

# Deciding Equivalence between Extended Datalog Programs. A Brief Survey.

Stefan Woltran  
*Technische Universität Wien*  
*A-1040 Vienna, Austria*

Joint work with:  
Thomas Eiter, Wolfgang Faber, Michael Fink, Hans Tompits

March 16, 2010

# Introduction

## The Answer-Set Programming Paradigm

- Answer-Set Programming (ASP) is proposed as declarative problem solving paradigm (term was coined by V. Lifschitz [1999,2002]).
- ASP has its roots in KR, logic programming, non-monotonic reasoning and deductive databases (Datalog).

## General Idea

- Reduce solving a problem instance to computing models of a corresponding theory / program.



# Introduction (ctd.)

## Answer-Set Programming

- We deal here with **non-monotonic logic programs** under **the answer-set semantics** (also known as stable model semantics).
  - ▶ easy formulation of concepts like transitive closure or inertia;
  - ▶ availability of efficient solvers, like DLV or Smodels.
- Numerous successful applications have been realized
  - ▶ planning, semantic-web reasoning, information integration, ...
- Realizations via domain-specific languages
  - ▶ front-end compiles problem instance from the domain into a logic program

# Introduction (ctd.)

## Shortcomings

- Currently there is a lack of support for **engineering ASP solutions**:
  - ▶ (offline) optimization techniques, testing and verification issues, modular programming methods.
- Main obstacle on the theoretical side:
  - ▶ Due to non-monotonicity, **equivalence** is a weaker concept than, e.g., in classical logic (no replacement-theorem!)

$$P_1 \equiv P_2 \not\Rightarrow Q \cup P_1 \equiv Q \cup P_2$$

- In other words, in ASP it is not possible to apply (standard) semantics to program fragments (or program parts).

# Motivation

## Example

$\{ \text{edge}(a, b) . \text{edge}(b, c) . \dots \}$   $KB$

$\left\{ \begin{array}{l} \text{path}(X, Y) :- \text{edge}(X, Y) . \\ \text{path}(X, Z) :- \text{path}(X, Y), \text{edge}(Y, Z) . \end{array} \right\}$   $Q1$

$\left\{ \begin{array}{l} \text{path}(X, Y) :- \text{edge}(X, Y) . \\ \text{path}(X, Z) :- \text{path}(X, Y), \text{path}(Y, Z) . \end{array} \right\}$   $Q2$

## Ordinary Equivalence (OE)

- Do  $Q1 \cup KB$  and  $Q2 \cup KB$  have the same output?

## More interesting problem: Query Equivalence (QE)

- Do  $Q1 \cup KB$  and  $Q2 \cup KB$  have the same output, for each  $KB$  (i.e., for any set of edges)?

# Motivation

## Example

$\{ \text{edge}(a, b) . \text{edge}(b, c) . \dots \}$   $KB$

$\left\{ \begin{array}{l} \text{path}(X, Y) :- \text{edge}(X, Y) . \\ \text{path}(X, Z) :- \text{path}(X, Y), \text{edge}(Y, Z) . \end{array} \right\}$   $Q1$

$\left\{ \begin{array}{l} \text{path}(X, Y) :- \text{edge}(X, Y) . \\ \text{path}(X, Z) :- \text{path}(X, Y), \text{path}(Y, Z) . \end{array} \right\}$   $Q2$

## Ordinary Equivalence (OE)

- Do  $Q1 \cup KB$  and  $Q2 \cup KB$  have the same output?

## More interesting problem: Query Equivalence (QE)

- Do  $Q1 \cup KB$  and  $Q2 \cup KB$  have the same output, for **each**  $KB$  (i.e., for any set of edges)?

## Motivation (ctd.)

### Example

$\{ \text{edge}(a, b) . \text{edge}(b, c) . \text{path}(c, d) . \dots \}$   $KB$

$\left\{ \begin{array}{l} \text{path}(X, Y) :- \text{edge}(X, Y) . \\ \text{path}(X, Z) :- \text{path}(X, Y) , \text{edge}(Y, Z) . \end{array} \right\}$   $Q1$

$\left\{ \begin{array}{l} \text{path}(X, Y) :- \text{edge}(X, Y) . \\ \text{path}(X, Z) :- \text{path}(X, Y) , \text{path}(Y, Z) . \end{array} \right\}$   $Q2$

### More interesting problem: Query Equivalence (QE)

- Do  $Q1 \cup KB$  and  $Q2 \cup KB$  have the same output, for each  $KB$  (i.e., for any set of edges)?

