Finite Model Property for LFD

Raoul Koudijs

ILLC Amsterdam

1 December 2021

The Logic of Functional Dependence (LFD)

The **logic of functional dependence (LFD)** is a highly expressive logic with many nice meta-properties, most notably it is decidable. However, decidability was only shown via a detour through quasi-models.

The Logic of Functional Dependence (LFD)

The **logic of functional dependence (LFD)** is a highly expressive logic with many nice meta-properties, most notably it is decidable. However, decidability was only shown via a detour through quasi-models.

It is interpreted over **generalised assignment models** just like the logic of Cylindric Relativized Set Algebras (CRS). Such models have natural modal companions that can serve as equivalent relational semantics.

The Logic of Functional Dependence (LFD)

The **logic of functional dependence (LFD)** is a highly expressive logic with many nice meta-properties, most notably it is decidable. However, decidability was only shown via a detour through quasi-models.

It is interpreted over **generalised assignment models** just like the logic of Cylindric Relativized Set Algebras (CRS). Such models have natural modal companions that can serve as equivalent relational semantics.

On the dependence side, LFD differs radically from dependence logics by studying the notion of **local dependence** rather than global dependence.

The Logic of Functional Dependence (LFD)

The **logic of functional dependence (LFD)** is a highly expressive logic with many nice meta-properties, most notably it is decidable. However, decidability was only shown via a detour through quasi-models.

It is interpreted over **generalised assignment models** just like the logic of Cylindric Relativized Set Algebras (CRS). Such models have natural modal companions that can serve as equivalent relational semantics.

On the dependence side, LFD differs radically from dependence logics by studying the notion of **local dependence** rather than global dependence.

My contributions were (i) proposing a notion of **dependence bisimulation** characterizing LFD and (ii) prove the **finite model property**.

Origins

FOL as a Modal Logic

Recall the Tarskian definition if truth of a first-order formula in a model *at some variable assignment*:

$$M, \alpha \models \exists x \phi \text{ iff } \exists d \in dom(M) \text{ s.t. } M, \alpha[x := d] \models \phi$$

FOL as a Modal Logic

Recall the Tarskian definition if truth of a first-order formula in a model *at some variable assignment*:

$$M, \alpha \models \exists x \phi \text{ iff } \exists d \in dom(M) \text{ s.t. } M, \alpha[x := d] \models \phi$$

Equivalently, this can be expressed as

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ with } \alpha =^{x} \beta \text{ and } M, \beta \models \phi$$

FOL as a Modal Logic

Recall the Tarskian definition if truth of a first-order formula in a model *at some variable assignment*:

$$M, \alpha \models \exists x \phi \text{ iff } \exists d \in dom(M) \text{ s.t. } M, \alpha[x := d] \models \phi$$

Equivalently, this can be expressed as

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ with } \alpha =^x \beta \text{ and } M, \beta \models \phi$$

where $=^x$ is the relation of agreeing 'up to x-values' (for $x \in V$)

$$s =^x t \qquad \text{iff} \qquad s_{\restriction V-\{x\}} = t_{\restriction V-\{x\}}$$

for which **the quantifier $\exists x$ becomes the $\Diamond$-modality**!

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

In this way, we may think of **quantifiers as modalities** and vice versa!

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

In this way, we may think of **quantifiers as modalities** and vice versa!

Standard first-order models arise by making 3 'negotiable choices':

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

In this way, we may think of **quantifiers as modalities** and vice versa!

Standard first-order models arise by making 3 'negotiable choices':
(i) States are identified with variable assignments

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

In this way, we may think of **quantifiers as modalities** and vice versa!

Standard first-order models arise by making 3 'negotiable choices':
(i) States are identified with variable assignments
(ii) R_x is the specific relation $=^x$ of agreeing 'up to x-values'

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

In this way, we may think of **quantifiers as modalities** and vice versa!

Standard first-order models arise by making 3 'negotiable choices':
(i) States are identified with variable assignments
(ii) R_x is the specific relation $=^x$ of agreeing 'up to x-values'
(iii) All assignments in the function space $dom(M)^V$ are available

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

In this way, we may think of **quantifiers as modalities** and vice versa!

Standard first-order models arise by making 3 'negotiable choices':
(i) States are identified with variable assignments
(ii) R_x is the specific relation $=^x$ of agreeing 'up to x-values'
(iii) All assignments in the function space $dom(M)^V$ are available

Without these additional choices, we obtain **abstract modal models** $\mathbb{M} = (A, (R_x)_{x \in V}, I)$ where $I(P\mathbf{x}) \subseteq A$ for each atomic formula $P\mathbf{x}$

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

In this way, we may think of **quantifiers as modalities** and vice versa!

Standard first-order models arise by making 3 'negotiable choices':
(i) States are identified with variable assignments
(ii) R_x is the specific relation $=^x$ of agreeing 'up to x-values'
(iii) All assignments in the function space $dom(M)^V$ are available

Without these additional choices, we obtain **abstract modal models** $\mathbb{M} = (A, (R_x)_{x \in V}, I)$ where $I(P\mathbf{x}) \subseteq A$ for each atomic formula $P\mathbf{x}$

Condition (i) in fact poses no restriction at all $s^*(x) := (s, x)$ (v.B.,1996)

Abstract Modal Frames

This gives rise to the following abstract modal pattern for quantification:

$$M, \alpha \models \exists x \phi \text{ iff } \exists \beta \text{ s.t. } R_x \alpha \beta \text{ and } M, \beta \models \phi$$

In this way, we may think of **quantifiers as modalities** and vice versa!

Standard first-order models arise by making 3 'negotiable choices':
(i) States are identified with variable assignments
(ii) R_x is the specific relation $=^x$ of agreeing 'up to x-values'
(iii) All assignments in the function space $dom(M)^V$ are available

Without these additional choices, we obtain **abstract modal models** $\mathbb{M} = (A, (R_x)_{x \in V}, I)$ where $I(P\mathbf{x}) \subseteq A$ for each atomic formula $P\mathbf{x}$

Condition (i) in fact poses no restriction at all $s^*(x) := (s, x)$ (v.B.,1996)
CRS is the logic obtained by giving up (iii) while keeping (i),(ii).

The Logic of Cylindric Relativized Set Algebras (CRS)

Dependence models are obtained from standard first-order models by dropping the existential condition (iii) encoding a full Cartesian product

The Logic of Cylindric Relativized Set Algebras (CRS)

Dependence models are obtained from standard first-order models by dropping the existential condition (iii) encoding a full Cartesian product

Definition (Dependence Model)

A dependence model $\mathbb{M} = (M, A)$ is a pair of a relational structure M and a team $A \subseteq dom(M)^V$ (set of 'available' variable assignments).

The Logic of Cylindric Relativized Set Algebras (CRS)

Dependence models are obtained from standard first-order models by dropping the existential condition (iii) encoding a full Cartesian product

Definition (Dependence Model)

A dependence model $\mathbb{M} = (M, A)$ is a pair of a relational structure M and a team $A \subseteq dom(M)^V$ (set of 'available' variable assignments).

CRS is just a first-order (relational) language (with no substitutions) where instead of ordinary quantification $\exists x$ we have a polyadic quantifier

$$(M, A), s \models \exists X \phi \text{ iff } \exists t \in A \text{ with } s =^X t \text{ and } (M, A), t \models \phi$$

The Logic of Cylindric Relativized Set Algebras (CRS)

Dependence models are obtained from standard first-order models by dropping the existential condition (iii) encoding a full Cartesian product

Definition (Dependence Model)

A dependence model $\mathbb{M} = (M, A)$ is a pair of a relational structure M and a team $A \subseteq dom(M)^V$ (set of 'available' variable assignments).

CRS is just a first-order (relational) language (with no substitutions) where instead of ordinary quantification $\exists x$ we have a polyadic quantifier

$$(M, A), s \models \exists X \phi \text{ iff } \exists t \in A \text{ with } s =^X t \text{ and } (M, A), t \models \phi$$

where $=^X$ is the relation of 'agreeing up to X' (X a *set* of variables)

The Logic of Cylindric Relativized Set Algebras (CRS)

Dependence models are obtained from standard first-order models by dropping the existential condition (iii) encoding a full Cartesian product

Definition (Dependence Model)

A dependence model $\mathbb{M} = (M, A)$ is a pair of a relational structure M and a team $A \subseteq dom(M)^V$ (set of 'available' variable assignments).

CRS is just a first-order (relational) language (with no substitutions) where instead of ordinary quantification $\exists x$ we have a polyadic quantifier

$$(M, A), s \models \exists X \phi \text{ iff } \exists t \in A \text{ with } s =^X t \text{ and } (M, A), t \models \phi$$

where $=^X$ is the relation of 'agreeing up to X' (X a *set* of variables)

$$s =^X t \qquad \text{iff} \qquad s_{\restriction V - \{x\}} = t_{\restriction V - \{x\}}$$

The Logic of Cylindric Relativized Set Algebras (CRS)

Dependence models are obtained from standard first-order models by dropping the existential condition (iii) encoding a full Cartesian product

Definition (Dependence Model)

A dependence model $\mathbb{M} = (M, A)$ is a pair of a relational structure M and a team $A \subseteq dom(M)^V$ (set of 'available' variable assignments).

