Back and Forth Between Team Semantics, Games, and Tarski Semantics

Erich Grädel

Berkeley Logic Colloquium, October 2016

Outline

- Team semantics and logics of dependence and independence
- Expressive power via translations into existential second-order logic
- Model checking games for logics with team semantics
- Inclusion logic, safety games, and greatest fixed-point logic.

Dependence and independence

What does it mean that "x depends on y" or that "x and y are independent" ?

Dependence and independence

What does it mean that "x depends on y" or that "x and y are independent" ?

Compare this to similarly looking statements such as "x divides y".

Dependence and independence

What does it mean that "x depends on y" or that "x and y are independent" ?

Compare this to similarly looking statements such as "x divides y".

To make sense of this we fix a structure $\mathfrak{A}$ in which the notion of divisibility is well-defined and an assignment $s : \{x, y\} \to A$ and use Tarski-style semantics to determine whether $\mathfrak{A} \models_s$ "x divides y".

Dependence and independence

What does it mean that "x depends on y" or that "x and y are independent" ?

Compare this to similarly looking statements such as "x divides y".

To make sense of this we fix a structure $\mathfrak{A}$ in which the notion of divisibility is well-defined and an assignment $s : \{x, y\} \to A$ and use Tarski-style semantics to determine whether $\mathfrak{A} \models_s$ "x divides y".

Dependence and independence are notions of a different kind! They manifest themselves not in single assignments, but only in larger amounts of data:

- tables or relations
- sets of plays in a game (e.g. strategies)
- sets of assignments.

A set of assigments (all with the same domain of variables) is called a team.

Logics of dependence and independence

Henkin, Enderton, Walkoe, . . . : partially ordered (or Henkin-) quantifiers
Hintikka and Sandu: Independence-friendly (IF) logic with explicit dependencies of quantifiers on each other

Semantics in terms of games with imperfect information

Logics of dependence and independence

Henkin, Enderton, Walkoe, . . . : partially ordered (or Henkin-) quantifiers
Hintikka and Sandu: Independence-friendly (IF) logic with explicit dependencies of quantifiers on each other

Semantics in terms of games with imperfect information

Claim: Impossibility of model-theoretic (compositional) semantics
never made precise, never proved

Logics of dependence and independence

Henkin, Enderton, Walkoe, . . . : partially ordered (or Henkin-) quantifiers
Hintikka and Sandu: Independence-friendly (IF) logic with explicit dependencies of quantifiers on each other

Semantics in terms of games with imperfect information

Claim: Impossibility of model-theoretic (compositional) semantics
never made precise, never proved

Hodges: model-theoretic semantics for IF-logic

Difference to Tarski semantics: a formula is not evaluated against a single assignment but against a set of assignments

This kind of semantics is an important achievement of independent interest.

Logics of dependence and independence

Modern framework: Do not state dependencies as annotations of quantifiers, but treat them as atomic statements on teams (Väänänen)

Dependence:

$$\models_X =(\overline{x},\overline{y}) \quad :\Longleftrightarrow \quad (\forall s \in X)(\forall s' \in X)(s(\overline{x}) = s'(\overline{x}) \to s(\overline{y}) = s'(\overline{y}))$$

Inclusion:

$$\models_X (\overline{x} \subseteq \overline{y}) \quad :\Longleftrightarrow \quad (\forall s \in X)(\exists s' \in X)(s(\overline{x}) = s'(\overline{y}))$$

Exclusion:

$$\models_X (\overline{x} \mid \overline{y}) \quad :\Longleftrightarrow \quad (\forall s \in X)(\forall s' \in X)(s(\overline{x}) \neq s'(\overline{y}))$$

Independence:

$$\models_X (\overline{x} \perp \overline{y}) \quad :\Longleftrightarrow \quad X(\overline{x}\overline{y}) = X(\overline{x}) \times X(\overline{y})$$

Here $X(\overline{x})$ is the set of all values $s(\overline{x})$ for $s \in X$.

Dependence atoms

Dependence atoms are expressions $=(\overline{x}, y)$.

Semantics: Let $\mathfrak{A}$ be a structure and X a team of assignments $s : V \to A$.

$\mathfrak{A} \models_X =(\overline{x}, y)$ if y depends on $\overline{x}$ in $\mathfrak{A}$ and X.

This means that for all $s, s' \in X$,

$$\bigwedge_{i=1}^{n} s(x_i) = s'(x_i) \implies s(y) = s'(y)$$

Dependence atoms

Dependence atoms are expressions $=(\overline{x}, y)$.

Semantics: Let $\mathfrak{A}$ be a structure and X a team of assignments $s : V \to A$.

$\mathfrak{A} \models_X =(\overline{x}, y)$ if y depends on $\overline{x}$ in $\mathfrak{A}$ and X.

This means that for all $s, s' \in X$,

$$\bigwedge_{i=1}^{n} s(x_i) = s'(x_i) \implies s(y) = s'(y)$$

Expanded form of dependence atoms $=(\overline{x}, \overline{y})$ saying that all variables of $\overline{y}$ depend on $\overline{x}$.

Dependence atoms

Dependence atoms are expressions $=(\overline{x}, y)$.

Semantics: Let $\mathfrak{A}$ be a structure and X a team of assignments $s : V \to A$.

$\mathfrak{A} \models_X =(\overline{x}, y)$ if y depends on $\overline{x}$ in $\mathfrak{A}$ and X.

This means that for all $s, s' \in X$,

$$\bigwedge_{i=1}^{n} s(x_i) = s'(x_i) \implies s(y) = s'(y)$$

Expanded form of dependence atoms $=(\overline{x}, \overline{y})$ saying that all variables of $\overline{y}$ depend on $\overline{x}$.

Dependence atoms are downwards closed.

Independence atoms

Definition. A team X satisfies the atom $x \perp y$ if

$$(\forall s, s' \in X)(\exists s'' \in X)(s''(x) = s(x) \wedge s''(y) = s'(y))$$

Independence atoms

Definition. A team X satisfies the atom $x \perp y$ if

$$(\forall s, s' \in X)(\exists s'' \in X)(s''(x) = s(x) \wedge s''(y) = s'(y))$$

Suppose you know X and you know that s is some assignment in X. You want to gather information about $s(y)$. What you know is that $s(y) \in X(y) := \{a : a = s'(y) \text{ for some } s' \in X\}$.

