The IDP System: What and why?

Bart Bogaerts
AI Lab, Vrije Universiteit Brussel

October 19, 2021

ARTIFICIAL
INTELLIGENCE
RESEARCH GROUP

Based on presentations of Marc Denecker

1. Introduction

2. Separating Knowledge from Problem: The KB Paradigm

3. FO(·), Informal Semantics and Inductive Definitions

4. Demo: IDP for Blocks World

5. Outlook

1. Introduction
2. Separating Knowledge from Problem: The KB Paradigm
3. FO(·), Informal Semantics and Inductive Definitions
4. Demo: IDP for Blocks World
5. Outlook

WHAT IS THE IDP SYSTEM?

- A Constraint Programming System
  (lazy clause generation solver)
- An Answer Set Solving System
  (ASP competitions, inductive definitions)
- A theorem prover
  (building on first-order logic)

WHAT IS THE IDP SYSTEM?

- A Constraint Programming System
  (lazy clause generation solver)
- An Answer Set Solving System
  (ASP competitions, inductive definitions)
- A theorem prover
  (building on first-order logic)
- None (or all) of the above:
  A knowledge base system

SOME DRIVING PRINCIPLES BEHIND IDP

- The Knowledge Base Paradigm: Knowledge and Problem should be separated
- Clear and precise Informal Semantics
- Focus on domain experts (rather than modelling experts)
  (limited core language, symmetry, function detection, grounding techniques, definitional structure)

SEPARATING KNOWLEDGE FROM PROBLEMS

One issue that fragments computational logic more than anything else:

the reasoning/inference task

STATE OF THE ART

- For every type of reasoning task, a new logic (or more than one):
  - Classical first order logic (FO):
    deduction
  - Deductive Databases (SQL, Datalog):
    query answering & other database operations
  - Answer set Programming (ASP):
    answer set computation
  - Abductive Logic Programming:
    abduction
  - Constraint Programming (CP):
    constraint solving
  - Planning languages PDDL :
    planning
  - Temporal logics :
    model checking
  - . . .

STATE OF THE ART

A declarative proposition:

Each lecturer teaches at least one course in the first bachelor

STATE OF THE ART

A declarative proposition:

Each lecturer teaches at least one course in the first bachelor

What is its purpose? What task is it to be used for?

- It could be a query to a database.

STATE OF THE ART

A declarative proposition:

Each lecturer teaches at least one course in the first bachelor

What is its purpose? What task is it to be used for?

- It could be a query to a database.
- It could be a constraint in a course assignment problem.

STATE OF THE ART

A declarative proposition:

Each lecturer teaches at least one course in the first bachelor

What is its purpose? What task is it to be used for?

- It could be a query to a database.
- It could be a constraint in a course assignment problem.
- It could be a desired property, to be proven from a formal specification of the course assignment domain.
- ...

STATE OF THE ART

A declarative proposition:

Each lecturer teaches at least one course in the first bachelor

What is its purpose? What task is it to be used for?

- It could be a query to a database.
- It could be a constraint in a course assignment problem.
- It could be a desired property, to be proven from a formal specification of the course assignment domain.
- ...

Depending on the task to be solved, we need a different system and a different logic to represent this proposition.

Is declarative knowledge not independent of the task (and hence, of a specific form of inference) ?

SEPARATING KNOWLEDGE FROM PROBLEMS

A logic theory is a bag of (descriptive) information.

- A logic theory cannot be executed.
- A logic theory is not a program.
- A logic theory is not a representation of a problem.

SEPARATING KNOWLEDGE FROM PROBLEMS

A logic theory is a bag of (descriptive) information.

- A logic theory cannot be executed.
- A logic theory is not a program.
- A logic theory is not a representation of a problem.

We use knowledge to solve a problem by applying the appropriate form of *inference*.

AN ILLUSTRATION

- What is the central “knowledge” in the graph coloring problem?
  - No two adjacent vertices have the same color.

AN ILLUSTRATION

- What is the central “knowledge” in the graph coloring problem?
  - No two adjacent vertices have the same color.
  - Represented in FO (classical first order logic)?