CRS is just a first-order (relational) language (with no substitutions) where instead of ordinary quantification $\exists x$ we have a polyadic quantifier

$$(M, A), s \models \exists X \phi \text{ iff } \exists t \in A \text{ with } s =^X t \text{ and } (M, A), t \models \phi$$

where $=^X$ is the relation of 'agreeing up to X' (X a *set* of variables)

$$s =^X t \qquad \text{iff} \qquad s_{\restriction V - \{x\}} = t_{\restriction V - \{x\}}$$

The CRS-quantifier $\forall\{x\}$ collapses to $\forall x$ over *full* models (i.e. dependence models $(M, dom(M)^V)$ encoding a standard model)

CRS and Locality

A disadvantage of this specific way of generalising first-order quantification is that CRS fails to satisfy a property called **Locality**, in contrast to FOL

Locality Whether $M, s \models \phi$ depends only on $\mathit{free}(\phi)$

CRS and Locality

A disadvantage of this specific way of generalising first-order quantification is that CRS fails to satisfy a property called **Locality**, in contrast to FOL

$$\textbf{Locality} \qquad \text{Whether } M, s \models \phi \text{ depends only on } \mathit{free}(\phi)$$

This lead to the following alternative proposal for the CRS-quantifiers (Marx, Venema), designed to satisfy Locality

$$(M, A), s \models \widetilde{\exists}\mathbf{x}\phi \text{ iff } \exists t \in A \text{ with } s =_{\mathit{Free}(\phi)-X} t \text{ and } (M, A), t \models \phi$$

which still collapses to standard quantification over full models.

CRS and Locality

A disadvantage of this specific way of generalising first-order quantification is that CRS fails to satisfy a property called **Locality**, in contrast to FOL

$$\textbf{Locality} \qquad \text{Whether } M, s \models \phi \text{ depends only on } \mathit{free}(\phi)$$

This lead to the following alternative proposal for the CRS-quantifiers (Marx, Venema), designed to satisfy Locality

$$(M, A), s \models \widetilde{\exists}\mathbf{x}\phi \text{ iff } \exists t \in A \text{ with } s =_{\mathit{Free}(\phi)-X} t \text{ and } (M, A), t \models \phi$$

which still collapses to standard quantification over full models.

However, there are conceptual problems arising from this move; e.g. $\exists X\phi \rightsquigarrow \exists X(\phi \wedge z = z)$ conjoining a tautology changes the meaning of the formula (witnessing assignment has to agree on $z \notin X$ with s as well)

Dependence via Assignment Gaps

Both proposals have problems generalizing all desirable properties of standard quantification to the generalised setting. Moreover, the general setting has to offer things *beyond* FOL that they fail to capture...

Dependence via Assignment Gaps

Both proposals have problems generalizing all desirable properties of standard quantification to the generalised setting. Moreover, the general setting has to offer things *beyond* FOL that they fail to capture...

The 'assignment gaps' create **dependencies between variables**!

Dependence via Assignment Gaps

Both proposals have problems generalizing all desirable properties of standard quantification to the generalised setting. Moreover, the general setting has to offer things *beyond* FOL that they fail to capture...

The 'assignment gaps' create **dependencies between variables**! Contrastingly, in FOL variables are manipulated completely *independent* of each other, witnessed by the FOL-validity

$$(\mathit{Commutation}) \qquad \exists x \exists y \phi \rightarrow \exists y \exists x \phi$$

which fails to be a CRS-validity. But we can only 'see' dependencies implicitly with CRS through such failures.

Dependence via Assignment Gaps

Both proposals have problems generalizing all desirable properties of standard quantification to the generalised setting. Moreover, the general setting has to offer things *beyond* FOL that they fail to capture...

The 'assignment gaps' create **dependencies between variables**! Contrastingly, in FOL variables are manipulated completely *independent* of each other, witnessed by the FOL-validity

$$(\mathit{Commutation}) \qquad \exists x \exists y \phi \rightarrow \exists y \exists x \phi$$

which fails to be a CRS-validity. But we can only 'see' dependencies implicitly with CRS through such failures.

The failure of the above Commutation axiom merely establishes a dependency in the sense of the negation of independence. Here we will look at the far stronger notion of *functional* dependence

Functional Dependence

A variable y functionally depends on a set of variables X **globally** if *whenever* the values of X are fixed, so is the y-value.

Functional Dependence

A variable y functionally depends on a set of variables X **globally** if *whenever* the values of X are fixed, so is the y-value. Dependence logics based on team semantics feature explicit 'dependence atoms'

$$(M, A) \models =(X, y) \quad \text{iff} \quad \forall s \in A \forall t \in A(\bigwedge_{x \in X} s(x) = t(x) \rightarrow s(y) = t(y))$$

Functional Dependence

A variable y functionally depends on a set of variables X **globally** if *whenever* the values of X are fixed, so is the y-value. Dependence logics based on team semantics feature explicit 'dependence atoms'

$$(M, A) \models =(X, y) \quad \text{iff} \quad \forall s \in A \forall t \in A(\bigwedge_{x \in X} s(x) = t(x) \to s(y) = t(y))$$

LFD studies the more fine-grained notion of **local dependence**:

$$(M, A), s \models D_X y \quad \text{iff} \quad \forall t \in A(\bigwedge_{x \in X} s(x) = t(x) \to s(y) = t(y)$$

which says that whenever the values of X are fixed to the *current* values, then also the y-value will be the same as the *current* one.

Functional Dependence

A variable y functionally depends on a set of variables X **globally** if *whenever* the values of X are fixed, so is the y-value. Dependence logics based on team semantics feature explicit 'dependence atoms'

$$(M, A) \models =(X, y) \quad \text{iff} \quad \forall s \in A \forall t \in A(\bigwedge_{x \in X} s(x) = t(x) \to s(y) = t(y))$$

LFD studies the more fine-grained notion of **local dependence**:

$$(M, A), s \models D_X y \quad \text{iff} \quad \forall t \in A(\bigwedge_{x \in X} s(x) = t(x) \to s(y) = t(y)$$

which says that whenever the values of X are fixed to the *current* values, then also the y-value will be the same as the *current* one.

LFD is classical (and decidable), in contrast to dependence logics based on team semantics $\rightsquigarrow$ dependence not intrinsically non-classical

A Typical Amsterdam Database Example

Row	Coffeeshop	Specialty	Price	Location
1	Dampkring	Skywalker	Moderate	Center
2	Paradox	Hindu Kush	Expensive	Center
3	Boerejongens	S5 Haze	Moderate	Center
4	Cheech & Chong's	S5 Haze	Cheap	West
5	Sensimillia	Tbisla	Cheap	West
6	Greenhouse	Hindu Kush	Expensive	East
7	The Stud	Tbisla	Cheap	East

A Typical Amsterdam Database Example

Row	Coffeeshop	Specialty	Price	Location
1	Dampkring	Skywalker	Moderate	Center
2	Paradox	Hindu Kush	Expensive	Center
3	Boerejongens	S5 Haze	Moderate	Center
4	Cheech & Chong's	S5 Haze	Cheap	West
5	Sensimillia	Tbisla	Cheap	West
6	Greenhouse	Hindu Kush	Expensive	East
7	The Stud	Tbisla	Cheap	East

global dependencies:
Coffeeshop $\rightarrow$ {Specialty, Price, Location}

A Typical Amsterdam Database Example

Row	Coffeeshop	Specialty	Price	Location
1	Dampkring	Skywalker	Moderate	Center
2	Paradox	Hindu Kush	Expensive	Center
3	Boerejongens	S5 Haze	Moderate	Center
4	Cheech & Chong's	S5 Haze	Cheap	West
5	Sensimillia	Tbisla	Cheap	West
6	Greenhouse	Hindu Kush	Expensive	East
7	The Stud	Tbisla	Cheap	East

global dependencies:

Coffeeshop $\rightarrow$ {Specialty, Price, Location}

{Specialty, Location} $\rightarrow$ {Price, Coffeeshop}

A Typical Amsterdam Database Example

Row	Coffeeshop	Specialty	Price	Location
1	Dampkring	Skywalker	Moderate	Center
2	Paradox	Hindu Kush	Expensive	Center
3	Boerejongens	S5 Haze	Moderate	Center
4	Cheech & Chong's	S5 Haze	Cheap	West
5	Sensimillia	Tbisla	Cheap	West
6	Greenhouse	Hindu Kush	Expensive	East
7	The Stud	Tbisla	Cheap	East

global dependencies:

Coffeeshop $\rightarrow$ {Specialty, Price, Location}

{Specialty, Location} $\rightarrow$ {Price, Coffeeshop}

local dependencies:

Location $\rightarrow$ Price at rows 4,5

A Typical Amsterdam Database Example

Row	Coffeeshop	Specialty	Price	Location
1	Dampkring	Skywalker	Moderate	Center
2	Paradox	Hindu Kush	Expensive	Center
3	Boerejongens	S5 Haze	Moderate	Center
4	Cheech & Chong's	S5 Haze	Cheap	West
5	Sensimillia	Tbisla	Cheap	West
6	Greenhouse	Hindu Kush	Expensive	East
7	The Stud	Tbisla	Cheap	East

global dependencies:

Coffeeshop $\rightarrow$ {Specialty, Price, Location}

{Specialty, Location} $\rightarrow$ {Price, Coffeeshop}

local dependencies:

Location $\rightarrow$ Price at rows 4,5

Specialty $\rightarrow$ Price at rows 1,2,5,6,7

A Typical Amsterdam Database Example

Row	Coffeeshop	Specialty	Price	Location
1	Dampkring	Skywalker	Moderate	Center
2	Paradox	Hindu Kush	Expensive	Center
3	Boerejongens	S5 Haze	Moderate	Center
4	Cheech & Chong's	S5 Haze	Cheap	West
5	Sensimillia	Tbisla	Cheap	West
6	Greenhouse	Hindu Kush	Expensive	East
7	The Stud	Tbisla	Cheap	East

global dependencies:

Coffeeshop $\rightarrow$ {Specialty, Price, Location}

{Specialty, Location} $\rightarrow$ {Price, Coffeeshop}

local dependencies:

Location $\rightarrow$ Price	at rows 4,5
Specialty $\rightarrow$ Price	at rows 1,2,5,6,7
Speciality $\rightarrow$ Location	at row 1 only

The Logic of Functional Dependence

What is to come?

