Building a Knowledge Base System for an Integration of Logic Programming and Classical Logic

Marc Denecker and Joost Vennekens

December 9, 2008

Introduction

- A celebration for a 20th anniversary
  - Stable semantics by Michael Gelfond and Vladimir Lifschitz
  - The origin of Answer Set Programming

Introduction

- A celebration for a 20th anniversary
  - Stable semantics by Michael Gelfond and Vladimir Lifschitz
  - The origin of Answer Set Programming
- A position session — a critique

> " All the attractive features of the semantics and the ASP approach notwithstanding, there are alternative approaches that are better suited to address KR challenges."

Introduction

- A celebration for a 20th anniversary
  - Stable semantics by Michael Gelfond and Vladimir Lifschitz
  - The origin of Answer Set Programming
- A position session — a critique

> " All the attractive features of the semantics and the ASP approach notwithstanding, there are alternative approaches that are better suited to address KR challenges."

- An alternative logic: (FO+Inductive Definitions)

FO(ID)

Introduction

- A celebration for a 20th anniversary
  - Stable semantics by Michael Gelfond and Vladimir Lifschitz
  - The origin of Answer Set Programming
- A position session — a critique

> " All the attractive features of the semantics and the ASP approach notwithstanding, there are alternative approaches that are better suited to address KR challenges."

- An alternative logic: (FO+Inductive Definitions)

FO(ID)

  - Stable semantics $\bigcap$ Well-founded semantics

Introduction

- A celebration for a 20th anniversary
  - Stable semantics by Michael Gelfond and Vladimir Lifschitz
  - The origin of Answer Set Programming
- A position session — a critique

> " All the attractive features of the semantics and the ASP approach notwithstanding, there are alternative approaches that are better suited to address KR challenges."

- An alternative logic: (FO+Inductive Definitions)

> FO(ID)

  - Stable semantics $\bigcap$ Well-founded semantics
  - Builds on other LP-traditions
    - Abductive logic programming
    - Deductive databases

Knowledge Base Systems

Informal semantics of LP

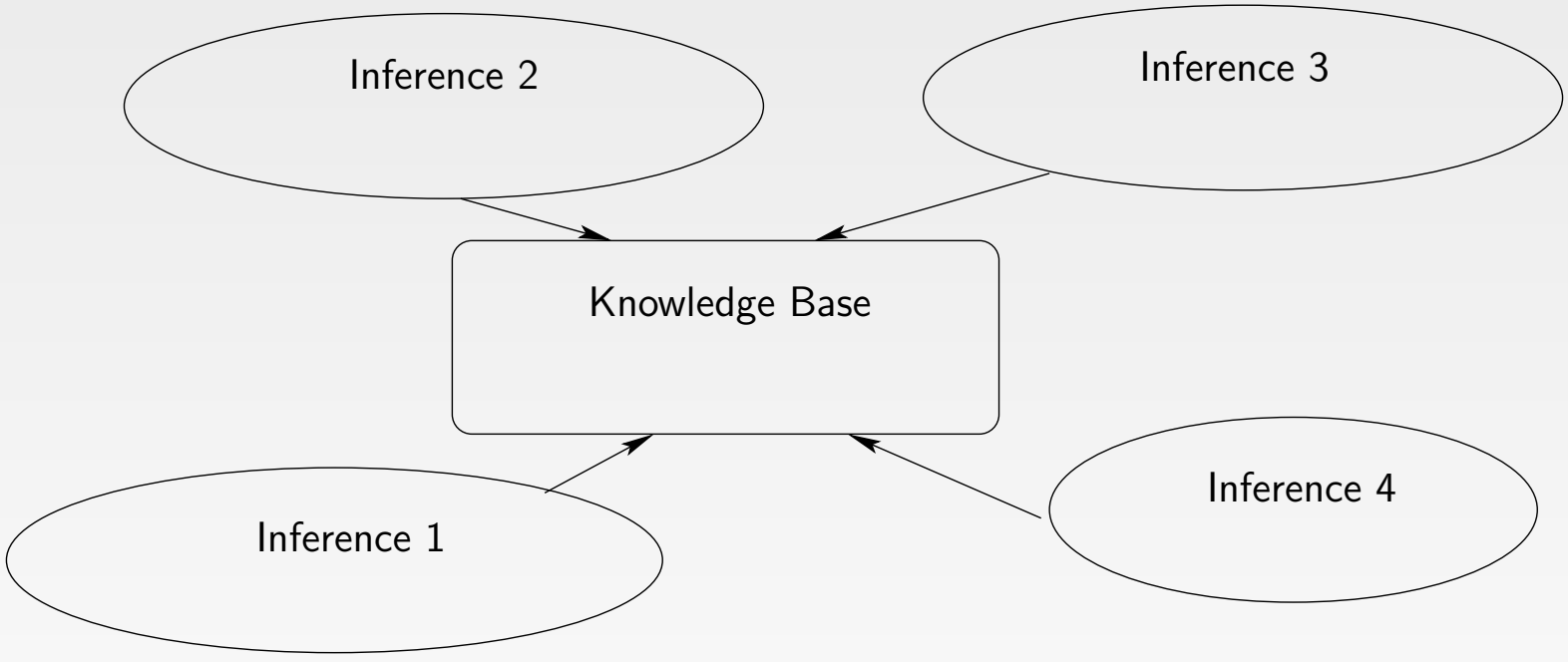
Formal definition of FO(ID)

