Coalgebra & Data

Clemens Kupke
University of Strathclyde
Glasgow, Scotland

Alcop 2015, Delft, 7 May 2015

Overview

- iteration-free coalgebraic PDL
  - brief overview
  - completeness
- Datalog$^{\pm}$
  - Intro: ontology-based data access & Datalog$^{\pm}$
  - the problem with negative information
  - normal Datalog$^{\pm}$
- Coalgebra & Data

Part 0: Basics of Coalgebraic Logics in 4 slides

Coalgebraic Modal Logic & PDL

- Observation: Kripke models are $\mathcal{P}$-coalgebras, ie, pairs (X, γ) with
$$\gamma : \mathrm{X} \to \mathcal{P}\mathrm{X}$$

- in this context X is usually a set

- Idea: Develop modal logic for T-coalgebras, where T is an endofunctor. Development should be parametric in T.

Coalgebraic Logic: Syntax

Given a modal similarity type Λ (ie., a collection of modal operators) and a set Var of propositional variables.

Definition

The set $\mathcal{F}(\Lambda)$ of formulas over Λ is defined a follows:

$$\mathcal{F}(\Lambda) \ni \varphi ::= \mathrm{p} \in \mathrm{Var} \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \heartsuit\varphi, \heartsuit \in \Lambda$$

Note

In this talk the (basic) similarity type will consist of one unary modality only!

Coalgebraic Logic: Semantics

In order to be able to interpret modal formulas we need

- a set functor T
- for every modal operator $\heartsuit \in \Lambda$ a natural transformation

$$\heartsuit : \mathrm{P} \longrightarrow \mathrm{PT},$$

where P denotes the contravariant power set functor.

Formulas are then interpreted over T-models $(\mathrm{X}, \gamma, \mathrm{V})$ consisting of

$$\gamma : \mathrm{X} \longrightarrow \mathrm{TX} \qquad \text{and} \qquad \mathrm{V} : \mathrm{Var} \longrightarrow \mathcal{P}(\mathrm{X}).$$

$$\begin{aligned} [\![\mathrm{p}]\!] &= \mathrm{V}(\mathrm{p}) \qquad \text{for } \mathrm{p} \in \mathrm{Var} \\ &\vdots \\ [\![\heartsuit\varphi]\!] &= \mathrm{P}\gamma(\heartsuit([\![\varphi]\!])) = \gamma^{-1}(\heartsuit([\![\varphi]\!])) \end{aligned}$$

Equivalently

$\heartsuit : \mathrm{P} \longrightarrow \mathrm{PT}$ is in one-to-one correspondence to

- $\widehat{\heartsuit} : \mathrm{T} \longrightarrow \mathrm{P}^{\mathrm{op}}\mathrm{P}$ (T-coalgebras to neighbourhood frames)

$$\mathrm{x} \models \heartsuit\varphi \quad \text{iff} \quad [\![\varphi]\!] \in (\widehat{\heartsuit} \circ \gamma)(\mathrm{x}).$$

- $\breve{\heartsuit} : \mathrm{T2} \longrightarrow 2$ ("allowed 0-1 patterns")

$$\begin{array}{ccccc} \mathrm{X} & & \xrightarrow{\chi_{[\![\varphi]\!]}} & & 2 \\ \downarrow{\scriptstyle \gamma} & & & & \\ \mathrm{T(X)} & & \xrightarrow{\mathrm{T}(\chi_{[\![\varphi]\!]})} & \mathrm{T(2)} & \xrightarrow{\breve{\heartsuit}} 2 \end{array}$$

$$(\mathrm{X}, \gamma, \mathrm{V}), \mathrm{x} \models \heartsuit\varphi \quad \text{iff} \quad \breve{\heartsuit}(\mathrm{T}(\chi_{[\![\varphi]\!]})(\mathrm{c}(\mathrm{x})) = 1.$$

Examples

- $\mathrm{T} = \mathcal{P}$, $\heartsuit = \Box$:

$$\begin{aligned} \heartsuit(\mathrm{U}) &= \{\mathrm{V} \subseteq \mathrm{X} \mid \mathrm{U} \subseteq \mathrm{V}\}, \\ \widehat{\heartsuit}(\mathrm{V}) &= \{\mathrm{U} \subseteq \mathrm{X} \mid \mathrm{U} \subseteq \mathrm{V}\} \text{ and} \\ \breve{\heartsuit}(\mathrm{V} \subseteq \mathcal{P}2) &= 1 \quad \text{iff} \quad 0 \notin \mathrm{V} \end{aligned}$$

- $\mathrm{T} = \mathcal{M}$, $\heartsuit = \Box$:

$$\begin{aligned} \heartsuit(\mathrm{U}) &= \{\mathrm{N} \in \mathcal{M}\mathrm{X} \mid \mathrm{U} \in \mathrm{N}\} \\ \widehat{\heartsuit}(\mathrm{N}) &= \mathrm{N} \\ \breve{\heartsuit}(\mathrm{N} \in \mathcal{M}2) &= 1 \quad \text{iff} \quad 1 \in \mathrm{N} \end{aligned}$$

$\vdots$

Part I: Coalgebraic PDL
(joint work H.H. Hansen, R.Leal)

Propositional Dynamic Logic (PDL)

Fischer & Ladner, 1977. Reason about program correctness.

$[\alpha]\varphi$ "after all successful executions of program α, φ holds"

- Syntax:

  formulas φ $\quad ::= \quad \mathrm{p} \in \mathrm{P}_0 \mid \neg\varphi \mid \varphi \vee \varphi \mid [\alpha]\varphi$

  programs $\alpha \in \mathrm{A}$ $\quad ::= \quad \mathrm{a} \in \mathrm{A}_0 \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?$

  composition $(;)$, choice $(\cup)$, iteration $(^*)$, tests $(\varphi?)$

- Multi-modal Kripke semantics: $\mathrm{M} = (\mathrm{X}, \{\mathrm{R}_\alpha \mid \alpha \in \mathrm{A}\}, \mathrm{V})$ where X is state space,
  - $\mathrm{R}_\alpha : \mathrm{X} \to \mathcal{P}(\mathrm{X})$ (relation, nondeterministic programs),
  - $\mathrm{V} \colon \mathrm{P}_0 \to \mathcal{P}(\mathrm{X})$ is a valuation.

$$\mathrm{M}, \mathrm{x} \models [\alpha]\varphi \quad \text{iff} \quad \forall \mathrm{y} \in \mathrm{X}.\, \mathrm{x}\mathrm{R}_\alpha\mathrm{y} \to \mathrm{M}, \mathrm{y} \models \varphi.$$

Standard PDL Models

- Def. $\mathrm{M} = (\mathrm{X}, \{\mathrm{R}_\alpha \mid \alpha \in \mathrm{A}\}, \mathrm{V})$ is standard if

$$\begin{aligned} \mathrm{R}_{\alpha;\beta} &= \mathrm{R}_\alpha \circ \mathrm{R}_\beta \ \text{(relation composition)} \\ \mathrm{R}_{\alpha\cup\beta} &= \mathrm{R}_\alpha \cup \mathrm{R}_\beta \\ \mathrm{R}_{\alpha^*} &= \mathrm{R}_\alpha^* \ \text{(reflexive, transitive closure)} \\ \mathrm{R}_{\varphi?} &= \{(\mathrm{x},\mathrm{x}) \mid \mathrm{x} \in [\![\varphi]\!]\} \end{aligned}$$

- Sound and (weakly) complete axiomatisation of standard models [Kozen & Parikh 1981]:
  PDL = Normal modal logic K (ML of Kripke frames) plus:

$$[\alpha;\beta]\varphi \leftrightarrow [\alpha][\beta]\varphi \qquad [\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$$

$$[\psi?]\varphi \leftrightarrow (\psi \to \varphi)$$

$$\varphi \wedge [\alpha][\alpha^*]\varphi \leftrightarrow [\alpha^*]\varphi \qquad \varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi) \to [\alpha^*]\varphi$$

Game Logic (GL)

Parikh, 1985. Strategic ability in determined 2-player games.