We saw some proposals for generalizing first-order quantification to dependence models, and the concept of local dependence with an example.

What is to come?

We saw some proposals for generalizing first-order quantification to dependence models, and the concept of local dependence with an example.

We go on to introduce the logic LFD and go into the proof of completeness and decidability from the original paper. We will need these for the proofs later on and they will connect LFD to the Guarded Fragment (GF).

What is to come?

We saw some proposals for generalizing first-order quantification to dependence models, and the concept of local dependence with an example.

We go on to introduce the logic LFD and go into the proof of completeness and decidability from the original paper. We will need these for the proofs later on and they will connect LFD to the Guarded Fragment (GF).

Then we can dive into the proof of the finite model property via Herwig's theorem, indicating where and how it differs from Grädel's result for GF.

What is to come?

We saw some proposals for generalizing first-order quantification to dependence models, and the concept of local dependence with an example.

We go on to introduce the logic LFD and go into the proof of completeness and decidability from the original paper. We will need these for the proofs later on and they will connect LFD to the Guarded Fragment (GF).

Then we can dive into the proof of the finite model property via Herwig's theorem, indicating where and how it differs from Grädel's result for GF.

Time permitting, we may go into more detail about the relation between CRS and LFD, and how LFD solves an open problem from (van Benthem, 1996) or even dependence bisimulations.

Syntax

Fix a finite set of variables V and a relational signature τ.

Syntax

Fix a finite set of variables V and a relational signature τ.

Definition (Syntax)

The language $LFD[V, \tau]$ over (V, τ) is recursively defined by:

$$\varphi ::= P\boldsymbol{x} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbb{D}_X\varphi \mid D_X y$$

where $X \subseteq_\omega V$ $y \in V$, $P \in \tau$ and $\boldsymbol{x} = (x_1, ..., x_n) \in V^{ar(P)}$.

Syntax

Fix a finite set of variables V and a relational signature τ.

Definition (Syntax)

The language $LFD[V, \tau]$ over (V, τ) is recursively defined by:

$$\varphi ::= P\boldsymbol{x} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbb{D}_X\varphi \mid D_X y$$

where $X \subseteq_\omega V$ $y \in V$, $P \in \tau$ and $\boldsymbol{x} = (x_1, ..., x_n) \in V^{ar(P)}$.

We pronounce the dependence atom $D_X y$ as "y (locally) depends on X" and the dependence quantifier $\mathbb{D}_X\varphi$ as "X (locally) determines truth of φ"

Syntax

Fix a finite set of variables V and a relational signature τ.

Definition (Syntax)

The language $LFD[V, \tau]$ over (V, τ) is recursively defined by:

$$\varphi ::= P\boldsymbol{x} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbb{D}_X\varphi \mid D_X y$$

where $X \subseteq_\omega V$ $y \in V$, $P \in \tau$ and $\boldsymbol{x} = (x_1, ..., x_n) \in V^{ar(P)}$.

We pronounce the dependence atom $D_X y$ as "y (locally) depends on X" and the dependence quantifier $\mathbb{D}_X\varphi$ as "X (locally) determines truth of φ"

Furthermore, we abbreviate $D_X Y := \bigwedge_{y \in Y} D_x y$ and $\mathbb{E}_X\varphi := \neg\mathbb{D}_X\neg\varphi$.

Semantics

Simply recall the definition of a dependence model!

Semantics

Simply recall the definition of a dependence model!

Definition (Dependence Models)

A **dependence model** $\mathbb{M}$ is a pair $\mathbb{M} = (M, A)$ of a relational first-order structure M, together with a fixed team $A \subseteq M^V$.

Semantics

Simply recall the definition of a dependence model!

Definition (Dependence Models)

A **dependence model** $\mathbb{M}$ is a pair $\mathbb{M} = (M, A)$ of a relational first-order structure M, together with a fixed team $A \subseteq M^V$.

This time, instead of the relation $=^x$ of agreeing 'up to x-values' we take $=_X$ as our fundamental update relation:

$$s =_X t \qquad \text{iff} \qquad s \upharpoonright X = t \upharpoonright X$$

Semantics

Simply recall the definition of a dependence model!

Definition (Dependence Models)

A **dependence model** $\mathbb{M}$ is a pair $\mathbb{M} = (M, A)$ of a relational first-order structure M, together with a fixed team $A \subseteq M^V$.

This time, instead of the relation $=^x$ of agreeing 'up to x-values' we take $=_X$ as our fundamental update relation:

$$s =_X t \qquad \text{iff} \qquad s \upharpoonright X = t \upharpoonright X$$

The semantics clauses are as follows (Booleans as usual):

$$s \models P\boldsymbol{x} \text{ iff } s(\boldsymbol{x}) \in I^M(P)$$
$$s \models \mathbb{D}_X\varphi \text{ iff } t \models \varphi \text{ holds for all } t \in A \text{ with } s =_X t$$
$$s \models D_X y \text{ iff } s =_X t \text{ implies } s =_y t \text{ for all } t \in A.$$

LFD satisfies Locality

For every $\varphi \in$LFD, we define its **free variables** by:

- $\mathit{free}(D_X y) = X$
- $\mathit{free}(\mathbb{D}_X \varphi) = X$
- $\mathit{free}(Px_1 ... x_n) = \{x_1, ..., x_n\}$
- $\mathit{free}(\neg\varphi) = \mathit{free}(\varphi)$
- $\mathit{free}(\varphi \wedge \psi) = \mathit{free}(\varphi) \cup \mathit{free}(\psi)$

LFD satisfies Locality

For every $\varphi \in$LFD, we define its **free variables** by:

- $\mathit{free}(D_X y) = X$
- $\mathit{free}(\mathbb{D}_X \varphi) = X$
- $\mathit{free}(Px_1 ... x_n) = \{x_1, ..., x_n\}$
- $\mathit{free}(\neg\varphi) = \mathit{free}(\varphi)$
- $\mathit{free}(\varphi \wedge \psi) = \mathit{free}(\varphi) \cup \mathit{free}(\psi)$

In contrast to CRS, LFD does satisfy Locality! (just like FOL)

(**Locality**): If $s =_X t$ then $s \models \varphi$ iff $t \models \varphi$, whenever $\mathit{free}(\varphi) \subseteq X$

First-Order Translation

Let V be finite and V' a set of copies with enumerations $\mathbf{v}, \mathbf{v}'$ resp.

First-Order Translation

Let V be finite and V' a set of copies with enumerations $\mathbf{v}, \mathbf{v}'$ resp.

- $tr(P\boldsymbol{x}) := P\boldsymbol{x}$
- tr commutes with $\neg, \wedge$
- $tr(\mathbb{D}_X\psi) := \forall \boldsymbol{z}(A\boldsymbol{v} \to tr(\psi))$
- $tr(D_X x) := \top_X(\bigwedge_{x \in X} x = x)$ (for $x \in X$)
- $tr(D_X y) := \forall \boldsymbol{z} \forall \boldsymbol{z'}((A\boldsymbol{v} \wedge A\boldsymbol{v}[\boldsymbol{z'}/\boldsymbol{z}]) \to y = y')$ (for $y \notin X$)

First-Order Translation

Let V be finite and V' a set of copies with enumerations $\mathbf{v}, \mathbf{v}'$ resp.

- $tr(P\boldsymbol{x}) := P\boldsymbol{x}$
- tr commutes with $\neg, \wedge$
- $tr(\mathbb{D}_X\psi) := \forall \boldsymbol{z}(A\boldsymbol{v} \to tr(\psi))$
- $tr(D_X x) := \top x(\bigwedge_{x \in X} x = x)$ (for $x \in X$)
- $tr(D_X y) := \forall \boldsymbol{z} \forall \boldsymbol{z'}((A\boldsymbol{v} \wedge A\boldsymbol{v}[\boldsymbol{z'}/\boldsymbol{z}]) \to y = y')$ (for $y \notin X$)

e.g. if $V = \{x, y, z\}$ we have $tr(\mathbb{D}_x Rxy) = \forall yz(Axyz \to Rxy)$ and $tr(D_x y) = \forall yz \forall y'z'((Axyz \wedge Axy'z') \to y = y')$

First-Order Translation

Let V be finite and V' a set of copies with enumerations $\mathbf{v}, \mathbf{v}'$ resp.