Independence atoms

Definition. A team X satisfies the atom $x \perp y$ if

$$(\forall s, s' \in X)(\exists s'' \in X)(s''(x) = s(x) \wedge s''(y) = s'(y))$$

Suppose you know X and you know that s is some assignment in X.
You want to gather information about $s(y)$. What you know is that
$s(y) \in X(y) := \{a : a = s'(y) \text{ for some } s' \in X\}$.

Suppose that I tell you $s(x)$ where $x \perp y$.
You cannot infer anything new about $s(y)$. Indeed, for all potential values $a \in X(y)$ there is an assignment $s'' \in X$ with $s''(x) = s(x)$ and $s''(y) = a$.

Independence atoms

Definition. A team X satisfies the atom $x \perp y$ if

$$(\forall s, s' \in X)(\exists s'' \in X)(s''(x) = s(x) \wedge s''(y) = s'(y))$$

Suppose you know X and you know that s is some assignment in X.
You want to gather information about $s(y)$. What you know is that
$s(y) \in X(y) := \{a : a = s'(y) \text{ for some } s' \in X\}$.

Suppose that I tell you $s(x)$ where $x \perp y$.
You cannot infer anything new about $s(y)$. Indeed, for all potential values $a \in X(y)$ there is an assignment $s'' \in X$ with $s''(x) = s(x)$ and $s''(y) = a$.

Generalizes to independence atoms $\overline{x} \perp \overline{y}$ on tuples of variables.

Notice that $\overline{x} \perp \overline{y}$ holds in X iff $X(\overline{xy}) = X(\overline{x}) \times X(\overline{y})$.

Exclusion and inclusion dependencies

Exclusion dependencies:

$$\vDash_X (\overline{x} \mid \overline{y}) \quad :\Longleftrightarrow \quad (\forall s \in X)(\forall s' \in X)(s(\overline{x}) \neq s'(\overline{y}))$$

The variables $\overline{x}$ and $\overline{y}$ have no common values in X: $X(\overline{x}) \cap X(\overline{y}) = \varnothing$.

Exclusion and inclusion dependencies

Exclusion dependencies:

$$\models_X (\overline{x} \mid \overline{y}) \quad :\Longleftrightarrow \quad (\forall s \in X)(\forall s' \in X)(s(\overline{x}) \neq s'(\overline{y}))$$

The variables $\overline{x}$ and $\overline{y}$ have no common values in X: $X(\overline{x}) \cap X(\overline{y}) = \emptyset$.

Inclusion dependencies:

$$\models_X (\overline{x} \subseteq \overline{y}) \quad :\Longleftrightarrow \quad (\forall s \in X)(\exists s' \in X)(s(\overline{x}) = s'(\overline{y}))$$

Every value of $\overline{x}$ in the team X appears also as a value for $\overline{y}$ in X: $X(\overline{x}) \subseteq X(\overline{y})$.

Logics of dependence and independence

Combine such atoms with logical connectives and quantifiers to obtain full-fledged logics for reasoning about dependence and independence.

Dependence logic: FO + dependence atoms $=(\overline{x}, y)$

Independence logic: FO + independence atoms $\overline{x} \perp \overline{y}$

Inclusion logic: FO + inclusion atoms $(\overline{x} \subseteq \overline{y})$ and so on.

Logics of dependence and independence

Combine such atoms with logical connectives and quantifiers to obtain full-fledged logics for reasoning about dependence and independence.

Dependence logic: FO + dependence atoms $=(\overline{x}, y)$

Independence logic: FO + independence atoms $\overline{x} \perp \overline{y}$

Inclusion logic: FO + inclusion atoms $(\overline{x} \subseteq \overline{y})$ and so on.

Note: Formulae are always assumed to be in negation normal form

Logics of dependence and independence

Combine such atoms with logical connectives and quantifiers to obtain full-fledged logics for reasoning about dependence and independence.

Dependence logic: FO + dependence atoms $=(\overline{x}, y)$

Independence logic: FO + independence atoms $\overline{x} \perp \overline{y}$

Inclusion logic: FO + inclusion atoms $(\overline{x} \subseteq \overline{y})$ and so on.

Note: Formulae are always assumed to be in negation normal form

All these logics require team semantics.

What precisely does it mean that, $\mathfrak{A} \models_X \psi(\overline{x})$?

Team semantics for first-order logic

$$\mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y}) \text{ for all } s \in X$$

Team semantics for first-order logic

$$\mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y}) \text{ for all } s \in X$$

- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad$ for all $s \in X$, $\mathfrak{A} \models_s \psi$ and $\mathfrak{A} \models_s \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad$ for all $s \in X$, $\mathfrak{A} \models_s \psi$ or $\mathfrak{A} \models_s \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad$ for all $s \in X$ there exist $a \in A$ with $\mathfrak{A} \models_{s[y \mapsto a]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad$ for all $s \in X$ and all $a \in A$, $\mathfrak{A} \models_{s[y \mapsto a]} \psi$

Team semantics for first-order logic

$$\mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y}) \text{ for all } s \in X$$

- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad$ for all $s \in X$, $\mathfrak{A} \models_s \psi$ and $\mathfrak{A} \models_s \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad$ for all $s \in X$, $\mathfrak{A} \models_s \psi$ or $\mathfrak{A} \models_s \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad$ for all $s \in X$ there exist $a \in A$ with $\mathfrak{A} \models_{s[y \mapsto a]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad$ for all $s \in X$ and all $a \in A$, $\mathfrak{A} \models_{s[y \mapsto a]} \psi$

In the presence of dependency atoms this flatness property breaks down

We need an inductive definition, defining the team semantics of a composite formula in terms of the team semantics of its building blocks.

