Fuzzy Prolog

Susana Muñoz-Hernández

Facultad de Informática

Universidad Politécnica de Madrid

28660 Madrid, Spain

susana@fi.upm.es

Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

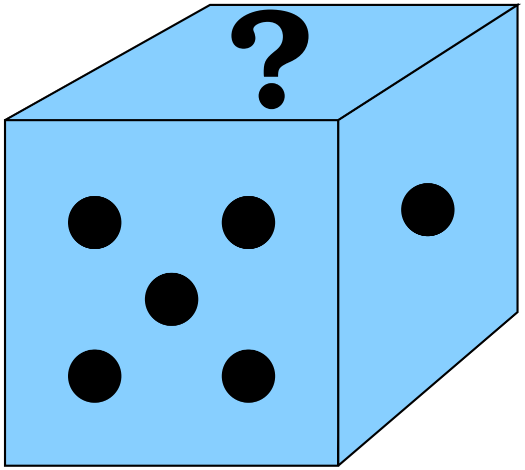
Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Modeling Real World

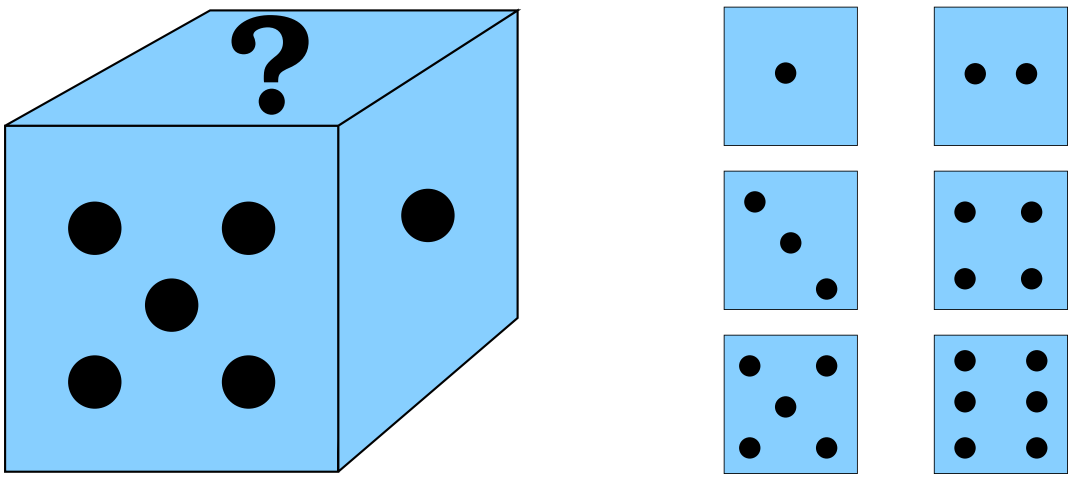
- Knowledge:
  - Uncertainty
  - Probability
  - Fuzziness
  - Incompleteness
  - Distributed knowledge
- Reasoning:
  - Logic

Uncertainty-Probability-Fuzziness



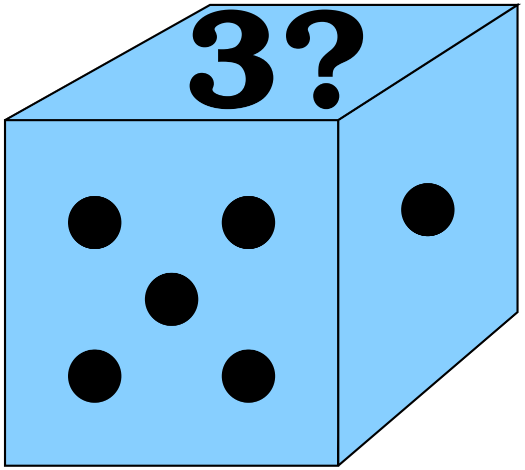
- If we through the cube... which value will appear at the top?

Uncertainty



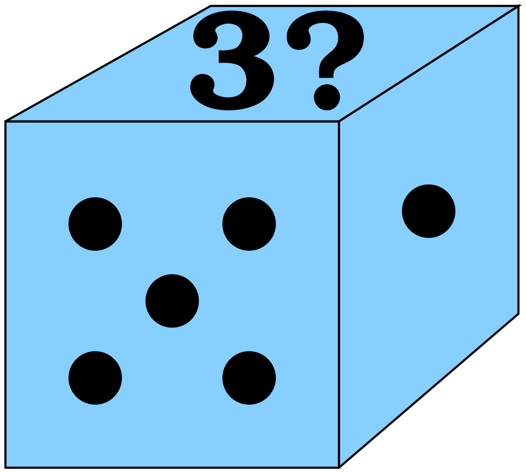
$$X = 1 \vee X = 2 \vee X = 3 \vee X = 4 \vee X = 5 \vee X = 6$$

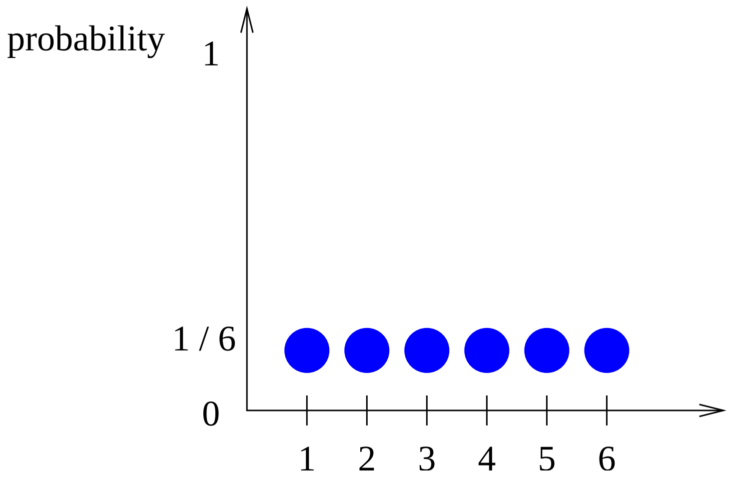
Uncertainty-Probability-Fuzziness



- If we through the cube... is it probable to obtain 3 at the top?

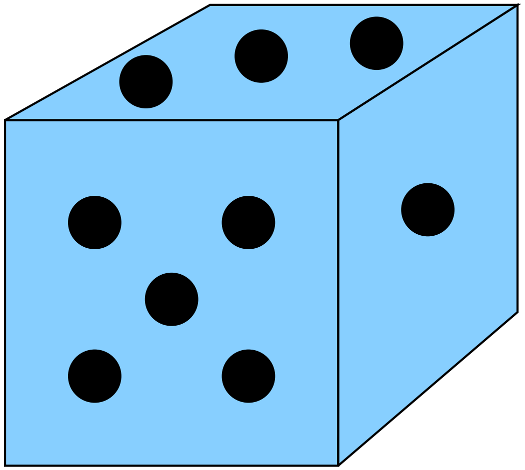
Probability





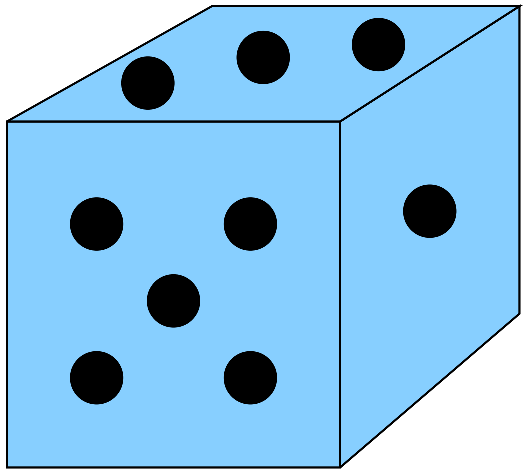
$$Pr(X = 3) = 1/6 = 0.16$$

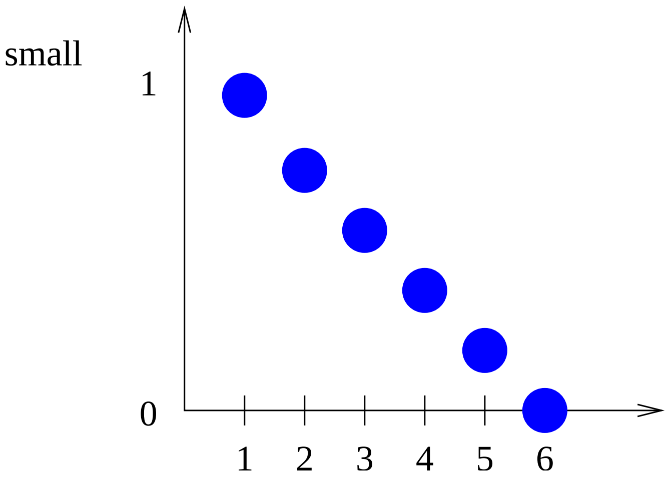
Uncertainty-Probability-Fuzziness



- If we obtain 3 at the top... is it a small value?

