Groundedness in Logics With a Fixpoint Semantics

Public Defence

Bart Bogaerts

Department of computer science, KU Leuven
Databases & theoretical computer science group, UHasselt

June 24, 2015

Content

Content

Research topics in KRR

1. Defining knowledge representation languages
2. Defining inference methods
3. Implementing inference engines
4. Studying complexity, expressivity and succinctness
5. Relationships between languages/systems/inferences/...
6. Applications

Knowledge Representation and Reasoning

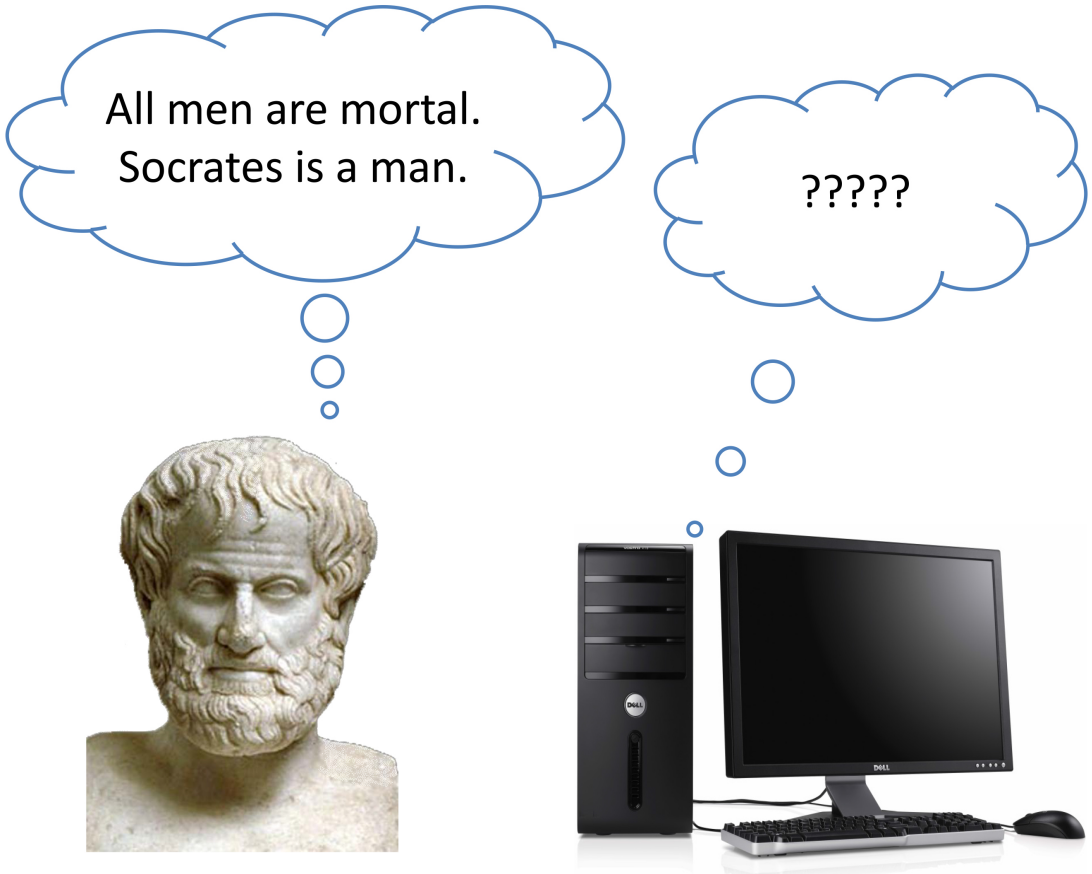
Goal

Representing knowledge about the world in a form that a computer system can utilise to solve complex tasks.

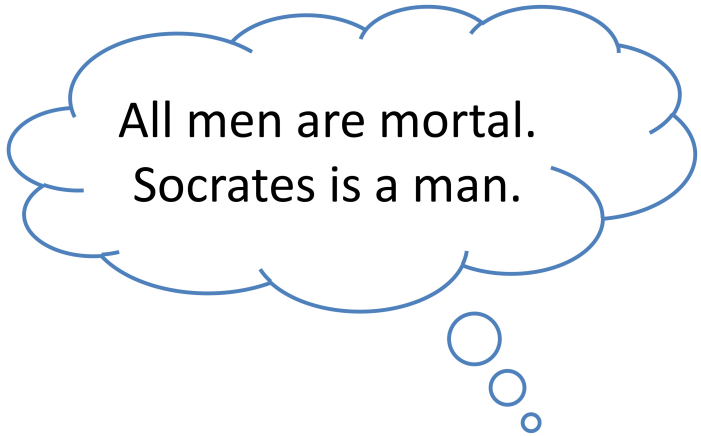
1. KR languages

Goal

Representing knowledge about the world in a form that a computer system can utilise to solve complex tasks.



1. KR Languages: First-Order Logic

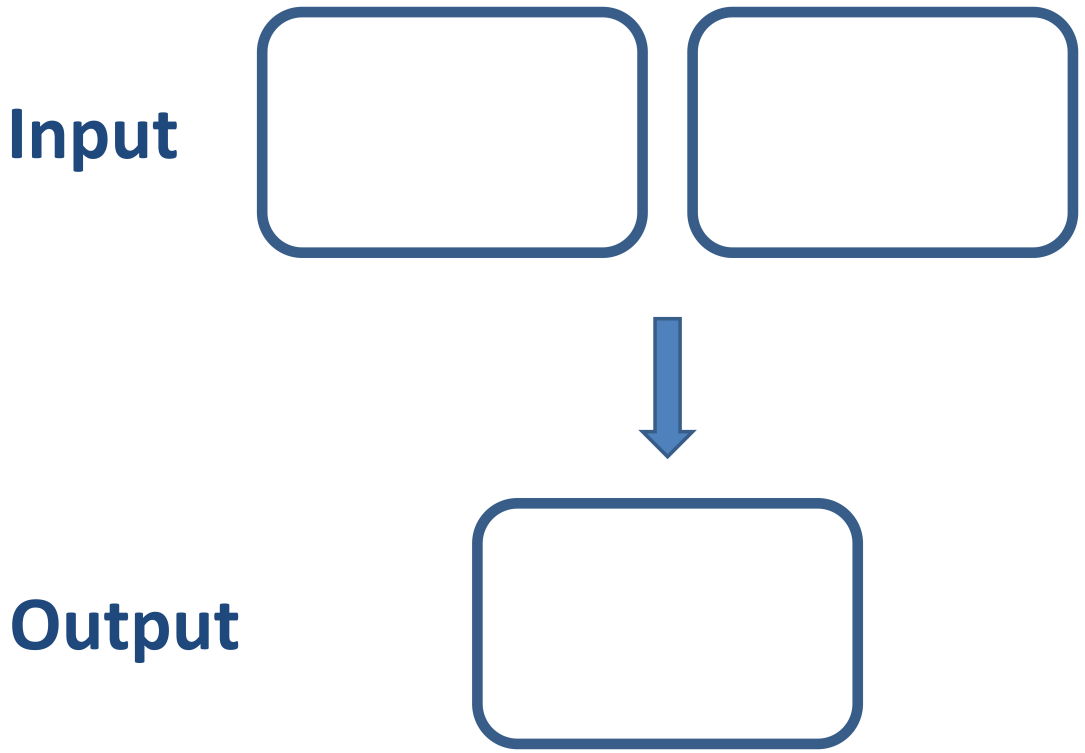


- First-order logic (FO):
  - All men are mortal: $\forall x : Man(x) \Rightarrow Mortal(x)$
  - Socrates is a man: $Man(Socrates)$
- Certain types of knowledge inexpressible in FO

