A Knowledge Base System project for FO(.)

Marc Denecker
Knowledge Representation & Reasoning (KRR)
Katholieke Universiteit Leuven

July 12, 2009

Acknowledgements

Maurice Bruynooghe, Victor Marek, Eugenia Ternovska, Mirek Truszczyński, Joost Vennekens

(Ex-)Members of KRR
Stephen Bond, Maarten Mariën, Nikolay Pelov, Hou Ping, Bert Vannuffelen, Hanne Vlaeminck, Johan Wittocx

Structure of the talk

1. The conceptual framework
   - The KBS computational paradigm
   - FO(·): an integration of classical first order logic (FO) and Logic Programming (LP)
   - Common sense KR in FO(·)
2. Implementation: progress report

Structure of the talk

- The conceptual framework
  - History and motivation of this work

Logic Programming: the early days

The enthousiasm of the early days

"Never had I experienced such ease in getting a complex program coded and running"

– D.H.D. Warren in Foreword to The Art of Prolog

Logic Programming: the early days

- 1980-1990: LP split up in separate scientific fields
  - LP for Programming
  - LP for Knowledge Representation (KR)

LP can make a contribution to KR
Kowalski, Sergot, Gelfond, Lifschitz, ...

Logic Programming: the early days

- 1980-1990: LP split up in separate scientific fields
  - LP for Programming
  - LP for Knowledge Representation (KR)

> LP can make a contribution to KR
> Kowalski, Sergot, Gelfond, Lifschitz, ...

- $\pm$1990: Disappointment

> LP has strong limitations as a KR language
> Representation of incomplete knowledge

Logic Programming: the early days

- 1980-1990: LP split up in separate scientific fields
  - LP for Programming
  - LP for Knowledge Representation (KR)

> LP can make a contribution to KR
> Kowalski, Sergot, Gelfond, Lifschitz, ...

- $\pm$1990: Disappointment

> LP has strong limitations as a KR language
> Representation of incomplete knowledge

- 1990-1995: Two KR fields growing out of LP:
  - Abductive Logic Programming (ALP)
  - Answer Set Programming (ASP)

Abductive Logic Programming

- 1986: pioneered by Eshghi, Kowalski, Shanahan, Kakas, Missiaen, . . . (I entered the field in 1990 as a Ph.D.)

> We experienced ALP as a rich KR language resolving some of LP's main problems for KR.

Abductive Logic Programming

- 1986: pioneered by Eshghi, Kowalski, Shanahan, Kakas, Missiaen, . . . (I entered the field in 1990 as a Ph.D.)

  We experienced ALP as a rich KR language resolving some of LP's main problems for KR.

- 1994: Research visit at University of Toronto (Ray Reiter)
  - (Ray Reiter championed KR using FO)
  - I gave a seminar:

    Showing ALP's use for KR by mapping Situation Calculus in ALP

Abductive Logic Programming

- 1986: pioneered by Eshghi, Kowalski, Shanahan, Kakas, Missiaen, . . . (I entered the field in 1990 as a Ph.D.)

  We experienced ALP as a rich KR language resolving some of LP's main problems for KR.

- 1994: Research visit at University of Toronto (Ray Reiter)
  - (Ray Reiter championed KR using FO)
  - I gave a seminar:

    Showing ALP's use for KR by mapping Situation Calculus in ALP

  - Ray was not impressed :-)
  - FO versus LP&ALP : a clash of concepts

Goal

How to clarify (A)LP's KR contribution to the broader KR community?

- It does not suffice to show that a problem can also be represented in ALP or ASP
- FO-KR researchers do not want to give up FO

Goal

How to clarify (A)LP's KR contribution to the broader KR community?

- It does not suffice to show that a problem can also be represented in ALP or ASP
- FO-KR researchers do not want to give up FO

Strategy

- Add the key-features of ALP to FO
- Integrating (A)LP into FO

A long process

Abductive Logic Programming

$\Downarrow$

Incomplete Logic Programming (1993)

$\Downarrow$

Open Logic Programming-FOL (OLP-FOL)(1996)

$\Downarrow$

ID-logic (2000)

$\Downarrow$

FO(ID)

Integrating FO and LP: Why?

Integrating FO and LP: Why?

The study of logic and Knowledge Representation is scattered over many fields. There are significant, but poorly understood overlaps. Science should bring clarity and understanding in here.

Integrating FO and LP: Why?

Integrating FO and LP: Why?

> The study of logic and Knowledge Representation is scattered over many fields. There are significant, but poorly understood overlaps. Science should bring clarity and understanding in here.

A tight (non-hybrid) integration is the best solution to explain contributions.

> In a tight integration, when fundamental concepts and terminologies are brought on a par, the contributions of different components will emerge.

Why FO?

FO is the base KR logic

Why FO?

- FO's historical roots are in KR

"All our ideas are compounded from a very small number of simple ideas, which form the alphabet of human thought."
"Complex ideas proceed from these simple ideas by a uniform and symmetrical combination, analogous to arithmetical multiplication."

– Leibniz

Why FO?

- FO's historical roots are in KR

  "All our ideas are compounded from a very small number of simple ideas, which form the alphabet of human thought."
  "Complex ideas proceed from these simple ideas by a uniform and symmetrical combination, analogous to arithmetical multiplication."

  – Leibniz

- De Morgan, Boole, Frege studied the key combination operators:

  conjunction, disjunction, negation, universal and existential quantification.

Why FO?

- A claim:

> These operators are fundamental for KR. Every "interesting" KR-language will have a substantial overlap with FO, in one form or the other.

Why FO?

- A claim:

  These operators are fundamental for KR. Every "interesting" KR-language will have a substantial overlap with FO, in one form or the other.

- E.g. constraints in ASP

Integrating FO and (A)LP: Challenges

- Many things to reconcile:

(Abductive) Logic Programming	Classical logic (FO)
Abduction	Deduction
Operational semantics	-
Non-monotone	Monotone
NAF `not`	Classical negation $\neg$
Herbrand structures	Arbitrary structures
Domain Closure Assumption (DCA)	No domain assumption
Constructors (UNA)	Free functions
Belief sets	Possible worlds
...	...

Integrating FO and (A)LP: Challenges

- Many things to reconcile:

(Abductive) Logic Programming	Classical logic (FO)
Abduction	Deduction
Operational semantics	-
Non-monotone	Monotone
NAF `not`	Classical negation $\neg$
Herbrand structures	Arbitrary structures
Domain Closure Assumption (DCA)	No domain assumption
Constructors (UNA)	Free functions
Belief sets	Possible worlds
...	...

Integrating FO and LP: Challenges

Reconciling a deductive logic with an abductive logic?

Integrating FO and LP: Challenges

Reconciling a deductive logic with an abductive logic?