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\quad \mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\varnothing\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\quad \mathfrak{A} \models_X \psi(\overline{y}) \iff \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\varnothing\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\quad \mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\varnothing\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\quad \mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\varnothing\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

Notice that $(\varphi \vee \varphi)$ is, in general, not equivalent to φ

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\quad \mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\varnothing\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

Notice that $(\varphi \vee \varphi)$ is, in general, not equivalent to φ

An example from dependence logic:

$=(y)$ means that the value of y is constant in the given team

$=(y) \vee =(y)$ means that y takes at most two values in the given team

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\quad \mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\emptyset\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\quad \mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\emptyset\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

Choose (for every $s \in X$) an arbitrary non-empty set of witnesses for $\exists x \ldots$ rather than just a single witness: lax semantics as opposed to strict semantics.

For FO and dependence logic the difference is immaterial, but for some other logics, only lax semantics guarantees the locality principle:

$$\mathfrak{A} \models_X \varphi \quad \Longleftrightarrow \quad \mathfrak{A} \models_{X \restriction \mathrm{free}(\varphi)} \varphi$$

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\quad \mathfrak{A} \models_X \psi(\overline{y}) \quad \Longleftrightarrow \quad \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\varnothing\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

Team semantics: inductive definition

- For f.o.-literals $\psi(\overline{y})$: $\mathfrak{A} \models_X \psi(\overline{y}) \iff \mathfrak{A} \models_s \psi(\overline{y})$ for all $s \in X$
- semantics of dependency atoms as defined above
- $\mathfrak{A} \models_X \psi \wedge \varphi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_X \varphi$
- $\mathfrak{A} \models_X \psi \vee \varphi \quad :\Longleftrightarrow \quad X = Y \cup Z$ such that $\mathfrak{A} \models_Y \psi$ and $\mathfrak{A} \models_Z \varphi$
- $\mathfrak{A} \models_X \exists y \psi \quad :\Longleftrightarrow \quad \exists F : X \to \mathcal{P}(A) \setminus \{\varnothing\}$ such that $\mathfrak{A} \models_{X[y \mapsto F]} \psi$
- $\mathfrak{A} \models_X \forall y \psi \quad :\Longleftrightarrow \quad \mathfrak{A} \models_{X[y \mapsto A]} \psi$

For sentences we define: $\mathfrak{A} \models \psi \;:\Longleftrightarrow \mathfrak{A} \models_{\{\varnothing\}} \psi$

Notice that we cannot reasonably replace $\{\varnothing\}$ by $\varnothing$ since the empty team satisfies all formulae: $\mathfrak{A} \models_\varnothing \psi$ for all ψ

Example: Defining 3-SAT in dependence logic

Represent an instance $\varphi = \bigwedge_{i=1}^{m}(Y_{i_1} \vee Y_{i_2} \vee Y_{i_3})$ of 3-SAT by a team

$Z_\varphi = \{(i, j, X, \sigma) : \text{in clause } i \text{ at position } j, \text{ the variable } X \text{ appears with parity } \sigma\}$

Example: The formula $\varphi = (X_1 \vee \neg X_2 \vee X_3) \wedge (X_2 \vee X_4 \vee \neg X_5)$, is described by the team

clause	position	variable	parity
1	1	X_1	+
1	2	X_2	-
1	3	X_3	+
2	1	X_2	+
2	2	X_4	+
2	3	X_5	-

Example: Defining 3-SAT in dependence logic

Represent an instance $\varphi = \bigwedge_{i=1}^{m}(Y_{i_1} \vee Y_{i_2} \vee Y_{i_3})$ of 3-SAT by a team

$Z_\varphi = \{(i, j, X, \sigma) : \text{in clause } i \text{ at position } j, \text{ the variable } X \text{ appears with parity } \sigma\}$

Example: The formula $\varphi = (X_1 \vee \neg X_2 \vee X_3) \wedge (X_2 \vee X_4 \vee \neg X_5)$, is described by the team

clause	position	variable	parity
1	1	X_1	+
1	2	X_2	-
1	3	X_3	+
2	1	X_2	+
2	2	X_4	+
2	3	X_5	-

Proposition. φ is satisfiable if, and only if, the team Z_φ is a model of

$$=(\text{clause,position}) \vee =(\text{clause, position}) \vee =(\text{variable,parity})$$

From team semantics to Tarski semantics

A team of assignments $s : \{x_1, \ldots, x_k\} \to A$ can be viewed as a relation $X \subseteq A^k$.

The translation from team semantics to Tarski semantics requires that we go to existential second-order logic Σ^1_1.

Proposition. Every formula $\psi(x_1, \ldots, x_n)$ in dependence or independence logic, with vocabulary τ, can be translated into a Σ^1_1-sentence ψ^* of vocabulary $\tau \cup \{X\}$ such that

$$\mathfrak{A} \models_X \psi(\overline{x}) \quad \Longleftrightarrow \quad (\mathfrak{A}, X) \models \psi^*$$

Indeed this holds for any extension of FO by atomic properties of teams that are first-order expressible (on the relation describing the team).

To understand the expressive power of a logic with team semantics we have to identify the fragment of Σ^1_1 to which it corresponds.

Dependence logic: translation into Σ^1_1

Construct translation $\psi(\overline{x}) \mapsto \psi^*(X)$ from dependence logic into Σ^1_1 such that $\mathfrak{A} \models_X \psi(\overline{x}) \iff (\mathfrak{A}, X) \models \psi^*$.

Exploit downwards closure: If $\mathfrak{A} \models_X \psi$ and $Y \subseteq X$, then $\mathfrak{A} \models_Y \psi$

Dependence logic: translation into Σ^1_1

Construct translation $\psi(\overline{x}) \mapsto \psi^*(X)$ from dependence logic into Σ^1_1 such that $\mathfrak{A} \models_X \psi(\overline{x}) \iff (\mathfrak{A}, X) \models \psi^*$.