Fuzziness





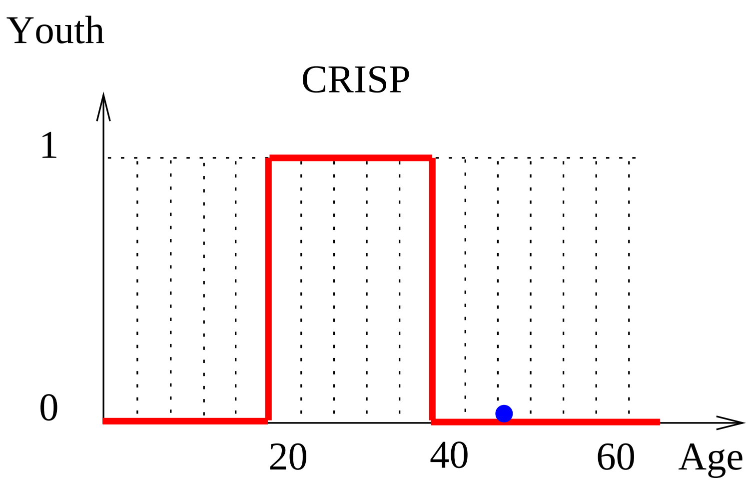
$$small(X = 3) = 0.6$$

Value 3 is slightly small

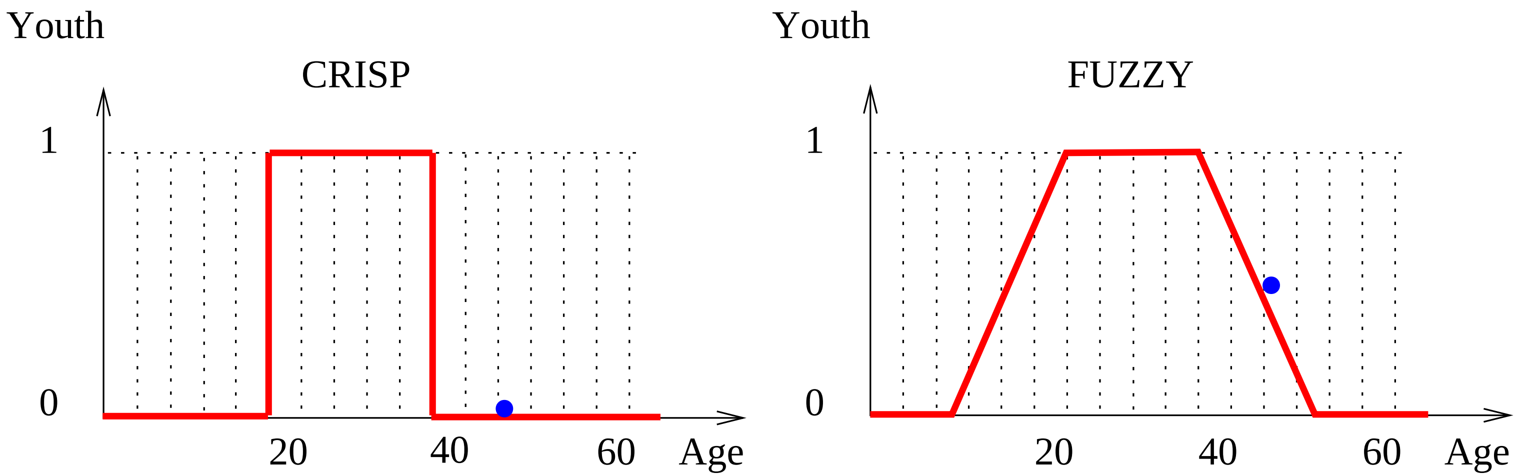
Fuzziness level: Example

Let’s define the concept of youth

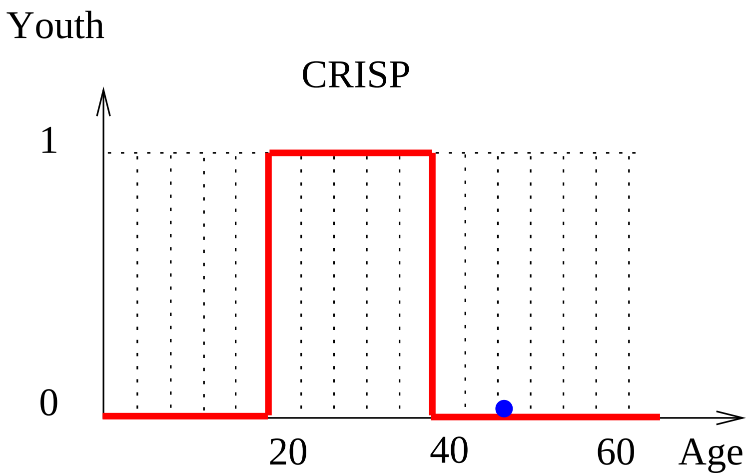
Fuzziness level

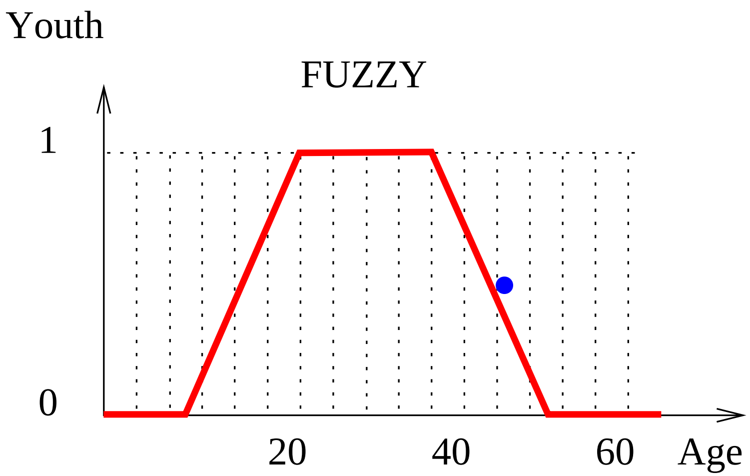


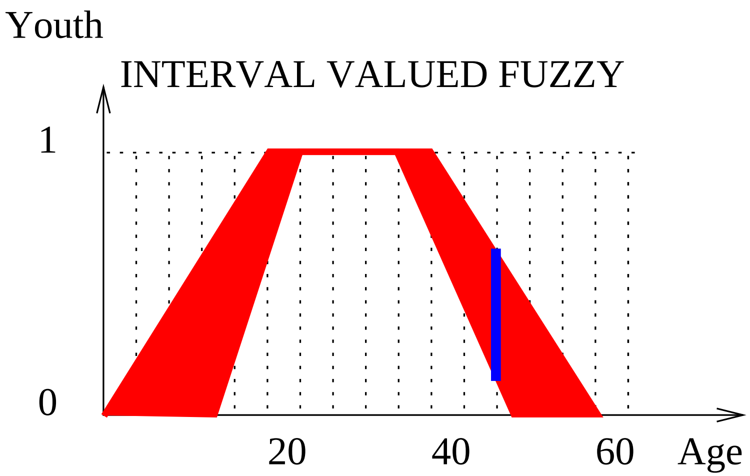
Fuzziness level



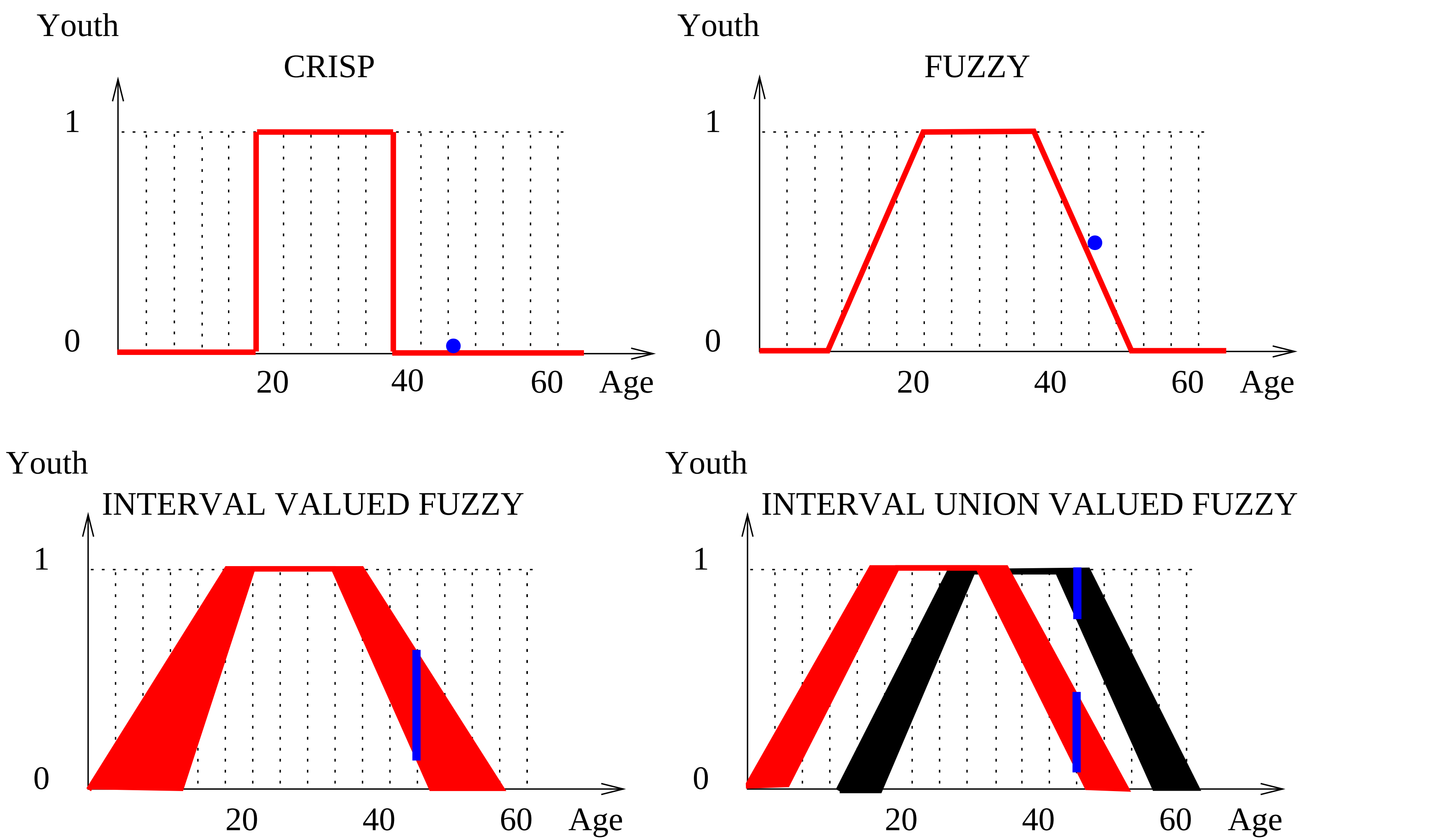
Fuzziness level







Fuzziness level



Truth value (Fuzziness level)

The value of youth of a 42 years-old man

- $V = 0$
- $V = 0.5$
- $V \in [0.2, 0.6]$
- $V \in [0.2, 0.5] \bigcup [0.8, 1]$

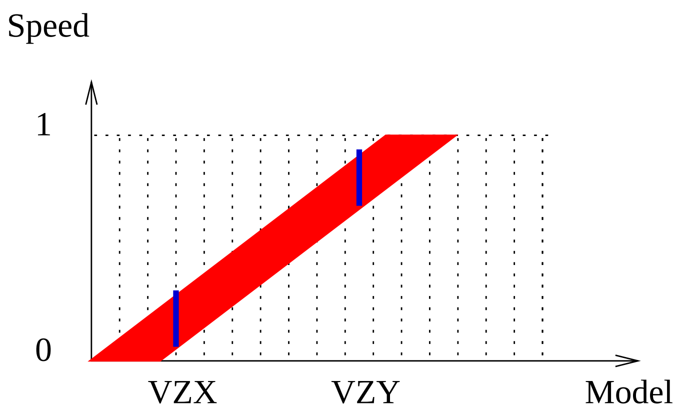
Truth value (Fuzziness level)

The value of youth of a 42 years-old man

- $V = 0$
  $(V = 0)$
- $V = 0.5$
  $(V = 0.5)$
- $V \in [0.2, 0.6]$
  $(0.2 \leq V \wedge V \leq 0.6)$
- $V \in [0.2, 0.5] \bigcup [0.8, 1]$
  $(0.2 \leq V \wedge V \leq 0.5) \vee (0.8 \leq V \wedge V \leq 1)$

Fuzziness - Uncertainty

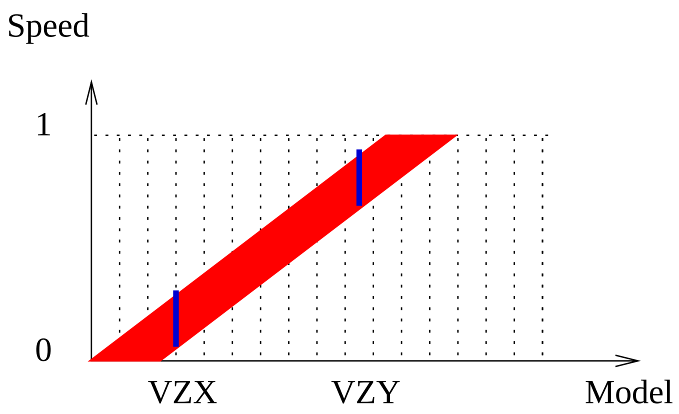
- New Laptop is a branch of computers with two laptop models (VZX and VZY). One model is very slow and the other one is very fast.