### Different problem: Uniform Equivalence (UE)

- Do  $Q1 \cup KB$  and  $Q2 \cup KB$  have the same output, for **any set of facts** (i.e., also paths may be part of the input)?

## Motivation (ctd.)

### A different view on the example ...

$$\left\{ \begin{array}{l} \text{edge}(a, b) . \text{edge}(b, c) . \dots \\ \dots \quad :- \quad \text{path}(X, Y) . \\ \dots \quad :- \quad \dots \end{array} \right\} \quad P$$

$$\left\{ \begin{array}{l} \text{path}(X, Y) \quad :- \quad \text{edge}(X, Y) . \\ \text{path}(X, Z) \quad :- \quad \text{path}(X, Y) , \text{edge}(Y, Z) . \end{array} \right\} \quad M1$$

$$\left\{ \begin{array}{l} \text{path}(X, Y) \quad :- \quad \text{edge}(X, Y) . \\ \text{path}(X, Z) \quad :- \quad \text{path}(X, Y) , \text{path}(Y, Z) . \end{array} \right\} \quad M2$$

### Strong Equivalence (SE)

- Do  $M1 \cup P$  and  $M2 \cup P$  have the same output for **any program**  $P$ ?

## Motivation (ctd.)

### A different view on the example ...

$$\left\{ \begin{array}{l} \text{edge}(a, b) . \text{edge}(b, c) . \dots \\ \dots \quad :- \text{path}(X, Y) . \\ \dots \quad :- \dots \end{array} \right\} \quad P$$

$$\left\{ \begin{array}{l} \text{path}(X, Y) \quad :- \text{edge}(X, Y) . \\ \text{path}(X, Z) \quad :- \text{path}(X, Y) , \text{edge}(Y, Z) . \end{array} \right\} \quad M1$$

$$\left\{ \begin{array}{l} \text{path}(X, Y) \quad :- \text{edge}(X, Y) . \\ \text{path}(X, Z) \quad :- \text{path}(X, Y) , \text{path}(Y, Z) . \end{array} \right\} \quad M2$$

### Application specific equivalence

- Do  $M1 \cup P$  and  $M2 \cup P$  have the same output (for each  $P$ , where `edge` appears only in rule heads and `path` only in rule bodies)?

# “Traditional” Datalog

## Terminology

- Datalog = ASP with Horn programs.

## Results for Datalog

- QE is undecidable (if domain is infinite) [Shmueli; SIGMOD 1987].
- UE is decidable [Sagiv, 1988].
- UE and SE coincide [Maher, 1988] (thus also SE decidable).
- OE and UE are complete for EXP [Eiter,Gottlob,Manilla; TODS 1997].

## Results for general ASP?

- How do these results carry over to Datalog **extensions**?
  - (default) negation in rule bodies;
  - disjunctions in rule heads.

# ”Traditional” Datalog

## Terminology

- Datalog = ASP with Horn programs.

## Results for Datalog

- QE is undecidable (if domain is infinite) [Shmueli; SIGMOD 1987].
- UE is decidable [Sagiv, 1988].
- UE and SE coincide [Maher, 1988] (thus also SE decidable).
- OE and UE are complete for EXP [Eiter,Gottlob,Manilla; TODS 1997].

## Results for general ASP?

- How do these results carry over to Datalog **extensions**?
  - ▶ (default) negation in rule bodies;
  - ▶ disjunctions in rule heads.

# Background

## Syntax

- A (disjunctive logic) program is a finite set of rules

$$r : a_1 \vee \dots \vee a_l \leftarrow b_1, \dots, b_m, \text{not } c_1, \dots, \text{not } c_n;$$

- ▶  $a_i, b_j, c_k$  are atoms over terms built from countable sets of constants (domain) and variables.
- ▶ **not** denotes default negation.