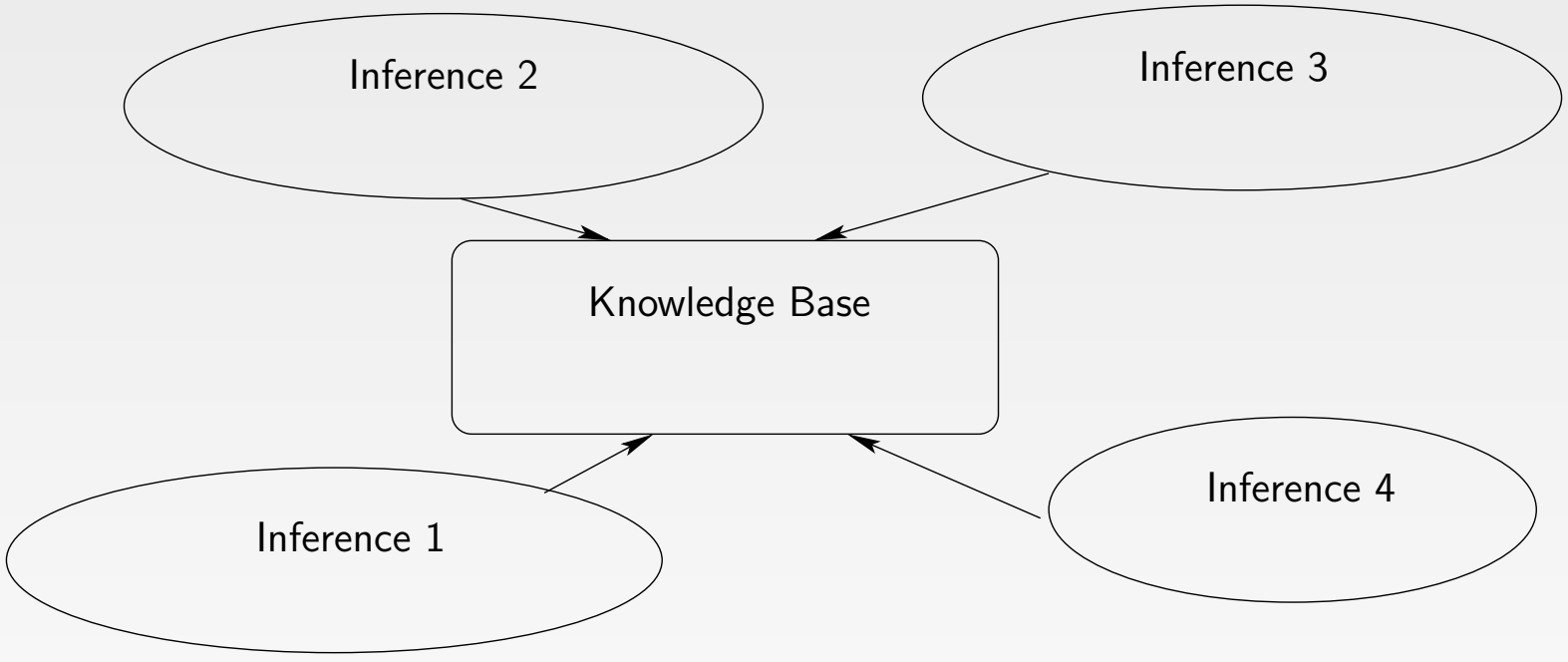
$$\Downarrow$$

The Knowledge Base System paradigm

Structure of the talk

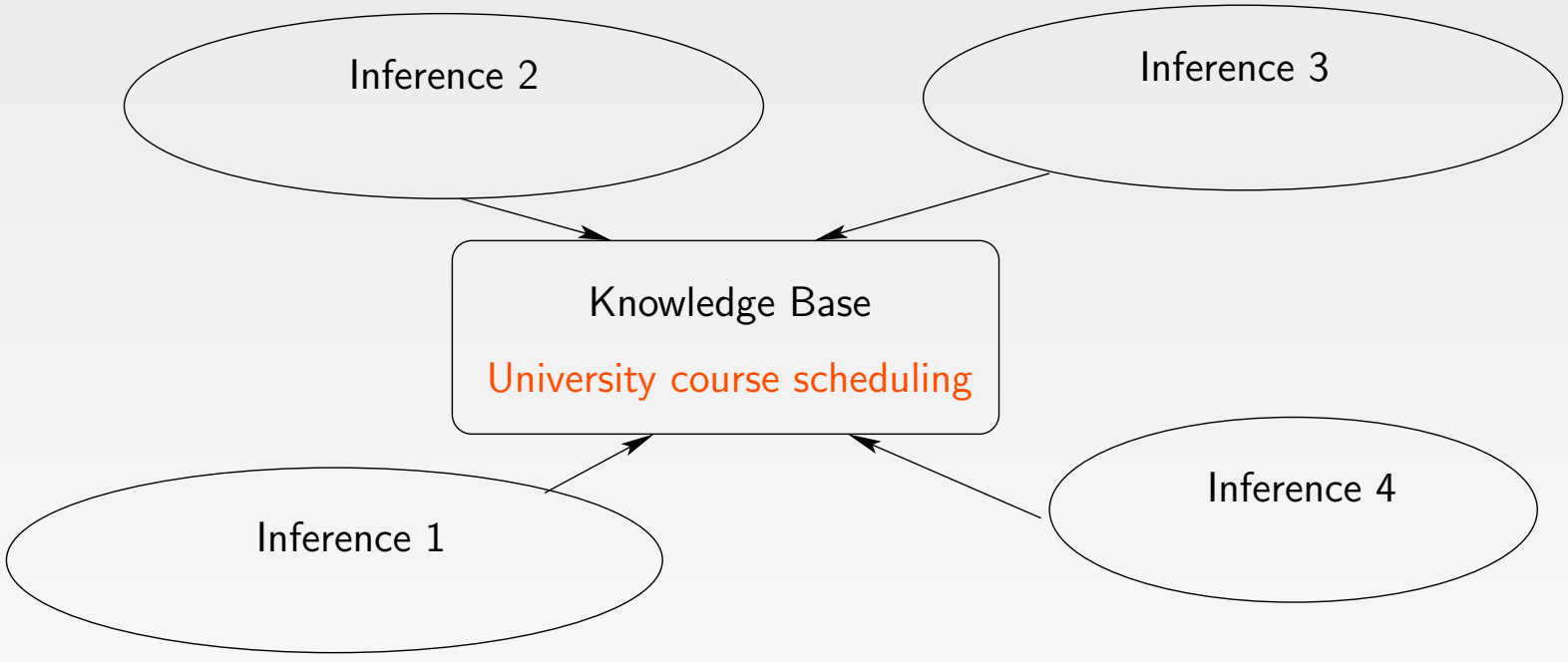
- The conceptual framework
  - History and motivation of this work
  - The Knowledge Base System paradigm

A Knowledge Base System (KBS)



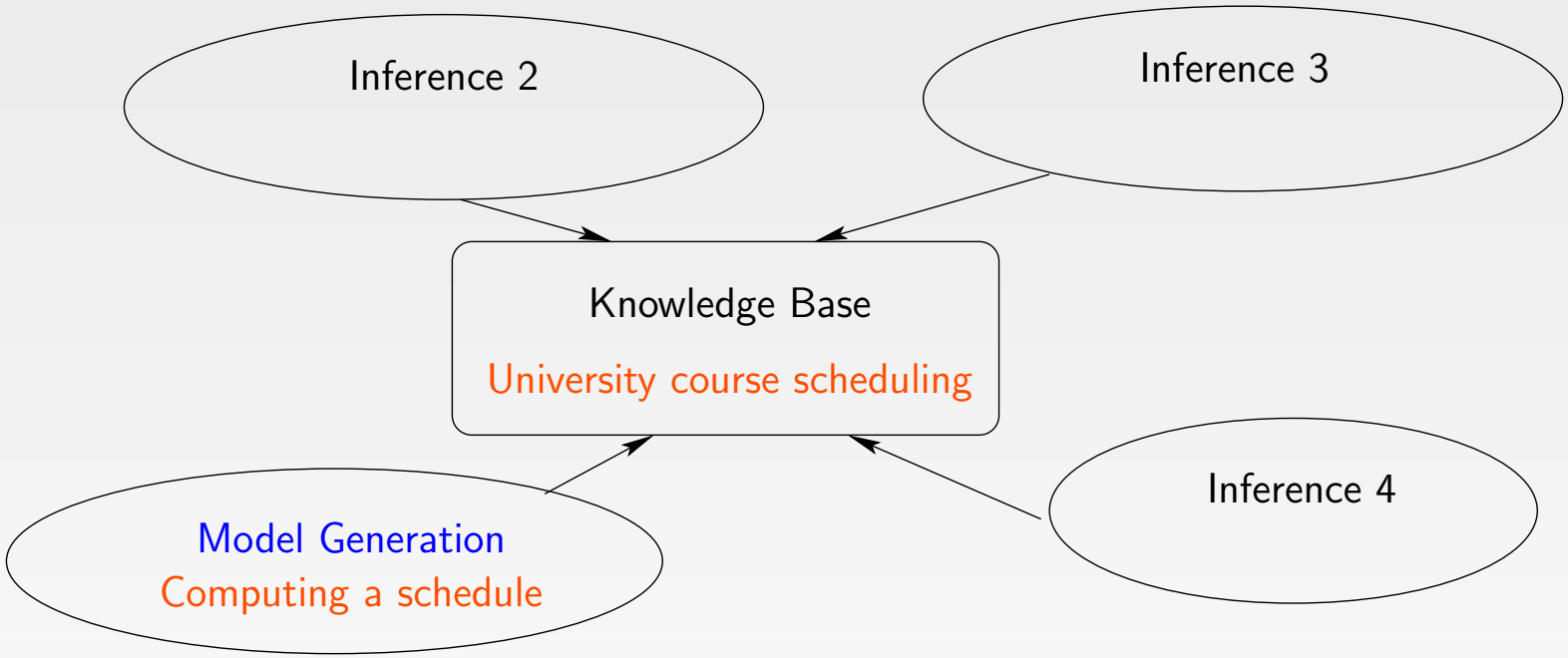
- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System (KBS)



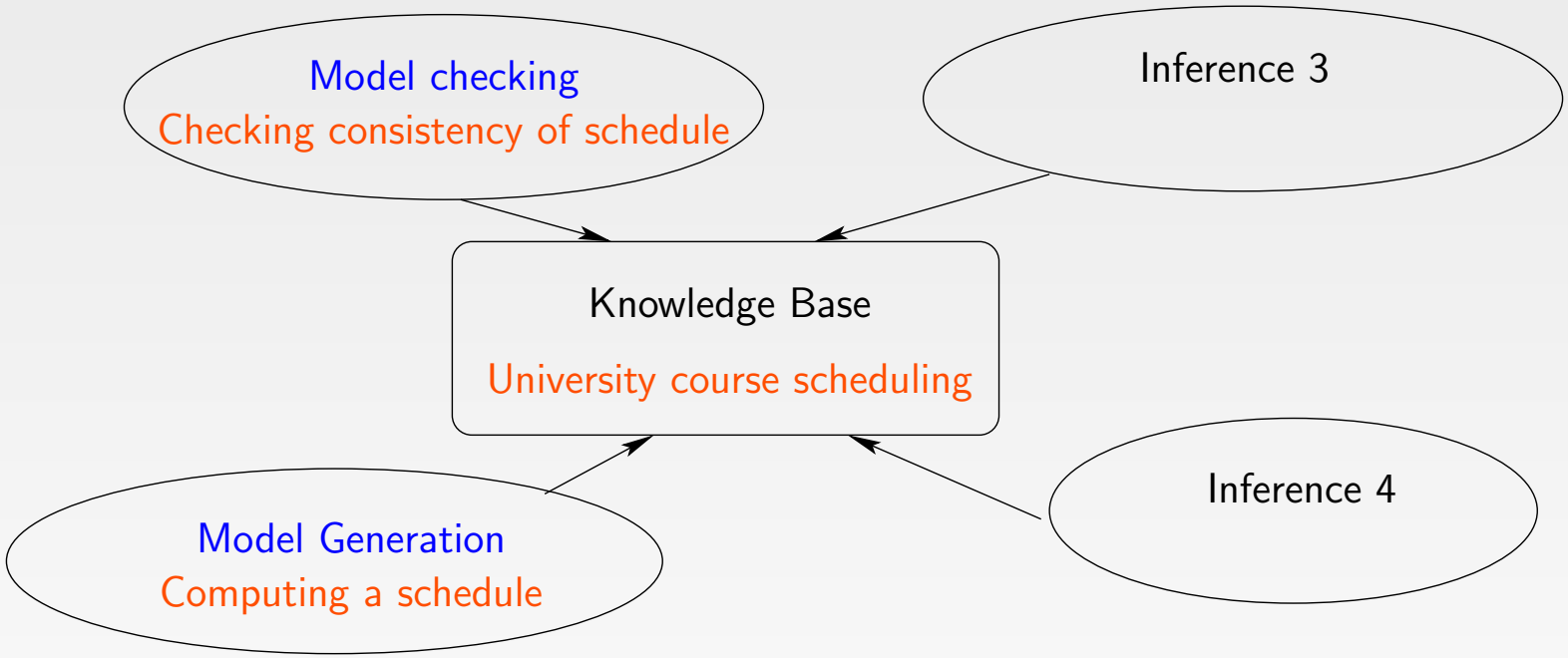
- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System (KBS)