Knowledge representation with FO(ID)

Implementation: progress report

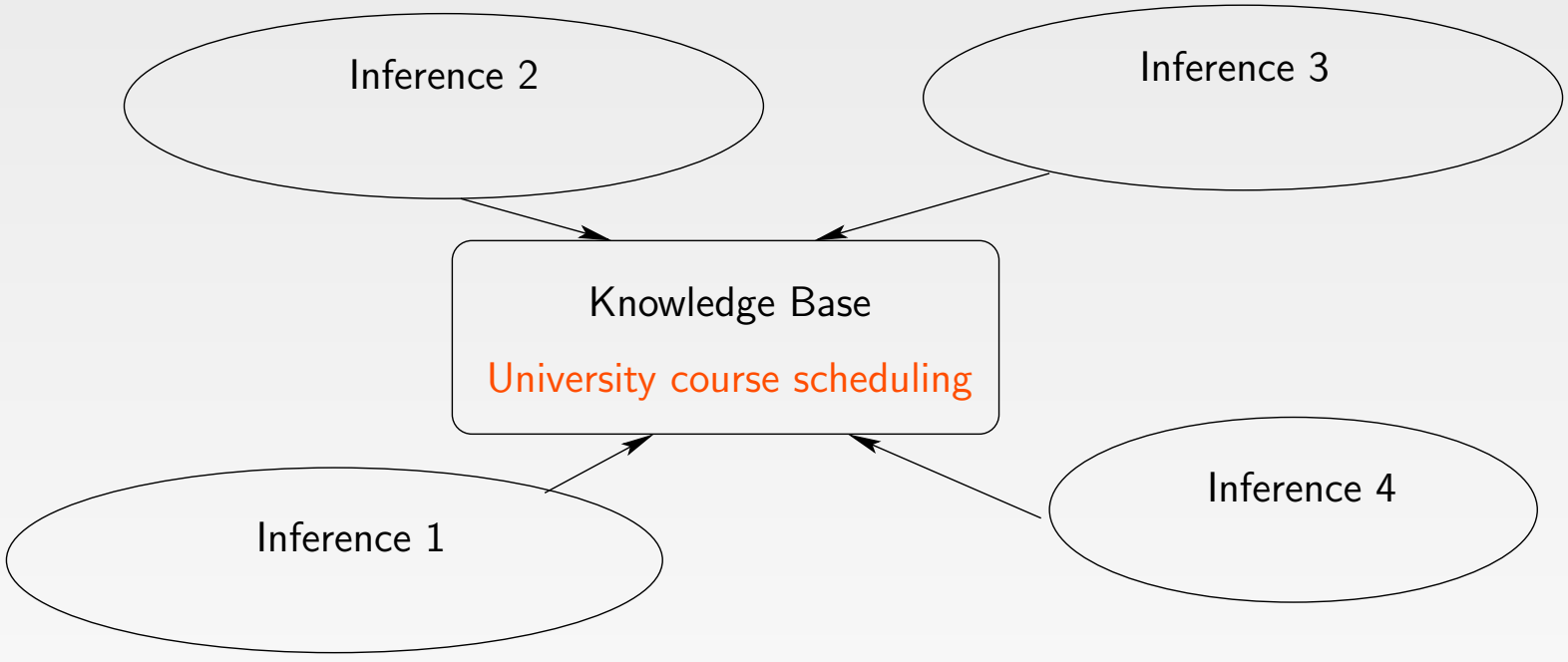
Conclusion

A Knowledge Base System



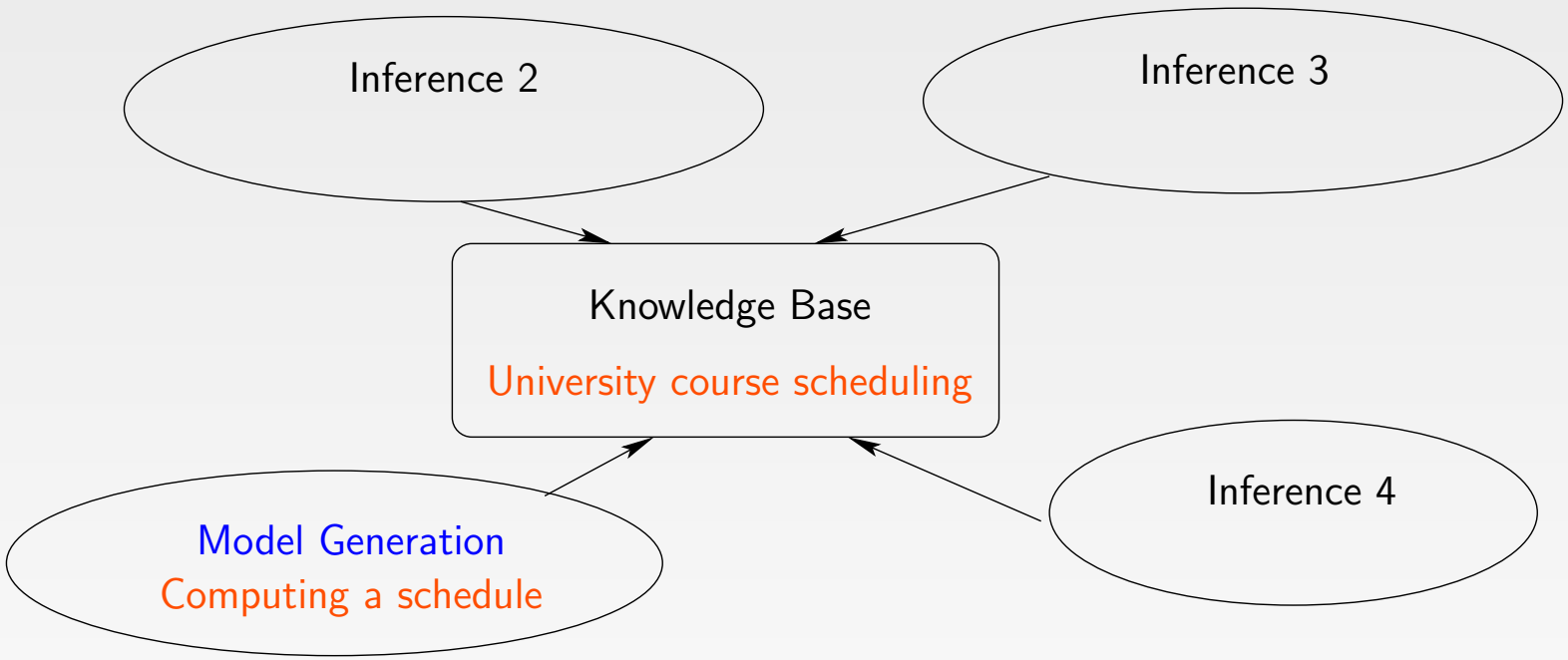
- Manages a declarative Knowledge Base