## Rule/Program Satisfaction

- Let  $I \subseteq At$  be an interpretation, i.e., a set of ground atoms. Then,
  - ▶  $I \models r$  iff  $I \cap hd(r) \neq \emptyset$  whenever  $bd^+(r) \subseteq I$  and  $I \cap bd^-(r) = \emptyset$ ;
  - ▶  $I \models P$  iff  $I \models r$ , for each  $r \in P$ ;  $I$  is then called a **model** of  $P$ .

$$(hd(r) = \{a_1, \dots, a_l\}, bd^+(r) = \{b_1, \dots, b_m\}, bd^-(r) = \{c_1, \dots, c_n\})$$

# Background

## Syntax

- A (disjunctive logic) program is a finite set of rules

$$r : a_1 \vee \dots \vee a_l \leftarrow b_1, \dots, b_m, \mathbf{not} c_1, \dots, \mathbf{not} c_n;$$

- ▶  $a_i, b_j, c_k$  are atoms over terms built from countable sets of constants (domain) and variables.
- ▶ **not** denotes default negation.

## Rule/Program Satisfaction

- Let  $I \subseteq At$  be an interpretation, i.e., a set of ground atoms. Then,
  - ▶  $I \models r$  iff  $I \cap hd(r) \neq \emptyset$  whenever  $bd^+(r) \subseteq I$  and  $I \cap bd^-(r) = \emptyset$ ;
  - ▶  $I \models P$  iff  $I \models r$ , for each  $r \in P$ ;  $I$  is then called a **model** of  $P$ .

$$(hd(r) = \{a_1, \dots, a_l\}, bd^+(r) = \{b_1, \dots, b_m\}, bd^-(r) = \{c_1, \dots, c_n\})$$

## Background (ctd.)

### Answer Sets

- Define the **reduct**,  $P^I$ , of a program  $P$  w.r.t.  $I$  as

$$P^I = \{hd(r) \leftarrow bd^+(r) \mid r \in Gr(P), bd^-(r) \cap I = \emptyset\}.$$

- $I$  is an **answer set** of  $P$  iff  $I$  is a minimal model of  $P^I$ .
  - ▶ The set of all answer sets of  $P$  is denoted by  $AS(P)$ .

### Intuition

- 1st Step: One assumes an interpretation  $I$  and evaluates default negation with respect to  $I$ ;
- 2nd Step: The remaining program has to justify  $I$ , i.e. each atom true in  $I$  has to be derived by  $P^I$ .

## Background (ctd.)

### Strong Equivalence (SE) [Lifschitz, Pearce, Valverde; TOCL 2001]

- Two programs  $P$ ,  $Q$  are **strongly equivalent** iff, for any program  $R$ ,  $AS(P \cup R) = AS(Q \cup R)$ .

### Uniform Equivalence (UE) [Eiter, Fink; ICLP 2003]

- Two programs  $P$ ,  $Q$  are **uniformly equivalent** iff, for any finite set  $F$  of facts,  $AS(P \cup F) = AS(Q \cup F)$ .
  - ▶ A fact is a ground rule of form  $a \leftarrow$ .

# Strong Equivalence vs. Uniform Equivalence

Example: Consider Programs  $P_1, P_2$

$$P_1 = \{a \vee b \leftarrow\} \quad P_2 = \{a \leftarrow \text{not } b; b \leftarrow \text{not } a\}.$$

We have  $AS(P_1) = AS(P_2) = \{\{a\}, \{b\}\}$ .

$P_1$  and  $P_2$  are not strongly equivalent

- Take  $R = \{a \leftarrow b, b \leftarrow a\}$ .

Then,  $AS(P_1 \cup R) = \{\{a, b\}\}$  and  $AS(P_2 \cup R) = \emptyset$ .