Exploit downwards closure: If $\mathfrak{A} \models_X \psi$ and $Y \subseteq X$, then $\mathfrak{A} \models_Y \psi$

- First-order literals $\alpha(\overline{x})$ are translated into $\forall \overline{x}(X\overline{x} \to \alpha(\overline{x}))$
- Dependency atoms $=(x_{i_1}, \ldots, x_{i_k}, x_i)$ are translated into
  $\forall \overline{x} \forall \overline{y}\Big(X\overline{x} \wedge X\overline{y} \wedge \bigwedge_{j=1}^{k}(x_{i_j} = y_{i_j}) \to (x_i = y_i)\Big)$
- $(\psi(\overline{x}) \wedge \varphi(\overline{x}))^* := \psi^*(X) \wedge \varphi^*(X)$
- $(\psi(\overline{x}) \vee \varphi(\overline{x}))^* := \exists Y \exists Z(\forall \overline{x}(X\overline{x} \to (Y\overline{x} \vee Z\overline{x}) \wedge \psi^*(Y) \wedge \varphi^*(Z)$
- $(\exists y\, \psi)^* := \exists Y(\forall \overline{x} \exists y(X\overline{x} \to Y\overline{x}y) \wedge \psi^*(Y))$
- $(\forall y\, \psi)^* := \exists Y(\forall \overline{x} \forall y(X\overline{x} \to Y\overline{x}y) \wedge \psi^*(Y))$.

Expressive power of dependence logic

By downwards closure, the sentences ψ^* have to be downwards monotone in the team predicate:
If $(\mathfrak{A}, X) \models \psi^*$ and $Y \subseteq X$ then $(\mathfrak{A}, Y) \models \psi^*$.

Expressive power of dependence logic

By downwards closure, the sentences ψ^* have to be downwards monotone in the team predicate:
If $(\mathfrak{A}, X) \models \psi^*$ and $Y \subseteq X$ then $(\mathfrak{A}, Y) \models \psi^*$.

Thus dependence logic corresponds to a strict fragment of existential second-order logic.

Expressive power of dependence logic

By downwards closure, the sentences ψ^* have to be downwards monotone in the team predicate:
If $(\mathfrak{A}, X) \models \psi^*$ and $Y \subseteq X$ then $(\mathfrak{A}, Y) \models \psi^*$.

Thus dependence logic corresponds to a strict fragment of existential second-order logic.

Theorem (Kontinen and Väänänen 2009)
The expressive power of formulae $\psi(x_1, \ldots, x_n)$ of dependence logic is precisely that of existential second-order sentences with the predicate for the team occurring only negatively.

Expressive power of dependence logic

By downwards closure, the sentences ψ^* have to be downwards monotone in the team predicate:
If $(\mathfrak{A}, X) \models \psi^*$ and $Y \subseteq X$ then $(\mathfrak{A}, Y) \models \psi^*$.

Thus dependence logic corresponds to a strict fragment of existential second-order logic.

Theorem (Kontinen and Väänänen 2009)
The expressive power of formulae $\psi(x_1, \ldots, x_n)$ of dependence logic is precisely that of existential second-order sentences with the predicate for the team occurring only negatively.

While sentences of dependence logic can express all NP-properties of finite structures, only the downwards closed NP-properties of teams are definable by open formulae of dependence logic.

From team semantics to Tarski semantics

Let L be any extension of FO by atomic properties of teams.

Question: Which fragment of Σ^1_1 is captured by L ?

From team semantics to Tarski semantics

Let L be any extension of FO by atomic properties of teams.

Question: Which fragment of Σ^1_1 is captured by L ?

- FO (without dependencies) corresponds to the set of Σ^1_1 sentences of the form $\forall \overline{x}(X\overline{x} \to \varphi(\overline{x}))$ where X does not occur in φ.

From team semantics to Tarski semantics

Let L be any extension of FO by atomic properties of teams.

Question: Which fragment of Σ^1_1 is captured by L ?

- FO (without dependencies) corresponds to the set of Σ^1_1 sentences of the form $\forall \overline{x}(X\overline{x} \to \varphi(\overline{x}))$ where X does not occur in φ.
- Exclusion logic and dependence logic captures precisely the Σ^1_1-sentences in which the team predicate appears only negatively.

From team semantics to Tarski semantics

Let L be any extension of FO by atomic properties of teams.

Question: Which fragment of Σ^1_1 is captured by L ?

- FO (without dependencies) corresponds to the set of Σ^1_1 sentences of the form $\forall \overline{x}(X\overline{x} \to \varphi(\overline{x}))$ where X does not occur in φ.
- Exclusion logic and dependence logic captures precisely the Σ^1_1-sentences in which the team predicate appears only negatively.
- FO + inclusion + exclusion $\equiv$ independence logic $\equiv$ Σ^1_1.
  Thus, all NP-properties of teams are definable in independence logic.

From team semantics to Tarski semantics

Let L be any extension of FO by atomic properties of teams.

Question: Which fragment of Σ^1_1 is captured by L ?

- FO (without dependencies) corresponds to the set of Σ^1_1 sentences of the form $\forall \overline{x}(X\overline{x} \to \varphi(\overline{x}))$ where X does not occur in φ.
- Exclusion logic and dependence logic captures precisely the Σ^1_1-sentences in which the team predicate appears only negatively.
- FO + inclusion + exclusion $\equiv$ independence logic $\equiv$ Σ^1_1.
  Thus, all NP-properties of teams are definable in independence logic.
- Inclusion logic coincides with the posGFP-fragment of LFP.
  Thus, on ordered finite structures, inclusion logic captures precisely the polynomial-time properties of teams.

Goal. Explain this result using a game-theoretic approach

Model-Checking Games

The model checking problem for a logic L (with classical Tarski-semantics)

Given: structure $\mathfrak{A}$

formula $\psi(\overline{x}) \in L$

assignment $s : \text{free}(\psi) \to A$

Question: $\mathfrak{A} \models_s \psi$?

Model-Checking Games

The model checking problem for a logic L (with classical Tarski-semantics)

Given: structure $\mathfrak{A}$
formula $\psi(\overline{x}) \in L$
assignment $s : \text{free}(\psi) \to A$

Question: $\mathfrak{A} \models_s \psi$?

Reduce model checking problem $\mathfrak{A} \models_s \psi$ to strategy problem for model checking game $G(\mathfrak{A}, \psi, s)$, played by

- Falsifier (also called Player 1), and
- Verifier (also called Player 0), such that

$$\mathfrak{A} \models_s \psi \iff \text{Verifier has winning strategy for } G(\mathfrak{A}, \psi, s)$$

Model-Checking Games

The model checking problem for a logic L (with classical Tarski-semantics)

Given: structure $\mathfrak{A}$
formula $\psi(\overline{x}) \in L$
assignment $s : \text{free}(\psi) \to A$

Question: $\mathfrak{A} \models_s \psi$?