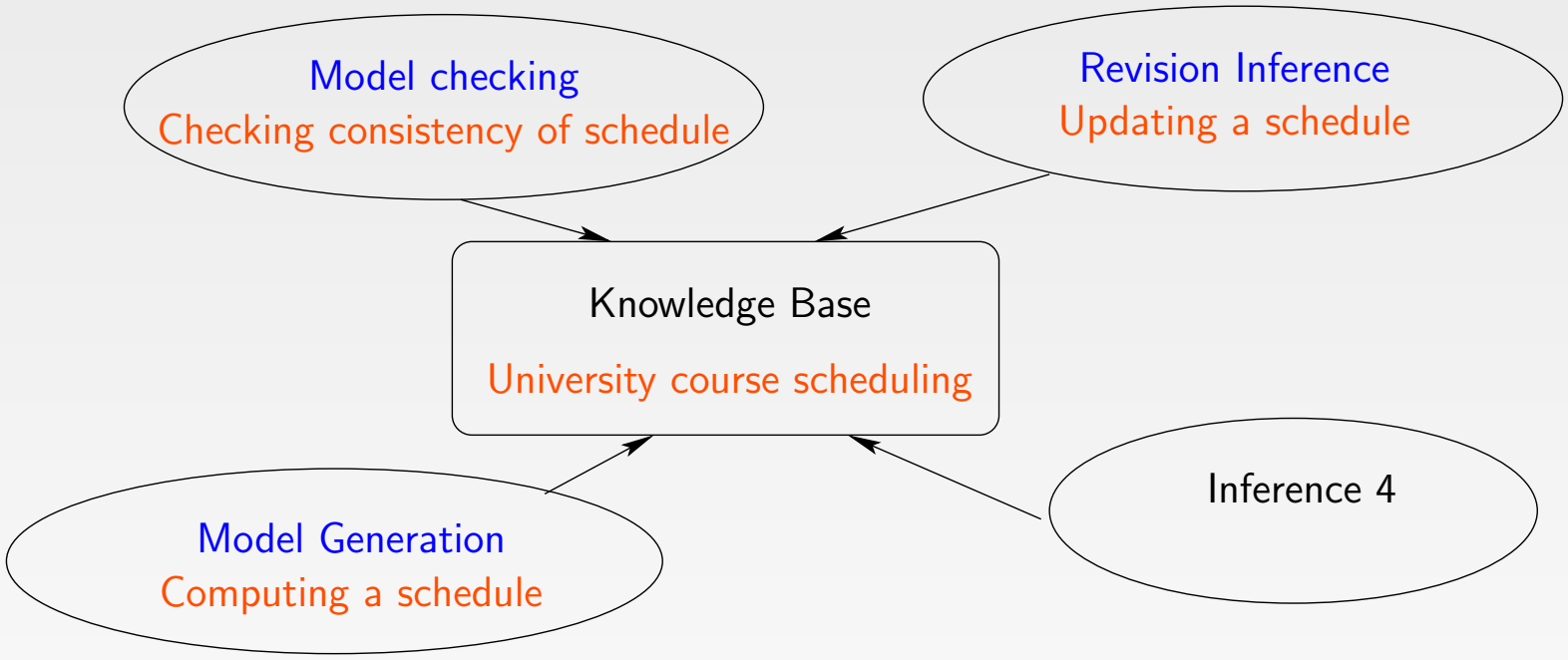
- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System (KBS)



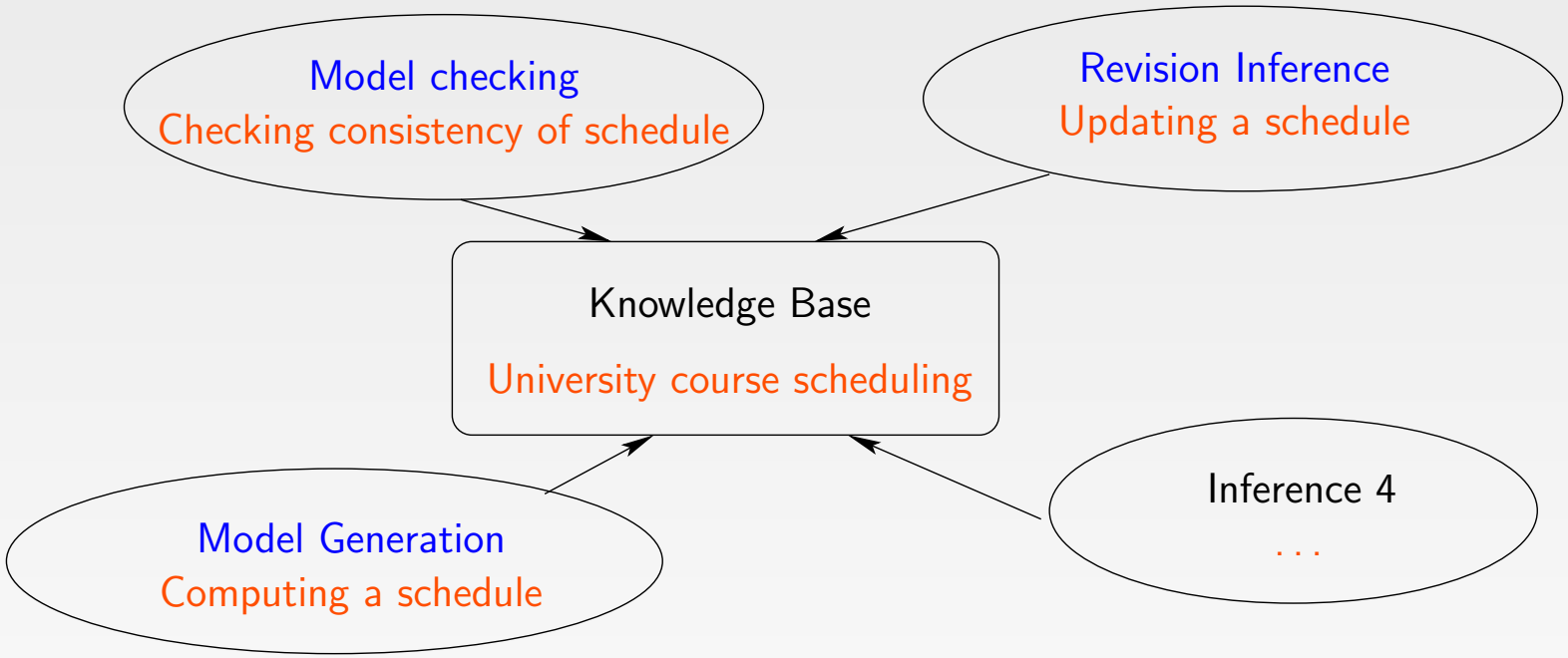
- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System (KBS)



- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System (KBS)



- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference to solve different types of tasks.

The KBS computational paradigm

- A paradigm for declarative problem solving

The KBS computational paradigm

- A paradigm for declarative problem solving
- Differs from declarative programming paradigms:
  - ASP, LP, CLP, . . . :
    - A declarative language + unique form of inference
    - A declarative program encodes a solution for a problem

The KBS computational paradigm

- A paradigm for declarative problem solving
- Differs from declarative programming paradigms:
  - ASP, LP, CLP, . . . :
    - A declarative language + unique form of inference
    - A declarative program encodes a solution for a problem
  - The KBS paradigm
    - A declarative language + multiple forms of inference
    - A KB is merely a specification of domain knowledge, which can be reused for solving different types of problems

The KBS computational paradigm

- A paradigm for declarative problem solving
- Differs from declarative programming paradigms:
  - ASP, LP, CLP, . . . :
    - A declarative language + unique form of inference
    - A declarative program encodes a solution for a problem
  - The KBS paradigm
    - A declarative language + multiple forms of inference
    - A KB is merely a specification of domain knowledge, which can be reused for solving different types of problems
- A KB language is a declarative language pur sang.