$\langle\gamma\rangle\varphi$ "player 1 has strategy in γ to ensure outcome satisfies φ"
("player 1 is effective for φ")

- Syntax: PDL syntax extended with dual operation on games:
  - $\gamma_1;\gamma_2$: play γ_1 then γ_2,
  - $\gamma_1 \cup \gamma_2$: player 1 chooses to play γ_1 or γ_2,
  - γ^*: player 1 chooses when to stop.
  - γ^{d}: players switch roles.
- Semantics: Game model $\mathrm{M} = (\mathrm{X}, \{\mathrm{E}_\gamma \mid \gamma \in \Gamma\}, \mathrm{V})$ where $\mathrm{E}_\gamma : \mathrm{X} \longrightarrow \mathcal{PP}(\mathrm{X})$ is monotonic neighbourhood function: If $\mathrm{U} \in \mathrm{E}_\gamma(\mathrm{x})$ and $\mathrm{U} \subseteq \mathrm{U}'$ then $\mathrm{U}' \in \mathrm{E}_\gamma(\mathrm{x})$.

  $\mathrm{U} \in \mathrm{E}_\gamma(\mathrm{x})$ iff player 1 is effective for U in γ starting in x.

  Modal semantics: $\mathrm{M}, \mathrm{x} \models \langle\gamma\rangle\varphi \quad$ iff $\quad [\![\varphi]\!] \in \mathrm{E}_\gamma(\mathrm{x})$

Standard GL Models

- Standard GL model: similar to PDL notion,

$$\mathrm{U} \in \mathrm{E}_{\gamma^{\mathrm{d}}}(\mathrm{x}) \text{ iff } \mathrm{X} \setminus \mathrm{U} \notin \mathrm{E}_{\gamma}(\mathrm{x}).$$

- GL = monotonic modal logic M (ML of mon. nbhd. frames) plus

$$\langle \gamma ; \delta \rangle \varphi \leftrightarrow \langle \gamma \rangle \langle \delta \rangle \varphi \qquad \langle \gamma \cup \delta \rangle \varphi \leftrightarrow \langle \gamma \rangle \varphi \vee \langle \delta \rangle \varphi$$

$$\langle \psi ? \rangle \varphi \leftrightarrow (\psi \wedge \varphi) \qquad \langle \gamma^{\mathrm{d}} \rangle \varphi \leftrightarrow \neg \langle \gamma \rangle \neg \varphi$$

$$\varphi \vee \langle \gamma \rangle \langle \gamma^{*} \rangle \varphi \rightarrow \langle \gamma^{*} \rangle \varphi \qquad \frac{\varphi \vee \langle \gamma \rangle \varphi \rightarrow \psi}{\langle \gamma^{*} \rangle \varphi \rightarrow \psi}$$

- Without dual: sound and (weakly) complete [Parikh 1985].
- Without iteration: sound and strongly complete [Pauly 2001].
- Completeness of full GL still open.

Towards Coalgebraic Dynamic Logic

Basic observation:

- $\mathcal{P}$ is monad $(\mathcal{P}, \eta, \mu)$ with:
  $\eta_X(x) = \{x\}, \quad \mu_X(\{U_i \mid i \in I\}) = \bigcup_{i \in I} U_i$.
- $\mathcal{M}$ is a monad $(\mathcal{M}, \eta, \mu)$ with:
  $$\begin{array}{lcl} \eta_X(x) & = & \{U \subseteq X \mid x \in U\} \\ \mu_X(W) & = & \{U \subseteq X \mid \eta_{\mathcal{P}(X)}(U) \in W\} \end{array}$$
- Composition of programs and games is Kleisli composition.

Basic setup:

- Action/program $X \to TX$ where T a Set-monad (T describes computation type, side-effects, ...)
- Sequential composition as Kleisli composition $*_T$.
- Multi-program setting: $X \to (TX)^A$ (A-labelled T-coalgebra) where A is a set of program labels.

Coalgebra-Algebra

Two perspectives:

$\xi \colon X \longrightarrow (TX)^A$	T^A-coalgebra, modal logic
$\widehat{\xi} \colon A \longrightarrow (TX)^X$	algebra homomorphism, program operations

Questions:

- What are "program" operations like $\cup$ and ^{d}?
- What is a standard model?
- Which compositionality axioms?
- How to prove soundness and completeness?

Pointwise Program Operations via Natural Operations

- An n-ary natural operation on T is a natural transformation $\sigma\colon \mathrm{T}^{\mathrm{n}} \longrightarrow \mathrm{T}$
- $\sigma\colon \mathrm{T}^{\mathrm{n}} \longrightarrow \mathrm{T}$ yields pointwise operation on $(\mathrm{TX})^{\mathrm{X}}$, e.g.,

$$\sigma^{\mathrm{X}}_{\mathrm{X}}(\mathrm{c}_1, \mathrm{c}_2)(\mathrm{x}) = \sigma_{\mathrm{X}}(\mathrm{c}_1(\mathrm{x}), \mathrm{c}_2(\mathrm{x}))$$

- Given finitary signature functor Σ,
  a natural Σ-algebra is natural transformation $\theta\colon \Sigma\mathrm{T} \longrightarrow \mathrm{T}$,
  and yields pointwise Σ-algebra $\theta^{\mathrm{X}}_{\mathrm{X}}\colon \Sigma((\mathrm{TX})^{\mathrm{X}}) \longrightarrow (\mathrm{TX})^{\mathrm{X}}$.

Natural and Pointwise Operations: Examples

Natural operations on $\mathcal{P}$:

- Union $\cup\colon \mathcal{P} \times \mathcal{P} \longrightarrow \mathcal{P}$ is a natural operation, since

$$f[U \cup U'] = f[U] \cup f[U'] \quad (\mathcal{P}f(U) = f[U])$$

  The pointwise extension of $\cup\colon \mathcal{P} \times \mathcal{P} \longrightarrow \mathcal{P}$ is union of relations $(R_1 \cup R_2)(x) = R_1(x) \cup R_2(x)$.

- Observation: Intersection and complement are not natural operations on $\mathcal{P}$.

Natural operations on $\mathcal{M}$:

- $\cup$ and $\cap$ (since preserved by f^{-1}).
- Dual operation ${}^{d}\colon \mathcal{M} \longrightarrow \mathcal{M}$ where for all $N \in \mathcal{M}(X)$, and $U \subseteq X$, $U \in N^{d}$ iff $X \setminus U \notin N$.
  Dual game operation is the pointwise extension.

Standard dynamic models

Given a countable set A_0 of atomic programs, and a signature functor Σ. Let $\mathrm{A} = \Sigma \cup \{;\}$-terms over A_0.