AN ILLUSTRATION

- What is the central “knowledge” in the graph coloring problem?
  - No two adjacent vertices have the same color.
  - Represented in FO (classical first order logic)?

$$\forall x \forall y (G(x, y) \Rightarrow Col(x) \neq Col(y))$$

AN ILLUSTRATION

- What is the central “knowledge” in the graph coloring problem?
  - No two adjacent vertices have the same color.
  - Represented in FO (classical first order logic)?

$$\forall x \forall y (G(x, y) \Rightarrow Col(x) \neq Col(y))$$

- How to solve a graph coloring problem in FO?
- What kind of inference do we need to apply to this formula to solve this problem?

"What kind of inference do we need to solve this problem?"

- A question that until recently, for FO, was not asked.

“What kind of inference do we need to solve this problem?”

- A question that until recently, for FO, was not asked.
- Many saw FO as the logic of deductive reasoning.
  - In some fields, this is still the dominating view.

“What kind of inference do we need to solve this problem?”

- A question that until recently, for FO, was not asked.
- Many saw FO as the logic of deductive reasoning.
  - In some fields, this is still the dominating view.
- Deduction is utterly useless for solving the graph coloring problem.

"What kind of inference do we need to solve this problem?"

- A question that until recently, for FO, was not asked.
- Many saw FO as the logic of deductive reasoning.
  - In some fields, this is still the dominating view.
- Deduction is utterly useless for solving the graph coloring problem.
- Instead, people developed new logics to handle problems like this:
  - Constraint Programming Languages
    - Ilog, Zinc, Constraint Logic Programming, ...
  - Answer Set Programming (ASP).

“What kind of inference do we need to solve this problem?”

- A question that until recently, for FO, was not asked.
- Many saw FO as the logic of deductive reasoning.
  - In some fields, this is still the dominating view.
- Deduction is utterly useless for solving the graph coloring problem.
- Instead, people developed new logics to handle problems like this:
  - Constraint Programming Languages
    - Ilog, Zinc, Constraint Logic Programming, . . .
  - Answer Set Programming (ASP).
- Form of inference needed here:

model generation/expansion

Shouldn’t it be possible to solve multiple types of tasks using the same language?

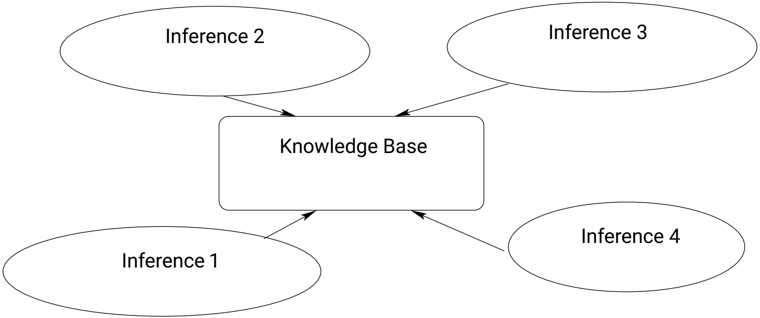
SEPARATING KNOWLEDGE FROM PROBLEMS

A logic theory is a bag of (descriptive) information.

- A logic theory cannot be executed.
- A logic theory is not a program.
- A logic theory is not a representation of a problem.

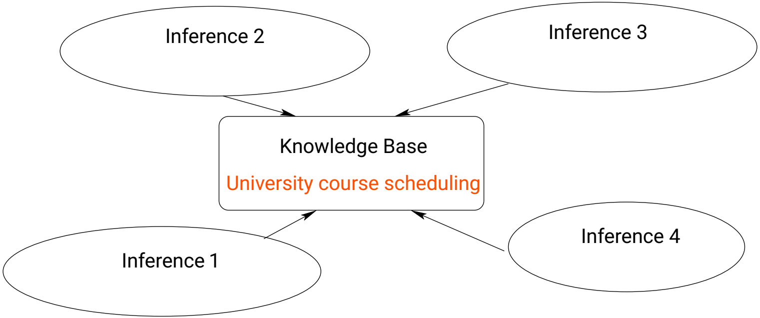
We use knowledge to solve a problem by applying the appropriate form of *inference*.