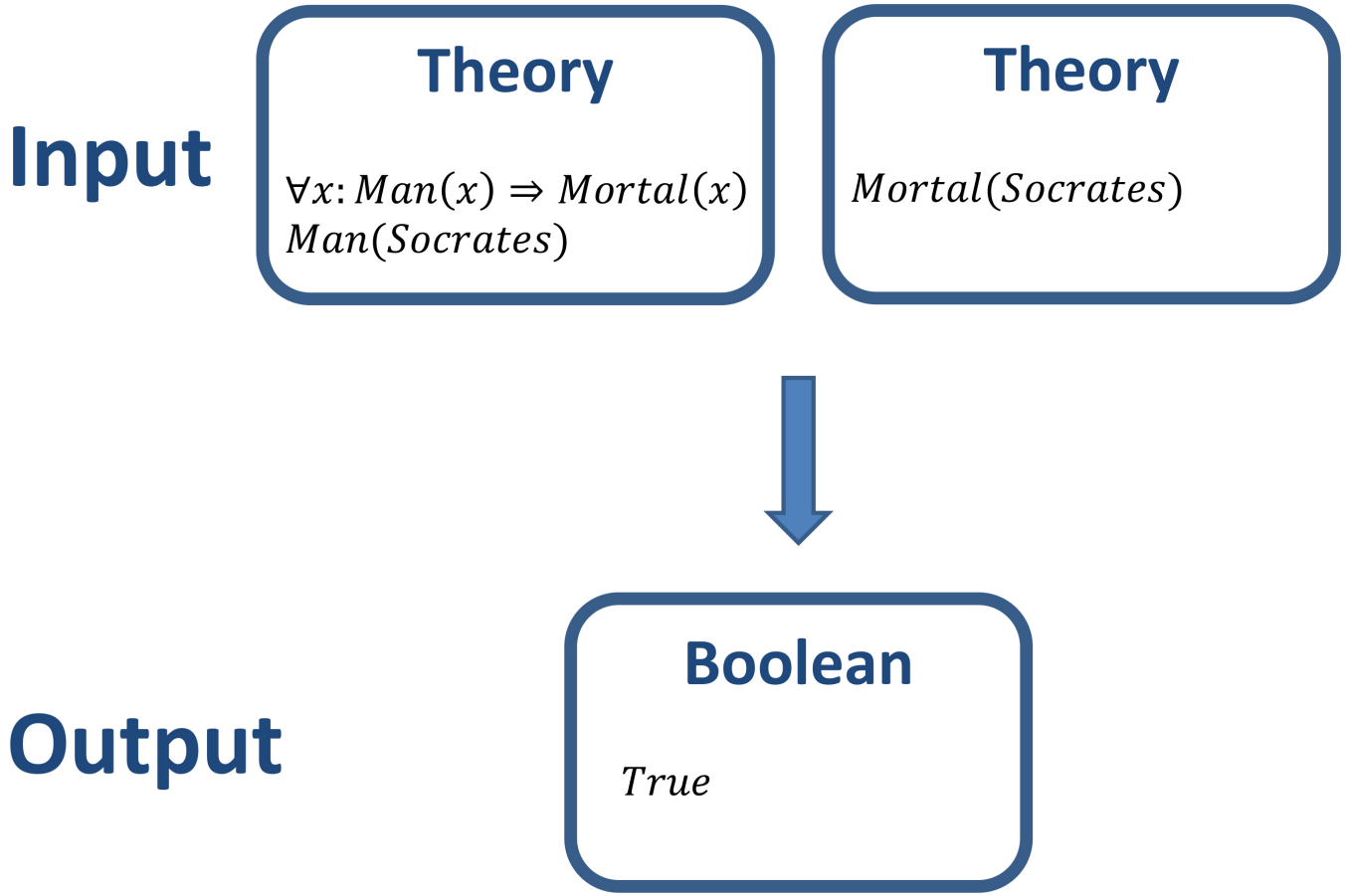
2. Inference Methods

Goal

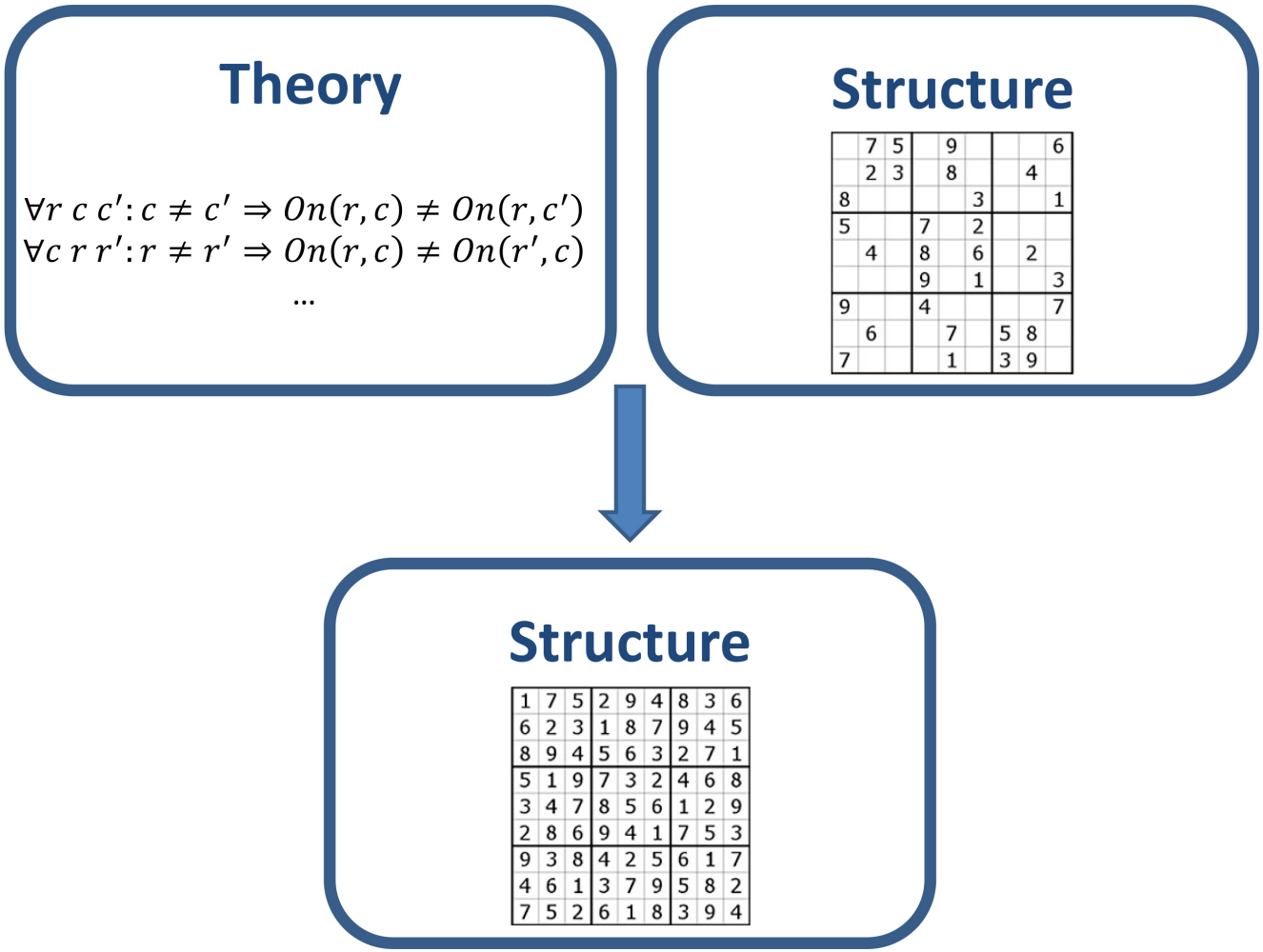
Representing knowledge about the world in a form that a computer system can utilise to solve complex tasks.