- $tr(P\boldsymbol{x}) := P\boldsymbol{x}$
- tr commutes with $\neg, \wedge$
- $tr(\mathbb{D}_X\psi) := \forall \boldsymbol{z}(A\boldsymbol{v} \to tr(\psi))$
- $tr(D_X x) := \top_X(\bigwedge_{x \in X} x = x)$ (for $x \in X$)
- $tr(D_X y) := \forall \boldsymbol{z}\forall \boldsymbol{z'}((A\boldsymbol{v} \wedge A\boldsymbol{v}[\boldsymbol{z'}/\boldsymbol{z}]) \to y = y')$ (for $y \notin X$)

e.g. if $V = \{x, y, z\}$ we have $tr(\mathbb{D}_x Rxy) = \forall yz(Axyz \to Rxy)$ and $tr(D_x y) = \forall yz\forall y'z'((Axyz \wedge Axy'z') \to y = y')$
The correspondence $\mathbb{M} = (M, A) \rightsquigarrow T(\mathbb{M})$ is one-to-one, where

$$I^{T(\mathbb{M})}(A) := \{s(\mathbf{v}) \mid s \in A\}$$

Theorem (Baltag & Van Benthem, 2021)

$$\mathbb{M}, s \models \psi \qquad \textit{iff} \qquad T(\mathbb{M}), s \models tr(\psi)$$

Consequences

It follows that ψ and $tr(\psi)$ are equi-satisfiable and so LFD inherits **Compactness, Löwenheim-Skolem** properties and **recursive enumerability** of its validities from FOL.

Consequences

It follows that ψ and $tr(\psi)$ are equi-satisfiable and so LFD inherits **Compactness, Löwenheim-Skolem** properties and **recursive enumerability** of its validities from FOL.

Note that **tr maps into the GF except for tr($\mathbf{D_x y}$)** ! The same translation reduces CRS-SAT to GF-SAT (van Benthem, 2005).

Consequences

It follows that ψ and $tr(\psi)$ are equi-satisfiable and so LFD inherits **Compactness, Löwenheim-Skolem** properties and **recursive enumerability** of its validities from FOL.

Note that **tr maps into the GF except for tr($\mathbf{D_{xy}}$)** ! The same translation reduces CRS-SAT to GF-SAT (van Benthem, 2005).

Our observation suggest building on Grädel's proof of the fmp for GF, which uses Herwig's theorem on extending partial isomorphisms.

Consequences

It follows that ψ and $tr(\psi)$ are equi-satisfiable and so LFD inherits **Compactness, Löwenheim-Skolem** properties and **recursive enumerability** of its validities from FOL.

Note that **tr maps into the GF except for tr($\mathbf{D_x y}$)** ! The same translation reduces CRS-SAT to GF-SAT (van Benthem, 2005).

Our observation suggest building on Grädel's proof of the fmp for GF, which uses Herwig's theorem on extending partial isomorphisms.

We deviate from Grädel's approach by using *quasi-models* (here called 'type models') rather than a Scott Normal form.

Type Models

The **closure** Ψ of an LFD-formula ψ is defined as the closure of $\{\psi\}$ under subformulas, single negations and all $D_X y$ for X, y occurring in ψ.

Type Models

The **closure** Ψ of an LFD-formula ψ is defined as the closure of $\{\psi\}$ under subformulas, single negations and all $D_X y$ for X, y occurring in ψ.

Then $\Delta \subseteq \Psi$ is a **type** (for Ψ) if it satisfies:

(a) $\neg\psi \in \Delta$ iff $\psi \notin \Delta$

(b) $(\psi \wedge \chi) \in \Delta$ iff $\psi \in \Delta$ and $\chi \in \Delta$

(c) if $\mathbb{D}_X\psi \in \Delta$, then $\psi \in \Delta$

(d) $D_X x \in \Delta$ for all $x \in X$

(e) $D_X Y, D_Y Z \in \Delta$ implies $D_X Z \in \Delta$

Type Models

The **closure** Ψ of an LFD-formula ψ is defined as the closure of $\{\psi\}$ under subformulas, single negations and all $D_X y$ for X, y occurring in ψ.

Then $\Delta \subseteq \Psi$ is a **type** (for Ψ) if it satisfies:

(a) $\neg\psi \in \Delta$ iff $\psi \notin \Delta$

(b) $(\psi \wedge \chi) \in \Delta$ iff $\psi \in \Delta$ and $\chi \in \Delta$

(c) if $\mathbb{D}_X\psi \in \Delta$, then $\psi \in \Delta$

(d) $D_X x \in \Delta$ for all $x \in X$

(e) $D_X Y, D_Y Z \in \Delta$ implies $D_X Z \in \Delta$

$$\Delta \sim_X \Delta' \quad \text{iff} \quad \{\psi \in \Delta \mid \mathit{free}(\psi) \subseteq D_X^{\Delta}\} = \{\psi \in \Delta' \mid \mathit{free}(\psi) \subseteq D_X^{\Delta}\}$$

Type Models

The **closure** Ψ of an LFD-formula ψ is defined as the closure of $\{\psi\}$ under subformulas, single negations and all $D_X y$ for X, y occurring in ψ.

Then $\Delta \subseteq \Psi$ is a **type** (for Ψ) if it satisfies:

(a) $\neg\psi \in \Delta$ iff $\psi \notin \Delta$

(b) $(\psi \wedge \chi) \in \Delta$ iff $\psi \in \Delta$ and $\chi \in \Delta$

(c) if $\mathbb{D}_X\psi \in \Delta$, then $\psi \in \Delta$

(d) $D_X x \in \Delta$ for all $x \in X$

(e) $D_X Y, D_Y Z \in \Delta$ implies $D_X Z \in \Delta$

$$\Delta \sim_X \Delta' \quad \text{iff} \quad \{\psi \in \Delta \mid \mathit{free}(\psi) \subseteq D_X^\Delta\} = \{\psi \in \Delta' \mid \mathit{free}(\psi) \subseteq D_X^\Delta\}$$

where $D_X^\Delta = \{y \in V \mid D_X y \in \Delta\}$ the **dependence-closure** of X w.r.t Δ.

Type Models

The **closure** Ψ of an LFD-formula ψ is defined as the closure of $\{\psi\}$ under subformulas, single negations and all $D_X y$ for X, y occurring in ψ.

Then $\Delta \subseteq \Psi$ is a **type** (for Ψ) if it satisfies:

(a) $\neg\psi \in \Delta$ iff $\psi \notin \Delta$

(b) $(\psi \wedge \chi) \in \Delta$ iff $\psi \in \Delta$ and $\chi \in \Delta$

(c) if $\mathbb{D}_X\psi \in \Delta$, then $\psi \in \Delta$

(d) $D_X x \in \Delta$ for all $x \in X$

(e) $D_X Y, D_Y Z \in \Delta$ implies $D_X Z \in \Delta$

$$\Delta \sim_X \Delta' \quad \text{iff} \quad \{\psi \in \Delta \mid \mathit{free}(\psi) \subseteq D_X^\Delta\} = \{\psi \in \Delta' \mid \mathit{free}(\psi) \subseteq D_X^\Delta\}$$

where $D_X^\Delta = \{y \in V \mid D_X y \in \Delta\}$ the **dependence-closure** of X w.r.t Δ.

A **type model** is a set of (Ψ-)types $\mathfrak{M}$ such that:

(e) if $\mathbb{E}_X\psi \in \Delta$ then $\exists \Delta' \in \mathfrak{M}$ with $\Delta \sim_X \Delta'$ and $\psi \in \Delta$

(f) $\sim_\emptyset$ is the universal relation on $\mathfrak{M}$

Representing Type Models as Dependence Models

Dependence models induce type models $\{type(s) \mid s \in A\}$.

Representing Type Models as Dependence Models

Dependence models induce type models $\{type(s) \mid s \in A\}$.Conversely, every type model can be 'represented' as a dep. model of bounded tree-width. Completeness and decidability are then immediate.

Representing Type Models as Dependence Models

Dependence models induce type models $\{type(s) \mid s \in A\}$.Conversely, every type model can be 'represented' as a dep. model of bounded tree-width. Completeness and decidability are then immediate.

Fix a type Δ_0.

Representing Type Models as Dependence Models

Dependence models induce type models $\{type(s) \mid s \in A\}$.Conversely, every type model can be 'represented' as a dep. model of bounded tree-width. Completeness and decidability are then immediate.

Fix a type Δ_0. For every path $\pi = (\Delta_0 \sim_{X_1} \ \ldots \ \sim_{X_n} \Delta_n)$ ($lh(\pi) = n+1$ and $last(\pi) = \Delta_n$) through $\mathfrak{M}$ one associates a **path-assignment** v_π.

Representing Type Models as Dependence Models

Dependence models induce type models $\{type(s) \mid s \in A\}$.Conversely, every type model can be 'represented' as a dep. model of bounded tree-width. Completeness and decidability are then immediate.

Fix a type Δ_0. For every path $\pi = (\Delta_0 \sim_{X_1} \ \ldots \ \sim_{X_n} \Delta_n)$ ($lh(\pi) = n+1$ and $last(\pi) = \Delta_n$) through $\mathfrak{M}$ one associates a **path-assignment** v_π.

$$\begin{array}{lll} \text{if } \pi = (\Delta_0) \text{ is the root}, & v_\pi = (\pi, v) & \text{for all } v \in V \\ \text{if } \pi = (\pi' \sim_X last(\pi)), & v_\pi(v) = (\pi', x) & \text{if } v \in D_X^{last(\pi)} \\ & v_\pi(v) = (\pi, v) & \text{if } v \notin D_X^{\Delta_n} \end{array}$$

Representing Type Models as Dependence Models

Dependence models induce type models $\{type(s) \mid s \in A\}$.Conversely, every type model can be 'represented' as a dep. model of bounded tree-width. Completeness and decidability are then immediate.