Reduce model checking problem $\mathfrak{A} \models_s \psi$ to strategy problem for model checking game $G(\mathfrak{A}, \psi, s)$, played by

- Falsifier (also called Player 1), and
- Verifier (also called Player 0), such that

$$\mathfrak{A} \models_s \psi \iff \text{Verifier has winning strategy for } G(\mathfrak{A}, \psi, s)$$

$\Longrightarrow$ Model checking via construction of winning strategies

Model-checking game for first-order logic

The game $\mathcal{G}(\mathfrak{A}, \psi, t)$ for a structure $\mathfrak{A}$ and $\psi(\overline{x}) \in \mathrm{FO}$.

Positions: (φ, s) φ is a subformula of ψ and $s : \mathrm{free}(\varphi) \to A$

Verifier moves:

$(\varphi_1 \vee \varphi_2, s) \to (\varphi_i, s \upharpoonright \mathrm{free}(\varphi_i))$ $(i = 1, 2)$

$(\exists x \varphi, s) \to (\varphi, s[x \mapsto a])$ $(a \in A)$

Falsifier moves

$(\varphi_1 \wedge \varphi_2, s) \to (\varphi_i, s \upharpoonright \mathrm{free}(\varphi_i))$ $(i = 1, 2)$

$(\forall x \varphi, s) \to (\varphi, s[x \mapsto a])$ $(a \in A)$

Terminal positions: φ atomic / negated atomic

$$\begin{matrix}\text{Verifier}\\ \text{Falsifier}\end{matrix} \text{ wins at } (\varphi, s) \iff \mathfrak{A} \begin{matrix}\models_s\\ \not\models_s\end{matrix} \varphi$$

From reachability games to second-order reachability games

Games for FO are reachability games, played on trees (or acyclic graphs)
Every terminal position is either winning or losing.

A winning strategy S from initial position v has to make sure that every play from v that is consistent with S reaches a winning terminal position.

From reachability games to second-order reachability games

Games for FO are reachability games, played on trees (or acyclic graphs)
Every terminal position is either winning or losing.

A winning strategy S from initial position v has to make sure that every play from v that is consistent with S reaches a winning terminal position.

For games with team semantics we need second-order reachability games, played on forests of game trees.
Now, every set of terminal positions is either winning or losing.

A winning strategy S from a set X of initial positions must make sure that the set of all terminal positions that are reachable from X by a play that is consistent with S is a winning set.

From reachability games to second-order reachability games

Games for FO are reachability games, played on trees (or acyclic graphs)
Every terminal position is either winning or losing.

A winning strategy S from initial position v has to make sure that every play from v that is consistent with S reaches a winning terminal position.

For games with team semantics we need second-order reachability games, played on forests of game trees.
Now, every set of terminal positions is either winning or losing.

A winning strategy S from a set X of initial positions must make sure that the set of all terminal positions that are reachable from X by a play that is consistent with S is a winning set.

It is important to admit nondeterministic strategies. In second-order reachability games these are more powerful than deterministic ones.

Model-checking game for logics with team semantics

General construction, for formulae $\psi(\overline{x})$ with any kind of dependency atoms.

A game graph $\mathcal{G}(\mathfrak{A}, \psi)$ is a forest of game trees whose roots are initial positions (ψ, s) for $s : \mathrm{free}(\psi) \to A$. The game trees are defined as for FO.

Model-checking game for logics with team semantics

General construction, for formulae $\psi(\overline{x})$ with any kind of dependency atoms.

A game graph $\mathcal{G}(\mathfrak{A}, \psi)$ is a forest of game trees whose roots are initial positions (ψ, s) for $s : \mathrm{free}(\psi) \to A$. The game trees are defined as for FO.

The winning condition is a second-order reachability condition: Terminal positions are (α, s) where α is a dependence atom or a first-order literal.
To be winning, a strategy S must guarantees that for every such α,

$$\mathfrak{A} \models_{\mathrm{Team}(S,\alpha)} \alpha$$

where $\mathrm{Team}(S, \alpha) := \{s : \text{the strategy } S \text{ admits the position } (\alpha, s)\}$.

Model-checking game for logics with team semantics

General construction, for formulae $\psi(\overline{x})$ with any kind of dependency atoms.

A game graph $\mathcal{G}(\mathfrak{A}, \psi)$ is a forest of game trees whose roots are initial positions (ψ, s) for $s : \mathrm{free}(\psi) \to A$. The game trees are defined as for FO.

The winning condition is a second-order reachability condition: Terminal positions are (α, s) where α is a dependence atom or a first-order literal.
To be winning, a strategy S must guarantees that for every such α,

$$\mathfrak{A} \models_{\mathrm{Team}(S,\alpha)} \alpha$$

where $\mathrm{Team}(S, \alpha) := \{s : \text{the strategy } S \text{ admits the position } (\alpha, s)\}$.

Theorem. $\mathfrak{A} \models_X \psi$ if, and only if, Verifier has a winning strategy in $\mathcal{G}(\mathfrak{A}, \psi)$ from the set $\{(\psi, s) : s \in X\}$ of initial positions defined by X.

Model-checking game for logics with team semantics

General construction, for formulae $\psi(\overline{x})$ with any kind of dependency atoms.

A game graph $\mathcal{G}(\mathfrak{A}, \psi)$ is a forest of game trees whose roots are initial positions (ψ, s) for $s : \mathrm{free}(\psi) \to A$. The game trees are defined as for FO.

The winning condition is a second-order reachability condition: Terminal positions are (α, s) where α is a dependence atom or a first-order literal.
To be winning, a strategy S must guarantees that for every such α,

$$\mathfrak{A} \models_{\mathrm{Team}(S,\alpha)} \alpha$$

where $\mathrm{Team}(S, \alpha) := \{s : \text{the strategy } S \text{ admits the position } (\alpha, s)\}$.

Theorem. $\mathfrak{A} \models_X \psi$ if, and only if, Verifier has a winning strategy in $\mathcal{G}(\mathfrak{A}, \psi)$ from the set $\{(\psi, s) : s \in X\}$ of initial positions defined by X.

