Tutorial on IDP

Ingmar Dasseville, Pieter Van Hertum, Marc Denecker

September 6, 2016
Modeling and solving in mathematics

An example domain:

_I am one year older than double the age of my son._
_The sum of our ages lies between 70 and 80._

A mathematical modeling:

- Choose symbols for abstraction:
  - $x$: my age
  - $y$: my son’s age

- Express information (in equations):
  - $x - 2y = 1$
  - $x + y \geq 70$
  - $x + y \leq 80$

Now, we can solve problems with this modeling.
Modeling and solving in mathematics

An example domain:

I am one year older than double the age of my son.
The sum of our ages lies between 70 and 80.

A mathematical modeling:

- Choose symbols for abstraction:
  - x: my age
  - y: my son’s age

- Express information (in equations):
  \[
  x - 2y = 1 \\
  x + y \geq 70 \\
  x + y \leq 80
  \]

Now, we can solve problems with this modeling.
Modeling and solving in mathematics

An example domain:

I am one year older than double the age of my son.
The sum of our ages lies between 70 and 80.

A mathematical modeling:

- Choose symbols for abstraction:
  - $x$: my age
  - $y$: my son’s age
- Express information (in equations):
  - $x - 2y = 1$
  - $x + y \geq 70$
  - $x + y \leq 80$

Now, we can solve problems with this modeling.
Modeling and solving in mathematics

An example domain:

I am one year older than double the age of my son.
The sum of our ages lies between 70 and 80.

A mathematical modeling:

- Choose symbols for abstraction:
  - \(x\): my age
  - \(y\): my son’s age

- Express information (in equations):
  - \(x - 2y = 1\)
  - \(x + y \geq 70\)
  - \(x + y \leq 80\)

Now, we can solve problems with this modeling.
Ingredients:

- **Symbols**: Mathematical variables $x, y$
- **Constraints** over symbols $x - 2y = 1$
- **Assignments** of values to symbols $x=3, y=5$
- Some assignments **satisfy** constraints, others don’t.

What problem can we solve? Many!
Ingredients:

- **Symbols**: mathematical variables \( x, y \)
- **Constraints** over symbols \( x - 2y = 1 \)
- **Assignments** of values to symbols \( x=3 \quad y=5 \)
- Some assignments **satisfy** constraints, others don’t.

What problem can we solve? **Many!**
Ingredients:

- **Symbols**: mathematical variables \(x, y\)
- **Constraints** over symbols \(x - 2y = 1\)
- **Assignments** of values to symbols \(x=3 \quad y=5\)
- Some assignments satisfy constraints, others don’t.

What problem can we solve? Many!
Modeling Information; Solving problems

We can solve multiple problems with this specification:

- Search one or more satisfying assignments.
- Evaluate if assignment \( x=50 \) \( y=26 \) satisfies the equations.
- What is the solution in case \( y=26 \).
- Is it entailed that my son is adult?
  - Is he adult in every satisfying assignment?
- Search satisfying assignment where my age \( x \) is maximal.
- What are the possible values for \( x \)?
- ...

precision, compactness, reuse
Modeling Information; Solving problems

We can solve multiple problems with this specification:

- Search one or more satisfying assignments.
- Evaluate if assignment \( x=50 \ y=26 \) satisfies the equations.
- What is the solution in case \( y=26 \).
- Is it entailed that my son is adult?
  - Is he adult in every satisfying assignment?
- Search satisfying assignment where my age \( x \) is maximal.
- What are the possible values for \( x \)?
- ...

precision, compactness, reuse
Modeling Information; Solving problems

We can solve multiple problems with this specification:

- Search one or more satisfying assignments.
- Evaluate if assignment $x=50 \ y=26$ satisfies the equations.
- What is the solution in case $y=26$.
- Is it entailed that my son is adult?
  - Is he adult in every satisfying assignment?
- Search satisfying assignment where my age $x$ is maximal.
- What are the possible values for $x$?
- ...

precision, compactness, reuse
Modeling Information; Solving problems

We can solve multiple problems with this specification:

▶ Search one or more satisfying assignments.
▶ Evaluate if assignment $x=50 \ y=26$ satisfies the equations.
▶ What is the solution in case $y=26$.
▶ Is it entailed that my son is adult?
  ▶ Is he adult in every satisfying assignment?
▶ Search satisfying assignment where my age $x$ is maximal.
▶ What are the possible values for $x$?
▶ . . .