- ▶  $(P_1 \cup R)^{\{a,b\}} = P_1 \cup R \Rightarrow \{a, b\} \in AS(P_1 \cup R);$
- ▶  $(P_2 \cup R)^{\{a,b\}} = R \Rightarrow \{a, b\} \notin AS(P_2 \cup R).$

$P_1$  and  $P_2$  are uniformly equivalent

- Let  $F$  be any set of facts:

- ▶ for  $F \cap \{a, b\} = \emptyset$ ,  $AS(P_1 \cup F) = AS(P_2 \cup F) = \{\{a\} \cup F, \{b\} \cup F\};$
- ▶ otherwise  $AS(P_1 \cup F) = AS(P_2 \cup F) = \{F\}.$

# Strong Equivalence vs. Uniform Equivalence

Example: Consider Programs  $P_1, P_2$

$$P_1 = \{a \vee b \leftarrow\} \quad P_2 = \{a \leftarrow \text{not } b; b \leftarrow \text{not } a\}.$$

We have  $AS(P_1) = AS(P_2) = \{\{a\}, \{b\}\}$ .

$P_1$  and  $P_2$  are not strongly equivalent

- Take  $R = \{a \leftarrow b, b \leftarrow a\}$ .

Then,  $AS(P_1 \cup R) = \{\{a, b\}\}$  and  $AS(P_2 \cup R) = \emptyset$ .

- ▶  $(P_1 \cup R)^{\{a,b\}} = P_1 \cup R \Rightarrow \{a, b\} \in AS(P_1 \cup R)$ ;
- ▶  $(P_2 \cup R)^{\{a,b\}} = R \Rightarrow \{a, b\} \notin AS(P_2 \cup R)$ .

$P_1$  and  $P_2$  are uniformly equivalent

- Let  $F$  be any set of facts:

- ▶ for  $F \cap \{a, b\} = \emptyset$ ,  $AS(P_1 \cup F) = AS(P_2 \cup F) = \{\{a\} \cup F, \{b\} \cup F\}$ ;
- ▶ otherwise  $AS(P_1 \cup F) = AS(P_2 \cup F) = \{F\}$ .

# Strong Equivalence vs. Uniform Equivalence

Example: Consider Programs  $P_1, P_2$

$$P_1 = \{a \vee b \leftarrow\} \quad P_2 = \{a \leftarrow \text{not } b; b \leftarrow \text{not } a\}.$$

We have  $AS(P_1) = AS(P_2) = \{\{a\}, \{b\}\}$ .

$P_1$  and  $P_2$  are not strongly equivalent

- Take  $R = \{a \leftarrow b, b \leftarrow a\}$ .

Then,  $AS(P_1 \cup R) = \{\{a, b\}\}$  and  $AS(P_2 \cup R) = \emptyset$ .

- ▶  $(P_1 \cup R)^{\{a,b\}} = P_1 \cup R \Rightarrow \{a, b\} \in AS(P_1 \cup R);$
- ▶  $(P_2 \cup R)^{\{a,b\}} = R \Rightarrow \{a, b\} \notin AS(P_2 \cup R).$

$P_1$  and  $P_2$  are uniformly equivalent

- Let  $F$  be any set of facts:

- ▶ for  $F \cap \{a, b\} = \emptyset$ ,  $AS(P_1 \cup F) = AS(P_2 \cup F) = \{\{a\} \cup F, \{b\} \cup F\};$
- ▶ otherwise  $AS(P_1 \cup F) = AS(P_2 \cup F) = \{F\}.$

# Strong Equivalence

Theorem [Eiter,Fink,Tompits,W; AAI 2005]

- Given program  $P$ , define its *extended active domain* as  $U_P^+ = U_P \cup \{c_1, \dots, c_n\}$ , where  $c_i$  are new constants from  $U$  and  $n$  is the max. number of variables in a rule of  $P$ .
- Then,  $P$  and  $Q$  are strongly equivalent iff  
 $G(P, U_{P \cup Q}^+)$  is strongly equivalent to  $G(Q, U_{P \cup Q}^+)$ .

SE for ground case is well understood [Turner, LPNMR 2001]

- Let  $SE(P) = \{(X, Y) \mid X \subseteq Y, X \models P^Y, Y \models P\}$ . Then  $P$  and  $Q$  are strongly equivalent iff  $SE(P) = SE(Q)$ .
- (This results reformulates the seminal result in [Lifschitz,Pearce,Valverde; TOCL 2001] which was given in terms of logic HT).