Inference Methods: Deduction



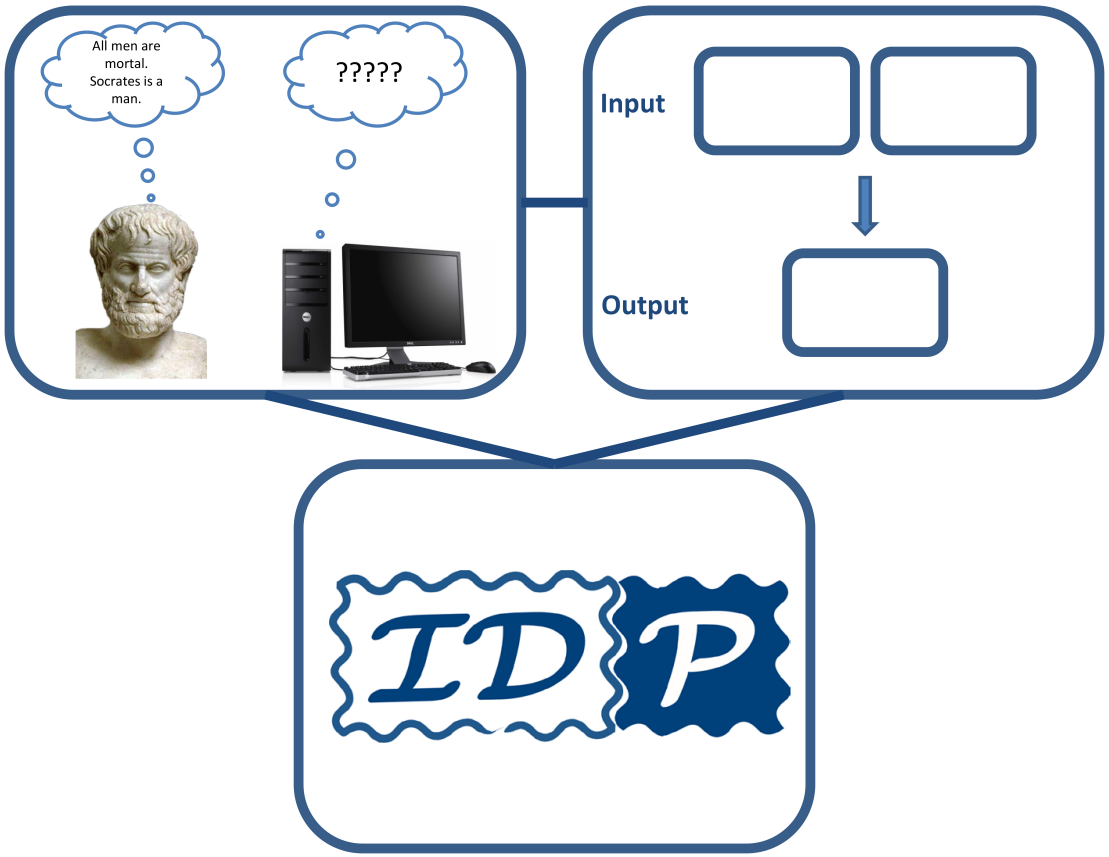
Inference Methods: Model Expansion



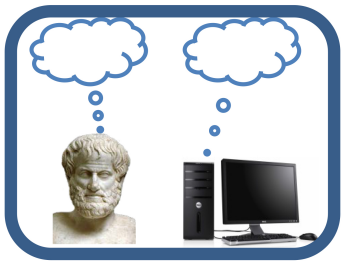
3. Inference Engines

Goal

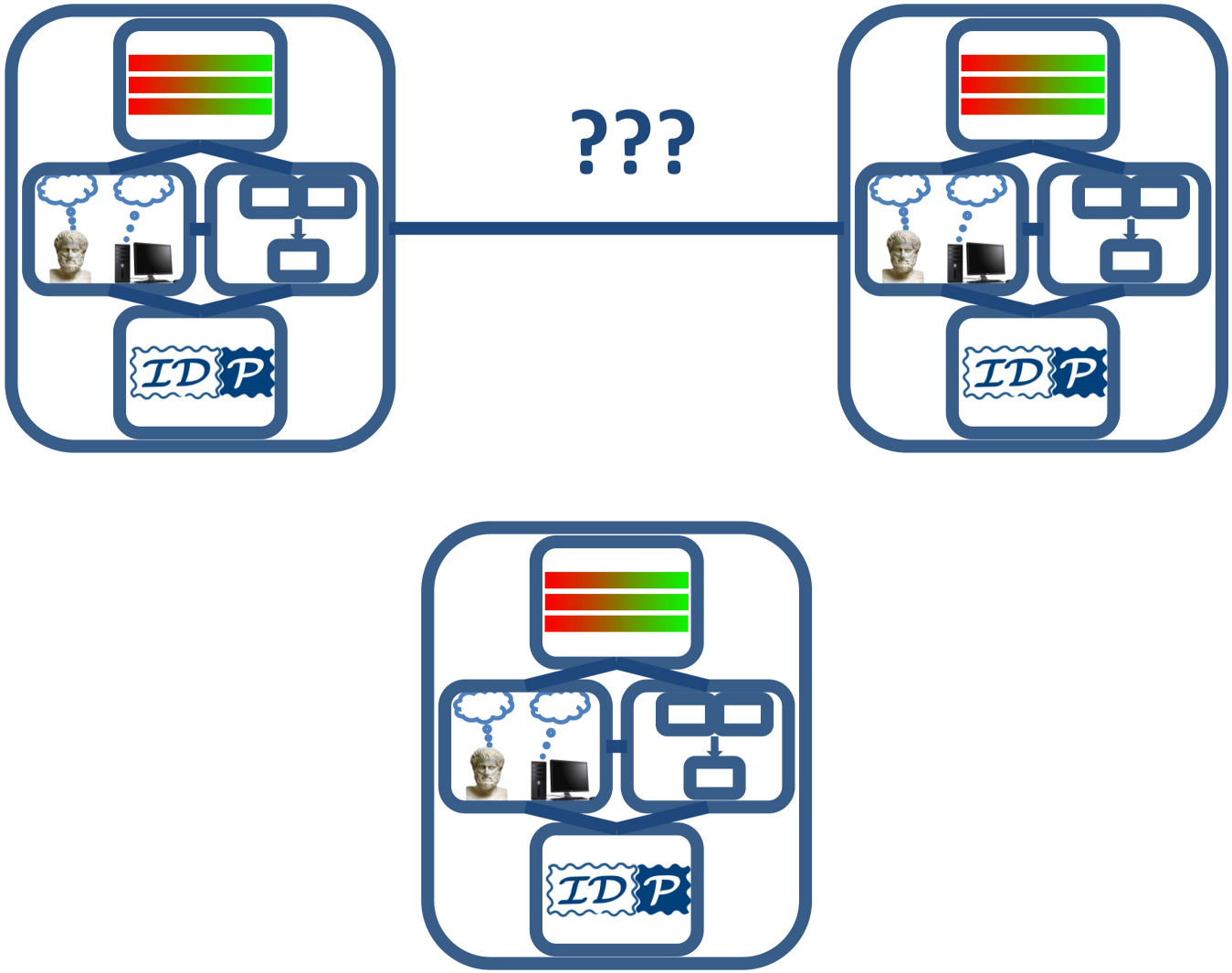
Representing knowledge about the world in a form that a computer system can utilise to solve complex tasks.