We define:

- Given natural algebra $\theta \colon \Sigma\mathrm{T} \longrightarrow \mathrm{T}$ then $\xi \colon \mathrm{X} \longrightarrow (\mathrm{TX})^{\mathrm{A}}$ is θ-standard iff

$$\widehat{\xi} \colon \mathrm{A} \longrightarrow (\mathrm{TX})^{\mathrm{X}} \quad \text{is a} \quad \Sigma\text{-algebra homomorphism.}$$

- If T is a monad, then $\xi \colon \mathrm{X} \longrightarrow (\mathrm{TX})^{\mathrm{A}}$ is ;-standard iff

$$\text{for all } \alpha, \beta \in \mathrm{A}, \quad \widehat{\xi}(\alpha;\beta) = \widehat{\xi}(\alpha) * \widehat{\xi}(\beta).$$

Sound Axioms for Pointwise Operations

- Example: PDL axiom for choice $[\alpha \cup \beta]\mathrm{p} \leftrightarrow [\alpha]\mathrm{p} \wedge [\beta]\mathrm{p}$.
- Idea: $\widehat{\heartsuit}\colon \mathrm{T} \to \mathcal{N}$ turns operations θ on T into operations χ on $\mathcal{N}$.

$$\begin{array}{ccc} \mathrm{T}^{\mathrm{n}} & \xRightarrow{\widehat{\heartsuit}^{\mathrm{n}}} & \mathcal{N}^{\mathrm{n}} \\ \Big\Downarrow{\scriptstyle \theta} & & \Big\Downarrow{\scriptstyle \chi} \\ \mathrm{T} & \xRightarrow{\widehat{\heartsuit}} & \mathcal{N} \end{array} \qquad \text{For example:} \qquad \begin{array}{ccc} \mathcal{P} \times \mathcal{P} & \xRightarrow{\widehat{\Box}^{\mathrm{n}}} & \mathcal{N} \times \mathcal{N} \\ \Big\Downarrow{\scriptstyle \cup} & & \Big\Downarrow{\scriptstyle \cap} \\ \mathcal{P} & \xRightarrow{\widehat{\Box}} & \mathcal{N} \end{array}$$

From $\chi\colon \mathcal{N}^{\mathrm{n}} \to \mathcal{N}$, we get rank-1 formula $\varphi(\chi, \alpha_1, \ldots, \alpha_{\mathrm{n}}, \mathrm{p})$ (not in this talk).

Lemma

If $\xi\colon \mathrm{X} \to (\mathrm{TX})^{\mathrm{A}}$ is θ-standard and $\chi\colon \mathcal{N}^{\mathrm{n}} \to \mathcal{N}$ is such that $\widehat{\heartsuit} \circ \theta = \chi \circ \widehat{\heartsuit}^{\mathrm{n}}$, then the rank-1 formula $[\underline{\theta}(\alpha_1, \ldots, \alpha_{\mathrm{n}})]\mathrm{p} \leftrightarrow \varphi(\chi, \alpha_1, \ldots, \alpha_{\mathrm{n}}, \mathrm{p})$ is valid in ξ.

Coalgebraic Logic (Def)

A (modal) logic is a triple $\mathcal{L} = (\Lambda, \mathcal{A}, \Theta)$ where

- Λ is a similarity type,
- $\mathcal{A} \subseteq \mathrm{Prop}(\Lambda(\mathrm{Prop}(\mathrm{Var})))$ is a set of rank-1 axioms, and
- $\Theta \subseteq \mathcal{F}(\Lambda)$ is a set of frame conditions

If $\varphi \in \mathcal{F}(\Lambda)$, we write $\vdash_{\mathcal{L}} \varphi$ if φ can be derived from $\mathcal{A} \cup \Theta$ with the help of propositional reasoning (tautologies + MP), uniform substitution, and the congruence rule.

$$\frac{\varphi \leftrightarrow \psi}{\heartsuit\varphi \leftrightarrow \heartsuit\psi}$$

Dynamic Syntax

Given

- Σ, a signature (functor).
- P_0, a countable set of atomic propositions.
- A_0, a countable set of atomic programs.

we define

$$\begin{array}{lll} \text{formulas } \mathcal{F}(\mathrm{P}_0, \mathrm{A}_0, \Sigma) \ni \varphi & ::= & \mathrm{p} \in \mathrm{P}_0 \mid \neg\varphi \mid \varphi \vee \varphi \mid [\alpha]\varphi \\ \text{programs } \mathrm{A}(\mathrm{P}_0, \mathrm{A}_0, \Sigma) \ni \alpha & ::= & \mathrm{a} \in \mathrm{A}_0 \mid \alpha;\alpha \mid \sigma(\alpha_1, \ldots, \alpha_\mathrm{n}) \end{array}$$

where $\sigma \in \Sigma$ is n-ary.

(Tests are incorporated later)

(T, θ)-Dynamic Logic

Given

- base logic $\mathcal{L}_{\mathrm{b}} = (\{\Box\}, \mathrm{Ax}(\Box, \mathrm{T}), \emptyset)$ (rank-1)
- $\theta \colon \Sigma\mathrm{T} \longrightarrow \mathrm{T}$ and set A_0 of atomic actions.

We define

$$\begin{aligned}
\Lambda &= \{[\alpha] \mid \alpha \in \mathrm{A}\},\\
\mathrm{Ax} &= \mathrm{Ax}(\Box, \mathrm{T})_{\mathrm{A}} \cup \text{“}\theta\text{-axioms”}\ ,\\
\mathrm{Fr} &= \{[\alpha;\beta]\mathrm{p} \leftrightarrow [\alpha][\beta]\mathrm{p} \mid \alpha, \beta \in \mathrm{A}, \text{some fresh } \mathrm{p} \in \mathrm{P}_0\},\\
\mathcal{L}(\theta) &= (\Lambda, \mathrm{Ax}, \emptyset),\\
\mathcal{L}(\theta, ;) &= (\Lambda, \mathrm{Ax}, \mathrm{Fr}).
\end{aligned}$$

$\mathcal{L}(\theta)$ and $\mathcal{L}(\theta, ;)$ are (T, θ)-dynamic logics over $\mathcal{L}_{\mathrm{b}}$.

Conditions for Soundness

Sequential composition axiom: $[\alpha;\beta]\mathrm{p} \leftrightarrow [\alpha][\beta]\mathrm{p}$.

Recall: $\widehat{\heartsuit} : \mathrm{T} \longrightarrow \mathrm{P}^{\mathrm{op}}\mathrm{P} \quad \overset{1-1}{\leftrightarrow} \quad \breve{\heartsuit} : \mathrm{T2} \longrightarrow 2$

Lemma

If $\xi : \mathrm{X} \longrightarrow (\mathrm{TX})^{\mathrm{A}}$ is ;-standard, and $\widehat{\heartsuit} : \mathrm{T} \longrightarrow \mathrm{P}^{\mathrm{op}}\mathrm{P}$ is a monad morphism, then the axiom $[\alpha;\beta]\mathrm{p} \leftrightarrow [\alpha][\beta]\mathrm{p}$ is valid in ξ, for all $\alpha, \beta \in \mathrm{A}$.