precision, compactness, reuse
Modeling Information; Solving problems

We can solve multiple problems with this specification:

➤ Search one or more satisfying assignments.
➤ Evaluate if assignment $x=50 \ y=26$ satisfies the equations.
➤ What is the solution in case $y=26$.
➤ Is it entailed that my son is adult?
  ➤ Is he adult in every satisfying assignment?
➤ Search satisfying assignment where my age $x$ is maximal.
➤ What are the possible values for $x$?
➤ . . .

precision, compactness, reuse
Modeling Information; Solving problems

We can solve multiple problems with this specification:

- Search one or more satisfying assignments.
- Evaluate if assignment $x=50 \ y=26$ satisfies the equations.
- What is the solution in case $y=26$.
- Is it entailed that my son is adult?
  - Is he adult in every satisfying assignment?
- Search satisfying assignment where my age $x$ is maximal.
  - What are the possible values for $x$?
  - ...

precision, compactness, reuse
We can solve multiple problems with this specification:

- Search one or more satisfying assignments.
- Evaluate if assignment $x=50 \ y=26$ satisfies the equations.
- What is the solution in case $y=26$.
- Is it entailed that my son is adult?
  - Is he adult in every satisfying assignment?
- Search satisfying assignment where my age $x$ is maximal.
- What are the possible values for $x$?
- ...

... precision, compactness, reuse
Modeling Information; Solving problems

We can solve multiple problems with this specification:

- Search one or more satisfying assignments.
- Evaluate if assignment $x=50$, $y=26$ satisfies the equations.
- What is the solution in case $y=26$.
- Is it entailed that my son is adult?
  - Is he adult in every satisfying assignment?
- Search satisfying assignment where my age $x$ is maximal.
- What are the possible values for $x$?
- ...
Modeling Information; Solving problems

We can solve multiple problems with this specification:

- Search one or more satisfying assignments.
- Evaluate if assignment $x=50 \ y=26$ satisfies the equations.
- What is the solution in case $y=26$.
- Is it entailed that my son is adult?
  - Is he adult in every satisfying assignment?
- Search satisfying assignment where my age $x$ is maximal.
- What are the possible values for $x$?
- ...
Now in IDP.

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=AgePuzzle
General principle and terminology

- set of symbols → Vocabulary
  - $x, y$
- set of constraints → Theory
  - $x+y \geq 30$
- Assignment of values to symbols → Structure
  - $x=53 \ y=26$
  - May be partial $y=26 \ x=?$
- Assignments that satisfy constraints → Models
Terminology

A **modeling** is a theory.

A **model** is a structure satisfying the theory.
Modeling and Solving

- Modeling = Specification = Knowledge representation

  Write a theory of which the world is a satisfying assignment

- Solving means ... depends on the sort of problem.
Modeling and Solving

- Modeling = Specification = Knowledge representation

Write a theory of which the world is a satisfying assignment

- Solving means . . .
  depends on the sort of problem.
Modeling and Solving

- Modeling = Specification = Knowledge representation

  Write a theory of which the world is a satisfying assignment

- Solving means . . .
  depends on the sort of problem.
A theory is not a program.
A theory is not a problem description.
A theory is a description of a class of satisfying assignments.
Modeling and Solving

In IDP:

- `modelexpand(<theory>,<inputstructure>)`
- `minimise(<theory>,<inputstructure>,<term>)`
- `optimalpropagate(<theory>,<inputstructure>)`
- ...

A short list, but extremely flexible.
From linear equations to logic

- The principles remain the same.
  - Symbols, theories, structures, satisfaction.

- More complex symbols, theories, values and structures.

\[ \forall d[\text{dep}]s[\text{shift}] : \#\{n[nurse] : NurseAt(n, d, s)\} > 3. \]

In words:

For all department \(d\), shift \(s\): the number of elements of the set of nurses \(n\) that work at \(d\) during \(s\) is greater than 3.
Coloring graphs

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=MapColoring
More complex symbols. . .
  ▶ From numerical “variables” to set, relation and function “variables”.

More complex “constraints”.

More domains and more complex ones.
  ▶ From integer or real numbers to multiple and arbitrary domains.