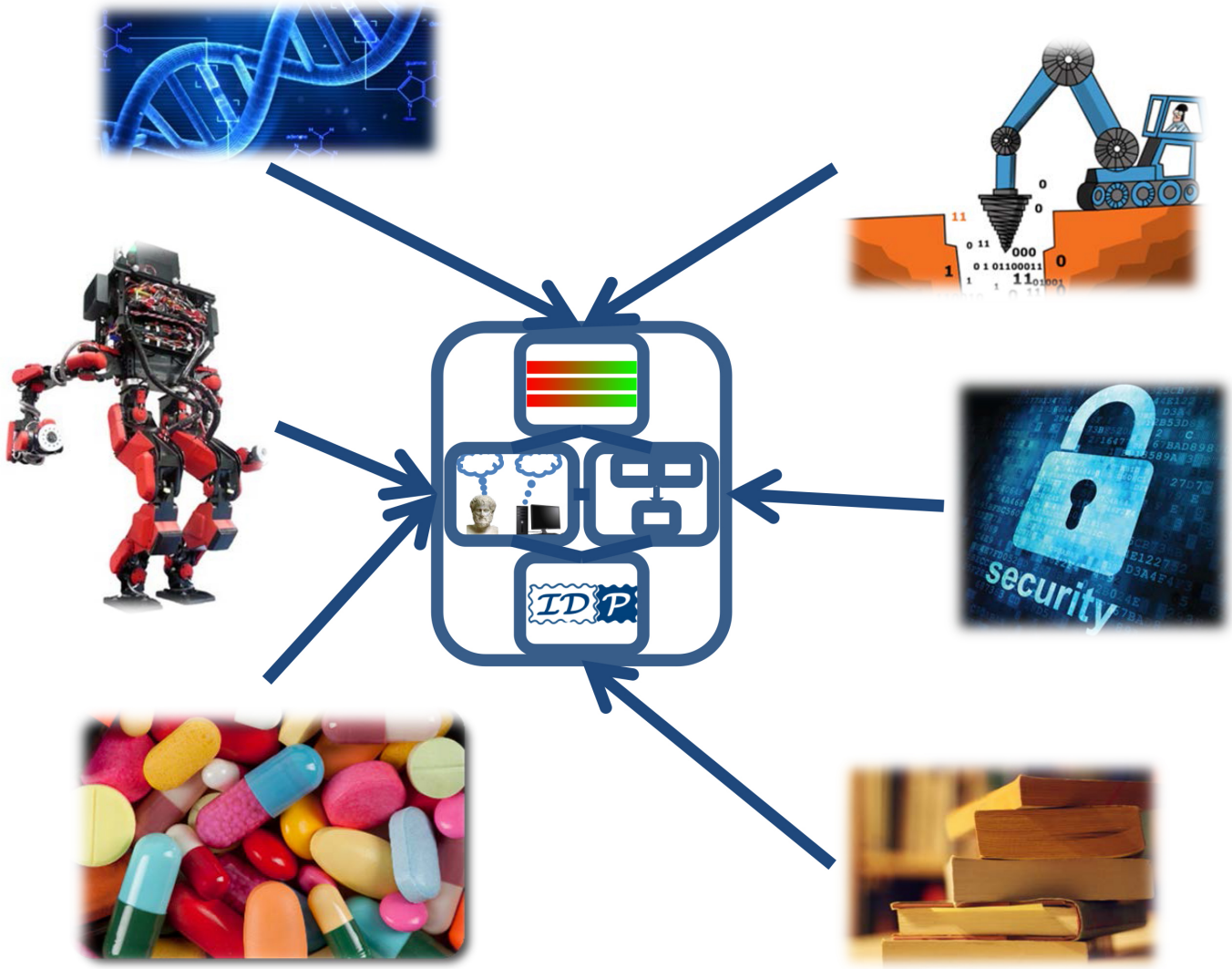
4. Expressivity, Succinctness, Complexity



5. Relationship Between Languages/Systems/Inferences/...



6. Applications



Content

Knowledge Representation in Practice

How to use IDP to solve real-world problems?

1. Collect knowledge about the problem domain
2. Model the knowledge in the IDP language
3. Map your problem to one or more inference methods
4. Run the system

Case study: Course scheduling

1. Collecting knowledge

- Every course consists of several (at least one) sessions.
- Every session is given in a room with capacity at least as high as the number of students enrolled for that session.
- Instructors can only teach one session simultaneously.
- ...

2. Modelling knowledge

- Every course consists of several (at least one) sessions.
- Every session is given in a room with capacity at least as high as the number of students enrolled for that session.
- Instructors can only teach one session simultaneously.
- ...

Vocabulary:

- Types: *course*, *session*, *room*, *capacity*($\mathbb{N}$), *student*, *instructor*, *time* ...

2. Modelling knowledge

- Every course consists of several (at least one) sessions.
- Every session is given in a room with capacity at least as high as the number of students enrolled for that session.
- Instructors can only teach one session simultaneously.
- ...

Vocabulary:

- Types: *course*, *session*, *room*, *capacity*($\mathbb{N}$), *student*, *instructor*, *time* ...
- Relations/Functions:
  - *partOf*(*session*, *course*)
  - *locationOf*(*session*) $\rightarrow$ *room*
  - *capacityOf*(*room*) $\rightarrow \mathbb{N}$
  - *enrolledIn*(*student*, *session*)
  - *teacherOf*(*session*) $\rightarrow$ *instructor*
  - ...

2. Modelling knowledge

- Every course consists of several (at least one) sessions.
- Every session is given in a room with capacity at least as high as the number of students enrolled for that session.
- Instructors can only teach one session simultaneously.
- ...

$\forall c[course] : \exists s[session] : partOf(s, c)$

2. Modelling knowledge

- Every course consists of several (at least one) sessions.
- Every session is given in a room with capacity at least as high as the number of students enrolled for that session.
- Instructors can only teach one session simultaneously.
- ...

$\forall c[course] : \exists s[session] : partOf(s, c)$

$\forall s[session] : capacityOf(locationOf(s)) \geq \#\{st[student] : enrolledIn(st, s)\}.$

2. Modelling knowledge

- Every course consists of several (at least one) sessions.
- Every session is given in a room with capacity at least as high as the number of students enrolled for that session.
- Instructors can only teach one session simultaneously.
- ...