Notice that the winning condition refers to the entire set of terminal positions that are reachable by plays that are consistent with the strategy.

Complexity

Theorem. While classical reachability games can be solved in linear time, the problem whether a given second-order reachability game admits a winning strategy for Player 0, is NP-complete.

Complexity

Theorem. While classical reachability games can be solved in linear time, the problem whether a given second-order reachability game admits a winning strategy for Player 0, is NP-complete.

The size of a model checking game $\mathcal{G}(\mathfrak{A}, \psi)$ on a finite structure $\mathfrak{A}$ is bounded by $|\psi| \cdot |A|^{\text{width}(\psi)}$, where $\text{width}(\psi) := \max\{|\text{free}(\varphi)| : \varphi \text{ subformula of } \psi\}$.

Theorem. Let L be any extension of first-order logic by atomic formulae on teams that can be evaluated in polynomial time. Then the model-checking problem for L on finite structures is in NEXPTIME. For formulae of bounded width, the model-checking problem is in NP.

Complexity

Theorem. While classical reachability games can be solved in linear time, the problem whether a given second-order reachability game admits a winning strategy for Player 0, is NP-complete.

The size of a model checking game $\mathcal{G}(\mathfrak{A}, \psi)$ on a finite structure $\mathfrak{A}$ is bounded by $|\psi| \cdot |A|^{\text{width}(\psi)}$, where $\text{width}(\psi) := \max\{|\text{free}(\varphi)| : \varphi \text{ subformula of } \psi\}$.

Theorem. Let L be any extension of first-order logic by atomic formulae on teams that can be evaluated in polynomial time. Then the model-checking problem for L on finite structures is in NEXPTIME. For formulae of bounded width, the model-checking problem is in NP.

The problem is in fact NEXPTIME-complete for most logics with team semantics, even in the case where $\mathfrak{A}$ is just the set $\{0, 1\}$ and $X = \{\varnothing\}$.

Inclusion logic

Inclusion logic: FO + inclusion atoms $\overline{x} \subseteq \overline{y}$.

Recall that $\mathfrak{A} \models_X \overline{x} \subseteq \overline{y}$ if for all $s \in X$ there exists a $t \in X$ with $t(\overline{y}) = s(\overline{x})$.

Inclusion logic is an interesting variant of a dependence logic, with somewhat unusual properties.

Contrary to dependence logic, it is not closed under subteams, but it is closed under unions of teams: If $\mathfrak{A} \models_X \psi$ and $\mathfrak{A} \models_Y \psi$ then also $\mathfrak{A} \models_{X \cup Y} \psi$

Hence, in terms of expressive power, inclusion logic is incomparable to dependence logic and exclusion logic.

Example: Infinite descending chains

$$(A, <) \models \exists x \exists y (y < x \wedge y \subseteq x) \quad \Longleftrightarrow \quad (A, <) \text{ is not well-founded}$$

Example: Infinite descending chains

$$(A, <) \models \exists x \exists y (y < x \wedge y \subseteq x) \quad \Longleftrightarrow \quad (A, <) \text{ is not well-founded}$$

A (nondeterministic) strategy S for this sentence just amounts to a selection of a team X of assignments $s : (x, y) \mapsto (a, b)$.

Example: Infinite descending chains

$$(A, <) \models \exists x \exists y (y < x \wedge y \subseteq x) \quad \Longleftrightarrow \quad (A, <) \text{ is not well-founded}$$

A (nondeterministic) strategy S for this sentence just amounts to a selection of a team X of assignments $s : (x, y) \mapsto (a, b)$.

All atoms are reachable by the opponent. Hence $\mathrm{Team}(S, y < x) = \mathrm{Team}(S, y \subseteq x) = X$. Thus, S is winning if:

Example: Infinite descending chains

$$(A, <) \models \exists x \exists y (y < x \wedge y \subseteq x) \quad \Longleftrightarrow \quad (A, <) \text{ is not well-founded}$$

A (nondeterministic) strategy S for this sentence just amounts to a selection of a team X of assignments $s : (x, y) \mapsto (a, b)$.

All atoms are reachable by the opponent. Hence $\mathrm{Team}(S, y < x) = \mathrm{Team}(S, y \subseteq x) = X$. Thus, S is winning if:

$(A, <) \models_X (y < x)$: For all $s : (x, y) \mapsto (a, b)$ in X, we have that $b < a$

Example: Infinite descending chains

$$(A, <) \models \exists x \exists y (y < x \wedge y \subseteq x) \quad \Longleftrightarrow \quad (A, <) \text{ is not well-founded}$$

A (nondeterministic) strategy S for this sentence just amounts to a selection of a team X of assignments $s : (x, y) \mapsto (a, b)$.

All atoms are reachable by the opponent. Hence $\mathrm{Team}(S, y < x) = \mathrm{Team}(S, y \subseteq x) = X$. Thus, S is winning if:

$(A, <) \models_X (y < x)$: For all $s : (x, y) \mapsto (a, b)$ in X, we have that $b < a$

$(A, <) \models_X (y \subseteq x)$: For all $s : (x, y) \mapsto (a, b)$ in X, we have another assignment $t : (x, y) \mapsto (b, c)$ in X.

Example: Infinite descending chains

$$(A, <) \models \exists x \exists y (y < x \wedge y \subseteq x) \quad \Longleftrightarrow \quad (A, <) \text{ is not well-founded}$$

A (nondeterministic) strategy S for this sentence just amounts to a selection of a team X of assignments $s : (x, y) \mapsto (a, b)$.

All atoms are reachable by the opponent. Hence $\mathrm{Team}(S, y < x) = \mathrm{Team}(S, y \subseteq x) = X$. Thus, S is winning if:

$(A, <) \models_X (y < x)$: For all $s : (x, y) \mapsto (a, b)$ in X, we have that $b < a$

$(A, <) \models_X (y \subseteq x)$: For all $s : (x, y) \mapsto (a, b)$ in X, we have another assignment $t : (x, y) \mapsto (b, c)$ in X.

Hence we have a winning strategy in the model-checking game if, and only if, $(A, <)$ has an infinite descending chain.