More complex values.
  ▶ From numbers to sets, relations, functions.
Vocabulary

typically V{
  type human
  type num isa int
  P(num)
  Married(human,human)
  MySonIsAdult
  Age(human): num
  Boss: human
}
**Terminology**

A symbol for which we do not know the value:

- In mathematical modeling: a **variable**
- In logic: a **constant**

In logic, a variable is something different:

\[ \forall x : \text{Man}(x) \Rightarrow \text{Human}(x). \]

A logical variable is more “variable” than a mathematical or constraint variable.
Terminology

A symbol for which we do not know the value:
- In mathematical modeling: a variable
- In logic: a constant

In logic, a variable is something different:

$$\forall x : Man(x) \Rightarrow Human(x).$$

A logical variable is more “variable” than a mathematical or constraint variable.
Terminology

A symbol for which we do not know the value:

- In mathematical modeling: a variable
- In logic: a constant

In logic, a variable is something different:

\[ \forall x : \text{Man}(x) \Rightarrow \text{Human}(x). \]

A logical variable is more “variable” than a mathematical or constraint variable.
Structures: assignments

structure S:V{
    num={2..100}
    Node={A..D}
    Human={Pieter;Ingmar;Marc}
    Edge={A,B; B,C; C,D}
    MySonIsAdult = {()}
    Cost={ Delhaize,dreft -> 2; Colruyt, dreft -> 2}
    Boss=Marc
}

- A structure $S$ of vocabulary $V$
- Lefthand side: symbol of $V$
- Righthand side: value of type of $V$
Structures express data.

Structures may be partial.

Structures need to specify a finite domain for every type.
(We are working on it)
Structures express data.

Structures may be partial.

Structures need to specify a finite domain for every type. (We are working on it)
Structures express data.

Structures may be partial.

Structures need to specify a finite domain for every type. (We are working on it)
**Constructed types**

```
type day constructed from
    { mon; tue; wed; thu; fri; sat; sun }
```

Specifies multiple things:

- type symbol `day`
- constant symbols of type `day`
- values per constant: constant and value is the same here
  ```
  mon = mon
  ```
- a value for type `day`
  ```
  day = { mon; tue; wed; thu; fri; sat; sun }
  ```
Theory

theory T: V{
  ...
}

- Theory $T$ written in vocabulary $V$
- Contains formulas and definitions.
The Einstein Puzzle

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=Einstein

Under “File” select “The Einstein Puzzle”.
## Logical symbols

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Logical symbols</th>
<th>IDP-symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>... and ...</td>
<td>... ∧ ...</td>
<td>... &amp; ...</td>
</tr>
<tr>
<td>... or ...</td>
<td>... ∨ ...</td>
<td>...</td>
</tr>
<tr>
<td>If ... then ...</td>
<td>... ⇒ ...</td>
<td>... =&gt; ...</td>
</tr>
<tr>
<td>... if and only if ...</td>
<td>... ⇔ ...</td>
<td>... &lt;==&gt;</td>
</tr>
<tr>
<td>not ...</td>
<td>¬ ...</td>
<td>~ ...</td>
</tr>
<tr>
<td>for all ...</td>
<td>∀ (x : \ldots)</td>
<td>! (x : \ldots)</td>
</tr>
<tr>
<td>there exists ...</td>
<td>(\exists x : \ldots)</td>
<td>? (x : \ldots)</td>
</tr>
<tr>
<td>there exists n ...</td>
<td>(\exists n \ x : \ldots)</td>
<td>?(n \ x : \ldots)</td>
</tr>
<tr>
<td>...</td>
<td>(\exists &lt; n \ x : \ldots)</td>
<td>?&lt;(n \ x : \ldots)</td>
</tr>
<tr>
<td>...</td>
<td>(\exists &gt; n \ x : \ldots)</td>
<td>?&gt;(n \ x : \ldots)</td>
</tr>
</tbody>
</table>
Type inference

```
type human
type num
Age(human,num)
...
∀ x:  ∃ y:  Age(x,y).
```

Type inference infers:
```
∀ x[human]:  ∃ y[num]:  Age(x,y).
```
Type inference

type human

type num

Age(human,num)

... 

\[\forall x: \exists y: \text{Age}(x,y).\]

