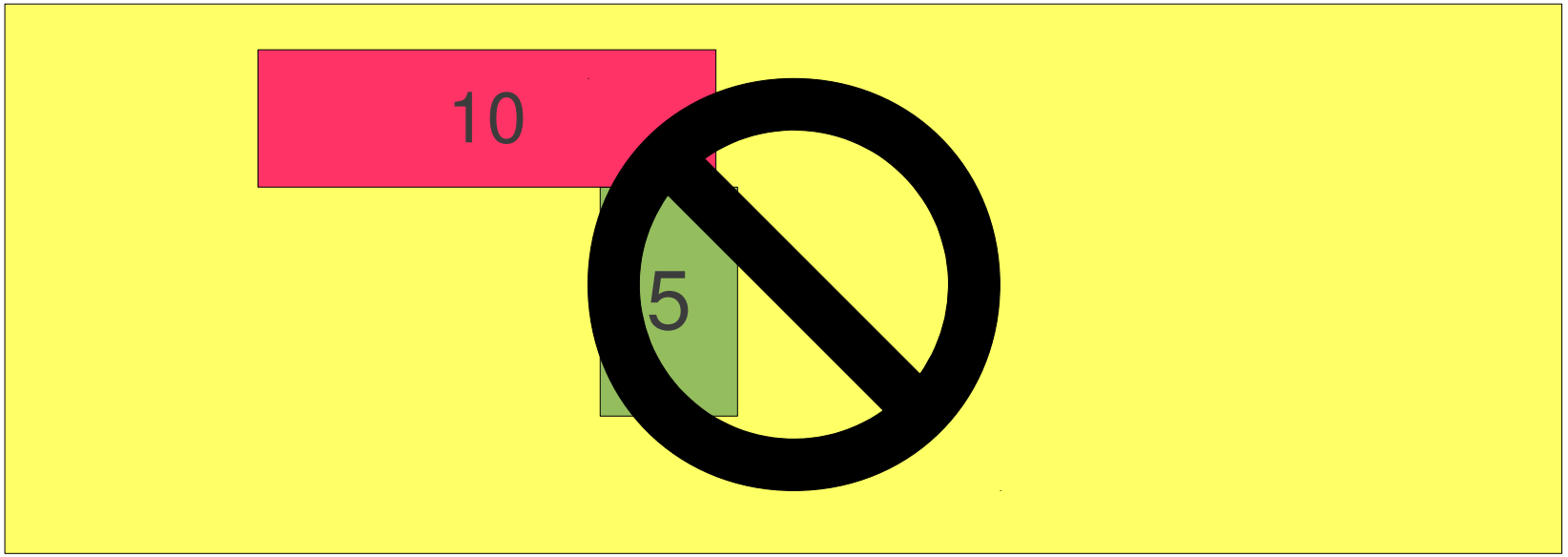


Meta: If build(A) then arch(A).

Creencias: To arch(A) do col(C1, C2), beam(B).

To col(C1,C2) do c(C1), c(C2), do(C1), do(C2).

To beam(B) do b(B), do_b(B).