  - VZX speed $[0.02, 0.08]$
  - VZY speed $[0.75, 0.90]$

Fuzziness - Uncertainty

- New Laptop is a branch of computers with two laptop models (VZX and VZY). One model is very slow and the other one is very fast.



  - VZX speed $[0.02, 0.08]$
  - VZY speed $[0.75, 0.90]$

- If a client buys a New Laptop computer, the truth value, V, of its speed will be [0.02, 0.08] $\bigcup$ [0.75, 0.90]

$$(0.02 \leq V \wedge V \leq 0.08) \vee (0.75 \leq V \wedge V \leq 0.90)$$

Modeling Real World

- Knowledge:

$$\text{Constraints} \leftarrow \begin{cases} Uncertainty \\ Probability \\ Fuzziness \\ Incompleteness \\ Distributed\ knowledge \end{cases}$$

- Reasoning:

$$\text{Prolog} \quad \leftarrow \quad Logic$$

Fuzzy Prolog

- Existing Fuzzy Prolog systems:
  - Prolog-Elf
  - Fril Prolog
  - f-Prolog
- Our Fuzzy Prolog approach:
  - Truth Value (union of sub-intervals) $\mathcal{B}([0,1])$
  - Aggregation operators (min, max, luka, ...)
  - CLP($\mathcal{R}$) based implementation

Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Syntax

- If A is an atom, $A \leftarrow v$ is a *fuzzy fact*, where v, a truth value, is an element in $\mathcal{B}([0,1])$ expressed as constraints over the domain $[0,1]$.
- Let $A, B_1, \ldots, B_n$ be atoms. A *fuzzy clause* is a clause of the form $A\ v \leftarrow_F B_1\ v_1, \ldots, B_n\ v_n$ where F is an *aggregation operator* of truth values represented as constraints over the domain $[0,1]$. The interval-aggregation induces a union-aggregation.
- A *fuzzy query* is a tuple $v \leftarrow A\ ?$ where A is an atom, and v is a variable (possibly instantiated) that represents a truth value in $\mathcal{B}([0,1])$.

Aggregation Operators

- A function $f : [0,1]^n \longrightarrow [0,1]$ that verifies $f(0,\ldots,0) = 0$, $f(1,\ldots,1) = 1$, and in addition it is monotonic and continuous, then it is called `aggregation operator`

Aggregation Operators

- A function $f : [0, 1]^n \rightarrow [0, 1]$ that verifies $f(0, \ldots, 0) = 0$, $f(1, \ldots, 1) = 1$, and in addition it is monotonic and continuous, then it is called `aggregation operator`
- Given an aggregation $f : [0, 1]^n \rightarrow [0, 1]$ an `interval-aggregation` $F : \mathcal{E}([0, 1])^n \rightarrow \mathcal{E}([0, 1])$ is defined as follows:

$$F([x_1^l, x_1^u], ..., [x_n^l, x_n^u]) = [f(x_1^l, ..., x_n^l), f(x_1^u, ..., x_n^u)]$$

Union Aggregation

- Given an interval-aggregation $F : \mathcal{E}([0,1])^n \rightarrow \mathcal{E}([0,1])$ defined over intervals, a `union-aggregation` $\mathcal{F} : \mathcal{B}([0,1])^n \rightarrow \mathcal{B}([0,1])$ is defined over union of intervals as follows:

$$\mathcal{F}(B_1, \ldots, B_n) = \cup\{F(\mathcal{E}_1, ..., \mathcal{E}_n) \mid \mathcal{E}_i \in B_i\}$$

Interpretation

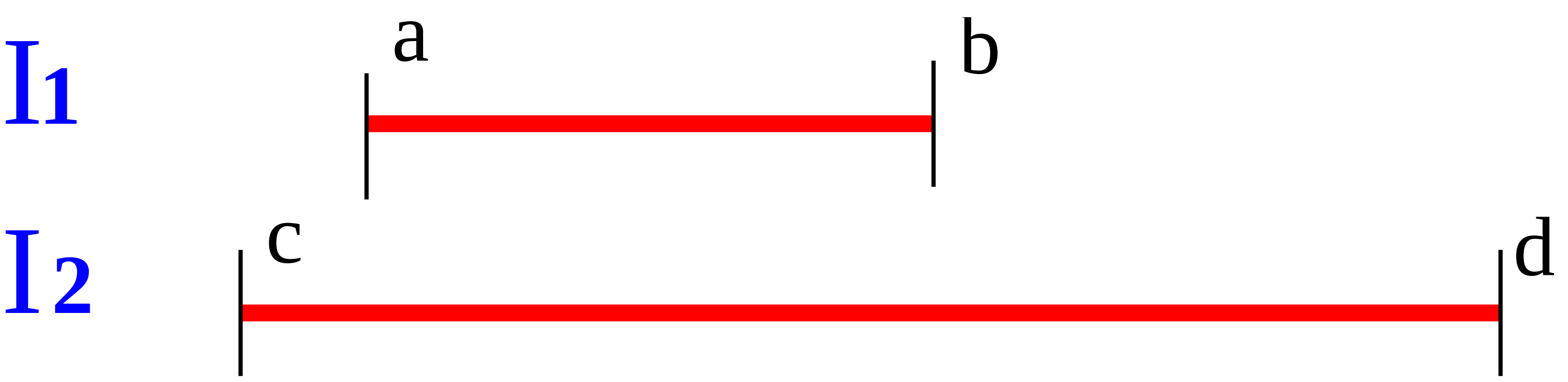
An *interpretation* I consists of the following:

1. a subset B_I of the *Herbrand Base*,
2. a mapping V_I, to assign a truth value, in $\mathcal{B}([0,1])$, to each element of B_I.

The Borel Algebra $\mathcal{B}([0,1])$ is a complete lattice under $\subseteq_{BI}$, that denotes Borel inclusion, and the Herbrand Base is a complete lattice under $\subseteq$, that denotes set inclusion, therefore a set of all *interpretations* forms a complete lattice under the relation $\sqsubseteq$ defined as follows.

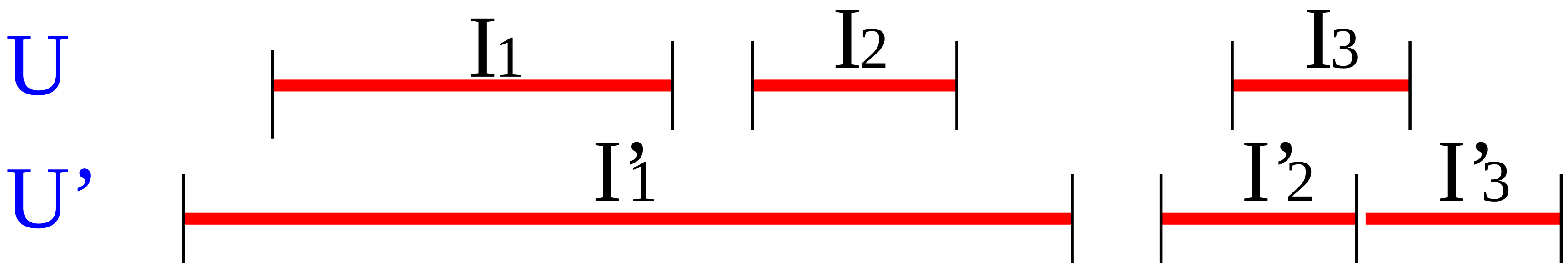
Interval Inclusion

Def:[interval inclusion $\subseteq_{II}$] Given two intervals $I_1 = [a, b]$, $I_2 = [c, d]$ in $\mathcal{E}([0, 1])$, $I_1 \subseteq_{II} I_2$ iff $c \leq a$ and $b \leq d$.



Borel Inclusion

Def:[Borel inclusion $\subseteq_{BI}$] Given two unions of intervals $U = I_1 \cup \cdots \cup I_N$, $U' = I'_1 \cup \cdots \cup I'_M$ in $\mathcal{B}([0,1])$, $U \subseteq_{BI} U'$ if and only if $\forall x \in I_i$, $i \in 1..N$, $\exists I'_j \in U'\ .\ x \in I'_j$ where $j \in 1..M$.



Interpretation Inclusion - Valuation

Def:[interpretation inclusion $\sqsubseteq$] $I \sqsubseteq I'$ iff $B_I \subseteq B_{I'}$ and for all $B \in B_I$, $V_I(B) \subseteq_{BI} V_{I'}(B)$, where $I = \langle B_I, V_I \rangle$, $I' = \langle B_{I'}, V_{I'} \rangle$ are interpretations.

Def:[valuation] A *valuation* σ of an atom A is an assignment of elements of U to variables of A. So $\sigma(A) \in B$ is a ground atom.

Model

Def: [model] Given an *interpretation* $I = \langle B_I, V_I \rangle$

- I is a *model* for a *fuzzy fact* $A \leftarrow v$, if for all valuation σ, $\sigma(A) \in B_I$ and $v \subseteq_{BI} V_I(\sigma(A))$.
- I is a *model* for a *clause* $A \leftarrow_F B_1, \ldots, B_n$ when the following holds:
  for all valuation σ, if $\sigma(B_i) \in B_I$, $1 \leq i \leq n$, and $v = \mathcal{F}(V_I(\sigma(B_1)), \ldots, V_I(\sigma(B_n)))$ then $\sigma(A) \in B_I$ and $v \subseteq_{BI} V_I(\sigma(A))$, where $\mathcal{F}$ is the union aggregation obtained from F.
- I is a *model* of a *fuzzy program*, if it is a *model* for the facts and clauses of the program.

Semantics Equivalence

Given a program P, the three semantics:

1. Least model $lm(P)$, under the $\sqsubseteq$ ordering.
2. Declarative meaning $lfp(T_P)$, least fixpoint for a consequence operator $T_P(I)$.
3. Success set $SS(P)$ of a transitional system.

are equivalent: $SS(P) = lfp(T_P) = lm(P)$.