Type inference infers:

\[\forall x\text{[human]}: \exists y\text{[num]}: \text{Age}(x,y).\]
Aggregates

number of elements of P

\[ \# \{ x, y : \ P(x, y) \} . \]

sum of \( x+y \), for all \( (x,y) \in P \)

\[ \text{sum}\{ x, y : \ P(x, y) : x+y \} . \]

minimum of set \( \{ \ x : Q(x) \& R(x) \} \):

\[ \text{min}\{ x : Q(x) \& R(x) : x \} . \]

maximum:

\[ \text{max}\{ x : Q(x) \& R(x) : x \} . \]

Nesting is allowed, as in:

\[ P_{nest} = \text{sum}\{x[n\text{um}] : x = \# \{ y : Q(x,y) \} : x \} . \]
Aggregates

number of elements of \( P \)
\[
\# \{ x, y : P(x, y) \}.
\]

sum of \( x+y \), for all \( (x, y) \in P \)
\[
\text{sum}\{x, y : P(x, y) : x+y\}.
\]

minimum of set \( \{ x : Q(x) \& R(x) \} \):
\[
\text{min}\{x : Q(x) \& R(x) : x\}.
\]

maximum :
\[
\text{max}\{x : Q(x) \& R(x) : x\}.
\]

Nesting is allowed, as in:
\[
P_{nest} = \text{sum}\{x[num] : x= \# \{ y : Q(x, y) \} : x\}.
\]
Aggregates

number of elements of P
\[ \#{x,y: P(x,y)}. \]

sum of \(x + y\), for all \((x,y) \in P\)
\[ \text{sum}\{x,y: P(x,y) : x+y\}. \]

minimum of set \{ \(x : Q(x) \& R(x)\)\}:
\[ \text{min}\{x: Q(x) \& R(x) : x\}. \]

maximum :
\[ \text{max}\{x: Q(x) \& R(x) : x\}. \]

Nesting is allowed, as in:
\[ P_{nest} = \text{sum}\{x[n\text{um}] : x= \#{y: Q(x,y)} : x \}. \]
Aggregates

number of elements of \( P \)
\[
\#\{ x, y : P(x, y) \}.
\]

sum of \( x+y \), for all \( (x, y) \in P \)
\[
\sum\{ x, y : P(x, y) : x+y \}.
\]

minimum of set \( \{ x : Q(x) \& R(x) \} \):
\[
\min\{ x : Q(x) \& R(x) : x \}.
\]

maximum:
\[
\max\{ x : Q(x) \& R(x) : x \}.
\]

Nesting is allowed, as in:
\[
P_{nest} = \sum\{ x[num] : x = \#\{ y : Q(x, y) \} : x \}.
\]
Aggregates

number of elements of P
\[ \#\{x,y : P(x,y)\} \]
sum of \( x+y \), for all \((x,y) \in P\)
\[ \text{sum}\{x,y : P(x,y) : x+y\} \]
minimum of set \\{ \( x : Q(x) \& R(x) \) \}:
\[ \text{min}\{x : Q(x) \& R(x) : x\} \]
maximum:
\[ \text{max}\{x : Q(x) \& R(x) : x\} \]
Nesting is allowed, as in:
\[ P_{nest} = \text{sum}\{x[num] : x = \#\{y : Q(x,y)\} : x\} \]
http://dtai.cs.kuleuven.be/krr/idp-ide/?present=Agg

Experiment with different input/output.

- Compute aggregates from structure.
- Compute structures from aggregates.
- Compute minimal structure from some aggregates.
Definitions

{ ! x[human]: WorkingAge(x) <- Age(x)>18 & Age(x)<65. }

{ ! x y: Parent(x,y) <- Mother(x,y).
  ! x y: Parent(x,y) <- Father(x,y).
}

{ Day(mon). ... Day(sun). }

▶ { Rules }
▶ One rule:
  ▶ ! x y: Atom <- Body .
  ▶ In front only ! , not ?
  ▶ Only atomic head
  ▶ <- : definitional operator
  ▶ Body is a formula.
▶ Atomic rules Day(mon).
Definitions