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System



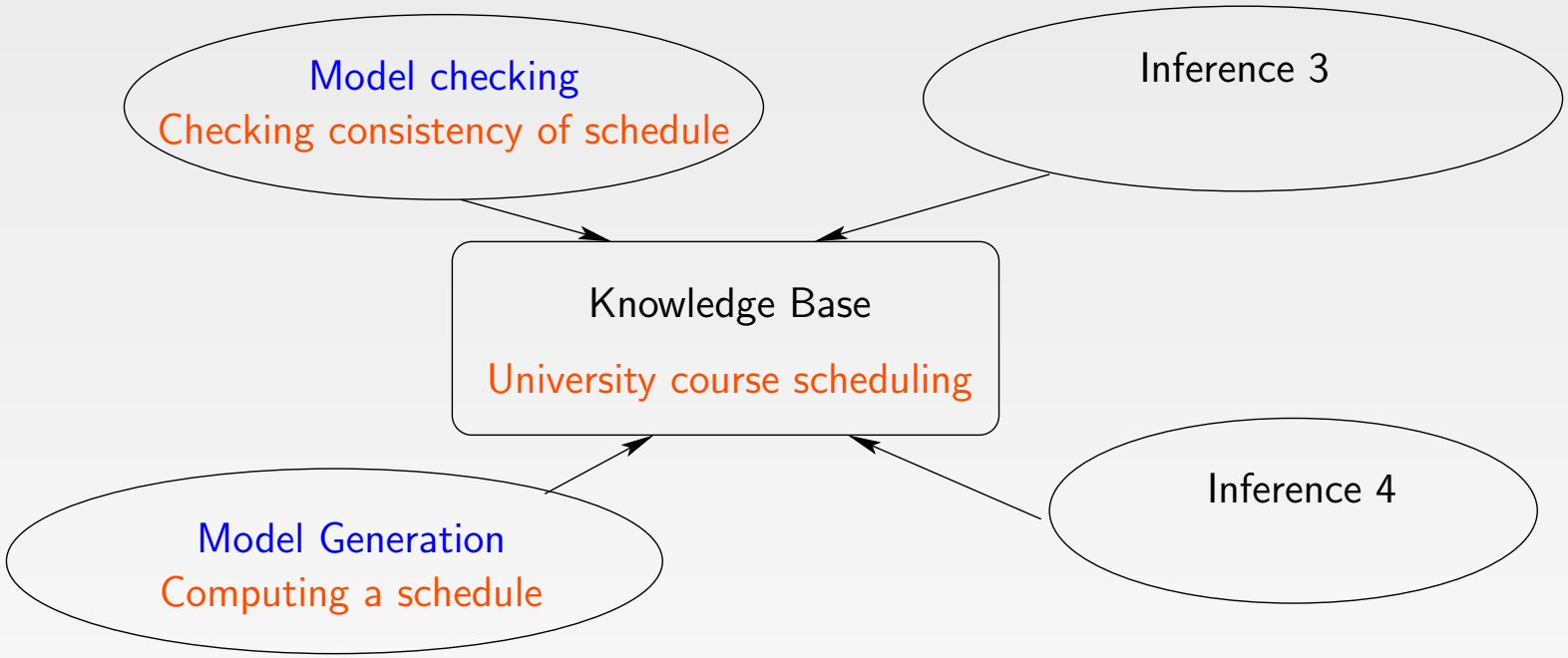
- Manages a declarative Knowledge Base
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System