Operational Semantics

- A sequence of transitions between different states of a system
- State: $\langle Goal,\ Valuation,\ Constraint\rangle$
- Initial State: $\langle A, \emptyset, true\rangle$
- Final State: $\langle \emptyset, \sigma, S\rangle$

Examples:

$\langle p(X,Y), \emptyset, true\rangle$, ... , $\langle \emptyset, \{X = 3, Y = 3\}, true\rangle$

$\langle bachelor(S,M), \emptyset, true\rangle$, ... , $\langle \emptyset, \{S = completed\}, M \geq 5\rangle$

Operational Semantics

A *transition* in the *transition system* is defined as:

1. $\langle A \cup a, \sigma, S \rangle \rightarrow \langle A\theta, \sigma \cdot \theta, S \wedge \mu_a = v \rangle$
   if $h \leftarrow v$ is a fact of the program P, θ is the mgu of a and h, and μ_a is the truth variable for a, and $solvable(S \wedge \mu_a = v)$. ($solvable(c) \equiv c$ has solution in $[0, 1]$ of $\mathcal{R}$)
2. $\langle A \cup a, \sigma, S \rangle \rightarrow \langle (A \cup B)\theta, \sigma \cdot \theta, S \wedge c \rangle$
   if $h \leftarrow_F B$ is a rule of the program P, θ is the mgu of a and h, c is the constraint that represents the truth value obtained applying the union-aggregator F on the truth variables of B, and $solvable(S \wedge c)$.
3. $\langle A \cup a, \sigma, S \rangle \rightarrow fail$ if none of the above are applicable.

Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Fuzzy Programs Syntax

```
tall(john,V):~
   [0.8,0.9].
tall(john,[0.8,0.9]):~.
```

```
good_player(X,V):~min
   tall(X,Vt),
   swift(X,Vs).
```

```
f_digit :#
   fuzzy digit/1.
```

```
not_small :#
   fnot  small/2.
```

Fuzzy → CLP($\mathcal{R}$) Translation

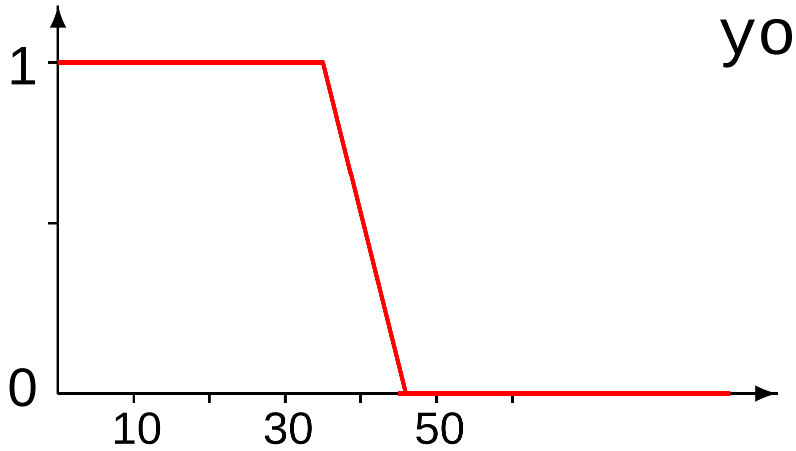
```
good_player(X,V):~min
    tall(X,Vt),
    swift(X,Vs).
```

```
not_small :#
    fnot  small/2.
```

```
good_player(X,V) :-
    tall(X,Vt),
    swift(X,Vs),
    minim([Vt,Vs],V),
    V.>=.0, V.=<.1.
```

```
not_small(X,V) :-
    small(X,Vs),
    V .=. 1 - Vs.
```

Syntactic Sugar




```
young :#
  fuzzy_predicate([(0,1),
                   (35,1),
                   (45,0),
                   (90,0)]).
```

```
young(X,1):-
  X .>=. 0,
  X .<. 35.
```

```
young(X,V):-
  X .>=. 35,
  X .<. 45,
  10*V.=.45-X.
```

```
young(X,0):-
  X .>=. 45,
  X .=<. 120.
```

Initial Evaluation

- Implementation over CLP($\mathcal{R}$): SIMPLICITY
- Aggregation operator: GENERALITY
- Definition of new operators: FLEXIBILITY
- Using Prolog resolution: EFFICIENCY

Available implementation:

http://clip.dia.fi.upm.es/Software/Ciao/

Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Combining Crisp and Fuzzy Logic

```
student(john).
student(peter).

-----------------------------------

age_about_15(john,1):~.
age_about_15(susan,0.7):~.
age_about_15(nick,0):~.

-----------------------------------

teenager_student(X,V):~
        student(X),         % CRISP
        age_about_15(X,Va).% FUZZY
```

```
?- student(john).
yes
?- student(nick).
no            FALSE
?- age_about_15(john,V).
V = 1
?- age_about_15(nick,V).
V = 0
?- age_about_15(peter,V).
no            UNKNOWN
?- teenager_student(john,V).
V .=. 1
?- teenager_student(susan,V).
V .=. 0
?- teenager_student(peter,V).
no            UNKNOWN
```

Solution: Default Knowledge

```
student(john).
student(peter).

:-default(f_student/2,0).

f_student(X,1):-
         student(X).
------------------------------------

:-default(age_about_15/2,[0,1]).

age_about_15(john,1):~.
age_about_15(susan,0.7):~.
age_about_15(nick,0):~.
------------------------------------

:-default(teenager_student/2,[0,1]).

teenager_student(X,V):~
        f_student(X,Vs),
        age_about_15(X,Va).
```

```
?- f_student(john,V).
V = 1
?- f_student(nick,V).

V = 0                FALSE

?- age_about_15(john,V).
V = 1
?- age_about_15(nick,V).
V = 0
?- age_about_15(peter,V).

V .>=. 0, V .<=. 1    UNKNOWN

?- teenager_student(john,V).
V .=. 1
?- teenager_student(susan,V).
V .=. 0
?- teenager_student(peter,V).

V .>=. 0, V .<=. 1    UNKNOWN
```

Default Value

We assume there is a function *default* which implement the Default Knowledge Assumptions. It assigns an element of $\mathcal{B}([0,1])$ to each element of the Herbrand Base.

- If the Closed World Assumption is used, then $default(A) = [0,0]$ for all A in Herbrand Base.
- If Open World Assumption is used instead, $default(A) = [0,1]$ for all A in Herbrand Base.

Interpretation

An *interpretation* I consists of the following:

1. a subset B_I of the *Herbrand Base*,
2. a mapping V_I, to assign
   (a) a truth value, in $\mathcal{B}([0,1])$, to each element of B_I, or
   (b) $default(A)$, if A does not belong to B_I.

Operational Semantics

A *transition* in the *transition system* is defined as:

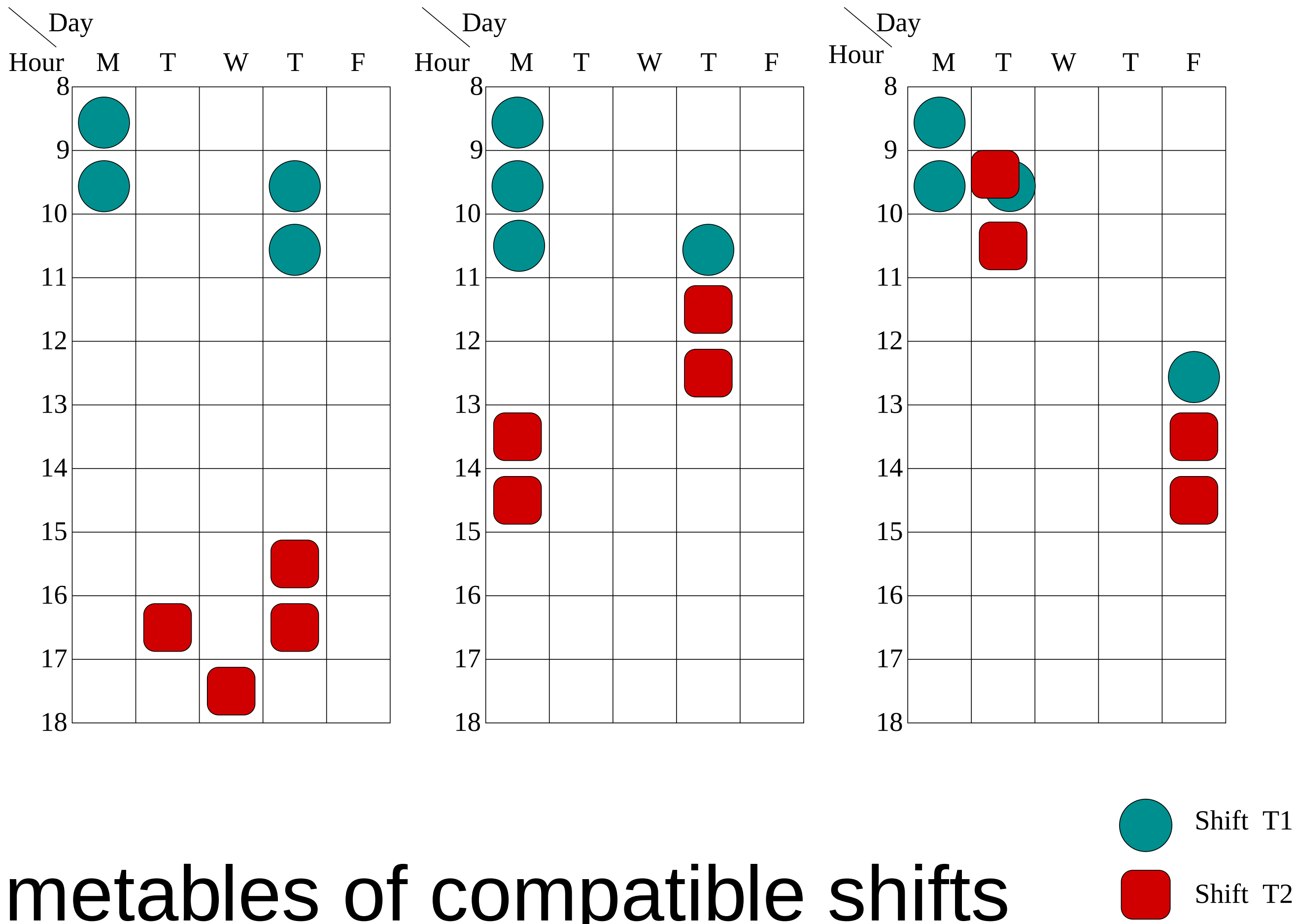
1. $\langle A \cup a, \sigma, S \rangle \rightarrow \langle A\theta, \sigma \cdot \theta, S \wedge \mu_a = v \rangle$
   if $h \leftarrow v$ is a fact of the program P, ...
2. $\langle A \cup a, \sigma, S \rangle \rightarrow \langle (A \cup B)\theta, \sigma \cdot \theta, S \wedge c \rangle$
   if $h \leftarrow_F B$ is a rule of the program P, ...
3. $\langle A \cup a, \sigma, S \rangle \rightarrow fail$ if none of the above are applicable.

Operational Semantics

A *transition* in the *transition system* is defined as:

1. $\langle A \cup a, \sigma, S \rangle \rightarrow \langle A\theta, \sigma \cdot \theta, S \wedge \mu_a = v \rangle$
   if $h \leftarrow v$ is a fact of the program P, ...
2. $\langle A \cup a, \sigma, S \rangle \rightarrow \langle (A \cup B)\theta, \sigma \cdot \theta, S \wedge c \rangle$
   if $h \leftarrow_F B$ is a rule of the program P, ...
3. $\langle A \cup a, \sigma, S \rangle \rightarrow fail$ if none of the above are applicable.
   $\langle A \cup a, \sigma, S \rangle \rightarrow \langle A, \sigma, S \wedge \mu_a = v \rangle$
   if none of the above are applicable and
   $solvable(S \wedge \mu_a = v)$ where $\mu_a = default(a)$.