# Strong Equivalence

Theorem [Eiter,Fink,Tompits,W; AAI 2005]

- Given program  $P$ , define its *extended active domain* as  $U_P^+ = U_P \cup \{c_1, \dots, c_n\}$ , where  $c_i$  are new constants from  $U$  and  $n$  is the max. number of variables in a rule of  $P$ .
- Then,  $P$  and  $Q$  are strongly equivalent iff

$G(P, U_{P \cup Q}^+)$  is strongly equivalent to  $G(Q, U_{P \cup Q}^+)$ .

SE for ground case is well understood [Turner, LPNMR 2001]

- Let  $SE(P) = \{(X, Y) \mid X \subseteq Y, X \models P^Y, Y \models P\}$ . Then  $P$  and  $Q$  are strongly equivalent iff  $SE(P) = SE(Q)$ .
- (This results reformulates the seminal result in [Lifschitz,Pearce,Valverde; TOCL 2001] which was given in terms of logic HT).

# Strong Equivalence (ctd.)

## Consequence

- Strong equivalence between non-ground programs remains *decidable* (even for countable domain).
- In fact, the problem is  $\text{coNEXP}$ -complete.
  - ▶ Implementation [Eiter,Faber,Traxler; LPNMR 2005]: reduction to a decidable FO-logic fragment (as suggested by [Lin; KR 2002]).
- If predicate arity is bounded, the problem is only  $\Pi_2^P$ -complete [Eiter,Faber,Fink,W; AMAI 2007].

## Alternative Characterisations

- Via first-order variants of logic HT [Lifschitz,Pearce,Valverde; LPNMR 2007. Pearce,Valverde; ICLP 2008].

# Strong Equivalence (ctd.)

## Consequence

- Strong equivalence between non-ground programs remains *decidable* (even for countable domain).
- In fact, the problem is  $\text{coNEXP}$ -complete.
  - ▶ Implementation [Eiter,Faber,Traxler; LPNMR 2005]: reduction to a decidable FO-logic fragment (as suggested by [Lin; KR 2002]).
- If predicate arity is bounded, the problem is only  $\Pi_2^P$ -complete [Eiter,Faber,Fink,W; AMAI 2007].

## Alternative Characterisations

- Via first-order variants of logic HT [Lifschitz,Pearce,Valverde; LPNMR 2007. Pearce,Valverde; ICLP 2008].

# Uniform Equivalence

## Results [Eiter,Fink,Tompits,W; AAI 2005, IJCAI 2007]

- Uniform equivalence is *undecidable* for *normal programs*.
  - ▶ Shown via a mapping of QE between linear Horn programs (undecidable, see [Feder,Saraiya; ICDT 1992]) to UE between normal programs;
  - ▶ Undecidability holds already for definite Horn programs which contain a single constraint with a negative body atom.
- A slightly different mapping (using [Shmueli; PODS 1987]) yields undecidability of UE between disjunctive programs of bounded predicate arities.

## Characterisation

- via UE-models (for any finite grounding):  
$$UE(P) = \{(X, Y) \in SE(P) \mid \forall Z : X \subset Z \subset Y \Rightarrow (Z, Y) \notin SE(P)\}.$$
- Alternative characterisations in terms of FO-HT [Fink; TPLP 2010].

# Uniform Equivalence

## Results [Eiter,Fink,Tompits,W; AAI 2005, IJCAI 2007]

- Uniform equivalence is *undecidable* for *normal programs*.
  - ▶ Shown via a mapping of QE between linear Horn programs (undecidable, see [Feder,Saraiya; ICDT 1992]) to UE between normal programs;
  - ▶ Undecidability holds already for definite Horn programs which contain a single constraint with a negative body atom.
- A slightly different mapping (using [Shmueli; PODS 1987]) yields undecidability of UE between disjunctive programs of bounded predicate arities.