Status of KBS paradigm

- The KBS paradigm is an implicit goal in Logic-based AI
  - Philosophy of logic-based AI focussed mostly on deductive reasoning (McCarthy, 1959),
  - In practice, many forms of inference are being developed (e.g. for FO, LP)

Status of KBS paradigm

- The KBS paradigm is an implicit goal in Logic-based AI
  - Philosophy of logic-based AI focussed mostly on deductive reasoning (McCarthy, 1959),
  - In practice, many forms of inference are being developed (e.g. for FO, LP)

- Widely considered to be unfeasible for rich languages such as FO

Unfeasability of KBS

- The Expressivity versus Tractability trade-off
  – the nightmare of KR researchers ?

Unfeasability of KBS

- The Expressivity versus Tractability trade-off
  – the nightmare of KR researchers ?
  - To specify all knowledge required to solve a real-world problem often requires a rich, expressive language.
  - Expressive languages are intractable or worse.

Unfeasability of KBS

- The Expressivity versus Tractability trade-off
  – the nightmare of KR researchers ?
  - To specify all knowledge required to solve a real-world problem often requires a rich, expressive language.
  - Expressive languages are intractable or worse.
- FO is "undecidable" (i.e., deduction)

Unfeasability of KBS

Solutions?

- The old medicine: Restricting expressivity of FO
  - E.g. Description logics, LP

But in these languages it is often hard or even impossible to specify the background knowledge required to solve many practical tasks.

Unfeasability of KBS

Solutions?

- The old medicine: Restricting expressivity of FO
  - E.g. Description logics, LP

But in these languages it is often hard or even impossible to specify the background knowledge required to solve many practical tasks.

- The new medicine:

  Useful "leight weight" forms of inference, computationally feasible for rich languages

  - Finite model checking, Query answering
  - Finite or bounded model generation (as in ASP)
  - Model revision
  - Approximate reasoning
  - ...

  An exciting prospect!

KBS changes our view on Logic

- Deductive logic, Abductive logic, Inductive logic , . . . ?
  - The inference is, by definition, part of the logic
  - A logic as a study of some form of inference

KBS changes our view on Logic

- Deductive logic, Abductive logic, Inductive logic , . . . ?
  - The inference is, by definition, part of the logic
  - A logic as a study of some form of inference
- The KBS paradigm leads to another view on logic.
  - It decouples the language from a specific form of inference
  - Logic as merely a formal language for expressing information

KBS changes our view on Logic

- Deductive logic, Abductive logic, Inductive logic , . . . ?
  - The inference is, by definition, part of the logic
  - A logic as a study of some form of inference
- The KBS paradigm leads to another view on logic.
  - It decouples the language from a specific form of inference
  - Logic as merely a formal language for expressing information
- In the KBS view, there is no conflict between abduction and deduction.

A requirement for a KB language

- A KB language is not a programming language
  - A KB theory does not encode a problem
  - A KB theory has no operational semantics
  - The KB is only a specification of the problem domain.
  - Used by human experts to communicate domain knowledge to the computer (and vice versa)

A requirement for a KB language

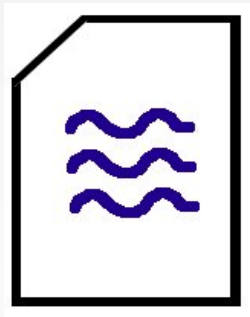
- A KB language is not a programming language
  - A KB theory does not encode a problem
  - A KB theory has no operational semantics
  - The KB is only a specification of the problem domain.
  - Used by human experts to communicate domain knowledge to the computer (and vice versa)

- This imposes a strong requirement on a KB-language:
  - Human experts need to be able to develop, interpret, extend, maintain a KB purely on the basis of
    - the KB's declarative semantics
    - the experts understanding of what the KB expresses about the problem domain.

A KR requirement on KB-language

A requirement for a KB language

Its informal semantics should be as objective, clear and precise as possible.



Informal Semantics

Theory

What it expresses

What is Informal semantics?

The informal semantics of a formula is what it expresses about the problem domain.

- E.g. the informal semantics of FO.

What is Informal semantics?

The informal semantics of a formula is what it expresses about the problem domain.

- E.g. the informal semantics of FO.
  - $\forall x(Human(x) \supset Male(x) \vee Female(x))$

What is Informal semantics?

The informal semantics of a formula is what it expresses about the problem domain.

- E.g. the informal semantics of FO.
  - $$\forall x(Human(x) \supset Male(x) \vee Female(x))$$
  - Under the suitable interpretation of the symbols, the informal semantics of this FO sentence is perfectly clear:

    Humans are male or female.

Relevance to KR

- The informal semantics of an expression is a thought
  - Informal, ethereal
- This makes its study difficult!
- Studied in Philosophical Logic,
  ignored in Computational Logic

Relevance to KR

- The informal semantics of an expression is a thought
  - Informal, ethereal
- This makes its study difficult!
- Studied in Philosophical Logic,
  ignored in Computational Logic
- It is important for declarative problem solving

Position

Informal semantics is fundamental for Knowledge Representation

I am not here in the happy position of a mineralogist who shows his audience a rock-crystal: I cannot put a thought in the hands of my readers with the request that they should examine it from all sides. Something in itself not perceptible by sense, the thought is presented to the reader — and I must be content with that — wrapped up in a perceptible linguistic form.

– Gottlob Frege, *Der Gedanke*

Informal semantics of the integration of FO and LP

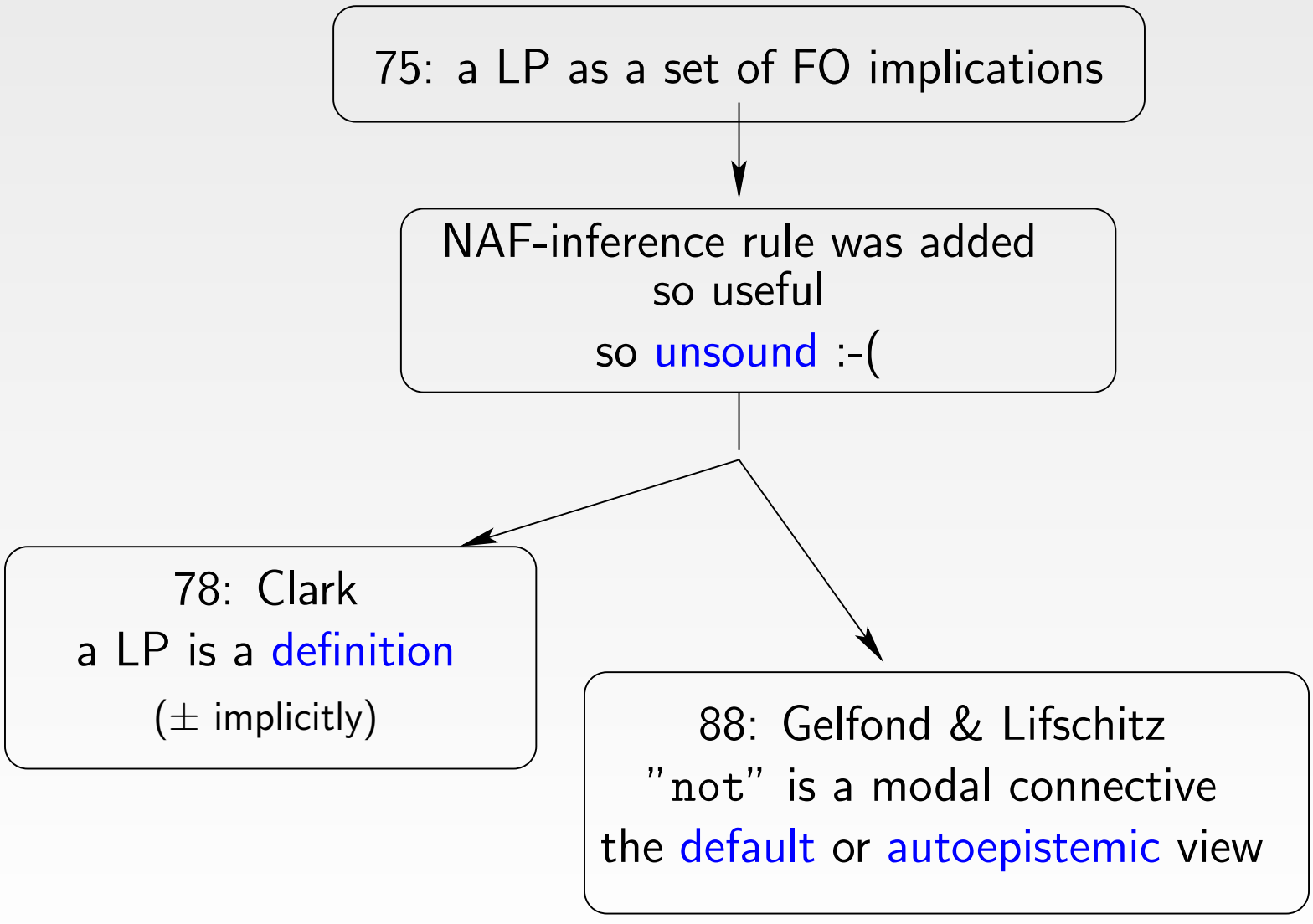
FO satisfies the informal semantics requirement for a KB-language.

What about LP?

Informal semantics of LP?

- In LP literature, the informal semantics of a logic program is sometimes called its declarative reading
- In LP, it is a blurred concept.
- Looking back in the history of LP
  - Many studies of formal semantics of LP
  - Only a few authors take position about the informal semantics.

History of LP's informal semantics



Informal semantics of LP: Definitions versus Defaults?

Two fundamentally different views on a logic program,
Two views on "`not`":

- a logic program as a default/autoepistemic theory
  - "`not`" as a non-derivability operator
    - "I do not know ..."
    - "It is consistent to assume the falsity of ..."
  - The view underlying Answer Set Programming

Informal semantics of LP: Definitions versus Defaults?