A KNOWLEDGE BASE SYSTEM (KBS)



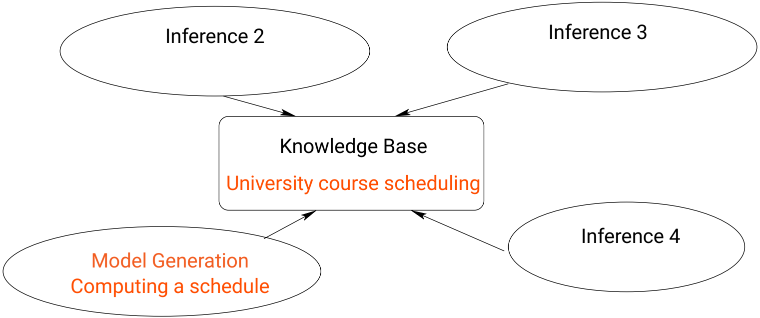
- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference:

A KNOWLEDGE BASE SYSTEM (KBS)



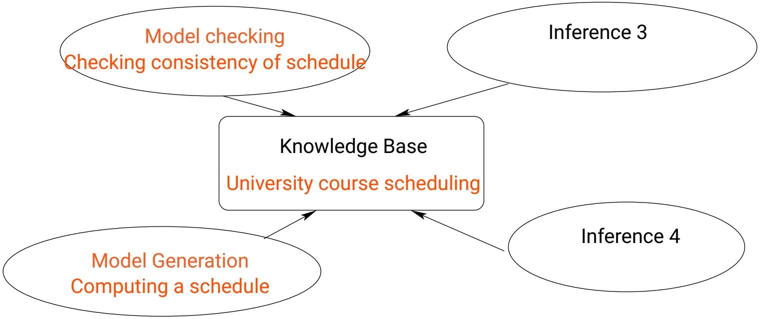
- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference:

A KNOWLEDGE BASE SYSTEM (KBS)



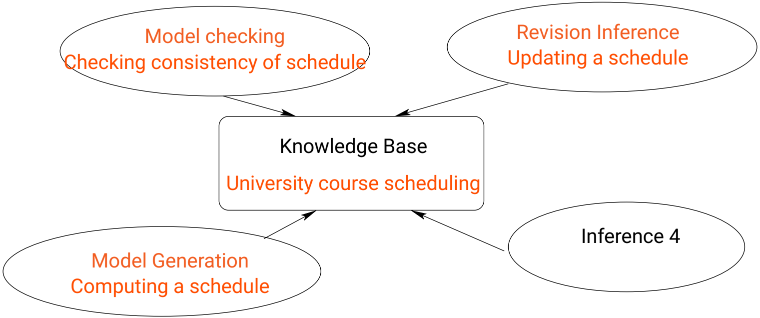
- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference:
  - Model generation: Computing a schedule

A KNOWLEDGE BASE SYSTEM (KBS)



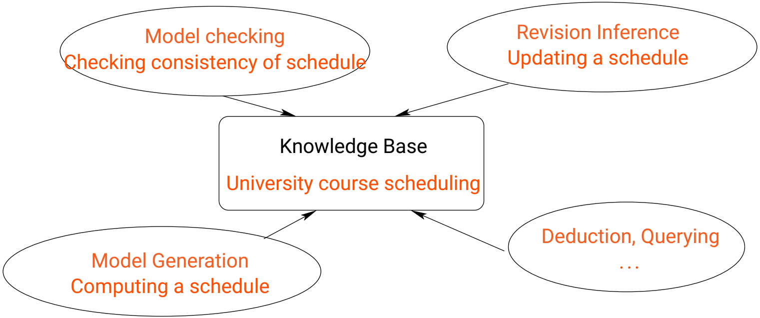
- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference:
  - Model generation: Computing a schedule
  - Model checking: Verifying consistency of a schedule

A KNOWLEDGE BASE SYSTEM (KBS)



- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference:
  - Model generation: Computing a schedule
  - Model checking: Verifying consistency of a schedule
  - Update and Revision: Updating a given schedule

A KNOWLEDGE BASE SYSTEM (KBS)



- Manages a declarative Knowledge Base (KB): a theory
- Equiped with different forms of inference:
  - Model generation: Computing a schedule
  - Model checking: Verifying consistency of a schedule
  - Update and Revision: Updating a given schedule
  - Deduction for verification of the KB
    Querying of defined predicates, ...