$\forall c[course] : \exists s[session] : partOf(s, c)$

$\forall s[session] : capacityOf(locationOf(s)) \geq \#\{st[student] : enrolledIn(st, s)\}.$

$\forall s_1, s_2[session] : teacherOf(s_1) = teacherOf(s_2) \Rightarrow noOverlap(s_1, s_2).$

3. Choosing inference methods

- Verifying whether a given schedule is valid
  - Model checking
- Finding a valid schedule
  - (Optimal) model expansion
- Updating scheduling when unexpected events occur
  - Revision
- Interactively (with a domain expert) crafting a schedule
  - E.g., propagation, explanation

4. Running the system

IDP Web-IDE File Configure main() Run Help IDP Homepage

```
  }

  Diploma = {
    anne,aardrijkskunde;
    bert,biologie; bert,lo;
    chris,chemie;
    dirk,economie;
    erik,engels; erik,nederlands;
    frans,frans; frans,latijn;
    gert,fysica; gert,wiskunde;
    hans,geschiedenis;
    ilona,godsdienst;
    jenny,informatica;
  }

  LesurenPerWeek = {
    wetwisk,aardrijkskunde  ↦ 1;  wetwisk,biologie      ↦ 2;
    wetwisk,chemie          ↦ 2;  wetwisk,economie      ↦ 0;
    wetwisk,engels          ↦ 3;  wetwisk,frans         ↦ 4;
    wetwisk,fysica          ↦ 2;  wetwisk,geschiedenis  ↦ 2;
    wetwisk,godsdienst      ↦ 2;  wetwisk,informatica   ↦ 1;
    wetwisk,latijn          ↦ 0;  wetwisk,lo            ↦ 2;
    wetwisk,nederlands      ↦ 4;  wetwisk,wiskunde      ↦ 6;
  }

} /* structure S */


theory T : V {
  // Een leerkracht geeft les aan een groep op een bepaald tijdstip
  { GeeftLes(Docent(groep,vak),groep,dag,uur) ← Les(vak,groep,dag,uur). }

  // Een vak moet per week het juiste aantal lesuren gegeven worden
  ∀ vak groep : #{ dag uur : Les(vak,groep,dag,uur) }
                = LesurenPerWeek(groep,vak).

  // Een leerkracht kan op een tijdstip maar 1 les geven
  ∀ leerkracht dag uur : #{ groep : GeeftLes(leerkracht,groep,dag,uur) } < 2.

  // Een leerkracht moet een diploma hebben voor de vakken die hij/zij geeft
  ∀ leerkracht groep vak : Docent(groep,vak) = leerkracht
                           ⇒ Diploma(leerkracht,vak).

  // Geen les op woensdagnamiddag
  ¬(∃ vak groep uur : Les(vak,groep,Woensdag,uur) ∧ uur > 4).

  // Een groep kan per tijdstip maar 1 les krijgen
  ∀ groep dag uur : #{ vak : Les(vak,groep,dag,uur) } < 2.
}


procedure main(){
    print(modelexpand(T,S)[1])
}
```

```
Warning: Verifying and/or autocompleting structure S
Warning: Variables uur dag vak groep are not quantified. At
tmp2471.tidp:91:5
Warning: Derived sort Dag for variable dag. At tmp2471.tidp:94:20
Warning: Derived sort Groep for variable groep. At tmp2471.tidp:94:9
Warning: Derived sort Uur for variable uur. At tmp2471.tidp:94:24
Warning: Derived sort Vak for variable vak. At tmp2471.tidp:94:5
Warning: Derived sort Dag for variable dag. At tmp2471.tidp:98:16
Warning: Derived sort Groep for variable groep. At
tmp2471.tidp:98:29
Warning: Derived sort Leerkracht for variable leerkracht. At
tmp2471.tidp:98:5
Warning: Derived sort Uur for variable uur. At tmp2471.tidp:98:20
Warning: Derived sort Groep for variable groep. At
tmp2471.tidp:101:16
Warning: Derived sort Dag for variable dag. At tmp2471.tidp:108:11
Warning: Derived sort Groep for variable groep. At
tmp2471.tidp:108:5
Warning: Derived sort Uur for variable uur. At tmp2471.tidp:108:15
Warning: Derived sort Vak for variable vak. At tmp2471.tidp:108:24
structure  : V {
  Dag = { 1..5 }
  Groep = { "wetwisk" }
  Leerkracht = { "anne"; "bert"; "chris"; "dirk"; "erik"; "frans"; "ge
  Lesuren = { 0..8 }
  Uur = { 1..7 }
  Vak = { "aardrijkskunde"; "biologie"; "chemie"; "economie"; "engels"
  Diploma = { "anne","aardrijkskunde"; "bert","biologie"; "bert","lo";
  GeeftLes = { "anne","wetwisk",4,5; "bert","wetwisk",2,5; "bert","wet
  Les = { "aardrijkskunde","wetwisk",4,5; "biologie","wetwisk",2,5; "b
  Volgt = { "wetwisk","aardrijkskunde"; "wetwisk","biologie"; "wetwisk
  Dinsdag = 2
  Docent = { "wetwisk","aardrijkskunde"-
>"anne"; "wetwisk","biologie"->"bert"; "wetwisk","chemie"-
>"chris"; "wetwisk","economie"->"dirk"; "wetwisk","engels"-
>"erik"; "wetwisk","frans"->"frans"; "wetwisk","fysica"-
>"gert"; "wetwisk","geschiedenis"->"hans"; "wetwisk","godsdienst"-
>"ilona"; "wetwisk","informatica"->"jenny"; "wetwisk","latijn"-
>"frans"; "wetwisk","lo"->"bert"; "wetwisk","nederlands"-
>"erik"; "wetwisk","wiskunde"->"gert" }
  Donderdag = 4
  LesurenPerWeek = { "wetwisk","aardrijkskunde"-
>1; "wetwisk","biologie"->2; "wetwisk","chemie"-
>2; "wetwisk","economie"->0; "wetwisk","engels"-
>3; "wetwisk","frans"->4; "wetwisk","fysica"-
>2; "wetwisk","geschiedenis"->2; "wetwisk","godsdienst"-
>2; "wetwisk","informatica"->1; "wetwisk","latijn"-
>0; "wetwisk","lo"->2; "wetwisk","nederlands"-
>4; "wetwisk","wiskunde"->6 }
  Maandag = 1
  Vrijdag = 5
  Woensdag = 3
}
```

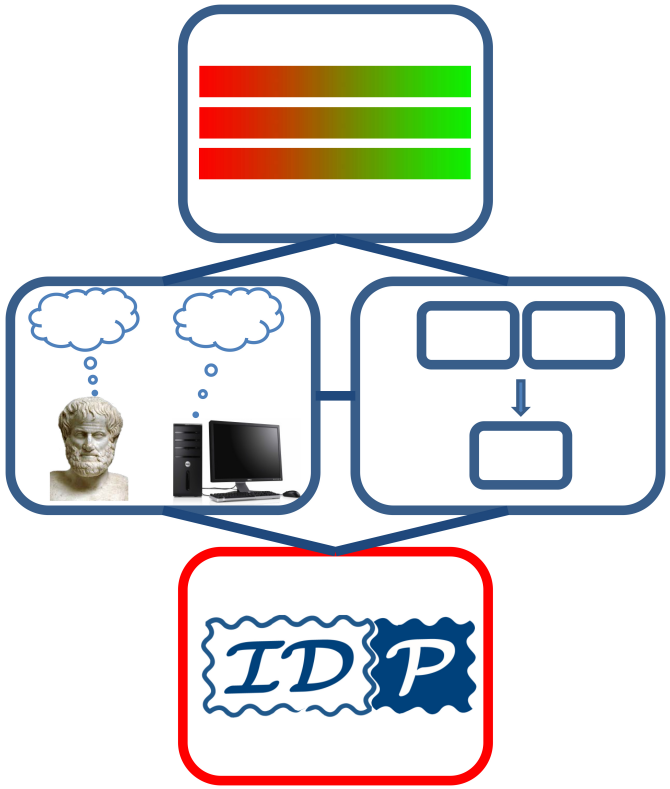
Content

Various research topics

1. The IDP Knowledge Base System
2. Symmetry
3. The Linear Time Calculus
4. Applications of KRR in Data Mining
5. FO(C)
6. Knowledge Compilation
7. Groundedness