Two fundamentally different views on a logic program,
Two views on "`not`":

- a logic program as a default/autoepistemic theory
  - "`not`" as a non-derivability operator
    - "I do not know ..."
    - "It is consistent to assume the falsity of ..."
  - The view underlying Answer Set Programming
- a logic program as a definition (Clark, Schlipf)
  - NAF-inference derives "`not p`" only if $\neg p$ is entailed.
  - "`not`" is classical negation $\neg$.
  - In this view, it is the rule operator that is non-classical.

Informal semantics of LP

- Both are internally consistent views on the LP-formalism, with their own merits ...

Informal semantics of LP

- Both are internally consistent views on the LP-formalism, with their own merits ...
- We embrace the definitional view:
  - An informal semantics of mathematical precision (conf. informal semantics requirement),
  - reconciling LP's NAF with FO's classical negation.

Formal semantics of LP as ID

- We embrace Clark's informal semantics

Formal semantics of LP as ID

- We embrace Clark's informal semantics
- but not his formal semantics
  - Completion semantics is a FO semantics
  - Inductive definitions are not FO expressible in general
    - E.g. transitive closure
  - Completion semantics is too weak

Formal semantics of LP as ID

- We embrace Clark's informal semantics
- but not his formal semantics
  - Completion semantics is a FO semantics
  - Inductive definitions are not FO expressible in general
    - E.g. transitive closure
  - Completion semantics is too weak
- What is the right formal semantics?

Structure of the talk

- The conceptual framework
  - History and motivation of this work
  - The Knowledge Base System paradigm
  - Extending FO with inductive definitions

Inductive definitions in mathematics

The transitive closure T_G of a graph G is defined inductively:
- $(x, y) \in T_G$ if $(x, y) \in G$;
- $(x, y) \in T_G$ if for some vertex z,

$$(x, z), (z, y) \in T_G.$$

We define $\mathfrak{A} \models \varphi$ by structural induction:
- $\mathfrak{A} \models q$ if $q \in \mathfrak{A}$;
- $\mathfrak{A} \models \alpha \wedge \beta$ if $\mathfrak{A} \models \alpha$ and $\mathfrak{A} \models \beta$;
- $\mathfrak{A} \models \neg\alpha$ if $\mathfrak{A} \not\models \alpha$

(i.e., if not $\mathfrak{A} \models \alpha$);

- Inductive definition: an informal concept of mathematical precision

Inductive definitions in mathematics

The transitive closure T_G of a graph G is defined inductively:
- $(x, y) \in T_G$ if $(x, y) \in G$;
- $(x, y) \in T_G$ if for some vertex z,

$$(x, z), (z, y) \in T_G.$$

We define $\mathfrak{A} \models \varphi$ by structural induction:
- $\mathfrak{A} \models q$ if $q \in \mathfrak{A}$;
- $\mathfrak{A} \models \alpha \wedge \beta$ if $\mathfrak{A} \models \alpha$ and $\mathfrak{A} \models \beta$;
- $\mathfrak{A} \models \neg\alpha$ if $\mathfrak{A} \not\models \alpha$

(i.e., if not $\mathfrak{A} \models \alpha$);

- Inductive definition: an informal concept of mathematical precision
- A definition as a set of LP-like informal rules (with negation)
  - One rule specifies a sufficient condition
  - Together, they form a necessary condition.

Inductive definitions in mathematics

The transitive closure T_G of a graph G is defined inductively:
- $(x, y) \in T_G$ if $(x, y) \in G$;
- $(x, y) \in T_G$ if for some vertex z,

$$(x, z), (z, y) \in T_G.$$

We define $\mathfrak{A} \models \varphi$ by structural induction:
- $\mathfrak{A} \models q$ if $q \in \mathfrak{A}$;
- $\mathfrak{A} \models \alpha \wedge \beta$ if $\mathfrak{A} \models \alpha$ and $\mathfrak{A} \models \beta$;
- $\mathfrak{A} \models \neg\alpha$ if $\mathfrak{A} \not\models \alpha$

(i.e., if not $\mathfrak{A} \models \alpha$);

- Inductive definition: an informal concept of mathematical precision
- A definition as a set of LP-like informal rules (with negation)
  - One rule specifies a sufficient condition
  - Together, they form a necessary condition.
- Definitions are very generic by using "parameters":
  - A logic program including transitive closure, also specifies the value of G
  - The definition does not. It therefore specifies T_G for every "parameter" G.

FO(ID)'s syntax of definitions

Definition

An FO(ID) definition Δ is a set of definitional rules:

$$\forall \mathbf{x}(P(\mathbf{t}) \leftarrow \varphi)$$

where φ is a FO formula

- Δ's defined predicates: predicates in the head
- Δ's "parameters": all other symbols in Δ
  - Called the open symbols of Δ

FO(ID)'s semantics of definitions

Thesis: formal versus informal semantics

The parametrized variant of the well-founded semantics (Van Gelder, 1993) correctly formalizes common forms of inductive definitions in mathematics.

(D_, 1998, D_, Bruynooghe, Marek 2001, D_, Ternovska 2008)

- Well-founded semantics (Van Gelder, Ross, Schlipf 1991)
- Extension for arbitrary structures, FO bodies and open predicates (Van Gelder, 1993)

How do you argue such a thesis?

The argument: an outline

An inductive definition defines a relation by describing how to construct it:

- Adding an element when the condition of its rule becomes true.
- Excluding an element after establishing that no rule can derive it. E.g. after deriving that $I \models \varphi$, derive that $I \not\models \neg\varphi$.

Show that the constructions that we do when we reason about an inductive definition, match the constructions as described by the formal semantics.

The argument: an outline

An inductive definition defines a relation by describing how to construct it:

- Adding an element when the condition of its rule becomes true.
- Excluding an element after establishing that no rule can derive it. E.g. after deriving that $I \models \varphi$, derive that $I \not\models \neg\varphi$.

Show that the constructions that we do when we reason about an inductive definition, match the constructions as described by the formal semantics.

The argument: an outline

An inductive definition defines a relation by describing how to construct it:

- Adding an element when the condition of its rule becomes true.
- Excluding an element after establishing that no rule can derive it. E.g. after deriving that $I \models \varphi$, derive that $I \not\models \neg\varphi$.

Show that the constructions that we do when we reason about an inductive definition, match the constructions as described by the formal semantics.

Such an argument can be made using the notion of induction sequences defined in (D_, Vennekens LPNMR07).

Formal semantics of a FO(ID) definition

Definition

A structure $\mathfrak{A}$ satisfies a definition Δ if $\mathfrak{A}$ is the two-valued well-founded model of Δ in $\mathfrak{A}|_{Open(\Delta)}$.

A two-valued semantics:

- A three-valued well-founded model means that the definition contains a flaw and is treated as an inconsistency.

Definition of FO(ID)

Definition

A FO(ID) theory is a set of FO sentences and FO(ID) definitions. A structure is a model of a theory if it is a model of each of its elements.

(D_, 2000, D_, Ternovska, 2005, D_, Ternovska, TOCL 2008)

Definition of FO(ID)

Definition

A FO(ID) theory is a set of FO sentences and FO(ID) definitions. A structure is a model of a theory if it is a model of each of its elements.

(D_, 2000, D_, Ternovska, 2005, D_, Ternovska, TOCL 2008)

Claim

FO(ID) satisfies the informal semantics requirement for a KB-language

Integrating FO and LP

- What is the KR contribution of LP to FO?

> (Inductive) definitions have many important applications. In general such definitions are not expressible in FO.

Integrating FO and LP

- What is the KR contribution of LP to FO?

  (Inductive) definitions have many important applications. In general such definitions are not expressible in FO.

- FO(ID):
  - a useful combination of complementary language constructs
  - a conceptually clean, tight (non-hybrid) integration of FO and LP

Turning FO(ID) into a practical KB language FO($\cdot$)

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge (conf. ASP!)

Turning FO(ID) into a practical KB language FO(·)

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge (conf. ASP!)

$\Rightarrow$ FO(ID)

Turning FO(ID) into a practical KB language FO(·)

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge (conf. ASP!)

$\Rightarrow$ FO(ID,Types)

  - Types

Turning FO(ID) into a practical KB language FO(·)

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge (conf. ASP!)

$\Rightarrow$ FO(ID, Types, Agg)

  - Types
  - Aggregates

Turning FO(ID) into a practical KB language FO(·)

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge (conf. ASP!)

$\Rightarrow$ FO(ID,Types,Agg,Arit)

  - Types
  - Aggregates
  - Arithmetic

Turning FO(ID) into a practical KB language FO($\cdot$)

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge (conf. ASP!)

$\Rightarrow$ FO(ID,Types,Agg,Arit,DefFun)

  - Types
  - Aggregates
  - Arithmetic
  - Definitions of functions

Turning FO(ID) into a practical KB language FO(·)

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge (conf. ASP!)

$\Rightarrow$ FO(ID,Types,Agg,Arit,DefFun,...)

- Types
- Aggregates
- Arithmetic
- Definitions of functions
- ...

FO(·)

FO(ID) inductive definitions capture most common forms of induction, but not all.