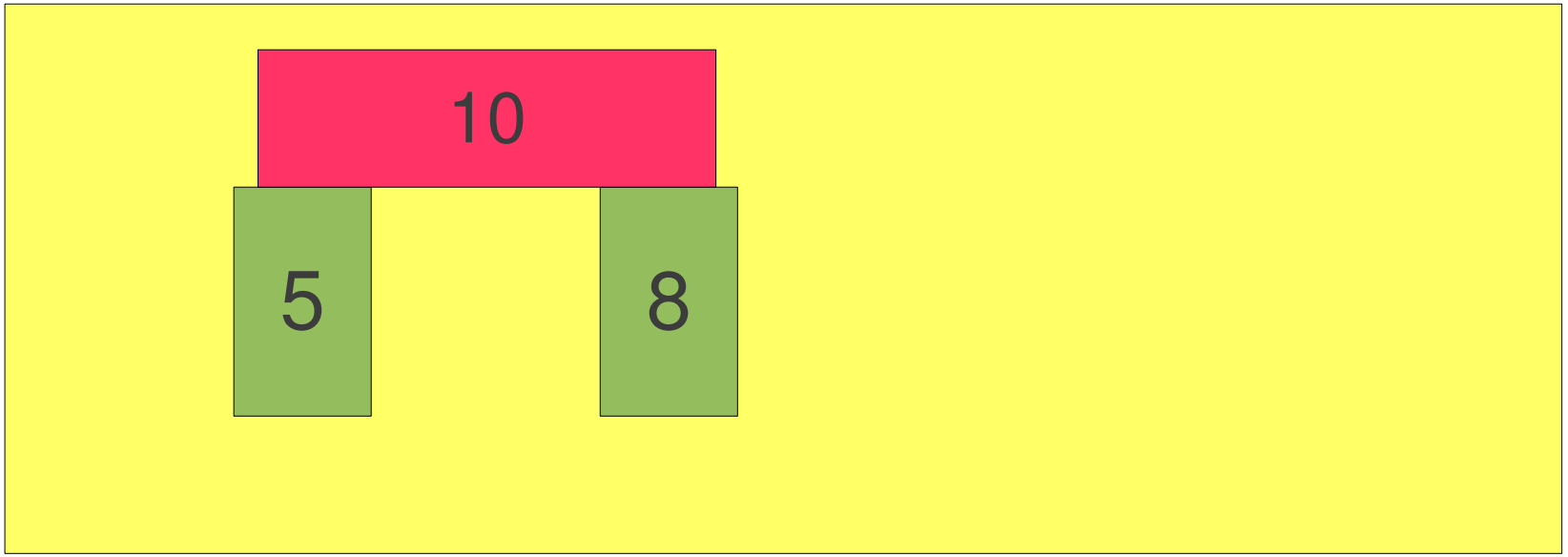


Meta: If build(A) then arch(A).

Creencia: To arch(A) do col(C1, C2), beam(B).

To col(C1,C2) do neq(C1, C2), c(C1), c(C2), do(C1), do(C2).

To beam(B) do b(B), do_b(B).



Veamos

Base de Conocimiento Inicial

```
arch ( A) : - col (C1 , C2 ), beam (B ).
col (C1 , C2 ) : - c ( C1 ) , c ( C2 ), do ( C1 ), do ( C2 ).
beam ( B) : - b( B), dob (B ).
abd ( do ). abd ( dob ). abd (c ). abd (b ).
observables ( c(_ )). observables ( b(_ )).
```

Traza de la Ejecución

```
?- learn (C ).
a (( col ( G234 , G235 ): - c( G234 ), c( G235 ), do ( G234 ), do ( G235 )) , [ obj ( G234 ),
obj ( G235 )])
... specialised into :
col ( G234 , G235 ): - neq ( G234 , G234 ), c( G234 ), c ( G235 ) , do ( G234 ) , do
( G235 )
3.. Found clause : col ( G627 , G628 ): - neq ( G627 , G628 )
C = [ ( col ( G627 , G628 ): - neq ( G627 , G628 )) ,
( col ( G234 , G235 ): - neq ( G234 , G234 ), c ( G234 ) , c( G235 ), do ( G234 ), do
( G235 ))]
```

Meta: **If someone ask me to tell a story then I tell a story.**

Creencias: **A person tells a story if she selects a sentence with the noun before the pronoun.**

A person tells a story if she selects a sentence with the pronoun before the noun.