THE FO(·)-KBS PROJECT

- On the logical level: FO(·)
  - Study “knowledge” by principled development of expressive KR languages
  - Clear informal semantics
  - Expressive language, rich enough so that the information, relevant to solve a problem CAN be represented.
  - Model-theoretic semantics, in the Tarskian style.
  - Classical first-order logic (FO) as foundation, extended where necessary.

THE FO($\cdot$)-KBS PROJECT

- On the logical level: FO($\cdot$)
  - Study “knowledge” by principled development of expressive KR languages
  - Clear informal semantics
  - Expressive language, rich enough so that the information, relevant to solve a problem CAN be represented.
  - Model-theoretic semantics, in the Tarskian style.
  - Classical first-order logic (FO) as foundation, extended where necessary.

(FO($\cdot$)= family of extensions of FO)

THE FO(·)-KBS PROJECT

- On the application level:
  - Towards a typology of tasks and computational problems in terms of (the same) logic and inference.
  - Eagerly searching for novel ways of using declarative specifications to solve problems.

THE FO(·)-KBS PROJECT

- On the inference level:
  - Building solvers for various forms of inference for FO(·)
  - Integrating various solving techniques from various declarative programming paradigms in one Knowledge Base System.
    (In IDP3, with the Lua scripting language)

1. Introduction
2. Separating Knowledge from Problem: The KB Paradigm
3. FO(·), Informal Semantics and Inductive Definitions
4. Demo: IDP for Blocks World
5. Outlook

WHY FO AS A FOUNDATION ?

FO: the language that failed in the seventies?

- Too expressive for building “practical” systems?
  - Undecidability
  - Expressivity/Efficiency trade-off

- FO is not suitable for describing common sense knowledge?
  - Nonmonotonic reasoning

- FO as a language is too difficult for practical use?
  - E.g., quantifiers

WHY FO AS A FOUNDATION ?

- FO, the outcome of 18's and 19's century's research in

  "laws of thought"

  - E.g., Leibniz, De Morgan, Boole, Frege, Peirce

WHY FO AS A FOUNDATION ?

- FO, the outcome of 18's and 19's century's research in

  "laws of thought"

  - E.g., Leibniz, De Morgan, Boole, Frege, Peirce

- FO is about a small set of connectives:

$$\wedge, \vee, \neg, \forall, \exists, \Leftrightarrow, \Rightarrow$$

  - Essential for KR, the right semantics in FO

WHY FO AS A FOUNDATION ?

- FO, the outcome of 18's and 19's century's research in

  "laws of thought"

  - E.g., Leibniz, De Morgan, Boole, Frege, Peirce

- FO is about a small set of connectives:

  $$\wedge, \vee, \neg, \forall, \exists, \Leftrightarrow, \Rightarrow$$

  - Essential for KR, the right semantics in FO

- Crystal clear informal semantics

  $$\forall x(Human(x) \Rightarrow Man(x) \vee Woman(x))$$

  means

  All humans are men or women

WHY FO AS A FOUNDATION ?

- FO, the outcome of 18's and 19's century's research in

  "laws of thought"

  - E.g., Leibniz, De Morgan, Boole, Frege, Peirce

- FO is about a small set of connectives:

  $$\wedge, \vee, \neg, \forall, \exists, \Leftrightarrow, \Rightarrow$$

  - Essential for KR, the right semantics in FO

- Crystal clear informal semantics

  $$\forall x(Human(x) \Rightarrow Man(x) \vee Woman(x))$$

  means

  All humans are men or women

- Model semantics as a way to formalize meaning.

CLAIMS

(1) Every expressive declarative modelling language has a substantial overlap with FO, in one form or the other.