Safety games for inclusion logic

Safety games are games that may admit infinite plays, and where Verifier has to avoid a given set of losing terminal positions but wins all infinite plays.

Safety games for inclusion logic

Safety games are games that may admit infinite plays, and where Verifier has to avoid a given set of losing terminal positions but wins all infinite plays.

The second-order reachability game $\mathcal{G}(\mathfrak{A}, \psi)$ for a formula $\psi \in \mathrm{FO}(\subseteq)$ can be modified to a safety game $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ as follows:

From positions $(\overline{x} \subseteq \overline{y}, s)$, Verifier can move to any position $(\overline{x} \subseteq \overline{y}, t)$ such that $t(\overline{y}) = s(\overline{x})$. From there, Falsifier can make an arbitrary move upwards in the game forest.

From a given set $X \subseteq I$ of initial positions, Verifier must make sure to avoid
- terminal positions (φ, s), where φ is a first-order literal with $\mathfrak{A} \not\models_s \varphi$, and
- initial positions outside X

Safety games for inclusion logic

Safety games are games that may admit infinite plays, and where Verifier has to avoid a given set of losing terminal positions but wins all infinite plays.

The second-order reachability game $\mathcal{G}(\mathfrak{A}, \psi)$ for a formula $\psi \in \mathrm{FO}(\subseteq)$ can be modified to a safety game $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ as follows:

From positions $(\overline{x} \subseteq \overline{y}, s)$, Verifier can move to any position $(\overline{x} \subseteq \overline{y}, t)$ such that $t(\overline{y}) = s(\overline{x})$. From there, Falsifier can make an arbitrary move upwards in the game forest.

From a given set $X \subseteq I$ of initial positions, Verifier must make sure to avoid
- terminal positions (φ, s), where φ is a first-order literal with $\mathfrak{A} \not\models_s \varphi$, and
- initial positions outside X

Proposition. Every winning strategy $S = (W, F)$ from X for $\mathcal{G}(\mathfrak{A}, \psi)$ is also a winning strategy for $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ that avoids $I \setminus X$, and vice versa.

Inclusion logic and greatest fixed points

Existence of an infinite descending chain is not first-order definable, and not even in $L_{\infty\omega}$. However, there is a simple definition by a greatest fixed point:

$(A, <)$ is not well-founded $\iff$ $(A, <) \models \exists x[\mathbf{gfp}\, Cx \,.\, \exists y(y < x \wedge Cy)](x)$

Inclusion logic and greatest fixed points

Existence of an infinite descending chain is not first-order definable, and not even in $L_{\infty\omega}$. However, there is a simple definition by a greatest fixed point:

$$(A, <) \text{ is not well-founded} \iff (A, <) \models \exists x[\mathbf{gfp}\, Cx\,.\, \exists y(y < x \wedge Cy)](x)$$

Given a team X, the maximal subteam $X_{\max} \subseteq X$ satisfying an inclusion statement $x_i \subseteq x_j$ is also definable as a greatest fixed point:

$$\overline{x} \in X_{\max} \iff (\mathfrak{A}, X) \models [\mathbf{gfp}\, Y\overline{x}\,.\, X\overline{x} \wedge \exists \overline{y}(Y\overline{y} \wedge y_j = x_i)](\overline{x})$$

Further $\mathfrak{A} \models_X (x_i \subseteq x_j)$ if, and only if $(\mathfrak{A}, X) \models \forall \overline{x}(X\overline{z} \to [\mathbf{gfp}\, Y\overline{x}. \ldots](\overline{x}))$.

Inclusion logic and greatest fixed points

Existence of an infinite descending chain is not first-order definable, and not even in $L_{\infty\omega}$. However, there is a simple definition by a greatest fixed point:

$$(A, <) \text{ is not well-founded} \iff (A, <) \vDash \exists x[\mathbf{gfp}\, Cx \,.\, \exists y(y < x \wedge Cy)](x)$$

Given a team X, the maximal subteam $X_{\max} \subseteq X$ satisfying an inclusion statement $x_i \subseteq x_j$ is also definable as a greatest fixed point:

$$\overline{x} \in X_{\max} \iff (\mathfrak{A}, X) \vDash [\mathbf{gfp}\, Y\overline{x} \,.\, X\overline{x} \wedge \exists \overline{y}(Y\overline{y} \wedge y_j = x_i)](\overline{x})$$

Further $\mathfrak{A} \vDash_X (x_i \subseteq x_j)$ if, and only if $(\mathfrak{A}, X) \vDash \forall \overline{x}(X\overline{z} \to [\mathbf{gfp}\, Y\overline{x}. \ldots](\overline{x}))$.

Claim. This generalizes from basic inclusion atoms to all formulae of inclusion logic.

Least fixed-point logic and its posGFP-fragment

LFP extends first-order logic by least and greatest fixed points of monotone definable operators. It is a logic of great importance in finite model theory.

Theorem (Immerman, Vardi)
On ordered finite structures, LFP captures PTIME.

Least fixed-point logic and its posGFP-fragment

LFP extends first-order logic by least and greatest fixed points of monotone definable operators. It is a logic of great importance in finite model theory.

Theorem (Immerman, Vardi)
On ordered finite structures, LFP captures PTIME.

posGFP is the fragment of LFP that makes use of greatest fixed points only (which may appear only positively).

posGFP is at the bottom level of the alternation hierarchy of LFP. In general, and for instance on $(\mathbb{N}, +, \cdot)$, this hierarchy is strict.

While, in general LFP $\subseteq \Delta^1_2 \setminus \Sigma^1_1$, we have that posGFP $\subseteq \Sigma^1_1$.

Least fixed-point logic and its posGFP-fragment

LFP extends first-order logic by least and greatest fixed points of monotone definable operators. It is a logic of great importance in finite model theory.

Theorem (Immerman, Vardi)
On ordered finite structures, LFP captures PTIME.

posGFP is the fragment of LFP that makes use of greatest fixed points only (which may appear only positively).

posGFP is at the bottom level of the alternation hierarchy of LFP. In general, and for instance on $(\mathbb{N}, +, \cdot)$, this hierarchy is strict.