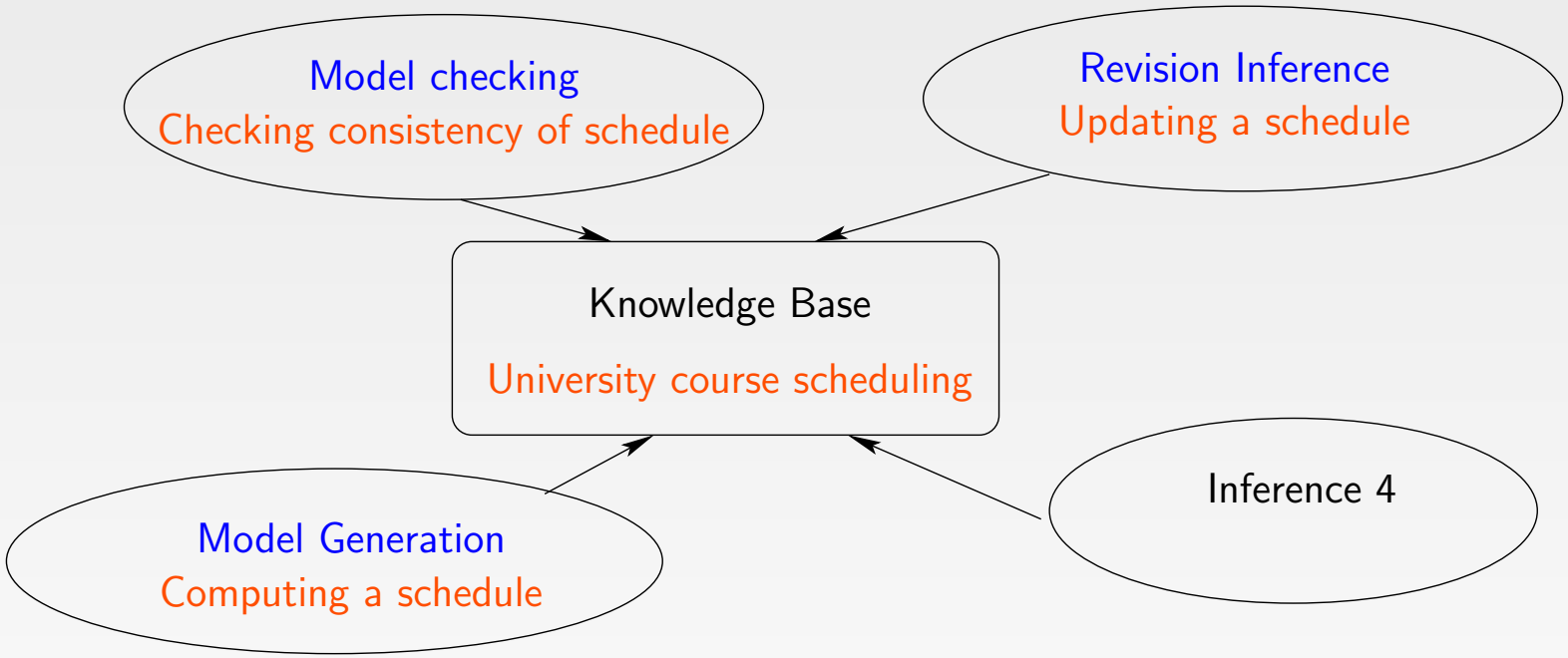
- Manages a declarative Knowledge Base
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System