Other natural forms of induction

- A form of induction not formalized by FO(ID) definitions:

  Coinduction

  - (Monotone) induction: least fixpoint construction
  - Coinduction: greatest fixpoint construction
- Logics of integrated induction/coinduction:
  - μ-calculus, Fixpoint logic
  - Coinductive Logic Programming
    (Simon, Mallya, Bansal, Gupta, 2006)
- Important theoretical and practical applications: bisimulation, modal, temporal and dynamic logics, automata theory, verification, programming with cyclic datastructures, ...

Coinduction versus (non-monotone) FO(ID) induction

- A research question

> How do these inductive principles compare?

Coinduction versus (non-monotone) FO(ID) induction

- A research question

  How do these inductive principles compare?

- Some initial work (Ping, D_, ASPOCP 2009)
  1. Definition of FO(FD) — FD: Fixpoint Definitions
     - Nested least and greatest fixpoints in CoLP syntax
  2. Theorem : non-monotone induction $\subseteq$ coinduction

     FO(ID) has a simple embedding in FO(FD), using a coinductive definition nested in an inductive definition.

  3. Model generation algorithms for propositional FO(FD) definitions (towards extending the IDP system)

Structure of the talk

- The conceptual framework
  - History and motivation of this work
  - The Knowledge Base System paradigm
  - FO(·): integration of FO and LP
  - Common sense KR in FO(ID)

Common sense KR in FO(ID)

- Inductive definitions are important in mathematics
- But mathematical knowledge is so different from common sense knowledge!
- Are inductive definitions useful for common sense?

Common sense KR in FO(ID)

- Inductive definitions are important in mathematics
- But mathematical knowledge is so different from common sense knowledge!
- Are inductive definitions useful for common sense?

Claim

Inductive definitions are strongly linked to two forms of common sense knowledge:

- Closed World Assumption
- Causality

Inductive definitions and common sense KR

Claim

The concept of inductive definition is a natural, precise, useful form of Closed Word Assumption (CWA)

- The CWA principle: "every atom not derived by a rule is false"
  That is part of the principle of ID!!!

Inductive definitions and common sense KR

Claim

The concept of inductive definition is a natural, precise, useful form of Closed Word Assumption (CWA)

- The CWA principle: "every atom not derived by a rule is false" That is part of the principle of ID!!!
- A parametrized form of CWA :
  - CWA is not applied to open atoms of a definition
  - as in ASP!

Representing defaults by definitions

- A "default theory" with exceptions and exceptions to exceptions

$$\left\{\begin{array}{l} \forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\ \forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\ \forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\ \forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \mathit{Penguin}(\mathit{Tweety}) \leftarrow \\ \mathit{Bird}(\mathit{Fred}) \leftarrow \\ \forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \end{array}\right\} \begin{array}{l} \models \mathit{Flies}(\mathit{Fred}) \\ \models \neg \mathit{Flies}(\mathit{Tweety}) \end{array}$$

Representing defaults by definitions

- A "default theory" with exceptions and exceptions to exceptions

$$\left\{\begin{array}{l} \forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\ \forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\ \forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\ \forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \mathit{Penguin}(\mathit{Tweety}) \leftarrow \\ \mathit{Bird}(\mathit{Fred}) \leftarrow \\ \forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \end{array}\right\} \begin{array}{l} \models \mathit{Flies}(\mathit{Fred}) \\ \models \neg \mathit{Flies}(\mathit{Tweety}) \end{array}$$

- The (mathematical) view as a definition gives us precise insight in the meaning of this rule set.

Representing defaults by definitions

- A "default theory" with exceptions and exceptions to exceptions

$$\left\{\begin{array}{l} \forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\ \forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\ \forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\ \forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \mathit{Penguin}(\mathit{Tweety}) \leftarrow \\ \mathit{Bird}(\mathit{Fred}) \leftarrow \\ \forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \end{array}\right\} \begin{array}{l} \models \mathit{Flies}(\mathit{Fred}) \\ \models \neg \mathit{Flies}(\mathit{Tweety}) \end{array}$$

- The (mathematical) view as a definition gives us precise insight in the meaning of this rule set.
- Where is the default nature?

Representing defaults by definitions

- A "default theory" with exceptions and exceptions to exceptions

$$\left\{\begin{array}{l} \forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\ \forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\ \forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\ \forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \mathit{Penguin}(\mathit{Tweety}) \leftarrow \\ \mathit{Bird}(\mathit{Fred}) \leftarrow \\ \forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \end{array}\right\} \begin{array}{l} \models \mathit{Flies}(\mathit{Fred}) \\ \models \neg \mathit{Flies}(\mathit{Tweety}) \end{array}$$

- The (mathematical) view as a definition gives us precise insight in the meaning of this rule set.
- Where is the default nature?
  - This definition is not entirely true. It is an approximation of reality, tuned to the situation and available knowledge.

Non-monotonicity & Elaboration Tolerance

$$\left\{\begin{array}{l} \forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\ \forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\ \forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\ \forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \mathit{Penguin}(\mathit{Tweety}) \leftarrow \\ \mathit{Bird}(\mathit{Fred}) \leftarrow \\ \forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \end{array}\right\} \models \neg \mathit{Flies}(\mathit{Tweety})$$

Non-monotonicity & Elaboration Tolerance

$$\left\{\begin{array}{l} \forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\ \forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\ \forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\ \forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\ \\ \mathit{Penguin}(\mathit{Tweety}) \leftarrow \quad \mathit{SuperPenguin}(\mathit{Tweety}) \leftarrow \\ \mathit{Bird}(\mathit{Fred}) \leftarrow \\ \forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \end{array}\right\} \models \mathit{Flies}(\mathit{Tweety})$$

Non-monotonicity & Elaboration Tolerance

$$
\left\{
\begin{array}{l}
\forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\
\forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\
\\
\forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\
\forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\
\forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\
\forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{BrokenWing}(x) \\
\\
\mathit{Penguin}(\mathit{Tweety}) \leftarrow \quad \mathit{SuperPenguin}(\mathit{Tweety}) \leftarrow \\
\mathit{Bird}(\mathit{Fred}) \leftarrow \\
\forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \\
\forall x\ \mathit{BrokenWing}(x) \leftarrow \mathbf{f}
\end{array}
\right\}
$$

Non-monotonicity & Elaboration Tolerance

$$\left\{\begin{array}{l}
\forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\
\forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\
\\
\forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\
\forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\
\forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\
\forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{BrokenWing}(x) \\
\\
\mathit{Penguin}(\mathit{Tweety}) \leftarrow \quad \mathit{SuperPenguin}(\mathit{Tweety}) \leftarrow \\
\mathit{Bird}(\mathit{Fred}) \leftarrow \quad \mathit{BrokenWing}(\mathit{Fred}) \leftarrow \\
\forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \\
\forall x\ \mathit{BrokenWing}(x) \leftarrow \mathbf{f}
\end{array}\right\} \models \neg \mathit{Flies}(\mathit{Fred})$$

Non-monotonicity & Elaboration Tolerance

$$
\left\{
\begin{array}{l}
\forall x\ \mathit{Bird}(x) \leftarrow \mathit{Penguin}(x) \\
\forall x\ \mathit{Penguin}(x) \leftarrow \mathit{SuperPenguin}(x) \\
\\
\forall x\ \mathit{Flies}(x) \leftarrow \mathit{Bird}(x) \wedge \neg \mathit{Ab}_F(x) \\
\forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{Penguin}(x) \wedge \neg \mathit{Ab}_{FP}(x) \\
\forall x\ \mathit{Ab}_{FP}(x) \leftarrow \mathit{SuperPenguin}(x) \\
\forall x\ \mathit{Ab}_F(x) \leftarrow \mathit{BrokenWing}(x) \\
\\
\mathit{Penguin}(\mathit{Tweety}) \leftarrow \quad \mathit{SuperPenguin}(\mathit{Tweety}) \leftarrow \\
\mathit{Bird}(\mathit{Fred}) \leftarrow \quad \mathit{BrokenWing}(\mathit{Fred}) \leftarrow \\
\forall x\ \mathit{SuperPenguin}(x) \leftarrow \mathbf{f} \\
\forall x\ \mathit{BrokenWing}(x) \leftarrow \mathbf{f}
\end{array}
\right\}
$$

- The same good non-monotonic properties as ASP
  - Rules are non-monotonic modules
  - Defined atoms are false by default

$\Rightarrow$ Elaboration tolerance

Integrating non-monotonicity in FO

FO(ID) is an extension of FO with non-monotonic modules encorporating a form of CWA of mathematical precision

- In comparison, the notion of CWA is a vague intuition

Integrating non-monotonicity in FO

FO(ID) is an extension of FO with non-monotonic modules encorporating a form of CWA of mathematical precision

- In comparison, the notion of CWA is a vague intuition

Definitional rules provide a useful form of nonmonotonic modularity.

Integrating non-monotonicity in FO

FO(ID) is an extension of FO with non-monotonic modules encorporating a form of CWA of mathematical precision

- In comparison, the notion of CWA is a vague intuition

Definitional rules provide a useful form of nonmonotonic modularity.

(FO(ID) is not only an extension of FO with inductive definitions)

Causality

- Causation : basic properties
  - No spontaneous generation
    - Effects and changes do not occur without external cause
  - Acyclicity: no phenomenon directly or indirectly causes itself
  - causal processes : propagation of effects
    - $\sim$ ramification in temporal reasoning

Causality

- Causation : basic properties
  - No spontaneous generation
    - Effects and changes do not occur without external cause
  - Acyclicity: no phenomenon directly or indirectly causes itself
  - causal processes : propagation of effects
    - $\sim$ ramification in temporal reasoning
- Causal processes $\sim$ inductive constructions

Causality

- Causation : basic properties
  - No spontaneous generation
    - Effects and changes do not occur without external cause
  - Acyclicity: no phenomenon directly or indirectly causes itself
  - causal processes : propagation of effects
    - $\sim$ ramification in temporal reasoning
- Causal processes $\sim$ inductive constructions

Hypothesis

An inductive definition is a causal theory applied to the platonic world of mathematics

(D_, Theseider-Dupré, Van Belleghem, 1998)

Causation — Inductive definitions

- Causal reasoning:
  - a crucial form of common sense reasoning
  - innate in the human cognition?