1. The IDP Knowledge Base System

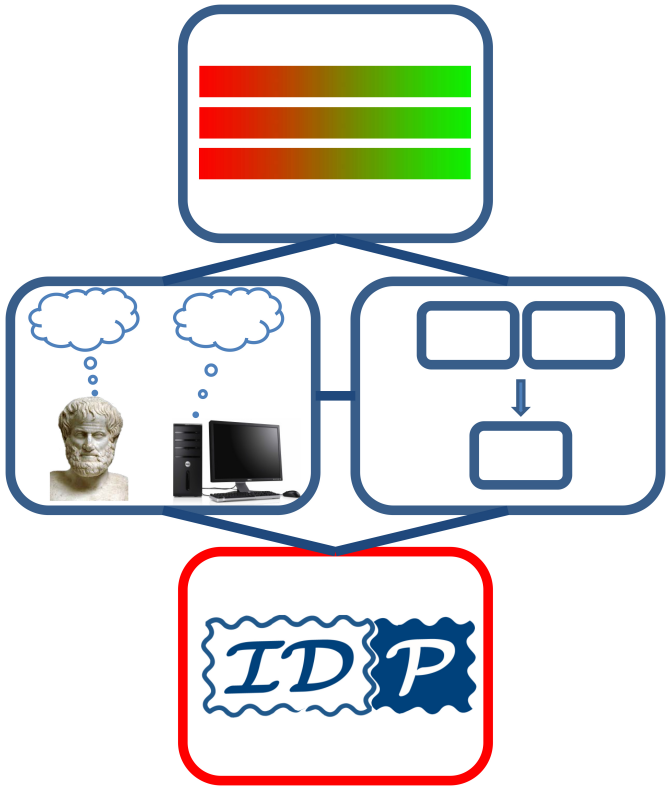
- The IDP language: rich extension of FO
- Many inferences methods
- Ground-and-solve



- Predicate Logic as a Modelling Language: The IDP System B. De Cat, B. Bogaerts, M. Bruynooghe and M. Denecker, 2015 (accepted)
- Meta-level Representations in the IDP Knowledge Base System: Towards Bootstrapping Inference Engine Development B. Bogaerts, J. Jansen, B. De Cat, G. Janssens, M. Bruynooghe and M. Denecker, LaSh, 2014
- MiniSat(ID) for Satisfiability Checking and Constraint Solving B. De Cat, B. Bogaerts and M. Denecker, ALP Newsletter, 2014.
- Model Expansion in the Presence of Function Symbols Using Constraint Programming B. De Cat, B. Bogaerts, J. Devriendt and M. Denecker, ICTAI, 2013

2. Symmetry

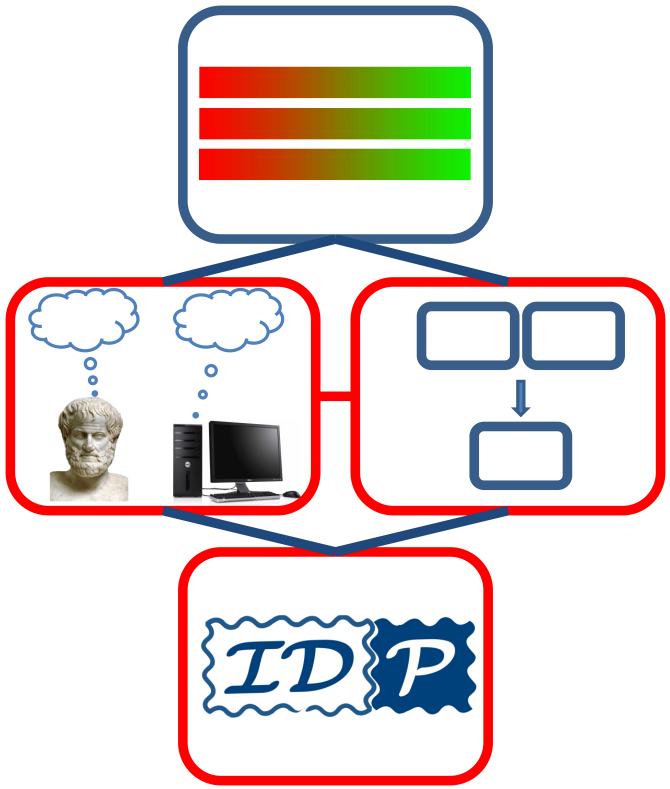
- Detecting and exploiting symmetries
- Increased efficiency



- BreakIDGlucose: On The Importance of Row Symmetry J. Devriendt, B. Bogaerts and M. Bruynooghe, CSPSAT, 2014
- Symmetry Propagation: Improved Dynamic Symmetry Breaking in SAT J. Devriendt, B. Bogaerts, C. Mears, B. De Cat and M. Denecker, ICTAI, 2012

3. The Linear Time Calculus

- Language: temporal version of the IDP language for modelling dynamic domains
- Inferences: progression, simulation, verification of invariants, ...
- Implementation in IDP

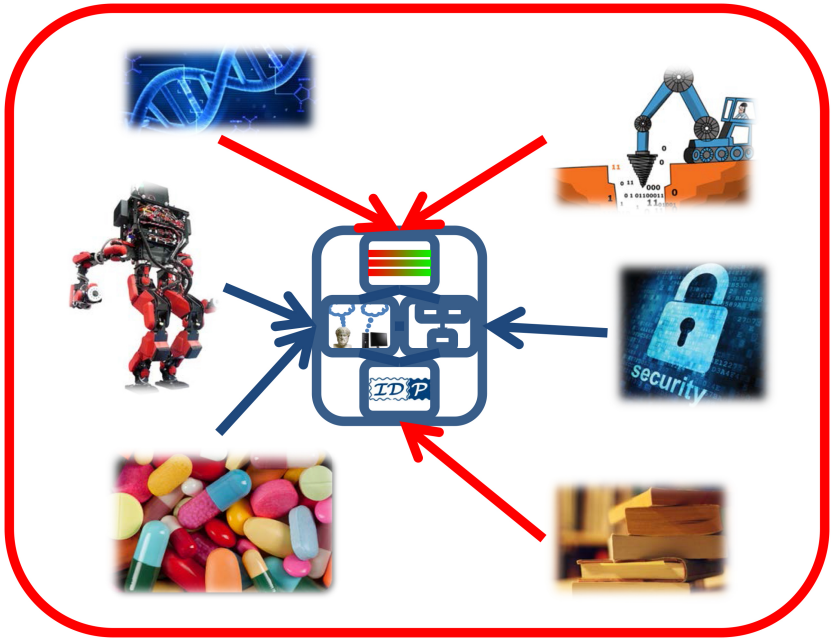


- Simulating Dynamic Systems Using Linear Time Calculus Theories B. Bogaerts, B. De Cat, J. Jansen, M. Bruynooghe, B. De Cat, J. Vennekens and M. Denecker, TPLP 14(4-5), 2014.

