Implementing a Fuzzy Prolog

Moritz Knapp

Seminar für Computerlinguistik
Motivation

- Prolog - a Popular AI Language
- The Problem of Fuzzy Data

A Solution - Basing Prolog on Fuzzy Logic

- By Defining New Predicates
- By Building Fuzziness into the Interpreter

The Ciao Prolog System

- A Prolog Programming Environment
- Ciao’s fuzzy package
- Example Program
1. Motivation
   - Prolog - a Popular AI Language
   - The Problem of Fuzzy Data

2. A Solution - Basing Prolog on Fuzzy Logic
   - By Defining New Predicates
   - By Building Fuzziness into the Interpreter

3. The Ciao Prolog System
   - A Prolog Programming Environment
   - Ciao’s fuzzy package
   - Example Program
Standard Prolog (based on two-valued logic) is widely used in Artificial Intelligence research.

- natural language processing (i.e. parsing)
- database systems
- expert ('knowledge-based') systems
  - medicine
  - finances
  - law
  - ...

Moritz Knapp
Implementing a Fuzzy Prolog
Standard Prolog (based on two-valued logic) is widely used in Artificial Intelligence research.

- natural language processing (i.e. parsing)
- database systems
- expert ('knowledge-based') systems
  - medicine
  - finances
  - law
  - ...
Prolog in Artificial Intelligence

Standard Prolog (based on two-valued logic) is widely used in Artificial Intelligence research.

- natural language processing (i.e. parsing)
- database systems
- expert ('knowledge-based') systems
  - medicine
  - finances
  - law
  - ...

Moritz Knapp
Implementing a Fuzzy Prolog
Prolog in Artificial Intelligence

Standard Prolog (based on two-valued logic) is widely used in Artificial Intelligence research.

- natural language processing (i.e. parsing)
- database systems
- expert (‘knowledge-based’) systems
  - medicine
  - finances
  - law
  - ...

Moritz Knapp
Implementing a Fuzzy Prolog
Outline

1 Motivation
   - Prolog - a Popular AI Language
   - The Problem of Fuzzy Data

2 A Solution - Basing Prolog on Fuzzy Logic
   - By Defining New Predicates
   - By Building Fuzziness into the Interpreter

3 The Ciao Prolog System
   - A Prolog Programming Environment
   - Ciao’s fuzzy package
   - Example Program
Problem: Sometimes, the data given is not 'crisp':

- *John is 35. Is John young?*
- We want something other than yes or no (like *fairly* young)
- 'John is young' has a **degree of truth**!

How can we handle this?
Problem: Sometimes, the data given is not ‘crisp’:

- *John is 35. Is John young?*
- We want something other than yes or no (like *fairly* young)
- ‘*John is young*’ has a **degree of truth**!

How can we handle this?
Problem: Sometimes, the data given is not ‘crisp’:

- John is 35. Is John young?
- We want something other than yes or no (like fairly young)
- ’John is young’ has a degree of truth!

How can we handle this?
Prolog in Artificial Intelligence

Problem: Sometimes, the data given is not 'crisp':

- *John is 35. Is John young?*
- We want something other than yes or no (like *fairly* young)
- *'John is young'* has a **degree of truth**!

How can we handle this?
Outline

1 Motivation
   - Prolog - a Popular AI Language
   - The Problem of Fuzzy Data

2 A Solution - Basing Prolog on Fuzzy Logic
   - By Defining New Predicates
   - By Building Fuzziness into the Interpreter

3 The Ciao Prolog System
   - A Prolog Programming Environment
   - Ciao’s fuzzy package
   - Example Program
A statement is **true to some degree**. The degree of truth marks the degree of membership in the set of true statements.

- fuzzy facts are associated with truth values in the real interval $[0, 1]$
- fuzzy rules have truth values determined by the combination of its goals’ truth values
- common combinations of truth values:
  - minimum (conjunction)
  - maximum (disjunction)
  - complement (negation)
Degrees of Truth

A statement is **true to some degree**. The degree of truth marks the degree of membership in the set of true statements.

- fuzzy facts are associated with truth values in the real interval $[0, 1]$
- fuzzy rules have truth values determined by the combination of its goals’ truth values
- common combinations of truth values:
  - minimum (conjunction)
  - maximum (disjunction)
  - complement (negation)
Degrees of Truth

A statement is **true to some degree**. The degree of truth marks the degree of membership in the set of true statements.

- fuzzy facts are associated with truth values in the real interval $[0, 1]$
- fuzzy rules have truth values determined by the combination of its goals’ truth values

common combinations of truth values:
- minimum (conjunction)
- maximum (disjunction)
- complement (negation)
A statement is true to some degree. The degree of truth marks the degree of membership in the set of true statements.