## Characterisation

- via UE-models (for any finite grounding):  
$$UE(P) = \{(X, Y) \in SE(P) \mid \forall Z : X \subset Z \subset Y \Rightarrow (Z, Y) \notin SE(P)\}.$$
- Alternative characterisations in terms of FO-HT [Fink; TPLP 2010].

## Strong Equivalence vs. Uniform Equivalence (ctd.)

For positive programs SE and UE coincide

- Observation: reduct makes the difference.
- For positive programs,  $P^Y = P$  holds for any  $Y$ .
- Thus,  $(X, Y) \in SE(P)$  yields  $(X, X) \in SE(P)$ .
- It follows that  $SE(P) \neq SE(Q)$  implies  $UE(P) \neq UE(Q)$ .
- Hence, UE implies SE; the other direction is by definition.

Consequence:

- UE remains decidable for positive disjunctive programs.

## Strong Equivalence vs. Uniform Equivalence (ctd.)

For positive programs SE and UE coincide

- Observation: reduct makes the difference.
- For positive programs,  $P^Y = P$  holds for any  $Y$ .
- Thus,  $(X, Y) \in SE(P)$  yields  $(X, X) \in SE(P)$ .
- It follows that  $SE(P) \neq SE(Q)$  implies  $UE(P) \neq UE(Q)$ .
- Hence, UE implies SE; the other direction is by definition.

Consequence:

- UE remains decidable for positive disjunctive programs.

# Overview of Results

## General case / bounded predicate arity

	disj.	positive	normal	Horn
SE	$\text{coNEXP}/\Pi_2^P$	$\text{coNEXP}/\Pi_2^P$	$\text{coNEXP}/\Pi_2^P$	EXP/coNP
UE	undec./undec.	$\text{coNEXP}/\Pi_2^P$	undec./?	EXP/coNP
OE	$\text{coNEXP}^{\text{NP}}/\Pi_3^P$	$\text{coNEXP}^{\text{NP}}/\Pi_3^P$	$\text{coNEXP}/\Pi_2^P$	EXP/coNP

[Eiter,Fink,Tompits,W; AAI 2005, IJCAI 2007; Eiter, Faber, Fink, W; AMAI 2007].

# Overview of Results

## General case / bounded predicate arity

	disj.	positive	normal	Horn
SE	$\text{coNEXP}/\Pi_2^P$	$\text{coNEXP}/\Pi_2^P$	$\text{coNEXP}/\Pi_2^P$	EXP/coNP
UE	undec./undec.	$\text{coNEXP}/\Pi_2^P$	undec./?	EXP/coNP
OE	$\text{coNEXP}^{\text{NP}}/\Pi_3^P$	$\text{coNEXP}^{\text{NP}}/\Pi_3^P$	$\text{coNEXP}/\Pi_2^P$	EXP/coNP

[Eiter,Fink,Tompits,W; AAI 2005, IJCAI 2007; Eiter, Faber, Fink, W; AMAI 2007].

# Stratification

## Interesting observation for ground programs

- SE (also UE) between stratified programs is coNP (thus as hard as for normal programs) [Eiter,Fink,Tompits,W; IJCAI 2007].
- for the hardness reduction in the case of SE the compared programs either have to be cyclic or possess different stratifications.

## Non-ground programs . . .

- UE is decidable for
  - ▶ monadic programs (follows from [Halevy,Mumick,Sagiv,Shmueli; JACM 2001]);
  - ▶ programs with joint stratification [Levy,Sagiv; VLDB 1993].
- but the general case still open.

# Stratification

## Interesting observation for ground programs

- SE (also UE) between stratified programs is coNP (thus as hard as for normal programs) [Eiter,Fink,Tompits,W; IJCAI 2007].
- for the hardness reduction in the case of SE the compared programs either have to be cyclic or possess different stratifications.

## Non-ground programs ...

- UE is decidable for
  - ▶ monadic programs (follows from [Halevy,Mumick,Sagiv,Shmueli; JACM 2001]);
  - ▶ programs with joint stratification [Levy,Sagiv; VLDB 1993].
- but the general case still open.

# Parameterised Notions of Equivalence

Definition [W; TPLP 2008. Truszczyński,W; TPLP 2009].