Fix a type Δ_0. For every path $\pi = (\Delta_0 \sim_{X_1} \ \ldots \ \sim_{X_n} \Delta_n)$ ($lh(\pi) = n+1$ and $last(\pi) = \Delta_n$) through $\mathfrak{M}$ one associates a **path-assignment** v_π.

$$\begin{array}{lll} \text{if } \pi = (\Delta_0) \text{ is the root}, & v_\pi = (\pi, v) & \text{for all } v \in V \\ \text{if } \pi = (\pi' \sim_X last(\pi)), & v_\pi(v) = (\pi', x) & \text{if } v \in D_X^{last(\pi)} \\ & v_\pi(v) = (\pi, v) & \text{if } v \notin D_X^{\Delta_n} \end{array}$$

The collection of such paths forms a tree under $\preceq$. For every node π there is a $|V|$-sized guarded submodel $v_\pi[V]$

Bounded Tree-width Model: Drawing the Frame

(drawing)

Representing Type Models as Dependence Models

We can build a model M of bounded tree-width on such objects (π, x) by:

$$\mathbf{u} \in I(P) \qquad \text{iff} \qquad \mathbf{u} = v_\pi(\mathbf{x}) \text{ for some } v_\pi \in A \text{ with } P\mathbf{x} \in \mathit{last}(\pi)$$

Together with our team A this is a dependence model $\mathbb{M} = (M, A)$ which represents $\mathfrak{M}$, i.e. $\{\mathit{type}(v_\pi) \mid v_\pi \in A\} = \mathfrak{M}$.

Representing Type Models as Dependence Models

We can build a model M of bounded tree-width on such objects (π, x) by:

$$\mathbf{u} \in I(P) \qquad \text{iff} \qquad \mathbf{u} = v_\pi(\mathbf{x}) \text{ for some } v_\pi \in A \text{ with } P\mathbf{x} \in \mathit{last}(\pi)$$

Together with our team A this is a dependence model $\mathbb{M} = (M, A)$ which represents $\mathfrak{M}$, i.e. $\{\mathit{type}(v_\pi) \mid v_\pi \in A\} = \mathfrak{M}$.

Theorem (Truth Lemma)

$$\mathbb{M}, v_\pi \models \psi \qquad \mathit{iff} \qquad \psi \in \mathit{last}(\pi)$$

This holds because for each type $\Delta \in \mathfrak{M}$ there is a path $\pi_\Delta = (\Delta_0 \sim_\emptyset \Delta)$ such that $\mathit{type}(v_{\pi_\Delta}) = \mathit{last}(\pi_\Delta) = \Delta$ by the above truth lemma.

Representing Type Models as Dependence Models

We can build a model M of bounded tree-width on such objects (π, x) by:

$$\mathbf{u} \in I(P) \qquad \text{iff} \qquad \mathbf{u} = v_\pi(\mathbf{x}) \text{ for some } v_\pi \in A \text{ with } P\mathbf{x} \in \mathit{last}(\pi)$$

Together with our team A this is a dependence model $\mathbb{M} = (M, A)$ which represents $\mathfrak{M}$, i.e. $\{\mathit{type}(v_\pi) \mid v_\pi \in A\} = \mathfrak{M}$.

Theorem (Truth Lemma)

$$\mathbb{M}, v_\pi \models \psi \qquad \mathit{iff} \qquad \psi \in \mathit{last}(\pi)$$

This holds because for each type $\Delta \in \mathfrak{M}$ there is a path $\pi_\Delta = (\Delta_0 \sim_\emptyset \Delta)$ such that $\mathit{type}(v_{\pi_\Delta}) = \mathit{last}(\pi_\Delta) = \Delta$ by the above truth lemma.

The fmp asks whether every type model can be represented by a **finite** dependence model!

Comparison to GF

An entirely similar construction for GF is given in (Andréka, Németi & van Benthem, 1998).

Comparison to GF

An entirely similar construction for GF is given in (Andréka, Németi & van Benthem, 1998).

"Grädel": Cut the generating tree after the finite stage after which no new types are created. Apply Herwig's theorem, making *all* local symmetries into global ones. Then there is a guarded-bisimulates between this finite model and our infinite unravelling.

Comparison to GF

An entirely similar construction for GF is given in (Andréka, Németi & van Benthem, 1998).

"Grädel": Cut the generating tree after the finite stage after which no new types are created. Apply Herwig's theorem, making *all* local symmetries into global ones. Then there is a guarded-bisimulates between this finite model and our infinite unravelling.

Me: Cut at level 3 and apply Herwig's theorem w.r.t. a *special choice of symmetries*. Encode dependence formulas with atoms $R^{X,y}\mathbf{x}$ and use *extra conditions* guaranteed by the theorem to ensure dependencies will be respected in the process of extension. Then there is a dependence bisimulation between the finite Herwig extension and our infinite bounded tree-width model.

Comparison to GF

An entirely similar construction for GF is given in (Andréka, Németi & van Benthem, 1998).

"Grädel": Cut the generating tree after the finite stage after which no new types are created. Apply Herwig's theorem, making *all* local symmetries into global ones. Then there is a guarded-bisimulates between this finite model and our infinite unravelling.

Me: Cut at level 3 and apply Herwig's theorem w.r.t. a *special choice of symmetries*. Encode dependence formulas with atoms $R^{X,y}\mathbf{x}$ and use *extra conditions* guaranteed by the theorem to ensure dependencies will be respected in the process of extension. Then there is a dependence bisimulation between the finite Herwig extension and our infinite bounded tree-width model.

There is also a proof of the fmp through the modal semantics using am even stronger version of Herwig's theorem

High level Description of the Proof

Observe that, *since there are only finitely many types, there is some finite stage after which no new types are created*. We cut our model that deep.

High level Description of the Proof

Observe that, *since there are only finitely many types, there is some finite stage after which no new types are created*. We cut our model that deep.

We **cut** $\mathbb{M}$ **at height 3** as each type $\Delta \in \mathfrak{M}$ occurs as $type(v_{\pi_\Delta})$ for the path $\pi_\Delta = (\Delta_0 \sim_\emptyset \Delta)$. Call the resulting dependence model $\mathbb{M}_{cut}$

High level Description of the Proof

Observe that, *since there are only finitely many types, there is some finite stage after which no new types are created*. We cut our model that deep.

We **cut** $\mathbb{M}$ **at height 3** as each type $\Delta \in \mathfrak{M}$ occurs as $type(v_{\pi_\Delta})$ for the path $\pi_\Delta = (\Delta_0 \sim_\emptyset \Delta)$. Call the resulting dependence model $\mathbb{M}_{cut}$

We want to apply **Herwig's theorem** to $T(\mathbb{M}_{cut})$, making sure the Herwig extension will respect the interpretation of dependence atoms in the underlying types.

High level Description of the Proof

Observe that, *since there are only finitely many types, there is some finite stage after which no new types are created*. We cut our model that deep.

We **cut** $\mathbb{M}$ **at height 3** as each type $\Delta \in \mathfrak{M}$ occurs as $type(v_{\pi_\Delta})$ for the path $\pi_\Delta = (\Delta_0 \sim_\emptyset \Delta)$. Call the resulting dependence model $\mathbb{M}_{cut}$

We want to apply **Herwig's theorem** to $T(\mathbb{M}_{cut})$, making sure the Herwig extension will respect the interpretation of dependence atoms in the underlying types.

The key idea is to use fresh atomic formulas $R^{X,y}\mathbf{x}$ to **encode the dependence atoms** $D_X y$. If we can show that $R^{X,y}\mathbf{x} \leftrightarrow D_X y$ holds in the Herwig extension, we would be done by Grädel's result.

Herwig's theorem

We have to go beyond Grädel's use of Herwig's theorem.

Herwig's theorem

We have to go beyond Grädel's use of Herwig's theorem.

Theorem (Herwig)

Let σ be a finite relational language, C a finite σ-structure and $p_1, ..., p_k$ partial isomorphisms on C. Then there exists a finite extension C^+ of C that satisfies the following conditions:

(i) *Every p_i extends to a unique automorphism $\widehat{p_i}$ of C^+ [inducing a subgroup $\langle \widehat{p_1}, ..., \widehat{p_k} \rangle$ of $Aut(C^+)$]*

(ii) *If a tuple $\boldsymbol{c}$ from C^+ is guarded or a singleton, then there exists an automorphism $f \in \langle \widehat{p_1}, ..., \widehat{p_k} \rangle$ such that for each $c_i \in \boldsymbol{c}$, $f(c_i) \in C$.*

(iii) *If $\exists f \in \langle \widehat{p_1}, ..., \widehat{p_k} \rangle$ and $c, c' \in C$ such that $f(c) = c'$, then either $f = id$ or there is a unique $p \in \langle p_1, ..., p_k \rangle$ such that $\widehat{p} = f$*

Herwig's theorem

We have to go beyond Grädel's use of Herwig's theorem.