- fuzzy facts are associated with truth values in the real interval $[0, 1]$
- fuzzy rules have truth values determined by the combination of its goals’ truth values
- common combinations of truth values:
  - minimum (conjunction)
  - maximum (disjunction)
  - complement (negation)
Outline

1 Motivation
   - Prolog - a Popular AI Language
   - The Problem of Fuzzy Data

2 A Solution - Basing Prolog on Fuzzy Logic
   - By Defining New Predicates
   - By Building Fuzziness into the Interpreter

3 The Ciao Prolog System
   - A Prolog Programming Environment
   - Ciao’s fuzzy package
   - Example Program
Example

Our data:
- average temperatures of locations
- average number of hours of sunshine at location

Query:
- How "hot and sunny" is location X?

`hot_and_sunny_area(X, CHI):-`  
  `av_temp(X, T), hot(T, CHI1),`  
  `av_sun(X, S), sunny(S, CHI2),`  
  `combine(CHI1, CHI2, CHI).`
Our data:
- average temperatures of locations
- average number of hours of sunshine at location

Query:
- How "hot and sunny" is location X?

```
hot_and_sunny_area(X, CHI) :-
    av_temp(X, T), hot(T, CHI1),
    av_sun(X, S), sunny(S, CHI2),
    combine(CHI1, CHI2, CHI).
```
Example

Our data:
  - average temperatures of locations
  - average number of hours of sunshine at location

Query:
  - How ”hot and sunny” is location X?

```prolog
hot_and_sunny_area(X, CHI) :-
  av_temp(X, T), hot(T, CHI1),
  av_sun(X, S), sunny(S, CHI2),
  combine(CHI1, CHI2, CHI).
```
Our data:

- average temperatures of locations
- average number of hours of sunshine at location

Query:

- How "hot and sunny" is location X?

```prolog
hot_and_sunny_area(X, CHI) :-
    av_temp(X, T), hot(T, CHI1),
    av_sun(X, S), sunny(S, CHI2),
    combine(CHI1, CHI2, CHI).
```
Additional Arguments for Truth Values

Our data:
- average temperatures of locations
- average number of hours of sunshine at location

Query:
- How ”hot and sunny” is location X?

```
hot_and_sunny_area(X, CHI) :-
    av_temp(X, T), hot(T, CHI1),
    av_sun(X, S), sunny(S, CHI2),
    combine(CHI1, CHI2, CHI).
```
Additional Arguments for Truth Values

Our data:
- average temperatures of locations
- average number of hours of sunshine at location

Query:
- How "hot and sunny" is location X?

hot_and_sunny_area(X, CHI):-
    av_temp(X, T), hot(T, CHI1),
    av_sun(X, S), sunny(S, CHI2),
    combine(CHI1, CHI2, CHI).
Our data:
- average temperatures of locations
- average number of hours of sunshine at location

Query:
- How "hot and sunny" is location X?

```prolog
hot_and_sunny_area(X, CHI) :-
    av_temp(X, T), hot(T, CHI1),
    av_sun(X, S), sunny(S, CHI2),
    combine(CHI1, CHI2, CHI).
```
Outline

1 Motivation
   - Prolog - a Popular AI Language
   - The Problem of Fuzzy Data

2 A Solution - Basing Prolog on Fuzzy Logic
   - By Defining New Predicates
   - By Building Fuzziness into the Interpreter

3 The Ciao Prolog System
   - A Prolog Programming Environment
   - Ciao’s fuzzy package
   - Example Program
Handling Fuzziness Automatically

- The Prolog interpreter is reimplemented to handle fuzziness.

  Reduced notation:

  ```prolog
  hot_and_sunny_area(X):-
    av_temp(X, T), hot(T),
    av_sun(X, S), sunny(S).
  ```

  - more natural, cleaner
  - more readable
  - behaves like standard Prolog, if there is no fuzziness
Handling Fuzziness Automatically

- The Prolog interpreter is reimplemented to handle fuzziness
- Reduced notation:

```prolog
hot_and_sunny_area(X):-
    av_temp(X, T), hot(T),
    av_sun(X, S), sunny(S).
```

- more natural, cleaner
- more readable
- behaves like standard Prolog, if there is no fuzziness
The Prolog interpreter is reimplemented to handle fuzziness

Reduced notation:

```
hot_and_sunny_area(X):-
  av_temp(X, T), hot(T),
  av_sun(X, S), sunny(S).
```

- more natural, cleaner
- more readable
- behaves like standard Prolog, if there is no fuzziness
Outline

1 Motivation
   - Prolog - a Popular AI Language
   - The Problem of Fuzzy Data

2 A Solution - Basing Prolog on Fuzzy Logic
   - By Defining New Predicates
   - By Building Fuzziness into the Interpreter

3 The Ciao Prolog System
   - A Prolog Programming Environment
   - Ciao’s fuzzy package
   - Example Program
The **Ciao** Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance ...
  - constraint programming
    - X .=. Y+Z, X .=<. 2*Y
  - alternative computation rules
    - breadth-first, iterative deepening ...
  - external interfaces (C, Java, ...)
  - ...