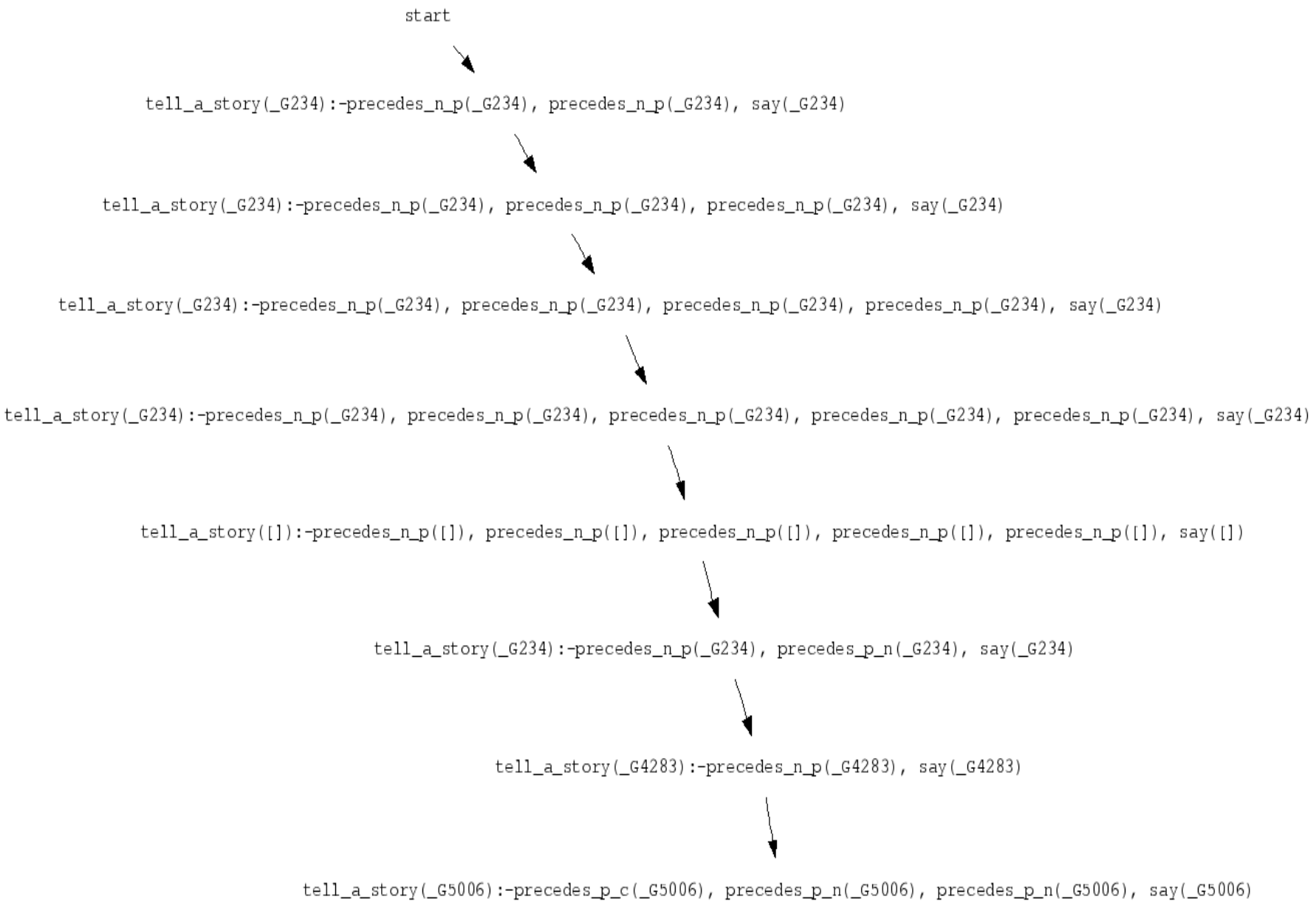
Susie had a shower after she got up
(Susie se duchó luego de levantarse).

After Susie got up, she had a shower
(Luego de que Susie se levantara, se duchó)

After she got up, Susie had a shower.
(Luego de levantarse, Susie se duchó).

* She had a shower after Susie got up
(Se duchó luego de que Susie se levantara)

Veamos


start
tell_a_story(_G234):-precedes_n_p(_G234), precedes_n_p(_G234), say(_G234)
tell_a_story(_G234):-precedes_n_p(_G234), precedes_n_p(_G234), precedes_n_p(_G234), say(_G234)
tell_a_story(_G234):-precedes_n_p(_G234), precedes_n_p(_G234), precedes_n_p(_G234), precedes_n_p(_G234), say(_G234)
tell_a_story(_G234):-precedes_n_p(_G234), precedes_n_p(_G234), precedes_n_p(_G234), precedes_n_p(_G234), precedes_n_p(_G234), say(_G234)
tell_a_story([]):-precedes_n_p([]), precedes_n_p([]), precedes_n_p([]), precedes_n_p([]), precedes_n_p([]), say([])
tell_a_story(_G234):-precedes_n_p(_G234), precedes_p_n(_G234), say(_G234)
tell_a_story(_G4283):-precedes_n_p(_G4283), say(_G4283)
tell_a_story(_G5006):-precedes_p_c(_G5006), precedes_p_n(_G5006), precedes_p_n(_G5006), say(_G5006)

* She had a shower after Susie got up

(Ella se duchó luego de que Susie se levantara)

“Una anáfora no puede preceder y controlar a su antecedente”

“An anaphor may not both precede and command its antecedent”