Causation — Inductive definitions

- Causal reasoning:
  - a crucial form of common sense reasoning
  - innate in the human cognition?

- Causal reasoning: the source of our remarkable and largely subconscious ability to reason with inductive definitons?

Causation — Inductive definitions

Formal investigations

- Applying (non-monotone) inductive definitions to ramification and causal processes

  (D_, Theseider-Dupré, Van Belleghem, 1998; D_, Ternovska, 2007)
- CP-logic

CP-logic

- Typical investigations into causality (e.g., Judea Pearl) are concerned with non-deterministic, probabilistic causality
- CP-logic rules:

$$\underbrace{(Goal\colon 0.5) \vee (Broken(Window)\colon 0.1)}_{\textit{probabilistic effect}} \leftarrow \underbrace{Kick(Ball)}_{\textit{cause}}.$$

CP-logic

- Typical investigations into causality (e.g., Judea Pearl) are concerned with non-deterministic, probabilistic causality
- CP-logic rules:

$$\underbrace{(Goal\colon 0.5) \vee (Broken(Window)\colon 0.1)}_{probabilistic\ effect} \leftarrow \underbrace{Kick(Ball)}_{cause}\,.$$

CP-logic

- Typical investigations into causality (e.g., Judea Pearl) are concerned with non-deterministic, probabilistic causality
- CP-logic rules:

$$\underbrace{(Goal\colon 0.5) \vee (Broken(Window)\colon 0.1)}_{probabilistic\ effect} \leftarrow \underbrace{Kick(Ball)}_{cause}.$$

- Deterministic rules: a special case

$$Goal : 1 \leftarrow Kick(Ball)$$

- Theorem (Vennekens, D , Bruynooghe, 2009)

A deterministic CP-logic theory coincides with a FO(ID) definition

In view of these observations, one expects some a strong with ASP?

Relationship to ASP

- ASP and FO(ID) are conceptually very different
- In practice, there is a strong congruence between FO(ID) and a core language of ASP (without disjunction)
- An answer set program as an FO(ID) theory:
  - Constraints $\sim$ FO sentences
  - Rules $\sim$ a definition
    - an atom is defined, unless ...
    - it is declared to be open (various ways):

$$in(X) \leftarrow \texttt{not}\ out(X) \qquad out(X) \leftarrow \texttt{not}\ in(X) \qquad \mid \qquad in(x) \vee \texttt{not}\ in(x) \leftarrow \qquad \mid \qquad 0 \leq \{in(X)\} \leq 1 \leftarrow$$

FO(ID) and ASP side by side

Hamiltonian circuit

FO(ID):	ASP
$\left\{ \begin{array}{l} vertex(a) \leftarrow \\ \ldots \end{array} \right\} \quad \left\{ \begin{array}{l} edge(a,b) \leftarrow \\ \ldots \end{array} \right\}$	$\begin{array}{ll} vertex(a) \leftarrow & edge(a,b) \leftarrow \\ \ldots & \ldots \end{array}$
$\left\{ \begin{array}{l} \forall X, Y\ (reached(Y) \leftarrow \\ \qquad start(X) \wedge in(X,Y)) \\ \forall X, Y\ (reached(Y) \leftarrow \\ \qquad reached(X) \wedge in(X,Y)) \end{array} \right\}$ $\{\ start(a) \leftarrow\ \}$	$reached(Y) \leftarrow start(X), in(X,Y)$ $reached(Y) \leftarrow reached(X), in(X,Y)$ $start(a) \leftarrow$
	$in(X,Y) \leftarrow \texttt{not}\ out(X,Y)$ $out(X,Y) \leftarrow \texttt{not}\ in(X,Y)$
$\forall X, Y\ (in(X,Y) \supset edge(X,Y))$ $\forall X\ (vertex(X) \supset reached(X))$ $\forall X, Y, Z\ (in(Y,X) \wedge in(Z,X) \supset Y = Z)$ $\forall X, Y, Z\ (in(X,Y) \wedge in(X,Z) \supset Y = Z)$	$\bot \leftarrow in(X,Y), \texttt{not}\ edge(X,Y)$ $\bot \leftarrow vertex(X), \texttt{not}\ reached(X)$ $\bot \leftarrow in(Y,X), in(Z,X), \texttt{not}\ Y = Z$ $\bot \leftarrow in(X,Y), in(X,Z), \texttt{not}\ Y = Z$

FO(ID) and ASP side by side

Hamiltonian circuit

FO(ID):	ASP
$\{\ vertex(a) \leftarrow\ \}$ $\{\ edge(a,b) \leftarrow\ \}$	$vertex(a) \leftarrow$ $edge(a,b) \leftarrow$
$\{\ \ldots\ \}$ $\{\ \ldots\ \}$	$\ldots$ $\ldots$
$\{\ \forall X, Y\ (reached(Y) \leftarrow start(X) \wedge in(X,Y))$	$reached(Y) \leftarrow start(X), in(X,Y)$
$\forall X, Y\ (reached(Y) \leftarrow reached(X) \wedge in(X,Y))\ \}$	$reached(Y) \leftarrow reached(X), in(X,Y)$
$\{\ start(a) \leftarrow\ \}$	$start(a) \leftarrow$
	$in(X,Y) \leftarrow \texttt{not}\ out(X,Y)$
	$out(X,Y) \leftarrow \texttt{not}\ in(X,Y)$
$\forall X, Y\ (in(X,Y) \supset edge(X,Y))$	$\bot \leftarrow in(X,Y), \texttt{not}\ edge(X,Y)$
$\forall X\ (vertex(X) \supset reached(X))$	$\bot \leftarrow vertex(X), \texttt{not}\ reached(X)$
$\forall X, Y, Z\ (in(Y,X) \wedge in(Z,X) \supset Y = Z)$	$\bot \leftarrow in(Y,X), in(Z,X), \texttt{not}\ Y = Z$
$\forall X, Y, Z\ (in(X,Y) \wedge in(X,Z) \supset Y = Z)$	$\bot \leftarrow in(X,Y), in(X,Z), \texttt{not}\ Y = Z$

ASP $\longrightarrow$ FO(ID) ($\pm$):

- Adding brackets on the right places
- Introducing declarations of open atoms

Relationship to ASP, formally

- A linear modular transformation back and forth (East, 2004; Mariën, Gilis, D_, 2004)

- Stable semantics versus Well-founded semantics
  - Most ASP programs have a unique three-valued well-founded model and multiple stable models
  - What about the parametrized well-founded models?
    - Open atoms are freely choosen
    - There are many such models, and typically two-valued

A model of a definition is a two-valued well-founded model, and hence a stable model!

Relationship to ASP

ASP versus FO(ID)

- ASP and FO(ID) representations are often isomorphic
- ASP and FO(ID) methodologies are similar

Structure of the talk

- The conceptual framework
- Implementation: progress report

Problem solving in the KBS paradigm

A new way of thinking about declarative problem solving:

- given a desired specification of a problem domain
- what kind of inference is needed to solve the problem?

Implementation of KBS

Forms of inference under development:

- Model generation: the IDP system
- Approximate reasoning (KR 2008)
- Revision inference

Structure of the talk

- The conceptual framework
- Implementation: progress report
  - Model expansion/generation

The IDP system

(Wittocx,Mariën,D_ 2008)

- Its purpose : generate models for a FO($\cdot$)theory with a given finite domain D.

The IDP system

(Wittocx,Mariën,D_ 2008)

- Its purpose : generate models for a FO($\cdot$)theory with a given finite domain D.
- Technology: grounding + SAT + ASP technology
  - Incorporating state-of-the-art stable semantics algorithms in MiniSat.

The IDP system

(Wittocx,Mariën,D_ 2008)

- Its purpose : generate models for a FO($\cdot$)theory with a given finite domain D.
- Technology: grounding + SAT + ASP technology
  - Incorporating state-of-the-art stable semantics algorithms in MiniSat.
- Results:
  - An Answer Set Programming system using FO($\cdot$)
  - A rich input language
    - Currently the only model generation for full first-order logic
    - plus ID's, Types, Agg, Arithmetic, partial functions, ...

The Second ASP competition

- organized by KRR, March-June 2009
- 16 teams - 14 single system teams
- three FO(·)systems
  - Enfragmo - Simon Fraser University
  - amsolver - University of Kentucky
  - IDP - K.U.Leuven
- IDP ranks 3'th on 14 in three of the five categories (including the global category)
  - Thanks to Johan Wittocx

Model revision

- Often, not only a single solution to a search problem is needed but also *revisions* of this model under new circumstances (train rescheduling, reconfiguration of a computer network, ...)