Remark:

- Kelly & Power, 1993: $\dfrac{\text{Monad morphism } \mathrm{T} \longrightarrow \mathrm{P}^{\mathrm{op}}\mathrm{P}}{\text{Eilenberg-Moore algebra } \mathrm{T2} \longrightarrow 2}$

Examples

- Example: $\heartsuit$ for Kripke $\Diamond$ corr. to free algebra $\mathcal{P}\mathcal{P}(1) \longrightarrow \mathcal{P}(1)$, so $\widehat{\heartsuit}\colon \mathcal{P} \longrightarrow \mathrm{P}^{\mathrm{op}}\mathrm{P}$ is monad morphism. Also $\neg\heartsuit\neg$.
- Example: Monotonic λ, $\widehat{\lambda}\colon \mathcal{M} \longrightarrow \mathrm{P}^{\mathrm{op}}\mathrm{P}$ is natural inclusion, hence monad morphism.
- Bad Example: for the sub-distribution monad $\mathcal{D}_\omega$ there appears to be no interesting EM-algebra $\mathcal{D}^\omega 2 \longrightarrow 2$ (and: difficult to imagine what an axiom for sequential composition would look like)

Our conclusion

Need to move to many-valued logics when discussing probabilistic systems (similarly for weighted).

Strong Completeness Result

If base logic $\mathcal{L}$ satisfies conditions for quasi-canonical T-model, then

- $\mathcal{L}(\theta)$ is sound and strongly complete wrt θ-standard T^{A}-models (standard methods from coalgebraic modal logic, quasi-canonical model theorem)
- $\mathcal{L}(\theta, ;)$ is sound and strongly complete wrt $\theta, ;$-standard T^{A}-models (use quasi-canonical model for $\mathcal{L}(\theta)$ to generate $\theta, ;$-standard model, show quasi-canonical)

Key property of the canonical model

For all MCSs Γ and all formulas φ we have

$$\gamma(\Gamma) \in \heartsuit(\hat{\varphi}) \quad \text{iff} \quad \heartsuit\varphi \in \Gamma$$

where $\hat{\varphi} = \{\Delta \in \mathrm{MCS} \mid \varphi \in \Delta\}$.

Adding Tests

Informally: given formula φ, program $\varphi?$ tests whether φ holds. If the test fails, the program aborts, otherwise do nothing.

- Syntax: $\varphi?$ is a program, when φ is a formula. Formulas and programs defined by mutual induction.
- Semantics: need T to be "pointed": for each set X, TX contains a distinguished element $\bot_{TX}$ ("abort"), and for all $f\colon X \to Y$, $Tf(\bot_{TX}) = \bot_{TY}$.
- Extend dynamic coalgebraic semantics $\xi\colon X \to (TX)^A$,

$$\widehat{\xi}(\varphi?)(x) = \begin{cases} \eta_X(x) & \text{if } x \in [\![\varphi]\!]^{\mathfrak{M}} \\ \bot_{TX} & \text{otherwise} \end{cases}$$

(standard wrt tests, $\widehat{\xi}$ and $[\![\varphi]\!]$ def'd by mutual induction.)

Axiomatising Tests

In PDL: $[\varphi?]p \leftrightarrow (\varphi \to p)$ or $\langle\varphi?\rangle p \leftrightarrow (\varphi \wedge p)$
In GL: $\langle\varphi?\rangle p \leftrightarrow (\varphi \wedge p)$

- Predicate lifting $\heartsuit\colon P \longrightarrow P \circ T$ is
  - box-like if for all X and $U \subseteq X$, $\bot_{TX} \in \heartsuit_X(U)$.
  - diamond-like if for all X and $U \subseteq X$, $\bot_{TX} \notin \heartsuit_X(U)$.

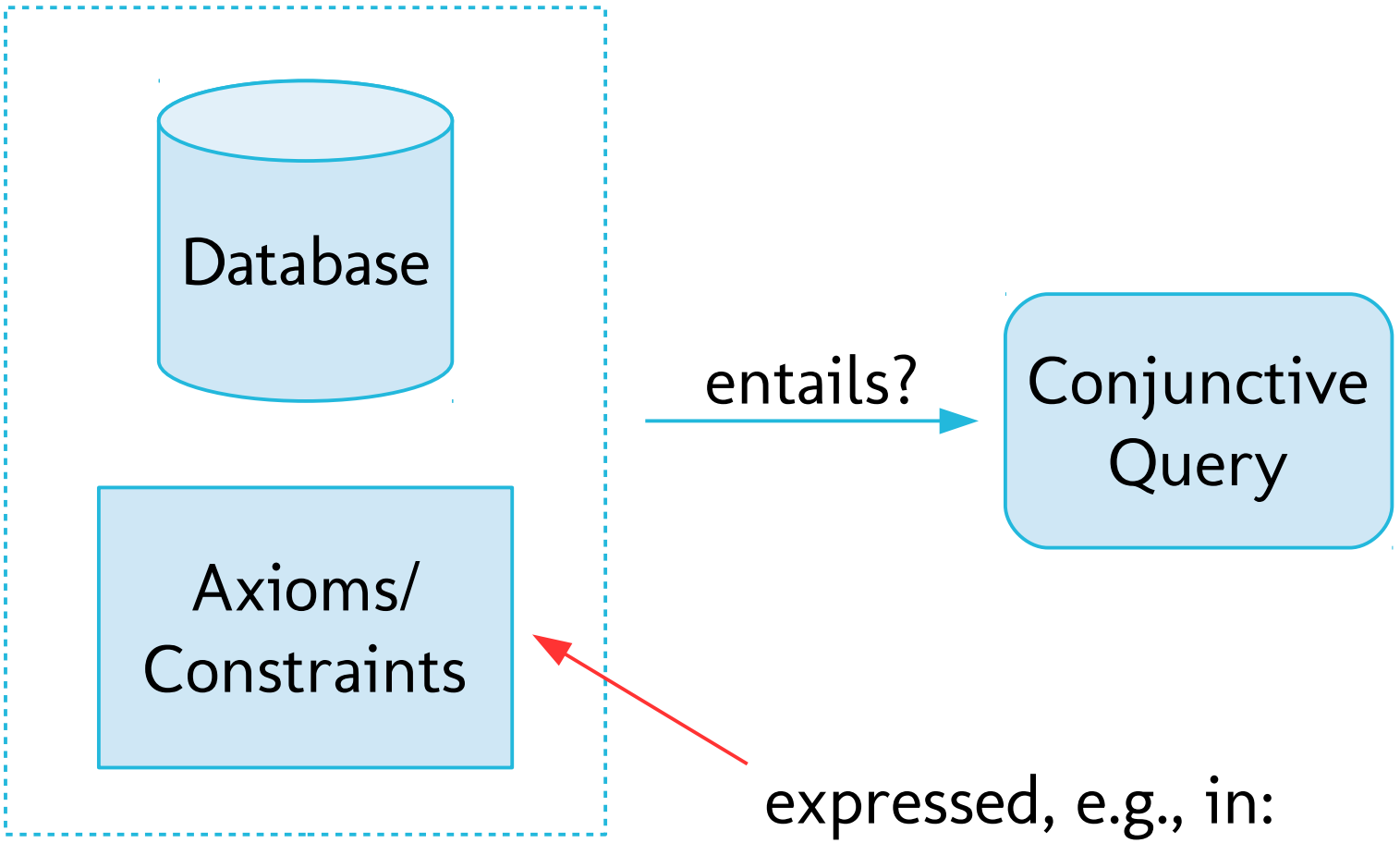
  Lemma: Any $\heartsuit\colon P \longrightarrow P \circ T$ either box-like or diamond-like.
- Axioms:
  – If $\heartsuit$ in dynamic semantics is box-like,
  then add $[\varphi?]p \leftrightarrow (\varphi \to p)$ to Fr,
  – If $\heartsuit$ in dynamic semantics is diamond-like,
  then add $[\varphi?]p \leftrightarrow (\varphi \wedge p)$ to Fr.
- Theorem: $\mathcal{L}(\theta, ;, ?)$ is strongly complete wrt dynamic models.
  (modify quasi-canonical model, extend to standard model, show quasi-canonical)