While, in general LFP $\subseteq \Delta^1_2 \setminus \Sigma^1_1$, we have that posGFP $\subseteq \Sigma^1_1$.

Theorem (Immerman) On finite structures, LFP $\equiv$ posGFP $\subseteq \Delta^1_1$.

posGFP and safety games

The model checking games for general LFP-formulae are parity games, which are not known to solvable in polynomial time.

However, the model-checking games for posGFP are safety games.

posGFP and safety games

The model checking games for general LFP-formulae are parity games, which are not known to solvable in polynomial time.

However, the model-checking games for posGFP are safety games.

How can we use the fact, that posGFP and FO($\subseteq$) have the same kind of model-checking games, to provide translations between the two logics ?

posGFP and safety games

The model checking games for general LFP-formulae are parity games, which are not known to solvable in polynomial time.

However, the model-checking games for posGFP are safety games.

How can we use the fact, that posGFP and FO($\subseteq$) have the same kind of model-checking games, to provide translations between the two logics ?

For this we need two further facts on model-checking games:

Interpretability: For every formula ψ (in any of the two logics) there exists a first-order interpretation I_ψ that interprets $\mathcal{G}(\mathfrak{A}, \psi)$ in $\mathfrak{A}$ (for every $\mathfrak{A}$).

Winning is definable: In both logics there exist formulae win(x) defining, on every safety game $\mathcal{G}$, the set of positions from which Verifier wins.

Game-based translations between FO($\subseteq$) and posGFP

Given a formula $\psi(\overline{x}) \in \mathrm{FO}(\subseteq)$ take the first-order interpretation I_ψ that interprets the safety game $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ in $\mathfrak{A}$. That is: $I_\psi : (\mathfrak{A}, X) \mapsto \mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$.

But this also means that every formula φ on $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ translates into a formula $I_\psi(\varphi)$ that expresses in $(\mathfrak{A}, X)$ what φ does in $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$.

Game-based translations between FO($\subseteq$) and posGFP

Given a formula $\psi(\overline{x}) \in \mathrm{FO}(\subseteq)$ take the first-order interpretation I_ψ that interprets the safety game $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ in $\mathfrak{A}$. That is: $I_\psi : (\mathfrak{A}, X) \mapsto \mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$.

But this also means that every formula φ on $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ translates into a formula $I_\psi(\varphi)$ that expresses in $(\mathfrak{A}, X)$ what φ does in $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$.

Apply this to $\mathrm{win}(y) \in \mathrm{posGFP}$ which defines the winning positions in safety games. I_ψ translates $\mathrm{win}(y)$ into another posGFP-formula $\mathrm{win}^*(\overline{y})$ expressing in $(\mathfrak{A}, X)$ the relevant winning condition of the model-checking game.

Combining this with the definability of the input positions associated with X, we obtain a posGFP-formula $\varphi(\overline{x})$ that is equivalent to $\psi(\overline{x})$, in the sense that

$$\mathfrak{A} \models_X \psi \qquad \Longleftrightarrow \qquad (\mathfrak{A}, X) \models_s \varphi(\overline{x}) \text{ for all } s \in X$$

Game-based translations between FO($\subseteq$) and posGFP

Given a formula $\psi(\overline{x}) \in \mathrm{FO}(\subseteq)$ take the first-order interpretation I_ψ that interprets the safety game $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ in $\mathfrak{A}$. That is: $I_\psi : (\mathfrak{A}, X) \mapsto \mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$.

But this also means that every formula φ on $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$ translates into a formula $I_\psi(\varphi)$ that expresses in $(\mathfrak{A}, X)$ what φ does in $\mathcal{G}_{\mathrm{safe}}(\mathfrak{A}, \psi)$.

Apply this to $\mathrm{win}(y) \in \mathrm{posGFP}$ which defines the winning positions in safety games. I_ψ translates $\mathrm{win}(y)$ into another posGFP-formula $\mathrm{win}^*(\overline{y})$ expressing in $(\mathfrak{A}, X)$ the relevant winning condition of the model-checking game.

Combining this with the definability of the input positions associated with X, we obtain a posGFP-formula $\varphi(\overline{x})$ that is equivalent to $\psi(\overline{x})$, in the sense that

$$\mathfrak{A} \models_X \psi \qquad \Longleftrightarrow \qquad (\mathfrak{A}, X) \models_s \varphi(\overline{x}) \text{ for all } s \in X$$

An analogous argument works for the translation of posGFP into Inc.

Inclusion logic and least fixed-point logic

Theorem. (Galliani and Hella) For every formula $\varphi(\overline{z}) \in \mathrm{FO}(\subseteq)$ one can construct a formula $\psi(X, \overline{z})$ in posGFP, and vice versa, such that, for all $\mathfrak{A}$ and all X

$$\mathfrak{A} \models_X \varphi \quad \Longleftrightarrow \quad (\mathfrak{A}, X) \models \forall \overline{z}(X\overline{z} \to \psi(X, \overline{z}))$$

Thus the maximal team satisfying φ coincides with $\mathbf{gfp}(\psi)$.

For the case of sentences, ψ and φ are equivalent.

Corollary. For sentences, inclusion logic and posGFP have the same expressive power.

Corollary. On finite structures, inclusion logic and LFP have the same expressive power. In particular, on ordered finite structures, inclusion logic captures PTIME.

Further work

On finite structures, the equivalence between positive greatest fixed point logic, safety games and inclusion logic lifts to an equivalence between corresponding counting extensions:

Theorem. (Grädel, Hegselmann)
Inclusion logic with counting corresponds to threshold safety games and to fixed-point logic with counting.

Thus, inclusion logic with counting comes rather close to capturing PTIME.

Summary

There is a rich collection of logics of dependence and independence, on the basis of different atomic dependency properties

All these logics are based on team semantics

In terms of expressive power, logics of dependence and independence are equivalent to existential second-order logic or natural fragments of it.

The model-checking games for logics with team semantics are second-order reachability games. These game provide evaluation procedures and complexity results for these logics.

For inclusion logic the games can be simplified to classical safety games.

Inclusion logic is equivalent to the posGFP fragment of fixed-point logic. This generalizes to counting extensions.