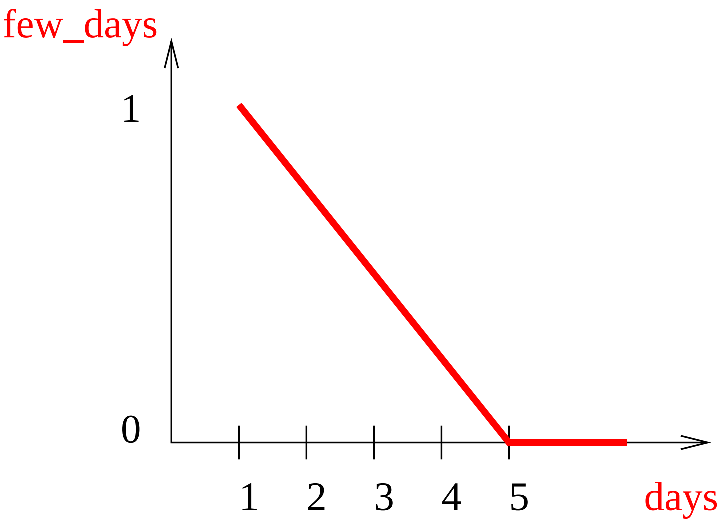
Example (I) - Shifts Compatibility

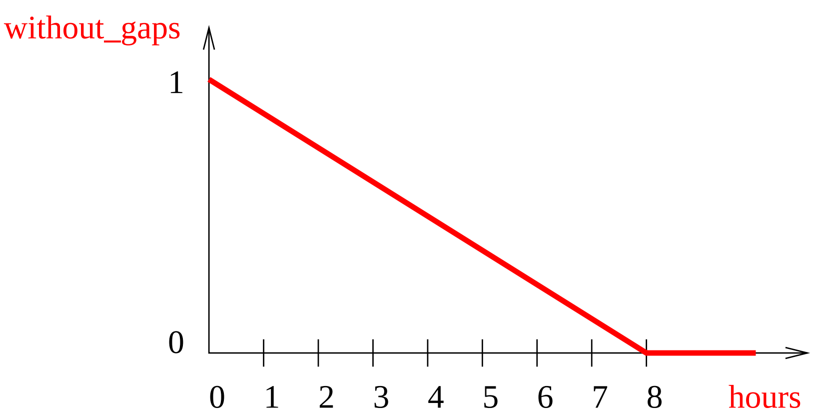


Timetables of compatible shifts

Example (II) - Crisp and Fuzzy

```
compatible(T1,T2,V):~ min
    correct_shift(T1),
    correct_shift(T2),
    disjoint(T1,T2),
    append(T1,T2,T),
    number_of_days(T,D),
    few_days(D,Vf),
    number_of_free_hours(T,H),
    without_gaps(H,Vw).
```



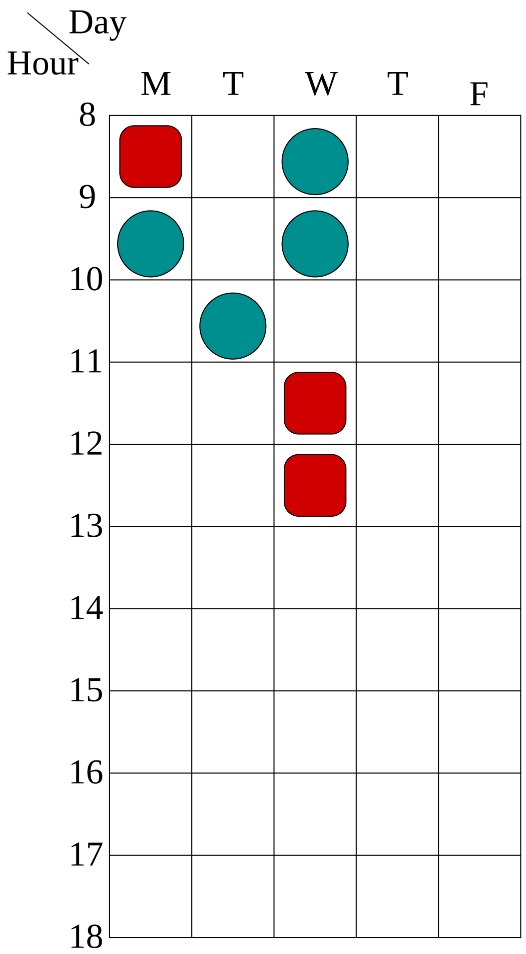


Example (III) - Default Values

```
f_correct_shift(T,1):- correct_shift(T).

:- default(f_correct_shift/2,[0,0]).     % CWA

f_disjoint(T1,T2,1):- disjoint(T1,T2).

:- default(f_disjoint/3,[0,0]).          % CWA

few_days(D,V):- ...

:- default(few_days/2,[0.25,0.75]).      % DEFAULT

without_gaps(H,V):- ...

:- default(without_gaps/2,[0,1]).        % OWA
```

Example (IV) - Constructive Answers

```
?- compatible(
   [(mo,9),(tu,10),(we,8),(we,9)],
   [(mo,8),(we,11),(we,12),(D,H)],
                          V ), V .>. 0.7 .

V = 0.9,  D = we, H = 10 ? ;
V = 0.75, D = mo, H = 10 ? ;
no
```



Evaluation

- Representation of real problems: INCOMPLETENESS
- Crisp + Fuzzy logic: EXPRESIVITY
- $[0, 1]$ to represent total uncertainty ($0 \leq v \wedge v \leq 1$). Lack of information do not stop the evaluation: ACCURACY
- Provides answers: CONSTRUCTIVE

Evaluation and Further Work

- Representation of real problems: INCOMPLETENESS
- Crisp + Fuzzy logic: EXPRESIVITY
- $[0,1]$ to represent total uncertainty ($0 \leq v \wedge v \leq 1$). Lack of information do not stop the evaluation: ACCURACY
- Provides answers: CONSTRUCTIVE

Further Work:

- Constructive negative queries
- Discrete fuzzy sets
- Applications (collaborative agents)

Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Negation in Prolog - As Failure

```
student(john).
student(peter).
```

```
-------------------------------------
POSITIVE QUERIES
-------------------------------------

?- student(john).
yes
?- student(rick).
no

-------------------------------------

?- student(X).
X = john ? ;
X = peter ? ;
no
```

```
-------------------------------------
NEGATIVE QUERIES
-------------------------------------

?- +\ student(john).
no
?- +\ student(rick).
yes

-------------------------------------

?- +\ student(X).
no
```

Negation in Prolog - Constructive

```
student(john).
student(peter).
```

```
------------------------------------
POSITIVE QUERIES
------------------------------------

?- student(john).
yes
?- student(rick).
no

------------------------------------

?- student(X).
X = john ? ;
X = peter ? ;
no
```

```
------------------------------------
NEGATIVE QUERIES
------------------------------------

?- +\ student(john).
no
?- +\ student(rick).
yes

------------------------------------

?- neg(student(X)).
X = rick ?;
X = anne ?;
X = rose ?;
...
```

Negation in Prolog - Constructive

```
student(john).
student(peter).
```

```
----------------------------------------
POSITIVE QUERIES
----------------------------------------

?- student(john).
yes
?- student(rick).
no

----------------------------------------

?- student(X).
X = john ? ;
X = peter ? ;
no
```

```
----------------------------------------
NEGATIVE QUERIES
----------------------------------------

?- +\ student(john).
no
?- +\ student(rick).
yes

----------------------------------------

?- neg(student(X)).
X =/= john, X =/= peter ?;
no
```

Constructive Negation

```
?- neg(member(X,[1,2])).
X =/= 1, X =/= 2 ?;
no
```

```
?- neg((X =/= 0,member(X,[1,2]))).
X = 0 ?;
X =/= 0, X =/= 1, X =/= 2 ?;
no
```

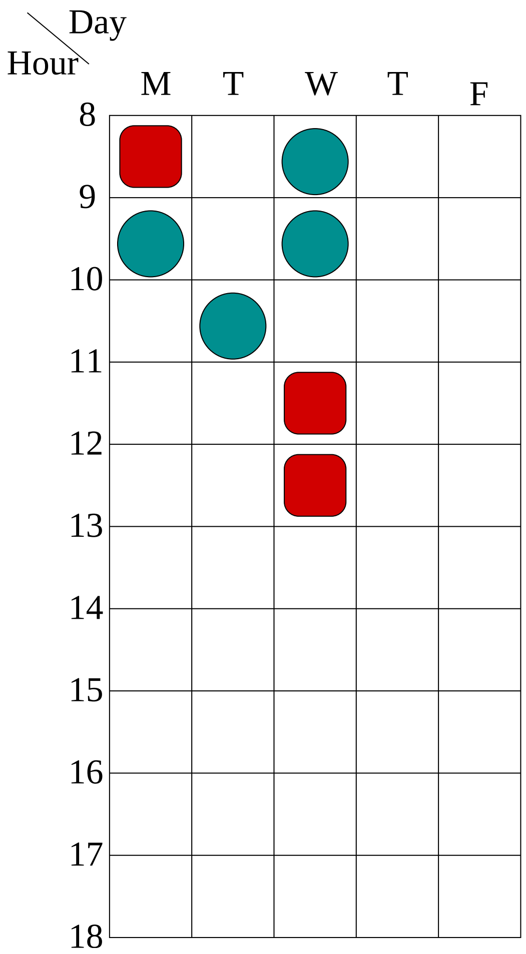
Fuzzified Crisp Predicates

```
student(john).
student(peter).
------------------------------------
f1_student(X,V):-
        student(X),!,
        V .=. 1.
f1_student(X,0).
------------------------------------
f2_student(X,V):-
        student(X),
        V .=. 1.
f2_student(X,V):-
        neg(student(X)),
        V .=. 0.
```

```
?- f1_student(X,1).
X = john ? ;
X = peter ? ;
no
------------------------------------

?- f1_student(X,0).
no

------------------------------------
?- f2_student(X,0).
X =/= john, X =/= peter ? ;
no
```

Example (V) - Constructive Answers

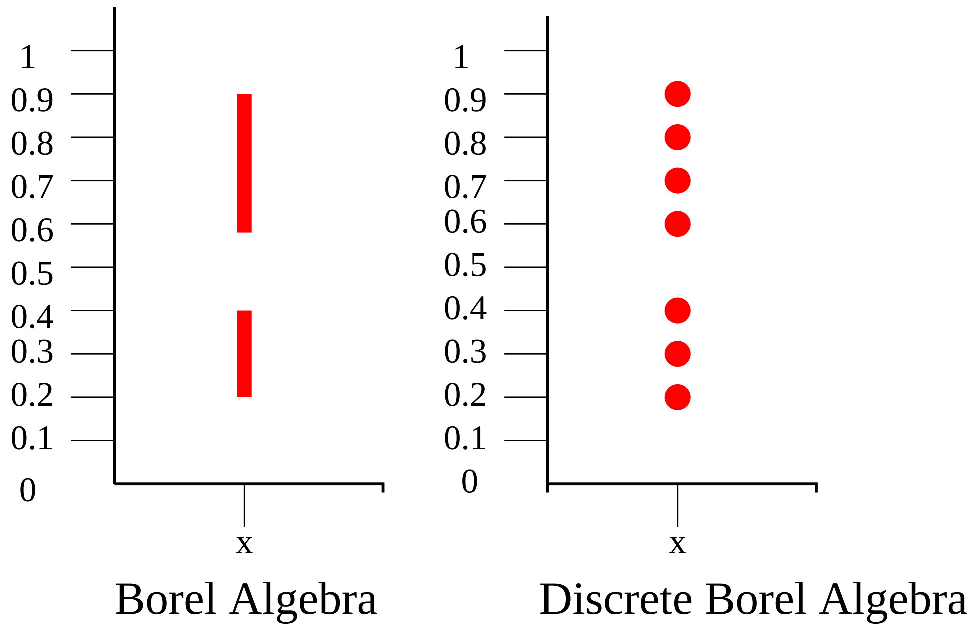
```
?- compatible(
     [(mo,9), (tu,10),
      (we,8), (we,9)],
     [(mo,8), (we,11),
      (we,12), (D,10)], V),
    V .>. 0 .

D =/= tu ? ;
no
```



Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Borel Alg. vs Discrete Borel Alg.