- includes an **emacs** interface
- License: GNU Library General Public License (LGPL).
- Homepage:
  http://clip.dia.fi.upm.es/Software/Ciao/

Moritz Knapp
Implementing a Fuzzy Prolog
Motivation
A Solution - Basing Prolog on Fuzzy Logic
The Ciao Prolog System

A Prolog Programming Environment
The Ciao Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance . . .
  - constraint programming
    - $X = . Y + Z$, $X = < 2 \times Y$
  - alternative computation rules
    - breadth-first, iterative deepening . . .
  - external interfaces (C, Java, . . .)
  - . . .

- includes an emacs interface

- License: GNU Library General Public License (LGPL).
- Homepage:
  http://clip.dia.fi.upm.es/Software/Ciao/

Moritz Knapp
Implementing a Fuzzy Prolog
The Ciao Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance ...
  - constraint programming
    - $X \ =\ = \ Y+Z$, $X \ =\ < \ 2*Y$
  - alternative computation rules
    - breadth-first, iterative deepening ...
  - external interfaces (C, Java, ...)
  - ...

- includes an emacs interface

License: GNU Library General Public License (LGPL).

Homepage:
http://clip.dia.fi.upm.es/Software/Ciao/
A Prolog Programming Environment

The Ciao Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance ... 
  - constraint programming
    - $X =. Y+Z$, $X =<. 2*Y$
  - alternative computation rules
    - breadth-first, iterative deepening ... 
  - external interfaces (C, Java, ...)
  - ...

- includes an emacs interface

License: GNU Library General Public License (LGPL).

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

Moritz Knapp
Implementing a Fuzzy Prolog
A Prolog Programming Environment

The **Ciao** Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance . . .
  - constraint programming
    - $X .=. Y+Z$, $X .=<. 2*Y$
  - alternative computation rules
    - breadth-first, iterative deepening . . .
  - external interfaces (C, Java, . . .)
  - . . .
- includes an emacs interface
- License: GNU Library General Public License (LGPL).
- Homepage:
  - http://clip.dia.fi.upm.es/Software/Ciao/
The Ciao Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance ...
  - constraint programming
    - $X.\ =.\ Y+Z$, $X.\ =<.\ 2*Y$
  - alternative computation rules
    - breadth-first, iterative deepening ...
  - external interfaces (C, Java, ...)
  - ...

- includes an emacs interface
- License: GNU Library General Public License (LGPL).
- Homepage:
  http://clip.dia.fi.upm.es/Software/Ciao/

Moritz Knapp

Implementing a Fuzzy Prolog
A Prolog Programming Environment

The Ciao Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance . . .
  - constraint programming
    - \( X = . Y + Z, X =<. 2*Y \)
  - alternative computation rules
    - breadth-first, iterative deepening . . .
  - external interfaces (C, Java, . . .)
  - . . .

- includes an emacs interface

- License: GNU Library General Public License (LGPL).
- Homepage:
  http://clip.dia.fi.upm.es/Software/Ciao/
The Ciao Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance . . .
  - constraint programming
    - $X = Y + Z$, $X <= 2*Y$
  - alternative computation rules
    - breadth-first, iterative deepening . . .
  - external interfaces (C, Java, . . .)
  - . . .

- includes an emacs interface

License: GNU Library General Public License (LGPL).

Homepage:
http://clip.dia.fi.upm.es/Software/Ciao/
A Prolog Programming Environment

The Ciao Prolog Development System

- extends Prolog with
  - object-oriented programming
    - modules, classes, inheritance . . .
  - constraint programming
    - \( X =. Y+Z, X =< 2*Y \)
  - alternative computation rules
    - breadth-first, iterative deepening . . .
  - external interfaces (C, Java, . . .)
    - . . .
  - includes an emacs interface
  - License: GNU Library General Public License (LGPL).
  - Homepage:
    - http://clip.dia.fi.upm.es/Software/Ciao/
The emacs Interface

```prolog
:- module([functions,clpq]).

% A function
fact(0) :- 1.
fact(N) :- N * "fact(--N) :- N > 0.

% A predicate
append([], X, X).
append([X|Y], Z, [X|W]) :-
    append(Y, Z, W).

% Using constraints (CLP(R))
fib(X, Y) :- X =. 0, Y =. 0.
fib(X, Y) :- X =. 1, Y =. 1.
fib(N, F) :- N >. 1,
    N1 =. N - 1,
    N2 =. N - 2,
    fib(N1, F1),
    fib(N2, F2),
    F =. F1+F2.
```