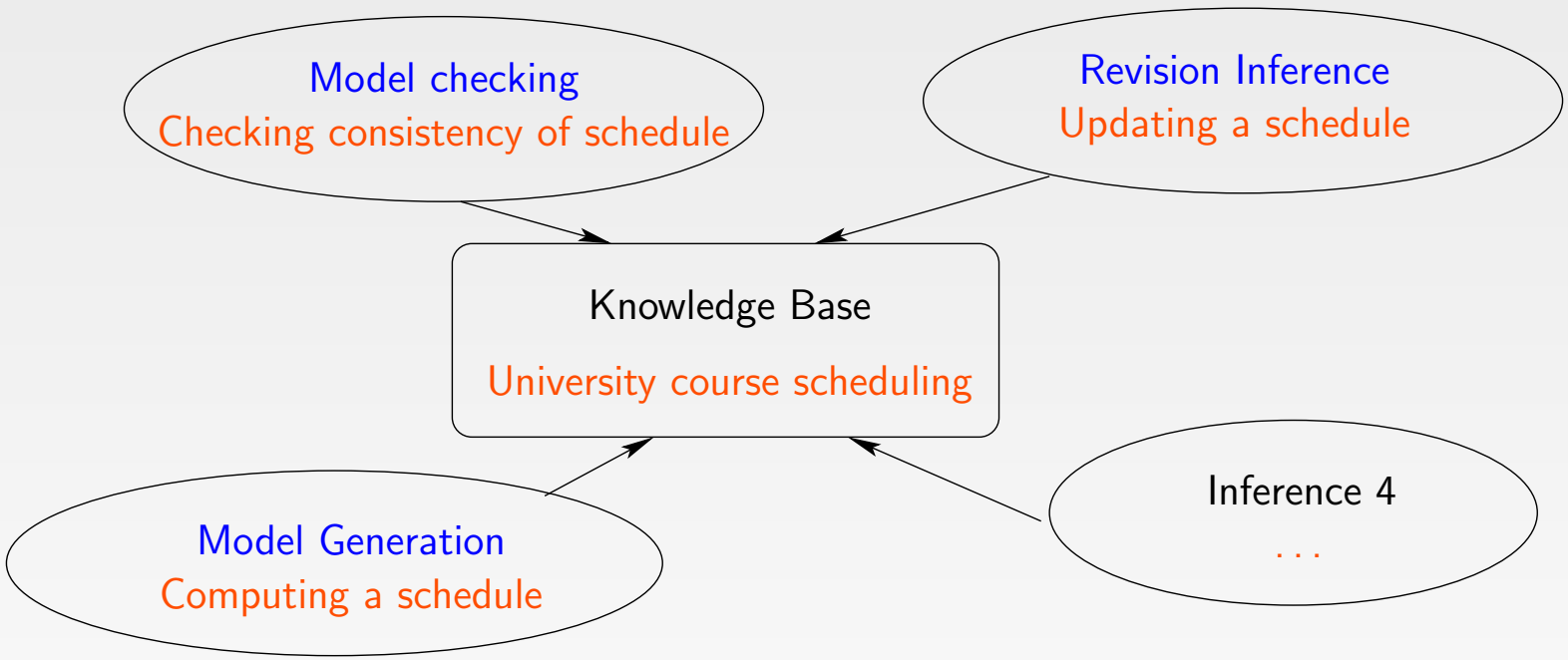
- Manages a declarative Knowledge Base
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System



- Manages a declarative Knowledge Base
- Equiped with different forms of inference to solve different types of tasks.

A Knowledge Base System



- Manages a declarative Knowledge Base
- Equiped with different forms of inference to solve different types of tasks.

KBS versus Declarative progamming paradigms

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

KBS versus Declarative progamming paradigms

- 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 goes beyond this by allowing multiple forms of inference
  - 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.

KBS versus Declarative progamming paradigms

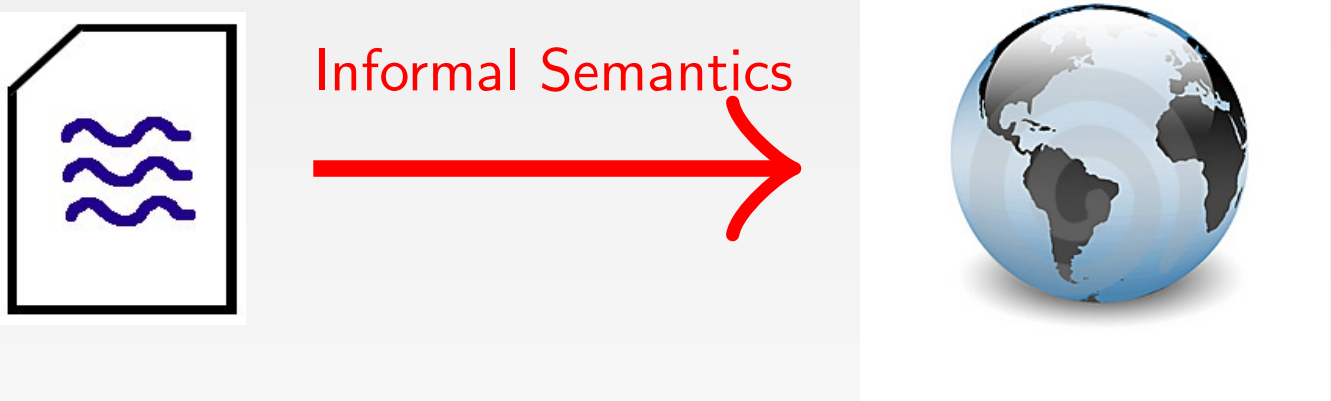
- Declarative programming paradigms: ASP, LP, CLP, ...
  - A declarative language + unique form of inference.
  - A declarative program encodes a solution for a problem.

- The KBS-paradigm goes beyond this by allowing multiple forms of inference
  - 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.

- This imposes a strong requirement on a KB-language:
  - Human experts need to be able to develop, interpret, maintain a KB purely on the basis of its declarative semantics.

A requirement on KB-language

In a KBS, the link between a KB-theory and what it states about the problem domain must be exceptionally clear.



Position 1

A strong requirement for KB-language:

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

Informal semantics?

- The informal semantics of FO.

Informal semantics?

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

Informal semantics?

- The informal semantics of FO.
  - $$\forall x(Human(x) \supset Male(x) \vee Female(x))$$
  - The informal semantics of this FO sentence is perfectly clear:

    Humans are male or female.

Informal semantics?

- The informal semantics of FO.
  - $$\forall x(Human(x) \supset Male(x) \vee Female(x))$$
  - The informal semantics of this FO sentence is perfectly clear:

    Humans are male or female.