Theorem (Herwig)

Let σ be a finite relational language, C a finite σ-structure and $p_1, ..., p_k$ partial isomorphisms on C. Then there exists a finite extension C^+ of C that satisfies the following conditions:

(i) *Every p_i extends to a unique automorphism $\widehat{p_i}$ of C^+ [inducing a subgroup $\langle \widehat{p_1}, ..., \widehat{p_k} \rangle$ of $Aut(C^+)$]*

(ii) *If a tuple $\boldsymbol{c}$ from C^+ is guarded or a singleton, then there exists an automorphism $f \in \langle \widehat{p_1}, ..., \widehat{p_k} \rangle$ such that for each $c_i \in \boldsymbol{c}$, $f(c_i) \in C$.*

(iii) *If $\exists f \in \langle \widehat{p_1}, ..., \widehat{p_k} \rangle$ and $c, c' \in C$ such that $f(c) = c'$, then either $f = id$ or there is a unique $p \in \langle p_1, ..., p_k \rangle$ such that $\widehat{p} = f$*

we make a **choice of partial isomorphisms** and make crucial use of **(iii)**.

Cut-Off Model: Expansion and Partial Isomorphisms

$p_\pi : v_\pi(\mathbf{v}) \mapsto v_{\pi_\Delta}(\mathbf{v})$
where $last(\pi) = \Delta$

$$I(R^{X,y}) := \{s(\mathbf{x}) \mid s \in A\}$$

The Herwig Extension

By Herwig's theorem we obtain a dependence model $\mathbb{M}^+_{cut} = (M^+_{cut}, A^+_{cut})$ w.r.t. the partial iso's $p_\pi : v_\pi(\mathbf{v}) \mapsto v_{\pi_\Delta}(\mathbf{v})$ for $lh(\pi) = 3, last(\pi) = \Delta$.

The Herwig Extension

By Herwig's theorem we obtain a dependence model $\mathbb{M}^{+}_{cut} = (M^{+}_{cut}, A^{+}_{cut})$ w.r.t. the partial iso's $p_\pi : v_\pi(\mathbf{v}) \mapsto v_{\pi_\Delta}(\mathbf{v})$ for $lh(\pi) = 3, last(\pi) = \Delta$.

Crucially, it follows that $s(\mathbf{v}) \mapsto v_\pi(\mathbf{v})$ of level 2 under some $f \in \langle \widehat{p_1}, ..., \widehat{p_k} \rangle$

The Herwig Extension

By Herwig's theorem we obtain a dependence model $\mathbb{M}^+_{cut} = (M^+_{cut}, A^+_{cut})$ w.r.t. the partial iso's $p_\pi : v_\pi(\mathbf{v}) \mapsto v_{\pi_\Delta}(\mathbf{v})$ for $lh(\pi) = 3, last(\pi) = \Delta$.

Crucially, it follows that $s(\mathbf{v}) \mapsto v_\pi(\mathbf{v})$ of level 2 under some $f \in \langle \widehat{p_1}, ..., \widehat{p_k} \rangle$

(drawing)

Key Lemmas

We let $type(s) = last(\pi)$ for $s \overset{f}{\mapsto} v_\pi$ of level 2. This is well-defined by (iii) and properties of $\langle p_1, .., p_k \rangle$, and note that $type(v_\pi) = last(\pi)$.

Key Lemmas

We let $type(s) = last(\pi)$ for $s \overset{f}{\mapsto} v_\pi$ of level 2. This is well-defined by (iii) and properties of $\langle p_1, .., p_k \rangle$, and note that $type(v_\pi) = last(\pi)$.

Lemma (Type Lemma)

For all $s, t \in A^+_{cut}$, $s =_X t$ *implies* $type(s) \sim_X type(t)$

Key Lemmas

We let $type(s) = last(\pi)$ for $s \overset{f}{\mapsto} v_\pi$ of level 2. This is well-defined by (iii) and properties of $\langle p_1, .., p_k \rangle$, and note that $type(v_\pi) = last(\pi)$.

Lemma (Type Lemma)

For all $s, t \in A^+_{cut}$, $s =_X t$ *implies* $type(s) \sim_X type(t)$

Lemma (Encoding lemma)

$$\mathbb{M}^+_{cut}, s \models R^{X,y}\mathbf{x} \leftrightarrow D_X y \qquad \forall s \in A^+_{cut}$$

($\leftarrow$) follows from Łos-Tarski as witnesses for $\neg D_X y$ in the type are given in one-step.

Key Lemmas

We let $type(s) = last(\pi)$ for $s \overset{f}{\mapsto} v_\pi$ of level 2. This is well-defined by (iii) and properties of $\langle p_1, .., p_k \rangle$, and note that $type(v_\pi) = last(\pi)$.

Lemma (Type Lemma)

For all $s, t \in A^+_{cut}$, $s =_X t$ *implies* $type(s) \sim_X type(t)$

Lemma (Encoding lemma)

$$\mathbb{M}^+_{cut}, s \models R^{X,y}\mathbf{x} \leftrightarrow D_X y \qquad \forall s \in A^+_{cut}$$

($\leftarrow$) follows from Łos-Tarski as witnesses for $\neg D_X y$ in the type are given in one-step.($\rightarrow$) is the most non-trivial part of the proof and uses (iii) and properties of $\langle p_1, ..., p_k \rangle$

Encoding Lemma Proof

Lemma

If $p \in \langle p_1, ..., p_k \rangle$ *is such that* $pv_\pi =_X v_{\pi'}$, *there are* $v_\rho, v_{\rho'} \in A_{cut}$ *with* $v_\pi =_X v_\rho, v_{\pi'} =_X v_{\rho'}$ *and* $pv_\rho = v_{\rho'}$ *which implies that* $last(\rho) = last(\rho')$.

$[D_x y \in last(\sigma)$ and $v_\sigma =_X v_{\sigma'}$ implies $v_\sigma =_y v_{\sigma'}$ for $v_\sigma, v_{\sigma'} \in A_{cut}]$

Finite Model Property via Bisimulation

Phil Pützstück and me independently proposed equivalent definitions of **dependence bisimulations** characterizing LFD as a fragment of FOL.

Finite Model Property via Bisimulation

Phil Pützstück and me independently proposed equivalent definitions of **dependence bisimulations** characterizing LFD as a fragment of FOL.

We finish the proof by giving a dependence bisimulation

$$Z := \{(s, v_\pi) \in A^+_{cut} \times A \mid type(s) = last(\pi)\}$$

between the **finite** Herwig extension $\mathbb{M}^+_{cut}$ and the **infinite** unravelling $\mathbb{M}$. It follows that $\mathbb{M}^+_{cut}$ is a finite dependence model representing $\mathfrak{M}$.

Finite Model Property via Bisimulation

Phil Pützstück and me independently proposed equivalent definitions of **dependence bisimulations** characterizing LFD as a fragment of FOL.

We finish the proof by giving a dependence bisimulation

$$Z := \{(s, v_\pi) \in A^+_{cut} \times A \mid type(s) = last(\pi)\}$$

between the **finite** Herwig extension $\mathbb{M}^+_{cut}$ and the **infinite** unravelling $\mathbb{M}$. It follows that $\mathbb{M}^+_{cut}$ is a finite dependence model representing $\mathfrak{M}$.

Theorem (Finite Model Property)

LFD has the finite model property (w.r.t. the intended semantics)

Extra Topics

(time permitting)

Comparing LFD and CRS: Axiomatization

The authors provide a complete Hilbert-style axiomatization as well as a complete sequent-calculus with a restricted form of cut-elimination

Comparing LFD and CRS: Axiomatization

The authors provide a complete Hilbert-style axiomatization as well as a complete sequent-calculus with a restricted form of cut-elimination

Table 1 The proof system **LFD**

(I)	**Axioms and rules of classical propositional logic**
(II)	**Axioms and rules for dependence modalities $\mathbb{D}$**
($\mathbb{D}$-Necessitation)	From φ, infer $\mathbb{D}_X\varphi$
($\mathbb{D}$-Distribution)	$\mathbb{D}_X(\varphi \to \psi) \to (\mathbb{D}_X\varphi \to \mathbb{D}_X\psi)$
($\mathbb{D}$-Introduction)	$\varphi \to \mathbb{D}_X\varphi$, provided that $Free(\varphi) \subseteq X$
($\mathbb{D}$-Elimination)	$\mathbb{D}_X\varphi \to \varphi$
(III)	**Axioms for dependence atoms D**
(Projection)	$D_X x$, provided that $x \in X$
(Transitivity)	$(D_X Y \wedge D_Y Z) \to D_X Z$
(IV)	**Axiom for $\mathbb{D}$-D interaction**
(Transfer)	$(D_X Y \wedge \mathbb{D}_Y\varphi) \to \mathbb{D}_X\varphi$

Comparing LFD and CRS: Axiomatization

CRS (without substitutions) is completely axiomatized by poly-modal S5 for the $\exists X$ modalities plus the schemata

$$P\mathbf{x} \to \forall Y P\mathbf{x}, \qquad \neg P\mathbf{x} \to \forall Y \neg P\mathbf{x} \qquad \text{where no } y \in Y \text{ occurs in } \mathbf{x}$$

for each atomic formula $P\mathbf{x}$ (where $\forall Y$ is shorthand for $\neg\exists Y\neg$)