CLAIMS

(1) Every expressive declarative modelling language has a substantial overlap with FO, in one form or the other.

- For some languages, the syntax, conceptuology, terminology may obscure the relationship.
  - SQL
  - ALLOY
  - Zinc
  - Answer Set Programming

CLAIMS

(1) Every expressive declarative modelling language has a substantial overlap with FO, in one form or the other.

- For some languages, the syntax, conceptuology, terminology may obscure the relationship.
  - SQL
  - ALLOY
  - Zinc
  - Answer Set Programming

(2) But FO is not enough for practical KR.

COMPUTATIONAL COMPLEXITY

- Deductive reasoning in FO is undecidable.

COMPUTATIONAL COMPLEXITY

- Deductive reasoning in FO is undecidable.
- Much richer “logics” exist and are in use in industry (SQL, ILOG, Zinc, ProB)
- They are used for simpler forms of inference: tractable (P) or almost (NP).
- Such forms of inference have apparently many applications than deduction.

COMPUTATIONAL COMPLEXITY

- Deductive reasoning in FO is undecidable.
- Much richer “logics” exist and are in use in industry (SQL, ILOG, Zinc, ProB)
- They are used for simpler forms of inference: tractable (P) or almost (NP).
- Such forms of inference have apparently many applications than deduction.

In a knowledge-centered logic, we must accept that some sort of problems are untractable or undecidable, while other problems are solvable.

- That is life, that is how knowledge is.
- We still hope to be able to solve many problems.

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

$\Rightarrow$ FO(   )

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

$\Rightarrow$ FO(Types)

- Types

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

$\Rightarrow$ FO(Types,ID)

- Types
- (Inductive) Definitions

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

⇒ FO(Types,ID,Agg)

  - Types
  - (Inductive) Definitions
  - Aggregates

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

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

- Types
- (Inductive) Definitions
- Aggregates
- (Bounded) Arithmetic

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

⇒ FO(Types,ID,Agg,Arit,FD)

  - Types
  - (Inductive) Definitions
  - Aggregates
  - (Bounded) Arithmetic
  - Coinductive Definitions

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

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

  - Types
  - (Inductive) Definitions
  - Aggregates
  - (Bounded) Arithmetic

  - Coinductive Definitions
  - Modal operators

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

$\Rightarrow$ FO(Types,ID,Agg,Arit,FD,Mod,HO,...)

  - Types
  - (Inductive) Definitions
  - Aggregates
  - (Bounded) Arithmetic

  - Coinductive Definitions
  - Modal operators
  - Higher Order logic
  - ...

The FO(·) language framework

FO(·): TURNING FO INTO A PRACTICAL KR LANGUAGE

- FO does not suffice for knowledge representation, modelling, specification.

⇒ FO(Types,ID,Agg,Arit,FD,Mod,HO,...)
  - Types
  - **(Inductive) Definitions**
  - Aggregates
  - (Bounded) Arithmetic

  - Coinductive Definitions
  - Modal operators
  - Higher Order logic
  - ...

SOME PROTOTYPICAL INDUCTIVE DEFINITIONS.

The two most common forms of ID's.

Monotone induction

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

Induction over well-founded order

We define $\mathfrak{A} \models \varphi$ by induction on the structure of φ

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

PROPERTIES OF INFORMAL ID'S

- Linguistically, a set of informal rules (with negation)
- Semantically, two principles:
  - Non-constructively, the least set closed under rule application
  - Constructively, the set obtained by iterated rule application.
- These two principles coincide – Tarski!

PROPERTIES OF INFORMAL ID´S

- Linguistically, a set of informal rules (with negation)
- Semantically, two principles:
  - Non-constructively, the least set closed under rule application
  - Constructively, the set obtained by iterated rule application.
- These two principles coincide – Tarski!
- Only for monotone definitions!

INFORMAL (INDUCTIVE) DEFINITIONS (ID´S)

- Definitions in mathematics: a special sort of knowledge:
  - of mathematical precision
  - broadly used
  - intuitively well understood
  - but not scientifically well-understood

ADDING ID'S TO FO

- Inductive definitions frequently occur in KR and formal specifications
- ID's cannot be expressed in FO in general.
  - Compactness theorem

$\Rightarrow$ It is necessary to extend FO with them.