A *discrete-interval* $[X_1, X_N]_d$ is a set of a finite number of values, $\{X_1, X_2, ..., X_{N-1}, X_N\}$, between X_1 and X_N, $0 \leq X_1 \leq X_N \leq 1$, such that $\exists\ 0 < \epsilon < 1.\ X_i = X_{i-1} + \epsilon,\ i \in \{2..N\}$.

Discrete Union Aggregation

- Given a discrete-interval-aggregation $F : \mathcal{E}_d([0,1])^n \rightarrow \mathcal{E}_d([0,1])$ defined over discrete-intervals, a `discrete-union-aggregation` $\mathcal{F} : \mathcal{B}_d([0,1])^n \rightarrow \mathcal{B}_d([0,1])$ is defined over union of discrete-intervals as follows:

$$\mathcal{F}(B_1, \ldots, B_n) = \cup\{F(\mathcal{E}_{d,1}, ..., \mathcal{E}_{d,n}) \mid \mathcal{E}_{d,i} \in B_i\}$$

Introduction to CLP($\mathcal{FD}$)

- CLP($\mathcal{FD}$): (Arithmetical) constraints over Finite Domains $\mathcal{FD}$: Each variable ranges over a finite set of integers
- *Resolution* is a combination of:
  - Propagation: excludes problem inconsistent values from the range of the variables (deterministic)
  - Labeling: assigns values to variables (expensive search process which fires more propagation)

Introduction to CLP($\mathcal{FD}$)

- Example:

```
main(X,Y,Z) :-
      [X,Y,Z] in 1..5,
      X-Y .=.  2*Z,
      X+Y .≥. Z,
      labeling([X,Y,Z]).
```

Fuzzy $\rightarrow$ CLP($\mathcal{FD}$) Translation

```
youth(45,V) ::~
  [0.2,0.5] v [0.8,1]
```

```
good_player(X,V)::~min
  tall(X,Vt),
  swift(X,Vs).
```

```
youth(45,V) :-
  V in 2..5,
  V in 8..10.
```

```
good_player(X,V) :-
  tall(X,Vt),
  swift(X,Vs),
  minim([Vt,Vs],V),
  V in 0..100.
```

Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Constraint Satisfaction Problems (CSP)

CANDIDATES

0.55
0.15
0.32
0
1
0.89
...
0.31

GOAL
V= ?