Comparing LFD and CRS: Axiomatization

CRS (without substitutions) is completely axiomatized by poly-modal S5 for the $\exists X$ modalities plus the schemata

$$P\mathbf{x} \to \forall Y P\mathbf{x}, \qquad \neg P\mathbf{x} \to \forall Y \neg P\mathbf{x} \qquad \text{where no } y \in Y \text{ occurs in } \mathbf{x}$$

for each atomic formula $P\mathbf{x}$ (where $\forall Y$ is shorthand for $\neg\exists Y\neg$)

Observe that [$\mathbb{D}$-Elim] is just (T) and

$$[\mathbb{D}\text{-Introduction}] \quad \varphi \to \mathbb{D}_X\varphi \qquad \text{whenever } \mathit{free}(\varphi) \subseteq X$$

boils down to a couple of relevant instances,

Comparing LFD and CRS: Axiomatization

CRS (without substitutions) is completely axiomatized by poly-modal S5 for the $\exists X$ modalities plus the schemata

$$P\mathbf{x} \to \forall Y P\mathbf{x}, \qquad \neg P\mathbf{x} \to \forall Y \neg P\mathbf{x} \qquad \text{where no } y \in Y \text{ occurs in } \mathbf{x}$$

for each atomic formula $P\mathbf{x}$ (where $\forall Y$ is shorthand for $\neg\exists Y\neg$)

Observe that [$\mathbb{D}$-Elim] is just (T) and

$$[\mathbb{D}\text{-Introduction}] \quad \varphi \to \mathbb{D}_X\varphi \qquad \text{whenever } \mathit{free}(\varphi) \subseteq X$$

boils down to a couple of relevant instances,including

$$Px_1 \ldots x_n \to \mathbb{D}_{\{x_1,\ldots,x_n\}} Px_1 \ldots x_n \qquad (\mathit{schema})$$

$$\neg Px_1 \ldots x_n \to \mathbb{D}_{\{x_1,\ldots,x_n\}} \neg Px_1 \ldots x_n \qquad (\mathit{schema})$$

$$\mathbb{D}_X\varphi \to \mathbb{D}_X\mathbb{D}_X\varphi \qquad (4)$$

$$\neg\mathbb{D}_X\varphi \to \mathbb{D}_X\mathbb{D}_X\varphi \qquad (5)$$

Comparing LFD and CRS: Translations

Over a *finite* set V of variables, CRS-quantifiers and dependence quantifiers are inter-definable:

$$\forall X\phi \equiv \mathbb{D}_{V-X}\phi \qquad\qquad \mathbb{D}_X\phi \equiv \forall V - X\phi$$

Comparing LFD and CRS: Translations

Over a *finite* set V of variables, CRS-quantifiers and dependence quantifiers are inter-definable:

$$\forall X\phi \equiv \mathbb{D}_{V-X}\phi \qquad\qquad \mathbb{D}_X\phi \equiv \forall V - X\phi$$

Even the local versions are inter-definable with dependence quantifiers:

$$\widetilde{\forall}_X\phi \equiv \mathbb{D}_{Free(\phi)-X}\phi \qquad\qquad \mathbb{D}_X\phi \equiv \forall_{Free(\phi)-X}(\phi \wedge \top_X)$$

where $\top_X$ is any tautology in variables X (e.g. $\bigwedge_{x\in X} x = x$)

Comparing LFD and CRS: Translations

Over a *finite* set V of variables, CRS-quantifiers and dependence quantifiers are inter-definable:

$$\forall X\phi \equiv \mathbb{D}_{V-X}\phi \qquad\qquad \mathbb{D}_X\phi \equiv \forall V - X\phi$$

Even the local versions are inter-definable with dependence quantifiers:

$$\widetilde{\forall}_X\phi \equiv \mathbb{D}_{Free(\phi)-X}\phi \qquad\qquad \mathbb{D}_X\phi \equiv \forall_{Free(\phi)-X}(\phi \wedge \top_X)$$

where $\top_X$ is any tautology in variables X (e.g. $\bigwedge_{x\in X} x = x$)

For an infinite V, the two notions seem to be independent; $\mathbb{D}_{V-x}\varphi$ and $\forall V - X\varphi$ are not formulas if V is infinite.

Modal Semantics

We now turn to the modal perspective on LFD, to see how it challenges CRS's update relation $=^x$ as the right way of representing abstract modal models/frames as dependence models.

Modal Semantics

We now turn to the modal perspective on LFD, to see how it challenges CRS's update relation $=^x$ as the right way of representing abstract modal models/frames as dependence models.

A **general relational model** is a structure $\mathbb{A} = (A, \sim_X, D_X y, P\mathbf{x})$ s.t.

(1) All $\sim_X$ are equivalence relations

(2) $\{(X, y) \mid s \in ||D_X y||\}$ satisfies Projection and Transitivity

(3) $||P\mathbf{x}||$ is a union of sets of the form $[s]_X$

(4) $\sim_\emptyset$ is the universal relation

(5) $||D_X y|| \subseteq \{s \in A \mid [s]_X \subseteq [s]_y\}$

(6) $\sim_{X \cup Y} \subseteq \sim_X \cap \sim_Y$

Modal Semantics

We now turn to the modal perspective on LFD, to see how it challenges CRS's update relation $=^x$ as the right way of representing abstract modal models/frames as dependence models.

A **general relational model** is a structure $\mathbb{A} = (A, \sim_X, D_X y, P\mathbf{x})$ s.t.

(1) All $\sim_X$ are equivalence relations

(2) $\{(X, y) \mid s \in ||D_X y||\}$ satisfies Projection and Transitivity

(3) $||P\mathbf{x}||$ is a union of sets of the form $[s]_X$

(4) $\sim_\emptyset$ is the universal relation

(5) $||D_X y|| \subseteq \{s \in A \mid [s]_X \subseteq [s]_y\}$

(6) $\sim_{X \cup Y} \subseteq \sim_X \cap \sim_Y$

A **standard relational model** is a a general relational model $\mathbb{A}$ where (5),(6) are equalities, then all information is carried by the reduct $\mathbb{A} = (A, \sim_x, P\mathbf{x})$ because the interpretations for $\sim_X$ and $D_X y$ are determined by the relations $\sim_x$ for $x \in V$.

Correspondences

If $\mathbb{M} = (M, A)$ is a dependence model then $\mathit{rel}(\mathbb{M}) = (A, =_x, P\mathbf{x})$ is an equivalent standard relational model.

Correspondences

If $\mathbb{M} = (M, A)$ is a dependence model then $rel(\mathbb{M}) = (A, =_x, P\mathbf{x})$ is an equivalent standard relational model.

If $\mathbb{A}$ is a standard relational model then $dep(\mathbb{A}) = (M, A^{\sim})$ is an equivalent dependence model, with assignments $s^{\sim} \in A^{\sim}$:

$$s^{\sim}(x) = [s]_x$$

Correspondences

If $\mathbb{M} = (M, A)$ is a dependence model then $rel(\mathbb{M}) = (A, =_x, P\mathbf{x})$ is an equivalent standard relational model.

If $\mathbb{A}$ is a standard relational model then $dep(\mathbb{A}) = (M, A^{\sim})$ is an equivalent dependence model, with assignments $s^{\sim} \in A^{\sim}$:

$$s^{\sim}(x) = [s]_x$$

Similarly, it can be show that the non-standard general relational models can be unravelled into standard ones via modal techniques

Correspondences

If $\mathbb{M} = (M, A)$ is a dependence model then $rel(\mathbb{M}) = (A, =_x, P\mathbf{x})$ is an equivalent standard relational model.

If $\mathbb{A}$ is a standard relational model then $dep(\mathbb{A}) = (M, A^{\sim})$ is an equivalent dependence model, with assignments $s^{\sim} \in A^{\sim}$:

$$s^{\sim}(x) = [s]_x$$

Similarly, it can be show that the non-standard general relational models can be unravelled into standard ones via modal techniques

Theorem (Baltag & Van Benthem, 2021)

Type models, dependence models and standard and general relational models all provide equivalent semantics for LFD

Representing Abstract Modal Models

Call the *skeleton* of a dependence model $\mathbb{M} = (M, A)$ the abstract modal model $(A, (=^x)_{x \in V}, I)$ induced by it. What characterizes the skeletons?

[1] The path principles for $=^X$ are **duals** of the inclusions $\sim_X \cap \sim_Y \subseteq \sim_{X \cup Y}$, $\sim_X \subseteq \sim_x$.

Representing Abstract Modal Models

Call the *skeleton* of a dependence model $\mathbb{M} = (M, A)$ the abstract modal model $(A, (=^x)_{x \in V}, I)$ induced by it. What characterizes the skeletons?

Theorem (Van Benthem, 1996)

An abstract modal frame $(A, (R_x)_{x \in V})$ is isomorphic to the frame of a skeleton iff all R_x are equivalence relations and satisfy certain 'path principles' on the sequential compositions

[1] The path principles for $=^X$ are **duals** of the inclusions $\sim_X \cap \sim_Y \subseteq \sim_{X \cup Y}, \sim_X \subseteq \sim_x$.

Representing Abstract Modal Models

Call the *skeleton* of a dependence model $\mathbb{M} = (M, A)$ the abstract modal model $(A, (=^x)_{x \in V}, I)$ induced by it. What characterizes the skeletons?