FO(ID) (DENECKER 2000, DENECKER& TERNOVSKA 2008)

An FO(ID) theory:

- FO sentences
- Definitions: sets of rules

FO(ID) (DENECKER 2000, DENECKER& TERNOVSKA 2008)

An FO(ID) theory:

- FO sentences
- Definitions: sets of rules

Example:

$$\left\{ \begin{array}{l} \forall x \forall y (R(x,y) \leftarrow G(x,y)) \\ \forall x \forall y (R(x,y) \leftarrow \exists z (G(x,z) \land R(z,y))) \end{array} \right\}$$
$$\forall x \forall y R(x,y)$$

expresses that ...

FO(ID) (DENECKER 2000, DENECKER& TERNOVSKA 2008)

An FO(ID) theory:

- FO sentences
- Definitions: sets of rules

Example:

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

expresses that ... R is the reachability graph of graph G and G is a connected graph

FO(ID) (DENECKER 2000, DENECKER& TERNOVSKA 2008)

An FO(ID) theory:

- FO sentences
- Definitions: sets of rules

Example:

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

$$\forall x \forall y R(x,y)$$

expresses that ... R is the reachability graph of graph G and G is a connected graph

Claim (KR 2014)

Rules under well-founded semantics provide a uniform formalism for expressing the most common forms of definitions.

FO(ID) AS A RULE FORMALISM

- Many rule-based formalisms
  - Logic Programming
  - Datalog
  - Answer Set Programming
  - Description logics with rules
  - Abductive Logic Programming
  - Business rule systems
- FO(ID) overlaps with many of them and provides two precise, well-understood declarative sorts of rules
  - Material implications, definitional rules

1. Introduction
2. Separating Knowledge from Problem: The KB Paradigm
3. FO($\cdot$), Informal Semantics and Inductive Definitions
4. Demo: IDP for Blocks World
5. Outlook

SMALL DEMONSTRATION

Block's World Application

- Blocks can either be on the table or stacked
- Robot (arm) can grab a block to move it (and later put it down)
- Entire stacks can be moved
- If too large a stack is picked up, it falls down (on the table)

BLOCK'S WORLD VOCABULARY

Types

- Time points (discrete time)
- Objects (including table

Discrete time (functions)

- Start point
- Successor

Relations

- Fluents (On, Holds)
- Actions (Take, Put)
- Derived Fluents (Above, Fall)

BLOCK'S WORLD THEORIES

Some different theories

- Theory describing the temporal domain
- Goal theory (for planning)
- Invariant theory (to prove)

BLOCK'S WORLD DEMO

Let's look at

- IDP Syntax
- Language Features (Definitions, aggregates, . . .)
- Different inference methods
- Lua programming environment

1. Introduction
2. Separating Knowledge from Problem: The KB Paradigm
3. FO(·), Informal Semantics and Inductive Definitions
4. Demo: IDP for Blocks World
5. Outlook

FUTURE

- IDP3 (as demonstrated) no longer actively maintained
- New version coming
  - MinisatID replaced by Z3
  - Lua replaced by python
  - Strong focus on extensibility of the language

MAIN PUBLICATIONS

- IDP:

  B. De Cat, B. Bogaerts, M. Bruynooghe, G. Janssens and M. Denecker Predicate Logic as a Modeling Language: The IDP System. Chapter in Declarative Logic Programming: Theory, Systems, and Applications, p. 279-323, 2018.

- The KBS-paradigm:

  Denecker, Marc; Vennekens, Joost. Building a knowledge base system for an integration of logic programming and classical logic, ICLP 2008

- The logic FO(ID):

  Denecker, Marc; Ternovska, Eugenia. A logic of nonmonotone inductive definitions, ACM Transactions on Computational Logic, volume 9, issue 2, 2008.

- The well-founded semantics works well for IDs:

  Marc Denecker, Joost Vennekens: The Well-Founded Semantics Is the Principle of Inductive Definition, Revisited. KR 2014