CONSTRAINTS

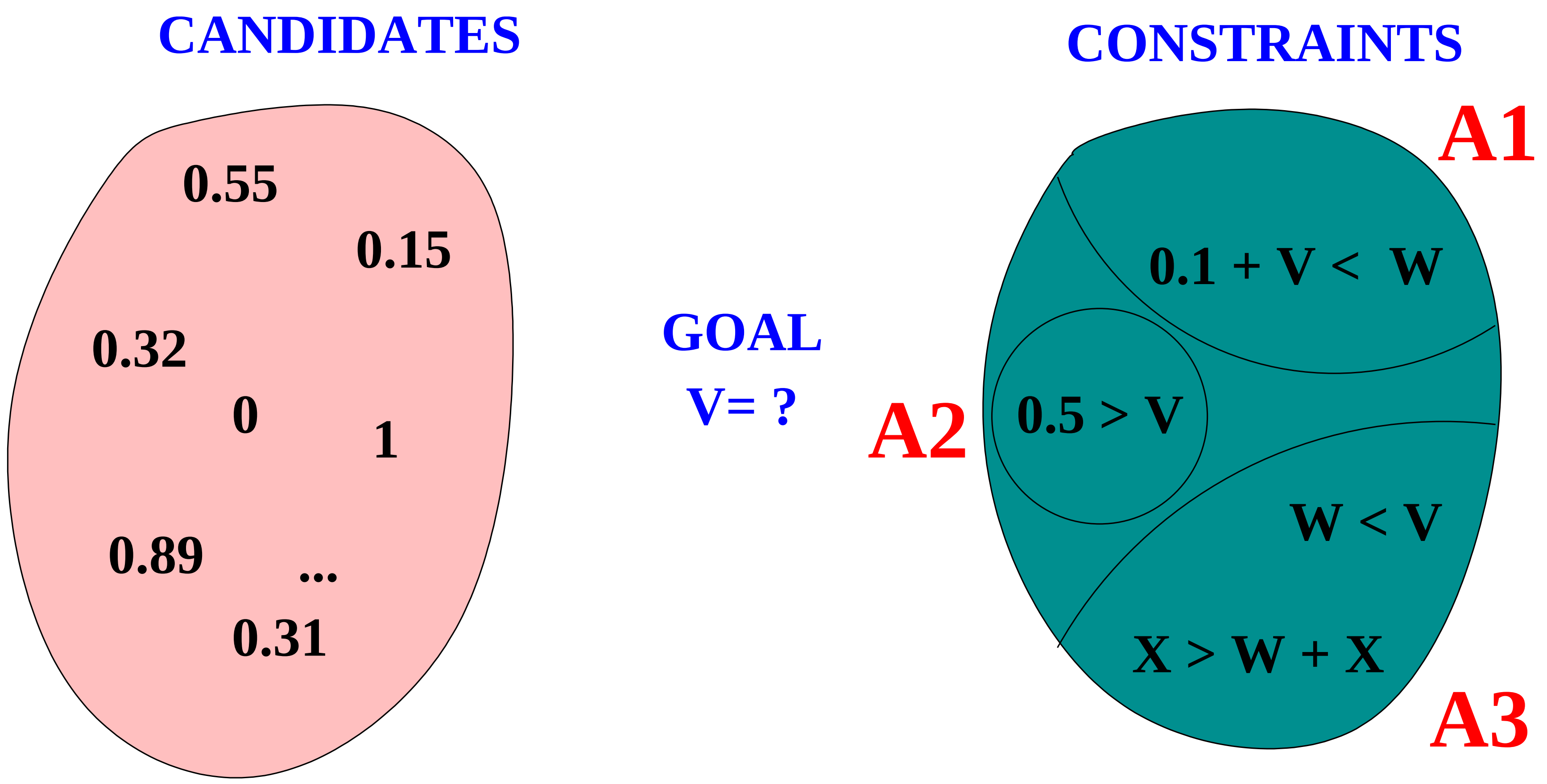
$0.1 + V < W$

$0.5 > V$

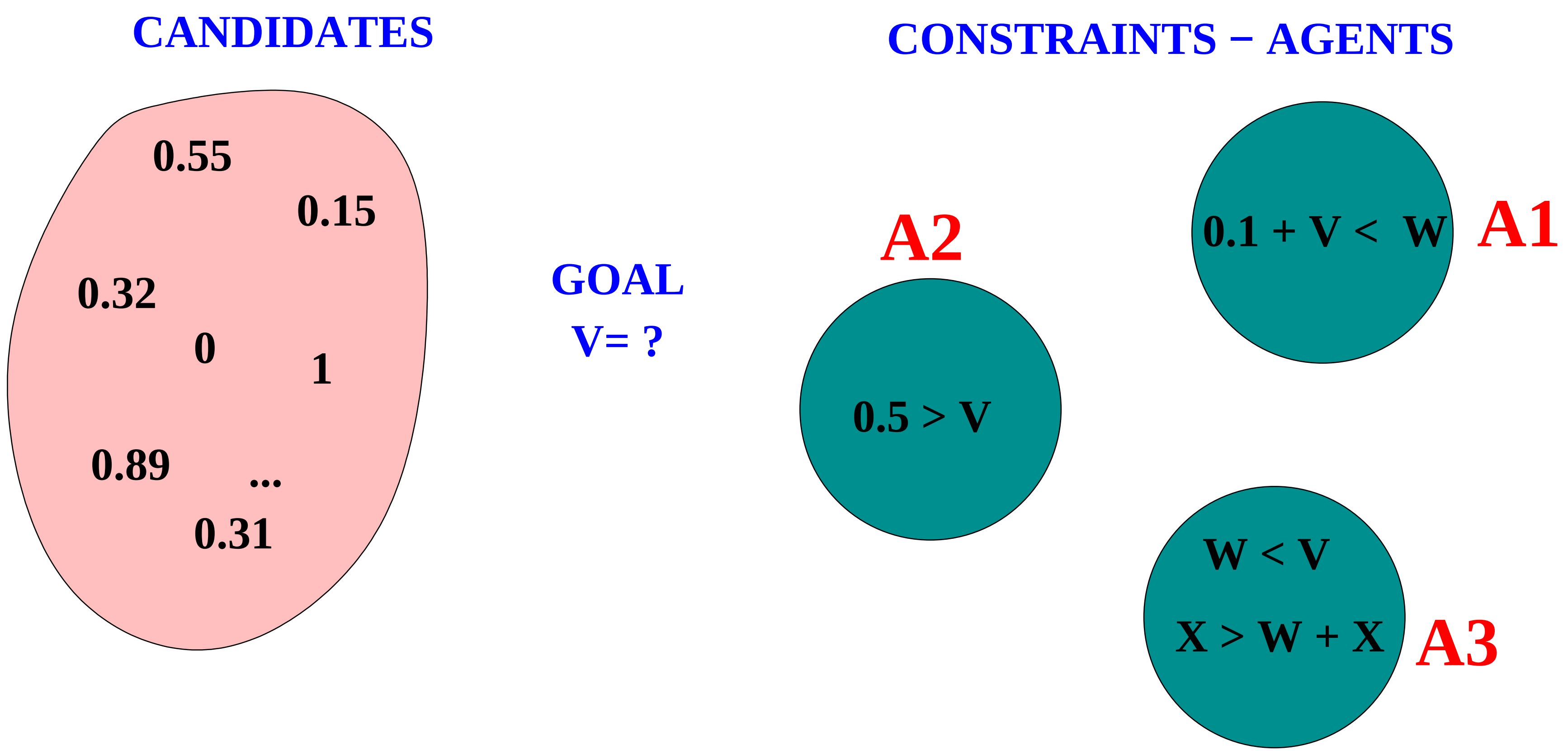
$W < V$

$X > W + X$

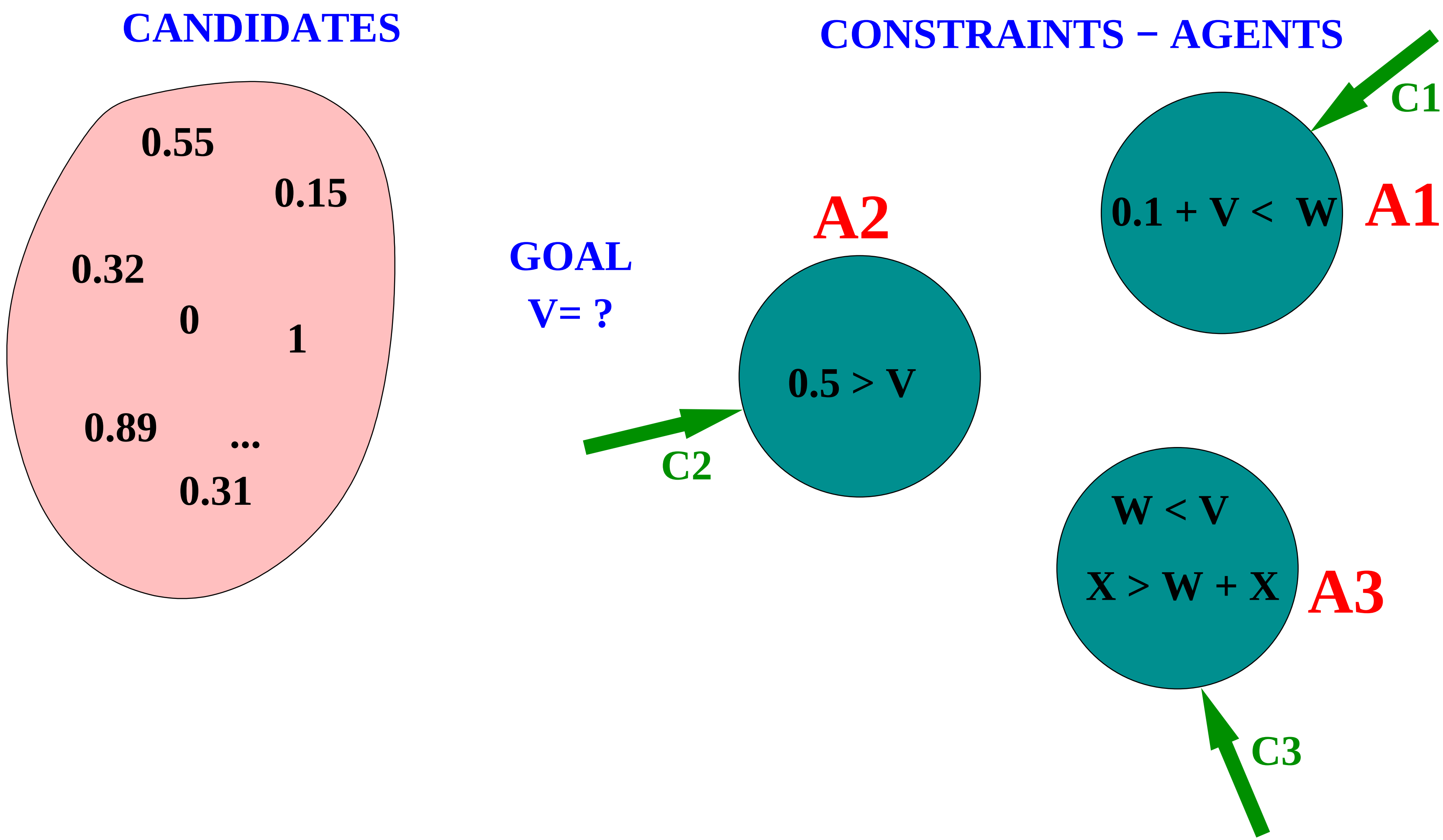
Distributed CSP



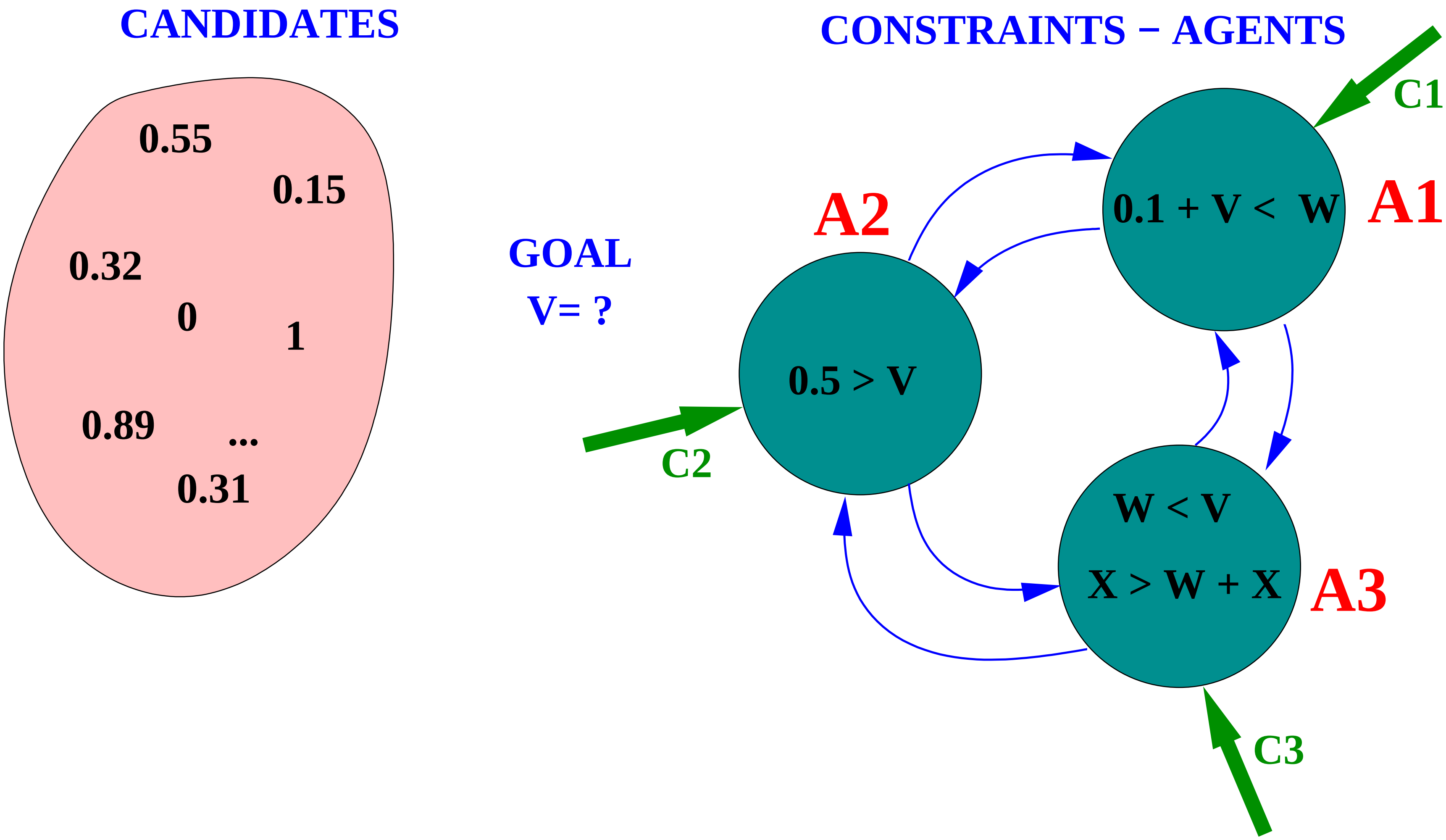
Distributed CSP (I)



Distributed CSP (II)



Distributed CSP (III)



Asynchronous Backtracking Algorithm

- Each agent owns exactly one variable and asynchronously assigns a value to its variable and sends it to the other agents (for evaluation)
- Each agent has a partial knowledge of the problem determined by the agents connected to it: *agent view*
- Messages exchanged:
  - *ok?* : assignment made by the agent
  - *nogood* : *agent view* which detects an inconsistency
  - *ack* : acknowledgement ($\equiv$ consistency)

Extended ABT (I)

- CLP($\mathcal{FD}$) resolution simplifies having multiple variables in each agent
- Coordination between distributed propagation and labeling
- We use the Chandy-Lamport algorithm for detecting when an agent is stable if:
  - It has no queued message
  - Its *agent view* is complete (there are no messages in transit)

Extended ABT (II)

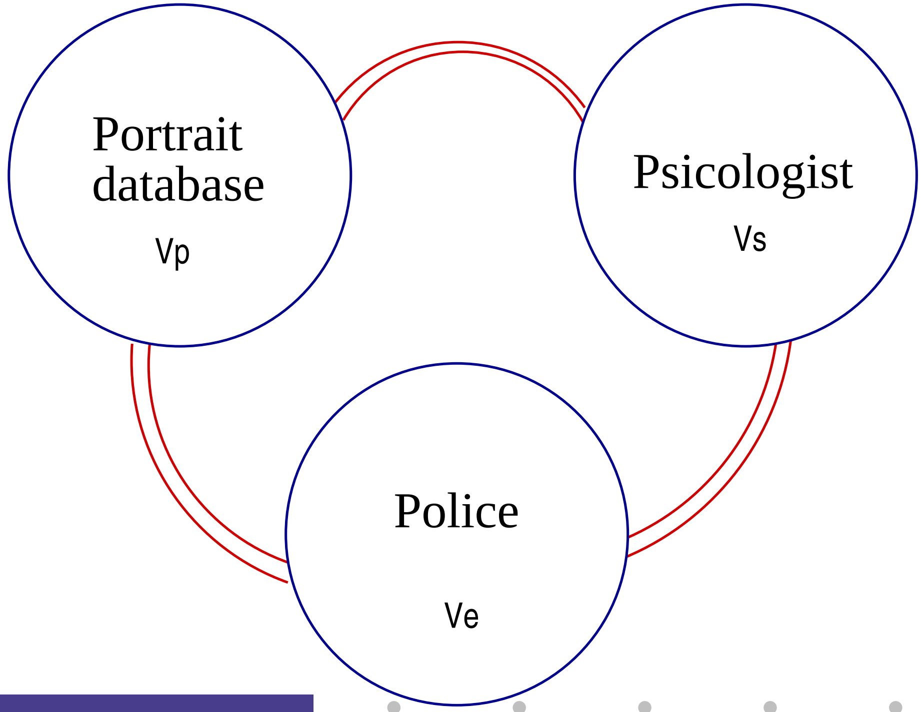
- Distributed propagation ends when termination is detected (all agents are in a stable state) $\longrightarrow$ obtains a global fixpoint without inconsistent values
- We use Dijkstra-Scholten algorithm that provide a reduction of the search space $\rightarrow$ minimal exchange of propagation messages: minimal spanning tree