Moritz Knapp
Implementing a Fuzzy Prolog
Outline

1 Motivation
   • Prolog - a Popular AI Language
   • The Problem of Fuzzy Data

2 A Solution - Basing Prolog on Fuzzy Logic
   • By Defining New Predicates
   • By Building Fuzziness into the Interpreter

3 The Ciao Prolog System
   • A Prolog Programming Environment
   • Ciao’s fuzzy package
   • Example Program
Using the fuzzy Package

Packages can be imported:

:- use_package(fuzzy).
Interface Extensions: New Declaration

- **aggr/1**
  
  ```prolog
  :- aggr prod.
  prod(X,Y,Z):- Z =. X*Y
  ```

  Aggregators implement the combination of truth values. Predefined aggregators:

  - `min`
  - `prod` - \(a \times b\)
  - `luka` - \(\max\{0, a + b - 1\}\)
  - `max`
  - `dprod` - \(a + b - a \times b\)
  - `dluka` - \(\min\{a + b, 1\}\)
Interface Extensions: New Predicates

#:2 and fuzzy_predicate/1

- for defining fuzzy predicates by listing a piecewise linear continuous membership function

young :\# fuzzy_predicate([(0,1),(35,1),(45,0),(120,0)])

~:/2

- for fuzzy clauses:

  live_long(X,Mu) :~ min
  young(X,X1),
  healthy(X,X2).
Interface Extensions: New Operators

- `fnot/1 [1150,fx]` - fuzzy negation of a fuzzy predicate
- `:=/2 [1200,xfx]`
- `:=/1 [1200,xf]`
- `#:2 [1200,xfx]`
- `=>/1 [1175,fx]`
- `aggr/1 [1150,fx]`
- ...

Moritz Knapp Implementing a Fuzzy Prolog
Motivation

A Solution - Basing Prolog on Fuzzy Logic

The Ciao Prolog System

A Prolog Programming Environment

Ciao’s fuzzy package

Example Program

Outline

1. Motivation
   - Prolog - a Popular AI Language
   - The Problem of Fuzzy Data

2. A Solution - Basing Prolog on Fuzzy Logic
   - By Defining New Predicates
   - By Building Fuzziness into the Interpreter

3. The Ciao Prolog System
   - A Prolog Programming Environment
   - Ciao’s fuzzy package
   - Example Program
small :#
    fuzzy_predicate([(1,1),(2,1),(3,0.7),(4,0.3),(5,0),(6,0)]).
large :#
    fuzzy_predicate([(1,0),(2,0),(3,0.3),(4,0.7),(5,1),(6,1)]).

die1(X,M) :-
    small(X,M).

die2(X,M) :-
    large(X,M).

two_dice(X,Y,M):~ prod
    die1(X,M1),
    die2(Y,M2).

sum(2,M) :-
    two_dice(1,1,M1).

sum(5,M) :~ dprod
    two_dice(4,1,M1),
    two_dice(1,4,M2),
    two_dice(3,2,M3),
    two_dice(2,3,M4).
small :#
    fuzzy_predicate([(1,1),(2,1),(3,0.7),(4,0.3),(5,0),(6,0)]).

large :#
    fuzzy_predicate([(1,0),(2,0),(3,0.3),(4,0.7),(5,1),(6,1)]).

die1(X,M) :~
    small(X,M).

die2(X,M) :~
    large(X,M).

two_dice(X,Y,M):~ prod
die1(X,M1),
die2(Y,M2).

sum(2,M) :~
    two_dice(1,1,M1).

sum(5,M) :~ dprod
two_dice(4,1,M1),
two_dice(1,4,M2),
two_dice(3,2,M3),
two_dice(2,3,M4).
Summary

• fuzzy data is a challenge that can be handled in different ways
  • [Ciao] is one possible solution
  • nice IDE for experimenting with fuzziness
  • also worth a look: reimplementations of the Prolog interpreter (see literature)
Summary

- fuzzy data is a challenge that can be handled in different ways
- **Ciao** is one possible solution
  - nice IDE for experimenting with fuzziness
  - also worth a look: reimplementations of the Prolog interpreter (see literature)
Summary

- Fuzzy data is a challenge that can be handled in different ways.
- **Ciao** is one possible solution.
- Nice IDE for experimenting with fuzziness.
- Also worth a look: reimplementations of the Prolog interpreter (see literature).
Summary

- fuzzy data is a challenge that can be handled in different ways
- Ciao is one possible solution
- nice IDE for experimenting with fuzziness
- also worth a look: reimplementations of the Prolog interpreter (see literature)
Thanks for Listening
S. Gottwald.  
*Fuzzy Sets and Fuzzy Logic.*  

L.A. Zadeh.  
Fuzzy Sets.  

The Implementation of FProlog - A Fuzzy Prolog Interpreter.  