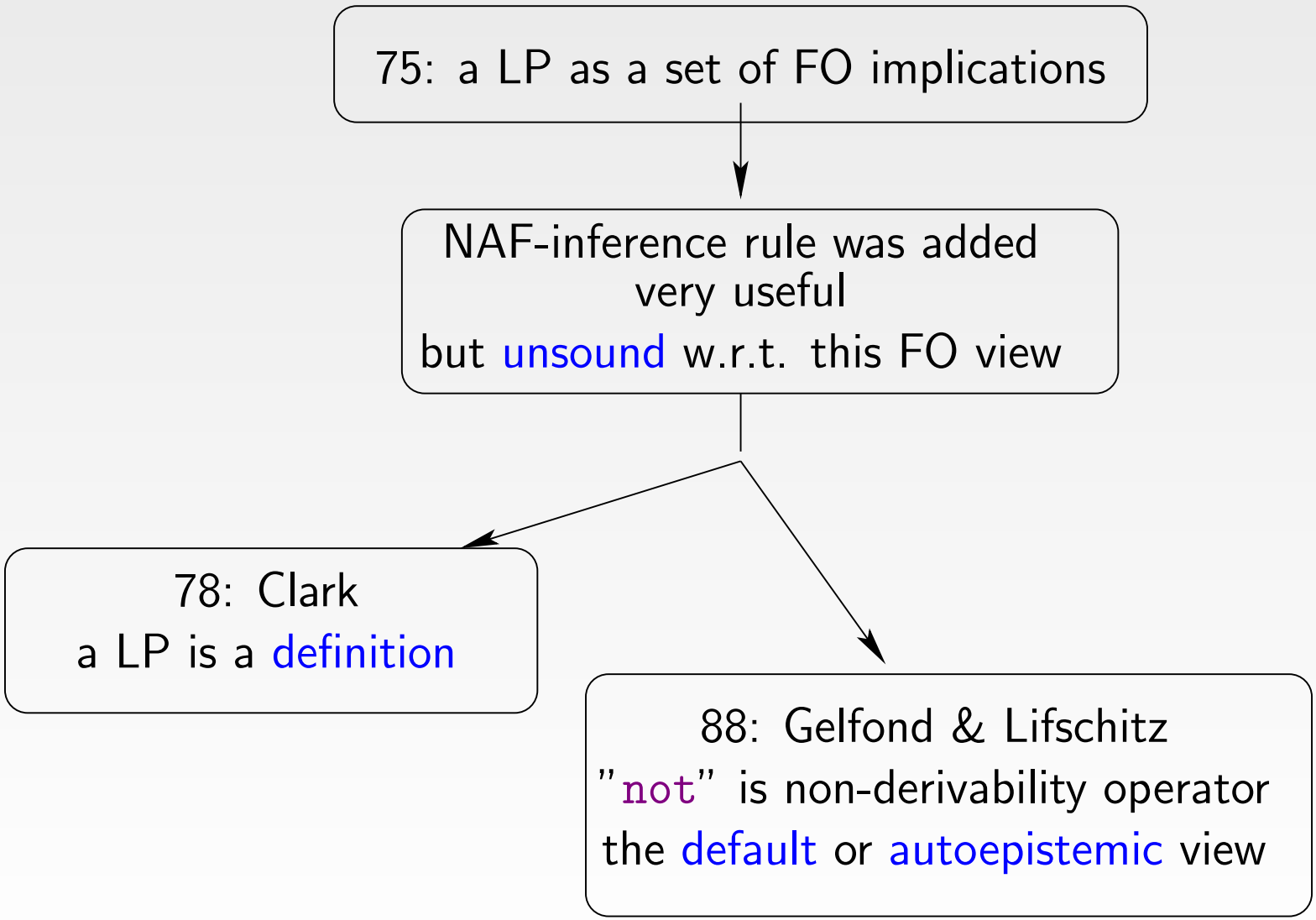
FO satisfies the requirement for a KB-language.

LP as a KB-language?

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 fairly blurred concept.
- If we look 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 ..."

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 ..."
- a logic program as a definition
  - 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 but ...

Informal semantics of LP

- Both are internally consistent views on the LP-formalism, with their own merits but . . .
- We embrace the definitional view.

Claim

The definition view yields an informal semantics

- of mathematical precision
- with wider applicability.

Informal semantics of LP

- Both are internally consistent views on the LP-formalism, with their own merits but . . .
- We embrace the definitional view.

Claim

The definition view yields an informal semantics

- of mathematical precision
- with wider applicability.

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

Informal semantics of LP

- Both are internally consistent views on the LP-formalism, with their own merits but ...
- We embrace the definitional view.

Claim

The definition view yields an informal semantics

- of mathematical precision
- with wider applicability.

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

- What is the right formal semantics?

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 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 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 $\mathfrak{A} \models \alpha$);

- A definition as a set of informal rules (with negation in body)
  - 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 $\mathfrak{A} \models \alpha$);

- A definition as a set of informal rules (with negation in body)
  - One rule specifies a sufficient condition
  - Together, they form a necessary condition.
- More accurately, an inductive definition defines a relation by describing how to construct it.
  - Rules as productions, to be applied iteratively.