4. Applications of KRR in Machine Learning and Data Mining

Applications of IDP:

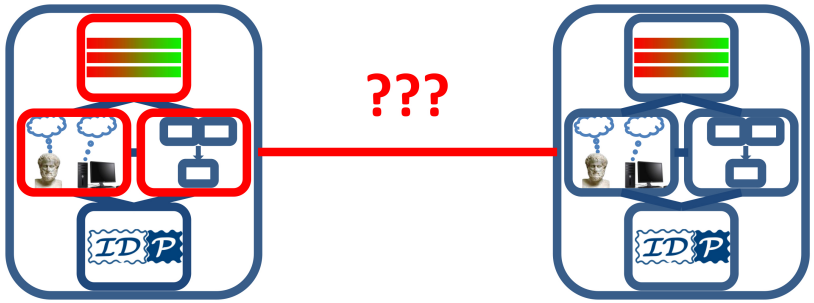
- Stemmatology
- Phylogenetic trees
- Finite state automata



- Predicate Logic as a Modeling Language: Modeling and Solving some Machine Learning and Data Mining Problems with IDP^3 M. Bruynooghe, H. Blockeel, B. Bogaerts, B. De Cat, S. De Pooter, J. Jansen, A. Labarre, J. Ramon, M. Denecker and S. Verwer, TPLP, 2015. (accepted)
- Analyzing manuscript traditions using constraint-based data mining T. Andrews, H. Blockeel, B. Bogaerts, M. Bruynooghe, M. Denecker, S. De Pooter, C. Macé and J. Ramon, CoCoMiLe, 2012.
- Modeling Machine Learning and Data Mining Problems with $\mathrm{FO}(\cdot)$ H. Blockeel, B. Bogaerts, M. Bruynooghe, B. De Cat, S. De Pooter, M. Denecker, A. Labarre, J. Ramon and S. Verwer, ICLP Technical Communication, 2012.

5. FO(C)

- Language for expressing complex cause-effect relationships
- Studied complexity of inference in FO(C)
- (Inference by) reduction of FO(C) to the IDP language

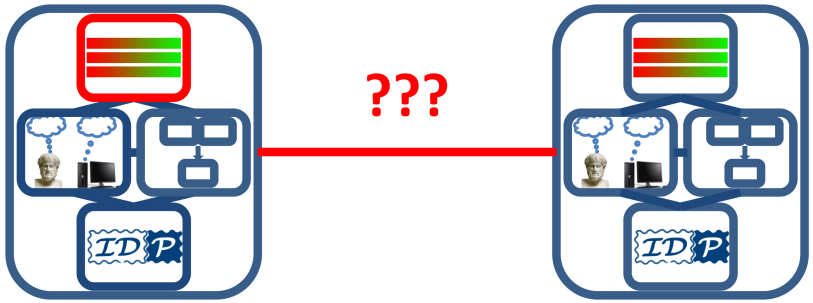


- Inference in the FO(C) Modelling Language B. Bogaerts, J. Vennekens, M. Denecker and J. Van den Bussche, ECAI, 2014
- FO(C) and Related Modelling Paradigms B. Bogaerts, J. Vennekens, M. Denecker and J. Van den Bussche, NMR, 2014
- FO(C): A Knowledge Representation Language of Causality B. Bogaerts, J. Vennekens, M. Denecker and J. Van den Bussche, ICLP technical communication, 2014

6. Knowledge Compilation

Knowledge Compilation:

- Compares languages by studying complexity of inference and relative succinctness
- Provides transformations between these languages



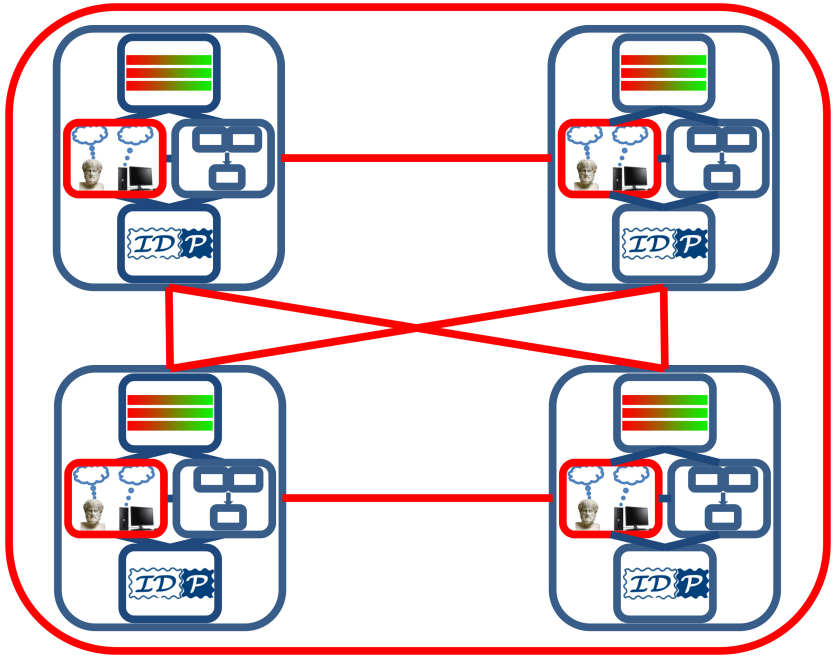
Our addition:

- Transformation from logic programs into Boolean circuits
  - Efficient (polynomial)
  - No auxiliary variables
  - Exploits the constructive nature of well-founded semantics
  - Enables approximate inference
- Theory is developed in *Approximation Fixpoint Theory*

- Knowledge compilation of logic programs using approximation fixpoint theory B. Bogaerts, G. Van den Broeck, TPLP 15(4-5), 2015 (conditionally accepted).

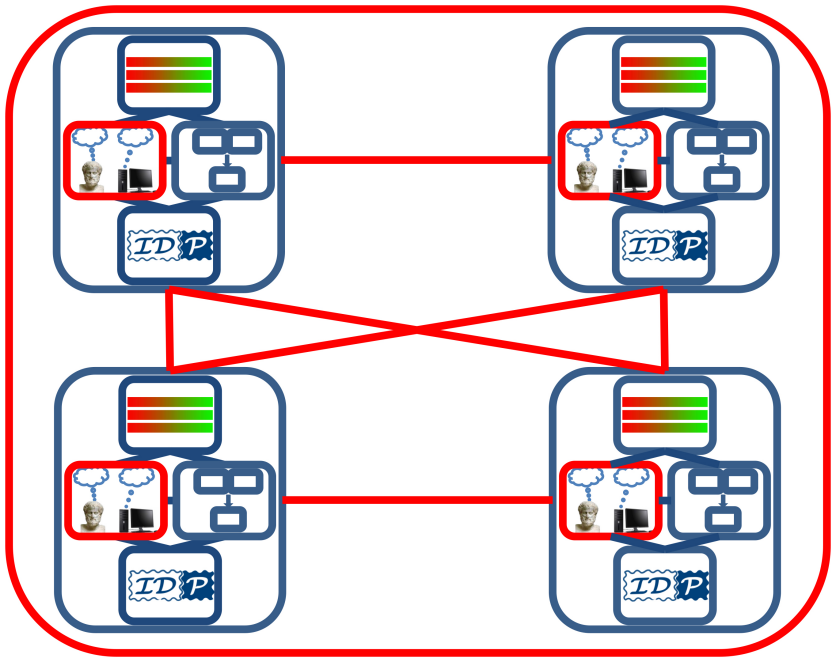
Approximation Fixpoint Theory (AFT)

- Abstract algebraical framework (Denecker, Marek, Truszczyński)
- Describes semantics ...
  - Supported semantics
  - Stable semantics
  - Kripke-Kleene semantics
  - Well-founded semantics
- ... of different logics
  - logic programming
  - autoepistemic logic (AEL)
  - default logic
  - Dung's argumentation frameworks
  - abstract dialectical frameworks



7. Groundedness

- Extension of AFT
- (Partial) grounded fixpoints: abstract concept that captures common intuitions from several fields
- Used to classify semantics in these fields
- Refined constructive semantics (for AEL)



- Grounded Fixpoints and Their Applications in Knowledge Representation B. Bogaerts, J. Vennekens and M. Denecker, Artificial Intelligence 224, 2015.
- Partial Grounded Fixpoints B. Bogaerts, J. Vennekens and M. Denecker, IJCAI, 2015.
- Grounded Fixpoints B. Bogaerts, J. Vennekens and M. Denecker, AAAI, 2015.
- On Well-Founded Set-Inductions and Locally Monotone Operators B. Bogaerts, J. Vennekens and M. Denecker. (unpublished)

Content

Content

Groundedness

- Different research domains:
  - Logic programming,
  - Autoepistemic logic,
  - Default logic,
  - Dung's argumentation frameworks,
  - Abstract dialectical frameworks.
- Similar intuitions: facts (or models)
  - are *grounded*,
  - are supported by *cycle-free* arguments,
  - are not *unfounded*,
  - can be built *from the ground up*.