Collaborative Fuzzy Problems

- Collaborative Fuzzy Problems can be modelled using a combination of:
  - Discrete Fuzzy Prolog
  - An implementation of Extended ABT (for distributed reasoning)
- CLP($\mathcal{FD}$) is the link between these components
- This work has been implemented in Ciao Prolog

Example (I)

- Criminal identification of suspects
- Distributed knowledge about:
  - physical aspects (Vp)
  - psychical aspects (Vs)
  - evidences (Ve)



Example (II)

- Discrete fuzzy program:

```
suspect(Person, V) ::~ inter_m
          allocate_vars([Vp, Vs, Ve]),
          physically_suspect(Person, Vp, Vs),
          psychically_suspect(Person, Vs, Vp),
          evidences(Person, Ve, Vp, Vs).
```

- Transformed CLP($\mathcal{FD}$) program:

```
suspect(Person, V) :-
          allocate_vars([Vp, Vs, Ve]),
          V in 0..10,
          physically_suspect(Person, Vp, Vs),
          psychically_suspect(Person, Vs, Vp),
          evidences(Person, Ve, Vp, Vs),
          inter_m([Vp, Vs, Ve], V).
```

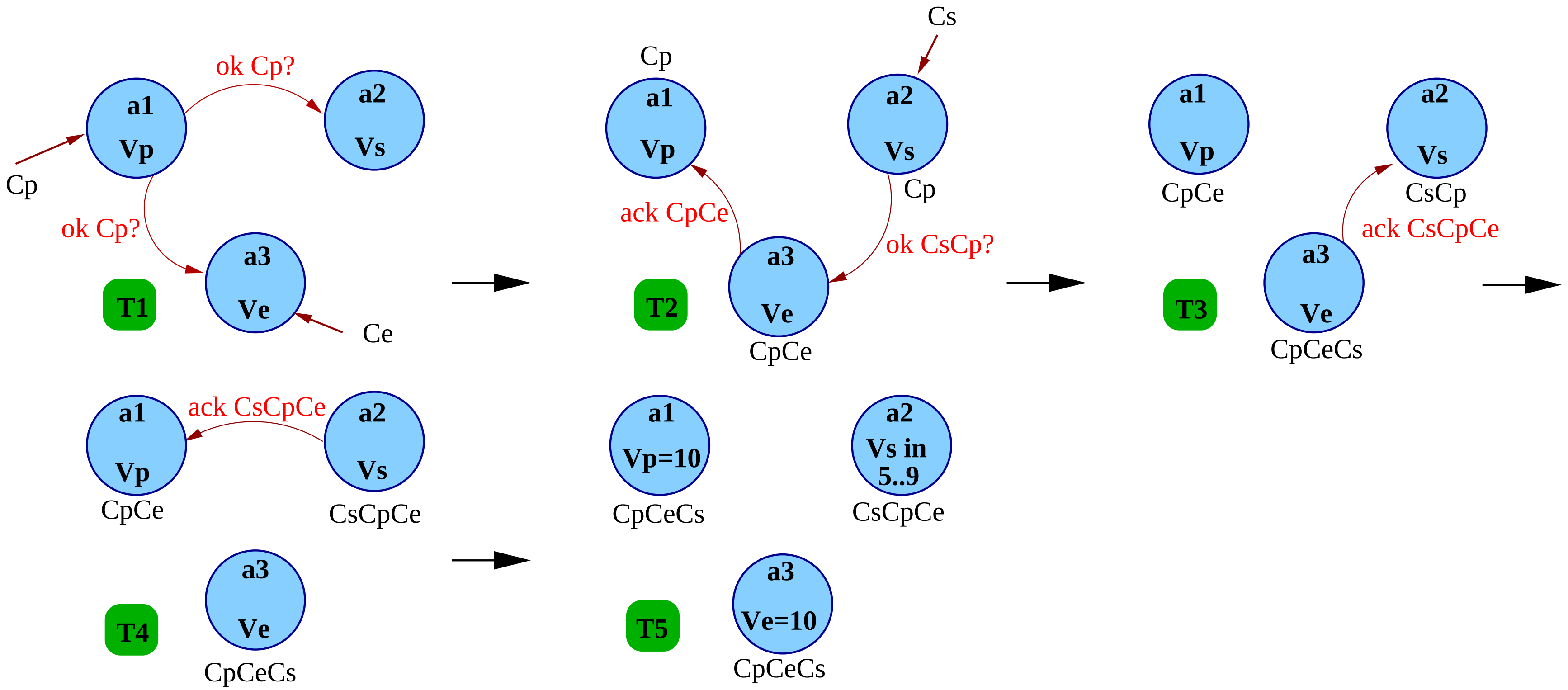
Distributed Knowledge

- Partial knowledge stored in each agent is formulated in terms of constraint expressions

```
Cp:   physically_suspect(Person, Vp, Vs) :-
              scan_portrait_database(Person, Vp),
              Vp * Vs .>=. 50 @ a1.
Cs:   psychically_suspect(Person, Vs, Vp) :-
              psicologist_diagnostic(Person, Vs),
              Vs .<. Vp @ a2.
Ce:   evidences(Person, Ve, Vp, Vs) :-
              police_database(Person, Ve),
              (Ve .>=. Vp,
               Ve .>=. Vs) @ a3.

scan_portrait_database(peter, Vp) :- Vp in 4..10.
psicologist_diagnostic(peter, Vs) :- Vs in 3..10.
police_database(peter, Ve) :- Ve in 7..10.
```

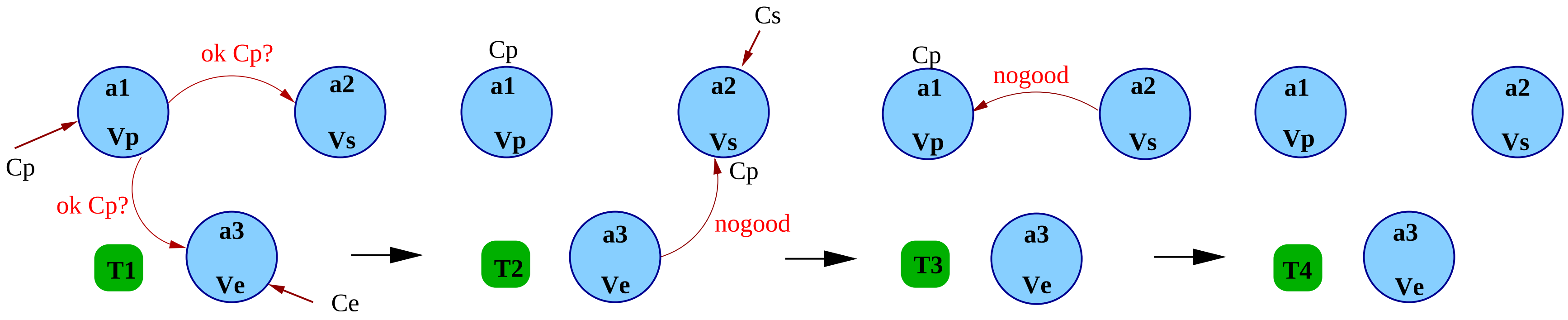
Agent Interaction (I)

- Collaborative fuzzy agents interaction for `suspect(peter,V)`:



Agent Interaction (II)

- Collaborative fuzzy agents interaction for `suspect(jane,V)`:



Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Multi-adjoint logic

- Rules with a truth degree of credibility

$$< \mathcal{R}; \alpha >$$

- Fuzzy Prolog rule syntax

$fpred(args)$ **cred** $(aggrC, \alpha)$ $:\sim$ $aggrO$

$$fpred1(args1),\ \ldots,\ fpredN(argsN).$$

- Example

```
good_player(J) cred (prod, 0.8) :~ prod
           swift(J), tall(J), experience(J).
```

Multi-adjoint logic

- Fuzzy Prolog fact syntax

$$fpred(args)\ \mathbf{value}\ truth\_value.$$

- Example

```
experience(john) value 0.9 .
experience(karl) value 0.9 .
experience(mike) value 0.9 .
experience(lebron) value 0.4 .
experience(deron) value 0.3 .
```

Default values

- Represent incomplete information
- Fuzzy Prolog default value syntax

$$:- \mathbf{default}\ (fpred/arity,\ default\_value).$$

- Example

```
:- default (experience/1, 0.9).
experience(lebron) value 0.4 .
experience(deron) value 0.3 .
```

Type Properties

- Fuzzy Prolog type properties syntax

$$:- \textbf{prop}\ type\_name/arity.$$

$$:- \textbf{set\_prop}\ fpred(args) => type\_name(args).$$

- Example

```
:- prop typePlayer/1.
typePlayer(john).
    ...
typePlayer(deron).
:- set_prop experience(J) => typePlayer(J).
```

Complete Example I

```
:- module(good_player,_,[rfuzzy]).
:- prop typePlayer/1.

(good_player(J) cred (prop,0.8)) :~ prop
          swift(J), tall(J), experience(J).

typePlayer(john).
     ...
typePlayer(deron).

:- set_prop experience(J) >= typePlayer(J).
```

Complete Example II

```
:- default (experience/1, 0.9).
experience(lebron) value 0.4 .
experience(deron) value 0.3 .

:- default (tall/1, 0.6).
tall(john) value 0.4 .
tall(karl) value 0.8 .

:- default (swift/1, 0.7).
swift(john) value 1 .
```

Fuzzy Queries

- Fuzzy Prolog queries syntax

$$?-\ fpred(args, V).$$

- Example

```
?- good_player(john,V).

V= 0.288 ?;

no
```

Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Work proposals

- “Models of Inexact Reasoning” work
  - Modeling a Problem with Fuzzy/RFuzzy Prolog
  - Semantics for Fuzzy queries language
  - ...
- Practical Project / Master thesis
  - Implementation of credibility with intervalsl
  - Fuzzy web interface
  - New Fuzzy Prolog version with credibility
  - Comparison with other Fuzzy Prolog (tool & semantics)
  - Credibility with intervals
  - ...

Work “Models of Inexact Reasoning”

- Choose a topic (with the acknowledge of the professor)
- Send **Feb 20th** by e-mail to $susana@fi.upm.es$
  - Source files of the report (better .tex, otherwise .doc)
  - Report with tests, explanations, etc. (.pdf)
  - Source files of the developed code (.pl)
  - Examples files (.pl)

Overview

- Basics
  - Introduction
  - Description
  - Implementation
- Extensions
  - Incompleteness
  - Constructive negative Queries
  - Discrete Fuzzy Sets
  - Collaborative Fuzzy Agents
  - Fuzzy Rules with Credibility: RFuzzy
- Work proposals

Fuzzy Prolog

Susana Muñoz-Hernández

Facultad de Informática

Universidad Politécnica de Madrid

28660 Madrid, Spain

susana@fi.upm.es