Theorem (Van Benthem, 1996)

An abstract modal frame $(A, (R_x)_{x \in V})$ is isomorphic to the frame of a skeleton iff all R_x are equivalence relations and satisfy certain 'path principles' on the sequential compositions

But what if we drop (ii) insistence on the update relation $=^x$?

[1] The path principles for $=^X$ are **duals** of the inclusions $\sim_X \cap \sim_Y \subseteq \sim_{X \cup Y}, \sim_X \subseteq \sim_x$.

Representing Abstract Modal Models

Call the *skeleton* of a dependence model $\mathbb{M} = (M, A)$ the abstract modal model $(A, (=^x)_{x \in V}, I)$ induced by it. What characterizes the skeletons?

Theorem (Van Benthem, 1996)

An abstract modal frame $(A, (R_x)_{x \in V})$ is isomorphic to the frame of a skeleton iff all R_x are equivalence relations and satisfy certain 'path principles' on the sequential compositions

But what if we drop (ii) insistence on the update relation $=^x$?

Let us think of the skeletons in a new way! $Skel^*(\mathbb{M}) = (A, (=_x)_{x \in V}, I)$[1]

[1] The path principles for $=^X$ are **duals** of the inclusions $\sim_X \cap \sim_Y \subseteq \sim_{X \cup Y}, \sim_X \subseteq \sim_x$.

Representing Abstract Modal Models

Call the *skeleton* of a dependence model $\mathbb{M} = (M, A)$ the abstract modal model $(A, (=^x)_{x \in V}, I)$ induced by it. What characterizes the skeletons?

Theorem (Van Benthem, 1996)

An abstract modal frame $(A, (R_x)_{x \in V})$ is isomorphic to the frame of a skeleton iff all R_x are equivalence relations and satisfy certain 'path principles' on the sequential compositions

But what if we drop (ii) insistence on the update relation $=^x$?

Let us think of the skeletons in a new way! $Skel^*(\mathbb{M}) = (A, (=_x)_{x \in V}, I)$[1]

Theorem (Now, 2021)

An abstract modal model $(A, (R_x)_{x \in V}, I)$ is isomorphic to a skeleton* iff (i) each R_x is an equivalence relation and (ii) $||P\mathbf{x}||$ is a union of sets of the form $[s]_X = \{t \in A \mid (s, t) \in \bigcap_{x \in X} R_x\}$

[1] The path principles for $=^X$ are **duals** of the inclusions $\sim_X \cap \sim_Y \subseteq \sim_{X \cup Y}, \sim_X \subseteq \sim_x$.

Dependence Bisimulations

Dependence bisimulations can be equivalently defined in at 2 ways.

Dependence Bisimulations

Dependence bisimulations can be equivalently defined in at 2 ways. Phil lets the [atom] clause run over dependence atoms as well, and has one pair of zigzag clauses (let $V^{s,t} = \{v \mid s =_v t\}$)

[**forth**] For each $t \in A$ there is some $t' \in A'$ with $s' =_{V^{s,t}} t'$ and $(t, t') \in Z$

Dependence Bisimulations

Dependence bisimulations can be equivalently defined in at 2 ways. Phil lets the [atom] clause run over dependence atoms as well, and has one pair of zigzag clauses (let $V^{s,t} = \{v \mid s =_v t\}$)

[**forth**] For each $t \in A$ there is some $t' \in A'$ with $s' =_{V^{s,t}} t'$ and $(t, t') \in Z$

while i chose for a purely relation [atom] clause and designed extra conditions on the zigzag clauses to ensure preservation of $D_X y$

[**forth**+] For each $t \in A$ there is some $t' \in A$ with $s' =_{V^{s,t}} t'$, $(t, t') \in Z$ and $V^{s,t}$ is **dependence-closed** at s'

Dependence Bisimulations

Dependence bisimulations can be equivalently defined in at 2 ways. Phil lets the [atom] clause run over dependence atoms as well, and has one pair of zigzag clauses (let $V^{s,t} = \{v \mid s =_v t\}$)

[**forth**] For each $t \in A$ there is some $t' \in A'$ with $s' =_{V^{s,t}} t'$ and $(t, t') \in Z$

while i chose for a purely relation [atom] clause and designed extra conditions on the zigzag clauses to ensure preservation of $D_X y$

[**forth**+] For each $t \in A$ there is some $t' \in A$ with $s' =_{V^{s,t}} t'$, $(t, t') \in Z$ and $V^{s,t}$ is **dependence-closed** at s'

Theorem (Pützstuck, me, 2020)

LFD is the dependence bisimulation-invariant fragment of FOL

Dependence Bisimulations

Dependence bisimulations can be equivalently defined in at 2 ways. Phil lets the [atom] clause run over dependence atoms as well, and has one pair of zigzag clauses (let $V^{s,t} = \{v \mid s =_v t\}$)

[**forth**] For each $t \in A$ there is some $t' \in A'$ with $s' =_{V^{s,t}} t'$ and $(t, t') \in Z$

while i chose for a purely relation [atom] clause and designed extra conditions on the zigzag clauses to ensure preservation of $D_X y$

[**forth**+] For each $t \in A$ there is some $t' \in A$ with $s' =_{V^{s,t}} t'$, $(t, t') \in Z$ and $V^{s,t}$ is **dependence-closed** at s'

Theorem (Pützstuck, me, 2020)

LFD is the dependence bisimulation-invariant fragment of FOL

My definition allows for more efficient algorithms for bisimilarity checking that are not exponential in $|V|$.

Invariance

Proof:

Conclusion

Summary

We have discussed the origins of LFD arising from the study of generalised assignment models and CRS. If time has permitted it we have seen how LFD is a sense dual to CRS, and how it solves an open problem posed by Johan in 2005. The connection with dependence logics remains relatively unexplored...

Summary

We have discussed the origins of LFD arising from the study of generalised assignment models and CRS. If time has permitted it we have seen how LFD is a sense dual to CRS, and how it solves an open problem posed by Johan in 2005. The connection with dependence logics remains relatively unexplored...

We introduced LFD and proved completeness and decidability via an infinite representation of type models. Then we went through the proof of the fmp through Herwig's theorem, pinpointing the differences with GF.

Summary

We have discussed the origins of LFD arising from the study of generalised assignment models and CRS. If time has permitted it we have seen how LFD is a sense dual to CRS, and how it solves an open problem posed by Johan in 2005. The connection with dependence logics remains relatively unexplored...

We introduced LFD and proved completeness and decidability via an infinite representation of type models. Then we went through the proof of the fmp through Herwig's theorem, pinpointing the differences with GF.

The size of the Herwig extension is non-elementary in case of unbounded arities, but in our case it can be bounded by a $|V|$-tower of exponentials. The modal proof of the fmp ensures maximal arity 2. One may also find smaller models using an appropriate notion of *Rosati cover* as with GF.

Further Work

A major open question concerns the **computational complexity** of LFD. We are currently looking at polynomial-reductions to and from GF^k-SAT.

Further Work

A major open question concerns the **computational complexity** of LFD. We are currently looking at polynomial-reductions to and from GF^k-SAT.

One can also look at local versions of other dependency atoms, such as independence and inclusion. Phil Pütztstück has shown that LFD extended with either of these becomes undecidable. However, **is the pure logic of independence decidable?**.

Further Work

A major open question concerns the **computational complexity** of LFD. We are currently looking at polynomial-reductions to and from GF^k-SAT.

One can also look at local versions of other dependency atoms, such as independence and inclusion. Phil Pütztstück has shown that LFD extended with either of these becomes undecidable. However, **is the pure logic of independence decidable?**.

Design a **fixpoint extension** of LFD and prove its decidability.

Further Work

A major open question concerns the **computational complexity** of LFD. We are currently looking at polynomial-reductions to and from GF^k-SAT.

One can also look at local versions of other dependency atoms, such as independence and inclusion. Phil Pütztstück has shown that LFD extended with either of these becomes undecidable. However, **is the pure logic of independence decidable?**.

Design a **fixpoint extension** of LFD and prove its decidability.

I also want to study more closely the relation between **named relational databases** and dependence models, i.e. LFD as a query-language.

Further Work

A major open question concerns the **computational complexity** of LFD. We are currently looking at polynomial-reductions to and from GF^k-SAT.

One can also look at local versions of other dependency atoms, such as independence and inclusion. Phil Pütztstück has shown that LFD extended with either of these becomes undecidable. However, **is the pure logic of independence decidable?**.

Design a **fixpoint extension** of LFD and prove its decidability.

I also want to study more closely the relation between **named relational databases** and dependence models, i.e. LFD as a query-language.

knowledge wh-, mixed readings, dynamic logic of group-readings, logic of continuous dependence, causality, strategic dependence, dynamical systems.....

References

Johan van Benthem:

- *Exploring Logical Dynamics* (1996)
- *Guards, Bounds and Generalised Semantics* (2005)
- *CRS and Guarded Logics: A Fruitful Contact* (2009)

Hajnal Andréka, István Németi & Johan van Benthem

- Modal Languages and Bounded Fragments of Predicate Logic (1998)

Alexandru Baltag & Johan van Benthem

- *A Simple Logic of Functional Dependence* (2021)

Phil Pützstück & Erich Grädel (2021)

- *Logics of Dependence and Independence: the Local Variants* (2021)

Finite Model Property and Bisimulation for LFD (2021) (me)