Ronald Langacker, 1969

“U

El nuevo Agente Aprendiz

$$
\begin{aligned}
&cycle(KB, Goals, T) \leftarrow \\
&\quad demo(KB, Goals, Goals', Rt) \wedge . \\
&\quad Rt \leq n \wedge . \\
&\quad act(KB, Goals', Goals'', Examples, RevisableRules, T + Rt) \wedge . \\
&\quad learn(KB, KB', Examples, RevisableRules, T + Rt + 1, Rl) \wedge . \\
&\quad Rl \leq m \wedge . \\
&\quad cycle(KB', Goals'', T + Rt + 1 + Rl)
\end{aligned}
$$

```
/
**********************************************************************
executing */
% an implementation of the act predicate

executing([[Influences, Rest, CN, HF, HH]|RestG],
                [[NewInfluences, Rest, NewCN, HF, HH]|RestG],
                Examples, Rules_to_reLearn ) :-
    testing(Influences, Feedback),
    observing(Raw_Observations),
    append(Raw_Observations, Feedback, Observations),
    criticising(Observations, Obs, Examples,
Rules_to_reLearn),
    and_append(Obs, Influences, NewInfluences)
```

```
/*****************************************************************************
critising */
% criticising builds the list of examples requires by the learning task.
% criticising( Feedback, Observations, Examples, Rules to review).
%
criticising([], true, [], []).

criticising([failed(Action, Goal)|RestActs], R, [-Goal|RestExamples],
New_Rules) :-
    saving(Action :- true),
    get_rules_for(Goal, This_Rules),
    criticising(RestActs, R, RestExamples, Rules),
    append( Rules, This_Rules, New_Rules).

% criticising([succeded(Action, Goal)|RA], R, [+Goal|RestExamples],
Rules) :-
criticising([succeded(Action, Goal)|RA], R, RestExamples, Rules) :-
    saving(Action :- true),
    criticising(RA, R, RestExamples, Rules).

criticising([Obs|RA], (todo(see, Obs), RObs), Examples, Rules) :-
    criticising(RA, RObs, Examples, Rules).
```

```
% Specialisation by refining the false clause and
adding the
% refined version instead of the other

specialise(Cls,Done,Example,Clauses):-
        false_clause(Cls,Done,Example,C),
       specialise_clause(C,a(S, _)),
       C = a(To_be_removed, _),
       remove_one(To_be_removed, Cls, Cls1),
       process_examples([S|Cls1],[],[-Example|
Done],Clauses).
```

```
% specialise_clause(C,S)
% <- S is a minimal specialisation of C under theta-subsumption

specialise_clause(Current,Spec) :- add_literal(Current,Spec).

specialise_clause(Current,Spec) :- apply_subs(Current,Spec).

add_literal(a((H:-true),Vars),a((H:-L),Vars)):-!,
        literal(L,LVars),
      not(would_unify(L, H)), % preventing recursive clauses
      ss(LVars, Vars). % LVars strict subset of Vars,  using a new
predicate

add_literal(a((H:-B),Vars),a((H:-L,B),Vars)):-
        literal(L,LVars), not_too_big((L,B)), % Problem-specific
condition
      not(would_unify(L, H)), % preventing recursive clauses
      ss(LVars, Vars). %
...
```

.. Perfectionist vs condescending, adaptive agents

$$g(X_{1,}.., X_k) \leftarrow P_1 \vee P_2 \vee .. \vee P_n$$

$$(P_1 \vee P_2 \vee .. \vee P_n)\theta \qquad Pi \equiv (C_{i1} \wedge C_{i2} \wedge .. \wedge C_{ir_i})$$

1. A Pi fails, i. e., at least an action in that plan failed, for every possible value of its variables. That is, that Pi has never succeeded.
2. Every Pi fails, for every possible value of their variables.
3. A Pi have been tried a certain number of times, m and it has failed in all of them (this could be seen as attempting PiΘj, for 0<j<n), where each Θj is a different substitution for the variables in the goal.
4. Every Pi has failed a given number of time, m, as just defined.

A perfectionist agent will go for a revision of its definition with option 3 and m=1. A condescending (lazy) agent will allow for option 4 and a bigger value for m.

Qué sigue?

- Agentes Aprendices para Optimización Basada en Simulación
- Agentes en Servicios Web

Tesis disponibles

Gracias

Jacinto Dávila
Universidad de Los Andes
Centro de Simulación, CESIMO.
La Hechicera,
Mérida, Venezuela.

jacinto@ula.ve
http://webdelprofesor.ula.ve/ingenieria/jacinto
http://galatea.sourceforge.net