PDL Conclusion

- possible criticism: no new results; PDL without iteration not interesting
- one seemingly new result for the lift monad $1 + \mathrm{X}$
- adding *-operator is (important) work in progress; uses coalgebraic weak completeness proof & a strengthened coherence condition for quasi-canonical models

Part II: Datalog$^{\pm}$
(joint work with Gottlob, Hernich, Lukasiewicz)

Ontology-Based Data Access



Intuition: ontology unifies and completes the data

Consider a hotel database (collection of atoms)

$$\mathrm{D} = \{\mathrm{Hotel(a)}, \mathrm{4Star(a)}, \mathrm{4Star(b)}\}$$

the rules

$$\begin{aligned} \mathrm{Hotel}, \mathrm{4Star} &\sqsubseteq \exists \mathrm{Pool} \\ \mathrm{4Star} &\sqsubseteq \mathrm{Hotel}, \end{aligned}$$

and the query

$$\mathrm{Q(x)} \leftarrow \exists \mathrm{y}\, \mathrm{Hotel(x)} \wedge \mathrm{Pool(x, y)}.$$

The certain answers (choice of semantics) for the query will be

$$\begin{array}{ll} \emptyset & \text{without ontology} \\ \{\mathrm{a}, \mathrm{b}\} & \text{with ontology} \end{array}$$

Another ontology language: Datalog$^{\pm}$

[Cali, Gottlob, Lukasiewicz] A general Datalog-based framework for tractable query answering over ontologies.
Journal of Web Semantics (2012)

Motivation for Datalog$^{\pm}$

- relations of arbitrary arity
- ontology languages for data access need to be lightweight: lightweight DLs exist, but definitions are involved
- integration of database typical reasoning such as "negation-as-failure-to-prove"

  (if there is no flight connection between Edinburgh and Amsterdam in the database, then we conclude $\neg\text{Connection}(\text{EDI}, \text{AMS})$ - this does not mean that it follows from the facts in the DB using logical deduction)

Datalog$^{\pm}$ Programs

$$\text{Author}(x) \longrightarrow \exists y, z(\text{Article}(x, y) \wedge \text{publishedAt}(y, z))$$

$$\text{publishedAt}(x, y) \wedge \text{publishedAt}(x, z) \longrightarrow y = z$$

$$\text{publishedAt}(x, y) \wedge \text{Conference}(y) \wedge \text{Journal}(y) \longrightarrow \bot$$

Using DL-Notation:

$$\text{Author} \sqsubseteq \exists \text{Article}\ \exists \text{publishedAt}$$
$$\text{funct}\ \ \text{publishedAt}$$
$$\exists \text{publishedAt}^{-} \sqcap \text{Conference} \sqsubseteq \neg \text{Journal}$$

Datalog$^{\pm}$ Programs: General Shape

A program is a finite set of Datalog$^{\pm}$ rules:

$$R_1(\overline{x}_1) \wedge \cdots \wedge R_k(\overline{x}_k) \longrightarrow \psi$$

where

- $R_i(\overline{x}_i)$ are atoms,
- ψ is of one of the following forms:
  - $\psi \equiv \exists \overline{z}\ (S_1(\overline{y}_1) \wedge \ldots \cdots \wedge S_n(\overline{y}_n))$, where the $\overline{y}_i$'s contain only variables in $\overline{z}$ or in the rule body, or
  - $\psi \equiv y_1 = y_2$, where y_1 and y_2 occur in the rule body, or
  - $\psi \equiv \bot$

Simplification: in the talk we will only consider Boolean queries.

Semantics: two equivalent definitions

For a given database D and Datalog$^{\pm}$-rules Σ:

Semantics I: Certain answers A query holds if it holds in all possible models of $D \cup \Sigma$

Semantics II: Canonical model A query holds if it holds in the minimal model of $D \cup \Sigma_f$ where Σ_f is the skolemisation of Σ, e.g., a rule

$$R_1(x_1, \ldots, x_k) \longrightarrow \exists y.S(\overline{x}, y)$$

is replaced by

$$R_1(x_1, \ldots, x_k) \longrightarrow S(\overline{x}, g(x_1, \ldots, x_k))$$

where g is a new function symbol.