{ ! x[human]: WorkingAge(x) <-
  Age(x)>18 & Age(x)<65. }

{ ! x y: Parent(x,y) <- Mother(x,y).
! x y: Parent(x,y) <- Father(x,y).
}

{ Day(mon). ... Day(sun). }

▶ { Rules }
▶ One rule:
  ▶ ! x y: Atom <- Body .
  ▶ In front only ! , not ?
  ▶ Only atomic head
  ▶ <- : definitional operator
  ▶ Body is a formula.
▶ Atomic rules Day(mon) .
Definitions

{ ! x[human]: WorkingAge(x) <-
    Age(x)>18 & Age(x)<65. }

{ ! x y: Parent(x,y) <- Mother(x,y).
! x y: Parent(x,y) <- Father(x,y). }

{ Day(mon). ... Day(sun). }

▷ { Rules }
▷ One rule:
   ▷ ! x y: Atom <- Body .
▷ In front only ! , not ?
▷ Only atomic head
▷ <- : definitional operator
▷ Body is a formula.
▷ Atomic rules Day(mon).
Definitions

\{ ![x][human]: WorkingAge(x) <- \\
\hspace{1cm} \text{Age}(x)>18 \& \text{Age}(x)<65. \} \\
\{ ![x \ y]: Parent(x,y) <- Mother(x,y). \\
\hspace{1cm} ![x \ y]: Parent(x,y) <- Father(x,y). \} \\
\{ Day(mon). \ldots \hspace{1cm} Day(sun). \} \\

\hspace{1cm} \{ Rules \} \\
\hspace{1cm} \{ One rule: \}
\hspace{1cm} ![x \ y]: Atom <- Body .
\hspace{1cm} In front only! \ , \ not? \\
\hspace{1cm} Only atomic head \\
\hspace{1cm} <- : definitional operator \\
\hspace{1cm} Body is a formula.

\hspace{1cm} Atomic rules Day(mon).
Definitions

{ ! x[human]: WorkingAge(x) <- Age(x)>18 & Age(x)<65. }

{ ! x y: Parent(x,y) <- Mother(x,y).
  ! x y: Parent(x,y) <- Father(x,y). }

{ Day(mon). ... Day(sun). }

▷ { Rules }
▷ One rule:
  ▷ ! x y: Atom <- Body .
  ▷ In front only ! , not ?
  ▷ Only atomic head
  ▷ <- : definitional operator
  ▷ Body is a formula.

▷ Atomic rules Day(mon) .
Definitions

{ ! x[human]: WorkingAge(x) <-
  Age(x)>18 & Age(x)<65. }

{ ! x y: Parent(x,y) <- Mother(x,y).
! x y: Parent(x,y) <- Father(x,y). }

{ Day(mon). ... Day(sun). }
Never confuse a definition for a set of implications

Compare:

\{
  ! x y: \text{Parent}(x,y) \leftarrow \text{Mother}(x,y).
  ! x y: \text{Parent}(x,y) \leftarrow \text{Father}(x,y).
\}
and

\begin{align*}
  ! x y: & \quad \text{Parent}(x,y) \leq \text{Mother}(x,y). \\
  ! x y: & \quad \text{Parent}(x,y) \leq \text{Father}(x,y).
\end{align*}

Suppose A is not mother nor father of B.

- definition says: A is not parent of B.
- implications say: nothing! A could be parent of B or not.

For a defined atom to be true, at least one case has to be the case.
Never confuse a definition for a set of implications

Compare:
\[
\{ \\
! x \ y: \text{Parent}(x,y) \leftarrow \text{Mother}(x,y). \\
! x \ y: \text{Parent}(x,y) \leftarrow \text{Father}(x,y). \\
\}
\]

and
\[
! x \ y: \text{Parent}(x,y) \leq \text{Mother}(x,y). \\
! x \ y: \text{Parent}(x,y) \leq \text{Father}(x,y).
\]

Suppose A is not mother nor father of B.

- definition says: A is not parent of B.
- implications say: nothing! A could be parent of B or not.

For a defined atom to be true, at least one case has to be the case.
Never confuse a definition for a set of implications

Compare:

\[
\{ \\
! x \ y: \ Parent(x,y) \leftarrow \ Mother(x,y). \\
! x \ y: \ Parent(x,y) \leftarrow \ Father(x,y). \\
\}
\]

and

\[
! x \ y: \ Parent(x,y) \leq \ Mother(x,y). \\
! x \ y: \ Parent(x,y) \leq \ Father(x,y).
\]

Suppose A is not mother nor father of B.