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 $\mathfrak{A} \models \alpha$);

- A definition as a set of informal rules (with negation in body)
  - One rule specifies a sufficient condition
  - Together, they form a necessary condition.
- More accurately, an inductive definition defines a relation by describing how to construct it.
  - Rules as productions, to be applied iteratively.
- Definitions may have "parameters" and be very generic.
  - 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.

FO(ID)'s semantics of definitions

A language-filosophical thesis

(A parametrized variant of) the well-founded semantics correctly formalizes the common forms of inductive definitions in mathematics.

[Denecker 98, Denecker Bruynooghe Marek 2001, Denecker Ternovska 2007]

Definition of FO(ID)

Definition

A FO[ID]-theory is a set of FO-sentences and FO(ID)-definitions.

Definition of FO(ID)

Definition

A FO[ID]-theory is a set of FO-sentences and FO(ID)-definitions.

Claim

FO(ID) satisfies the requirement for a KB-language (having a clear and precise informal semantics)

FO(ID) for KR: motivation?

FO(ID) for KR: motivation?

- **Position 2**

  FO is the base KR language. Every interesting KR-language has a substantial overlap with FO.

FO(ID) for KR: motivation?

- **Position 2**

  FO is the base KR language. Every interesting KR-language has a substantial overlap with FO.

- **Position 3**

  (Inductive) definitions have many applications, not only in mathematics, but also in software domains and common sense KR.

FO(ID) for KR: motivation?

- **Position 2**

  FO is the base KR language. Every interesting KR-language has a substantial overlap with FO.

- **Position 3**

  (Inductive) definitions have many applications, not only in mathematics, but also in software domains and common sense KR.

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

Inductive definitions and common sense KR

- ID's are important in mathematics but are they useful for common sense KR?

Inductive definitions and common sense KR

- ID's are important in mathematics but are they useful for common sense KR?
-

Position 4

The concept of inductive definition is a natural, precise and very useful instance of CWA.

  - The CWA principle "every atom not derived by a rule is false" is also included in the principle of ID.

Inductive definitions and common sense KR

- ID's are important in mathematics but are they useful for common sense KR?
-

Position 4

The concept of inductive definition is a natural, precise and very useful instance of CWA.

  - The CWA principle "every atom not derived by a rule is false" is also included in the principle of ID.
  - The CWA underlying ID's is very similar to the form of CWA explicitly or implicitly used in many ASP applications.

Definitional rules as non-monotonic modules

Position 5

Definitional rules provide a useful form of nonmonotonic modularity.

- Incrementally building knowledge representations.
- Elaboration tolerance.

Turning FO(ID) into a KB-language

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

Turning FO(ID) into a KB-language

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge.
- $\Rightarrow$ FO(ID)

Turning FO(ID) into a KB-language

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge.
- $\Rightarrow$ FO(ID,Types)
  - Types

Turning FO(ID) into a KB-language

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge.
- $\Rightarrow$ FO(ID,Types,Agg)
  - Types
  - Aggregates

Turning FO(ID) into a KB-language

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge.
- $\Rightarrow$ FO(ID,Types,Agg,Arit)
  - Types
  - Aggregates
  - Arithmetic

Turning FO(ID) into a KB-language

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge.
- $\Rightarrow$ FO(ID,Types,Agg,Arit,ParFun)
  - Types
  - Aggregates
  - Arithmetic
  - Partial functions

Turning FO(ID) into a KB-language

- FO(ID) is not nearly "expressive" enough for a compact and modular representation of domain knowledge.
- $\Rightarrow$ FO(ID,Types,Agg,Arit,ParFun,...)
  - Types
  - Aggregates
  - Arithmetic
  - Partial functions
  - ...

FO($\cdot$)

Implementation of KBS

Forms of inference under development:

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

The IDP system

[Wittocx,Mariën,Denecker 2008]

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

The IDP system

[Wittocx,Mariën,Denecker 2008]