Model revision

- Often, not only a single solution to a search problem is needed but also *revisions* of this model under new circumstances (train rescheduling, reconfiguration of a computer network, ...)
- Typical requirements for revision:
  - New solution is close to previous one.
  - Efficiently computable (minimize downtime)

Declarative approach

- Given:
  - Theory T describing the requirements for a solution
    (may be the same theory as used for finding the first solution)
  - Previous solution $M \quad (M \models T)$
  - Set of atoms C describing the changed circumstances

Declarative approach

- Given:
  - Theory T describing the requirements for a solution (may be the same theory as used for finding the first solution)
  - Previous solution $M \quad (M \models T)$
  - Set of atoms C describing the changed circumstances
- Find: Structure M' such that $M' \models T$ and $M'(A) \neq M(A)$ for every $A \in C$.

Declarative approach

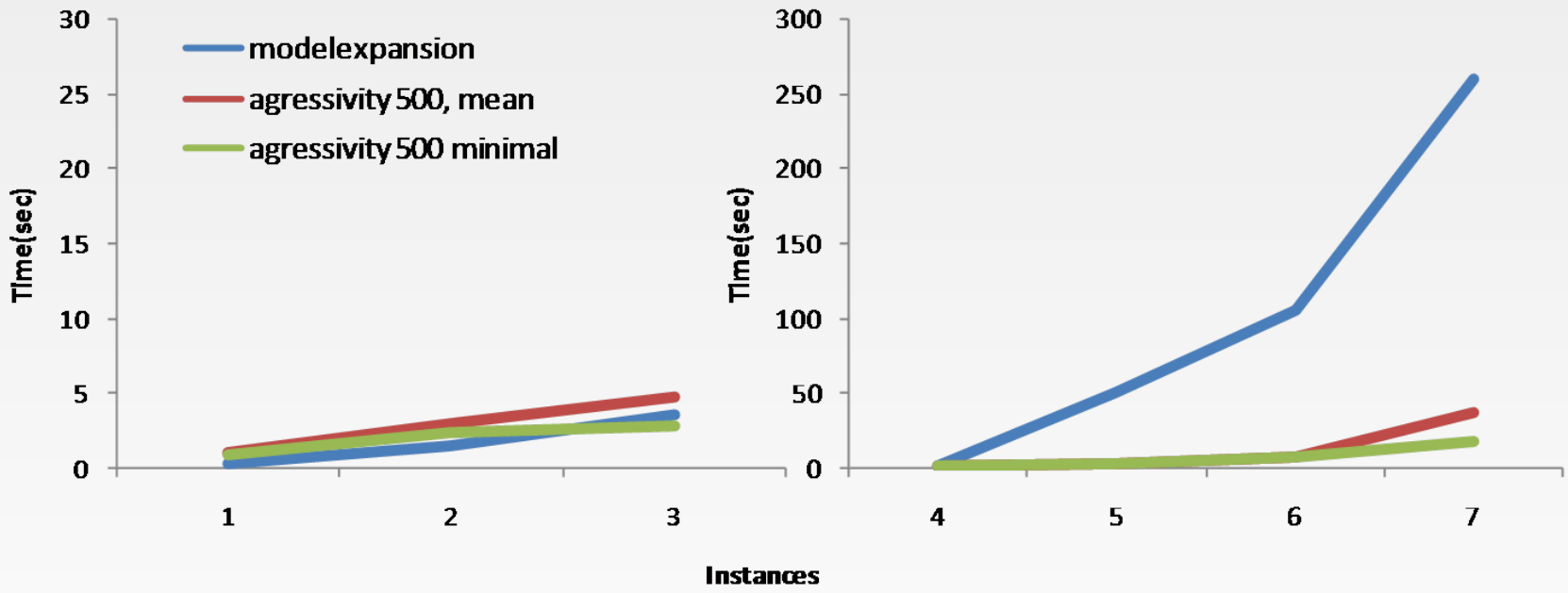
- Given:
  - Theory T describing the requirements for a solution (may be the same theory as used for finding the first solution)
  - Previous solution $M \quad (M \models T)$
  - Set of atoms C describing the changed circumstances
- Find: Structure M' such that $M' \models T$ and $M'(A) \neq M(A)$ for every $A \in C$.

Example (Network configuration)

$T = \{\exists c\ \mathit{MailServer}(c).\quad \exists c\ \mathit{DHCPServer}(c).\quad \ldots\}$
$M = \{\mathit{MailServer}(PC_1), \ldots\}$
$C = \{\mathit{MailServer}(PC_1)\}\quad$ (e.g., PC_1 is broken)
Solution: $M' = \{\mathit{MailServer}(PC_2), \ldots\}$

Results (Train rescheduling)

Time needed to calculate completely new solution
vs. time needed to calculate revision



Future work: heuristics to find close to "optimal" revisions.
(Current algorithm makes sometimes 5× more changes than needed)

A Logical Framework for Knowledge-intensive Software

Problem setting

- Fact: applications with a lot of domain knowledge are typically hard to implement and/or adapt. E.g.
  - configuration software,
  - taxonweb, ...
- Solution: Knowledge Base Paradigm
  - Express all relevant domain knowledge using a natural knowledge representation language,
  - Solve tasks using a range of reasoning mechanisms.

A Logical Framework for Knowledge-intensive Software

Some examples of knowledge based approaches.

- Business Rules: language directed towards one specific form of reasoning
- OWL: restricted language in order to keep decidability of deduction
- Constraint Programming: lacks a unified general language for expressing the knowledge

Important observation

- In different approaches we see one trend:
- The languages used in these approaches are all converging towards syntactical variants of First Order Logic

A Logical Framework for Knowledge-intensive Software

Our contribution

- Use FO($\cdot$)(a rich extension of FO) as language for expressing the domain knowledge
- We have identified a number of typical tasks we want to perform in knowledge-intensive applications.
- We have shown that we can formulate these tasks as logical inference tasks on a FO($\cdot$)theory.
- See [Vlaeminck et al, PPDP'09]

Debugging of FO theories

- A debugging method for FO theories, by tracing a propagation based solver.
- See [Wittocx et al, ICLP'09]

FO(ID) as extension of DL with rules

$$\begin{array}{ccccc} \text{DL} & + & \text{Rules} \\ \cap & & \cap \\ \text{FO} & (& \text{ID} &) \end{array}$$

- Adding rules to DL is hot topic in Semantic Web community
- FO(ID) induces a way of extending DL with definitional rules
  - Strong semantic integration
  - Conceptually clean

J.Vennekens, M.Denecker. FO(ID) as an Extension of DL with Rules. European Semantic Web Conference (ESWC) 2009.

The KBS as a conceptual framework, as a researc ideal
Can it be built? Not for a long time problably. Fine-tuning the KB to a specific computational problem will needed for a long time.

Integrating FO and (A)LP: Challenges

- Many things to reconcile:

(Abductive) Logic Programming	Classical logic (FO)
Abduction	Deduction
Operational semantics	-
Non-monotone	Monotone
NAF `not`	Classical negation $\neg$
Herbrand structures	Arbitrary structures
Domain Closure Assumption (DCA)	No domain assumption
Constructors (UNA)	Free functions
Belief sets	Possible worlds
...	...

Integrating FO and (A)LP: Challenges

- Many things to reconcile:

(Abductive) Logic Programming	Classical logic (FO)
Abduction	Deduction
Operational semantics	-
Non-monotone	Monotone
NAF `not`	Classical negation $\neg$
Herbrand structures	Arbitrary structures
Domain Closure Assumption (DCA)	No domain assumption
Constructors (UNA)	Free functions
Belief sets	Possible worlds
...	...

FO(ID)

- A logic satisfying the Informal Semantics requirement for KBS
- A view on the KR contribution of LP to FO.
- A tight integration of LP and FO
- A formal extension of ALP
- Inductive definitions as a precisely understood form of CWA, and useful to express defaults
- An integration of non-monotonic modules in FO

In the process, we get solutions for some research questions in ASP for free:

- Like FO, FO(ID) is an open domain logic, but DCA can be expressed
- Like FO, FO(ID) has free function symbols, but UNA can be expressed

ASP versus FO(ID)

- A useful fragment of ASP
  - implicit CWA
  - no classical negation
  - no disjunction in head
- Strongly corresponds to (a fragment of) FO(ID)
  - FO(ID): by default, an atom is open, unless defined
  - ASP: by default, an atom is closed

Informal and formal constructions

The transitive closure T_G of a graph G is defined inductively:
- $(x, y) \in T_G$ if $(x, y) \in G$;
- $(x, y) \in T_G$ if for some vertex z,

$$(x, z), (z, y) \in T_G.$$

We define $\mathfrak{A} \models \varphi$ by structural induction:
- $\mathfrak{A} \models q$ if $q \in \mathfrak{A}$;
- $\mathfrak{A} \models \alpha \wedge \beta$ if $\mathfrak{A} \models \alpha$ and $\mathfrak{A} \models \beta$;
- $\mathfrak{A} \models \neg\alpha$ if $\mathfrak{A} \not\models \alpha$

(i.e., if not $\mathfrak{A} \models \alpha$);

versus

$$\left\{ \begin{array}{l} \forall xy\ (Tr(x,y) \leftarrow G(x,y)) \\ \forall xy\ (Tr(x,y) \leftarrow \\ \quad \exists z(lTr(x,z) \wedge \\ Tr(z,y))) \end{array} \right\}$$

$$\left\{ \begin{array}{l} \forall i, p\ (Sat(i,p) \leftarrow \\ Holds(p,i)) \\ \forall i, f_1, f_2\ (Sat(i, and(f_1, f_2)) \leftarrow \\ \quad Sat(i, f_1) \wedge Sat(i, f_2)). \\ \forall i, f\ (Sat(i, not(f)) \leftarrow \\ \quad \neg Sat(i, f)) \end{array} \right\}$$