- definition says: A is not parent of B.
- implications say: nothing! A could be parent of B or not.

For a defined atom to be true, at least one case has to be the case.
Is that a big difference?

You bet!

Let the numbers speak!

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=DefImp

If the number of humans is $n$, the definition has 1 model, the set of implications has $2^{n^2}$ models.
Is that a big difference?

You bet!

Let the numbers speak!

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=DefImp

If the number of humans is $n$, the definition has 1 model, the set of implications has $2^{n^2}$ models.
No fixed input/output connected to definitions in FO(.).

- E.g., any of Parent, Father, Mother could be given in structure or not.

A definition only states a logical relationship between defined symbols and parameter symbols:

- Parent in terms of Father, Mother

IDP will try to satisfy it with whatever is given.
Inductive definitions

\[
\begin{align*}
! x & : \text{Reachable}(A). \\
! x & : \text{Reachable}(x) \leftarrow \\
& \quad \text{? } z : \text{Reachable}(z) \& \text{Edge}(z, x).
\end{align*}
\]

Compare this with inductive definitions in mathematics:

Definition
The set of reachable nodes from A are defined by induction:

- A is reachable.
- if z is reachable and there is an edge from z to x, then x is reachable.

The reachability relation is the least relation that satisfies those rules.
There is no fixed dataflow.
  ▶ From a known **Edge**, compute **Reachable**.
  ▶ From a known **Reachable**, compute **Edge**.
  ▶ Or both half partially known.

Below, **Reachable** is known to be the set of all nodes.

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=TransClosure
See one inductive definition and 3 FO formulas that many think are equivalent, and convince yourself that none of them are equivalent. 
http://dtai.cs.kuleuven.be/krr/idp-ide/?present=DefClark
Here our tour over IDP finishes.

Now, we can build software solutions with these ingredients.
Study programme selection

https://dtai.cs.kuleuven.be/software/idp/examples/courseselection

Interactive configuration

- a hard problem for standard software technologies
- 5 forms of inference on the same theory, to provide 5 forms of inference.
Course scheduling

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=CourseScheduling

IDP is used in several schools already, and in our department for certain scheduling tasks.
(the web-server may not have the ressources to visualize the solution)
Planning: Hanoi

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=Hanoi

- Run
- Click beside the tower to see the plan in action.
All visualisations were made with IDP.
Conclusion

Flexibility due to

- natural specification language
- rich expressivity
- no fixed data flow
- no fixed problem
- multiple forms of inference
Future

- Challenges everywhere (language, efficiency, other forms of inference)
- For use in software development, the main problem of IDP is communication with the “world”: 
  - calling,
  - be called,
  - interacting

with other programs in other languages.
End
Simulating binary quantification

Consider:

- “Every person older than 65 is retired”
- “There exists a person older than 65 that is retired”

Note the symmetry! “every” vs “exists”

Translation in logic:

- “Every person older than 65 is retired”
  
  \(! x \text{[person]}: \text{Age}(x) \geq 65 \Rightarrow \text{Retired}(x).\)

- “There exists a person older than 65 that is retired”
  
  \(? x \text{[person]}: \text{Age}(x) \geq 65 \land \text{Retired}(x).\)

Symmetry is broken:

\(! x : \ldots \Rightarrow \ldots \) versus \(?x: \ldots \land \ldots \)
Simulating binary quantification

Consider:

- “Every person older than 65 is retired”
- “There exists a person older than 65 that is retired”

Note the symmetry! “every” vs “exists”

Translation in logic:

- “Every person older than 65 is retired”
  \[
  ! \ x[\text{person}] : \ \text{Age}(x)\geq65 \Rightarrow \text{Retired}(x).
  \]

- “There exists a person older than 65 that is retired”
  \[
  ? \ x[\text{person}] : \ \text{Age}(x)\geq65 \& \text{Retired}(x).
  \]

Symmetry is broken:

\[
! \ x : \ \text{..} \Rightarrow \ \text{..} \ \text{versus} \ ? \ x : \ \text{..} \ & \ \text{..}
\]
Unsatisfiable theories

The following example shows how to search for unsatisfiable subtheories in a theory using the command “printcore(theory,Structure)”.

http://dtai.cs.kuleuven.be/krr/idp-ide/?present=PrintCore