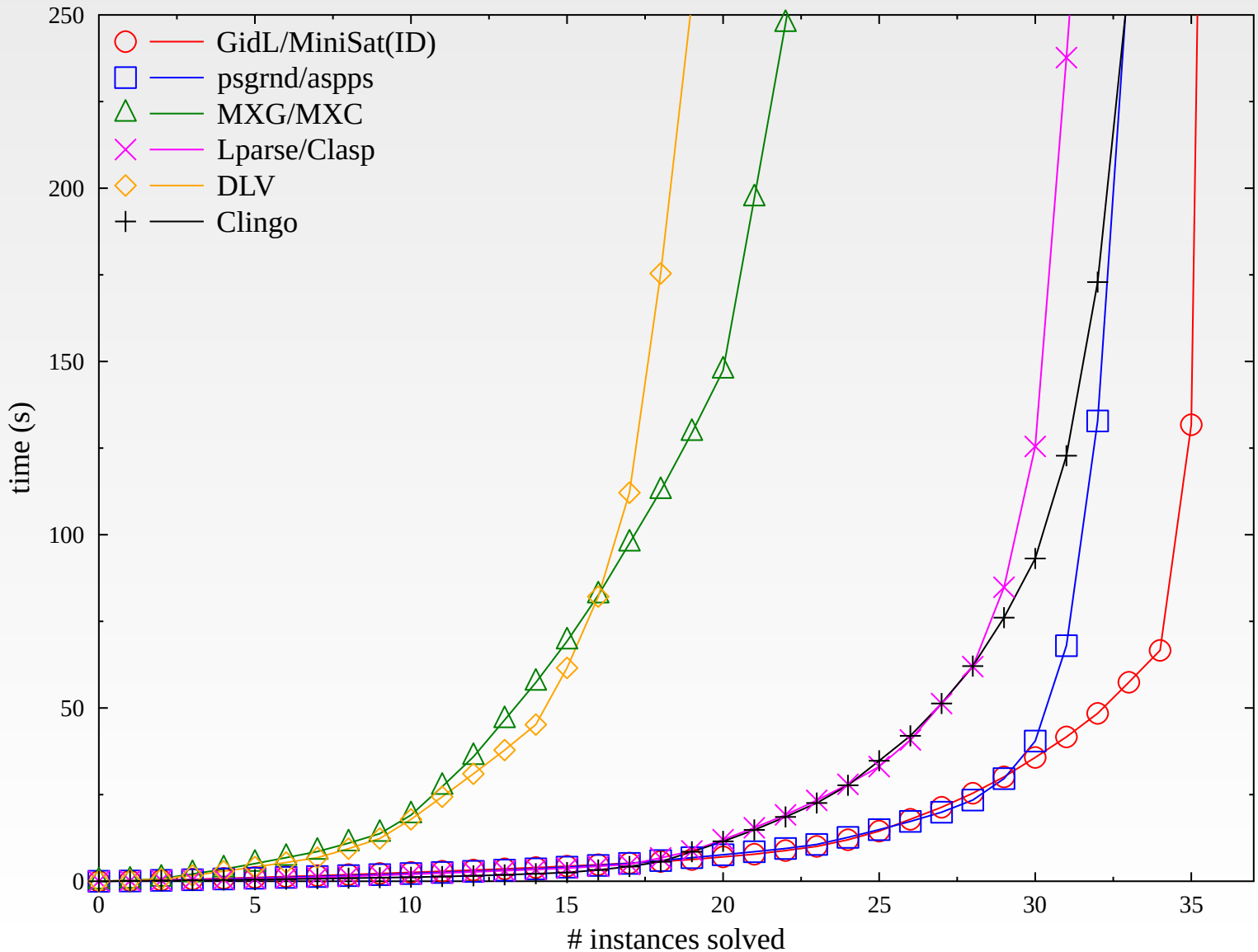
- Its purpose : generate models for a FO($\cdot$) theory with a given finite domain D.
- Technology: grounding + SAT + ASP technology
  - Incorporating state-of-the-art stable semantics algorithms in MiniSat.

The IDP system

[Wittocx,Mariën,Denecker 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, ...
  - According to our tests, the fastest ASP system.

LaSh'08 experiments



Conclusion

Why caring about FO(ID)?

Conclusion

Why caring about FO(ID)?

- For the elegance and generality of its syntax.
- For the clarity and precision of its informal semantics
  - FO(ID) can be taught without explaining its formal semantics.
- For the many occurrences of definitions in applications
- For efficiency (in the long run)
  - Technology of the SAT and SMT communities.

Conclusion

Why caring about FO(ID)?

- For the elegance and generality of its syntax.
- For the clarity and precision of its informal semantics
  - FO(ID) can be taught without explaining its formal semantics.
- For the many occurrences of definitions in applications
- For efficiency (in the long run)
  - Technology of the SAT and SMT communities.

- Integrating LP with FO is necessary, in the long run.
  - To explain the role and contribution of LP to the larger KR community.
  - For the unity and coherence of our science.

Relation to other LP formalisms

FO(ID) is continuing other traditionFO(ID) is continuing other traditions in LP: s in LP:

- LP formalisms where the definition view is fitting:
  - Abductive Logic Programming
  - Deductive databases
- These formally correspond to fragments of FO(ID)

FO(ID) and ASP side by side

Although ASP and FO(ID) are conceptually very different, in practical use they are quite similar.
Hamiltonian path

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

Confusion about well-founded semantics

- Mismatch between well-founded semantics and FO
  - A LP has a unique, three-valued well-founded model which seems a bad fit with FO's multiple 2-valued models.
  - (As opposed to stable semantics, which can have multiple 2-valued stable models.)
- Solution:
  - The parametrised version of well-founded semantics does not fix the interpretation of the parameters of a definition, and therefore allows multiple models.
  - Three-valued well-founded models are not accepted, only 2-valued models.

Limits

- In mathematics, not every set of informal rules specifies a correct definition:

  We define $\mathfrak{A} \models \varphi$ by structural induction:

  - ...
  - $\mathfrak{A} \models \psi$ if $\mathfrak{A} \models \psi \wedge \phi$.
  - $\mathfrak{A} \models \psi$ if $\mathfrak{A} \not\models \neg\psi$.

- Mathematically unacceptable: this is not structural induction because it defines satisfaction of formulas in terms of satisfaction of larger formulas.
- It is this type of definitions for which the formal versions have 3-valued well-founded semantics.
- In FO(ID) we do the same as mathematicians: we reject the definition. If IDP discovers that a definition has a 3-valued WFM, it writes a error message.