Logic Programming

- Completion semantics (Clark, 1978) ungrounded models
- Perfect model semantics (Przymusinski, 1988)
- Well-founded semantics (Van Gelder et. al., 1988)
- Stable semantics (Gelfond and Lifschitz, 1988)

$$\{p \leftarrow p\}: \text{ self-supporting model } \{p\}$$

Autoepistemic Logic (AEL)

- Expansion semantics (Moore, 1985) ungrounded expansions
- "Honest theories" (Halpern and Moses, 1985)
- Moderately grounded expansions (Konolige, 1988)
- Strongly grounded expansions (Konolige, 1988)
- Constructive tightly grounded autoepistemic reasoning (Niemelä, 1991)
- Stable semantics (Denecker, Marek and Truszczyński, 2000)
- Well-founded semantics (Denecker, Marek and Truszczyński, 2000)

$\{Kp \Rightarrow p\}$: self-supporting expansions in which p is known

Grounded fixpoints

- What is the problem with logic programs as $\{p \leftarrow p\}$ or AEL theories such as $\{Kp \Rightarrow p\}$?
- Algebraical study (fixpoint theory)
- Application to logic programming, autoepistemic logic, default logic, Dung's argumentation frameworks and abstract dialectical frameworks

Grounded fixpoints

Given

- *Complete lattice* $\langle L, \leq \rangle$*: every set* $S \subseteq L$ *has a least upper bound* $\bigvee S$ *and a greatest lower bound* $\bigwedge S$
- *Operator* $O : L \to L$

Definition (Grounded)

We call $x \in L$ *grounded* for O if for each $v \in L$ such that $O(x \wedge v) \leq v$, it holds that $x \leq v$.

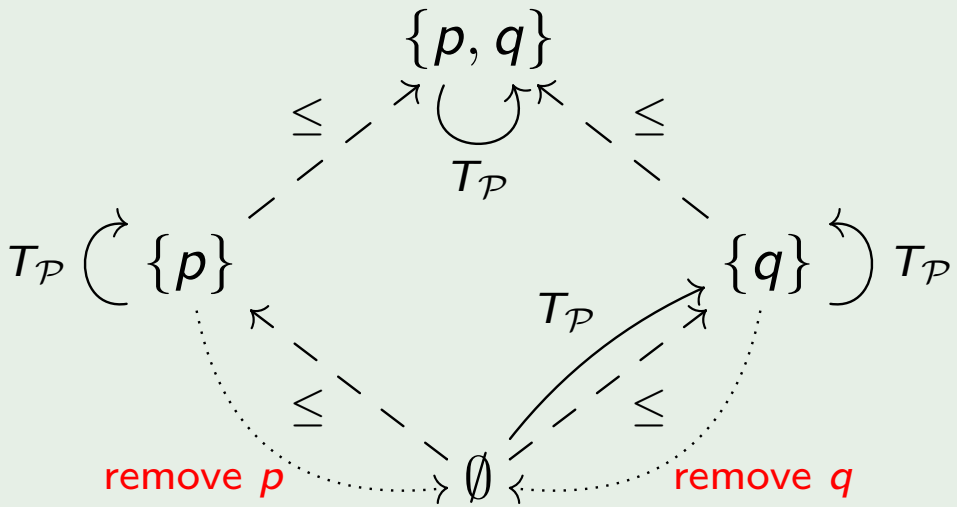
Intuition

- $L = 2^F$, $\leq = \subseteq$
- *In this case:* x *is grounded for* O *if it only contains facts that are sanctioned by* O*: whenever we remove facts from* x*, at least one of them is rederived.*

Groundedness

Example

$$\mathcal{P} = \left\{ \begin{array}{l} p \leftarrow p. \\ q \leftarrow \neg p \vee q. \end{array} \right\}$$



Properties of grounded fixpoints

Proposition

All grounded fixpoints of O are minimal fixpoints of O.

Proposition

A monotone operator has exactly one grounded fixpoint, namely its least fixpoint.

Proposition

All A-stable fixpoints of O are grounded fixpoints of O.

Proposition

The well-founded fixpoint of a symmetric approximator A of O approximates all grounded fixpoints of O.

Grounded fixpoints in logic programming

- Study of existing semantics: stable and two-valued well-founded semantics are "grounded"
- Closely related to unfounded sets
- Grounded fixpoints induce a new semantics
  - Two-valued
  - Purely algebraical
  - Simple
  - Easily extensible
- Study complexity

Other applications of grounded fixpoints

- Autoepistemic logic
- Default logic
- Dung's argumentation frameworks
- Abstract dialectical frameworks

Results

- Abstract algebraical definition of groundedness
- Studied relation with other types of fixpoints from AFT
- Corresponds to intuitions in many different domains
- Study of existing semantics
- New induced semantics with attractive properties

Content

Motivation

- Grounded fixpoints: limited to two-valued setting
- Sometimes partial fixpoints (three-valued models) are interesting:
  - Central objects of study
  - More general
  - As an analysis tool (e.g., in debugging)

Partial Grounded Fixpoints

Definition

Let A be an approximator of O. A point $(x, y) \in L^c$ is *A-grounded* if for every $v \in L$ with $A(x \wedge v, y \wedge v)_2 \leq v$, also $y \leq v$.

Properties of partial grounded fixpoints

Proposition

If A is a symmetric approximator of O, then x is grounded for O if and only if (x, x) is A-grounded.

Proposition

All consistent (partial) A-stable fixpoints are A-grounded.

Theorem

The well-founded fixpoint of a symmetric approximator A of O is the least precise A-grounded fixpoint.

Proposition

If A and B are approximators of O and $A \leq_p B$, then all consistent B-grounded points are also A-grounded.

Application (Logic Programming)

- Study semantical relationship with well-founded and stable models
- Closely related to unfounded sets
- Study complexity

Content

Motivation

- Denecker and co-authors
  - strongly argued for a constructive semantics for various non-monotonic logics (LP, AEL, ...)
  - claimed that the well-founded semantics is always strong enough to construct the intended model for "sound" theories
- Claim was proven for logic programs (viewed as inductive definitions) (Denecker and Vennekens, 2014)
- Claim was made (without proof) for AEL (Denecker, Marek and Truszczyński, 2011)
- However ...

Motivation

- ... still in 2011, Hanne Vlaeminck came up with this example:

Example

$$\{q \Leftrightarrow \neg Kp, r \Leftrightarrow \neg Kq\}.$$

- Clear intended model
- Well founded semantics fails to identify it (construction is too weak)
- The unique grounded fixpoint

Goal

Refine well-founded semantics to

- *Maintain advantages of a constructive semantics*
- *Also capture this kind of examples*

Results

- Generalisation (AEL): monotonically stratified theories
- Generalisation (AFT): locally monotone operator
- New constructive semantics (AFT): well-founded set-semantics based on groundedness

Theorem

All locally monotone operators O have a unique grounded fixpoint; this fixpoint is the unique element in the well-founded set of O.

Questions?

?