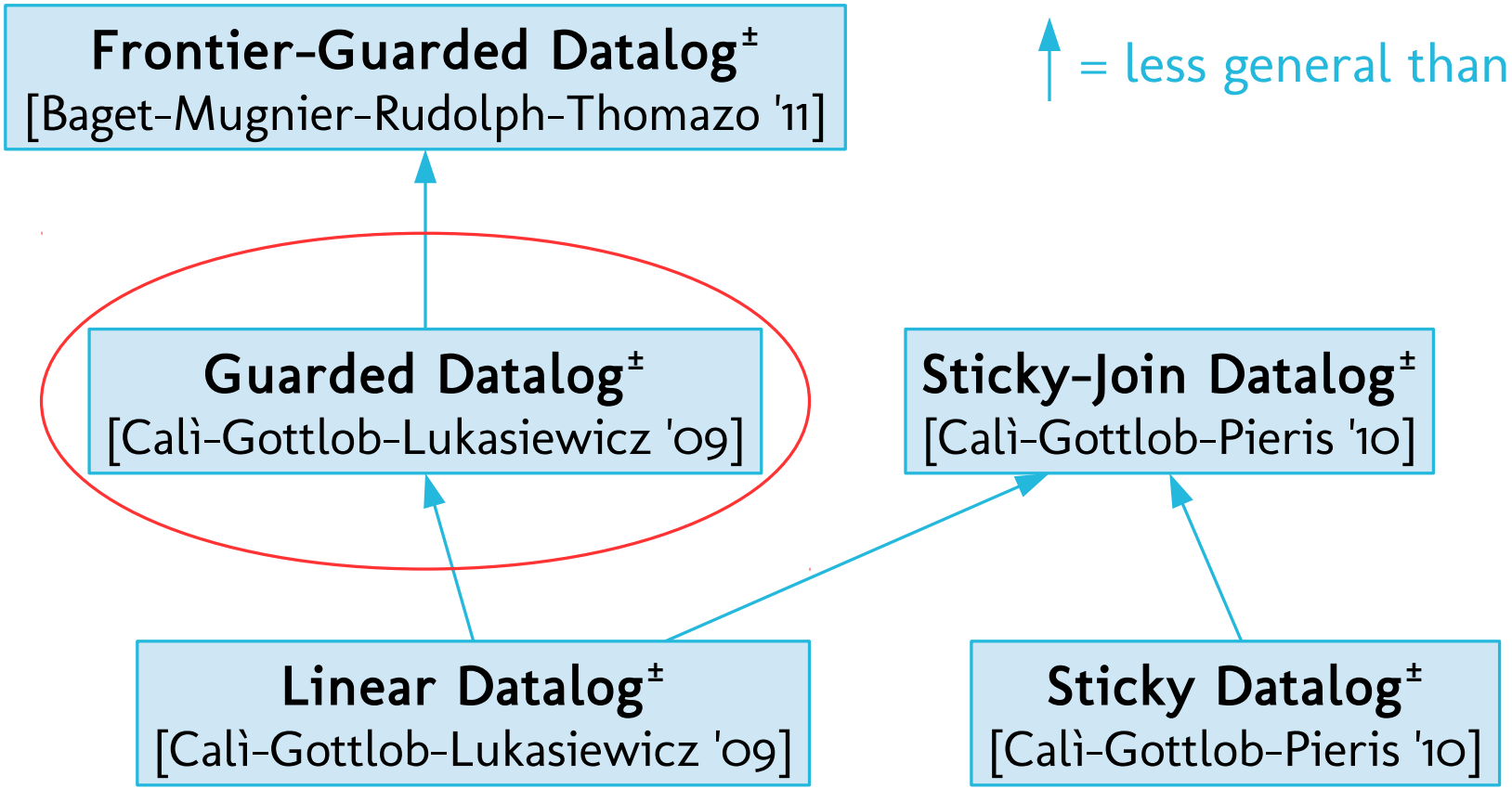
Logic Programming

- Skolemisation turns a Datalog$^{\pm}$ program Σ into a logic program!
- Query answering relative to a Datalog$^{\pm}$ program can be done using logic programming techniques.
- Nevertheless is Datalog$^{\pm}$ interesting on its own: programs have particular syntactic shapes, need to restrict to "tractable" fragments
- "Tractable" here means polynomial in the data complexity.

Data complexity (Vardi 1982)

- complexity of answering query Q relative to a database D and a program Σ is measured in data complexity
- this means: Q and Σ are fixed - size of the input is the size of D
- Idea: size of D the dominating factor

Some Tractable Cases (Incomplete)



Adding negated atoms

The minimal model of a logic program is obtained as the least fixpoint of a monotone operator

$$\mathrm{T_P} : \mathcal{P}(\mathrm{At}) \longrightarrow \mathcal{P}(\mathrm{At})$$

such that M is the smallest set of atoms that is closed under application of a (substituted) rule.

Simple Example (propositional program) with negation

$$\begin{array}{rcl} \neg \mathrm{q}, \mathrm{p} & \longrightarrow & \mathrm{q} \\ & \longrightarrow & \mathrm{p} \end{array}$$

$\mathrm{T_P}(\emptyset) = \{\mathrm{p}\}, \mathrm{T_P^2}(\emptyset) = \{\mathrm{p}, \mathrm{q}\}$

$\mathrm{T_P}(\emptyset) = \{\mathrm{p}\}, \mathrm{T_P^2}(\emptyset) = \{\mathrm{p}, \mathrm{q}\}, \mathrm{T_P^3}(\emptyset) \stackrel{?}{=} \{\mathrm{p}\} \Rightarrow \mathrm{T_P}$ not monotone!

Solutions

The addition of nonmonotonic negation to logic programs is well researched, we focused on two options:

- well-founded semantics: canonical model does exist, but monotone operator more complicated and model is three-valued (`F`,`T`,`U`)

- stable semantics: two valued models, but no canonical model - in particular, models cannot be obtained as unique least fixpoint of a monotone operator

Problem: No previously existing complexity (or even decidability) results for logic programs involving function symbols.

Well-Founded Semantics: Definition

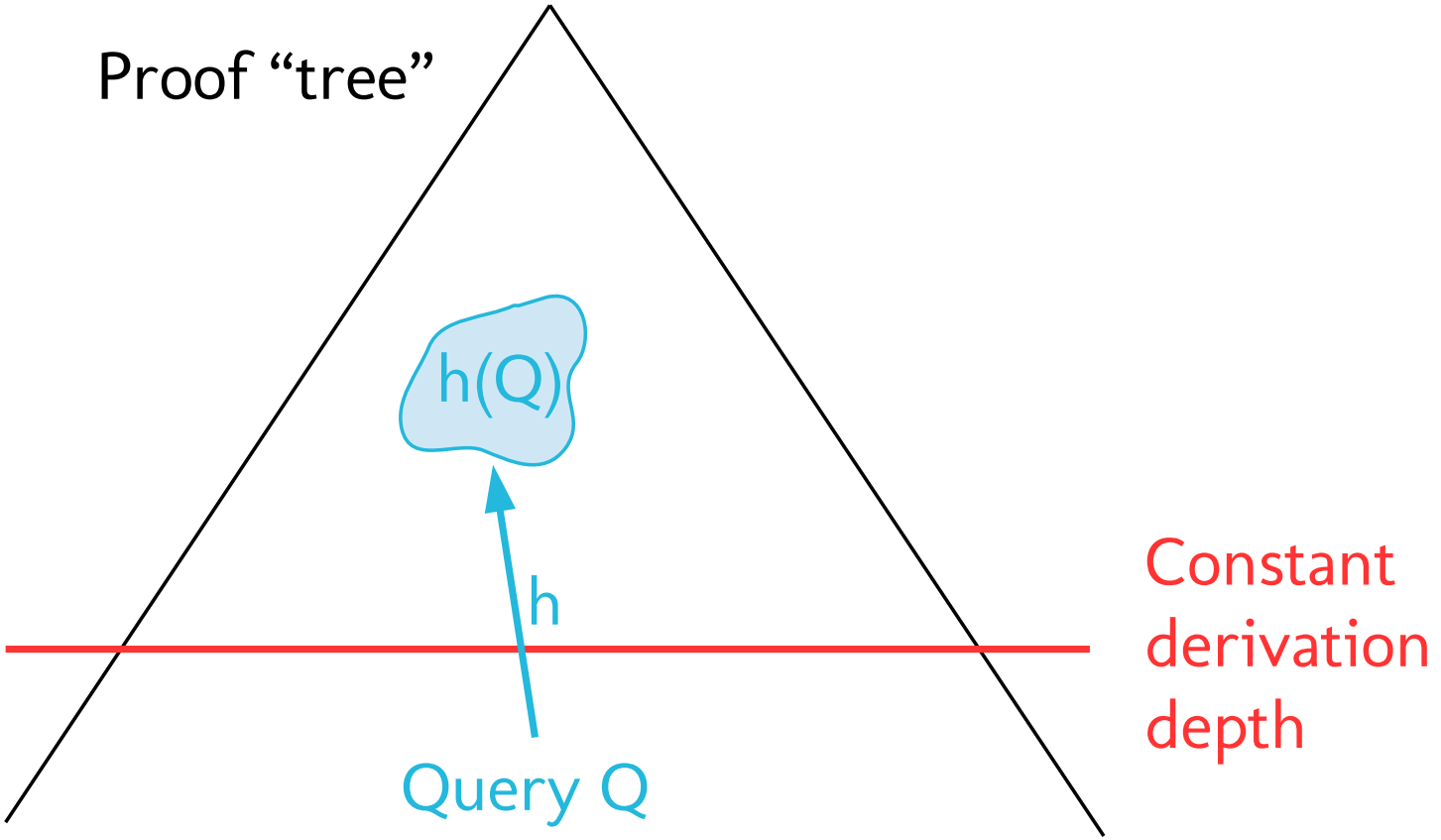
van Gelder-Ross-Schlipf '91

$$\text{Number}(0), \text{Even}(0)$$
$$\text{Number}(x) \longrightarrow \text{Number}(s(x))$$
$$\text{Number}(x) \wedge \neg\text{Even}(x) \longrightarrow \text{Even}(s(x))$$

$\text{Number}(0), \text{Even}(0)$
$\text{Number}(s(0)), \neg\text{Even}(s(0))$
$\text{Number}(s^2(0)), \text{Even}(s^2(0))$

- Start with empty set of literals.
- In each step
  - Apply the rules to infer new atoms.
  - Add negations of atoms that can no longer be derived.
- This converges to the well-founded model!