- $\mathcal{HB}(A, B)$  – all programs  $P$  such that  $hd(P) \subseteq A$  and  $bd(P) \subseteq B$
- Programs  $P$  and  $Q$  are  $(A, B)$ -equivalent, in symbols,  $\equiv_{(A,B)}$ , if for every program  $R \in \mathcal{HB}(A, B)$ ,  $AS(P \cup R) = AS(Q \cup R)$ .

Landscape for  $\equiv_{(A,B)}$  with  $A \subseteq B$  or  $B \subseteq A$

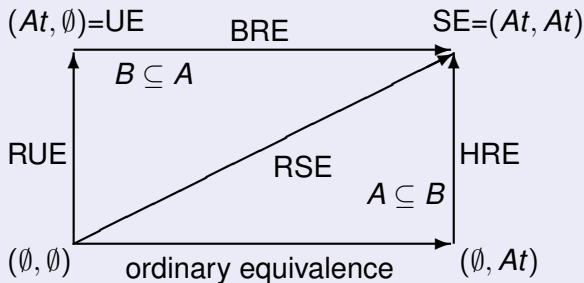


# Parameterised Notions of Equivalence

Definition [W; TPLP 2008. Truszczyński,W; TPLP 2009].

- $\mathcal{HB}(A, B)$  – all programs  $P$  such that  $hd(P) \subseteq A$  and  $bd(P) \subseteq B$
- Programs  $P$  and  $Q$  are  $(A, B)$ -equivalent, in symbols,  $\equiv_{(A,B)}$ , if for every program  $R \in \mathcal{HB}(A, B)$ ,  $AS(P \cup R) = AS(Q \cup R)$ .

Landscape for  $\equiv_{(A,B)}$  with  $A \subseteq B$  or  $B \subseteq A$



# Conclusion

## Summary

- Equivalence in Answer-Set Programming has received increasing interest in recent years.
- (Slightly) different access compared to datalog.
- Still, ASP can learn from datalog results.
  - ▶ Query-reachability [Levy,Sagiv; PODS 1992]
  - ▶ Recursion elimination [Gaifman,Mairson,Sagiv,Vardi; JACM 1993]
  - ▶ Bounding number of variables per rule [Vardi; PODS 1995]
- Open questions:
  - ▶ Decidability/undecidability frontier for UE (stratified programs, bounded arities)
  - ▶ Characterisation of stratified/acyclic/HCF programs in terms of SE/UE-models

## Conclusion (ctd.)

### Future Work: Put theory to practice!

- Software-Engineering tools for Answer-Set programmers
  - ▶ Modular ASP: Helsinki.
  - ▶ SE: Vienna.
- Does (offline-)optimization pay off?
  - ▶ ... how to improve grounders for ASP-systems.
- Use parameterized equivalence checks for program verification
  - ▶ already in use for logic-programming course at our institute
  - ▶ equivalence notions could support E-learning facilities

# Conclusion (ctd.)

Lot of further work has been done. . .

- Program transformations

- ▶ [Eiter,Fink,Tompits,W; LPNMR/KR 2004. Eiter,Fink,Tompits,Traxler,W; KR 2006. Lin,Chen; JAIR 2007. Wong; JAIR 2008. Pührer,Tompits; LPNMR 2009].

- Further language extensions (weak/weight constr., aggr.)

- ▶ [Turner; TPLP 2003. Liu,Truszczyński; JAIR 2006].

- Non-standard comparisons

- ▶ [Eiter,Tompits,W; IJCAI 2005. Oetsch,Tompits,W; AAAI 2007. Oetsch,Tompits; ICLP 2008].

- Logical underpinning

- ▶ [Lifschitz,Pearce,Valverde; TOCL 2001. De Jongh,Hendriks; TPLP 2003. Lifschitz,Pearce,Valverde; LPNMR 2007. Fink; ICLP 2008].

- Other LP semantics

- ▶ well-founded semantics [Cabalar,Odintsov,Pearce,Valverde; ICLP 2006], supported semantics [Truszczyński,W; AAAI 2008].