Proof in the positive case



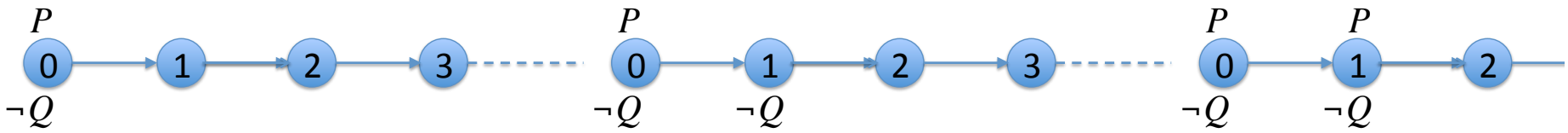
This fails in the negative case

Deciding whether a literal belongs to $\mathrm{WFS}(\mathrm{D}, \Sigma)$ may require an infinite number of iterations:

$$\mathrm{R}(0,1), \mathrm{P}(0)$$
$$\mathrm{R}(x,y) \longrightarrow \mathrm{R}(y, \mathrm{f}(x,y))$$

$$\mathrm{R}(x,y) \wedge \neg \mathrm{P}(x) \longrightarrow \mathrm{Q}(y)$$

$$\mathrm{R}(x,y) \wedge \mathrm{P}(x) \wedge \neg \mathrm{Q}(y) \longrightarrow \mathrm{P}(y)$$
$$\mathrm{R}(x,y) \wedge \neg \mathrm{P}(y) \longrightarrow \mathrm{S}(0)$$



Forward Proofs

Schlipf '95

- Forward proof of an atom $\mathrm{R}(\bar{\mathrm{a}})$ from a program P:

$$\alpha_1 \xrightarrow{\mathrm{r_1}} \alpha_2 \xrightarrow{\mathrm{r_2}} \alpha_3 \xrightarrow{\mathrm{r_3}} - - - \xrightarrow{\mathrm{r_n}} \mathrm{R}(\bar{\mathrm{a}})$$

  i.e., a series of rule applications ignoring negative side atoms.

- $\neg\mathrm{R}(\bar{\mathrm{a}})$ will be derived if every forward proof for it "uses" a negative literal $\neg\mathrm{S}(\bar{\mathrm{b}})$, with $\mathrm{S}(\bar{\mathrm{b}})$ already known to be true.

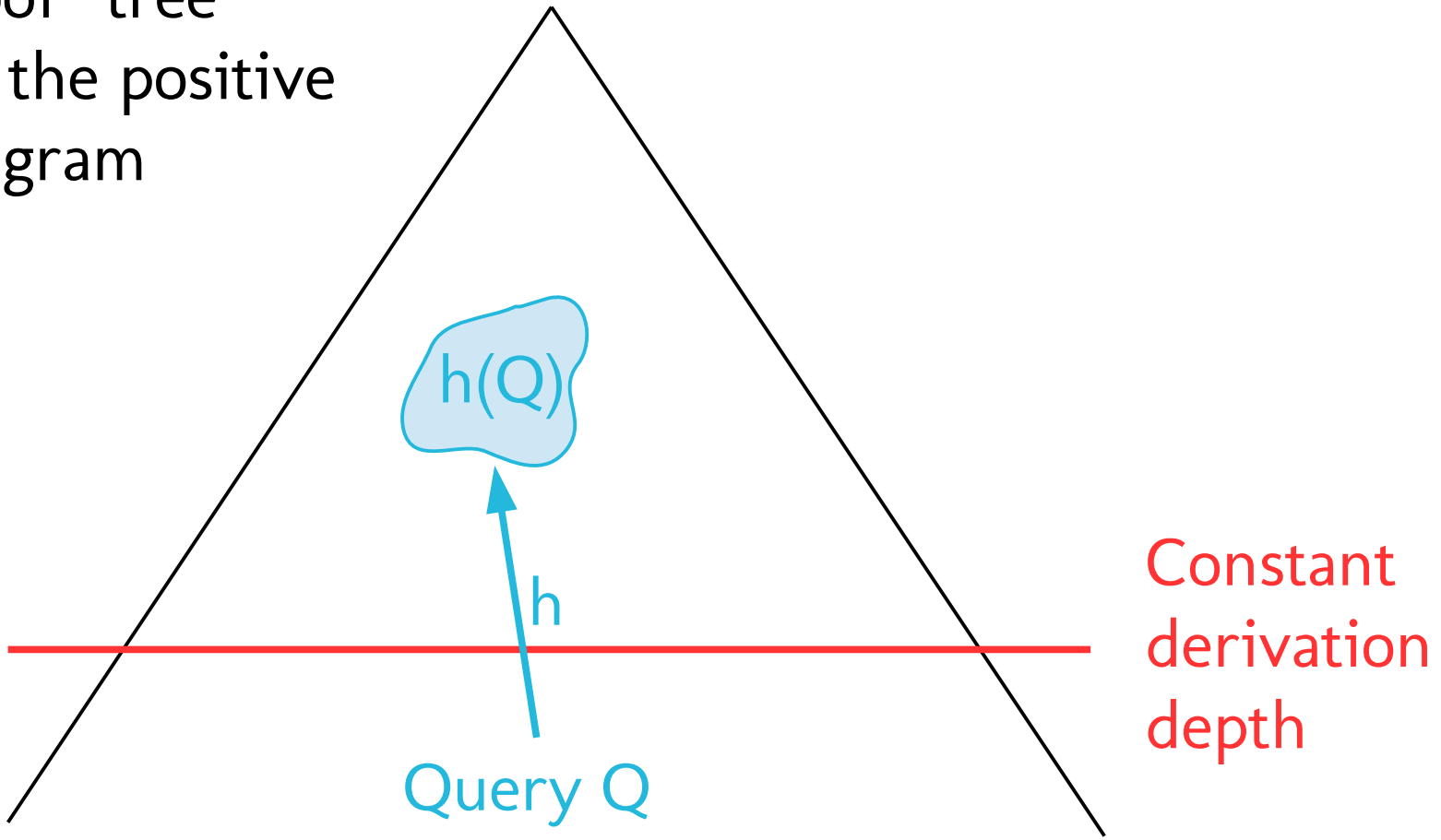
- $\mathrm{R}(\bar{\mathrm{a}})$ will be derived if there exists a forward proof such that all side literals are already known to be true.

Query answering

- alternating algorithm that either tries to find a forward proof of a given atom or to show that no such proof for a given negative literal exists
- configurations of the algorithm roughly correspond to atoms and subsets of their type (in WFS(P))
- key observation: we can identify configurations that are "X-isomorphic" (where X is the set of relevant constants)

Back to the positive case

Proof “tree”
for the positive
program



Complexity results

Input	A database D, a guarded normal Datalog$^{\pm}$ program Σ, and a Boolean conjunctive query Q with negation
Question	Is Q true in WFS(D, Σ)?

- PTIME-complete in data complexity
- EXPTIME-complete if predicate's arities are bounded by a constant
- 2-EXPTIME-complete in general

A hidden assumption

- the translation into logic programming implies that we treat all elements of our models as distinct
- Example:

$$\text{Employee}(x) \longrightarrow \exists y\, \text{hasEmployer}(x, y)$$

  together with $D = \{\text{Employee(John)}, \text{Employee(Sam)}\}$.

- Answer of the query

$$\exists x(\text{hasEmployer}(\text{John}, x) \wedge \neg\, \text{hasEmployer}(\text{Sam}, x))$$

  depends on whether we generate for John and Sam distinct employers by applying the rule
- $\Rightarrow$ Equality-Friendly Well-founded Semantics

Guarded Fixed Point Logic

The set of formulas of GFP over $\mathcal{R}$ is built from atomic formulas over $\mathcal{R}$ (including equality atoms) using Boolean combinations, and the following two additional formula formation rules:

I. If α is an atomic formula over $\mathcal{R}$ containing the variables in x, and ψ is a GFP formula over $\mathcal{R}$ whose free variables occur in α, then $\exists \overline{\mathrm{x}}\,(\alpha \wedge \psi)$ and $\forall \overline{\mathrm{x}}\,(\alpha \rightarrow \psi)$ are GFP formulas over $\mathcal{R}$. The formula α is called guard.

II. Let R be a k-ary predicate, $\overline{\mathrm{x}}$ a k-tuple of variables, and $\psi(\mathrm{R}, \overline{\mathrm{x}})$ a GFP formula over $\mathcal{R} \cup \{\mathrm{R}\}$ whose free variables occur in $\overline{\mathrm{x}}$, and where R appears only positively (in the scope of an even number of negation symbols) and not in guards. Then, $[\mathrm{lfp}_{\mathrm{R},\overline{\mathrm{x}}}\,\psi](\overline{\mathrm{x}})$ and $[\mathrm{gfp}_{\mathrm{R},\overline{\mathrm{x}}}\,\psi](\overline{\mathrm{x}})$ are GFP formulas over $\mathcal{R}$ with free variables $\overline{\mathrm{x}}$.

Example Formula GFP

The following GFP formula says that binary relation E is well-founded, i.e., no element is the endpoint of an infinite E-path:

$$\forall x, y \left(E(x, y) \rightarrow [\mathrm{lfp}_{W,x}\, \forall y \big(E(y, x) \rightarrow W(y) \big)](x) \right) .$$

[Grädel & Walukiewicz] 2-ExpTime decidability
(ExpTime with bounded arities)

Translation of WFS into GFP (Idea)

Construct a GFP sentence efwfs(P) whose models closely correspond to the databases in EFWFS(P), i.e., such that for all queries ("covered NBCQs") Q over the schema of P, we have EFWFS(P) $\models$ Q iff efwfs(P) $\models$ Q*.

- The key is to "existentially quantify" all the instances of NTGDs that we use to compute the WFS, and to mimic the fixed-point definition of the WFS using those instances.
- Fixpoint in WFS is modeled with lfp (derivable atoms) and gfp (those atoms that certainly cannot be derived).
- Upper bound on set of derived positive atoms and coveredness for derived negative atoms provides guards.

Stable semantics

- Both approaches also work with the stable semantics
- Data Complexity increases to coNP
- Intuition: Need to check query on all stable models

Ref's

- [Gottlob, Hernich, K., and Lukasiewicz] Equality-friendly well-founded semantics and applications to description logics. AAAI 2012

- [Hernich, K., Lukasiewicz and Gottlob] Well-founded semantics for extended datalog and ontological reasoning. PODS 2013

- [Gottlob, Hernich, K., and Lukasiewicz] Stable model semantics for guarded existential rules and description logics. KR2014

Part III: The connections (Future Work!)

Datalog$^{\pm}$

Issues

- query-rewriting using backward-chaining: very useful - not sufficiently explored
- need for reasoning with probabilities, weight, preferences and combinations
- need to operate over semi-structured data

Goals

- Use backward-chaining algorithm from coalgebraic LP to obtain "parallellizable" query-rewriting algorithm
- Extend this to Datalog$^{\pm}$ with nonmonotonic negation
- Extend Datalog$^{\pm}$ to Coalgebraic Datalog$^{\pm}$ for other types of data.

Coalgebraic Datalog$^{\pm}$

- Goals:
  - extend Datalog$^{\pm}$ with features such as probabilities, weights and preferences
  - provide efficient algorithms for query-rewriting and query answering
- Two Approaches:
  - generalise coalgebraic LP to other functors
  - add fixpoint operators to coalgebraic predicate logic to create coalgebraic LFP or GFP
- [Komendantskaya, Schmidt, and Power] Coalgebraic logic programming: from semantics to implementation. Journal of Logic and Computation (2014)
- [Litak, Pattinson, Sano, and Schröder] Coalgebraic predicate logic. ICALP (2012)

Coalgebraic semi-structured data

- represent tree and graph-structured data coalgebraically
- develop theory of data-labelled coalgebras, similar to recent work on XML trees

  [Figueira, Figueira, and Areces] Basic Model Theory of XPath on Data Trees. ICDT 2014.
- develop theory of automata operating on data-labelled structures

Coalgebraic (core) XPath

- our starting point is core XPath for data graphs:

> The path formulae of the two flavors of GXPath are given below. In both cases a ranges over Σ.
>
> Path expressions of *Regular graph XPath*, denoted by $\mathsf{GXPath}_{\mathrm{reg}}$, are given by:
>
> $$\alpha, \beta := \varepsilon \mid \_ \mid a \mid a^- \mid [\varphi] \mid \alpha \cdot \beta \mid \alpha \cup \beta \mid \overline{\alpha} \mid \alpha^*$$
>
> Path expressions of *Core graph XPath* denoted by $\mathsf{GXPath}_{\mathrm{core}}$ are given by:
>
> $$\alpha, \beta := \varepsilon \mid \_ \mid a \mid a^- \mid a^* \mid a^{-*} \mid [\varphi] \mid \alpha \cdot \beta \mid \alpha \cup \beta \mid \overline{\alpha}$$

- build coalgebraic core XPath starting from coalgebraic PDL:
  - add *
  - add non-natural operations
  - extend path-expressions to properties of the data, e.g. $\alpha^=$, $\alpha^{\neq}$ or regular expressions with memory
  - probabilistic or weighted graphs

On the connection (G)XPath & PDL

- [Libkin, Martens, and Vrgoc] Querying graph databases with XPath. ICDT (2013)
- [Alechina, Immermann] Reachability Logic: An Efficient Fragment of Transitive Closure Logic. Logic Journal of the IGPL (2000)
- [ten Cate, Marx] Navigational XPath: calculus and algebra. ACM SIGMOD Record (2007)
- [ten Cate, Fontaine, Litak] Some modal aspects of XPath. Journal of Applied Non-Classical Logics (2010)

Further steps

- Ontological query answering for path queries.

- [Cardelli, Ghelli] TQL: a query language for semistructured data based on the ambient logic. Mathematical Structures in Computer Science (2004)

- long-term: "continuous" queries over streaming data?

Thanks!